Hochschule Reutlingen – Data Mining

Deep Learning

Überblick

Mike Wieder
29.5.2017

Inhaltsverzeichnis

1 Was versteht man unter Deep Learning?

Deep Learning ist eine Teildisziplin von machine learning und basiert auf tiefen neuronalen Netzen. Diese sind dem menschlichen Gehirn nachempfunden und deren Aufbau kann in 3 Schichten, wobei jede davon aus einer Vielzahl von Neuronen besteht, unterteilt werden.

In der ersten Schicht, dem input layer, wird die Dateneingabe, beispielsweiße durch ein Neuron für jeden Pixel eines Bildes vorgenommen. Die Werte der einzelnen Pixel werden ohne Kontext zueinander flach als Linie dargestellt (Siehe Abbildung 1-1). Anschließend werden die Daten an das hidden layer weitergegeben, in welchem die Daten weiterverarbeitet werden. Das hidden layer besteht bei tiefen Netzen aus mehreren Ebenen, welche aus den Eingabedaten immer weitere Informationen erkennen bis der output layer erreicht ist, welcher das Ergebnis der Auswertung liefert und eine Vorhersage trifft. Jedes Neuron im output layer liefert einen Wert zwischen 0 und 1, welcher die Wahrscheinlichkeit darstellt, dass dieses Neuron die richtige Antwort darstellt. Die Summe aller Ausgabewerte ist 1.

Bei dieser Art von neuronalen Netzen handelt es sich um ein Deep feedforward neural network, da die Neuronen immer nur nach vorne kommunizieren.

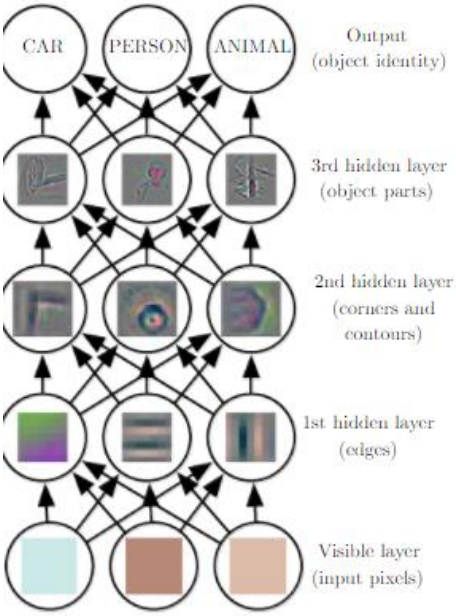

Abbildung 0-1 - Aufbau eines neuronalen Netzes

```
sports car, sport car (score = 0.89596)
grille, radiator grille (score = 0.01607)
convertible (score = 0.01372)
car wheel (score = 0.00793)
passenger car, coach, carriage (score = 0.00246)
```

Abbildung 0-2 - Ergebnis einer Bildanalyse

Jedes Neuron des hidden layers führt dieselbe Aufgabe durch und summiert alle anliegenden gewichteten Werte der vorherigen Neuronen auf und addiert eine Konstante(Bias), was dann an eine nichtlineare Funktion, auch activation function genannt (z.B. Softmax, Sigmoid), weitergegen wird. Das Resultat wird wieder an die darauffolgenden Neuronen weitergegeben und der Vorgang wiederholt sich bis das output Layer erreicht ist.

Das Netz muss davor jedoch trainiert werden, wobei Lerndaten, bei denen der Zielwert bekannt ist, durch das Netz laufen und die Gewichte und der Bias angepasst werden. Dies geschieht mit Hilfe des Gradiantenverfahrens. Bei jedem Schritt des Lernvorgangs entwickelt das Netz eine Vorhersage, zu welchem die Parameter je nach Abweichung zu der richtigen Lösung angepasst werden.

Deep Learning kann dazu verwendet werden um Aufgaben, welche einfach von Menschen gelöst werden auch durch Maschinen zu lösen. Dazu gehören beispielsweiße Schrift-, Bild- und Spracherkennung oder selbstfahrende Fahrzeuge.

2 Geschichtliche Entwicklung von Deep Learning

Der erste Deep Learning Algorithmus wurde bereits 1965 von Ivakhenko und Lapa vorgestellt. 1979 wurde von Fukushima das erste convolutional neural network vorgestellt, welches die Eingabewerte zuerst nach nützlichen Informationen filtert und so Muster erkennen kann.

In den 80er Jahren wurde weiter an dem Verbessern des Einlernens der Netze, vor allem an der Verbesserung des Gradiantenverfahrens, geforscht.

Die erste praktische Anwendung wurde 1989 von LeCun entwickelt und wurde dazu benutzt um handgeschriebene Ziffern zu erkennen. Sie basierte auf einem convolutional neural network. Das Trainieren des Netzes dauerte jedoch 3 Tage und die Gewichte der Neuronen mussten teilweise manuell gestellt werden, weshalb es nie praktisch eingesetzt wurde.

Durch die hohen Rechenkosten und der fehlende Erfolg der bisherigen Forschung wurde Deep Learning in den 1990er und 2000er nur wenig gefördert und verwendet.

Durch die Weiterentwicklung von GPUs und dadurch die Möglichkeit wesentlich größere Trainingsdatensätze zu verwenden ist Deep Learning seit 2009 eines der wichtigsten Forschungsthemen im Gebiet der künstlichen Intelligenz. Fortschritte in der Forschung des menschlichen Gehirns helfen ebenfalls bei der Entwicklung neuronaler Netze.

3.0 Grundbegriffe

Um die Funktionsweise zu verstehen müssen zuerst die Grundbegriffe und Funktionen geklärt werden. Dazu gehört die Funktionsweise und der Aufbaue der einzelnen Bestandteile eines neuronalen Netzes.

3.1 Neuronen

Neuronen sind der wichtigste Bestandteil eines neuronalen Netzes. Ihre Aufgabe ist es eine Reihe von Eingabewerten zu verarbeiten und auf einen einzelnen Ausgabewert abzubilden. Neuronen werden nach der Art der verwendetet activation function (Siehe 3.4) unterschieden.

ine spezielle Art von Neuron ist der Perzeptron, dieser bildet binäre Eingabedaten auf einen binären Ausgabewert ab. Der Nachteil hiervon ist, dass eine kleine Veränderung der Eingabedaten das Ergebnis komplett von 0 zu 1 ändern kann. Bei neuronalen Netzen ist dieses Verhalten nicht erwünscht, weshalb Neuronen verwendet werden die mehrere Ausgabewerte liefern können.

3.2 Gewichte

Bei den Gewichten handelt es sich um Stellschrauben eines jeden Neurons, die das Netz zum Einlernen von Aufgaben benötigt. Jedes Neuron besitzt für jeden Eingabewert ein separates Gewicht. Durch den Wert dieser Gewichte können nutzvolle von nutzlosen Informationen unterschieden werden. Sollte beispielsweiße die Aufgabe eines Neurons sein ein Stoppschild zu erkennen und die Eingabewerte stellen unterschiedliche Farben dar, so ist nur die Information relevant ob das Schild rot ist. Das Neuron, das angibt ob das Schild rot ist oder nicht würde eine höhere Gewichtung bekommen als die anderen Neuronen.

3.3 Bias

Der Bias ist eine weitere Stellschraube eines jeden Neurons, im Gegensatz zu den Gewichten jedoch besitzt jedes Neuron nur einen Bias. Der Bias ist eine Konstante, die auf alle Eingabewerte aufaddiert wird.

Sollte von einem Neuron beispielsweiße der Ausgabewert ~1 erwünscht sein falls alle Eingabewerte schwarz sind (bei RGB-Kodierung = 0) so ist dies ohne die Verwendung eines Bias nicht möglich (siehe Abbildung 3.1-1). Der Ausgabewert würde immer 0.5 sein, da alle Eingabewerte 0 sind. Durch das addieren eines Bias können wir den Ausgabewert näher an die geforderte 1 bringen.

3.4 Activation function

Die activation function ist ein zentraler Bestandteil eines jeden Neurons des hidden layers und die Wahl der richtigen Funktionen ist von großer Bedeutung für die Leistung des Netzes.

Die activation function dient dazu, eine Reihe von Eingabewerten auf einen Zielbereich abzubilden. Die Eingabewerten bestehen aus den Werten der Neuronen des vorherigen Layers inkl. der Gewichtungen und dem Bias.

Das Besondere daran ist, dass es sich um eine nichtlineare Funktion handelt, dadurch wird das gesamte Netz nichtlinear was zu einer höheren Vielzahl an möglichen Abbildungen führt. Ohne die Funktion würden die Neuronen lediglich lineare Transformationen durch das weitere Multiplizieren mit den Gewichten und dem Bias durchführen. Dies ist jedoch nicht ausreichend um komplexere Aufgaben zu lösen.

Eine activation function hat die Form

$$y=\sigma(Wx+b)$$

wobei W=Gewichte, x=Eingabewerte und b=Bias.

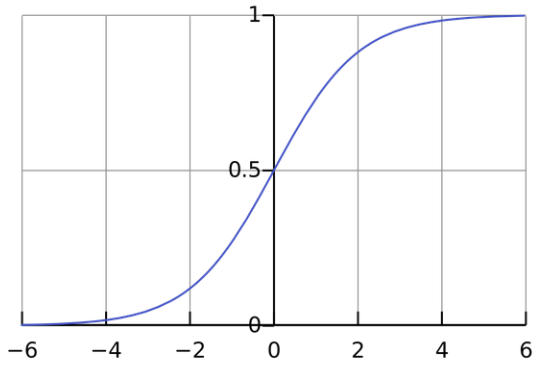

Abbildung 3.4-1 - Softmax function

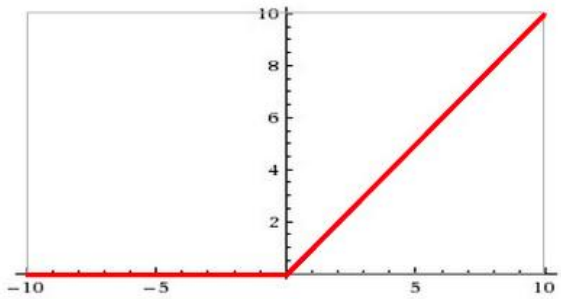

Abbildung 3.4-2 - RELU function

3.5 Gradiantenverfahren

Das Gradiantenverfahren ist ein wichtiger Bestandteil bei dem Lernen eines neuronalen Netzes. Davor müssen die vom Netz erzeugten Vorhersagen mit dem richtigen Ergebnis verglichen werden indem der Abstand (z.B. Cross-Entropy) dazwischen berechnet wird. Das Ziel des Gradiantenverfahrens hierbei ist es diesen Abstand zu minimieren.

Der Abstand ist von den Gewichten, dem Bias, sowie den Eingabewerten abängig. Der Gradiant zeigt „bergab“, bis ein lokales Minimum erreicht wird und somit in die Richtung des geringsten Abstandes zwischen der erzeugten und der richtigen Lösung zeigt. Die Gewichte und der Bias werden dementsprechend angepasst.

Es wird jedoch immer nur ein gewisses Stück weit dem Gradianten gefolgt (Lernrate), da man ansonsten häufig die Werte über das Minimum hinaus anpasst und es somit nie erreicht. Über die Lernrate kann somit die Lerndauer des Netzes und die erreichte Präzision eingstellt werden.

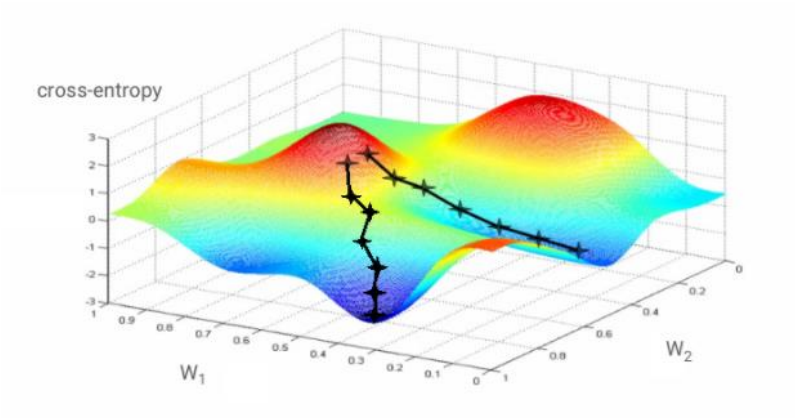

Abbildung 3.5-1 - Cross Entropy als Funktion von 2 Gewichten

3.4.1 Vanishing gradient problem

Ein Problem bei tiefen neuronalen Netzen ist, dass je näher die hidden Layer am input Layer liegen sie einen starken Verfall des Lernerfolgs verzeichnen müssen, je länger das Training durchgeführt wird. Dies liegt daran, dass der Gradiant sich durch die Multiplikation aller Ableitungen der darauffolgenden activation functions berechnet. Da diese in der Regel |>1| sind wird der Gradient immer kleiner und der Lernerfolg sinkt somit auch immer weiter.

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

$$\frac{\partial C}{\partial b_3} = \overbrace{\sigma'(z_3)\,w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

Oben ist ein Gradiant des ersten hidden layers zu sehen und unten ein Gradiant des dritten hidden layers.

3.6 Convolutional neural network

Ein Convolutional neural network funktioniert vom Aufbau her wie der visuelle Cortex des Menschen. Hierbei liegt an den Neuronen des ersten ersten hidden Layers nichtmehr jedes einzelne input Neuron an, sondern ein gewisser Bereich. Die Eingabewerte behalten hierbei auch ihre ursprüngliche Form bei und werden nichtmehr umgeformt dargestellt.

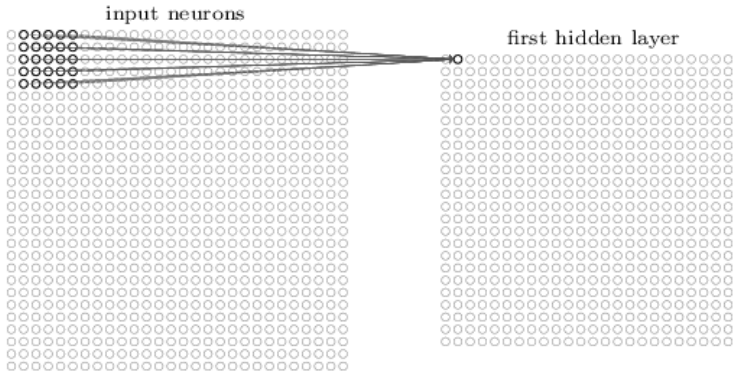

Abbildung 3.6-1 - Aufbau eines CNN

Der Vorteil hiervon ist, dass gewisse Strukturen und Zusammenhänge der Eingabewerte nichtmehr verloren gehen. Die Gewichte und der Bias der hidden Neuronen werden hier geteilt. Jedes hidden Neuron besitzt für jede Verbindung zu den Input Neuronen dieselben Gewichte und denselben Bias. Sollten die Neuronen 25 Inputs erfassen besitzen sie somit 25 Gewichte.

Alle Neuronen werden somit darauf eingestimmt bestimmte Muster (z.B eine Kante in einem Bild) zu erkennen. Um mehrere Unterschiedliche Muster zu erkennen können mehrere Schichten verwendet werden. Diese Schichten bezeichnet man als convolutional feature Layer

Daraufhin folgen ein oder mehrere Pooling Layer, welche die Informationen aus dem feature Layer auf einen kleineren Bereich zusammenfassen, sie fassen beispielsweiße ein 2x2 großen Bereich in einem einzelnen Neuron zusammen.

Am Ende werden alle Neuronen des Pooling Layers mit allen Neuronen des Output Layers verbunden und es wird wie gewohnt eine Vorhersage getroffen.

4 Tensorflow

Tensorflow ist eine von Google entwickle Bibliothek für machine Learning mit dem Schwerpunkt auf Sprach- und Bilderkennung durch neuronale Netze. Sie basiert auf Pyhton und C++, bietet jedoch auch eine API für Java, Go und einige weitere Sprachen.

Zusätzlich wird mit Tensorboard eine einfache Visualisierung der erzeugten Zwischen- und Endergebnisse angeboten.

5 Quellen

http://neuralnetworksanddeeplearning.com/

https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-history-training/