Karlsruhe Institute of Technology

Maschinelles Lernen im Kontext der Programmierung in natürlicher Sprache

Seminararbeit
von

Philipp Weinmann

An der Fakultät für Informatik
Institut für Programmstrukturen
und Datenorganisation (IPD)

Betreuer: Dipl. Inform. Alexander Wachtel

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Die Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT) habe ich befolgt.

Karlsruhe, 14/01/19 .....

(**Philipp Weinmann**)

Inhaltsverzeichnis

Abstract

Der Forschungsbereich Programmieren in Natürlicher Sprache findet erst seit kurzem größeres Interesse. Dies geschieht in einer Zeit, in der das Maschinelle Lernen an Bedeutung gewinnt. Es ist daher interessant zu betrachten, welche Anwendungsmöglichkeiten Maschinelles Lernen in diesem Bereich der Informatik bietet. In dieser Arbeit werden wir betrachten, was unter dem Begriff Maschinelles Lernen zu verstehen ist und gehen auf Künstliche Neuronalen Netze (ANNs), die Teil von Maschinellem Lernen sind, genauer ein. Des Weiteren werden wir die Vorteile der Anwendung von ANNs für die maschinelle Übersetzung betrachten und bewerten.

Kapitel 1

Einleitung

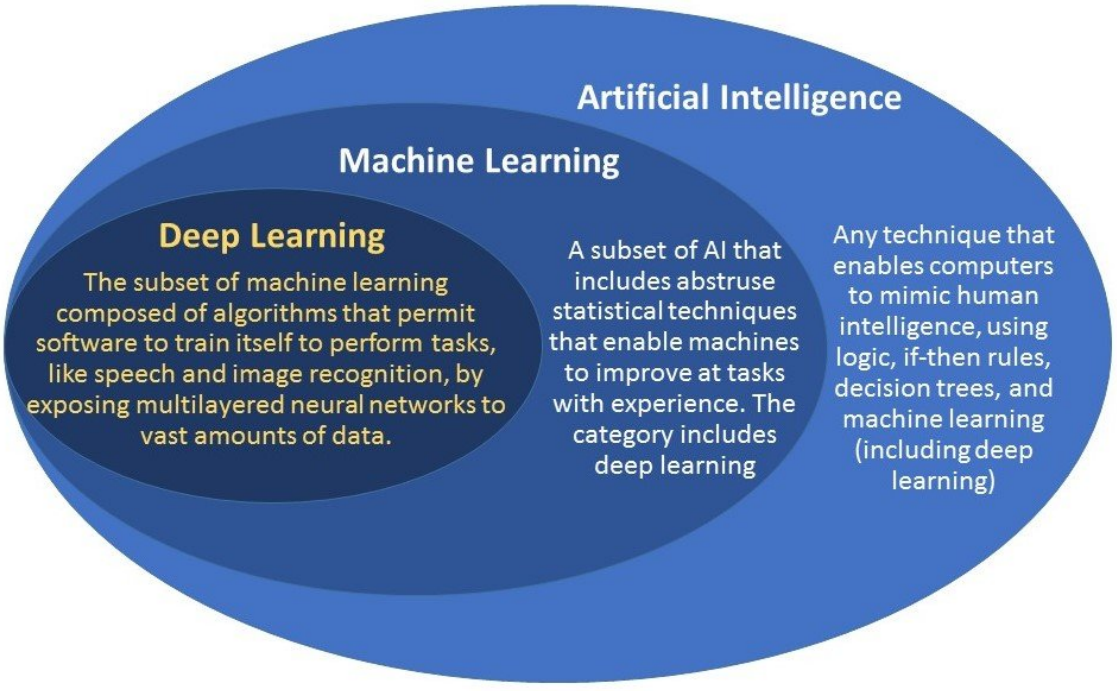

Abbildung 1.1: Veranschaulichung, wie Maschinelles Lernen einzuordnen ist. [Dha17]

Maschinelles Lernen (ML) ist, wie in Abbildung 1.1 beschrieben, ein Teilbereich der Künstlichen Intelligenz (KI). Es ist ein Oberbegriff für die künstliche Generierung von Wissen aus Erfahrung [Ete18]. Weil es oft schwierig ist, bei einer Entscheidung zu erkennen, welche Option das beste Ergebnis liefert, versuchen moderne Programme dies maschinell zu lösen. Dank statistischer Auswertungen können Algorithmen entstehen, die mit einer gewissen Wahrscheinlichkeit korrekte Ergebnisse liefern, ohne das der Programmierer sich Gedanken machen muss, wie der Algorithmus im Detail aufgebaut ist.

1.1 Anwendungen im Bereich der Informatik

In jedem Gebiet, in dem große Mengen an Daten zur Verfügung stehen bzw. generiert werden können, ist ML theoretisch anwendbar.[Kou18]. Obwohl die Theorie hinter dieser Art der Datenauswertung seit langem bekannt ist, so finden mächtigere Algorithmen, die z.B. auf Künstlichen Neuronalen Netzen (ANNs) basieren, erst seit kurzem verbreitete Anwendung. Dies wird allgemein auf die verbesserte Rechenleistung modernen Computer zurückgeführt. [Hwa18]

Anwendungsbereiche sind zum Beispiel:

- Gesichtserkennung
- Spamerkennung (Email)
- Umwandlung von gesprochener Sprache zu Text
- Handschrifterkennung
- Autonomes Fahren
- Automatische Medikamentenentwicklung
- Maschinelle Übersetzungen

In dieser Ausarbeitung werden wir uns insbesondere für Künstliche Neuronale Netze (ANNs) sowie deren Anwendung für maschinelle Übersetzungen interessieren.

1.2 Maschinelles Lernen: Ein erstes Beispiel

Hören sie sich dieses Beispiel an: https://www.audioblocks.com/stock-audio/playground-children-playing.html
Sie erkennen sofort, dass es sich um spielende Kinder handelt. Unser Gehirn schafft es mit extremer Genauigkeit komplexe Geräusche zu erkennen und zu analysieren. Auch wenn wir nicht ausmachen können, was jedes einzelne Kind ruft, so wissen wir instinktiv dass es sich um Kinder handelt. Schauen wir uns einmal die Wellenfunktion eines Abschnittes dieses Audiofiles an:

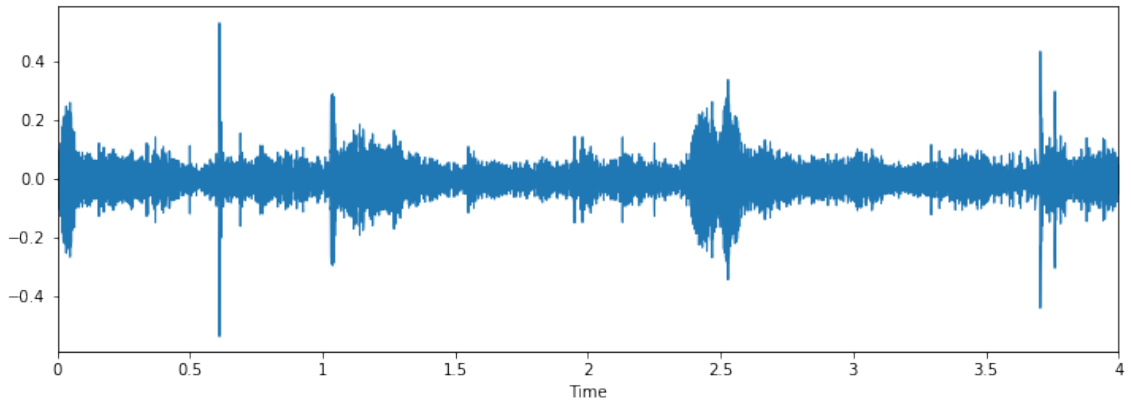

Abbildung 1.2: Audiofile spielender Kinder. Luftdruck/Umgebungsdruck in Funktion der Zeit (Sekunden). [SF17]

Betrachten wir Wellenfunktion eines Presslufthammers, bekommen wir eine sehr unterschiedliche Wellenfunktion:

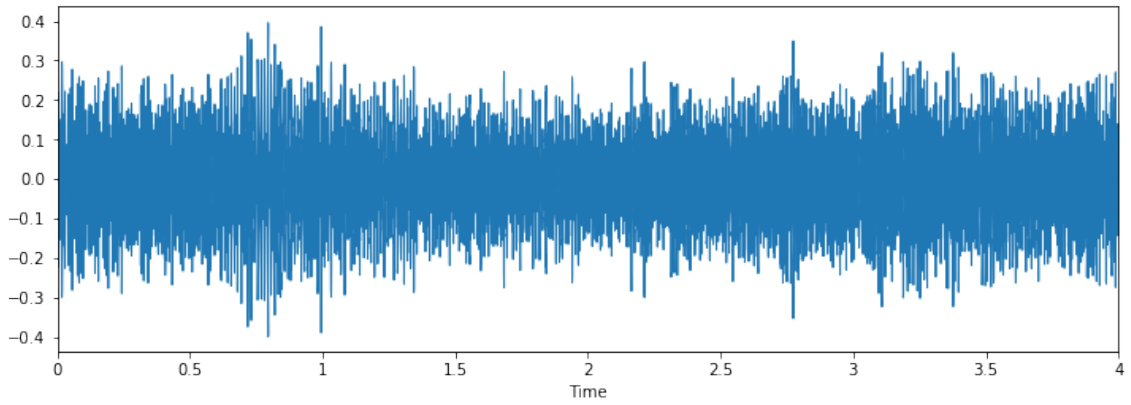

Abbildung 1.3: Audiofile eines Presslufthammers. Luftdruck/Umgebungsdruck in Funktion der Zeit (Sekunden). [SF17]

Wir können einige klare Unterschiede ausmachen. Falls wir nun ein Programm schreiben wollen, das erkennt ob ein Audiofile spielende Kinder oder das Geräusch eines Presslufthammers enthält, so können wir anhand der Wellenfunktion dieser Geräusche einige Lösungsansätze folgern. Wir bemerken zum Beispiel, dass es bei kreischenden Kindern deutlich mehr Ausreißer gibt als bei dem Geräusch eines Presslufthammers. Wir könnten also ein Programm schreiben, das ein Audiofile nach der Anzahl an Ausreißern pro Sekunde (**APS**) erkennt, d.h. klassifiziert.

```
String classify(int[] audiofile) {
        float APS = getAPS(audiofile);  // AusreisserProSekunde
        if (APS > (spielkinderDurchschnittsAPS +
            presslufthammerDurchschnittsAPS) / 2)){
                return "SpielendeKinder";
        } else {
                return "PresslufthHammer";
        }
}
```

Um diesen Code ausführen zu können muss bekannt sein, welchen Wert die Ausreißer pro Sekunde (**APS**) von spielenden Kindern bzw. von Presslufthämmern im Durchschnitt annehmen. Um dies Abzuschätzen, ist ein großer Datensatz an Audiodateien, bei denen bekannt ist um welche Geräusche es sich handelt, nötig. Falls dieser vorhanden sind, kann folgendes Programm geschrieben werden, welches den Durchschnittswert der Ausreißer pro Sekunde (**APS**) für die jeweilige Kategorie ermittelt:

```
float mittelwertAPS = (spielkinderDurchschnittsAPS +
    pressluftHammerDurchschnittsAPS) / 2);
     String classify(int[] audiofile, String category) {
                float APS = getAPS(audiofile);
                        if (APS > mittelwertAPS){
                                updateAPS(APS, category);

                                return "SpielendeKinder";
                        } else {
                                updateAPS(APS, category);
                                return "PressluftHammer";
                        }
    }
void updateAPS(int APS, String category) {
        if(category.equals(spielendeKinder)) {
                spielendeKinderAPSList.add(APS);
        } else {
                pressluftHammerAPSList.add(APS);
        }
}
```

Dieses zweite Programm ermittelt noch immer, welcher Kategorie die Audiofiles angehören. Daher kann es, falls bekannt ist um welche der beiden Kategorien es sich handelt weiterverwendet werden. Die Ausreißer pro Sekunde (**APS**) werden damit mit jedem neuen vorklassifizierten Audiofile (*labeled data*) präziser und das Programm lernt mit der Zeit. Dies ist ein Beispiel sehr rudimentärem Maschinellem Lernen.
Es gibt mächtigere Programme, in denen der Algorithmus nicht nur den Wert einer Variablen erkennt, sondern auch die Kriterien zur Unterscheidung zwischen Kategorien oder sogar Kategorien selber ermittelt.

1.3 Aufkommen von Maschinellem Lernen

Maschinelles Lernen generell und insbesondere Neuronale Netze erfreuen sich seit einigen Jahren großer Aufmerksamkeit. Anfangs wurden erst statistische Analysemethoden entdeckt und verbessert [BPC63][Leg05][Mar06]. Ab der zweiten Hälfte des 20. Jahrhunderts, mit der Entwicklung der ersten Computern, werden mehrere Pionierarbeiten über das Maschinelle Lernen publiziert. 1950 formuliert Alan Turing die Turing Learning Machine [Mac50], ein Jahr später entwickeln und bauen zwei Wissenschaftler das erste neuronale Netz [MM51]. 1957 entwickelt Frank Rosenblatt den ”perceptron”[Ros58], ein erstes Modell eines vollständig verbundenen neuronalen Netzes (Fully connected Neural Network), siehe Abbildung 1.4.

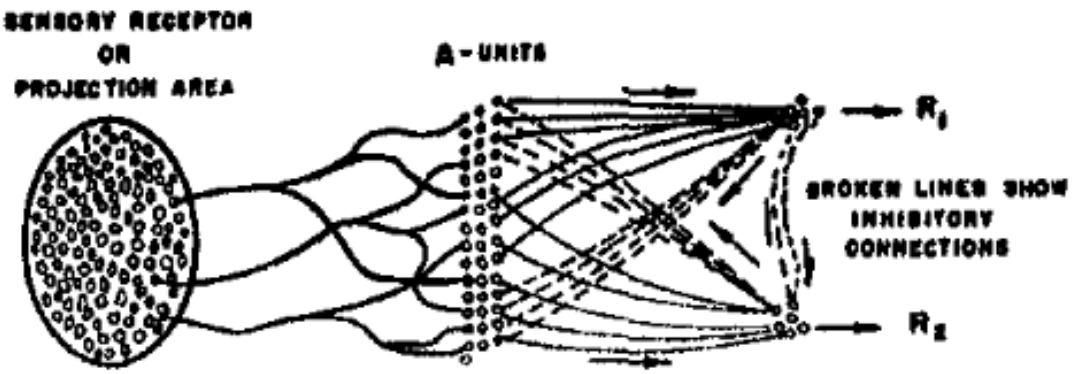

Abbildung 1.4: Zeichnung eines Perceptrons in der ursprünglichen Arbeit. [Ros58]

Mit der Weiterentwicklung der Computerhardware und deren allgemeinen Nutzung werden ab den 90er Jahren Algorithmen entwickelt, die große Mengen an Daten analysieren können. Anfang des 21. Jahrhunderts und insbesondere ab 2010 hat die Rechnerleistung so sehr zugenommen, dass sehr rechenintensive Anwendungen von Maschinellem Lernen, wie das *Deep learning*, praktikabel werden [TYRW14] [Mic].

Kapitel 2

Künstliche Neuronale Netze (ANNs)

Ein technischer Durchbruch ist oft *nur* die gelungene Nachahmung eines in der Natur vorkommenden Phänomens. ANNs sind der Versuch die Funktion des menschlichen Gehirns nachzuahmen. Während diese Unterfangen nicht oder nur teilweise gelungen sind [Ada18], haben sich ANNs dennoch als sehr nützlich erwiesen.

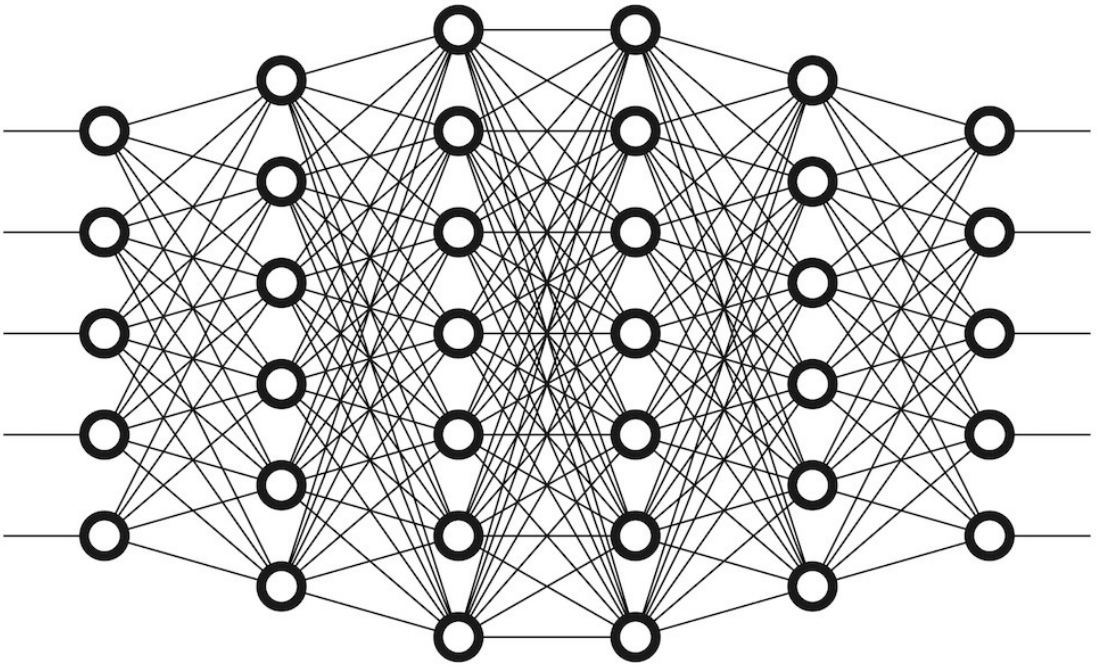

Abbildung 2.1: Topologie eines Fully connected Neural Network.[uem]

Diese Netze sind Programmiergerüste (*Engl: Frameworks*) für Algorithmen, die auf maschinellem Lernen basieren. Beschrieben werden sie mit Begriffen aus der Biologie. Ein ANN, besteht aus einer gewissen Anzahl an Knoten, sogenannten *Neuronen*, die bei unterschiedlichen Eingaben unterschiedlich aktiviert werden. Diese sind mit weiteren Neuronen über sogenannte *Synapsen* verbunden und versenden je nach ANNs mehr oder weniger starke Signale über diese Verbindungen. Wie in Abbildung 2.1 dargestellt, können Neuronen in Lagen (*Layers*) zusammengefasst werden, welche die Stufen des ANNs darstellen, die mithilfe der Breitensuche bestimmt werden können. Die erste Lage wird als *inputlayer* beschrieben und die letzte Lage wird *outputlayer* genannt. Jede Lage zwischen diesen beiden werden als *hidden layers* beschrieben, weil sie nicht mit der Systemumgebung kommunizieren.
Wie in unserem Beispiel erwähnt, ist das Ziel von neuronalen Netzen Eigenschaften zu erkennen, die gewisse Objekte gemeinsam haben. ANNs sollen:

- Klassifizieren
- Zusammenhänge erkennen

Es folgt nun eine kurze Beschreibung einiger ausgewählter Bestandteile von ANNs, um über deren Funktionsweise einen Überblick zu bekommen.

2.1 Aktivierungsfunktion

Informationen werden in ANN über Signale von Neuron zu Neuron weitergegeben. Jedoch werden diese Signale nicht direkt weitergegeben sonder müssen umgewandelt werden. Das Ausgabesignal eines Neurons wird durch die Aktivierungsfunktion berechnet. Diese bildet den Aktivierungswert auf ein Intervall ab und ist daher eine nichtlineare Funktion. Ohne diese Nichtlinearität könnten komplexe Datensätze wie Bilder, Videos sowie Audio nicht analysiert werden. Prägnant wird dies durch folgendes Zitat beschrieben:

"Neural-Networks are considered Universal Function Approximators. It means that they can compute and learn any function at all. Almost any process we can think of can be represented as a functional computation in Neural Networks." [Wal17] Dies bedeutet, das ANN sehr vielfältig angewendet werden können. Eine früher beliebte Aktivierungsfunktion war die *normalisierte Sigmoid Funktion.*

$$f(x) = \frac{1}{1+e^{-x}},\ x\ die\ Eingabe,\ f(x)\ die\ Ausgabe\ des\ Neurones. \tag{2.1}$$

Während dies die wohl allgemein bekannteste Aktivierungsfunktion ist, verwenden moderne ANNs heute eine Variante der *Rectifier Funktion*, welche sich besser eignet zum trainieren von ANNs [LBH15]

$$g(x) = max(0,x),\ x\ die\ Eingabe,\ g(x)\ die\ Ausgabe\ des\ Neurones. \tag{2.2}$$

Eine beliebte Approximation dieser Funktion ist z.B. die analytische Funktion

$$h(x) = log(1+e^{x}),\ x\ die\ Eingabe,\ h(x)\ die\ Ausgabe\ des\ Neurones. \tag{2.3}$$

2.2 Propagierungsfunktion

Jede Verbindung zwischen zwei Neuronen besitzt eine Gewichtung (*Engl: weight*). Diese beschreibt wie viel Einfluss sie auf die Aktivierung des verbundenen Neurons ausübt.

Die *Propagierungsfunktion* berechnet den Input $p_j^l(t)$ des Neurons j anhand der Outputs $o_i(t)$ der Neuronen in der vorangehenden Lage l.

$$p_j^{(l+1)}(t) = \sum_i o_i^{(l)}(t) w_{ij}^{(l)}$$

Um eine Aktivierungsschwelle (*Engl: threshold*) einzuführen, kann ein sogenannter *bias* der Summe hinzugefügt werden. Dies ist nötig, wenn ein Neuron nur ab einem bestimmten Aktivierungswert von Bedeutung ist.

$$p_j^{(l+1)}(t) = \sum_i o_i^{(l)}(t) w_{ij}^{(l)} - bias$$

Ein großer Vorteil von ANNs ist, dass sie als eine Folge von Matrixmultiplikationen und Anwendung der Aktivierungsfunktion darstellbar sind, welche besonders effizient durch Computerprozessoren berechnet werden können.

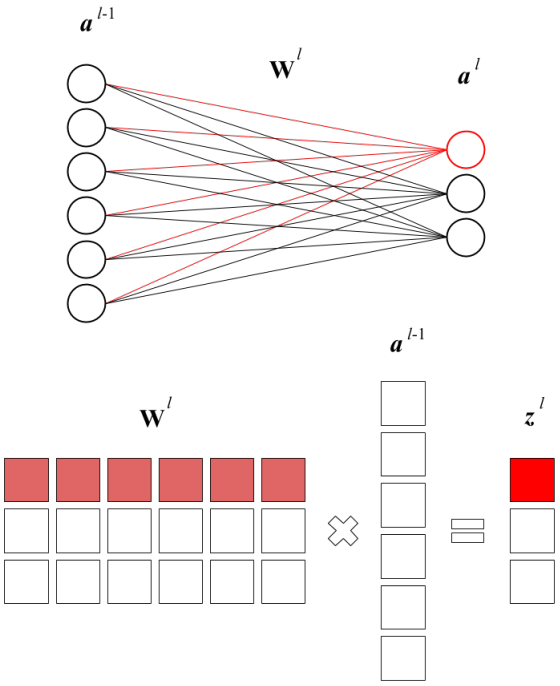

Abbildung 2.2: Neuronales Netz in Matrixdarstellung. Hierbei ist $a^l = f(z^l)$ wobei f die Aktivierungsfunktion darstellt. [Hal16]

2.3 Trainieren eines ANN

Wie zuvor beschrieben, benötigt ein ANN massive Datensätze. Zum Trainieren wird ein Algorithmus genutzt, der anhand dieser Daten die *weights* und *biases* so einstellt, dass das Netzwerk seine Funktion erfüllt. Dies ist wegen der massiven Anzahl an Parametern (Größenordnungen von 10^6 Parametern werden bei modernen Netzen erreicht [SLJ+15]) nichttrivial. Ein weitverbreiteter Algorithmus ist die Fehlerrückführung (*Backpropagation*).

2.4 Backpropagation

Der Backpropagation Algorithmus verläuft folgendermaßen [Bib19]: erst werden Eingabemuster durch das ANN propagiert, daraufhin wird die Ausgabe des Netzes, d.h. die Aktivierung des letzten Layers verglichen mit der gewünschten Aktivierung und der Fehler des Netzes durch die Summe der Quadrate der Abweichungen berechnet.

$$E = \frac{1}{2}\sum_{i=1}^{n}(t_i - o_i)^2, \ mit$$

E die Fehlerfunktion
t_i der gewünschte Output beim Muster i
o_i der wirkliche Output beim Muster i
n die Anzahl an Mustern, die dem Netz vorgestellt werden

Das Ziel ist es nun die Summe der Fehlerfunktionen über alle möglichen Inputs zu minimieren. Dies ist nicht möglich ohne alle möglichen Kombinationen von *weights* (und falls vorhanden *biases*) auszuprobieren, jedoch ist es möglich dank des Gradientenverfahrens lokalen Minimas nahe kommen.

2.5 Fehlerminimierung

Sei w_{ij} das Gewicht der Verbindung des Neurones i zum Neuron j. Wir versuchen dieses nun so anzupassen, das die Fehlerfunktion minimiert wird. Dazu muss die partielle Ableitung der Fehlerfunktion E berechnet werden:

$$\frac{\partial E}{\partial w_{ij}} \tag{2.4}$$

mit einem Lernfaktor η kann eingestellt werden, wie schnell das Neuronale Netz sich dem Lokalen Minimum nähert. Jedoch können zu starke Korrekturen *"über das Ziel hinausschießen"* d.h. einem lokalen Minimum außer durch Zufall nicht nahe genug kommen und somit den Nutzen des Trainierens beschränken. Wir definieren infolgedessen die Änderung des Gewichtes Δw_{ij} als:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{2.5}$$

Und somit können wir die Änderung der Gewichte folglich vornehmen:

$$w_{ij}^{neu} = w_{ij}^{alt} + \Delta w_{ij},\ mit \tag{2.6}$$

w_{ij}^{neu} *der neue Wert des Gewichts der Verbindung zwischen Neuronen i und j*
w_{ij}^{alt} *der alte Wert des Gewichts der Verbindung zwischen Neuronen i und j*
Δw_{ij} *die Änderung des Gewichts der Verbindung zwischen Neuronen i und j*

2.6 Limitationen und Gefahren von Neuronalen Netzen

Während sich ANNs im Moment großer Beliebtheit erfreuen, ist es wichtig einiger ihrer Nachteile bewusst zu sein. Wie zuvor erwähnt, werden durch das Gradientenverfahren nur lokale Minima gefunden. Die Performance eines ANN nach dem Trainieren ist daher stark von den Anfangswerten der weights und biases abhängig und kann unter Umständen auf enttäuschendem Niveau stagnieren, unabhängig davon wie viele Daten zur Verfügung stehen. Das Trainieren von ANNs ist noch immer sehr Rechnerleistungsintensiv und benötigt große Mengen an vorklassifizierten Daten (labeled data), die nicht immer zur Verfügung stehen. Durch diese neuen Technologien sind Daten sehr wertvoll geworden. Während ML zwar nicht verantwortlich ist für den unvorsichtigen Umgang und Handel mit unseren Daten, so verstärken sie diesen Trend. Es wird oft angenommen, das es nicht möglich sei ein ANN zu verstehen, da es sich um eine *black box* handele. Obwohl einige Fortschritte im Verständnis der inneren Funktion von trainierten ANNs gemacht worden sind [RSG16], ist es noch immer schwierig trainierte ANNs oder ähnliche Konstrukte zu verstehen. Es sollte immer damit gerechnet werden, dass ein neuronales Netz auf völlig vorhergesehene Weise auf eine neue Eingabe reagiert. Es können kaum Garantien für das Verhalten eines neuronalen Netzes gegeben werden. Anwendungen von ANNs und der damit verbundene Kontrollverlust werfen ethische Fragen auf, mit denen wir uns auseinandersetzen müssen [Gib18].

Trotz dieser Bedenken hat Maschinelles Lernen große Fortschritte in vielen Bereichen der Informatik ermöglicht. Im Bereich der Programmierung in Natürlicher Sprache basiert ein großer Teil der Anwendungssoftware auf Maschinellem Lernen. Ein Beispiel dieser Anwendungssoftware ist sind automatische Übersetzungen, welche heutzutage auf ANNs

basieren.

Kapitel 3

Maschinelle Übersetzungen (MT)

3.1 Neuronale Maschinenübersetzung (NMT)

Während Chatbots noch weit davon entfernt sind den Turingtest zu bestehen, so haben Maschinelle Übersetzungen dank ANNs quasi menschliche Fehlerraten erreicht [goo16].

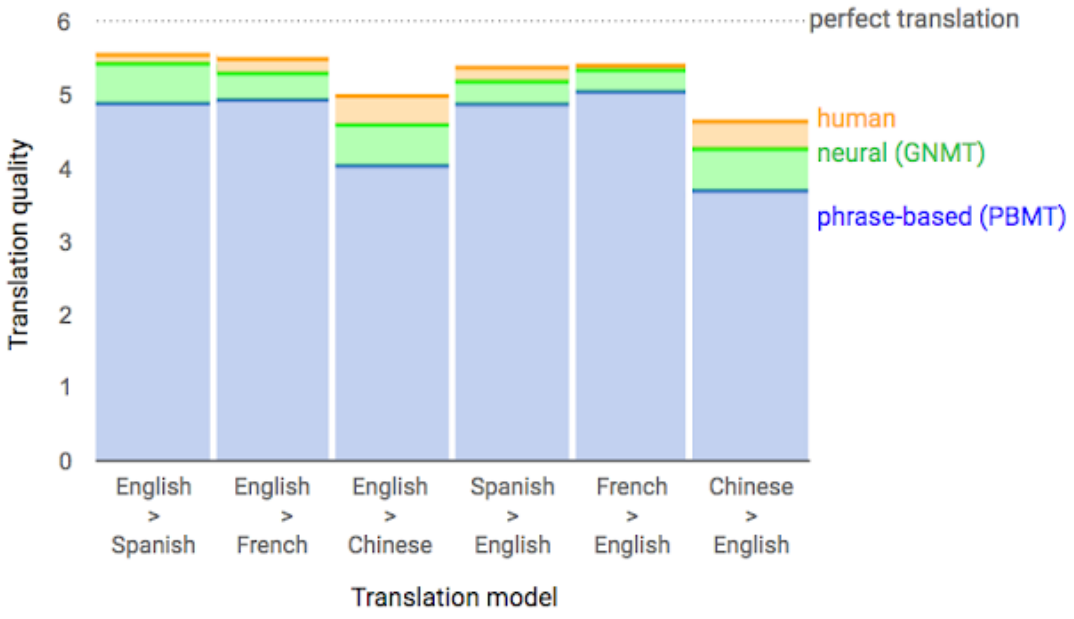

Abbildung 3.1: Personen mit exzellenten Sprachkenntnissen in der Ursprungs und Zielsprache wurden gebeten Übersetzungen von Google Translate auf einer Skala von 0 bis 6 zu bewerten. [goo16]

Wie in Abbildung 3.1 zu erkennen ist, hat NMT signifikative Verbesserungen in MT ermöglicht. Dank ANNs behauptet Google z.B. ihre Fehlerraten um 60% gesenkt zu ha-

ben [WSC+16a]. Andere Dienste wie Microsoft sowie Yahoo melden ähnliche Fortschritte [edi18].

3.2 Long short-term memory

Moderne neurale Übersetzungsmaschinen (NMT) nutzen Long short-term memory Netze [WSC+16b]. Eine Form dieser Netzwerke sind die Rekursiven Neuronale Netzen. Diese ANNs haben die Eigenschaft, dass sie zyklische Verbindungen erlauben und somit sequentielle Eingaben verarbeiten können. Diese gelten als besonders schwierig zu implementieren sind aber essentiell zur Analyse von Daten wie gesprochener oder geschriebener Text. Betrachten wir als Beispiel das Elman Netzwerk:

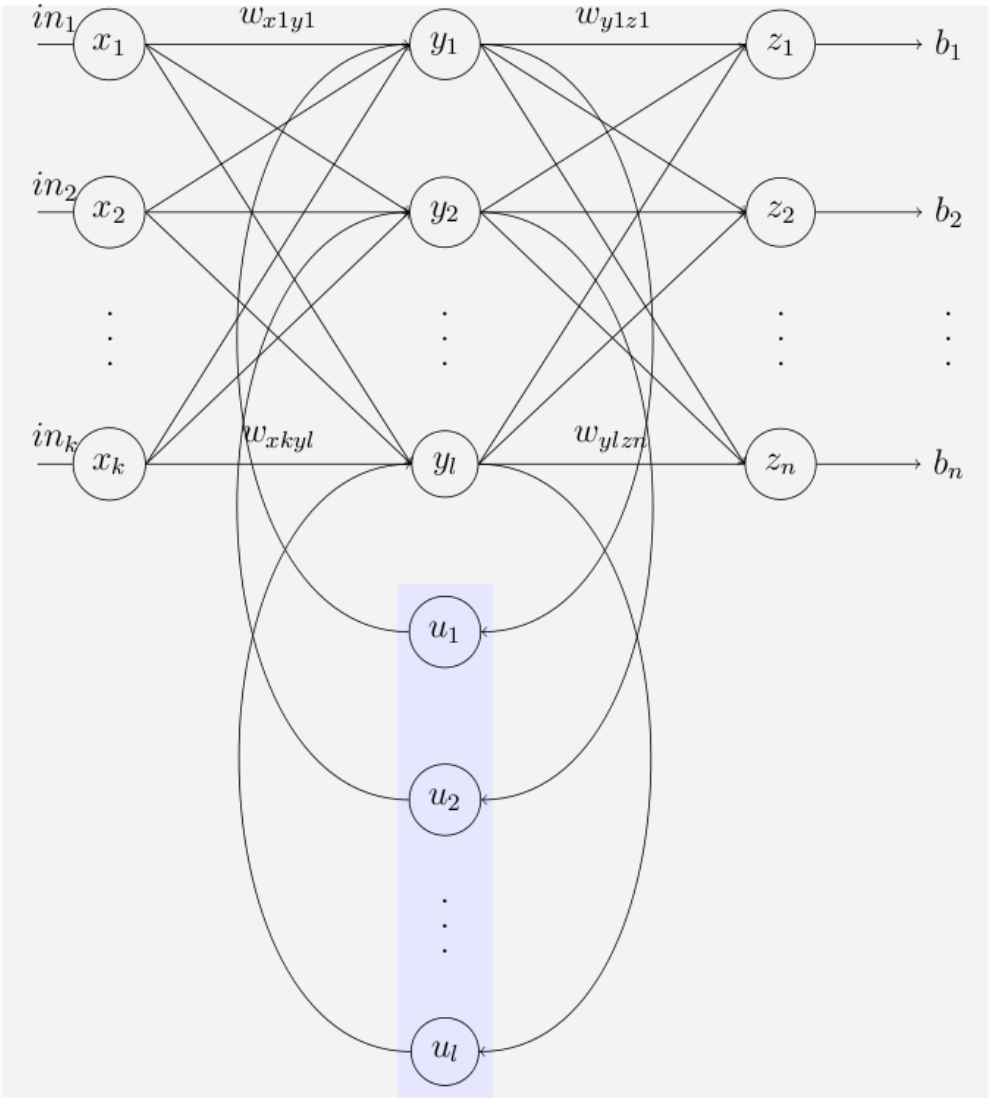

Abbildung 3.2: Das Elman Netzwerk[Cmd]

In Abbildung 3.2 sind Rückwärtsverbindungen dargestellt. Es gibt einerseits eine Vorwärtsverbindung von y_l nach u_l sowie eine Rückwärtsverbindung von u_l nach y_l. Die Rückwärtsverbindungen merken sich eine Kopie des Aktivierungsgraded ihres Ausgangsneurons. Dadurch können Eingaben gespeichert werden und bei neuen Eingaben in Betracht gezogen werden. Dadurch wird sequentielle Betrachtung von Daten ermöglicht. In der Praxis werden weitaus komplexere neuronale Netze verwendet [DPW96], welche aber auf denselben

Mechanismen basieren wie die hier beschriebenen. Nennenswert sind z.B. Convolutional Neural Networks, das Hopfield Netzwerk, die Boltzmann Maschine, sowie Autoencoders. Diese im Detail zu beschreiben würde jedoch den Rahmen dieser Arbeit sprengen.

Kapitel 4

Bewertung

ML ist ein mächtiges Werkzeug in vielen Bereichen der Informatik. Insbesondere ANNs haben Teile der Informatik schon revolutioniert. Jedoch besitzen diese Werkzeuge, dieselben Probleme die wir Menschen auch haben. Sie sind nur so gut wie die Daten, die ihnen zum Trainieren zur Verfügung gestellt werden. Des Weiteren sind genaue Lösungswege nur schwer zu durchschauen. Sprachen verändern sich ständig, daher ist es von nutzen, das Übersetzungstools ebenso dem allgemeinen Sprachgebrauch anpassen.

Literaturverzeichnis

[Ada18] Adan, Yariv: Do neural networks really work like neurons? In: *Medium* (2018), Oct. `https://medium.com/swlh/do-neural-networks-really-work-like-neurons-667859dbfb4f`

[Bib19] *Backpropagation – Wikipedia.* `https://de.wikipedia.org/wiki/Backpropagation#Algorithmus`. Version: Jan 2019. – [Online; accessed 13. Jan. 2019]

[BPC63] Bayes, Thomas ; Price, Richard ; Canton, John: An essay towards solving a problem in the doctrine of chances. (1763)

[Cmd] The Elman Simple Recurrent Neural Network https://commons.wikimedia.org/wiki/File:Elman_srnn.png

[Dha17] Aufgerufen: 03/12/18 https://www.geospatialworld.net/blogs/difference-between-ai%EF%BB%BF-machine-learning-and-deep-learning/

[DPW96] Dellaert, Frank ; Polzin, Thomas ; Waibel, Alex: Recognizing emotion in speech. In: *Fourth International Conference on Spoken Language Processing*, 1996

[edi18] editor, Microsoft blog: *Customized neural machine translation with Microsoft Translator*. https://www.microsoft.com/en-us/research/blog/customized-neural-machine-translation-microsoft-translator/. Version: May 2018

[Ete18] Eter, Nicole: *Deep Learning in der AugenheilkundeDeep learning in ophthalmology*. https://link.springer.com/article/10.1007/s00347-018-0713-1#citeas. Version: May 2018

[Gib18] Gibbs, Samuel: *Google's AI is being used by US military drone programme*. https://www.theguardian.com/technology/2018/mar/07/google-ai-us-department-of-defense-military-drone-project-maven-tensorflow. Version: Mar 2018

[goo16] google: *A Neural Network for Machine Translation, at Production Scale*. https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html. Version: Sep 2016

[Hal16] Hallstroem, Erik: *Backpropagation from the beginning – Erik Hallstroem – Medium*. https://medium.com/@erikhallstrm/backpropagation-from-the-beginning-77356edf427d. Version: Dec 2016

[Hwa18] Hwang, Tim: Computational Power and the Social Impact of Artificial Intelligence. In: *CoRR* abs/1803.08971 (2018). http://arxiv.org/abs/1803.08971

[Kou18] Kour, Bhupinder: *The Rise of Machine Learning and AI is Improving Lives in 2018*. https://www.smartdatacollective.com/rise-of-machine-learning-ai-improving-lives/. Version: Jan 2018

[LBH15] LeCun, Yann ; Bengio, Yoshua ; Hinton, Geoffrey: *Deep learning*. https://www.nature.com/articles/nature14539. Version: May 2015

[Leg05] Legendre, Adrien M.: *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805

[Mac50] Machinery, Computing: Computing machinery and intelligence-AM Turing. In: *Mind* 59 (1950), Nr. 236, S. 433

[Mar06] Markov, Andreĭ A.: An example of statistical investigation of the text Eugene Onegin concerning the connection of samples in chains. In: *Science in Context* 19 (2006), Nr. 4, S. 591–600

[Mic] Microsoft: *Microsoft Translator launching Neural Network based translations for all its speech languages*. https://blogs.msdn.microsoft.com/translation/2016/11/15/microsoft-translator-launching-neural-network-based-translations-for-all-its-speech-languages/

[MM51] Frank Rosenblatt paper description http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/

[Ros58] Rosenblatt, Frank: The perceptron: a probabilistic model for information storage and organization in the brain. In: *Psychological review* 65 (1958), Nr. 6, S. 386

[RSG16] Ribeiro, Marco T. ; Singh, Sameer ; Guestrin, Carlos: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, S. 1135–1144

[SF17] Shaikh, Faizan ; Faizan: *Getting Started with Audio Data Analysis (Voice) using Deep Learning*. https://www.analyticsvidhya.com/blog/2017/08/audio-voice-processing-deep-learning/. Version: Aug 2017

[SLJ+15] Szegedy, Christian ; Liu, Wei ; Jia, Yangqing ; Sermanet, Pierre ; Reed, Scott ; Anguelov, Dragomir ; Erhan, Dumitru ; Vanhoucke, Vincent ; Rabinovich, Andrew: Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, S. 1–9

[TYRW14] Taigman, Yaniv ; Yang, Ming ; Ranzato, Marc'Aurelio ; Wolf, Lior: Deepface: Closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, S. 1701–1708

[uem] uemit: *Neuronales Netz Bild*. https://machine-learning-blog.de/2017/11/02/was-ist-deep-learning/. – Aufgerufen: 09.01.2017

[Wal17] Walia, Anish S.: *Activation functions and it's types-Which is better?* https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f. Version: May 2017

[WSC+16a] Wu, Yonghui ; Schuster, Mike ; Chen, Zhifeng ; Le, Quoc V. ; Norouzi, Mohammad ; Macherey, Wolfgang ; Krikun, Maxim ; Cao, Yuan ; Gao, Qin ; Macherey, Klaus u. a.: Google's neural machine translation system: Bridging the gap between human and machine translation. In: *arXiv preprint arXiv:1609.08144* (2016)

[WSC+16b] Wu, Yonghui ; Schuster, Mike ; Chen, Zhifeng ; Le, Quoc V. ; Norouzi, Mohammad ; Macherey, Wolfgang ; Krikun, Maxim ; Cao, Yuan ; Gao, Qin ; Macherey, Klaus ; Klingner, Jeff ; Shah, Apurva ; Johnson, Melvin ; Liu, Xiaobing ; Kaiser, Lukasz ; Gouws, Stephan ; Kato, Yoshikiyo ; Kudo, Taku ; Kazawa, Hideto ; Stevens, Keith ; Kurian, George ; Patil, Nishant ; Wang, Wei ; Young, Cliff ; Smith, Jason ; Riesa, Jason ; Rudnick, Alex ; Vinyals, Oriol ; Corrado, Greg ; Hughes, Macduff ; Dean, Jeffrey: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. In: *CoRR* abs/1609.08144 (2016). http://arxiv.org/abs/1609.08144

Glossar

ANN Artificial Neural Network ein Künstliches Neuronales Netz..

Ausreißer Ein Funktionswert der um ein vielfaches von seinem zeitlich vorgehenden Wert abweicht..

Fully connected Neural Network ein Neuronales Netz in dem jede Lage mit der nächsten verbunden ist..

KI Jegliches Programm das es einer Maschine ermöglicht auf ihre Umwelt zu reagieren..

ML Maschinelles Lernen künstliche generierung von Wissen aus Erfahrung..

MT Machine Translation eine automatische Übersetzung von geschriebener oder gesprochener Sprache in eine andere Sprache bzw. Form..

NMT Neural machine translation automatische Übersetzung von geschriebener oder gesprochener Sprache in eine andere Sprache bzw. Form mithilfe von Künstlichen Neuronalen Netzen (ANN)..

Turingtest ein von Alan Turing entwickelter Test ob eine Maschine ein dem Menschen gleichwertiges Denkvermögen hat.