deepLID

Sprachidentifikation mit Deep Learning

Joel André

Maturaarbeit
Betreuungsperson:
Beni Keller

Kantonsschule Zug
2019

Inhaltsverzeichnis

1 Einleitung

2 Deep learning

Die Begriffe *Deep Learning*, *maschinelles Lernen* und, *künstliche Intelligenz*, werden oft fälschlicherweise auswechselbar verwendet, und in eine Schublade geräumt. Es gibt allerdings eine ganz klare, und für das Verständnis wichtige Hierarchie zwischen den Wörtern. Um Klarheit zu verschaffen werden darum alle Gebiete aufgeführt.

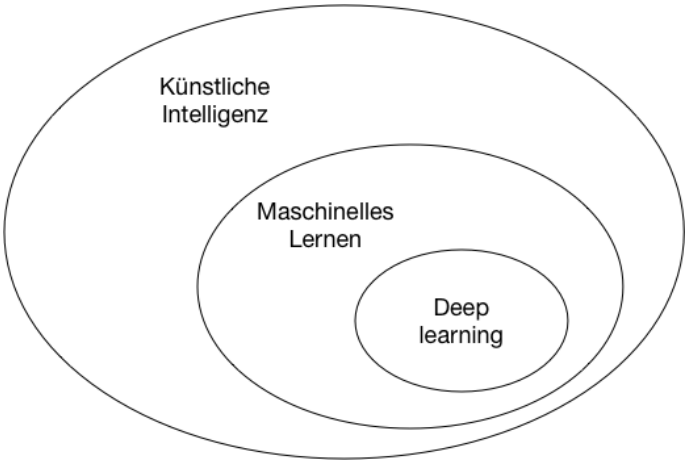


Abbildung 1: Künstliche Intelligenz, Maschinelles Lernen und Deep Learning

Das Gebiet der künstlichen Intelligenz gibt es schon so lange wie den Computer selbst. Die Frage wie schlau ein Computer werden kann, beschäftigt uns bis heute. Al anerkannte Definition für KI gilt, das Bestreben intellektuelle Aufgaben, die normalerweise von Menschen gelöst werden zu automatisieren.

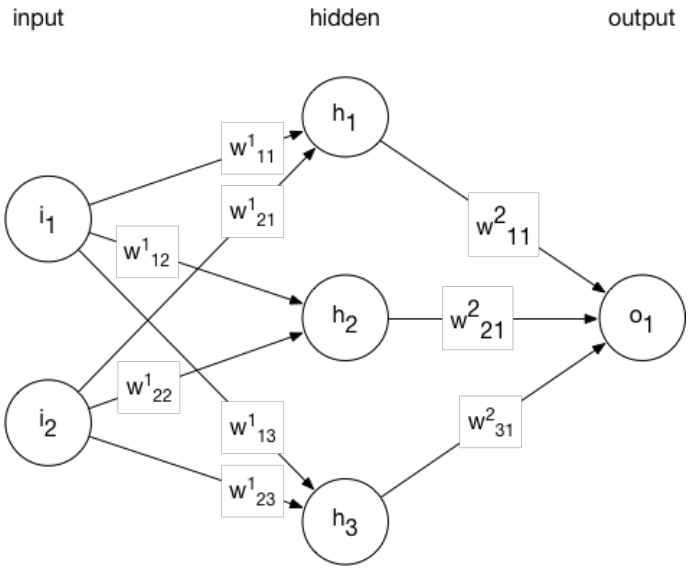
Erste Erfolge erreichte man zum Beispiel mit Schachcomputern, die handgeschriebene Regeln befolgten. Diese Form von künstlicher Intelligenz hatte aber schnell Grenzen, da viele Prozesse schlicht zu komplex waren um sie unter angemessenem Aufwand mit Regeln zu beschreiben. An diesem Punkt wurde man auf maschinelles Lernen aufmerksam.

Der Ablauf von maschinellem Lernen ist grundlegend anders als konventionelles Programmieren. Der Entwickler muss keinen Programmcode mit festen Regeln schreiben, im Gegenteil. Man liefert dem Computer Eingabe und Ausgabe, und der Computer lernt selber die Regeln. Maschinelles Lernen entstand erst so richtig um 1990, wurde aber schnell zum grössten Teilgebiet der künstlichen Intelligenz.

Beim maschinellem Lernen, lernt die Software im Grunde eine nützlichere Darstellungsweise der Daten bzw. der Eingabe. Anhand dieser anderen Darstellungsweise kann der Computer die Antwort einfach erkennen. Wenn der Computer stufenweise nützlichere Repräsentationen bestimmt kann er zunehmend komplexe Probleme in einfachere Zwischenschritten lösen. Genau das ist deep learning. Es beschreibt also mehr das Konzept von stufenweisem Lernen als eine Methode selber. Ein weit verbreitete Mehode sind allerdings *tiefe künstliche neuronalen Netzen*[vgl. 1].

2.1 Künstliche Neuronale Netze

Künstliche neuronale Netze hat man sich, wie der Name schon preisgibt, von der Natur abgesehen. Ähnlich wie in unserem Gehirn gibt es Neuronen (Knoten, z.b o_1) und dazwischenliegende Verbindungen. Die Verbindungen haben ein Gewicht



$$o_1 = \sigma\Big(\sum_i h_i \cdot w^2_{i1}\Big)$$

Abbildung 2: Grafische Darstellung eines künstlichen neuronalen Netzwerks

w. Der Wert eines Knotens ist eine Funktion der Summe all er seiner eingehenden Verbindungen. Diese Funktion wird Aktivierungsfunktion σ genannt. Eine bekannte Aktivierungsfunktion ist zum Beispiel die Sigmoid Funktion: $exp(x) = \frac{1}{1+e^{-x}}$[4]. Die Aktivierungsfunktion ist wichtig damit das Netwerk auch nicht lineare Repräsentationen lernen kann. Es ist bewiesen dass solche neuronale Netze jede Funktion abbilden können[3, Kap. 4].

Die mathematischen Operationen in einem neuronalen Netwerk lassen sich alle als Matrizen-Operationen berechnen. Mit Matrizen kann der Computer auf einer Grafikkarte und mit einem BLAS (*Basic linear algebra system*) sehr effizient rechnen.

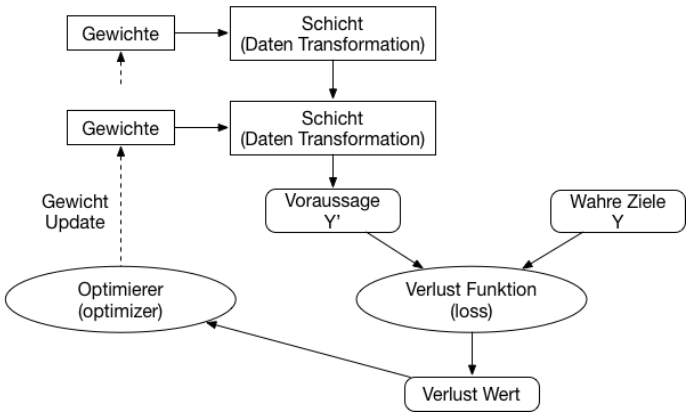


Abbildung 3: Grafische Darstellung eines künstlichen neuronalen Netzwerks

Die Gewichte sind die Parameter. Um diese zu erlernen braucht es eine *Verlust Funktion* die uns angiebt, wie weit die Ausgabe von den korrekten Zielen entfernt ist. Anhand des Verlustwerts passt das Netzwerk die Gewichte schrittweise an. Die-

sen Teil übernimmt der *Optimierer*. Ein einfacher Optimierer ist zum Beispiel das Gradientenverfahren. Da bei diesem Typ von neuronalen Netzwerken alle Knoten miteinander verbunden sind, wird es oft *dense neural network* genannt.

2.2 Convolutional Neural Networks

Convolutional Neural Networks sind eine sehr weit verbreitete Methode im Feld von *Computer Vision*. Der fundamentale Unterschied zwischen dem oben besprochenen *dense network* und einem *CNN* ist, dass ein *CNN* lokale Muster erkennen kann, wo hingegen das vorherige Netzwerk nur globale Muster erkennen konnte. Das heisst, dass ein Muster an einer bestimmten Stelle angetroffen wird, an jeder anderen Stelle ebenfalls erkannt wird. [1]

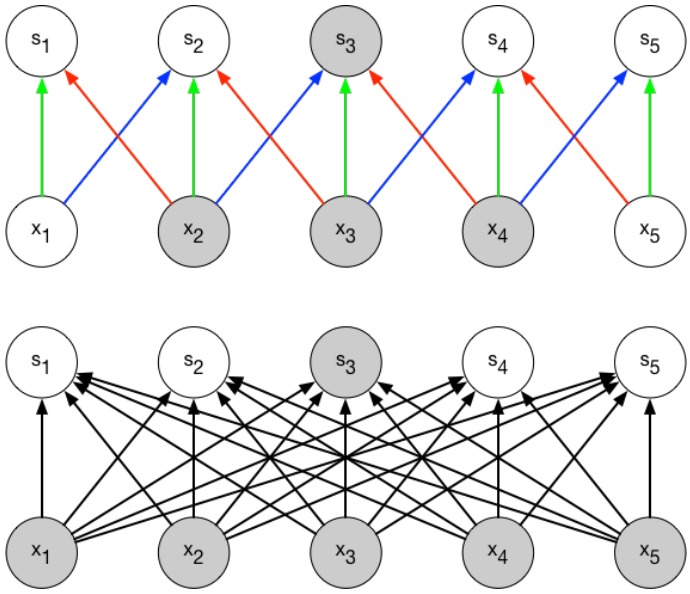


Abbildung 4: (*Oben*)1D Convolution mit *kernel* der Grösse 3. s_3 wird durch 3 inputs beeinflusst. (*Unten*) *Dense Network*. s_3 wird durch alle inputs beeinflusst.[2]

Ein weiterer Vorteil von *Convolutional neural networks* ist, dass sie eine räumliche hyrarchie von Mustern erlernen können. Wenn die Eingabe das Bild einer Katze ist, wird zum Beispiel die erste Schicht unterschiedliche Kanten erkennen, die zweite Schicht dann einzelne Merkmale (z.b Augen), und so weiter.

Damit das gilt, muss aber der analysierte Bereich eines Knotens, von Schicht zu Schicht grösser werden. Deshalb wird meisten nach jedem *convolution layer* ein *pooling layer* gesetzt. Das *pooling layer* fasst mehrere Datenpunkte zusammen um dem nächsten Netzwerk eine grösseren Analysebereich zu verschaffen.

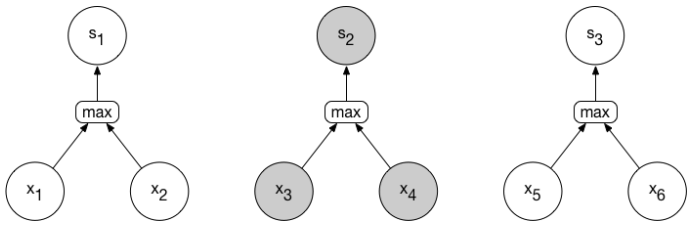


Abbildung 5: Abbildung eines 1D Max-Pooling layer. s_2 ist $\max(x3, x4)$

2.3 Recurrent Neural Networks

2.4 [andere Tricks]

3 Daten

3.1 Daten Auswahl

3.2 Audio Signal Processing

4 Resultate

4.1 Modelle

4.2 Web Interface

5 Diskussion

5.1 Ausbaumöglichkeiten

6 Bibliotheken

6.1 Numpy

6.2 Matplotlib

6.3 Keras

6.4 Flask

Literatur

[1] François Chollet. *Deep learning with Python*. Shelter Island, NY: Manning Publications Co., 2018.

[2] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[3] Michal Nilson. *Neural Networks and Deep Learning*. Dez. 2017. URL: `%5Curl%7Bhttp://neuralnetworksanddeeplearning.com/index.html%7D` (besucht am 21. 07. 2018).

[4] Tariq Rashid. *Neuronale Netze selbst programmieren*. Heidelberg: O'Reilly, 2017.

Abbildungsverzeichnis