Künstliche Intelligenz

→ siehe Präsentation 1 Einführung KI

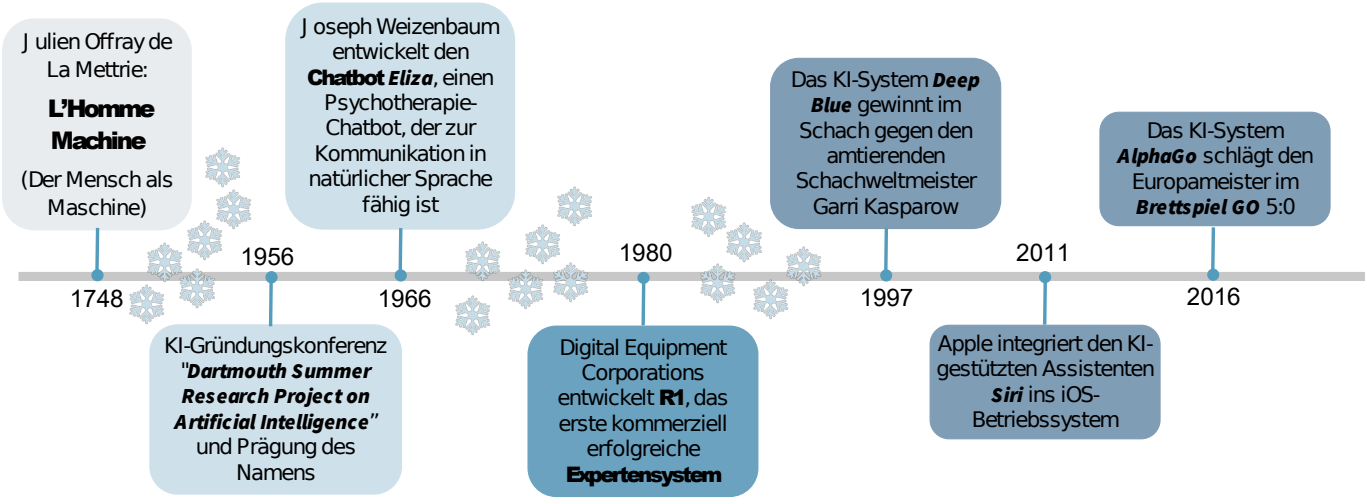
Begriff Intelligenz

Der Begriff „***Künstliche Intelligenz***“ (kurz: ***KI*** oder engl. AI - artificial intelligence) wurde erst 1956 bei einer Konferenz zu intelligenten Maschinen entworfen. Die Idee dahinter wurde aber z.B. bereits im Jahr 1748 in Julien de La Mettries Werk „L’Homme Machine“ diskutiert.

Das Verständnis, was ***künstliche Intelligenz*** ausmacht, unterlag bisher einem stetigen Wandel.

- 1983: Künstliche Intelligenz befasst sich mit der Frage, wie man Computer dazu bringen kann, Dinge zu tun, die Menschen im Moment noch besser können.
  Ziel: Handlung
- 1992: Künstliche Intelligenz ist die Erforschung von Rechenverfahren, die es ermöglichen, wahrzunehmen, zu schlussfolgern und zu handeln.
  Ziel: Interaktion
- 2022: Künstliche Intelligenz […] ist ein Teilgebiet der Informatik, das sich mit Wikipedia der Automatisierung intelligenten Verhaltens und dem Maschinellen Lernen befasst.
  Ziel: Automatisierung & automatisches Lernen

Wichtige Meilensteine zur künstlichen Intelligenz



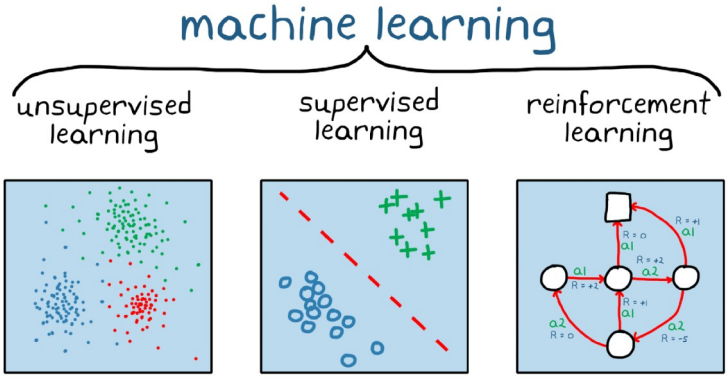
Grundideen maschinellen Lernens

Lernen stellt einen fortlaufenden, nicht abschließenden Prozess dar, in dem das Verhalten, Wissen und Können durch die Erfahrungen angepasst und dadurch das Ergebnis verbessert wird. Dieses Vorgehen kann einfach auf maschinelles Lernen übertragen werden. Die Maschine verwendet Daten und Algorithmen, um neue Ergebnisse, Fähigkeiten oder Entscheidungen zu generieren.

Aufgaben

- ***Klassifikation***: Zuordnung von Datensätzen zu bekannten Gruppen basierend auf ihren Merkmalen
- ***Regression***: Vorhersage von numerischen Werten basierend auf den Zusammenhängen
- ***Assoziation***: Bestimmen von Wahrscheinlichkeiten, dass miteinander auftretende Ereignisse in Zusammenhang stehen
- ***Clustering***: Organisation von Daten durch die Aufteilung in Gruppen mit ähnlichen Merkmalen

Varianten des maschinellem Lernen:



Reinforcement Learning (verstärkendes Lernen)

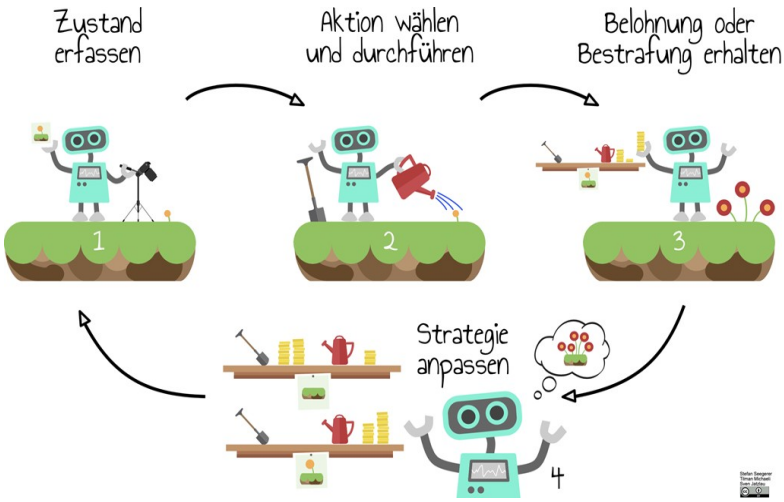
Lernverfahren zur Erlernung einer Strategie, um zu die zu erhaltene Belohnung zu maximieren.

Der Algorithmus erfasst hier selbstständig Informationen, die im Hinblick auf sein Ziel relevant sind (1).

Aus seinen vorgegebenen Handlungsmöglichkeiten wählt der Algorithmus abhängig von dieser Analyse ein Vorgehen aus (2).

Wählt der Algorithmus die Alternative aus, die ihn näher an sein Ziel bringt, wird er belohnt. Hat eine Entscheidung einen negativen Einfluss auf die Zielerreichung, wird er bestraft (3).

Durch die positiven und negativen Rückmeldungen passt der Algorithmus seine Strategie an. Aktionen, die zum Ziel führen, werden „bestärkt“ und nicht erfolgreiche geschwächt (4).



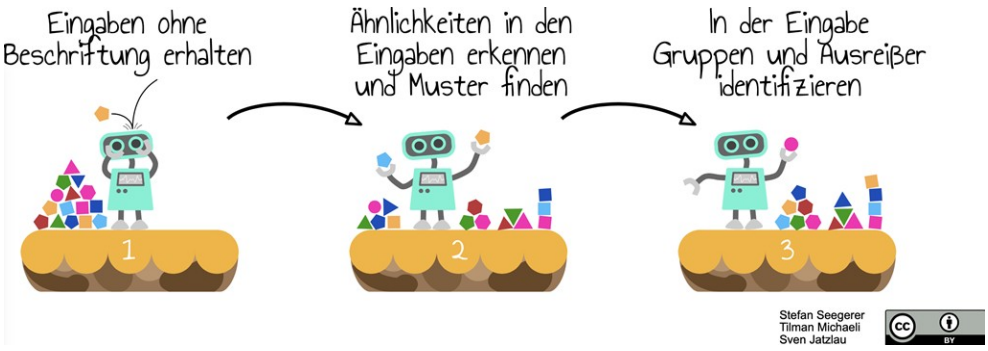
Unsupervised Learning (unüberwachtes Lernen)

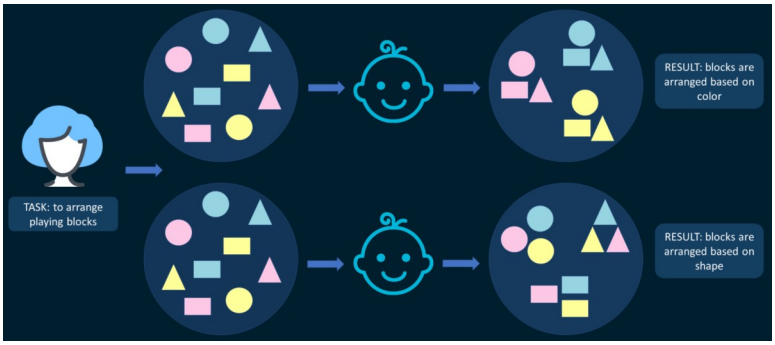
Lernverfahren zur Erkennung von Mustern innerhalb der Daten ohne (im Voraus) bekannten Ergebnissen

Dem unüberwachten Lernen stehen keine benannten Daten zur Verfügung (1).

Der Algorithmus verwendet die Ähnlichkeiten der Daten in einzelnen Merkmalen und versucht dadurch, die Daten zu gruppieren. Dabei geht man davon aus, dass ähnliche Daten auch ähnliche Merkmalsausprägungen haben (2).

Danach verbleiben diejenigen Daten, die in keine der Gruppen passen (sogenannte Ausreißer) (3).





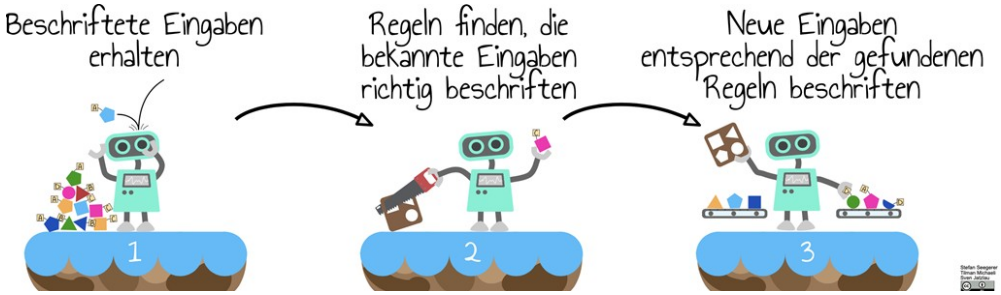
Supervised Learning (überwachtes Lernen)

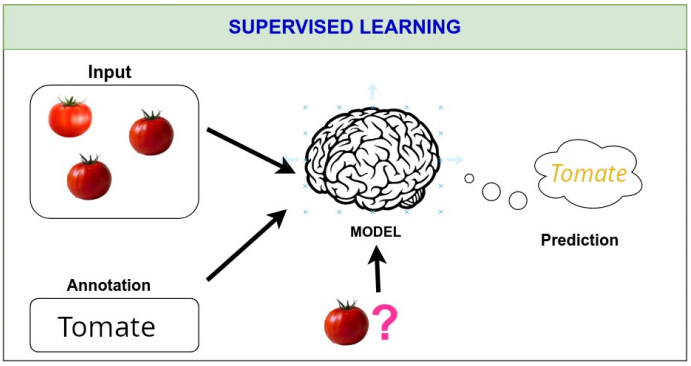
Lernverfahren, um Regeln zu bekannten Ergebnissen zu erlernen

Der Algorithmus erhält bereits gelabelte (in Klassen zugeordnete) Daten (1).

Daraus erstellt der Algorithmus Regeln, die eine Zuordnung der Daten zu den Klassen ermöglichen (2).

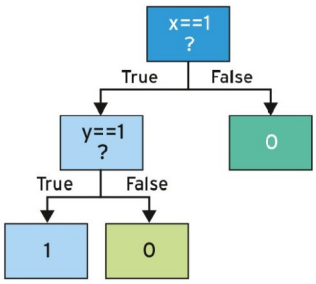
Mit den Regeln werden neue, unbeschriftete Daten klassifiziert (3).





Entscheidungsbaumalgorithmus

→ siehe Präsentation 2 Entscheidungsbaum



Ein ***Entscheidungsbaum*** soll zur Einteilung von Daten in Gruppen (***Klassifikation***) dienen. Er besteht aus ***Knoten*** und ***Kanten***, wobei die inneren Knoten Auswahlmöglichkeiten repräsentieren, die auf ein bestimmtes ***Merkmal*** (***Attribut***) des zu klassifizierenden Datenpunkts (Objekts) angewendet werden. Die ***Kanten*** repräsentieren die unterschiedlichen ***Attributwerte***, wobei diese ggf. auch gruppiert sein können. Der Datenpunkt wird entsprechend der Beschriftung des Blattes am Ende des durchlaufenden Pfades ***klassifiziert*** (gelabelt).

Ziel des Algorithmus ist es, durch die Informationen, die sich in den Trainingsdaten befinden, die Knoten und Kanten des Baumes von der Wurzel zu den Blättern so zu entwerfen, dass der Baum die Trainingsdaten richtig klassifiziert.

Die Auswahl desjenigen ***Attributs*** in einem ***Knoten***, das den meisten ***Informationsgewinn*** liefert, wird zum Beispiel mittels ***Fehleranzahlanalyse*** bestimmt. Dabei werden für alle noch nicht verwendeten Attribute die Aufteilungen anhand der Merkmale jeweils eines Attributs ermittelt und die richtigen und die falschen Zuordnungen der Individuen gezählt.

Nach dem Attribut mit dem größten Informationsgewinn (oder den wenigsten Fehlern) werden die Individuen nun gruppiert und jeweils für diese Teilgruppen werden die nächsten besten Attribute gesucht.

Konkreter Algorithmus zum Entscheidungbaum

Wiederhole für die Datenmenge D solange, bis ein oder mehrere Abbruchkriterien erfüllt sind:

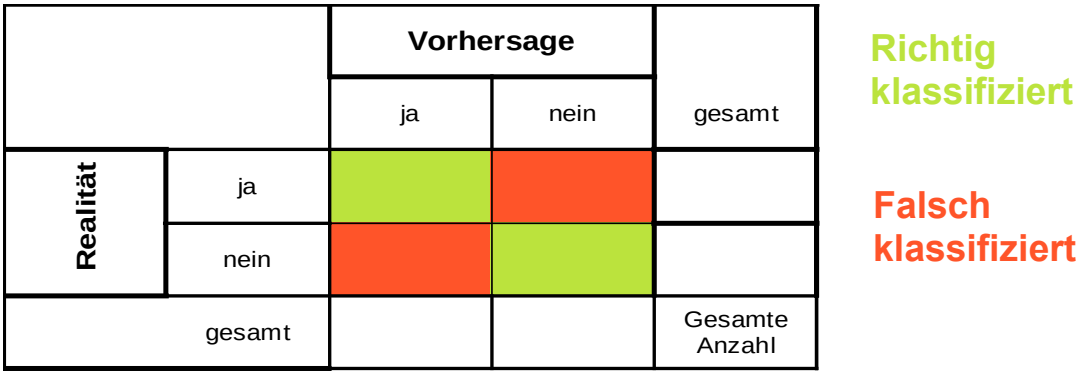
> ***Bestimme das Attribut A, das hinsichtlich des gewählten Aufteilungskriteriums den größten Informationsgewinn bzw. den kleinsten Fehler liefert.***
>
> ***Teile die Datenmenge D anhand einer gewählten Regel der Attributwerte von A in D_i auf.***

Abbruchkriterien:

- **alle Merkmale verwendet**
- **geringer Informationsgewinn bei weiterer Aufteilung**
- **Anzahl der verbleibenden Datensätze zu gering**
- **Datensätze komplett klassifiziert**

Güte des Entscheidungsbaums

Mit Hilfe der Testdaten lässt sich entsprechend überprüfen, wie gut der Baum neue Daten bereits klassifiziert. Dazu verwenden wir wieder andere, beschriftete Datensätze (Individuen). Jedes Individuum der Testdaten wird anhand des Baumes klassifiziert und mit den richtigen Beschriftungen jeweils verglichen. Dabei können vier Fälle unterschieden werden:



Diese Aufteilung nennen wir ***Konfusionsmatrix***. Mit den eingetragenen Werten lassen sich verschiedene statistische Gütekriterien bestimmen. Wir betrachten dabei die Genauigkeit, die angibt, wie viel Prozent richtig klassifiziert sind:

$$\textbf{Genauigkeit} = \frac{Anzahl\ richtig\ klassifiziert}{Anzahl\ richtig\ klassifiziert + Anzahl\ falsch\ klassifiziert}$$

Aufgabe 6:
Erstelle analog zum ersten Beispiel der Präsentation einen Entscheidungsbaum und bewerte anschließend mit Hilfe der Konfusionsmatrix die Genauigkeit Deines Baums.

Hole Dir mit Deinem Nachbarn dazu zu zweit einen Satz der Bildkarten des Beispiels ***Hitzefrei***. Alternativ kannst du auch die Exceltabellen zu Hitzefrei heranziehen.

Aufgabe 7:
Erstelle analog zum ersten Beispiel der Präsentation einen Entscheidungsbaum und bewerte anschließend mit Hilfe der Konfusionsmatrix die Genauigkeit Deines Baums.

Die Datengrundlage sind Trainingsdaten und Testdaten zu ***Fische2***.

In der Praxis verwendet man speziell entwickelte Programme oder Programmiersprachen, die einen bei der Auswertung helfen. Ein Vertreter hierzu ist das Data Mining-Programm ***Orange***.

→ siehe Präsentation 3 Entscheidungsbaum mit Orange

Hyperparameter

Bei der Erstellung der Entscheidungsbäume mit geeigneter Software können zusätzliche Bedingungen (***Hyperparameter***) gestellt werden, die sich auf die Struktur des Baumes auswirken. Dies können sein:

- Maximale Tiefe des Baums,
- Aufteilen der Knoten beenden bei erreichter Genauigkeit,
- Minimale Anzahl von Objekten in einem Blatt
- die binäre Aufspaltung erzwingen oder
- falls eine bestimme prozentuale Klassifikation möglich

Der Entscheidungsbaumalgorithmus sucht nach ***Mustern*** in Daten und erstellt daraus ein ***Modell***. Er eignet sich besonders gut bei Daten, deren Ausprägungen sich in ***Kategorien*** eingeteilt sind. Es gibt keine Garantie, dass der "beste" bzw. überhaupt "ein guter" Entscheidungsbaum gefunden wird. Entscheidungsbäume sind nur dann sinnvoll anwendbar, wenn in den Daten Muster vorhanden sind. Werden die Trainingsdaten zu genau gelernt (***overfitting***), kann ein zu sehr angepasster Baum entstehen, der sich bei leicht geänderten Merkmalsausprägungen nicht mehr verwenden lässt. Er lässt sich durch Stutzen des Baumes (***pruning***) wieder so umwandeln, dass er mehr generalisiert ist. Andersherum kann es vorkommen, dass vorliegende Muster nicht oder nur schwach gelernt werden (***underfitting***). Hier muss man weitere Testdaten vorhalten.

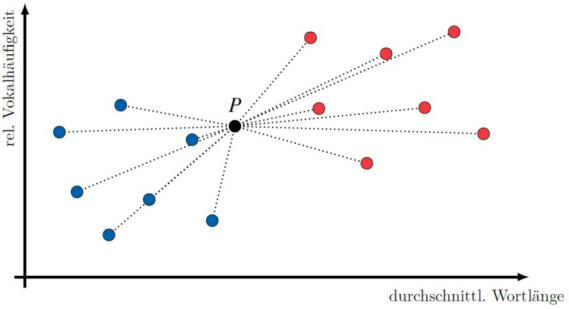
Aus diesen Gründen kommt der Testphase eine große Bedeutung zu, in der die Qualität des erstellten Modells überprüft wird.

k-nächster-Nachbarn-Algorithmus (optionale Vertiefung für Schnelle)

Der ***k-nächste-Nachbarn-Algorithmus*** kann ebenfalls zur ***Klassifikation*** genutzt werden. Er benötigt ebenfalls beschriftete Eingabedaten.
Zur Klassifizierung eines neuen Datenpunkts P zieht man eine ***gegebene Anzahl k*** von Trainingsdatenpunkten heran, die ihm in Bezug auf seine Merkmalsausprägungen "***am ähnlichsten***" sind. Man ordnet ihm diejenige Klasse zu, die unter diesen k Trainingsdatenpunkten am häufigsten vorkommt. Sollten zwei Klassen gleich oft vorkommen, so muss man manuell vorher eine Zuordnung vorgeben.

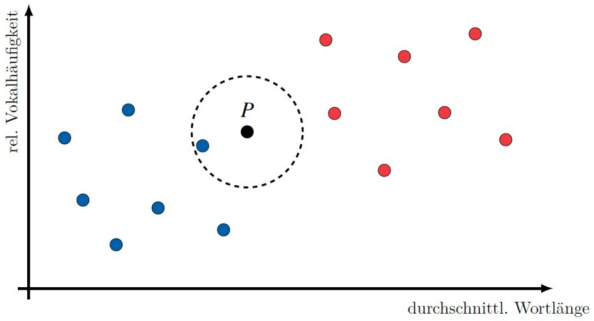
Die ***Ähnlichkeitsbeziehung*** löst man häufig mit dem ***euklidischen Abstand***. Dabei müssen die Ausprägungen der Merkmal aber so in Zahlen repräsentiert sein, dass keine ***Verzerrungen*** auftreten.

Transformiert man zum Beispiel drei Ausprägungen wie sonnig, wolkig und bedeckt in Zahlenwerte 1, 2 und 3, so hat sonnig zu bedeckt einen größeren Abstand als wolkig zu bedeckt.

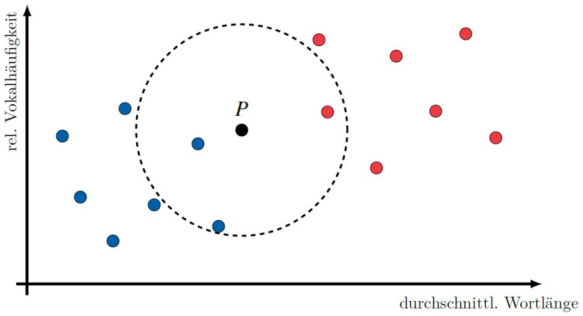


Algorithmus

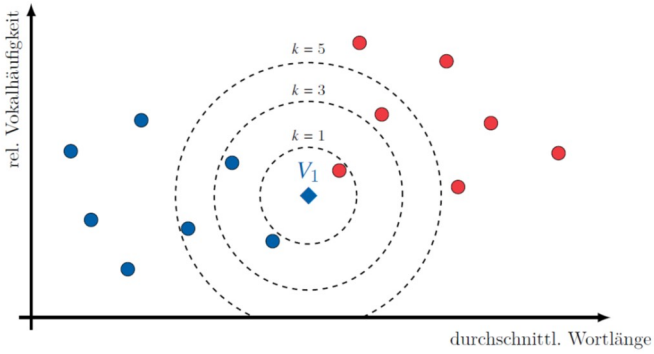
1. ***Vorbereitung***
   a. ***Trainingsdaten bereitstellen***
   b. ***Hyperparameter k festlegen***
2. ***Abstand des neuen Datenpunkts zu allen Trainingsdatenpunkten berechnen***
3. ***Trainingsdaten nach Abstand aufsteigend sortieren***
4. ***Neuen Datenpunkt anhand des Parameters k klassifizieren***



Das



Bestimmen des besten Wertes für k kann mit Hilfe mehrerer bereits gelabelter Datenpunkte durchgeführt werden. Dazu werden bekannte Datenpunkte von den Trainingsdaten als ***Validierungsdaten*** getrennt. Diese



Daten werden mit jeweils ***verschiedenen Werten für k klassifiziert*** und der Wert für ***k*** gewählt, der am häufigsten zum richtigen Ergebnis führt.

Da bei jedem neuen Datenpunkt wieder alle bisherigen Datenpunkte wieder betrachtet und ausgewertet werden (Abstandsberechnung), wird der knN-Algorithmus als ***lazy-learner*** bezeichnet.

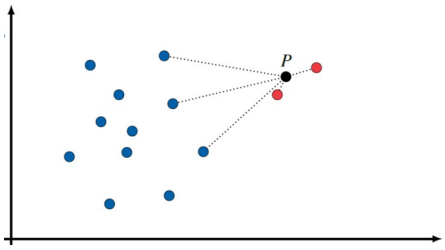
Hyperparameter

Als ***Hyperparameter*** können neben dem Wert ***k*** noch weitere Parameter gestellt werden, die sich auf die Bewertung auswirken. Dies können sein:

- Auswahl der Abstandsmetrik
  Neben dem normalen Abstand gibt es noch andere Berechnungsmöglichkeiten (***Metriken***) wie die ***Manhattan-Distance*** oder die ***Minkowski-Metrik***.
- Gewichtung der verschiedenen Merkmale

Grenzen von Entscheidungsbäumen

Je mehr Merkmale (man spricht von ***Dimensionen***) betrachtet werden, desto unzuverlässiger können die ***Klassifikationsergebnisse*** werden, d. h. mit zunehmender Anzahl der Dimensionen des Merkmalraums werden die Unterschiede zwischen der maximalen gemessenen Distanz und der minimalen gemessenen Distanz zweier beliebiger Datenpunkte geringer. Da der k-nächste-Nachbarn-Algorithmus auf der Abstandsberechnung basiert, kann er deswegen nicht mehr zuverlässig klassifizieren. Man spricht in diesem Zusammenhang vom ***Fluch der Dimensionalität***.



Des Weiteren kann eine ungünstige Verteilung in der Anzahl oder Lage der bekannten Datenpunkte Einfluss auf das Klassifikationsergebnis haben. Dominiert eine Klasse in ihrer Anzahl, so wird, so wird diese kaum beachtet. Gleiches gilt für die Lage der Punkte.

Aufgabe 12 (für Schnelle):

Bearbeite den ***Arbeitsauftrag*** aus dem Vorlagenverzeichnis zu k-nächste-Nachbarn-Algorithmus!

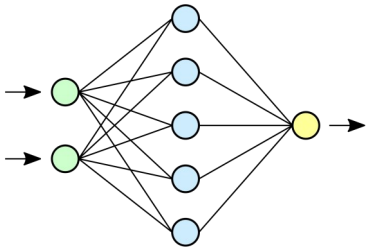
Neuronale Netze

→ siehe Präsentation 4 Perzeptron

Aufgabe 13:

Bildet Gruppen zu jeweils 8 Schülern und holt für jede Gruppe eine Materialkiste.

Spielt mehrere Runden des Wollperzeptrons mit wechselnden Aufgaben!



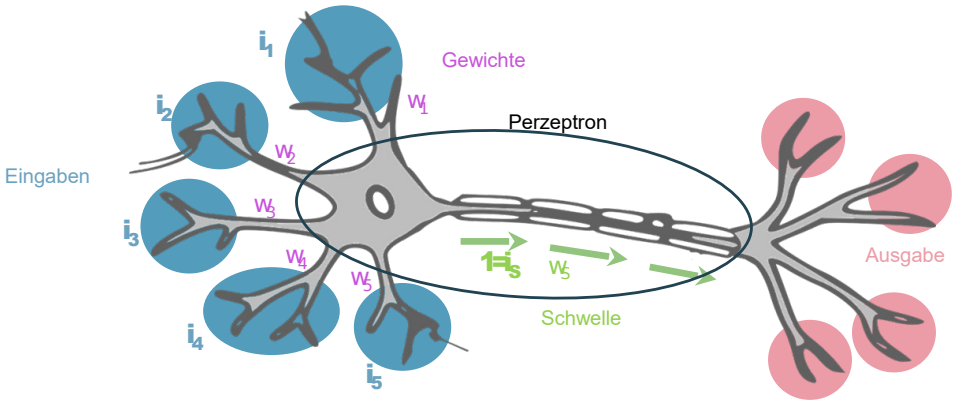
Künstliche neuronale Netze (***KNN***) sind Netze aus ***künstlichen Neuronen***. Sie stellen einen Forschungszweig der ***künstlichen Intelligenz*** dar. ***Künstliche neuronale Netze*** haben ein biologisches Vorbild. Bei KNNs geht es allerdings mehr um eine Abstraktion (Modellbildung) zur Informationsverarbeitung, weniger um das Nachbilden biologischer neuronaler Netze. KNNs sind universelle ***Funktionsapproximatoren*** (KNN können für beliebige Funktionen eingesetzt werden und passen sich dabei immer stärker an ihren Zweck an). Beim Trainieren des Netzes werden dabei an den Verbindungen zwischen den Neuronen Gewichte als Zahlenwerte anhand einer ***Fehlerkostenfunktion*** gebildet und aktualisiert.

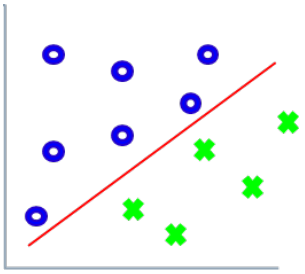
Perzeptron

→ Präsentation

Das ***Perzeptron*** (nach engl. Perception: ***Wahrnehmung***) ist einer einzelnen Nervenzelle nachempfunden. Es wurde von Frank Rosenblatt 1957 vorgestellt. Es besteht aus einem ***künstlichen Neuron***, das mit anpassbaren Gewichtungen und einem Schwellenwert arbeitet.

Der ***Zweck*** eines Perzeptrons ist ebenfalls die ***Klassifikation*** von Daten. Es stellt eine Anwendung des ***supervised learnigs*** dar. Mit seiner Hilfe lassen sich Daten in zwei

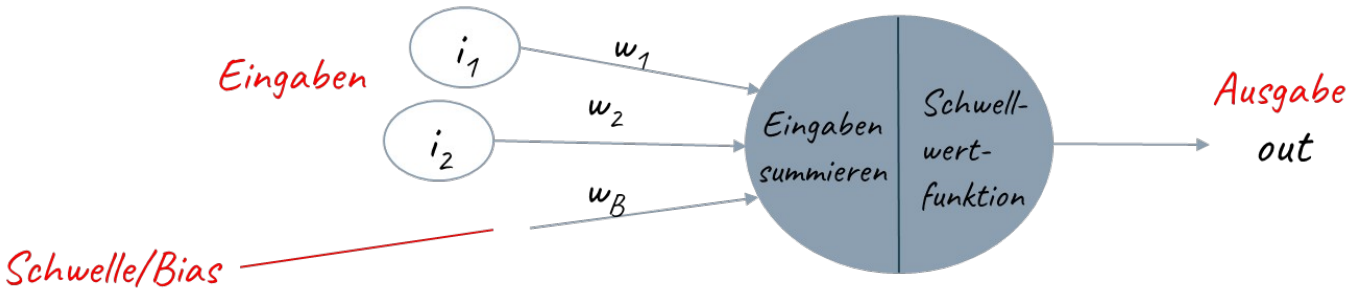




Klassen einteilen. Da sich zwischen den beiden Klassen eine „Trenngerade“ einfügen lässt, sprich man auch von linear separierbaren Daten. Ein Perzeptron nimmt die Eingabedaten (die Ausprägungen der Merkmale als Zahlenwerte) entgegen und bestimmt mit diesen

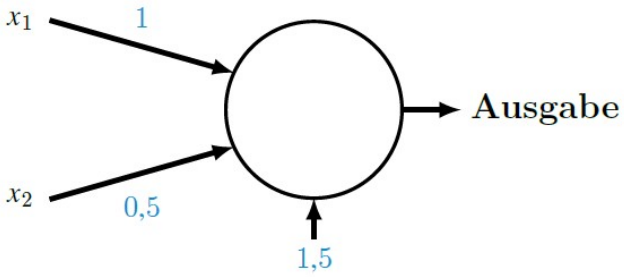
zusammen mit den zugehörigen Gewichten einen Wert. Anhand des Schwellwerts (Bias) kann man die Zugehörigkeit zu den beiden Klassen regeln. Liegt der Wert unterhalb, wird die eine Klasse ausgegeben, liegt er oberhalb, dann die andere.

Lernprozess



Das Perzeptron muss es trainiert werden, damit es zuverlässig klassifizieren. Dazu werden gelabelte Daten als Trainingsdaten benötigt. Vor dem ersten Trainingprozess werden die Gewichte und der Schwellenwert durch zufällige Werte vorbelegt. Das Perzeptron berechnet dann für einen Trainingsdatenpunkt die Ausgabe (Label) und vergleicht diese mit dem tatsächlichen Label (Beschriftung). Wir benennen die beiden Label jeweils dabei mit **0** und **1**.

Beispiel: Wir nehmen vier Punkte und benutzen x_1 als i_1 und x_2 als i_2.

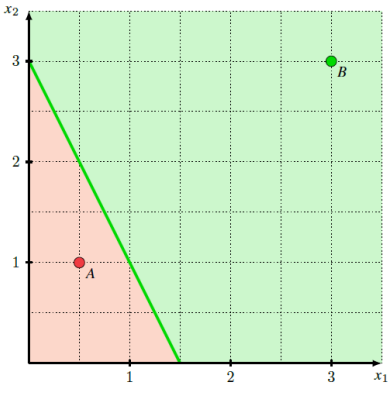


Daten-punkt	x_1	x_2	Label
A	0,5	1	0 (rot)
B	3	3	1 (grün)
C	0,25	2	1 (grün)
D	1,5	1	0 (rot)

Dann gibt es zwei Fälle:

1. Berechnetes Label und Beschriftung stimmen überein

   Das Perzeptron hat die Klassifizierung für diesen Datenpunkt **richtig** durchgeführt. Eine Anpassung der Gewichte und des Schwellenwerts ist somit nicht erforderlich.



2. Berechnetes Label und Beschriftung stimmen **nicht** überein

   Das Perzeptron hat die Klassifizierung für diesen Datenpunkt **falsch** durchgeführt. Eine Anpassung der Gewichte und des Schwellenwerts ist somit erforderlich. Wir unterscheiden zwei Fälle

   a. Berechnetes Label ist kleiner als das erwartete Label

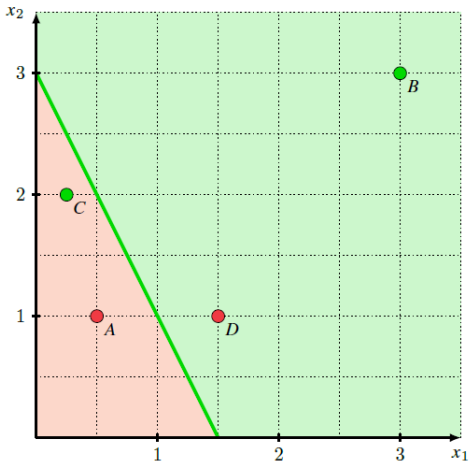
      Die gewichtete Summe liegt also unterhalb der Schwelle, also hat das Ergebnis den Wert 0, sollte aber 1 haben (Punkt C).

      Um die Trennlinie der beiden Halbräume in Richtung des aktuellen Datenpunktes zu bewegen, werden die Gewichte w_1 und w_2 erhöht und der Schwellenwert w_B verringert.

   b. Berechnetes Label ist größer als das erwartete Label

      Die gewichtete Summe liegt also unterhalb der Schwelle, also hat das Ergebnis den Wert 0, sollte aber 1 haben (Punkt D).

      Um die Trennlinie der beiden Halbräume in Richtung des aktuellen Datenpunktes zu bewegen, werden die Gewichte w_1 und w_2 erniedrigt und der Schwellenwert w_B erhöht.



Die Richtung, in die wir die Gewichte und den Schwellenwert verändern müssen, können wir uns aus dem berechneten Label (output → o) und dem erwarteten Label (target → t) selbst berechnen:

$$\delta = t - o$$

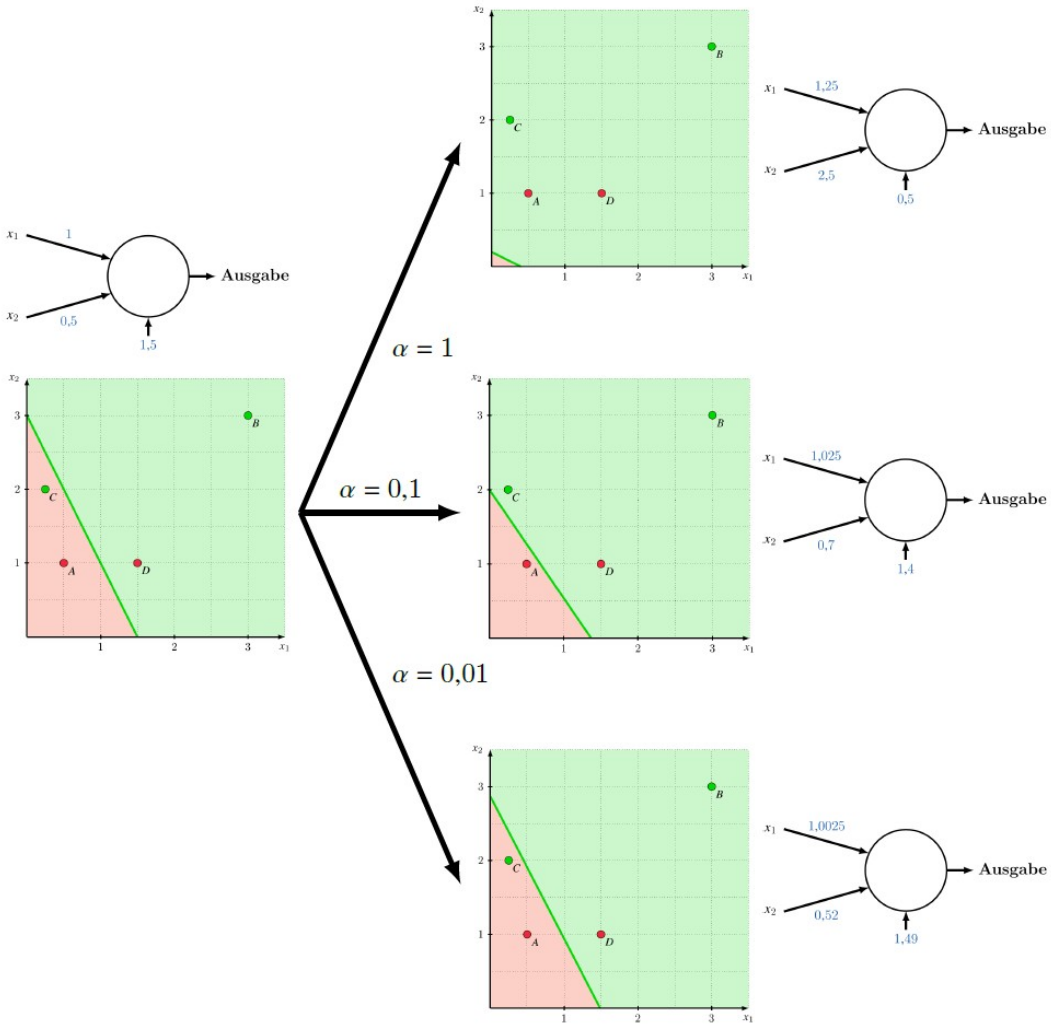
Um auf die Geschwindigkeit, mit dem das Perzeptron sich an die Punkte anpasst, zu steuern, benutzen wir einen zusätzlichen Wert, die sogenannte ***Lernrate*** α. Damit ergibt sich:

$$w_1 = w_1 + \alpha \cdot \delta \cdot i_1$$
$$w_2 = w_2 + \alpha \cdot \delta \cdot i_2$$
$$w_B = w_B - \alpha \cdot \delta \cdot 1$$

Zusammenfassend haben wir damit die ***Delta-Lernregel*** für ein Perzeptron:

$$w_k = w_k + \alpha \cdot \delta \cdot i_k$$

Beispiel für Lernrate:



Berechnung der neuen Gewichten für Punkt $C\left(\frac{1}{4};2\right)$:

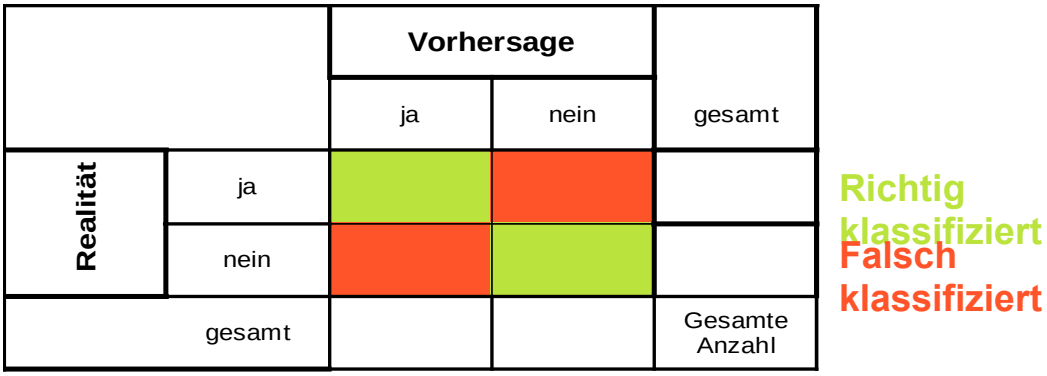
Ausgangslage der Gewichte und des Schwellenwerts:

$w_1=1 \quad w_2=\frac{1}{2} \quad w_B=1{,}5 \quad \delta=1-0=1$

$w_1=1+1\cdot 1\cdot\frac{1}{4}=1{,}25$	$w_1=1+0{,}1\cdot 1\cdot\frac{1}{4}=1{,}025$	$w_1=1+0{,}01\cdot 1\cdot\frac{1}{4}=1{,}0025$
$w_2=0{,}5+1\cdot 1\cdot 2=2{,}5$	$w_2=0{,}5+0{,}1\cdot 1\cdot 2=0{,}7$	$w_2=0{,}5+0{,}01\cdot 1\cdot 2=0{,}52$
$w_B=1{,}5-1\cdot 1=0{,}5$	$w_B=1{,}5-0{,}1\cdot 1=1{,}4$	$w_B=1{,}5-0{,}01\cdot 1=1{,}49$

Güte

Mit Hilfe der Testdaten lässt sich wiederum überprüfen, wie gut der Baum „unbekannte“ Datensätze (Testdaten) klassifiziert. Jedes Individuum der Testdaten wird vom Perzeptron klassifiziert und mit den richtigen Beschriftungen jeweils verglichen. Dies übertragen wir wieder in die ***Konfusionsmatrix***:



Wir betrachten dabei die Genauigkeit, die angibt, wie viel Prozent richtig klassifiziert sind:

$$\textbf{Genauigkeit} = \frac{Anzahl\ richtig\ klassifiziert}{Anzahl\ richtig\ klassifiziert + Anzahl\ falsch\ klassifiziert}$$

Anhand der Genauigkeit können wir ablesen, ob sich das Modell bereits zum Einsatz eignet oder wir es noch weiter trainieren müssen. Dazu können wir entweder die Trainingsdaten zufällig durchmischen und damit nochmals trainieren oder uns auf die Suche nach neuen Daten machen.

Programmierung des Perzeptrons (nur NTG)

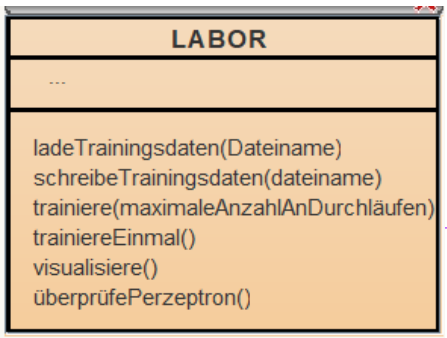
Aufgabe 14:

Verwende für diese Aufgabe die BlueJ-Vorlage Java_Perzeptron.

Das Projekt besteht aus zwei Klassen: LABOR und PERZEPTRON

Die Klasse LABOR ist bereits fertig implementiert und bietet die Steuerung für die Objekte der Klasse PERZEPTRON an. Verwende in dieser Aufgabe für Zahlenwerte ausschließlich den ***Datentyp*** double.

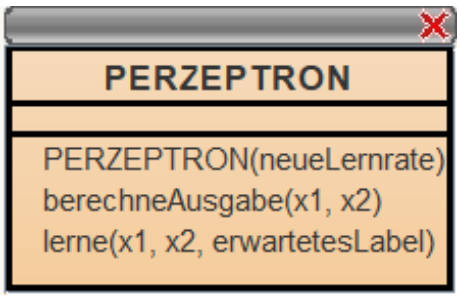
In der Klasse Labor sind die nebenstehenden Methoden realisiert. Über die Methode visualisiere() lässt sich ein 2D-Koordinatensystem einblenden, in das mit der Maus (Links- und Rechtsklick) gelabelte Punkte eingetragen



werden können, um damit das Perzeptron zu trainieren. Zugleich wird auch die gelernte Klassifikationsgerade angezeigt.

Mit Hilfe der beiden Methoden trainiere und trainiereEinmal werden alle Datensätze mehrmals bzw. einmal durchlaufen, um die Gewichte zu erlernen.

Die Klasse PERZEPTRON muss vervollständigt werden. Ändere nichts an den Signaturen der drei Methoden ab!



a) Füge notwendige Attribute ein, damit ein Perzeptron- Objekt seine Informationen speichern kann.

b) Ergänze den Rumpf des Konstruktors. Dieser erwartet als Parameter die Lernrate η.

Überprüfe anhand von Werte, ob der Konstruktor korrekt arbeitet!

c) Ergänze den Rumpf der Methode berechneAusgabe(...), die für die beiden Eingaben x_1 und x_2 das Label in Form von 0.0 oder 1.0 bestimmt.

Überprüfe anhand von Werten, ob die Methode korrekt arbeitet!

d) Ergänze den Rumpf der Methode lerne(…), die für die Eingabe eines Datenpunktes mit dem bekannten (nicht berechneten) Label einen Lernprozess durchläuft.

Überprüfe anhand von Werten, ob die Methode korrekt arbeitet!

e) Erzeuge ein neues Objekt der Klasse PERZEPTRON sowie damit ein neues Objekt der Klasse LABOR. Lass das 2d-Koordinatensystem anzeigen. Erzeuge durch Klicken mit der Maus zwei eindeutig getrennte Punktwolken (Beachte hierzu den Hinweis unter dem Diagramm). Trainiere damit einmal das Perzeptron.

Hat der Trainingsprozess nach einem Durchlauf bereits das gewünschte Ergebnis?

f) Wiederhole den Trainingsprozess mit Hilfe der Methode trainiereEinmal(), bis das Perzeptron die beiden Punktewolken sauber voneinander trennt.

g) Speichere Deinen Datensatz ab. Die Methode erwartet einen Dateinamen. Dieser darf noch nicht im BlueJ-Projekt vorhanden sein.

Tausche Deinen Datensatz mit dem eines Mitschülers aus und wiederhole damit Teilaufgabe f).

h) Erzeuge mehrmals wieder jeweils ein neues Objekt der beiden Klassen und erzeuge andere Datensätze, und trainiere damit das zugehörige Perzeptron.

Verhalten sich die einzelnen Perzeptrons immer wie erwartet?

Ausblick (optional)

Aufgabe 15:

Für diese Aufgaben wird das Tool Playground aus dem tensorflow-Framework (https://playground.tensorflow.org) benötigt. Bitte über Link öffnen (anklicken mit <STRG>) Im Link sind Einstellungen für den Einstieg gespeichert.

Mit Hilfe des Playgrounds lassen sich komplexere Situationen als ein einzelnes Perzeptron untersuchen.
Beim Start findest Du die Situation eines einzelnen Perzeptrons vor. Laut Definition wird unter einem Perzeptron aber ein Verbund mehrer künstlicher Neuronen (einzelne Perzeptrons) verstanden, die zwischen zwei Klassen differenzieren.

a) Betätige den Playbutton oben rechts und betrachte dabei schnell den Output.

b) Verändere ein paar Parameter auf der linken Seite unterhalb von Data und lass jeweils das Perzeptron trainieren.
Wie hat sich die Änderung auf den Output ausgewirkt?

c) Vergrößere nun schrittweise die Lernrate und vergleiche den Output.

d) Erweitere nun das Perzeptron um Hidden Layers. Gehe hier schrittweise vor: erst ein, dann zwei usw. Lass jeweils das Perzeptron trainieren.
Wie macht sich die Hinzunahme weiterer Neuronen auf die Ausgabe bemerkbar?

e) Verändere nun die Datenquelle zu der mit den vier Bereichen (rechts oben) und versuche mit möglichst wenig Neuronen insgesamt eine vollständige Klassifizierung zu erreichen, indem Du mit den Features experimentierst und die Anzahl der Layers bzw. jeweils der Neuronen einstellst.

f) Für Schnelle:
Wiederhole die letzte Aufgabe e) mit den beiden anderen Data-Inputs!

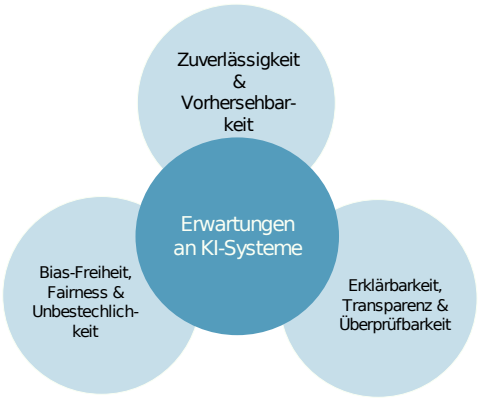
Ethische Aspekte und Auswirkungen des Einsatzes von KI-Systemen

Aufgabe 16:

Das Plakat Mensch und Maschine listet viele Anwendungen und Einsatzszenarien sowie Herausforderungen im Zusammenhang mit Computersystemen allgemein und im Zusammenhang mit KI-Systemen.

Bildet Vierergruppen und findet die für euch wichtigsten Stichpunkte auf dem Plakat. Notiert euch dabei, warum ihr diese Entscheidung getroffen habt.

Der Einsatz neuer Technologien muss immer gesellschaftlich geregelt werden. Dabei werden folgenden Schlüsseleigenschaften betrachtet:



Bei jedem Punkt muss beachtet werden, dass es sowohl positive als auch negative Einflussfaktoren existieren.

Zur Zeit stehen drei Themen besonders im Fokus

1. Verzerrung durch Bias

   Sampling Bias: Rückschlüsse aus Daten ziehen, die nicht repräsentativ sind für die Gesamtheit (Population), die man verstehen/ analysieren will

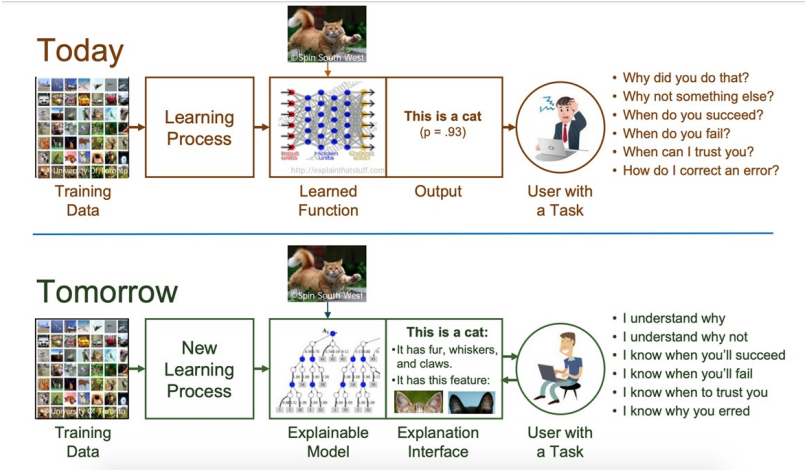
2. Zuverlässigkeit

   Manipulation hat Auswirkungen auf das Ergebnis des Entschlusses

Beide Schilder wurden als Höchstgeschwindigkeit von 85 mph erkannt.

3. Erklärbarkeit der Entscheidungen der KI-Systeme

   Die Entscheidungen werden mehr anerkannt, wenn das System auch angeben kann, warum es die Entscheidung so getroffen hat.



Aufgabe 17:

Bildet Vierergruppen und wählt eines der folgenden Themen aus! Erstellt zu eurem ausgewählten Thema ein Plakat auf einer DIN A3-Seite, das die wichtigsten Aspekte beinhaltet.

Die folgenden Links beleuchten jeweils ein Thema:

https://www.zdf.de/nachrichten/politik/ki-regeln-gesetz-ai-act-eu-ethik-experten-100.html

https://www.3sat.de/themen/kuenstliche-intelligenz-106.html

https://www.br.de/fernsehen/ard-alpha/sendungen/campus/doku/verbrechen-sicherheit-ueberwachung-campus-doku-100.html

Diskussionswürdig:

→ Personenüberwachung in China *(Video: Totale Kontrolle China setzt auf Gesichtserkennung)*