Peter Baeumle-Courth

Approximation, Reduktion und Regelextraktion: Semantikbeschreibung für Neuronale Netze

- 2004 -

Informatik

Approximation, Reduktion und Regelextraktion: Semantikbeschreibung für Neuronale Netze

Inaugural-Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften im Fachbereich
Mathematik und Informatik
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms-Universität Münster

Vorgelegt von
Peter Baeumle-Courth, geb. Baeumle,
aus Freiburg i. Br. (Baden-Württemberg)
- 2004 -

Dekan: Prof. Dr. K. Hinrichs
Erster Gutachter: Prof. Dr. W.-M. Lippe
Zweiter Gutachter: Prof. Dr. X. Jiang
Tag der mündlichen Prüfung: 12.Januar 2005
Tag der Promotion: 12.Januar 2005

Abstract

Die vorliegende Dissertation befasst sich mit der Problemstellung, aus einem mehrschichtigen Neuronalen Feedforward-Netz, das für die Außenwelt als Black Box wirkt, eine semantische Beschreibung zu extrahieren. Als Form dieser Beschreibung des Netz-Verhaltens dient die Struktur einer Regelbasis bestehend aus menschlich verständlichen und durch Experten gut zu interpretierenden IF-THEN-Regeln.

Hierbei steht das Ziel im Vordergrund, eine möglichst allgemeine Methodik der Regelextraktion zu entwickeln, so dass die resultierenden Verfahren flexibel eingesetzt werden können.

Nach einer Einführung in die zentralen Grundlagen von Neuronalen Netzen, Fuzzy-Systemen und der Kombination dieser Ansätze werden exemplarisch einige bereits bekannte Regelextraktionsverfahren konkret vorgestellt. Es zeigt sich allerdings, dass die hierbei betrachteten Neuronalen Netze jeweils einen eng definierten Aufbau besitzen.

Die vorliegende Arbeit löst sich von solchen Vorgaben und betrachtet allgemeine dreischichtige Feedforward-Netze mit sog. sigmoiden Ausgabefunktionen. Methodisch werden einzelne Aspekte sequentiell nacheinander behandelt, so dass ggf. für eine spätere Anwendung komponentenartig benötigte Teile zum Einsatz kommen können.

Zunächst befasst sich die Arbeit mit Möglichkeiten, die vorgegebenen rechen-intensiven sigmoiden Funktionen durch einfachere Abbildungen, sogenannte Rampenfunktionen, zu approximieren. Im darauffolgenden Schritt wird ausgearbeitet, unter welchen Bedingungen das Neuronale Netz vereinfacht werden kann, konkret: mit welcher Güte-Abschätzung eine Reduktion der Anzahl der Neuronen in der verborgenen Schicht möglich ist. Schließlich findet dann auf dieser Grundlage die eigentliche Regelextraktion statt, für die verschiedene Verfahren entwickelt und an konkreten numerischen Beispielen miteinander verglichen werden.

Flankierend zur theoretischen Ausarbeitung entstand im Zuge der vorliegenden Dissertation auch die prototypische Realisierung einer Java-Applikation (“Javanna”) zur Illustration der Regelextraktion für Neuronale Netze mit skalarer Ausgabe sowie einer Klassenbibliothek, mit der mehrschichtige Feedforward-Netze abgebildet und die hier präsentierten Regelextraktionsverfahren umgesetzt werden können.

Danksagung

Meinen ganz besonderen Dank möchte ich Herrn Prof. Dr. W.-M. Lippe aussprechen, der mich in den vergangenen Jahren mit kompetentem Rat in zahlreichen motivierenden Gesprächen bei meiner Arbeit begleitet hat.

Herrn Prof. Dr. X. Jiang bekunde ich herzlich meinen Dank für seine Bereitschaft, die vorliegende Arbeit als Zweitgutachter zu beurteilen.

Dr. Steffen Niendieck, Dr. Andreas Tenhagen, Dr. Thomas Feuring und Dipl.-Math. Christoph Mertens danke ich ebenfalls für eine Vielzahl konstruktiver Diskussionen; der Meinungsaustausch mit ihnen sowie mit den Studenten der Soft Computing-AG haben mir als externem Doktoranden die fachliche Arbeit erleichtert.

Schließlich möchte ich meiner Frau Marianne Courth für ihre Unterstützung im privaten Umfeld und nicht zuletzt das abschließende Lektorat danken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Die Entwicklung der Menschen ist gekennzeichnet von Wachstum, sowohl in qualitativer wie in quantitativer Hinsicht. Dies zeigt sich im Anwachsen der Weltbevölkerung, in der zentralen Grundüberzeugung des vorherrschenden Wirtschaftssystems, in dem persönlichen Anspruch, die Lebensqualität immer weiter zu verbessern, in der extrem beschleunigten Zunahme des Wissens der Menschheit beispielsweise auf den Gebieten der Wissenschaft und Technik, dabei besonders der Medizin, der Informatik und den Biowissenschaften.

Dabei ist in der jüngeren Vergangenheit (verglichen mit der gesamten menschlichen Entwicklung) mit der elektronischen Datenverarbeitung eine neue Qualität an Werkzeugen, an Hilfsmitteln entstanden. Waren zuvor Werkzeuge (u.a. Maschinen) die Unterstützung bei im weitesten Sinne manuellen Tätigkeiten des Menschen, so ist die Informatik neben der Mathematik oder der Biologie längst ebenfalls ein Bereich, in dem auch die Denkleistungen des Menschen selbst den bzw. einen zentralen Gegenstand der Forschung darstellen.

“Wie denkt der Mensch?”, das ist eine der grundlegenden Fragestellungen, die erforscht werden. “Wie zieht ein Mensch Schlussfolgerungen?” ist eine spezielle Frage, die u.a. im Rahmen der sogenannten “Künstlichen Intelligenz” (KI) als auch des “Soft Computing” behandelt wird. Unter dem Begriff “Soft Computing” werden i.w. die Themengebiete (Künstliche) Neuronale Netze, Fuzzy-Systeme und Genetische Algorithmen zusammengefasst. Die Behandlung symbolischer Regeln und Schlüsse fällt dagegen in den Bereich der “Künstlichen Intelligenz” (KI).

John D. Barrow führt in seinem Buch [Barr2001] in sehr kompakter Form durch das historische Auf und Ab der mathematischen Disziplinen, nicht zuletzt der Mathematischen Logik, sowie der Errungenschaften von Mathematikern wie Kurt Gödel und Alfred Tarski. Dort wird auch ausgeführt, dass (unter gewissen Minimalvoraussetzungen) die Semantik eines formalen logischen Systems nicht vollständig sein kann, sich also niemals selbst (ganz) erklären kann.

Da aus diesen prinzipiellen Überlegungen heraus gesagt werden kann, dass das menschliche Denken niemals “alles” über das menschliche Denken selbst wissen, erfassen, begreifen kann, muss die zu behandelnde Fragestellung selbstverständlich präziser formuliert (bzw. eingeschränkt) werden.

Im Rahmen der vorliegenden Arbeit soll es darum gehen, das Verhalten eines Künstlichen Neuronalen Netzes dadurch besser verstehen zu lernen, dass eine Regelbasis angegeben wird, welches das Verhalten des Netzes möglichst gut darstellt. Dieses “möglichst gut” muss selbstverständlich wiederum präzisiert werden. Es kann beispielsweise bedeuten, dass eine vorgegebene numerische Präzision bei Zahlenwerten erzielt wird; es kann aber ebensogut bedeuten, dass ein gewisser prozentualer Anteil von Anwendungsfällen korrekt klassifiziert wird. Die “Güte” muss jeweils in der konkreten Situation definiert werden.

Dies ist nicht nur für den Forschenden von Interesse, vielmehr gibt es zahlreiche praktische Anwendungsfälle, in denen eine durch ein Neuronales Netz veranlasste Konsequenz auch nachvollziehbar begründet werden muss, etwa wenn es im Bankbereich um die (Nicht-) Bewilligung eines Kreditantrages geht, im Versicherungsbereich um das Eingehen eines Lebensversicherungsvertrages auf einer gegebenen Datengrundlage oder im besonders

sensiblen medizinischen Bereich um die Diagnoseunterstützung eines Krankheitsbildes mit daraus resultierender Medikamentierung. In allen Fällen muss ein Experte[1] seiner Umwelt seine Entscheidung begründen können.

Es sollen dabei jedoch - anders als bei Expertensystemen aus dem Bereich der KI - die Vorteile, die die Künstlichen Neuronalen Netze bieten, weiterhin nutzbar bleiben. Insbesondere richten wir unser Augenmerk im Laufe dieser Arbeit hauptsächlich auf die Interpretation bereits vorliegender Netze; Aspekte wie das Trainieren von Neuronalen Netzen oder das Modifizieren eines Netzes im Rahmen der Interpretation sollen hier nicht im Vordergrund stehen.

Die vorliegende Arbeit ist wie folgt aufgebaut: In **Teil I** werden einige der Grundlagen des Soft Computing referiert, soweit diese im Rahmen der Arbeit benötigt werden oder dem prinzipiellen Verständnis dienen. Teil I umfasst die Kapitel 1 bis 4.

Kapitel 1 führt in die Thematik der Künstlichen Neuronalen Netze ein; dabei wird zunächst kurz das biologische Vorbild präsentiert, einige historische Anmerkungen werden gegeben. Anschließend werden verschiedene Aspekte wie das Trainieren Neuronaler Netze, die Art und Weise, wie ein Neuronales Netz Wissen repräsentiert, die Approximationsfähigkeit sowie eine Reihe praktischer Anwendungsgebiete von Netzen vorgestellt.

In Kapitel 2 wird ein Einblick in die Fuzzy-Theorie und den Anwendungsbereich von Fuzzy-Systemen gegeben. Dies umfasst Fuzzy-Mengen, -Zahlen, -Arithmetik und - nicht zuletzt - die Grundlagen der Fuzzy-Logik. Eine Vorstellung von Fuzzy-Entscheidungssystemen wie den Fuzzy-Controllern nach Mamdani sowie Sugeno und Takagi rundet dieses Kapitel ab.

Kapitel 3 stellt Kombinationsmöglichkeiten von Neuronalen Netzen und Fuzzy-Anwendungen vor. Es werden die Vor- und Nachteile der beiden Ansätze diskutiert und Beispiele für Fuzzy-Neuro- und Neuro-Fuzzy-Systeme gegeben. Bekannte Ansätze wie das NEFCON-System werden kurz dargestellt.

In Kapitel 4 wird dargestellt, dass unter gewissen Voraussetzungen Neuronale Netze als Fuzzy-Entscheidungssysteme beispielsweise im Sinne eines Mamdani-Controllers interpretiert werden können. Schwerpunktmäßig wird auf die Arbeiten von Tenhagen [Tenh2000] und Niendieck [Nien1998] eingegangen, worin Abbildungen zwischen Neuronalen Netzen und Fuzzy-Entscheidungssystemen behandelt werden. Ergänzend wird hier auch die funktionale Äquivalenz von Neuronalen Netzen mit sog. radialen Basisfunktionen (RBF-Netzen) und Sugeno-Takagi-Controllern diskutiert.

Teil II (Kapitel 5 bis 7) befasst sich mit dem Kernthema der Arbeit, der "Regelextraktion aus Neuronalen Netzen".

Kapitel 5 stellt die wesentlichen Grundlagen der Regelextraktion dar. Dabei geht es um die Klärung, wie eine Regel bzw. eine Menge von Regeln, eine sog. Regelbasis, formuliert

[1] Im Rahmen dieser Arbeit wird aus Gründen der besseren Lesbarkeit nur die "einfache", männliche Form aufgeführt. Selbstverständlich soll diese Begrifflichkeit weibliche Personen, z.B. hier Expertinnen, mit umfassen.

werden kann, und um die verschiedenen Vorgehensweisen, wie Regeln extrahiert werden können.

Schließlich wird ein Überblick über verschiedene Verfahren der Regelextraktion gegeben; insbesondere werden die Validity Interval Analysis von Thrun und der Ansatz von Moraga vorgestellt, der auf dem Einsatz von sog. kompensierenden Operatoren basiert.

In Kapitel 6 geht es um "Approximation und Reduktion", d.h. um Möglichkeiten im Vorfeld der konkreten Regelextraktion aus einem Neuronalen Netz verschiedene Näherungen und Vereinfachungen durchzuführen. Wesentlich sind hier die Approximation der Sigmoiden durch eine Funktion wesentlich einfacherer Bauart, die sogenannte Rampenfunktion, sowie die Betrachtungen, wann und um welchen Preis in einem Neuronalen Netz die Anzahl der Neuronen in der verborgenen Schicht reduziert werden kann. U.a. wird hier der Least Weight Algorithmus vorgestellt, ein konkretes Verfahren zur Reduktion der Neuronenanzahl. Dabei werden stets entsprechende Güte-Aussagen formuliert, d.h. die Qualität der jeweiligen Näherung wird auch quantitativ erfassbar gemacht. Daneben wird in diesem Kapitel eine Reihe von Resultaten zu den hier diskutierten 3-Schicht-Netzen vorgestellt, beispielsweise das Symmetrie-Verhalten oder die Plateaubildung der Netzausgabefunktion.

Kapitel 7 stellt Verfahren zur Regelextraktion vor, die prinzipiell auf verschiedenste Netze angewendet werden können. Neben den elementareren Verfahren der äquidistanten Intervall-Unterteilung und der Numerischen Variationsmethode wird ein höherdimensionales Verfahren, die Regelextraktion durch schrittweise Verbesserung, vorgestellt. Ergänzend wird die parallel zur theoretischen Arbeit realisierte Java-Anwendung mit dem Arbeitstitel "Javanna" vorgestellt, mit der einige dieser Algorithmen praktisch erprobt werden können. Daneben sind die Hauptklassen für eine mögliche Wiederverwendung in weiteren java-basierten Netz- Applikationen konzipiert worden.

Abb. E.1: Schematischer Verfahrensablauf

Es sei noch angemerkt, dass im Rahmen dieser Arbeit die neue deutsche Rechtschreibung verwendet wird. Eine Ausnahme bilden lediglich die Zitate, bei denen die Schreibweise des jeweiligen Originals beibehalten wurde.

Weiterhin werden teilweise englischsprachige Begriffe verwendet, da diese im fachlichen Kontext gebräuchlich sind, auch wenn dadurch manche Formulierungen sehr gemischt-sprachig klingen.

I. Teil Grundlagen des Soft Computing

1 Künstliche Neuronale Netze

Das Konzept des Künstlichen Neuronalen Netzes lehnt sich dem biologischen Vorbild an. Dort werden in Nervensystemen Daten in Form von elektrischen Impulsen transportiert, die zu biochemischen Reaktionen führen. (Biologische) Neuronen sind miteinander vernetzt, deren Gesamtheit in Form des Neuronalen Netzes ist (in gewissen Grenzen) lernfähig und kann sein Verhalten an verschiedene (Umwelt-)Situationen anpassen. Dieser sog. *konnektionistische Ansatz* steht im Kontrast zur Arbeitsweise gewohnter DV-Software.

Das "Paradigma des Konnektionismus" (vgl. [Zell1997], S. 26) besagt, dass Informationsverarbeitung als Zusammenwirken einer sehr großen Anzahl einfacher Grundeinheiten (Neuronen o. Zellen) angesehen wird, die verstärkende oder dämpfende Signale an andere Einheiten senden.

Die "klassische" Informationsverarbeitung im Kontext der EDV ist wesentlich durch Algorithmen und Datenstrukturen geprägt. Das bedeutet, dass zunächst festgelegt wird, *was womit* getan werden soll, - und auch, *wie* es getan werden soll.

Interessanterweise löst sich die Software-Entwicklung mit dem objektorientierten Paradigma ein Stück weit davon, auf einer höheren Stufe bereits festlegen zu müssen, *wie* etwas en detail zu geschehen hat. Unterstützt von korrespondierenden Software-Entwicklungswerkzeugen (Tools) werden in zunehmendem Maße wiederkehrende Basisarbeiten (weg)gekapselt. Natürlich finden auch hier schlussendlich alle Einzelaktionen wirklich statt, für den auf einer höheren Ebene agierenden Software-Entwickler können allerdings zahlreiche Teilaktivitäten in Form einer black box einfach angestoßen werden, ohne dass der (Mikro-)Algorithmus bekannt sein oder gar explizit formuliert werden müsste. Damit kann sich der Arbeitsprozess der kreativen Problemlösung auf komplexere Dinge konzentrieren, ohne sich unnötigerweise mit zahlreichen, letztlich aber wohlbekannten Teilproblemen und deren Lösungen aufhalten zu müssen.

Für eine Problemlösung werden also zunächst geeignete Datenstrukturen generiert und anschließend (meist im Zuge schrittweiser Verfeinerung) Algorithmen entwickelt, die präzise definieren, in welcher Abfolge und ggf. unter welchen Voraussetzungen bestimmte Operationen auszuführen sind.

Es soll uns für eine einfache Gegenüberstellung von algorithmischen und konnektionistischen Vorgehensweisen genügen, sequentiell orientierte Algorithmen zu betrachten. Parallele Prozesse und Nebenläufigkeit werden hier bewusst ausgeklammert.

In unserem Zusammenhang resultiert daraus ein Konflikt: Einerseits sind Künstliche Neuronale Netze praktisch nur durch den Einsatz elektronischer Datenverarbeitung nutzbar, d.h. ein solches Netz muss durch Software emuliert werden. Andererseits "kennen" wir das Netz zunächst nur durch seine formale Beschreibung. Das bedeutet, dass wir - von einfachen Spezialfällen abgesehen - die DV-Umsetzung eines Neuronalen Netzes nicht ohne weiteres

verifizieren können. Es wäre zwar möglich, die DV-Realisierung zu *falsifizieren*, wenn wir nämlich an mindestens einem konkreten Beispiel zeigen können, dass das "echte" Netz zu einer vorgegebenen Eingabe eine andere Ausgabe als die Software-Variante liefert; für die Verifikation taugt dieses Vorgehen jedoch nicht.

Insbesondere dann wird eine solche Verifikation nicht gelingen, wenn das Netz durch entsprechende Lernverfahren auf eine bestimmte Situation adaptiert, also verändert worden ist. Auch in diesem Punkt ist es somit sehr hilfreich, möglichst gute Regeln zu kennen, die das Verhalten des Netzes beschreiben.

Im weiteren Verlauf der Arbeit soll es um sogenannte *Multilayer-Feedforward-Netze* (*MLFF*) gehen; diese werden in den nächsten Abschnitten detaillierter vorgestellt werden.

Bevor wir uns ausführlicher um Grundlagen Künstlicher Neuronaler Netze kümmern, wollen wir zunächst kurz die biologischen Vorbilder vorstellen.

1.1 Biologische Vorbilder

Zum besseren Verständnis und zur Motivierung des Neuronen-Begriffs im Rahmen der Künstlichen Neuronalen Netze sollen nachfolgend kurz einige Aspekte der natürlichen Neuronen präsentiert werden. Sehr viel ausführlicher findet sich eine Einführung in die Thematik biologischer Neuronen z.B. bei [Zell1997].

1.1.1 Aufbau eines Neurons

Grundlegende Bausteine in biologischen Systemen sind Neuronen. Ein Neuron ist dabei eine Zelle, die elektrochemische Reize aufnehmen und darauf durch die Erzeugung neuer elektrischer Impulse reagieren kann.

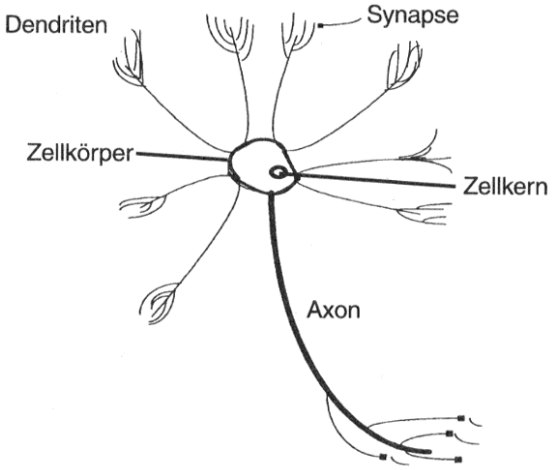


Abb. 1.1: Skizzenhafte Darstellung eines Neurons
Quelle: [Patt1997], S.19

Neuronen setzen sich, wie im Bild gezeigt, zusammen aus dem Zellkern und dem Zellkörper, einer Reihe sogenannter Dendriten, die über Synapsen Impulse erhalten, sowie einem Axonstrang, der für die Aus- bzw. Weitergabe des generierten Impulses sorgt.

1.1.2 Aktivierung eines Neurons

Die Dendriten eines Neurons erhalten fortlaufend elektrische Reize, die je nach Situation von den Synapsen verstärkt oder abgeschwächt und in das Innere der Zelle weitergeleitet werden. Dadurch ändert sich das elektrische Potenzial der Zelle. Sofern über eine gewisse Zeitdauer die Summe der eintreffenden Reize ein bestimmtes Potenzial überschreitet, wird das Neuron aktiv und gibt seinerseits einen elektrischen Impuls über das Axon an andere Neuronen weiter. Sind dagegen die eintreffenden Reize zu gering, so kehrt das Neuron in sein Ruhepotenzial zurück, es findet keine Ausgabe über das Axon statt.

Wir wollen für das Weitere festhalten, dass - von den biologischen Fakten abstrahierend - ein Neuron mehrere Eingänge besitzen kann, die Eingaben verstärken oder abschwächen können, und dass es eine Ausgabe besitzt, die ihrerseits als Eingabe für nachfolgende Neuronen dienen kann.

1.1.3 Nervensysteme

Aus Sicht der Informationsverarbeitung im Gehirn ist das Nervensystem die wesentliche Komponente. Es setzt sich aus einer großen Zahl von Neuronen (in der Hirnrinde größenordnungsmäßig 10^{11} Neuronen) zusammen, die einerseits eigenständig sind, andererseits aber erst durch ihr Zusammenwirken das Verarbeiten komplexer Informationen leisten können. Eine gute einführende Übersicht über Struktur und Funktionsweise des menschlichen Gehirns findet sich bei [Zell1997], S.59ff.

Dabei existieren die unterschiedlichsten Verbindungsstrukturen zwischen den Neuronen; mathematisch verwenden wir dafür den Begriff der Netz-Topologie. Es gibt sowohl vergleichsweise einfache Schichtenmodelle, als auch rückgekoppelte (rekursive) Netz-Strukturen.

Bis heute ist der Aspekt der Lernfähigkeit von Nervensystemen auf der Ebene einzelner Zellen nicht richtig verstanden; es erhebt sich dabei auch die Frage, ob eine einzelne Zelle überhaupt in dem Sinne “lernfähig” ist, oder ob dies nicht immanent (ausschließlich) eine Eigenschaft des komplexeren Systems ist, zu dem die Zelle gehört.

Tatsache ist, dass die Lernfähigkeit des gesamten Systems früh vorbereitet wird; bereits in einer frühen Wachstumsphase wird das Nervensystem ausgeprägt und wächst später im Vergleich dazu nur noch marginal. Nach aktuellem Erkenntnisstand finden Lernvorgänge in Form physischer und biochemischer Veränderungen der Neuronen und der zwischen ihnen geschalteten Synapsenstruktur statt.

1.2 Grundlagen Künstlicher Neuronaler Netze

Künstliche Neuronale Netze (KNN) sind informationsverarbeitende Systeme, die aus einer (endlichen) Anzahl von Zellen (sog. *Neuronen*) bestehen, die in einer topologischen Struktur von Verbindungen miteinander verbunden sind. Dies ist eine Vorab-Erläuterung, die die formale Definition eines Neuronalen Netzes nicht ersetzen soll; vgl. hierzu Abschnitt 1.2.2 (S. 19 ff). In englischsprachiger Literatur werden diese Netze mit *ANN* für *artificial neural networks* abgekürzt. Ausführliche Einführungen in diese Thematik finden sich u.a. bei Zell [Zell1997], Rojas [Roja1996], Brause [Brau1995], Grauel [Grau1992], Kratzer [Krat1993] oder Hecht-Nielsen [Hech1990].

Eine sehr kurze Vorstellung in das Konzept des Künstlichen Neuronalen Netzes vermittelt auch Dewdney in seinem "Turing Omnibus" ([Dewd1995], S.259ff); dieses Buch ist im übrigen eine sehr empfehlenswerte Lektüre für jeden, der einen gewissen Überblick über Theorie und Praxis ausgewählter Kapitel der Informatik erhalten möchte.

Nachfolgend sollen in relativ kompakter Form die für die vorliegende Arbeit benötigten Grundlagen vorgestellt werden.

1.2.1 Geschichtlicher Abriss der Neuronalen Netze

Die Geschichte der (Künstlichen) Neuronalen Netze ist beinahe so alt wie die der ersten programmierbaren Computer. Die nachstehende Zusammenstellung lehnt sich an die entsprechenden Kapitel bei [Patt1997] und [Zell1997] sowie die Ausführungen bei [Tenh2000] an.

Schon im Jahre 1943 stellen McCulloch und Pitts in [McCu1943] künstliche Zellen vor, die über zwei mögliche Statuszustände verfügen und ein Schwellenwertverhalten zeigen. Wird mit der Summe der Eingaben der jeweilige Schwellenwert überschritten, gibt die Zelle eine 1 aus, andernfalls eine 0. McCulloch und Pitts zeigen, dass bereits mit Netzen dieses einfachen Typs prinzipiell jede logische oder arithmetische Funktion dargestellt werden kann.

1949 beschreibt der Physiologe Donald O. Hebb in [Hebb1949] die aus heutiger Sicht klassische Lernregel, derzufolge eine Synapse immer verstärkender wirkt, wenn ein über sie eintreffender Reiz oft zu einer Ausgabeaktivität und zur Erregung des nachfolgenden Neurons führt. Zahlreiche spätere Lernalgorithmen wurden durch die Hebbsche Lernregel beeinflusst. Auf die Anpassung und das Lernen eines Neuronalen Netzes werden wir in Abschnitt 1.2.3 ausführlicher zu sprechen kommen.

1957/58 entwickeln Frank Rosenblatt, Charles Wightman und andere am MIT den ersten Neurocomputer, das Mark I Perceptron (vgl. hierzu [Rose1958]). Dieser Computer wurde für Mustererkennungsprobleme eingesetzt und modelliert außer den Schwellwerten auch multiplikative Gewichte, die der jeweiligen Situation angepasst werden können. 1959 erscheint Rosenblatts Buch "Principles of Neurodynamics" [Rose1959], in dem verschiedene Varianten des Perceptron-Modells vorgestellt werden.

Kurz darauf stellen 1960 Bernard Widrow und Marcian E. Hoff in [Widr1960] das "adaptive linear element" (Adaline) vor, ein schnell lernendes und an verschiedene Situationen anpassbares System. Das Adaline arbeitet mit Schwellenwertfunktionen und reellwertigen

Ausgaben. Das Verfahren zum Einstellen der Gewichte findet bis heute in Form der δ-Regel Einsatz.

Einen guten Überblick über die Entwicklungen dieser Zeit gibt [Nils1965].

1969 weisen Marvin Minsky und Seymour Papert in ihrem Buch "Perceptrons" [Mins1969] darauf hin, dass das mathematische Modell des Perceptrons zahlreiche Problemklassen überhaupt nicht lösen kann. Für einige Jahre sorgt dieses Resultat zu einem deutlichen Rückgang der Forschung auf dem Gebiet der Neuronalen Netze.

Als Beispiel soll hier das *XOR-Problem* erwähnt werden, das mit einem Perceptron nicht gelöst werden kann. Hierbei geht es um die Repräsentation der logischen "exclusive or"-Funktion f mit f(0,0):=0, f(1,0):=1, f(0,1):=1, f(1,1):=0. Es zeigt sich, dass ein Perceptron nur sog. "linear separable Klassen" in der Ausgabe trennen kann. (Zwei disjunkte Mengen A und B in einem n-dimensionalen Vektorraum sind linear trenn- oder separierbar, wenn es eine sie trennende Hyperebene gibt. Siehe hierzu [Roja1996], S. 60f, und [Zell1997], S.99ff.)

1974 publiziert Paul Werbos im Rahmen seiner Dissertation [Werb1974] erstmals das Backpropagation-Lernverfahren, das allerdings zwölf Jahre später von Rumelhart u.a. (in der sogenannten PDP-Gruppe[2]) mit deutlich größerer Wirkung veröffentlicht wird [Rume1986]. Die hier behandelte Klasse von Neuronalen Netzen kann prinzipiell auch nichtlineare Probleme lösen.

Seit etwa 1980 hat das Gebiet der Künstlichen Neuronalen Netze wieder deutlich an Forschungsinteresse gewonnen. Teilweise ist dies begründet durch finanzielle Unterstützung etlicher Projekte (u.a. der amerikanischen DARPA[3]), wesentlich hat aber auch die rasante Weiterentwicklung der seriellen Computertechnologie dazu beigetragen, mit der die Simulation der eigentlich massiv parallelen Neuronalen Netze mit akzeptabler Performance umgesetzt werden kann.

Zu den Meilensteinen der Entwicklung der Neuronalen Netze gehören auch die von Teuvo Kohonen entwickelten und nach ihm benannten Netze ([Koho1982], [Koho1984], [Koho1997]).

Schließlich seien auch die von Stephen Grossberg erfundenen ART-Netzwerke erwähnt, (vgl. hierzu die Arbeit von Cohen und Grossberg [Cohe1983]), die dem biologischen Vorbild entsprechend in der Lage sind, einmal Gelerntes je nach Relevanz dauerhaft abzuspeichern oder aber auch wieder zu vergessen und so Raum für neu zu Lernendes zu bereiten. Die Abkürzung ART steht hierbei für "Adaptive Resonanztheorie".

1.2.2 Einführung in den Aufbau Künstlicher Neuronaler Netze

Nachfolgend sollen die (im Rahmen dieser Arbeit) wichtigsten Begriffe und Notationen aus dem Bereich der Grundlagen Neuronaler Netze vorgestellt werden. Für weitergehende Informationen sei auf die einschlägige Literatur verwiesen. Eine kleine Literaturauswahl wurde bereits zu Beginn des Abschnitts 1.2 auf S. 18 genannt.

---

[2] PDP steht in diesem Zusammenhang für *parallel distributed processing*.

[3] DARPA steht für *Defense Advanced Research Projects Agency*, die zentrale Forschungsbehörde des US-amerikanischen Verteidigungsministeriums.

Die Zellen eines Künstlichen Neuronalen Netzes sind dem biologischen Neuronen-Vorbild angelehnt, naturgemäß aber stark idealisiert wie im nachstehenden Bild gezeigt.

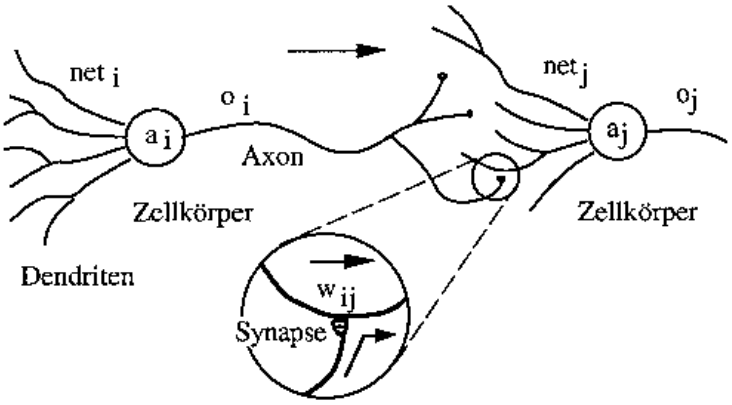


Abb. 1.2: Idealisierte Darstellung zweier Neuronen
Entnommen aus [Zell1997], S. 71

Künstliche Neuronen bestehen in dieser vereinfachten Darstellung aus den Dendriten, die die Eingabe zu diesem Neuron aufsummieren (im Bild mit net_i bzw. net_j dargestellt), dem Zellkörper und einem Axon, das die Ausgabe (o_i resp. o_j) der Zelle nach außen bzw. über Synapsen an die Dendriten nachfolgender Zellen weiterleitet. Die Stärke dieser Synapsen wird i.a. durch ein numerisches Verbindungsgewicht (w_{ij}) repräsentiert.

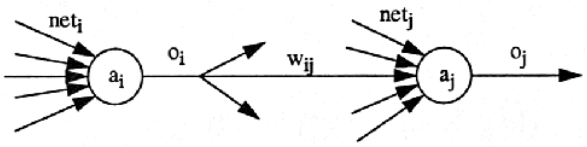


Abb. 1.3: Skizzenhafte Darstellung der Datenweitergabe zwischen Neuronen
Entnommen [Zell1997], S. 72

Präzisieren wir, was wir unter einem (Künstlichen) Neuron verstehen wollen. Mit dieser Arbeitsdefinition folgen wir weitgehend der Darstellung von Tenhagen in [Tenh2000].

Da wir uns grundsätzlich mit Künstlichen Neuronalen Netzen und Künstlichen Neuronen befassen werden, lassen wir zur besseren Lesbarkeit das Attribut "künstlich" im Folgenden meist weg und betonen umgekehrt, wenn die Begrifflichkeiten der biologischen Vorbilder gemeint sind.

Definition 1.1 (Neuron)

Ein *Neuron* ist eine datenverarbeitende Einheit, die sich als Tupel $(\vec{w}, S, f_{act}, f_{out})$ schreiben lässt. Es nimmt von seiner Außenwelt Daten in Form eines Eingabevektors $\vec{x} = (x_1, ..., x_m)$ (mit $m \geq 1$ skalaren Werten) in Empfang. Dabei sind $\vec{w} = (w_1, ..., w_n)$ der *Gewichtsvektor* (mit $n \geq 1$), S der sogenannte *lokale Speicher* der Einheit, $f_{act} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ die *Aktivierungsfunktion* und $f_{out} : \mathbb{R} \to \mathbb{R}$ die *Ausgabefunktion* des Neurons.

Die Funktion $f_{out} \circ f_{act}$ heißt *Transferfunktion* des Neurons.

Die *Ausgabe* y des Neurons ist definiert durch $y := f_{out}(f_{act}(\vec{x}, \vec{w}))$. Zu einem gegebenen Netz mit festem Gewichtsvektor $\vec{w} = (w_1, ..., w_n)$ wird die Ausgabe(funktion) des Neurons im Folgenden auch mit $out(\vec{x})$ notiert.

Die Aktivierungs- und die Ausgabefunktion können hierbei noch von den Informationen im lokalen Speicher *S* abhängen. In dem lokalen Speicher können (zusätzlich zum Gewichtsvektor $\vec{w}$) weitere Parameter abgelegt werden. Aus Gründen der Übersichtlichkeit verzichtet man jedoch meist darauf, *S* in die Parameterliste dieser Funktionen explizit mit aufzunehmen. Abbildung 1.4 zeigt eine grafische Darstellung für ein solches Neuron.

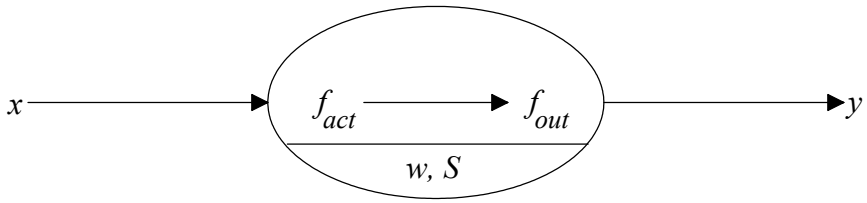


Abb. 1.4: Schematische Darstellung eines Neurons

Bemerkung 1.2

a) In der Praxis ist häufig $m = n$, und die Aktivierungsfunktion f_{act} wird als das Skalarprodukt $\vec{w} \cdot \vec{x} = \sum_{i=1}^{n} w_i x_i$ definiert.
   Wir wollen, wiederum zur einfacheren Verständlichkeit der Formeln, bis auf weiteres nicht zwischen einem Vektor und seinem Transponierten unterscheiden.

b) Zunächst einmal betrachten wir den reellwertigen Bereich, d.h. die hier genannten Skalare (etwa die o.e. Gewichtsvektoren) können als reelle Zahlen interpretiert werden. Später können dies aber auch Fuzzy-Zahlen sein.

Selbstverständlich interessieren wir uns nicht für ein isoliertes Neuron, sondern für den Zusammenschluss mehrerer Neuronen zu einem Neuronalen Netz; erst dieses verfügt über die relevanten Eigenschaften wie Lernfähigkeit etc.

Definition 1.3 (Neuronales Netz)

Gegeben sei eine endliche, nichtleere Menge N von Neuronen. Ein *Neuronales Netz* ist ein um Verbindungen zur "Außenwelt" erweiterter, gerichteter Graph (N,V), wobei die Menge der Kanten V die Menge der Verbindungen zwischen je zwei Neuronen definiert.

Es gilt weiterhin: Jedes Neuron kann eine beliebige, endliche Menge an Verbindungen empfangen, über die die Eingaben für die Aktivierungsfunktion eintreffen.

Jedes Neuron kann seine Ausgabe ebenfalls über eine beliebige, endliche Menge von Verbindungen weitergeben. Der Deutlichkeit halber sei hier betont: ein Neuron generiert in einer konkreten Situation nur genau *einen* Ausgabewert; dieser kann allerdings über mehrere Verbindungen weitergeleitet werden.

In Ergänzung zur strengen Graphendefinition gibt es Verbindungen, die außerhalb des Graphen beginnen und zu Neuronen aus N führen. Diese Neuronen werden Eingabeneuronen

genannt. Entsprechend gibt es bei endlich vielen Neuronen abgehende Verbindungen in die Außenwelt; diese Neuronen heißen Ausgabeneuronen.

Die so definierte Graphenstruktur wird *Netztopologie* genannt.

Die Begrifflichkeiten aus der Graphentheorie können ausführlicher u.a. bei [Jung1994] nachgelesen werden.

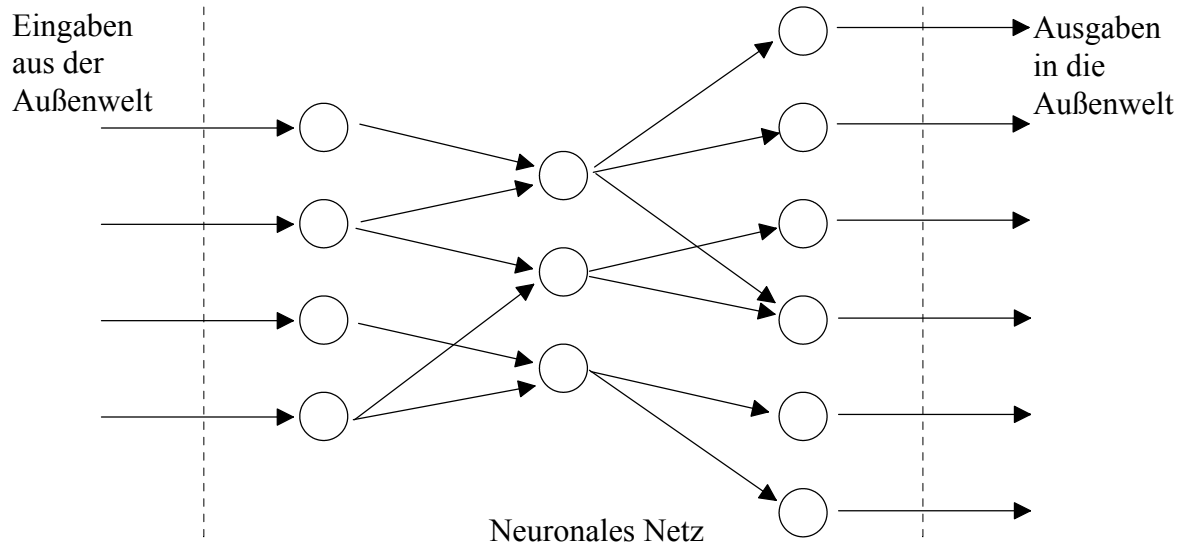


Abb. 1.5: Veranschaulichung eines Neuronalen Netzes

Bemerkung 1.4

Obiges Bild zeigt bewusst nicht das auf Grund der Definition "allgemeinstmögliche" Netz, sondern deutet bereits die für den weiteren Verlauf dieser Arbeit relevante Spezialisierung an.

Bemerkung 1.5

Um für die Praxis brauchbare Netze zu erhalten, könnte die Definition des Neuronalen Netzes weiter verschärft werden. Beispielsweise liegt es nahe, dass alle Neuronen aus der Menge N auch in entsprechenden Verbindungen vorkommen, d.h. dass der Graph aus einer Zusammenhangskomponente besteht. Mehr noch: sinnvollerweise wird man fordern, dass die Ausgabe jedes Neurons an andere Neuronen oder die Außenwelt weitergegeben wird, d.h. dass keine "toten Enden" existieren.

Wir wollen den Formalismus jedoch nicht zu weit führen, da wir uns ohnedies auf speziellere Netztopologien konzentrieren werden. Hierzu definieren die wesentlichen Begriffe des vorwärtsgerichteten Netzes und der Schicht eines (solchen) Neuronalen Netzes. Eine allgemeinere Definition des Begriffes "Schicht" findet sich z.B. bei [Tenh2000].

Definition 1.6 (Vorwärtsgerichtetes Netz, Feedforward-Netz)

Ist der Graph des Neuronalen Netzes (N,V) azyklisch, so spricht man von einem *vorwärtsgerichteten* oder *Feedforward-Netz*.

Bei einem solchen Netz gibt es also keinen Pfad gerichteter Verbindungen, der von einem Neuron zu sich selbst zurückführt.

Definition 1.7 (Schicht, Eingabeschicht, Ausgabeschicht, verborgene Schicht)

Es sei (N,V) ein vorwärtsgerichtetes Neuronales Netz.

Existieren endlich viele, nichtleere und paarweise disjunkte Teilmengen S_1, S_2, ..., S_m von N, so dass für $1 \leq i \leq m$ und alle Neuronen aus S_i gilt: alle eintreffenden Verbindungen kommen aus der Außenwelt oder von Neuronen aus S_j mit j<i und alle abgehenden Verbindungen gehen in die Außenwelt oder zu Neuronen aus S_k mit i<k, so heißen die Mengen S_1, S_2, ..., S_m *Schichten* des Neuronalen Netzes.

Das Netz wird aufsteigend angeordnetes (Schichten-)Netz mit m Schichten genannt. Speziell heißt S_1 die *Eingabeschicht* des Netzes, falls S_1 genau die Menge aller Neuronen des Netzes ist, die eine eintreffende Verbindung aus der Außenwelt besitzen. Analog wird S_m als *Ausgabeschicht* des Netzes bezeichnet, falls S_m genau die Menge aller Neuronen ist, die eine abgehende Verbindung in die Außenwelt besitzen. Die dazwischenliegenden Schichten S_2, S_3, ..., S_{m-1} werden *verborgene Schichten* genannt.

Bemerkung

Für die Fälle $m = 1$ und $m = 2$ gibt es natürlich keine verborgenen Schichten. In der Praxis ist jedoch $m \geq 3$.

Bemerkung

Einige weitere Begriffsbildungen zu Neuronalen Netzen sollen kurz aufgeführt werden.

1. Besitzt der Graph des Neuronalen Netzes Zyklen, so heißt das Netz *rekurrent*. Sofern eine "Hauptrichtung" des Datenflusses feststeht, werden die Verbindungen, die gegen diese Richtung laufen und zu den Zyklen führen, *Rückkopplungen* genannt.
   Dabei wird unterschieden zwischen *direkten* Rückkopplungen, bei denen ein Neuron mit sich selbst verbunden ist, und *indirekten* Rückkopplungen, bei denen der betreffende Zyklus mehrere Neuronen umfasst.
2. Ist jedes Neuron mit jedem anderen Neuron verbunden, so nennt man das Netz *total verbunden*; ist dazu noch jedes Neuron auch mit sich selbst verbunden, so heißt das Netz *total verbunden mit direkter Rückkopplung*.
3. In einem aufsteigend angeordneten Netz gemäß Definition 1.7 können Verbindungen zwischen Neuronen aus nicht benachbarten Schichten (also aus S_i und S_j mit j > i+1) existieren; solche "abkürzenden" Verbindungen werden *Shortcuts* oder *Shortcut Connections* genannt.

Definition 1.8 (vollständig schichtweise verbunden)

Ein aufsteigend angeordnetes Netz heißt *vollständig schichtweise verbunden* bzw. *vollständig verbundenes Schichtennetz*, wenn jedes Neuron einer Schicht mit genau allen Neuronen der Nachbarschichten verbunden ist.

Bemerkung 1.9

Für Neuronen der Eingabe- bzw. Ausgabeschicht ist damit selbstverständlich auch gemeint, dass bei ersteren alle Eingaben aus der Außenwelt kommen bzw. bei letzteren alle Ausgaben in die Außenwelt gehen (und dies die einzigen Schichten sind, in denen Neuronen direkten Kontakt zur Außenwelt besitzen).

In einem vollständig verbundenen Schichtennetz läuft der Datenfluss sequentiell durch die Schichten: Außenwelt-Eingabe → Schicht 1 → ... → Schicht m → Ausgabe-Außenwelt.

Definition 1.10 (K-L-M-Netz)

Im Folgenden bezeichnen wir ein vollständig verbundenes Neuronales Netz mit drei Schichten kurz als *K-L-M-Netz*. Dabei ist *K* die Anzahl der Eingabeneuronen, *L* die der verborgenen (oder Hidden) Neuronen und *M* die Anzahl der Ausgabeneuronen.

Bemerkung 1.11

Soweit bisher behandelt, wird ein Neuronales Netz wesentlich durch seine Topologie, den zugrundeliegenden Verbindungsgraphen, sowie den Aufbau der einzelnen Neuronen (und dort insbesondere die Aktivierungs- und Ausgabefunktion und den Gewichtsvektor $\overrightarrow{w}$) bestimmt.

Als Aktivierungsfunktion wird, wie in Bemerkung 1.2 bereits erwähnt, oftmals das Skalarprodukt verwendet. Es sind jedoch statt dieser gewichteten Summen auch andere Funktionen nutzbar, beispielsweise eine Metrik, die den Abstand zwischen dem Gewichtsvektor und dem Eingabevektor misst. Auch die verwendete Metrik kann variieren (Euklidische Metrik, Hamming-Distanz). Vgl. Hierzu [Patt1997].

Typische Beispiele von Ausgabefunktionen (vgl. nachstehende Abbildungen) sind die Identität(sfunktion)[4], eine binäre Schwellenwertfunktion, der Tangens hyperbolicus tanh(x) oder eine sigmoide Funktion wie z.B. die logistische Funktion $x \to \frac{1}{1+e^{-x}}$. (Vgl. hierzu die nachfolgende Definition.)

[4] Je nach behandelter Problemstellung werden die Ergebnisse der Neuronen der Ausgabeschicht oft nicht weiter transformiert, sondern unverändert nach außen gegeben.

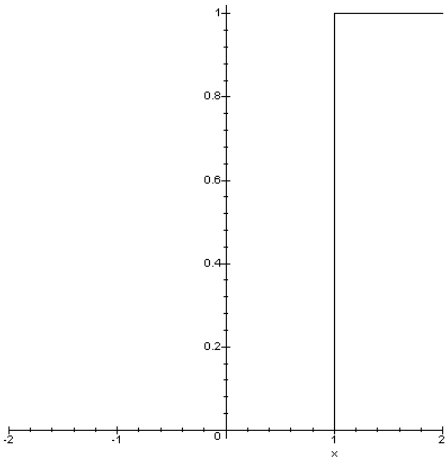


Abb. 1.6: Beispiel einer binären Schwellenwertfunktion ($x \to 1$ *falls* $x \geq 1$, *sonst* 0)

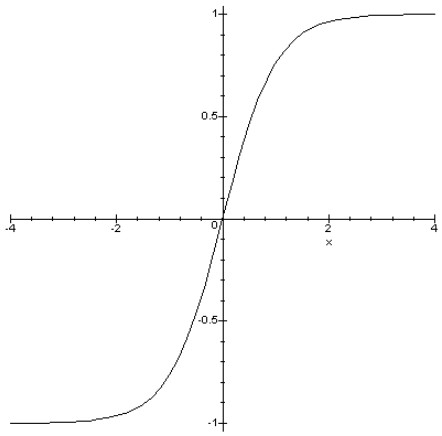


Abb. 1.6a: Tangens hyperbolicus

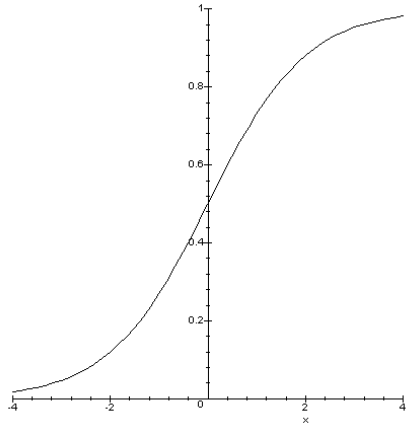


Abb. 1.6b: Logistische Funktion $x \to \frac{1}{1+e^{-x}}$.

Definition 1.12 (Sigmoide Funktion)

Eine stetige, monoton wachsende Funktion $f\colon \mathbb{R} \to [0,1]$ heißt (eine) *sigmoide Funktion*, wenn ihre Funktionswerte "von 0 nach 1 verlaufen", d.h. mathematisch:

$$\lim_{x\to\infty} f(x) = 1 \quad \text{und} \quad \lim_{x\to-\infty} f(x) = 0.$$

Notation 1.13

Speziell werden wir im Folgenden als "die" *sigmoide Funktion* die auf $\mathbb{R}$ definierte logistische Funktion $x \to \frac{1}{1+e^{-x}}$ bezeichnen, die wir auch mit *sigm* abkürzen wollen.

Bemerkung 1.14

Die ersten beiden Ableitungen der Sigmoiden *sigm* berechnen sich zu

$$sigm' : x \to \frac{e^{-x}}{(1+e^{-x})^2} \text{ , } sigm'' : x \to \frac{2(e^{-x})^2}{(1+e^{-x})^3} - \frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-2x}-e^{-x}}{(1+e^{-x})^3}.$$

Für die erste Ableitungsfunktion der Sigmoiden gilt:

$$sigm'(x) = sigm(x) \cdot (1 - sigm(x)).$$

Nachstehend werden diese drei Funktionen graphisch dargestellt. Insbesondere nimmt die erste Ableitungsfunktion der Sigmoiden ihr Maximum von $\frac{1}{4}$ bei $x = 0$ an.

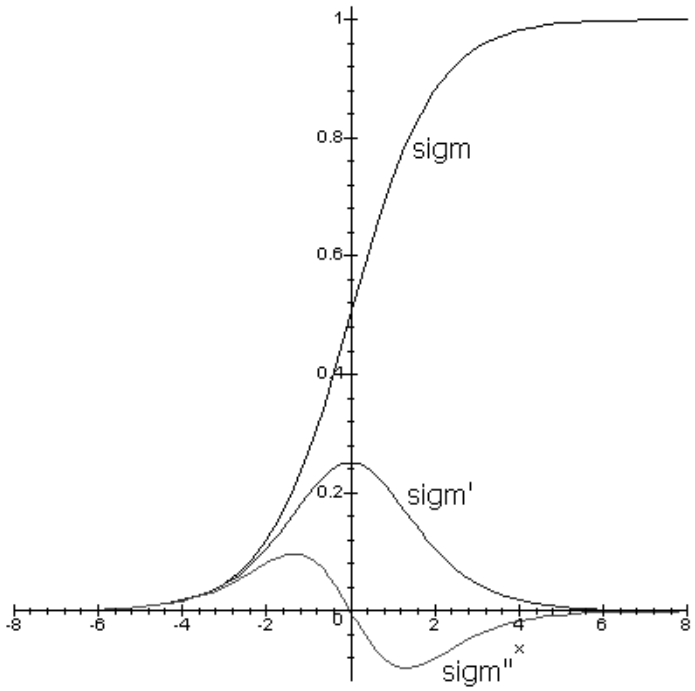


Abb. 1.7: Die Graphen der sigmoiden Funktion *sigm* und ihrer ersten beiden Ableitungen *sigm'*, *sigm''*

Bemerkung 1.15 (Umkehrfunktion der Sigmoiden)

Die Sigmoide Funktion *sigm* ist streng monoton wachsend und stetig, also auch umkehrbar. Ihre Umkehrfunktion $sigm^{-1}$ hat die Gestalt $sigm^{-1}(x) = -\ln(\frac{1}{x} - 1)$ und ist auf dem Intervall]0,1[definiert.

Beispiel 1.16

Zur Illustration wird nachstehend ein sehr einfaches Neuronales Netz mit zwei Eingabeneuronen, zwei Neuronen in der verborgenen Schicht und einem Ausgabeneuron dargestellt.

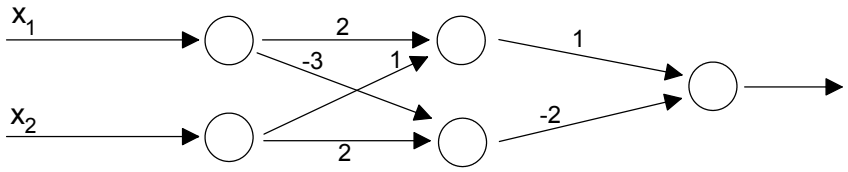


Abb. 1.8: Darstellung eines einfachen 2-2-1-Netzes

Die Gewichte zwischen den Neuronen sind im obigen Bild eingetragen. Die formale Aktivierungsfunktion ist das Skalarprodukt, die Ausgabefunktion die Sigmoide.

Damit ergibt sich im hier gezeigten Netz zur Eingabe eines Vektors $\vec{x} := (x_1, x_2)$ die Ausgabe

$$out(\vec{x}) = 1 \cdot sigm(2x_1 + 1x_2) - 2 \cdot sigm(-3x_1 + 2x_2).$$

Zur Visualisierung wird in den beiden nächsten Abbildungen der Funktionsgraph der Netzausgabefunktion aufgetragen.

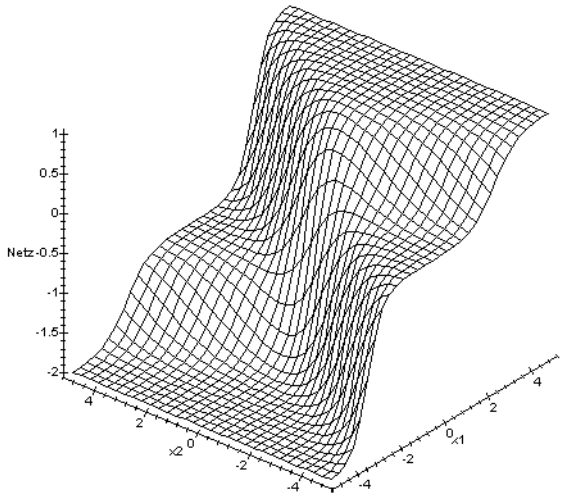


Abb. 1.9: Netzausgabe des 2-2-1-Netzes

Im vorigen Bild wird in jeder Eingabedimension der Bereich zwischen -5 und +5 aufgetragen; im zweiten Graphen ist das etwas "makroskopischere" Verhalten über dem Eingabebereich$[-25,+25]\times[-25,+25]$ dargestellt.

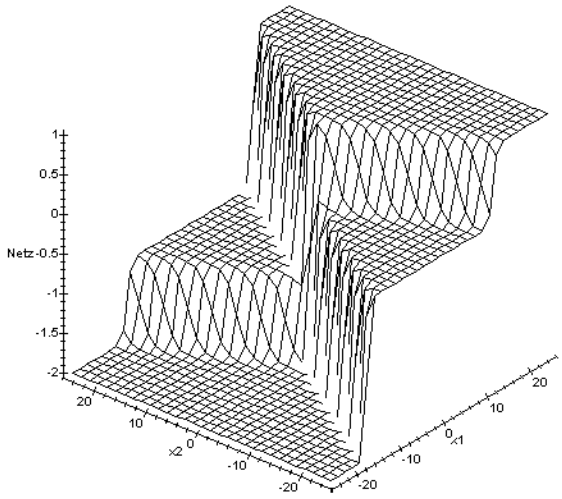


Abb. 1.10: Netzausgabe des 2-2-1-Netzes über einem größeren Eingabebereich

Neben den sigmoiden Ausgabefunktionen werden in der Praxis für verschiedene Netztypen noch andere Funktionen eingesetzt. Wir greifen die sogenannten "RBF-Netze" auf, da wir später ein Beispiel damit formulieren werden. Dazu definieren wir zunächst den passenden Funktionstypus.

Definition 1.17 (Radial Basis Function, RBF)

Eine Funktion $\psi : \mathbb{R}^K \to \mathbb{R}$ heißt *radiale Basisfunktion* (*radial basis function*, *RBF*), wenn gilt: ψ lässt sich darstellen in der Form $\psi(\vec{x}) = h(\left\| \vec{x} - \vec{m} \right\|)$, wobei h eine monoton fallende Funktion $h : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ ist mit $\lim_{x\to\infty} h(x) = 0$ und $\vec{m} \in \mathbb{R}^K$ ein festgewählter Vektor, das sogenannte *Zentrum* der Funktion.

Beispiel 1.18

Nachstehende Abbildung illustriert eine Funktion h ("Glockenkurve") mit den oben beschriebenen Eigenschaften, $h(x) := \exp(-0.5x^2)$.

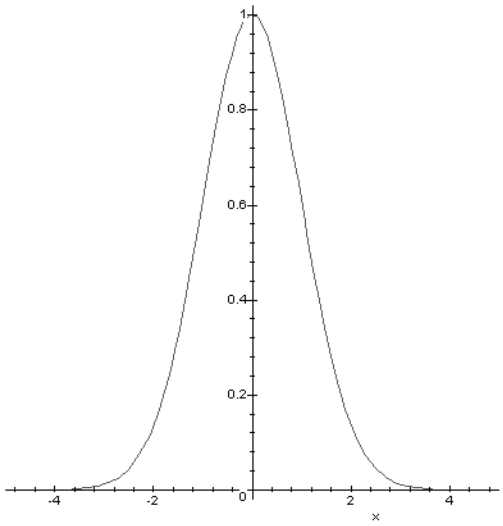


Abb. 1.11: Graph einer Glockenkurve h

Für K=2 kann damit eine radiale Basisfunktion konstruiert werden. Dazu sei das Zentrum $\vec{m} := \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ gewählt. Die Funktion ψ mit $\psi(\vec{x}) = h(\|\vec{x} - \vec{m}\|)$ hat dann folgenden Funktionsgraphen.

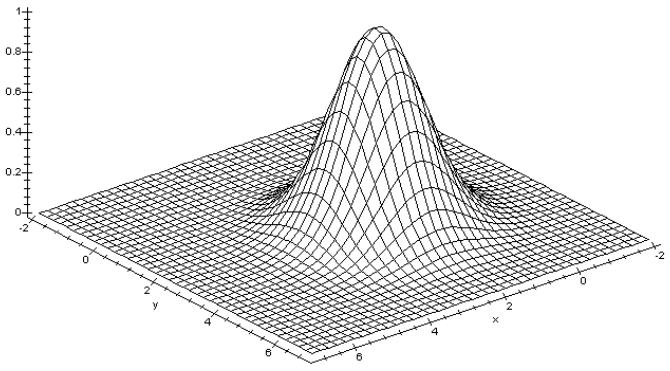


Abb. 1.12: Graph einer RBF

Man sieht, dass ψ eine Funktion ist, die nahe um das Zentrum $\vec{m} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ "groß" ist und von dort aus radial gegen 0 abfällt.

Kommen wir hiermit zu den RBF-Netzen, - das sind Feedforward-Netze mit im einfachsten Fall nur einer verborgenen Schicht und einem Ausgabeneuron. Auf diesen Fall wollen wir im Rahmen unserer Arbeit die Definition beschränken, auch wenn eine Verallgemeinerung ohne weiteres möglich ist.

Definition 1.19 (RBF-Netz)

Ein 3-schichtiges Feedforward-Netz, bei dem die Neuronen der verborgenen Schicht als Aktivierungsfunktion eine radiale Basisfunktion besitzen, heißt *RBF-Netz*. Die Eingabeschicht verfüge über K und die verborgene Schicht über L Neuronen; die Ausgabeschicht bestehe aus nur einem Neuron.

Die Ausgabe des RBF-Netzes wird beschrieben durch

$$out(\vec{x}) = \sum_{1 \le i \le L} W_i \cdot \psi(\| \vec{x} - \vec{c_i} \|),$$

dabei ist $\vec{x}$ der Eingabevektor, $W := (W_1, ..., W_L)$ der Vektor der Gewichte zwischen den Neuronen der verborgenen Schicht und dem Ausgabeneuron, $\vec{c_i}$ sind die Zentren der L radialen Basisfunktionen.

Als sog. *normierte Ausgabe* des RBF-Netzes wird definiert:

$$out_{norm}(\vec{x}) = \frac{\sum_{1 \le i \le L} W_i \cdot \psi(\| \vec{x} - \vec{c_i} \|)}{\sum_{1 \le i \le L} \psi(\| \vec{x} - \vec{c_i} \|)}.$$

Beispiel 1.20

Nachstehender Plot zeigt exemplarisch qualitativ die Ausgabefunktion eines RBF-Netzes mit fünf Neuronen in der verborgenen Schicht.

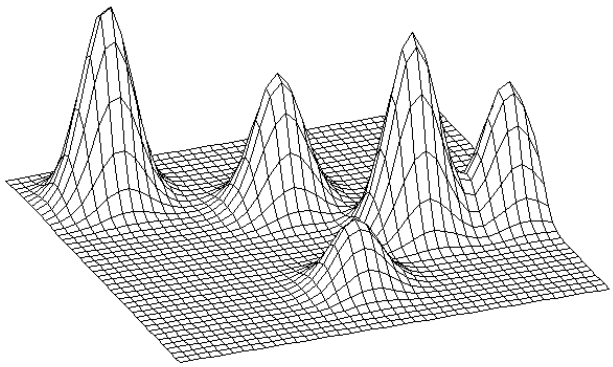


Abb. 1.13: Qualitative Darstellung der Ausgabe eines RBF-Netzes

Bemerkung 1.21

Ein konkretes, numerisches Beispiel für ein RBF-Netz findet sich später in Abschnitt 4.5.

Bemerkung 1.22

Wie eingangs erwähnt, kann unsere Definition eines RBF-Netzes in mancherlei Hinsicht verallgemeinert werden. Für unsere Zwecke genügt jedoch diese Form eines solchen Netzes. Weitergehende Ausführungen zu RBF-Netzen und radialen Basisfunktionen finden sich z.B. bei Brause in [Brau1995], S. 286ff, und bei Zell in [Zell1997], S. 225ff.

1.2.3 Training und Adaption Neuronaler Netze

Ein Neuronales Netz wird in der Praxis für einen konkreten Zweck eingesetzt, d.h. es soll gewisse Berechnungen durchführen, Daten klassifizieren, Muster erkennen etc. Mathematisch formuliert, soll ein Neuronales Netz eine vorgegebene Funktion darstellen - oder zumindest in einem zu präzisierenden Sinne approximieren.

Aus theoretischer Sicht gibt es ein besonders bemerkenswertes Resultat, das auf Kolmogorov [Kolm1957] zurückgeht und hier in einer auf Neuronale Netze adaptierten Formulierung von Hecht-Nielsen [Hech1990] kurz vorgestellt wird. Vgl. "Kolmogorov's Mapping Neural Network Existence Theorem" in [Hech1990] (S. 122).

Das ursprüngliche Resultat von Kolmogorov ist ein reiner Existenzsatz und bezieht sich auf die Darstellbarkeit höherdimensionaler (stetiger) Funktionen durch eindimensionale:
Ist $f: [0,1]^n \to \mathbb{R}^m$ eine stetige Funktion, dann lässt sich jede Komponente f_i darstellen in der Form $f_i(x) = \sum_{k=1}^{2n+1} g_i(\sum_{j=1}^{n} \lambda^k h(x_j + k\varepsilon) + k)$, $1 \le i \le m$; dabei sind λ eine reelle Konstante und h eine reellwertige, stetige, monoton wachsende Funktion, die beide nur von n, nicht aber von der darzustellenden Funktion f abhängen! Die positive rationale Konstante ε kann beliebig klein vorausgesetzt werden; von ihr und von der Funktion f hängen die reellwertigen, stetigen Funktionen g_i ab.

Satz 1.23 (Darstellungssatz von Kolmogorov)

Es seien $n, m \in \mathbb{N}$. Sei $f: [0,1]^n \to \mathbb{R}^m$ eine stetige Funktion; dann existiert ein dreischichtiges Neuronales Feedforward-Netz mit einer n-elementigen Eingabeschicht, einer verborgenen Schicht mit $2n+1$ Neuronen und einer Ausgabeschicht mit m Elementen, das die Funktion f exakt implementiert. Das bedeutet: zu einer Eingabe $\vec{x} = (x_1, ..., x_n)$ liefern f und das Neuronale Netz dasselbe Ergebnis, dieselbe Ausgabe.

Nun ist dies allerdings ein reines Existenztheorem; es ist derzeit kein Rezept bekannt, das zu einer beliebig vorgegebenen stetigen Funktion $f: [0,1]^n \to \mathbb{R}^m$ konstruktiv das entsprechende Netz angeben würde.

Bemerkung 1.24

Kolmogorovs Darstellungssatz lässt sich insoweit auf $L_2([0,1]^n)$, den Raum der quadrat-integrierbaren Funktionen über $[0,1]^n$, erweitern, als sich jede solche Funktion zu einer vorgegebenen ε-Genauigkeit durch ein geeignetes 3-schichtiges Backpropagation-Netz approximieren lässt. Vgl. hierzu [Hech1990], S. 132. Die Approximation versteht sich im Sinne der L_2-Norm.

Da man - von sehr einfachen Fällen abgesehen - zu einem gegebenen Problem kein Neuronales Netz "aus dem Ärmel schütteln" kann, ist es immanent wichtig, dass das Netz sich an die Problemstellung anpassen, adaptieren kann.

Hierzu dient ein Lernverfahren, ein Algorithmus, der das Ausgangsnetz dergestalt verändert, dass das gestellte Problem durch das modifizierte Netz möglichst gut gelöst wird. Die grobe Netztopologie wird man häufig durch Erfahrungswerte wählen können. Die numerischen Werte (wie die Anzahl der Neuronen oder die Gewichte) können in gewissen Grenzen

ausprobiert oder durch Zufallswerte vorbelegt werden. Auf die Qualitätsbeurteilung und Gütemessung wird in Abschnitt 1.2.4 (S. 36 ff) eingegangen werden.

Dem Neuronalen Netz werden in einer Trainingsphase eine Reihe von Eingaben (und je nach Vorgehensweise auch Soll-Ausgaben) vorgelegt, die gelernt werden sollen. Im Idealfall ist das Netz nach absolviertem Training generalisierungsfähig, d.h. es hat nicht nur die Trainingseingaben gelernt, sondern liefert auch für andere Eingaben korrekte oder zumindest gut approximierende Ausgaben.

Und trotz der obigen Existenzaussage ist jedoch keine allgemeine Aussage darüber möglich, ob sich stets zu einer gegebenen Funktion ein entsprechendes Netz durch ein geeignetes Lernverfahren anpassen lässt.

Diese Adaption des Netzes, der Lernprozess, kann generell in verschiedenen Formen stattfinden:

1. Veränderung der Gewichte,
2. Veränderung der Aktivierungs- oder der Ausgabefunktion,
3. Aufbau neuer Verbindung,
4. Entfernen bestehender Verbindungen,
5. Aufnahme neuer Neuronen,
6. Entfernen bestehender Neuronen.

Das Entfernen von Verbindungen kann natürlich dazu führen, dass sinnvollerweise auch ein Neuron entfernt werden muss. Entsprechend hat das Hinzunehmen oder Entfernen von Neuronen (Knoten) Einfluss auf die Verbindungsstruktur.

Es muss festgehalten werden, dass die numerische Computer-Simulation des Neuronalen Netzes und des trainierenden Lernprozesses manche Modifikationsmethoden nur unter erheblichem Aufwand ermöglicht. Auch deshalb ist die klassisch dominierende Vorgehensweise von Lernverfahren zweifelsohne die der fortgesetzten Veränderung der Gewichte.

Ein weiteres Unterscheidungsmerkmal von Lernverfahren ist das zugrundeliegende Paradigma. Üblich sind hier die folgenden Kategorien (vgl. [Lipp2004], [McCl1986]).

1. Überwachtes Lernen[5]:
   Hier werden dem zu trainierenden Netz Eingabe-Ausgabe-Kombinationen vorgelegt; anhand der Abweichung der Ist-Ausgabe des momentanen Netzes von der vorgegebenen Soll-Ausgabe kann das Netz mit einer entsprechenden Lernregel modifiziert werden. Hier ist zu beachten, dass ein Vergleich mit einer Soll-Ausgabe direkt nur für die Neuronen möglich ist, die auch etwas an die Außenwelt liefern; für Neuronen von verborgenen Schichten kann ein solcher Vergleich nicht durchgeführt werden.
2. Bestärkendes Lernen[6]:
   Gegenüber der vorherigen Form wird beim bestärkenden Lernen dem Netz lediglich qualitativ mitgeteilt, ob die zu einer Eingabe ermittelte Ausgabe als korrekt oder inkorrekt eingestuft wird.

---

[5] engl.: supervised learning.
[6] engl.: reinforcement learning.

Dieser Ansatz des Lernens führt zwar zu langsameren Algorithmen als das überwachte Lernen, dafür ist es gemessen am biologischen Vorbild plausibler.

3. Nicht überwachtes Lernen[7]:
   Hier gibt es keine externe Lehrer-Instanz. Das Neuronale Netz versucht über das Lernverfahren selbständig, in den vorgelegten Daten Klassifizierungsmuster zu erkennen und Ähnlichkeitsklassen zu bilden.
   Bekanntestes Beispiel Verfahren unüberwachten Lernens sind die Selbstorganisierenden Karten (self-organizing maps) von Kohonen ([Koho1982]).

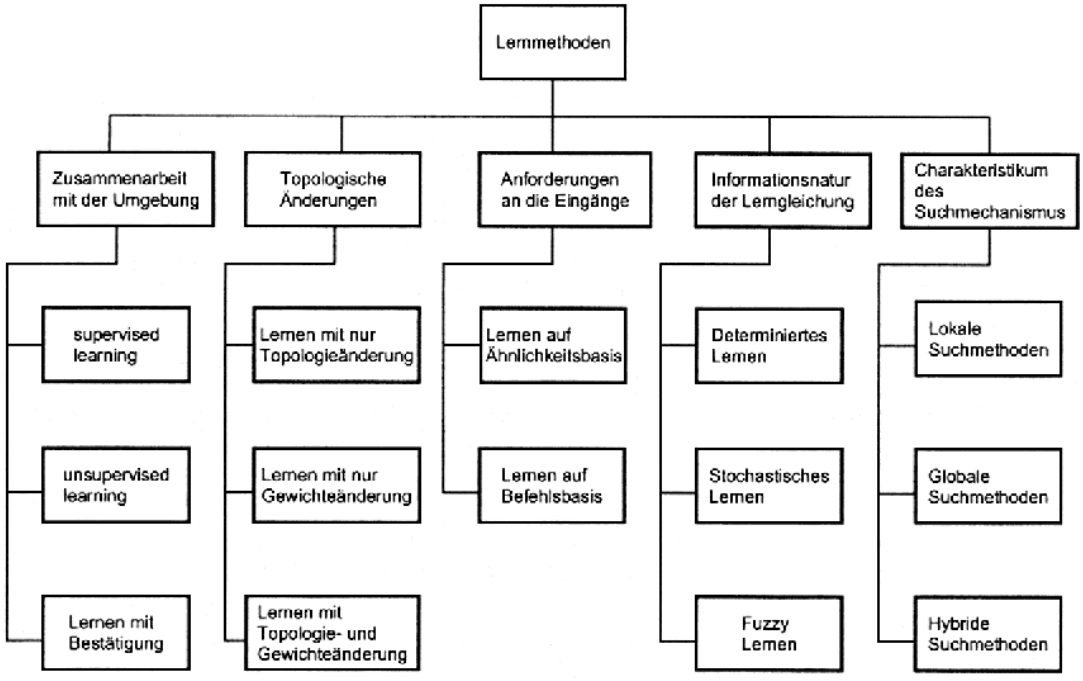


Abb. 1.14: Klassifizierung der Lernverfahren für Neuronale Netze
(Die Abbildung ist dem Buch von Aliev et al ([Alie2000]) entnommen.)

Aliev u.a. Geben die in Abb. 1.14 dargestellte Unterteilung von Lernverfahren für Neuronale Netze. Wir wollen nachfolgend einige der bekannten Lernregeln (zit. bei [Zell1997], S. 84ff) kurz vorstellen. Die Schreibweise w_{ij} für das Gewicht an der Verbindung von Neuron i zu Neuron j lehnt sich der Darstellung in den Abbildungen 1.2 und 1.3 (s. S. 20) an.

Bemerkung 1.25 (Hebbsche Lernregel)

Donald O. Hebb stellt in [Hebb1949] eine Lernregel auf, die als Grundlage zahlreicher moderner Lernverfahren dient. Es beruht auf einem Verstärkungsmechanismus einer Neuronen-Verbindung, wenn beide Neuronen gleichzeitig eine starke Aktivierung erfahren. Das heißt, dass das betreffende Verbindungsgewicht in dieser Situation vergrößert wird.

Mathematisch formuliert: $\Delta w_{ij} = \eta \cdot o_i \cdot a_j$; dabei ist Δw_{ij} die Veränderung des Gewichtes von Neuron i zu Neuron j, η eine konstante Lernrate, o_i die Ausgabe des Neurons i und a_j die Aktivierung von Neuron j.

---

[7] engl.: unsupervised learning, self-organized learning.

Es sei angemerkt, dass in der hier gezeigten Urform der Hebbschen Lernregel bei einer positiven Lernrate η die Gewichte nur größer werden können, sofern die Ausgabe- und die Aktivierungsfunktion nichtnegativ sind, wie es etwa bei binären Problemstellungen (mit den Werten 0 und 1) der Fall ist. Häufig werden in einer solchen Situation stattdessen z.B. die Werte -1 und +1 verwendet.

Bemerkung 1.26 (Delta-Regel oder Widrow-Hoff-Regel)

Bei dieser Lernregel ist die Gewichtsänderung proportional zur Abweichung der erwarteten von der tatsächlichen Ausgabe des Neurons.

$$\Delta w_{ij} = \eta \cdot o_i \cdot (t_j - o_j)$$

Hier ist o_j die Ausgabe des Neurons j, die verglichen wird mit der Soll-Ausgabe t_j, dem sog. "teaching value". Je nachdem, ob die tatsächliche Ausgabe größer oder kleiner als t_j ist, kann das Gewicht entsprechend vergrößert oder verkleinert werden. Schreibt man für $(t_j - o_j)$ kurz δ_j, so lautet diese Regel $\Delta w_{ij} = \eta \cdot o_i \cdot \delta_j$.

Bemerkung 1.27 (Backpropagation-Regel)

Eine Verallgemeinerung der Delta-Regel für Neuronen mit nichtlinearen Aktivierungsfunktionen ist Backpropagation. Die Aktivierungsfunktionen müssen semilinear (monoton und differenzierbar) sein. Auch diese Regel für die Gewichtsänderungen schreibt sich

$$\Delta w_{ij} = \eta \cdot o_i \cdot \delta_j,$$

wobei allerdings die Berechnung der δ_j komplexer ist. Für den in der Praxis häufig eingesetzten Fall der logistischen Aktivierungsfunktion $x \to \frac{1}{1+e^{-x}}$ ergibt sich die folgende Fallunterscheidung. Vgl. Bemerkung 1.2: es wird hier davon ausgegangen, dass sich die Eingabe eines bestimmten Neurons über das Skalarprodukt $\sum_{i=1}^{n} w_i x_i$ berechnen lässt.

$$\delta_j = \left\{ \begin{array}{ll} o_j(1-o_j)(t_j-o_j) & \textit{falls j eine Ausgabezelle ist} \\ o_j(1-o_j)\sum_k (\delta_k w_{jk}) & \textit{falls j kein Ausgabeneuron ist} \end{array} \right\}$$

Dabei läuft der Summationsindex k über alle Nachfolgerneuronen des hier betrachteten Neurons j, d.h. es werden rückwärts, ausgehend von der Ausgabeschicht, die Gewichtsänderungen ermittelt. Daher rührt auch der Name "*back propagation*", "Rückwärtsverteilung".

Für die allgemeine Formulierung der Backpropagation-Regel sei z.B. auf [Zell1997] verwiesen.

Bemerkung 1.28

Im Vergleich zum Vorbild der biologischen Neuronen und Netze ist festzuhalten, dass die hier vorgestellte Modellbildung stark idealisiert. Einige Aspekte werden dabei zwar übernommen, zahlreiche jedoch auch deutlich vereinfacht (oder sogar weggelassen). Vgl. hierzu [Zell1997], S.51ff.

Übernommen werden neben der grundsätzlichen Konzeption von Neuronen und Netzen u.a. die hohe Parallelität einer großen Zahl von Neuronen sowie die Veränderbarkeit von

Verbindungsgewichten oder auch der Netztopologie. Der Begriff “groß” ist hier relativ zu sehen; in der Computersimulation sind Netze mit einigen Tausend Neuronen bereits als groß zu bezeichnen, das biologische Vorbild des Gehirns wird auf die Größenordnung von 10^{11} Neuronen geschätzt!

Stark reduziert wird jedoch bei der Simulation die Zahl der Verbindungen pro Neuron, abstrahiert wird auch in der Frage der zeitlichen Vorgänge beim biologischen Vorbild.

Das Ziel des Trainierens eines Neuronalen Netzes ist nicht nur das möglichst gute Erlernen auf einer gegebenen Trainingsdatenmenge, sondern ebenfalls eine gewisse Generalisierungsfähigkeit. Muster, die ähnlich sind zu während des Trainings vorgelegten Daten, sollen möglichst genauso korrekt verarbeitet werden. Aber auch zu anderen Daten(-Eingaben) erhofft man sich von einem praktisch einsetzbaren Netz brauchbare Approximationsleistungen.

Im allgemeinen bedeutet dies, dass ein guter Kompromiss gefunden werden muss zwischen dem “zu perfekten” Erlernen einer gegebenen Menge von Trainingsdaten auf der einen und dem “möglichst guten” Verarbeiten der sonstigen, im praktischen Einsatz relevanten Daten auf der anderen Seite.

Zur praktischen Beschleunigung und Verbesserung des Lernverhaltens gehört auch, dass ggf. die Daten in sinnvoller Weise vorbereitet, vorverarbeitet werden. Nicht immer liegen reale Daten oder Situationsbeschreibungen in Form von numerischen Werten vor; bei qualitativen Merkmalen (wie etwa Eingruppierungen in endlich viele Kategorien) wird häufig eine künstliche Nummerierung gewählt, die für das Neuronale Netz aufgrund der konkret festgelegten Zahlenwerte bereits eine Bewertung darstellt. Ggf. hilft hier auch eine Umsetzung in (mehrere) binäre Muster. Dies meint, dass anstelle der numerischen Darstellung 1, 2, 3, etc. für jede der betrachteten Kategorien die binären Werte 0 und 1 (“liegt nicht vor”, “liegt vor”) verwendet werden können. In manchen Situationen hat dies den Vorteil, dass auch eine Mehrfachzuordnung einzelner Datenmuster zu Kategorien erfolgen kann; allerdings erhöht sich die Komplexität durch eine solche Modellierung. Eine solche Vorgehensweise wird in [Lipp1996] vorgestellt.

Aber selbst bei Vorliegen numerischer Werte, hilft häufig eine geschickte Skalierung, ein günstigeres Lernverhalten zu erzielen. Wenn nicht gezielt Gründe dafür sprechen, eine Eingangskomponente (im Szenario einer mehrdimensionalen Eingabe) besonders stark zu betonen, dann kann i.d.R. davon ausgegangen werden, dass eine nivellierende Skalierung hilfreich ist. Alleine wegen numerischer Algorithmen sollten die Größenordnungen der zu verarbeitenden Daten nicht sehr weit auseinanderliegen.

Naturgemäß kann generell nicht vorhergesagt werden, wie das trainierte Netz auf Daten außerhalb der Trainingsmenge reagieren wird. Darum wird die Generalisierungsfähigkeit eines Neuronalen Netzes anhand von Testdaten überprüft. Die Trainings- und die Testdatenmenge sind dabei als disjunkt vorauszusetzen.

Um sogenanntes *Overtraining* zu vermeiden, kann bisweilen auch der Einsatz einer dritten, zu den zuvor genannten Mengen paarweise disjunkten, Datenmenge zur Validierung herangezogen werden. Während des Netz-Trainings wird in gewissen Abständen mit der Validierungsmenge die Leistung des Netzes (d.h. die Güte der Approximation oder

Klassifizierung) festgestellt; sobald diese (in geeigneter Weise gemessene) Leistung abfällt, kann das Training beendet werden.

Dieses Phänomen des Overtrainings ist von der numerischen Funktionsapproximation bekannt: werden die Stützstellen (sprich: Trainingsdaten) zu gut approximiert (gelernt), so wird oftmals die Näherungsleistung außerhalb der Stützstellen deutlich schlechter.

Insgesamt kann festgehalten werden, dass ein Training eines Neuronalen Netzes i.a. nur zu einem brauchbaren Ergebnis führen wird, wenn eine ausreichende und - in Hinblick auf das jeweilige Problem - repräsentative Datenmenge vorhanden ist.

1.2.4 Qualität eines Künstlichen Neuronalen Netzes

Die Leistungsmessung eines Neuronalen Netzes kann in verschiedener Hinsicht erfolgen (vgl. hierzu [Zell1997], S. 413ff). Es kann beispielsweise ermittelt werden, wie gut ein bestimmter Lernalgorithmus arbeitet oder welche Approximationsqualität das erreichte Netz auf einer Testmenge besitzt. Auch hierbei können zahlreiche Faktoren berücksichtigt werden, z.B. die Trainingsdauer bis zum Erreichen des Abbruchkriteriums oder der Aufwand für einen Zyklus (gemessen in elementaren Rechenschritten).

Bei einem (Feedforward-)Netz, das eine Funktion $f: A \to \mathbb{R}^m$, $A \subseteq \mathbb{R}^n$, approximieren soll, wird häufig der nachstehend vorgestellte mittlere quadratische Fehler als Gütemaß herangezogen.

Das gesamte Netz lässt sich als Funktion $Y(\vec{x}, W)$ auffassen, die vom Eingabevektor $\vec{x}$ und den Gewichten W des Netzes abhängig ist. W lässt sich dabei, je nach Schreibweise, als Vektor aller Gewichte oder als Gewichts-Tensor auffassen. Für die hier getroffene Definition ist dieser Unterschied im Formalismus allerdings unerheblich.

Die Approximation des Netzes ist gut, wenn die Funktionen $\vec{x} \to Y(\vec{x}, W)$ und $\vec{x} \to f(\vec{x})$ in einem gewissen Sinn nahe beieinander liegen.

Definition 1.29 (Mittlerer Quadratischer Fehler, Mean Squared Error)

a) Der *quadratische Fehler* des Netzes zum Eingabevektor $\vec{x}$ und den Gewichten W ist definiert als $F(x, W) := |f(\vec{x}) - Y(\vec{x}, W)|^2$.

b) Der *Mittlere Quadratische Fehler* (*Mean Squared Error,* abgekürzt *MSE*) des Netzes (in Abhängigkeit von den Gewichten) ist definiert durch $F(W) := \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} F(\vec{x_k}, W)$. Hierbei ist $(\vec{x_k})_{k\in\mathbb{N}}$ eine (theoretische, unendliche) Folge von Eingabevektoren.

Bemerkung 1.30

Die hier auftretende Grenzwertbildung ist in der Praxis wiederum durch eine Approximation zu ersetzen, denn es stehen im konkreten Anwendungsfall natürlich nicht unendlich viele Eingabevektoren (Daten) zur Verfügung.

Dafür bietet der Mittlere Quadratische Fehler den Vorteil, dass er "fast sicher existiert", d.h. dass der Grenzwert "fast sicher konvergiert", und die Fehlerfunktion ebenfalls "fast sicher"

stetig und differenzierbar (in W) ist. Vgl. hierzu und zu den nachfolgenden Aussagen Hecht-Nielsen [Hech1990], S. 112ff.

Auch wenn es eine Reihe anderer Fehlerfunktionen gibt, so ist der Mittlere Quadratische Fehler dennoch der meistverbreitete. Er betont starke Abweichungen stärker als schwache, - ein Verhalten, das dem natürlichen Empfinden entspricht -, und mehrfach vorgelegte Eingabewerte fließen entsprechend mehrfach in die Berechnung des Fehlers mit ein.

Allerdings lässt sich diese Fehlerfunktion nicht als Universalwerkzeug verwenden. Auf dem Gebiet der Muster- oder Bilderkennung scheitert beispielsweise die simple Form der Abstandsmessung zwischen Pixelwerten o.ä. Und generell hängt vieles von einer geeigneten Codierung der Problemstellung ab.

Betrachtet man die MSE-Funktion $W \rightarrow F(W)$, so wird durch sie eine Oberfläche über dem Raum der möglichen Gewichte definiert. Zur Veranschaulichung nehmen wir an, das betrachtete Netz habe L Gewichte, W sei also als L-dimensionaler Vektor interpretierbar. Für jedes spezielle W gibt dann F(W) die Höhe dieser Oberfläche, der sogenannten Fehleroberfläche, an.

Im nachstehenden Bild wird - wenngleich extrem vereinfacht für den Fall von L=2 - zur Veranschaulichung eine solche mögliche Fehleroberfläche gezeigt; der vertikale Strich markiert das Minimum.

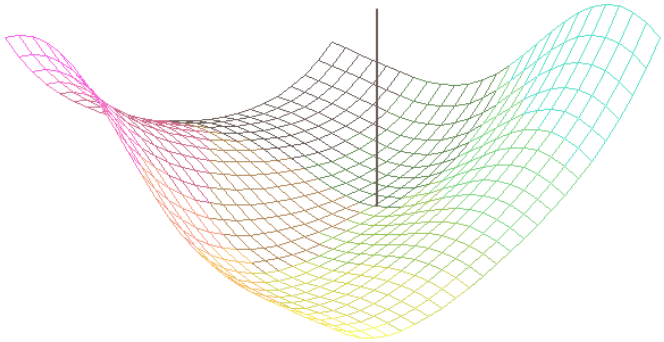


Abb. 1.15: Skizzenhafte Darstellung einer Fehleroberfläche mit Minimum

Ziel des Lernprozesses ist es (bei den hier betrachteten Netzen und Problemstellungen) somit, einen Gewichtsvektor W_0 zu finden, so dass F an dieser Stelle (möglichst) ein absolutes Minimum auf der Fehleroberfläche annimmt.

Bemerkung 1.31

Eine Reihe von allgemein nutzbaren Problemstellungen für ein praktisches Benchmarking neuronaler Lernverfahren sowie entsprechende Software wird in [Zell1997] vorgestellt (vgl. dort S. 420ff).

1.2.5 Wissensspeicher und Regeldarstellung

Das Neuronale Netz kann als Wissensspeicher aufgefasst werden in dem Sinn, dass logische und funktionale Zusammenhänge darin enthalten sind bzw. durch das Netz repräsentiert werden. Statt Wissensspeicher könnte man auch den moderneren Ausdruck der *Knowledge*

Base verwenden; allerdings sollte unterschieden werden zwischen dem (reinen) Wissen um Zusammenhänge und dem Verwalten von ggf. riesigen Datenmengen zum Zweck der Analyse und der möglichen Extraktion von solchen Zusammenhängen. Im weiten Feld der Informatik wird der Begriff der Knowledge Base auch für die Zusammenfassung von beidem verwendet.

Dabei erschließt sich die Wissensrepräsentation eines Künstlichen Neuronalen Netzes dem Betrachter nicht unmittelbar von selbst. Denn im Gegensatz zur Vorgehensweise logischer Systeme, die mit Schlussfolgerungen und Regeln arbeiten, besteht der konnektionistische Ansatz gerade darin, ein komplexes System von Neuronen als Gesamtheit einen bestimmten Sachverhalt lernen zu lassen, ohne dabei die Rolle eines einzelnen, isolierten Neurons zu betonen oder explizit zu kennen.

Ein solches Neuronales Netz, das, wie wir gesehen haben, auch auf spezielle Situationen trainiert worden sein kann, repräsentiert Wissen, das je nach Aufbau des Netzes mehr oder weniger gut in Form von Regeln ausgedrückt werden kann, ohne dass wir diese Regeln aber i.a. direkt aus der Topologie und den Gewichten des Netzes ablesen könnten. Da es jedoch eine Funktionsvorschrift zwischen dem Eingabe- und dem Ausgaberaum verkörpert, treffen für das Netz in diesem Sinn alle Aussagen (und somit auch Regeln) zu, die für die zugrundeliegende Funktion gelten.

Umgekehrt ist aber auch zu klären, in welcher Form beispielsweise ein a-priori-Wissen über Zusammenhänge zwischen den Trainingsdaten genutzt und in das Netz gebracht werden könnte. Abgesehen von einigen Vorüberlegungen zu einer geschickten Initialisierung der Gewichte ist ein Einbringen von Vorwissen in das Netz generell nicht möglich.

Es stellt sich somit die Frage, wie - noch dazu möglichst universell - die Arbeitsweise eines Neuronalen Netzes mit dem Wunsch, mit logischen IF-THEN-Regeln operieren zu können, in Einklang gebracht werden kann. Da es im Kontext der hier betrachteten Systeme üblich ist, die englischsprachigen Begriffe “IF” und “THEN” zu verwenden (und nicht auf deutsch “WENN” und “DANN”), wollen auch wir hier dieser Gewohnheit folgen.

Ansätze, das durch ein Neuronales Netz immanent gespeicherte Wissen durch Regeln darzustellen, gibt es mittlerweile zahlreiche; einige davon werden in Kapitel 5 vorgestellt. Allerdings gehen die bisher präsentierten Verfahren fast durchweg von Neuronalen Netzen mit ganz speziellen Strukturen aus. Teilweise werden auch nur Netze betrachtet, die eigens für den Zweck der Regelrepräsentation modelliert worden sind.

1.2.6 Approximationsfähigkeit Neuronaler Netze

Zunächst erinnern wir an Kolmogorovs Darstellungssatz 1.23 (sh. S. 31), der besagt, dass zu einer stetigen Funktion $f: [0,1]^n \to \mathbb{R}^m$ ein dreischichtiges Neuronales Netz mit $2n+1$ Neuronen in der verborgenen Schicht existiert, welches die Funktion (exakt) realisiert.

In dem so postulierten Netz treten jedoch in der Praxis unbekannte Funktionen auf. Wie bereits erwähnt, ist Satz 1.23 eher von theoretisch-grundlegender Bedeutung. Um pragmatisch vorgehen zu können, sind Aussagen wünschenswert, die mit bekannten Funktionstypen operieren, z.B. hier mit sigmoiden Funktionen. Wie bereits erwähnt, werden wir weiterhin die logistische Funktion als konkreten Repräsentanten der sigmoiden Funktionstypen

betrachten. Dafür geht es dann nicht mehr um die exakte Darstellung einer Funktion durch ein Neuronales Netz, sondern um die Approximation (auf einer kompakten Grundmenge). Hierzu zunächst eine Definition.

Definition 1.32 (Universeller Approximator)

Es sei $K \subseteq \mathbb{R}^n$ eine kompakte Menge und Φ eine Menge von Abbildungen $\varphi : K \to \mathbb{R}^m$. Dann heißt Φ ein *universeller Approximator (bezüglich der Norm* $\|\cdot\|$), wenn es zu jeder stetigen Funktion $h : K \to \mathbb{R}^m$ und jedem $\varepsilon > 0$ ein $\varphi \in \Phi$ gibt, so dass $\|h - \varphi\| < \varepsilon$.

Bemerkung 1.33

Die in der obigen Definition verwendete Norm $\|\cdot\|$ ist im konkreten Kontext festzulegen. Oftmals handelt es sich dabei um die L_p- (oder speziell für p=2 die L_2-) Norm über einer Grundmenge M:

$$\|f\|_p := (\int_M |f(x)|^p dx)^{1/p}, p \geq 1.$$

Bemerkung 1.34

Die hier erwähnte Menge Φ kann beispielsweise eine Familie bestimmter Neuronaler Netze sein, denn jedes solche Netz beschreibt eine bestimmte Funktion $\varphi : K \to \mathbb{R}^m$.

Bemerkung 1.35

Prinzipiell sind in Definition 1.32 auch Normen wie die Maximumsnorm

$$\|f\|_{\max} := \max_K \{ |f(x)| : x \in K\}$$

denkbar. (K ist eine kompakte Menge, f eine stetige Funktion auf K, so dass $|f|$ sein Maximum auch annimmt.)

Folglich ist dann Φ ein universeller Approximator bezüglich der Maximumsnorm, wenn es zu jeder stetigen Funktion $h : K \to \mathbb{R}^m$ und jedem $\varepsilon > 0$ ein $\varphi \in \Phi$ gibt, so dass gilt:

$$\|h(x) - \varphi(x)\| < \varepsilon \ \forall x \in K.$$

Wir werden jedoch sehen, dass wegen gewisser Symmetrieeigenschaften der von uns betrachteten Neuronalen Netze die Menge dieser Netze kein universeller Approximator bezüglich der Maximumsnorm ist. (Vgl. hierzu das Resultat 6.29 auf S. 128.)

Eine Reihe von Arbeiten befassen sich mit Universalitätsaussagen über die Approximationsfähigkeit Neuronaler Netze, vgl. die Zusammenstellung bei Feuring in [Feur1995], S. 32ff. Stellvertretend sei das Resultat von Hornik wiedergegeben. Dessen originale Formulierung bezieht sich auf L_p-integrable Funktionen; für unsere Zwecke genügt jedoch die hier dargestellte Version des Satzes.

Satz 1.36 (Dreischichtige Netze sind universelle Approximatoren)

Es seien $K \subseteq \mathbb{R}^n$ eine kompakte Menge und Φ die Menge aller dreischichtigen Neuronalen Feedforward-Netze mit Eingabemenge K und nicht-konstanten, stetigen Ausgabefunktionen. Dann lässt sich jede quadratintegrable Funktion $h : K \to \mathbb{R}^m$ zu einer vorgegebenen Genauigkeit durch ein geeignetes Netz aus Φ approximieren (bezüglich der L_2-Norm).

Ein Beweis hierzu findet sich in [Horn1991].

❑

Mit diesem Resultat können wir uns nachfolgend in unseren Betrachtungen auf dreischichtige Feedforward-Netze konzentrieren.

1.2.7 Einsatzgebiete Neuronaler Netze

Abschließen wollen wir diese Einführung in die Künstlichen Neuronalen Netze mit einem Überblick über einige typische Anwendungsgebiete. Damit soll der Leser gleichzeitig einen gewissen Beispielvorrat erlangen, auf den in späteren Kapiteln bei Bedarf zurückgegriffen werden kann.
Wir verweisen hierbei hauptsächlich auf [Zell1997] und [Patt1997]; dort werden die hier genannten Anwendungsbeispiele etwas ausführlicher erläutert. Weitere Beispiele aus dem Gebiet betriebswirtschaftlicher Anwendungen finden sich z.B. in [Biet1998].

Beispiel 1.37 (Allgemeine Abbildungen)

Wie bereits erwähnt, dienen Neuronale Netze dazu, allgemeine Abbildungsvorschriften (Funktionen) anhand von Trainingsdaten (näherungsweise) zu erlernen. Allerdings müssen gewisse Voraussetzungen wie z.B. Stetigkeit - je nach verwendetem Netztyp - von der zu lernenden bzw. zu approximierenden Funktion erfüllt werden.

Liegen genügend viele und brauchbare Daten vor, so kann ein geeignetes Neuronales Netz auf den funktionalen Zusammenhang zwischen einer m-dimensionalen Eingabe und der zugehörigen n-dimensionalen Ausgabe trainiert werden. Mit “brauchbar” ist dabei gemeint, dass ein Netz nur dort eine gewisse Generalisierungsfähigkeit aufweisen kann, wo es zuvor auch trainiert hat. Liegen in einem einfachen Fall beispielsweise alle Trainingsdaten im positiven Bereich, so wird man keine Approximationsfähigkeit des Netzes im negativen Bereich erwarten können.

Beispiel 1.38 (Mustererkennung)

Neuronale Netze sind gut geeignet, Muster zu erkennen, z.B. optische Muster in Bildern, Grafiken, aber auch bei Sprach- oder Handschriftenerkennung. Klassische Methoden auf diesen Gebieten weisen sehr häufig den Nachteil extrem hoher Verarbeitungszeiten auf.

Beispiel 1.39 (Prognosen und Risikoabschätzung)

Der Wunsch, Vorhersagen treffen zu können, ist weit verbreitet. Ob es im Bankenbereich um die Einschätzung der Kreditwürdigkeit eines Kunden oder die künftige Wertentwicklung von

Aktion geht, oder ob ein Strom- oder Telekommunikationsanbieter zukünftige Absatzzahlen prognostizieren möchte: Neuronale Netze erweisen sich auch auf diesem Gebiet als geeignetes Mittel, sofern ein ausreichendes Maß an Erfahrungs- bzw. Trainingsdaten vorliegt, mit denen das Netz lernen kann. Dabei werden in der Praxis mitunter zahlreiche Neuronale Netze parallel eingesetzt und fortlaufend anhand der neuen Erfahrungswerte weitertrainiert.

Beispiel 1.40 (Optimierung)

Viele Problemstellungen haben zahlreiche Nebenbedingungen (z.B. das klassische Traveling Salesman Problem (TSP)[8], sh. etwa [Hopf1985]), so dass das Trial-and-Error-Prinzip aus Komplexitätsgründen in der Praxis nicht geeignet ist, das Problem zu lösen. Unter den jeweiligen Nebenbedingungen soll dann möglichst auch noch die optimale, die beste Lösung (in Bezug auf eine vorgegebene Kostenfunktion) gefunden werden.

Beispiel 1.41 (Automatische Steuerung)

Ein weiteres Anwendungsgebiet ist die Steuerung autonomer Fahrzeuge, also das Steuern durch algorithmische oder neuronale Mechanismen. Eines der prominentesten Systeme ist *ALVINN*, *Autonomous Land Vehicle in a Neural Network*, an der Carnegie-Mellon-Universität (sh. [Pome1991], zit. nach [Zell1997], S. 541ff).

[8] Ein Handlungsreisender muss eine vorgegebene Anzahl von Städten bereisen unter Minimierung z.B. der Wegstrecke (oder auch der benötigten Reisezeit).

2 Fuzzy-Theorie und Fuzzy-Systeme

In diesem Kapitel sollen die Grundlagen aus den Gebieten der Fuzzy-Theorie und der Fuzzy-Systeme vorgestellt werden, auf die im späteren Verlauf teilweise zurückgegriffen wird. Insbesondere der Begriff des Fuzzy-Controllers oder -Reglers spielt im Weiteren eine Rolle. Für weitergehende Einführungen sei exemplarisch auf [Böhm1993] und [Borg2003] verwiesen.

2.1 Einführung und kurzer geschichtlicher Abriss

Die Fuzzy-Logik, auf der das gesamte Gebiet der Fuzzy-Theorie und -Systeme beruht, ist eine Erweiterung der klassischen zweiwertigen Logik. Statt "fuzzy" könnten wir auch den deutschen Begriff "unscharf" verwenden; der englischsprachige Begriff hat sich jedoch durchgesetzt.

Während Aristoteles das Gesetz vom ausgeschlossenen Dritten postulierte, worauf die Beweisführung durch Widerspruch, die "reductio ad absurdum", beruht, vermutete Platon, dass es etwas Drittes zwischen "wahr" und "falsch" geben könnte. Dieser Gedanke ist in gewissem Sinne bereits eine erste Vorahnung dessen, was wir heute in Form anderer (nicht-binärer) Logiken kennen, u.a. der Fuzzy-Logik, die in Abschnitt 2.4 detaillierter vorgestellt wird. Angemerkt sei ebenfalls, dass die Beweisführung durch Widerspruch, die "*reductio ad absurdum*", auf diesem Postulat beruht!

Bereits in den Zwanziger Jahren des vorigen Jahrhunderts stellt der Mathematiker und Philosoph Bertrand Russell fest, dass das Gesetz des ausgeschlossenen Dritten nur für präzise Formalismen, nicht jedoch für beliebige vage (z.B. umgangssprachliche) Beschreibungen gilt. (Sh. hierzu [Russ1923], zitiert bei [Feur1995].)

Zur selben Zeit bewiesen Logiker (Post, Tarski, Lukasiewicz), dass es widerspruchsfreie nicht-zweiwertige Logikkalküle gibt, die ohne das Prinzip des ausgeschlossenen Dritten auskommen. Zu diesen sog. Lukasiewicz-Logiken sei auf [Böhm1993], S. 209ff, verwiesen. Zunächst behandelte man dreiwertige Logiken, es wurden jedoch ebenfalls Logiken mit mehr als drei und schließlich mit unendlich vielen Wahrheitswerten formuliert. Vgl. Hierzu [Barr2001], S.35 ff. Bemerkenswert ist dabei insbesondere die Feststellung: "*Obwohl (...) zulässige Systeme drei verschiedene Zustände für Aussagen zulassen, werden alle Aussagen* über *sie mit Hilfe der traditionellen zweiwertigen Logik gemacht. Darüber hinaus ist es (...) unmöglich, die Widerspruchsfreiheit eines Systems allein im System zu beweisen. Es scheint keine allumfassende übergreifende Superlogik zu geben, in deren Rahmen sich jede mögliche Logik als Sonder- oder Grenzfall sehen läßt.*"

Damit ist der Weg geebnet für die Fuzzy-Logik, die Wahrheitswerte im reellen Kontinuum des Intervalls [0,1] verwendet. Im Jahre 1965 veröffentlicht Lotfi Asker Zadeh den Beitrag "Fuzzy Sets" (sh. [Zade1965]), in dem er die Mathematik der "unscharfen Mengen" beschreibt.

Als grundlegende Arbeiten zu Fuzzy-Mengen, Fuzzy-Zahlen und deren Arithmetik sowie zur Fuzzy-Logik sind u.a. diejenigen von Dubois und Prade [Dubo1980] sowie Kosko

[Kosk1987] zu nennen. Sicherlich die praxisrelevanteste Anwendung der Fuzzy-Theorie liegt auf dem Gebiet der Fuzzy-Regelung (Fuzzy-Controller), die ihren wesentlichen Ausgangspunkt in den Arbeiten von Mamdani ([Mamd1974], [Mamd1975]) besitzt.

2.2 Grundlagen der Fuzzy-Mengenlehre

In der klassischen Mengenlehre wird eine Menge M (auf einer vorausgesetzten Grundmenge X) dadurch definiert, dass für alle Elemente x von X festgelegt wird, ob x zu M gehört (Notation: $x \in M$) oder nicht ($x \notin M$). Dabei können Mengen in aufzählender Schreibweise notiert werden ($M := \{a, b, c, d\}$) oder prädikativ, d.h. in beschreibender Form (zum Beispiel $M := \{x \mid x \in \mathbb{N},\ x \text{ ist eine gerade Zahl }\}$.

Formal lässt sich eine solche Menge M durch eine zweiwertige (Zugehörigkeits-, Mitgliedschafts- oder auch charakteristische) Funktion $\mu : X \to \{0, 1\}$ beschreiben, so dass $\mu(x) = 1 \Leftrightarrow x \in M$ gilt (und somit natürlich $\mu(x) = 0$ für $x \notin M$).

Die Fuzzy-Mengenlehre verallgemeinert diesen Ansatz, indem diese Funktion Werte im gesamten reellen Intervall [0,1] annehmen kann.

2.2.1 Fuzzy-Mengen

Definition 2.1 (Fuzzy-Menge, Zugehörigkeitsfunktion, Zugehörigkeitsgrad)

Sei X eine (nichtleere) Grundmenge. Eine *Fuzzy-Menge* a über X ist eine Menge geordneter Paare $\{(x, \mu_a(x)) \mid x \in X\}$, wobei $\mu_a : X \to [0, 1]$ eine Funktion ist, die den Grad der Zugehörigkeit angibt, mit dem ein Element $x \in X$ zur Menge a gehört. μ_a wird Zugehörigkeitsfunktion genannt.

Man spricht davon, dass ein Element x zum Grad $\mu_a(x)$ zur Menge a gehört bzw. den Zugehörigkeitsgrad $\mu_a(x)$ besitzt. Allgemein sagt man, dass ein Element x zur Menge a gehört, wenn $\mu_a(x) > 0$.

Die Menge aller Fuzzy-Mengen über einer Menge X wird mit *FM(X)* bezeichnet.

Bemerkung 2.2

1. Man bezeichnet im Fuzzy-Kontext reelle Werte gelegentlich auch als “crispe” oder “scharfe” Werte.
2. Prinzipiell kann die Grundmenge X selbst eine Fuzzy-Menge über einer anderen Menge X' sein; damit gelangt man zu Fuzzy-Mengen höherer Ordnung. Zunächst einmal genügt es aber für unsere Zwecke, sich X als die Menge der reellen Zahlen oder eine Teilmenge davon, z.B. ein Intervall, vorzustellen.

Bemerkung 2.3 (Notation von Zadeh)

Zadeh führte für Fuzzy-Mengen die folgenden, rein formal zu verstehenden Notationen ein (vgl. [Zade1965] sowie [Böhm1993]). Ist X eine endliche Grundmenge $\{x_1, x_2, ..., x_n\}$, so kann a auch geschrieben werden als

$$a = \frac{\mu_a(x_1)}{x_1} + \frac{\mu_a(x_2)}{x_2} + ... + \frac{\mu_a(x_n)}{x_n} = \sum_{1 \le i \le n} \frac{\mu_a(x_i)}{x_i},$$

wobei Summanden für x_i mit $\mu_a(x_i) = 0$ auch weggelassen werden können. Für eine unendliche Grundmenge X wird formal-analog das Integralzeichen verwendet:

$$a = \int_X \frac{\mu_a(x)}{x}.$$

Beispiel 2.4

Nachstehendes Bild illustriert die Zugehörigkeitsfunktionen der crispen Zahl 3 (im hier beschriebenen Sinne) und einer Fuzzy-Menge a über $\mathbb{R}$, die mit einer gewissen Unschärfe die Zahl 2 repräsentiert.

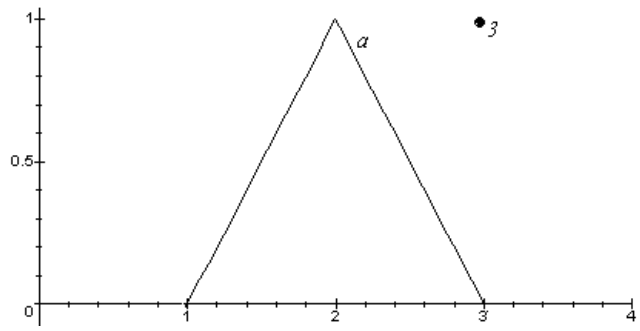


Abb. 2.1: Grafische Darstellung einer Fuzzy-Menge a und der crispen Menge (Zahl) 3

Bemerkung 2.5

Es soll an dieser Stelle angemerkt werden, dass es zwar eine formale Ähnlichkeit zwischen wahrscheinlichkeitstheoretischen Interpretationen und den hier dargestellten Fuzzy-Konzepten gibt. Die semantische Interpretation ist jedoch sehr unterschiedlich.

Betrachten wir hierzu die beiden Aussagen: "Die Bremsen dieses Wagens funktionieren zum Grad 0,95" sowie "Die Bremsen dieses Wagens funktionieren mit der Wahrscheinlichkeit[9] 0,95".

Während man bei der ersten Aussage jedes normale Bremsmanöver erfolgreich absolvieren wird, weil 95% der Bremsleistung ausreichen werden, kann man bei der zweiten Aussage Glück oder Pech haben, denn entweder hat man einen Wagen mit funktionierenden Bremsen, oder man hat mit einer (wie auch immer präzisierten) fünfprozentigen Wahrscheinlichkeit ein nicht bremsbares Fahrzeug!

[9] Rein formal muss hier betont werden, dass eine wahrscheinlichkeitstheoretische Aussage über ein einzelnes Individuum keinen Sinn macht.

Wahrscheinlichkeitstheoretische Interpretationen beziehen sich (in ihrer Anwendbarkeit) stets auf eine größere Zahl, während der Fuzzy-Ansatz für eine einzelne Ausprägung unscharfe Werte zulässt.

Bemerkung 2.6

Fuzzy-Mengen sind offensichtlich vollständig durch ihre Zugehörigkeitsfunktion charakterisiert. So entspricht die Grundmenge X in eindeutiger Weise derjenigen Fuzzy-Menge, für die $\mu(x) = 1 \ \forall x \in X$ ist, die leere Menge $\oslash$ entspricht der Fuzzy-Menge, die durch $\mu(x) = 0 \ \forall x \in X$ beschrieben wird.

Die Teilmengenbeziehung kann im Fuzzy-Kontext einfach wie folgt festgelegt werden.

Definition 2.7 (Fuzzy-Teilmenge)

Es seien a, b Fuzzy-Mengen über X. Dann heißt a eine (Fuzzy-)Teilmenge von b, $a \subset b$, wenn $\mu_a(x) \leq \mu_b(x) \ \forall x \in X$ gilt.

Bemerkung 2.8

Wir wollen hier (und später), um die Notationen nicht zu überfrachten, für die Fuzzy-Sachverhalte keine neuen Symbole einführen und gehen davon aus, dass im jeweiligen Zusammenhang die Bedeutung eines Symbols eindeutig erkennbar ist.

2.2.2 Operationen auf Fuzzy-Mengen

Nachfolgend sollen einige grundlegende Operationen auf Fuzzy-Mengen vorgestellt werden.

Definition 2.9 (Komplement einer Fuzzy-Menge)

Es sei a eine Fuzzy-Menge über X. Dann ist das Komplement von a definiert durch $\overline{a} := \{ (x, \mu_{\overline{a}}(x)) \mid x \in X \}$ mit $\mu_{\overline{a}}(x) := 1 - \mu_a(x)$.

Beispiel und Bemerkung 2.10

Sei $X := [0,4]$, a wie in Abb. 2.1 (S. 44); dann besitzt $\bar{a}$ die im nachstehenden Bild gezeigte Zugehörigkeitsfunktion. Insbesondere ist das Komplement einer Fuzzy-Menge wieder eine Fuzzy-Menge.

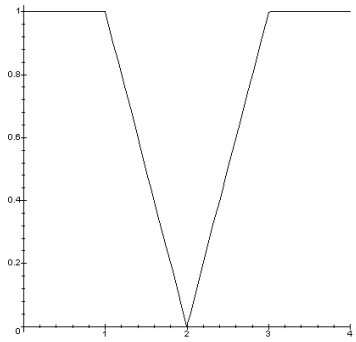


Abb. 2.2: Fuzzy-Komplement von a (vgl. Abb. 2.1)

Definition 2.11 (Höhe, normal)

a) Ist a eine Fuzzy-Menge über X, dann heißt $height(a) := \sup\{\mu_a(x) \mid x \in X\}$ die *Höhe* der Fuzzy-Menge a.

b) Eine Fuzzy-Menge $a \in FM(X)$ heißt *normal*, wenn $height(a) = 1$ ist.

Auf der Grundlage des Maximum- und des Minimum-Operators führte Zadeh in [Zade1965] die Bildung von Vereinigung und Durchschnitt von Fuzzy-Mengen ein.

Definition 2.12 (Vereinigung, Durchschnitt)

Es seien a, b Fuzzy-Mengen über X. Dann wird definiert:

a) Vereinigung: $a \cup b := \{(x, \mu_{a\cup b}(x)) \mid x \in X,\ \mu_{a\cup b}(x) := \max\{\mu_a(x), \mu_b(x)\}\}$

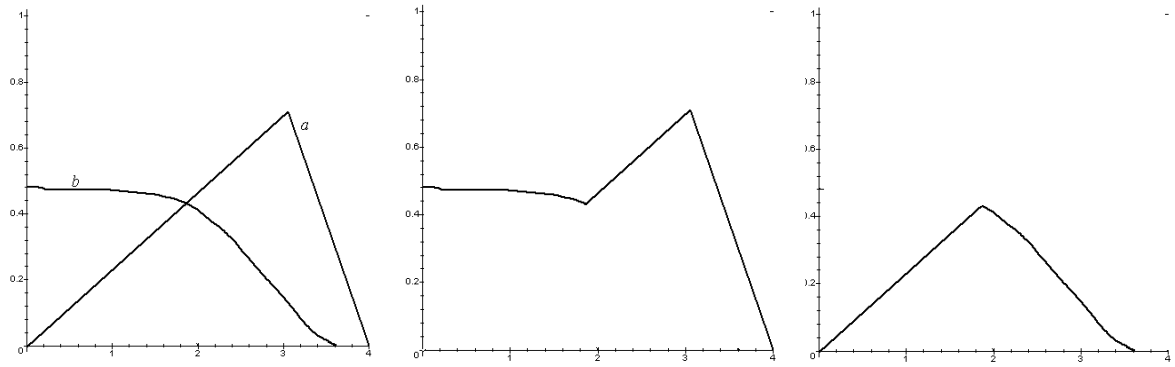
b) Durchschnitt: $a \cap b := \{(x, \mu_{a\cap b}(x)) \mid x \in X,\ \mu_{a\cap b}(x) := \min\{\mu_a(x), \mu_b(x)\}\}$

Insbesondere sind die Vereinigung und der Durchschnitt zweier Fuzzy-Mengen über X wieder Fuzzy-Mengen über X.

Beispiel 2.13

Im nachstehenden Bild sind zwei Fuzzy-Mengen a und b über $X := [0,4]$ dargestellt (Abbildungsteil a); daneben sind die Vereinigung (b) und der Durchschnitt (c) wiedergegeben, deren Zugehörigkeitsfunktion gemäß obiger Definition über das punktweise

Maximum bzw. Minimum der Zugehörigkeitsfunktionen der beiden Ausgangsmengen festgelegt ist.



a) Zwei Fuzzy-Mengen a und b b) Vereinigung $a \cup b$ c) Durchschnitt $a \cap b$

Abb. 2.3: Darstellung von Vereinigung und Durchschnitt zweier Fuzzy-Mengen

Bemerkung 2.14

Von der speziellen Wahl der beiden Operatoren max und min kann generell abstrahiert werden; üblicherweise werden zwei Klassen von Operatoren betrachtet: die sogenannten t-Normen und die t-Conormen (vgl. [Dubo1985]), die wie folgt definiert werden.

Definition 2.15 (t-Norm, t-Conorm, s-Norm)

a) Eine *t-Norm* ist eine Funktion $t : [0,1] \times [0,1] \to [0,1]$ mit folgenden Eigenschaften.
   i) Randbedingungen: $t(1,1) = 1,\ t(a,1) = t(1,a) = a\ \forall a \in [0,1]$
   ii) Monotonie: $t(a,b) \le t(c,d)\ \ \forall a,b,c,d \in [0,1]$ *mit* $a \le c$ *und* $b \le d$
   iii) Kommutativität: $t(a,b) = t(b,a)\ \ \forall a,b \in [0,1]$
   iv) Assoziativität: $t(a,t(b,c)) = t(t(a,b),c)\ \ \forall a,b,c \in [0,1]$

b) Eine *t-Conorm* (oder auch *s-Norm*) ist eine Funktion $s : [0,1] \times [0,1] \to [0,1]$ mit folgenden Eigenschaften.
   i) Randbedingungen: $s(0,0) = 0,\ s(a,0) = s(0,a) = a\ \forall a \in [0,1]$
   ii) Monotonie: $s(a,b) \le s(c,d)\ \ \forall a,b,c,d \in [0,1]$ *mit* $a \le c$ *und* $b \le d$
   iii) Kommutativität: $s(a,b) = s(b,a)\ \ \forall a,b \in [0,1]$
   iv) Assoziativität: $s(a,s(b,c)) = s(s(a,b),c)\ \ \forall a,b,c \in [0,1]$

Bemerkung 2.16

Wie man leicht sieht, ist der min-Operator eine t-Norm, der max-Operator eine t-Conorm. Diese Konzepte verallgemeinern somit diese beiden Operatoren.

Zu jeder t-Norm t korrespondiert offensichtlich genau eine t-Conorm s durch folgende Beziehung:

$$s(a,b) = 1 - t(1-a, 1-b),\ a,b \in [0,1]$$

Für weitere Eigenschaften zu t-Normen und t-Conormen sei auf [Klir1988] verwiesen.

Auch Relationen auf Fuzzy-Mengen sind möglich. Für das Weitere benötigen wir die Definition des kartesischen Produktes von Fuzzy-Mengen, womit die Fuzzy-Erweiterung des kartesischen Produktes der Grundmengen gemeint ist. Da Fuzzy-Mengen über einer Menge X Teilmengen von $X \times [0,1]$ sind, können natürlich prinzipiell auch die klassischen kartesischen Produkte gebildet werden. In unserem Zusammenhang möchten wir aber auf dem kartesischen Produkt der Ausgangsmengen wiederum eine reellwertige Zugehörigkeitsfunktion erhalten.

Definition 2.17 (Kartesisches Produkt von Fuzzy-Mengen)

Es seien $X_1, X_2, ..., X_n$ Grundmengen, $a_1 \in FM(X_1), a_2 \in FM(X_2), ..., a_n \in FM(X_n)$. Dann wird das (fuzzy-)kartesische Produkt $a_1 \times ... \times a_n$ definiert über die Zugehörigkeitsfunktion:

$$\mu_{a_1 \times ... \times a_n}(x_1 \times ... \times x_n) := \min\{\mu_{a_i}(x_i) \mid 1 \le i \le n\}.$$

Damit ist das kartesische Produkt $a_1 \times ... \times a_n$ eine Fuzzy-Menge über $X_1 \times ... \times X_n$.

Um das Sprachliche nicht unnötig aufzublähen, sprechen wir nachfolgend einfach vom kartesischen Produkt von Fuzzy-Mengen, da ein Missverständnis ausgeschlossen scheint.

2.3 Fuzzy-Zahlen und Fuzzy-Arithmetik

Zur “unscharfen” (oder sagen wir “abweichungstoleranten”) Beschreibung von Sachverhalten (wie z.B. “warmer Kaffee” anstelle von “die Kaffeetemperatur beträgt 51 Grad”) erweist es sich als hilfreich, spezielle Fuzzy-Mengen zu verwenden, die in gewissem Sinn als Verallgemeinerung gewöhnlicher (reeller) Zahlen dienen können; damit gelangen wir zu dem Konzept der Fuzzy-Zahl.

Im Folgenden wollen wir als Grundmenge X die Menge der reellen Zahlen $\mathbb{R}$ betrachten. In naheliegender Weise lassen sich die Begrifflichkeiten aber auch auf Teilintervalle von $\mathbb{R}$ übertragen.

2.3.1 Fuzzy-Zahlen

Wir wollen solche Fuzzy-Mengen a über $\mathbb{R}$ als Fuzzy-Zahl bezeichnen, bei denen es anschaulich unmittelbar einleuchtet, welche reelle Zahl r sie repräsentieren. Das heißt, dass wir an der Stelle r die Zugehörigkeit 1 erwarten, nach links bzw. nach rechts sollen die Zugehörigkeitswerte kleiner werden.

Präzisieren wir diesen Wunsch durch folgende Definitionen.

Definition 2.18 (Konvexe Fuzzy-Menge)

Eine Fuzzy-Menge a über $\mathbb{R}$ heißt konvex, wenn gilt:

$$\forall x_1, x_2 \in \mathbb{R}\ \forall \lambda \in [0,1]: \min(\mu_a(x_1), \mu_a(x_2)) \le \mu_a(\lambda x_1 + (1-\lambda)x_2).$$

Beispiel 2.19

Nachstehendes Bild zeigt eine konvexe Fuzzy-Menge a sowie deren Komplementmenge $\bar{a}$, die nicht konvex ist.

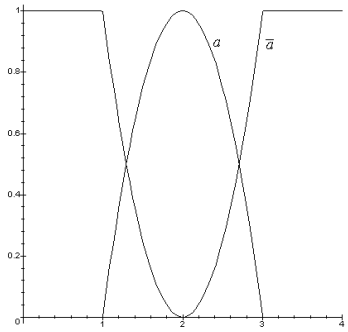


Abb. 2.4:Konvexe Fuzzy-Menge a und deren nicht-konvexes Komplement

Definition 2.20 (Fuzzy-Zahl, Modalwert)

Eine konvexe Fuzzy-Menge a über $\mathbb{R}$ mit stückweise stetiger Zugehörigkeitsfunktion μ_a heißt *Fuzzy-Zahl*, wenn es genau ein $x_0 \in \mathbb{R}$ mit dem Zugehörigkeitsgrad $\mu_a(x_0) = 1$ gibt. Der Wertx_0 heißt *Modalwert* von a.

Bemerkung 2.21

Eine Fuzzy-Zahl ist insbesondere eine normale Fuzzy-Menge.

Bemerkung 2.22

Die in Abb. 2.4 gezeigte Fuzzy-Menge a ist gleichzeitig eine Fuzzy-Zahl. Offensichtlich repräsentiert sie in gewisser Weise die reelle Zahl 2, die auch der Modalwert von a ist.

Das ebenfalls eingezeichnete Komplement von a ist jedoch keine konvexe Fuzzy-Menge, also auch keine Fuzzy-Zahl.

Notation 2.23

Im Folgenden wollen wir die Menge der Fuzzy-Zahlen mit F bezeichnen.

Beispiel 2.24

Abb. 2.5 zeigt eine konvexe Fuzzy-Menge, die jedoch nicht normal ist. An der Stelle 2 hat die Zugehörigkeitsfunktion zwar ihr Maximum, dieses ist aber nicht 1.

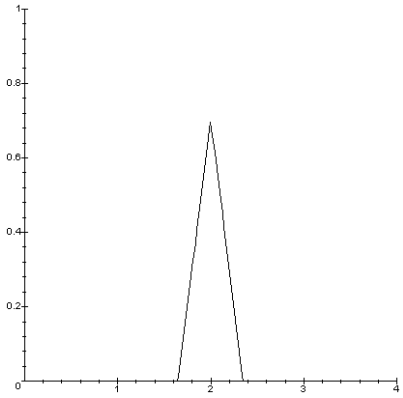


Abb. 2.5: Keine Fuzzy-Zahl: eine nicht-normale Fuzzy-Menge (Höhe ist nicht 1)

Definition 2.25 (Träger)

Es sei $a \in F$. Als *Träger* (oder engl. *support*) von a, supp(a), wird der topologische Abschluss der Menge $\{x \in \mathbb{R} \mid \mu_a(x) > 0\}$ definiert.

Bemerkung 2.26

In F findet sich jede reelle Zahl r (bei geeigneter Interpretation) wieder. Dies führt zur folgenden Definition.

Definition 2.27 (Singleton)

Die Fuzzy-Zahl mit der 1-Punkt-Zugehörigkeitsfunktion

$$\mu_a(x) := \begin{cases} 1 & \text{für } x = r \\ 0 & \text{sonst} \end{cases}$$

repräsentiert die crispe Zahl r und wird *Singleton* genannt.

Manche Eigenschaften reeller Zahlen wird man auf Fuzzy-Zahlen übertragen können, wenn auch nicht immer vollständig. Ein Beispiel sind die Einordnungen in positive und negative Zahlen.

Definition 2.28 (Positive und negative Fuzzy-Zahlen)

Eine Fuzzy-Zahl a heißt positiv, wenn $\mu_a(x) = 0 \; \forall x \leq 0$ ist. Entsprechend heißt eine Fuzzy-Zahl a negativ, wenn $\mu_a(x) = 0 \; \forall x \geq 0$ ist.

Beispiel 2.29

Nachstehend werden in Abb. 2.6 drei Fuzzy-Zahlen gezeigt. a ist positiv, b ist negativ, die Fuzzy-Zahl c ist jedoch weder positiv noch negativ, da sie im positiven wie im negativen reellen Bereich positive Zugehörigkeitsgrade besitzt.

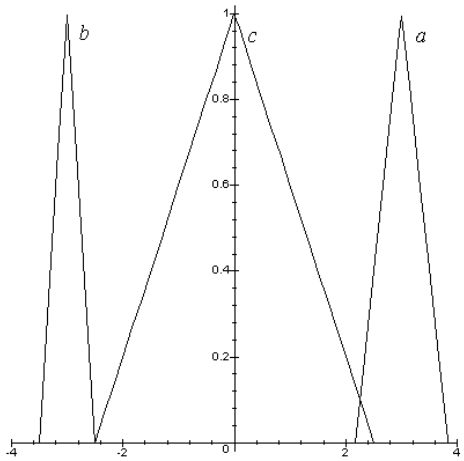


Abb. 2.6: Drei Fuzzy-Zahlen: a positiv, b negativ, c weder positiv noch negativ

Definition 2.30 (Fuzzy-Null)

(Jedes) $a \in F$ mit Modalwert 0 heißt (eine) Fuzzy-Null.

Bemerkung 2.31

Offensichtlich gibt es unendlich viele Fuzzy-Nullen.

Bemerkung 2.32

Definition 2.20 legt den Begriff der Fuzzy-Zahl in nachvollziehbarer Weise fest; für die praktische Arbeit - insbesondere die Umsetzung in Software-Lösungen - ist es aber erforderlich, weitere Bedingungen zu formulieren, die die Vielfalt der Fuzzy-Zahlen einschränken.

Dubois und Prade [Dubo1978] führten die sogenannte LR-Darstellung ein, mit der sich eine Fuzzy-Zahl durch eine linksseitige und eine rechtsseitige Referenzfunktion beschreiben lässt (vgl. Hierzu auch [Böhm1993], S.13ff).
Ein für die DV-Implementierung besonders nützlicher Spezialfall sind die nachstehend definierten triangulären Fuzzy-Zahlen.

Definition 2.33 (Trianguläre Fuzzy-Zahl, rechte und linke Unschärfe)

$a \in F$ heißt trianguläre Fuzzy-Zahl, wenn sich die Zugehörigkeitsfunktion μ_a schreiben lässt in der Form

$$\mu_a(x) := \begin{cases} \frac{x}{a_l} - \frac{a_m - a_l}{a_l} & \text{für } a_m - a_l \le x \le a_m \\ -\frac{x}{a_r} + \frac{a_m + a_r}{a_r} & \text{für } a_m < x \le a_m + a_r \\ 0 & \text{sonst} \end{cases}$$

für geeignete Werte $a_l > 0, a_m \in \mathbb{R}, a_r > 0$.

a_l heißt die linke, a_r die rechte Unschärfe von a. (a_m ist der schon definierte Modalwert von a.) Die Fuzzy-Zahl a wird in diesem Fall auch mit (a_m,a_l,a_r) notiert.

Die Menge der triangulären Fuzzy-Zahlen wird mit $\hat{F}$ bezeichnet.

Beispiel 2.34

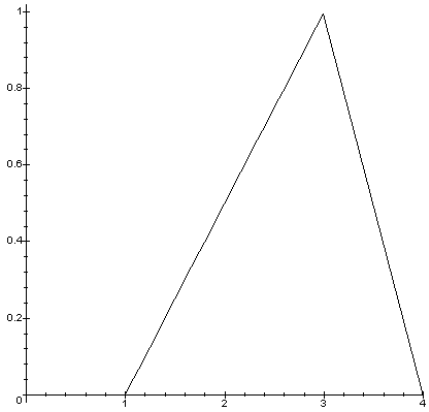


Abb. 2.7: Trianguläre Fuzzy-Zahl a, $(a_m,a_l,a_r)=(3,2,1)$

Im obigen Bild wird die trianguläre Fuzzy-Zahl $a \in \widehat{F}$ dargestellt, die bei 3 ihren Modalwert besitzt und eine linke Unschärfe von 2 sowie eine rechte Unschärfe von 1 aufweist.

Bemerkung 2.35

Während die Definition 2.20 der (allgemeinen) Fuzzy-Zahl auch "unendlich weit" verschwommene Fuzzy-Mengen zugelassen hat, bei denen also der Träger die gesamte Menge $\mathbb{R}$ oder ein einseitig unbeschränktes Intervall ist, werden sinnvollerweise mit Definition 2.33 insbesondere nur noch Fuzzy-Zahlen mit kompakten Trägern betrachtet.

Bemerkung und Notation 2.36

Eine trianguläre Fuzzy-Zahl $a \in \widehat{F}$ ist durch die Angabe der linken und der rechten Grenze ihres Trägers (a_λ bzw. a_ρ) sowie des Modalwertes a_m vollständig und eindeutig festgelegt. Dies ermöglicht eine DV-Implementierung solcher Fuzzy-Zahlen mit vergleichsweise geringem Aufwand, da je (triangulärer Fuzzy-)Zahl nur drei reelle Werte verwaltet werden müssen.

Hierfür schreiben wir $[a_\lambda, a_m, a_\rho]$, und es gilt der Zusammenhang: $a_\lambda = a_m - a_l$, $a_\rho = a_m + a_r$.

Beispiel 2.37

Die Fuzzy-Zahl *(3,2,1)* aus Abb. 2.7 lässt sich somit auch schreiben als [1,3,4].

Für viele praktische Anwendungen ist es wichtig, aus einer Fuzzy-Menge wieder eine crispe Menge zu gewinnen. Dies geschieht u.a. über die nachfolgend definierten Niveaumengen, die Bereiche angeben, über denen die betreffende Zugehörigkeitsfunktion einen vorgegebenen Wert überschreitet.

Definition 2.38 (α-Niveaumenge)

Es sei a eine Fuzzy-Menge über X, weiter sei $\alpha \in [0,1]$. Dann heißt die Menge

$$a_\alpha := \{ x \in X \mid \mu_a(x) \geq \alpha \}$$

α-Niveaumenge (oder auch *α-cut* oder *α-Schnitt*) von a. (Für $\alpha = 0$ ist diese Niveaumenge gerade die gesamte Grundmenge X.)

Die entsprechende Menge mit der strikten Ungleichungsbedingung $a_\alpha^* := \{ x \in X \mid \mu_a(x) > \alpha \}$ heißt *strenge α-Niveaumenge von a*.

Bemerkung 2.39

Auch wenn wir eingangs erwähnt haben, dass wir in diesem Abschnitt die Grundmenge $X = \mathbb{R}$ oder ggf. eine Intervall-Teilmenge davon betrachten wollen, so verwenden wir hier weiterhin auch das Symbol X um anzudeuten, dass einige Definitionen und Resultate auch unabhängig vom Spezialfall der reellen Zahlen formuliert werden können.

Ist X eine crispe Menge (z.B. $X = \mathbb{R}$), so sind auch die α-Niveaumengen crisp.

Beispiel 2.40

Nachstehendes Bild zeigt eine solche α-Niveaumenge: zur triangulären Fuzzy-Zahl a mit Modalwert 2, rechter Grenze 4 und linker Grenze 0 ist die Niveaumenge für $\alpha = 0.4$: $a_{0.4} = \{ x \in X \mid \mu_a(x) \geq 0.4 \} = [0.8, 3.2]$.

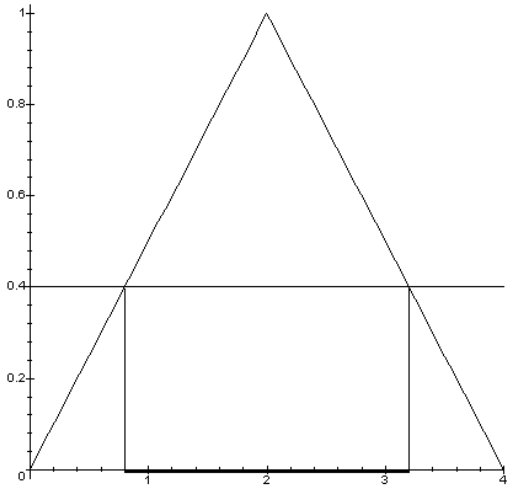


Abb. 2.8: Darstellung der 0.4-Niveaumenge von a = [0,2,4]: das Intervall [0.8, 3.2]

Bemerkung 2.41

In Definition 2.25 wurde der Träger einer Fuzzy-Zahl definiert. Ausgedrückt durch die Notation der (strikten) α-Niveaumengen ist der Träger gerade der topologische Abschluss der strikten 0-Niveaumenge $\{ x \in X \mid \mu_a(x) > 0 \}$. Dies ist jedoch generell nicht die 0-Niveaumenge $a_0 = \{ x \in X \mid \mu_a(x) \geq 0 \} = X$!

Im obigen Beispiel ist der Träger von a das Intervall *supp*(a) = [0,4], während die (nicht-strikte) 0-Niveaumenge trivialerweise ganz $\mathbb{R}$ ist.

2.3.2 Das Extensionsprinzip von Zadeh

Damit trianguläre Fuzzy-Zahlen eingesetzt werden können, ist für diese eine Arithmetik erforderlich. Im Folgenden werden das *Extensionsprinzip* von Zadeh kurz vorgestellt und die wesentlichen Schritte skizziert, wie daraus Addition und Multiplikation hergeleitet bzw. motiviert werden. Bei manchen Autoren wird auch der Begriff *Erweiterungsprinzip* verwendet.

L. Zadeh stellt in [Zade1965] das sog. Extensionsprinzip vor, mit dem klassische Konzepte in die Fuzzy-Welt systematisch übertragen werden können. Dieses wollen wir zunächst an einer einfachen Situation erläutern.

Betrachten wir dazu zwei Grundräume X und Y; $f: X \to Y$ sei eine Funktion, a eine Fuzzy-Menge über X, also $a \in FM(X)$.

Um die Funktion f zu "fuzzifizieren", also auf die Fuzzy-Mengen über X zu übertragen, wird zu a ein "passendes" $b \in FM(Y)$ gesucht. Naheliegend ist die Überlegung, dass für jedes Paar $(x,y) \in X \times Y$ mit $y = f(x)$ die Zugehörigkeitsfunktion $\mu_b(y)$ durch $\mu_a(x)$ bestimmt wird, sich der Zugehörigkeitsgrad von y bzgl. b also durch den von x bzgl. a berechnen lässt.

Da die Funktion f jedoch nicht injektiv sein muss, kann es zu $y \in Y$ mehrere Urbilder in X geben. In einem solchen Falle wird definitionsgemäß der größte auftretende Zugehörigkeitsgrad bzw. allgemein das Supremum der Zugehörigkeitsgrade genommen.
Unter den möglicherweise unendlich vielen Urbildern muss es selbstverständlich keines geben, das den größten Zugehörigkeitsgrad besitzt.

Das Extensionsprinzip lautet für eindimensionale Funktionen $f: X \to Y$ somit:

Definition 2.42 (Extensionsprinzip)

Es sei $f: X \to Y$ eine Funktion. Dann wird durch f einem $a \in FM(X)$ genau ein $b \in FM(Y)$ zugeordnet durch die folgende Definition der Zugehörigkeitsfunktion.

$$\mu_b(y) := \begin{cases} \sup\{\mu_a(x) \mid x \in X,\ y = f(x)\} & falls\ f^{-1}(y) \neq \emptyset \\ 0 & sonst \end{cases}$$

Die Menge b heißt dann auch Fuzzy-Fortsetzung von a unter f.
(Mit f^{-1}(y) wird hier die Urbildmenge zu y bezeichnet, nicht die i.a. nicht existierende Umkehrfunktion an der Stelle y.)

Bemerkung 2.43

Für eine bijektive Funktion f gilt offensichtlich die einfachere Beziehung $\mu_b(y) = \mu_a(x)$ für Paare (x,y) mit $y = f(x)$.

Beispiel 2.44

Es seien $X = Y = \mathbb{R}$. Zur Funktion $f(x) := 2x$ ergibt sich aufgrund der Bijektivität von f die Fuzzy-Fortsetzung einfach durch die Beziehung $\mu_b(y) = \mu_a(\frac{y}{2})$.

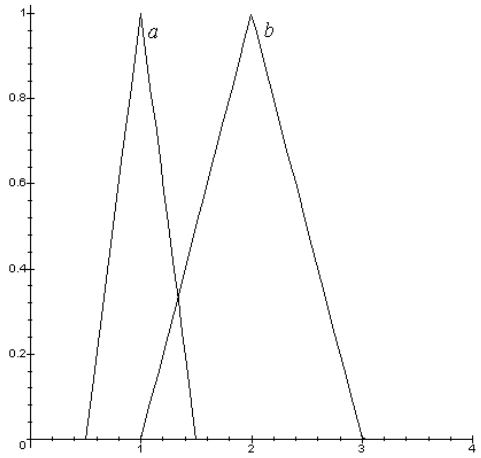


Abb. 2.9: Die Fuzzy-Menge *a* und ihre Erweiterung *b* unter *f(x) = 2x*

Zur Fuzzy-Menge $a = [0.5, 1, 1.5]$ ergibt sich somit die Menge $b = [1, 2, 3]$, wie im obigen Bild gezeigt.

Beispiel 2.45

In ähnlicher Weise erhält man bei Betrachtung der Funktion $f(x) := -x$ die Fuzzy-Version der Negation, des Vorzeichenwechsels. Zur Fuzzy-Menge $a = [1, 2, 3]$ ergibt sich hier die Menge $b = [-3, -2, -1]$, also die Spiegelung an der y-Achse.

In der allgemeineren Fassung formuliert das Extensionsprinzip die Erweiterung einer Funktion, die auf einem mehrdimensionalen Grundraum erklärt ist. Hierzu wird das in Definition 2.17 (S. 48) eingeführte kartesische Produkt von Fuzzy-Mengen verwendet.

Definition 2.46 (Mehrdimensionales Extensionsprinzip)

Es seien $X_1, X_2, ..., X_n$ und Y Grundmengen, $a_1 \in FM(X_1), a_2 \in FM(X_2), ..., a_n \in FM(X_n)$, $f: X_1 \times ... \times X_n \to Y$.

Dann lässt sich die Fuzzy-Menge $a := a_1 \times ... \times a_n$ mittels f abbilden auf eine Fuzzy-Menge $b \in FM(Y)$ durch die Festlegung der Zugehörigkeitsfunktion:

$$\mu_b(y) := \begin{cases} \sup\{\min\{\mu_{a_1}(x_1), ..., \mu_{a_n}(x_n)\} \mid x_1 \in X_1, .., x_n \in X_n,\ y = f(x_1, .., x_n)\} & \text{für } f^{-1}(y) \neq \emptyset \\ 0 & \text{sonst} \end{cases}$$

Bemerkung 2.47

Um in der mehrdimensionalen Version ebenfalls einen elementaren, reellen Zugehörigkeitswert zu erhalten, wird also jeweils der kleinste der n auftretenden Zugehörigkeitswerte genommen.

2.3.3 Fuzzy-Arithmetik

Als Anwendung des mehrdimensionalen Extensionsprinzips kann die Fuzzy-Erweiterung der klassischen Rechenoperationen in Angriff genommen werden.

Da wir uns im Folgenden aus den genannten Gründen auf trianguläre Fuzzy-Zahlen beschränken wollen, seien die relevanten Resultate zur Fuzzy-Arithmetik nur kurz dargestellt (teilweise zitiert nach Tenhagen [Tenh2000], S. 51ff).
Insbesondere sei darauf hingewiesen, dass nicht alle von den reellen Zahlen bekannten Eigenschaften im Rahmen der Fuzzifizierung erhalten bleiben.

Bemerkung 2.48

a) Im nachfolgenden Satz 2.49 wird aus dem Extensionsprinzip die Addition zweier triangulärer Fuzzy-Zahlen hergeleitet.

b) Die Fuzzy-Subtraktion resultiert unter Zuhilfenahme der Negation aus der Addition.

c) Die Fuzzy-Multiplikation ergibt sich im Prinzip durch die Anwendung des mehrdimensionalen Extensionsprinzips auf die reelle Multiplikation; um innerhalb der Menge der triangulären Fuzzy-Zahlen zu bleiben, wird in Definition 2.52 diese Multiplikation in einer angepassten Form festgelegt.

d) Als Fuzzy-Inversion wird die Extension der Funktion f mit $f(x) := \frac{1}{x}$ verstanden. Allerdings ist das Fuzzy-Produkt einer Fuzzy-Zahl mit der so berechneten "Fuzzy-Inversen" i.a. nicht die reelle Zahl 1.

e) Somit ergibt sich die Fuzzy-Division durch Multiplikation mit der "Fuzzy-Inversen".

f) Die reelle Zahl 0 (bzw. präzise formuliert: das ihr entsprechende Singleton) ist das neutrale Element der Fuzzy-Addition. Es lässt sich jedoch zeigen, dass es für Fuzzy-Zahlen, die keine reellen Zahlen repräsentieren (also keine Singletons sind), keine additiv inverse Fuzzy-Zahl gibt. Algebraisch formuliert ist $(\widehat{F}, +)$ keine Gruppe.
Das Zeichen + steht an dieser Stelle selbstverständlich für die hier diskutierte Fuzzy-Addition. Wiederum aus Gründen der besseren Lesbarkeit wollen wir keine spezielle Notation für die Fuzzy-Rechenoperationen einführen.

g) Die reelle Zahl 1 (bzw. das entsprechende Singleton) ist das neutrale Element bezüglich der Fuzzy-Multiplikation. Es ist aber auch $(\widehat{F} \setminus \{0\}, *)$ keine Gruppe, da es zu einer Fuzzy- Zahl a (ausgenommen die Singletons) keine multiplikativ Inverse gibt.

Formulieren wir nachstehend die benötigten Rechenoperationen für trianguläre Fuzzy-Zahlen konkret.

Satz 2.49 (Fuzzy-Addition)

Es seien $a = [a_\lambda, a_m, a_\rho]$, $b = [b_\lambda, b_m, b_\rho] \in \widehat{F}$. Dann ergibt sich die Summe der beiden Fuzzy-Zahlen mittels des Extensionsprinzips zu $c = [a_\lambda + b_\lambda, a_m + b_m, a_\rho + b_\rho] \in \widehat{F}$.

Insbesondere ist die Summe zweier triangulärer Fuzzy-Zahlen wieder eine solche.

Beweis:

Es sei $c := a + b$. Gemäß dem mehrdimensionalen Extensionsprinzip (vgl. Def. 2.46) ergibt sich die Zugehörigkeitsfunktion für c zu $\mu_c(z) = \sup\{\min\{\mu_a(x), \mu_b(y)\} \mid x + y = z\}$ für alle z.

Offensichtlich gilt (nur) für $c_m := a_m + b_m$, dass $\mu_c(c_m) = 1$ ist, denn μ_a und μ_b nehmen den (maximalen) Wert 1 genau an den Stellen a_m bzw. b_m an.

Gemäß Definition 2.33 gilt für die Zugehörigkeitsfunktion einer triangulären Fuzzy-Zahl a mit $a = (a_m, a_l, a_r) = [a_\lambda, a_m, a_\rho] \in \widehat{F}$

$$\mu_a(x) = \begin{cases} \frac{x}{a_l} - \frac{a_m - a_l}{a_l} & \text{für } a_m - a_l \le x \le a_m \\ -\frac{x}{a_r} + \frac{a_m + a_r}{a_r} & \text{für } a_m < x \le a_m + a_r \\ 0 & \text{sonst} \end{cases} .$$

Es sei kurz daran erinnert: die Schreibweise mit den runden Klammern gibt neben dem Modalwert die linke und die rechte Unschärfe der triangulären Fuzzy-Zahl an; die Notation mit den eckigen Klammern soll an die Intervalldarstellung erinnern, hier werden die linke Grenze, der Modalwert und die rechte Grenze angegeben.

Ersetzen wir die linken und rechten Unschärfen (a_l bzw. a_r) durch die linke und rechte Grenze (vgl. die Bemerkung auf S. 52) mittels der Beziehungen $a_\lambda = a_m - a_l$, $a_\rho = a_m + a_r$, so erhalten wir für a und b die folgenden Zugehörigkeitsfunktionen.

$$\mu_a(x) = \begin{cases} \frac{x - a_\lambda}{a_m - a_\lambda} & \text{für } a_\lambda \le x \le a_m \\ \frac{a_\rho - x}{a_\rho - a_m} & \text{für } a_m < x \le a_\rho \\ 0 & \text{sonst} \end{cases}$$

$$\mu_b(x) = \begin{cases} \frac{x - b_\lambda}{b_m - b_\lambda} & \text{für } b_\lambda \le x \le b_m \\ \frac{b_\rho - x}{b_\rho - b_m} & \text{für } b_m < x \le b_\rho \\ 0 & \text{sonst} \end{cases}$$

Damit ist auch die Erkenntnis trivial, dass c links von $a_\lambda + b_\lambda$ und rechts von $a_\rho + b_\rho$ den Zugehörigkeitsgrad 0 besitzt, $\mu_c(z) = \sup\{\min\{\mu_a(x), \mu_b(y)\} \mid x + y = z\} = 0$ für $z \le a_\lambda + b_\lambda$ bzw. $z \ge a_\rho + b_\rho$, da stets mindestens einer der Zugehörigkeitsgrade von a oder b bei der Minimumsbildung gleich 0 ist.

Betrachten wir z-Werte mit $a_\lambda + b_\lambda < z \le a_m + b_m = c_m$. (Die Situation für z-Werte rechts von c_m, also $a_m + b_m = c_m < z < a_\rho + b_\rho$, ist analog und wird hier nicht weiter ausgeführt.)

Damit sind auch nur x- und y-Werte mit $x \le a_m$, $y \le b_m$ zu betrachten, denn andernfalls wäre o.B.d.A. $y > b_m$ und somit zwangsläufig $x < a_m$, so dass für ein geeignetes $\delta > 0$ gelten würde: $\mu_a(x + \delta) > \mu_a(x)$, $\mu_b(y - \delta) > \mu_b(y)$, $x + \delta + y - \delta = x + y = z$, d.h. für das betrachtete Paar (x,y) läge nicht das Maximum über alle $\min\{\mu_a(x), \mu_b(y)\}$ mit $x + y = z$ vor.

Dann wird für z der Term $\min\{\mu_a(x), \mu_b(y)\}$ für x, y mit $x + y = z$ genau dann maximal, wenn die beiden Zugehörigkeitsfunktionen gleich sind, $\mu_a(x) = \mu_b(y)$. (Andernfalls könnten wieder durch eine entsprechende δ-Korrektur beide Zugehörigkeitsfunktionen vergrößert werden. Wäre etwa $\mu_a(x) < \mu_b(y)$, so gäbe es ein geeignetes $\delta > 0$ mit der Eigenschaft $\mu_a(x) < \mu_a(x + \delta) < \mu_b(y - \delta) < \mu_b(y)$, $x + \delta + y - \delta = x + y = z$, d.h. wiederum hätte bei (x,y) nicht das entsprechende Maximum über $\min\{\mu_a(x), \mu_b(y)\}$ mit $x + y = z$ vorgelegen.)

Es sei also z gegeben mit $a_\lambda + b_\lambda < z \le a_m + b_m = c_m$; gesucht werden x und y mit den beiden Bedingungen $x + y = z$ und $\mu_a(x) = \mu_b(y)$, das heißt eingesetzt in die Geradengleichungen:

$$\frac{x-a_\lambda}{a_m-a_\lambda} = \frac{y-b_\lambda}{b_m-b_\lambda} = \frac{(z-x)-b_\lambda}{b_m-b_\lambda}.$$

Formt man die Gleichheit zwischen dem ersten und dem letzten Ausdruck um, so erhält man

$$x = \frac{(z-b_\lambda)(a_m-a_\lambda)+a_\lambda(b_m-b_\lambda)}{(a_m-a_\lambda)+(b_m-b_\lambda)}.$$

Setzen wir diesen Wert in die Gleichung vorherige Gleichung $\frac{x-a_\lambda}{a_m-a_\lambda}$ein, so ergibt sich

$$\frac{\frac{(z-b_\lambda)(a_m-a_\lambda)+a_\lambda(b_m-b_\lambda)}{(a_m-a_\lambda)+(b_m-b_\lambda)}-a_\lambda}{a_m-a_\lambda} = \frac{z(a_m-a_\lambda)-b_\lambda(a_m-a_\lambda)+a_\lambda(b_m-b_\lambda)-a_\lambda((a_m-a_\lambda)+(b_m-b_\lambda))}{(a_m-a_\lambda)((a_m-a_\lambda)+(b_m-b_\lambda))} = \frac{z}{c_m-c_\lambda} - \frac{c_\lambda}{c_m-c_\lambda},$$

wobei für den letzten Ausdruck $c_\lambda = a_\lambda + b_\lambda$ gesetzt wurde.

Entsprechend wird $c_\rho = a_\rho + b_\rho$ gesetzt, und es folgt insgesamt, dass sich die Zugehörigkeitsfunktion von $c = a + b$ für alle z schreiben lässt als

$$\mu_c(z) = \mu_{a+b}(z) = \begin{cases} \frac{z-c_\lambda}{b_m-c_\lambda} & \text{für } c_\lambda \le z \le c_m \\ \frac{c_\rho-z}{c_\rho-c_m} & \text{für } c_m < z \le c_\rho \\ 0 & \text{sonst} \end{cases},$$

womit gezeigt ist, dass auch die Summe zweier triangulärer Fuzzy-Zahlen wieder eine solche ist.

❑

Beispiel 2.50

Die nachstehende Abbildung illustriert die Fuzzy-Addition exemplarisch. Die beiden triangulären Fuzzy-Zahlen a = [1,3,5] und b = [5,6,7] werden addiert und ergeben c = [6,9,12].

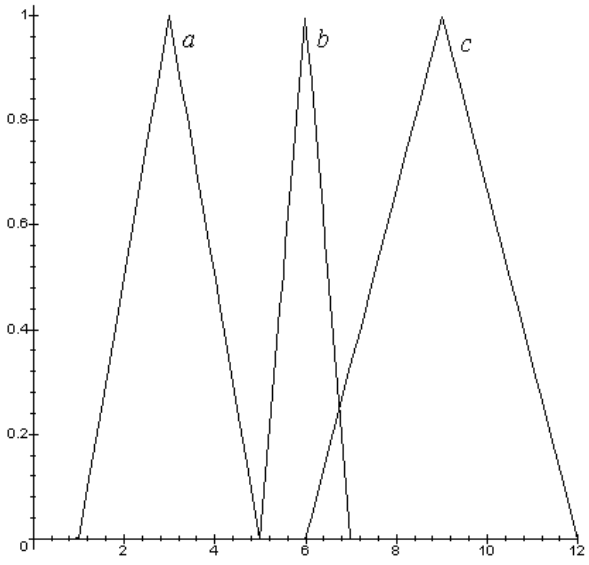


Abb. 2.10: Summe von zwei triangulären Fuzzy-Zahlen

Bemerkung 2.51

Die auf dem Extensionsprinzip beruhende Addition von triangulären Fuzzy-Zahlen verdeutlicht ein Problem: die Unschärfen werden bei fortgesetzter Addition niemals kleiner, i.a. jedoch größer. Das bedeutet, dass auch aus den entsprechenden Zahlen gewonnenen Aussagen in der Regel immer unschärfer werden.

Aus dem Extensionsprinzip folgt natürlich auch eine Multiplikation. Wie Tenhagen ausführt, ist das Produkt zweier triangulärer Fuzzy-Zahlen zwar wieder eine Fuzzy-Zahl, aber i.a. keine mit Dreiecksgestalt. Vgl. hierzu das Beispiel in [Tenh2000], S. 53.

Für die Praxis wird daher an dieser Stelle mit einer Approximation gearbeitet. Die Multiplikation wird wie folgt definiert.

Definition 2.52 (Fuzzy-Multiplikation)

Es seien $a = [a_\lambda, a_m, a_\rho]$, $b = [b_\lambda, b_m, b_\rho] \in \hat{F}$. Dann wird das Produkt der beiden Fuzzy-Zahlen definiert durch $c := [c_\lambda, c_m, c_\rho] \in \hat{F}$ mit $c_m := a_m + b_m$, $c_\lambda := \min\{a_\lambda b_\lambda, a_\lambda b_\rho, a_\rho b_\lambda, a_\rho b_\rho\}$ und $c_\rho := \max\{a_\lambda b_\lambda, a_\lambda b_\rho, a_\rho b_\lambda, a_\rho b_\rho\}$.

Bemerkung 2.53

Die überraschend kompliziert wirkenden Terme für die linken und rechten Trägergrenzen mit der Minimums- bzw. Maximumsbildung rühren daher, dass bei unterschiedlichem Vorzeichen der Trägergrenzen diese vertauscht werden müssen.

Für den Fall von ausschließlich positiven Trägergrenzen von a und b reduzieren sich die beiden Formeln auf die einfachen Fälle $c_\lambda = a_\lambda b_\lambda$ und $c_\rho := a_\rho b_\rho$.

Bemerkung 2.54

1. Ohne Beweis sei angemerkt, dass für die Addition und die Multiplikation in $\hat{F}$ das Assoziativ- und das Kommutativgesetz gelten.
2. Sofern nur positive trianguläre Fuzzy-Zahlen beteiligt sind, gilt auch das Distributivgesetz: $(a+b)c = ac + bc$.

2.4 Fuzzy-Logik

Der Begriff “Fuzzy-Logik” ist seit den Achtziger Jahren populär geworden insbesondere durch die herausragenden Erfolge japanischer Elektronik (z.B. bei Kameras), bei der Fuzzy-Logik praktisch eingesetzt wurde. Eine recht kompakte Einführung in die Thematik der Fuzzy-Logik findet sich bei [Hell1997]; für einen etwas ausführlicheren Einstieg sei außerdem noch einmal auf [Böhm1993] verwiesen. Eine gute Darstellung mehrwertiger Logiken findet sich darüber hinaus bei [Resc1969].

2.4.1 Grundelemente der Logik

Allgemein, auch später im Kontext der Fuzzy-Logik, benötigen wir Prinzipien, wie (gültige) Schlussfolgerungen gezogen werden können. Umgangssprachlich sind dies Wenn-dann-Aussagenverknüpfungen, bei denen aus einer (eventuell zusammengesetzten) Prämisse eine (möglicherweise ebenfalls zusammengesetzte) Konklusion (Folgerung) resultiert.

Nachstehende Abbildung skizziert die Vorgehensweise. Aus einer oder mehreren Aussagen im Wenn-Teil des Satzes, den sog. Einzel-Prämissen, wird zunächst durch logische Verknüpfung (z.B. das logische *Und*) eine Prämisse ermittelt; im Zuge der Schlussfolgerung wird deren Wahrheitswert mit dem der Konklusion[10] in Verbindung gesetzt.

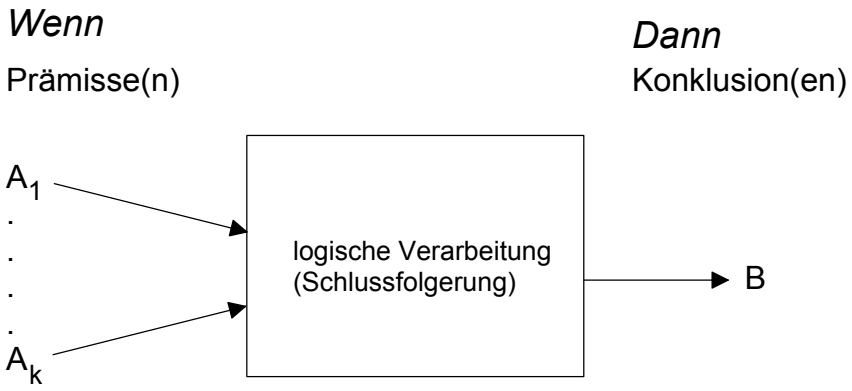


Abb. 2.11: Schematische Darstellung des logischen Schließens (angelehnt an [Böhm1993], S. 199)

Beispiele aus der klassischen (zweiwertigen) Logik sind Aussagensätze wie "wenn x gerade ist, dann ist x+1 ungerade" oder "wenn die Temperatur unter 0 Grad Celsius sinkt und es feucht ist, dann besteht die Gefahr von Glatteis auf den Straßen".

Definition 2.55 (Wahrheitswerte, klassische Wahrheitswertefunktion)

a) Im Folgenden werden die klassischen Wahrheitswerte "wahr" und "falsch" durch die numerischen Werte 1 bzw. 0 dargestellt.

b) Eine Funktion w, die logischen Ausdrücken (Wahrheits-)Werte aus { 0, 1 } zuordnet, wird Wahrheitswertefunktion genannt.

c) Es seien $x_1, ..., x_n$ logische Ausdrücke. Zu einem zusammengesetzten, n-stelligen logischen Ausdruck $A(x_1, ..., x_n)$ wird dann die Wahrheitswertefunktion w definiert durch $w(A(x_1, ..., x_n)) := A(w(x_1), ..., w(x_n))$.

Definition 2.56 (Logische Verknüpfungen)

Grundlegende zweiwertige Verknüpfungen von Aussagen in der klassischen Logik sind in der nachstehenden Übersicht, der sogenannten Wahrheitswertetafel, aufgeführt. Die genannten Verknüpfungen werden durch diese Tabelle formal definiert.

---

[10] Generell kann sich auch die Konklusion aus mehreren Teilen zusammensetzen; aus Gründen der Einfachheit wurde hier nur eine Konklusion B dargestellt.

Dazu kommt als wesentliche einstellige Operation die Negation ("nicht *a*", Notation: ¬*a*), die den Wahrheitswert von *a* gerade umkehrt (d.h.: ist *a* wahr (=1), so ist ¬*a* falsch (=0) u.u.).

a	b	Konjunktion (und) $a \wedge b$	Disjunktion (oder) $a \vee b$	Subjunktion (wenn - dann) $a \rightarrow b$	Bijunktion (genau dann - wenn) $a \leftrightarrow b$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1
1="wahr", 0="falsch"					

Tab. 2.1: Zweiwertige Verknüpfungen der klassischen Aussagenlogik

Beispiel 2.57

Teil c) der vorigen Definition 2.55 soll kurz an einem Beispiel erläutert werden.

Gesucht sei der Wahrheitswert des zusammengesetzten Ausdrucks $A(x_1,x_2) := x_1 \wedge \neg x_2$. Nun sind die verschiedenen Möglichkeiten der Wahrheitswerte von x_1, x_2 zu betrachten. Sei etwa $w(x_1) = 1,\ w(x_2) = 0$. Dann besagt die obige Definition:
$w(A(x_1,x_2)) = w(x_1 \wedge \neg x_2) = A(w(x_1), w(x_2)) = w(x_1) \wedge \neg w(x_2) = 1 \wedge \neg 0 = 1 \wedge 1 = 1.$

Die anderen drei Fälle ergeben sich analog; die Betrachtung kann natürlich wiederum in einer Wahrheitswertetafel übersichtlich festgehalten werden.

Für das Weitere sind sogenannte aussagenlogische Folgerungen (Implikationen) von großer Bedeutung. Wir definieren diese daher explizit.

Definition 2.58 (Aussagenlogische Folgerungen, Implikationen)

Es seien $k \geq 1$ und $A_1, ..., A_k, B$ n-stellige Aussagen. Dann heißt B eine aussagenlogische Folgerung (oder eine Implikation) aus $A_1, ..., A_k$, wenn für jede Belegung $w(x_1), ..., w(x_n)$ gilt:

Ist $w(A_1(x_1, ..., x_n)) = w(A_2(x_1, ..., x_n)) = ... = w(A_k(x_1, ..., x_n)) = 1$, dann $w(B(x_1, ..., x_n)) = 1$.

Bemerkung und Notation 2.59

Für eine solche aussagenlogische Folgerung schreiben wir kürzer auch

$$A_1(x_1, ..., x_n) \wedge A_2(x_1, ..., x_n) \wedge ... \wedge A_k(x_1, ..., x_n) \Rightarrow B(x_1, ..., x_n).$$

Aussagenlogisch formuliert bedeutet dies, dass die Subjunktion

$$A_1(x_1, ..., x_n) \wedge A_2(x_1, ..., x_n) \wedge ... \wedge A_k(x_1, ..., x_n) \rightarrow B(x_1, ..., x_n)$$

stets wahr, also eine Tautologie, ist.

Bemerkung 2.60

Wir beschränken uns künftig o.B.d.A. auf Folgerungen, bei denen die Prämisse nur aus konjunktiven Verknüpfungen besteht, die Konklusion keine Konjunktion und keine Disjunktion enthält. Wie in [Tenh2000] (S.57f) ausgeführt, lassen sich Implikationen, die nur

Konjunktionen oder Disjunktionen in Prämisse und Konklusion enthalten, durch ein logisch äquivalentes System von aussagenlogischen Folgerungen der Bauart $A_1 \wedge A_2 \wedge ... \wedge A_k \Rightarrow B$ ersetzen.

2.4.2 Schlussfiguren der klassischen Logik

Für logische Schlussfolgerungen werden immer wieder Grundableitungsregeln eingesetzt, die zum Teil eigene Bezeichnungen erhalten haben. Drei dieser sogenannten Schlussfiguren sollen hier kurz in Form von Definitionen vorgestellt werden. Dabei werden jedoch nur die Namen dieser Folgerungsregeln definiert; die Gültigkeit der logisch-äquivalenten Umformung ist dagegen eine Aussage, die nachzuweisen ist. (Vgl. hierzu [Böhm1993], S. 203ff.)

Es seien *a*, *b* und *c* logische Ausdrücke.

Definition 2.61 (Modus ponens, Abtrennungsregel)

Die nachstehende Ableitungsregel wird Modus ponens oder Abtrennungsregel genannt.

$a \rightarrow b$	*wenn a, dann b*
a	*es gilt a*
b	*also gilt auch b*

Aussagenlogisch kann die Regel geschrieben werden in der Form $((a \rightarrow b) \wedge a) \rightarrow b$.

Definition 2.62 (Modus tollens, Widerlegungsregel)

Als Modus tollens oder Widerlegungsregel wird die folgende Schlussfigur bezeichnet.

$a \rightarrow b$	*wenn a, dann b*
$\neg b$	*b gilt jedoch nicht*
$\neg a$	*also gilt auch a nicht*

Aussagenlogisch kann diese Regel geschrieben werden als $((a \rightarrow b) \wedge \neg b) \rightarrow \neg a$.

Definition 2.63 (Modus barbara, Kettenschlussregel)

Eine Form logischer Transitivität stellt die Kettenschlussregel, der Modus barbara, dar.

$a \rightarrow b$	*wenn a, dann b*
$b \rightarrow c$	*wenn b, dann c*
$a \rightarrow c$	*somit: wenn a, dann c*

Diese Regel kann geschrieben werden in der Form $((a \rightarrow b) \wedge (b \rightarrow c)) \rightarrow (a \rightarrow c)$.

Bemerkung 2.64

Die Kettenschlussregel kann unmittelbar auf mehr als zwei beteiligte Folgerungen erweitert werden.

2.4.3 Fuzzy-Aussagenlogik

Betrachten wir nun die für das Folgende wesentliche Verallgemeinerung der logischen Prinzipien gegenüber der klassischen Logik.

Ausgangspunkt ist bei allen nicht-zweiwertigen Logiken die Überlegung, dass es "zwischen" *wahr* und *falsch* noch andere Bewertungen, andere "Wahrheitswerte" geben kann. Sprachliche Beschreibungen wie "vielleicht", "sicherlich" usw. unterstreichen, dass in der Realität die klassische zweiwertige Logik nicht immer die optimale Form darstellen muss.

Im Rahmen der Fuzzy-Logik treten nun an die Stelle der "absoluten" Aussagen mit ihren binären Wahrheitswerten "wahr" und "falsch" die entsprechenden Wahrheitswertefunktionen oder Zugehörigkeitsgrade. Als Wahrheitswerte werden hier alle reellen Zahlen aus dem Intervall [0,1] zugelassen. Der Wert 1 repräsentiert dabei semantisch eine vollständig sichere Wahrheit, 0 steht für ein ebenso sicheres "falsch". Alle reellen Werte dazwischen, wie zuvor schon bei den Zugehörigkeitsfunktionen kennengelernt, stellen einen entsprechenden Erfüllungsgrad dar.

Definition 2.65 (Fuzzy-Wahrheitswertefunktion)

Es sei A die Menge der zulässigen Ausdrücke. (Der Begriff des "zulässigen Ausdrucks" soll an dieser Stelle nicht formal definiert werden; hierzu sei auf [Böhm1993], S. 180ff und S. 218ff, verwiesen.)
Dann ist $w : A \to [0,1]$ eine Wahrheitswertefunktion.
Ist A eine n-stellige Aussage, so ist $w(A(x_1,...,x_n)) := A(w(x_1),...,w(x_n))$.

Definition 2.66 (Fuzzy-logische Verknüpfungen)

Die Fuzzy-Negation $\neg a$ wird definiert mittels $w(\neg a) := 1 - w(a)$.

Die Erweiterungen der klassischen zweistelligen Operationen sind in nachstehender Tabelle zusammengefasst.

Fuzzy-Konjunktion (und) $a \wedge b$	Fuzzy-Disjunktion (oder) $a \vee b$	Fuzzy-Subjunktion (wenn - dann) $a \to b$	Fuzzy-Bijunktion (genau dann - wenn) $a \leftrightarrow b$
$w(a \wedge b) :=$ $\min(w(a), w(b))$	$w(a \vee b) :=$ $\max(w(a), w(b))$	$w(a \to b) :=$ $\min(1, 1 + w(b) - w(a))$	$w(a \leftrightarrow b) :=$ $1 - \lvert w(a) + w(b) \rvert$

Tab. 2.2: Wahrheitswerte zweiwertiger Verknüpfungen der Fuzzy-Aussagenlogik

Bemerkung 2.67

a) Die in Definition 2.66 angegebenen Rechenoperationen für die Verknüpfungen sind nicht die einzig möglichen. Generell lässt sich anstelle der Maximum- und Minimum-Bildung das Konzept der t-Normen und -Conormen (vgl. Def. 2.15 auf S. 47) hier zum Einsatz bringen.

b) Wir verzichten hier aus Gründen der besseren Lesbarkeit darauf, den Fuzzy-Operationen neue Symbole zuzuordnen, d.h. wir notieren diese genauso wie die klassischen

Verknüpfungen und gehen davon aus, dass im Kontext stets eindeutig zu verstehen ist, welche Operation gemeint ist.

c) Mit den hier getroffenen Definitionen lässt sich zeigen (vgl. etwa [Böhm1993].), dass zahlreiche Aussagen für die (Fuzzy-)Negation, Konjunktion und Disjunktion ebenfalls gelten, z.B. die Kommutativ-, Assoziativ-, Distributiv- und DeMorganschen Gesetze.

d) Es gilt jedoch (entsprechend dem Verhalten des Komplements bei Fuzzy-Mengen) i.a. nicht: $w(a \wedge \neg a) = 0$, $w(a \vee \neg a) = 1$.

2.4.4 Schlussfiguren der Fuzzy-Logik

Für die in Abschnitt 2.5 vorgestellten Fuzzy-Entscheidungssysteme benötigen wir für das Ziehen logischer Schlussfolgerungen im Fuzzy-Kontext den klassischen Figuren (wie dem Modus ponens, Def. 2.61) entsprechende Regeln.

Zunächst wollen wir kurz darstellen, dass der klassische Modus ponens im Kontext der Fuzzy-Logik nicht allgemeingültig ist. Das folgende Beispiel stammt aus [Böhm1993], S. 262.

Beispiel 2.68

Betrachten wir zwei Größen *X* und *Y*, denen die Merkmale "groß" und "klein" zugeordnet werden können. Auf einer (recht willkürlichen) numerischen Skala seien "groß" und "klein" modelliert durch die folgenden Zugehörigkeitsgrade.

Skala	1	2	3	4
groß	0	0,2	0,7	1
klein	1	0,5	0,1	0

Die Schlussfigur des Modus ponens bedeutet:

$$\frac{\begin{array}{l} X \text{ ist groß} \rightarrow Y \text{ ist klein} \\ X \text{ ist groß} \end{array}}{Y \text{ ist ?}}$$

Sofern der klassische Modus ponens gelten würde, wäre das "?" durch "klein" zu ersetzen.

Nun müssen wir die verschiedenen möglichen Ausprägungen von *X* und *Y* kombinieren und überprüfen, welchen Wahrscheinlichkeitswert *Y* jeweils besitzt. Sollten sich die oben gezeigten Werte für "klein", also 1, 0.5, 0.1 und 0, ergeben, so wäre im konkreten Beispiel der Modus ponens erfüllt. Sollte sich dagegen mindestens in einem Fall ein anderer Wert ergeben, dann wäre gezeigt, dass der klassische Modus ponens im Rahmen der Fuzzy-Logik mit den hier diskutierten Fuzzy-Operationen (Konjunktion mittels Minimum-, Disjunktion mittels Maximumbildung) nicht allgemeingültig ist.

Betrachten wir für "*X* ist groß" und "*Y* ist klein" zunächst exemplarisch die Ausprägungswerte 3 und 2. Dann ist w(*X*=3) = 0.7 und w(*Y*=2) = 0.5 (gemäß oben festgelegter Skala).

Die Aussage “wenn *X* groß ist, ist *Y* klein” kann auch formuliert werden als “nicht (*X* ist groß) oder (*Y* ist klein)”. Das heißt: w(nicht (*X* ist groß)) = 1 - w(*X* ist groß) = 1 - 0.7 = 0.3. Entsprechend ist w(*Y* ist klein) = 0.5. Die Oder-Verknüpfung (Disjunktion) wird über den Maximum-Operator durchgeführt, d.h. max(0.3,0.5)=0.5.

Die Aussage “*X* ist groß” hat den Wahrheitswert w(*X* ist groß) = 0.7; die Semantik des Wenn-Dann wird über das Maximum dieser Minima realisiert: für *X*=3, *Y*=2 gelangt man so zu min(0.5, 0.7) = 0.5; analog sind die Terme für *X*=1, *X*=2 und *X*=4 zu bilden. Man erhält dabei die Minima min(0,1)=0, min(0.2,0.8)=0.2, min(1,0.5)=0.5 . Das Maximum dieser vier Werte 0.5, 0, 0.2 und 0.5 ist 0.5.

Etwas übersichtlicher kann das Ganze durch das Tableau auf S. 65 dargestellt werden.

Zunächst wird rechts oben die Wahrheitswertetafel für “*X ist groß* → *Y ist klein*” errechnet. Für das Wertepaar (3,2) haben wir dies oben explizit durchgeführt und 0,5 erhalten.

Links unten wird die Wahrheitswertetafel für “*X ist groß*” notiert.

Schließlich wird rechts unten die Wahrheitswertetafel für die Implikation des Modus ponens wie folgt errechnet. Stellt man sich gedanklich in eine der Zellen rechts unten, so sind - ähnlich wie bei der Bildung eines Skalarproduktes - die einzelnen Werte der Zeile links und der Spalte darüber jeweils positionsweise miteinander mit *Und*, sprich dem Minimum, zu verknüpfen; aus den so erhaltenen einzelnen Minima wird anschließend das Maximum genommen und in die betreffende Zelle eingetragen.

X ist groß → *Y ist klein*

X=	*Y*= 1	2	3	4
1	1	1	1	1
2	0,8	0,8	0,8	0,8
3	0,7	0,5	0,3	0,3
4	1	0,5	0,1	0

X ist groß

0	0,2	0,7	1
1	2	3	4

1	0,5	0,3	0,3
1	2	3	4

Y ist ?

Auf die 0,3 in Zelle 4 rechts unten (bei “*Y ist ?*”) gelangt man also folgendermaßen:
$\max\{\min\{0,1\},\min\{0.2,0.8\},\min\{0.7,0.3\},\min\{1,0\}\} = \max\{0,0.2,0.3,0\} = 0.3.$

Insgesamt ist festzustellen, dass das Tableau rechts unten nicht der Aussage “*Y ist klein*” entspricht, denn diese besitzt die Belegung (1, 0.5, 0.1, 0)!

Das bedeutet: der klassische Modus ponens ist hier im Rahmen der Fuzzy-Logik mit den angewendeten Minimum- und Maximumbildungen nicht gültig!

Daher wird im Kontext der Fuzzy-Logik eine Abschwächung der Modus ponens-Regel formuliert, die nachstehend definiert werden soll.

Definition 2.69 (Generalisierter Modus ponens)

Es seien A, A' Fuzzy-Mengen über einer Grundmenge G_1, B, B' Fuzzy-Mengen über einer Grundmenge G_2. Dann heißt die nachfolgend dargestellte Schlussfigur generalisierter Modus ponens.

$$\frac{X \text{ ist } A \rightarrow Y \text{ ist } B \qquad X \text{ ist } A'}{Y \text{ ist } B'}$$

Hierbei werden die Wenn-dann-Aussage der Prämisse durch einen Fuzzy-Implikations-Operator, das Ziehen der Schlussfolgerung durch eine sog. Kompositionsregel der Inferenz realisiert.

Bemerkung 2.70

Es ist zu beachten, dass Definition 2.69 zunächst rein formal zu sehen ist; für reale Anwendbarkeit müssen die Mengen A, A' und B, B' natürlich paarweise etwas miteinander zu tun haben. Vgl. hierzu auch [Alie2000], S. 79ff. Im klassischen Modus ponens waren sie jeweils gleich, $A = A'$, $B = B'$.

Erst durch die Festlegung, durch welche Operatoren die Inferenz mathematisch dargestellt wird, ist die obige Definition anwendbar. Für eine ausführlichere Diskussion möglicher Implikationsoperatoren sei wiederum auf [Böhm1993], S. 263ff, verwiesen. Im Folgenden werden einzelne hiervon genauer vorgestellt und diskutiert werden.

Beispiel 2.71

An einem konkreten Beispiel soll der generalisierte Modus ponens illustriert werden.

Wir betrachten einen Zusammenhang zwischen der Körpergröße eines Autofahrers X und dem Maß, wie geeignet ein bestimmter Wagentyp Y für ihn ist.

Die Grundmenge G_1 sei das Intervall [0,250], das für die Körpergröße in Zentimetern stehen soll. Die Grundmenge G_2 beschreibe numerisch in Form der Zahlenwerte { 0, 1, ..., 10 } die Skala von “überhaupt nicht geeignet” (=0) bis “sehr gut geeignet” (=10).

Es seien: $A := [160, 170, 180]$, $A' := [170, 180, 190]$, $B := \frac{0.5}{8} + \frac{0.75}{9} + \frac{1}{10}$. Hierbei sind die Mengen A, A' trianguläre Fuzzy-Zahlen (vgl. Notation auf S. 52), und die diskrete Fuzzy-Zahl B ist gemäß der Zadehschen Notation (siehe Bemerkung S. 44) geschrieben.

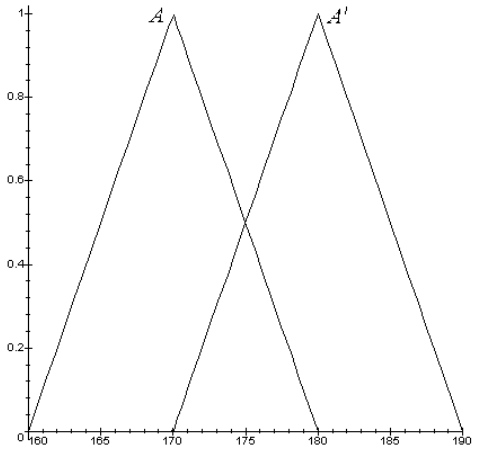


Abb. 2.12: Die Fuzzy-Mengen A und A'

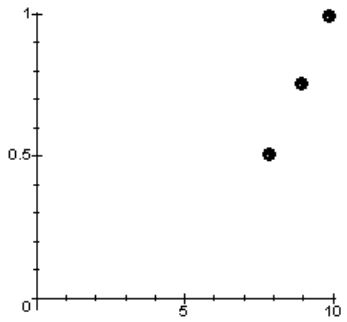


Abb. 2.13: Die Fuzzy-Menge *B*
(Der Zugehörigkeitswert 0 wurde der Übersichtlichkeit wegen nicht markiert.)

Verwenden wir den Zadehschen Implikationsoperator $I_Z(x,y) := \max(\min(x,y), 1-x)$, so erhalten wir hier folgendes Zugehörigkeitstableau für die Implikation "if X is A then Y is B". Um die Darstellung etwas übersichtlicher zu gestalten haben wir hier den Grundbereich auf die Werte 160, 165, 170, ..., 190 bzw. 7, 8, 9, 10 eingeschränkt.

if X is A then Y is B	Zugehörigkeitsgrad		0	0,5	0,75	1
Zugehörigkeitsgrad		B	7	8	9	10
0	A	160	0	0	0	0
0,5		165	0	0,5	0,5	0,5
1		170	0	0,5	0,75	1
0,5		175	0	0,5	0,5	0,5
0		180	0	0	0	0
0		185	0	0	0	0
0		190	0	0	0	0

Tab. 2.3: Tableau der Implikation "if X is A then Y is B"

Zu lesen sind die Einträge hier z.B.: Für die konkreten Werte X=170 und Y=9 ist die Aussage "if X is A then Y is B" mit dem Zugehörigkeitsgrad 0,75 erfüllt.

Rechnerisch: $\mu_A(170) = 1$, $\mu_B(9) = 0,75$, und der Zadehsche Implikationsoperator liefert $I_Z(1, 0.75) := \max(\min(1, 0.75), 1-1) = 0.75$.

A'						
160	165	170	175	180	185	190
0	0	0	0,5	1	0,5	0

Tab. 2.4: Zugehörigkeitsgrade (Auszug) zur Menge A'

Die Inferenz wird über die max-min-Komposition realisiert.

if X is A then Y is B		B	0	0,5	0,75	1
			7	8	9	10
0	A 160		0	0	0	0
0,5	165		0	0,5	0,5	0,5
1	170		0	0,5	0,75	1
0,5	175		0	0,5	0,5	0,5
0	180		0	0	0	0
0	185		0	0	0	0
0	190		0	0	0	0

A'								B'	7	8	9	10
160	165	170	175	180	185	190			**0**	**0,5**	**0,5**	**0,5**
0	0	0	0,5	1	0,5	0						

Tab. 2.5: Inferenz des generalisierten Modus ponens

Die rechts unten dargestellten Zugehörigkeitsgrade zu “Y is B'” erhalten wir zu gegebenem Y-Wert y durch komponentenweise Bildung der Minima von $\mu_{if\ X\ is\ A\ then\ Y\ is\ B}(x,y)$ mit $\mu_{A'}(x)$ und anschließendem Ermitteln des Maximums:

$$\mu_{Y\ is\ B'}(y) = \max\{\ \min\{\ \mu_{if\ X\ is\ A\ then\ Y\ is\ B}(x,y), \mu_{A'}(x)\} \mid x \in A\ \}.$$

2.5 Fuzzy-Entscheidungssysteme

Die Methoden der Fuzzy-Logik werden in der Praxis vorwiegend im Kontext der technischen Steuerung in Form sogenannter Fuzzy-Controller zum Einsatz gebracht.

Weiterhin kommt Fuzzy-Logik im Rahmen der Mustererkennung und in Expertensystemen zur Anwendung. Auch hier findet eine Fuzzy-Regelbasis ihren Einsatz. Übergreifend wird daher der Begriff des Fuzzy-Entscheidungssystems verwendet, der die Fuzzy-Controller ebenso umfasst wie die Fuzzy-Expertensysteme. Vgl. hierzu [Tenh2000], S. 61ff.

Der traditionelle Weg einer Problemlösung wird über den Versuch einer möglichst guten mathematischen Modellierung beschritten; hierfür ist sowohl fachspezifisches Know-How aus dem jeweiligen Einsatzgebiet, als auch mathematisches Wissen erforderlich. Darüber hinaus sind für viele praktische Problemstellungen keine exakten mathematischen Modelle bekannt.

Setzt man klassische Regeln ein, so stößt man rasch auf das Problem der Grenzbereiche. Hierzu betrachten wir ein kleines Beispiel (zitiert nach [Hell1997], S. 163f).

Beispiel 2.72

Betrachten wir ein Regelungssystem für einen Druckkessel, bei dem Temperatur und Druck zum Öffnen oder Schließen eines Notventils führen können. Eine “crispe” Schlussregel könnte hier lauten: “Wenn die Temperatur >= 400 Grad (Celsius) und der Druck >= 30 Bar beträgt und das Hauptventil geschlossen ist, dann öffne das Notventil, andernfalls schließe es.”

So verständlich und einsichtig diese Regel ist, es treten in der Praxis sofort Fragen auf, die v.a. die Grenzwerte betreffen, teilweise aber auch den Fall fehlender Informationen betreffen.

1. Was soll (bei geschlossenem Hauptventil) mit dem Notventil bei einer Temperatur von 390 Grad und einem Druck von 40 Bar passieren? Gemäß der obigen Regel mit der "Und"-Verknüpfung wird das Notventil geschlossen, - aber ist dies auch fachlich die gewünschte Reaktion?
2. Was passiert, wenn die Temperatur auf 600 Grad steigt, der Druck aber unter 30 Bar liegt? Wäre hier das Öffnen des Notventils erforderlich?
3. Was soll geschehen, wenn die Temperatur (z.B. wegen eines technischen Defekts) nicht bekannt ist, der Druck aber sehr deutlich über die genannte Grenze von 30 Bar gestiegen ist?

Was das klassische System dieser "starren" Regeln in der obigen Form nicht berücksichtigen kann, das sind Situationen, in denen teilweise Informationen nicht (oder nicht präzise) vorliegen; weiterhin kann es sein, dass aufgrund der starren Regeln fachlich unkorrekt reagiert wird, wenn bei mehrteiligen konjunktiv verknüpften Prämissen einige Prämissen-Teile erfüllt sind, mindestens ein Teil jedoch nicht.

Durch die Fuzzy-Logik besteht die Möglichkeit, Regeln "weich" zu formulieren, was in einem bestimmten Formalismus die Modellierung von (z.B. menschlicher) Unsicherheit zulässt. So kann etwa auf einen "sehr hohen" Druckanstieg auch dann mit dem Öffnen des Notventils reagiert werden, wenn die Temperaturwerte entweder nicht kritisch sind oder evtl. in dem Moment gar nicht vorliegen.

Nachfolgend wollen wir zwei grundlegende Modelle von Fuzzy-Controllern vorstellen, die von Mamdani sowie von Sugeno und Tagaki. Dabei folgen wir weitgehend den Ausführungen von Feuring und Tenhagen ([Feur1995], [Tenh2000]).

Weiterhin führt Feuring in seiner Arbeit aus[11], dass die in den nächsten Abschnitten vorgestellten Fuzzy-Controller universell in dem Sinne sind, dass sich jede stetige Funktion (auf einem kompakten Definitionsbereich) durch einen solchen Controller beliebig genau approximieren lässt.

Bemerkung 2.73 (Konzept des Mamdani-Controllers)

Bereits in den Siebziger Jahren hat L. Zadeh das Konzept der Fuzzy-Regelung in [Zade1972] und [Zade1973] vorgestellt. Eine erste Anwendung fand dies durch Mamdani und Assilian ([Mamd1975]) im Kontext einer Dampfmaschinensteuerung.

[11] Sh. dort S. 64ff.

Definieren wir zunächst formal, was wir unter einem Mamdani-Controller verstehen wollen. Die Formulierung dieser Definition stammt aus [Tenh2000], S.62.

Definition 2.74 (Mamdani-Controller)

Ein *Mamdani-Controller* ist eine Steuereinheit, die Messdaten erhält und hieraus Steuerungsdaten berechnet und ausgibt. Die vier Hauptkomponenten sind:

1. ein *Fuzzifizierer*, der crispe Eingaben in Fuzzy-Mengen umwandelt;
2. eine *Regelbasis*, d.h. eine endliche Menge von Regeln der Form
   IF (x_1 IS a_1) AND ... AND (x_n IS a_n) THEN (y IS b); dabei sind a_1, ..., a_n und b Fuzzy-Mengen.
3. eine *Entscheidungslogik*, die diese Regeln auf die fuzzifizierten Eingaben anwendet und Fuzzy-Werte berechnet;
4. ein *Defuzzifizierer*: hier werden die Fuzzy-Ausgaben der Entscheidungslogik in korrespondierende crispe (d.h. reelle) Werte umgesetzt.

Dabei sind die x- und y-Werte in der Praxis reelle Zahlen; prinzipiell kann das hier auftretende y auch aus einem höherdimensionalen reellen Raum stammen.

Die nachstehende Abbildung illustriert den Aufbau eines solchen Controllers anschaulich.

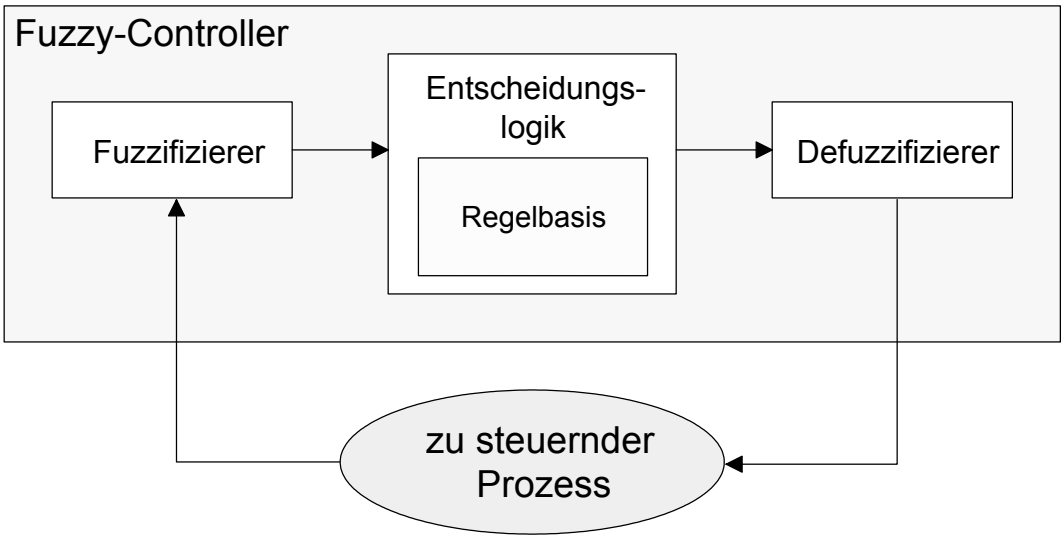


Abb. 2.14: Aufbau eines Fuzzy-Controllers nach Mamdani

Der Ablauf lässt sich schematisch wie folgt beschreiben. Der zu steuernde Prozess liefert crispe Werte x_1, ..., x_n. Diese werden vom Fuzzifizierer zu Fuzzy-Mengen transformiert, meist dadurch, dass ein sogenannter Singleton-Fuzzifizierer einen crispen Wert x_i auf das zugehörige Singleton a_{x_i} abbildet (vgl. Definition 2.27 auf S. 50). Entsprechend wird dem Vektor x die höherdimensionale Fuzzy-Menge a_x zugeordnet.

Eine Bedingung der Form "IF x IS a" beschreibt das Maß der Zugehörigkeit, "wie sehr" x zur Fuzzy-Menge a zuzurechnen ist; dies geschieht durch Berechnung des Zugehörigkeitsgrades $\mu_a(x)$.

Die mehrteiligen Prämissen sind anschließend konjunktiv zu verknüpfen; häufig wird hierfür der min-Operator verwendet, der den kleinsten auftretenden Zugehörigkeitsgrad ermittelt. Generell sind hier aber beliebige t-Normen einsetzbar (vgl. Def. 2.15).

Die Fuzzy-Menge a_x wird mittels einer Regel R aus der Regelbasis auf eine Fuzzy-Menge $a_x \circ R$ abgebildet, für deren Zugehörigkeitsfunktion gilt:

$$\mu_{a_x \circ R}(z) = \mu_{a_1}(x_1) * \ldots * \mu_{a_n}(x_n) * \mu_b(z).$$

Diese Formel gilt für die hier betrachteten Singleton-Fuzzifizierungen. Das Zeichen $*$ steht für die dabei verwendete t-Norm, also etwa den min-Operator. Die Konjunktion erstreckt sich hierbei auch auf $\mu_b(z)$, da die Zugehörigkeit von z zu b durch die Zugehörigkeitsgrade der betreffenden Prämissen begrenzt werden soll.

Im Extremfall: wenn der Zugehörigkeitsgrad der Prämissen bzw. eines Prämissenteils 0 ist, kann die betreffende Regel für die vorliegenden Werte auch keine Aussagekraft für sich beanspruchen.

Bemerkung 2.75 (Defuzzifizierung)

Für die in Definition 2.74 angesprochene Defuzzifizierung, also die Abbildung einer Fuzzy-Menge zu einer crispen Zahl, existieren verschiedene Möglichkeiten. Drei Varianten sollen hier kurz vorgestellt werden.

Eine einfache Methode ist die *Maximum-Defuzzifizierung*. Dabei wird zu einer Fuzzy-Menge a der oder ein Punkt des Trägers ausgewählt, für den der Zugehörigkeitsgrad $\mu_a(x)$ maximal wird. Gibt es nicht nur einen Punkt mit maximalem Zugehörigkeitsgrad, so ist dies noch keine eindeutige Festlegung; man kann - mit einer gewissen Willkür - definieren, dass man den kleinsten Wert x mit maximalem Zugehörigkeitsgrad $\mu_a(x)$ bestimmt.

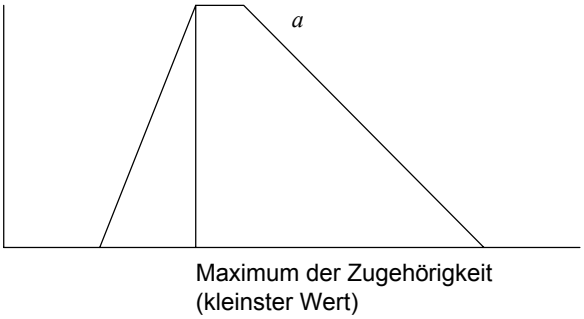


Abb. 2.15: Beispiel zur Defuzzifizierung - Maximum-Methode

Während die Berechnung des defuzzifizierten Wertes nach der Maximum-Methode sehr schnell geht, ist festzuhalten, dass von der genauen Gestalt der Fuzzy-Menge auf diese Weise viel Information verloren geht.

Einen anderen Weg beschreitet die *Schwerpunkt-Defuzzifizierung* (*center of gravity*). Hier wird als crisper Wert y derjenige verwendet, der unter dem Schwerpunkt der durch die Zugehörigkeitsfunktion erzeugten Fläche liegt. Mathematisch heißt dies:

$$y := \frac{\int x \cdot \mu_a(x)\, dx}{\int \mu_a(x)\, dx}.$$

Dabei erstrecken sich die Integrale jeweils über den Träger der Zugehörigkeitsgrad μ_a oder äquivalent über die Menge der reellen Zahlen, da außerhalb des Trägers $\mu_a(x) = 0$ ist. Sollte der Träger aus singulären, diskreten Punkten bestehen, so ist die o.g. Integraldarstellung durch die entsprechende diskrete Summe zu ergänzen bzw. zu ersetzen: $y = \frac{\sum x \cdot \mu(x)}{\sum \mu(x)}$.

Der Nachteil an der Schwerpunktmethode ist die i.a. aufwändige Berechnung. Dafür liefert sie zumindest für Fuzzy-Mengen, die als Träger ein Intervall besitzen, ein brauchbares Resultat, das - anders als die Maximummethode - die konkrete Gestalt der Fuzzy-Menge mit all ihren Punkten berücksichtigt. Besteht der Träger jedoch beispielsweise aus zwei separaten Intervallen, dann könnte der hier ermittelte Schwerpunktwert außerhalb des Trägers liegen, was nicht sinnvoll ist.

Eine dritte Vorgehensweise ist die Methode "*Mittelwert der Maxima*" (*mean of max*). Hier wird der Mittelwert aller Punkte ermittelt, über denen die Zugehörigkeitsfunktion μ_a das Maximum annimmt. Gibt es im Spezialfall nur genau einen solchen Punkt, ist das Resultat natürlich dasselbe wie bei der Maximummethode.

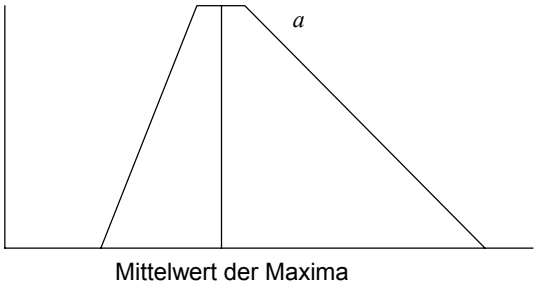


Abb. 2.16: Beispiel zur Defuzzifizierung - Mittelwert der Maxima-Methode

Im Gegensatz zur einfachen Mittelwertmethode werden hierbei zumindest alle Punkte der Fuzzy-Menge mit maximaler Zugehörigkeit berücksichtigt. Wiederum ist allerdings das Resultat unplausibel, wenn die Fuzzy-Menge nicht ein Intervall als Träger besitzt.

Welches Defuzzifizierungsverfahren in der Praxis gewählt wird, hängt angesichts der verschiedenen Vor- und Nachteile von der jeweiligen Problemstellung ab.

Bemerkung 2.76 (Fuzzy-Controller nach Sugeno und Takagi)

Das Konzept des Mamdani-Controllers wurde von Sugeno und Takagi modifiziert ([Suge1985], [Suge1985a]). Hier wird bereits im Anweisungsteil, dem Resultat der Regelerzeugung, direkt ein crisper Wert ermittelt, so dass die Defuzzifizierung nicht mehr erforderlich ist, was den Rechenaufwand bei der Steuerung deutlich reduziert.

Die *r* Regeln haben hierbei die Form

$$\text{“IF } (x_1 \text{ IS } a_1) \text{ AND } \ldots \text{ AND } (x_n \text{ IS } a_n) \text{ THEN } y = f_i(x_1,\ldots,x_n)\text{”},$$

wobei die Funktionen $f_1, \ldots, f_r$ bereits crispe Ergebnisse liefern und der Index *i* die betreffende Regel Nr. *i* bezeichnet.

Sind $a_1, \ldots, a_r$ die Grade, zu denen die Regeln R_1, ..., R_r erfüllt sind, so ist die Ausgabe des Sugeno-Controllers gerade $y = \frac{\sum_{1 \le i \le r} a_i f_i(x_1,\ldots,x_n)}{\sum_{1 \le i \le r} a_i}$.

Allerdings kann bei diesem Konzept keine Regelbasis mit rein linguistischen Ausdrücken generiert werden, da für jede einzelne Regel der funktionale Zusammenhang (f_i) ermittelt werden muss und dieser i.a. keine semantisch-äquivalente umgangssprachliche Formulierung besitzt.

Bemerkung 2.77 (Universalität von Fuzzy-Controllern)

Fuzzy-Controller finden u.a. deshalb so weit verbreiteten Einsatz, weil sich die Art der generierten Regeln (vor allem gemäß dem Konzept von Mamdani) sehr stark an umgangssprachlichen Formulierungen orientiert.

Darüber hinaus ist festzuhalten, dass sich mit den hier vorgestellten Fuzzy-Controller-Konzepten alle stetigen Funktionen beliebig genau approximieren lassen. Vgl. hierzu die Ausführungen von Feuring, [Feur1995], S. 64ff.

3 Kombination von Neuronalen Netzen und Fuzzy-Anwendungen

Die weitverbreitete praktische Anwendung von Fuzzy-Systemen wurde bereits in Abschnitt 2.5 (sh. S.68) hervorgehoben. Positive Synergien lassen sich dabei durch das Zusammenspiel von Fuzzy-Systemen mit dem Konzept der Neuronalen Netze erzielen. Hierzu soll in diesem Kapitel ein kurzer Überblick gegeben werden.

Für das weitere Vorgehen mit dem Ziel, möglichst gute Regeln aus Neuronalen Netzen zu extrahieren, wollen wir ebenfalls in diesem Kapitel das Fundament legen und die Approximationsfähigkeit und die (in noch zu präzisierendem Sinne) universelle Einsetzbarkeit Neuronaler Netze beschreiben.

Prinzipiell können die Methoden der Fuzzy-Systeme und der Neuronalen Netze in zwei Richtungen kombiniert werden. Die folgende tabellarisch vorgestellte Unterteilung orientiert sich an Zadeh; diese ist jedoch nicht die einzig praktizierte Vorgehensweise, andere Autoren nehmen auch andere Klassifizierungen vor.

Übersicht der verwendeten Begrifflichkeiten
(zitiert nach [Tenh2000], S. 76)

Fuzzy-Neuro-Systeme	**Neuro-Fuzzy-Systeme**
Fusioniert: - Systeme, die Fuzzy-Zahlen verwenden - Stabilitäts- und Qualitätsaspekte klassischer Neuronaler Netze	Kooperativ: - Optimierung von Komponenten von Fuzzy-Entscheidungssystemen
Hybrid: - Automatisierte Parametersteuerung von Neuronalen Netzen	Hybrid: - Optimierung kompletter Fuzzy-Entscheidungssysteme

Zunächst wollen wir kurz die Vor- und Nachteile der beiden Ansätze, also von Neuronalen Netzen und Fuzzy Entscheidungssystemen, beleuchten. Dabei folgen wir teilweise den Ausführungen von Tenhagen in [Tenh2000].

3.1 Vor- und Nachteile der Konzepte

So verschieden die Konzepte der Neuronalen Netze und der Fuzzy-Systeme sind, so günstig erweisen sich die jeweiligen Vorteile, die sich geschickt ergänzen können.

Zu den wesentlichen Vorteilen Neuronaler Netze gehören:

1. deren Lernfähigkeit und Adaptivität;
2. die Parallelverarbeitung innerhalb des Systems;
3. der Lernprozess kann ohne problemspezifisches Vorwissen ablaufen.

Dabei sollen aber auch einige Merkmale genannt werden, die sich eher nachteilig für den praktischen Einsatz auswirken:

1. ein vergleichsweise hoher Rechenaufwand;
2. Trainings- und Testdaten müssen in ausreichend großem Umfang vorliegen (um ein gutes Netz zu generieren); darüber hinaus gibt es keine Fehlerabschätzung für Daten außerhalb dieser Mengen;
3. der Transfer des "Black Box"-Netzes von und zu einer menschlichen Wissensrepräsentation ist sehr schwierig, d.h. es kann in der Praxis nicht oder nur in geringem Maße vorhandenes Wissen in ein neues Netz integriert werden, ebenso ist die Extraktion von Regeln aus einem (trainierten und damit gegenüber der Startsituation veränderten) Netz sehr schwierig.

Dieser letzte Punkt wird im Folgenden zentral für unsere weitere Arbeit sein (vgl. Kapitel 5). Im Kontext von Multilayer Feedforward-Netzen werden hier einige Resultate vorgestellt werden. Insgesamt bleibt jedoch festzuhalten, dass es praktisch nicht möglich ist, aus einem *beliebigen* Neuronalen Netz eine in jedem Falle überschaubare und damit wirklich "hilfreiche" Regelbasis zu extrahieren, mit der ein solches Netz besser verständlich gemacht werden könnte.

Fuzzy-Entscheidungssysteme weisen auf der anderen Seite einigermaßen komplementäre Stärken und Schwächen auf. Zu deren Vorteilen gehören:

1. die einfache Wissensintegration und -extraktion in Form von Regeln; gute Interpretierbarkeit des Systems;
2. keine Trainingsphase; somit sind auch keine Trainings- und Testdaten erforderlich.

Demgegenüber zählen zu den negativen Aspekten von Fuzzy-Systemen:

1. für den nutzbaren Einsatz ist umfangreiches, problemspezifisches Wissen erforderlich (zur Einstellung der Regelbasis);
2. das System ist aufwändig (manuell) durch Experten zu konfigurieren;
3. die Fuzzy-Regelbasis ist nicht adaptiv, d.h. sie passt sich nicht dynamisch an Erfordernisse an, so dass keine fortlaufende Selbst-Optimierung des Systems stattfinden kann.

Durch eine geeignete Kombination der beiden Konzepte kann jedoch versucht werden, die verschiedenen Vorteile nutzbar zu machen.

3.2 Fusionierte Fuzzy-Neuro-Systeme

Einmal können Neuronale Netze mit Methoden der Fuzzy-Theorie "angereichert" werden, d.h. konkret werden anstelle crisper Zahlen (oder Vektoren) Fuzzy-Mengen (beispielsweise Fuzzy- Zahlen allgemeiner oder spezieller Form) eingesetzt. Hier hat sich der Begriff des "fusionierten Fuzzy-Neuro-Systems" durchgesetzt ebenso wie der des "Fuzzy-Neuronalen Netzes". Dabei kann variiert werden, welche Zahlen fuzzy sein können, oftmals alle (Ein-

und Ausgaben ebenso wie Gewichte und Parameter), bisweilen aber auch nur eine Teilmenge davon.

Einer der Vorteile dieser Netze ist, dass sie mit unscharfen (sprich: ungenauen) Daten operieren können, so dass eine besondere Datenvorverarbeitung in vielen Fällen nicht erforderlich ist. Darüber hinaus sind aber auch Resultate bekannt, die die Stabilitäts- und Qualitätseigenschaften klassischer Neuronaler Netze betreffen (vgl. [Feur1995]).

Dadurch, dass ein solches Fuzzy-Neuronales-Netz unscharfe Daten in Form von Fuzzy-Zahlen verarbeiten kann, entfällt eine aufwändige Datenvorverarbeitung. Aufgrund der Eigenschaften der fuzzy-arithmetischen Operationen (vgl. Bemerkung 2.51) ist jedoch das Aufstellen von Lernverfahren für diese Netze nicht unproblematisch.

In [Lipp1995b], [Lipp1995c], [Lipp1998] werden eine Reihe von Lernalgorithmen für solche Fuzzy-Neuronale-Netze vorgestellt[12].

3.3 Hybride Fuzzy-Neuro-Systeme

Außerdem kann ein Fuzzy-System dazu genutzt werden, Einfluss auf den Lernprozess eines Neuronalen Netzes zu nehmen. Menschliches Expertenwissen kann zur Konstruktion von Fuzzy-Expertensystemen dazu genutzt werden, die generell recht aufwändige Steuerung der Lernparameter beim Training eines Neuronalen Netzes zu übernehmen (sh. [Lipp1995a]). Damit erhält man ein sog. “hybrides Fuzzy-Neuronales System”.

Unter hybriden Fuzzy-Neuro-Systemen versteht man also Neuronale Netze, die durch Zuhilfenahme von Fuzzy-Methoden verbessert werden. Oftmals gestaltet sich die Lernphase eines Neuronalen Netzes auf ein gegebenes Problem aufwändig, da eine ganze Reihe von Parametern günstig eingestellt werden müssen.

Hier können aus Vorerfahrung oftmals mit spezifischem Wissen sinnvolle Bereiche für bestimmte Parameter angegeben werden. Fuzzy-Entscheidungssysteme können konstruiert werden, die die Steuerung der Lernparameter eines Neuronalen Netzes übernehmen.

3.4 Kooperative Neuro-Fuzzy-Systeme: Optimierung von Fuzzy-Entscheidungssystemen durch Neuronale Netze

Auf der anderen Seite kann das Konzept des Neuronalen Netzes zur Optimierung von gegebenen Fuzzy-Entscheidungssystemen dienen, denn es hat sich in der Praxis gezeigt, dass ein erstes “grobes” Fuzzy-System für ein Problem vergleichsweise schnell generiert werden kann, sich anschließend aber die Optimierung oftmals als recht mühsam erweist. Bei diesem zweiten Ansatz wird üblicherweise von Neuronalen Fuzzy- oder etwas kürzer von Neuro-Fuzzy-Systemen gesprochen.

Man unterscheidet hier zwischen kooperativen und hybriden Ansätzen, je nachdem, wie stark die beiden Systeme “Fuzzy” und “Neuronales Netz” voneinander getrennt sind.

---

[12] Hier zitiert nach [Feur2000], S. 77.

Bei einem kooperativen System wirken im Kern zwei eigenständige Teilsysteme. Tenhagen stellt in seiner Arbeit ein fuzzifiziertes Kohonen-Netz vor, also ein Fuzzy-Neuronales-Netz, mit dem die Partitionierungen für Ein- und Ausgaberäume eines Fuzzy-Entscheidungssystems gefunden werden können (vgl. [Tenh2000], S. 82ff).

Beispiel 3.1 (Fuzzy Associative Memories, FAM)

Feuring stellt in [Feur1995] die sog. *Fuzzy Associative Memories* (*FAM*) von Kosko als Beispiel für ein kooperatives Neuro-Fuzzy-System vor.

Bei einem solchen Assoziativspeicher handelt es sich um eine Abbildung von einer endlichen Fuzzy-Menge in eine zweite. Ist $X := \{x_1, ..., x_n\}$ eine endliche Menge, dann lässt sich jede Fuzzy-Menge $\tilde{a}$ über X, $\tilde{a} \in FM(X)$, als Element des n-dimensionalen Quaders $[0,1]^n$ auffassen, wobei in jeder Dimension gerade der entsprechende Zugehörigkeitswert steht: $\tilde{a} = (\mu_{\tilde{a}}(x_1), ..., \mu_{\tilde{a}}(x_n))$. (Zur formalen Definition einer Fuzzy-Menge sei auf Def. 2.1 hingewiesen.)

Sind $Y := \{y_1, ..., y_p\}$ eine zweite endliche Menge und $\tilde{b} \in FM(Y)$, so wird ein FAM beschrieben durch eine linguistische Regel R der Art

$$IF\ \xi_1 \in \tilde{a}_1\ AND\ ...\ AND\ \xi_m \in \tilde{a}_m\ THEN\ \eta \in \tilde{b}\ .$$

Mathematisch ist dies also als Abbildung

$$R:\ \underbrace{[0,1]^n \times ... \times [0,1]^n}_{m-mal} \rightarrow [0,1]^p$$

aufzufassen. Ein FAM-System besteht aus einer Kombination mehrerer solcher Regeln, wie nachstehend skizziert.

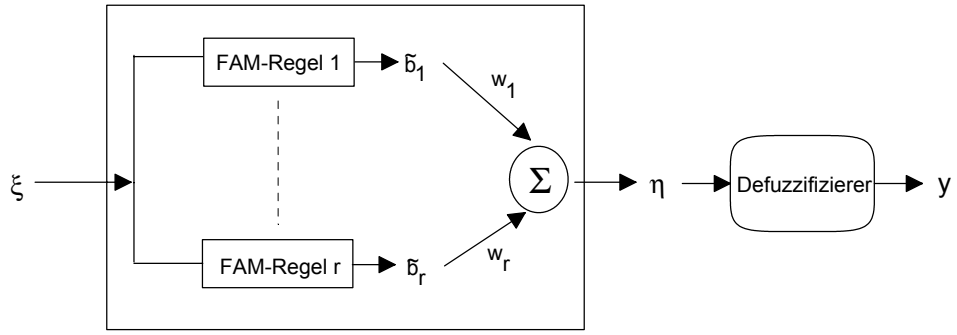


Abb. 3.1: Skizze eines FAM-Systems (vgl. [Feur1995], S. 79)

Ein solches FAM-System ist von der Struktur mit einem zweischichtigen Neuronalen Netz vergleichbar. Zur Erzeugung von Regeln verwendet Kosko entsprechend ein Zweischichtnetz; in unserer obigen Darstellung besitzt die Eingabeschicht *m+1* Neuronen für die m Eingabegrößen (Messgrößen) und die eine Ausgabedimension (Stellgröße). Die Größe der Ausgabeschicht korrespondiert mit der Anzahl der linguistischen Terme für die Mess- und Stellgrößen. Seien für die Messgröße i gerade p_i linguistische Terme und für die Stellgröße q vorgesehen, so verfügt das Neuronale Netz über $q \cdot \prod_{1 \le i \le m} p_i$ Ausgabeneuronen. Dies entspricht der Anzahl aller möglichen Regeln, die aus den linguistischen Termen der m Messgrößen gebildet werden können. Das anschließende Trainieren des Neuronalen Netzes

dient der Entscheidungsfindung, welche dieser Regeln eine hohe Bedeutung besitzen und welche nicht.

Auf die arithmetischen Operationen innerhalb eines FAM-Systems und den Lernvorgang des skizzierten Neuronalen Netzes soll an dieser Stelle nicht weiter eingegangen werden; hierzu sei auf [Feur1995] (S. 77ff) verwiesen. Es geht hier lediglich darum, exemplarisch aufzuzeigen, dass bei diesem Modell ein Neuronales Netz genutzt wird, um Regeln eines Fuzzy-Systems zu generieren.

3.5 Hybride Neuro-Fuzzy-Systeme

Der hybride Ansatz beschreibt einheitliche Systeme, deren Struktur im Kern meist einem Neuronalen Netz entspricht, das die Regelbasis eines Fuzzy-Controllers repräsentiert.

Beispiel 3.2 (NEFCON)

Eines der bekanntesten hybriden Neuro-Fuzzy-Systeme ist NEFCON (Neural Fuzzy Controller), das Fuzzy-Entscheidungssysteme dadurch optimiert, dass diese in die Struktur eines dreischichtigen Neuronalen Netzes überführt werden. Die nachstehende Abbildung zeigt exemplarisch die Situation im Falle von $r = 4$ Regeln. Vgl. hierzu [Borg2003], S. 330ff, sowie [Nien2003], S. 89ff.

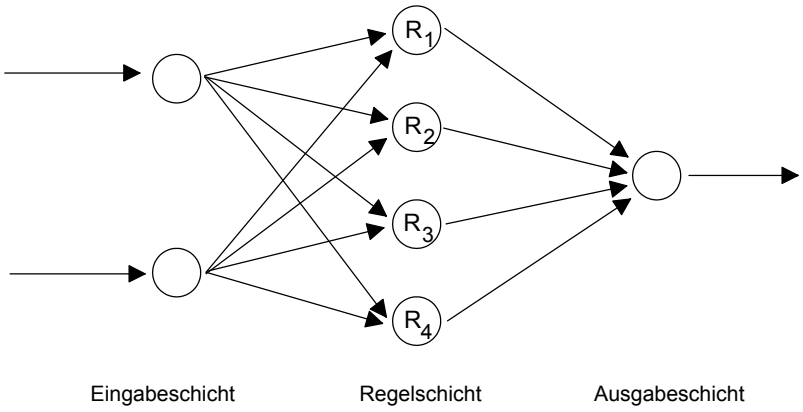


Abb. 3.2: Aufbau eines NEFCON-Netzes

Das Neuronale Netz beim NEFCON-Ansatz verwendet als Gewichte Fuzzy-Mengen anstelle crisper Zahlenwerte. Wird ein Fuzzy-Controller mit r Regeln, einer n-dimensionalen Eingabe und einer eindimensionalen Ausgabe repräsentiert, so hat das entsprechende Netz n Neuronen in der Eingabeschicht, in der verborgenen Schicht r Neuronen, die jeweils eine bestimmte Regel vertreten, und ein Ausgabeneuron. Die Gewichte im Netz sind Fuzzy-Mengen, die in der Praxis linguistische Terme verkörpern, beispielsweise “hoch” oder “mittel”. Die Gewichte von der Eingabeschicht zur Regelschicht stehen für die Prämissen, die Gewichte zur Ausgabeschicht (bzw. zu dem einen Ausgabeneuron) für die Konklusionen der Regeln.

Hat beispielsweise Regel Nr. 2 die Form “IF x_1 IS hoch AND x_2 IS mittel THEN y IS niedrig”, dann bedeutet dies, dass das Gewicht vom ersten Eingabeneuron zum Neuron R_2

eine Fuzzy-Menge zur Darstellung von “hoch” ist; entsprechend besitzt die Verbindung vom zweiten Eingabeneuron zum Neuron R_2 eine Fuzzy-Menge für “mittel”. Schließlich liegt an der Verbindung von Neuron R_2 zum Ausgabeneuron ein Fuzzy-Gewicht “niedrig” an.

Für das Training werden zwei unterschiedliche Lernverfahren für die Fuzzy-Mengen und die Regeln eingesetzt. Voraussetzung für die Anwendbarkeit der Lernverfahren ist die Monotonie der Zugehörigkeitsfunktionen der Fuzzy-Mengen, die als Gewichte zum Ausgabeneuron fungieren, auf deren Träger. Daher werden an dieser Stelle oftmals Fuzzy-Mengen mit “zackenförmiger” Zugehörigkeitsfunktion verwendet, wie im nachstehenden Bild dargestellt.

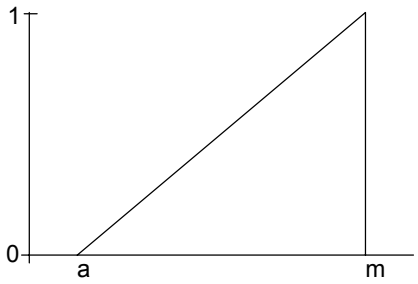


Abb. 3.3: Fuzzy-Menge mit zackenförmiger Zugehörigkeitsfunktion

Beispiel 3.3 (Lin und Lee)

Auf der gleichen Grundidee basiert auch das System von Lin und Lee; hier wird ein fünfschichtiges Neuronales Netz zu dem ursprünglichen Fuzzy-Entscheidungssystem aufgebaut. Die Schichten repräsentieren - wie im nachstehenden Bild dargestellt - sukzessive die Eingabe, die Eingabepartitionierungen, die Regeln, die Ausgabepartitionierungen und schließlich die Ausgabe.

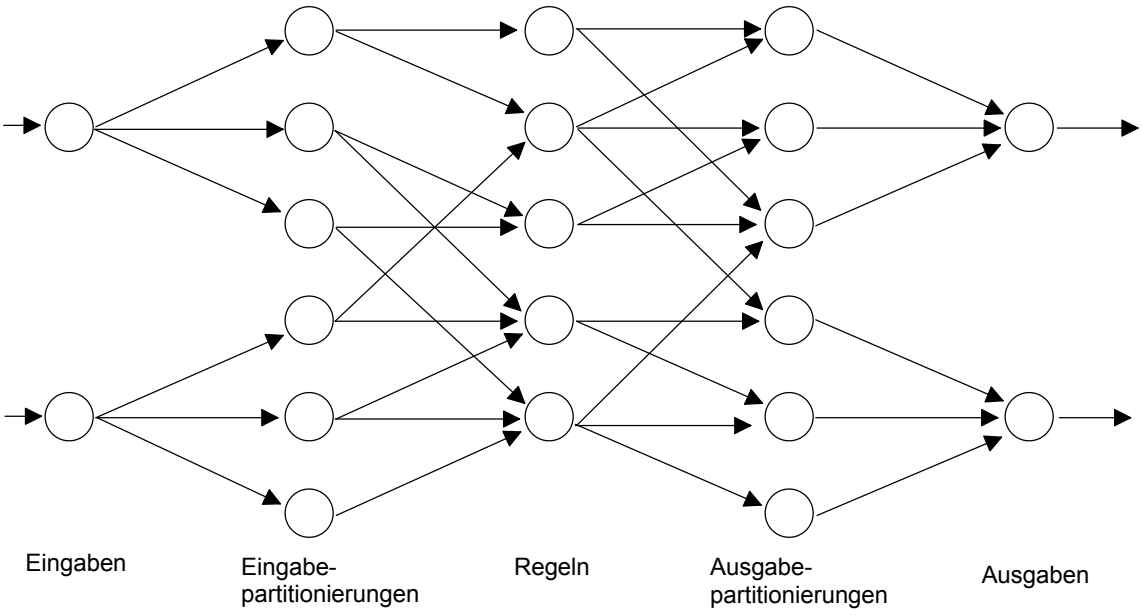


Abb. 3.4: Struktur eines Netzes nach Lin und Lee

Vgl. [LinL1991], [LinL1993] sowie für eine ausführlichere Veranschaulichung [Feur1995], S. 82ff, und [Nien2003], S. 79ff.

Bemerkung 3.4

Der Vorteil dieser Ansätze gegenüber reinen Neuronalen Netzen kann darin gesehen werden, dass nach erfolgter Trainingsphase von Experten die im Controller generierten Regeln begutachtet und interpretiert werden können.

Umgekehrt ist jedoch festzuhalten, dass bei diesen Modellen Neuronale Netze einer sehr konkret vorgegebenen Struktur zum Einsatz kommen. Für unser Ziel, Regeln aus möglichst allgemeinen Feedforward-Netzen mit nur wenigen Vorbedingungen zu extrahieren, sind diese spezifischen Situationen zu einengend.

4 Neuronale Netze als Fuzzy-Entscheidungssysteme

Wir wollen in diesem Kapitel darstellen, dass unter bestimmten Voraussetzungen ein (bereits trainiertes) Neuronales Netz als Fuzzy-Entscheidungssystem (FES) gemäß der Definition 2.74 eines Mamdani-Controllers bzw. eines Sugeno-Controllers (vgl. 2.76) interpretiert werden kann.

In diesem Fall interessiert natürlich eine möglichst einfache, menschlich verständliche Formulierung der entsprechenden Regeln. Ließe man beliebig komplizierte Regeln zu, so wäre die Fragestellung trivial, denn die hier betrachteten Neuronalen Netze beschreiben stets eine deterministische, funktionale Zuordnung von Ein- zu Ausgabewerten. Ebenso sollte für den praktischen Einsatz die Anzahl der Regeln nicht zu groß sein.

Die Dissertation von A. Tenhagen [Tenh2000] und die Arbeit von St. Niendieck [Nien1998] behandeln Abbildungen zwischen bestimmten Neuronalen Netzen und Fuzzy-Entscheidungssystemen. Wie die Ausführungen zeigen, müssen von dem Neuronalen Netz eine Reihe von Vorgaben erfüllt sein, damit es in dieser Weise als Fuzzy-Entscheidungssystem interpretiert bzw. formal in ein solches überführt werden kann. (Vgl. hierzu [Tenh2000], S. 127ff.)

4.1 Das Niendieck-Tenhagen-Netz

Zur Veranschaulichung des Ausgangspunktes wird nachstehend der allgemeine Aufbau eines Niendieck-Tenhagen-Netzes kurz dargestellt. Die folgende Abbildung illustriert den Spezialfall einer eindimensionalen Netz-Ausgabe.

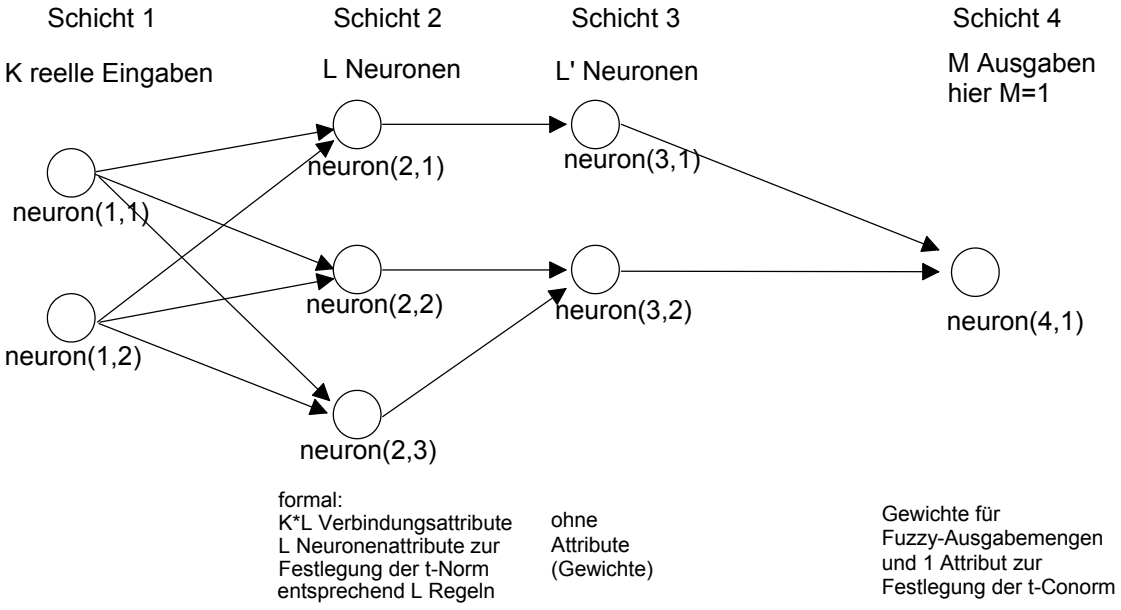


Abb. 4.1: Grobskizze eines Niendieck-Tenhagen-Netzes

Die erste Schicht dieses Netzaufbaus dient dazu, crispe Eingaben entgegenzunehmen. *K* entspricht der Dimension des Eingaberaumes.

In Schicht 2 werden über Fuzzy-Mengen als Gewichte die Prämissen der verschiedenen Regeln abgespeichert. Dabei vertritt jedes Neuron eine Regel. L steht somit für die Anzahl der Regeln.

Jedes Neuron aus Schicht 3 steht für eine Konklusionsmenge. Da maximal so viele Konklusionsmengen auftreten können wie Regeln, ist die Anzahl der Neuronen in der dritten Schicht kleiner oder gleich der der zweiten Schicht, $L' \leq L$.

Schließlich sorgt die vierte Schicht dafür, dass über einen Defuzzifizierungsmechanismus crispe Ausgaben (der Dimension M) generiert werden. (Vgl. Bemerkung 2.75 zu möglichen Defuzzifizierungsmethoden.)

Betrachten wir, wie konkret Regeln eines Fuzzy-Entscheidungssystems in einem solchen Netz repräsentiert werden. Die nachfolgende Abbildung zeigt nur einen Teil des obigen Netzes. Dieser Ausschnitt ist gerade der relevante Teil für eine bestimmte Regel.

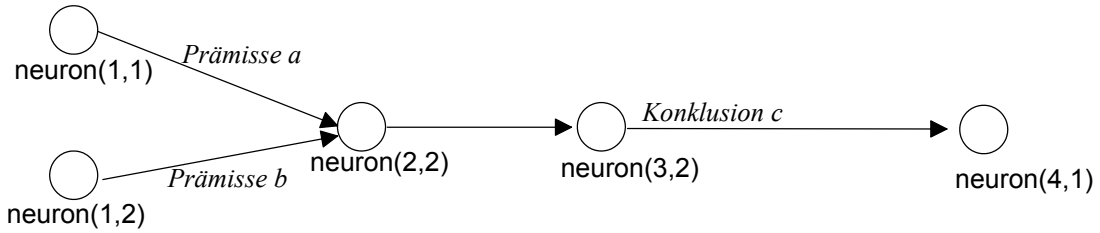


Abb. 4.2: Ausschnitt aus dem Netz von Abb. 4.1

Als Gewichte zwischen der Eingabe und der zweiten Schicht fungieren Fuzzy-Mengen (*a* und *b*), die in den Regeln als Prämissen auftreten. Über den AND-Operator verknüpft werden diese weitergeführt und führen in Schicht 3 zu dem Neuron, das die betreffende Konklusion *c* (ebenfalls eine Fuzzy-Menge) vertritt. Im Bild wird also die Regel

$$IF\ x_1\ IS\ a\ AND\ x_2\ IS\ b\ THEN\ y\ IS\ c$$

dargestellt.

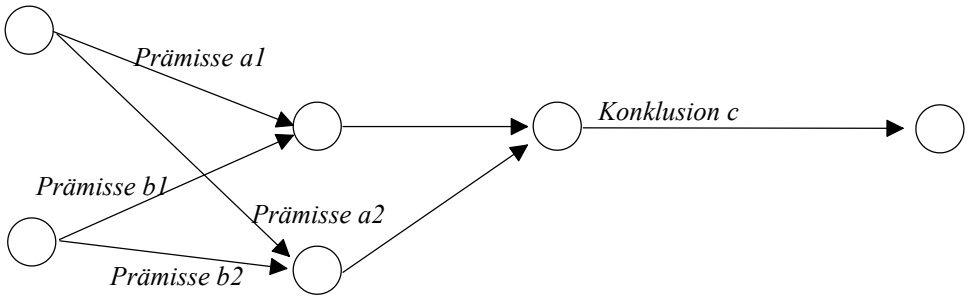


Abb. 4.3: Erweiterter Ausschnitt aus dem Netz von Abb. 4.1

Erweitern wir die Sicht auf das Netz, so erkennen wir, dass in dem Neuron der dritten Schicht, das für eine bestimmte Konklusion steht, alle Verbindungen eintreffen, die zu Regeln mit eben dieser Schlussfolgerung gehören. In Abb. 4.3 sind dies die beiden Regeln

$$IF\ x_1\ IS\ a1\ AND\ x_2\ IS\ b1\ THEN\ y\ IS\ c,$$

$$IF\ x_1\ IS\ a2\ AND\ x_2\ IS\ b2\ THEN\ y\ IS\ c.$$

4.2 Interpretation spezieller 3-Schicht-Netze als Fuzzy-Entscheidungssystem

Gegeben sei ein 3-Schicht-Netz mit (zunächst) einem Ausgabeneuron. Die Eingaben und Aktivierungsfunktionen seien wie bei Niendieck [Nien1998] und Tenhagen [Tenh2000] vorausgesetzt. Das bedeutet im Einzelnen (vgl. nachstehende Abbildung):

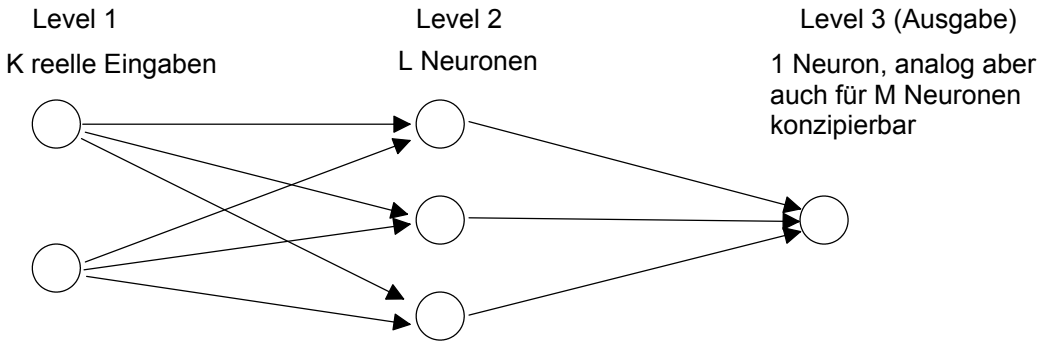


Abb. 4.4: Skizze des 3-Schicht-Netzes

Das Netz besitze K Neuronen in der Eingabeschicht, die reelle Eingaben x_1, ... , x_K entgegennehmen. Die als vollständig mit der ersten Schicht verbunden angenommene zweite Schicht bestehe aus L Neuronen, die wiederum alle mit dem Ausgabeneuron der 3.Schicht verbunden seien. Dieser Wert L korrespondiert später mit der Zahl der repräsentierten Regeln.

Die Annahme der vollständigen Verbundenheit ist keine ernsthafte Einschränkung, da wir hier nicht über das Trainieren eines solchen Netzes sprechen. Ggf. müssten im Folgenden lediglich Subindizierungen gewählt werden, was indes die Lesbarkeit unnötigerweise erschweren würde. Es gilt jedoch die Voraussetzung, dass jedes Neuron der zweiten Schicht mit mindestens einem Neuron der Eingabeschicht verbunden sein muss. Andernfalls müsste das betreffende isolierte Neuron der zweiten Schicht aus dem Netz entfernt werden.

Zur weiteren Notation: die schon verwendeten K und L seien natürliche Zahlen, ein klein geschriebener Index k bzw. l deutet nachfolgend an, dass er aus dem Bereich 1..K bzw. 1..L stammt bzw. über diesen Bereich rangiert. Mit w_{kl} wird das Fuzzy-Verbindungsgewicht zwischen dem k-ten Neuron N_{1k} der ersten und dem l-ten Neuron N_{2l} der zweiten Schicht bezeichnet[13]. Dieses Gewicht w_{kl} ist also eine trianguläre Fuzzy-Zahl x. Für diese wollen wir

[13] Wir folgen an dieser Stelle der Notation von Tenhagen und Niendieck um einen Abgleich mit den Originalquellen zu erleichtern, auch wenn wir an anderer Stelle die Reihenfolge der Indizierung der

die Schreibweise (x_L,x_M,x_R) verwenden, wobei x_L die linke Grenze, x_M den Modalwert und x_R die rechte Grenze bezeichnet.

Wichtig: für festes k sind die w_{kl}, $1 \le l \le L$, als *paarweise verschieden* vorauszusetzen! Vgl. hierzu die entsprechende Anmerkung in Tenhagen [Tenh2000], S. 129. Dass diese Einschränkung jedoch nicht wesentlich ist, werden wir in Satz 6.2 auf S. 108 zeigen.

Die Aktivierungs- und Ausgabefunktionen der Neuronen sehen wie folgt aus.

Schicht 1 (Eingabeschicht):
Die reellwertigen Eingaben x_1, ..., x_K werden unverändert durchgereicht. D.h. formal: $f_{akt,1,k} = f_{aus,1,k} = id$ (identische Abbildung).

Schicht 2:
Die Aktivierungsfunktionen $f_{akt,2,l}$ berechnen zu jeder Verbindung einen Zugehörigkeitswert anhand der Fuzzy-Gewichte w_{kl}, $1 \le k \le K$.

$$f_{akt,2,1}(x_1, ..., x_K) = (\mu_{w_{1l}}(x_1), ..., \mu_{w_{Kl}}(x_K)) \qquad \text{(Für K > 1 ist dies ein Vektor.)}$$

Mit der festgelegten t-Norm t_{AND} (konkret dem min-Operator) wird die Ausgabefunktion von Neuron N_{2l} ermittelt:

$$f_{aus,2,l}(z_1, ..., z_K) = t_{AND}(z_1, ..., z_K),$$

also

$$f_{aus,2,l} \circ f_{akt,2,l}(x_1, ..., x_K) = t_{AND}(\mu_{w_{1l}}(x_1), ..., \mu_{w_{Kl}}(x_K)).$$

Das 3-Schicht-Netz wird nun (mehr oder weniger formal) um eine weitere verborgene Schicht erweitert.

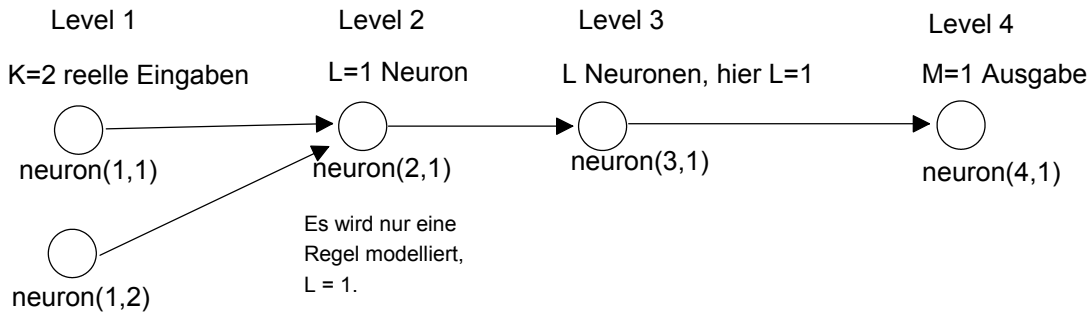


Abb. 4.5: Vereinfachte Skizze des erzeugten 4-Schicht-Netzes

(Neue) Schicht 3:
In Form einer 1:1-Erweiterung werden L Neuronen in der neuen verborgenen Schicht 3 eingefügt; jedes solche Neuron N_{3l} hat genau eine Verbindung von Schicht 2 (von Neuron N_{2l}) kommend und genau eine Verbindung zur Ausgabeschicht mit dem Fuzzy-Gewicht b_l.

Die Aktivierungsfunktion und die Ausgabefunktion sind jeweils die Identität, $f_{akt,3,l} = f_{aus,3,l} = id$.

Gewichte umdrehen und damit eine Vektorschreibweise erleichtern.

Schicht 4 (Ausgabeschicht):
Das Neuron N_{41} besitzt als Aktivierungsfunktion

$$f_{akt,4,1}(y_1, ..., y_L) = \max\{y_l \,\widetilde{\min}\, b_l | 1 \le l \le L\},$$

wobei die y_l alle aus [0,1] stammen und mit $\widetilde{\min}$ das Fuzzy-Minimum (vgl. [Tenh2000], S. 71) bezeichnet wird; anschaulich wird dabei im Ausdruck $y \,\widetilde{\min}\, b$ die Fuzzy-Zahl b auf der (reellwertigen) Höhe y abgeschnitten.

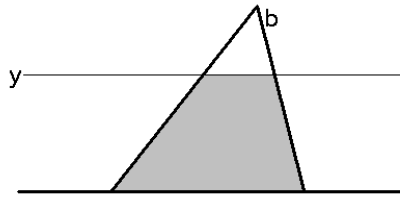


Abb. 4.6: Die graue Fläche stellt $y \,\widetilde{\min}\, b$ dar.

Die Funktion $f_{akt,4,l}$ liefert somit eine Fuzzy-Menge als Ergebnis.

Die zugehörige Ausgabefunktion ist dann eine geeignete Defuzzifizierungsfunktion *Defuzz()*, die zu einer Fuzzy-Menge M einen entsprechenden reellen Wert liefert: $f_{aus,4,l}(M) = Defuzz(M)$.

Numerisches Beispiel 4.1

Es seien K=L=2. Die Gewichte seien trianguläre Fuzzy-Zahlen in der Darstellung (linke Grenze, Modalwert, rechte Grenze); es seien $w_{1,1}$ = (-2,0,2), $w_{1,2}$ = (0,2,4), $w_{2,1}$ = (-1, 0, 1), $w_{2,2}$ = (0,2,5). Die Fuzzy-Gewichte zur Ausgabeschicht seien b_1 = (-4,-2,0) und b_2 = (0,2,4). Mit x_1=1 und x_2=3 ergibt sich daraus folgende Berechnung.

$$f_{aus,2,1} \circ f_{akt,2,1}(1,3) = t_{AND}(\mu_{(-2,0,2)}(1), \mu_{(-1,0,1)}(3)) = t_{AND}(\tfrac{1}{2}, 0) = \min(\tfrac{1}{2}, 0) = 0$$

$$f_{aus,2,2} \circ f_{akt,2,2}(1,3) = t_{AND}(\mu_{(0,2,4)}(1), \mu_{(0,2,5)}(3)) = t_{AND}(\tfrac{1}{2}, \tfrac{2}{3}) = \min(\tfrac{1}{2}, \tfrac{2}{3}) = \tfrac{1}{2}$$

Diese Werte, 0 und 1/2, werden durchgereicht bis zur Schicht 4.

$$f_{akt,4,1}(y_1, ..., y_L) = \max\{0 \,\widetilde{\min}\, b_1, \tfrac{1}{2} \,\widetilde{\min}\, b_2\} = \max\{0 \,\widetilde{\min}\, (-4-2, 0), \tfrac{1}{2} \,\widetilde{\min}\, (0,2,4)\} =: M$$

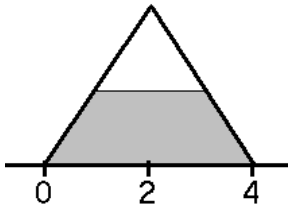


Abb. 4.7: Die Ausgabe $f_{aus,4,1}(M)$

Die endgültige Ausgabe ist schließlich $f_{aus,4,1}(M) = Defuzz(M)$, d.h. das Trapez (0,0), (1,1/2), (3,1/2), (4,0) [vgl. obiges Bild] wird in einen crispen Wert transformiert: die Defuzzifizierung dieser Menge liefert bei gängigen Verfahren wie der Mittelwert- oder der Schwerpunktmethode den Wert 2.

4.3 Äquivalenz von Neuronalen Netzen und Fuzzy-Entscheidungssystemen

Es sei ein Netz gemäß dem vorigen Abschnitt bzw. gemäß der hierzu funktional äquivalenten 4-Schicht-Erweiterung (siehe S. 84) gegeben.

Weiterhin sei ein Fuzzy-Entscheidungssystem (FES) wie folgt gegeben.

Es besitze die folgenden L Regeln:

$$(R_1) \quad \text{IF } x_1 \text{ IS } w_{1,1} \text{ AND } ... \text{ AND } x_K \text{ IS } w_{K,1} \text{ THEN y IS } b_1$$
$$...$$
$$(R_L) \quad \text{IF } x_1 \text{ IS } w_{1,L} \text{ AND } ... \text{ AND } x_K \text{ IS } w_{K,L} \text{ THEN y IS } b_L$$

Wiederum wird hier "o.B.d.A." angenommen, dass alle Regelprämissen stets sämtliche Eingaben verwenden. Selbstverständlich ist es auch hier andernfalls nur eine Angelegenheit einer komplizierteren Indizierung, den allgemeinen Fall darzustellen - jedoch auf Kosten der Lesbarkeit der Formeln!

Die t-Norm t_{AND} und die t-Conorm t_{OR} seien wie im Neuronalen Netz, hier wird *min* bzw. *max* verwendet. (Zur Thematik der t-Normen und t-Conormen sei auf Def. 2.15 verwiesen.)

Ebenso gehen wir davon aus, dass das Neuronale Netz und das Fuzzy-Expertensystem mit derselben Defuzzifizierungsmethode arbeiten.

Damit formulieren wir nachstehend ein zentrales Resultat von Tenhagen [Tenh2000], die Äquivalenz des o.g. Neuronalen Netzes und des Fuzzy-Entscheidungssystems.

Satz 4.2 (Äquivalenz von FES und Neuronalem Netz)

Das oben beschriebene Fuzzy-Entscheidungssystem und das Neuronale Netz sind äquivalent, d.h. sie liefern zu einer beliebigen Eingabe $(x_1,...,x_K)$ denselben Ergebnis- bzw. Ausgabewert.

Beweis:

Sei $(x_1,...,x_K)$ aus $\mathbb{R}^K$ gegeben. Dann sind im FES die Prämissen zu betrachten. Deren Erfüllungsgrade sind $e_l = t_{AND}(\mu_{w_{1l}}(x_1), ..., \mu_{w_{Kl}}(x_K)) = \min(\mu_{w_{1l}}(x_1), ..., \mu_{w_{Kl}}(x_K))$, $1 \le l \le L$.

Bei der Berechnung der Konklusionen (vgl. (6.6) bei Tenhagen [Tenh2000], S. 133) ergibt sich die folgende Fuzzy-Menge M:

$$M = t_{OR}(e_1 \,\widetilde{\min}\, b_1, ..., e_L \,\widetilde{\min}\, b_L) = \max(e_1 \,\widetilde{\min}\, b_1, ..., e_L \,\widetilde{\min}\, b_L)$$

Diese Menge wird schließlich mittels *Defuzz()* defuzzifiziert.

Offensichtlich liefern das Neuronale Netz sowie das Fuzzy-Entscheidungssystem denselben Output (für t_{AND}=min und t_{OR}=max):

$$Defuzz(\underset{1\le l\le L}{t_{OR}} (b_l \,\widetilde{\min}\, \underset{1\le k\le K}{t_{AND}} (\mu_{w_{k,l}}(x_k)))) = Defuzz(\max_{1\le l\le L} (b_l \,\widetilde{\min}\, \min_{1\le k\le K} (\mu_{w_{k,l}}(x_k))))$$

□

4.4 Charakteristika beim Ansatz von Tenhagen

Die von Tenhagen betrachteten Netze weisen folgende charakteristischen Eigenschaften auf (vgl. [Tenh2000], S. 128):

1. Jedes Neuron der zweiten Schicht empfängt mindestens eine Verbindung von Schicht 1; diese Verbindung ist mit einem Fuzzy-Gewicht versehen, mit dem eine Fuzzy-Zugehörigkeitsfunktion ermittelt wird.
2. Jedes Neuron der zweiten Schicht besitzt genau eine Verbindung in die dritte Schicht.
3. Jedes Neuron der dritten Schicht ist mit mindestens einem Neuron der zweiten Schicht verbunden.
4. Jedes Neuron der dritten Schicht besitzt genau eine Verbindung in die vierte (Ausgabe-) Schicht. Das heißt, zu jedem Neuron der vierten Schicht kann die Gruppe der hiermit verbundenen Neuronen der dritten Schicht betrachtet werden; diese Verbindungen besitzen ebenfalls wieder Fuzzy-Gewichte, mittels derer Fuzzy-Zugehörigkeitsfunktionen berechnet werden.

Insbesondere ist festzuhalten, dass die Berechnungsvorschriften bei den Verbindungen bereits sehr stark an den Aufbau der Fuzzy-Entscheidungssysteme angelehnt sind.

4.5 Sugeno-Controller und RBF-Netze

Zum Abschluss dieses Kapitels soll noch ein anderes Beispiel kurz präsentiert werden, das die funktionale Äquivalenz eines Fuzzy-Controllers mit einem speziellen Neuronalen Netz aufzeigt. Hierbei handelt es sich um einen Controller nach Sugeno und Takagi (vgl. 2.76) und ein Radial Basis Function-Netz (RBF-Netz) gemäß Definition 1.19.

Von Jang [Jang1993] und Hunt [Hunt1996] wurde gezeigt, dass ein (3-schichtiges) RBF-Netz äquivalent zu einem bestimmten Fuzzy-Controller ist, wobei die Anzahl der Regeln des Controllers der Anzahl der Neuronen der versteckten Schicht des RBF-Netzes entspricht.

Betrachten wir zunächst das RBF-Netz mit K Neuronen in der Eingabeschicht, L Neuronen in der Hidden Schicht und einem Ausgabeneuron. Mit den Notationen aus Definition 1.19 (vgl. S. 30) wird die normierte Ausgabe des Netzes zum Eingabevektor $\vec{x}$ beschrieben durch

$$out_{norm}(\vec{x}) = \frac{\sum_{1 \le i \le L} W_i \cdot \psi(\|\vec{x} - \vec{c_i}\|)}{\sum_{1 \le i \le L} \psi(\|\vec{x} - \vec{c_i}\|)},$$

wobei $W := (W_1, ..., W_L)$ der Vektor der Gewichte zwischen den Neuronen der verborgenen Schicht und dem Ausgabeneuron und die $\vec{c_i}$ die Zentren der L radialen Basisfunktionen $\vec{x} \to \psi(\|\vec{x} - \vec{c_i}\|)$ sind.

Nehmen wir zum Vergleich dazu einen Sugeno-Controller (vgl. 2.76 auf S. 72) mit r Regeln R_1, ..., R_r und zugehörigen Erfüllungsgraden $\alpha_1, ..., \alpha_r$, dann ist die Ausgabe des Controllers gegeben durch

$$y = \frac{\sum_{1 \le i \le r} a_i f_i(x_1,...,x_n)}{\sum_{1 \le i \le r} a_i}.$$

Gibt es im Sugeno-Controller nun gerade $r = L$ Regeln und haben diese die spezielle Form

$$\text{IF } x_1 \text{ IS } a_1 \text{ AND } ... \text{ AND } x_n \text{ IS } a_n \text{ THEN } y = W_i,$$

das heißt liefern die Fuzzy-Regeln konstante (crispe) Ergebniswerte, dann ist der Output des Controllers zu schreiben als

$$y = \frac{\sum_{1 \le i \le L} a_i W_i}{\sum_{1 \le i \le L} a_i}.$$

Werden schließlich die radialen Basisfunktionen $\vec{x} \to \psi(\left\| \vec{x} - \vec{c_i} \right\|)$ als Zugehörigkeitsfunktionen der Fuzzy-Mengen (bzw. Zugehörigkeitsgrade a_i der Regeln) im Controller eingesetzt, dann resultiert als Ausgabewert des Controllers gerade die normierte Ausgabe des zuvor diskutierten RBF-Netzes:

$$y = \frac{\sum_{1 \le i \le L} W_i \cdot \psi(\left\| \vec{x} - \vec{c_i} \right\|)}{\sum_{1 \le i \le L} \psi(\left\| \vec{x} - \vec{c_i} \right\|)} = out_{norm}(\vec{x}).$$

Das bedeutet: ein Sugeno-Controller mit den hier beschriebenen Eigenschaften kann semantisch äquivalent durch ein entsprechendes RBF-Netz dargestellt werden. Dies bietet den Vorteil, dass Lernverfahren für RBF-Netze genutzt werden können und auf diese Weise letztlich der Sugeno-Controller optimiert werden kann.

Bemerkung 4.3

Gegenüber der originalen Formulierung von Jang [Jang1993] und Hunt [Hunt1996] wurde hier zur besseren Verständlichkeit ein etwas vereinfachtes Szenario gewählt. [Küfe2001] liefert eine zusammenfassende Darstellung des Sachverhaltes in einer weitergehend parametrisierten Form z.B. in Bezug auf die konkret eingesetzten radialen Basisfunktionen. Wir werden uns jedoch in den folgenden Kapiteln schwerpunktmäßig mit möglichst generellen Feedforward-Netzen beschäftigen.

II. Teil Regelextraktion aus Neuronalen Netzen

5 Grundlagen der Regelextraktion

In diesem zweiten Teil der vorliegenden Arbeit geht es um den Zusammenhang zwischen Neuronalen Netzen und einer für Menschen verständlichen Wissens- oder Regelbasis sowie um die Darstellung der Möglichkeiten, wie konkrete Regeln (bzw. Regelbasen) aus Neuronalen Netzen gewonnen werden können.

Zunächst werden in diesem Kapitel grundlegende Begriffe der Regelextraktion erläutert und ein Überblick über bekannte Ansätze der Regelextraktion aus Neuronalen Netzen gegeben[14], bevor anschließend in den darauffolgenden Kapiteln spezielle Verfahren zur Vereinfachung von Neuronalen Netzen und Regelextraktionsalgorithmen präsentiert werden.

Dabei wollen wir nicht die Optimierung von Fuzzy-Controllern oder das Lernen von Neuronalen Netzen betrachten; hierzu sei beispielsweise auf [Lipp1999] verwiesen. Wir gehen im Folgenden von bereits trainierten Neuronalen Feedforward-Netzen[15] aus und wollen untersuchen, unter welchen Bedingungen an das jeweilige Neuronale Netz Regeln daraus extrahiert werden können.

5.1 Lernen, Wissen und die Notwendigkeit von Regelextraktion

Lernt ein Mensch, so wird vieles in Form von Regelzusammenhängen abgespeichert; auf diese Weise wird ein bestimmtes Wissen konserviert. Dabei können diese Regeln direkt von einem (wie auch immer gearteten) Lehrer vermittelt werden, oftmals werden aber auch hinreichend viele selbst erlebte oder beobachtete Beispiele zu eigenen Regeln verdichtet. Umgekehrt besteht beim Menschen aber meist das Bedürfnis, eine Regel durch ein oder mehrere Beispiele angewendet zu sehen. Gelegentlich ist auch ein Gegenbeispiel für das Verständnis eines Menschen hilfreich, insbesondere dann, wenn die Prämisse der Regel "sehr oft" zutrifft. Ein Gegenbeispiel nutzt die äquivalente, negierte Formulierung einer Regel (im Rahmen der klassischen Logik) aus. Statt "IF A THEN B" wird "IF NOT B THEN NOT A" beispielhaft verdeutlicht. Das ist die die in 2.62 vorgestellte Widerlegungsregel ("Modus tollens"), vgl. S.62.

Neben dem Kennen von Regeln ist für den Menschen auch ein Vorrat an Beispielen sowie eine gewisse Übung erforderlich. Im Zusammenhang mit Neuronalen Netzen kann ein

---

[14] Eine sehr kompakte Einführung zu diesem Thema findet sich bei Stoffers [Stof2000].

[15] Wir fokussieren hier ausschließlich auf Feedforward-Netze. Andere Netz-Architekturen werden in der Literatur ebenfalls behandelt, z.B. behandeln Hammer, Rechtien, Strickert und Villmann die Regelextraktion von selbstorganisierenden Netzen (vgl. [Hamm2002]), Andrews und Geva befassen sich in [Andr2002] mit lokalen Cluster-Netzen. In [Sun2001] wird die "Extraktion von Wissen" während des Lernprozesses eines Neuronalen Netzes behandelt.

entsprechender Beispielvorrat durch Dateneingabe in das Netz generiert werden; damit stellt sich dann auch Übung, Erfahrung mit dem betreffenden Netz ein.

Was für das tiefere menschliche Verständnis des Netzes schließlich fehlt, sind Regeln. Aus diesem Grund ist es wünschenswert, zu einem Neuronalen Netz eine - natürlich möglichst überschaubare - Anzahl von Regeln zu kennen, die das Verhalten des Netzes beschreiben.

Für den praktischen Einsatz von Neuronalen Netzen ist es indes nicht nur ein “nice to have”, nicht nur ein schöner Wunsch, sondern eine sehr wichtige Aufgabe, das Verhalten des Netzes in menschlich verständlicher Form auszudrücken, speziell durch Angabe von Regeln. Ohne eine solche Interpretation bleibt das Neuronale Netz eine “black box” und kann in vielen technischen und ökonomischen, insbesondere jedoch in sicherheitsempfindlichen Anwendungen nicht eingesetzt werden.

Auch der Einsatz Neuronaler Netze z.B. für Prognosen bei Banken, Versicherungen oder Krankenkassen erfordert solche verständlichen Regeln, denn ein Kunde möchte speziell für einen ablehnenden Bescheid eine Begründung bekommen und diese nachvollziehen können. Nicht zuletzt wird auch der Kundenberater verstehen wollen, warum “seine Software” ggf. eine ablehnende Entscheidung vorschlägt.

Darüber hinaus ist speziell auf dem Gebiet der Software-Entwicklung darauf hinzuweisen, dass das Testen und Validieren von Programmen zur Emulation von Neuronalen Netzen teilweise unmöglich ist, da i.a. das Verhalten des “echten”, durch die Software zu modellierenden Netzes auf beliebigen Daten nicht vor dem ersten praktischen Durchlauf bekannt oder zumindest nicht generell vorhersagbar ist.

Selbstredend können in komplexeren Situationen Testfälle (abgesehen von trivialen Wertekombinationen) nicht mehr von Hand durchgerechnet werden, um die korrekte Funktionsweise des Software-Netzes zu überprüfen! Auch hier kann eine Regel-Darstellung helfen, eine Software-Implementierung eines Neuronalen Netzes zu verifizieren oder wenigstens plausibel erscheinen zu lassen.

Prinzipiell ist es zwar vorstellbar, dass man eine Software-Implementierung durch Zuhilfenahme einer zweiten, unabhängig erstellten Software-Lösung zur selben Aufgabenstellung validieren möchte. Einerseits hat eine solche numerische Plausibilitätskontrolle aber keinen formalen Beweischarakter (und ersetzt auch keine umfangreichen Testverfahren), andererseits würde sich bei solch einer Vorgehensweise der Entwicklungsaufwand buchstäblich verdoppeln. Erneut wird deutlich, dass ein durch Regeln dargestelltes bzw. präzisiertes Wissen dem menschlichen Software-Tester das Überprüfen einer Software-Lösung für Neuronale Netze gravierend erleichtern kann.

5.2 Regel-Typen

Regeln sind generell Wenn-Dann-Aussagen bzw. -Aussageformen. Wir werden im Folgenden, insbesondere bei formalen Notationen, das englische IF-THEN verwenden. Dabei können die Regeln jedoch verschiedene Konzepte verfolgen.

Das prinzipielle Muster lautet “IF Prämisse (erfüllt) THEN Folgerung (gilt)”. Generell lässt man bei der Prämisse Konjunktionen (Und- bzw. AND-Verknüpfungen) zu. Disjunktionen

(Oder- bzw. OR-Verknüpfungen) in der Prämisse können dagegen auch durch separate Regeln ersetzt werden.

Spielraum gibt es bei der Gestaltung von Prämissen und Folgerungen. Diese können zunächst einmal in diskreten, qualitativen Klassifizierungsmustern bestehen:

IF x_1 gehört zu A_1 AND x_2 gehört zu A_2 THEN y gehört zu C.

Statt "gehört zu" wird auch kürzer "IS" geschrieben. Selbstverständlich können hier und im Folgenden auch mehr als zwei konjunktive Prämissenteile miteinander verknüpft werden.

Häufig werden Daten jedoch nicht (nur) qualitativ klassifiziert, sondern quantitativ verarbeitet. Hierzu passt dann der Regeltyp, der reellwertige Intervalle einsetzt:

IF $x_1 \in [a_1, b_1]$ AND $x_2 \in [a_2, b_2]$ THEN $y \in [c, d]$,

wobei es natürlich nicht darauf ankommt, ob hier geschlossene, offene oder halboffene Intervalle verwendet werden. In Zusammenhang mit digitaler Datenverarbeitung ist es ohnehin vergleichsweise selten, dass zwei Gleitkommaspeicherplätze den exakt identischen Wert annehmen.

Statt Klassifizierungsmustern oder -mengen oder reellen Intervallen können bei der Regelformulierung auch Fuzzy-Mengen verwendet werden; dann erhält man entsprechend Fuzzy-Regeln, wie wir sie im Zusammenhang mit Fuzzy-Controllern (vgl. Def. 2.74) bereits kennengelernt haben.

Es gibt indes noch weitere Möglichkeiten der Regelformulierung. Towell und Shavlik [Towe1993] verwenden eine "M-of-N-Formulierung":

IF (M von N Prämissen treffen zu) THEN Konklusion (gilt).

Gelegentlich mag es in konkreten Fällen hilfreich sein, eine Negation nutzen zu können. Dies führt zu Regeln der Bauart

IF x_1 IS A_1 AND NOT (x_2 IS A_2) THEN y gehört zu C.

Schließlich sei angemerkt, dass es in verschiedenen Situationen hilfreich sein kann, auch das aus der Algorithmen-Darstellung bekannte "ELSE" bei Regelformulierungen zuzulassen. Wir werden dies gleich in Bemerkung 5.4 zur Illustration aufgreifen.

Bemerkung 5.1

Nimmt unabhängig von x eine Variable y immer den Wert 0 an, so formulieren wir die Regel "IF true THEN y=0" natürlich kurz als "y=0". Formal ist dies aber als Wenn-Dann-Aussage darstellbar. Auch ist es entsprechend möglich, dass bei einer Regel, deren Prämissen über einem K-dimensionalen Eingaberaum formuliert sind, nicht tatsächlich alle K Größen $x_1, ..., x_K$ eine Rolle spielen. In einem solchen Fall wird auf Prämissen-Teile der Form "... AND x_j beliebig" verzichtet, die Regel also nur mit den tatsächlich relevanten Eingangsgrößen dargestellt.

Bemerkung 5.2

Es kann auch darüber nachgedacht werden, bewusst unvollständige Regelbasen zu formulieren, d.h. Mengen von Regeln, die nicht den gesamten Eingaberaum abdecken. Dies kann dann Sinn machen, wenn die Regelbasis hauptsächlich für das menschliche Verständnis generiert werden und nicht für eine exakte Darstellung des Netzes dienen soll. Denn zum einen kann ein Mensch *zu viele* Regeln ohnehin nicht erfassen und überblicken, zum zweiten denkt sich ein menschlicher Experte ein stetiges Zwischenverhalten zwischen zwei durch Regeln erfassten Eingabebereichen meist implizit hinzu. Ist das tatsächliche Verhalten des Neuronalen Netzes in diesem Bereich wirklich stetig (und vielleicht auch monoton), dann würden weitere Regeln die Verständlichkeit eventuell mindern.

Wir werden in unserem weiteren Vorgehen daher bewusst die Anzahl der zu erzeugenden Regeln in den konkreten Verfahren als Steuerungsparameter definieren.

5.3 Arten der Regelextraktion und Qualitätskriterien

Grundsätzlich unterscheidet man zwei verschiedene Vorgehensweisen der Regelextraktion, *pädagogische* und *analytische* Algorithmen.

Bei pädagogischen Verfahren werden anhand von Beispieldaten Regeln formuliert. Mit anderen Worten: es wird das Verhalten des Neuronalen Netzes im Sinne einer Black Box für “viele” Beispieldaten protokolliert und in Form von mehr oder weniger einfachen Regeln zusammengefasst. Hier wird somit versucht, eine möglichst passende Abbildungsvorschrift zwischen der (Netz-)Eingabe und der Ausgabe zu finden.

Demgegenüber betrachten analytische Algorithmen die einzelnen Neuronen eines Netzes und beschreiben deren Verhalten. Damit dieser Ansatz für ein nichttriviales Netz erfolgreich sein kann, müssen die Ausgaben der Neuronen in den verborgenen Schichten und der Ausgabeschicht binäre Ja-Nein-Entscheidungen darstellen. Somit kann jedes dieser Neuronen als Repräsentant einer entsprechenden Regel interpretiert werden. Die einzelnen Regeln der Neuronen können dann für das gesamte Netz zu einer Regelbasis zusammengefasst werden. Vgl. hierzu [Andr1995], S. 376.

Zhou et al sprechen in [Zhou2000] von *architecture-analysis-based* und *function-analysis-based* Vorgehensweisen bei der Regelextraktion, meinen damit im Wesentlichen jedoch die beiden oben beschriebenen Ansätze.

Als dritte Form haben sich sogenannte *eklektische* Verfahren etabliert, bei denen die beiden zuvor genannten Prinzipien gemischt werden. So kann etwa eine (analytische) Regelextraktion für jede Neuronenschicht durchgeführt werden, bei der jede Schicht insgesamt als Black Box aufgefasst wird.

Von einfachen Beispielen abgesehen liefern analytische Verfahren somit einen besseren Einblick in die innere Struktur und Arbeitsweise eines Neuronalen Netzes als die pädagogischen Algorithmen. Allerdings kann ein analytisches Verfahren hierbei zu einer derart umfangreichen Regelbasis führen, dass dies wiederum die menschliche Verständlichkeit der Regelmenge erheblich einschränkt.

Andrews, Diederich und Tickle haben in [Andr1995] (S. 377) eine mehrstufige Taxonomie aufgestellt, anhand derer Regelextraktionsverfahren eingeordnet werden können. Ein wichtiges Qualitätskriterium für Regeln bzw. Regelbasen, das darin aufgeführt wird, ist die sogenannte *Verständlichkeit*: die Anzahl der Regeln in einer Regelbasis und die Anzahl der Prämissen in den einzelnen Regeln darf nicht "zu hoch" sein.

Bei der Extraktion von Regeln aus einem Neuronalen Netz wird häufig eine *höchstmögliche Allgemeingültigkeit* angestrebt, das bedeutet, dass alle Teilprämissen einer Regel auch wirklich erforderlich sind. Die Regel würde also ungültig werden, sollte eine Teilbedingung weggelassen werden.

In der Praxis kann dies zum Abschluss eines Regelextraktionsverfahrens dadurch sichergestellt werden, dass eventuelle Regeln mit derselben Folgerung (Konklusion) daraufhin überprüft werden, ob ihre Prämissen evtl. miteinander verschmolzen oder im Sinne einer solchen Allgemeingültigkeit neu gebündelt werden können. Allerdings kann der Wunsch nach höchstmöglicher Allgemeingültigkeit dazu führen, dass sich überlappende Prämissenbereiche in verschiedenen Regeln verwendet werden müssen, was die Verständlichkeit der Regelbasis nicht zwingend verbessern würde. Hierzu das nachstehende einfache Beispiel zur Illustration.

Beispiel 5.3

Wir betrachten das einfache AND-Beispiel von zwei Eingangsgrößen x_1, x_2. Nur, wenn beide den Wert 1 besitzen, ist der Output $y = 1$, andernfalls ist $y = 0$. Die "elementaren" Regeln hierzu sind die folgenden.

R1: IF $x_1 = 1$ AND $x_2 = 1$ THEN $y = 1$,

R2: IF $x_1 = 0$ AND $x_2 = 1$ THEN $y = 0$,

R3: IF $x_1 = 1$ AND $x_2 = 0$ THEN $y = 0$,

R4: IF $x_1 = 0$ AND $x_2 = 0$ THEN $y = 0$.

Man erkennt unmittelbar, dass die Regeln R2, R3 und R4 nicht "höchstmöglich allgemeingültig" sind, denn die Teilprämisse $x_1 = 0$ bzw. $x_2 = 0$ genügt jeweils für die Konklusion $y = 0$. Bei den resultierenden drei Regeln R1, R2', R3' überlappen sich jedoch die Prämissen von R2' und R3', so dass in nichttrivialen und weniger übersichtlichen Fällen in der Praxis über die Widerspruchsfreiheit der Regelbasis nachgedacht werden müsste.

R1: IF $x_1 = 1$ AND $x_2 = 1$ THEN $y = 1$,

R2': IF $x_1 = 0$ THEN $y = 0$,

R3': IF $x_2 = 0$ THEN $y = 0$.

Bemerkung 5.4 (Die ELSE-Alternative)

Wie man leicht sieht, kann die Situation im vorherigen Beispiel durch Hinzunahme der Möglichkeit einer "ELSE"-Formulierung sehr einfach dargestellt werden durch eine einzige Regel:

R1': IF $x_1 = 1$ AND $x_2 = 1$ THEN $y = 1$ ELSE $y = 0$.

Äquivalent zu “ELSE” wäre auch eine Priorisierung, also die Einführung einer Reihenfolge für die Regeln einer Regelbasis. Damit ließe sich diese Situation darstellen durch zwei formale Regeln, vgl. Bemerkung 5.1:

R1: IF $x_1 = 1$ AND $x_2 = 1$ THEN $y = 1$,

R2’’: $y = 0$.

Als Pseudocode einer an C angelehnten Sprache wäre dies etwa in Form einer Funktion *rules()* implementierbar, die den y-Wert berechnet und zurückgibt.

```
int rules(int x1, int x2)     // Implementierung einer priorisierten
{                             // Regel-Liste
   if (x1==1 and x2==1)
   {
      return 1;
   }
   return 0;
}
```

Bemerkung 5.5 (Ursache und Wirkung)

Es sei abschließend noch angemerkt, dass eine Menge von Regeln zwar einen Zusammenhang zwischen Input- und Outputgrößen formuliert, dass dieser Zusammenhang aber nicht zwangsläufig der von Ursache und Wirkung sein muss!

Die Regel “IF $x = a$ THEN $y = b$” besagt zwar, dass die Größe y den Wert b besitzt, falls $x = a$ ist, das heißt aber nicht, dass diese Prämisse auch die Ursache darstellt. Ebensogut kann es verborgene Ursachen geben, die dazu führen, dass gleichzeitig $x = a$ und $y = b$ gelten.

5.4 Exakte und approximative Regelextraktion

Es gibt Situationen, bei denen ein Neuronales Netz so speziell beschaffen ist, dass eine Darstellung seiner Semantik durch eine Regelbasis sogar im exakten, präzisen Sinne möglich ist. Meistens jedoch wird durch eine endliche Regelbasis lediglich eine Annäherung an das Ausgabeverhalten eines Neuronalen Netzes möglich sein. Für eine formale Klarheit wollen wir diese zwei Möglichkeiten der Bedeutung des Begriffs Regelextraktion begrifflich unterscheiden.

Definition 5.6 (Exakte Regelextraktion)

Als *exakte Regelextraktion* wird in unserem Kontext der Mechanismus bezeichnet, zu einem Neuronalen Netz (ggf. unter gewissen zusätzlichen Voraussetzungen) funktionsäquivalente Regeln zu formulieren, d.h. eine Regelbasis aufzustellen, die zu beliebigen (zulässigen) Eingabedaten zu exakt denselben Ausgabedaten führt wie das Neuronale Netz.

Beispiel 5.7

1. In Satz 4.2 wurde die Äquivalenz eines Neuronalen Netzes mit einem Fuzzy-Entscheidungssystem (FES) für ein eigens dafür konstruiertes Netz gezeigt. Die Regeln des FES repräsentieren das Netz in exakter Weise.
2. In Abschnitt 4.5 werden ein Sugeno-Controller und ein semantisch äquivalentes RBF-Netz konstruiert. Auch hier verkörpern die Regeln im Controller in exakter Weise das Neuronale Netz.

In den meisten Situationen der Praxis ist es indes nicht oder nicht einfach möglich, ein Fuzzy-Expertensystem bzw. eine Regelbasis anzugeben, die ein bestimmtes Neuronales Netz semantisch äquivalent darstellen. Neben der oben genannten Definition 5.6 ist daher für praktische Belange eine approximative Regelextraktion möglich und nötig. Hier muss das Verhalten des gegebenen Neuronalen Netzes durch die Regeln nicht identisch repräsentiert werden, sondern nur numerisch angenähert innerhalb vorgegebener Toleranzen.

Definition 5.8 (Approximative Regelextraktion)

Wir sprechen von *approximativer* oder *numerischer Regelextraktion*, wenn zu einem vorgegebenen Neuronalen Netz, einer festgelegten Definitions- bzw. Eingabemenge, einem Fehlermaß F sowie einer Toleranz Δ eine Regelbasis aufgestellt wird, so dass die mit dem Fehlermaß F bewerteten Abweichungen der Ausgaben von Regelbasis und Neuronalem Netz innerhalb der genannten Toleranz bleiben.

Mathematisch präziser formuliert: Seien N die Netzausgabefunktion, R die Ausgabefunktion der Regelbasis, D die Definitionsmenge, F ein Fehlermaß und $\Delta > 0$ eine einzuhaltende Toleranz. Dann ist R eine (F,Δ)-*approximative Regelbasis zum Netz* N, wenn gilt: $F(N,R,D) < \Delta$.

Beispiel 5.9

Es seien D eine endliche Teilmenge des $\mathbb{R}^K$, N ein Neuronales Netz und R eine Regelbasis (bzw. jeweils deren Ausgabefunktionen mit Werten in $\mathbb{R}^M$), F sei definiert durch $F(N,R,D) := \frac{1}{|D|} \sum_{x \in D} |N(x) - R(x)|^2$, wobei mit |D| die Anzahl der Elemente der Menge D bezeichnet werde und der Term $|N(x) - R(x)|$ für die euklidische Norm in $\mathbb{R}^M$ stehe.

Dann ist F das sog. *mittlere quadratische Fehlermaß* (*mean square error*, *MSE*) und R eine Approximation an das Netz N in diesem Sinne.

Im weiteren Gang der Arbeit werden wir Regelextraktion hauptsächlich in dieser approximativen Bedeutung verfolgen.

Beispiel 5.10

Ersetzt man im vorigen Beispiel F durch das Maximum,

$$F(N,R,D) := \max \{ |N(x) - R(x)| : x \in D \},$$

so erhält man ein strengeres, punktweise operierendes Fehlerkriterium. Wir werden jedoch noch sehen, dass bis auf triviale Spezialfälle dieses Fehlermaß zu rigide ist für eine

praktikable Regelextraktion. Außerdem beziehen sich die universellen Approximationsaussagen für Neuronale Netze ebenfalls nicht auf Maximum-Normen, sondern auf mittlere quadratische Abweichungen oder ähnliches. Vgl. hierzu Bemerkung 1.35 (S. 39) sowie 6.29 (S. 128).

5.5 Ein Überblick über Ansätze der Regelextraktion

In der Vergangenheit sind bereits zahlreiche Ansätze einer Regelextraktion aus Neuronalen Netzen verfolgt worden. Teilweise werden dabei fertig trainierte Netze betrachtet, zum Teil geht die Regelextraktion Hand in Hand mit Lernverfahren für das Neuronale Netz. Einige Arbeiten (vgl. etwa Zhao und De Souza [Zhao2001]) setzen Genetische Algorithmen zur Optimierung bzw. Komplexitätsreduktion einer zunächst sehr großen Regelbasis ein[16].

Hayashi, Buckley und Czogala [Haya1992] liefern Verfahren zur wechselseitigen Approximation von Fuzzy-Entscheidungssystemen und Neuronalen Netzen; dabei wird zu einer gegebenen Trainingsmenge eines Netzes eine Regelbasis konstruiert, deren Qualität damit von der Güte der verwendeten Trainingsdaten abhängt.

Auch die Arbeit von Yi und Oh [Yi1992] behandelt einen Ansatz, aus Neuronalen Netzen bereits während des Trainings Fuzzy-Regeln zu generieren. Erwähnenswert ist hierbei eine Form der Gewichtung der erzeugten Regeln durch eine sogenannte "*relative Wichtigkeit*" ("*relative importance*").

Ein Zusammenhang zwischen bestimmten Typen von Neuronalen Netzen und Regeldarstellungen (etwa in Form von Regelbasen innerhalb entsprechender Fuzzy-Controller) wird in zahlreichen Arbeiten formuliert und präzisiert. So wurde beispielsweise von Jang [Jang1993] und Hunt [Hunt1996] gezeigt (sh. hierzu Abschnitt 4.5), dass ein (3-schichtiges) Radial Basis Function (RBF)-Netz äquivalent zu einem bestimmten Fuzzy-Controller ist, wobei die Anzahl der Regeln des Controllers der Anzahl der Neuronen der versteckten Schicht des RBF-Netzes entsprechen. Zu einem Fuzzy-Controller, der mehrere Eingangsgrößen und/oder eine größere Anzahl Regeln besitzt, wird das entsprechende RBF-Netz somit allerdings schnell sehr groß. Umgekehrt bedeutet dieser Sachverhalt, dass zu einem vorgegebenen 3-Schicht-RBF-Netz mit L Neuronen in der verborgenen Schicht auch eine äquivalente Basis mit L Regeln aufgestellt werden kann. Darauf aufbauend stellen Wu und Tam [Wu1999] ein relativ einfaches Modell zur Extraktion eines Fuzzy-Entscheidungssystems aus einem RBF-Netz vor.

Pop, Hayward und Diederich formulieren in [PopH1994] den RULENEG-Algorithmus zur Regelextraktion aus Neuronalen Netzen, bei dem ganz oder teilweise negierte Prämissen zum Einsatz kommen.

Andrews, Diederich und Tickle [Andr1995] geben einen auch heute noch sehr lesenswerten Überblick über verschiedene Regelextraktionsverfahren für bereits trainierte neuronale Netze. Eine vergleichende Bewertung einer ganzen Reihe von Regelextraktionsalgorithmen führt Neumann [Neum1998] durch. In einem weiteren Überblicksbeitrag legen Tickle,

[16] Zum Themenkomplex der Genetischen Algorithmen sei stellvertretend auf das Buch von Weicker [Weic2002] verwiesen.

Andrews, Golea und Diederich [Tick1998] dar, dass einer der künftigen Forschungsschwerpunkte bei der Regelextraktion aus Neuronalen Netzen auf reellwertigen Ausgaben liegen wird. Als ein Beispiel hierfür formulieren Saito und Nakano [Sait2002] einen Algorithmus zur Regelextraktion, der gleichzeitig numerische und nominale, d.h. klassifizierende Variablen berücksichtigt. Die Konklusion wird dabei als eine funktionale Abhängigkeit von der Eingabe ausgedrückt, d.h. in Abhängigkeit des Eingabevektors $\vec{x}$ hat die Regel-Ausgabe die Form $y = f(\vec{x})$.

Ishibuchi, Nii und Tanaka [Ishi1999] präsentieren einen Ansatz, positive und negative linguistische Fuzzy-Regeln für trainierte Netze im Rahmen von Musterklassifikationsproblemen zu extrahieren. Der Begriff "negativ" bezeichnet im Kontext der hier betrachteten Regeln die Negation der Prämisse oder der Konklusion; beispielsweise ist "IF x NOT groß THEN ..." eine solche "negative" Formulierung. Interessant sind speziell die negierten Formulierungsmöglichkeiten in den Regeln, jedoch ist die Komplexität der Berechnung für etwas umfangreichere Netze sehr hoch.

Kasabov und Woodford betrachten in ihrer Arbeit [Kasa1999] Möglichkeiten zum Einfügen und Extrahieren von Regeln bei sich verändernden Neuronalen Netzen; hierbei wird eine spezielle 5-Schicht-Architektur aufgebaut.

Ebenfalls auf einer ganz dezidierten Architektur, einer 4-Schicht-Architektur mit Treppenfunktionen als Aktivierung in der ersten verborgenen Schicht, arbeitet das "DIMLP"-Modell ("*discretized interpretable multi layer perceptron*") von Bologna [Bolo2000]. Dort werden auf der Grundlage konkreter (Beispiel-)Daten Regeln extrahiert, die sich an Hyperebenen des Eingaberaumes orientieren. Ähnlich arbeitet auch der Ansatz von Moraga, den wir in Abschnitt 5.5.2 vorgestellen werden.

Tsukimoto [Tsuk2000] stellt einen analytischen Algorithmus zur Regelextraktion bei trainierten Neuronalen Netzen vor, der monotone Ausgabefunktionen (z.B. sigmoide Funktionen) für die Neuronen voraussetzt. Diese Arbeit wird in einer Seminarausarbeitung von Marc Rogge überzeugend dargestellt, vgl. [Rogg2001].
Das präsentierte Verfahren weist im Gegensatz zu vielen anderen Verfahren polynomiale Komplexität auf. Im Wesentlichen wird die Verhaltensweise der einzelnen Neuronen durch boolesche Funktionen approximiert, aus denen dann die Regeln des Netzes formuliert werden. Gleichzeitig weisen Snyders und Omlin in [Snyd2001] nach, dass mit polynomialem Aufwand keine *exakte* Regelextraktion möglich ist.

Castellano und Fanelli [Cast2000] konstruieren während des Lernens eine Fuzzy-Regelbasis für dreischichtige Netze; hierbei entspricht die Anzahl der Neuronen gerade der Anzahl der extrahierten Regeln. Wir werden dagegen im Folgenden die Anzahl der zu extrahierenden Regeln nicht an die Anzahl der Neuronen in der verborgenen Schicht koppeln, da dies eine erhebliche Einschränkung bedeuten würde.

Setiono und Leow [Seti2000] stellen im Kontext von Klassifikationsproblemen mit "FERNN" einen Algorithmus zur "schnellen Extraktion von Regeln" vor, der auf der zentralen Idee beruht, weniger relevante Verbindungsvektoren von der Eingabe- zur Hidden-Schicht zu entfernen (bzw. die betreffenden Vektorkomponenten auf 0 zu setzen). Die hierbei extrahierten Regeln sind dabei in disjunktiver Normalform oder als sog. "M-of-N"-Regeln formuliert. Eine "M-von-N"-Regel hat eine Prämisse der Bauart "IF M von N Bedingungen sind erfüllt THEN ...".

Quek und Zhou [Quek2001] stellen zwei Algorithmen zur Extraktion linguistischer Regeln aus speziellen Fuzzy-Neuronalen Netzen mit (wiederum genau) fünf Schichten vor.

Zhou et al [Zhou2000] sowie Jiang, Zhou und Chen [Jian2002] stellen weitere Verfahren im Kontext von Kategorisierung vor (u.a. unter Einsatz statistischer Methoden), bei denen die generierten Regeln in einer Prioritätenliste verwaltet werden, mit anderen Worten: in der hierdurch vorgegebenen Reihenfolge wird eine Eingabe der Regelbasis vorgelegt, und bei der ersten Regel, deren Prämisse diese Eingabe beinhaltet, wird die resultierende Ausgabe ermittelt.
Formal entspricht dies einer IF-THEN-ELSE-Verkettung von Regeln:

IF Prämisse1 THEN Konsequenz1 ELSE IF Prämisse2 THEN Konsequenz2 ELSE IF ...

Die Bewertung von Regeln spielt auch in der Arbeit von Guillaume und Charnomordic [Guil2001] eine Rolle. Hier werden (auf dem praktischen Anwendungsgebiet der französischen Käse-Produktion) Fuzzy-Controller trainiert und durch Genetische Algorithmen optimiert. Interessant in dem Zusammenhang mit unserer Arbeit ist die Regelbewertung, die die Autoren auf der Grundlage einer umfangreichen Datenbasis vornehmen. Anhand dieser Datenbasis werden die einzelnen Regeln bewertet und ggf. - zum Beispiel bei zu geringem Einfluss auf das Gesamtergebnis - aus der Regelbasis entfernt.

In den folgenden beiden Abschnitten werden ergänzend noch zwei weitere Ansätze vorgestellt.

5.5.1 Validity Interval Analysis (VIA)

Als einer der prominenten Ansätze für ein pädagogisches Regelextraktionsverfahren soll hier die sog. *Validity Interval Analysis* (*VIA*) von Sebastian Thrun vorgestellt werden (vgl. [Thru1995] sowie die Seminarausarbeitung [Plög2000]).

Bei der VI-Analyse handelt es sich um ein recht allgemein angelegtes Verfahren, das zunächst eine Menge von Regeln aus einem fast beliebigen Neuronalen Feedforward-Netz generiert, anschließend wird die Konsistenz der Regeln zu dem Netz überprüft. An das Netz wird im Kern nur die Anforderung gestellt, dass die Aktivierungsfunktionen der Neuronen stetig und monoton sind, d.h. der in der vorliegenden Arbeit betrachtete Typ der sigmoiden Funktion wird davon mit erfasst.

Wir wollen anhand eines konkreten Beispiels die Vorgehensweise des Verfahrens erläutern. Hierfür betrachten wir das nachfolgend skizzierte Netz. Wir verzichten gegenüber allgemeineren Darstellungsmöglichkeiten auf die Verwendung von Schwellenwerten bei den einzelnen Neuronen und gehen davon aus, dass die Eingabeneuronen wie gewohnt die Identität als Aktivierungsfunktion besitzen und die anderen Neuronen das Skalarprodukt der gewichteten Input-Größen. Als Ausgabefunktion verwenden die Neuronen der verborgenen Schicht die Sigmoide.

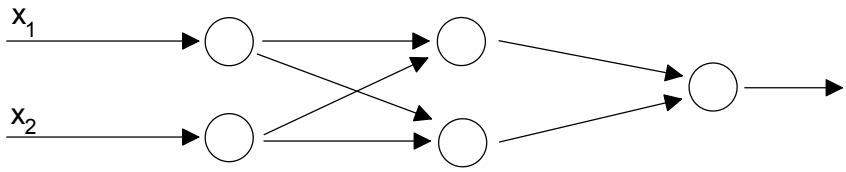


Abb. 5.1: Darstellung eines 2-2-1-Netzes

Die grundlegende Idee der "validity intervals" ist es, Intervalle zu finden, die dem Aktivierungsbereich einer Einheit, also einem einzelnen Neuron oder einer Neuronenschicht, entsprechen.

Betrachten wir ein bestimmtes einzelnes Neuron (in der verborgenen Schicht). Dann werde mit P die Menge der Vorgängerneuronen bezeichnet. In untenstehender Skizze ist somit P die Menge der Neuronen in der Eingabeschicht.

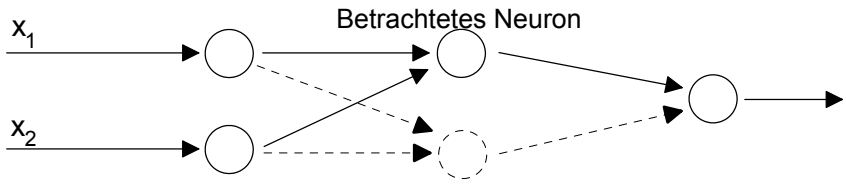


Abb. 5.2: Betrachtung eines einzelnen Neurons der verborgenen Schicht

Nummerieren wir für einen Moment die Neuronen des Netzes einfach durch, so dass die Indizes 1 und 2 für die beiden Eingabeneuronen, 3 und 4 für die Neuronen der verborgenen Schicht und 5 für das Ausgabeneuron stehen.

Es sei für die folgenden Ausführungen net_j die Aktivierung bei Neuron j; für ein Eingabeneuron ist dies also gerade der von der Außenwelt eingehende Wert x_j. Für ein anderes Neuron j ergibt sich $net_j = \sum_{k \in P} w_{jk} x_k$.

Ziel der VI-Analyse ist es, geeignete Intervalle für die jeweiligen Aktivierungen zu finden. Hierzu soll ein System von linearen Ungleichungen aufgestellt werden, auf das anschließend Methoden der linearen Optimierung angewendet werden können.

Da die Sigmoide nichtlinear ist, projiziert das Verfahren die entsprechenden Intervalle zurück: an Stelle eines Intervalls $[a_j, b_j]$ zu Ergebniswerten der Form $sigm(\sum_{k \in P} w_{jk} x_k)$ wird das korrespondierende Eingabeintervall $[sigm^{-1}(a_j), sigm^{-1}(b_j)]$ betrachtet. Hierbei ist $sigm^{-1}$ die Umkehrfunktion der Sigmoiden, vgl. Bem. 1.15.

Lässt man für die ursprünglichen Grenzen die Werte 0 und 1 zu, so kann es sich bei $[sigm^{-1}(a_j), sigm^{-1}(b_j)]$ um ein uneigentliches Intervall handeln, das die Grenzen $-\infty$ oder $+\infty$ (oder beide) beinhalten kann.

Damit wird nun das folgende System von Ungleichungen aufgestellt, wobei wir hier der kompakteren Darstellung wegen mit P die Menge der entsprechenden Indizes notieren wollen; betrachten wir exemplarisch Neuron 3, so ist also $P = \{1, 2\}$.

$$\forall k \in P \ : \ a_k \le x_k \le b_k \ ,$$

$$a_3' \le \sum_{k \in P} w_{1k} x_k \le b_3' \quad \text{mit} \quad a_3' := sigm^{-1}(a_3),\ b_3' := sigm^{-1}(b_3).$$

Dieses Ungleichungssystem kann äquivalent auch in Intervalldarstellung notiert werden.

$$\forall k \in P\ :\ x_k \in [a_k, b_k],$$
$$\sum_{k \in P} w_{1k} x_k \in [a_3', b_3'].$$

Für Neuron 4 (bzw. allgemein die weiteren Neuronen) werden in entsprechender Weise Ungleichungen (resp. Intervall-Aussagen) formuliert.

Um die o.e. Intervalle zu verfeinern bzw. zu verbessern werden nun in der sogenannten *Vorwärtsphase* Kandidaten für neue a_j- und b_j-Werte betrachtet: Zu dem stellvertretend betrachteten Neuron 3 werden die beiden Ausdrücke $\tilde{a}_3 := sigm(\min_{x_k} \{\sum_{k \in P} w_{1k} x_k\})$ und $\tilde{b}_3 := sigm(\max_{x_k} \{\sum_{k \in P} w_{1k} x_k\})$ definiert. Dabei handelt es sich um die Funktionswerte der kleinsten bzw. größten möglichen Aktivierungen[17] an dem Neuron, wenn die x_k jeweils über ihre Bereiche $[a_k, b_k]$ rangieren. Stellt sich heraus, dass $\tilde{a}_3 > a_3$ gilt, dann ist eine bessere (d.h. größere) untere Schranke an Stelle von a_3 gefunden und dieser Wert wird durch $\tilde{a}_3$ ersetzt:$a_3 := \tilde{a}_3$. Entsprechend verfährt man, wenn $\tilde{b}_3 < b_3$ gilt: $b_3 := \tilde{b}_3$. Wie zuvor wird nun $a_3' := sigm^{-1}(a_3),\ b_3' := sigm^{-1}(b_3)$ gesetzt.

Auch für die Neuronen der Eingabeschicht kann anschließend in der sogenannten *Rückwärtsphase* eventuell eine Verbesserung gefunden werden.
Hier werden die Kandidaten $\tilde{a}_k := \min_{x_k} x_k$ und $\tilde{b}_k := \max_{x_k} x_k$ betrachtet, wobei die x_k über die zulässigen Werte rangieren, die sich durch die Nebenbedingungen der Vorwärtsphase ergeben haben.

Dies wollen wir an einem numerischen Beispiel verdeutlichen.

Betrachten wir das nachfolgende Netz und darin wieder ein Neuron der Hidden Schicht speziell.

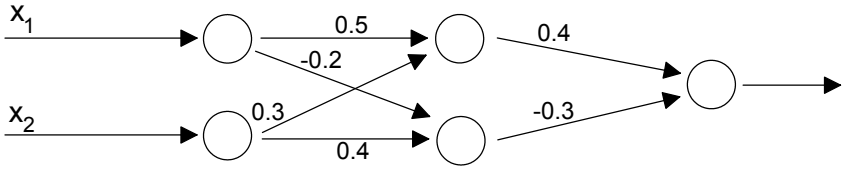


Abb. 5.3: Numerisches Beispiel

Es seien zu Beginn $[a_1, b_1] := [a_2, b_2] := [a_3, b_3] := [0, 1]$.

Damit lauten die initialen Ungleichungen:

$$0 \le x_1 \le 1,\ 0 \le x_2 \le 1\ ,$$
$$-\infty \le 0.5x_1 + 0.3x_2 \le +\infty \quad \text{wegen} \quad -\infty = sigm^{-1}(0),\ +\infty = sigm^{-1}(1).$$

Dabei ist die letzte Ungleichungskette natürlich nur die formalistische Anwendung auf die konkreten Werte 0 und 1.

---

[17] Mathematisch präzise sind *min* und *max* nur für kompakte Intervalle korrekte Formulierungen. Wir wollen hier jedoch nur das Verfahren exemplarisch darstellen und verzichten daher auf die formale Strenge.

Dann betrachten wir in der Vorwärtsphase:

$$\tilde{a}_3 := sigm(\min_{x_k}\{\sum_{k\in P} w_{1k}x_k\}) = sigm(\min_{x_k}\{0.5x_1 + 0.3x_2\}) = sigm(0) = 0.50$$

und

$$\tilde{b}_3 := sigm(\max_{x_k}\{\sum_{k\in P} w_{1k}x_k\}) = sigm(\max_{x_k}\{0.5x_1 + 0.3x_2\}) = sigm(0.8) = 0.70$$

Diese Werte ergeben sich wegen $0 \le x_1 \le 1$ und $0 \le x_2 \le 1$. Der zweite Wert ist auf zwei Nachkommastellen gerundet wiedergegeben.

In diesem Fall gilt gleichermaßen $\tilde{a}_3 = 0.50 > a_3 = 0$ und $\tilde{b}_3 = 0.70 < b_3 = 1$. Damit wird als neues Intervall $[a_3, b_3] := [0.50, 0.70]$ definiert.

Die Berechnungen für das zweite Neuron der verborgenen Schicht verlaufen analog und werden hier im Detail weggelassen. Man erhält dort ebenfalls auf zwei Nachkommastellen gerundet $[a_4, b_4] := [0.45, 0.60]$.

In der Rückwärtsphase wird nun geprüft, ob das Intervall der zulässigen bzw. gültigen Werte für die Eingangsgrößen verkleinert, d.h. optimiert, werden kann.

Aufgrund der o.g. Werte liegt die Aktivierung $net_3 = 0.5x_1 + 0.3x_2$ für Neuron 3 zwischen 0.50 und 0.70. Löst man $net_3 = 0.5x_1 + 0.3x_2$ auf nach x_1, so ergibt sich: $x_1 = \frac{net_3 - 0.3x_2}{0.5}$ bzw. $x_2 = \frac{net_3 - 0.5x_1}{0.3}$.

Der Kandidat für ein neues a_1 ist damit $\tilde{a}_1 := \min \frac{net_3 - 0.3x_2}{0.5} = \frac{0.5 - 0.3\cdot 1}{0.5} = 0.4$. Dieser Wert ist größer als der vorherige Wert $a_1 = 0$, d.h. es wird nun redefiniert a_1:=0.4.

Analog ergibt sich als Kandidat für ein neues b_1 $\tilde{b}_1 := \max \frac{net_3 - 0.3x_2}{0.5} = \frac{0.7 - 0.3\cdot 0}{0.5} = 1.4$. Dieser Wert ist jedoch nicht kleiner als der bisherige Wert b_1, so dass es hier bei $b_1 = 1$ bleibt.

Insgesamt ist damit das neue Intervall $[a_1, b_1] := [0.4, 1]$.

Entsprechende Rechnung für a_2 und b_2:
$\tilde{a}_2 = \min \frac{net_3 - 0.5x_1}{0.3} = \frac{0.5 - 0.5\cdot 1}{0.3} = 0$, $\tilde{b}_2 = \max \frac{net_3 - 0.5x_1}{0.3} = \frac{0.7 - 0.5\cdot 0}{0.3} = \frac{7}{3} > 1$.

In beiden Fällen ergibt sich jedoch keine Verkleinerung des Intervalls, es bleibt hier somit bei den vorherigen Werten $[a_2, b_2] := [0, 1]$.

Führt man die entsprechenden Rechnungen durch in Hinblick auf Neuron 4, dessen Aktivierungsbereich $[a_4, b_4] = [0.45, 0.60]$ ist, dann erhält man eine (weitere) Verfeinerung des ersten Intervalls $[a_1, b_1]$ zu $[a_1, b_1] := [0.3, 1]$. Die Schnittmenge beider Verfeinerungen ist $[0.4, 1] \cap [0.3, 1] = [0.3, 1] =: [a_1, b_1]$.

Damit kann nun die Ausgabe des Netzes betrachtet werden. Diese lässt sich schreiben als

$$out := 0.4 \cdot neuron_3 - 0.3 \cdot neuron_4,$$

wobei nach obigen Ungleichungen für die von den Neuronen 3 und 4 weitergegebenen Werte $neuron_3 \in [0.50, 0.70]$ und $neuron_4 \in [0.45, 0.60]$ gelten, wenn $x_1 \in [0.3, 1]$ und $x_2 \in [0, 1]$ erfüllt sind.

Damit kann die Netzausgabe abgeschätzt werden:

$$0.02 = 0.4 \cdot 0.50 - 0.3 \cdot 0.60 \le out \le 0.4 \cdot 0.70 - 0.3 \cdot 0.45 = 0.145.$$

Damit ergibt sich auf natürliche Weise die folgende Regel:

$$IF\ x_1 \in [0.3, 1]\ AND\ x_2 \in [0, 1]\ THEN\ y \in [0.02, 0.145].$$

Bei einem angenommenen Eingaberaum $[0, 1] \times [0, 1]$ kann diese Regel in die kompaktere Darstellung

$$IF\ x_1 \geq 0.3\ \ THEN\ y \in [0.02, 0.145]$$

verkürzt werden.

Wir haben an diesem numerischen Beispiel konkret gesehen, wie die VI-Analyse Regeln durch Betrachtung der zugrundeliegenden Intervalle bzw. Überprüfung der betreffenden Konsistenzbedingungen optimieren kann.

Es muss noch ergänzt werden, wie das Verfahren zu (ersten) Regeln gelangt. Dabei bieten sich zwei prinzipielle Vorgehensweisen an: Bottom-Up, vom Spezifischen zum Allgemeinen, sowie umgekehrt, Top-Down, vom Allgemeinen zum Spezifischen verfeinernd.

Im Bottom-Up-Ansatz wird von konkret vorliegenden Trainingsdaten ausgegangen. Jeder solche Datensatz kann als Regel mit degenerierten Intervallen aufgefasst werden, da die präzisen Werte vorliegen. Sinngemäß: $IF\ \overrightarrow{x} = \overrightarrow{x_0}\ THEN\ y \in C$.

Die Verallgemeinerung kann dadurch erreicht werden, dass die Prämissen-Intervalle jeweils um kleine Beträge vergrößert werden können. Die so entstandenen Intervalle werden sodann mit der oben beschriebenen VI-Methode geprüft, d.h. es wird zu verifizieren versucht, ob auch für alle Eingaben innerhalb der neuen Intervalle die Konklusion "y gehört zur Klasse C". Gelingt dies, so ist eine allgemeinere Regel gefunden.

Der von Thrun beschriebene Top-Down-Ansatz geht umgekehrt von sehr allgemeinen Regeln aus, etwa "alles gehört zu C". Durch Zerlegen der Prämissen in disjunkte Intervalle gelangt man zu spezifischeren Regeln bzw. Regelkandidaten; kann ein solcher Kandidat wie zuvor beschrieben verifiziert werden, so ist eine konkrete spezifischere Regel gefunden.

5.5.2 Der Ansatz von Moraga

Claudio Moraga betrachtet Fuzzy-IF-THEN-Regeln, bei denen die Teilprämissen nicht mittels einer t-Norm (z.B. dem min-Operator) verknüpft werden wie etwa beim ANFIS-Modell (vgl. [Jang1993a] sowie [Borg2003], S. 233ff). Stattdessen verwendet Moraga sogenannte *kompensierende Operatoren* ("*compensating aggregation operators*"), mit denen die Prämissen von Fuzzy-Regeln "weicher" verknüpft werden. Wir werden auf diesen Punkt nach einigen mathematischen Ausführungen zurückkommen.

Für die nachfolgenden Ausführungen sei speziell auf [Temm1999], [Mora2000], [Mora2000a], [Mora2000b], [Mora2001] und [Mora2002] verwiesen.

Ein weiterer Aspekt des Ansatzes von Moraga ist, die große Anzahl generierter Regeln gegenüber ANFIS zu reduzieren. Hat ein ANFIS-Modell fünf Eingangsgrößen mit jeweils drei linguistischen Variablen zu ihrer Beschreibung, so erzeugt ANFIS $3^5 = 243$ Regeln, womit die Verständlichkeit der Regelbasis höchst eingeschränkt ist.

Zur Erläuterung eines solchen kompensierenden Operators benötigen wir die Definition der *Symmetrischen Summation*. Dazu vorab ein Resultat (vgl. [Temm1999]).

Bemerkung 5.11

Es sei f die sigmoide Funktion. Dann gilt: $f(x+y) = \frac{1}{1+e^{-x-y}} = \frac{1}{1+e^{-x}e^{-y}}$.

Wegen

$$e^{-x} = 1 + e^{-x} - 1 = \frac{\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}}{\frac{1}{1+e^{-x}}} = \frac{1-\frac{1}{1+e^{-x}}}{\frac{1}{1+e^{-x}}} = \frac{1-\frac{1}{1+e^{-x}}}{\frac{1}{1+e^{-x}}} = \frac{1-f(x)}{f(x)}$$

folgt

$$f(x+y) = \frac{1}{1+(\frac{1-f(x)}{f(x)})(\frac{1-f(y)}{f(y)})} = \frac{f(x)f(y)}{(1-f(x))(1-f(y))+f(x)f(y)}.$$

Hierauf basiert die folgende Definition.

Definition 5.12 (Symmetrische Summation)

Die Abbildung $\oplus :]0,1[\times]0,1[\to]0,1[$, $a \oplus b := \frac{a \cdot b}{(1-a)\cdot(1-b)+a \cdot b}$, heißt *symmetrische Summation*.

Bemerkung 5.13

Die Struktur $(]0,1[,\oplus)$ bildet eine abelsche Gruppe mit dem neutralen Element $\frac{1}{2}$. Das inverse Element zu $f(x)$ ist $f(-x) = 1 - f(x)$ bzw. zu y entsprechend 1-y, denn $y \oplus (1-y) = \frac{1}{2}$.

Für Werte in $]0, \frac{1}{2}[$ verhält sich diese Operation wie eine t-Norm t, für Werte größer $\frac{1}{2}$ wie die dazu duale t-Conorm s. Es ist speziell $s = t$, denn nach Definition der dualen t-Conorm gilt (vgl. Bemerkung 2.16):

$$s(a,b) = 1 - t(1-a, 1-b) = 1 - \frac{(1-a)(1-b)}{ab+(1-a)(1-b)} = \frac{(1-a)(1-b)+ab-(1-a)(1-b)}{(1-a)(1-b)+ab} = \frac{ab}{(1-a)(1-b)+ab} = t(a,b).$$

Exemplarisch betrachten wir die Monotonie. Seien hierzu $a, b \in]0,1[$ und $0 < h < 1-b$. Dann prüfen wir, ob die Ungleichung $a \oplus b \le a \oplus (b+h)$ erfüllt ist.

Es gilt: $a \oplus b \le a \oplus (b+h) \Leftrightarrow \frac{ab}{(1-a)(1-b)+ab} \le \frac{a(b+h)}{(1-a)(1-b-h)+a(b+h)}$

$\Leftrightarrow ab[(1-a)(1-b-h)+a(b+h)] \le a(b+h)[(1-a)(1-b)+ab]$

$\Leftrightarrow ab(1-a)(1-b) - ab(1-a)h + aba(b+h) \le ab(1-a)(1-b) + ah(1-a)(1-b) + a(b+h)ab$

$\Leftrightarrow -ab(1-a)h \le ah(1-a)(1-b).$

Die letzte Ungleichung ist stets erfüllt, da die linke Seite kleiner oder gleich 0 und die rechte Seite nichtnegativ ist.

Bemerkung 5.14

Die Abbildung $\psi : (\mathbb{R}, +) \rightarrow (]0,1[, \oplus)$ mit $\psi(x) := f(x)$ ist ein Isomorphismus.

Beweis:

Dass ψ bijektiv ist, folgt aus Bem. 1.15, da es sich um die Sigmoide handelt.

Bleibt zu zeigen, dass die algebraische Struktur der Addition + auf die Struktur $\oplus$ transformiert wird.

Zunächst ist $\psi(0) = f(0) = \frac{1}{2}$. Für $x, y \in \mathbb{R}$ ist nach Bemerkung 5.11 und Definition 5.12

$$\psi(x+y) = \frac{f(x)f(y)}{(1-f(x))(1-f(y))+f(x)f(y)} = f(x) \oplus f(y).$$

❑

Bemerkung 5.15

Das obige Resultat lässt sich verallgemeinern für Funktionen f des Typs $x \rightarrow \frac{1}{1+k^{-x}}$, $k > 1$ *reell* . Dieser Funktionstyp wird in der Literatur auch *verallgemeinerte logistische Funktion* genannt. Für $k = e$ erhalten wir gerade die gewohnte sigmoide Funktion.

Bemerkung 5.16

Sieht man sich den Kurvenverlauf einer t-Norm wie z.B. dem min-Operator an und vergleicht diesen mit dem Graph des kompensierenden Operators “symmetrische Summation” $\oplus$, dann fällt auf, dass letzterer Werte über $\frac{1}{2}$ stärker zum Ergebnis 1 schiebt als der Minimum-Operator. Umgekehrt werden Werte unter $\frac{1}{2}$ stärker zum Ergebniswert 0 gedrückt. Dies wird in den beiden nachstehenden Abbildungen illustriert.

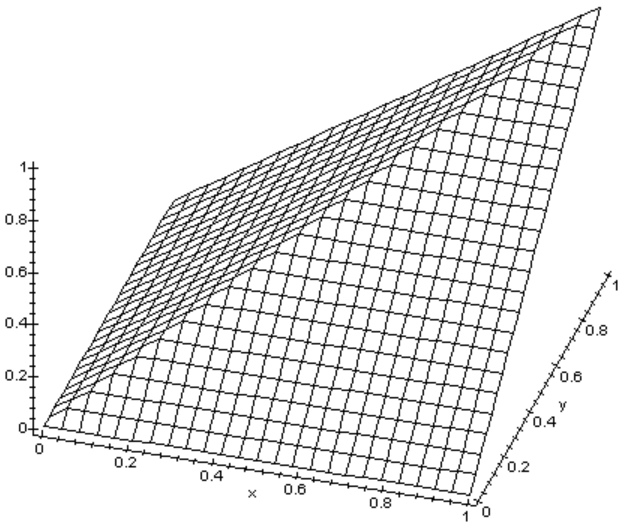


Abb. 5.4: Funktionsplot der t-Norm min-Operator

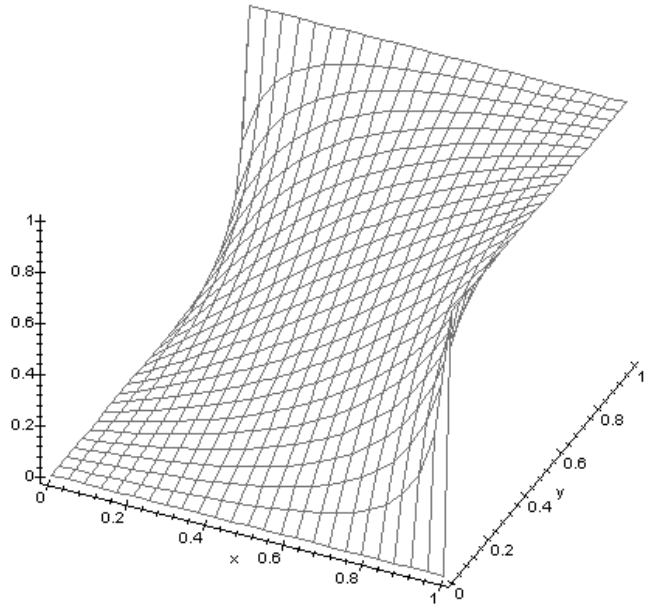


Abb. 5.5: Funktionsplot der symmetrischen Summation

Moraga nennt in [Mora2001] ein kleines Beispiel. Eine Fuzzy-Regel über einen Kinofilm kann beispielsweise linguistisch formulieren[18]: "WENN (A) der Film einen Preis gewonnen hat UND (B) der Eintrittspreis akzeptabel ist, DANN kaufe eine Eintrittskarte".

Sollte nun die (Teil-)Prämisse B := "Eintrittspreis ist akzeptabel" einen Zugehörigkeitswert von 0,51 besitzen, dann ist es für die Konsequenz unerheblich, ob der Film viele oder höherwertige Preise gewonnen hat ($\mu_{\mathrm{Pr}eis\_gewonnen}(x_1) = 0.99$) oder nur einen Provinzpreis ergattern konnte ($\mu_{\mathrm{Pr}eis\_gewonnen}(x_1) = 0.52$). In beiden Fällen hätte die Konsequenz (Kaufen einer Eintrittskarte) bei Zugrundelegen des min-Operators maximal den Grad 0,51. Moraga führt aus, dass dies in vielen Fällen nicht dem menschlichen Handeln entspricht, bei dem die Bereitschaft zu Kompromissen vorhanden ist und ein Film mit einer Reihe von Oskars durchaus einen höheren Eintrittspreis rechtfertigen kann.

Nehmen wir die o.g. Zahlen und berechnen das Resultat anhand des kompensierenden Operators "symmetrische Summation": $0.99 \oplus 0.51 = \frac{0.99 \cdot 0.51}{(1-0.99)\cdot(1-0.51)+0.99\cdot 0.51} \approx 0.99$. Das bedeutet: die Konsequenz, eine Eintrittskarte zu kaufen, wird aufgrund des sehr hohen Zugehörigkeitsgrades der ersten Teilprämisse deutlich verstärkt.

Im anderen Fall, $\mu_{\mathrm{Pr}eis\_gewonnen}(x_1) = 0.52$, fällt die Entscheidung wesentlich weniger deutlich aus: $0.52 \oplus 0.51 = \frac{0.52 \cdot 0.51}{(1-0.52)\cdot(1-0.51)+0.52\cdot 0.51} \approx 0.53$.

Daher verwendet Moraga bei seinem Ansatz den symmetrischen Summationsoperator für die nachfolgend skizzierte Regelextraktion.

---

[18] In dieser Situation verwenden wir ausnahmsweise das deutsche "WENN-DANN", damit der Satz verständlich bleibt.

Bemerkung 5.17 (Moraga-Regelextraktion)

Wir gehen von einem dreischichtigen Feedforward-Netz mit K Eingabeneuronen und L Neuronen in der Hidden Schicht aus. Dann ist die Ausgabe eines Neurons der verborgenen Schicht (mit *f* als der sigmoiden Funktion)

$$h(x_1, ..., x_K) = f(\Sigma w_i x_i) = \bigoplus f(w_i x_i),$$

wobei mit $w_i, 1 \le i \le L$, die Gewichte zu dem betreffenden Neuron bezeichnet werden. Der letzte Ausdruck steht für die symmetrische Summation wie zuvor definiert, lediglich angewendet auf L statt zwei Argumente.

Formal kann der Ausdruck $f(w_i x_i)$ auch als Funktion $f_{(i)}$ aufgefasst werden mit der Zuordnungsvorschrift $f_{(i)}(x) := f(w_i x)$. Da f und damit auch $f_{(i)}$ eine Funktion mit Werten in [0,1] (bzw. in]0,1[) ist, kann $f_{(i)}$ auch als Fuzzy-Menge interpretiert werden. Sei A_i der linguistische Ausdruck zur Fuzzy-Menge $f_{(i)}$. Dann generiert Moragas Ansatz aus

$$h(x_1, ..., x_K) = f(\Sigma w_i x_i) = \bigoplus f(w_i x_i) = \bigoplus f_{(i)}(x_i)$$

zu dem betreffenden Neuron genau eine Fuzzy-Regel der Form

$$IF\ x_1\ is\ in\ A_1\ AND\ ...\ AND\ x_L\ is\ in\ A_L\ THEN\ h = \bigoplus f_{(i)}(x_i).$$

Ähnlich wie beim Controller-Modell von Takagi und Sugeno wird hier eine Konklusion formuliert, nur nicht in linearer Form, sondern in Gestalt der symmetrischen Summation. Ebenso erhält man auf diese Weise zu den *L* Neuronen der verborgenen Schicht *L* Regeln, die als additives Fuzzy-System wirken.

Je nach Vorzeichen des Gewichtswertes w_i sind die Zugehörigkeitsgrade der Prämissen A_i für $x_i > 0$ oder $x_i < 0$ größer als 0,5. Insofern sind die effektiv wirkenden Regeln bei Moragas Ansatz faktisch sehr eingeschränkt in ihrem Aufbau, da in jeder Eingabedimension stets ein Halbraum für die Prämisse verwendet wird. Daneben ist festzuhalten, dass die Anzahl der zu extrahierenden Regeln mit der Anzahl der Hidden Neuronen korrespondiert, also zu einem konkreten Netz fest vorgegeben ist. Dies ist eine Einschränkung, der wir bei unseren weiteren Betrachtungen nicht unterliegen wollen.

6 Approximation und Reduktion

Dieses Kapitel stellt Formalismen vor, mit denen zu klassischen Feedforward-Netzen mit der sigmoiden (logistischen) Ausgabefunktion Regeln generiert werden können. Dabei finden an verschiedenen Stellen Approximationsschritte statt, d.h. unter Inkaufnahme der daraus resultierenden Näherungen kann das Verfahren zu vereinfachten Situationen führen.

Um anschließend aus dem Netz eine möglichst einfache und menschlich gut verständliche Regelbasis extrahieren zu können, wird versucht, die Anzahl der Neuronen in der verborgenen Schicht zu reduzieren.

Es zeigt sich, dass es hierfür stellenweise sehr hilfreich ist, zunächst die sigmoide Ausgabefunktion durch eine Funktion einfacherer Bauart, die sogenannte *Rampenfunktion*, zu ersetzen bzw. zu approximieren. Dieser Schritt ermöglicht es im übrigen auch, den Rechenaufwand des Neuronalen Netzes zu senken. Als nächstes kann dann die Anzahl der Neuronen in der verborgenen Schicht reduziert werden, im Extremfall unter gewissen - naturgemäß sehr speziellen - Voraussetzungen sogar auf nur ein einziges Neuron.

Bei den vorgestellten Verfahren wird das Augenmerk auch auf die insbesondere für den Praxiseinsatz wichtigen Genauigkeitsaussagen gerichtet. So werden jeweils Fehlerbetrachtungen durchgeführt und Aussagen zur Güte der Methoden formuliert.

Schließlich werden (im nachfolgenden Kapitel) verschiedene Regelextraktionsmechanismen vorgestellt, die im Prinzip in jeder Phase des hier skizzierten Ablaufes angewendet werden können.

Bemerkung 6.1

Da der in der Praxis verwendete Regeltyp

$$IF\ x_1\ IN\ A_1\ AND\ ...\ AND\ x_K\ IN\ A_K\ THEN\ y_1\ IN\ B_1\ AND\ ...\ AND\ y_M\ IN\ B_M$$

im mehrdimensionalen Fall eine Und-verknüpfte Konklusion besitzt, werden die nachfolgenden Formulierungen auf den eindimensionalen Fall zugeschnitten; dies stellt jedoch keine besonders gravierende Einschränkung dar, da ein K-L-M-Netz als “Summe” von M “eindimensionalen” K-L-1-Netzen interpretiert werden kann. Dort, wo später über die Anzahl der Regeln gesprochen wird, erhöht sich diese jedoch durch diese Zerlegung im allgemeinen Fall, da die Prämissen für die einzelnen Dimensionen unabhängig voneinander formuliert werden.

6.1 Approximation der Sigmoiden

Wir beginnen damit, die sigmoide Funktion durch eine geeignete und einfacher zu berechnende Funktion zu approximieren. Die Motivation hierfür resultiert aus der Weiterführung der Betrachtung des folgenden Hilfssatzes, der ausführt, dass es tatsächlich gerechtfertigt ist anzunehmen, dass die Gewichtsvektoren, die zu einem bestimmten Neuron hinführen, paarweise verschieden sind. Vgl. hierzu die Bemerkung auf Seite 84: dort wurde vorausgesetzt, dass je zwei Gewichtsvektoren verschieden voneinander sind.

Umgekehrt zeigt der Satz, dass zwei Neuronen, zu denen dieselben Gewichtsvektoren gehören, auch zu einem einzigen Neuron zusammengefasst werden können, womit bereits der Weg zum nachfolgenden Abschnitt 6.2 bereitet ist, in dem es um die Möglichkeiten der Reduktion der Neuronenanzahl geht.

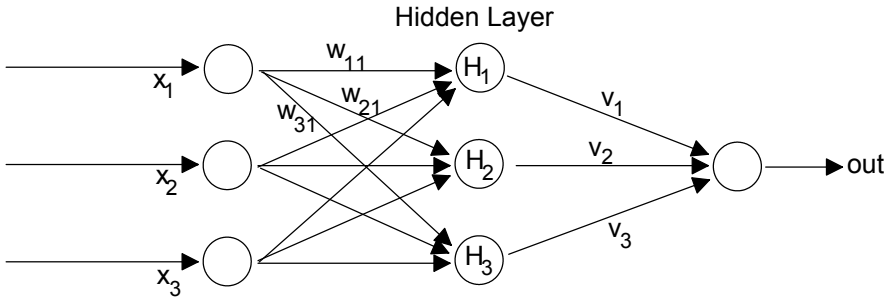


Abb. 6.1: Skizze eines beispielhaften 3-Schicht-Netzes (Typ "3-3-1")

Satz 6.2 (Zusammenfassung bei gleichen Gewichtsvektoren)

Es sei ein K-L-1-Netz gegeben; die Gewichtsvektoren zu den Neuronen des Hidden Layers werden mit $\overrightarrow{w_1}, \ldots, \overrightarrow{w_L}$ bezeichnet[19], die Gewichte zum Ausgabeneuron mit $v_1, \ldots, v_L$. Die Aktivierungsfunktion der Neuronen im Hidden Layer werde mit *s* bezeichnet.

Sind zwei Gewichtsvektoren $\overrightarrow{w_{i_1}}$ und $\overrightarrow{w_{i_2}}$ gleich, so können die beiden Neuronen H_{i_1}, H_{i_2} durch ein anderes Neuron H_0 ersetzt werden; für den Gewichtsvektor $\overrightarrow{w_0}$ zu diesem Neuron und das Gewicht v_0 von diesem Neuron zum Ausgabeneuron gilt dann: $\overrightarrow{w_o} = \overrightarrow{w_{i_1}} = \overrightarrow{w_{i_2}}$, $v_0 = v_{i_1} + v_{i_2}$.

Beweis:

Die beiden Neuronen H_{i_1}, H_{i_2} leisten zu der Ausgabe des Netzes $out = \sum_{1 \le j \le L} v_j \cdot s(\overrightarrow{w_j} \cdot \overrightarrow{x})$ den Beitrag $v_{i_1} \cdot s(\overrightarrow{w_{i_1}} \cdot \overrightarrow{x}) + v_{i_2} \cdot s(\overrightarrow{w_{i_2}} \cdot \overrightarrow{x})$. Mit $\overrightarrow{w_{i_1}} = \overrightarrow{w_{i_2}}$ kann dieser Ausdruck äquivalent ersetzt werden durch $(v_{i_1} + v_{i_2}) \cdot s(\overrightarrow{w_{i_1}} \cdot \overrightarrow{x}) = v_0 \cdot s(\overrightarrow{w_0} \cdot \overrightarrow{x})$, womit die Behauptung bewiesen ist.

❑

Bemerkung

Der Buchstabe *s* in der Formulierung des Satzes deutet an, dass dies typischerweise eine sigmoide Funktion ist. Der Beweis zeigt jedoch, dass es bei diesem Resultat nicht auf die spezielle Gestalt der Funktion *s* ankommt.

Beispiel 6.3

Das soeben formulierte Resultat soll anhand eines einfachen Zahlenbeispiels illustriert werden.
Gegeben sei ein 2-2-1-Netz wie nachstehend abgebildet. Dieses Beispiel betrachtet der

[19] So sind z.B. im Vektor $\overrightarrow{w_1}$ die Gewichte w_{11} vom ersten Eingabeneuron zum ersten Neuron des Hidden Layers, w_{12} vom zweiten Eingabeneuron wiederum zum ersten Neuron des Hidden Layers usw. enthalten.

Übersichtlichkeit halber nur die beiden Neuronen, die anschließend zusammengefasst werden.

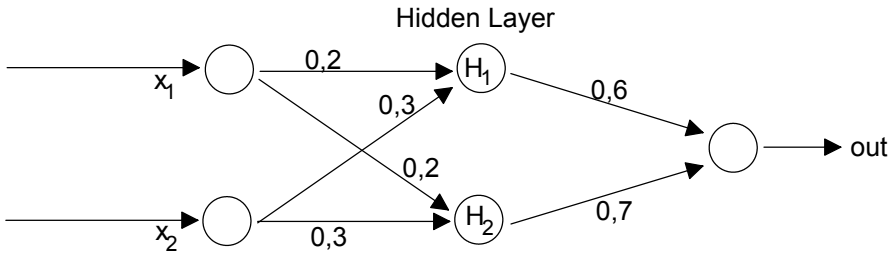


Abb. 6.2: Beispiel eines 2-2-1-Netzes

Die Gewichtsvektoren $\overrightarrow{w_i}$ hin zum i-ten Neuron im Hidden Layer seien $\overrightarrow{w_1} = \overrightarrow{w_2} = \begin{pmatrix} 0,2 \\ 0,3 \end{pmatrix}$, die Gewichte zur Ausgabeschicht $v_1 = 0,6$ und $v_1 = 0,7$.

Dann berechnet sich gemäß Satz 6.2 die Netzausgabe *out* des 2-2-1-Netzes zu

$out(\overrightarrow{x}) = v_1 \cdot s(\overrightarrow{w_1} \cdot \overrightarrow{x}) + v_2 \cdot s(\overrightarrow{w_2} \cdot \overrightarrow{x}) = 0,6 \cdot s(0,2x_1 + 0,3x_2) + 0,7 \cdot s(0,2x_1 + 0,3x_2)$

$= (0,6 + 0,7) \cdot s(0,2x_1 + 0,3x_2) = 1,3 \cdot s(0,2x_1 + 0,3x_2).$

Dies ist aber auch genau die Ausgabe des in der folgenden Abbildung gezeigten vereinfachten (“reduzierten”) 2-1-1-Netzes, bei dem nur noch ein Neuron in der verborgenen Schicht vorliegt.

Der Gewichtsvektor $\overrightarrow{w}$ hin zum Hidden-Neuron ist $\overrightarrow{w} = \overrightarrow{w_1} = \overrightarrow{w_2} = \begin{pmatrix} 0,2 \\ 0,3 \end{pmatrix}$, das Gewicht zur Ausgabeschicht $v = v_1 + v_2 = 0,6 + 0,7 = 1,3$, die Ausgabe errechnet sich damit als $out(\overrightarrow{x}) = 1,3 \cdot s(0,2x_1 + 0,3x_2)$.

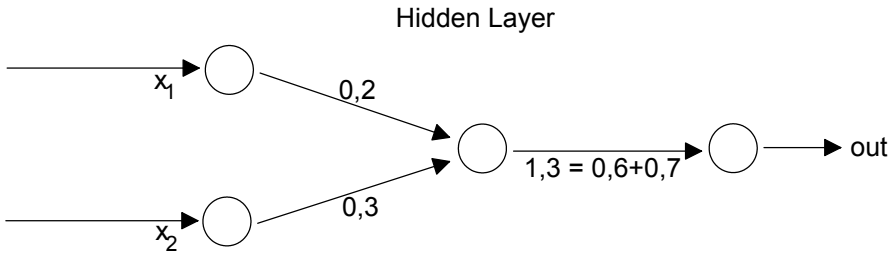


Abb. 6.3: Das reduzierte 2-1-1-Netz

Folgerung 6.4

Der vorherige Satz 6.2 lässt sich durch iterierte Anwendung auch auf mehr als zwei Neuronen des Hidden Layers mit identischen Gewichtsvektoren übertragen. Insgesamt lässt sich die Menge aller Neuronen des Hidden Layers mit denselben Gewichtsvektoren durch ein einziges Neuron mit entsprechend angepasstem Gewicht zur Ausgabeschicht ersetzen. Dieser Satz kann entsprechend auch auf mehrschichtige Netze übertragen werden, die wir allerdings im Rahmen dieser Arbeit nicht weiter behandeln.

Ebenso können auch Neuronen des Hidden Layers mit *ähnlichen* Gewichtsvektoren zur Ausgabeschicht zusammengefasst werden, dann jedoch lediglich im Sinne einer Approximation.

Bemerkung 6.5

Aufgrund der Nichtlinearität der sigmoiden Funktion gibt es kein ähnlich einfaches Resultat zur Reduktion der Neuronen-Anzahl im Sinne einer exakten Darstellung, wenn die betreffenden Gewichtsvektoren paarweise verschieden sind.

Es kann jedoch eine numerische Approximation an die sigmoide Funktion vorgenommen werden. Hierzu können beispielsweise Treppenfunktionen verwendet werden, die mit zunehmender Granularität die sigmoide Funktion beliebig genau annähern können. Wir betrachten im Folgenden stellvertretend für die Gruppe der sigmoiden Funktionen die spezielle logistische Funktion $x \to 1/(1+e^{-x})$, vgl. Bemerkung 1.11 auf S. 24. Sinngemäß sind die getätigten Aussagen aber auch auf andere sigmoide Funktionen anwendbar.

Mathematisch ist auch zu präzisieren, auf welcher Grundmenge und mit welcher Norm diese Approximationsaussage gilt. Für den praktischen Einsatz werden wir stets eine kompakte Grundmenge betrachten; als Norm können die Maximum- oder die L^2-Norm verwendet werden. Aufgrund des damit verbundenen Rechenaufwandes scheint es indes sinnvoller zu sein, eine andere Form der Approximation zu wählen.

Hierzu definieren wir sogenannte “Rampenfunktionen”, die eine im zentralen Bereich um den Punkt $(0, \frac{1}{2})$ symmetrische Gerade beschreiben, die bei den Funktionswerten 0 bzw. 1 abgeschnitten und jeweils konstant fortgesetzt werden (vgl. nachstehende Skizze). Die hier vorgestellte Definition zielt direkt auf die Approximation der sigmoiden Funktion ab; daher wird zugunsten der einfacheren Lesbarkeit auf weitere Parametrisierungen dieses Funktionstyps verzichtet.

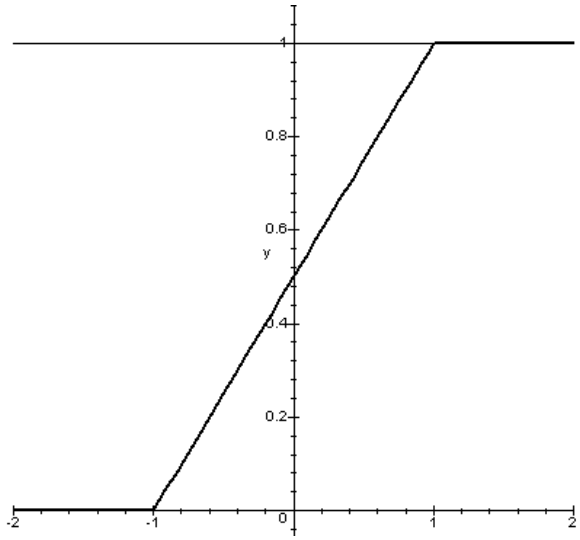


Abb. 6.4: Darstellung der Rampenfunktion mit x_0=1

Definition 6.6 (Rampenfunktion)

Es sei x_0 eine positive reelle Zahl. Dann nennen wir die Funktion R_{x_0} mit der stückweisen Abbildungsvorschrift

$$\begin{aligned} R_{x_0} &: x \to \tfrac{1}{2} + \tfrac{x}{2x_0} && \text{für } -x_0 \le x \le x_0, \\ R_{x_0} &: x \to 0 && \text{für } x < -x_0 \quad \text{und} \\ R_{x_0} &: x \to 1 && \text{für } x > x_0 \end{aligned}$$

Rampenfunktion (zum Parameter x_0).

Das Intervall $[-x_0, x_0]$ nennen wir *Kernintervall* oder *Kernbereich* der Funktion R_{x_0}.

Notation und Bemerkung 6.7

Bezeichnen wir wie bereits in Definition 1.12 erwähnt nachfolgend mit *sigm* als spezielle sigmoide Funktion die logistische Funktion mit der Abbildungsvorschrift $sigm(x) := \frac{1}{1+e^{-x}}$. Dann können wir diese - wie die nachstehende Abb. 6.5 verdeutlicht - durch eine geeignet gewählte Rampenfunktion R_{x_0} annähern. Beschränkt man sich in den anschließenden Betrachtungen auf das Kernintervall $[-x_0, x_0]$, so kann für weitere Betrachtungen auf die dort geltende Linearität von R_{x_0} zurückgegriffen werden.

Auf dem Kernintervall besitzt die Rampenfunktion R_{x_0} die Steigung $\frac{1}{2x_0}$. Jede der hier definierten Rampenfunktionen ist symmetrisch um den Punkt $(0, \frac{1}{2})$.

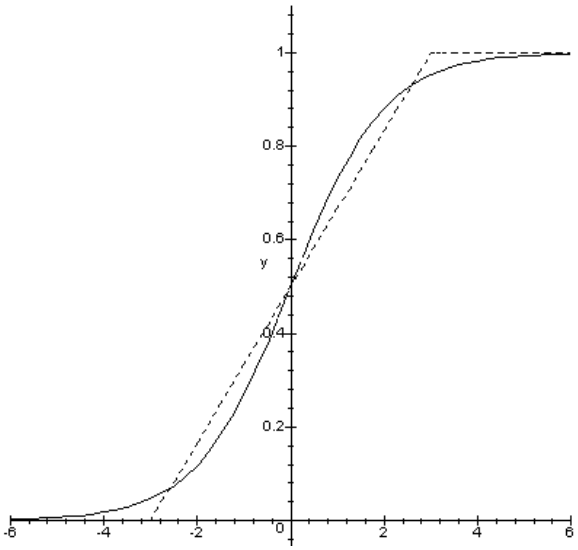


Abb. 6.5: Approximation der sigmoiden Funktion
durch eine (gestrichelt dargestellte) Rampenfunktion (hier: x_0=3)

Satz 6.8 (Approximation der sigmoiden Funktion durch eine Rampenfunktion)

Die sigmoide Funktion *sigm* kann durch Rampenfunktionen R_{x_0} mit $x_0 > 0$ angenähert werden.

1. Für $x_0 = 2$ erhalten wir auf dem Intervall [-2,2] die im Punkt $(0, \frac{1}{2})$ anliegende Tangente an die sigmoide Funktion.
2. Für $x_0 > 2$ wird die maximale Abweichung zwischen der sigmoiden Funktion und R_{x_0} auf dem Kernintervall an der Stelle $x_1 = \ln(x_0 - 1 + \sqrt{x_0(x_0-2)})$ angenommen und beträgt $|R_{x_0}(x_1) - sigm(x_1)| = \left|\frac{1}{2} + \frac{1}{2x_0}x_1 - \frac{1}{1+e^{-x_1}}\right| =$
$\left|\frac{1}{2} + \frac{1}{2x_0}\ln(x_0 - 1 + \sqrt{x_0(x_0-2)}) - \frac{x_0-1+\sqrt{x_0(x_0-2)}}{x_0+\sqrt{x_0(x_0-2)}}\right|$.
3. Außerhalb des Kernintervalls liegt die maximale Abweichung an der Stelle x_0 vor. Diese beträgt $1 - sigm(x_0) = 1 - 1/(1+e^{-x_0}) = e^{-x_0}/(1+e^{-x_0}) = 1/(1+e^{x_0})$.
Für $0 < x_0 < 2$ ist dies auch gleichzeitig die maximale Abweichung der beiden Funktionen auf dem gesamten Definitionsbereich.

Beweis:

Aus Symmetriegründen betrachten wir explizit nur den positiven Definitionsbereich.

1. Die Aussage ist evident, da die Steigung der sigmoiden Funktion an der Stelle 0 den Wert 0.25 besitzt: $(\frac{\partial}{\partial x}sigm)(0) = \frac{e^{-0}}{(1+e^{-0})^2} = \frac{1}{4}$; dies ist auch die Steigung der Rampenfunktion R_2 (in $x = 0$ bzw. auf dem gesamten Kernbereich).
2. Sei nun $x_0 > 2$. Für den hier betrachteten positiven Teil des Kernbereichs gilt $0 < x < x_0$.
Wir betrachten die erste Ableitung der Differenz der beiden (auf dem betrachteten Intervall differenzierbaren) Funktionen und ermitteln mit deren Nullstelle das Maximum dieser Differenzfunktion:
$\frac{\partial}{\partial x}(\frac{1}{1+e^{-x}} - \frac{1}{2} - \frac{1}{2x_0}x) = \frac{e^{-x}}{(1+e^{-x})^2} - \frac{1}{2x_0} = 0.$
Dies formen wir äquivalent um zu $2x_0 = \frac{(1+e^{-x})^2}{e^{-x}} = \frac{(1+e^{x})^2}{e^{x}}$ und substitutieren $z := e^x$.

(Wegen $x > 0$ ist somit $z > 1$.)
Umgeformt erhalten wir die folgende quadratische Gleichung in z.
$0 = z^2 + (2 - 2x_0)z + 1$
Unter den vorliegenden Nebenbedingungen $(x_0 > 2, z > 1)$ gelangen wir zur einzigen Lösung der Gleichung: $z = x_0 - 1 + \sqrt{x_0(x_0 - 2)}$.
Ersetzen wir schließlich z wieder durch x, so erhalten wir für den x-Wert des gesuchten Maximums x_1: $x_1 = \ln(x_0 - 1 + \sqrt{x_0(x_0 - 2)})$.
Setzen wir diesen Wert ein, so erhalten wir:

$$R_{x_0}(x_1) - sigm(x_1) = \tfrac{1}{2} + \tfrac{1}{2x_0}x_1 - \tfrac{1}{1+e^{-x_1}}$$
$$= \tfrac{1}{2} + \tfrac{1}{2x_0}\ln(x_0 - 1 + \sqrt{x_0(x_0 - 2)}) - \frac{1}{1+e^{-\ln(x_0-1+\sqrt{x_0(x_0-2)})}}$$
$$= \tfrac{1}{2} + \tfrac{1}{2x_0}\ln(x_0 - 1 + \sqrt{x_0(x_0 - 2)}) - \frac{1}{1+1/(x_0-1+\sqrt{x_0(x_0-2)})}$$
$$= \tfrac{1}{2} + \tfrac{1}{2x_0}\ln(x_0 - 1 + \sqrt{x_0(x_0 - 2)}) - \frac{x_0-1+\sqrt{x_0(x_0-2)}}{x_0+\sqrt{x_0(x_0-2)}},$$ w.z.b.w.

Da die Rampenfunktion an dieser Stelle unterhalb der Sigmoiden liegt, ist dieser Ausdruck negativ; daher treten in der Formulierung des Satzes Betragsstriche auf.

Für $0 < x_0 < 2$ liegt offensichtlich der maximale Abstand der beiden Funktionen auf dem Kernbereich der Rampenfunktion an der Stelle x_0 vor, denn die Steigung der sigmoiden Funktion ist hier (streng) monoton fallend; dadurch vergrößert sich der Abstand zur approximierenden Geraden bis zum Wert x_0. Ab dieser Stelle nähert sich dann die sigmoide Funktion dem Funktionswert 1 der Rampenfunktion an.

3. Außerhalb des Kernbereiches ist (für x > 0 gemäß Vorbemerkung) nur die Abweichung der sigmoiden Funktion von der konstanten Funktion 1 zu berechnen. Diese berechnet sich zu
$1 - sigm(x) = 1 - 1/(1 + e^{-x}) = e^{-x}/(1 + e^{-x}) = 1/(1 + e^{x})$
und ist aus Monotoniegründen maximal an der Stelle $x = x_0$.

❑

Beispiel 6.9

Die nachfolgenden Bilder illustrieren das Verhalten der Rampenfunktion bzw. deren betragsmäßige Abweichung von der Sigmoiden für verschiedene Werte von x_0.
Zum besseren Erkennen wurde die Abweichung zwischen der Rampenfunktion und der Sigmoiden mit dem Faktor 10 hochskaliert.

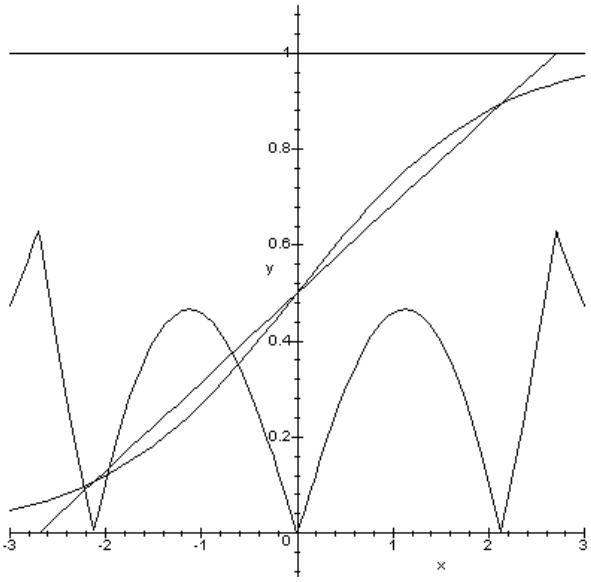


Abb. 6.6: Rampenfunktion, Sigmoide und zehnfach skalierte betragsmäßige Abweichung für x_0=2.7

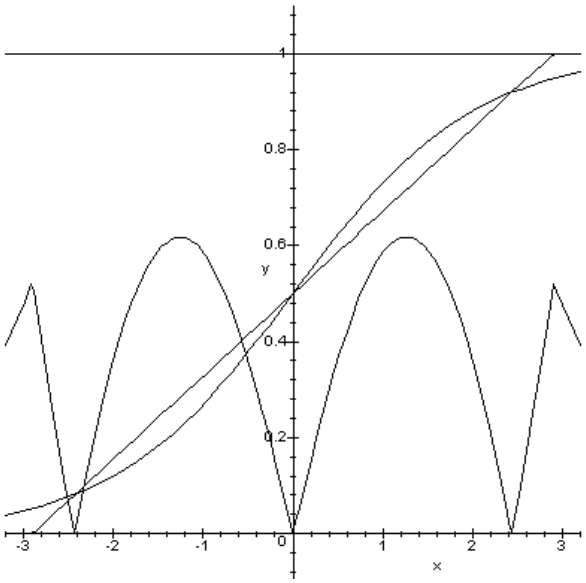


Abb. 6.7: Rampenfunktion, Sigmoide und zehnfach skalierte betragsmäßige Abweichung für x_0=2.9

In beiden Fällen stellt man als ungefähre quantitative Schätzung (unter Berücksichtigung des im Bild verwendeten Skalierungsfaktors 10) fest, dass die maximale Abweichung der Rampenfunktion von der Sigmoiden etwa 0.06 beträgt.

Folgerung 6.10

Der Approximationssatz 6.8 liefert die explizite Angabe, bei welchem x-Wert die maximale Abweichung zwischen den beiden betrachteten Funktionen in Abhängigkeit von dem Parameter x_0 vorliegt: innerhalb des Kernbereiches ist dies $x_1 = \ln(x_0 - 1 + \sqrt{x_0(x_0-2)})$, außerhalb des Kernbereiches x_0.

Die Abweichungen betragen dort $\left| \frac{1}{2} + \frac{1}{2x_0} \ln(x_0 - 1 + \sqrt{x_0(x_0-2)}) - \frac{x_0 - 1 + \sqrt{x_0(x_0-2)}}{x_0 + \sqrt{x_0(x_0-2)}} \right|$ bzw. $1/(1+e^{x_0})$.

Aus Stetigkeitsgründen wird die maximale Abweichung minimal, wenn diese beiden Ausdrücke gleich sind, d.h. für x_0 mit der Eigenschaft

$$\left| \frac{1}{2} + \frac{1}{2x_0} \ln(x_0 - 1 + \sqrt{x_0(x_0-2)}) - \frac{x_0 - 1 + \sqrt{x_0(x_0-2)}}{x_0 + \sqrt{x_0(x_0-2)}} \right| = 1/(1+e^{x_0})$$

bzw. äquivalent dazu

$$\left| \frac{1}{2} + \frac{1}{2x_0} \ln(x_0 - 1 + \sqrt{x_0(x_0-2)}) - \frac{x_0 - 1 + \sqrt{x_0(x_0-2)}}{x_0 + \sqrt{x_0(x_0-2)}} \right| - 1/(1+e^{x_0}) = 0. \qquad (*)$$

Die nachstehenden beiden Grafiken veranschaulichen die linke Seite dieser letzten Gleichung (*) in Abhängigkeit von x_0, die erste zeigt einen "makroskopischen" Blick über dem x-Bereich 2..10, die zweite zoomt hinein in den Bereich der Nullstelle.

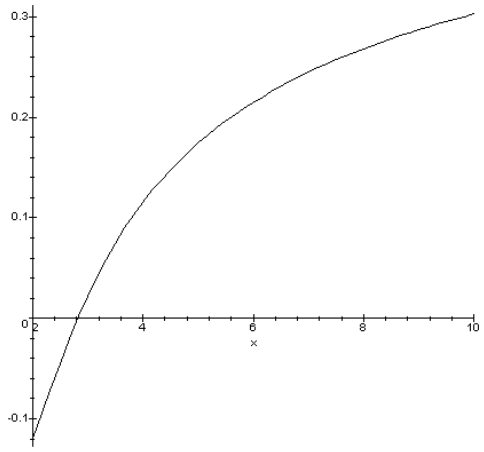


Abb. 6.8: Grafische Darstellung des Ausdrucks von Gleichung (*)

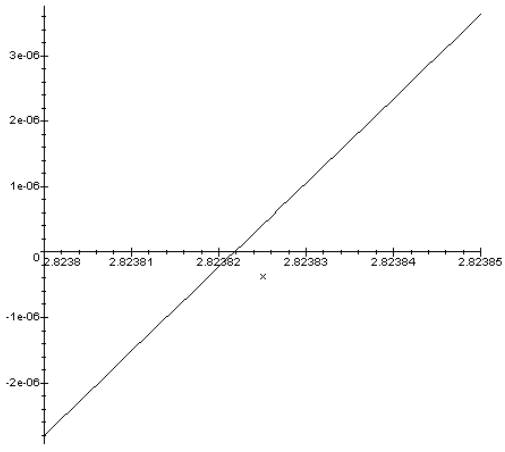


Abb. 6.9: Grafische Ermittlung der Nullstelle (sh. Gleichung (*) auf Seite 115)

Bemerkung 6.11 (Abweichung der Rampenfunktion von der Sigmoiden)

Der betreffende Wert x_0 für die Nullstelle kann numerisch approximativ bestimmt werden; wir begnügen uns für das Folgende mit dem Näherungswert $x_0 = 2.82382$; hierfür beträgt die maximale Abweichung $\delta_{\max}$ der Sigmoiden von der Rampenfunktion $R_{x_0} = R_{2.82382}$ etwa $\delta_{\max} \approx 0.056$.
Mit etwas größerer Präzision liefert der Ausdruck $1/(1+e^{x_0})$ den (wiederum numerischen Näherungs-)Wert 0.05605. Wir werden im Folgenden der einfacheren Lesbarkeit jedoch einfach die Bezeichnung $\delta_{\max}$ bzw. den Näherungswert 0.056 verwenden.

Satz 6.12 (Abweichung der Netz-Ausgabe bei Einsatz der Rampenfunktion)

Betrachten wir ein dreischichtiges Feedforward-Netz vom Typ K-L-1 mit einem Ausgabeneuron wie schematisch in Abbildung 6.1 (vgl. S. 108) dargestellt. Dann folgt aus dem bisher Gesagten, dass sich die maximale (betragsmäßige) Abweichung der Netz-Ausgaben bei Ersetzen der sigmoiden Funktion in der inneren Schicht durch die Rampenfunktion $R_{x_0} = R_{2.82382}$ berechnen lässt zu $\sum_{1\le j\le L} |v_j| \cdot \delta_{\max} \approx \sum_{1\le j\le L} |v_j| \cdot 0.056$.

Beweis:

Die Gewichtsvektoren zu den L Neuronen des Hidden Layers werden wieder mit $\overrightarrow{w_1}, \ldots, \overrightarrow{w_L}$ bezeichnet, die Gewichte zum Ausgabeneuron mit $v_1, \ldots, v_L$. Zu dem Eingabevektor $\overrightarrow{x}$ ergibt sich die Ausgabe unter Verwendung der sigmoiden Funktion *sigm* zu $\sum_{1\le j\le L} v_j \cdot sigm(\overrightarrow{w_j} \cdot \overrightarrow{x})$; bei Verwendung der Rampenfunktion R_{x_0} (mit $x_0 = 2.82382$) erhalten wir die entsprechende Netz-Ausgabe $\sum_{1\le j\le L} v_j \cdot R_{x_0}(\overrightarrow{w_j} \cdot \overrightarrow{x})$. Es resultiert unter Verwendung von Bemerkung 6.11 die betragsmäßige Abschätzung für den Abstand dieser beiden Ausgaben:

$$\left| \sum_{1\le j\le L} v_j \cdot sigm(\overrightarrow{w_j} \cdot \overrightarrow{x}) - \sum_{1\le j\le L} v_j \cdot R_{x_0}(\overrightarrow{w_j} \cdot \overrightarrow{x}) \right| \le$$
$$\sum_{1\le j\le L} |v_j| \cdot |sigm(\overrightarrow{w_j} \cdot \overrightarrow{x}) - R_{x_0}(\overrightarrow{w_j} \cdot \overrightarrow{x})| \le \sum_{1\le j\le L} |v_j| \cdot \delta_{\max} \approx \sum_{1\le j\le L} |v_j| \cdot 0.056 .$$

❑

Bemerkung 6.13

a) Diese Ungleichung liefert eine "*worst case*"-Abschätzung bei approximativer Berechnung der Netzausgabe mittels der Rampenfunktion anstelle der Sigmoiden. In der Praxis wird der Fehler bei realen Datensätzen naturgemäß meist geringer ausfallen, da sich die Abweichungen der Näherungswerte vorzeichenbehaftet teilweise aufheben werden.

b) Trivialerweise ist die Approximation durch die Rampenfunktion bei Eingabe des Nullvektors $\overrightarrow{0}$ präzise, denn mit $\overrightarrow{w_j} \cdot \overrightarrow{0} = 0$ sind $sigm(0) = \frac{1}{2}$, $R_{x_0}(0) = \frac{1}{2}$, und es gilt:
$$\sum_{1\le j\le L} v_j \cdot sigm(\overrightarrow{w_j} \cdot \overrightarrow{0}) = \sum_{1\le j\le L} v_j \cdot \tfrac{1}{2} = \sum_{1\le j\le L} v_j \cdot R_{x_0}(\overrightarrow{w_j} \cdot \overrightarrow{0}).$$
Das heißt: zur Eingabe des Nullvektors liefert das Netz stets die Ausgabe $\sum_{1\le j\le L} \frac{1}{2} v_j$.

Numerisches Beispiel 6.14 (Sinus-Approximation)

Messen wir die auftretenden Abweichungen, wenn die sigmoide Funktion durch die o.e. Rampenfunktion $R_{x_0} = R_{2.82382}$ ersetzt wird, am Beispiel eines einfachen Feedforward-Netzes, das die Funktion $\frac{1}{2}(1 + \sin(x))$ berechnet. Diese modifizierte Sinus-Funktion mit Werten zwischen 0 und 1 wurde gewählt, da hier die sigmoide Funktion ein günstiger Baustein zur Approximation ist.

Die in Anhang A1 (S. 172) aufgeführten Resultate zeigen, dass erwartungsgemäß der *absolute* Fehler nach Satz 6.12 abgeschätzt werden kann und die theoretische Schranke der Abweichung in diesem Fall nicht erreicht wird; allerdings zeigen sich, *prozentual* auf den Ergebniswert bezogen, der mit der sigmoiden Funktion erzielt wird, gravierende Abweichungen von etwa 180% dort, wo dieser Ergebniswert nahe 0 liegt, was aus der Mischung von positiven und negativen Gewichtsvektoren v_i resultiert!

Dies ist ein numerisch plausibles Ergebnis, zeigt aber, dass das Ersetzen der Ausgabefunktion durch die Rampenfunktion nur für bestimmte Bereiche sinnvoll zu sein scheint. Dort allerdings kann deren Verwendung auch die Performance der Software-Implementierung des Netzes durchaus steigern, da keine Exponentialfunktion berechnet werden muss. Ggf. kann möglicherweise auch die konkrete Problemstellung "von der 0 weg" skaliert werden; dadurch wird der hier gezeigte extreme Effekt vermieden, der wegen der Division durch einen Wert "nahe bei 0" aufgetreten ist.

Bemerkung 6.15

Gemäß Satz 1.36 (S. 40) sind dreischichtige Netze mit der Rampenfunktion als nichtkonstanter, stetiger Ausgabefunktion universelle Approximatoren auf einem Kompaktum; d.h. auch Neuronale Netze, die statt der Sigmoiden die Rampenfunktion verwenden, können zur Darstellung von quadratintegrablen Funktionen aus $L_2(K)$ auf einer kompakten Grundmenge K eingesetzt werden.

6.2 Reduktion der Neuronenanzahl in der verborgenen Schicht

Unter gewissen einschränkenden Bedingungen können wir bei Approximation eines Netzes mittels der Rampenfunktion das gesamte K-L-1-Netz durch ein K-1-1-Netz ersetzen, d.h. die gesamte verborgene Schicht kann durch ein einziges Neuron substituiert werden! Sofern diese Voraussetzungen nicht gegeben sind, kann dieses K-1-1-Netz immerhin als Approximation für das Ausgangsnetz dienen. Dies präzisiert der nachfolgende Substitutionssatz. Im Anschluss daran werden dann alternative Möglichkeiten einer Reduktion der Neuronenanzahl präsentiert.

Satz 6.16 (Substitutionssatz)

Es sei ein K-L-1-Multilayer-Feedforward-Netz gegeben.

Für $1 \le i \le L$ sei $\overrightarrow{w_i}$ der Gewichtsvektor zum i-ten Neuron der verborgenen Schicht, v_i sei entsprechend das Gewicht von diesem Neuron zum Ausgabeneuron. Ferner sei wiederum $x_0 = 2.82382$.

Es gelte weiterhin:

1. Auf der betrachteten Eingabemenge $E \subset \mathbb{R}^K$ gelte für $1 \le i \le L$: $\overrightarrow{w_i} \cdot \overrightarrow{x} \in [-x_0, x_0]$.
2. $v_1 + v_2 + ... + v_L \neq 0$.
3. Als Aktivierungsfunktion des Netzes wird die Rampenfunktion $R_{x_0} = R_{2.82382}$ verwendet.

Gilt auf der betrachteten Eingabemenge $E \subset \mathbb{R}^K$ auch für $\overrightarrow{w} := \frac{v_1\overrightarrow{w_1}+v_2\overrightarrow{w_2}+...+v_L\overrightarrow{w_L}}{v_1+v_2+...+v_L}$, dass $\overrightarrow{w} \cdot \overrightarrow{x} \in [-x_0, x_0]$ erfüllt ist, so ist das ursprüngliche K-L-1-Netz äquivalent ersetzbar durch das nachfolgend beschriebene K-1-1-Netz, bei dem die verborgene Schicht aus lediglich einem Neuron besteht. Hierbei werden die bisherigen L Neuronen der verborgenen Schicht zusammengefasst zu einem Neuron mit dem oben definierten Gewichtsvektor $\overrightarrow{w}$ und dem Gewichtsvektor $v := v_1 + v_2 + ... + v_L$ zum Ausgabeneuron.

Beweis:

Die Ausgabe des K-L-1-Ausgangsnetzes berechnet sich zu

$$out(\overrightarrow{x}) = \sum_{1 \le i \le L} v_i \cdot R_{x_0}(\overrightarrow{w_i} \cdot \overrightarrow{x}) = \sum_{1 \le i \le L} (\tfrac{1}{2} v_i + \tfrac{1}{2x_0} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x}) = \sum_{1 \le i \le L} \tfrac{1}{2} v_i + \tfrac{1}{2x_0} \sum_{1 \le i \le L} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x}.$$

Unter Berücksichtigung von $\overrightarrow{w} \cdot \overrightarrow{x} \in [-x_0, x_0]$ wird die Ausgabe des neuen Netzes wie folgt ermittelt:

$$out_{neu}(\overrightarrow{x}) = v \cdot R_{x_0}(\overrightarrow{w} \cdot \overrightarrow{x}) = v \cdot (\tfrac{1}{2} + \tfrac{1}{2x_0} \overrightarrow{w} \cdot \overrightarrow{x}) = \tfrac{1}{2} v + \tfrac{1}{2x_0} v \cdot \overrightarrow{w} \cdot \overrightarrow{x} =$$

$$= \tfrac{1}{2} v + \tfrac{1}{2x_0} v \cdot (\tfrac{v_1\overrightarrow{w_1}+v_2\overrightarrow{w_2}+...+v_L\overrightarrow{w_L}}{v}) \cdot \overrightarrow{x} = \tfrac{1}{2} v + \tfrac{1}{2x_0} (v_1\overrightarrow{w_1} + v_2\overrightarrow{w_2} + ... + v_L\overrightarrow{w_L}) \cdot \overrightarrow{x} =$$

$$= \sum_{1 \le i \le L} \tfrac{1}{2} v_i + \tfrac{1}{2x_0} \sum_{1 \le i \le L} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x}, \text{ w.z.b.w.}$$

❑

Bemerkung 6.17

In dem Szenario des vorigen Satzes regelt somit der Vektor $\overrightarrow{z} := \sum_{1 \le i \le L} v_i \cdot \overrightarrow{w_i}$ das Verhalten des gesamten Netzes.

Bemerkung 6.18

Für den Spezialfall $L = 2$ für zwei Neuronen mit denselben Gewichtsvektoren ($\overrightarrow{w_1} = \overrightarrow{w_2}$) überschneidet sich die Aussage des Substitutionssatzes 6.16 mit der von Satz 6.2 (S. 108), wenn man letzteren auf die Rampenfunktion als Ausgabefunktion überträgt.

In diesem Fall, $L = 2$, ergeben sich mit $\overrightarrow{w_1} = \overrightarrow{w_2}$ für das neue Neuron die Beziehungen $v := v_1 + v_2$ und $\overrightarrow{w} := \frac{v_1\overrightarrow{w_1}+v_2\overrightarrow{w_2}}{v_1+v_2} = \frac{v_1\overrightarrow{w_1}+v_2\overrightarrow{w_1}}{v_1+v_2} = \frac{(v_1+v_2)\overrightarrow{w_1}}{v_1+v_2} = \overrightarrow{w_1} = \overrightarrow{w_2}$; dies sind aber gerade die Beziehungen, die in Satz 6.2 (im Prinzip für eine beliebige Ausgabefunktion) festgestellt worden sind.

Für das weitere Vorgehen stellen wir einen Begriff und eine Formel bereit: wenn zwei oder mehr Neuronen (der verborgenen Schicht) durch eines ersetzt werden sollen, dann wollen wir von einem Substitutionsneuron sprechen.

Definition 6.19 (Substitutionsneuron)

Gegeben sei ein K-L-1-Netz. Die Gewichtsvektoren von der Eingabeschicht zum i-ten Neuron in der verborgenen Schicht werden mit $\overrightarrow{w_i}$ bezeichnet, das Gewicht von diesem Hidden Neuron zur Ausgabeschicht sei v_i. Es gelte $v_1 + v_2 + ... + v_n \neq 0$.

Sollen n Neuronen der verborgenen Schicht (- o.B.d.A. mit den Indizes 1,2,...,n -) aus dem Netz entfernt und durch ein neues Neuron ersetzt werden, so nennen wir das neue Neuron *Substitutionsneuron*. Dieses erhält den Gewichtsvektor $\overrightarrow{w}$ von der Eingabeschicht bzw. das Gewicht v zur Ausgabeschicht gemäß den nachstehenden Beziehungen:

$$v := v_1 + v_2 + ... + v_n \ , \ \overrightarrow{w} := \frac{v_1\overrightarrow{w_1} + v_2\overrightarrow{w_2} + ... + v_n\overrightarrow{w_n}}{v_1 + v_2 + ... + v_n}.$$

Bemerkung

Der Substitutionssatz 6.16, der diese Formeln für $n = L$ anwendet, legt eine Modifikation nahe. *(*$n = L$ bedeutet, dass *alle* Neuronen der verborgenen Schicht durch *ein* neues Neuron substituiert werden.)
Ist die Approximation durch das resultierende Netz mit *einem* Hidden-Neuron nicht ausreichend, so kann mit *zwei* Neuronen in der Hidden-Schicht gearbeitet werden. Selbstverständlich kann dann schrittweise die Anzahl der Neuronen in der verborgenen Schicht wieder "hochgefahren" werden. Wir wollen an dieser Stelle aber nur den qualitativ noch gut zu diskutierenden Fall von zwei Neuronen in der Hidden-Schicht diskutieren und für den allgemeinen Fall auf Satz 6.38 (S. 131) verweisen.

Satz 6.20 (Erweiterter Substitutionssatz)

Es sei ein K-L-1-Feedforward-Netz gegeben; weiterhin seien die Voraussetzungen wie in Satz 6.16 erfüllt.

Zunächst werden die bisherigen L Neuronen der verborgenen Schicht gruppiert in die mit positivem Verbindungsgewicht zur Ausgabeschicht bzw. mit negativem Gewicht.
O.B.d.A. seien die ersten L_1 Gewichte zur Ausgabeschicht positiv, die restlichen negativ; ein Gewicht 0 zur Ausgabeschicht wird naheliegenderweise nicht auftreten bzw. wäre nicht relevant.

In Formeln bedeutet dies: $v_i > 0 \ \forall i : 1 \leq i \leq L_1, \ \ v_i < 0 \ \forall i : L_1 + 1 \leq i \leq L$.

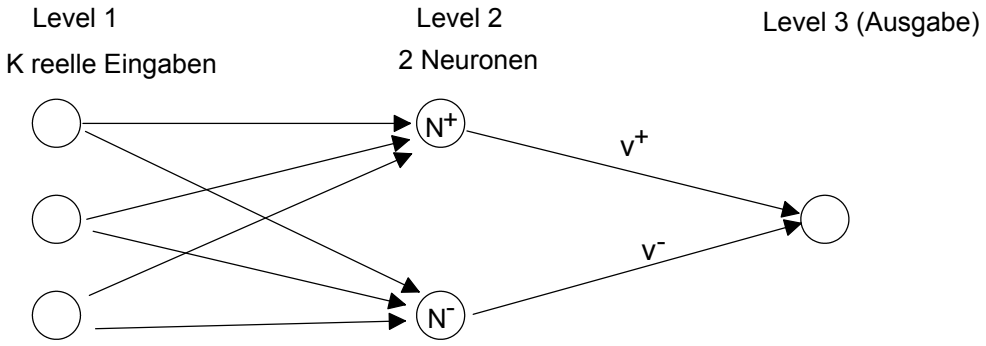


Abb. 6.10: Skizze des K-2-1-Netzes

Es werden zwei Neuronen (N^+, N^-) in der neuen verborgenen Schicht generiert, die analog dem bisherigen Vorgehen die korrespondierenden Neuronen zusammenfassen.

Die Gewichtsvektoren von der Eingabeschicht zu diesen beiden Neuronen werden mit $\overrightarrow{w^+}$ bzw. $\overrightarrow{w^-}$, deren Gewichte zur Ausgabeschicht mit v^+ bzw. v^- bezeichnet. Diese werden wie folgt festgelegt:

$$v^+ := v_1 + v_2 + ... + v_{L_1},\ v^- := v_{L_1+1} + ... + v_L\ ,$$

$$\overrightarrow{w^+} := \frac{v_1\overrightarrow{w_1}+v_2\overrightarrow{w_2}+...+v_{L_1}\overrightarrow{w_{L_1}}}{v_1+v_2+...+v_{L_1}},\ \overrightarrow{w^-} := \frac{v_{L_1+1}\overrightarrow{w_{L_1+1}}+...+v_L\overrightarrow{w_L}}{v_{L_1+1}+v_2+...+v_L}.$$

Mit anderen Worten: nachdem zunächst über die positiven und negativen Gewichte zur Ausgabeschicht gruppiert wird, wird anschließend für jede der beiden Gruppen ein Substitutionsneuron in der verborgenen Schicht bereitgestellt und mit den oben definierten Werten zur Ein- bzw. Ausgabeschicht verbunden.

Gilt auf der betrachteten Eingabemenge $E \subset \mathbb{R}^K$ $\overrightarrow{w^+} \cdot \overrightarrow{x} \in [-x_0, x_0]$ und $\overrightarrow{w^-} \cdot \overrightarrow{x} \in [-x_0, x_0]$, so ist das ursprüngliche K-L-1-Netz äquivalent ersetzbar durch das beschriebene K-2-1-Netz, bei dem die verborgene Schicht aus genau zwei Neuronen besteht.

Beweis:

Der Beweis ist strukturell äquivalent zu dem des ersten Substitutionssatzes. Wie zuvor berechnet sich die Ausgabe des ursprünglichen K-L-1-Netzes zu

$$out(\overrightarrow{x}) = \sum_{1\le i\le L} v_i \cdot R_{x_0}(\overrightarrow{w_i} \cdot \overrightarrow{x}) = \sum_{1\le i\le L} (\tfrac{1}{2}v_i + \tfrac{1}{2x_0}v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x}) =$$

$$= \sum_{1\le i\le L} \tfrac{1}{2}v_i + \tfrac{1}{2x_0} \sum_{1\le i\le L} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x}$$

$$= \sum_{1\le i\le L_1} \tfrac{1}{2}v_i + \tfrac{1}{2x_0} \sum_{1\le i\le L_1} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x} + \sum_{L_1+1\le i\le L} \tfrac{1}{2}v_i + \tfrac{1}{2x_0} \sum_{L_1+1\le i\le L} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x}.$$

Unter Berücksichtigung von $\overrightarrow{w} \cdot \overrightarrow{x} \in [-x_0, x_0]$, $\overrightarrow{w^+} \cdot \overrightarrow{x} \in [-x_0, x_0]$ und $\overrightarrow{w^-} \cdot \overrightarrow{x} \in [-x_0, x_0]$ wird die Ausgabe des neuen Netzes wie folgt berechnet:

$$out_{neu}(\overrightarrow{x}) = v^+ \cdot R_{x_0}(\overrightarrow{w^+} \cdot \overrightarrow{x}) + v^- \cdot R_{x_0}(\overrightarrow{w^-} \cdot \overrightarrow{x}) =$$

$$= v^+ \cdot (\tfrac{1}{2} + \tfrac{1}{2x_0}\overrightarrow{w^+} \cdot \overrightarrow{x}) + v^- \cdot (\tfrac{1}{2} + \tfrac{1}{2x_0}\overrightarrow{w^-} \cdot \overrightarrow{x}) =$$

$$= \tfrac{1}{2}v^+ + \tfrac{1}{2x_0}v^+ \cdot \overrightarrow{w^+} \cdot \overrightarrow{x} + \tfrac{1}{2}v^- + \tfrac{1}{2x_0}v^- \cdot \overrightarrow{w^-} \cdot \overrightarrow{x} =$$

$$= \tfrac{1}{2}v^+ + \tfrac{1}{2x_0}v^+ \cdot \left(\frac{v_1\overrightarrow{w_1}+v_2\overrightarrow{w_2}+\ldots+v_{L_1}\overrightarrow{w_{L_1}}}{v^+}\right) \cdot \overrightarrow{x} + \tfrac{1}{2}v^- + \tfrac{1}{2x_0}v^- \cdot \left(\frac{v_{L_1+1}\overrightarrow{w_{L_1+1}}+\ldots+v_L\overrightarrow{w_L}}{v^-}\right) \cdot \overrightarrow{x} =$$

$$= \sum_{1\le i\le L_1} \tfrac{1}{2}v_i + \tfrac{1}{2x_0} \sum_{1\le i\le L_1} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x} + \sum_{L_1+1\le i\le L} \tfrac{1}{2}v_i + \tfrac{1}{2x_0} \sum_{L_1+1\le i\le L} v_i \cdot \overrightarrow{w_i} \cdot \overrightarrow{x} = out(\overrightarrow{x});$$

somit gilt die Behauptung.

❑

Bemerkung

a) In der Praxis werden die Voraussetzungen der Substitutionssätze 6.16 und 6.20 naturgemäß nicht für alle Eingabevektoren erfüllt sein; in diesem Fall kann allerdings das K-1-1- bzw. K-2-1-Netz als Approximation des Ausgangsnetzes dienen.

b) Trivialerweise gilt, wie bereits in 6.13 angemerkt, dass jedes der o.g. für die Approximation verwendeten Netze zur Eingabe der 0 (bzw. im vektoriellen Fall des Nullvektors) das exakte Ergebnis $\sum_{1\le i\le L} \frac{1}{2}v_i$ liefert.

Nachfolgend wollen wir die Netze, die sich aus den beiden Substitutionssätzen 6.16 und 6.20 ergeben, in einer konkreten Simulation testen.

Numerisches Beispiel 6.21 (Inverses Pendel)

Ein bekanntes Beispiel, welches im Rahmen der Regelextraktion und -beschreibung herangezogen wird, ist die Aufgabenstellung des inversen Pendels. Wir beziehen uns in der konkreten Ausgangsdarstellung auf [Nauc1994] (S. 269 und S. 353). Dieses Beispiel, das von einem einfachen Fuzzy-Controller (vgl. Kap. 2.5) ausgeht, wollen wir im Folgenden wiederholt zur numerischen Demonstration aufgreifen.

Das inverse Pendel ist ein "auf dem Kopf" stehendes Pendel, das so balanciert werden soll, dass es möglichst aufrecht stehen bleibt. Das Pendel kann sich nur in der x-y-Ebene relativ zu seinem Verankerungspunkt im Winkelbereich θ von -90° bis +90° bewegen. Die Kraft F, mit der Einfluss genommen wird auf das Pendel, kann von dem momentanen Auslenkungswinkel θ und der Winkelgeschwindigkeit $\dot{\theta} = \frac{d\theta}{dt}$ abhängen. Es werde angenommen, dass die Winkelgeschwindigkeit zwischen -45°/s und +45°/s rangiert und die Kraft zwischen -10 N und +10 N betragen kann. Bezeichnen wir die Eingangsgrößen mit X_1 und X_2, die Ausgangsgröße mit Y, so haben wir die Intervalle [-90,+90] für X_1, [-45,+45] für X_2 und [-10,+10] für Y.

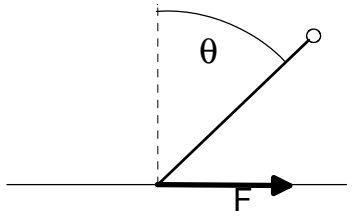


Abb. 6.11: Skizze des Inversen Pendels

Eine recht einfache Regelbasis für das Problem wird bei [Nauc1994] auf S. 353 gegeben. Dabei werden die numerischen Werte jeweils auf den Bereich [-1,+1] skaliert und auf die vier Fuzzy-Bereiche NE (“negative”), NZ (“negative near zero”), PZ (“positive near zero”) und PO (“positive”) abgebildet.

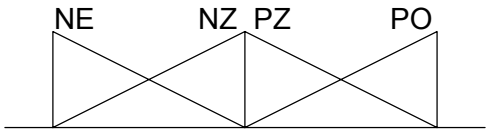


Abb. 6.12: Festlegung der linguistischen Terme

Das nachfolgende Tableau zeigt die sechzehn Regeln schematisch; beispielsweise
“IF X_1 IS NZ AND X_2 IS PZ THEN Y IS NZ”.

	X_1 NE	NZ	PZ	PO
X_2 NE	NE	NE	NE	PZ
NZ	NE	NZ	PZ	PO
PZ	NE	NZ	PZ	PO
PO	NZ	PO	PO	PO

Im Rahmen unserer numerischen Implementierung arbeiten wir mit den Werten -1, -0.5, +0.5 und +1; in crispen Werten wird NE mit dem Intervall [-1,-0.5] identifiziert, NZ mit]-0.5,0], PZ mit]0,+0.5] und PO mit]+0.5,+1]. Die Zugehörigkeit der rechten und/oder linken Grenze zu einem dieser Intervalle ist bei Verwendung von Gleitkommaarithmetik numerisch unerheblich.

X_2	X_1 -1 .. -0,5	-0,5 .. 0	0 .. 0,5	0,5 .. 1
-1 .. -0,5	-1	-1	-1	0,5
-0,5 .. 0	-1	-0,5	0,5	1
0 .. 0,5	-1	-0,5	0,5	1
0,5 .. 1	-0,5	1	1	1

Nachstehende Abbildung illustriert diese (stark vereinfachte) Ausgabefunktion schematisch.

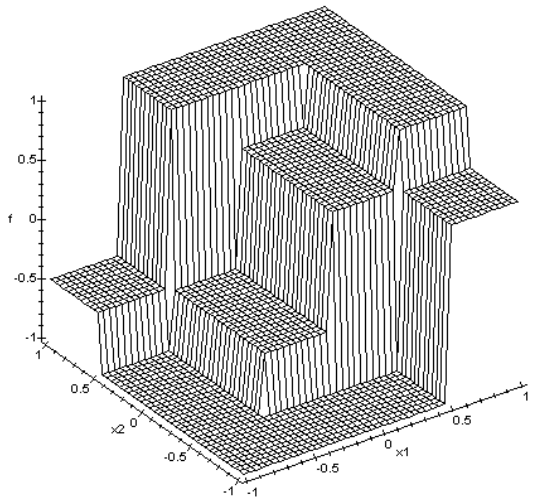


Abb. 6.13: Vereinfachte Darstellung des Inversen Pendels

In Anhang A2 (S. 174 ff) wird eine Simulation mit 250 Durchläufern dokumentiert. Dabei wird zunächst ein dreischichtiges Netz mit 16 Hidden Neuronen trainiert, so dass der mittlere quadratische Fehler (bei Zugrundelegen der sigmoiden Funktion) bei etwa 0,1 liegt. Anschließend werden die aus den beiden Substitutionssätzen 6.16 und 6.20 resultierenden Netze mit nur einem Neuron bzw. mit zwei Neuronen in der Hidden Schicht (und der Rampenfunktion als Ausgabe) generiert und die mittleren quadratischen Fehler über einer Testmenge aufgetragen.

Während sich deutlich zeigt, dass - erwartungsgemäß - das Netz mit nur einem Hidden Neuron generell eine sehr schlechte Approximation (mit einem mittleren MSE um ca. 0,5) leistet, ist die Annäherung mit dem 2-2-1-Netz nach Satz 6.20 (mit einem mittleren MSE bei ca. 0,2) deutlich besser. In einigen Durchläufen weisen diese Netze sogar eine Güte auf, die nahe bei der des Ausgangsnetzes liegt.

Bemerkung 6.22 (Zusammenfassung von Neuronen in der Hidden-Schicht)

Der nächste Verallgemeinerungsschritt wurde bereits angesprochen: anstelle der speziell zwei Neuronen in der Hidden-Schicht beim vorherigen Satz 6.20 kann auch eine größere Anzahl L' Neuronen generiert werden, wobei L'>2 und "klein" im Vergleich zur Ausgangsgröße L sein sollte.

Hierbei sollten diejenigen Hidden-Neuronen des Ausgangsnetzes zu jeweils einem neuen Neuron im K-L'-1-Netz zusammengefasst werden, deren Gewichte zur Ausgabeschicht in Ähnlichkeitsklassen liegen. Im Falle von L'=2 waren dies (im konkreten Fall) die Klasse der positiven und die Klasse der negativen Gewichte, für L'>2 sind verschiedene Vorgehensweisen denkbar.

So können einmal feste Intervalle I_j in Frage kommen (inklusive der beiden jeweils einseitig unbegrenzten "Reste", die die positiven bzw. negativen, betragsmäßig großen Gewichte bündeln), und alle Hidden-Neuronen, deren Gewichte in demselben Intervall I_j liegen, werden in analoger Weise zusammengefasst ("Intervallclusterung").

Es können jedoch auch gruppenweise Hidden-Neuronen mit "benachbarten" Gewichtswerten zusammengefasst werden; hierzu können alle Gewichte sortiert aufgetragen und in L' Gruppen aufgeteilt werden ("Gruppenclusterung").

Beides wird im nachfolgenden einfachen Beispiel kurz dargestellt.

Beispiel 6.23 (Illustration der Cluster-Verfahren)

Die untenstehende Tabelle illustriert die erwähnten Clusterbildungsverfahren an einem kleinen Beispiel. Ausgehend von L=9 Hidden-Neuronen sollen exemplarisch L'=4 Neuronen neu gebildet werden.

<table>
<tr><td>Nr. L des Hidden-Neurons</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td>Gewichtsvektoren v_i
zur Ausgabeschicht</td><td>-2,5</td><td>-1,1</td><td>-0,5</td><td>-0,4</td><td>-0,3</td><td>-0,2</td><td>0,4</td><td>0,6</td><td>1,5</td></tr>
<tr><td>Intervallclusterung
die L'-1 Intervallgrenzen seien -1, 0 und +1</td><td colspan="2">Cluster 1</td><td colspan="4">Cluster 2</td><td colspan="2">Cluster 3</td><td>Cl. 4</td></tr>
<tr><td>Gruppenclusterung
L/L' = 2,25 - d.h. je zwei Neuronen werden zusammen-gefasst, das letzte Neuron wird z.B. dem letzten Cluster zugeordnet.</td><td colspan="2">Cluster 1</td><td colspan="2">Cluster 2</td><td colspan="2">Cluster 3</td><td colspan="3">Cluster 4</td></tr>
</table>

Die Berechnung der Gewichte von der Eingabeschicht zu den neuen Hidden-Neuronen und von diesen zur Ausgabeschicht erfolgt in Analogie zu den in den Substitutionssätzen 6.16 und 6.20 dargestellten Formeln (sh. S. 117 und 119).

Die Gruppenclusterung zeigt ihre (potenzielle) Stärke darin, dass in einem Bereich vieler ähnlicher Gewichtswerte auch mehrere (neue) Neuronen generiert werden; umgekehrt erweist sie sich als anfällig gegenüber Ausreißern, im oben gezeigten Cluster 4 etwa werden die ehemaligen Gewichtswerte im Bereich von 0,4 bis 1,5 in einem einzigen Neuron repräsentiert.

Darauf basierend sind somit auch Mischformen denkbar, etwa das Vorabdefinieren eines "inneren Bereiches", in dem die meisten Gewichte liegen. Innerhalb dieses Bereiches kann dann eine Gruppenclusterung stattfinden, außerhalb werden zwei oder (aus Symmetrie-gründen) eine andere gerade Anzahl von Intervallen gebildet.

Bemerkung 6.24

Bei der hier diskutierten Zusammenfassung von Neuronen der Hidden Schicht ist wiederum darauf zu achten, dass die Summe der betreffenden Gewichtsvektoren v_i nicht verschwindet, da in diesem Falle das ersetzende Neuron nur formal vorhanden wäre, denn dessen Gewicht zur Ausgabeschicht, $v^{neu} := v_{i_1} + v_{i_2} + ...$, wäre 0.

Dies bedeutet bei der Intervallclusterung, dass günstigerweise kein Intervall definiert wird, welches die 0 als inneren Punkt enthält. Bei der Gruppenclusterung muss manuell auf die Einhaltung dieser Bedingung geachtet werden.

Im Rahmen numerischer Gleitkommaberechnungen wird eine exakte Gleichheit mit 0 selten auftreten; da aber in diesem Kontext in der Praxis mit Abweichungstoleranzen gearbeitet wird, bleibt die Problematik sinngemäß bestehen, da auch Werte "sehr nahe 0" nicht sinnvoll wären.

Numerisches Beispiel 6.25

In Anhang A6 auf S. 190 wird ein numerisches Beispiel dargestellt, welches illustriert, dass mehr Neuronen erwartungsgemäß ein besseres Resultat liefern können, jedoch nicht unbedingt müssen.
Insbesondere sind damit aber Situationen zu erkennen, in denen die Gruppenclusterung günstiger als die Intervallclusterung sein kann, dann nämlich, wenn innerhalb der Menge der Neuronen mit ähnlichen Gewichtswerten zur Ausgabeschicht eine bessere Trennschärfe erzielt werden kann.

Numerisches Beispiel 6.26 (Inverses Pendel)

Wir wollen an das zuvor präsentierte Beispiel 6.21 zum inversen Pendel anknüpfen. In Anhang A2 (S. 174 ff) sind neben den bislang bereits diskutierten Ergebnissen auch numerische Resultate für die beiden Clusterungsverfahren aufgeführt. Es zeigt sich anhand der dort dokumentierten Werte, dass die sog. *dynamische* Intervallclusterung häufig etwas besser ist als die Gruppenclusterung, dass jedoch insgesamt im Durchschnitt über die 250 Simulationsläufe mit einem MSE von ca. 0,22 bei der Gruppen- und 0,20 bei der Intervallclusterung eine ähnliche Qualität erreicht wird.

Im Anhang sind die Details dieser Clusterungsverfahren beschrieben. Konkret sind hier Netze mit acht Neuronen in der Hidden Schicht generiert worden, wobei das dynamische Intervallclusterungsverfahren seine Intervalle an den konkret auftretenden Gewichtswerten (zur Ausgabeschicht) orientiert, während das zum Vergleich ebenfalls dargestellte statische Verfahren Intervalle der festen Breite 1 verwendet.

Bei der Gegenüberstellung zeigt sich ebenfalls, dass es bei den hier generierten Netzen nicht wesentlich darauf ankommt, ob die Rampen- oder die sigmoide Funktion für die Ausgabe verwendet wird.

Um sich ein Bild davon zu machen, wie die Ausgabe eines solchen K-L'-1-Netzes (mit der Rampenfunktion als Ausgabefunktion der Hidden-Neuronen) qualitativ aussieht, wird nachfolgend ein einfaches Beispiel grafisch veranschaulicht.

Beispiel 6.27 (Ausgabe eines 2-4-1-Netzes mit Rampenfunktionen)

Die nachstehende Abbildung veranschaulicht eine Kombination von Rampenfunktionen, wie sie in einem solchen approximierenden K-L'-1-Netz als Ausgabe(funktion) auftritt.

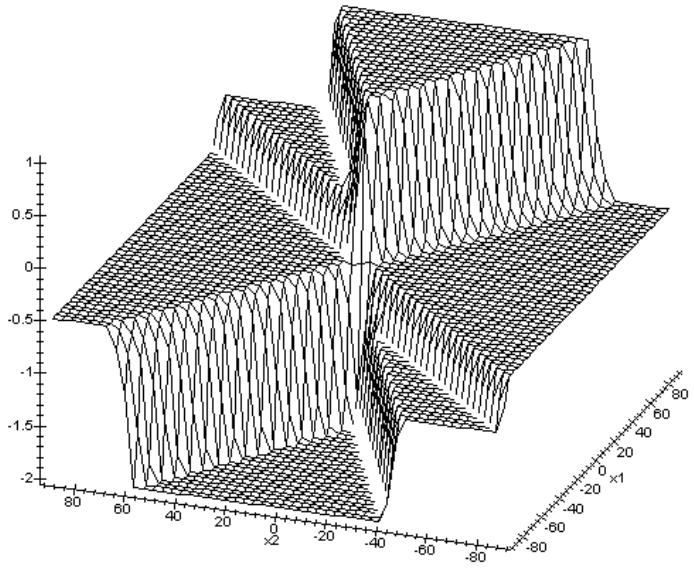


Abb. 6.14: Beispiel einer Kombination von Rampenfunktionen

Hier ist $K = 2$, $L' = 4$, die Gewichtsvektoren und -werte sind nachstehend aufgeführt. Wie bereits zuvor: die Vektoren sind die Gewichtsvektoren von der Eingabeschicht zur verborgenen Schicht, die Gewichte v_i sind die Gewichte von den Neuronen der Hidden Schicht zum Ausgabeneuron.

Die hier verwendeten Werte sind selbstverständlich willkürlich, jedoch so gewählt, dass der hier gezeigte Graph die qualitativen Zusammenhänge gut sichtbar illustriert.

$$\overrightarrow{w_1} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \overrightarrow{w_2} = \begin{pmatrix} -1 \\ -1.5 \end{pmatrix}, \overrightarrow{w_3} = \begin{pmatrix} 0.5 \\ 0.7 \end{pmatrix}, \overrightarrow{w_4} = \begin{pmatrix} -0.5 \\ 1 \end{pmatrix},$$

$$v_1 = 0.5,\ v_2 = -1,\ v_3 = 0.5,\ v_4 = -1.$$

Die Eingangsgrößen x_1 und x_2 wurden über dem Bereich [-90,+90] aufgetragen, damit die Plateaubildung gut zu erkennen ist. Die größten und kleinsten auftretenden Ausgabewerte (und damit Plateaus im obigen Bild) sind +1 und -2; dies korrespondiert mit den beiden positiven Gewichten v_1 und v_3 und den negativen Gewichten v_2 und v_4. Die Auf- und Abstiegsbereiche ("Hänge") richten sich orthogonal zu den Gewichtsvektoren $\overrightarrow{w_i}$ aus, denn dort sind die Ausdrücke $\overrightarrow{w_i} \cdot \overrightarrow{x} \approx 0$, die betreffenden Terme in der Netzausgabe ändern sich hier von 0 nach 1 bzw. von 0 nach v_i.

Interessant ist hier noch ein Blick auf die Abweichung, die entsteht, wenn die Rampenfunktion durch die sigmoide Funktion ersetzt wird. Der Graph der Netzausgabe bei Verwendung der sigmoiden Funktion sieht im oben verwendeten Maßstab nahezu gleich aus wie der in Abb. 6.14. Wir verzichten daher auf die Wiedergabe dieses fast identischen Schaubildes.

Nachstehend ist (mit einer gegenüber der vorherigen Abbildung abweichenden Skalierung der Funktionswerte) die Differenz der Ausgaben mit sigmoider Funktion und mit Rampenfunktion aufgetragen.

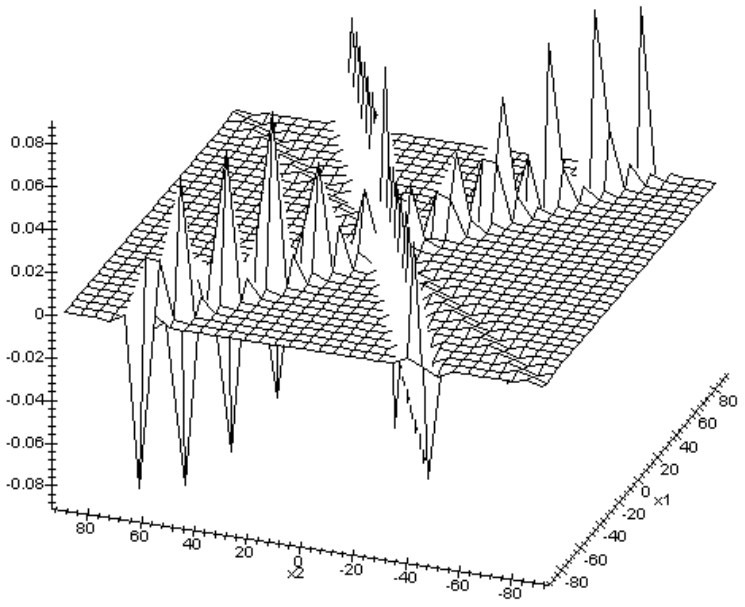


Abb. 6.15: Abweichung der Netzausgaben (Sigmoide vs. Rampenfunktion R_{x_0})

Die "Gebirge" dieser Abbildung zeigen die Bereiche, in denen die betragsmäßige Abweichung $\left| R_{x_0}(\overrightarrow{w_i} \cdot \overrightarrow{x}) - sigm(\overrightarrow{w_i} \cdot \overrightarrow{x}) \right|$ die größten Werte annimmt; dies ist dann der Fall, wenn die Funktionsargumente $\overrightarrow{w_i} \cdot \overrightarrow{x}$ im Kernbereich $[-x_0, x_0]$ liegen, die Eingabevektoren $\overrightarrow{x}$ also (nahezu) orthogonal zu jeweils einem der Gewichtsvektoren $\overrightarrow{w_i}$ stehen. (Wird mit α der Winkel zwischen $\overrightarrow{w_i}$ und $\overrightarrow{x}$ bezeichnet, so gilt $\overrightarrow{w_i} \cdot \overrightarrow{x} = \left|\overrightarrow{w_i}\right| \cdot \left|\overrightarrow{x}\right| \cdot \cos(\alpha)$.)

Zur weiteren Begründung dieses Sachverhaltes sei auf den Satz 6.8 verwiesen: die größten Abweichungen der Sigmoiden von der Rampenfunktion R_{x_0} befinden sich im Kernbereich $[-x_0, x_0]$ symmetrisch um die 0, wobei an der Stelle 0 selbst die Abweichung jedoch ebenfalls 0 beträgt.

Dies wird auch in der nachstehenden Abbildung illustriert, in der die (vorzeichenbehaftete) Differenz $sigm(\overrightarrow{w_i} \cdot \overrightarrow{x}) - R_{x_0}(\overrightarrow{w_i} \cdot \overrightarrow{x})$ aufgetragen ist.

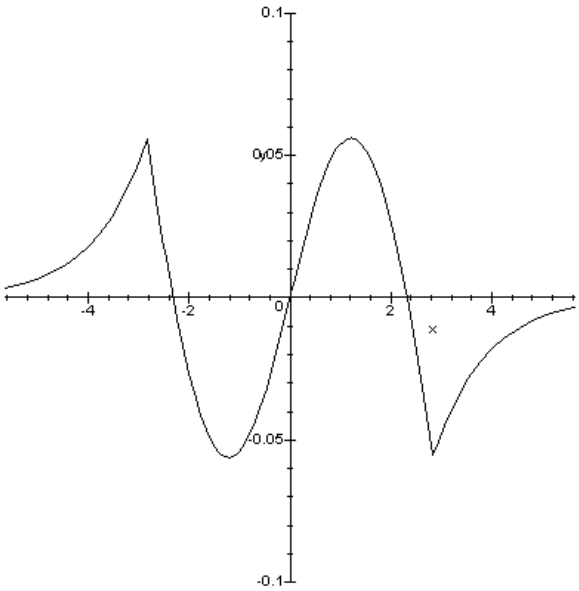


Abb. 6.16: Abweichung der Sigmoiden von der Rampenfunktion R_{x_0}

Bemerkung 6.28 (Qualitative Form der Ausgabe eines K-L-1-Netzes)

Es gilt generell, dass die Form der Ausgabe eines K-L-1-Netzes, welches die Rampenfunktion oder die Sigmoide als Ausgabefunktion im Hidden Layer verwendet, qualitativ so aussieht, wie in Abb. 6.14 exemplarisch gezeigt. Anschaulich gesprochen verfügt ein solches K-L-1-Netz über maximal *2L* Plateaus. Es sei noch einmal daran erinnert, dass im vorliegenden Kontext die Ausgabeneuronen mit der Identität als Ausgabefunktion wirken.

Unter einem solchen *Plateau* verstehen wir dabei - im Falle der Rampenfunktion - die Bereiche konstanter Funktionswerte der Netzausgabe; im Falle der Sigmoiden handelt es sich hierbei dementsprechend um Regionen, innerhalb derer die Netzausgabe noch innerhalb einer vorgegebenen Toleranz schwankt, da die Sigmoide die Funktionswerte 0 und 1 nicht annimmt, sondern sich diesen Werten nur asymptotisch nähert.

Die (maximale) Anzahl *2L* erklärt sich dadurch, dass durch Hinzunahme eines weiteren Hidden Neurons der Ausgaberaum orthogonal zum neuen Gewichtsvektor "durchschnitten" wird; hierdurch werden (im allgemeinen Fall) zwei der bisherigen Plateaus zerteilt und es entstehen auf diese Weise vier Plateaus. Im Höherdimensionalen entspricht ein solches Plateau einer Hyperebene.

Satz 6.29 (Punktsymmetrie der Netzausgabe)

Gegeben sei ein K-L-1-Netz mit der Ausgabefunktion *f* in der verborgenen Schicht; hierbei sei *f* die Sigmoide oder (eine) Rampenfunktion. Dann gilt für die Netzausgabefunktion *out*:

$$out(-\vec{x}) = \sum_{1 \le i \le L} v_i - out(\vec{x}).$$

Mit anderen Worten: Die Netzausgabefunktion ist punktsymmetrisch um $(\vec{0}, \frac{1}{2} \cdot \sum_{1 \le i \le L} v_i)$.

Beweis:

Für die hier betrachteten Ausgabefunktionen f gilt $f(x) = 1 - f(-x)$. Damit folgt:

$$out(-\vec{x}) = \sum_{1\le i\le L} v_i \cdot f(-\vec{w_i} \cdot \vec{x}) = \sum_{1\le i\le L} v_i \cdot (1 - f(\vec{w_i} \cdot \vec{x})) = \sum_{1\le i\le L} v_i - \sum_{1\le i\le L} v_i \cdot f(\vec{w_i} \cdot \vec{x})$$

$$= \sum_{1\le i\le L} v_i - out(\vec{x}) .$$

Insbesondere ist $out(\vec{0}) = \frac{1}{2} \cdot \sum_{1\le i\le L} v_i$.

❑

Bemerkung 6.30

Wie der obige Beweis zeigt, wird hier lediglich die Symmetrie der Funktion f um den Punkt (0, 1/2) benötigt.

Beispiel 6.31

Dieser Sachverhalt lässt sich exemplarisch in Abbildung 6.14 (sh. Beispiel 6.27 auf S. 126) gut nachvollziehen. Dort ist $\sum_{1\le i\le 4} v_i = -1$, und für die Funktionswerte der symmetrisch einander gegenüberliegenden Plateaus (i.e. +1,-2 sowie 0,-1 und -0.5, -0.5) gilt 1 = -1 - (-2), 0 = -1 - (-1) bzw. -0.5 = - 1 - (-0.5).

Beispiel 6.32

Wie in Bemerkung 1.35 (S. 39) bereits angesprochen, sind die hier betrachteten Neuronalen Netze keine universellen Approximatoren bezüglich der Maximumsnorm, da die vom Netz anzunähernde bzw. zu repräsentierende Ausgangsfunktion die entsprechenden Symmetrieeigenschaften nicht aufweisen muss. Daher ist "nur" eine Approximationsfähigkeit im L_p-Sinne (z.B. für p=2 im Sinne der mittleren quadratischen Abweichung) möglich.

Bemerkung 6.33

Betrachten wir statt eines K-L-1-Netzes eines vom Typ K-L-M, so besteht die Verallgemeinerung von Satz 6.29 darin, dass für jede Ausgabedimension andere Punktsymmetrien aufgebaut werden und sich die Varianten des qualitativ in Abbildung 6.14 gezeigten Verhaltens überlagern. Daher wenden wir uns im Folgenden wieder den Netzen vom Typ K-L-1 zu und treffen in den anschließenden Bemerkungen noch weitere Annahmen.

Bemerkung 6.34 (Normierung des Eingaberaumes oder der Gewichtsvektoren)

Da in den Ausgabefunktionen nur die Ausdrücke $\vec{w_i} \cdot \vec{x}$ eine Rolle spielen, können wahlweise die Gewichtsvektoren $\vec{w_i}$ oder die Eingaben $\vec{x}$ als normiert angenommen werden. Generell haben wir vorausgesetzt, dass der Eingaberaum aus einer kompakten Menge besteht.

Wegen der aufgezeigten Punktsymmetrie können wir weiterhin nicht nur annehmen, dass die Komponenten der Eingabevektoren x_i betragsmäßig durch 1 (nach oben) begrenzt werden,

wir können sogar voraussetzen, dass $x_i \in [0,1]$ gilt. Wir können also bei Bedarf o.B.d.A. davon ausgehen, dass der Eingaberaum eine Teilmenge von $[0,1]^K$ ist oder die Längen der Gewichtsvektoren $\overrightarrow{w_i}$ z.B. durch 1 beschränkt sind: $\left|\overrightarrow{w_i}\right| \leq 1$.

Zu einem gegebenen Netz mit reellwertiger Ausgabe kann auch der Ausgaberaum eingegrenzt bzw. abgeschätzt werden.

Satz 6.35 (Abschätzung der Netzausgabe)

Betrachten wir ein K-L-1-Netz mit den Gewichten v_i, $1 \leq i \leq L$, zum Ausgabeneuron. Die Ausgabefunktion f sei die Sigmoide oder (eine) Rampenfunktion.

Dann lässt sich der Wert der Netzausgabe wie folgt abschätzen:

$$\sum_{1\leq i\leq L,\, v_i<0} v_i \leq out(\overrightarrow{x}) = \sum_{1\leq i\leq L} v_i \cdot f(\overrightarrow{w_i} \cdot \overrightarrow{x}) \leq \sum_{1\leq i\leq L,\, v_i>0} v_i.$$

Sollte es keine Gewichte v_i mit $v_i < 0$ bzw. $v_i > 0$ geben, so ist der Wert der betreffenden Summe per definitionem 0.

Die Gleichheit kann dabei nur auftreten, wenn die Ausgabefunktion die Werte 0 und 1 auch annimmt, wie dies bei der Rampenfunktion der Fall ist. Im Falle der Sigmoiden handelt es sich um eine strikte Ungleichungsabschätzung.

Beweis:

Betrachten wir die Abschätzung der Netzausgabe nach oben.

Da die Ausgabefunktion f nur nichtnegative Werte annimmt, können wir zunächst dadurch nach oben abschätzen, dass wir die negativen Summenterme weglassen:

$$out(\overrightarrow{x}) = \sum_{1\leq i\leq L} v_i \cdot f(\overrightarrow{w_i} \cdot \overrightarrow{x}) \leq \sum_{1\leq i\leq L,\, v_i>0} v_i \cdot f(\overrightarrow{w_i} \cdot \overrightarrow{x}).$$

Mit $f \leq 1$ folgt

$$\sum_{1\leq i\leq L,\, v_i>0} v_i \cdot f(\overrightarrow{w_i} \cdot \overrightarrow{x}) \leq \sum_{1\leq i\leq L,\, v_i>0} v_i,$$

insgesamt also die behauptete Abschätzung nach oben.

Die Abschätzung der Netzausgabe nach unten folgt in analoger Weise.

❑

Bemerkung 6.36

Die Aussage von 6.35 lässt sich in naheliegender Weise auch für dreischichtige Feedforward-Netze mit mehr als einem Ausgabeneuron (K-L-M-Netze) formulieren; hier gilt die betreffende Ungleichung für jede Dimension separat.

Bemerkung 6.37 (Plateaus der Netzausgabe)

Die in Abb. 6.14 beispielhaft dargestellten Plateaus führen zu einer theoretisch sehr einfachen Möglichkeit der Regelextraktion. Für jeden Plateaubereich kann prinzipiell eine Regel der Form

$$IF\ \vec{x}\ IN\ Plateau_j\ THEN\ Ausgabe\ =\ Plateauwert_j$$

aufgestellt werden. Da es bei einem K-L-1-Netz (maximal) *2L* Plateaus gibt, wäre das Verhalten des Netzes in den Plateaubereichen - also für betragsgroße Eingabevektoren - mit ebenfalls *2L* Regeln zu beschreiben.

Allerdings wird in der Praxis generell der Bereich "nahe der 0" interessanter sein als das Gebiet außerhalb, da die Ausgabe des Neuronalen Netzes "nahe der 0" seine Trennschärfe demonstriert.

Über eine weitere Möglichkeit, die Anzahl der Neuronen in der verborgenen Schicht zu reduzieren, und die damit verbundene Güte-Aussage gibt der folgende Satz Auskunft. Hierbei werden - anders als bei den zuvor vorgestellten Cluster-Verfahren - gewisse Neuronen der Hidden-Schicht entfernt und keine neuen Neuronen gebildet.

Satz 6.38 (Least Weight Algorithmus (LWA) zur Reduktion der Neuronenanzahl)

Gegeben sei ein $K-L-1$-Netz; o.B.d.A. seien die Gewichte v_i zur Ausgabeschicht betragsmäßig absteigend sortiert, d.h. für deren Beträge gelte $|v_1| \geq |v_2| \geq |v_3| \geq ... \geq |v_L|$.
Ferner sei $1 \leq L' < L$ und die Ausgabefunktion der Hidden Neuronen, die Rampenfunktion oder die Sigmoide, werde wiederum mit *f* bezeichnet.

Dann können die $L-L'$ Neuronen in der verborgenen Schicht, die zu den in diesem Sinne $L-L'$ betragsmäßig kleinsten Gewichten gehören, entfernt werden und man erhält ein approximierendes $K-L'-1$-Netz, dessen Ausgabe um höchstens

$$\max\ \{\sum_{L'+1\leq i\leq L,\ v_i>0} v_i\ ,\ -(\sum_{L'+1\leq i\leq L,\ v_i<0} v_i)\}$$

von derjenigen des Ausgangsnetzes mit *L* Hidden Neuronen abweicht.

Beweis:

Die Ausgaben der Netze mit L bzw. L' Neuronen in der Hidden Schicht zu dem Eingabevektor $\vec{x}$ seien $out_L(\vec{x})$ bzw. $out_{L'}(\vec{x})$.
Damit sind $out_L(\vec{x}) = \sum_{1\leq i\leq L} v_i \cdot f(\vec{w_i} \cdot \vec{x})$ und $out_{L'}(\vec{x}) = \sum_{1\leq i\leq L'} v_i \cdot f(\vec{w_i} \cdot \vec{x})$.

Es folgt (unter Berücksichtigung, dass *f* nur Werte zwischen 0 und 1 annimmt):

$$\left|out_L(\vec{x}) - out_{L'}(\vec{x})\right| = \left|\sum_{L'+1\leq i\leq L} v_i \cdot f(\vec{w_i} \cdot \vec{x})\right| =$$

$$\left|\sum_{L'+1\leq i\leq L,\ v_i>0} v_i \cdot f(\vec{w_i} \cdot \vec{x}) + \sum_{L'+1\leq i\leq L,\ v_i<0} v_i \cdot f(\vec{w_i} \cdot \vec{x})\right| \leq$$

$$\max\ \{\sum_{L'+1\le i\le L,\ v_i>0} v_i \cdot f(\overrightarrow{w_i}\cdot\overrightarrow{x}),\ -(\sum_{L'+1\le i\le L,\ v_i<0} v_i \cdot f(\overrightarrow{w_i}\cdot\overrightarrow{x}))\} \le$$
$$\max\ \{\sum_{L'+1\le i\le L,\ v_i>0} v_i,\ -(\sum_{L'+1\le i\le L,\ v_i<0} v_i)\},$$ w.z.b.w.

❑

Bemerkung

Formal wäre im obigen Satz auch $L' = L$ möglich; in diesem Fall wäre das "Resultat" gerade wieder das Ausgangsnetz.

Das Resultat dieses Satzes soll zunächst an einer einfachen Skizze (Beispiel 6.39) veranschaulicht werden; anschließend wird in 6.42 eine numerische Gegenüberstellung dieses Verfahrens mit den zuvor präsentierten Methoden präsentiert.

Beispiel 6.39 (Veranschaulichung)

Die nachstehenden Abbildungen illustrieren exemplarisch das Vorgehen des beschriebenen LWA-Verfahrens. Ausgangspunkt ist ein 2-4-1-Netz mit den Gewichtsvektoren

$$\overrightarrow{w_1} = (0.5, 0.75),\ \overrightarrow{w_2} = (0.4, -0.2),\ \overrightarrow{w_3} = (-0.3, 0.2),\ \overrightarrow{w_4} = (0.1, 0.8)$$

von der Eingabe- zur Hidden-Schicht und den Gewichten

$$v_1 = 1,\ v_2 = 2,\ v_3 = -1,\ v_4 = -3$$

zur Ausgabeschicht.

Die Ausgabe dieses einfachen Netzes ist in der nachstehenden Abbildung dargestellt.

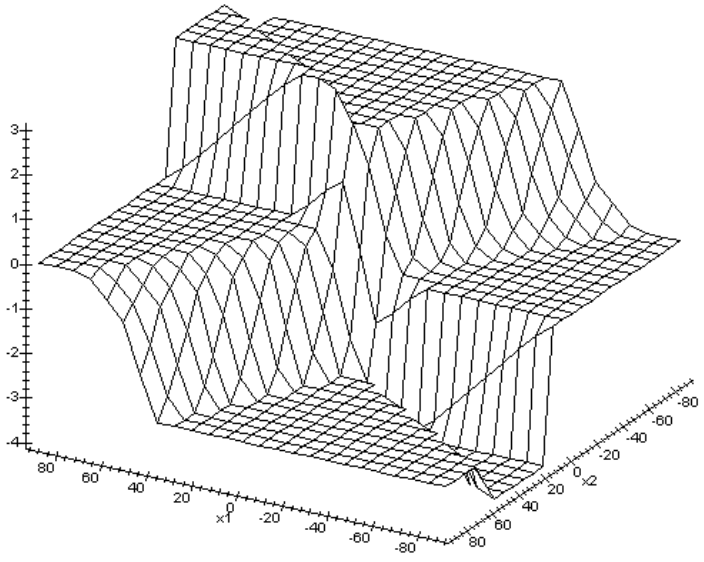


Abb. 6.17: Ausgabe des Netzes mit vier Neuronen in der verborgenen Schicht

Gemäß dem Algorithmus wird nun das (bzw. ein Neuron) mit dem betragsmäßig kleinsten Gewicht zur Ausgabeschicht aus dem Netz entfernt; hier sei dies das Neuron 1 mit $v_1 = 1$.

Das resultierende Netz verfügt noch über drei verborgene Neuronen; seine Ausgabe ist nachstehend abgebildet.

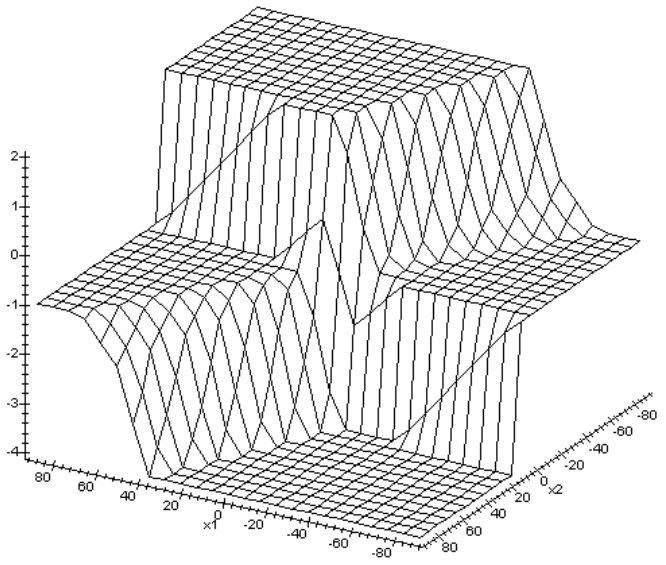


Abb. 6.18: Ausgabe des modifizierten Netzes mit noch drei Neuronen in der verborgenen Schicht

Man erkennt, dass sich der Maximalwert der Netzausgabe konsequenterweise von +3 zu +2 verändert hat.

Im nächsten Reduktionsschritt wird ein weiteres Neuron entfernt, nun das mit dem Gewicht $v_3 = -1$. (Die Nummerierung der Gewichte wird aus Gründen der besseren Verständlichkeit in Bezug auf das ursprüngliche Netz beibehalten.)

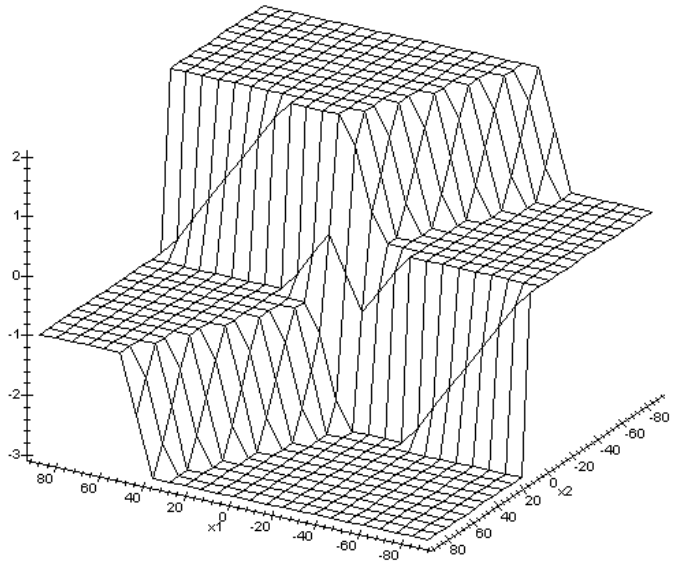


Abb. 6.19: Ausgabe des Netzes mit zwei Neuronen in der verborgenen Schicht

Es ist zu erkennen, dass die minimale Ausgabe des modifizierten Netzes nun bei -3 liegt.

Reduziert man noch weiter, so erhält man das Netz mit der nachfolgend gezeigten Ausgabe; es wurde das verborgene Neuron mit dem Gewicht $v_2 = 2$ entfernt. Mit nur noch einem Neuron in der Hidden Schicht liegen lediglich noch zwei Plateaus vor: hier bei 0 und -3.

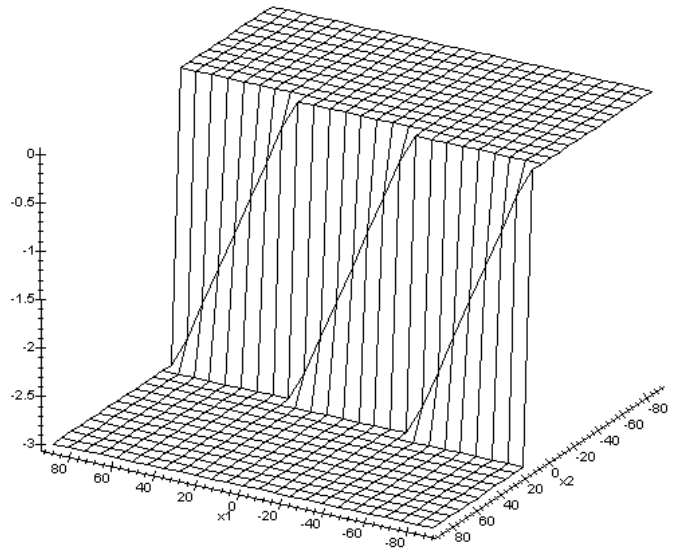


Abb. 6.20: Ausgabe des 2-1-1-Netzes mit nur noch einem Neuron in der verborgenen Schicht

Bemerkung 6.40

Beispiel 6.39 zeigt, dass sich bei jedem Schritt, in dem ein Neuron der verborgenen Schicht mit dem Gewicht v_i zur Ausgabeschicht entfernt wird, die Güte auf dem Eingabe-Halbraum $\{\vec{x} \mid \vec{x} \cdot \vec{w_i} > 0\}$ um maximal v_i (betragsmäßig) verschlechtert, während auf dem komplementären Halbraum nach wie vor die ursprüngliche Ausgabe erzielt wird. Möchte man stattdessen die *maximale* Abweichung über dem gesamten Eingaberaum minimieren, dann kann die gesamte Netzausgabe um $\frac{1}{2}v_i$ verschoben werden.

Bemerkung 6.41

Da beim Least Weight Algorithmus keine neuen Neuronen gebildet werden, vereinfacht sich hier die Netzstruktur gegenüber der Ausgangssituation. Dies ist für praktische numerische Implementierungen von Vorteil.

Numerisches Beispiel 6.42 (Inverses Pendel mit LWA)

Auch der Least Weight Algorithmus soll mit dem Beispiel des Inversen Pendels numerisch betrachtet werden. Aufsetzend auf Beispiel 6.26 werden zusätzlich zu den bisher diskutierten Verfahren LWA-Netze mit 8 bzw. 16 Neuronen in der Hidden Schicht generiert. Deren Güte (hinsichtlich des mittleren quadratischen Fehlers (MSE) der Abweichung von der angestrebten Soll-Ausgabe) wird in Anhang A3 (S. 181) exemplarisch für fünfzig Durchläufe aufgelistet und den Werten der anderen Netze gegenübergestellt.

Dabei zeigt sich bei dem Vergleich des aus dem Least Weight Algorithmus generierten Netzes mit acht Hidden Neuronen (LWA-2-8-1-Netz) mit dem korrespondierenden Netz mit 16 Neuronen in der verborgenen Schicht (LWA-2-16-1-Netz), dass erwartungsgemäß die Güte im letzteren Fall deutlich besser ist. Die Zahl 16 wurde mit Blick auf die konkrete Darstellung des Inversen-Pendel-Problems gewählt (vgl. 6.21, S. 121).

Da der Least Weight Algorithmus lediglich die Gewichte zwischen den Hidden Neuronen und den Ausgabeneuronen berücksichtigt, die Gestalt der konkreten Ausgabefunktion dabei unberücksichtigt lässt, ist weiterhin festzustellen, dass die Güte der LWA-Netze in den meisten Fällen etwas besser ist, wenn die Sigmoide verwendet wird anstelle der Rampenfunktion.

Bemerkung 6.43

Es ist offensichtlich (und auch beim vorherigen numerischen Beispiel zu beobachten), dass der Least Weight Algorithmus grundsätzlich für eine wachsende Anzahl Hidden Neuronen $L'(\leq L)$ sukzessive bessere Approximationen an das Ausgangsnetz liefert. Pragmatisch kann auf diese Weise in einem konkreten Anwendungsfall geprüft werden, für welches L' (erstmals) ein LWA-Netz einer gewissen vorgegebenen Güte generiert wird.

Bemerkung 6.44 (Relevanzanalyse)

Der Least Weight Algorithmus 6.38 arbeitet nach dem Ansatz, diejenigen Neuronen der Hidden Schicht zu eliminieren, deren Gewichte zur Ausgabeschicht betragsmäßig am kleinsten sind. Dieses Vorgehen wird von der Überlegung getragen, dass i.a. dies die Hidden Neuronen sein werden, deren Beitrag zur Gesamtausgabe am ehesten zu vernachlässigen ist.

Dabei wird allerdings nicht berücksichtigt, welche Gewichte an den Verbindungen von der Eingabeschicht zu einem solchen Hidden Neuron anliegen. Die "Relevanz" eines Neurons der verborgenen Schicht hängt bei einem konkreten Eingabevektor $\vec{x}$ faktisch nicht nur von dem Gewicht v zur Ausgabeschicht ab, sondern von dem Ausdruck $v \cdot f(\vec{w} \cdot \vec{x})$. (Zur einfacheren Lesbarkeit wurde hier auf die Indizierung der Gewichte verzichtet, da wir nur ein einzelnes Neuron der Hidden Schicht betrachten.)

Als sog. "Relevanzanalyse" könnte nun versucht werden, auf der Grundlage der Ausdrücke $v \cdot f(\vec{w} \cdot \vec{x})$ zu entscheiden, welches die "am wenigsten wichtigen" Neuronen in der Hidden Schicht sind, welche also am wenigsten zu der Gesamtausgabe beitragen (und diese im Sinne einer Vereinfachung ggf. zu entfernen).

Diese Vorgehensweise basiert jedoch auf der Kenntnis des speziellen Eingabevektors $\vec{x}$, so dass zunächst keine von diesem unabhängige Eliminierung von Neuronen der Hidden Schicht möglich wäre; diese könnte sich dann beispielsweise auf die Mittelung der oben erwähnten Relevanzen über eine repräsentative, endliche Menge von Eingabevektoren stützen. In der Praxis gibt es jedoch Verfahren, die sich auf konkrete Eingabedaten beziehen, vgl. hierzu Abschnitt 5.5 (S. 96).

Bemerkung 6.45

Bei den bisherigen Betrachtungen wurden fast ausschließlich K-L-1-Feedforward-Netze betrachtet, also Netze mit einer eindimensionalen Ausgabe(funktion). Es stellt sich die Frage, welche Resultate in kanonischer Form auf den höherdimensionalen Fall (K-L-M-Netze) übertragen werden können.

Ein solcher Transfer wäre sicherlich dann möglich, wenn es gelänge, ein (beliebiges) K-L-M-Netz in semantisch äquivalenter Form als ein "vektorwertiges" K-L-1-Netz aufzufassen. Dass dies jedoch nicht funktioniert, das klärt der nachfolgende Satz.

Satz 6.46

Es existiert keine semantikerhaltende Bijektion zwischen der Menge $N(K,L,M,\mathbb{R})$ der reellwertigen K-L-M-Netze und der Menge $N(K,L,1,\mathbb{R}^M)$ der vektorwertigen K-L-1-Netze mit M-dimensionaler Ausgabefunktion.

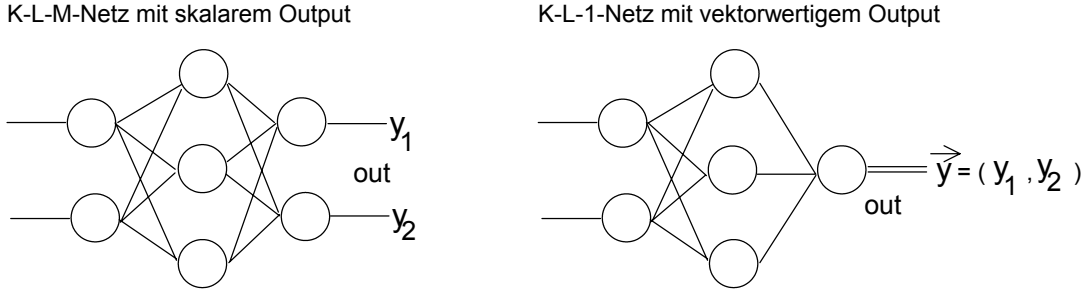


Abb. 6.21: Gegenüberstellung eines K-L-M-Netzes mit skalarem Output
und eines vektorwertigen K-L-1-Netzes für den Fall K=1, L=3 und M=2

Beweis:

Gehen wir von einem beliebigen K-L-M-Netz aus und versuchen die Konstruktion eines hierzu äquivalenten vektorwertigen K-L-1-Netzes.

Bezeichnen wir im K-L-M-Netz mit $\overrightarrow{w_i}$ den Gewichtsvektor zum i-ten Hidden Neuron und mit $\overrightarrow{v_j}$ den Vektor der Gewichte zum j-ten Ausgabeneuron; entsprechend sei $\overrightarrow{w_i^*}$ der Gewichtsvektor zum *i*-ten Hidden Neuron im vektorwertigen K-L-1-Netz, mit v_j^* werde das (reellwertige) Gewicht vom *j*-ten Hidden Neuron zum Ausgabeneuron bezeichnet. Mit *f* werde die Ausgabefunktion bezeichnet; diese sei wie üblich entweder die Sigmoide oder die Rampen- Funktion.

Semantisch äquivalent bedeutet, dass die Ausgaben der beiden Netze für alle Eingabevektoren $\overrightarrow{x}$ und für jedes *j* mit $1 \le j \le M$ gleich sind. Das heißt:

$$\forall \overrightarrow{x} \in \mathbb{R}^K \; \forall j : 1 \le j \le M \; : \; v_{j1} f(\overrightarrow{w_1} \cdot \overrightarrow{x}) + \ldots + v_{jL} f(\overrightarrow{w_L} \cdot \overrightarrow{x}) = v_1^* f(\overrightarrow{w_1^*} \cdot \overrightarrow{x}) + \ldots + v_L^* f(\overrightarrow{w_L^*} \cdot \overrightarrow{x}) \; .$$

Für jede der hier betrachteten Ausgabefunktionen ist f(0) = 0.5. Speziell für $\vec{x} = \vec{0}$ bedeutet dies daher: $\forall j : 1 \leq j \leq M : \sum_{1 \leq l \leq L} v_{jl} = \sum_{1 \leq l \leq L} v_l^*$, d.h. die Summe der Komponenten der Gewichtsvektoren $\vec{v_j}$ im K-L-M-Netz müsste für jedes *j* gleich sein. Das hieße: für jedes Neuron im Hidden Layer des (skalaren) K-L-M-Netzes müssten alle davon abgehenden Verbindungen dieselben Gewichte besitzen. Dies ist i.a. natürlich nicht der Fall.
Dies ist somit ein Widerspruch dazu, dass wir von einem beliebigen K-L-M-Netz ausgegangen sind, somit kann eine semantisch äquivalente Bijektion zwischen diesen Netz-Mengen nicht existieren.

❑

Bemerkung

1. Wie die Argumentation des Beweises zeigt, kann ein vektorwertiges K-L-1-Netz als (skalares) K-L-M-Netz aufgefasst werden, die Umkehrung ist jedoch (i. a.) falsch.
2. Der obige Satz präzisiert Bemerkung 6.1 von S. 107. Dort wurde davon gesprochen, dass ein K-L-M-Netz als "Summe" von *M* "eindimensionalen" K-L-1-Netzen interpretiert werden kann. Diese *M* K-L-1-Netze sind aber grundsätzlich unabhängig voneinander, denn die Gewichte zu den Ausgabeneuronen können und werden im allgemeinen Fall unterschiedlich sein.

6.3 Zusammenfassung

In diesem Kapitel wurden für dreischichtige Feedforward-Netze eine Reihe von Ansätzen zur numerischen Approximation vorgestellt. Zunächst wurde für die in der Praxis häufig eingesetzte Sigmoide (logistische Funktion) mit dem Konzept der Rampenfunktion eine einfacher zu berechnende Annäherung präsentiert. Es sei noch einmal darauf hingewiesen, dass zahlreiche der vorgelegten Resultate auch für andere Ausgabefunktionen neben der Sigmoiden übertragbar sind.

Im Anschluss wurden verschiedene Algorithmen entwickelt, die zu einem gegebenen Netz ein approximierendes Netz einfacherer Struktur liefern: die elementaren Substitutionssätze, die nur ein oder zwei Neuronen in der verborgenen Schicht generieren, sowie die Clusterbildungsverfahren und der Least Weight Algorithmus, die eine beliebige Zahl von Hidden Neuronen zulassen. Damit sind, wie auch die exemplarischen numerischen Betrachtungen unterstrichen haben, diese Verfahren für die Praxis als Approximation existierender Feedforward-Netze flexibel einsetzbar, da zu einer vorgegebenen Güte-Schranke ein entsprechendes Näherungsnetz konstruiert werden kann.

En passant wurden in diesem Kapitel einige Resultate vorgestellt, die allgemeine Aussagen zur Ausgabe von K-L-1-Netzen formulieren und illustrieren, etwa das qualitative Ausgabeverhalten, eine quantitative Abschätzung oder der interessante Aspekt der Punktsymmetrie.

7 Verfahren zur Regelextraktion

Dieses Kapitel befasst sich mit der Formulierung konkreter Verfahren zur Extraktion von Regeln aus Neuronalen Netzen. Das operative Ziel ist jeweils, aus einem gegebenen Neuronalen Netz eine Regelbasis zu extrahieren, die im Idealfall das Netz semantisch äquivalent repräsentiert und dieses einem Menschen hinsichtlich seiner Arbeitsweise verständlicher macht.

In der Praxis sieht man jedoch sehr schnell ein, dass eine *exakte* semantische Äquivalenz durch - womöglich sogar nur einige wenige - Regeln generell nicht erzielt werden kann. Aus diesem Grund wird von der Forderung nach strenger semantischer Äquivalenz abgewichen; Ziel ist im Folgenden die *numerische Approximation* von Neuronalen Netzen durch eine entsprechende Regelbasis (vgl. dazu Abschnitt 5.4 auf Seite 94).

Der Prognose von Tickle, Andrews, Golea und Diederich folgend (siehe hierzu auf S. 97 die entsprechende Bemerkung zu [Tick1998]), fokussieren wir dabei auf reellwertige Netz-Ausgaben und betrachten auch Netze mit reellwertigen Gewichten.

Wir sprechen im konkreten Kontext über die bislang bereits behandelten dreischichtigen Feedforward-Netze; einige der hier betrachteten Regelextraktionsmechanismen greifen aufgrund ihrer numerischen Methodik jedoch teilweise auch für andere Netze, insbesondere die von Tenhagen [Tenh2000] und Niendieck [Nien1998] behandelten speziellen Vier-Schicht-Netze.

Um Regelextraktionen für die Praxis handhabbar zu machen, ist es sinnvoll, die Anzahl der generierten Regeln beeinflussen zu können. Im Folgenden wird diese Anzahl als Stellgröße r dem (jeweiligen) Algorithmus mitgegeben werden.

Da es in der Praxis stets auch um eine Software-Implementierung geht, wird der Fokus auf eine Reduktion der Komplexität gelegt. Wir behandeln daher zunächst wieder K-L-1-Netze, betrachten also Netze, die eine K-dimensionale Eingabe entgegennehmen, L Neuronen in einer verborgenen Schicht besitzen, und die in der dritten Schicht über ein Ausgabeneuron verfügen. Als Ausgabefunktion der verborgenen Schicht wird wahlweise die sigmoide Funktion $x \to \frac{1}{1+e^{-x}}$ oder die Rampenfunktion R_{x_0} verwendet.

7.1 Regelextraktion für skalare Eingaberäume

Zunächst betrachten wir den einfachsten Fall: Neuronale Netze mit einer skalaren Eingabe, d.h. wir setzen vorerst K=1. Weiterhin wird der Eingabebereich ohne Einschränkung der praktischen Anwendung auf ein kompaktes Intervall $[x_{min},x_{max}]$ begrenzt.

Die generierten Regeln werden in zwei dualen Formen verwaltet; einmal wird eine Punkt-Darstellung der Art “IF x NEAR a THEN y NEAR b” angegeben, zum zweiten wird aber auch eine (reelle) Intervall-Darstellung “IF x IN (a_1,a_2) THEN y IN (b_1,b_2)” mitgeführt.

Die erste Darstellungsform lässt damit vom Ansatz her auch Fuzzy-Regeln zu, wenn a und b Fuzzy-Zahlen sind; die zweite Darstellung repräsentiert deutlicher die numerische Behandlung im Kontext reeller Zahlen.

Am Rande sei angemerkt, dass es numerisch auf die Zuordnung der Intervallränder zum linken oder rechten angrenzenden Intervall in der Praxis naturgemäß nicht ankommt.

Wir wollen nun zwei Regelextraktionsalgorithmen präsentieren, die eine vorgegebene Anzahl *r* von Regeln aus einem Neuronalen Netz generieren.

Bemerkung 7.1 (Regelextraktionsverfahren Äquidistante Unterteilung)

Das definierte Eingabe-Intervall $[x_{min},x_{max}]$ wird in *r* gleichgroße Teilintervalle T_i mit Mittelpunkt $x_{i,mittel}$ zerlegt. Über jedem dieser Teilintervalle wird (mittels einer für die numerischen Berechnungen vordefinierten Anzahl von Stützstellen) der Ergebnis-Mittelwert $y_{i,mittel}$ berechnet. Parallel dazu werden der kleinste und der größte auftretende Ergebniswert $y_{i,min}$ bzw. $y_{i,max}$ des Netzes auf diesem Teilintervall festgestellt.

Die *r* erzeugten Regeln ordnen nun die jeweiligen x- und y-Mittelwerte bzw. die x- und y-Intervalle zu: "IF *x* NEAR $x_{i,mittel}$ THEN *y* NEAR $y_{i,mittel}$" bzw. "IF *x* IN T_i THEN *y* IN $(y_{i,min},y_{i,max})$".

Als mögliche Verbesserung des soeben vorgestellten Verfahrens bietet es sich an, die Schwankung der Netzausgabe zu berücksichtigen. Das Neuronale Netz stellt eine Abbildungsvorschrift zwischen Ein- und Ausgaben vor; diese Funktion nennen wir der Einfachheit halber *f*. Das heißt: dort, wo *f* einen flachen Verlauf aufweist, kann ein größeres x-Intervall in der Prämisse einer Regel eingesetzt werden; variiert *f* in einem Bereich jedoch stark, dann ist dort eine Unterteilung in feinere x-Intervalle sinnvoll.

In der folgenden Definition betrachten wir die "Schwankung" einer solchen Funktion *f* über dem x-Intervall bezogen auf eine vorgegebene Partition.

Definition 7.2 (Numerische Variation)

Es seien das x-Intervall $[x_{\min}, x_{\max}]$, eine Funktion $f: [x_{\min}, x_{\max}] \to \mathbb{R}$ sowie eine Partitionierung $P = \{ x_1, ..., x_p \}$ dieses x-Intervalls gegeben.

Die *numerische Variation v(f,P)* von f über dem x-Intervall und bezogen auf die Partitionierung P wird definiert als

$$v(f,P) := \sum_i |f(x_{i+1}) - f(x_i)|,\ P = \{x_i | x_1 = x_{\min} \le x_2 \le ... \le x_p = x_{\max}\}.$$

Bemerkung 7.3

In die formale Notation der numerischen Variation wird das betreffende x-Intervall aus Gründen der Übersichtlichkeit nicht mit aufgenommen, da in unserem Kontext dieser Definitionsbereich als fest vorgegeben angenommen wird.

Beispiel 7.4

Bei einer monotonen Funktion *f* ist *v(f,P)* gerade $|f(x_{min})-f(x_{max})| = y_{max}-y_{min}$. ($y_{max}$ und y_{min} seien der größte bzw. kleinste Funktionswert über dem x-Intervall.)

Bemerkung 7.5

Denkbar ist zwar, dass die numerische Variation für eine nicht-monotone Funktion bei einer speziellen Partition kleiner (!) wird als $y_{max}-y_{min}$, gerade dann nämlich, wenn die Partitionierung gewisse monotone Teilstücke der Funktion "übersieht"; im allgemeinen wird jedoch die numerische Variation bei geeigneten Partitionen umso größer sein als der Wert $y_{max}-y_{min}$, desto mehr die Funktion "schwankt".

Bemerkung 7.6 (Regelextraktionsverfahren Numerische Variation)

Anhand der numerischen Variation wird nun in Analogie zum monotonen Fall von x_{min} ausgehend begonnen, das erste von *r* *x*-Teilintervallen zu finden, auf denen sich die (Netz-)Funktion (im Sinne dieser Variation) um $\frac{1}{r}v(f,P)$ geändert hat. D.h. für das erste Teilintervall bestimmen wir das kleinste *j* mit der Eigenschaft $\sum_{i\leq j} |f(x_{i+1})-f(x_i)| \geq \frac{1}{r}v(f,P)$.

Die nachfolgenden *x*-Teilintervalle werden entsprechend gefunden, sobald der entsprechende Anteil der numerischen Variation, also $\frac{i}{r}v(f,P)$, erreicht oder erstmals überschritten wurde. Das letzte Teilintervall ergibt sich hierbei automatisch.

Auf jedem dieser Teilintervalle wird nun wie beim vorherigen Regelextraktionsverfahren die Mittelwertbildung vorgenommen.

Bemerkung 7.7

Dieses Verfahren wird im nachstehenden Bild kurz an einem sehr einfachen Beispiel illustriert.

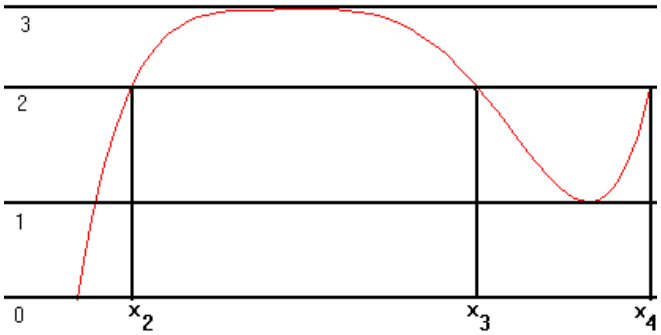


Abb. 7.1: Darstellung des Extraktionsprinzips mittels numerischer Variation

Beispiel: Die Netzausgabefunktion *f* soll verlaufen wie im obigen Bild gezeigt; dann ist die numerische Variation über einer Partition, die die relevanten Punkte beinhaltet, 6. (Denn von dem Wert 0 ausgehend erfolgt zunächst ein Anstieg auf 3, dann ein Abstieg auf 1, sodann wiederum ein Anstieg auf den Wert 2.)

Sollen drei Regeln extrahiert werden, so wird über der Partition geprüft, wann ein Drittel der numerischen Variation, hier also der Betrag 2, erreicht ist. Dies ist bei x_2 der Fall. Sofern die Partition den Maximumspunkt mit dem Funktionswert 3 umfasst, ist x_3 der nächste Punkt, bei dem eine weitere Schwankung um den Betrag 2 festgestellt wird. Hierbei wird natürlich wieder angenommen, dass auch dieser Punkt zur betrachteten Partition gehört.
Schließlich ergibt sich das letzte Teilintervall als das für die dritte Regel erforderliche Prämissenintervall.

Bemerkung 7.8

Der Vorteil des Numerischen Variationsverfahrens gegenüber dem Äquidistanzverfahren ist, dass i.a. keine gleichgroßen x-Intervalle (und damit Prämissen-Bereiche) formuliert werden; dort, wo die Netz-Funktion starke Schwankungen aufweist, wird feiner unterteilt werden, und hier werden auch mehr Regeln generiert als in einem Bereich mit wenig Schwankung der Ausgabefunktion des Netzes.

Bemerkung 7.9

Die beiden vorgestellten Verfahren setzen in ihrer allgemeinen Formulierung keine besondere Ausgabefunktion der Neuronen voraus, lediglich ihre numerische Approximationsgüte wird von dem Funktionsverhalten beeinflusst. Demgegenüber lassen sich zu einer konkreten Ausgabefunktion naheliegenderweise angepasste, optimierte Regeln formulieren.

So kann für die Rampenfunktion berücksichtigt werden, dass für Argumente außerhalb des Kernbereiches $[-x_0, x_0]$ die Funktionswerte konstant sind (vgl. dazu Definition 6.6 auf S. 111).

Bemerkung 7.10

Da der Funktionsverlauf innerhalb des Kernbereiches linear ist, bietet es sich weiterhin an, die Regeldarstellung flexibler zu gestalten. Anstatt eine Approximation durch "Treppenstufen" durchzuführen, wie es bei Regeln der Art "IF x IN (a,b) THEN y = y_0" der Fall ist, kann es hier empfehlenswert sein, eine Formulierung "IF x IN (a,b) THEN y = cx+d" zuzulassen.

Abstrahierend vom konkreten Beispiel der Rampenfunktion erhebt sich die Frage, inwieweit Regeln eines allgemeineren Typs "IF x IN (a,b) THEN y = h(x)" für bestimmte Funktionen h als sinnvoll erachtet werden. Selbstverständlich ginge diese Verallgemeinerung einher mit einer Erhöhung der Komplexität. Es sei jedoch daran erinnert, dass die Fuzzy-Controller nach dem Konzept von Sugeno und Takagi (sh. 2.76 auf S. 72) ebenfalls einen solchen Regeltypus zulassen.

7.2 Regelextraktion für höherdimensionale Eingaberäume

Im Falle höherdimensionaler Eingaberäume ist zunächst zu klären, wie die verschiedenen Eingabedimensionen miteinander gekoppelt werden sollen. Das klassische Modell bei einer K-dimensionalen Eingabe sind (kompakte oder zumindest beschränkte) Rechteckmengen (bzw. "Quader") der Form $[x_{1,1}, x_{1,2}] \times ... \times [x_{K,1}, x_{K,2}]$.

Wie in Bemerkung 6.37 (S. 131) bereits ausgeführt, weisen die hier betrachteten Netze eine typische Plateaubildung auf (für betragsgroße Eingabevektoren). Zu einem K-L-1-Netz können auf diese Weise die maximal 2L Plateaus recht einfach mit jeweils einer Regel beschrieben werden.

Im Bereich "nahe der 0" finden dagegen die deutlichen Veränderungen der Netzausgabefunktion statt. Daher konzentrieren wir uns im Folgenden auf einen kompakten Eingabebereich für das Neuronale Netz.

Numerisches Beispiel 7.11

Nachstehend wird zur Veranschaulichung die Netzausgabe eines exemplarischen 2-6-1-Netzes über dem Bereich $[-25,+25] \times [-25,+25]$ unter Verwendung der sigmoiden Ausgabefunktion im Hidden Layer dargestellt. Die sechs Gewichtsvektoren von der Eingabeschicht zu den Hidden Neuronen sind hierbei:

$$\overrightarrow{w_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \overrightarrow{w_2} = \begin{pmatrix} 0,2 \\ -0,6 \end{pmatrix}, \overrightarrow{w_3} = \begin{pmatrix} 0,7 \\ 1 \end{pmatrix}, \overrightarrow{w_4} = \begin{pmatrix} 0,3 \\ 0,25 \end{pmatrix}, \overrightarrow{w_5} = \begin{pmatrix} 0,8 \\ 1 \end{pmatrix}, \overrightarrow{w_6} = \begin{pmatrix} 0,5 \\ 1 \end{pmatrix};$$

die Gewichte zum Ausgabeneuron lauten:

$$v_1 = 0,2,\ v_2 = -0,3,\ v_3 = 0,2,\ v_4 = -0,3,\ v_5 = 0,3,\ v_6 = -0,1.$$

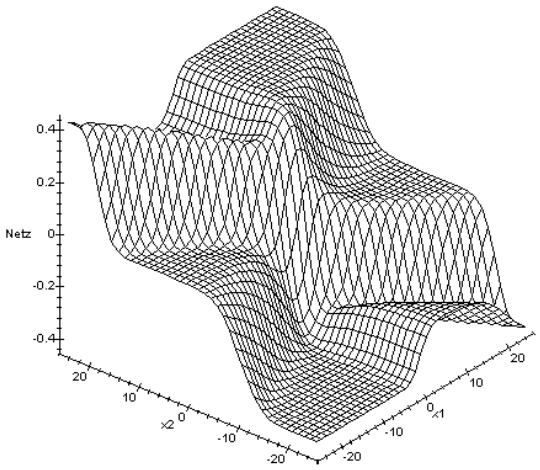


Abb. 7.2: Netzausgabe des 2-6-1-Netzes

Die nächste Abbildung zeigt einen Höhenlinien-Plot der Netzausgabe. Deutlich zu erkennen ist in beiden Darstellungen die Punktsymmetrie der Netze (vgl. hierzu Satz 6.29, S. 128); in dem konkreten Fall ist der Symmetriepunkt der Ursprung, $((0,0),0)$.

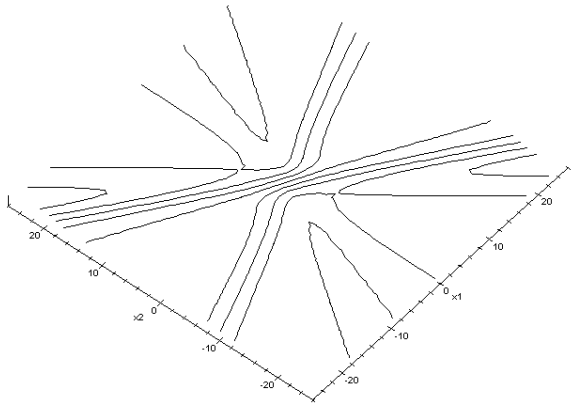


Abb. 7.3: Höhenlinien-Plot der Netzausgabe des 2-6-1-Netzes

Auf dieses Beispiel werden wir in Anschluss an die Formulierung des nächsten Satzes noch eingehen.

Bemerkung 7.12

Der anschließende Satz beschreibt ein Regelextraktionsverfahren für den Fall höherdimensionaler Eingaberäume. Das zuvor bereits mehrfach erwähnte Beispiel des Inversen Pendels (vgl. 6.21, S. 121) ist ein entsprechender Anwendungsfall, denn dort liegt ein zweidimensionaler Eingaberaum vor.

Für das Weitere stellen wir die (naheliegende) Notation der Partitionierung von Quadern (bzw. verallgemeinert "Hyper-Quadern") bereit; damit wird die Zerlegung eines Quaders im $\mathbb{R}^K$ formal präzisiert.

Definition 7.13 (Quaderpartitionierung)

Gegeben seien eine natürliche Zahl K sowie ein kompakter (Hyper-)Quader Q im $\mathbb{R}^K$, $Q := [x_{1,\min}, x_{1,\max}] \times ... \times [x_{K,\min}, x_{K,\max}]$.

Dann nennen wir die auf den Intervallpartitionierungen $x_{1,0} = x_{1,\min} < x_{1,1} < ... < x_{1,z_1} = x_{1,\max}$, $x_{2,0} = x_{2,\min} < x_{2,1} < ... < x_{2,z_2} = x_{2,\max}$, ..., $x_{K,0} = x_{K,\min} < x_{K,1} < ... < x_{K,z_K} = x_{K,\max}$ basierende Zerlegung von Q eine *Quaderpartitionierung*. Ist hierbei mindestens einer der auftretenden Indizes bei den einzelnen Intervallzerlegungen z_1, z_2, ..., $z_K > 1$, so sprechen wir von einer *echten Quaderpartitionierung*.

Bemerkung 7.14

Die einzelnen Elemente einer solchen Quaderpartitionierung haben wiederum die Form von Quadern im $\mathbb{R}^K$ und lassen sich schreiben als $[x_{1,j_1}, x_{1,j_1+1}] \times ... \times [x_{K,j_K}, x_{K,j_K+1}]$.
Eine solche Quaderpartitionierung enthält $z_1 z_2 ... z_K$ Elemente (Teilquader). Der Begriff der *echten* Quaderpartitionierung stellt dabei sicher, dass die Ausgangsmenge in mindestens zwei Teilquader aufgeteilt wird.

Der nachfolgend formal dargestellte Algorithmus dient dazu, aus einem vorgegebenen Neuronalen Netz und zu einer gewünschten Anzahl r eine Menge von r Regeln zu extrahieren. Hierzu wird der Eingaberaum, der ohne Einschränkung als begrenzter K-dimensionaler Quader angenommen wird, in - wiederum quaderförmige - Segmente zerlegt. Für jedes dieser Segmente wird numerisch bestimmt, wie stark dort die Netzausgaben von einem Mittelwert abweichen. Schließlich werden für diejenigen Segmente (weitere) Regeln formuliert, für die die signifikantesten Verbesserungen zu erzielen sind.

Bei diesem Verfahren werden *r-1* Regeln aufgestellt, deren Prämissen vom Typ

$$\vec{x} \; IN \; Quader$$

sind. Die letzte Regel ist als pauschale *ELSE*-Klausel zu interpretieren und deckt den restlichen Definitionsbereich bzw. Eingaberaum ab.

Gemäß Bemerkung 6.34 (sh. S. 129) kann vorausgesetzt werden, dass die Gewichtsvektoren $\vec{w_i}$ zwischen der Eingabe- und der verborgenen Schicht der Norm nach durch 1 beschränkt sind: $\left|\vec{w_i}\right| \leq 1$. Diese Eigenschaft wird in einem Korollar zu dem folgenden Satz verwendet; der Satz selbst wird zunächst für beliebige Gewichtsvektoren formuliert.

Satz 7.15 (Regelextraktion durch schrittweise Verbesserung)

Gegeben sei ein (dreischichtiges) Neuronales K-L-1-Netz, dessen Eingabemenge durch einen kompakten Quader $Q := [x_{1,\min}, x_{1,\max}] \times \ldots \times [x_{K,\min}, x_{K,\max}]$ im $\mathbb{R}^K$ definiert werde. Als Ausgabefunktion der Hidden Neuronen wird die sigmoide Funktion *sigm* verwendet, d.h. die Netzausgabe lässt sich beschreiben durch eine stetige Funktion $f: Q \to \mathbb{R}$. Weiterhin sei eine natürliche Zahl r vorgegeben, die angibt, wieviele Regeln aus dem Netz extrahiert werden sollen.

Dann liefert das nachfolgend beschriebene Verfahren eine Menge von r Regeln, deren maximale Abweichung von der Netzausgabe $\frac{1}{2}\left(\sum_{1 \leq i \leq L,\, v_i > 0} v_i - \sum_{1 \leq i \leq L,\, v_i < 0} v_i\right)$ beträgt; dabei sind die v_i die Gewichte zur Ausgabeschicht.

Algorithmus:

Es seien $y_{\max} := \sum_{1 \leq i \leq L,\, v_i > 0} v_i$ und $y_{\min} := \sum_{1 \leq i \leq L,\, v_i < 0} v_i$ (vgl. hierzu Satz 6.35, S. 130).
Sollte eine der beiden Summen leer sein, d.h. das Ausgangsnetz entweder keine positiven oder keine negativen Gewichtswerte zur Ausgabeschicht aufweisen, so ist die entsprechende Summe 0.

Als erste (pauschale) Regel (mit stets zutreffender Prämisse) wird formuliert:

$$IF \; \vec{x} \; IN \; Q \; THEN y = y_0 := \tfrac{1}{2}(y_{\max} + y_{\min}).$$

Das heißt, der Algorithmus beginnt mit der einfachen Regel, dass alle Ausgabewerte dem Mittelwert y_0 des (maximalen) Wertebereiches der Netzausgaben entsprechen.

Im Fall $r = 1$ endet der Algorithmus hier. Für $r > 1$ wird wie folgt verfahren:

Es wird eine echte Quaderpartitionierung von Q mit mindestens r Elementen generiert, d.h.: es existieren in dieser Partitionierung p (Teil-)Quader der Form $[x_{1,j_1}, x_{1,j_1+1}] \times ... \times [x_{K,j_K}, x_{K,j_K+1}], 1 \le i \le K, 0 \le j < z_i$, mit $p \ge r$; dabei gelte für sämtliche Kantenlängen dieser Quader:

$$(*) \quad x_{i,j_i+1} - x_{i,j_i} \le \frac{2}{K \cdot w_{\max}}, \ 1 \le i \le K, 0 \le j_i < z_i \ .$$

Dabei ist $w_{\max} := \max\{ |w_{lk}| \ : 1 \le l \le L, \ 1 \le k \le K \}$.

Für jeden dieser p Quader $Q_i, \ 1 \le i \le p,$ wird ein über die Eckpunkte des Quaders gemittelter Funktionswert gebildet:

$$y_i := \frac{1}{2^K} \sum_{t=1}^{2^K} f(\overrightarrow{e_{i,t}}).$$

Hierbei bezeichnet $\overrightarrow{e_{i,t}}$ den t-ten Eckpunkt des Quaders Q_i.

Nun werden die betragsmäßigen Abweichungen *abw(i)* dieser gemittelten Funktionswerte von y_0 betrachtet

$$abw(i) := |y_i - y_0|$$

und absteigend sortiert. D.h. mit einer geeigneten Permutation $\alpha : \{1, 2, ..., p\} \to \{1, 2, ..., p\}$ gilt:

$$abw(\alpha(1)) \ge abw(\alpha(2)) \ge ... \ge abw(\alpha(p)).$$

Für die *r-1* Partitionsquader $Q_{\alpha(1)}, Q_{\alpha(2)}, ..., Q_{\alpha(r-1)}$ werden Regeln aufgestellt:

$$\text{Regel } i\text{: } IF \ \overrightarrow{x} \ IN \ Q_{\alpha(i)} \ THEN \ y = y_{\alpha(i)}, \ 1 \le i \le r-1.$$

Schließlich wird die Ausgangsregel in ihrer Prämisse modifiziert, denn diese soll nur noch für die Eingaben zuständig sein, die nicht durch die o.g. Regeln abgedeckt werden. Diese Regel lautet daher nun:

$$IF \ \overrightarrow{x} \ IN \ Q \ AND \ \overrightarrow{x} \ NOT \ IN \ \bigcup_{1 \le i \le r-1} Q_{\alpha(i)} \ THEN \ y = y_0$$

bzw. äquivalent umgeformt

$$\text{Regel } r\text{: } IF \ \overrightarrow{x} \ IN \ Q \setminus \bigcup_{1 \le i \le r-1} Q_{\alpha(i)} \ THEN \ y = y_0.$$

Beweis der Behauptung, d.h. der aufgestellten Güte-Aussage:

Zu zeigen ist, dass die behauptete maximale Abweichung der aufgestellten Regelbasis von der Netzausgabe, die bei diesem Verfahren auftreten kann, $\frac{1}{2}\left(\sum_{1 \le i \le L, \ v_i > 0} v_i - \sum_{1 \le i \le L, \ v_i < 0} v_i\right)$ beträgt.

Wir definieren zur einfacheren Darstellung das Maximum aller Kantenlängen der Partitionsquader als $\delta := \max\{ x_{i,j_i+1} - x_{i,j_i} | 1 \le i \le K, 0 \le j_i < z_i \}$.
Aus der vorausgesetzten Ungleichung (*) folgt somit $\delta \le \frac{2}{K \cdot w_{\max}}$.

Es sei $\overrightarrow{x}$ ein Element des Eingaberaums.

1.Fall: $\overrightarrow{x}$ gehört zu einem der *r-1* Partitionsquader $Q_{\alpha(i)}, \ 1 \le i \le r-1$.

Hier wurde als Regel-Ausgabe der Wert $y_{a(i)} = \frac{1}{2^K} \sum_{t=1}^{2^K} f(\overrightarrow{e_{a(i),t}})$ definiert: das Mittel der Funktionswerte auf den Quaderecken $\overrightarrow{e_{a(i),t}}$. Wegen der Stetigkeit der Funktion f gibt es (mindestens) ein $\overrightarrow{x_0}$ aus $Q_{a(i)}$ mit $y_{a(i)} = f(\overrightarrow{x_0})$.

Die sigmoide Funktion *sigm* ist (überall) differenzierbar, ihre Ableitung lässt sich in der Form $\frac{\partial\, sigm}{\partial x}(x) = \frac{1}{1+e^{-x}} \cdot (1 - \frac{1}{1+e^{-x}})$ schreiben (vgl. Bemerkung 1.14 auf S. 26). Damit ist auch die Ausgabefunktion des Netzes $f(\overrightarrow{x}) = \sum_{j=1}^{L} v_j \cdot sigm(\overrightarrow{w_j} \cdot \overrightarrow{x})$ differenzierbar: sie besitzt sämtliche partiellen Ableitungen $\frac{\partial f}{\partial x_i}$ (in Bezug auf $x_1, x_2, ..., x_K$), die sich wie folgt ergeben:

$$\frac{\partial f}{\partial x_i}(\overrightarrow{x}) = \sum_{j=1}^{L} v_j w_{j,i} \cdot (\frac{1}{1+\exp(-\overrightarrow{w_j} \cdot \overrightarrow{x})} \cdot (1 - \frac{1}{1+\exp(-\overrightarrow{w_j} \cdot \overrightarrow{x})})), \quad 1 \le i \le K.$$

Für $0 \le z \le 1$ ist $0 \le z \cdot (1-z) \le \frac{1}{4}$. Der Ausdruck $z := \frac{1}{1+\exp(-\overrightarrow{w_j} \cdot \overrightarrow{x})}$ erfüllt diese Voraussetzung, somit resultiert die entsprechende Abschätzung: $0 \le \frac{1}{1+\exp(-\overrightarrow{w_j} \cdot \overrightarrow{x})} \cdot (1 - \frac{1}{1+\exp(-\overrightarrow{w_j} \cdot \overrightarrow{x})}) \le \frac{1}{4}$.

Gemeinsam mit $|w_{ji}| \le w_{\max}$ (für alle i und j mit $1 \le i \le K$ bzw.und $1 \le j \le L$) kann also die partielle Ableitung betragsmäßig abgeschätzt werden:

$$\left| \frac{\partial f}{\partial x_i}(\overrightarrow{x}) \right| \le \frac{1}{4} \cdot w_{\max} \cdot \sum_{j=1}^{L} |v_j|, \quad 1 \le i \le K.$$

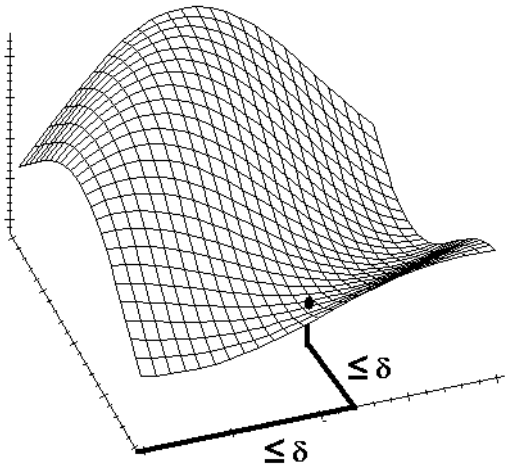


Abb. 7.4: Abschätzung des Funktionswertes auf einem Quader der Partition

Damit kann für zwei beliebige Elemente $\overrightarrow{x_a}$, $\overrightarrow{x_b}$ aus $Q_{a(i)}$ die Differenz $f(\overrightarrow{x_b}) - f(\overrightarrow{x_a})$ betragsmäßig abgeschätzt werden wie folgt.

$$\left| f(\overrightarrow{x_b}) - f(\overrightarrow{x_a}) \right| \le \delta \cdot \sum_{1 \le i \le K} \left| \frac{\partial f}{\partial x_i}(\overrightarrow{x}) \right| \le \delta \cdot \frac{1}{4} w_{\max} \cdot \sum_{1 \le i \le K} \sum_{j=1}^{L} |v_j| = \frac{1}{4} \delta w_{\max} K \sum_{j=1}^{L} |v_j| \le \frac{1}{2} \sum_{j=1}^{L} |v_j|.$$

Die letzte Ungleichung dieser Kette folgt aus der Voraussetzung $\delta \le \frac{2}{K \cdot w_{\max}}$.

Nun ist $\sum_{j=1}^{L} |v_j| = \sum_{1 \le i \le L,\, v_i > 0} v_i - \sum_{1 \le i \le L,\, v_i < 0} v_i = y_{\max} - y_{\min}$, so dass insgesamt resultiert:

$$\left| f(\overrightarrow{x_b}) - f(\overrightarrow{x_a}) \right| \le \frac{1}{2}(y_{\max} - y_{\min}).$$

Setzt man speziell $\overrightarrow{x_b} = \overrightarrow{x_0}$, so folgt für ein beliebiges Element $\overrightarrow{x_a}$ aus $Q_{a(i)}$:

$$\left|f(\overrightarrow{x_0}) - f(\overrightarrow{x_a})\right| = \left|y_{a(i)} - f(\overrightarrow{x_a})\right| \le \tfrac{1}{2}(y_{\max} - y_{\min}),$$

so dass die Behauptung im 1.Fall gezeigt ist.

2.Fall: $\overrightarrow{x}$ gehört zu keinem der *r-1* Partitionsquader $Q_{a(i)}$, $1 \le i \le r-1$, somit ist für dieses $\overrightarrow{x}$ Regel *r* zuständig.

Hier ist die Behauptung trivialerweise erfüllt, denn diese Regel liefert den Funktionswert $y_0 = \tfrac{1}{2}(y_{\max} + y_{\min}) = \tfrac{1}{2}(\sum_{1\le i\le L,\, v_i>0} v_i + \sum_{1\le i\le L,\, v_i<0} v_i)$. Gemäß Satz 6.35 liegt die Netzausgabe *y* stets zwischen $y_{\min} = \sum_{1\le i\le L,\, v_i<0} v_i$ und $y_{\max} = \sum_{1\le i\le L,\, v_i>0} v_i$.

Für $y > y_0$ ist $y - y_0 = y - \tfrac{1}{2}(y_{\max} + y_{\min}) \le y_{\max} - \tfrac{1}{2}(y_{\max} + y_{\min}) = \tfrac{1}{2}(y_{\max} - y_{\min})$,

für $y \le y_0$ ist $y_0 - y = \tfrac{1}{2}(y_{\max} + y_{\min}) - y \le \tfrac{1}{2}(y_{\max} + y_{\min}) - y_{\min} = \tfrac{1}{2}(y_{\max} - y_{\min})$.

Damit ist die Behauptung auch für den 2.Fall und somit insgesamt bewiesen.

□

Bemerkung 7.16

1. Eine solche Partitionierung, wie sie in diesem Satz benötigt wird, lässt sich naturgemäß auf verschiedene Weise generieren.
   Dies kann dadurch geschehen, dass in jeder der K Dimensionen mindestens $r^{1/K} + 1$ (auf die nächstgrößere natürliche Zahl aufgerundet) Unterteilungspunkte (Stützstellen) gewählt werden; es kommt jedoch jede Partitionierung mit *p* Elementen in Frage, für die (mit den Bezeichnungen gemäß Definition 7.13) gilt: $p := z_1 z_2 ... z_K \ge r$ und die die Nebenbedingung (*) von S. 145 erfüllt.
2. Ungleichung (*) macht deutlich, dass um so mehr Partitionsquader gebildet werden müssen, desto größer K und $w_{\max}$ sind.
3. Ist im obigen Satz im Spezialfall $p = r$, d.h. werden genausoviele Teilquader gebildet wie Regeln gewünscht werden, so ist die *r*-te Regel von derselben Form wie die *r-1* Regeln zuvor, da das Komplement der Partitionsquader $Q_{a(1)}, Q_{a(2)}, ..., Q_{a(r-1)}$ gerade dem letzten Quader entspricht: $Q_{a(r)} = Q \setminus \bigcup_{1\le i\le r-1} Q_{a(i)}$.
   Ggf. kann in diesem speziellen Fall auf den letzten Quader noch die im Algorithmus beschriebene Regelanpassung angewendet werden.
4. Der Form halber ist anzumerken, dass der vorige Satz 7.15 zu einer gegebenen Partitionierung des Eingaberaumes naturgemäß nicht beliebig viele Regeln generieren kann. Umgekehrt lässt sich aber zu jeder gewünschten endlichen Regelanzahl *r* eine geeignete Partitionierung angeben, mit der mindestens *r* Regeln extrahiert werden können.
5. Satz 7.15 geht davon aus, dass als Definitionsbereich (Eingaberaum) ein Quader im $\mathbb{R}^K$ vorliegt. Diese Voraussetzung wurde zur direkten und besseren Formulierung des Verfahrens gewählt; qualitativ gilt die Aussage des Satzes für eine beliebige kompakte Teilmenge des $\mathbb{R}^K$ in entsprechender Weise.
6. Der Algorithmus verwendet die Eckpunkte der einzelnen Partitionsquader zur Mittelwertbildung. Prinzipiell können aber natürlich auch andere Punkte des jeweiligen Quaders

dazu verwendet werden. Die Formulierung des Algorithmus über die Eckpunkte erscheint jedoch am einfachsten und verständlichsten.

Korollar 7.17

Gilt im vorigen Satz 7.15 für die Gewichtsvektoren $\overrightarrow{w_i}$ zwischen der Eingabe- und der verborgenen Schicht insbesondere die Normiertheitsbedingung $w_{\max} \leq 1$, so vereinfacht sich Bedingung (*) von S. 145 zu

$$x_{i,j_i+1} - x_{i,j_i} \leq \tfrac{2}{K}, \;\; 1 \leq i \leq K, 0 \leq j < z_i \; .$$

❑

Bemerkung 7.18

In Hinblick auf eine Software-Implementierung des o.g. Verfahrens sei angemerkt, dass die letzte Regel der Regelbasis,

$$\text{Regel } r\text{: } \mathit{IF}\ \overrightarrow{x}\ \mathit{IN}\ Q \setminus \bigcup_{1 \leq i \leq r-1} Q_{\alpha(i)}\ \mathit{THEN}\ y = y_0,$$

die Rolle einer letztendlichen Alternative, also einer abschließenden ELSE-Klausel, einnimmt.

Auch die Prämissen der vorhergehenden Regeln schließen sich paarweise gegenseitig aus, insgesamt überdecken die r Prämissen den gesamten Eingaberaum. Insofern kann eine konkrete (Software-)Implementierung in der formalen Struktur

IF *Prämisse-1*
THEN *Konklusion-1*
ELSE IF *Prämisse-2*
THEN *Konklusion-2*
ELSE IF ...
ELSE IF *Prämisse-(r-1)*
THEN *Konklusion-(r-1)*
ELSE
Konklusion-r

realisiert werden, wie sie auch in der folgenden Abbildung, einem Struktogramm, skizziert wird. Der Vorteil liegt in der Performanz, da nach dem Erfülltsein einer Prämisse (“true”) keine weiteren, unnötigen Bedingungen mehr geprüft werden müssen.

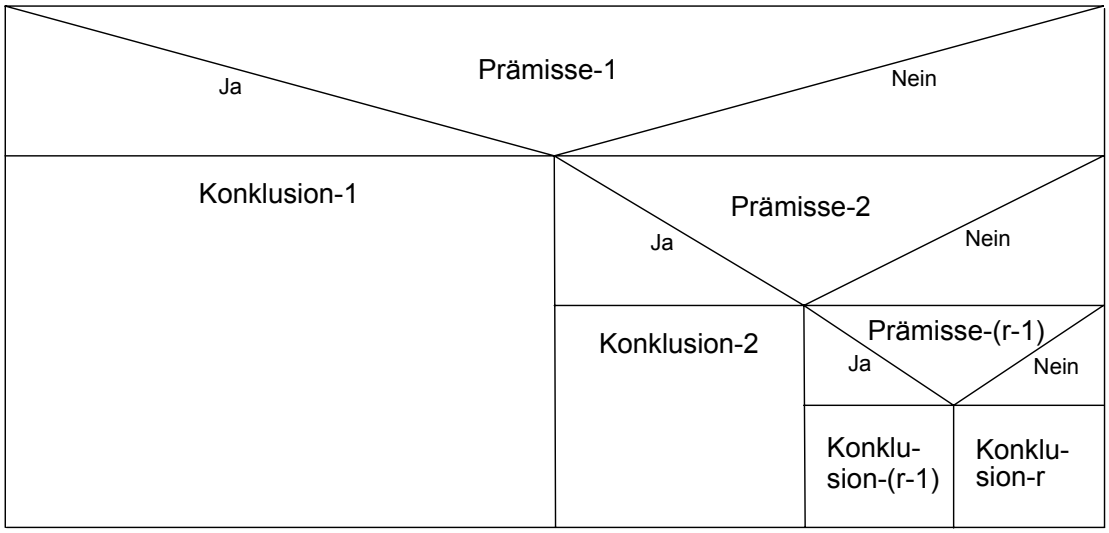


Abb. 7.5: Struktogramm - Kaskadierte IF-THEN-ELSE-Struktur

Rein formal kann eine solche kaskadierte IF-THEN-ELSE-Struktur natürlich umgeschrieben werden in gleichberechtigte IF-THEN-Formulierungen.

Beispiel 7.19

Für eine einfache qualitative Demonstration betrachten wir das (kleine) Neuronale Netz aus Beispiel 7.11 (S. 142).

Wenden wir das Regelextraktionsverfahren der schrittweisen Verbesserung auf dieses Netz an und lassen 16 Regeln erzeugen. So gelangen wir zu der folgenden Regelbasis, wobei die Ergebniswerte auf zwei Nachkommastellen gerundet worden sind[20].

Regel Nr. 1	if x in [-25.0, -12.5] x [-25.0, -12.5] then y = -0.29
Regel Nr. 2	if x in [12.5, 25.0] x [12.5, 25.0] then y = 0.29
Regel Nr. 3	if x in [0.0, 12.5] x [12.5, 25.0] then y = 0.25
Regel Nr. 4	if x in [-12.5, 0.0] x [-25.0, -12.5] then y = -0.25
Regel Nr. 5	if x in [-12.5, 0.0] x [12.5, 25.0] then y = 0.20
Regel Nr. 6	if x in [0.0, 12.5] x [-25.0, -12.5] then y = -0.20
Regel Nr. 7	if x in [-25.0, -12.5] x [12.5, 25.0] then y = 0.17
Regel Nr. 8	if x in [12.5, 25.0] x [-25.0, -12.5] then y = -0.17
Regel Nr. 9	if x in [-25.0, -12.5] x [-12.5, 0.0] then y = -0.15
Regel Nr. 10	if x in [12.5, 25.0] x [0.0, 12.5] then y = 0.15
Regel Nr. 11	if x in [0.0, 12.5] x [0.0, 12.5] then y = 0.14
Regel Nr. 12	if x in [-12.5, 0.0] x [-12.5, 0.0] then y = -0.14
Regel Nr. 13	if x in [-12.5, 0.0] x [0.0, 12.5] then y = 0.11
Regel Nr. 14	if x in [0.0, 12.5] x [-12.5, 0.0] then y = -0.11
Regel Nr. 15	if x in [-25.0, -12.5] x [0.0, 12.5] then y = 0.05
Regel Nr. 16	if x in [12.5, 25.0] x [-12.5, 0.0] then y = -0.05

[20] Die numerischen Daten finden sich in Anhang A5 auf S. 188.

Die Reihenfolge der Regeln ergibt sich aus dem Verfahren; früher genannte Regeln bringen gegenüber der provisorischen pauschalen Regel signifikantere Verbesserungen als später aufgelistete.

Nachstehend ist die Ausgabefunktion dieser Regelbasis aufgetragen. Der mittlere betragsmäßige Fehler liegt bei ca. 0,075, der mittlere quadratische Fehler bei etwa 0,008.

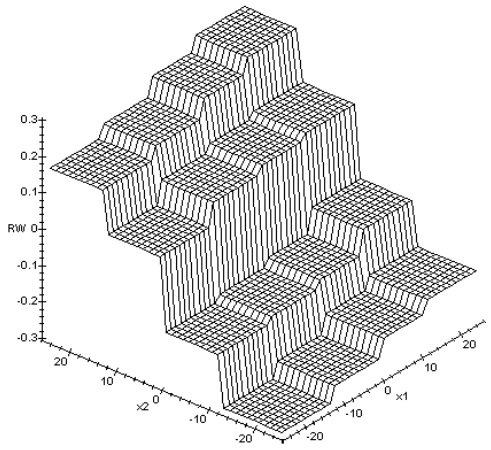


Abb. 7.6: Ausgabe der generierten Regelbasis mit 16 Regeln

Numerisches Beispiel 7.20

Wir wollen ebenfalls das Beispiel des Inversen Pendels (vgl. 6.26 auf S. 125 sowie 6.42 auf S.134) aufgreifen und das vorgestellte Regelextraktionsverfahren der schrittweisen Verbesserung (Satz 7.15) auch hierauf anwenden.

Eine Reihe von Simulationen zeigt, dass dieses Verfahren sehr gute Resultate aufweisen kann, wenn die Anzahl der zu extrahierenden Regeln für die behandelte Problemstellung günstig ist.
Die hier erwähnten Simulationsergebnisse sind in Anhang A4 auf S. 183 ff aufgelistet. Für diese Simulation wurde als Ausgangspartitionierung eine äquidistante Intervallunterteilung in beiden Dimensionen des Eingaberaums vorgenommen, so dass die Partitionierung mindestens r Rechtecke (Quader) enthält.

Im konkreten Fall gehen wir von der Inversen-Pendel-Ausgangsfunktion aus, die auf 16 Quadranten stückweise konstant definiert ist. Aufgrund dieser Situation kann das Verfahren der schrittweisen Verbesserung für den Fall von $r = 16$ Regeln (kurz: SV-16) seine Qualität beweisen: hier ergibt sich ein über 150 Datensätze gemittelter durchschnittlicher MSE von 0,0958. Damit ist SV-16 bei den konkreten Beispieldaten sogar besser als das Ausgangsnetz. Wird hingegen eine zu kleine Anzahl von zu erzeugenden Regeln gewählt, so sind naturgemäß Qualitätseinbußen hinzunehmen. SV-8 kommt auf einen mittleren MSE von 0,2483; zu Demonstrationszwecken wird im Anhang auch der Fall aufgetragen, bei dem lediglich fünf Regeln generiert werden; bei diesem SV-5 ergibt sich mit 0,3812 ein deutlich schlechterer MSE (im Mittel über die 150 Datensätze).

Für wachsende Regelanzahl r nähert sich SV-r in seiner Ausgabe immer weiter dem Original-Netz an; dies unterstreichen auch die numerischen Daten.

Aber, und das mag zunächst überraschen, eine zu groß festgelegte Regelanzahl *r* kann zu einer Verschlechterung führen. Exemplarisch sind im Anhang die Ergebnisse für SV-128, also das Erzeugen von 128 Regeln, notiert. Mit 0,1267 ist der hierbei erzielte durchschnittliche MSE jedoch schlechter als im "optimalen" Falle der 16 Regeln.
Zu interpretieren ist dies dahingehend, dass mit "zu vielen" Regeln das Netz "zu dicht" approximiert wird, obwohl sich dieses in manchen Bereichen nicht besonders gut an die Ausgangsfunktion annähert.

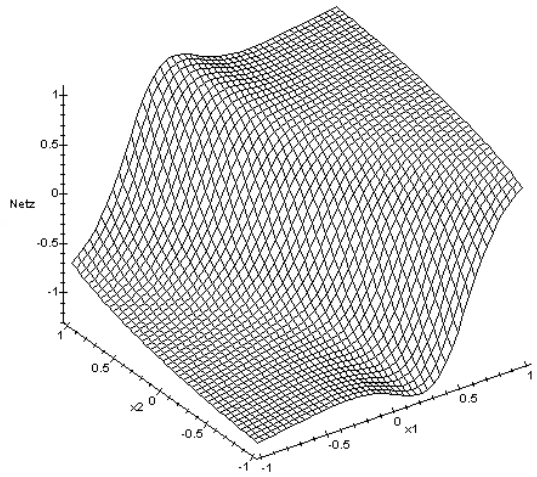


Abb. 7.7: Ausgabefunktion eines 2-24-1-Netzes zum Beispiel des Inversen Pendels

In obenstehender Abbildung wird die Ausgabefunktion eines auf die Problemstellung des Inversen Pendels trainiertes 3-Schicht-Netzes mit 24 Hidden Neuronen dargestellt.

Nachstehend ist zur Illustration ein Funktionsplot zu einer Regelbasis für das Inverse Pendel-Problem mit 16 Regeln aufgetragen. Gut erkennbar ist der qualitative Verlauf, der dem ursprünglichen Funktionsgraphen - wie in Abb. 6.13 (auf S. 123) dargestellt - recht nahe kommt. Dabei ist zu betonen, dass die generierte Regelbasis nicht aus dieser Funktionsvorschrift abgeleitet wurde, sondern aus dem approximierenden 3-Schicht-Netz. So ist auch zu erklären, dass die Ergebniswerte der Regelmenge den Wert +1 überschreiten können, denn das auf das Pendel-Problem trainierte Netz tut dies ebenfalls.

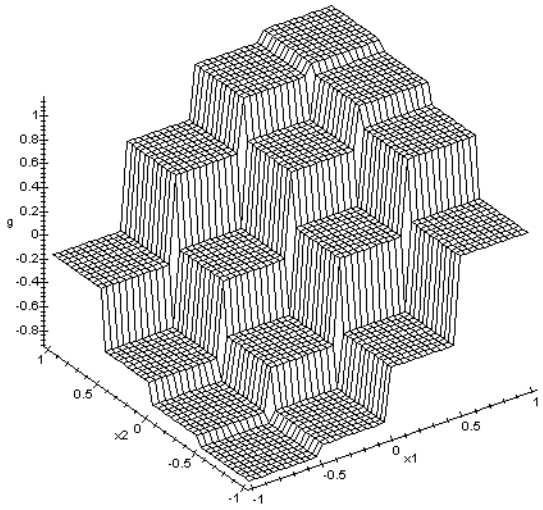


Abb. 7.8: Ein Beispiel-Plot einer SV-16-Regelbasis

Im Fall von 16 Regeln ist dies im konkreten Fall naturgemäß die Aufteilung in die in der obenstehenden Abbildung 7.8 gezeigten sechzehn Quadrate. Diese passen gut zur formulierten Aufgabenstellung, insofern ist es nachvollziehbar, dass die Regelextraktion SV-16 ein gutes Resultat erzielt. Es mag jedoch überraschen, dass das Regelextraktionsverfahren - im vorliegend diskutierten Fall - Approximationsschwächen des Ausgangsnetzes ein kleines Stück weit zu korrigieren imstande ist!

7.3 Regelextraktion bei vektorwertiger Netz-Ausgabe

Das Resultat von Satz 7.15 kann mit leichter Modikation an die Situation einer höherdimensionalen Netzausgabe adaptiert werden. Dabei wird im Folgenden im Interesse einer einfacheren Darstellung die Maximumsnorm auf dem $\mathbb{R}^M$ verwendet:

$$\left|\overrightarrow{x}\right| := \max\{|x_1|, ..., |x_M|\} \text{ für } \overrightarrow{x} \in \mathbb{R}^M.$$

Bemerkung 7.21

Dass wir hier die Maximumsnorm verwenden, schränkt die Allgemeinheit der Aussage nicht ein. Auf dem endlich-dimensionalen Vektorraum $\mathbb{R}^M$ sind alle Normen zueinander äquivalent. (Vgl. etwa [Bach1966], S. 122.)

Bemerkung 7.22

Im nachstehenden Satz wird ein dreischichtiges Netz mit M-dimensionaler Ausgabeschicht behandelt. Mit $\overrightarrow{v_m} = (v_{m,1}, ..., v_{m,L})$ (für $1 \le m \le M$) werden die darin auftretenden Gewichtsvektoren zur Ausgabeschicht notiert; im nachstehenden Bild eines 3-3-2-Netzes ist beispielhaft der Gewichtsvektor $\overrightarrow{v_1} = (v_{1,1}, v_{1,2}, v_{1,3})$ eingezeichnet.

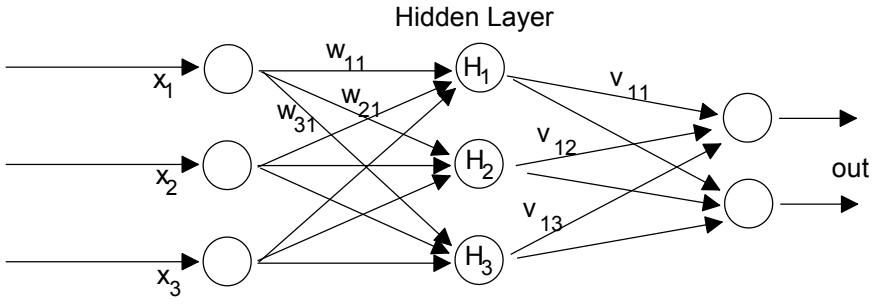


Abb. 7.9: Exemplarische Darstellung eines 3-3-2-Netzes mit dem Gewichtsvektor (v_{11},v_{12},v_{13})

Satz 7.23 (Regelextraktion durch schrittweise Verbesserung)

Gegeben sei ein (dreischichtiges) Neuronales K-L-M-Netz, dessen Eingabemenge durch einen kompakten Quader $Q := [x_{1,\min}, x_{1,\max}] \times ... \times [x_{K,\min}, x_{K,\max}]$ im $\mathbb{R}^K$ definiert werde. Als Ausgabefunktion der Hidden Neuronen wird die sigmoide Funktion *sigm* verwendet, d.h. die Netzausgabe lässt sich beschreiben durch eine stetige Funktion $f: Q \to \mathbb{R}^M$ mit den Komponenten $f_m: Q \to \mathbb{R},\ 1 \le m \le M$.

Weiterhin sei eine natürliche Zahl *r* vorgegeben, die angibt, wieviele Regeln aus dem Netz extrahiert werden sollen.

Dann liefert das nachfolgend beschriebene Verfahren eine Regelbasis mit *r* Regeln, deren maximale Abweichung von der Netzausgabe im Sinne der Maximumsnorm

$$\tfrac{1}{2} \max_{1 \le m \le M} \Big(\sum_{1 \le i \le L,\ v_{m,i} > 0} v_{m,i} - \sum_{1 \le i \le L,\ v_{m,i} < 0} v_{m,i} \Big)$$

beträgt. Hierbei ist $v_{m,i}$ das Gewicht vom *i*-ten Neuron der verborgenen Schicht zum *m*-ten Ausgabeneuron.

Algorithmus:

Es seien $\overrightarrow{y_{\max}}$ und $\overrightarrow{y_{\min}}$ die Vektoren im $\mathbb{R}^M$ mit den Komponenten $y_{\max,m} := \sum_{1 \le i \le L,\ v_{m,i} > 0} v_{m,i}$ und $y_{\min,m} := \sum_{1 \le i \le L,\ v_{m,i} < 0} v_{m,i}$.
Sollte eine dieser Summen leer sein, so ist die entsprechende Summe wiederum 0.

Als erste (pauschale) Regel (mit stets zutreffender Prämisse) wird formuliert:

$$IF\ \overrightarrow{x}\ IN\ Q\ THEN\ \overrightarrow{y} = \overrightarrow{y_0} := \tfrac{1}{2}(\overrightarrow{y_{\max}} + \overrightarrow{y_{\min}}).$$

Das heißt, der Algorithmus beginnt mit der einfachen Regel, dass alle Ausgabewerte dem (komponentenweisen) Mittelwert $\overrightarrow{y_0}$ des (maximalen) Wertebereiches der Netzausgaben entsprechen, wobei jede Ausgabedimension separat berechnet wird.

Im formalen Spezialfall $r = 1$ endet der Algorithmus hier.

Für $r > 1$ wird nun eine echte Quaderpartitionierung von Q mit mindestens r Elementen generiert, d.h. es existieren in dieser Partitionierung p (Teil-)Quader der Form

$$[x_{1,j_1}, x_{1,j_1+1}] \times ... \times [x_{K,j_K}, x_{K,j_K+1}],\ 1 \le i \le K,\ 0 \le j < z_i \text{ , mit } p \ge r;$$

dabei gelte für sämtliche Kantenlängen dieser Quader:

$$(*) \quad x_{i,j_i+1} - x_{i,j_i} \le \frac{2}{K \cdot w_{\max}},\ 1 \le i \le K, 0 \le j_i < z_i \ .$$

Hierbei ist $w_{\max} := \max\{\ |w_{lk}|\ : 1 \le l \le L,\ 1 \le k \le K\ \}$.

Für jeden dieser p Quader $Q_i,\ 1 \le i \le p,$ und für jede Ausgabedimension m, $1 \le m \le M$, wird ein über die Eckpunkte des Quaders gemittelter Funktionswert gebildet:

$$y_{i,m} := \frac{1}{2^K} \sum_{t=1}^{2^K} f_m(\overrightarrow{e_{i,t}}).$$

Hierbei wird mit $\overrightarrow{e_{i,t}}$ wiederum der t-te Eckpunkt des Quaders Q_i bezeichnet.

Nun werden die betragsmäßigen Abweichungen *abw(i)* dieser gemittelten Funktionswerte $\overrightarrow{y_i}$ von $\overrightarrow{y_0}$ in Bezug auf die Maximumsnorm betrachtet, das heißt:

$$abw(i) := \max_{1 \le m \le M} \left| y_{i,m} - y_{0,m} \right|.$$

Diese werden absteigend sortiert, das heißt, dass mit einer geeigneten Permutation $\alpha : \{1, 2, ..., p\} \to \{1, 2, ..., p\}$ gilt:

$$abw(\alpha(1)) \ge abw(\alpha(2)) \ge ... \ge abw(\alpha(p)).$$

Für die *r-1* Partitionsquader $Q_{\alpha(1)}, Q_{\alpha(2)}, ..., Q_{\alpha(r-1)}$ werden Regeln aufgestellt:

$$\text{Regel } i\text{: } IF\ \overrightarrow{x}\ IN\ Q_{\alpha(i)}\ THEN\ \overrightarrow{y} = \overrightarrow{y_{\alpha(i)}},\ 1 \le i \le r-1.$$

Schließlich wird die Ausgangsregel in ihrer Prämisse modifiziert, denn diese soll nur noch für die Eingaben zuständig sein, die nicht durch die o.g. Regeln abgedeckt werden. Diese Regel lautet somit

$$IF\ \overrightarrow{x}\ IN\ Q\ AND\ \overrightarrow{x}\ NOT\ IN \bigcup_{1 \le i \le r-1} Q_{\alpha(i)}\ THEN\ \overrightarrow{y} = \overrightarrow{y_0}$$

bzw.

$$\text{Regel } r\text{: } IF\ \overrightarrow{x}\ IN\ Q \setminus \bigcup_{1 \le i \le r-1} Q_{\alpha(i)}\ THEN\ \overrightarrow{y} = \overrightarrow{y_0}.$$

Beweis der Behauptung, d.h. der aufgestellten Güte-Aussage:

Zu zeigen ist, dass die behauptete maximale Ergebnis-Abweichung der aufgestellten Regelbasis von der Netzausgabe, die bei diesem Verfahren auftreten kann, nicht mehr als $\frac{1}{2} \max_{1 \le m \le M} \left(\sum_{1 \le i \le L,\ v_{m,i} > 0} v_{m,i} - \sum_{1 \le i \le L,\ v_{m,i} < 0} v_{m,i} \right)$ beträgt.

Wie zuvor definieren wir auch hier zur einfacheren Darstellung das Maximum aller Kantenlängen der Partitionsquader als $\delta := \max\{\ x_{i,j_i+1} - x_{i,j_i} | 1 \le i \le K, 0 \le j_i < z_i\ \}$.
Wegen (*) folgt $\delta \le \frac{2}{K \cdot w_{\max}}$.

Es sei $\overrightarrow{x}$ ein Element des Eingaberaums.

1.Fall: $\overrightarrow{x}$ gehört zu einem der *r-1* Partitionsquader $Q_{\alpha(i)}$, $1 \le i \le r-1$.

Hier wurde als Regel-Ausgabe der Wert $\overrightarrow{y_{\alpha(i)}} = \frac{1}{2^K} \sum_{t=1}^{2^K} f(\overrightarrow{e_{\alpha(i),t}})$ definiert: das komponentenweise Mittel der Funktionswerte auf den Quaderecken $\overrightarrow{e_{\alpha(i),t}}$. Wegen der Stetigkeit der Funktion *f* gibt es (mindestens) ein $\overrightarrow{x_0}$ aus $Q_{\alpha(i)}$ mit $\overrightarrow{y_{\alpha(i)}} = f(\overrightarrow{x_0})$.

Jede Komponente $f_m(\overrightarrow{x}) = \sum_{j=1}^{L} v_{m,j} \cdot sigm(\overrightarrow{w_j} \cdot \overrightarrow{x})$ der Ausgabefunktion des Netzes ist differenzierbar: sie besitzt sämtliche partiellen Ableitungen $\frac{\partial f_m}{\partial x_i}$ (in Bezug auf $x_1, x_2, ..., x_K$), die sich wie folgt ergeben:

$$\frac{\partial f_m}{\partial x_i}(\overrightarrow{x}) = \sum_{j=1}^{L} v_{m,j} w_{j,i} \cdot \left(\frac{1}{1+\exp(-\overrightarrow{w_j} \cdot \overrightarrow{x})} \cdot \left(1 - \frac{1}{1+\exp(-\overrightarrow{w_j} \cdot \overrightarrow{x})}\right)\right), \quad 1 \le i \le K.$$

Gemeinsam mit $|w_{ji}| \le w_{\max}$ (für alle *i* und *j* mit $1 \le i \le K$ bzw.und $1 \le j \le L$) können die partiellen Ableitungen wie zuvor betragsmäßig abgeschätzt werden:

$$\left|\frac{\partial f_m}{\partial x_i}(\overrightarrow{x})\right| \le \frac{1}{4} \cdot w_{\max} \cdot \sum_{j=1}^{L} |v_{m,j}|, \quad 1 \le i \le K, \ 1 \le m \le M.$$

Damit kann für zwei beliebige Elemente $\overrightarrow{x_a}$, $\overrightarrow{x_b}$ aus $Q_{\alpha(i)}$ die Differenz $f_m(\overrightarrow{x_b}) - f_m(\overrightarrow{x_a})$ betragsmäßig abgeschätzt werden wie folgt.

$$\left|f_m(\overrightarrow{x_b}) - f_m(\overrightarrow{x_a})\right| \le \delta \cdot \sum_{1 \le i \le K} \left|\frac{\partial f_m}{\partial x_i}(\overrightarrow{x})\right| \le \frac{1}{4}\delta w_{\max} \sum_{1 \le i \le K} \sum_{j=1}^{L} |v_{m,j}| = \frac{1}{4}\delta w_{\max} K \sum_{j=1}^{L} |v_{m,j}| \le \frac{1}{2} \sum_{j=1}^{L} |v_{m,j}|.$$

Die letzte Ungleichung dieser Kette folgt aus der Voraussetzung $\delta \le \frac{2}{K \cdot w_{\max}}$.

Nun ist $\sum_{j=1}^{L} |v_{m,j}| = \sum_{1 \le i \le L,\ v_i > 0} v_{m,i} - \sum_{1 \le i \le L,\ v_i < 0} v_{m,i} = y_{\max,m} - y_{\min,m}$, so dass insgesamt resultiert:

$$\left|f_m(\overrightarrow{x_b}) - f_m(\overrightarrow{x_a})\right| \le \frac{1}{2}(y_{\max,m} - y_{\min,m}).$$

Setzt man speziell $\overrightarrow{x_b} = \overrightarrow{x_0}$, so folgt für ein beliebiges Element $\overrightarrow{x_a}$ aus $Q_{\alpha(i)}$:

$$\left|f_m(\overrightarrow{x_0}) - f_m(\overrightarrow{x_a})\right| = \left|y_{\alpha(i),m} - f_m(\overrightarrow{x_a})\right| \le \frac{1}{2}(y_{\max,m} - y_{\min,m}).$$

Vektoriell geschrieben ist dies (mit $|\cdot|$ als der Maximumsnorm)

$$\left|f(\overrightarrow{x_0}) - f(\overrightarrow{x_a})\right| = \left|\overrightarrow{y_{\alpha(i)}} - f(\overrightarrow{x_a})\right| \le \frac{1}{2}\left|\overrightarrow{y_{\max}} - \overrightarrow{y_{\min}}\right| = \frac{1}{2} \max_{1 \le m \le M} \left(\sum_{1 \le i \le L,\ v_{m,i} > 0} v_{m,i} - \sum_{1 \le i \le L,\ v_{m,i} < 0} v_{m,i}\right),$$

so dass die Behauptung im 1.Fall gezeigt ist.

2.Fall: $\overrightarrow{x}$ gehört zu keinem der *r-1* Partitionsquader $Q_{a(i)}$, $1 \le i \le r-1$, somit ist für dieses $\overrightarrow{x}$ Regel *r* zuständig.

Diese Regel liefert den Funktionswert $\overrightarrow{y_0} = \frac{1}{2}(\overrightarrow{y_{\max}} + \overrightarrow{y_{\min}})$. Komponentenweise kann wegen $y_{\min,m} \le f_m(\overrightarrow{x}) \le y_{\max,m}$ daher wiederum abgeschätzt werden:

$$\left| f_m(\overrightarrow{x}) - y_{0,m} \right| = \left| f_m(\overrightarrow{x}) - \tfrac{1}{2}(y_{\max,m} + y_{\min,m}) \right| \le \tfrac{1}{2}(y_{\max,m} - y_{\min,m}).$$

Damit ist die Behauptung auch für den 2.Fall und somit insgesamt bewiesen.

❑

7.4 Implementierung von Regelextraktionsverfahren mit Java

Begleitend zu den theoretischen Ausführungen dieser Arbeit ist eine in der Programmiersprache Java implementierte, prototypische Applikation mit dem Titel "*Javanna*" entstanden, die die praktische Evaluierung einiger Regelextraktionsmechanismen exemplarisch ermöglicht. *Javanna* steht dabei als Akronym für "*Java-based neural network application*".

Nachfolgend soll ein kurzer Überblick über das Konzept, das User Interface und die Objektmodellierung dieser Anwendung gegeben werden. In Anhang A7 findet sich eine Aufstellung der wesentlichen Klassen sowie der Klassenhierarchie, auf der Begleit-CDROM zu dieser Arbeit sind die Software selbst und die komplette Dokumentation im HTML-Format zu finden.

7.4.1 Konzeption und Überblick

Die Java-Applikation ist entstanden, damit einzelne Schritte der Regelextraktion aus einem Neuronalen Netz auch praktisch nachvollzogen werden und interaktiv variiert werden können. Aus Gründen der Visualisierung wurde hierbei nur der reellwertige Netztypus 1-L-1 in der grafischen Oberfläche der konkreten exemplarischen Anwendung umgesetzt, d.h. ein Feedforward-Netz mit je einem Eingabe- und einem Ausgabeneuron sowie einer verborgenen Schicht mit interaktiv einstellbar vielen Neuronen. Die im Rahmen dieser Anwendung für die fachliche Logik zuständigen Klassen unterstützen gleichwohl allgemeinere Netztopologien und -konfigurationen und sind als eine Art Bibliothek generell einsetzbar.

Im Vordergrund der graphischen Beispielanwendung stehen die beiden Ansichten "Netz" und "Regeln", die auch in den folgenden beiden Bildschirmdarstellungen stellvertretend gezeigt werden.

Die erstgenannte Ansicht “Netz” stellt ein sehr einfaches 1-L-1-Netz mit reellwertigen Ein- und Ausgaben sowie Gewichten zur Verfügung. Der Parameter L, d.h. Die Anzahl der Neuronen in der verborgenen Schicht, wird beim Anlegen eines neuen Netzes abgefragt und vom Benutzer eingegeben. Die Gewichte des Netzes und der Eingabewert können interaktiv eingestellt werden, ebenso der Typ der Ausgabefunktion der inneren Neuronen. Aktuell wird hier mit den Typen “Sigmoide” und “Rampenfunktion” gearbeitet, zwischen denen via Checkbox-Auswahl umgeschaltet werden kann.

“Javanna” bietet die Möglichkeit, das einfache Backpropagation-Lernverfahren durchzuführen; hierzu kann in der Ansicht “Netzinfo” die Lernrate eingestellt werden. Netze können auch persistent gemacht werden, sie können in einer editierfähigen, reinen ASCII-Datei abgespeichert und aus einer solchen geladen werden. Das hierbei verwendet Abspeicherungsformat ist bewusst sehr elementar gehalten und wird in Anhang A7 kurz beschrieben.

In der vorliegenden Fassung trainiert das Backpropagation-Lernverfahren das Netz auf die Funktion $x \to \frac{1}{2}\sin(x) + \frac{1}{2}$. Die spezielle Modifikation der Sinus-Funktion wurde gewählt, damit insbesondere im Bereich um $x = 0$ ein gutes Lernverhalten sowohl mit der Sigmoiden als auch der Rampenfunktion möglich ist. In der unten gezeigten Ansicht “Netz” wird dem Benutzer auch die Soll-Ausgabe dieser Funktion zu dem jeweiligen Eingabewert angezeigt.

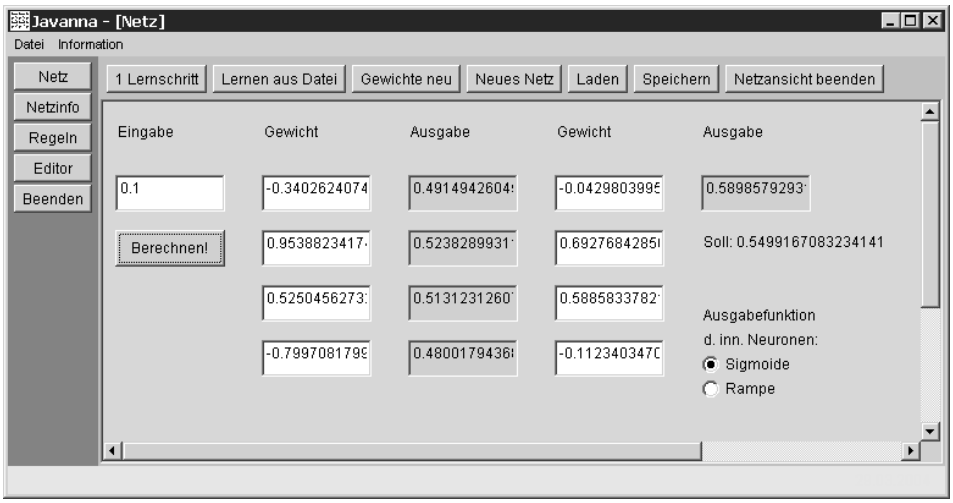


Abb. 7.10: Die Ansicht “Netz” innerhalb der Applikation

Die Ansicht “Regeln” ermöglicht es, die beiden Regeltypen “Äquidistante x-Unterteilung” und “Numerische Variation” interaktiv zu erproben (vgl. die Definitionen 7.1 und 7.2). Hierbei sind der x-Eingabebereich, die gewünschte Anzahl der Regeln und die angezeigte Genauigkeit einstellbar. Die extrahierten Regeln werden in dualer Form präsentiert: neben der “nahe-bei”- Formulierung, die einzelne reelle Werte verwendet, wird in Klammern auch die dazu korrespondierende Intervalldarstellung angegeben.

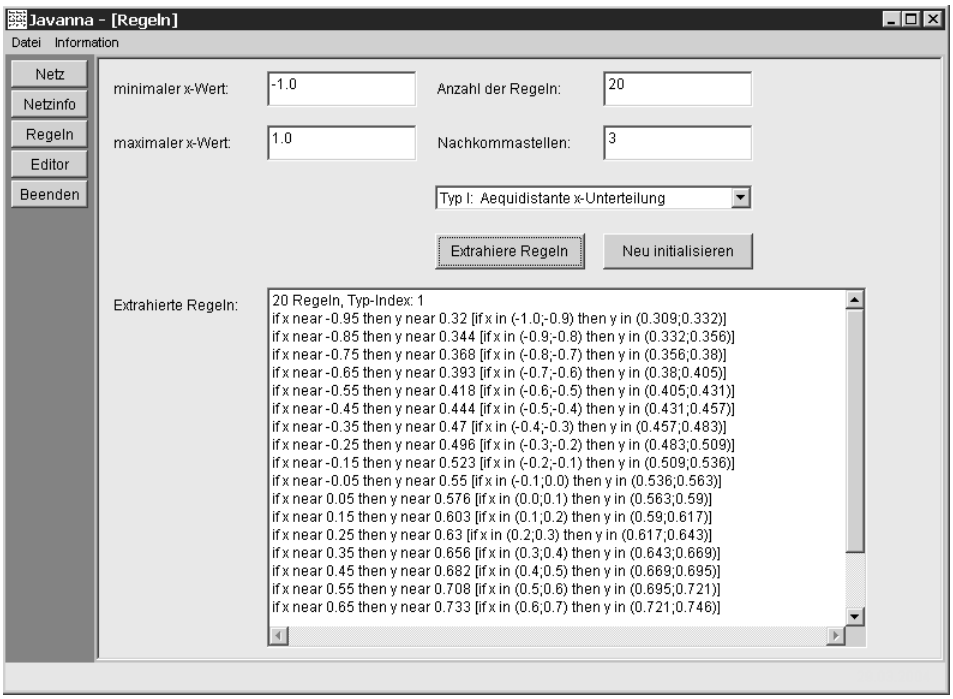


Abb. 7.11: Die Ansicht “Regeln” innerhalb der Applikation

Schließlich ist ergänzend ein kleiner Texteditor in die Applikation integriert, damit beispielsweise Netzdaten oder Regelbasen zwischengespeichert und editiert oder begleitende Notizen erstellt werden können.

7.4.2 Die Applikationsarchitektur

Die Anwendung “Javanna” wie auch das Package “javanna” sind objektorientiert konzipiert und realisiert. Eine Reihe von Klassen (z.B. Netz, Neuron, Verbindung) modelliert die entsprechende Realität eines Neuronalen Netzes. Mittels Vererbung werden spezielle Klassen implementiert, so erben die Klassen AusgabeNeuron und EingabeNeuron naturgemäß von der allgemeineren Klasse Neuron.

Ebenso liegen für die Darstellung von Regeln und Regelbasen entsprechend implementierte Klassen vor. Die abstrakten Klassen Regel und Regelbasis dienen dabei als Ausgangspunkt für konkrete Spezifikationen. So sind Regel1 und Regelbasis1 konkrete Klassen für den eindimensionalen Fall, der in der Anwendung Javanna zum Einsatz kommt. Mit Regel2 und Regelbasis2 liegen höherdimensionale Realisierungen vor; diese Klassen kommen in den Testsituationen zum Problem des Inversen Pendels zum Einsatz (6.21).

Die Hierarchie der wesentlichen Klassen ist in Anhang A7 dargestellt.

7.4.3 Dokumentation der Anwendung

Die Klassen aus dem Package javanna liegen dokumentiert in Form von HTML-Seiten vor, die mit dem Werkzeug *javadoc* aus dem Java Development Kit generiert worden sind.

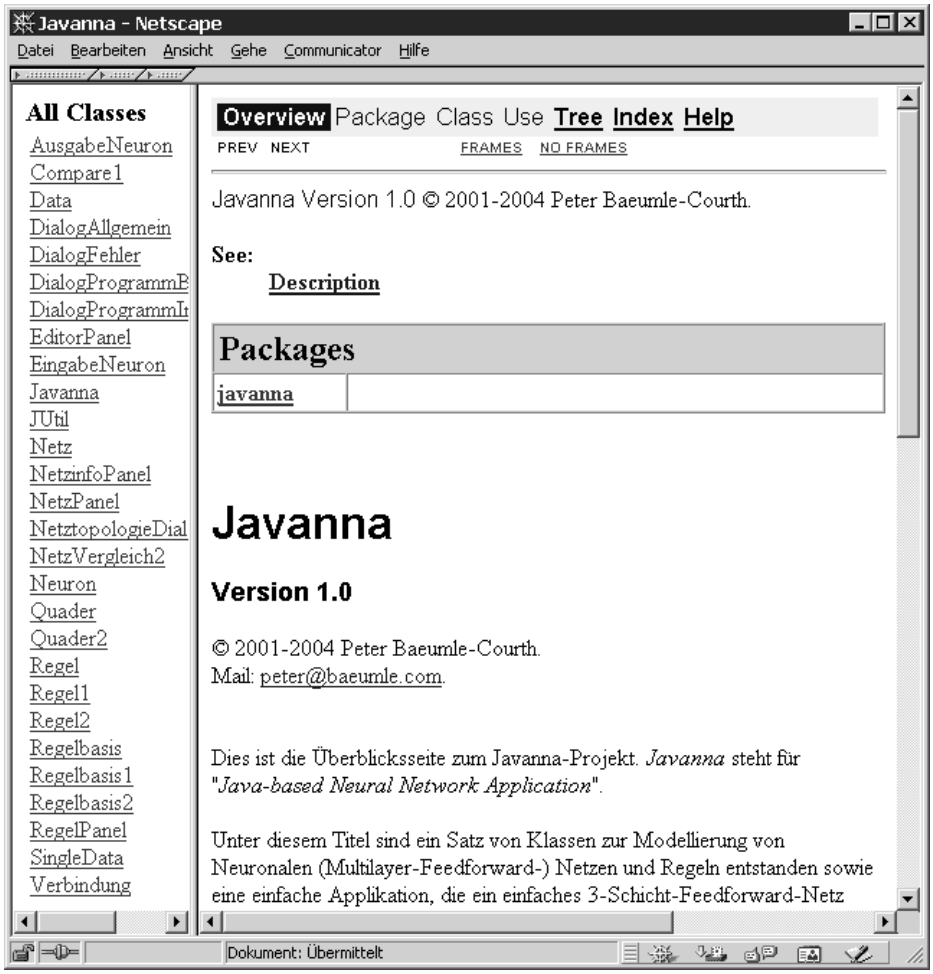


Abb. 7.12: Ansicht der mit javadoc generierten HTML-Dokumentation

Aus Platzgründen wird an dieser Stelle auf eine vollständige Wiedergabe dieser Dokumentation in schriftlicher Form verzichtet; diese findet sich in digitaler Form komplett auf der Begleit-CDROM zu dieser Arbeit. Anhang A7 (S. 192 ff) enthält darüber hinaus die Darstellung einer beispielhaften Programmnutzung der Anwendung “Javanna”.

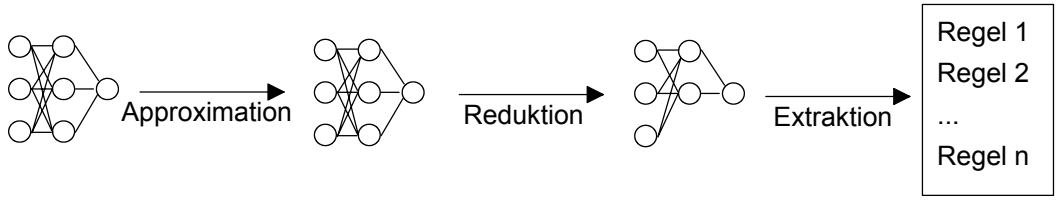
Fazit und Ausblick

In der vorliegenden Arbeit ging es um die Problemstellung, menschlich verständliche Regeln zu einem Neuronalen Netz anzugeben, das bedingt durch seine Architektur als Black Box agiert. Dabei war es Wunsch des Autors, eine möglichst generische Regelextraktions-methodik zu entwickeln, die keine gar zu einschränkenden Bedingungen an das Neuronale Netz stellen sollte.

Zahlreiche Ansätze, von denen einige in dieser Arbeit präsentiert wurden, nutzen das Zusammenspiel von Fuzzy-Entscheidungssystemen mit Neuronalen Netzen. Dabei geht es häufig um das Verbessern von Fuzzy-Controllern durch Neuronale Netze, indem geeignete Netze konstruiert werden. Auf diese Weise können Lernverfahren für Neuronale Netze zur Optimierung des Fuzzy-Controllers genutzt werden. Wie die Ausführungen zeigten, haben die hier beteiligten Netze jeweils einen ganz spezifischen Aufbau, der für die im Rahmen unserer Arbeit behandelte Aufgabenstellung zu einengend war.

Daher befasste sich diese Arbeit modulartig mit verschiedenen Komponenten, die gemeinsam oder auch einzeln zu einem besseren Verständnis der Wirkungsweise eines Neuronalen Netzes beitragen können.

Diese zentralen Etappen der Arbeit waren die Approximation der Sigmoiden durch die Rampenfunktion, die Reduktion der Neuronenanzahl in der verborgenen Schicht (bei dreischichtigen Feedforward-Netzen) sowie die Extraktion von Regeln aus einem Neuronalen Netz.



Durch die Approximation der Sigmoiden mittels der mathematisch bzw. numerisch wesentlich einfacheren Rampenfunktion gelang es, in gewissen Situationen die Anzahl der Neuronen in der verborgenen Schicht zu reduzieren. Aber auch der numerische Rechenaufwand für ein Neuronales Netz sinkt, wenn anstelle der auf der Exponentialfunktion basierenden Sigmoiden nur die stückweise lineare Rampenfunktion ermittelt werden muss. Güte-Aussagen über die entsprechenden Approximationen wurden ebenfalls im Rahmen der Arbeit formuliert.

Die Reduktion der Neuronenanzahl in der verborgenen Schicht wurde auch unabhängig von der Approximation durch die Rampenfunktion thematisiert. Mit dem Least Weight Algorithmus wurde ein Verfahren präsentiert, mit dem zu bekannten Güte-Niveaus die Anzahl der Neuronen reduziert werden kann.

Schließlich ermöglicht die dritte Komponente die eigentliche Regelextraktion; auch hierzu wurden verschiedene Algorithmen für verschiedene Netz-Situationen präsentiert, wobei der Algorithmus der schrittweisen Verbesserung (SV-Algorithmus) ohne stark einschränkende Bedingungen an das Neuronale Netz auskommt.

Diese drei Bausteine - die Approximation, die Reduktion und die Regelextraktion - können nun verschiedentlich miteinander kombiniert werden. Wie die numerischen Beispiele der Arbeit zeigten, ist es in manchen Situationen von Vorteil, die Sigmoide durch die Rampenfunktion zu ersetzen, also hier bereits eine erste Approximation vorzunehmen. Weiterhin kann es sinnvoll sein, Neuronale Netze mit sehr vielen Neuronen in der verborgenen Schicht zunächst zu reduzieren, wofür sich der Least Weight Algorithmus anbietet, bevor ein Regelextraktionsverfahren angesetzt wird.

Eine an diese Arbeit anknüpfende Forschung kann sich in natürlicher Weise vertiefend mit den drei Aspekten des Verfahrens - der Approximation, der Reduktion und der eigentlichen Regelextraktion - befassen.

Bei der Approximation der Sigmoiden, im vorliegenden Fall durch den Typus Rampenfunktion, kann untersucht werden, ob andere Näherungsfunktionen bessere Resultate erzielen lassen, wobei die numerische Komplexität nicht außer acht gelassen werden darf.

Für die Reduktion der Neuronenanzahl im Netz - und damit einer topologischen Vereinfachung - können weitere Ansätze untersucht werden. So muss beispielsweise nicht unbedingt aus n in gewisser Weise “ähnlichen” Neuronen genau ein substituierendes Neuron generiert werden: dies könnten auch zwei Neuronen sein, die sich die Arbeit gewissermaßen teilen.

Hinsichtlich der Regelextraktion kann es interessant sein, vor allem den höherdimensionalen Fall vertiefend zu betrachten. Das Variieren der bislang betrachteten Rechteckmengen hin zu anderen Typen von Eingabemengen (z.B. radial beschriebenen Kugelmengen) mag für gewisse Problemstellungen eine Verbesserung bedeuten ebenso wie eine Vorverarbeitung der Eingabedaten, d.h. evtl. kann eine Art von Koordinationtransformation vorgeschaltet werden.

Insgesamt kann festgehalten werden, dass eine Reihe von Fragestellungen rund um die hier erarbeitete Semantikbeschreibung eines Neuronalen Feedforward-Netzes die weitere Forschungsarbeit motivieren. Das Phasenmodell des entwickelten Verfahrens (Approximation, Reduktion und Regelextraktion) erleichtert mit seinem modularen Ansatz auch eine partielle Optimierung durch Ersetzen oder Verbessern einer einzelnen Komponente.

Abkürzungsverzeichnis

ANN
: Artificial Neural Network (siehe auch “KNN”)

ART
: Adaptive Resonanztheorie (s. S. 19)

DARPA
: Defense Advanced Research Projects Agency (s. S. 19)

FES
: Fuzzy-Entscheidungssystem

KI
: Künstliche Intelligenz

KNN
: Künstliches Neuronales Netz

LWA
: Least Weight Algorithmus (vgl. S. 131)

NN
: Neuronales Netz (siehe auch “KNN”)

O.B.d.A.
: Ohne Beschränkung der Allgemeinheit

PDP
: Parallel Distributed Computing (s. S. 19)

RBF
: Radial Basis Function, radiale Basisfunktion

SV, SV-*n*
: Abkürzung für den Algorithmus der “schrittweisen Verbesserung” (vgl. S. 144); die Schreibweise “SV-*n*” (mit $n > 0$) wird verwendet, wenn n Regeln mit dem Verfahren generiert werden sollen.

Notationen und Zeichenerklärung

❑	Zeichen für ein Beweisende (zur optischen Verdeutlichung)
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}_0^+$	Menge der nichtnegativen reellen Zahlen
$\overrightarrow{w} \cdot \overrightarrow{x}$	Skalarprodukt zweier Vektoren
Δw_{ij}	Veränderung des Gewichtswertes w_{ij}; allgemein ist Δz die Veränderung (Zuwachs oder Abnahme) der Größe z von einem Zeitpunkt zum nächsten: $z^{neu} := z^{alt} + \Delta z$.
$FM(X)$	Menge der Fuzzy-Mengen über X (siehe Definition 2.1 auf S. 43)
F	Menge der Fuzzy-Zahlen
$\hat{F}$	Menge der triangulären Fuzzy-Zahlen
L_2, $L_2(X)$	Menge der quadratintegrablen Funktion über einer gegebenen Grundmenge X
(a_m,a_l,a_r)	Darstellung einer triangulären Fuzzy-Zahl durch den Modalwert und die beiden Unschärfen (vgl. Def. 2.33, S. 51)
$[a_\lambda, a_m, a_\rho]$	Darstellung einer triangulären Fuzzy-Zahl durch die Trägergrenzen und den Modalwert (vgl. S. 52)
a_α	Niveaumenge einer Fuzzy-Menge (vgl. Def. 2.38, S. 53)
a_α^*	strenge Niveaumenge einer Fuzzy-Menge (vgl. Def. 2.38, S. 53)
$\delta_{\max}$	Maximale Abweichung der Rampenfunktion von der Sigmoiden (vgl. Bem. 6.11 auf S. 116)
$N(K,L,M,\mathbb{R})$	Menge der Neuronalen Feedforward-Netze mit drei Schichten, K Neuronen in der Eingabeschicht, L Neuronen in der verborgenen Schicht, M Neuronen in der Ausgabeschicht und reellwertigen Ausgaben
$N(K,L,1,\mathbb{R}^M)$	Menge der Neuronalen Feedforward-Netze mit drei Schichten, K Neuronen in der Eingabeschicht, L Neuronen in der verborgenen Schicht, einem Neuron in der Ausgabeschicht und M-dimensionalen (vektorwertigen) Ausgaben

Literaturverzeichnis

[Alie2000] Aliev, R., Bonfig, K. W., Aliew, F.
Soft Computing. Eine grundlegende Einführung
Berlin, 2000

[Andr1995] Andrews, R., Diederich, J., Tickle, A. B.
Survey and critique of techniques for extracting rules from trained artificial neural networks
Knowledge Based Systems vol. 8 no. 6 pp 373-389, 1995

[Andr2002] Andrews, R., Geva, S.
Rule extraction from local cluster neural nets
Neurocomputing, vol. 47, pp 1-20, 2002

[Bach1966] Bachman, G., Narici, L.
Functional Analysis
New York, 1966

[Barr2001] Barrow, J.D.
Ein Himmel voller Zahlen.
Hamburg, 2001 (2.Auflage)

[Biet1998] Biethahn, J., Hönerloh, A., Kuhl, J., Leisewitz, M.-C., Nissen, V., Tietze, M. [Hrsg.]
Betriebswirtschaftliche Anwendungen des Soft Computing
Braunschweig, 1998

[Böhm1993] Böhme, G.
Fuzzy-Logik
Berlin, 1993

[Bolo2000] Bologna, G.
Rule Extraction from a Multi Layer Perceptron with Staircase Activation Functions
IJCNN 2000 (International Joint Conference on Neural Networks), pp 419-424, 2000.

[Borg2003] Borgelt, C., Klawonn, F., Kruse, R., Nauck, D.
Neuro-Fuzzy-Systeme
Wiesbaden, 2003

[Brau1995] Brause, R.
Neuronale Netze
Stuttgart, 1995

[Cast2000] Castellano, G., Fanelli, A. M.
Fuzzy inference and rule extraction using a neural network
Neural Network World Journal 3, pp 361.371, 2000

[Cohe1983] Cohen, M.A., Grossberg, S.
Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks
IEEE Transactions on Systems, Man and Cybernetics, vol. 13, pp 815-826, 1983

[Dewd1995] Dewdney, A.K.
Der Turing Omnibus
Heidelberg, 1995

[Dubo1978] Dubois, D., Prade, H.
Operations on Fuzzy Numbers
International Journal of Systems Sciences, vol. 9, pp 613-626, 1978

[Dubo1980] Dubois, D., Prade, H.
Fuzzy Sets and Systems: Theory and Applications
New York, 1980

[Dubo1985] Dubois, D., Prade, H.
A Review of Fuzzy Set Aggregation Connectives
Information Sciences, vol. 36, pp 36-121, 1985

[Feur1995] Feuring, T.
Fuzzy-Neuronale Netze - Von kooperativen über hybride zu fusionierten vage-konnektionistischen Systemen
Dissertation, Westfälische Wilhelms-Universität Münster, 1995

[Grau1992] Grauel, A.
Neuronale Netze: Grundlagen und mathematische Modellierung
Mannheim, 1992

[Guil2001] Guillaume, S., Charnomordic, B.
Knowledge discovery for control purposes in food industry databases
Fuzzy Sets and Systems, vol. 122, pp 487-497, 2001

[Hamm2002] Hammer, B., Rechtien, A., Strickert, M., Villmann, T.
Rule Extraction from Self-Organizing Networks
Univ. of Osnabrück, Germany, 2002
Publ. im Internet: www.informatik.uni-osnabrueck.de/marc/papers/icannbb02.pdf
Abruf: 26.10.2004.

[Haya1992] Hayashi, Y., Buckley, J.J., Czogala, E.
Approximation between Fuzzy Expert Systems and Neural Networks
Proc. of the 2nd International Conference on Fuzzy Logic & Neural Networks, Iizuka, pp 135-139, 1992

[Hebb1949] Hebb, D.O.
The Organization of Behaviour
New York, 1949

[Hech1990] Hecht-Nielsen, R.
Neurocomputing
Reading, 1990

[Hell1997] Hellmich, R.
Einführung in intelligente Softwaretechniken
München, 1997

[Hopf1985] Hopfield, J.J., Tank, D.W.
Neural Computation of Decisions in Optimization Problems
Biological Cybernetics, vol. 52, pp 141-152, 1985

[Horn1991] Hornik, K.
Approximation capabilities of mutilayer feedforward networks
Neural Networks, vol. 4, pp 551-557, 1991.

[Hunt1996] Hunt, K. J.
Functional Equivalence Between Radial Basis Function Network and Fuzzy Inference System
IEEE Transactions on Neural Networks, vol. 3, pp 776-781, 1996

[Ishi1999] Ishibuchi, H., Nii, M., Tanaka, K.
Fuzzy-Arithmetic-Based Approach for Extracting Positive and Negative Linguistic Rules from Trained Neural Networks
IEEE International Fuzzy Systems Conference Proceedings, Aug. 22-25, 1999, Seoul, Korea, pp III-1387 - III-1387

[Jang1993] Jang, J. S. R.
Functional Equivalence Between Radial Basis Function Network and Fuzzy Inference System
IEEE Transactions on Neural Networks, vol. 1, pp 156-158, 1993

[Jang1993a] Jang, R., Shing, J.
ANFIS - Adaptive-Network-Bases Fuzzy Inference System
IEEE Transactions on Systems, Man and Cybernetics, vol 23, no. 3, pp 665-685, 1993

[Jian2002] Jiang, Y., Zhou, Z., Chen, Z.
Rule Learning based on Neural Network Ensemble
Proc. of the International Joint Conference on Neural Networks, Honolulu, 2002, vol. 2, pp 1416-1420

[Jung1994] Jungnickel, D.
Graphen, Netzwerke und Algorithmen
Mannheim, 1994

[Kasa1999] Kasabov, N., Woodford, B.
Rule Insertion and Rule Extraction from Evolving Fuzzy Neural Networks: Algorithms and Applications for Building Adaptive, Intelligent Expert Systems
IEEE International Fuzzy Systems Conference Proceedings, Aug. 22-25, 1999, Seoul, Korea, pp III-1406 - III-1411

[Klir1988] Klir, G.J., Folger, T.A.
Fuzzy Sets, Uncertainty, and Information
Englewood Cliffs, 1988

[Koho1982] Kohonen, T.
Self-organized formation of topologically correct feature maps
Biological Cybernetics, vol. 43, pp 59-69, 1982

[Koho1984] Kohonen, T.
Self-Organization and Associative Memory
Berlin, 1984

[Koho1997] Kohonen, T.
Self-Organizing Maps
Berlin, 1997

[Kolm1957] Kolmogorov, A. N.
On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition [auf Russisch]
Doklady Akad. Nauk USSR, vol. 114, pp 953-956, 1957 (zitiert nach [Hech1990])

[Kosk1987] Kosko, B.
Foundations of Fuzzy Estimation Theory
Ph. D. Thesis, Irvine, 1987 (zit. nach [Feur1995])

[Krat1993] Kratzer, K. P.
Neuronale Netze: Grundlagen und Anwendungen
München, 1993

[Küfe2001] Küfer, Th.
Modellierung und Optimierung von Sugeno-Controllern mit RBF-Netzen
Seminarausarbeitung, Universität Münster, 2001

[LinL1991] Lin, C.-T., Lee, C. S. G.
Neural-Network-Based Fuzzy Logic Control and Decision System
IEEE Transactions on Computers, Vol. 40, no. 12, 1991

[LinL1993] Lin, C.-T., Lee, C. S. G.
Reinforcement Structure/Parameter Learning for Neural-Network based Fuzzy Logic Control Systems,
Proceedings of the IEEE International Conference on Fuzzy Systems, San Francisco CA, pp 88-93, 1993

[Lipp1995a] Lippe, W.-M., Tenhagen, A., Feuring
A hybrid learning rule for a feedworward network
International Journal of Artificial Intelligence Tools, vol. 3, pp 407-416, 1995

[Lipp1995b] Lippe, W.-M., Mischke, L., Feuring, Th.
Supervised Learning in Fuzzy Neural Networks
Proceedings of the INNS World Congress on Neural Networks, Washington D.C., 1995

[Lipp1995c] Lippe, W.-M., Büscher, Th., Feuring, Th.
A fully fuzzified neural network based on the backpropagation algorithm
Proceedings of the INNS World Congress on Neural Networks, Washington D.C., 1995

[Lipp1996] Lippe, W.-M., Tenhagen, A., Feuring, Th., Lahl, H. Henke, D.
Separation of Hashish Signatures with Neural Networks
in: C.H. Dagli et al (Eds): Intelligent Engineering Systems through Artificial Neural Networks, vol. 6, pp 989-994, 1996.

[Lipp1998] Lippe, W.-M., Klingelbiel, A., Tenhagen, A.
A Gradient Descent Learning Rule for Fuzzy Neural Networks
Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing, Cancun, IASTED/ACTA Press, Anaheim, pp 493-497, 1998

[Lipp1999] Lippe, W.-M., Tenhagen, A., Niendieck, St.
Representing and Optimizing Fuzzy Controllers by Neural Networks
Proc. of the 8th IEEE International Conference on Fuzzy Systems, Seoul, 1999

[Lipp2004] Lippe, W.-M.
Skriptum Neuronale Netze
Münster, 2004

[Mamd1974] Mamdani, E.H.
Applications of Fuzzy Algorithms for Simple Dynamic Plant
Proceedings of the IEEE, vol. 121, pp 1585-1588, 1974

[Mamd1975] Mamdani, E.H., Assilian, S.
An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller
International Journal of Man-Machines Studies, vol. 7, pp 1-13, 1975

[McCl1986] McClelland, J.L., Rumelhart, D.E.
Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1 & 2
Cambridge, 1986

[McCu1943] McCulloch, W.S., Pitts, W.
A logical calculus of the ideas immanent in nervous activity
Bulletin of Mathematical Biophysics 5, pp 115-133, 1943

[Mins1969] Minsky, M.L., Papert, S.A.
Perceptrons
Cambridge, 1969

[Mora2000] Moraga, C.
Neuro-Fuzzy Modeling with Standard Feedforward Neural Networks
Proceedings European Symposium on Intelligent Techniques, Aachen, 2000

[Mora2000a] Moraga, C., Temme, K.-H.
S-Neural Networks are Fuzzy Models
Proceedings International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, pp 1518-1523, Madrid, 2000

[Mora2000b] Moraga, C.
Neuro-Fuzzy Modeling of Compensating Systems
in: Sincak, P., Vascak, J. (Eds.): Quo vadis Computational Intelligence?
Heidelberg, 2000

[Mora2001] Moraga, C.
Neuro-Fuzzy Modeling between minimum and maximum
Proc. Workshop on Computational Intelligency, Theory and Applications, pp 21-28, Nis (Yugoslavia), 2001

[Mora2002] Moraga, C.
Neuro-Fuzzy Systems for Rule Extraction
First International ICSC Congress on Neuro Fuzzy Technologies, La Habana, 2002

[Nauc1994] Nauck, D., Klawonn, F., Kruse, R.
Neuronale Netze und Fuzzy Systeme
Wiesbaden, 1994
(Eine spätere Auflage ist unter dem Titel “Neuro-Fuzzy-Systeme” als [Borg2003] in diesem Literaturverzeichnis ebenfalls berücksichtigt.)

[Neum1998] Neumann, J.
Classification and Evaluation of Algorithms for Rule Extraction From Artificial Neural Networks
Summer Project, Univ. of Edinburgh, 1998.

[Nien1998] Niendieck, St.
Eine universelle Repräsentation von Fuzzy-Controllern durch Neuronale Netze
Diplomarbeit, Westfälische Wilhelms-Universität Münster, 1998

[Nien2003] Niendieck, St.
Optimierung von Fuzzy-Controllern - Von Untersuchungen hybrider Neuro-Fuzzy-Systeme zum Entwurf des universellen konnektionistischen Modells MFOS
(Münsteraner-Fuzzy-Optimierungs- System)
Dissertation, Westfälische Wilhelms-Universität Münster, 2003

[Nils1965] Nilsson, N.
Learning Machines
New York, 1965

[Patt1997] Patterson, D.
Künstliche neuronale Netze
München, 1997

[Plög2000] Plöger, H.
Regelextraktion aus Neuronalen Netzen
Seminarausarbeitung, Westfälische Wilhelms-Universität Münster, 2000

[Pome1991] Pomerleau, D.A.
Rapidly adapting artificial neural networks for autonomous navigation
in: Lippmann, R.P., Moody, J.E., Touretzky, D.S. (Eds.): Advances in Neural Information Processing Systems, vol. 3, pp 429-435, 1991

[PopH1994] Pop, E., Hayward, R., Diederich, J.
RULENEG: extracting rules from a trained ANN by stepwise negation
QUT NRC, 1994.

[Quek2001] Quek, C., Zhou, R.W.
The POP learning algorithmus: reducing work in identifying fuzzy rules
Neural Networks, vol. 14, pp 1431-1445, 2001

[Resc1969] Rescher, N.
Many-Valued Logic
New York, 1969

[Rogg2001] Rogge, M.
Ein polynomialer Regelextraktions-Algorithmus
Seminarausarbeitung, Westfälische Wilhelms-Universität Münster, 2001

[Roja1996] Rojas, R.
Theorie der neuronalen Netze
Berlin, 1996

[Rose1958] Rosenblatt, F.
The perceptron: a probabilistic model for information storage and organization in the brain
Psychological Review 65, pp 386-408, 1958

[Rose1959] Rosenblatt, F.
Principles of Neurodynamics
New York, 1959

[Rume1986] Rumelhart, D.E., Hinton, G.E., Williams, R.J.
Learning internal representations by error propagation
in: Rumelhart, D.E., McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, pp 318-362, Cambridge, 1986

[Russ1923] Russell, B.
Vagueness
The Australasian Journal of Psychology and Philosophy, vol. 1, pp 84-92, 1923

[Sait2002] Saito, K., Nakano, R.
Extracting regression rules from neural networks
Neural Networks 15, pp 1279-1288, 2002

[Seti2000] Setiono, R., Leow, W. K.
FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks
Applied Intelligence 12, 1-2, pp 15-25, 2000

[Snyd2001] Snyders, S., Omlin, C. W.
Rule Extraction from Knowledge-Based Neural Networks with Adaptive Inductive Bias
ICONIP 2001, Shanghai

[Stof2000] Stoffers, Th.
Regelextraktion aus Künstlichen Neuronalen Netzen
Seminarausarbeitung, Westfälische Wilhelms-Universität Münster, 2000

[Suge1985] Sugeno, M., Takagi, T.
Fuzzy identification of systems and ist applications to modeling and control
IEEE Transactions on Systems, Man & Cybernetics, Vol. 15, pp 116-132, 1985

[Suge1985a] Sugeno, M.
An introductory survey of fuzzy control
Inform. Science, Vol. 36, pp 59-83, 1985

[Sun2001] Sun, R.
Beyond Simple Rule Extraction: Acquiring Planning Knowledge from Neural Networks
ICONIP 2001, Shanghai

[Temm1999] Temme, K.-H., Heider, R., Moraga, C.
Generalized Neural Networks for Fuzzy Modeling
Proceedings of the International Joint Conference of the Spanish and European Societies for Fuzzy Logic and Technology, Palma de Mallorca, 1999

[Tenh2000] Tenhagen, A.
Optimierung von Fuzzy-Entscheidungssystemen mittels konnektionistischer Methoden
Dissertation, Westfälische Wilhelms-Universität Münster, 2000

[Thru1995] Thrun, S.
Extracting Rules from Artificial Neural Networks with Distributed Representations
Advances in Neural Information Processing Systems 7, pp 505-512, 1995

[Tick1998] Tickle, A. B., Andrews, R., Golea, M., Diederich, J.
The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks.
IEEE Transactions on Neural Networks, 9, pp 1057-1068

[Towe1993] Towell, G., Shavlik, J.
The extraction of refined rules from knowledge-based neural networks
Machine Learning, vol 13, no 1, pp 71-101, 1993.

[Tsuk2000] Tsukimoto, H.
Extracting Rules from Trained Neural Networks
IEEE Transactions on Neural Networks, Vol. 11 No. 2, pp 377-389, 2000

[Weic2002] Weicker, K.
Evolutionäre Algorithmen
Stuttgart, 2002

[Werb1974] Werbos, P.J.
Beyond regression: new tools for prediction and analysis in the behavioral sciences
Ph.D. Thesis, Harvard University, Cambridge, 1974

[Widr1960] Widrow, B., Hoff, M.E.
Adaptive switching circuits
1960 IRE WESCON Convention Record, New York, pp 96-104, 1960

[Wu1999] Wu, A. I., Tam, P. K. S.
A Simplified Model of Fuzzy Inference System Constructed by Using RBF Neurons
IEEE International Fuzzy Systems Conference Proceedings, Aug. 22-25, 1999, Seoul, Korea, pp I-50 - I-54

[Yi1992] Yi, H.-J., Oh, K.-W.
Neural Network-based Fuzzy Production Rule Generation and ist Application to an Approximate Reasoning Approach
Proc. of the 2nd International Conference on Fuzzy Logic & Neural Networks, Iizuka, pp 333-336, 1992

[Zade1965] Zadeh, L.
Fuzzy Sets
Information and Control, no. 8, pp 338-353, 1965

[Zade1972] Zadeh, L.
A Rationale for Fuzzy Control
Journal of Dynamic Systems, Measurement, and Control, Vol. 94 No. 6, pp 3-4, 1972

[Zade1973] Zadeh, L.
Outline of a new approach to the analysis of complex systems and decision processes
IEEE Transactions on Systems, Man & Cybernetics, Vol. 3, pp 28-44, 1973

[Zell1997] Zell, A.
Simulation neuronaler Netze
München, 1997 (2.Auflage)

[Zhao2001] Zhao, Z., De Souza, R.
Fuzzy rule learning during simulation of manufacturing resources
Fuzzy Sets and Systems, vol. 122, pp 469-485, 2001

[Zhou2000] Zhou, Z.-H., Chen, S.-F., Chen, Z.-Q.
A Statistics based Approach for Extracting Priority Rules from Trained Neural Networks
IJCNN 2000 (International Joint Conference on Neural Networks), pp 401-406, 2000

Anhang

Nachfolgend finden sich eine Reihe von numerischen Beispielen zu den in der vorliegenden Arbeit dargestellten Verfahren. Weitere Ergänzungen sind auf der Begleit-CDROM enthalten; deren Inhalt wird dort in den Dateien inhalt.txt bzw. inhalt.html beschrieben.

Anhang A1 - Beispiel Sinus-Approximation (vgl. S. 117)

Das nachstehende Ablauflisting stammt von dem Java-Programm “Compare1” und illustriert ein sehr einfaches 1-3-1-Netz, das auf die modifizierte Sinus-Funktion

$$x \rightarrow \tfrac{1}{2}(1 + \sin(x))$$

trainiert worden ist. Sh. hierzu S. 108.

```
java javanna.Compare1 sinus-gelernt-100.net TEST.DAT
Compare1  (c) 2004 P. Baeumle-Courth
* Diagnose-Ausgabe des Netzes *
Anzahl Schichten: 3
Schicht 0: 1 Neuron/en.
  Neuron[0;0]-Gewichte:
  0.5104587  -1.0993515  0.050280392
Schicht 1: 3 Neuron/en.
  Neuron[1;0]-Gewichte:
  1.3217824
  Neuron[1;1]-Gewichte:
  -1.0596993
  Neuron[1;2]-Gewichte:
  0.73582333
Schicht 2: 1 Neuron/en.
x: -1.0  Sigmoide:  0.059524983  Rampe: 0.16665605
                    Abweichung [abs./rel.]:    0.107131064 1.7997664 [= 179.9%]
x: -0.9  Sigmoide:  0.09879205  Rampe:  0.19988579
                    Abweichung [abs./rel.]:    0.10109374 1.0232984
x: -0.8  Sigmoide:  0.13940078  Rampe:  0.2331155
                    Abweichung [abs./rel.]:    0.093714714 0.6722682
x: -0.7  Sigmoide:  0.1812717  Rampe:   0.26634526
                    Abweichung [abs./rel.]:    0.08507356 0.46931517
x: -0.6  Sigmoide:  0.22430494  Rampe:  0.29957482
                    Abweichung [abs./rel.]:    0.07526988 0.3355694
x: -0.5  Sigmoide:  0.26838055  Rampe:  0.33280456
                    Abweichung [abs./rel.]:    0.06442401 0.24004723
x: -0.4  Sigmoide:  0.31335896  Rampe:  0.3660343
                    Abweichung [abs./rel.]:    0.052675337 0.16809903
x: -0.3  Sigmoide:  0.35908324  Rampe:  0.399264
                    Abweichung [abs./rel.]:    0.040180773 0.11189821
x: -0.2  Sigmoide:  0.40538126  Rampe:  0.43249375
                    Abweichung [abs./rel.]:    0.027112484 0.06688144
x: -0.1  Sigmoide:  0.45206872  Rampe:  0.46572345
                    Abweichung [abs./rel.]:    0.013654739 0.030205008
```

```
x: 0.0  Sigmoide:   0.4989532  Rampe:   0.4989532
                    Abweichung [abs./rel.]:    0.0 0.0
x: 0.1  Sigmoide:   0.5458375  Rampe:   0.53218293
                    Abweichung [abs./rel.]:    0.01365459 0.02501585
x: 0.2  Sigmoide:   0.5925251  Rampe:   0.56541264
                    Abweichung [abs./rel.]:    0.027112484 0.045757525
x: 0.3  Sigmoide:   0.63882315  Rampe:  0.59864235
                    Abweichung [abs./rel.]:    0.040180802 0.06289816
x: 0.4  Sigmoide:   0.6845475  Rampe:   0.63187206
                    Abweichung [abs./rel.]:    0.052675426 0.07694927
x: 0.5  Sigmoide:   0.7295258  Rampe:   0.66510177
                    Abweichung [abs./rel.]:    0.06442404 0.088309474
x: 0.6  Sigmoide:   0.7736014  Rampe:   0.69833153
                    Abweichung [abs./rel.]:    0.07526988 0.09729801
x: 0.7  Sigmoide:   0.81663465  Rampe:  0.7315611
                    Abweichung [abs./rel.]:    0.08507353 0.104175754
x: 0.8  Sigmoide:   0.85850567  Rampe:  0.7647909
                    Abweichung [abs./rel.]:    0.09371477 0.10916034
x: 0.9  Sigmoide:   0.8991144  Rampe:   0.7980206
                    Abweichung [abs./rel.]:    0.10109377 0.112437055
x: 1.0  Sigmoide:   0.93838143  Rampe:  0.8312503
                    Abweichung [abs./rel.]:    0.10713112 0.11416586

Minimale betragsmaessige Abweichung:                        0.0
Maximale betragsmaessige Abweichung:                        0.10713112
Durchschnittliche betragsmaessige Abweichung:               0.06288861
Minimale betragsmaessige relative Abweichung:               0.0
Maximale betragsmaessige relative Abweichung:               1.7997664
Durchschnittliche betragsmaessige relative Abweichung:    0.27397695
Datenumfang:                                                21
```

Anhang A2 - Das Inverse Pendel

Zu dem in Beispiel 6.21 (S. 121) dargestellten Problem des inversen Pendels wurden numerische Simulationen vorgenommen. Die nachstehende Übersichtstabelle zeigt für 250 Durchläufe jeweils die mittleren quadratischen Fehler (MSE) verschiedener Netze; die numerischen Werte wurden zur besseren Lesbarkeit auf vier Nachkommastellen gerundet. Jede Spalte repräsentiert ein bestimmtes Neuronales Netz.

Spalte A: Das auf das Inverse-Pendel-Problem trainierte 2-16-1-Ausgangsnetz (mit der sigmoiden Ausgabefunktion, vgl. Bem. 6.7 auf S. 111);
Spalte B: Zum Vergleich dieselbe Netzstruktur wie in A, hier allerdings mit der Rampenfunktion berechnet (vgl. Definition 6.6). (Für die Praxis bedeutet die Berechnung der Rampenfunktion einen deutlich geringeren Prozessor-Aufwand als die der sigmoiden Funktion.)
Spalte C: Das 2-1-1-Netz gemäß Satz 6.16 mit der Rampenfunktion.
Spalte D: Zum Vergleich wiederum dieselbe Netzstruktur wie in C, diesmal mit der sigmoiden Ausgabefunktion.
Spalte E: Das 2-2-1-Netz gemäß Satz 6.20 mit der Rampenfunktion.
Spalte F: Zum Vergleich dieselbe Netzstruktur wie in E, wieder mit der sigmoiden Ausgabefunktion.
Spalten G und H: das 2-8-1-Netz, das gemäß Gruppenclusterung aus dem 2-16-1-Ausgangsnetz entsteht; aufgetragen sind wieder die mittleren quadratischen Fehler unter Verwendung von Rampen- (G) bzw. sigmoider (H) Ausgabefunktion.
Spalten I und J: das 2-8-1-Netz, das gemäß statischer Intervallclusterung aus dem 2-16-1-Ausgangsnetz entsteht; aufgetragen sind die mittleren quadratischen Fehler mit Rampen- (Spalte I) bzw. sigmoider (J) Ausgabefunktion. “Statische Intervallclusterung” bedeutet hier, dass jeweils Intervalle der Breite 1 (symmetrisch um die 0) generiert werden. Hinzu kommen “außen” die beiden semi-unendlichen (Rest-)Intervalle.
Spalten K und L: hier sind die 2-8-1-Netze aufgetragen (mit Rampen- bzw. sigmoider Ausgabefunktion), die gemäß einer “dynamischen Intervallclusterung” erzeugt worden sind. Hierbei wurde als Grundbereich das Intervall von dem kleinsten bis zum größten Gewicht zum Ausgabeneuron verwendet und in acht gleichbreite Intervalle aufgeteilt.

Durchlauf	A	B	C	D	E	F	G	H	I	J	K	L
1	0,1209	0,1394	0,5396	0,5398	0,3228	0,5009	0,3228	0,4232	0,1394	0,1209	0,1394	0,1209
2	0,1083	0,1115	0,5672	0,5672	0,1334	0,1341	0,1334	0,1340	0,1115	0,1083	0,1115	0,1083
3	0,1230	0,1576	0,5139	0,5141	0,1599	0,1941	0,1488	0,1462	0,1576	0,1201	0,1576	0,1225
4	0,1360	0,1761	0,4943	0,4943	0,2614	0,4558	0,2594	0,3938	0,2537	0,3208	0,2594	0,3951
5	0,1357	0,1704	0,4972	0,4975	0,2447	0,3657	0,2413	0,3017	0,2469	0,2971	0,2469	0,2454
6	0,1304	0,1707	0,5122	0,5128	0,2697	0,4096	0,2696	0,3283	0,2697	0,4013	0,2893	0,3358
7	0,1225	0,1529	0,5243	0,5249	0,1910	0,1689	0,2335	0,3149	0,2656	0,3082	0,2607	0,2389
8	0,1413	0,1835	0,4972	0,4979	0,1984	0,2552	0,2204	0,1902	0,2811	0,3146	0,1835	0,1410
9	0,1317	0,1749	0,5016	0,5015	0,2926	0,4416	0,2855	0,3568	0,2865	0,3552	0,2855	0,3540
10	0,1249	0,1451	0,5212	0,5213	0,2568	0,2252	0,1625	0,1724	0,1684	0,1815	0,1684	0,1883
11	0,1371	0,1795	0,5038	0,5043	0,1994	0,2475	0,2169	0,1893	0,2494	0,2742	0,1788	0,1371

12	0,1193	0,1440	0,5457	0,5458	0,2660	0,4039	0,2660	0,3351	0,2833	0,2582	0,2608	0,2886
13	0,1133	0,1420	0,5440	0,5447	0,2047	0,2360	0,1727	0,1622	0,3264	0,2803	0,2427	0,2019
14	0,1263	0,1652	0,5165	0,5166	0,1939	0,1864	0,2025	0,2185	0,3954	0,4778	0,2251	0,1936
15	0,1143	0,1326	0,5520	0,5522	0,1932	0,1629	0,1446	0,1720	0,1721	0,2031	0,1721	0,2044
16	0,1426	0,1696	0,4887	0,4897	0,1615	0,1781	0,1899	0,1829	0,1712	0,1717	0,1712	0,1608
17	0,1234	0,1385	0,5268	0,5270	0,2012	0,2352	0,1621	0,1474	0,1385	0,1245	0,1385	0,1239
18	0,1267	0,1424	0,5245	0,5247	0,2857	0,3861	0,2857	0,3519	0,1424	0,1273	0,1424	0,1272
19	0,1203	0,1540	0,5234	0,5234	0,1550	0,1730	0,1654	0,1820	0,1527	0,1234	0,1558	0,1223
20	0,1202	0,1525	0,5330	0,5338	0,2249	0,3988	0,1925	0,2670	0,2197	0,3078	0,2006	0,3552
21	0,1290	0,1655	0,5098	0,5103	0,1869	0,1581	0,1702	0,2005	0,2274	0,2067	0,2142	0,3012
22	0,1082	0,1372	0,5588	0,5588	0,3352	0,3415	0,4005	0,3747	0,1370	0,1072	0,1370	0,1072
23	0,1275	0,1622	0,5011	0,5020	0,1949	0,1399	0,1458	0,1322	0,1700	0,1629	0,1700	0,1500
24	0,1032	0,1060	0,5903	0,5903	0,1058	0,1027	0,1058	0,1026	0,1060	0,1029	0,1060	0,1029
25	0,1269	0,1411	0,5257	0,5259	0,2637	0,5082	0,2637	0,4142	0,1411	0,1303	0,2250	0,2881
26	0,1185	0,1336	0,5415	0,5417	0,1540	0,1631	0,1563	0,1858	0,2274	0,2823	0,2274	0,2836
27	0,1041	0,1060	0,5858	0,5858	0,1067	0,1057	0,1059	0,1034	0,1060	0,1036	0,1060	0,1042
28	0,1354	0,1718	0,4947	0,4952	0,2352	0,3819	0,2352	0,3522	0,1965	0,2181	0,1939	0,1637
29	0,1329	0,1718	0,5040	0,5043	0,1802	0,2405	0,2751	0,2987	0,4252	0,4558	0,2326	0,2003
30	0,1188	0,1507	0,5311	0,5316	0,1811	0,1695	0,1679	0,1642	0,2578	0,2152	0,2315	0,1796
31	0,1112	0,1144	0,5569	0,5569	0,1145	0,1106	0,1142	0,1105	0,1144	0,1113	0,1144	0,1111
32	0,1244	0,1615	0,5211	0,5213	0,2787	0,4770	0,2674	0,3553	0,2761	0,3334	0,2585	0,3837
33	0,1253	0,1617	0,5225	0,5234	0,2205	0,1686	0,1915	0,2143	0,2173	0,1995	0,2002	0,2278
34	0,1099	0,1130	0,5619	0,5619	0,1556	0,1546	0,1243	0,1232	0,1130	0,1098	0,1130	0,1097
35	0,1239	0,1555	0,5162	0,5162	0,1952	0,1495	0,1745	0,1724	0,2157	0,2282	0,1553	0,1210
36	0,1331	0,1740	0,5113	0,5120	0,2188	0,3062	0,2726	0,2388	0,1734	0,1290	0,1734	0,1311
37	0,1163	0,1385	0,5458	0,5460	0,2040	0,1630	0,1340	0,1269	0,1385	0,1159	0,1385	0,1163
38	0,1144	0,1173	0,5490	0,5490	0,3295	0,3999	0,2069	0,2067	0,1216	0,1276	0,1173	0,1145
39	0,1307	0,1623	0,5125	0,5131	0,2086	0,2927	0,1870	0,2600	0,1568	0,2283	0,1572	0,2212
40	0,1230	0,1379	0,5394	0,5396	0,3508	0,4124	0,1500	0,1290	0,1379	0,1227	0,1379	0,1227
41	0,1324	0,1705	0,5027	0,5031	0,1645	0,1978	0,2893	0,2882	0,5669	0,6683	0,2045	0,1705
42	0,1219	0,1555	0,5186	0,5185	0,1542	0,1726	0,1668	0,1741	0,1974	0,1844	0,1886	0,1503
43	0,1161	0,1380	0,5483	0,5486	0,3371	0,4023	0,3268	0,3786	0,1380	0,1157	0,1380	0,1159
44	0,1187	0,1344	0,5503	0,5505	0,2596	0,3509	0,3711	0,4690	0,4070	0,5149	0,4070	0,5149
45	0,1257	0,1546	0,5097	0,5106	0,1900	0,2218	0,2455	0,2500	0,1544	0,1225	0,1656	0,1470
46	0,1191	0,1494	0,5391	0,5393	0,2070	0,1963	0,2328	0,3037	0,1822	0,1836	0,2745	0,4040
47	0,1322	0,1671	0,5052	0,5055	0,2432	0,3852	0,2364	0,2907	0,2623	0,3133	0,2623	0,2756
48	0,1071	0,1212	0,5805	0,5805	0,3559	0,3426	0,1271	0,1114	0,1212	0,1069	0,1212	0,1069
49	0,1374	0,1689	0,4956	0,4967	0,2003	0,1640	0,2141	0,2683	0,4688	0,5361	0,1689	0,1338
50	0,1100	0,1223	0,5668	0,5669	0,3076	0,3125	0,2706	0,2573	0,1223	0,1096	0,1223	0,1100
51	0,1348	0,1770	0,5011	0,5016	0,2140	0,1668	0,1826	0,1772	0,2803	0,2819	0,2768	0,2457
52	0,1310	0,1742	0,5060	0,5060	0,3222	0,4392	0,2922	0,3543	0,3018	0,4232	0,2922	0,3515
53	0,1307	0,1498	0,5201	0,5203	0,1992	0,2819	0,2442	0,2694	0,1498	0,1288	0,1498	0,1307
54	0,1093	0,1357	0,5444	0,5449	0,2710	0,3192	0,1939	0,1943	0,2569	0,1895	0,2569	0,1912
55	0,1363	0,1747	0,4957	0,4961	0,1663	0,1982	0,3787	0,4289	0,4868	0,5778	0,2133	0,2129
56	0,1291	0,1636	0,5153	0,5161	0,1943	0,2721	0,2606	0,3314	0,2465	0,2039	0,4498	0,6174
57	0,1373	0,1766	0,4995	0,5001	0,1704	0,2135	0,5085	0,5690	0,2136	0,1766	0,2259	0,1911
58	0,1388	0,1829	0,4928	0,4930	0,1724	0,1864	0,2083	0,2210	0,3017	0,3487	0,2195	0,1886

59	0,1180	0,1706	0,5347	0,5349	0,3640	0,5761	0,3390	0,4828	0,3390	0,4882	0,3390	0,4814
60	0,1246	0,1596	0,5306	0,5312	0,1996	0,1898	0,2684	0,2507	0,3217	0,3839	0,3218	0,3843
61	0,1269	0,1594	0,5200	0,5208	0,1940	0,1836	0,2399	0,3198	0,1571	0,1314	0,1571	0,1314
62	0,1257	0,1542	0,5045	0,5056	0,1536	0,1599	0,1585	0,1503	0,1646	0,1779	0,1542	0,1247
63	0,1333	0,1718	0,5043	0,5047	0,1700	0,1728	0,3016	0,3816	0,2025	0,1700	0,5471	0,6724
64	0,1339	0,1625	0,5020	0,5025	0,1883	0,1358	0,1662	0,1522	0,2240	0,2008	0,1629	0,1442
65	0,1174	0,1493	0,5196	0,5195	0,1634	0,1773	0,2263	0,1914	0,2066	0,2514	0,2066	0,2202
66	0,1205	0,1512	0,5165	0,5165	0,1694	0,1721	0,2440	0,2411	0,2281	0,2677	0,2281	0,2346
67	0,1276	0,1690	0,5195	0,5201	0,2042	0,2611	0,2018	0,2111	0,3325	0,3716	0,2534	0,2943
68	0,1190	0,1282	0,5377	0,5376	0,3386	0,3973	0,3132	0,3725	0,3132	0,3823	0,1282	0,1171
69	0,1311	0,1650	0,5044	0,5052	0,1912	0,2677	0,1939	0,2720	0,1939	0,2753	0,2382	0,2454
70	0,1159	0,1315	0,5524	0,5526	0,3063	0,4585	0,3063	0,3663	0,1315	0,1151	0,1315	0,1151
71	0,1235	0,1506	0,5114	0,5115	0,1699	0,1857	0,1693	0,1503	0,2262	0,2599	0,1506	0,1211
72	0,1202	0,1349	0,5535	0,5537	0,1717	0,2296	0,2828	0,4304	0,3368	0,4616	0,3368	0,4638
73	0,1293	0,1737	0,5181	0,5191	0,2317	0,3045	0,1860	0,1746	0,3150	0,2840	0,2763	0,2136
74	0,1338	0,1761	0,5028	0,5029	0,1944	0,1887	0,1870	0,2119	0,2734	0,3189	0,2154	0,1780
75	0,1266	0,1645	0,5221	0,5229	0,2317	0,1771	0,2370	0,2432	0,4646	0,5460	0,3130	0,3708
76	0,1316	0,1657	0,5016	0,5022	0,1899	0,2240	0,2178	0,2325	0,1913	0,2112	0,1881	0,2980
77	0,1162	0,1469	0,5405	0,5410	0,2047	0,2450	0,2137	0,2709	0,3981	0,4264	0,1640	0,2334
78	0,1323	0,1596	0,5022	0,5030	0,2844	0,2451	0,1736	0,1884	0,1736	0,1912	0,1596	0,1320
79	0,1222	0,1408	0,5374	0,5377	0,3260	0,3749	0,2672	0,3218	0,2975	0,3578	0,1408	0,1215
80	0,1104	0,1237	0,5685	0,5686	0,2885	0,2928	0,1763	0,1727	0,1237	0,1092	0,1237	0,1092
81	0,1082	0,1229	0,5780	0,5781	0,3062	0,3082	0,2253	0,2253	0,1230	0,1077	0,1230	0,1070
82	0,1165	0,1368	0,5441	0,5442	0,2492	0,2068	0,1386	0,1363	0,1492	0,1597	0,1492	0,1604
83	0,1162	0,1570	0,5369	0,5374	0,2663	0,4068	0,2358	0,2641	0,2788	0,2054	0,2788	0,2065
84	0,1148	0,1376	0,5503	0,5505	0,2163	0,1667	0,1326	0,1290	0,1376	0,1143	0,1376	0,1146
85	0,1301	0,1721	0,5084	0,5085	0,2753	0,4699	0,2692	0,3263	0,2749	0,3492	0,2692	0,3193
86	0,1240	0,1579	0,5155	0,5156	0,1712	0,1647	0,2262	0,2592	0,5129	0,5273	0,1556	0,1264
87	0,1143	0,1289	0,5628	0,5630	0,1646	0,1991	0,2508	0,3492	0,2818	0,3850	0,2818	0,3842
88	0,1100	0,1243	0,5662	0,5665	0,2628	0,2787	0,1445	0,1285	0,1243	0,1102	0,1243	0,1102
89	0,1259	0,1579	0,5118	0,5120	0,2064	0,3842	0,1912	0,2903	0,2223	0,1689	0,1953	0,3105
90	0,1156	0,1506	0,5428	0,5428	0,2906	0,2260	0,1510	0,1858	0,2906	0,2557	0,1519	0,1869
91	0,1178	0,1409	0,5436	0,5438	0,3508	0,4074	0,3281	0,3835	0,1409	0,1176	0,1409	0,1176
92	0,1307	0,1717	0,4964	0,4966	0,2822	0,5425	0,2678	0,3479	0,2682	0,3640	0,1717	0,1301
93	0,1271	0,1548	0,5062	0,5063	0,1733	0,1888	0,1664	0,1530	0,2361	0,2555	0,2151	0,1701
94	0,1201	0,1396	0,5275	0,5277	0,1499	0,1778	0,1808	0,1631	0,1402	0,1206	0,1396	0,1191
95	0,1229	0,1561	0,5163	0,5162	0,1657	0,1612	0,1995	0,2069	0,4649	0,4214	0,2037	0,1720
96	0,1169	0,1482	0,5388	0,5395	0,2151	0,3004	0,2100	0,2775	0,2100	0,2836	0,1401	0,1976
97	0,1090	0,1319	0,5521	0,5521	0,2922	0,3240	0,5181	0,5294	0,3431	0,4806	0,5362	0,5563
98	0,1200	0,1534	0,5202	0,5202	0,1541	0,1779	0,1882	0,1984	0,4581	0,4161	0,1872	0,1500
99	0,1091	0,1482	0,5577	0,5577	0,2398	0,2040	0,2191	0,1958	0,2546	0,2470	0,2546	0,2506
100	0,1303	0,1682	0,5123	0,5131	0,2631	0,2185	0,1926	0,1904	0,2229	0,2198	0,3474	0,4227
101	0,1270	0,1638	0,5126	0,5127	0,2309	0,4010	0,2274	0,3193	0,2348	0,2999	0,2194	0,1825
102	0,1145	0,1297	0,5600	0,5601	0,2653	0,3908	0,1920	0,2424	0,1792	0,2302	0,1792	0,2302
103	0,1201	0,1414	0,5412	0,5415	0,3610	0,5753	0,3605	0,4720	0,3546	0,4084	0,1414	0,1200
104	0,1344	0,1691	0,5134	0,5140	0,2981	0,3846	0,2429	0,2175	0,3120	0,3533	0,3071	0,2909
105	0,1387	0,1840	0,4988	0,4990	0,1806	0,1991	0,2024	0,1746	0,2024	0,1735	0,1840	0,1357

106	0,1330	0,1681	0,4972	0,4977	0,2111	0,3569	0,2113	0,3343	0,1923	0,1617	0,1975	0,1920
107	0,1291	0,1625	0,5210	0,5217	0,1977	0,2172	0,1615	0,2205	0,1633	0,2366	0,1751	0,2006
108	0,1043	0,1073	0,5838	0,5839	0,1102	0,1053	0,1092	0,1047	0,1073	0,1043	0,1073	0,1043
109	0,1168	0,1363	0,5473	0,5476	0,3576	0,4935	0,3542	0,4205	0,3529	0,3879	0,1363	0,1168
110	0,1405	0,1743	0,5077	0,5082	0,3706	0,4421	0,2551	0,2161	0,2933	0,3647	0,2899	0,2670
111	0,1154	0,1403	0,5519	0,5517	0,3423	0,4421	0,3316	0,4595	0,2834	0,2986	0,6292	0,7989
112	0,1419	0,1717	0,4887	0,4898	0,1651	0,2034	0,1977	0,1824	0,2607	0,2239	0,1717	0,1412
113	0,1172	0,1638	0,5346	0,5349	0,3221	0,5159	0,2921	0,3409	0,3013	0,4334	0,3013	0,4275
114	0,1320	0,1647	0,5104	0,5112	0,2084	0,2655	0,1834	0,1475	0,4664	0,6502	0,1832	0,1695
115	0,1287	0,1606	0,5129	0,5135	0,2351	0,3918	0,2264	0,3032	0,2768	0,3709	0,2461	0,2046
116	0,1187	0,1458	0,5178	0,5178	0,1906	0,2175	0,1634	0,1489	0,2189	0,2595	0,2189	0,2258
117	0,1295	0,1478	0,5150	0,5152	0,2892	0,4050	0,2892	0,3961	0,1478	0,1321	0,1478	0,1307
118	0,1386	0,1807	0,4912	0,4918	0,2289	0,1798	0,1712	0,1803	0,2012	0,2019	0,1807	0,1358
119	0,1125	0,1435	0,5489	0,5492	0,2417	0,1845	0,1850	0,1614	0,2471	0,1948	0,2517	0,2331
120	0,1207	0,1547	0,5260	0,5262	0,2187	0,3856	0,2060	0,2858	0,2547	0,3299	0,2535	0,2705
121	0,1247	0,1575	0,5166	0,5166	0,2329	0,1673	0,1690	0,1870	0,3155	0,3840	0,2764	0,2947
122	0,1216	0,1495	0,5175	0,5176	0,1802	0,1967	0,1820	0,1634	0,2523	0,3064	0,2458	0,2239
123	0,1176	0,1484	0,5323	0,5328	0,1754	0,1787	0,2444	0,2739	0,1461	0,1226	0,1461	0,1229
124	0,1204	0,1577	0,5233	0,5231	0,3022	0,2507	0,1836	0,1880	0,2065	0,2174	0,2034	0,1981
125	0,1076	0,1095	0,5704	0,5703	0,1886	0,2118	0,1517	0,1523	0,1096	0,1075	0,1095	0,1076
126	0,1168	0,1420	0,5407	0,5405	0,2098	0,2000	0,2651	0,3482	0,3388	0,4342	0,5292	0,6651
127	0,1080	0,1367	0,5649	0,5648	0,2091	0,2320	0,1969	0,1756	0,2434	0,2147	0,1354	0,1070
128	0,1296	0,1691	0,5095	0,5097	0,1768	0,1868	0,2452	0,3229	0,4072	0,5230	0,3731	0,4802
129	0,1179	0,1464	0,5205	0,5206	0,2211	0,2439	0,1846	0,1818	0,2214	0,2798	0,2199	0,1955
130	0,1137	0,1520	0,5480	0,5482	0,2885	0,4097	0,2844	0,3421	0,2448	0,3264	0,2448	0,3060
131	0,1347	0,1804	0,4996	0,4997	0,2121	0,1960	0,1806	0,1888	0,2419	0,2821	0,2148	0,2348
132	0,1198	0,1544	0,5325	0,5332	0,2628	0,4342	0,2072	0,3324	0,2628	0,3988	0,2538	0,2646
133	0,1242	0,1635	0,5211	0,5212	0,2387	0,4122	0,2319	0,3496	0,2319	0,3629	0,2313	0,3359
134	0,1365	0,1787	0,4941	0,4943	0,2228	0,1668	0,1751	0,2247	0,3268	0,4416	0,3268	0,4313
135	0,1326	0,1609	0,4991	0,5001	0,2611	0,4404	0,2611	0,3824	0,1644	0,1469	0,1644	0,1587
136	0,1257	0,1650	0,5155	0,5155	0,2734	0,4446	0,2663	0,3248	0,2728	0,3933	0,2438	0,2544
137	0,1245	0,1646	0,5251	0,5258	0,1984	0,2656	0,1924	0,2023	0,2702	0,2307	0,2590	0,2244
138	0,1242	0,1428	0,5323	0,5325	0,2488	0,3376	0,1555	0,1327	0,1428	0,1240	0,1428	0,1240
139	0,1239	0,1415	0,5337	0,5339	0,1543	0,1725	0,2179	0,3010	0,2648	0,3633	0,2648	0,3630
140	0,1280	0,1458	0,5156	0,5159	0,2759	0,5289	0,2759	0,4455	0,2350	0,2924	0,2357	0,3144
141	0,1334	0,1595	0,5027	0,5032	0,1559	0,1816	0,1718	0,1571	0,1605	0,1491	0,1605	0,1440
142	0,1347	0,1785	0,5009	0,5009	0,1881	0,2002	0,1717	0,1770	0,2630	0,3149	0,2266	0,2553
143	0,1210	0,1349	0,5533	0,5535	0,1923	0,2744	0,4561	0,6022	0,1349	0,1176	0,1349	0,1167
144	0,1270	0,1691	0,5157	0,5159	0,1755	0,2143	0,1681	0,1589	0,2607	0,2156	0,2386	0,2107
145	0,1223	0,1399	0,5413	0,5415	0,2795	0,3621	0,2157	0,1980	0,1398	0,1201	0,1399	0,1206
146	0,1181	0,1613	0,5349	0,5352	0,3023	0,5113	0,2785	0,3173	0,2829	0,4261	0,2916	0,3567
147	0,1226	0,1581	0,5101	0,5105	0,2202	0,2469	0,2170	0,2261	0,2021	0,2366	0,2021	0,2076
148	0,1306	0,1680	0,5083	0,5086	0,3184	0,3487	0,2018	0,2144	0,2434	0,2644	0,2336	0,1977
149	0,1426	0,1916	0,4855	0,4857	0,1886	0,1660	0,1703	0,1534	0,2161	0,1807	0,2161	0,1824
150	0,1420	0,1810	0,4873	0,4884	0,1613	0,1602	0,1719	0,1534	0,1854	0,1730	0,1810	0,1423
151	0,1226	0,1507	0,5120	0,5120	0,1728	0,1874	0,1741	0,1531	0,2290	0,2668	0,2283	0,2197
152	0,1204	0,1369	0,5435	0,5437	0,3187	0,3997	0,3338	0,3631	0,3533	0,3857	0,1369	0,1204

153	0,1271	0,1664	0,5094	0,5094	0,2345	0,4182	0,2332	0,3542	0,1996	0,1632	0,2071	0,2196
154	0,1189	0,1369	0,5332	0,5335	0,1533	0,1609	0,1732	0,1658	0,1369	0,1212	0,1369	0,1204
155	0,1131	0,1601	0,5478	0,5480	0,2890	0,2282	0,2171	0,2072	0,2171	0,2083	0,2015	0,1717
156	0,1115	0,1273	0,5627	0,5630	0,2047	0,2433	0,2025	0,1897	0,1273	0,1112	0,1273	0,1115
157	0,1200	0,1471	0,5213	0,5216	0,1855	0,2190	0,2810	0,2637	0,4107	0,3718	0,2461	0,2800
158	0,1355	0,1724	0,4961	0,4967	0,1680	0,1534	0,2671	0,3204	0,3937	0,4929	0,1900	0,1632
159	0,1193	0,1348	0,5453	0,5456	0,3087	0,4463	0,3067	0,3970	0,1348	0,1195	0,1348	0,1194
160	0,1123	0,1288	0,5579	0,5582	0,1634	0,2049	0,1858	0,1652	0,1288	0,1118	0,1288	0,1122
161	0,1214	0,1565	0,5314	0,5323	0,2123	0,3543	0,1786	0,2478	0,1829	0,2992	0,1635	0,2406
162	0,1152	0,1414	0,5295	0,5302	0,2691	0,3032	0,1753	0,1722	0,2614	0,2323	0,2181	0,1743
163	0,1217	0,1551	0,5216	0,5216	0,1631	0,1694	0,1856	0,1896	0,1541	0,1205	0,2029	0,1616
164	0,1262	0,1637	0,5233	0,5238	0,1924	0,2743	0,3123	0,3023	0,2340	0,1849	0,2340	0,1914
165	0,1203	0,1357	0,5396	0,5398	0,3032	0,4367	0,3032	0,3875	0,1357	0,1196	0,1357	0,1191
166	0,1121	0,1254	0,5640	0,5642	0,1817	0,2291	0,2789	0,3371	0,3307	0,3703	0,1254	0,1120
167	0,1112	0,1460	0,5475	0,5477	0,2650	0,2771	0,2045	0,1761	0,2160	0,1748	0,1460	0,1102
168	0,1389	0,1843	0,4971	0,4974	0,1943	0,2256	0,1934	0,1819	0,2617	0,2718	0,1858	0,1410
169	0,1077	0,1219	0,5762	0,5763	0,3028	0,3062	0,1319	0,1178	0,1219	0,1076	0,1219	0,1076
170	0,1309	0,1674	0,5039	0,5043	0,1858	0,2317	0,1762	0,2236	0,1748	0,2463	0,1731	0,1875
171	0,1053	0,1084	0,5829	0,5829	0,1981	0,2206	0,1571	0,1471	0,1181	0,1205	0,1084	0,1051
172	0,1180	0,1536	0,5374	0,5382	0,2293	0,3733	0,2165	0,2949	0,4051	0,3371	0,2165	0,3025
173	0,1230	0,1450	0,5328	0,5330	0,1440	0,1476	0,1418	0,1263	0,1368	0,1231	0,1450	0,1225
174	0,1400	0,1853	0,4911	0,4911	0,1888	0,1757	0,1707	0,1570	0,2210	0,1778	0,2163	0,1990
175	0,1380	0,1772	0,5023	0,5029	0,2017	0,1687	0,2019	0,2038	0,2821	0,2911	0,1772	0,1333
176	0,1290	0,1655	0,5092	0,5094	0,2239	0,3451	0,2206	0,2718	0,2412	0,3018	0,2412	0,3023
177	0,1198	0,1506	0,5271	0,5274	0,1753	0,1652	0,1718	0,1642	0,2488	0,2220	0,1506	0,1181
178	0,1392	0,1853	0,4918	0,4919	0,1796	0,1659	0,1729	0,1548	0,2133	0,2060	0,2108	0,1764
179	0,1203	0,1476	0,5148	0,5149	0,1635	0,1861	0,3805	0,3706	0,5633	0,5837	0,1971	0,1635
180	0,1195	0,1472	0,5411	0,5409	0,2249	0,3365	0,2233	0,2261	0,2249	0,2992	0,2095	0,1996
181	0,1114	0,1256	0,5636	0,5638	0,1659	0,2113	0,2445	0,3024	0,2927	0,3497	0,2927	0,3500
182	0,1175	0,1325	0,5513	0,5515	0,2895	0,3706	0,2889	0,3665	0,1326	0,1170	0,1325	0,1174
183	0,1353	0,1775	0,4951	0,4953	0,2205	0,1655	0,1756	0,1853	0,3430	0,3807	0,1775	0,1348
184	0,1282	0,1630	0,5181	0,5190	0,2019	0,2911	0,2173	0,2677	0,4692	0,6032	0,3729	0,4939
185	0,1096	0,1244	0,5751	0,5752	0,3844	0,3688	0,1508	0,1373	0,1244	0,1096	0,1244	0,1096
186	0,1364	0,1792	0,4934	0,4933	0,2893	0,4291	0,2885	0,3749	0,2893	0,4225	0,2893	0,3854
187	0,1168	0,1476	0,5430	0,5429	0,2328	0,3675	0,2318	0,2690	0,3413	0,4535	0,2257	0,2520
188	0,1291	0,1535	0,5091	0,5094	0,3189	0,4958	0,3179	0,4407	0,3179	0,3864	0,1537	0,1360
189	0,1118	0,1479	0,5426	0,5425	0,2126	0,2400	0,2392	0,2013	0,2031	0,1636	0,2031	0,1670
190	0,1176	0,1523	0,5332	0,5334	0,2574	0,1854	0,1772	0,1820	0,2659	0,3094	0,2099	0,2221
191	0,1338	0,1602	0,5027	0,5035	0,2202	0,3372	0,2005	0,2717	0,2025	0,3188	0,1989	0,2729
192	0,1192	0,1543	0,5316	0,5323	0,2761	0,4122	0,2554	0,2986	0,2554	0,3188	0,3334	0,3328
193	0,1261	0,1428	0,5318	0,5320	0,2459	0,3402	0,2768	0,2747	0,3964	0,3994	0,1428	0,1262
194	0,1179	0,1359	0,5498	0,5499	0,1984	0,2826	0,2848	0,3171	0,3104	0,3426	0,1359	0,1176
195	0,1261	0,1773	0,5200	0,5206	0,2400	0,2962	0,2075	0,2520	0,2075	0,2572	0,2648	0,2331
196	0,1041	0,1070	0,5860	0,5860	0,1069	0,1037	0,1067	0,1034	0,1070	0,1039	0,1070	0,1039
197	0,1236	0,1542	0,5296	0,5299	0,2166	0,2859	0,2025	0,1916	0,2166	0,2687	0,1999	0,1900
198	0,1246	0,1550	0,5275	0,5281	0,2046	0,2878	0,1744	0,1850	0,2046	0,2326	0,2042	0,2259
199	0,1234	0,1530	0,5117	0,5118	0,1759	0,1835	0,1636	0,1605	0,2352	0,2383	0,2178	0,1773

200	0,1215	0,1544	0,5246	0,5246	0,2179	0,1596	0,1708	0,1801	0,2378	0,2398	0,1537	0,1202
201	0,1132	0,1172	0,5499	0,5498	0,1222	0,1158	0,1177	0,1130	0,1172	0,1143	0,1172	0,1142
202	0,1235	0,1419	0,5260	0,5262	0,1618	0,1475	0,1820	0,1972	0,1419	0,1229	0,2141	0,2356
203	0,1322	0,1650	0,5044	0,5048	0,2035	0,3592	0,1950	0,3301	0,1662	0,1304	0,1650	0,1303
204	0,1309	0,1663	0,5121	0,5126	0,1864	0,2418	0,2509	0,2805	0,4140	0,4931	0,3345	0,3925
205	0,1249	0,1666	0,5309	0,5306	0,3019	0,4424	0,2466	0,2781	0,2466	0,2832	0,2466	0,2797
206	0,1160	0,1314	0,5510	0,5513	0,3110	0,4197	0,3110	0,3800	0,1314	0,1161	0,1314	0,1161
207	0,1092	0,1247	0,5702	0,5705	0,1838	0,2230	0,1384	0,1177	0,1247	0,1089	0,1247	0,1089
208	0,1390	0,1721	0,4987	0,4995	0,2075	0,1679	0,2032	0,2688	0,4399	0,6093	0,3514	0,4645
209	0,1160	0,1277	0,5543	0,5545	0,1898	0,2367	0,3266	0,3672	0,4443	0,4872	0,1277	0,1161
210	0,1313	0,1655	0,5006	0,5009	0,2142	0,2819	0,2136	0,2616	0,2256	0,2901	0,2256	0,2686
211	0,1045	0,1072	0,5845	0,5845	0,1072	0,1041	0,1072	0,1045	0,1072	0,1041	0,1072	0,1042
212	0,1263	0,1489	0,5087	0,5091	0,1528	0,1641	0,1552	0,1443	0,1511	0,1490	0,1511	0,1396
213	0,1170	0,1478	0,5311	0,5312	0,1681	0,2017	0,1636	0,1638	0,2306	0,1840	0,1478	0,1137
214	0,1318	0,1716	0,5136	0,5142	0,2045	0,2456	0,2113	0,1857	0,2937	0,2767	0,2614	0,2019
215	0,1342	0,1563	0,5061	0,5064	0,1663	0,1386	0,1981	0,2848	0,2740	0,3634	0,2759	0,3803
216	0,1372	0,1587	0,5002	0,5005	0,1814	0,2493	0,1978	0,1820	0,1591	0,1471	0,1591	0,1421
217	0,1354	0,1723	0,5009	0,5011	0,2541	0,5055	0,2388	0,3138	0,2418	0,2874	0,2367	0,3511
218	0,1422	0,1757	0,4860	0,4871	0,2095	0,1564	0,1499	0,1404	0,1790	0,1641	0,1790	0,1700
219	0,1270	0,1626	0,5170	0,5174	0,2090	0,3563	0,2058	0,3143	0,2090	0,3209	0,1912	0,1698
220	0,1122	0,1297	0,5584	0,5587	0,1454	0,1564	0,1626	0,1745	0,1297	0,1117	0,2200	0,2448
221	0,1161	0,1286	0,5527	0,5529	0,1725	0,2180	0,2516	0,3244	0,3325	0,4186	0,3325	0,4192
222	0,1174	0,1493	0,5284	0,5287	0,2187	0,2475	0,2188	0,2288	0,2600	0,2755	0,2600	0,2793
223	0,1252	0,1475	0,5185	0,5187	0,1543	0,1747	0,1969	0,1786	0,1476	0,1331	0,1476	0,1307
224	0,1226	0,1557	0,5170	0,5170	0,1602	0,1649	0,1616	0,1618	0,2074	0,1927	0,1557	0,1214
225	0,1231	0,1408	0,5370	0,5372	0,1711	0,2398	0,2903	0,2691	0,1408	0,1207	0,1408	0,1217
226	0,1076	0,1220	0,5759	0,5761	0,2837	0,2929	0,1316	0,1170	0,1220	0,1076	0,1220	0,1075
227	0,1162	0,1523	0,5418	0,5425	0,2324	0,3831	0,2162	0,2757	0,2206	0,3249	0,2206	0,3097
228	0,1186	0,1465	0,5246	0,5248	0,2122	0,2376	0,1897	0,1806	0,2575	0,3323	0,1465	0,1175
229	0,1281	0,1656	0,5084	0,5086	0,2064	0,3602	0,2056	0,3329	0,2056	0,3417	0,1649	0,2421
230	0,1277	0,1588	0,5157	0,5165	0,2290	0,3574	0,2183	0,2730	0,2999	0,4049	0,2986	0,3133
231	0,1307	0,1722	0,5068	0,5068	0,2980	0,4581	0,2913	0,3529	0,2971	0,4464	0,2913	0,3539
232	0,1247	0,1731	0,5225	0,5230	0,2786	0,2640	0,1852	0,1752	0,2696	0,2473	0,3264	0,3747
233	0,1191	0,1590	0,5334	0,5341	0,2785	0,4562	0,2569	0,3283	0,2569	0,3316	0,2569	0,3264
234	0,1342	0,1662	0,5035	0,5045	0,1965	0,2549	0,2009	0,2677	0,1813	0,1642	0,1813	0,1526
235	0,1182	0,1492	0,5480	0,5478	0,2238	0,1953	0,2637	0,2780	0,3664	0,5060	0,2738	0,3183
236	0,1275	0,1643	0,5176	0,5180	0,1750	0,2294	0,2845	0,3250	0,5649	0,6504	0,2261	0,1861
237	0,1134	0,1241	0,5517	0,5516	0,3211	0,3701	0,2665	0,3055	0,1240	0,1170	0,1241	0,1134
238	0,1229	0,1559	0,5279	0,5285	0,2494	0,3661	0,1985	0,1978	0,5121	0,6347	0,5121	0,6344
239	0,1238	0,1595	0,5176	0,5176	0,2295	0,3918	0,2241	0,2865	0,2371	0,2892	0,2371	0,2507
240	0,1287	0,1666	0,5046	0,5050	0,2179	0,4191	0,2175	0,3647	0,1893	0,1598	0,2001	0,2252
241	0,1159	0,1283	0,5599	0,5601	0,1830	0,2306	0,3595	0,4063	0,4089	0,3963	0,1283	0,1140
242	0,1213	0,1546	0,5305	0,5313	0,2039	0,3061	0,2021	0,1857	0,2032	0,2432	0,2032	0,2432
243	0,1207	0,1567	0,5271	0,5275	0,1907	0,2524	0,1949	0,2717	0,1949	0,2751	0,1643	0,1371
244	0,1291	0,1600	0,4994	0,4996	0,1699	0,1491	0,2136	0,2143	0,1977	0,1970	0,1909	0,1597
245	0,1105	0,1227	0,5704	0,5705	0,2518	0,2840	0,3894	0,4390	0,4488	0,5064	0,1227	0,1105
246	0,1252	0,1469	0,5142	0,5145	0,1534	0,1700	0,1538	0,1406	0,1472	0,1359	0,1469	0,1248

247	0,1108	0,1374	0,5393	0,5398	0,2452	0,2720	0,3806	0,3562	0,5413	0,5140	0,3903	0,3168
248	0,1259	0,1649	0,5169	0,5174	0,2883	0,4028	0,2571	0,2775	0,2761	0,3632	0,2767	0,3074
249	0,1050	0,1079	0,5810	0,5810	0,1080	0,1046	0,1078	0,1045	0,1079	0,1048	0,1079	0,1048
250	0,1282	0,1568	0,5158	0,5166	0,2134	0,3141	0,1856	0,2667	0,2882	0,2858	0,2922	0,3302
Minimum	0,1032	0,1060	0,4855	0,4857	0,1058	0,1027	0,1058	0,1026	0,1060	0,1029	0,1060	0,1029
Maximum	0,1426	0,1916	0,5903	0,5903	0,3844	0,5761	0,5181	0,6022	0,5669	0,6683	0,6292	0,7989
Durchschnitt	**0,1233**	**0,1520**	**0,5271**	**0,5274**	**0,2224**	**0,2752**	**0,2211**	**0,2508**	**0,2415**	**0,2666**	**0,2084**	**0,2167**

Anhang A3 - Das Inverse Pendel mit dem Least Weight Algorithmus

Zum Beispiel 6.21 des Inversen Pendels (S. 121) werden nachstehend Simulationsergebnisse wiedergegeben, bei denen auch der Least Weight Algorithmus (LWA, Satz 6.38 auf S. 131) berücksichtigt ist. Vgl. a6.42 (S. 134).

Durchlauf	A	B	C	D	E	F	G	H	I	J	K	L
1	0,0969	0,1162	0,7709	1,0103	0,4232	0,5131	0,4124	0,4138	0,2738	0,2522	0,1290	0,1119
2	0,0885	0,1323	0,4107	0,3310	0,3176	0,2839	0,2325	0,1662	0,5536	0,4876	0,1365	0,0919
3	0,1685	0,1868	0,2267	0,2477	0,2590	0,2699	0,1868	0,1685	0,1448	0,1256	0,1789	0,1603
4	0,3018	0,3114	0,3298	0,3309	0,3135	0,3098	0,3114	0,3020	0,1619	0,1528	0,3623	0,3519
5	0,0994	0,1248	0,3489	0,2659	0,1603	0,1505	0,1852	0,1864	0,3309	0,2797	0,1420	0,1116
6	0,1177	0,1411	0,5691	0,6741	0,4822	0,4947	0,1429	0,1247	0,1834	0,1489	0,1201	0,0934
7	0,1359	0,1948	0,2767	0,2864	0,2326	0,2307	0,2450	0,1994	0,3194	0,2589	0,1940	0,1388
8	0,0866	0,1078	0,3019	0,5457	0,3019	0,4590	0,3019	0,4577	0,1308	0,1018	0,1130	0,0892
9	0,1154	0,1330	0,1580	0,1725	0,1727	0,1762	0,1330	0,1153	0,1919	0,1731	0,1397	0,1219
10	0,0896	0,1261	0,3242	0,6297	0,3242	0,5556	0,1261	0,0887	0,2132	0,1729	0,1299	0,0934
11	0,1389	0,1458	0,1573	0,1448	0,1534	0,1421	0,1458	0,1390	0,6953	0,6913	0,2338	0,2277
12	0,1155	0,1335	0,4312	0,7012	0,4309	0,6367	0,1544	0,1316	0,9613	0,9194	0,1936	0,1787
13	0,2578	0,2796	0,4070	0,5290	0,3950	0,4152	0,2961	0,2652	0,9240	0,9097	0,3137	0,2816
14	0,0781	0,1040	0,3268	0,4030	0,2698	0,2662	0,1174	0,1434	0,6386	0,6102	0,1194	0,0886
15	0,1041	0,1267	0,2852	0,3150	0,2930	0,3662	0,2826	0,3406	0,2750	0,2491	0,1273	0,1048
16	0,1261	0,1683	0,3686	0,3594	0,2393	0,2372	0,2297	0,2236	0,1729	0,1235	0,1841	0,1404
17	0,0951	0,1266	0,4256	0,4576	0,2258	0,2441	0,1266	0,0950	0,8945	0,8405	0,1293	0,0940
18	0,1404	0,1416	0,2168	0,2010	0,1718	0,1635	0,1416	0,1405	0,5387	0,5385	0,1474	0,1457
19	0,0973	0,1102	0,5064	0,5260	0,2581	0,2800	0,2619	0,2875	0,7722	0,7417	0,2560	0,2365
20	0,0993	0,1155	0,4469	0,4422	0,3657	0,3650	0,1155	0,1054	0,1325	0,1064	0,1279	0,1110
21	0,1007	0,1408	0,6261	0,9159	0,4603	0,5280	0,2036	0,1556	0,2033	0,1459	0,1498	0,1051
22	0,0982	0,1349	0,4823	0,9475	0,4823	0,8874	0,1349	0,0984	2,1165	2,0478	0,2901	0,2386
23	0,1337	0,1447	0,2675	0,3629	0,2675	0,3595	0,2329	0,3041	1,8697	1,8419	0,1720	0,1554
24	0,1245	0,1345	0,1894	0,1901	0,1693	0,1478	0,1345	0,1245	0,5229	0,5100	0,1639	0,1545
25	0,0832	0,0887	0,1048	0,0947	0,0987	0,0899	0,0887	0,0832	0,3009	0,2943	0,1069	0,1011
26	0,1152	0,1399	0,3180	0,3547	0,2010	0,1794	0,1583	0,1574	0,3166	0,2809	0,1992	0,1769
27	0,3840	0,3972	0,7069	0,8679	0,5559	0,6697	0,5332	0,6018	0,1271	0,1139	0,2466	0,2339
28	0,0879	0,1207	0,5473	0,7251	0,2831	0,3749	0,3000	0,3624	1,6126	1,5817	0,1760	0,1400
29	0,1478	0,1745	0,5084	0,3427	0,2492	0,2374	0,2544	0,2656	0,1850	0,1369	0,1474	0,1243
30	0,2537	0,2496	0,5865	0,6794	0,4870	0,5194	0,2496	0,2552	0,4308	0,4420	0,4411	0,4512
31	0,0774	0,1013	0,4648	0,5120	0,2600	0,2663	0,1478	0,1699	2,2301	2,2538	0,3434	0,3294
32	0,0976	0,1306	0,2738	0,4207	0,2722	0,3356	0,2496	0,1907	0,8265	0,7912	0,1476	0,1118
33	0,0757	0,1038	0,5727	0,6357	0,3105	0,3844	0,1038	0,0756	2,1226	2,1086	0,2533	0,2214
34	0,1025	0,1344	0,3398	0,6393	0,3365	0,5865	0,1344	0,1026	1,5448	1,4939	0,2077	0,1732
35	0,1169	0,1219	0,1660	0,1658	0,1877	0,2098	0,1219	0,1171	0,0980	0,0883	0,1154	0,1097
36	0,1015	0,1374	0,1659	0,1360	0,1526	0,1277	0,1332	0,0977	0,9668	0,9399	0,1770	0,1406
37	0,1181	0,1464	0,2467	0,3120	0,2209	0,2350	0,2009	0,2212	0,5247	0,5018	0,2484	0,2221
38	0,4787	0,5036	0,5833	0,6474	0,5833	0,6136	0,5036	0,4788	0,3227	0,2960	0,2002	0,1742
39	0,1060	0,1360	0,3220	0,2963	0,1735	0,1674	0,1951	0,1754	0,3845	0,3414	0,1491	0,1170
40	0,0807	0,1209	0,4746	0,5205	0,4481	0,4673	0,4249	0,4406	0,2295	0,1965	0,1281	0,0907
41	0,0925	0,1140	0,3456	0,3773	0,2154	0,2102	0,1692	0,1395	0,2131	0,1686	0,1508	0,1295
42	0,1251	0,1461	0,4425	0,4681	0,2791	0,2586	0,1628	0,1918	0,7918	0,7736	0,2090	0,1883
43	0,1051	0,1080	0,3359	0,3225	0,2073	0,2399	0,1083	0,1330	1,7975	1,7855	0,1097	0,1018
44	0,3057	0,3484	0,8025	0,9709	0,7151	0,7588	0,4602	0,4219	0,9079	0,8488	0,5779	0,5319
45	0,0891	0,1205	0,2493	0,1761	0,1769	0,1599	0,1205	0,0891	0,5858	0,5410	0,1296	0,0951
46	0,4133	0,4341	0,7966	0,7683	0,5147	0,5377	0,4964	0,5240	2,7875	2,7852	0,7202	0,7026
47	0,1642	0,1745	0,4011	0,4397	0,3575	0,3946	0,2284	0,2463	2,1580	2,1654	0,4150	0,4085
48	0,0999	0,1094	0,1246	0,1537	0,1198	0,0963	0,1094	0,1005	0,2407	0,2238	0,1129	0,1031
49	0,3632	0,3771	0,6533	0,9011	0,6532	0,7830	0,3771	0,3632	0,1591	0,1537	0,3148	0,3016
50	0,1085	0,1347	0,5509	0,7769	0,4567	0,5099	0,3087	0,4053	0,4543	0,4109	0,1334	0,1055
MinMSE	0,0757	0,0887	0,1048	0,0947	0,0987	0,0899	0,0887	0,0756	0,0980	0,0883	0,1069	0,0886
AvgMSE	**0,1461**	**0,1691**	**0,3988**	**0,4739**	**0,3137**	**0,3579**	**0,2239**	**0,2237**	**0,7108**	**0,6829**	**0,2083**	**0,1842**
MaxMSE	0,4787	0,5036	0,8025	1,0103	0,7151	0,8874	0,5332	0,6018	2,7875	2,7852	0,7202	0,7026

In den Spalten sind aufgetragen:
A: das auf das Inverse-Pendel-Problem trainierte 2-24-1-Netz mit der sigmoiden Ausgabefunktion im Hidden Layer;

B: die gleiche Netztopologie wie in A, aber mit der Rampenfunktion als Ausgabe;
C: das 2-2-1-Netz gemäß Satz 6.20 mit der Rampenfunktion;
D: das gleiche Netz wie in C zum Vergleich mit der Sigmoiden;
E und F: das 2-8-1-Netz gemäß der Gruppenclusterungsmethode - wiederum mit Rampenfunktion bzw. der Sigmoiden;
G: 2-8-1-Netz aus dem dynamischen Intervallclusterungsverfahren mit der Rampenfunktion;
H: das entsprechende Netz wie in G mit der Sigmoiden;
I: das aus dem Least Weight Algorithmus resultierende 2-8-1-Netz (mit der Rampenfunktion);
J: das LWA-2-8-1-Netz mit der Sigmoiden zum Vergleich;
K: das LWA-2-16-1-Netz mit der Rampenfunktion;
L: dito mit der Sigmoiden.

Schließlich sind unter “MinMSE”, “AvgMSE” und “MaxMSE” die minimalen, mittleren und maximalen MSE-Werte für jeden Netztyp im Rahmen dieser Datenreihe aufgeführt.

Anhang A4 - Das Inverse Pendel mit dem Verfahren der schrittweisen Verbesserung

Begleitend zu Beispiel 7.20 (S. 150) werden in der nachstehenden Tabelle Ergebnisse von 150 Simulationen aufgelistet, bei denen - ausgehend von einem trainierten 2-24-1-Netz zum Beispiel des Inversen Pendels - verschiedene Verfahren der vorliegenden Arbeit gegenübergestellt werden, speziell das Verfahren der schrittweisen Verbesserung gemäß Satz 7.23.
Die Betitelung der einzelnen Spalten übernimmt dabei von Anhang A3 die Abkürzungen für die dort bereits aufgeführten Approximations- und Reduktionsverfahren. Aus Gründen der Übersichtlichkeit werden jedoch bewusst nicht sämtliche Verfahren aus Anhang A3 hier erneut berücksichtigt.

Durchlauf	A	B	C	G	H	J	L	M	N	O	P
1	0,1102	0,1414	0,6373	0,1414	0,1103	0,4821	0,1370	0,3436	0,2230	0,0678	0,1102
2	0,1287	0,1293	0,1542	0,1293	0,1287	0,1317	0,1339	0,3235	0,2300	0,0764	0,1253
3	0,2003	0,2169	0,3137	0,2169	0,2003	0,4645	0,2508	0,4287	0,3459	0,1670	0,1878
4	0,1263	0,1546	0,3435	0,2755	0,3042	1,4309	0,1173	0,3723	0,2487	0,0899	0,1237
5	0,1675	0,1973	0,7764	0,1973	0,1691	0,2154	0,0941	0,4283	0,2778	0,1363	0,1621
6	0,1374	0,1792	0,8187	0,1792	0,1378	0,7249	0,1252	0,3793	0,2473	0,1003	0,1441
7	0,1075	0,1357	0,2768	0,2749	0,3536	0,1074	0,0913	0,3541	0,2340	0,0811	0,0981
8	0,1030	0,1318	0,4197	0,2572	0,2928	0,2975	0,1463	0,3528	0,2171	0,0646	0,0972
9	0,1748	0,1990	0,2426	0,2468	0,2347	0,1207	0,1781	0,4157	0,2947	0,1337	0,1716
10	0,1751	0,1846	0,3739	0,1849	0,1849	0,4100	0,1978	0,4304	0,3265	0,1576	0,1820
11	0,0947	0,1275	0,2877	0,2792	0,2318	0,2341	0,0846	0,3774	0,2329	0,0800	0,0993
12	0,1209	0,1282	0,1603	0,1282	0,1208	0,5732	0,1697	0,3731	0,2564	0,0914	0,1267
13	0,0992	0,1307	0,3016	0,4065	0,5086	0,7733	0,0908	0,3824	0,2282	0,0859	0,1060
14	0,1167	0,1784	0,2529	0,1857	0,2350	0,4667	0,1288	0,3686	0,2431	0,0989	0,1142
15	0,0984	0,1323	0,2649	0,2346	0,1866	0,5658	0,2229	0,3192	0,1949	0,0581	0,0987
16	0,1521	0,1680	0,4672	0,2426	0,2169	0,2913	0,1990	0,3917	0,2745	0,1052	0,1491
17	0,0894	0,1286	0,2929	0,1894	0,1957	0,7829	0,0851	0,3596	0,2175	0,0709	0,0871
18	0,1042	0,1240	0,3045	0,2313	0,2594	1,0018	0,1175	0,3295	0,2330	0,0546	0,0999
19	0,0964	0,1126	0,3166	0,1126	0,0966	1,0408	0,2323	0,3565	0,2119	0,0682	0,0963
20	0,2183	0,2430	0,5173	0,2463	0,2436	1,1105	0,1197	0,4776	0,3294	0,1858	0,2096
21	0,1211	0,1319	0,2875	0,1319	0,1213	0,4308	0,1327	0,3567	0,2112	0,0655	0,1117
22	0,0858	0,1207	0,3615	0,1207	0,0858	0,4097	0,1186	0,3379	0,2155	0,0514	0,0801
23	0,0910	0,0988	0,1335	0,0988	0,0910	0,6092	0,1119	0,3803	0,2156	0,0739	0,0871
24	0,1265	0,1370	0,6569	0,6359	0,7399	0,1909	0,1012	0,3713	0,2751	0,0856	0,1217
25	0,1129	0,1453	0,5251	0,4157	0,3501	0,1036	0,0863	0,3471	0,2231	0,0715	0,1190
26	0,4128	0,4076	0,5730	0,4076	0,4131	0,6026	0,3319	0,6244	0,4847	0,3466	0,3983
27	0,0930	0,1083	0,3743	0,1518	0,1720	0,3096	0,1193	0,3410	0,2215	0,0638	0,1001
28	0,1139	0,1329	0,7367	0,1329	0,1148	3,8348	0,7096	0,3369	0,2127	0,0711	0,1145
29	0,1195	0,1469	0,3167	0,1984	0,2092	0,2804	0,0964	0,3741	0,2326	0,0810	0,1221
30	0,1310	0,1631	0,5273	0,4398	0,6190	0,1236	0,1534	0,3564	0,2168	0,0858	0,1262
31	0,1049	0,1185	0,1889	0,1484	0,1344	0,1420	0,1263	0,3515	0,2224	0,0577	0,0908
32	0,1088	0,1445	0,3945	0,1900	0,1557	0,1477	0,1063	0,3619	0,2286	0,0742	0,1031
33	0,0828	0,1094	0,3641	0,2686	0,1985	0,1123	0,0839	0,3464	0,2279	0,0576	0,0838
34	0,1034	0,1444	0,2234	0,1425	0,1455	0,2334	0,1247	0,3725	0,2268	0,0710	0,1118
35	0,1288	0,1697	0,4596	0,2269	0,2699	0,1983	0,1221	0,3564	0,2287	0,0925	0,1169
36	0,1050	0,1401	0,4172	0,1609	0,1642	0,3153	0,1058	0,3256	0,2178	0,0546	0,0989
37	0,3756	0,4401	0,6869	0,4401	0,3749	2,8367	0,8942	0,6281	0,4597	0,3434	0,3569
38	0,0873	0,1040	0,1833	0,2001	0,1418	0,2409	0,0900	0,3594	0,2236	0,0571	0,0861
39	0,1048	0,1290	0,3888	0,1635	0,1238	0,2729	0,1067	0,3453	0,2057	0,0702	0,1053
40	0,1035	0,1322	0,2985	0,1415	0,1590	0,2151	0,1492	0,3618	0,2293	0,0760	0,1111
41	0,0981	0,1067	0,4769	0,1067	0,0992	0,6243	0,1190	0,3419	0,2091	0,0599	0,0907
42	0,3451	0,3843	0,6052	0,5819	0,6123	0,6739	0,3746	0,6519	0,4582	0,3114	0,3456
43	0,0777	0,1172	0,2488	0,2436	0,1442	0,8177	0,1797	0,3404	0,1900	0,0591	0,0794
44	0,1530	0,1682	0,2470	0,1682	0,1531	0,1198	0,2611	0,3876	0,2907	0,1122	0,1550
45	0,2746	0,3120	0,5336	0,3170	0,3656	0,8011	0,4607	0,5433	0,3780	0,2544	0,2666
46	0,1290	0,1443	0,7408	0,3935	0,5063	0,3089	0,2596	0,3966	0,2624	0,0949	0,1188
47	0,4893	0,5007	0,6779	0,5384	0,5394	0,9616	0,4309	0,8048	0,6249	0,4507	0,4832
48	0,1868	0,2113	0,4566	0,3311	0,3663	0,1225	0,1586	0,4331	0,3408	0,1492	0,1931
49	0,1282	0,1528	0,5087	0,1521	0,1282	1,0225	0,2342	0,3335	0,2267	0,0849	0,1178
50	0,0742	0,0960	0,2777	0,4989	0,6584	1,9232	0,3135	0,3597	0,2144	0,0571	0,0819

Durchlauf	A	B	C	G	H	J	L	M	N	O	P
51	0,0907	0,1151	0,2526	0,2095	0,2672	0,4122	0,0967	0,3535	0,2138	0,0493	0,0863
52	0,0820	0,1279	0,6410	0,1279	0,0818	0,1708	0,0929	0,3362	0,2029	0,0581	0,0845
53	0,0932	0,1067	0,3280	0,1068	0,1026	0,1148	0,1044	0,3658	0,2055	0,0658	0,0817
54	0,1533	0,1707	0,5338	0,3542	0,4256	0,5315	0,0839	0,4232	0,2650	0,1225	0,1522
55	0,0968	0,1362	0,2917	0,1362	0,0966	1,1166	0,1008	0,3631	0,2328	0,0621	0,0905
56	0,0985	0,1183	0,1740	0,3064	0,2627	0,4902	0,1847	0,3628	0,2398	0,0760	0,0994
57	0,1425	0,1604	0,4319	0,1980	0,2023	0,4699	0,2835	0,3945	0,2430	0,1004	0,1386
58	0,1863	0,1968	0,3677	0,1968	0,1887	1,2352	0,1013	0,4600	0,3099	0,1686	0,1854
59	0,1335	0,1681	0,3156	0,1681	0,1335	0,1700	0,1479	0,3983	0,2269	0,0870	0,1181
60	0,1010	0,1131	0,3802	0,2253	0,1507	0,1855	0,1133	0,3295	0,2229	0,0650	0,1009
61	0,0854	0,1032	0,2532	0,2381	0,2840	0,1107	0,0855	0,3321	0,2132	0,0530	0,0772
62	0,1315	0,1507	0,4314	0,1507	0,1318	0,9459	0,2409	0,3956	0,2724	0,1043	0,1334
63	0,1219	0,1415	0,4046	0,2947	0,2914	0,1065	0,0991	0,3657	0,2248	0,0746	0,1144
64	0,1001	0,1140	0,1493	0,2276	0,2559	0,1051	0,0993	0,3391	0,1999	0,0649	0,0960
65	0,0956	0,1143	0,1918	0,1143	0,0961	0,1370	0,1039	0,3514	0,1976	0,0528	0,0880
66	0,1092	0,1850	0,4363	0,3700	0,3743	4,2980	1,4501	0,3369	0,2136	0,0839	0,1048
67	0,0982	0,1359	0,3327	0,1397	0,1002	0,1428	0,0935	0,3539	0,2162	0,0627	0,0889
68	0,1412	0,1492	0,1539	0,1492	0,1408	0,1047	0,1231	0,3870	0,2416	0,1107	0,1333
69	0,0828	0,1163	0,4158	0,2613	0,3950	0,5981	0,0820	0,3537	0,2103	0,0572	0,0788
70	0,1176	0,1366	0,2842	0,2486	0,1950	0,1342	0,1037	0,3858	0,2458	0,0738	0,1115
71	0,0917	0,1105	0,2525	0,1809	0,1194	0,1786	0,0876	0,3529	0,2270	0,0477	0,0959
72	0,0779	0,1008	0,3696	0,2450	0,2263	0,5317	0,0771	0,3395	0,1909	0,0565	0,0738
73	0,0950	0,1387	0,2738	0,3003	0,1997	0,3088	0,1373	0,3585	0,1904	0,0488	0,0960
74	0,1179	0,1604	0,4926	0,1604	0,1171	0,3084	0,2009	0,3454	0,2470	0,0828	0,1178
75	0,0991	0,1424	0,4858	0,1424	0,0994	1,9994	0,3053	0,3756	0,2192	0,0611	0,0971
76	0,1241	0,1560	0,3145	0,2174	0,1652	1,3287	0,2125	0,3701	0,2474	0,0700	0,1143
77	0,2144	0,2407	0,4875	0,3473	0,3011	0,1636	0,1778	0,4778	0,3076	0,1825	0,2114
78	0,0940	0,1210	0,3837	0,3023	0,3036	1,0444	0,1846	0,3840	0,2304	0,0731	0,0888
79	0,1260	0,1497	0,4606	0,2064	0,2072	0,1074	0,1246	0,3794	0,2560	0,0938	0,1239
80	0,0747	0,0890	0,4168	0,0890	0,0748	0,6157	0,1559	0,3485	0,1773	0,0439	0,0740
81	0,1999	0,2351	0,5871	0,2352	0,1994	0,4848	0,2502	0,4517	0,3562	0,1610	0,2055
82	0,0938	0,1232	0,3604	0,1232	0,0938	0,5598	0,1222	0,3427	0,2021	0,0625	0,0739
83	0,1100	0,1305	0,1468	0,1305	0,1102	0,1151	0,1015	0,3346	0,2036	0,0669	0,1055
84	0,1076	0,1224	0,2025	0,1224	0,1078	0,1796	0,1115	0,3208	0,2138	0,0594	0,0934
85	0,1321	0,1556	0,3449	0,1649	0,1765	0,1263	0,1712	0,3773	0,2623	0,0948	0,1280
86	0,0984	0,0983	0,6063	0,0983	0,1001	0,1449	0,1014	0,3550	0,2206	0,0637	0,0993
87	0,0966	0,1064	0,1188	0,1064	0,0967	0,2246	0,0892	0,3499	0,2378	0,0873	0,1001
88	0,2091	0,2831	0,3023	0,3670	0,2897	0,2143	0,2663	0,4699	0,3542	0,2056	0,2064
89	0,0830	0,1071	0,2736	0,1071	0,0829	0,1835	0,0839	0,3583	0,2249	0,0552	0,0818
90	0,0757	0,0987	0,3556	0,0987	0,0769	1,3976	0,0939	0,3342	0,2072	0,0449	0,0736
91	0,2276	0,2389	0,3757	0,2664	0,2572	0,1050	0,1812	0,4646	0,3609	0,1895	0,2327
92	0,0751	0,0956	0,2074	0,1109	0,0928	0,1125	0,0723	0,3672	0,1924	0,0604	0,0790
93	0,0930	0,1126	0,5654	0,2521	0,2577	0,1116	0,1148	0,3308	0,2076	0,0636	0,0997
94	0,1168	0,1116	0,2418	0,2228	0,2327	0,5243	0,1153	0,3677	0,2424	0,0612	0,1065
95	0,1032	0,1122	0,1205	0,1122	0,1032	0,2680	0,1179	0,3411	0,2113	0,0763	0,1086
96	0,0897	0,1378	0,2269	0,1378	0,0893	0,3337	0,1506	0,3379	0,2287	0,0654	0,0870
97	0,1008	0,1246	0,7084	0,1248	0,1007	0,1225	0,0938	0,3592	0,2359	0,0586	0,1021
98	0,1823	0,1893	0,7007	0,1932	0,1862	0,1091	0,1854	0,4385	0,3258	0,1388	0,1880
99	0,0943	0,1345	0,3746	0,1692	0,1283	0,1852	0,1037	0,3690	0,2198	0,0590	0,0949
100	0,1127	0,1426	0,4904	0,1418	0,1252	0,3452	0,2154	0,3551	0,2252	0,0731	0,1089

Durchlauf	A	B	C	G	H	J	L	M	N	O	P
101	0,0890	0,1106	0,6451	0,3197	0,4525	0,2721	0,2321	0,3465	0,1882	0,0619	0,0807
102	0,1063	0,1119	0,1982	0,1119	0,1066	0,3253	0,1302	0,3366	0,2075	0,0606	0,1042
103	0,1274	0,1515	0,2714	0,2349	0,2062	0,1289	0,1232	0,3909	0,2286	0,0876	0,1292
104	0,1666	0,1905	0,5236	0,1905	0,1705	0,1464	0,0970	0,4188	0,2868	0,1312	0,1592
105	0,3547	0,3562	0,4766	0,3562	0,3554	0,1653	0,1674	0,5928	0,5055	0,3186	0,3526
106	0,1095	0,1535	0,4480	0,1560	0,1804	1,2056	0,2003	0,3862	0,2480	0,0892	0,1078
107	0,0947	0,1343	0,5972	0,3694	0,4233	0,8567	0,1064	0,3482	0,2206	0,0586	0,0903
108	0,0988	0,1212	0,4055	0,1212	0,0988	0,6722	0,0859	0,3681	0,2355	0,0704	0,1030
109	0,0910	0,1105	0,1693	0,1766	0,1787	0,1087	0,0914	0,3360	0,2225	0,0569	0,0857
110	0,1060	0,1311	0,2981	0,2354	0,2714	0,5030	0,1923	0,3555	0,2326	0,0879	0,1121
111	0,0913	0,1160	0,3831	0,1445	0,1312	0,7088	0,0974	0,3485	0,2022	0,0500	0,0771
112	0,0832	0,1167	0,3459	0,2489	0,2723	0,1615	0,0869	0,3219	0,1975	0,0553	0,0829
113	0,2403	0,2511	0,2803	0,2511	0,2399	0,1330	0,2016	0,5055	0,3752	0,1995	0,2322
114	0,1324	0,1630	0,4211	0,2673	0,2114	0,1480	0,1048	0,3775	0,2434	0,0811	0,1184
115	0,1078	0,1262	0,2876	0,1262	0,1074	0,4202	0,1320	0,3335	0,2427	0,0849	0,1071
116	0,1312	0,1475	0,6589	0,2032	0,1872	0,3541	0,2508	0,3617	0,2460	0,0820	0,1366
117	0,1227	0,1308	0,3879	0,1308	0,1231	1,1094	0,0869	0,3830	0,2103	0,0935	0,1210
118	0,0917	0,1229	0,5470	0,3059	0,3683	0,1627	0,0907	0,3482	0,2078	0,0579	0,0946
119	0,0933	0,1141	0,3861	0,1141	0,0941	0,1488	0,1040	0,3404	0,2030	0,0653	0,0912
120	0,0897	0,0942	0,1660	0,0942	0,0897	0,3331	0,1037	0,3547	0,2029	0,0658	0,0971
121	0,1136	0,1264	0,2932	0,2468	0,1784	2,3951	0,5366	0,3616	0,2599	0,0894	0,1089
122	0,2888	0,3056	0,3036	0,3056	0,2878	0,1916	0,3625	0,5476	0,3916	0,2597	0,2802
123	0,0834	0,1325	0,6062	0,3277	0,3432	0,4766	0,1337	0,3519	0,2089	0,0687	0,0936
124	0,0822	0,1128	0,1820	0,1128	0,0824	0,2051	0,1342	0,3277	0,2002	0,0513	0,0880
125	0,0790	0,1054	0,4549	0,1608	0,1517	0,3446	0,1039	0,3508	0,2090	0,0541	0,0900
126	0,0972	0,1231	0,6305	0,4651	0,5145	0,1300	0,1120	0,3320	0,1985	0,0609	0,0963
127	0,0994	0,1509	0,2145	0,2714	0,2091	0,3749	0,1047	0,3280	0,2186	0,0643	0,0980
128	0,0989	0,1122	0,3703	0,2921	0,2828	0,1116	0,0874	0,3337	0,2045	0,0743	0,0947
129	0,1007	0,1268	0,6994	0,1268	0,1006	0,4426	0,0987	0,3642	0,2143	0,0658	0,0991
130	0,1487	0,1798	0,4527	0,1798	0,1485	0,2186	0,1991	0,4395	0,2689	0,1097	0,1456
131	0,1253	0,1387	0,1719	0,1388	0,1253	0,1706	0,1093	0,4401	0,2521	0,1086	0,1219
132	0,0962	0,1122	0,3132	0,1776	0,1940	0,2293	0,4657	0,3557	0,2156	0,0479	0,0834
133	0,0934	0,1161	0,4688	0,1150	0,1016	0,1515	0,1058	0,3121	0,2109	0,0504	0,0953
134	0,1127	0,1437	0,5164	0,1437	0,1124	0,1950	0,1177	0,3802	0,2358	0,0914	0,1131
135	0,2939	0,3153	0,3521	0,3620	0,3985	0,3749	0,2948	0,4888	0,3591	0,2510	0,2961
136	0,1146	0,1353	0,3073	0,1544	0,1250	0,5320	0,1455	0,3742	0,2525	0,0773	0,1075
137	0,1035	0,1098	0,1982	0,1098	0,1037	0,1200	0,1088	0,3394	0,2140	0,0831	0,1058
138	0,0958	0,1199	0,2350	0,1335	0,1124	0,2716	0,1008	0,3257	0,2115	0,0675	0,0934
139	0,1040	0,1334	0,3426	0,1537	0,1605	0,9478	0,1171	0,3439	0,2214	0,0649	0,1030
140	0,1242	0,1434	0,5009	0,2230	0,2781	0,4628	0,1223	0,3616	0,2775	0,0775	0,1236
141	0,1062	0,1514	0,2909	0,1908	0,2903	0,5239	0,1467	0,3508	0,2261	0,0876	0,1103
142	0,1618	0,1993	0,4221	0,5427	0,6832	0,3059	0,1773	0,4164	0,2738	0,1223	0,1624
143	0,1145	0,1241	0,7543	0,1241	0,1144	0,1387	0,1035	0,3452	0,2255	0,0833	0,1092
144	0,1214	0,1751	0,4452	0,4076	0,4212	0,3962	0,1174	0,3528	0,2510	0,0892	0,1237
145	0,0862	0,1054	0,2695	0,1054	0,0878	0,1105	0,0921	0,3537	0,2138	0,0633	0,0845
146	0,0932	0,1098	0,4364	0,1117	0,1051	0,2573	0,1064	0,4067	0,2286	0,0854	0,0948
147	0,1073	0,1490	0,3223	0,1490	0,1072	0,3459	0,1405	0,3641	0,2283	0,0794	0,1090
148	0,0978	0,1295	0,3760	0,2222	0,1685	0,7507	0,1048	0,3240	0,2013	0,0673	0,1006
149	0,2658	0,2959	0,6382	0,4609	0,5197	0,2454	0,1793	0,5259	0,3561	0,2363	0,2605
150	0,1534	0,2295	0,6467	0,2348	0,1766	0,6472	0,1127	0,3911	0,2465	0,1176	0,1383
MinMSE	0,0742	0,0890	0,1188	0,0890	0,0748	0,1036	0,0723	0,3121	0,1773	0,0439	0,0736
AvgMSE	**0,1291**	**0,1544**	**0,3902**	**0,2208**	**0,2175**	**0,4957**	**0,1687**	**0,3812**	**0,2483**	**0,0958**	**0,1267**
MaxMSE	0,4893	0,5007	0,8187	0,6359	0,7399	4,2980	1,4501	0,8048	0,6249	0,4507	0,4832

In den Spalten sind aufgetragen:

A: das auf das Inverse-Pendel-Problem trainierte 2-24-1-Netz mit der sigmoiden Ausgabefunktion im Hidden Layer;

B: die gleiche Netztopologie wie in A, aber mit der Rampenfunktion als Ausgabe;

C: das 2-2-1-Netz gemäß Satz 6.20 mit der Rampenfunktion;

G: 2-8-1-Netz aus dem dynamischen Intervallclusterungsverfahren mit der Rampenfunktion;

H: das entsprechende Netz wie in G mit der Sigmoiden;
J: das aus dem Least Weight Algorithmus resultierende 2-8-1-Netz (mit der Sigmoiden);
L: das LWA-2-16-1-Netz mit der Sigmoiden;
M: Regelextraktion nach dem Verfahren der schrittweisen Verbesserung (SV) mit 5 Regeln;
N: SV-Regelextraktion mit 8 Regeln;
O: SV-Regelextraktion mit 16 Regeln;
P: schließlich Anwendung des SV-Verfahrens mit der höheren Anzahl von 128 Regeln.

Schließlich sind wiederum unter “MinMSE”, “AvgMSE” und “MaxMSE” die minimalen, mittleren und maximalen MSE-Werte für jeden Netztyp im Rahmen dieser Datenreihe aufgeführt. Dabei beziehen sich diese Abweichungen jeweils auf die Ausgangsfunktion, die das Inverse Pendel (mit einer Partition von 16 einfachen Rechtecken auf dem Definitionsbereich) modelliert.

Anhang A5

Zu Beispiel 7.11 (S. 142) bzw. 7.19 (S. 149):
Neuronales 2-6-1-Netz und hieraus generierte Regelbasis SV-16

Nachfolgend die beispielhaften Daten zu o.g. einfachem Beispiel. Aus Platzgründen sind die Daten an dieser Stelle nur auszugsweise wiedergegeben; komplett finden diese sich auf der Begleit-CDROM.

x und y sind die Eingabegrößen, "Netz" ist die Netzausgabe, "RW" die Ausgabe der generierten Regelmenge. |Netz-RW| ist die betragsmäßige Differenz von Netzausgabe und Regelbasis-Ergebnis. Schließlich ist $|Netz-RW|^2$ die quadratische Differenz von Netz- und Regelbasis-Ausgabe.

Alle Werte sind gerundet auf drei Nachkommastellen.

x	y	Netz	RW	|Netz-RW|	$\|Netz-RW\|^2$
-25,000	-25,000	-0,300	-0,294	0,006	0,000
-25,000	-24,000	-0,300	-0,294	0,006	0,000
-25,000	-23,000	-0,300	-0,294	0,006	0,000
-25,000	-22,000	-0,300	-0,294	0,006	0,000
-25,000	-21,000	-0,300	-0,294	0,006	0,000
-25,000	-20,000	-0,300	-0,294	0,006	0,000
-25,000	-19,000	-0,300	-0,294	0,006	0,000
-25,000	-18,000	-0,299	-0,294	0,005	0,000
-25,000	-17,000	-0,298	-0,294	0,004	0,000
-25,000	-16,000	-0,297	-0,294	0,003	0,000
-25,000	-15,000	-0,295	-0,294	0,001	0,000
-25,000	-14,000	-0,290	-0,294	0,004	0,000
-25,000	-13,000	-0,283	-0,294	0,011	0,000
-25,000	-12,000	-0,270	-0,152	0,118	0,014
-25,000	-11,000	-0,250	-0,152	0,098	0,010
-25,000	-10,000	-0,219	-0,152	0,067	0,005
[...]					
25,000	-10,000	0,009	-0,046	0,055	0,003
25,000	-9,000	0,004	-0,046	0,050	0,003
25,000	-8,000	0,002	-0,046	0,048	0,002
25,000	-7,000	0,001	-0,046	0,047	0,002
25,000	-6,000	0,001	-0,046	0,047	0,002
25,000	-5,000	0,001	-0,046	0,047	0,002
25,000	-4,000	0,001	-0,046	0,047	0,002
25,000	-3,000	0,001	-0,046	0,047	0,002
25,000	-2,000	0,001	-0,046	0,047	0,002
25,000	-1,000	0,001	-0,046	0,047	0,002
25,000	0,000	0,002	0,152	0,150	0,022
25,000	1,000	0,004	0,152	0,148	0,022
25,000	2,000	0,007	0,152	0,145	0,021
25,000	3,000	0,012	0,152	0,140	0,020
25,000	4,000	0,021	0,152	0,131	0,017
25,000	5,000	0,036	0,152	0,116	0,013
25,000	6,000	0,059	0,152	0,092	0,009

25,000	7,000	0,093	0,152	0,059	0,003
25,000	8,000	0,135	0,152	0,017	0,000
25,000	9,000	0,180	0,152	0,028	0,001
25,000	10,000	0,219	0,152	0,067	0,005
25,000	11,000	0,250	0,152	0,098	0,010
25,000	12,000	0,270	0,152	0,118	0,014
25,000	13,000	0,283	0,294	0,011	0,000
25,000	14,000	0,290	0,294	0,004	0,000
25,000	15,000	0,295	0,294	0,001	0,000
25,000	16,000	0,297	0,294	0,003	0,000
25,000	17,000	0,298	0,294	0,004	0,000
25,000	18,000	0,299	0,294	0,005	0,000
25,000	19,000	0,300	0,294	0,006	0,000
25,000	20,000	0,300	0,294	0,006	0,000
25,000	21,000	0,300	0,294	0,006	0,000
25,000	22,000	0,300	0,294	0,006	0,000
25,000	23,000	0,300	0,294	0,006	0,000
25,000	24,000	0,300	0,294	0,006	0,000
25,000	25,000	0,300	0,294	0,006	0,000
				MAE	MSE
				0,075	0,008

MAE = mean absolute error
MSE = mean square error

Anhang A6 - Beispiel Clusterung

Nachstehend wird ein Auszug aus einem Tabellenkalkulationsblatt wiedergegeben, das zur exemplarischen Berechnung der Ausgaben eines 1-6-1-Netzes sowie dazu korrespondierender 1-3-1- bzw. 1-4-1-Netze dient. Vgl. hierzu S. 125.

Beispiel: Ersetzen mehrerer Neuronen der Hidden-Schicht durch ein oder mehrere Neuronen

Clusterung: Intervall und Gruppen; hier wird illustriert, dass mehr Neuronen (1-4-1-Netz, das könnte die Gruppenclusterung sein) besser sein können, aber nicht müssen.

Gegeben: 1-6-1-Netz

	1-6-1-NETZ		1-3-1-NETZ		1-4-1-NETZ	
i	wi	vi	wj	vj	wk	vk
1	1,5	4	1,5	4	1,5	4
2	2,5	2	3,3636364	3,3	2,6	2,5
3	3	0,5				
4	8	0,5			5,75	0,8
5	2	0,3				
6	1	0	1	0	1	0

Epsilon: 0,0001 (dient für Betragsabschätzungen in Spalte H)

Anm.: Formel für wj bzw wk:
Summe (vi wi) / Summe(vi) über beteiligte i

Beispieldaten:

Eingabe x	Output 1-6-1	Output 1-3-1	Output 1-4-1	Delta3	Delta4	D4 besser D3?
-1	1,0960447	0,9376093	1,0366861	-0,158435	-0,059359	+
-0,9	1,3124023	1,0438484	1,2580175	-0,268554	-0,054385	+
-0,8	1,5443576	1,2277493	1,4793489	-0,316608	-0,065009	+
-0,7	1,7763129	1,5305306	1,7006803	-0,245782	-0,075633	+
-0,6	2,0082682	1,833312	1,9220117	-0,174956	-0,086257	+
-0,5	2,2402235	2,1360933	2,1433431	-0,10413	-0,09688	+
-0,4	2,4721788	2,4388746	2,4388746	-0,033304	-0,033304	
-0,3	2,741656	2,741656	2,741656	0	0	
-0,2	3,0444373	3,0444373	3,0444373	0	0	
-0,1	3,3472187	3,3472187	3,3472187	0	0	
0	3,65	3,65	3,65	0	0	
0,1	3,9527813	3,9527813	3,9527813	-4,4E-016	0	
0,2	4,2555627	4,2555627	4,2555627	8,88E-016	0	
0,3	4,558344	4,558344	4,558344	8,88E-016	0	
0,4	4,8278212	4,8611254	4,8611254	0,0333042	0,0333042	
0,5	5,0597765	5,1639067	5,1566569	0,1041302	0,0968805	+
0,6	5,2917318	5,466688	5,3779883	0,1749563	0,0862566	+
0,7	5,5236871	5,7694694	5,5993197	0,2457823	0,0756327	+
0,8	5,7556424	6,0722507	5,8206511	0,3166084	0,0650087	+
0,9	5,9875977	6,2561516	6,0419825	0,2685539	0,0543848	+
1	6,2039553	6,3623907	6,2633139	0,1584354	0,0593586	+
1,25	6,6107872	6,6279883	6,6279883	0,0172012	0,0172012	
1,5	6,893586	6,893586	6,893586	0	0	
1,75	7,1591837	7,1591837	7,1591837	0	0	
2	7,3	7,3	7,3	0	0	

Zur Erläuterung: Alle der hier gezeigten Netze verwenden die Rampenfunktion als Ausgabefunktion in der versteckten Schicht. Das 1-6-1-Netz besteht aus 6 Neuronen im Hidden Layer, die von der Eingabeschicht die Gewichte w_i und zur Ausgabeschicht die Gewichte v_i besitzen. Das 1-3-1-Netz entsteht aus dem 1-6-1-Netz dadurch, dass die (Hidden) Neuronen 2 bis 5 zu einem Substitutionsneuron (gemäß Definition 6.19, sh. S. 119) zusammengefasst werden; entsprechend werden für das 1-4-1-Netz die beiden Neuronen 2 und 3 bzw. 4 und 5 jeweils durch ein Substitutionsneuron in der verborgenen Schicht ersetzt.

Exemplarisch werden in der Tabelle die Ausgabewerte ("Output ...") dieser drei Netze dargestellt; daneben werden mit "Delta3" bzw. "Delta4" die Abweichungen des 1-3-1- bzw. des 1-4-1-Netzes vom Ausgangsnetz aufgeführt. Die Spalte "D4 besser als D3?" schließlich markiert die Fälle, in denen das 1-4-1-Netz eine bessere Approximation liefert als das 1-3-1-Netz.

Anhang A7 - Auszug der Dokumentation zu “Javanna”

Nachstehend wird ein kleiner Auszug aus der mit “javadoc” generierten API-Dokumentation zum Java-Package “javanna” wiedergegeben. Die komplette Dokumentation im HTML-Format befindet sich auf der Begleit-CDROM.

A7.1 Überblick über die Klassen aus dem Package “javanna”

Nachfolgend werden die Klassen des Paketes “javanna” kurz vorgestellt. Hierbei werden innere (Hilfs-)Klassen nicht dargestellt.

Die Klasse *Javanna* ist die Einstiegsklasse (main-Klasse) für die exemplarische graphische Applikation. Sie kann (unter Windows-Betriebssystemen) von der Begleit-CDROM durch Aufruf der Stapelverarbeitungsdatei “javanna.bat”, die auf verschiedenen Verzeichnisebenen vorhanden ist, gestartet werden. Voraussetzung ist eine installierte Java Runtime-Umgebung der Version 1.3. Auf der Begleit-CDROM findet sich das komplette Software Development Kit für Java (JDK 1.3) im Verzeichnis /JavaSDK-1-3-1.

Zusammenfassung der Klassen (Class Summary)

AusgabeNeuron	Implementierung eines AusgabeNeurons durch Vererbung von der Klasse Neuron.
Compare1	Compare1 ist eine Klasse fuer eine einfache Konsolenanwendung, die die Approximationsguete der (theoretisch bestimmten, optimalen) Rampenfunktion zur logistischen (sigmoiden) Funktion anhand von Beispielnetzen des Typs 1-N-1 numerisch ermittelt. Bestimmt werden die durchschnittliche, die maximale und die minimale Abweichung, die ueber die Datensaetze gesehen von den beiden Implementierungen erzielt werden.
Data	Klasse zur Modellierung einer Datenmenge.
DialogAllgemein	Allgemein verwendbare Dialogbox mit einer kurzen Textmeldung. Diese Klasse wurde von der zu Beginn eingesetzten Entwicklungsumgebung *Symantec Visual Café* zunaechst automatisch generiert; anschliessend wurden einige Teile manuell angepasst.
DialogFehler	Dialogbox mit einer kurzen Fehlermeldung.
DialogProgrammBeenden	Dialogbox zum Beenden des Programms.
DialogProgrammInformation	Dialogbox mit einer kurzen Programminformation.
EditorPanel	Diese Klasse realisiert ein sehr einfaches Editor-Panel fuer die Bearbeitung z.B. der Netz- oder Trainingsdaten.

EingabeNeuron	Implementierung eines EingabeNeurons durch Vererbung von der Klasse Neuron.
Javanna	Die Klasse Javanna ist die Einstiegs- bzw. Hauptklasse der Anwendung. Hier ist die statische main-Methode platziert.
JUtil	Die Klasse JUtil (Javanna Utility) ist eine Hilfsklasse mit einer Reihe von statischen public-Methoden.
Netz	Modellierung eines Neuronalen Netzes, Implementierung der Netz-Logik.
NetzinfoPanel	Die Klasse NetzinfoPanel stellt ein Panel zur textuellen Anzeige des Netzes zur Verfuegung; kommuniziert mit der fachlichen Klasse Netz.
NetzPanel	Die Klasse NetzPanel stellt ein Panel zur grafischen Anzeige des Netzes zur Verfuegung; kommuniziert mit der fachlichen Klasse Netz.
NetztopologieDialog	Diese Klasse generiert einen sehr einfachen Dialog zur Bestimmung der Netztopologie. Derzeit ist dies nur die Anzahl der Neuronen im Hidden Layer.
Neuron	Implementierung eines einzelnen Neurons.
Quader	Die Klasse Quader implementiert einen Quader im n-dimensionalen Raum (Rn).
Quader2	Die Klasse Quader2 implementiert einen Quader im zweidimensionalen Raum (R2).
Regel	Die abstrakte Klasse Regel stellt den Grundrahmen einer Regel vom allgemeinen Typ "IF x IN QUADER-DES-RK THEN y IN QUADER-DES-RM" dar.
Regel1	Die Klasse Regel1 repraesentiert einen einfachen Regel-Typus, der zu eindimensionalen reellen Funktionen passt.
Regel2	Die Klasse Regel2 repraesentiert einen einfachen Regel-Typus, der zu reellwertigen Funktionen von zwei Veraenderlichen passt (und somit zu Netzen mit zwei Eingabeneuronen).
Regelbasis	Regelbasis ist die (abstrakte) Zusammenfassung der Regeln (vom abstrakten Typ Regel) zu einem Netz sowie die entsprechende Konfiguration (beispielsweise der erlaubten Wertebereiche).

Regelbasis1	Regelbasis1 ist die Zusammenfassung der Regeln (vom Typ Regel1) zu einem Netz sowie die entsprechende Konfiguration (beispielsweise der erlaubten Wertebereiche). Diese (elementare) Klasse deckt den 1-1-Fall ab, also eindimensionale Eingaben und ebenfalls eindimensionale Ausgaben.
Regelbasis2	Regelbasis2 ist die Zusammenfassung der Regeln (vom Typ Regel2) zu einem Netz sowie die entsprechende Konfiguration (beispielsweise der erlaubten Wertebereiche). Diese Klasse deckt den K-1-Fall ab, also K-dimensionale Eingaben und ebenfalls eindimensionale Ausgaben.
RegelPanel	Die Klasse RegelPanel stellt ein Panel zur textuellen Anzeige der Regeln dar.
SingleData	Klasse zur Modellierung eines Input-Output-Datensatzes.
Verbindung	Darstellung einer gerichteten Verbindung zwischen zwei Neuronen zweier Schichten.

A7.2 Klassenhierarchie

Auszugsweise wird nachstehend (ebenfalls aus der API-Dokumentation entnommen) die Hierarchie der wesentlichen Klassen im Package javanna dargestellt. Jede Einrückungsstufe stellt eine Vererbungsebene dar. Wieder wird aus Gründen der Übersichtlichkeit auf die Erwähnung von Hilfsklassen wie den ActionListener- oder WindowListener-Klassen verzichtet.

- ☐ class java.lang.Object
  - ☐ class javanna.***Compare1***
  - ☐ class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - ☐ class java.awt.Container
      - ☐ class java.awt.Panel
        - ☐ class javanna.***EditorPanel***
        - ☐ class javanna.***NetzinfoPanel***
        - ☐ class javanna.***NetzPanel***
        - ☐ class javanna.***RegelPanel***
      - ☐ class java.awt.Window
        - ☐ class java.awt.Dialog
          - ☐ class javanna.***DialogAllgemein***
            - ☐ class javanna.***DialogFehler***
          - ☐ class javanna.***DialogProgrammBeenden***
          - ☐ class javanna.***DialogProgrammInformation***
          - ☐ class javanna.***NetztopologieDialog***
        - ☐ class java.awt.Frame
          - ☐ class javanna.***Javanna***
  - ☐ class javanna.***Data***
  - ☐ class javanna.***JUtil***
  - ☐ class javanna.***Netz***
  - ☐ class javanna.***NetzVergleich2***
  - ☐ class javanna.***Neuron***
    - ☐ class javanna.***AusgabeNeuron***
    - ☐ class javanna.***EingabeNeuron***
  - ☐ class javanna.***Quader***
    - ☐ class javanna.***Quader2***
  - ☐ class javanna.***Regel***
    - ☐ class javanna.***Regel1***
    - ☐ class javanna.***Regel2***
  - ☐ class javanna.***Regelbasis***
    - ☐ class javanna.***Regelbasis1***
    - ☐ class javanna.***Regelbasis2***
  - ☐ class javanna.***SingleData***
  - ☐ class javanna.***Verbindung***

A7.3 Dateiformat für Neuronale Netze

Das Dateiformat für abgespeicherte (in unserem Rahmen: dreischichtige) Netze ist einfach aufgebaut, wie das nachstehende Beispiel zeigt.

```
# Datei: beispiel.net - 1-3-1-Netz

# Topologie: Anzahl der Neuronen in den einzelnen Schichten:
1 3 1

# Gewichte:
# Von der Eingabeschicht zur Hidden-Schicht
0.7232851 -0.6753568 0.2984641

# Von der Hidden-Schicht zur Ausgabeschicht
-0.46105385
0.94017935
0.6875841

# Ende der Datei beispiel.net
```

Zunächst einmal sind Kommentarzeilen, die mit "#" oder "//" beginnen, möglich, ebenso werden leere Zeilen als formale Kommentarzeilen toleriert.

In der ersten (Nicht-Kommentar-)Zeile stehen drei natürliche Zahlen K, L, M entsprechend der Anzahl der Neuronen in der Eingabe-, der verborgenen sowie der Ausgabe-Schicht. Danach folgen in K Zeilen jeweils L Gewichte (double-Werte) von den Eingabeneuronen zu den Neuronen der verborgenen Schicht. Schließlich folgen L Zeilen mit den jeweils M Gewichten von den Neuronen der verborgenen Schicht zu den Ausgabeneuronen.

Mit diesem bewusst einfach gehaltenen Dateiformat für (zunächst dreischichtige) Netze soll ein eventueller Datenaustausch mit anderer Software zur Emulation Neuronaler Netze erleichtert bzw. ermöglicht werden.

A7.4 Beispielhafter Programmablauf

Nachstehend wird ein kurzer Programmlauf mit der graphischen Anwendung “Javanna” dokumentiert.

Das Programm kann unter Microsoft Windows-Betriebssystemen mit der Batchdatei “javanna.bat”, die in verschiedenen Verzeichnissen der Begleit-CDROM zu finden ist, gestartet werden. Ansonsten kann die Applikation aus dem Verzeichnis /javanna/classes auch mit dem Kommando “`java -cp . javanna.Javanna`” gestartet werden. Voraussetzung ist ein installiertes Java Runtime Environment der Version 1.3, das in der Microsoft-Windows-Version im Verzeichnis /JavaSDK-1-3-1 verfügbar ist.

Nach dem Start der Anwendung zeigt sich die graphische Benutzeroberfläche.

Abb. A7.1: Nach dem Programmstart

Zunächst ist mittels des Schaltknopfes “Netz” ein neues Netz zu definieren. Die vorliegende Anwendung unterstützt 1-L-1-Netze, d.h. eingestellt werden kann die Anzahl der Neuronen in der verborgenen Schicht.

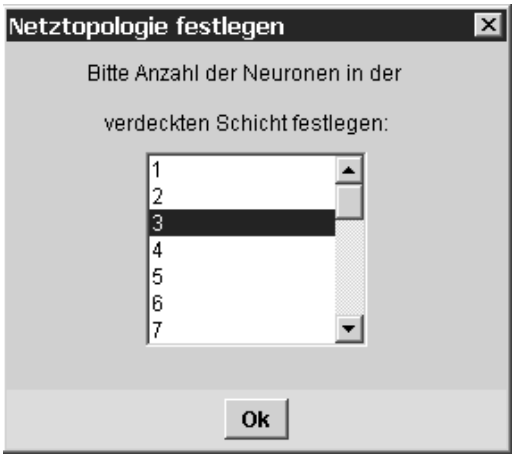


Abb. A7.2: Festlegen der Netztopologie

Wählen wir für die Netztopologie den Wert 3, dann erhalten wir die nachstehende Darstellung.

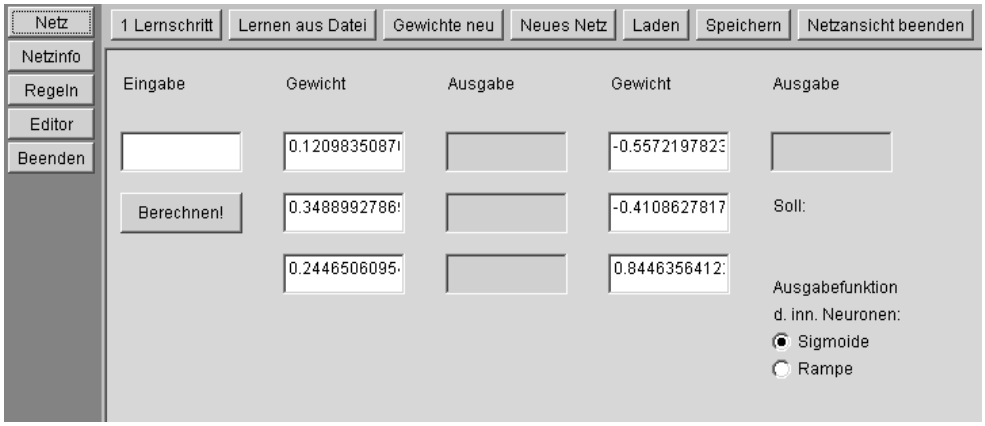


Abb. A7.3: Ein 1-3-1-Netz mit Zufallsgewichten

Die hell unterlegten Textfelder (für Eingabe und die Gewichtswerte) können frei editiert werden. Auf diese Weise können auch manuell Netze konstruiert und Ein-Ausgabe-Muster

erprobt werden. Die im Bild zu sehenden Werte für die Gewichte sind Zufallswerte, die mit "Gewichte neu" jederzeit neu generiert werden können. Daneben kann das aktuell bearbeitete Netz gespeichert und ein bereits in einer Datei vorhandenes Netz geladen werden.

Mit "1 Lernschritt" wird ein einzelner Lernschritt gemäß dem Backpropagation-Verfahren durchgeführt mit der Lernrate, die in der Ansicht "Netzinfo" eingestellt werden kann. Der Vorgabewert hierfür ist 0,1. Sind in einer Datei bereits Eingabewerte gespeichert, so kann mit "Lernen aus Datei" das Lernen beschleunigt werden, indem das Backpropagation-Verfahren auf diesen Eingabewerte operiert. Gelernt wird in dem vorliegenden Programm die Funktion $x \to \frac{1}{2}\sin(x) + \frac{1}{2}$.

Betrachten wir exemplarisch ein einfaches 1-2-1-Netz in der Anwendung. Wir stellen die Gewicht zur Demonstration manuell auf die Werte +1 und -1 ein.

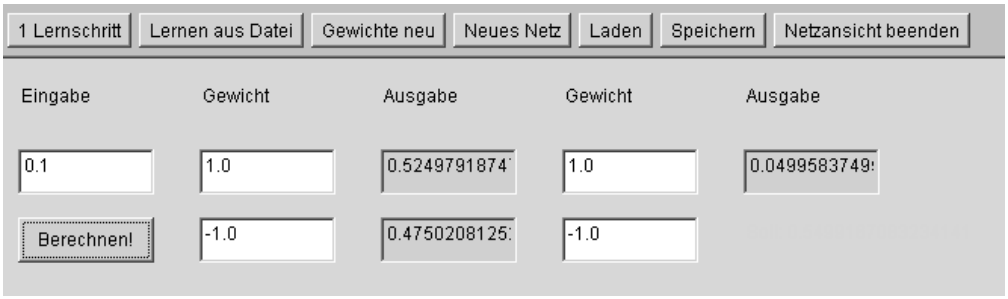


Abb. A7.4: Ein manuell eingestelltes 1-2-1-Netz

Wollen wir das Netz später wieder einmal verwenden, können wir es speichern. Die Applikation besitzt einen kleinen Editor. Aktivieren wir diesen und laden die eben gespeicherte Netz-Datei, dann sehen wir im folgenden Bild noch einmal das Datenformat am konkreten Beispiel.

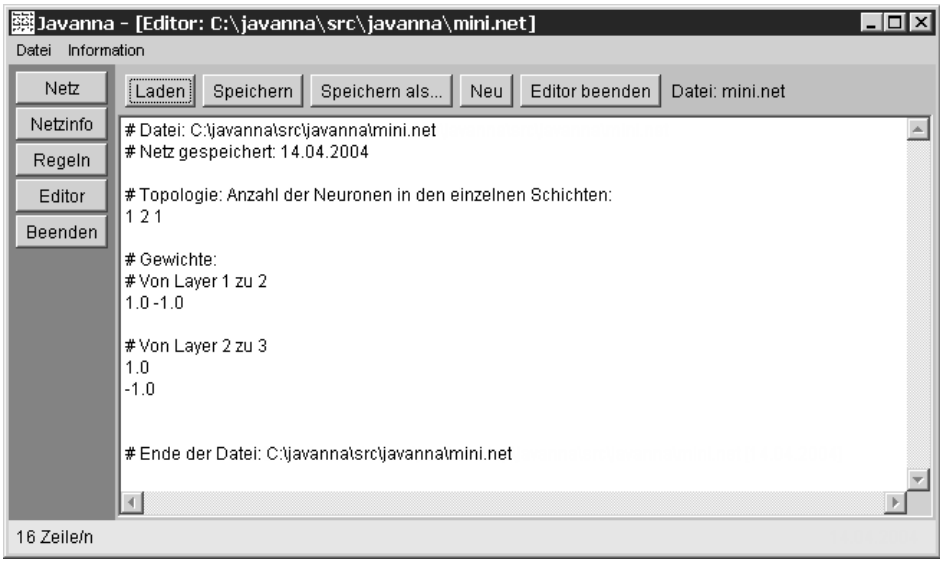


Abb. A7.5: Der Editor mit geladener Netz-Datei

Das Ausgabeverhalten dieses sehr kleinen Netzes kann recht einfach ermittelt werden und wir können uns mittels der Ansicht "Regeln" eine vorzugebende Anzahl Regeln generieren lassen. Wir spielen dies mit vier Regeln gemäß Typ I "Äquidistante x-Unterteilung" durch.

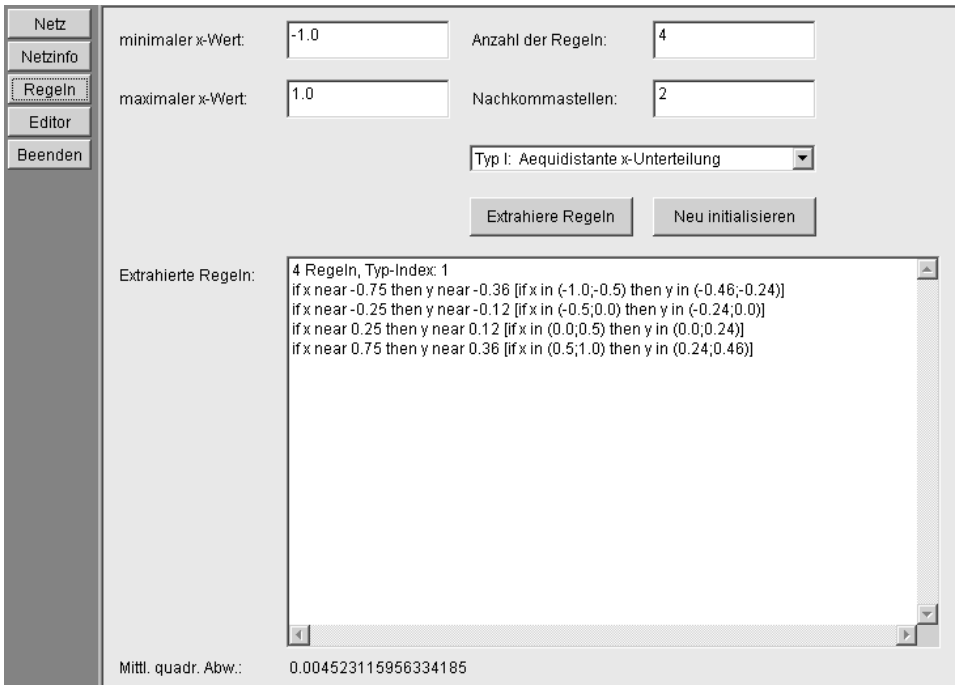


Abb. A7.6: Ansicht auf die erzeugten Regeln

Die Regeln werden nach Festlegung des Eingabebereiches, der gewünschten Anzahl Regeln und der Anzahl der anzuzeigenden Nachkommastellen textuell in dualer Form dargestellt: einmal in Form von fuzzy-ähnlichen “NEAR”-Aussagen, ein zweites Mal in Intervallnotation (siehe obige Abbildung).

In der Ansicht “Regeln” wird auch die mittlere quadratische Abweichung der Regeln im Vergleich zu dem Neuronalen Netz angezeigt. Diese mittlere quadratische Abweichung wird in dem Programm über 100 äquidistante Stützstellen über dem zugrundeliegenden Eingabebereich ermittelt.

Eine ergänzende Anmerkung: wählt man eine sehr große Anzahl von Regeln und lässt sich nur wenige Nachkommastellen anzeigen, so erhält man “optisch” eventuell Regeln scheinbar mehrfach; dies liegt dann aber lediglich an der zu ungenauen Anzeige, durch Heraufsetzen der Anzahl der Nachkommastellen kann diese Situation natürlich vermieden werden.

Lebenslauf

Name:	Peter Baeumle-Courth, geb. Baeumle
Geburtsdatum:	28.Mai 1960
Geburtsort:	Freiburg i. Br. (Baden-Württemberg)
Familienstand:	verheiratet, 2 Kinder
Eltern:	Ernst Baeumle, verst. 1972 Regina Baeumle, geb. Kolich
Staatsangehörigkeit:	deutsch
Schulbildung:	1966 - 1970 Grundschule (Karlsschule Freiburg i. Br.) 1970 - 1979 Kepler-Gymnasium Freiburg i. Br.
Abitur:	30.Mai 1979 in Freiburg i. Br.
Studium:	Mathematik mit Nebenfach Physik (Diplom) an der Albert-Ludwigs-Universität Freiburg i. Br. - Oktober 1979 - Oktober 1985
Prüfungen:	23.Juli 1981 Vordiplom in Mathematik, Nebenfach Physik 31.Oktober 1985 Diplom in Mathematik mit Nebenfach Physik - beide an der Albert-Ludwigs-Universität Freiburg i. Br.
Beruflicher Werdegang:	1981 - 1985 Studentische Hilfskraft am Mathematischen Institut und am Institut für Stochastik der Universität Freiburg i. Br. 1985 - 1988 Wissenschaftlicher Mitarbeiter am Lehrstuhl von Prof. Dr. Pfanzagl, Universität zu Köln 1988 - 1997 Dozent für Mathematik und Angewandte Informatik am b.i.b. Bildungszentrum für informationsverarbeitende Berufe e.V. in Bergisch Gladbach Seit 1998 Dozent an der FHDW Fachhochschule der Wirtschaft (in Trägerschaft des b.i.b.) in Bergisch Gladbach für die Schwerpunktbereiche Software-Entwicklung und Mathematik
Beginn der Dissertation:	Juni 2001 am Institut für Informatik der Westfälischen Wilhelms-Universität Münster bei Prof. Dr. W.-M. Lippe