



# **Implementierung eines neuronalen Netzwerkes zur Zeichenerkennung in SetIX**

## **Studienarbeit**

Studiengang Angewandte Informatik

Duale Hochschule Baden-Württemberg Mannheim

von

Lucas Heuser und Johannes Hill

Bearbeitungszeitraum:	05.09.2016 - 29.05.2017
Matrikelnummer, Kurs:	-, TINF14AI-BI
Matrikelnummer, Kurs:	-, TINF14AI-BI
Ausbildungsfirma:	Roche Diagnostics GmbH, Mannheim
Abteilung:	Scientific Information Services
Betreuer der DHBW-Mannheim:	Prof. Dr. Karl Stroetmann

---

UNTERSCHRIFT DES BETREUERS

# Eidesstattliche Erklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit mit dem Thema

*Implementierung eines neuronalen Netzwerkes zur Zeichenerkennung in SetIX*

selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Mannheim, den 21. März 2017

---

LUCAS HEUSER

---

JOHANNES HILL

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Studienarbeit and DHWB	1
1.2	GitHub Link	1
1.3	Was ist künstliche Intelligenz	1
1.4	Aktuelle Relevanz/Themen von neuronalen Netzen	1
1.5	Ziel der Arbeit	1
1.6	Aufbau der Arbeit	1
<b>2</b>	<b>Theorie</b>	<b>2</b>
2.1	SetlX	2
2.2	MNIST	2
2.3	Perceptrons	2
2.4	Sigmoid Neurons	4

# Abbildungsverzeichnis

2.1	Percetron mit den Eingaben $x_1, x_2, x_3$ und der Ausgabe <i>output</i> .	2
2.2	Unterschiedliche Möglichkeiten der Entscheidungsfindung.	3
2.3	Percetron mit zwei Eingaben -2 und einem Bias von 3.	3
2.4	NAND Gatter mit den Eingaben $x_1$ und $x_2$ .	4
2.5	NAND Gatter Aufbau mit Perceptrons.	4
2.6	Vereinfachter NAND Gatter Aufbau mit Perceptrons.	4
2.7	Modifizieren von Weights und Biases schaffen lernendes Netzwerk.	5

# Kapitel 1

## Einleitung

### 1.1 Studienarbeit and DHWB

### 1.2 GitHub Link

### 1.3 Was ist künstliche Intelligenz

### 1.4 Aktuelle Relevanz/Themen von neuronalen Netzen

### 1.5 Ziel der Arbeit

Die menschliche Wahrnehmung ist

### 1.6 Aufbau der Arbeit

# Kapitel 2

## Theorie

### 2.1 SetIX

### 2.2 MNIST

### 2.3 Perceptrons

Ein Perceptron ist ein mathematisches Modell zur Abbildung eines künstlichen Neurons in einem Netzwerk. Es wird für die Entscheidungsfindung herangezogen, indem verschiedene Aussagen abgewägt werden. Hierbei nimmt das Perceptron eine Menge von Eingaben  $x_n$  mit  $n \in \{1, \dots, n\}$  und berechnet einen einzigen binären Ausgabewert (siehe Abb. 2.1). Für die Berechnung der Ausgabe werden

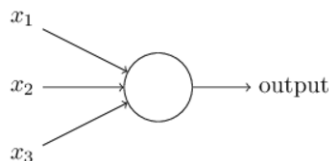


Abbildung 2.1: Perceptron mit den Eingaben  $x_1, x_2, x_3$  und der Ausgabe *output*.

sogenannte *Weights*  $w_n$  mit  $n \in \{1, \dots, n\}$  eingeführt, welche die Gewichtung der jeweiligen Eingabe festlegen. Der Ausgabewert *output* wird mittels der gewichteten Summe  $\sum_j w_j x_k$  und einem definierten Grenzwert *threshold* bestimmt.

$$\text{output} := \begin{cases} 0 & \text{falls } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{falls } \sum_j w_j x_j > \text{threshold} \end{cases} \quad (2.1)$$

Werden die *Weights* und der *Threshold* variiert, entstehen unterschiedliche Modelle zur Entscheidungsfindung. Hierbei ist zu beachten, dass eine Minimierung des *Thresholds* den binären Ausgabewert 1 mit einer höheren Wahrscheinlichkeit bedingt.

Der Aufbau des Netzwerks leitet sich aus den unterschiedlichen Modellen der Entscheidungsfindung ab und wird mit Hilfe der Perceptrons abgebildet (siehe Abb. 2.2). Eine Entscheidungsmöglichkeit wird hierbei durch das Perceptron dargestellt. Weiterhin wird eine Spalte von Perceptrons als *Layer* bezeichnet. Der erste Layer fällt Entscheidungen auf Basis der Eingabewerte, indem er diese abwägt. Jedes Perceptron des zweiten Layers hingegen, wägt für die Entscheidungsfindung die Resultate des ersten Layers ab. Ein Perceptron auf dem zweiten Layer kann somit eine Entscheidung auf einer abstrakteren

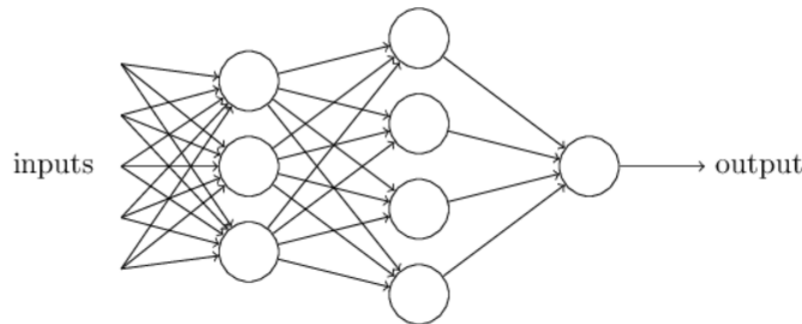


Abbildung 2.2: Unterschiedliche Möglichkeiten der Entscheidungsfindung.

und komplexeren Ebene durchführen. Auf diese Weise kann sich ein vielschichtiges Netzwerk von Perceptrons in ein anspruchsvolles Modell zur Entscheidungsfindung entwickeln.

Im folgenden wird die mathematische Beschreibung von Perceptrons vereinfacht, indem Änderungen an der Notation für  $\sum_j w_j x_j > \text{threshold}$  vorgenommen werden. Für die Beschreibung der Summe  $\sum_j w_j x_j$  werden die Vektoren  $w$  und  $x$  eingeführt, wodurch sich die Schreibweise  $w \cdot x \equiv \sum_j w_j x_j$  ergibt. Des Weiteren wird der `threshold` auf die andere Seite der Ungleichung gezogen und erhält die Bezeichnung *Bias*,  $b \equiv -\text{threshold}$ .

$$\text{output} := \begin{cases} 0 & \text{falls } w \cdot x + b \leq 0 \\ 1 & \text{falls } w \cdot x + b > 0 \end{cases} \quad (2.2)$$

Somit nimmt der Bias Einfluss auf die binäre Ausgabe `output`, weshalb der Bias eine Ausgabe mit einem hohen positiven oder negativen Wert mit einer höheren Wahrscheinlichkeit bedingen kann.

Im vorherigen Abschnitt wurden Perceptrons als ein Methode für die Entscheidungsfindung beschrieben. Ein weiterer Anwendungsfall besteht in der Berechnung von logischen Funktion wie z.B. AND, OR und NAND. Fällt die Betrachtung auf ein Perceptron mit 2 Eingaben deren Gewichtung jeweils den Wert -2 aufweisen und einen Bias von 3, so ergibt sich folgende Abbildung (siehe Abb. 2.3).

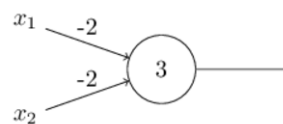
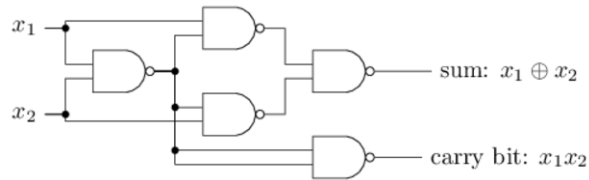


Abbildung 2.3: Percetron mit zwei Eingaben -2 und einem Bias von 3.

Weisen die Eingaben  $x_1, x_2$  den Wert 0 auf, ergibt sich für den `output` den Wert 1. Sind die Eingabewerte für  $x_1, x_2$  -1 ergibt sich für den `output` den Wert -1. Der Aufbau beschreibt somit einen NAND Gatter.

$$\begin{aligned} w \cdot x + b &= \text{output} \\ (-2) * 0 + (-2) * 0 + 3 &= 1 \\ (-2) * 1 + (-2) * 1 + 3 &= -1 \end{aligned}$$

NAND Gatter können verwendet werden, um die unterschiedlichsten Berechnungen durchzuführen. Im Folgenden fällt die Betrachtung auf die Addition von zwei Bits  $x_1$  und  $x_2$ . Für die Berechnung wird die bitweise Summe  $x_1 \oplus x_2$  gebildet, wobei ein `carrybit` den Wert 1 annimmt, sobald  $x_1$  und  $x_2$  gleich 1.

Abbildung 2.4: NAND Gatter mit den Eingaben  $x_1$  und  $x_2$ .

Um ein gleichwertiges Netzwerk abzuleiten, werden die NAND Gatter durch Perceptrons mit jeweils 2 Eingaben ersetzt. Hierbei weisen die Gewichtungen  $w_1, w_2$  den Wert -2 und der Bias  $b$  den Wert 3 auf.

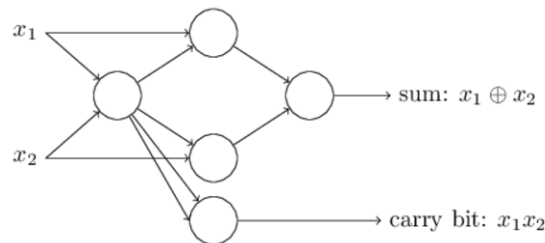


Abbildung 2.5: NAND Gatter Aufbau mit Perceptrons.

In einem weiteren Schritt wird die Abbildung eines NAND Gatter mit Perceptrons vereinfacht. Dazu werden mehrere Eingänge eines Perceptrons zu einem zusammengefasst, weshalb aus den zwei Eingaben -2 der Wert -4 resultiert. Ebenfalls werden die Eingaben in einem sogenannten *Input Layer* zusammengefasst, wobei durch die Notation eine Eingabe nicht mit einem Perceptron gleichzustellen ist.

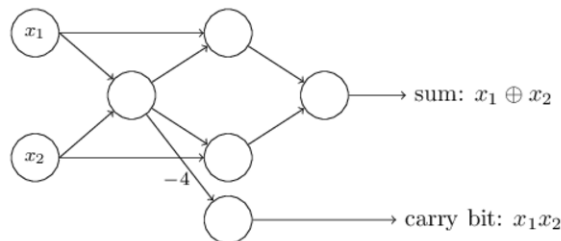


Abbildung 2.6: Vereinfachter NAND Gatter Aufbau mit Perceptrons.

Dieser Anwendungsfall zeigt, dass mit Perceptrons unterschiedliche Berechnungen durchgeführt werden können. Implementierte Lernalgorithmen können Gewichtungen sowie den Bias automatisch durch entsprechende Stimuli im Netzwerk anpassen und ermöglichen die Nutzung von künstliche Neuronen, die sich von herkömmlichen Logik Gattern unterscheiden. Neuronale Netze können somit über einen definierten Zeitraum lernen, wie bestimmte Probleme zu lösen sind.

## 2.4 Sigmoid Neurons

Für die Entwicklung lernender Algorithmen in einem Netzwerk mit Perceptrons, fällt unsere Betrachtung auf das Beispiel der Erkennung von handgeschriebenen Zahlen. Die Eingabe für das Netzwerk könnten die Raw Pixeldaten der eingescannten Bilder darstellen, welche die handgeschriebenen Zahlen abbilden. Das Ziel an dieser Stelle ist, dass das Netzwerk anhand der Veränderungen von *Weights* und



*Biases* lernt eine korrekte Klassifizierung der Zahlen vorzunehmen.

Das Modifizieren der *Weights* und *Biases* kann das Verhalten des Netzwerkes und deren Entscheidungsfindung zu Problemen beeinflussen. Angenommen die Erkennung und Klassifizierung einer Zahl wurde durch das Netzwerk falsch vorgenommen, so können durch kleine Veränderungen an den *Weights* und *Biases* eine Korrektur durchgeführt werden. Dieses stetige Modifizieren der Werte über einen definierten Zeitraum ermöglicht ein lernendes Netzwerk (siehe Abb. 2.7).

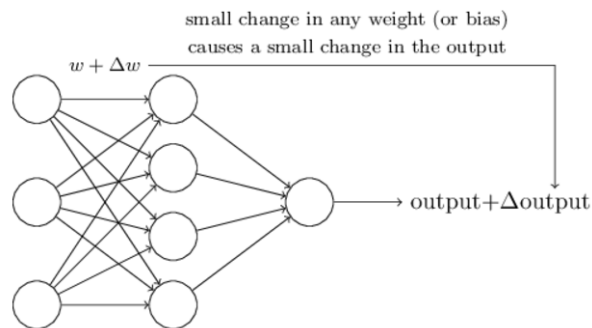


Abbildung 2.7: Modifizieren von Weights und Biases schaffen lernendes Netzwerk.

Dieses wünschenswerte Verhalten eines lernenden Netzwerkes kann durch Perceptrons nicht gewährleistet werden, da eine Veränderung der Weights und Biases den Ausgabewerte *output* eines Neurons umkehren kann. Dadurch kann das komplette Verhalten zur Klassifizierung von Zahlen beeinflusst werden, wobei zuvor falsch erkannte Zahlen nun richtig klassifiziert werden und umgekehrt. Mit Hilfe der Einführung des Sigmoid Neurons soll dieser Fehler behoben werden. Eine Änderung der Weights und Biases bei diesem künstlichen Neuron soll nur marginale Änderungen an dem Ausgabewert *output* vornehmen. Diese Erweiterung des Neurons begünstigt ein Netzwerk selbständig die Klassifizierung von Zahlen zu optimieren.

Der Aufbau des Sigmoid Neurons ähnelt dem Perceptron, wobei das Neuron eine Anzahl von Eingabewerten  $x_n$  mit  $n \in 1, ..n$  entgegennimmt und ausgehend von diesen Informationen den *output* ermittelt. Der wesentliche Unterschied zwischen diesen zwei Typen von Neuronen liegt in dem Wertebereich des *output*.