Herzlich Willkommen zum Abschnitt Theorie der Neuronalen Netzwerke!

Da wir uns nun ein paar elementare Eigenschaften von Machine Learning angeschaut haben, sind wir ready to go um uns die Theorie hinter Neuronalen Netzwerken anzuschauen.  
Denn dieses Wissen ist essenziell um diese in Tensorflow zu erstellen.  
  
Dabei ist der Abschnitt unterteilt, in einerseits einem praktischen Minimalbeispiels.  
Nämlich dem Perzeptron oder auch Rosenblatt-Neuron.  
Dieses funktioniert ähnlich zu unserer Linearen Regression.  
An diesem werden wir uns die elementaren Bestandteile anschauen und in den folgenden Videos näher auf diese eingehen.  
  
Das heutige Thema ist aber, ein paar Begrifflichkeiten und Limitierungen zu benennen.  
Wie ich bereits schon früher erwähnt habe, gehören Neuronale Netzwerke zur Klasse der Supervised Learning Methods.  
Dort hatten wir ja noch eine weitere Unterteilung in Regression oder Klassifikation.  
Ein Neuronales Netzwerk kann beide Aufgaben erfüllen, also je nach Problemstellung.  
  
Ein Neuronales Netzwerk besteht aus (möglicherweise) mehreren Neuronen und noch weiteren Bestandteilen, die wir im nächsten Video sehen werden.  
Diese Art des Netzwerkes wurde von dem vereinfachten Verständnis unseres Gehirns abgeleitet.  
Das Grundprinzip ist, dass Neuronen Datenempfangen (Reize) und ab einen gewissen Schwellwert diese Daten weitergeben bzw. „feuern“. Dieses Prinzip wollen auch wir in unserem Modell übernehmen.

Also müssen wir irgendwie Werten von anderen Neuronen erhalten, dann schauen ob die Summe groß genug ist, um sie weiterzugeben und am Ende des Modells haben wir dann einen weitergereichten Wert und anhand dessen können wir dann ein Beispiel Klassifizieren bzw. Einen durch die Regression berechneten Wert erhalten.  
  
Was wir dabei unbedingt beachten müssen, alle Trainings- und Testdaten unseres Neuronalen Netzwerkes müssen die gleiche Dimension haben UND bei allen Beispielen die dann in der späteren Anwendung auftauchen ebenfalls.  
Dazu ein kleines Beispiel: Wir haben einen 2D Vektor als Input und eine Reelle Zahl als Output.  
Also müssen alle Beispiele 2 Input Werte haben und man erhält immer einen Output Wert.  
Das muss einem vorher klar sein.