# GRUNDLAGEN, GRENZEN UND EINSATZ KÜNSTLICHER NEURONALER NETZE

mip

Jörg Kremer

Head of Consulting

Februar 2020

# VORWORT

*Quasi reflexartig werden für analytische Aufgaben aktuell gerne Ansätze aufgerufen, die irgendwie mit „Neuronalen Netzen“ und „Deep-Learning“ zu tun haben. Gerade für Fachanwender, die Chancen und Risiken dieser Technologie für ihre Problemstellungen bewerten müssen, lohnt daher ein intensiverer Blick auf die Grundlagen.*

*Dazu bietet das vorliegende kompakte E-Book ein gutes Fundament und geht dabei deutlich tiefer als manche populärwissenschaftliche Abhandlung. Herr Kremer versteht es, auf Basis dieses Faktenwissens die entscheidenden Schlussfolgerungen für die Grenzen und die Operationalisierbarkeit von Neuronalen Netzen in realen Projekten zu ziehen.*

*Dr. Roland Zimmermann*
*Professor für Wirtschaftsinformatik und Statistik*
*Prodekan der Fakultät Betriebswirtschaft*
*Technische Hochschule Nürnberg Georg Simon Ohm*

# VORWORT DES AUTORS

*Liebe Leserinnen und Leser,*

*vielen Dank für Ihr Interesse an meinem eBook. Tagtäglich liegt mein Augenmerk auf der Leitung und Steuerung unserer großen Kundenprojekte. Dabei geht es um die Verarbeitung der Daten, von der Datenhaltung und -integration bis hin zur Analyse und Visualisierung. Doch als ehemaliger Universitäts-Dozent schlägt mein Herz immer noch für die Wissenschaft. Dies ist meine Motivation, unsere Werk-studenten/innen bei ihren Arbeiten zu betreuen und gemeinsam mit unserem Team in der Welt der künstlichen Intelligenz (KI) zu forschen und zu experimentieren.*

*Rund um das Thema KI gibt es heute eine Menge an Studien und Artikeln. Trotzdem herrscht eine gewisse Unsicherheit dahingehend, was es im Einzelnen bedeutet und wie sich diese neuen Technologien heute schon einsetzen lassen. Das gilt ebenso für die künstlichen neuronalen Netze (KNN), die eine Teildisziplin der KI darstellen.*

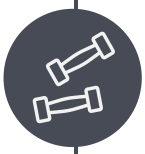
*Wie kann man sich also am besten dem Thema KNN nähern? In dem man zunächst das natürliche Vorbild, unser Gehirn, versteht. Seit Jahrhunderten wird intensiv geforscht, um zu begreifen, wie wir Informationen speichern, denken und lernen. Doch trotz aller Anstrengungen kann die Neurologie noch nicht erklären, wie wir Menschen zu unserem Bewusstsein kommen. Aber was erforscht ist und modellhaft erklärt werden kann, sind die biochemischen Abläufe im menschlichen Gehirn. Diese folgen algorithmischen Mustern, die relativ gut in mathematischen Modellen beschrieben werden können. Im Laufe der Jahre wurden diese Modelle immer weiter verfeinert und bilden die Grundlage zur Erstellung von KNN.*

*Ich möchte Ihnen die Unterschiede künstlicher neuronaler Netze im Vergleich zum menschlichen Gehirn aufzeigen und erklären. Die daraus resultierenden Herausforderungen zeigen Ihnen, wofür KNN heute bereits sinnvoll genutzt werden können. Sie werden verstehen, wo diese an ihre Grenzen stoßen und was Sie in der Praxis beachten sollten. Sie werden erkennen, wie umfangreich die Überlegungen im Vorfeld eines solchen Projektes sind und wie individuell der Aufbau erfolgen muss, um die meist unterschiedlichen Aufgabenstellungen zu lösen. Unabhängig davon, wie tief Sie in diese Thematik einsteigen wollen oder müssen, freue ich mich auf ein Feedback und einen regen Austausch mit Ihnen.*

*Mit besten Grüßen*

*Jörg Kremer*
*Head of Consulting*

# INHALTSVERZEICHNIS



# 1 Die Natur als (unerreichtes) Vorbild

Um künstliche neuronale Netze zu verstehen, ist es unabdingbar, die Grundlagen natürlicher neuronaler Netze, also von Gehirnen, zu verstehen. Sehr einfach definiert, ist ein künstliches neuronales Netz nichts anderes, als der Versuch, die Funktion eines Gehirns mit Hilfe von Computertechnik nachzubauen.

Für spezielle Aufgabenstellungen gelingt dies mittlerweile ganz gut, jedoch sind wir technologisch gesehen noch enorm weit von der Nachbildung komplexer natürlicher neuronaler Netze entfernt. Um die Gründe dafür zu verstehen, werden in diesem Artikel die zentralen Unterschiede und Problemfelder künstlicher neuronaler Netze im Vergleich ihrer natürlichen Vorbilder erklärt. Außerdem wird herausgearbeitet, wofür künstliche neuronale Netze heute bereits gut genutzt werden können und was es in der Praxis zu beachten gilt.

Dabei wird nicht auf wissenschaftliche Details eingegangen, da dies eine sehr umfassende Arbeit erfordern würde. Die zur Erstellung des Artikels genutzten Quellen befinden sich im Literaturverzeichnis auf Seite 1. Da es sich nicht um eine wissenschaftliche Arbeit handelt, wurde auf detaillierte Quellenangaben zu Gunsten der Lesbarkeit verzichtet.

## 1.1 Nervenzellen (Neuronen)

Ein menschliches Gehirn besitzt ca. 100 Milliarden Nervenzellen. Diese speziellen Zellen sind darauf spezialisiert, Informationen an andere Nerven-, Muskel- und Drüsenzellen zu übertragen. Die strukturelle und funktionale Vernetzung der Nervenzellen machen das Gehirn aus.

Jedes Neuron besteht vereinfacht formuliert aus einem Zellkörper, einem Axon und mehreren Dendriten, um Verbindungen zu anderen Nervenzellen herzustellen. Über die Dendriten werden elektrische Impulse von anderen Neuronen empfangen. Die Kontaktstellen zwischen den Nervenzellen werden als Synapsen bezeichnet. Axone verbinden sich über Synapsen mit Dendriten oder Zellköpern, so dass diese Impulse von vielen über die Synapsen verbundenen Neuronen erhalten.

Der Zellkörper besteht aus dem Zellkern (Nucleus) und Zytoplasma. Er fungiert als Speicher von elektrischen Ladungen, vergleichbar mit einem Kondensator oder einer Batterie. Die Aufladung wird initiiert durch Spannungsimpulse anderer Neuronen, wobei die Spannung steigt, je mehr Impulse einwirken. Wenn ein bestimmter Schwellenwert überschritten wird, entlädt sich der Zellkörper.

Der dadurch entstehende elektrische Impuls wird über das Axon an die Synapsen der verbundenen Neuronen verteilt, wo genau der gleiche Prozess abläuft. Jedes der ca. 100 Milliarden Neuronen ist mit 1.000 bis 10.000 anderen Neuronen verbunden und diese bilden das neuronale Netz.

Eine Verbindung zwischen zwei Neuronen kann man sich ähnlich wie zwei Kabel vorstellen, die aufeinandertreffen. Jedoch sind die Kabel nicht perfekt leitend verbunden. Zwischen ihnen liegt ein, für Elektronen unüberwindbarer, kleiner Spalt. In diesem Spalt befinden sich chemische Botenstoffe (Neurotransmitter), die durch die anliegende Spannung an den Synapsen ionisiert werden und die Ladung über den Spalt transportieren.

## 1.2 VERARBEITUNGSPROZESSE UND LERNEN

Der Prozess des Ladens und Entladens passiert durch Ladungsdifferenzen im Inneren und Äußeren der Zelle. Diesen Prozess kann jede Nervenzelle mehrere Male pro Sekunde abwickeln. Neben den elektrischen Impulsen werden auch chemische Botenstoffe ausgeschüttet und damit andere Zellen gesteuert, zum Beispiel um Muskelzellen zu aktivieren.

Wichtig ist außerdem, dass jedes Neuron nicht nur elektrische Ladung über verbundene andere Neuronen empfängt, sondern gleichsam auch abgibt. Ob und wann für ein Neuron die Feuerschwelle der elektrischen Ladung erreicht ist, ist ein ausgesprochen komplexer Prozess.

Das Training des neuronalen Netzes funktioniert, indem die Synapsen ihre Leitfähigkeit verändern können. Je häufiger Spannungsimpulse über eine Synapse übertragen werden, desto höher wird die Leitfähigkeit. Umgekehrt können Synapsen, die keine oder sehr wenige Spannungsimpulse übertragen, irgendwann absterben. In dem Prozess arbeiten alle Neuronen des Gehirns asynchron und parallel, jedoch mit einer Geschwindigkeit von circa einem Hertz, was verglichen mit einer CPU sehr langsam ist. Ein Gehirn erreicht seine unglaubliche Effizienz durch die massive Parallelverarbeitung.

Die nachfolgende Abbildung skizziert ein natürliches neuronales Netz:

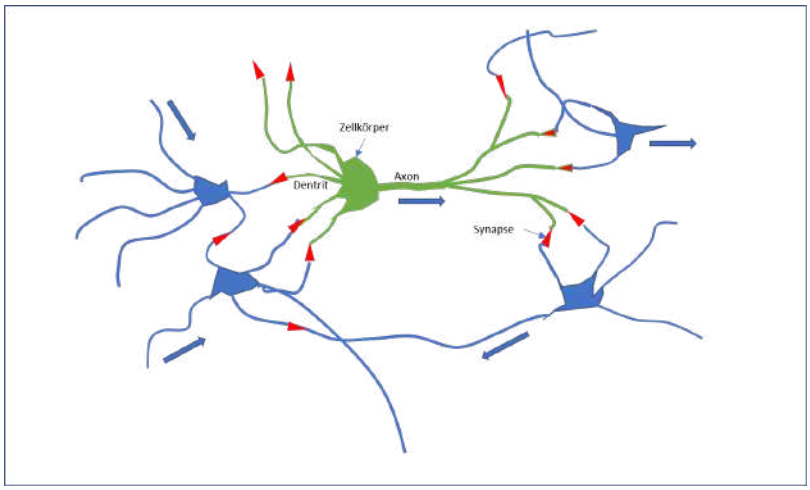



Abbildung 1: eigene Darstellung in Anlehnung an Wolfgang Ertel, Grundkurs künstliche Intelligenz, 3. Auflage, 2013, Kapitel 9.1

Im Gehirn sind verschiedene Bereiche für unterschiedliche Funktionen zuständig und die Verbindungen sind entsprechend aufgebaut. Die Reize auf das Gehirn sind typischerweise sehr vielfältig und passieren massiv parallel.

Etwa sehen wir, wie auf dem Herd ein Topf Milch überkocht, riechen die verbrannte Milch und spüren in der Nähe der Herdplatte die Hitze. Zudem ist unser Gehirn auf diese oder ähnliche Ereignisse trainiert, vor allem, wenn wir uns als Kind an einer heißen Herdplatte verbrannt haben. Diese Flut an Reizen führt dann zu Handlungsmustern, zum Beispiel: Herdplatte ausschalten, Topf mit Topflappen vom Herd nehmen, Fenster öffnen etc.. Diese Handlungsmuster werden über unser Gehirn, unser neuronales Netz gesteuert. Wenn wir eine Situation zum ersten Mal erleben, versuchen wir diese mit einer möglichst ähnlichen Situation zu assoziieren. Wenn dann die ausgelösten Handlungsmuster zu anderen Ergebnissen führen, trainieren wir dadurch unser Gehirn und modifizieren unsere Handlung gegebenenfalls. Dies alles passiert mit minimalsten Reaktionszeiten und extrem parallelisiert.

Eine zentrale Eigenschaft des Gehirns, die wir an dieser Stelle festhalten, ist die enorm hohe Zahl an Neuronen und die enorme Dichte der Vernetzung dieser Neuronen untereinander. Dass es auf absehbare Zeit unmöglich ist, ein solch komplexes Netzwerk mathematisch (auch nur andeutungsweise vollständig) zu beschreiben, ist mehr als offensichtlich. Wenn wir annehmen, dass die Information über Ort und Zeit eines Aktionspotentials, zum Beispiel einer Muskelreaktion, 20 Bits benötigt, ergibt sich bei 100 Milliarden Neuronen und einer Feuerrate von einem Hertz pro Sekunde 2,5 Terrabyte Informationsmenge, die erzeugt werden kann.

Modernste Computer und/oder Cluster besitzen durchaus mehrere 100.000 Prozessoren. Selbst im Cluster wird es jedoch noch sehr lange dauern, bis wir, wenn man vereinfacht jedes Neuron mit einem CPU-Kern gleichsetzt, die physikalischen Voraussetzungen haben, ein künstliches neuronales Netz in der Komplexität eines menschlichen Hirns aufzubauen.

# 2 Künstliche neuronale Netze

Künstliche neuronale Netze gehen auf die grundlegende Forschung von Warren McCulloch und Walter Pitts aus dem Jahr 1943 zurück. In ihrem Aufsatz „A logical calculus of the ideas immanent in nervous activity" stellten sie erstmalig ein mathematisches Modell des Neurons als zentrales Schaltelement des Gehirns vor. Durch die Formalisierung des Neurons und seiner Funktion wurde die Grundlage geschaffen, neuronale Netze künstlich nachzubauen.

## 2.1 ELEMENTE KÜNSTLICHER NEURONALER NETZE

### 2.1.1 Units

Jedes neuronale Netz besteht aus mehreren Units oder Knoten (Neuronen), die Informationen von anderen Units oder der Umwelt aufnehmen und in modifizierter Form an andere Units oder die Umwelt abgeben. Hierbei kann man zwischen

- Input-Units (die Informationen aus der Außenwelt aufnehmen),
- Hidden Units (die sich zwischen Input- und Output-Units befinden und eine interne Repräsentation der Außenwelt beinhalten) und
- Output-Units (die Informationen an die Außenwelt weitergeben)

unterschieden werden.

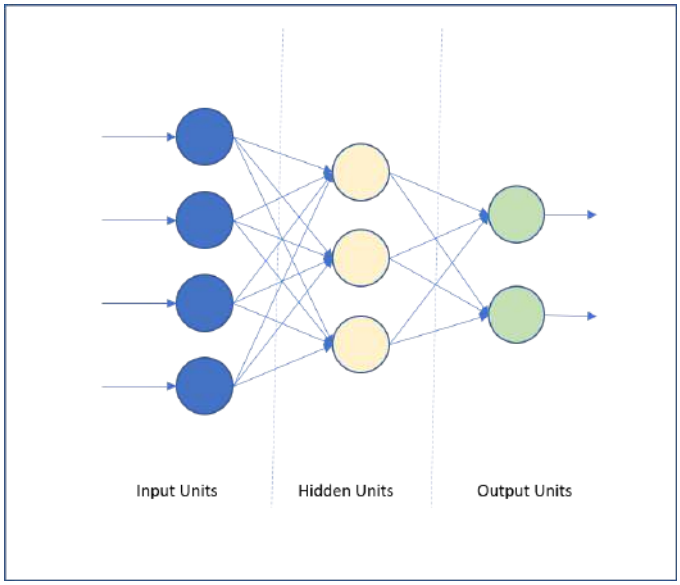



Abbildung 2: eigene Darstellung: schematisches künstliches neuronales Netz (in den meisten neuronalen Netzen gibt es mehrere Schichten von hidden Units)

### 2.1.2 Kanten

Die Units eines Netzes sind über sogenannte Kanten verbunden, wobei die Stärke der Verbindung, also der Einfluss zwischen zwei Units, über ein Gewicht ausgedrückt wird. Dabei ist wichtig, dass jede Kante genau mit einem individuellen Gewicht ausgestattet ist. Positive Gewichte drücken einen exzitatorischen (erregenden) Einfluss aus, während negative Gewichte inhibitorisch (hemmend) sind. Ein Gewicht von 0 bedeutet, dass sich die betroffenen beiden Units zum Betrachtungszeitpunkt nicht beeinflussen. Das Lernen eines künstlichen neuronalen Netzes geschieht durch Veränderung der Gewichtungen nach einer Reihe von Lernregeln.

### 2.1.3 Input und Netzinput

Der Input einer Unit ergibt sich durch Multiplikation des Outputs (des Aktivitätslevels) der sendenden Unit mit dem Gewicht zwischen den beiden Units. Da es mehr als eine eingehende Verbindung zu jeder Unit geben kann, muss der gesamte Netzinput betrachtet werden. Dieser errechnet sich anhand einer Propagierungsfunktion. Am verbreitetsten ist eine Linearkombination, bei der alle Einzelinputs summiert werden. Formal lässt sich der Input der Unit i (von der Unit j) darstellen als:

$\text{Input}_{ij} = a_j \cdot w_{ij}$, wobei a das Aktivitätslevel ist und w das Gewicht.

Der Netzinput lässt sich entsprechend darstellen als:

$$\text{Netzinput}_i = \sum_j Input_{ij} = \sum_j a_j w_{ij}$$

### 2.1.4 Aktivitätsfunktion, Aktivitätslevel und Output

Der Netzinput einer Unit wird mit Hilfe einer Aktivitätsfunktion in ein Aktivitätslevel der entsprechenden Unit transferiert. Der Aktivitätslevel wird dann als Output an andere Neuronen weiter versendet. Hierbei kommen typischerweise Sigmoide Funktionen zum Einsatz (logistische oder Tangens-Hyperbolicus Funktionen). Diese Funktionen liefern bei einem hohen negativen Betrag des Netzinput eine 0 (logistisch) bzw. -1 (Tangens-Hyperbolicus), steigen dann langsam an, werden dann steiler, ähnlich einer linearen Funktion, um sich dann bei hohen positiven Beträgen langsam dem Wert 1 anzunähern. Diese Funktionen begrenzen das Aktivitätslevel (ein verrauschter Input wird also mehr oder weniger ignoriert) und die Funktion ist im Gegensatz zu einer binären Funktion mit Schwellenfunktion immer differenzierbar (kennt also nicht nur 1 und 0).

### 2.1.5 Bias Unit

Bias Units haben keine eingehenden Verbindungen im Netz. Ihr Aktivitätslevel ist immer +1. Das Gewicht zwischen einer Bias Unit und einer anderen Unit kann jedoch positiv oder negativ sein. Mit Hilfe der Bias Units kann man Schwellen im neuronalen Netz definieren (negatives Gewicht), die andere Input-Units erst überschreiten müssen. Da die Schwelle von den Gewichten abhängt, ist sie durch das Trainieren des Netzes, im Gegensatz zur Schwelle der Aktivitätsfunktion, veränderbar. Eine weitere Verwendung besteht darin, eine Unit mit einem positiven Gewicht häufiger feuern zu lassen als gewöhnlich.

Wenn man das neuronale Netz aus Abbildung 1 in ein künstliches neuronales Netz überführt, entsteht schematisch folgendes Netz:

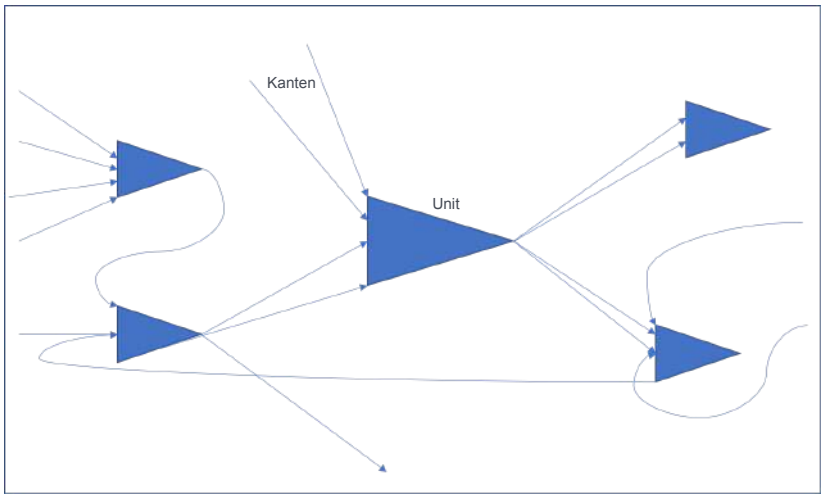



Abbildung 3: eigene Darstellung in Anlehnung an Wolfgang Ertel, Grundkurs künstliche Intelligenz, 3. Auflage, 2013, Kapitel 9.1

## 2.2 TRAINIEREN EINES KÜNSTLICHEN NEURONALEN NETZES

Künstliche neuronale Netze werden zunächst mit vorgegebenem Lernmaterial trainiert. Die Gewichte werden normalerweise mit zufälligen Werten initialisiert. Das entspricht vom Prinzip dem Hirn eines Menschen, das erst durch das Training mit Reizen die Synapsen ausbildet. Dadurch verändern sich die Gewichte zwischen den einzelnen Units. Die verwendete(n) Lernregel(n) bestimmen dabei, wie genau die Veränderungen der Gewichte vorgenommen werden. Grob kann man zwischen überwachtem (beaufsichtigtem) Lernen und nicht überwachtem (unbeaufsichtigtem) Lernen unterscheiden. Bei dem überwachten Lernen wird der Output vorgegeben und dann die Gewichte optimiert. Beim nicht überwachten Lernen hingegen erfolgen die Veränderungen der Gewichte anhand der Ähnlichkeit der Inputreize.

### 2.2.1 Hebb Regel (unbeaufsichtigtes Lernen)

Die Hebb Regel geht auf den Psychologen Donald Olding Hebb zurück. Vereinfacht besagt die Regel, dass eine starke Interaktion zwischen zwei Neuronen eine Erhöhung der Leitfähigkeit der Synapsen zwischen diesen beiden Neuronen bewirkt. Dadurch wird die Erzeugung eines Aktionspotentials von Neuron A auf Neuron B vergrößert. Übersetzt man dies auf künstliche neuronale Netze, wird das Gewicht zwischen zwei Units verändert, wenn beide Units aktiv sind. Hierbei sind drei Parameter entscheidend:

- Der Output der sendenden Unit $a_j$,
- der Output der empfangenden Unit $a_i$,
- der (positive) Lernparameter ε

Als Formel: $\Delta w_{ij} = \varepsilon a_i a_j$

Ein großer Nachteil der Hebb Regel ist, dass die Gewichte bei Units mit Werten zwischen 0 und 1 immer nur wachsen können. Durch Modellierung einer Zerfallskonstanten kann bei unbenutzten Gewichten diesem Problem entgegengewirkt werden. Die Hebb Regel lässt sich maximal auf Netze mit 2 Ebenen anwenden. Netze mit nur 2 Ebenen sind heutzutage eher unüblich, aber für ein grundsätzliches Verständnis, ist die Hebb Regel hilfreich.

### 2.2.2 Delta Regel (überwachtes Lernen)

Grundlage der Delta Lernregel ist der Vergleich zwischen dem beobachteten Output und dem gewünschten Output.Wenn die beobachtete Aktivität bei positivem Input zu niedrig ist, werden die Gewichte zwischen sendender und empfangender Unit erhöht. Im Falle eines zu niedrigen negativen Inputs werden die Gewichte entsprechend gesenkt.Bei einer zu positiven Aktivität werden umgekehrt bei positivem Input Gewichte gesenkt und bei negativem Input erhöht.Entspricht die beobachtete Aktivität der gewünschten Aktivität bleiben die Gewichte gleich.

Als Formel:: $\delta = a_{i\,erwartet} - a_{i\,beobachtet}$

$$\Delta w_{ij} = \varepsilon \delta_i a_j$$

Der Trainingsparameter ε steuert die Stärke der Gewichtsveränderung und wird zu Beginn des Trainings festgelegt. Auch die Delta Regel ist auf eine Anwendung von neuronalen Netzen mit maximal zwei Ebenen anwendbar. Sie kann angewandt werden, um lineare Funktionen aus Daten zu generieren. Bei der Modifikation der Delta Regel in eine inkrementelle Delta Regel kann anstelle einer linearen Funktion eine Klassifizierung definiert werden.

### 2.2.3 Backprogation (überwachtes Lernen)

Die bisher dargestellten Lernregeln sind lediglich auf Input- und Output-Units anwendbar. Sobald das künstliche neuronale Netz auch Hidden Units benötigt, beispielsweise um eine logische XOR-Verbindung abzubilden (a oder b aber nicht beides), werden Lernregeln benötigt, die auch Wirkung auf die Hidden Units ausüben.
Das Verfahren der Backprogation definiert eine Rechenregel, mit der die Gewichte der Hidden Units modifiziert werden können. Es setzt in der bekanntesten Variante (nach Rummelhart, Hinton und Williams, 1986) auf der inkrementellen Delta Regel auf. Die in heutigen Anwendungen genutzten künstlichen neuronalen Netze verwenden typischerweise dieses Lernverfahren.

#### *2.2.3.1 Algorithmus*

Der Algorithmus der Backprogation funktioniert nach folgendem Schema:

- Forward-Pass-Phase: Training des Netzes mit Inputreizen und Berechnung der Outputs.

- Fehlerbestimmung: Die gewünschten Outputs werden mit den in der Forward-Pass-Phase berechneten Outputs verglichen. Ändert sich das Delta zwischen gewünschten und berechneten Output nicht mehr signifikant, wird diese Phase abgebrochen und es folgt die dritte und letzte Phase des Algorithmus.

- Backward-Pass: Die Fehlerterme werden nun in umgekehrter Richtung (also Richtung Input Units) ausgebreitet. Die Gewichte werden so modifiziert, dass die Fehlerterme kleiner werden und nach mehreren Iterationen innerhalb der Güteschwelle liegen. Beginnend mit der Output-Schicht und der letzten Hidden Schicht geht man dabei Ebene für Ebene bis zur Input Schicht zurück.

In der Backward-Pass-Phase kommt zur Modifikation der Gewichte typischerweise das Gradientenabstiegsverfahren zum Einsatz.

### *2.2.3.2 Gradientenabstiegsverfahren*

Die folgende Abbildung veranschaulicht das Gradientenabstiegsverfahren:

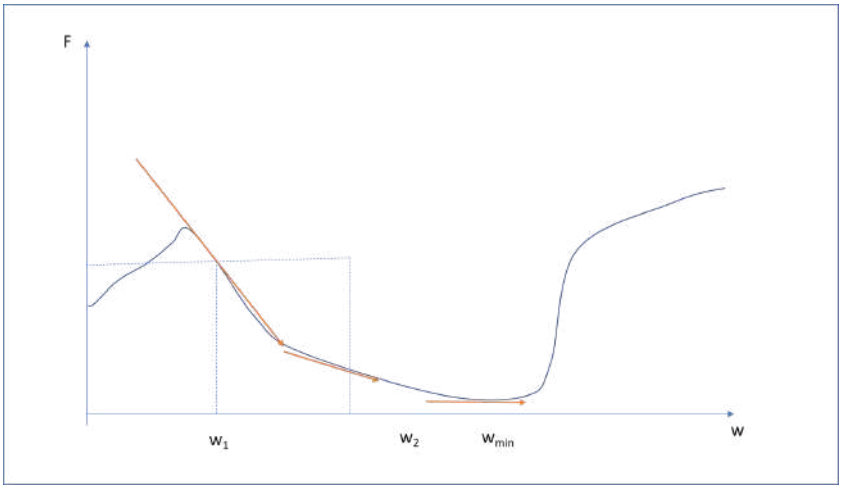



Abbildung 4:

Gradientenabstiegsverfahren, eigene

Darstellung in Anlehnung an

http://www.neuronalesnetz.de/backpropag

ation4.html

Das Verfahren beginnt mit einer zufällig gewählten Gewichtungskombination. Für diese Kombination wird der Gradient (Ableitung der mehrdimensionalen Analysis), dargestellt als orangener Vektor, berechnet. So wird das lokale Minimum berechnet. Dann wird das Gewicht gemäß der definierten Lernkonstante modifiziert (hier erhöht). Es wird erneut für die nun geltende Kombination der Gradient berechnet. In dem obigen Beispiel erhält man nach drei Iterationen das globale Minimum. Das Verfahren ist jedoch fehleranfällig.

### *2.2.3.3 Probleme des Gradientenabstiegsverfahrens*

Verfahrensbedingt ist es unklar, ob man ein lokales oder ein globales Minimum gefunden hat. Je mehr Dimensionen ein Netz hat, umso schwerwiegender tritt dieses Problem zu Tage, da die Fehlerterm-Funktion stark verklüftet sein kann. Hingegen kann der Gradient bei sehr flachen Plateaus so klein werden, dass mit dem Verfahren das nächste Tal mit dem nächsten Minimum gar nicht erreicht wird und das Verfahren somit festsitzt. Sehr kleine Täler der Fehlerterm-Funktion werden unter Umständen übersprungen, sodass wieder nur ein lokales Minimum gefunden wird. Je nach Verlauf kann das Verfahren auch oszillieren, also nach mehreren Iterationen wieder bei der Ausgangsgewichtung landen.

Um diese Probleme anzugehen, können entweder die Lernrate oder die Eingangsgewichte verändert werden. Typischerweise macht es Sinn, im Algorithmus mit einer höheren Lernrate zu starten und diese in kleinen Schritten zu senken, um das Training des neuronalen Netzes zu optimieren. Je geringer die Lernrate ist, desto langwieriger ist jedoch der Lernprozess. Mit geringen Lernraten ist die Gefahr des Oszillierens und des Überspringens globaler Minima geringer und es kann eine größere Datendichte bewältigt werden. Die Gefahr der Stagnation in einem Plateau steigt jedoch an. Die Schwächen des Gradientenverfahrens haben dazu geführt, dass andere Verfahren entwickelt wurden und werden. Das Gradientenverfahren wird allerdings auch heute noch häufig genutzt.

### 2.2.4 Competitive Learning (unüberwachtes Lernen)

Das neuronale Netz nimmt bei diesem Verfahren anhand der Ähnlichkeit der Input-Reize Kategorisierungen vor. Das geschieht wieder in drei Phasen:

- Erregungsphase: Für alle Output Units wird der Netzinput anhand dieser bekannten Formel berechnet.

$$netinput_i = \sum_j a_j w_{ij}$$

- Wettbewerbsphase: Die Output Unit mit dem höchsten Netzinput wird als Gewinner ermittelt.

- Gewichtsadjustierung: Nun werden die Gewichte bei allen Verbindungen, die zur Gewinner-Unit aus der Wettbewerbsphase führen, justiert. Die Modifikation erfolgt so, dass die Gewichte dem Input ähnlicher gemacht werden.

$$\Delta w_{ij} = \mathcal{E}(a_j - w_{ij})$$

Der Nachteil dieses Verfahrens ist, dass einzelne Output Units alle Input-Muster an sich reißen können und somit keine Kategorisierung mehr stattfindet.

## 2.3 WICHTIGE NETZTYPEN

Die verschiedenen Netztypen unterscheiden sich zunächst hinsichtlich der Anzahl der Ebenen, bzw. danach, ob es Hidden Units gibt oder nicht. In der zweiten Ebene unterscheiden sich die Netze hinsichtlich der verwendbaren Lernregeln. Nachfolgend werden einige gängige Netztypen kurz skizziert.

### 2.3.1 Pattern Associator

Pattern Associator Netze sind künstliche neuronale Netze, die zur Mustererkennung genutzt werden können. Sie bestehen aus einer Ebene von Input Units und einer Ebene von Output Units, also ohne Hidden Units. Das Grundprinzip beruht darauf, dass das Netz Assoziationen verschiedener Reizpaare erlernt. Das Training des Netzes erfolgt typischerweise mit der Hebb Regel oder der Delta Regel.

Mit Hilfe der Generalisierung (oder auch Diskrimination) unterschiedlicher Reize werden ähnliche Reize zu einer Reizgruppe kategorisiert. So können dann beispielsweise ein Muster bzw. ein Bild in die passende Kategorie eingeordnet werden, z.B. werden verschiedene Bärenarten der Kategorie Bär zugeordnet. Allerdings besteht hierbei die Anfälligkeit für eine Übergeneralisierung. Möglicherweise wird dann der Wal, obwohl er ein Säugetier ist, fälschlicherweise der Kategorie Fische zugeordnet.

Pattern Associator Netze sind relativ tolerant gegen interne und externe Schäden des Netzes. Auch wenn einzelne Units oder Kanten „absterben“ (interne Schäden), wird mit einer hohen Wahrscheinlichkeit der korrekte Output erzeugt. Auch unvollständiger fehlerhafter Input wird gut toleriert (z.B. Gesichtserkennung trotz verdeckter Augen durch eine Sonnenbrille). Das Netz bildet im Laufe der Lernphase Prototypen der verschiedenen Muster aus.

### 2.3.2 Rekurrente Netze

Die Besonderheit rekurrenter Netze besteht darin, dass Rückkopplungen von Units einer Schicht zu Units derselben oder einer vorangehenden Schicht existieren. Dieser Aufbau eignet sich besonders zur Analyse von Zeitreihen, Texten oder Videos, um Muster im Verlaufe der Zeit zu erkennen. Man kann zwischen verschiedenen Arten von Rückkopplung unterscheiden.

- Direkte Rückkopplungen bedeuten, dass der Output einer Unit zum (zusätzlichen) Input der gleichen Unit wird.
- Bei indirekten Rückkopplungen leitet eine Unit ihren Output an vorangegangene Schichten als Input zurück.
- Eine seitliche Rückkopplung bedeutet, dass innerhalb einer Schicht eine Unit ihren Output als Input an eine andere Unit der gleichen Schicht weitergibt.
- Bei einer vollständigen Verbindung sind alle Units miteinander verbunden (mit oder ohne direkte Rückkopplungen).

#### *2.3.2.1 Beispiel aus der Mustererkennung*

Wir illustrieren dies anhand folgenden Beispiels: In einem Gitternetz mit beispielsweise 10 mal 10 rechteckigen Feldern werden Zahlen dadurch dargestellt, dass jedes Feld entweder gefärbt (1) oder nicht gefärbt (0) ist. Wir erzeugen also ein künstliches neuronales Netz (in diesem Fall einen Autoassoziativspeicher mit vollständiger Vernetzung aller Units miteinander) mit 100 Neuronen und (100 * 99) / 2 = 4.950 Kanten.

#### *2.3.2.2 Hopfield Netz als Autoassoziativspeicher (autonomes Lernen mit modifizierter Hebb Regel)*

Um obiges Beispiel abzubilden, wählen wir ein Hopfield-Netz (nach John Hopfield, 1982) mit 100 Units. Die Units sind vollständig vernetzt, jedoch ist die Gewichtsmatrix symmetrisch und es gibt keine direkte Rückkopplung. Die Speicherung der zunächst zu erlernenden Muster geschieht in einer 10 x 10 Matrix, in der jedes Element entweder 1 oder -1 ist. Zunächst lernen wir also N binär kodierte Muster, die wir in den Vektoren $q^1, \dots, q^n$ speichern,

mit $q_j^i \in \{1, -1\}.$

Das Lernen der N Muster erfolgt mit einer der Hebb Regel verwandten Funktion:

$$w_{ij} = \frac{1}{n} \sum_{k-1}^{N} q_i^k q_j^k|$$

Immer dann, wenn die Pixel i und j den gleichen Wert haben, wird ein positiver Beitrag zum Gewicht $w_{ij}$ geliefert, ansonsten wird ein negativer Beitrag ausgegeben. Weil jeder Pixel durch ein Neuron abgebildet ist, werden also Gewichte zwischen Units gestärkt, die gleichzeitig den gleichen Wert haben.

Die Speicherung der Ziffern 1-4 in der Matrix könnte beispielsweise so aussehen und genau damit erfolgt die Lernphase:

Abbildung 5: Hopfield Netz Matrix Beispiel

Ein rekurrentes Hopfield Netz würde für eine 3 x 2 Pixel Matrix folgendermaßen aussehen:

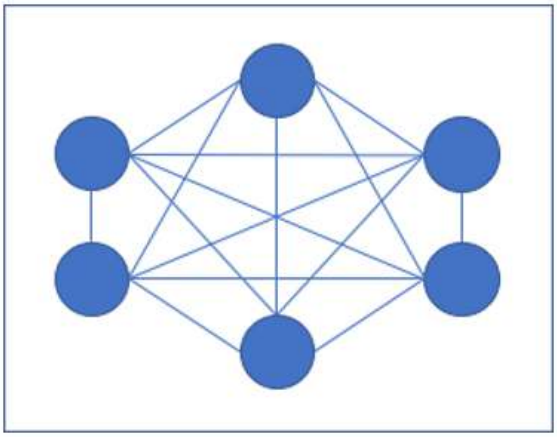


Abbildung 6: Hopfield Netz mit 6 Units und 3 Ebenen und 15 Kanten (und 15 Gewichten)

In der Abbildung erkennt man, dass jede der sechs Units mit jeder anderen Unit verbunden ist. Das heißt jede Unit bekommt Inputreize von allen anderen Units und erzeugt anhand der Aktivierungsfunktion auch Outputs die an alle Units außer an sich selbst geschickt werden. Beim Lernen bekommt je Iteration das Netz die Outputwerte der davorliegenden Iteration als Inputwerte. Dadurch verändern sich die Gewichte und nach n Iterationen konvergiert das Netz nicht weiter und bleibt stabil.

Für unser Beispiel brauchen wir jedoch 100 Units und 4.950 Kanten, was sich nicht mehr übersichtlich darstellen lässt.

Nachdem diese vier Muster gespeichert sind, kann es zur Mustererkennung genutzt werden. Hierzu geben wir ein neues Muster X als Input. Die Aktivierung aller Neuronen wird nun in einem asynchronen Prozess nach folgender Regel (Aktivierungsfunktion) mit n Iterationen durchlaufen, bis sich keine Aktivierungen mehr ändern.

$$x_i = \begin{cases} -1 \; falls \sum_{j-1, j \neq i}^{n} w_{ij} x_j < 0 \\ ansonsten \; 1 \end{cases}, wobei \; i \; für \; die \; Iteration \; steht$$

Bei 10% Rauschen konvergiert das Netz bei den gelernten 4 Ziffern in vielen Fällen nach zwischen 300 und 400 Iterationen zu den gelernten Mustern. Es gibt jedoch auch stabile Zustände, die nicht gelernt wurden und die etwa bei 20% Rauschen entstehen. Folgendes Beispiel veranschaulicht den Zustand im Laufe der Iterationen bis zur Konvergenz bei einem Inputreiz mit einer um 10% verrauschten Ziffer 3:

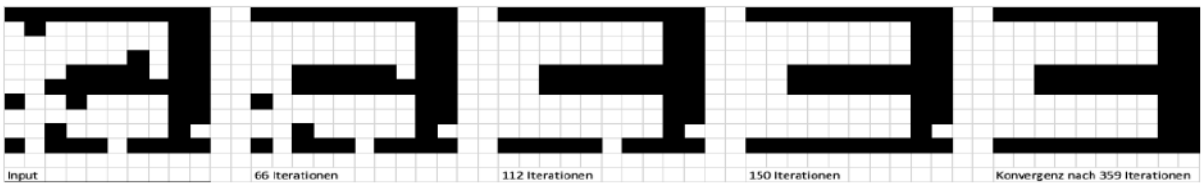



Abbildung 7: Grafische Darstellung Hopfield Assoziation bis zur Konvergenz

Wenn man zusätzlich die Ziffern 0,5,6,7,8,9 speichert, ist dieses Netz mit 100 Neuronen nicht mehr in der Lage die Information zu speichern, da der Speicherplatz nicht ausreicht. Schon bei gering verrauschten Mustern zeigt das Hopfield Netz ein chaotisches Verhalten.

Auf eine genaue mathematische Erklärung wird an dieser Stelle verzichtet. Das Beispiel soll dazu dienen, den grundsätzlichen Aufbau eines Autoassoziativspeicher zu verstehen und zu erkennen, dass selbst bei diesem relativ simplen Beispiel sehr bald systeminhärente Grenzen erreicht werden. Hopfield Netze haben sich auf Grund ihrer Instabilität in der Praxis nicht durchgesetzt, da sie durch die vollständige Vernetzung zu komplex zu handhaben sind, sowohl bzgl. der Hardware-Anforderungen als auch hinsichtlich des Aufwandes während des Lernens.

#### *2.3.2.3 Weitere rekurrente Netze*

Bei Netzen mit indirekter Rückkopplung geben Units ihren Output nur an die hinter ihnen liegende Ebenen zurück. Beispiele sind

- Simple recurrent networks
- Jordan Netze
- LSTM Netze (Long Short Term Memory)

Diese künstlichen neuronalen Netze können gut verwendet werden für

- Prognoserechnungen
- Sprachsimulation oder
- Attraktorennetze.

Letztere lernen mit Backprogation, die je nach Input mit einem von n stabilen Outputzuständen reagieren. Das heißt, es gibt z.B. 5 mögliche Outputs (etwa Kategorien), die je nach Input von der Aktivierungsfunktion ermittelt werden.

Im Rahmen dieser Arbeit wird auf diese rekurrenten Netze nicht weiter eingegangen.

### 2.3.3 Sonstige fortgeschrittene Netztypen

Neben den bisher beschriebenen Netztypen sind für bestimmte Anwendungsfälle (z.B. Wortkodierung) auch Varianten mit unbeaufsichtigtem Lernen gängig. Im Bereich der feed forward Netze (nicht rekurrent) haben sich insbesondere im Bereich der Bildklassifizierung Convolutional Neural Nets durchgesetzt. Da dieser Artikel nur einen Einstieg in die Materie der neuronalen Netze bieten soll, werden diese Ansätze hier nicht weiter beschrieben.

# 3 EIGENSCHAFTEN UND GRENZEN KÜNSTLICHER NEURONALER NETZE

Nachdem die elementaren Grundlagen künstlicher neuronaler Netze im Kapitel 2 dieser Arbeit gelegt wurden, leiten wir nun die sich daraus resultierenden zentralen Eigenschaften der Netze ab. Die vorangegangenen Kapitel haben zudem bereits ausführlich auf Grenzen und Probleme der Verfahren hingewiesen. Was das für die praktische Verwendung bedeutet, wird in diesem Kapitel zusammengefasst.

## 3.1 ZENTRALE EIGENSCHAFTEN

### 3.1.1 Parallelverarbeitung

Biologische neuronale Netze führen alle notwendigen Berechnungen parallel aus. Überträgt man diese Eigenschaft auf künstliche neuronale Netze, besteht bei heutigen Computern die Problematik, dass die Anzahl parallel verarbeiteter Berechnungen begrenzt ist. Allerdings sind moderne Prozessoren so schnell getaktet, dass im Vergleich zum Gehirn relativ gesehen weniger Prozessoren gebraucht werden. Jeder Prozessorkern ist in künstlichen neuronalen Netzen für n Units zuständig und die Berechnung wird je nach Komplexität des Netzes und der Verfügbarkeit von Prozessorkernen serialisiert. Allerdings ist man auch mit modernster Hardware noch relativ weit von den Verarbeitungsmöglichkeiten menschlicher Gehirne entfernt. Alleine die Rechengeschwindigkeit eines Hirns ist heute noch etwa 100-mal so hoch, wie die eines Computers.

### 3.1.2 Verteilte Speicherung

In neuronalen Netzwerken erfolgt die Speicherung von Informationen verteilt über alle oder zumindest eine Vielzahl der Gewichte des neuronalen Netzes und nicht lokal, wie etwa bei einer Festplatte. Am Beispiel des Hopfield Netzes aus Kapitel 2.4.2.2 haben wir gesehen, dass je nach Art und Aufbau des Netzes eine große Zahl von Units und Verbindungen benötigt werden, um eine ausreichende Menge an Informationen (zum Beispiel Gesichter für eine Gesichtserkennung) zu speichern. In Kombination mit den je nach Einsatzziel notwendigen zahlreichen Iterationen der Berechnung sind heutige künstliche neuronale Netze typischerweise auf bestimmte Einsatzszenarien spezialisiert und dann auch nur dafür nutzbar.

### 3.1.3 Fehlertoleranz

Wie wir in den vergangenen Kapiteln gesehen haben, sind künstliche neuronale Netze tolerant gegenüber internen Schäden („absterben“ einzelner Units) und gegen externe Fehler (z.B. Rauschen in einem Muster). Jedoch geht dies nicht so weit wie die Fähigkeiten eines Gehirns, wie beispielsweise nach einem Schlaganfall. Obwohl dabei Teile des Gehirns absterben, ist je nach Größe des Schadens, das Gehirn in der Lage, möglicherweise ohne Informationsverlust weiter zu funktionieren. Die Prozesse, die sich in einem solchen Fall abspielen, lassen sich in künstlichen Netzen nicht simulieren.

### 3.1.4 Probleme bei komplexen Kategorisierungen

Bei der Kategorisierung von Bildern ist es relativ aufwendig in künstlichen neuronalen Netzen notwendige Sonderregeln zu trainieren. So muss etwa bei einem Netz zur Bildkategorisierung von Tierbildern eine Sonderregel berücksichtigt werden, bei denen ein Bild eines Hais korrekt der Kategorie Fische zugeordnet wird, das Bild eines Delfins jedoch der Kategorie Meeressäuger.

### 3.1.5 Probleme des Lernens und des Trainierens

Es wurde ausführlich dargelegt, dass künstliche neuronale Netze je nach Konfiguration der Lernfunktion in verschiedene Probleme, wie etwas das Oszillieren, das Stagnieren oder das Finden (falscher) lokaler optimaler Lösungen geraten können. Außerdem müssen komplexe Netze auf der einen Seite mit sehr vielen Daten trainiert werden, auf der anderen Seite aber auch in der Lage sein, genug Daten zu speichern. Der gesamte Trade Off (Verhältnis zwischen Einsatz von Ressourcen / Zeit und Qualität des Netzes) gelingt bisher nur für spezialisierte Anwendungsfälle.

# 4 FAZIT

Neuronale Netze werden heute in verschiedensten Szenarien eingesetzt. Sie liefern sehr gute Ergebnisse bei der Kategorisierung von Datenmustern, Bildern oder Texten, eignen sich gut zur Bilderkennung oder auch für Prognoserechnungen.

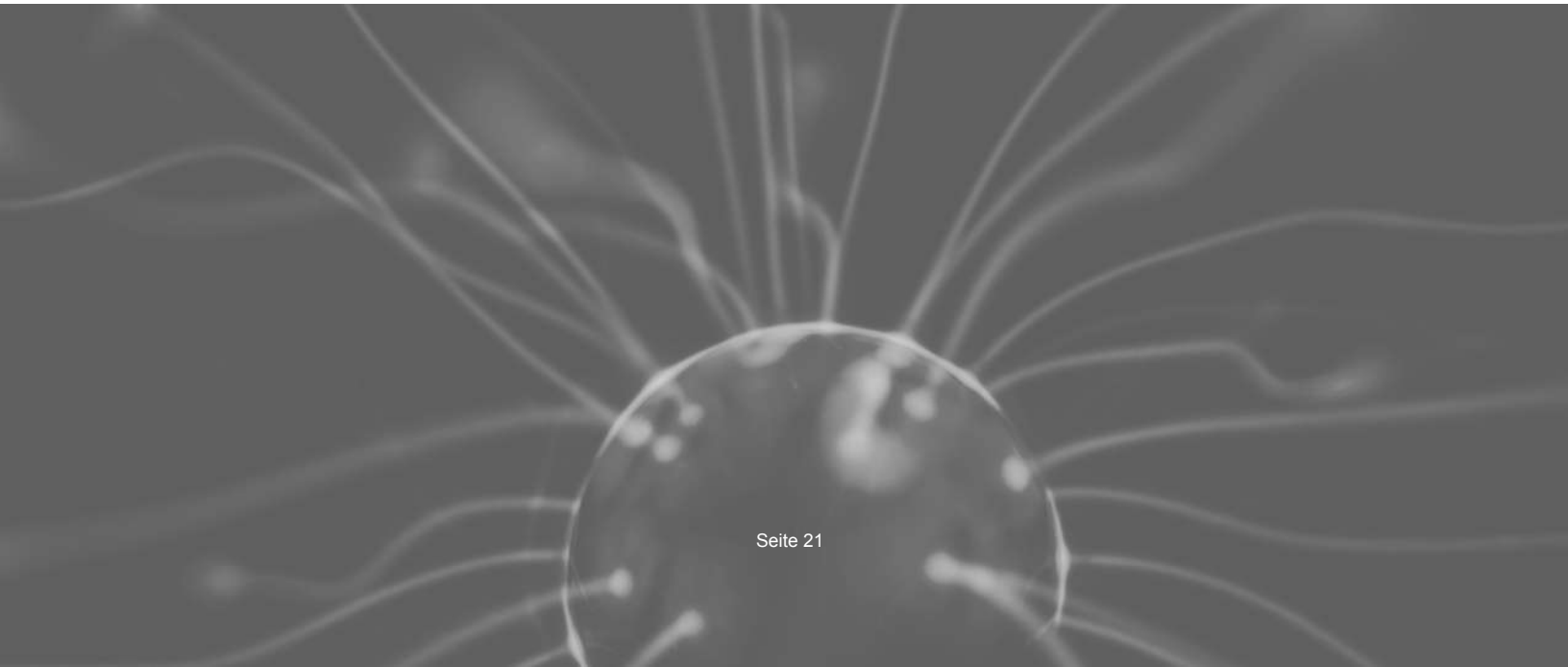
Dabei sind sie durch ihre Fehlertoleranz und ihre „Lernfähigkeit“ auf der einen Seite deutlich mächtiger und flexibler als die auf statistischen Verfahren beruhende künstliche Intelligenz.

Sie sind jedoch auf der anderen Seite sehr fordernd für die Hardware und wegen der systeminhärenten Probleme oft nur aufwendig umsetzbar. Zudem ist meistens eine große Anzahl an Trainingsdaten notwendig, um ein stabiles neuronales Netz zu entwickeln.

Es wird deutlich, dass für individuelle Probleme auch individuelle künstliche neuronale Netze aufgebaut werden müssen.

Services von der Stange zu nutzen, wie sie auf verschiedenen Cloud Plattformen existieren, machen nur Sinn, wenn das dort angebotene neuronale Netz möglichst genau für die zu lösende Aufgabe entworfen wurde.

Es ist typischerweise nicht möglich an Dingen, wie Lernparameter, Lernverfahren etc. etwas zu verändern. Normalerweise kann man diese vorgefertigten Netze nur mit neuen Daten trainieren, beispielsweise mit Bildern im Falle einer Bilderkennung.



## 4.1 PRAXISVORGEHEN

In der praktischen Anwendung ist es wichtig, zunächst zu identifizieren, ob es ein Problem zu lösen gilt, für das sich ein künstliches neuronales Netz eignet. Ist dies der Fall, beginnt die eigentliche Herausforderung. Nachfolgend gehen wir davon aus, dass das anzugehende Problem eine Zeitreihenprognose ist.

- Neben einem künstlichen neuronalen Netz bieten sich hierzu auch Verfahren des maschinellen Lernens oder mathematisch/statistische Verfahren an. Die Vor- und Nachteile müssen analysiert und abgewogen werden. Der Aufwand für die einzelnen Alternativen wird verglichen und so eine Entscheidungsgrundlage geschaffen.

- Entscheidet man sich zur Lösung des Problems durch die Nutzung eines künstlichen neuronalen Netzes, ist als erstes festzulegen, mit welchem Netztyp idealerweise gearbeitet wird und welche Lernmethode verwendet werden soll. Im Falle einer Zeitreihenprognose könnte sich ein einfaches rekurrentes neuronales Netz eignen.

- Als nächstes sollte überprüft werden, ob ein geeignetes Netz bereits schon einmal aufgebaut wurde (in Fall von Zeitreihenanalysen eher unwahrscheinlich) und nach einem neuen Training verwendet werden könnte.

- Ist kein geeignetes Netz vorhanden, muss ein neues Netz aufgebaut und trainiert werden.

- Auch während der Trainingsphase und der damit gegebenenfalls einhergehenden Modifikation des Netzes, sind Projekte in dem Umfeld beratungsintensiv. Wenn das Netz gut trainiert ist und nur noch angewandt wird (inklusive dem damit einhergehenden Training mit neuen Daten), kann es auch von Nicht-Experten genutzt werden.

## 4.2 VORSICHT STANDARD-ANGEBOTE

Wie in diesem Artikel aufgezeigt wurde, müssen künstliche neuronale Netze zumeist individuell für ein bestimmtes Praxisproblem aufgebaut und/oder trainiert werden. Es gibt nur in wenigen Fällen passende Lösungen von der Stange. Wählt man eine solche Lösung, so ist es typischerweise nicht möglich, das Netz anzupassen, die Lernregel zu bestimmen oder andere zentrale Parameter zu verändern. Dies kann durchaus zu nicht zufriedenstellenden Ergebnissen führen. Die Projekte im Bereich künstlicher neuronaler Netze sind vor allem Beratungsprojekte. Welche technischen Komponenten von welchen Herstellern zum Einsatz kommen ist sekundär.

# 5 IHR KONTAKT

Autor:
Jörg Kremer
Head of Consulting
Tel.: 089 589394 40
Joerg.Kremer@mip.de

Fürstenrieder Straße 267
81377 München

Niederlassung:
Rosensteinstraße 22
70191 Stuttgart



## LITERATURVERZEICHNIS

- http://www.neuronalesnetz.de/downloads/neuronalesnetz_de.pdf
- https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.130/Lehre/SS12/V-ENI/eni.pdf
- https://www.embedded-software-engineering.de/experten-stecken-die-grenzen-neuronaler-netze-ab-a-619763/
- Society of Neuroscience, Das Gehirn, eine kurze Zusammenfassung über das Gehirn und das Nervensystem, Universität Tübingen, 2010
- Wolfgang Ertel, Grundkurs künstliche Intelligenz, 3. Auflage, 2013, Kapitel 9
- David Kriesel, Ein kleiner Überblick über neuronale Netze, http://www.dkriesel.com/_media/science/neuronalenetze-de-zeta2-1col-dkrieselcom.pdf
- https://www.uni-heidelberg.de/presse/ruca/ruca07-1/vorbild.html