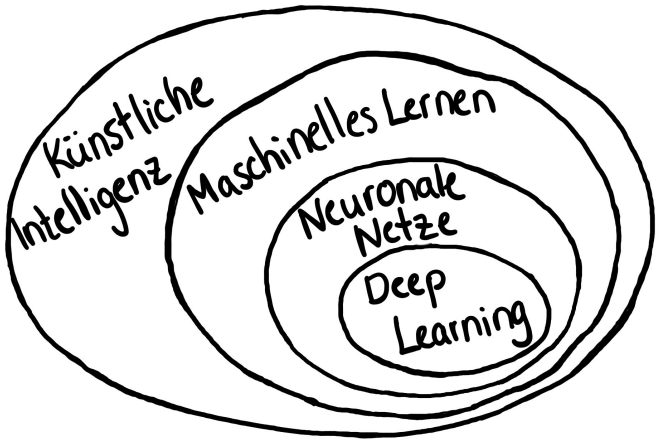
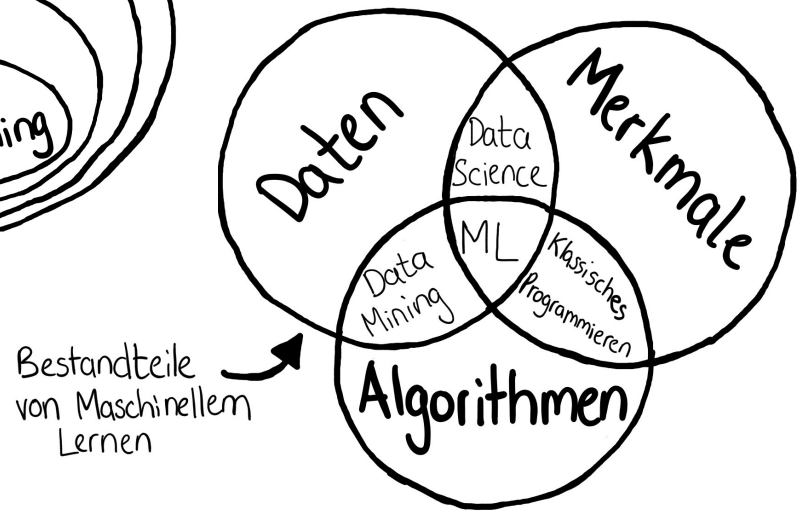
Maschinelles Lernen

Maschinelles Lernen ist das Gewinnen von **Wissen** aus **Erfahrung**. Wie wir Menschen auch, lernt eine Maschine aus Beispielen. Man spricht davon, dass ein **Algorithmus** mit Hilfe von **Trainingsdaten** in der **Lernphase** ein statistisches **Modell** erstellt.





Maschinelles Lernen kann man in die Bereiche **klassisches** maschinelles Lernen, **Reinforcement Learning** und **künstliche neuronale Netze** teilen. Im nächsten Kapitel findest du eine große Übersicht zu Algorithmen, die man dem maschinellem Lernen zuordnen kann.

klassisches maschinelles Lernen

Man spricht von klassischem maschinellen Lernen, weil die gemeinten Algorithmen oft schon sehr lange Bestand haben oder auf alten mathematischen Formeln basieren. In den folgenden Kapiteln werden wir uns hauptsächlich mit diesen Algorithmen auseinandersetzen.

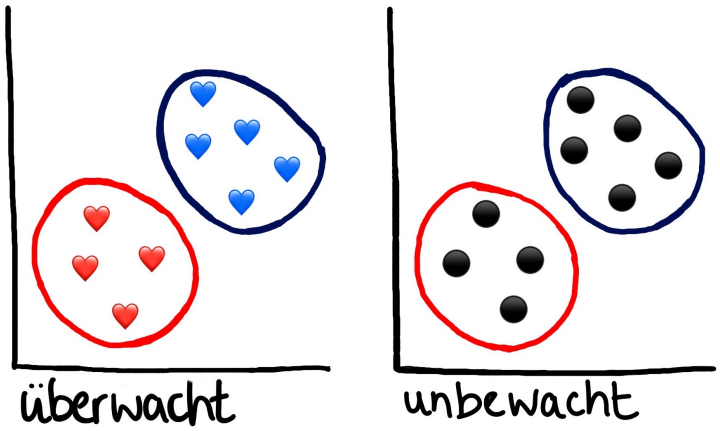
Reinforcement Learning

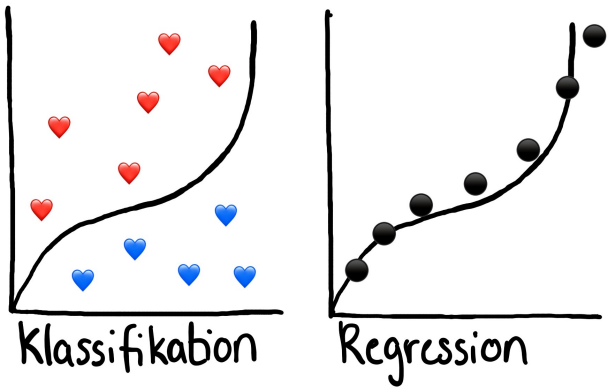
Reinforcement Learning kann man sich wie Hundetraining vorstellen. Während der Lernphase wird die Maschine belohnt, wenn sie etwas richtig macht und bestraft, wenn sie etwas falsch macht. Dabei ist die Maschine wie ein Hund auch darauf aus, die Belohnung zu maximieren und lernt somit richtig zu handeln.

künstliche neuronale Netze

Künstliche neuronale Netze sind der Vernetzung von Neuronen im Nervensystem eines Lebewesens nachempfunden. Die Grundlagen von KNNs existieren schon sehr lange. Allerdings haben sie in den letzten Jahren an großer Bedeutung gewonnen. In einem der folgenden Kapitel findest du mehr über sie heraus.

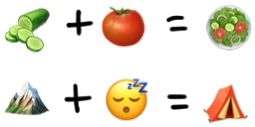
Beim maschinellen Lernen unterscheidet man zwischen **überwachtem** und **unbewachtem** Lernen. Dabei kann man sich das überwachte Lernen so vorstellen, dass die Maschine einen Lehrer vor sich hat, der ihr sagt, was richtig und was falsch ist. Der grundlegende Unterschied liegt dabei in den Daten, die die Maschine erhält. Beim überwachten Lernen erhält die Maschine die richtigen Antworten zu jedem Datenpunkt. Diese Antwort nennt man **Label** oder **Kategorie**. Da Menschen in der Realität allerdings auch nicht auf alles immer eine Antwort haben, oder es zu aufwendig ist, eine richtige Antwort zu vielen vielen Datenpunkten zu geben, existieren auch Algorithmen für das unbewachte Lernen.

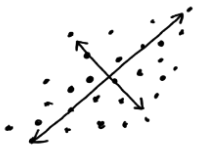


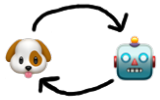


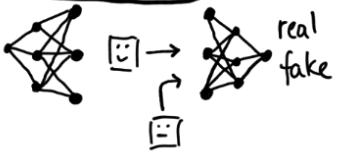
Überwachtes Lernen kann man wiederum in **Klassifikation** und **Regression** unterteilen. Hier liegt der Unterschied im Ergebnis. Bei der Klassifikation sind die Antworten, die die Maschine erhält, **Kategorien**. Ein typisches Beispiel ist die Unterteilung von E-Mails in Spam und Nicht-Spam.
Bei der Regression sind die Antworten **Zahlen**, zum Beispiel der Preis einer Aktie. Anhand von vergangenen Preisen einer Aktie könnte eine Maschine durch einen Regressions-Algorithmus den zukünftigen Preis einer Aktie **vorhersagen**.

Übersicht ML







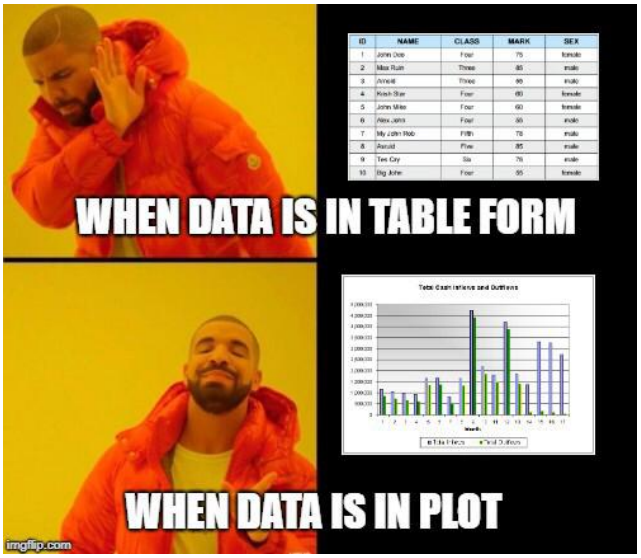


Naive Bayes
KNN
SVM
Logistische Regression
Klassifikation
Lineare Regression
Polynomiale Regression
Regression
überwacht
Mustersuche
Dimensionsreduktion
unüberwacht
Clustern
klassisches Lernen
Reinforcement Learning
Maschinelles Lernen
Stacking
Ensemble Methoden
Bagging
Boosting
Perzeptron (MLP)
künstliche neuronale Netze
erzeugende gegnerische Netze (GAN)
real
fake
Faltungsnetze
Autoencoder

Daten

Daten sind neben den Algorithmen der wichtigste Bestandteil für ein erfolgreiches Modell beim maschinellen Lernen. Man spricht in dem Zuge von ***„Garbage in, Garbage out“*** (Müll rein, Müll raus). Auch der beste Algorithmus wird keine guten Ergebnisse liefern, wenn die eingegebenen Daten schlecht sind. Deswegen ist es wichtig die Daten genau zu verstehen und gegebenenfalls vor der Lernphase aufzubereiten. Auch die Wahl des Algorithmus ist abhängig von der Datenlage. Im Folgenden lernst du alles über die **Datenanalyse** und **Datenaufbereitung**.

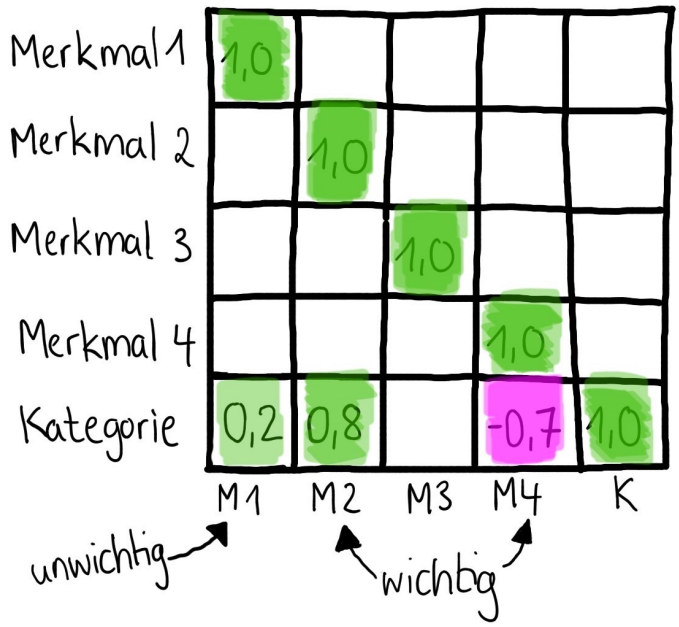
Daten können verschiedenste Formen haben. Wir werden uns hauptsächlich textbasierte Daten anschauen. Diese kann man sich vorstellen wie eine Excel-Tabelle. Daten können allerdings auch in Form von Bildern oder Audio bestehen.
Egal, welche Form ein Datensatz besitzt, ein Datensatz ist immer eine Sammlung Datenpunkten und für jeden Datenpunkt existieren dieselben **Merkmale** (*engl. features*). Um beim Beispiel der Excel-Tabelle zu bleiben, ist ein Datenpunkt eine Zeile in der Tabelle. Jedes Merkmal wird über eine Spalte repräsentiert, wobei die erste Zeile der Tabelle einem zumeist sagt, wie das Merkmal heißt. Bei überwachtem Lernen ist eine Spalte der Tabelle nicht ein Merkmal, sondern die



Weil wir uns häufig viele 100 Datenpunkte anschauen müssen, ist es nicht immer sinnvoll sich die Daten in Tabellenform anzuschauen. Um Daten zu **visualisieren**, gibt es verschiedene Techniken, die wir uns jetzt anschauen.

Heatmap

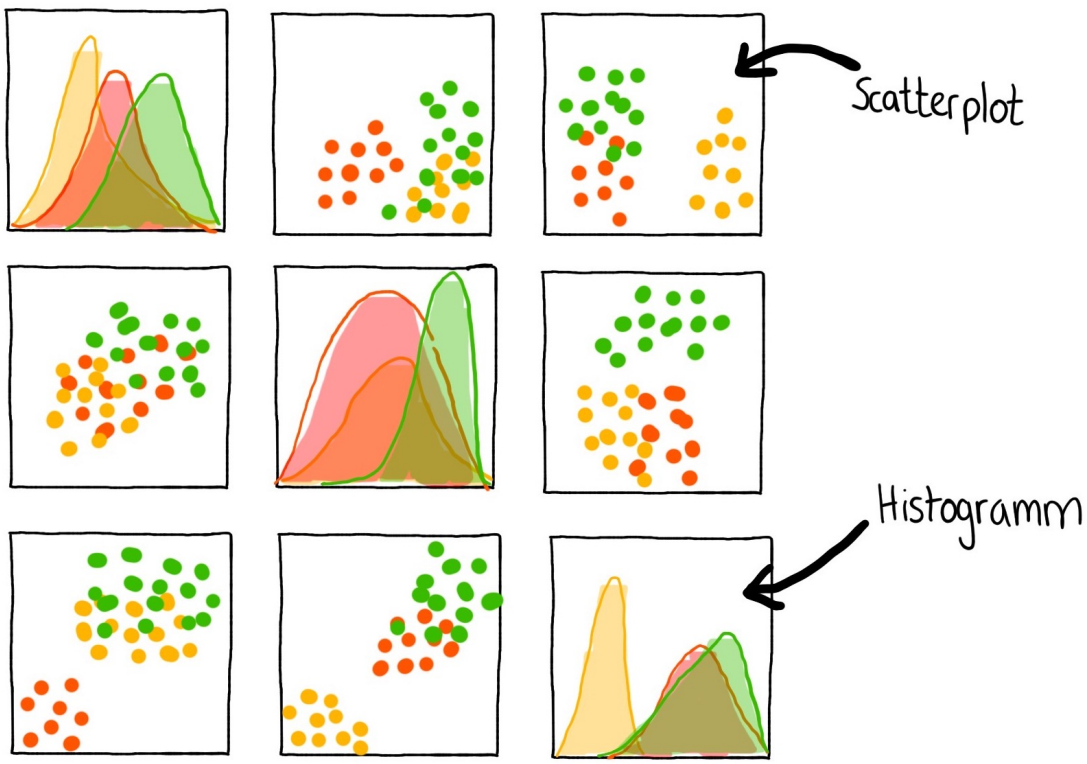
Die Heatmap ist ein beliebtes Werkzeug, um sich die **Datenkorrelation**, also den Zusammenhang beziehungsweise die Wechselwirkung zwischen den verschiedenen Merkmalen und der Kategorie, anzuschauen.



Ist ein Wert in der Heatmap sehr hoch **(größer als 0,5)**, dann bedeutet das, dass wenn der Wert des einen Merkmals steigt, dann steigt auch der Wert des anderen Merkmals (oder der Kategorie). Ist ein Wert sehr niedrig **(negativ, kleiner als -0,5)**, dann sinkt der Wert des einen Merkmals, wenn der Wert des anderen Merkmals steigt. Je größer die Wechselwirkung zwischen Merkmalen und Kategorie ist, desto wichtiger sind die Merkmale zumeist.

Pairplot

Ein weiteres wichtiges Werkzeug, was einem helfen kann die Daten besser zu verstehen, ist der Pairplot. Neben dem Zusammenhang zwischen Merkmalen, zeigt uns der Pairplot auch die Verteilung von einzelnen Merkmalen.

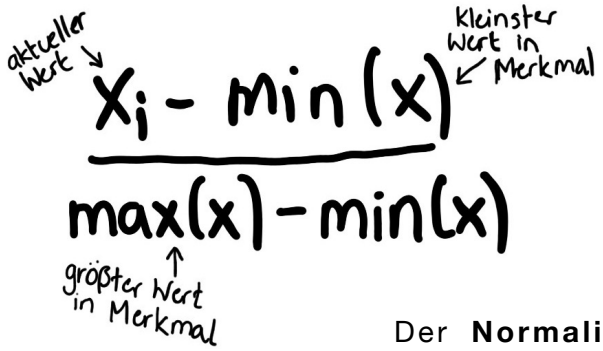


Das **Histogramm** zeigt die Verteilung der einzelnen Klassen. Im Bild im mittleren Histogramm kann man beispielsweise ablesen, dass alle Datenpunkte in der grünen Klasse einen sehr hohen ähnlichen Wert haben, während die Datenpunkte in der grünen Klasse für das dritte Merkmal (unteres Histogramm) zwar auch meist einen hohen Wert haben, allerdings liegen die Datenpunkte wohl verstreuter in dieser Klasse und haben dementsprechend nicht immer einen ähnlichen Wert für Merkmal drei.

Ein **Scatterplot** zeigt uns wie in der Heatmap auch schon die Korrelation zwischen den Merkmalen. Dazu können wir beim Blick auf den Pairplot schon erste Überlegungen machen, wie sich die Klassen gut separieren lassen könnten oder, wo sich im Falle eines Regressionsproblems eine Regel abzeichnet.

Daten skalieren

Oftmals bewegen Datenpunkte sich in einem großen Wertebereich. Manche Algorithmen, die mit der Distanz zwischen Datenpunkten arbeiten, können mit solchen Daten nur sehr schlecht arbeiten, weswegen das Skalieren beziehungsweise Normalisieren der Daten nicht ausbleiben darf. Dafür lernen wir drei Techniken kennen.



Der **Min-Max-Skalierer** ist die einfachste Methode, um Daten zu skalieren. Dabei wird der größte und kleinste Wert von allen Datenpunkten genommen und die nebenstehende Rechnung für jeden einzelnen Datenpunkt (xi) durchgeführt. Damit landen alle Datenpunkt in den Wertebereichen [0, 1] oder [-1, 1], je nachdem, ob der kleinste Wert negativ war.

Der **Normalisierer** funktioniert ähnlich wie der Min-Max-Skalierer. Jedoch ziehen wir hier den Mittelwert aller Datenpunkte von jedem Datenpunkt ab.

$$\frac{x_i - \text{mittelwert}(x)}{\max(x) - \min(x)}$$

$$\frac{x_i - \text{mittelwert}(x)}{\text{varianz}(x)}$$

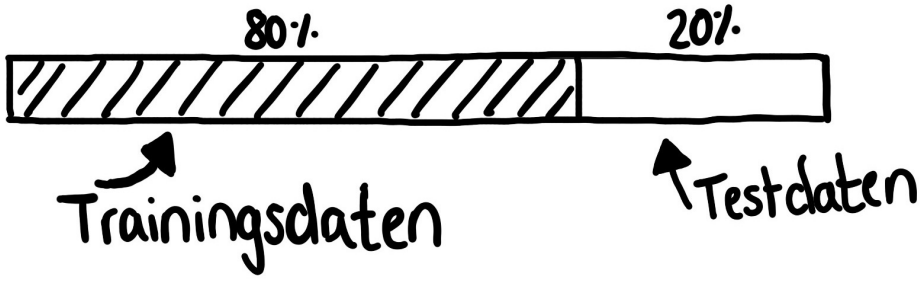
Auch die **Standardisierung** macht sich den Mittelwert zu Nutze. Jedoch teilen wir hier nicht durch die Differenz von größtem und kleinstem Wert, sondern durch die Varianz (auch Streuung) von allen Datenpunkten.

Während sich beim Min-Max-Skalierer nur der Wertebereich ändert, werden die Daten bei den anderen beiden Werkzeugen normalisiert beziehungsweise standardisiert, was den Wertebereich zu einer Art Normalverteilung werden lässt.

Trainingsdaten & Testdaten

Bevor es mit der Auswahl der Algorithmen und der Lernphase losgehen kann, müssen wir noch einen entscheidenen Schritt machen – die Daten müssen geteilt werden. Unser Datensatz muss für die Lernphase in **Trainings- und Testdaten** unterteilt werden. Das machen wir, weil wir am Ende einschätzen wollen, wie gut das trainierte Modell tatsächlich funktioniert. Dafür müssen wir dem Modell in der Lernphase einen Teil unserer Daten **vorenthalten**. Diesen Teil der Daten, den wir Testdaten nennen, geben wir dem Modell dann nach der Lernphase. Anhand der Antworten des Modells und der korrekten Antworten, die wir auch für die Testdaten haben, können wir dann schauen, wie oft das Modell richtig lag.

Dabei müssen wir darauf achten, dass das Modell immer noch genug Daten für das Training hat. Deswegen enthält man dem Modell meist nur einen kleinen Teil der Daten vor.

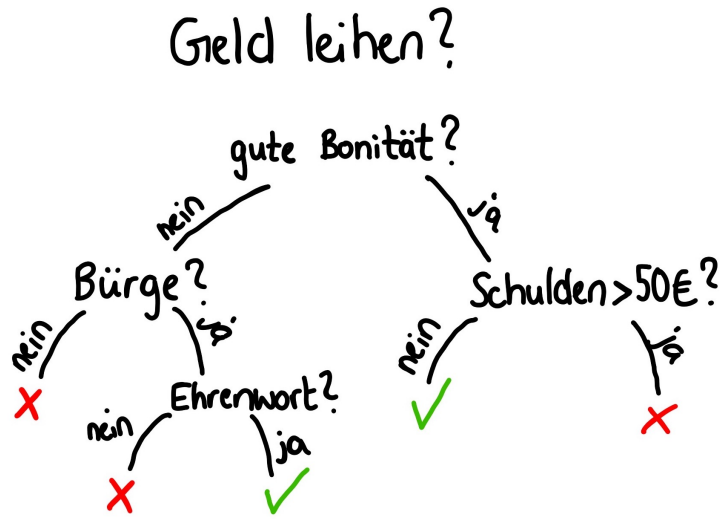


Klassifikation

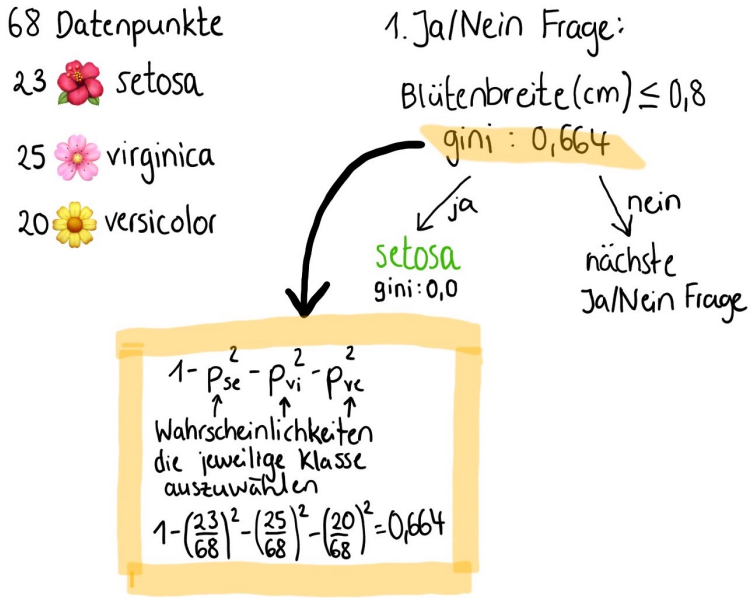
Im Folgenden lernst du alle Klassifikationsalgorithmen kennen, die du in **μanthano** zum Lösen von Klassifikationsproblemen nutzen kannst.

Entscheidungsbaum

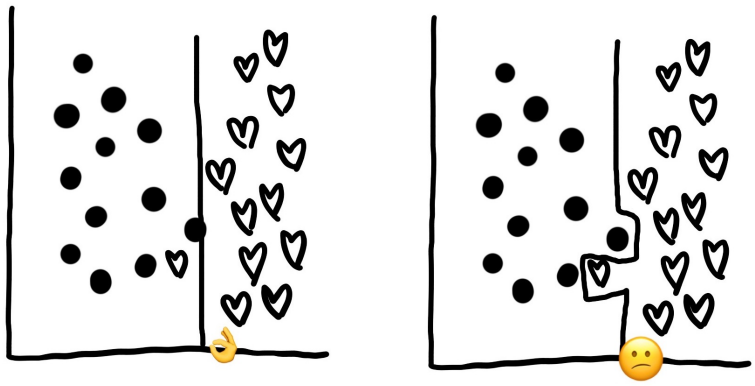
Der Entscheidungsbaum ist ein beliebtes klassisches ML Modell. Um etwas zu klassifizieren, werden alle Merkmale eines Datensatzes vom Algorithmus in Ja/Nein-Fragen modelliert. Der Algorithmus geht dabei vor, wie wir Menschen es auch manchmal tuen, wenn wir etwas entscheiden wollen. Je weiter oben sich eine Frage im Baum befindet, desto allgemeiner ist sie.



Die Fragen generiert der Algorithmus so, dass der **Gini-Index** bei jeder Frage möglichst **minimiert** wird. Der Gini-Index wird über die Wahrscheinlichkeit berechnet, dass ein Datenpunkt in dem aktuellen Abschnitt des Entscheidungsbaums falsch klassifiziert wird, also nicht seiner Klasse zugeordnet wird. Je kleiner der Gini-Index also ist, desto besser werden alle Datenpunkte klassifiziert. Bei einem unteren Blatt im Baum, wo sich der Algorithmus für eine Klasse entscheidet, sollte der Gini-Index dementsprechend bei 0,0 liegen.



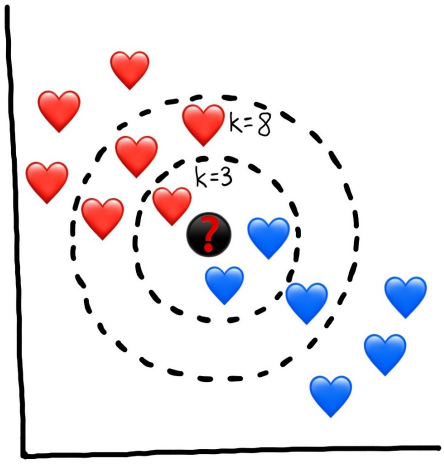
Du kannst entscheiden, ob der Baum seine Blätter über den Gini-Index oder zufällig generieren soll. Außerdem kannst du die maximale Tiefe des Baumes bestimmen, also wie viele Fragen maximal gestellt werden sollen, bevor der Baum zu einem Ergebnis kommen soll. Das ist wichtig, da es sonst passieren kann, dass das Modell auf die Trainingsdaten **„overfittet“** (überanpasst) wird und zu genaue Entscheidungsgrenzen gezogen werden.



k-nächste Nachbarn

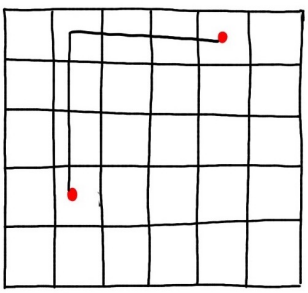
„Gleich und gleich gesellt sich gern“ – Ganz nach diesem Motto geht der k-nächste Nachbarn Algorithmus vor. Um einen neuen Datenpunkt zu klassifizieren, schaut sich dieser Algorithmus die Klassen der nächsten Nachbarn an und entscheidet dann nach Mehrheit. Diesem Algorithmus musst du dabei drei Dinge mitteilen.

1. Wie viele Nachbarn soll sich der Algorithmus anschauen? (das ominöse **k**)

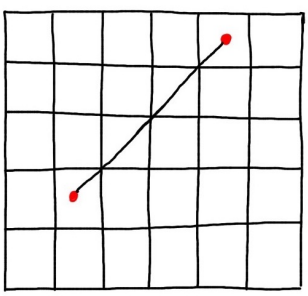


Wie im Bild zu erkennen ist, kann die Anzahl der zu betrachtenden Nachbarn einen erheblichen Einfluss auf das Ergebnis nehmen. Je nach Problem und Lage der Daten, muss das k angepasst werden.

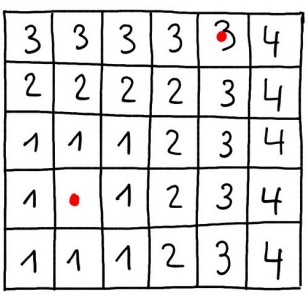
2. Welche Distanzmetrik soll der Algorithmus zur Berechnung der nächsten Nachbarn nutzen? Die Bilder im Folgenden zeigen dir die Arbeitsweise der verschiedenen Metriken.



manhattan



Euklid



Chebyshev

3. Welche Gewichtung soll der Algorithmus der verschiedenen Nachbarschaften bei der Entscheidung der Klasse zukommen lassen?

Uniform: Alle Datenpunkte in der Nachbarschaft werden gleich gewertet
Distanz: Nähere Nachbarn haben größeren Einfluss auf die Entscheidung für eine Klasse als fernere Nachbarn

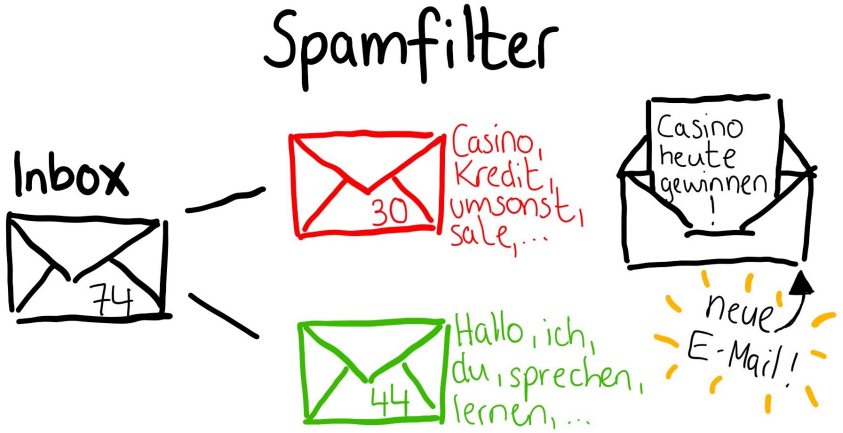
Naiver Bayes

Der naive Bayes Algorithmus basiert auf dem **Satz von Bayes**, den man aus dem Mathematikunterricht kennt. Mit ihm kann man die **bedingte Wahrscheinlichkeit** von zwei aufeinanderfolgenden Ereignissen bestimmen.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

P(A|B) ist die bedingte Wahrscheinlichkeit des Ereignisses A unter der Voraussetzung, dass Ereignis B vorher eingetreten ist.

Zum besseren Verständnis schauen wir uns Spamfilter an, bei denen früher oftmals der Naive Bayes angewendet wurde.



E-Mails werden aufgrund der Worte, die sie enthalten, als Spam oder Nicht-Spam markiert. Wir wollen jetzt also für eine neue E-Mail entscheiden, die das Wort Casino enthält, wie hoch die Wahrscheinlichkeit ist, dass diese E-Mail Spam ist. Dafür stellen wir folgende Rechnung an. *(Wir gehen davon aus, dass 18 E-Mails in den Trainingsdaten das Wort Casino enthalten, wobei 14 davon als Spam klassifiziert wurden und 4 E-Mails als Nicht-Spam)*

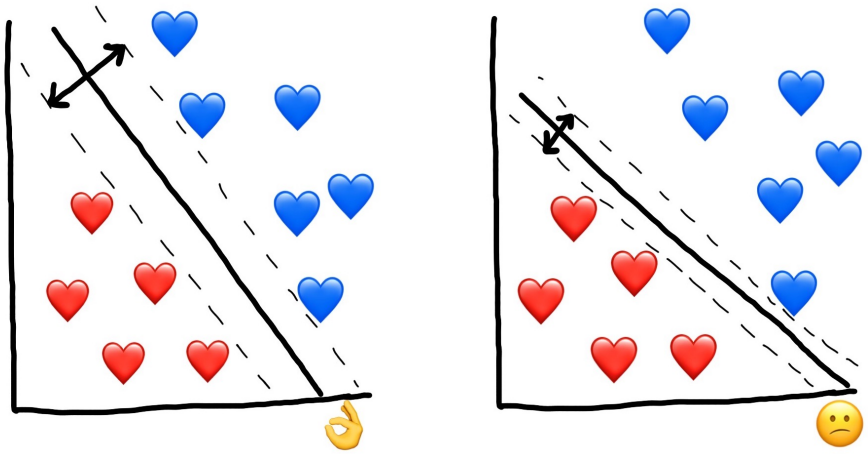
$$P(\text{Spam}|\text{Suchwort}) = \frac{P(\text{Suchwort}|\text{Spam}) \cdot P(\text{Spam})}{P(\text{Suchwort})}$$

$$P(\text{Spam}|\text{Casino}) = \frac{\frac{14}{30} \cdot \frac{30}{74}}{\frac{18}{74}} = 0{,}78$$

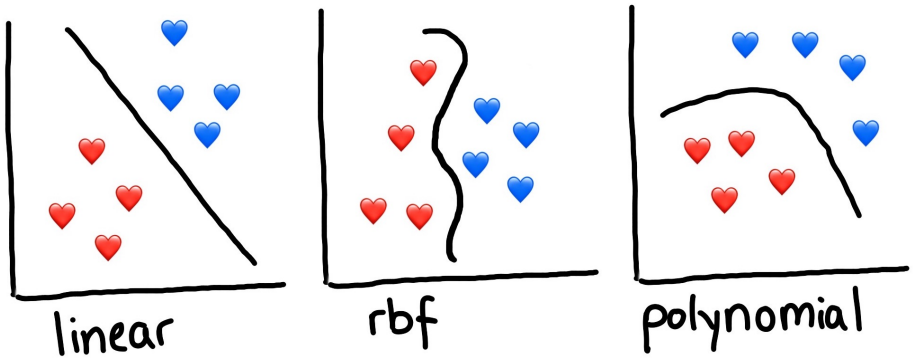
Der Naive Bayes geht allerdings davon aus, dass jedes Merkmal unabhängig vom anderen zu betrachten ist. Was bedeutet das im konkreten Fall unseres Spamfilters? Der Algorithmus lässt sich sehr leicht austricksen. Beispielsweise könnte am Ende einer Spam-E-Mail ein großer Absatz gefüllt mit Worten aus der Nicht-Spam-Gruppe stehen. Damit würde der Algorithmus eine solche Spam-E-Mail gegebenenfalls als Nicht-Spam klassifizieren. Um das zu umgehen werden Spamfilter heutzutage mit anderen Algorithmen umgesetzt – zum Beispiel mit dem Folgenden.

Support Vector Maschine

Um die doch komplexe Mathematik hinter den Support Vector Maschinen mal außen vor zu lassen, macht dieser Algorithmus in ganz einfachen Worten Folgendes: er versucht 2 parallele Linien (**Stützvektoren**) zwischen die Klassen zu legen, sodass der Abstand zwischen den beiden Linien maximal groß wird.



Da die Daten unter Umständen nicht immer linear separierbar sind, kannst du auch einen anderen **Kernel** auswählen, welcher vorgibt, wie die Entscheidungsgrenze ermittelt wird.



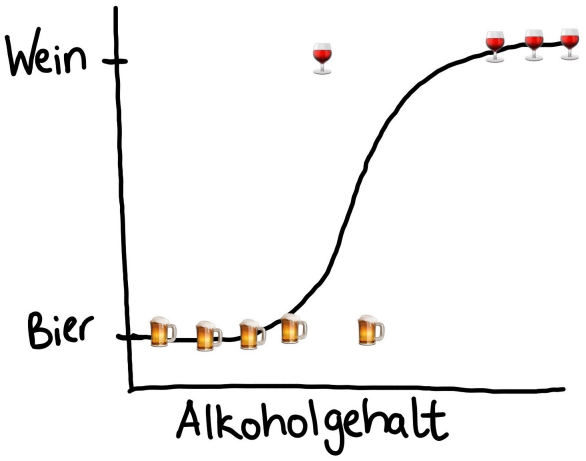
Beim polynominalen Kernel musst du zusätzlich den Grad des Polynoms setzen. Eine kleine Auffrischung zu Polynomen findest du im Kapitel ***Regression***.

Logistische Regression

Trotz des etwas verwirrenden Namens ist die Logistische Regression nicht etwa ein Algorithmus zum Lösen von Regressionsproblemen, sondern zum Lösen von Klassifikationsproblemen. In einfachen Worten, sagt er die Wahrscheinlichkeit des Auftretens eines Ereignisses vorher, indem es Datenpunkte an eine **Logitfunktion** anpasst.

$$sig(t) = \frac{1}{1 + e^{-t}}$$

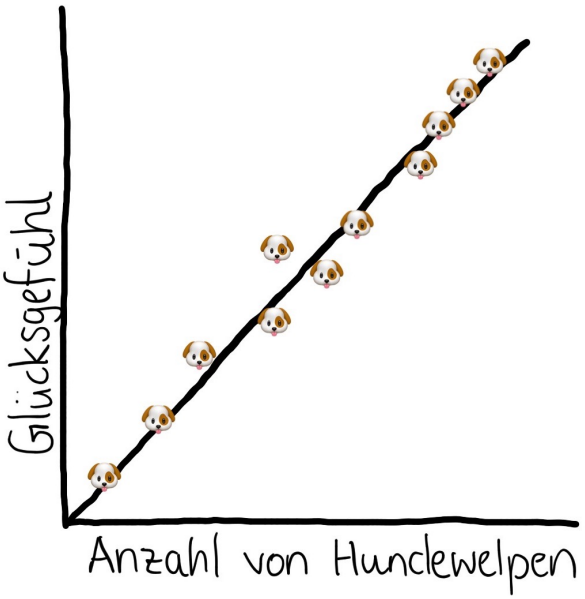
Man spricht in diesem Zusammenhang auch davon, dass man den **natürlichen Logarithmus einer Chance**, dass ein Ereignis eintritt, berechnet. Im Beispiel wollen wir anhand des Alkoholgehalts eines Getränks klassifizieren, ob es sich bei dem Getränk um Wein oder Bier handelt. Dazu haben wir einen Datensatz, der viele viele Biere und Weine mit ihrem Alkoholgehalt enthält. Auf diesen Datensatz passen wir nun im Training die Logitfunktion an, um für neue Datenpunkte voherzusagen, ob es sich um Bier oder Wein handelt.



Regression

Im Folgenden lernst du Regressionsalgorithmen kennen, die du in **μanthano** zum Lösen von Regressionsproblemen nutzen kannst.

Lineare Regression



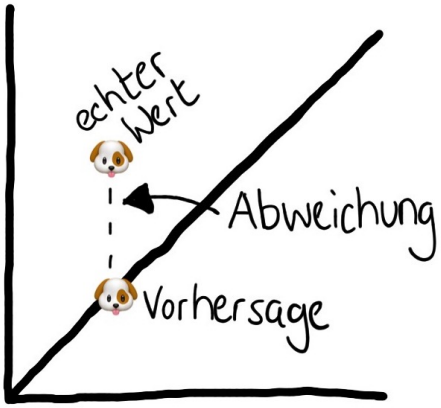
Zeichne eine Linie durch alle deine Datenpunkte – Zack, fertig! Regression

Ganz so einfach ist das mit der Regression natürlich nicht, aber die Grundidee stimmt schon einmal.

Bei der **linearen Regression** versucht der Algorithmus eine lineare Funktion zu finden, die möglichst auf die Datenpunkte passt.

Dabei ermittelt der Algorithmus die beste lineare Funktion für einen Datensatz, indem er versucht die **quadratische Abweichung** zu minimieren. Die Abweichung errechnet sich aus der Differenz von allen echten Datenpunkten und den von einer linearen Funktion vorhergesagten Werten.

Damit sich positive und negative Abweichungen nicht aufheben, nutzt der Algorithmus die quadratische Abweichung.



$2^2 = 4$

$(-2)^2 = 4$

$2 - 2 = 0$ ✗

$4 + 4 = 8$ ✓

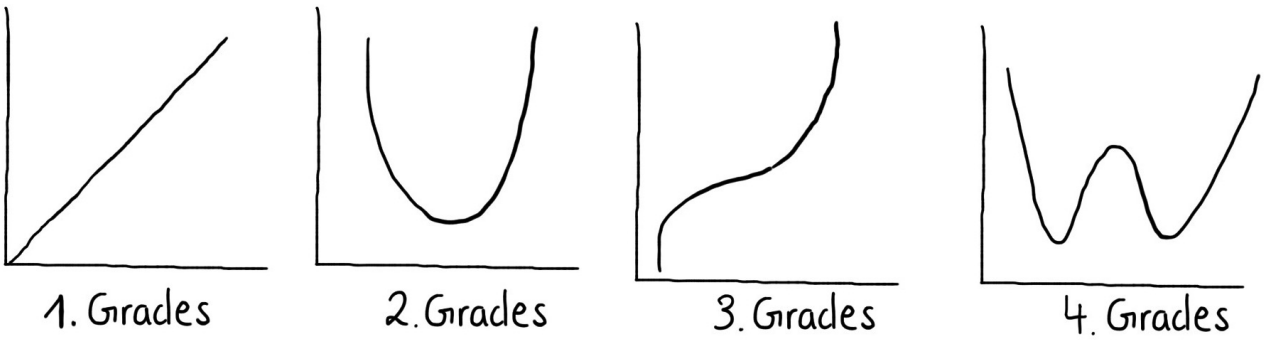
Polynomiale Regression

Da Daten nicht immer linear verlaufen, gibt es neben der linearen Regression auch andere Algorithmen. Eine davon ist die **polynominale Regression**.

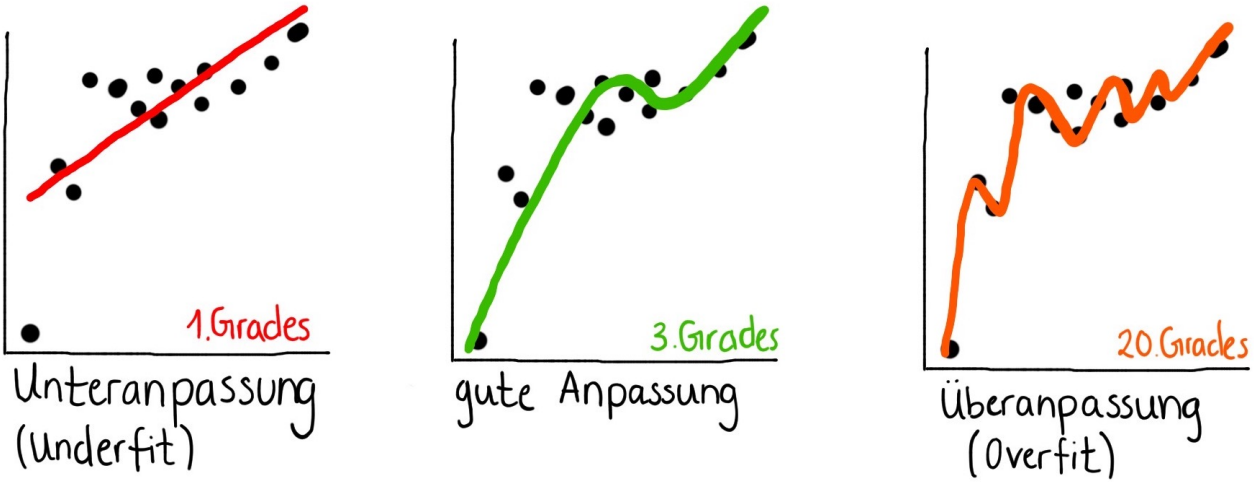
Ein **Polynom** ist einfach ein mehrgliedriger Term.

$$5x^4+2x^3+7x^2-12x+9$$

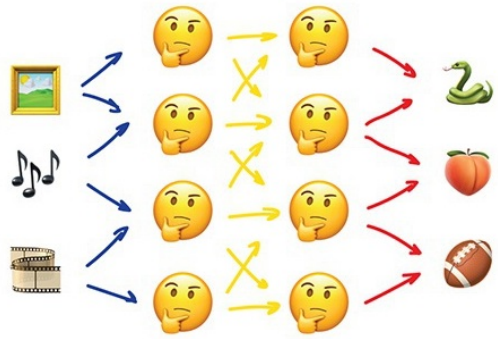
Ein Polynom hat immer einen bestimmten **Grad**. Bei dem Term oben handelt es sich um ein Polynom 4. Grades.



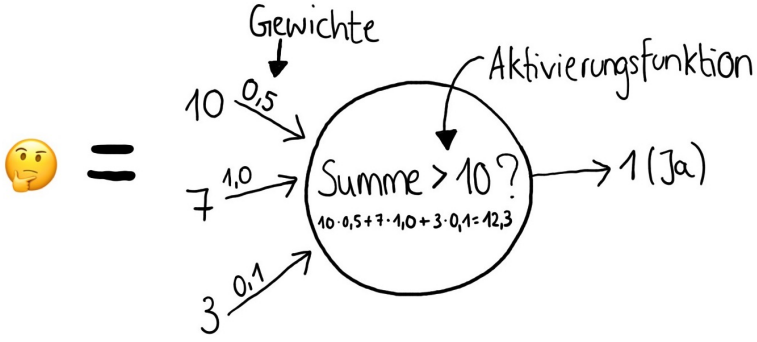
Den Grad kannst du bei der polynominalen Regression anpassen, je nachdem wie es für den zu verarbeitenden Datensatz Sinn macht. Dabei sollte man aber nicht über das Ziel hinausschießen und das Model **überanpassen**. Du musst immer daran denken, dass das Modell nicht nur für die Trainingsdaten gute Ergebnisse liefern soll sondern auch für neue Daten, die eventuell etwas anders aussehen als die Trainingsdaten.



Künstliche Neuronale Netze (Mehrlagiges Perzeptron)

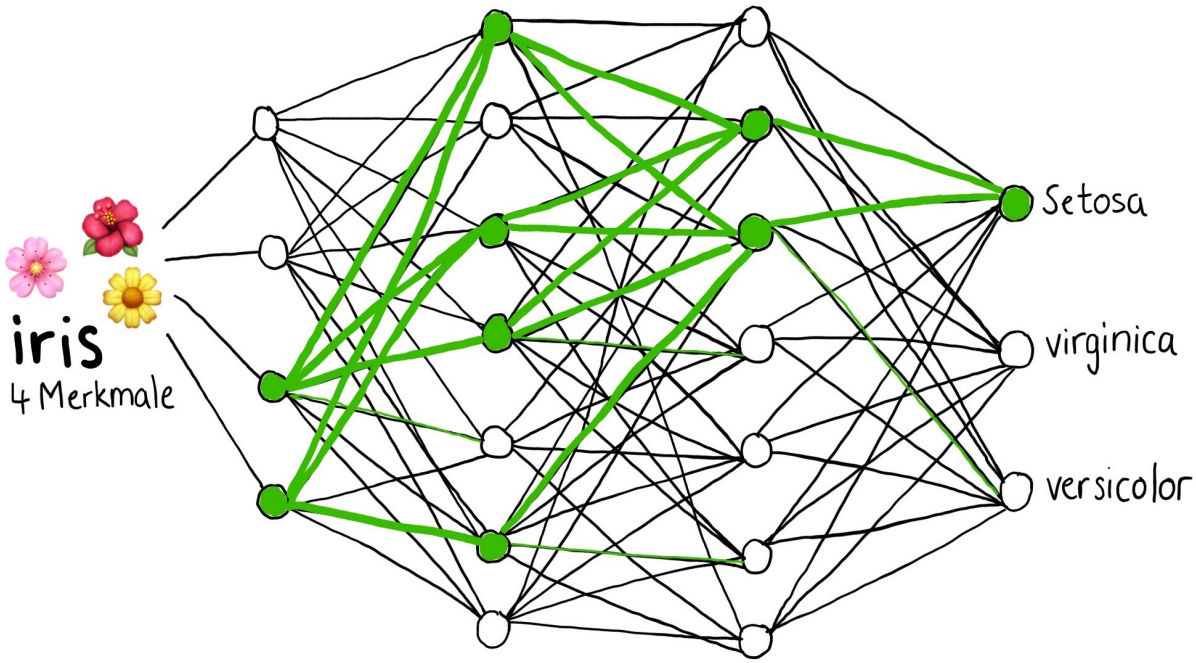


Ein **künstliches neuronales Netz** ist eine Sammlung von Neuronen und Verbindungen zwischen diesen. Ein **Neuron** ist mehr oder weniger eine Funktion mit vielen Eingaben und einer Ausgabe. Jede **Verbindung** ist in einem neuronalen Netz mit einem Gewicht versehen. Dieses **Gewicht** sagt dem Neuron, wie stark es jede Eingabe gewichten soll. Beim Training werden die Werte der Gewichte angepasst.
KNNs können sowohl bei der Klassifikation als auch bei der Regression eingesetzt werden.



Die Neuronen sind in einem künstlichen neuronalen Netz in Schichten angeordnet. Jedes Netz besitzt eine **Eingangsschicht** und eine **Ausgangsschicht** und beliebig viele **versteckte Schichten**. Die Anzahl der Neuronen pro Schicht kann bei jeder Schicht unterschiedlich sein.

Man spricht davon, dass ein Neuron ***feuert***, wenn die Aktivierungsfunktion einen gewissen Schwellwert erreicht *(siehe Abbildung oben)*.



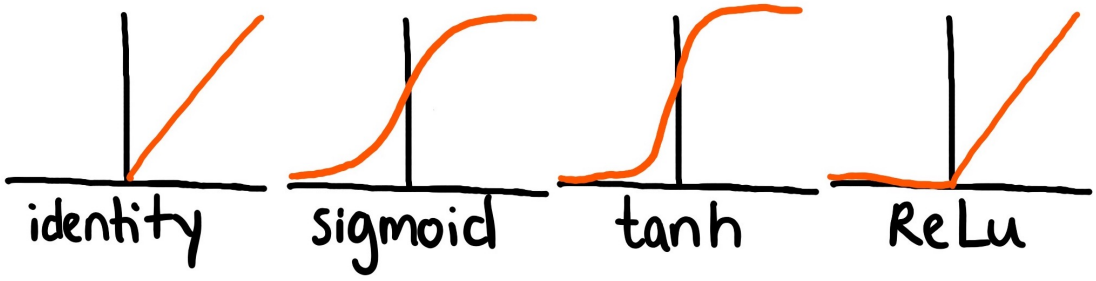
Wie oben beschrieben, werden die Gewichte an den Verbindungen beim Training angepasst. Diesen Vorgang nennt man **Backpropagation**. Beim Training werden die Trainingsdaten immer wieder durch das Netz gereicht. Am Ende wird geschaut, ob das richtige Ergebnis herauskam. Je nach Ergebnis wird dann der Weg rückwärts abgelaufen und die Gewichte an den Verbindungen werden angepasst.

Wenn du ein künstliches neuronales Netz entwickelst, kannst du neben der Anzahl der Schichten und der Neuronen pro Schicht sogenannte Hyperparameter anpassen. Diese sind im Folgenden näher erklärt.

Aktivierungsfunktion

Es gibt verschiedene Aktivierungsfunktionen, die man für die Neuronen auswählen kann. Wenn du genauer wissen willst, was hinter den verschiedenen Funktionen steckt, kannst du dir folgenden Artikel durchlesen (**ACHTUNG***: es wird mathematisch und ein bisschen komplizierter*):

Understanding Activation Functions in Neural Networks



Für viele Probleme funktioniert ***ReLU*** (Rectified Linear Unit) sehr gut.

Lernalgorithmus

Der Lernalgorithmus bestimmt, wie genau die Gewichte beim Training angepasst werden. ***Adam*** funktioniert gut bei großen Datensätzen, während ***lbfgs*** unter Umständen besser und schneller bei kleineren Datensätzen ist. Außerdem gibt es noch den Algorithmus Stochastic Gradient Descent (***sgd***).

Lernrate

Die Lernrate bestimmt, wie schnell oder langsam die Gewichte beim Training angepasst werden sollen. Damit man die optimalen Werte möglichst schnell erreicht, darf die Lernrate nicht zu klein gewählt sein. Sie darf aber auch nicht zu groß gewählt sein, damit man die optimalen Werte nicht verpasst. Ein guter Anfang ist der Wert **0,001**.

maximale Iterationen

Die maximale Anzahl von Iterationen zeigt an, wie oft dem Netz alle Trainingsdaten gezeigt werden, bevor der Lernprozess abgeschlossen werden soll.
300 ist eine gute Zahl für den Anfang. Je nach Problem muss diese Zahl allerdings erhöht werden.