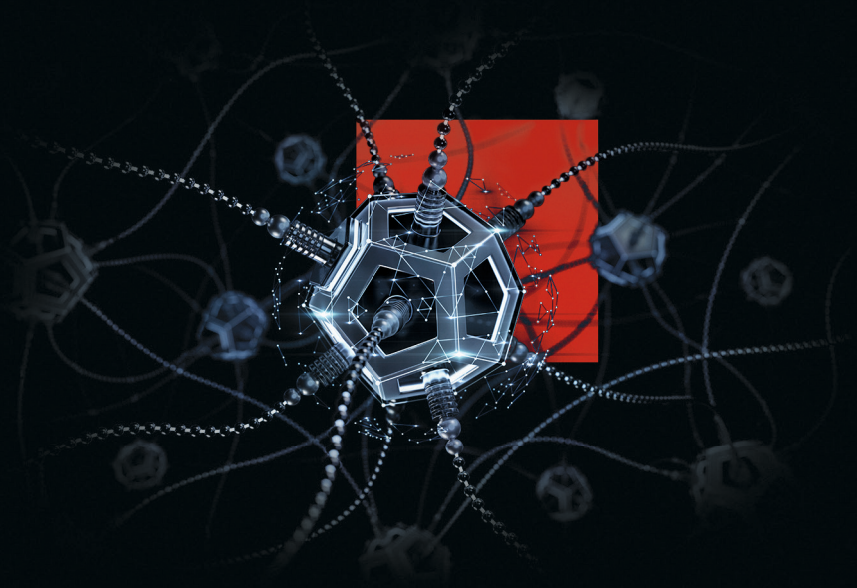
daniel BASLER



Neuronale Netze mit C# programmieren

Mit praktischen Beispielen für Machine Learning im Unternehmenseinsatz

Beispielcode unter
plus.hanser-fachbuch.de

HANSER

Basler

Neuronale Netze mit C# programmieren

Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial.

Geben Sie auf **plus.hanser-fachbuch.de** einfach diesen Code ein:

plus-5db90-her61

Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

www.hanser-fachbuch.de/newsletter

Daniel Basler

Neuronale Netze mit C# programmieren

Mit praktischen Beispielen für Machine Learning im Unternehmenseinsatz

HANSER

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

Copy editing: Walter Saumweber, Ratingen
Umschlagdesign: Marc Müller-Bremer, *www.rebranding.de*, München
Umschlagrealisation: Max Kostopoulos
Titelmotiv: © *istockphoto.com/*Artystarty
Layout: Manuela Treindl, Fürth
Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-46229-8

E-Book-ISBN: 978-3-446-46426-1

E-Pub-ISBN: 978-3-446-46635-7

Inhalt

Vorwort

Neuronale Netze sind seit einiger Zeit überall im Gespräch. Als Softwareentwickler stellt man fest, dass es zurzeit keinen anderen Bereich in der Softwareentwicklung gibt, der sich so rasant verändert und weiterentwickelt.

Eine attraktive Alternative zu Python für den Entwurf von neuronalen Netzen ist eine modulare und objektorientierte Programmiersprache wie Microsoft C#. Die Objektorientierung bietet den Vorteil, dass sich sehr gut modulare Machine-Learning-Modelle entwerfen lassen, die konfigurierbar, erweiterbar und wiederverwendbar sind.

Dieses Buch richtet sich an C#-Entwickler, die einen möglichst umfassenden Blick über neuronale Netze erlangen wollen. Es möchte Sie beim Kennenlernen, Experimentieren und Arbeiten mit neuronalen Netzen und Machine-Learning-Modellen anleiten und unterstützen. Dabei wendet es sich an im Umgang mit neuronalen Netzen unerfahrene Programmierer.

Sie sollten über Grundkenntnisse in der Programmierung mit C# verfügen, sodass Begriffe wie Variablen, Schleifen etc. Ihnen vertraut sind. Wegen des mathematischen Anteils müssen Sie sich keine Sorgen machen. Um die Buchinhalte zu verstehen, sind wirklich nur gute Kenntnisse in dem Konzept der linearen Algebra erforderlich.

Insgesamt erhebt das Buch auch keinen Anspruch auf Vollständigkeit. Es verzichtet auf tiefgehende mathematische und programmiertechnische Details, die nicht wirklich notwendig sind, um die Programmierung von neuronalen Netzen und Machine Learning zu verstehen.

Anhand von Anwendungsbeispielen lernen Sie neuronale Netze und Machine-Learning-Modelle zu entwickeln. Sie lernen auf diese Weise dynamische Datenstrukturen, Feedforward-Netze, Backpropagation-Algorithmen sowie Convolutional Neural Networks und Natural Language Processing kennen. Das Buch möchte Ihnen die Leistungsvielfalt neuronaler Netze vermitteln und Ihnen helfen diese in eigenen Programmierprojekten zu nutzen.

Den Mitarbeiterinnen und Mitarbeitern des Hanser-Verlages, besonders Frau Sylvia Hasselbach, danke ich für die Sorgfalt und Unterstützung bei der Veröffentlichung dieses Buches.

Ihnen, liebe Leserin und lieber Leser, wünsche ich viel Freude und Erfolg beim Kennenlernen und Arbeiten mit neuronalen Netzen mit C#.

Daniel Basler

Herford, April 2021

Aufbau des Buches

Die wichtigsten Begriffe und Konzepte werden gleich in den ersten beiden Kapiteln erläutert. Die weiteren Kapitel sind thematisch so aufgebaut, dass man das Buch von vorne nach hinten durcharbeiten kann.

Nachfolgend erhalten Sie einen Überblick über den Inhalt des Buches:

- **Kapitel 1** führt in das Thema künstliche Intelligenz ein. Dabei geht es um die Anwendungsgebiete, den Aufbau von neuronalen Netzen und das Neuron als Grundbaustein.
- **Kapitel 2** beschreibt die Konzepte und Methoden von Machine Learning. Es nennt wichtige Algorithmen und zeigt ein erstes einfaches neuronales Netz als Programmierbeispiel.
- **Kapitel 3** beschreibt Schritt für Schritt den Aufbau eines künstlichen neuronalen Netzes in C# und geht auf die wichtigsten Mathematik-Grundlagen ein. Anhand von Aufgabenstellungen werden mehrschichtige neuronale Netze entwickelt. Des Weiteren beschäftigen Sie sich in diesem Kapitel mit Lernalgorithmen und der Programmierung des Backpropagation-Algorithmus.
- **Kapitel 4** zeigt das Training von neuronalen Netzen. Sie lernen hier unterschiedliche Lernprozesse kennen und implementieren Trainingsmethoden in ein neuronales Netz.
- **Kapitel 5** führt Sie weiter zu den sogenannten Recurrent Neural Networks (RNN), die hauptsächlich in der Verarbeitung von Textsequenzen oder Zeitreihen eingesetzt werden. Sie lernen, wie die einzelnen Schichten in einem RNN aufgebaut sind.
- **Kapitel 6** erläutert ausführlich den Aufbau von Convolutional Neural Networks (CNN), die vornehmlich für die Verarbeitung von Bild- und Audiodateien eingesetzt werden. Ein CNN ist ein spezielles mehrschichtiges Feedforward-Netz.
- **Kapitel 7** beschäftigt sich mit den im Moment aktuellen Machine Learning Frameworks, die Sie bei der Entwicklung von Machine-Learning-Modellen in C# unterstützen können.
- **Kapitel 8** beschreibt ein weiteres nützliches ML-Framework im .NET Umfeld. Der sogenannte SciSharp-Technologie-Stack ermöglicht es, ML-Modelle für TensorFlow mit C# zu erstellen. Des Weiteren werden in diesem Kapitel die Frameworks Keras.NET und NeuralNetwork.NET vorgestellt.
- **Kapitel 9** gibt einen Einblick in Machine Learning as a Service und stellt Amazon Lex und Azure Cognitive Services an einem Anwendungsbeispiel näher vor.
- **Kapitel 10** setzt die in den vorherigen Kapiteln aufgezeigten Methoden in einzelnen Beispielanwendungen für Zeitreihenanalyse, Bildklassifikation, Objekterkennung und Natural Language Processing um.

Programmbeispiele

Um den Praxisbezug zu gewährleisten, wird gezeigt, wie die beschriebenen Themen in C#-Programme umgesetzt werden. In den meisten Fällen wird der vollständige Beispielcode abgebildet, um die Programmierung in C# zu verdeutlichen. Sie können die Buchbeispiele komplett von meinem GitHub-Repository herunterladen (*https://github.com/DanielBasler/NeuralNetwork*) oder von der Plus.Hanser-Webseite:

Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel die Code-Beispiele aus dem Buch. Geben Sie auf *plus.hanser-fachbuch.de* einfach diesen Code ein:

`plus-5db90-her61`

Zielsetzung des Buches

Mein Ziel ist es, Ihnen ein solides Fundament für das Entwerfen und Entwickeln von neuronalen Netzen und Machine-Learning-Modellen an die Hand zu geben, unterstützt von praktischen Beispielen. Des Weiteren möchte ich, dass Sie ein Gefühl für den Programmier-Aufwand von neuronalen Netzen und deren Leistung bekommen und die Vielfalt der Einsatzmöglichkeiten aber auch der Anforderungen kennenlernen.

Alle hier verwendeten Softwaretools, mit Ausnahme der KI-Services, sind kostenlos und Open Source. Die Beispiele lassen sich mit Visual Studio Community Version 2019 entwickeln und auf einem „normalen PC“ ohne besondere Hardware-Power erstellen und ausführen.

1 Künstliche Intelligenz

„Okay Google“, „Alexa, spiel bitte ‚Eight days a week‘ von den Beatles“ oder „hey Siri“... kennen Sie vermutlich alle. Die sogenannte künstliche Intelligenz, kurz nur noch als KI bezeichnet, ist schon längst in unserem Alltag angekommen und dringt in immer mehr Bereiche vor.

Das heißt, wir sind alle mittendrin, ob es um Sprachassistenten als Helfer auf dem Smartphone, im Auto oder zu Hause geht, die digitale Vernetzung in allen Lebensbereichen schreitet mit unglaublicher Geschwindigkeit voran. Mit der digitalen Transformation, durch die auch die KI allgegenwärtig wird bzw. ist, müssen sich auch Unternehmen auf eine Umstrukturierung und gegebenenfalls sogar auf eine Neuorientierung ihrer Geschäftsprozesse einstellen. Heute müssen Sie sich als Entwickler mit Begriffen wie digitaler Zwilling, Product Performance Management und digitale Fabrik auseinandersetzen. So stellt zum Beispiel das Zusammenspiel von Robotic Process Automation (RPA) und künstlicher Intelligenz ganz neue Symbiosen im Bereich der Automation dar.

In der Diskussion über KI tauchen heute sehr viele verschiedene Begriffen auf. Es ist die Rede von Machine Learning, neuronalen Netzen, Representation Learning, Natural Language Processing (NLP) oder auch Deep Learning. Selbst Begriffe wie Big Data und Data Science werden in die Runde geworfen. Sie sehen also, wer sich mit KI befasst, wird sehr schnell mit einem Begriffswirrwarr von Schlagwörtern überzogen, die durchaus für Verwirrung sorgen können.

Man kann KI als Überbegriff sehen, unter dem unterschiedliche Techniken und Bezeichnungen versammelt sind, und bevor wir uns dem Hauptthema des Buches – der Programmierung neuronale Netze – widmen, möchte ich einige Begriffe und Technologien klären, die in diesem Buch benutzt werden.

1.1 Grundlagen

Dabei ist KI keine neue Wissenschaft oder Technologie. Die Anfänge der KI-Forschung gehen bis in die 1950er-Jahre zurück, in denen Alan Turing den Aufsatz „Computing Machinery and Intelligence“ [1] vorgelegt hat. Auf Turing geht auch der nach ihm benannte Turing-Test zurück, der dazu dient, zu unterscheiden, ob eine Maschine ein gleichwertiges Denkvermögen aufweist wie ein Mensch oder nicht.

Zum ersten Mal wurde der Begriff „artifical intelligence“ von John McCarthy [2] verwendet bzw. geprägt. Laut McCarthy ist KI eine Informations- und Ingenieurswissenschaft, die dem Herstellen „intelligenter“ Maschinen und speziellen intelligenten Computerprogrammen gewidmet ist. Für McCarthy besteht der rechnerische Teil der Intelligenz in der Fähigkeit, die Ziele in der Welt zu erreichen. Das heißt für ihn, ein Computer soll so gebaut oder programmiert werden, dass er eigenständig Programme bearbeitet und löst, aus den gemachten Fehlern lernt, Entscheidungen trifft und mit seiner Umgebung kommuniziert.

Aufgrund dessen kann man auch vereinfacht sagen, KI beschäftigt sich mit der Entwicklung von Systemen, die eigenständig Probleme lösen und analog zu menschlichen Denk- und Verhaltensmustern intelligent handeln.

Dass die KI seit den letzten Jahren einen dermaßen großen Boom erlebt, hat sie folgenden Faktoren zu verdanken:

- Die Menge der zur Verfügung stehenden Daten hat extrem zugenommen. Durch Big Data [3] erweitern sich auch die Anwendungsfelder für den Einsatz von KI.
- Die Daten die über Big Data gewonnen werden, lassen sich immer preisgünstiger speichern. Die Preise für Storage sind in den vergangenen Jahren deutlich gesunken. Das macht das Speichern und Auswerten großer Datenmengen inzwischen für Unternehmen rentabel.
- Grafikprozessoren (GPU) werden günstiger und leistungsfähiger. Durch den Einsatz von GPUs können Berechnungen massiv parallelisiert und dadurch beschleunigt werden. Vor allem in diesem Bereich waren die Fortschritte in den vergangenen Jahren enorm.
- Frameworks und Cloud Services erleichtern den Einsatz. Neben der Rechenleistung und dem Speicher in der Cloud können hier auch ML-Modelle erstellt und auf verschiedenen Rechnern in der Cloud-Infrastruktur getestet werden. Hier haben Frameworks wie TensorFlow und vorgefertigte KI-Services aus der Cloud die Einstiegsschwelle deutlich gesenkt.

Durch diese Faktoren finden immer mehr KI-Lösungen ihren Weg in die Praxis. Allerdings sprechen wir hier nur über die sogenannte schwache KI, die inzwischen sehr gut erforscht und sich in zahlreichen Produkten und Diensten etabliert hat.

1.1.1 Schwache künstliche Intelligenz

Als schwache künstliche Intelligenz (engl. *weak AI* oder *narrow AI*) werden Systeme bezeichnet, die auf die Lösung konkreter Anwendungsprobleme fokussieren.

Das heißt, bei der schwachen KI geht es um die Simulation eines gewissen Bereiches des intelligenten Verhaltens mit Mitteln der Mathematik und der Informatik. Mit schwacher KI haben wir es mittlerweile im Alltag ständig zu tun. Beispiele hierfür sind unteranderen:

- Zeichen- und Texterkennung
- Bilderkennung
- Spracherkennung
- Automatische Übersetzung
- Expertensysteme
- Navigationssysteme
- Autovervollständigung und Korrekturvorschläge bei Suchvorgängen

Besondere Aufmerksamkeit erhalten hier auch die sogenannten intelligenten Chatbots, die den Kundensupport revolutionieren sollen. Aber auch die Buchhaltung wird heute schon mit schwacher KI automatisiert und optimiert, ebenso wie der Posteingang beim E-Mail-Verkehr und Aufgaben im Backoffice allgemein.

1.1.2 Starke künstliche Intelligenz

Starke künstliche Intelligenz (engl. *strong AI* oder *general AI*) hat das Ziel, eine Intelligenz zu erschaffen, die menschliches Denken, Bewusstsein und Emotionen oder die gleichen intellektuellen Fertigkeiten von Menschen erreichen oder übertreffen kann.

Starke KI müsste somit folgende Eigenschaften aufweisen:

- Logisches Denkvermögen
- Entscheidungsfähigkeit auch bei Unsicherheit
- Planungs- und Lernfähigkeit
- Fähigkeit zur Kommunikation in natürlicher Sprache
- Kombinieren aller Fähigkeiten

Allerdings ist es bis heute noch nicht gelungen, eine starke KI zu entwickeln. Auch die Diskussion, ob eine solche Entwicklung überhaupt möglich ist, hält weiterhin an. Verstärkt finden wir heute schon Ansätze in der Form der hybriden KI.

1.1.3 Hybride künstliche Intelligenz

Bei der hybriden künstlichen Intelligenz handelt es sich um eine KI-Technologie die versucht in Verbindung mit agiler Optimierung in der Unternehmensplanung oder auch in der Operationsforschung eine Reihe von Prozess- Technologien und -Modellen mit Machine Learning und Deep Learning zu vereinen.

Das heißt, man macht sich die Algorithmen aus Machine Learning zunutze um entsprechende ML-Modelle zu entwickeln, welche wiederum mit menschlichem Expertenwissen hinsichtlich Geschäftsprozessen, Verhaltensmustern und Planungszielen erweitert und trainiert werden. Somit ist es möglich, potenzielle Entscheidungsräume innerhalb dieser Modelle in kürzester Zeit durch den Experten auszuloten und zu bewerten.

Google nutzt die hybride künstliche Intelligenz schon bei der Sprachsuche. Wird von dem ML-System eine Äußerung richtig verstanden, folgt darauf das Markieren eines Treffers durch einen menschlichen Experten. Das ML-System ist so angelegt, dass es sich durch diese Markierung selbst trainieren kann, es verbessert so seine Erkennungsmöglichkeiten, je häufiger es eingesetzt wird.

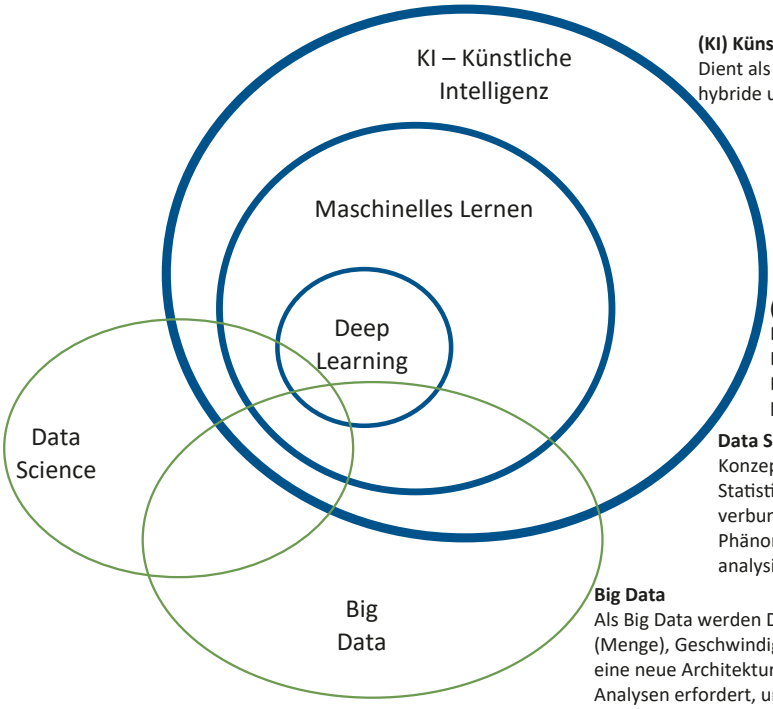
1.2 Themenfelder der künstlichen Intelligenz

In den meisten Fällen wird KI als Bezeichnung für Computersysteme benutzt, die Aufgaben abarbeiten, nachdem sie mit großen Datenmengen trainiert wurden. Die Industrie fasst den Begriff KI noch weiter, hier versteht man KI als technologische Methode, die es ermöglicht, menschliche Wahrnehmung und Handeln durch Maschinen nachzubilden. Die Einsatzgebiete der KI sind dabei sehr vielfältig. So gehört auch der erfolgreiche Einsatz in der Robotik dazu, wenn es z. B. im Maschinenbau um die Bearbeitung in Bereichen von Tausendstel-Millimetern geht.

Des Weiteren spielt auch Robotic Process Automation (RPA) eine große Rolle. Hier werden Routine-Aufgaben automatisiert, indem der Software-Roboter diese nachahmt. In einigen Fällen kann sogar der gesamte Geschäftsprozess durch einen Software-Roboter abgebildet werden.

Allerdings wird KI immer nur als reiner Oberbegriff behandelt, es ist daher wichtig zu klären, was man wirklich meint, wenn man über künstliche Intelligenz, maschinelles Lernen und Deep Learning spricht und wie sich die Begriffe zueinander verhalten.

Das Mengendiagramm in Bild 1.1 zeigt, dass künstliche Intelligenz ein Umfangreiches Set an Methoden, Verfahren und Technologien bildet. So stellt jeder Kreis im Mengendiagramm eine entsprechende KI-Technologie dar und somit auch, dass die KI zwar das maschinelle Lernen und damit auch Deep Learning einschließt, darauf aber nicht begrenzt ist. Im täglichen Umgang werden die aufgezeigten Begriffe sehr häufig synonym gebraucht.



(KI) Künstliche Intelligenz
Dient als Oberbegriff für schwache, hybride und starke KI

(ML) Maschinelles Lernen (Machine Learning)
ML gibt Systemen die Möglichkeit zu lernen, ohne explizit programmiert zu sein. ML definiert den Hauptbestandteil des KI-Systems

(DL) Deep Learning
DL ist eine Teilmenge des maschinellen Lernens. Mehrschichtige neuronale Netze, die aus riesigen Datenmengen lernen

Data Science
Konzept zur Vereinheitlichung von Statistiken, Datenanalyse und den damit verbundenen Methoden, um aktuelle Phänomene mit Daten zu verstehen und zu analysieren

Big Data
Als Big Data werden Daten bezeichnet, deren Größe (Menge), Geschwindigkeit, Vielfalt und Komplexität eine neue Architektur, Techniken, Algorithmen und Analysen erfordert, um diese zu verwalten und Werte daraus zu gewinnen

Bild 1.1 Verwandte Themenfelder in der KI

1.2.1 Machine Learning

Machine Learning, kurz als ML bzw. auch als maschinelles Lernen bezeichnet, ist eines der bekanntesten Teilgebiete der Künstlichen Intelligenz und eine Datenanalyse-Methode. ML verfolgt das Ziel, aus Daten zu lernen und möglichst treffende Vorhersagen zu generieren. Das heißt, ein künstliches System lernt aus Beispielen und kann nach Beendigung der Lernphase das Gelernte verallgemeinern. Es lernt hierbei aber die Beispiele nicht einfach auswendig, sondern es erkennt in den Lerndaten Gesetzmäßigkeiten.

Daher werden im Machine Learning Daten gesammelt und aufbereitet, um sie mithilfe von speziellen Machine-Learning-Algorithmen auf einem oder mehreren Rechnern zu trainieren. Dabei leitet der Algorithmus aus den Daten ein Modell ab, das in der Lage ist, bestimmte Eigenschaften in den Daten zu erkennen. Das Ergebnis eines ML-Algorithmus ist immer ein Machine-Learning Model. Fast alle Modelle, die ML-Algorithmen aus Beispielen erzeugen, sind letztlich statistische Modelle, weshalb Statistik ein Fundament für die Theorie des maschinellen Lernens bildet.

Für den Softwarebereich hat Thomas Mitchell es wie folgt beschrieben: Ein Computerprogramm lernt beim Lösen einer bestimmten Klasse von Aufgaben (T), wenn seine messbare Leistung (P) sich mit der Erfahrung (E) im Lauf der Zeit erhöht [4]. Diese Logik zu finden wird maschinelles Lernen genannt. Ausführlicher wird ML im nachfolgenden Kapitel 2 erläutert.

1.2.2 Deep Learning

Deep Learning, kurz als DL oder auch als tiefgehendes Lernen bezeichnet, ist eine Teildisziplin des maschinellen Lernens und wird derzeit am häufigsten im Zusammenhang mit dem Begriff KI verwendet. Die Grundlage von DL sind künstliche neuronale Netze und somit eine spezielle Methode der Informationsverarbeitung. DL wird besonders auf große Datenmengen angewandt, die mithilfe von neuronalen Netzen analysiert werden. Über diese Technologie schafft es das System, Strukturen zu erkennen, Informationen zu sortieren und zu evaluieren. Dabei vollzieht sich ein permanenter Prozess, das Gelernte wird immer wieder mit neuen Inhalten verknüpft und erweitert. Dies führt mit hoher Wahrscheinlichkeit dazu, dass ein richtiges Ergebnis erkannt und ausgegeben wird. DL kommt zum Einsatz, wenn andere ML-Verfahren an ihre Grenzen stoßen.

Auch der Prozess und die benötigten Ressourcen unterscheiden sich im Vergleich zum Machine Learning. Bei ML gibt der Mensch Testdaten mit korrekten Antworten vor, sodass der Algorithmus basierend auf diesen manuell klassifizierten Daten lernt. Beim Deep Learning fällt die Vorgabe der Lösung weg, stattdessen wird die Zuordnung und Klassifizierung der Daten automatisiert. Mehr Informationen zu DL finden Sie in Abschnitt 2.9.

1.2.3 Cognitive Computing

Cognitive Computing stellt einen Teilbereich der KI dar, der eine natürliche, möglichst menschliche Interaktion mit Maschinen anstrebt. Hierfür werden bei Cognitive Computing Technologien der KI genutzt, um menschliche Denkprozesse zu simulieren. Dabei geht es darum, auf Basis von Erfahrungen eigene Lösungen und Strategien zu entwickeln.

Mithilfe von KI und Cognitive Computing soll ein System entstehen, das Bilder und Sprache interpretiert und auch noch schlüssig antworten kann. Die wichtigste Voraussetzung für das Cognitive Computing ist demnach die Fähigkeit, aus den gemachten Erfahrungen selbstständig zu lernen. Gleichzeitig soll das System die eigenen Lösungsansätze ständig hinterfragen.

Ein wesentliches Merkmal des Cognitive Computing ist, das riesige Datenmengen unterschiedlichster Art gespeichert und binnen kürzester Zeit verarbeitet werden müssen. Da die Daten in der Regel in unstrukturierter Form vorliegen, lassen sich herkömmliche relationale arbeitende Systeme nicht für das Cognitive Computing effizient einsetzen. Daher kommen hier Techniken aus dem Big-Data-Umfeld in Verbindung mit Data Science zum Einsatz.

Das bekannteste Beispiel für Cognitive Computing ist wohl der Sieg 2011 von IBM Watson bei Jeopardy. Hierbei war das System in der Lage, Fragen selbstständig und sinnvoll zu beantworten.

1.2.4 Big Data und Data Science

Big Data ist ein allgemeiner Begriff, der für die Beschreibung umfangreicher Mengen unstrukturierter und semi-strukturierte Daten verwendet wird. Demzufolge bezeichnet Big Data die immer rasanter wachsenden Datenmengen zum Beispiel aus den sozialen Netzwerken. Die wichtigsten Datenquellen für Big Data sind:

- Mobile Internetnutzung
- Social Media
- Geo-Tracking
- Cloud Computing
- Vitaldaten-Messung
- Media-Streaming

Der Begriff Big Data meint aber nicht nur die Daten selbst, sondern auch deren Analyse und Nutzung. Hierfür wird Big Data auch mit Cloud Computing und Machine Learning in Verbindung gebracht, um die Echtzeit-Analyse von großen Datenmengen zu bewerkstelligen. Da die gesammelten Datenmengen allerdings komplex, schnelllebig und unstrukturiert sind, stellt sich die Aufbereitung und Analyse sehr häufig als schwierig und aufwendig dar. An dieser Stelle kommt dann Data Science zum Einsatz.

Bei Data Science handelt es sich um eine Wissenschaft, die sich mit der Extraktion von Wissen aus großen Datenmengen (auch Big Data) beschäftigt. Mit verschiedenen Methoden aus der Datenanalyse und Visualisierung werden die gesammelten Daten erfasst und die benötigten relevanten Informationen extrahiert. Gegenwärtig werden im Data Science Techniken und Methoden aus der Mathematik, der Statistik, der Stochastik und der Informatik angewendet.

1.2.5 Predictive Analytics

Predictive Analytics stellt eine Analysemethode dar, die sowohl neue als auch historische Daten zur Vorhersage von Aktivitäten, Verfahren und Trends verwendet. Man nutzt hier Machine Learning um Vorhersagemodelle zu erstellen, die einen numerischen Wert für die Wahrscheinlichkeit des Eintretens eines bestimmten Ereignisses berechnen. Bei Predictive Analytics kommen folgende Methoden zum Einsatz:

- **Logistische Regression:** Eine Analysemethode, die zur Vorhersage eines Datenwertes auf Grundlage früherer Beobachtungen eines Datensatzes verwendet wird.
- **Zeitreihenanalyse:** Eine Darstellung von Datenpunkten in aufeinanderfolgenden Zeitintervallen.
- **Entscheidungsbaum:** Ein Diagramm, das mithilfe einer Verzweigungsmethode jedes mögliche Ergebnis einer Entscheidung darstellt.

Durch den Einsatz von ML hat Predictive Analytics in den letzten Jahren große Fortschritte gemacht und viel neue Aufmerksamkeit erfahren.

1.2.6 Natural Language Processing

Natural Language Processing, kurz NLP, beschreibt Techniken und Methoden zur maschinellen Verarbeitung natürlicher Sprache. Das heißt, NLP versucht, natürliche Sprache zu erfassen und mithilfe von Regeln und maschinengestütztem Lernen zu verarbeiten. NLP kommt als Teilbereich in folgenden Anwendungen zum Einsatz:

- Spracherkennung
- Segmentierung zuvor erfasster Sprache in einzelne Wörter und Sätze
- Erkennen der Grundformen der Wörter und Erfassung grammatischer Informationen
- Extraktion der Bedeutung von Sätzen und Satzteilen

Ein praktisches Beispiel für den Einsatz von NLP finden Sie in Kapitel 8.

Bild 1.2 zeigt noch einmal alle Teilbereiche der KI in Bezug auf Handeln, Wahrnehmung und Lernen.

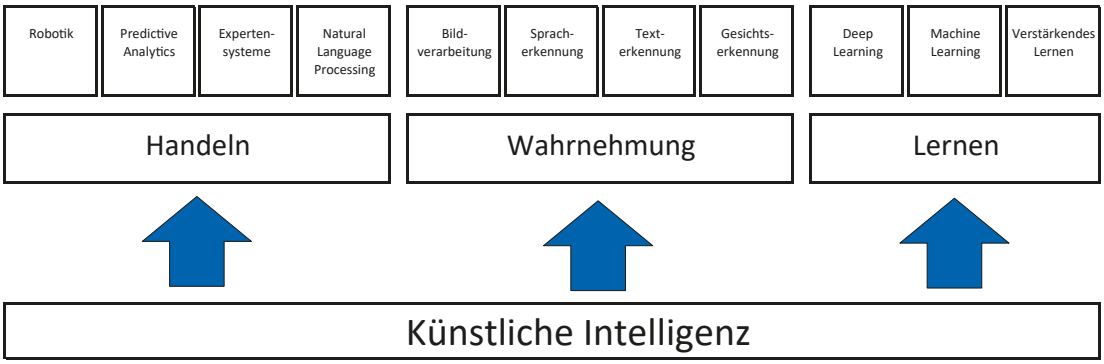


Bild 1.2 Die Teilbereiche Handeln, Wahrnehmen und Lernen in der KI

1.3 KI-Service-Plattformen

Alle großen Internetkonzerne bieten inzwischen eine Zusammenstellung verschiedener APIs (Application Programming Interface), SDKs (Software Development Kit) und Services (Dienste) an, um Ihnen als Entwickler dabei zu helfen, ihre Programme intelligenter und benutzerfreundlicher zu machen.

Über diese KI-Plattformen können Sie auf ausgereifte und schon fertige Services für Lösungen auf Basis von Machine-Learning-Modellen oder Services für Sprach- und Text-Erkennung zurückgreifen. Neben den verschiedenen KI-Services und -Tools bieten diese Plattformen durch die Nutzung in der Cloud auch eine gute Unterstützung bei der Infrastruktur, der Sicherheit, der Verfügbarkeit und natürlich auch den Vorteil der zur Verfügung stehenden Rechenleistung.

Neben diesen KI-Services veröffentlichen die Anbieter wie Google, Amazon, Microsoft und IBM auch den entsprechenden Code ihrer Open Source ML- und DL-Frameworks. So können Sie als Entwickler an der direkten Weiterentwicklung der Frameworks teilnehmen und sich in der jeweiligen Community austauschen.

1.3.1 Amazon

Amazon betreibt mit seinem Web Service (AWS) den größten Cloud Computing Dienst weltweit. Neben den bekannten Infrastruktur-Diensten bietet Amazon auch Services für künstliche Intelligenz an.

Mit Deep Scalable Sparse Tensor Network Engine (DSSTNE) hat Amazon seine Programmbibliothek für maschinelles Lernen als Open Source zu Verfügung gestellt. Der Schwerpunkt von DSSTNE liegt auf Verfahren, die bei der Entwicklung von ML-Modellen mit nur wenigen eigenen Trainingsdaten auskommen, da das erstellte ML-Modell über einen Cloud Service verteilt trainiert werden kann. Die weiteren KI Services von Amazon bieten die folgenden Möglichkeiten.

Amazon SageMaker

Der Amazon SageMaker ist ein von Amazon vollständig verwalteter Service, der jedem Entwickler die Möglichkeit bietet, schnell Modelle für Machine Learning zu erstellen und diese Modelle zu trainieren, um sie dann für eine Anwendung bereitzustellen. Über dieses Tool wird der gesamte ML-Zyklus von der Datenaufbereitung bis zur Bereitstellung der ML-Modelle abgedeckt.

Amazon Lex

Amazon Lex ist ein Service zur Erstellung einer Schnittstelle für Sprache und Text. Amazon Lex bietet automatische Spracherkennung zur Umwandlung von Sprache in Text und eine Erkennung der Textabsicht, um eine realistische Gesprächsinteraktion in der Anwendung zu gewährleisten.

Amazon Lex bildet den Kern von Amazon Alexa und stellt somit jedem Entwickler diese Technologie zur Verfügung. Somit soll es möglich sein, schnell und einfach komplexe Chatbots zu entwickeln.

Amazon Translate

Amazon Translate ist ein neuraler, maschineller Übersetzungsservice, der kostengünstig Übersetzungen liefert. Hierbei bedeutet neuronale maschinelle Übersetzung, dass Amazon Deep-Learning-Modelle verwendet, um im Vergleich zu herkömmlichen regelbasierten Übersetzungsalgorithmen eine genauere und natürlicher klingende Übersetzung zu liefern.

Amazon Polly

Amazon Polly ist ein Service, der Text in realistische Sprachausgabe verwandelt. Hier kommt über den Text-to-Speech-Service Deep Learning zum Einsatz, um natürlich klingende menschliche Sprache zu synthetisieren. Somit soll es möglich sein, neue sprachfähige Produkte zu entwickeln.

Amazon Rekognition

Hinter Amazon Rekognition steht ein hochgradig skalierbares Deep-Learning-Modell, das es ermöglicht Objekte, Menschen, Text, Szenen und Aktivitäten in Bildern und Videos zu identifizieren. Der Service bietet weiterhin auch eine sehr genaue Gesichtsanalyse- und Gesichtsfunktionen an, mit denen Sie Gesichter analysieren und vergleichen können. Sie benötigen für diesen Service keinerlei Machine-Learning-Kenntnisse.

Amazon RoboMaker

Mit dem Amazon RoboMaker steht eine umfassende Cloud-Lösung für die Entwicklung von Robotersystemen zur Verfügung. RoboMaker ermöglicht das Simulieren, Testen und das sichere Bereitstellen von Roboteranwendungen im großen Umfang.

Apache MXNet

Amazon bietet einige optimierte Services für das Deep-Learning-Framework Apache MXNet an. Bei MXNet handelt es sich um ein schnelles und skalierbares Schulungs- und Interferenz-Framework mit einer kompakten und entwicklerfreundlichen API für Machine Learning. Apache MXNet bietet des Weiteren interessante Einblicke in den Aufbau tiefgreifender Deep-Learning-Systeme für Entwickler.

1.3.2 Google

Google gehört heute zu den führenden Unternehmen im Bereich KI. Neben der Forschung betreibt Google auch eigene KI-Systeme auf der Google Cloud Plattform, die Ihnen als Entwickler zur Verfügung stehen. Die bekannteste Programmbibliothek für ML von Google ist TensorFlow.

TensorFlow

TensorFlow ist Googles plattformunabhängiges Open-Source-Framework für maschinelles Lernen. Der Schwerpunkt des Systems liegt in der Verarbeitung von Sprache und Bildern. In der Forschung und im Produktivbetrieb wird das Framework derzeit in verschiedenen Google-Produkten eingesetzt. Hierzu zählen die Google-Spracherkennung, Gmail, Google Fotos, Google Maps und die Google-Suche. Mehr Informationen zu TensorFlow finden Sie in Abschnitt 7.2 „TensorFlow“.

Google Vision API

Die Google Vision API bietet über eine REST API-Schnittstelle leistungsstarke vorabtrainierte Modelle für Machine Learning. Sie können über die Schnittstelle Bilder mit Labels versehen und diese Bilder dann in kurzer Zeit Millionen von vordefinierten Kategorien zuordnen. Außerdem ist es möglich, Objekte und Gesichter zu erkennen sowie gedruckten und handgeschriebenen Text zu lesen.

Cloud AI Platform

Die Cloud AI Platform ermöglicht es Entwicklern, den vollständigen Zyklus der Machine-Learning-Entwicklung in einem Projekt zu managen. Sie unterstützt mit einem integrierten Toolset alle Prozessketten von der Erstellung der Datenbasis bis zu Ausführung und Bereitstellung des ML-Projekts.

AutoML Video Intelligence

AutoML Video Intelligence stellt eine grafische Benutzeroberfläche zur Verfügung, mit der Sie eigene ML-Modelle zur Klassifizierung und Nachverfolgung von Objekten in Videos trainieren können.

Translation API Basic

Die Google Translation API übersetzt Texte aus Websites und Anwendungen in mehr als 100 Sprachen. Translation API verwendet hierfür vortrainierte neuronale Machine-Learning-Übersetzungen und liefert damit schnelle, dynamische Ergebnisse.

AutoML Natural Language

AutoML Natural Language bietet eine Benutzeroberfläche, um eigene ML-Modelle zum Klassifizieren, Extrahieren und Erkennen von Stimmungen trainieren zu können. Sie können Ihre Trainingsdaten direkt über die Oberfläche hochladen und anschließend einfach Ihr erstelltes ML-Modell testen.

Vortrainierte Modelle

Für viele gängige Anwendungsfälle im Bereich Sprachen- und Textanalyse sowie in der Bildverarbeitung stehen Ihnen von Google schon vortrainierte ML-Modelle als API-Schnittstelle für die Nutzung in eigenen Anwendungen zur Verfügung. Da Google laufend weitere Fortschritte in der KI-Forschung erzielt, haben Sie immer Zugriff auf die neusten vollständig trainierten ML-Modelle.

1.3.3 Microsoft Cognitive Services

Neben dem bekannten Open-Source-Framework ML.NET bietet das Softwareunternehmen Microsoft weitere APIs, SDKs und Dienste rund um das Thema KI an. Diese werden als Cognitive Services bezeichnet und sind über die Azure-Plattform erreichbar.

Gesichtserkennungs-API

Die Gesichtserkennungs-API nutzt Algorithmen, um menschliche Gesichter auf Bildern zu erkennen. Diese API bietet Erweiterungen wie Gesichtserkennung, Gesichtsabgleich und Gesichtsgruppierung, um Gesichter basierend auf ihrer Ähnlichkeit in Gruppen einzuordnen.

Video API

Die Video API erlaubt, das Erkennen von gesprochenen Wörtern, geschriebenem Text, Gesichtern, Emotionen, Marken oder Szenen automatisch aus Video- oder Audiodateien zu extrahieren. Über eine einfache Benutzeroberfläche kann die API-Schnittstelle konfiguriert werden.

Speech-Service

Der Speech-Service dient zur Vereinheitlichung von Spracherkennung, Sprachsynthese und Sprachübersetzung. Bis Anfang 2020 war dieser Bereich noch in einzelne API-Schnittstellen aufgeteilt. Auch Sprachassistenten können mit diesem umfangreichen Service entwickelt werden.

Freihanderkennungs-API

Diese KI-API unterstützt Sie bei der Erkennung digitaler Freihandinhalte wie handschriftlicher Texte, Formen und Layouts geschriebener Dokumente. Es handelt sich hierbei aber nicht um eine optische Zeichenerkennung wie bei OCR (Optical Character Recognition), sondern um eine Analyse der Daten anhand der Bewegung über Eingabetools wie digitaler Stifte oder Fingerzeichen. Der erkannte Inhalt wird dann über die API-Schnittstelle als JSON-Antwort zurückgeliefert.

Cognitive Toolkit

Auch das Cognitive Toolkit ist ein kostenloses Open-Source-Deep-Learning-Framework, mit dem künstliche neuronale Netzen trainiert werden können. Das Toolkit beschreibt hierbei neuronale Netze als eine Reihe von Rechenschritten über einen gerichteten Graphen. Sie können mit dem Cognitive Toolkit beliebige ML-Modelltypen einfach realisieren und diese auch kombinieren. Das Toolkit ist mit vielen verschiedenen Programmiersprachen nutzbar und ermöglicht auch einen zuverlässigen Betrieb mit großen Datenmengen.

1.3.4 IBM

Auch IBM bietet über seine Watson Data Platform verschiedene Dienste für KI-Aktivitäten an. Diese werden als Watson Developer Cloud Services zur Verfügung gestellt.

Watson Natural Language Classifier

Der Natural Language Classifier unterstützt Sie, die Sprache kurzer Texte zu verstehen. Ein Klassifikationsmerkmal, der Classifier, lernt anhand Ihrer Beispieldaten und kann anschließend Informationen zu Texten zurückgeben, anhand derer das ML-Modell nicht trainiert wurde.

Watson Natural Language Understanding

Der Natural Language Understanding-Service dient zur Analyse von sprachlichen Merkmalen in Texteingaben. Es wird versucht, Kategorien, Konzepte oder auch Emotionen aus dem Text zu extrahieren.

Watson Visual Recognition

Auch bei IBMs Visual Recognition werden Deep-Learning-Algorithmen verwendet, um in Bildern Objekte und Szenen erkennen zu können.

Vortrainierte Modelle

Wie Google, stellt auch IBM vortrainierte Modelle für Machine Learning zur Verfügung. Hier haben die ML-Modelle aber einen eindeutigen Bezug auf entsprechende Anwendungsfälle. So gibt es ML-Modelle für die Fertigung, für Versicherungen, den Einzelhandel und den Schulungsbereich.

Watson Assistant

Bei IBMs Watson Assistant handelt es sich um eine sogenannte Konversations-KI-Plattform für die Erstellung von virtuellen Assistenten (Chatbots). Hier ist der Assistent ein kognitiver Bot, den Sie an Ihre Anforderungen anpassen und bereitstellen können, um Ihren Anwendern kontextbezogene Unterstützung anzubieten. Watson Assistant erstellt dynamisch ein Modell für Machine Learning, das auf Ihre definierten aber auch auf ähnliche Benutzeranfragen abgestimmt ist.

Wie Sie sehen, bieten die Konzerne schon eine beachtliche Anzahl von nützlichen Schnittstellen und KI-Diensten an, die auch entsprechend weiterentwickelt werden, und es kommen ständig neue APIs oder KI-Services hinzu.

Durch das Zur-Verfügung-stellen einer entsprechenden API oder auch eines Service ist es für Sie sehr viel einfacher, auf schon vorhandene Funktionen und Modelle zurückzugreifen. Profitieren Sie hier einfach von den Weiterentwicklungen und dem Austausch in der jeweiligen Community, um Ihre Programmentwicklung noch effektiver zu gestalten.

Dieses Buch möchte Sie darin unterstützen, selber Modelle für entsprechende Frameworks zu erstellen oder schon vorhandene zu modifizieren. Nutzen Sie hierfür aber auch die Vorteile der großen Lernplattformen von Microsoft, Google und Co.

1.4 Künstliche neuronale Netze

Unter den lernenden Algorithmen sind die künstlichen neuronalen Netze die Schlüsseltechnologie für Machine und Deep Learning. Um aber die Möglichkeit von ML und DL auszukosten und richtig zu nutzen, ist es hilfreich, wenn man weiß, wie neuronale Netze tatsächlich funktionieren.

Die Zusammenhänge im Detail versteht man am besten, wenn man ein eigenes neuronales Netz programmiert. Das so erworbene Wissen lässt sich dann bei der Verwendung eines KI-Services oder eines entsprechenden Frameworks sicher und effektiv nutzen.

1.4.1 Funktionsweise

Künstliche neuronale Netze (engl. *artifical neural network* – ANN), kurz als KNN bezeichnet, bestehen aus Knoten, die Sie als Analogie zu menschlichen Nervenzellen sehen können. Diese Knoten stellen die einzelnen Neuronen dar, die auch gerne als Units oder Einheiten bezeichnet werden.

Sie dienen dazu, Informationen aus der Umwelt oder von anderen Neuronen aufzunehmen und diese wiederum an andere Neuronen in modifizierter Form weiterzuleiten. Diese Neuronen sind in Schichten, den sogenannten Layers angeordnet, in denen sich mehrere Neuronen befinden. Jedes Neuron einer Schicht (Layer) ist mit allen Neuronen der nächsten Schicht verbunden. Somit besteht ein KNN aus Neuronen mit gerichteten und gewichteten Verbindungen. Bild 1.3 zeigt die schematische Darstellung eines neuronalen Netzes.

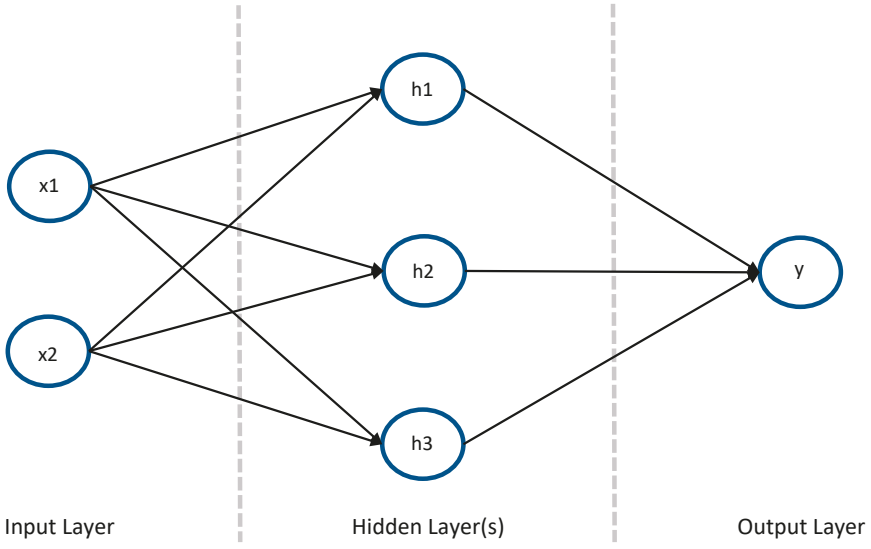


Bild 1.3 Schematische Darstellung eines künstlichen neuronalen Netzes (KNN)

Die Schichten eines neuronalen Netzes werden im Regelfall in folgende Kategorien unterteilt:

- **Input Layer:** Die Eingabeschicht repräsentiert die Eingabedaten in das neuronale Netz, also die gewünschten Daten, auf die eine Klassifikation angewendet wird.
- **Hidden Layer:** Die verborgene Zwischenschicht bildet sich wie folgt: Zwischen den Neuronen der Eingangsschicht und deren Ausgangsschicht befinden sich eine oder auch mehrere versteckte Schichten. Gibt es mehr als nur eine versteckte Schicht, handelt es sich um ein sogenanntes tiefes neuronales Netz, auch bekannt unter dem Begriff Deep Neural Network.

  Theoretisch ist die Anzahl der möglichen verborgenen Schichten in einem künstlichen neuronalen Netz unbegrenzt. In der Praxis bewirkt jede hinzukommende verborgene Schicht jedoch auch einen Anstieg der benötigten Rechenleistung.
- **Output Layer:** Die Ausgangsschicht ist mit den Neuronen der versteckten Schicht verbunden und repräsentiert das Ergebnis der Informationsverarbeitung.

1.4.2 Netztypen

Je nachdem, welche Aufgabe ein neuronales Netz erledigen soll, ob eine einfache Klassifikation, Bildanalysen oder Übersetzungen, werden neuronale Netze nach unterschiedlichen Gesichtspunkten aufgebaut. Dabei unterscheidet man zwischen grundsätzlich verschiedenen Strukturen, die als Topologien bezeichnet werden. Die Topologie eines Netzes, also die Zuordnung der Verbindungen zwischen den Neuronen, muss abhängig von der Aufgabenstellung gewählt werden. Die meisten heute im Einsatz befindlichen neuronalen Netze kombinieren verschiedene Formen der Netz-Architektur. Die folgende Aufzählung zeigt die wichtigsten Netztypen:

- *Single-Layer Perceptron (Perzeptron)* ist die einfachste Form eines neuronalen Netzes. Es besteht nur aus einem einzigen Neuron. Mehr dazu erfahren Sie in Abschnitt 2.8.
- *Feedforward Neural Network (vorwärtsgekoppelte Netze)* sind Netze, in denen die Verbindungen nur in eine Richtung gehen, von der Eingabe zur Ausgabe.
- *Recurrent Neural Network (rekurrente Netze)* sind Netze, in denen es auch Verbindungen in Rückwärtsrichtung gibt, sodass Rückkopplungen entstehen können (siehe auch Kapitel 5).
- *Convolutional Neural Network* sind Netze, die eine Sonderform des künstlichen neuronalen Netzes darstellen. Sie werden insbesondere im Bereich der Bild- und Audioverarbeitung eingesetzt. Dieses Netz lernen Sie in Kapitel 6 genauer kennen.
- *Asynchrone Netze* sind Netze, in denen die Neuronen nicht alle auf einmal (synchron), sondern nacheinander in zufälliger Reihenfolge aktiviert werden.
- *Symmetrische Netze* sind Netze, die von jedem Betrachtungspunkt eines Neurons gleich aussehen. Das heißt, die Neuronen in einem symmetrischen Netz verhalten sich identisch. Die Symmetrie erleichtert die Programmierung, da keine besonderen Fälle bei der Verbindung der Neuronen zu beachten sind.
- *Selbstassoziative Netze* sind Netze, in denen Eingabe- und Ausgabe-Neuronen übereinstimmen.
- *Zyklische Netze* sind Netze, in denen sich einige Neuronen gegenseitig stimulieren. Auch hier gibt es die Möglichkeit einer Rückkopplung.

Eine Betrachtung und Beschreibung aller möglichen KNNs würde den Rahmen dieses Buches sprengen. Daher beschränken wir uns hier auf die Beschreibung der grundlegenden Strukturen von Feedforward Neural Networks, Recurrent Neural Networks und den Convolutional Neural Networks.

Feedforward Neural Network

Ein Feedforward Neural Network, kurz FNN, besitzt beliebig viele Layer (Schichten). Die Daten werden hierbei von einem Layer über verschiedene gewichtete Verbindungen zum nächsten Layer weitergeleitet. Der Informationsfluss findet ausschließlich vorwärtsgerichtet von den Input Layer über die Hidden Layer zu den Output Layer statt. Es existieren somit keine Verbindungen, welche zurückführen. Des Weiteren handelt es sich bei einem FNN immer um ein vollständig verbundenes (fully connected) neuronales Netz, da immer sämtliche Neuronen einer Schicht mit allen der darauffolgenden verbunden sind. Das FNN wird sehr häufig für die Ermittlung einer Prognose verwendet. Eine praktische Programmieraufgabe mit diesem Netz finden Sie in Abschnitt 3.4.

Recurrent Neural Network

Ein Recurrent Neural Network, kurz RNN, kann, im Gegensatz zu einem FNN, Informationen wieder in vorherige Layers (Schichten) leiten. Somit existieren in RNNs Verbindungen, bei denen Informationen bestimmte Neuronen-Verbindungen des neuronalen Netzes rückwärts und anschließend erneut vorwärts durchlaufen können. In den meisten Fällen werden RNN in der Spracherkennung, Übersetzungen und Handschrifterkennung eingesetzt.

Hierbei sind folgende Varianten möglich:

- **Direkte Rückkopplung:** Ein Neuron nutzt seinen Output als erneuten Input.
- **Indirekte Rückkopplung:** Der Output eines Neurons wird als Input eines Neurons in einer vorgelagerten Schicht (Layer) verwendet.
- **Seitliche Rückkopplung:** Der Output eines Neurons wird als Input eines Neurons in derselben Schicht (Layer) verwendet.
- **Vollständige Verbindung:** Der Output eines Neurons wird von jedem anderen Neuron im Netz als zusätzlicher Input verwendet.

Dieses Netz lernen Sie ausführlich in Kapitel 5, „Recurrent Neural Network“ kennen.

Convolution Neural Network

Ein Convolutional Neural Network, kurz CNN, ist ein spezielles neuronales Netz. Es besitzt mehrere Faltungsschichten und ist dadurch in der Lage, Input in Form einer Matrix zu verarbeiten. Die Architektur eines CNN unterscheidet sich deutlich von der eines klassischen FNN.

Die Topologie gestaltet sich bei CNNs aus den Convolutional Layers und Pooling Layers. Hierdurch ist eine Untersuchung des Inputs aus verschiedenen Perspektiven möglich und es eignet sich daher besonders für das maschinelle Lernen im Bereich Bild- und Spracherkennung. Mehr zu CNNs erfahren Sie in Kapitel 6.

1.4.3 Anwendungsbereiche

Die Bandbreite der Anwendungsbereiche für künstliche neuronale Netze hat entsprechend der Weiterentwicklung von Hard- und Software zugenommen. KNNs werden heute in den unterschiedlichsten Bereichen zur Informationsverarbeitung eingesetzt. Typische bekannte Anwendungen sind:

- Bilderkennung
- Spracherkennung
- Mustererkennung
- Schrifterkennung
- Frühwarnsysteme
- Simulation komplexer Systeme

Vor allem Bild-, Muster- und Spracherkennung werden in Industrieanwendungen aus den Bereichen Qualitätssicherung, Konstruktion und Entwicklung erfolgreich eingesetzt. In der Automatisierung und Produktion sorgen sie für eine Optimierung der Prozesse.

1.5 Grundbaustein Neuron

Ein künstliches neuronales Netz ist ein mathematisches Modell, das durch das Vorbild vernetzter biologischer Nervenzellen, den sogenannten Neuronen, inspiriert worden ist. Somit stellt das Neuron auch in dem mathematischen Modell den Grundbaustein dar. Aus mathematischer Sicht besteht das Neuron im Wesentlichen aus einer gewichteten Verknüpfung der Eingänge, einer Aktivierungsberechnung und einer Ausgabefunktion. Das künstliche Neuron im Modell stellt einfach einen Prozessor mit einer bestimmten Anzahl von Ein- und Ausgaben dar. Hierbei unterscheidet man bei der Aktivität eines Neurons zwischen dem Aktiv-Zustand, dem Senden eines Signals, und dem Inaktiv- oder Ruhezustand.

Jede Verbindung der Neuronen kennzeichnet den Datenfluss von einem Neuron zu einem anderen. Bild 1.4 zeigt die Notwendigen Elemente in einem Neuron. Die übermittelten Daten sind einfach Zahlwerte, die unterschiedlich gewichtet sind. Das Neuron verknüpft die eingehenden Werte $X_1, ..., X_n$ und die Gewichtungen $W_1, ..., W_n$ mathematisch.

Das heißt, die Eingabesignale $X_1, ..., X_n$ werden mit den Gewichtsfaktoren $W_1, ..., W_n$ multipliziert. Alle Eingabeinformationen mit den Gewichten der Verbindungen sind somit zu einer einzigen Netzeingabe verknüpft. Die Netzeingabe stellt somit die Summe aller Informationen dar, die aus dem Netz an das Neuron weitergegeben werden. Dementsprechend erhält man über die verwendete Propagierungsfunktion $p(X_1, \ldots, X_n)$ das zusammengefasste Gesamt- bzw. das Netto-Eingabesignal.

Wenn die Summe über das Produkt aller Eingaben mit den entsprechenden Gewichten größer als der Schwellenwert für die Aktivierungsfunktion ist, wird die Ausgabe auf aktiv gesetzt, sonst ist die Ausgabe passiv. Der Schwellenwert kann auch durch einen weiteren Eingang, dem sogenannten *Bias bn*, bestimmt werden.

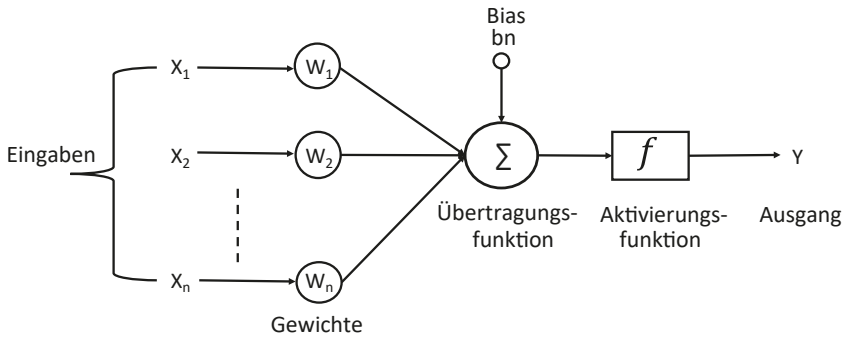


Bild 1.4 Schema des künstlichen Neurons

Bei dem Bias handelt es sich um einen Initialwert. Das Resultat entscheidet darüber, ob das Neuron aktiviert ist und somit zum Ergebnis beitragen soll. Es zählt hierbei die einfache Regel, dass ein Wert $v \leq 0$ einer Nicht-Aktivierung und ein Wert $v > 0$ einer Aktivierung entspricht.

Somit kann man das Neuron in folgende Teile untergliedern:

- Eine oder mehrere eingehende Verbindungen X_1 bis X_n, welche numerische Ausgangssignale von anderen Neuronen empfangen. Dabei wird jeder Verbindung eine Gewichtung W_1 bis W_n zugewiesen, die verwendet wird, um das jeweilige gesendete Signal zu verstärken bzw. zu drosseln.
- Eine Aktivierungsfunktion, die den numerischen Wert des Ergebnis-Signals bestimmt, welches das Neuron aussendet.
- Eine oder eventuell mehrere Ausgangsverbindungen, die das Ergebnis-Signal zu den anderen Neuronen übertragen.

Bias

Je nachdem, welches Problem mit dem neuronalen Netz gelöst werden soll, kann es sein, dass eine bestimmte Aktivierungsfunktion nicht optimal ist. Hierfür gibt es den Bias-Wert. Dieser Wert ist ein zusätzliches Gewicht, das bestimmt, wie groß eine gewichtete Summe sein muss, um den Schwellenwert der Aktivierungsfunktion zu überschreiten. Der Bias-Wert verschiebt somit das Grundniveau der Aktivierungsfunktion. In der Praxis spricht man beim Bias auch von der systematischen Verzerrung, wobei dieser Wert für die Berechnung einfach als weiteres Gewicht behandelt wird.

1.5.1 Aktivierungsfunktion

Die Aktivierungsfunktion, auch als Transfer- oder Übertragungsfunktion bekannt, stellt die Verbindung zwischen dem Netzinput und dem Aktivitätslevel eines Neurons dar.

Hierbei handelt es sich um eine mathematische Funktion, die auf das Zwischenergebnis, also die Summe der Eingaben multipliziert mit den Gewichtungen (Bild 1.4), angewendet wird.

Somit entscheidet die Aktivierungsfunktion, ab welchem Potenzial das Neuron aktiv wird. Dementsprechend lässt sich für das Neuron ein Aktivierungspotenzial simulieren. Zusätzlich werden die Ausgabewerte der Neuronen einer Schicht normalisiert.

Die Aktivierungsfunktion hat für gewöhnlich einen dynamischen Ausgabebereich zwischen –1 und 1 oder zwischen 0 und 1, wobei bei einigen Aktivierungsfunktionen die Ausgabegrenzen breiter sein können bzw. auch bis unendlich gehen. Die verschiedenen Aktivierungsfunktionen unterscheiden sich in ihrer Komplexität und Ausgabeart. Bild 1.5 zeigt die am häufigsten verwendeten Aktivierungsfunktionen für neuronale Netze.

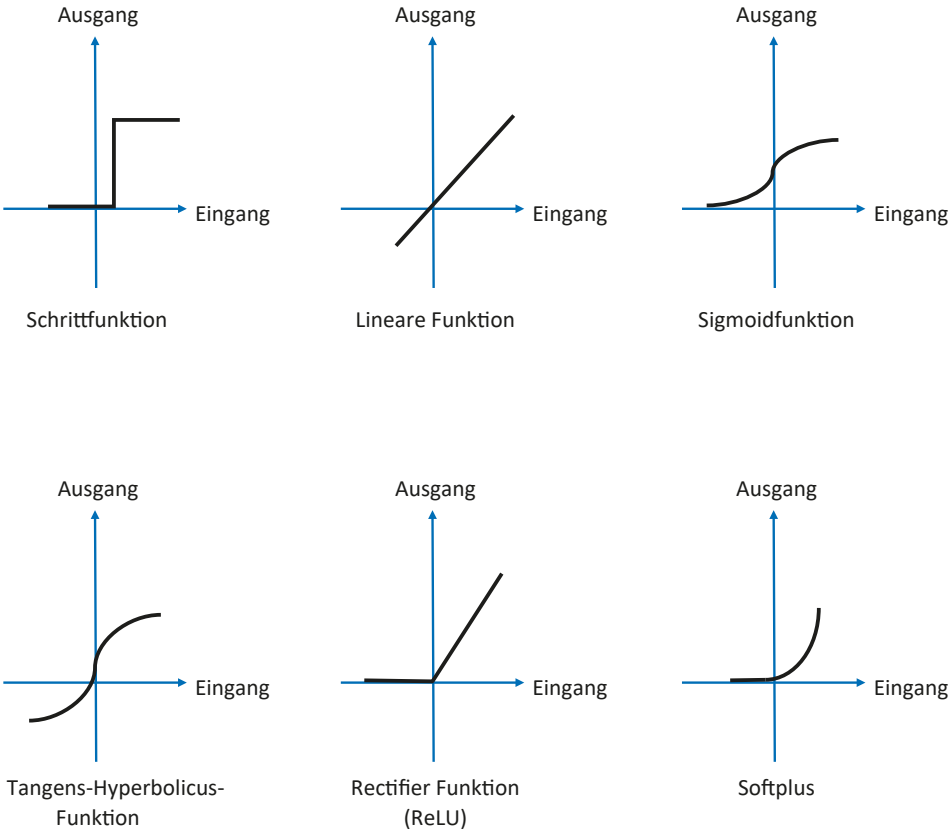


Bild 1.5 Aktivierungsfunktionen

Bei der Auswahl einer Aktivierungsfunktion gibt es keine einfache Faustregel. Alles hängt von den Netzeingabedaten ab und auch davon in welcher Form die Daten nach dem Aktivierungsfunktionsaufruf umgewandelt werden sollen. In der Praxis haben sich aus mathematischer Sicht die Sigmoid-, die Tangens-Hyperbolicus (tanh)-, aber auch die ReLU-Funktion als nützlich erwiesen, da sie sich besonders gut für die Klassifizierung und Mustererkennung eignet. Insgesamt unterscheidet man zwischen folgenden Aktivierungsfunktionen:

- **Schrittfunktion (binäre Schwellenfunktion):** Hier gibt es nur zwei Zustände des Aktivitätslevels, 0 (bzw. manchmal auch –1) oder 1.

- **Lineare Funktion:** Hier ist der Zusammenhang zwischen Netzinput und Aktivitätslevel linear. Die lineare Funktion ermöglicht auch das Arbeiten mit einer Schwelle. Das heißt, bevor der Zusammenhang zwischen den beiden Größen linear wird, muss eine zuvor festgelegte Schwelle überschritten werden.
- **Sigmoidfunktion:** Diese Art von Aktivierungsfunktion wird in vielen Modellen verwendet, die kognitive Prozesse in KNNs simulieren. Es handelt sich um eine differenzierbare reelle Funktion mit einer durchweg positiven oder negativen ersten Ableitung und genau einem Wendepunkt. Mit der Sigmoidfunktion sind Berechnungen wesentlich leichter durchzuführen als mit anderen s-förmigen Funktionen. Sie werden die Sigmoidfunktion in den Praxisbeispielen in diesem Buch des Öfteren einsetzen.
- **Tangens-Hyperbolicus-Funktion (tanh):** Diese Aktivierungsfunktion verhält sich relativ ähnlich wie die Sigmoidfunktion. Die Funktion $\tanh(x)$ ist ebenso sigmoid, stetig und differenzierbar und hat als Wertebereich das Intervall (–1, 1). Auch diese Funktion werden Sie später noch in den Praxisbeispielen kennenlernen.
- **Rectifier-Funktion (ReLU):** Diese Funktion steht im Kontext künstlicher neuronaler Netze als Gleichrichter und definiert eine Aktivierungsfunktion, die als Positivteil ihres Arguments definiert ist. Für den Einsatz im Deep Learning ist diese Funktion dabei, die weit verbreitete Sigmoidfunktion abzulösen. Haupteinsatzzweck ist das Convolutional Neural Network (CNN).
- **Softplus-Funktion:** Auch Softplus ist eine neuere Funktion als Sigmoid oder tanh. Sowohl die ReLU-Funktion als auch Softplus sind weitgehend identisch, außer in der Nähe von 0, hier ist die Softplus-Funktion glatt und differenzierbar. Softplus bietet eine Alternative zu den anderen Funktionen, da sie schnell differenzierbar und ihre Ableitung leicht nachvollziehbar ist.

Wie Sie sehen, weisen die beschriebenen Aktivierungsfunktionen allesamt große mathematische Ähnlichkeiten auf. Es gibt keine universelle Regel für die Auswahl einer Aktivierungsfunktion und es gibt auch keine Aktivierungsfunktion, die für alle Anwendungsfälle funktioniert. So ist die lineare Aktivierungsfunktion ideal für die Lösung eines Regressionsproblems. Die Sigmoidfunktion kann den linearen Übergang nur annähend beschreiben, jedoch niemals exakt abbilden. Die ReLU-Funktion ist die gängigste Aktivierungsfunktion in Convolutional Neural Networks, die bei der Erkennung von Texten und Objekten in Bildern verwendet wird. Persönlich verwende ich sehr gerne die tanh-Funktion, weil sie gut begrenzt und sehr schnell zu berechnen ist, aber vor allem, weil sie für meine Anwendungsbereiche am besten funktioniert. Sie können aber auch Aktivierungsfunktionen mischen und anpassen, um eine spezielle Lösung für Ihren Anwendungsbereich zu entwickeln.

Am Ende jeden Neurons steht die Ausgabefunktion. Da diese Funktion fast ausschließlich nur ihr Argument zurückgibt, wird sie auch als identische Abbildung oder Identität bezeichnet. An einem einfachen Beispiel sehen Sie in Bild 1.6, wie man sehr schnell und einfach errechnen kann, ob das Neuron aktiv wird oder ruht. Als Aktivierungsfunktion wird die einfache Schrittfunktion verwendet.

Um jetzt zu bestimmen, ob das Neuron N_1 aktiv wird, müssen Sie zuerst die Netto-Eingabe ermitteln. Die Netto-Eingabe (*net*) entspricht, wie schon erläutert der Summe der gewichteten Eingangssignale. Daher kann sie wie folgt berechnet werden:

$$net = X_1 \cdot W_1 + X_2 \cdot W_2 = 1 \cdot 0{,}6 + 1 \cdot (-0{,}4) = 0{,}2.$$

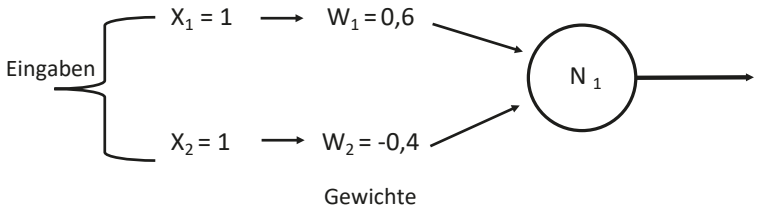


Bild 1.6 Neuron mit Eingabewerte und Gewichten

Die Netto-Eingabe *net* mit dem Wert 0,2 stellt das Argument der Schrittfunktion dar, die bestimmt, ob das Neuron aktiv wird. Die Berechnung $f(net) = f(0{,}2) = 1$ stellt das Neuron aktiv, da 0,2 > 0 und dadurch bestimmt wird, dass das Neuron aktiv wird.

Das Beispiel zeigt das einfachste künstliche neuronale Netz mit nur einem Neuron. Es wird als Linear Threshold Unit, kurz LTU, bezeichnet und stellt mathematisch ein Schwellenwertelement dar, das eine Verarbeitungseinheit für Zahlen mit n Eingängen $X_1, ..., X_n$ und einem Ausgang bildet. Das Element hat einen Schwellenwert und jeder Eingang ist mit einem Gewicht versehen. Mit den geeigneten Gewichten und einem entsprechenden Schwellenwert lässt sich mit einer LTU eine logische UND-Verknüpfung nachbilden.

Wie die Aktivierungsfunktion praktisch eingesetzt wird, lernen Sie in Kapitel 3, wenn es um den Aufbau eines ersten neuronalen Netzes in C# geht.

1.5.2 Matrizendarstellung

Wie Sie an dem einfachen Beispiel gesehen haben, kann die Berechnung der Eingaben, Gewichte und der Ausgaben bei einem größeren Netz zu sehr viel Arbeit führen. Stellen Sie sich den Rechenaufwand bei einem neuronalen Netz mit drei Schichten zu je 50 Knoten vor. Allein das Notieren aller erforderlichen Berechnungen wäre eine riesige Aufgabe. Hier kann die Matrizenmultiplikation sehr nützlich sein.

Künstliche neuronale Netze lassen sich glücklicherweise auch sehr schnell als Matrizen darstellen. Bild 1.7 zeigt die schematische Darstellung und die entsprechende Matrizenschreibweise.

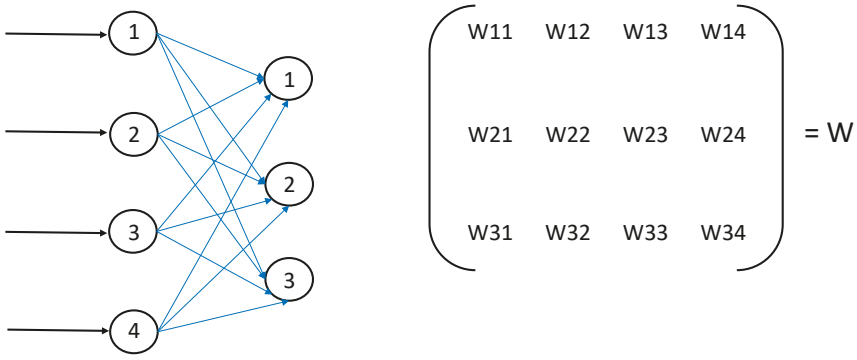


Bild 1.7 Matrizenschreibweise

Diese Darstellung hat den Vorteil, dass die zuvor gezeigten Berechnungen mathematisch relativ einfach sind und zusammenfassend vorgenommen werden können. Infolgedessen stellt die Matrix eine mathematische Einheit dar, genau wie eine einzelne Zahl.

So besteht eine Matrix W aus einer Menge von Elementen W_{ij}. In der hier verwendeten Darstellung bedeutet W_{ij} die Verbindung von Zelle *i* nach Zelle *j*. Da das Lernen in neuronalen Netzen in den Gewichten stattfindet und diese das gelernte Wissen des Netzes speichern, werden die Netzgewichte als Matrix dargestellt. In der Schreibweise $W = w_{ij}$ gilt dann:

- Ein Eintrag in der Matrix W mit $W_i = 0$ gibt an, dass keine Verbindung zwischen *i* und *j* existiert.
- Ein Eintrag in der Matrix W mit $W_i < 0$ gibt an, dass Neuron *i* seinen Nachfolger *j* durch ein Gewicht der Stärke w_{ij} kennt.
- Ein Eintrag in der Matrix W mit $W_i > 0$ gibt an, dass Neuron *i* seinen Nachfolger *j* durch ein Gewicht der Stärke w_{ij} anregt.

Beachten Sie auch, dass die Matrixelemente nicht zwingend Zahlen sein müssen, sie können auch Größen sein, denen Sie einen Namen geben, aber keinen nummerischen Wert zuweisen.

Bei der Berechnung von mehrschichtigen Netzen, zum Beispiel von einem dreischichtigen Netz mit einem Hidden Layer, würden Sie zwei Gewichtsmatrizen benötigen, bei zwei Hidden Layern drei Matrizen usw. Die Multiplikation von Matrizen ist etwas gewöhnungsbedürftig. Als Voraussetzung für die Durchführbarkeit der Multiplikation muss die Anzahl der Spalten der linksstehenden Matrix gleich der Anzahl der Zeilen der rechtsstehenden Matrix sein.

Die sich ergebende Matrix hat so viele Zeilen wie die rechte Matrix und so viele Spalten wie die linke Matrix. Aber keine Angst vor der Mathematik. Die Programmiersprache C# stellt mit der Methode *Matrix.Multiply(Matrix, Matrix)* eine sehr schnelle und effektive Matrixberechnung für unsere Aufgabenstellungen zur Verfügung.

1.6 Architekturprinzipien

Die Struktur eines neuronalen Netzes, ob als FNN, RNN oder auch CNN, besitzt eine große Bedeutung für die Ergebnisse des Lernprozesses. Die Netzarchitektur bestimmt somit das zu verwendende Lernverfahren zur Berechnung der Netzgewichte zwischen den Neuronen.

Das einfachste Prinzip finden Sie bei den einschichtigen Netzen, die nur aus einer Neuronenschicht bestehen und bei denen eine bestimmte Anzahl von Neuronen einen Input-Vektor aufnehmen und andere Neuronen einen Output-Vektor ausgeben. Die Erweiterung des Prinzips stellen dann die geschichteten Netze dar. Diese repräsentieren eine Hierarchie und sind damit leistungsfähiger als einschichtige Netze.

Die jeweilige Topologie sollten Sie daher auf jeden Fall vor der Entwicklung eines neuronalen Netzes festlegen:

- Größe der Eingangs-/Ausgangschicht steht fest
- Anzahl der verdeckten Schichten ist variabel

- Größe der verdeckten Schichten ist variabel
- Entweder strenge Schichtenarchitektur oder beliebige vorwärts gerichtete Schichtenarchitektur

Leider gibt es hier kaum Standard-Regeln. Sie können aber, wenn Sie Ihr erstes eigenes einfaches Netz entwickeln mit folgenden Annahmen experimentieren:

- Es gibt eine verdeckte Schicht und eine Ausgangsschicht.
- Der Eingabevektor hat meist 20 bis 200 Elemente.
- Der Ausgabevektor hat meist 2 bis 100 Elemente.
- Die Anzahl der Neuronen in der verdeckten Schicht liegt bei 50 bis 500.

Des Weiteren sollten Sie die neuronale Netzstruktur hinsichtlich des Anwendungsbereichs festlegen, welcher einer der folgenden sein kann:

- Mustererkennung
- Optimierungsprobleme
- Roboterkontrolle und Überwachung
- Entscheidungstheorie und Klassifizierung

Aber keine Angst, Sie lernen in den nachfolgenden Kapiteln, welche neuronale Netzstruktur für welchen Verwendungszweck am besten geeignet ist.