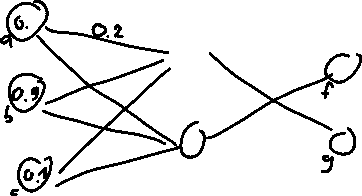
Kurzzusammenfassung neuronale Netze.

Ein neuronales Netzwerk ist eine Variante des machine learning. Es geht dabei darum mithilfe von schon gelösten Beispielen den Computer selbst bestimmte Regeln oder Muster zu lernen, damit er ähnliche Aufgaben ohne Hilfe lösen kann. Ein neuronales Netz bedient sich dabei als Inspiration beim Gehirn von Lebewesen. Die Details sind nicht annähernd biologisch korrekt. Es hat sich jedoch eine korrespondierende Nomenklatur durchgesetzt. Ein neuronales Netz besteht aus Neuronen und Axonen, welche die Neuronen miteinander verbinden. Ein Neuron repräsentiert dabei einfach eine Zahl zwischen null und eins. Das Neuron gibt über die Axonen ein Signal, dieser vorher erwähnte Wert zwischen 0 und 1, dem nächsten Neuron weiter. Das Netz ist in Schichten aufgebaut wobei klassischerweise ein Neuron mit allen Neuronen aus der nächsten Schicht verbunden ist. Die erste Schicht repräsentiert die Eingabe. Im Falle eines Bildes währen das zum Beispiel die Helligkeitswerte jedes Pixels des Bildes. Nun wird eine sogenannte Aktivierungsfunktion auf diese Neuronen angewendet. Es gibt dabei nicht eine einzige Funktion. Alle Aktivierungsfunktionen, haben jedoch eine wichtige Charakteristik. Ihre Ausgabe ist nämlich bei jeglicher Eingabe zwischen null und eins. Danach wird das Signal den Axonen entlang weitergeleitet. Jedem Axon wird dabei ein Gewicht zugeordnet, welches mit dem ursprünglichen Signal multipliziert wird. Beim nächsten Neuron werden die Signale aller Axonen, die zu diesem Neuron führen aufsummiert. Dann wird auf diese Werte wieder die Aktivierungsfunktion angewendet und dann wird das Signal an die nächste Schicht weitergegeben. Um das Ganze hoffentlich etwas verständlicher zu machen habe ich hier eine Skizze eines eingefügt, um zu zeigen, wie das Signal von einem Neuron zum nächsten weitergegeben wird. Das Konzept einer Aktivierungsfunktion ist schwierig zu zeichnen. Sie würde jedoch auf die resultierende Summe von 0.61 angewendet.



Wir haben also die Gleichung:

0.3 \* 0.2 + 0.9 \* 0.5 + 0.1 \* 1 = 0.61

Die letzte Schicht ist die sogenannte Ausgabeschicht. Die Werte in dieser Schicht repräsentieren, welche Lösung Anfangs gestellten Problem der Computer mit welcher Wahrscheinlichkeit für richtig hält.

Damit das Netz aber irgendetwas lernen kann, muss zuerst einmal ein System gefunden, welches festlegt wie gut oder schlecht eine Ausgabe des Netzes ist. Dies ist die sogenannte Fehlerfunktion. Sie bestimmt wie schlecht das Netz abgeschnitten hat. Dies tut sie indem sie die Unterschiede zwischen der Aufgabe jedem Neuron mit der erwarteten Ausgabe dieses Neurons vergleicht. Oft wird das Quadrat dieser Differenz benutzt. Es ist wichtig zu realisieren, dass die Fehlerfunktion einen Fehler für jeden Neuron einzeln ausgibt. Aufgrund dieses Fehlers können nun die Gewichte aller Axonen, die zu diesem Neuron führen, angepasst werden. Mit der Backpropagation wird nun für jeden Neuron auch in den versteckten Schichten ein Fehler bestimmt. Dieser Fehler geht ähnlich wie ein normales Signal durch das Netz einfach in Gegenrichtung. Die Fehlerfunktion stoppt vor der letzten Ebene, da von dort die Eingabe kommt und diese sowieso immer variiert. Nun werden die Gewichte angepasst. Es gilt dabei, dass der Fehler jedes Knoten minimiert werden soll. Wie die Gewichte angepasst werden sollen, um diese Idealkonfiguration zu erreichen, wird das Gradienten Verfahren angewendet. Es wird also diese Fehlerfunktion als tatsächliche Funktion angeschaut und dann wird die Ableitung dieser Funktion bestimmt. Dies ist die mathematisch komplizierteste Operation im Netz, da es sich bei der Fehlerfunktion um die Verschachtelung verschiedener Prozesse handelt. Ich werde hier nicht weiter auf die dahinterliegende Mathematik eingehen, da sie schwierig ohne Formeln zu beschreiben ist und das Darstellen von Formeln auf Word nicht besonders angenehm ist. Wichtig ist zu realisieren, dass wir mit diesem Prozess wohl nie einen perfekten Zustand erreichen., da die Gewichte proportional dazu angepasst werden, wie stark die Fehlerfunktion sinkt bei diesen Anpassungen. Um dem Programm das Lösen breiterer Aufgaben beizubringen wird der Durchschnitt der Fehler mehrerer Trainingsbeispiele benutzt. Dadurch wird verhindert, dass das Netz nur ganz spezifische Dinge erkennt.