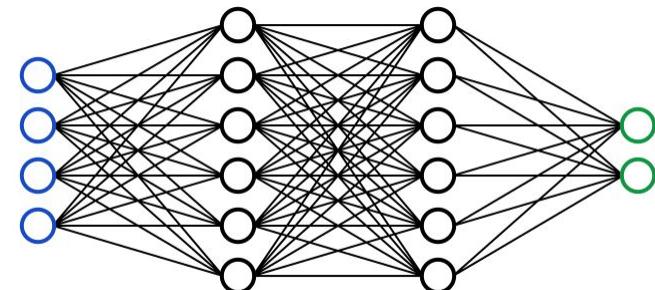
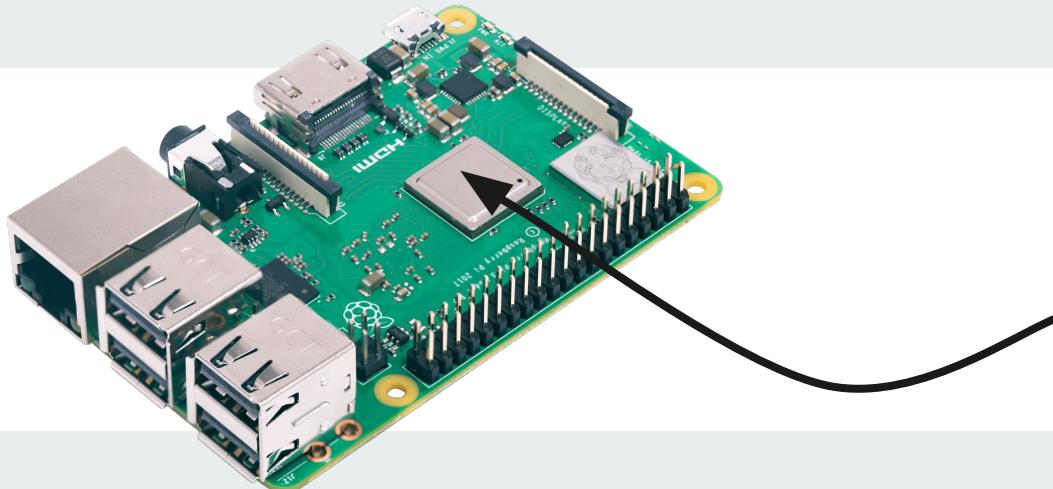

Low Power Lab

für Anwendungen mit Machine Learning

Alexander Hagg (TREE/EMT)

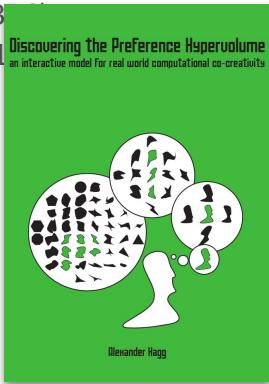


Wer bin ich?

Alex(ander Hagg)

Werdegang:

- Bachelor CS (H-BRS)
- Master Autonomous Systems (H-BRS)
- PhD LIACS, Universiteit Leiden (NL)

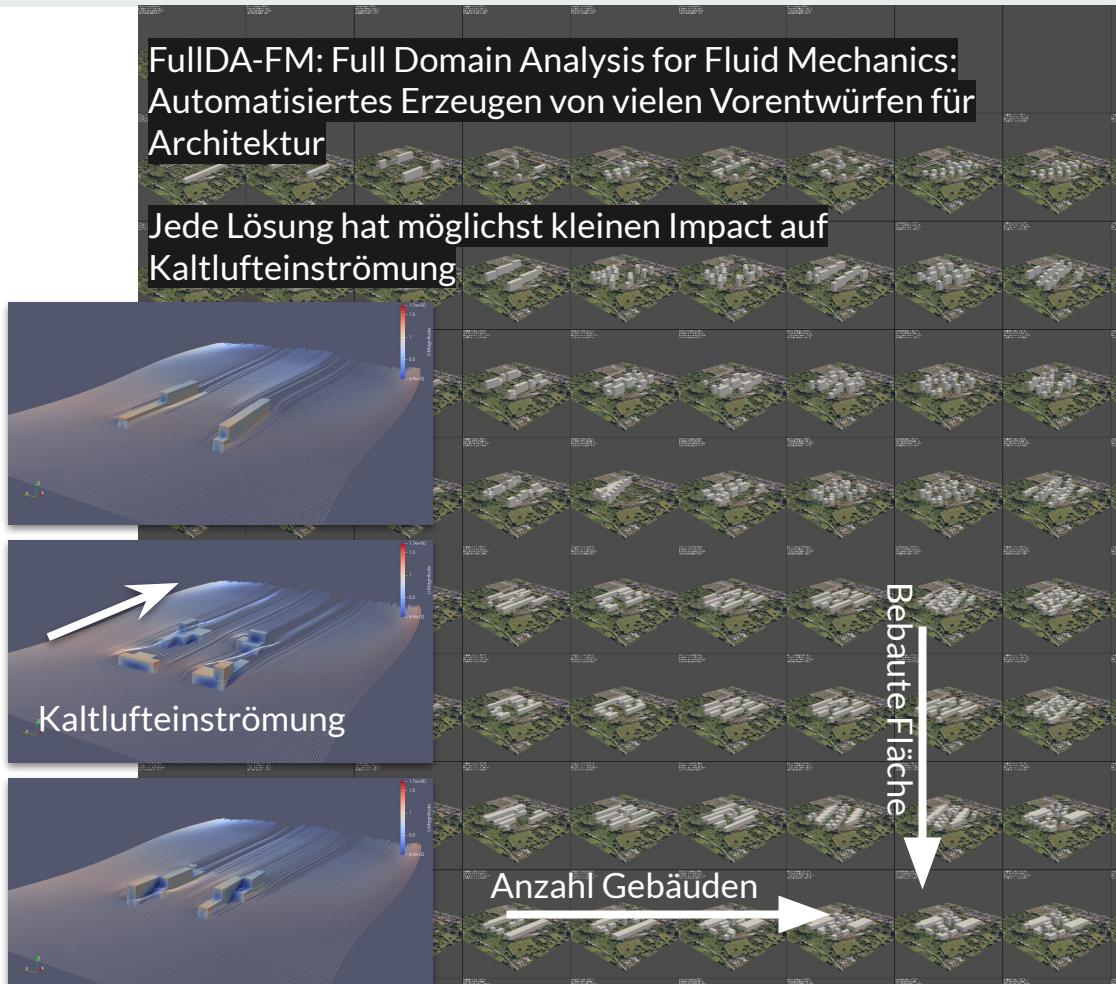


Hauptinteresse:

Kokreative Designprozesse für
IngenieurInnen

- Divergente Optimierung
- Surrogatmodellierung
- Generative ML-Modelle
- Strömungsmechanik
- Robotik (MAS, Fraunhofer IAI)
- Lehre (Genetic Algorithms, Neuroevolution, Machine Learning (Projekt), Algebra, Robotics Prog.)

alexander.hagg@h-brs.de



Wichtige Informationen



Projektarbeit in Gruppen von 2-3 Personen (10:00-17:00)

Wir werden folgendes verwenden:

1. Raspberry PI Plattform + Kameramodul
2. Linux-basiertes Betriebssystem
3. Bash Scriptsprache um Betriebssystem zu bedienen
4. Python Programmiersprache (+ externe Bibliotheken) für ML, Datenverarbeitung und Analyse

-> Wir nutzen in den Übungen hier vor allem Scikit, Scikit-learn, aber es gibt noch viele andere Bibliotheken für ML!

Welchen Output erwarte ich von euch?

1. Präsentation Woche 1: Projektidee, Anforderungen, Attrappe,
2. Präsentation Woche 2: Zwischenstand, Funktionalität zeigen
3. Präsentation Woche 3: Tage des offenen Projekts

Ziele



Eigene Kreativität und Praktische Anwendung stehen zentral!

- Grundlagen Linux, Bash, Python, Softwareentwicklung
- Überblick über Methoden des maschinellen Lernens (ML)
- Sammeln erster Erfahrungen mit Machine Learning
- Praktischer Einsatz, Sensoren, Programmierung, Algorithmen
- Erfahrung mit Hardwareeinschränkungen
- Selbstständiges Arbeiten, Teamarbeit, Präsentation und Dokumentation

Projektmöglichkeiten



1. Interaktion mit Benutzer auf Basis von Kameradaten
2. Garrulus: Segmentierung oder Klassifizierung von Pflanzen für Dronenflüge
3. Eigene Projektideen sind gewollt!
 - a. Durch Eigeninteresse/Hobby
 - b. Durch Verbindung mit einer anderen Veranstaltung
 - c. Inspiration Anwendungen anderer Menschen (online/github/...)

! In diesem Projekt geht es darum, **erste** Praxiserfahrungen mit ML zu bekommen, nicht um Vollzeitsoftwareentwickler zu werden.

! Das Ziel des Projekts ist es, maschinelles Lernen für etwas **Nützliches** (oder **Lustiges**) zu verwenden

Beispielprojekte

Vortrainierte Modelle

Gesichtsmaskenerkennung - <https://github.com/inmicro/Face-Mask-Detection-Raspberry-pi-2021>

Audio/Spracherkennung (Einzelne Wörter) - https://github.com/nyumaya/nyumaya_audio_recognition

Objekterkennung - https://github.com/jiteshsaini/model_garden

Noise suppression and acoustic echo cancellation - <https://github.com/SaneBow/PiDTLN>

Virtual Assistant - <https://github.com/reywahl/Assistant>

Deep-Learning-based examples - <https://qengineering.eu/deep-learning-examples-on-raspberry-32-64-os.html>



Modelle müssen trainiert werden

Audioklassifikation - <https://github.com/GianlucaPaolocci/Sound-classification-on-Raspberry-Pi-with-Tensorflow>

Gesichtserkennung - <https://github.com/kunalyelne/Face-Recognition-using-Raspberry-Pi>

Audio/Spracherkennung (Einzelne Wörter) - <https://github.com/ShawnHymel/tflite-speech-recognition>

Handschrifterkennung mit virtuellem Stift - https://github.com/coding-ai/raspberrypi_handwritten_recognition

Weitersuchen:

z.B. <https://github.com/search?p=7&q=raspberry+machine-learning&type=Repositories>

(Semi-)Autonomes Wiederaufforstungsprojekt mit Dronen und selbstentwickeltem Sensorarray

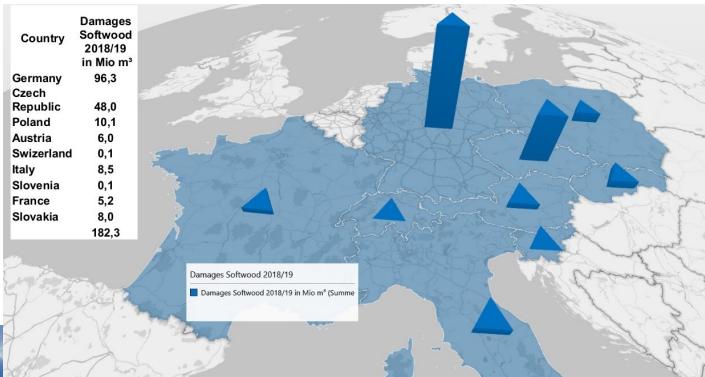


Hochschule
Bonn-Rhein-Sieg

University of Applied Sciences



Ministry for Environment, Agriculture,
Conservation and Consumer Protection
of the State of North Rhine-Westphalia





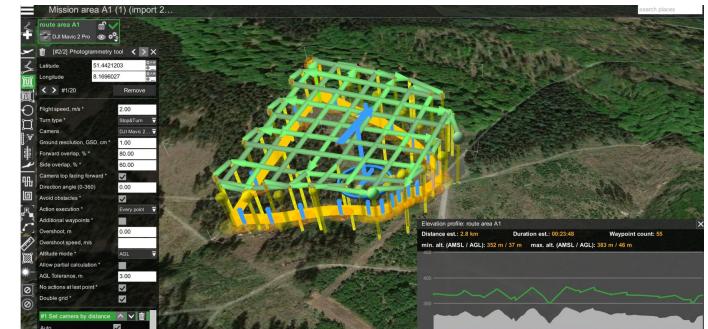
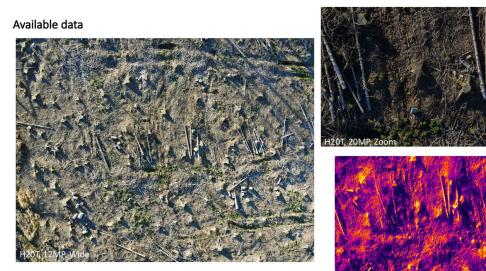
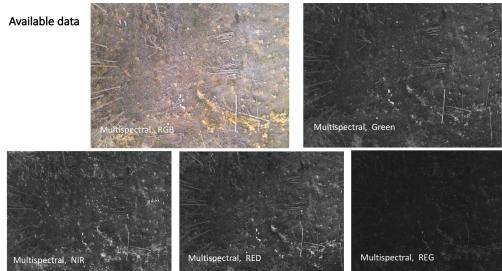
Ministry for Environment, Agriculture,
Conservation and Consumer Protection
of the State of North Rhine-Westphalia



Nutze Machine Learning um Bilddaten der Drone zu kategorisieren.

Dies ist eine **forschungsnahe** Arbeit, und bedarf ein hohes Maß an **Eigeninteresse**

Ansprechpartner Ahmad Drak (english only), der erst in ein paar Wochen wieder da ist. Daten habe ich aber.



Überblick Woche 1



Tag	Vormittag	Nachmittag	
Montag	VL: Einleitung Machine Learning VL: Projektplanung	<ul style="list-style-type: none">- Raspberry Pi anschließen, Linux kennenlernen- Repository und virtuelle Umgebung anlegen mit Bash commands- Übung 1 und 2: Segmentierung (scikit-learn, scikit-image)- Gruppen bilden aus 2 oder 3 Personen, Ideenentwicklung	16 Uhr: Ideen präsentieren
Dienstag	VL Clustering, Klassifikation, Regression	<ul style="list-style-type: none">- Übung: Klassifikation mit einfaches neuronalen Netzwerk (sklearn.neural_network)- Gruppen: Projektthema bestimmen, Planung (Arbeitspakete), notwendige externe Bibliotheken, (inkl. optional benötigte HW, selber besorgen)	16 Uhr: Themen präsentieren, inkl. optionale HW (mit Angabe, wo ihr die HW herbekommt)
Mittwoch	Gruppen: Attrappe/Mockup erstellen		
Donnerstag	Gruppen: Attrappe/Mockup erstellen		
Freitag	Gruppen: Präsentation vorbereiten: <ul style="list-style-type: none">- Idee- Anforderungen- Erste Ergebnisse- Schwierigkeiten, Planung für die nächsten 2 Wochen (Arbeitspakete, Arbeitsverteilung)		15 Uhr: Attrappe/Mockup präsentieren

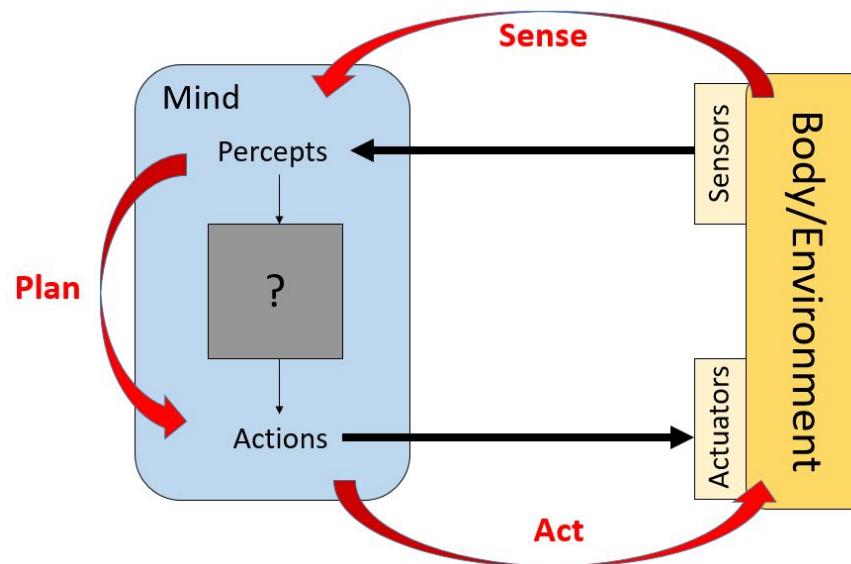
Woche 2: Projektplan selbstständig durcharbeiten, Arbeitspakete implementieren, testen,
-> **Freitag 15 Uhr: grobe Funktionalität präsentieren**

Woche 3: Ergebnisse evaluieren und Kurzpräsentation vorbereiten.
-> **Projektpräsentation Tag des Offenen Projekts am Freitag**

Montag: Einführung Machine Learning

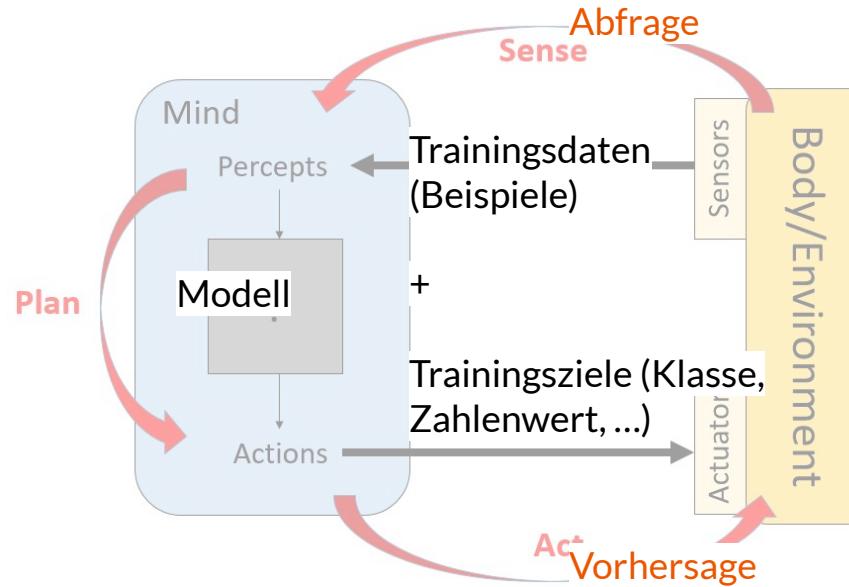
Was ist Künstliche Intelligenz?

Künstliche Intelligenz = eine Interaktionsschleife!



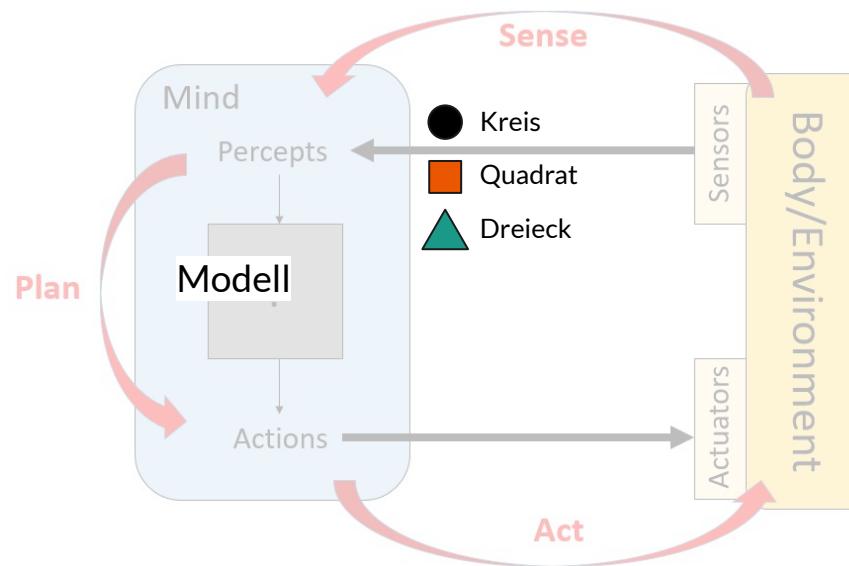
Was ist Machine Learning?

Forschungsfeld, das Computern die Fähigkeit gibt, zu lernen, ohne explizit programmiert zu werden.



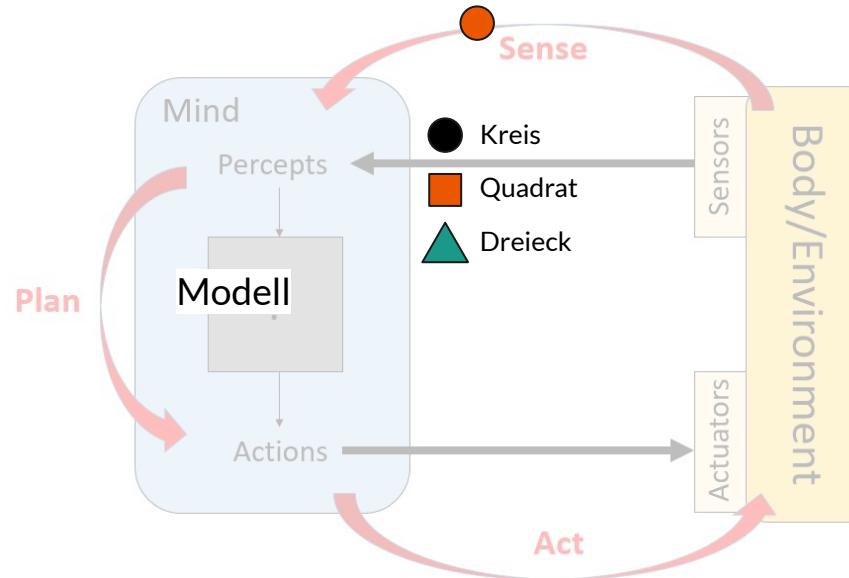
Was ist Machine Learning?

Forschungsfeld, das Computern die Fähigkeit gibt, zu lernen, ohne explizit programmiert zu werden.



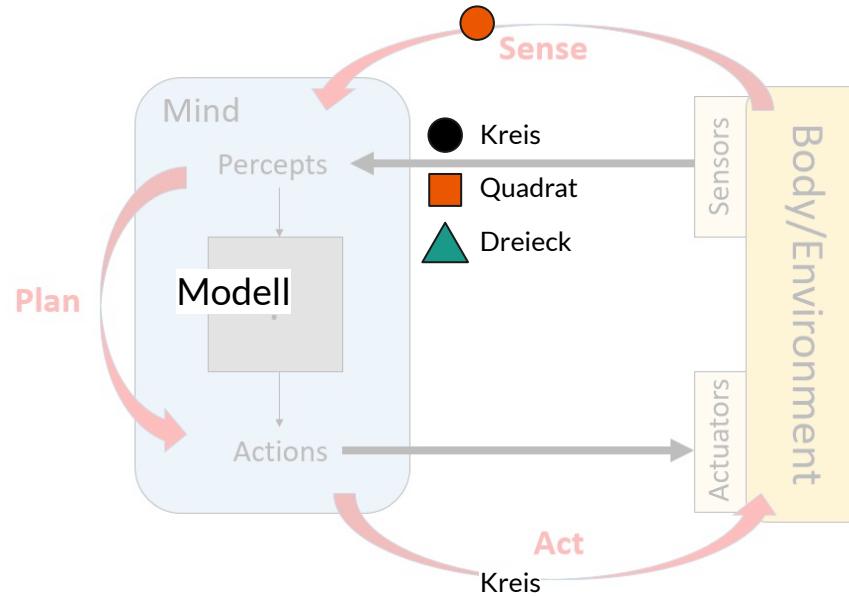
Was ist Machine Learning?

Forschungsfeld, das Computern die Fähigkeit gibt, zu lernen, ohne explizit programmiert zu werden.



Was ist Machine Learning?

Forschungsfeld, das Computern die Fähigkeit gibt, zu lernen, ohne explizit programmiert zu werden.

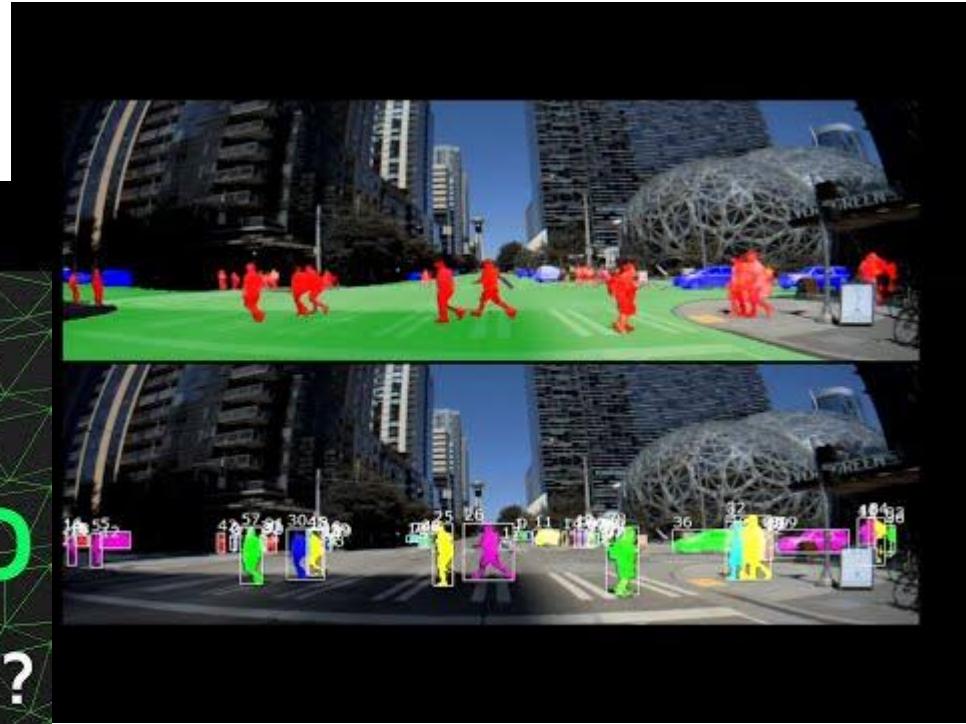
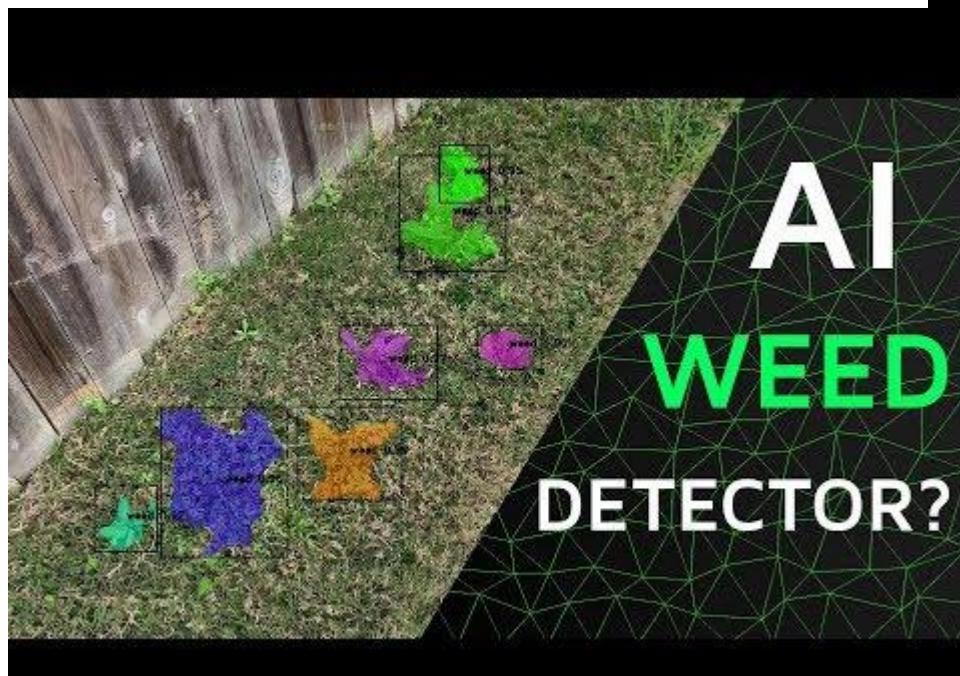


Aufgaben

- Segmentierung und Clustering
- Klassifizierung
- Dimensionsreduktion
- Anomaliedetektion
- Regression
- Zeitreihenvorhersage
- Navigation/Pfadplanung
- Kontroll/Steuerung
- usw. usf.

Segmentierung

Einteilen einer Datenmenge in
zusammenhängenden Regionen auf Basis
bestimmter Eigenschaften

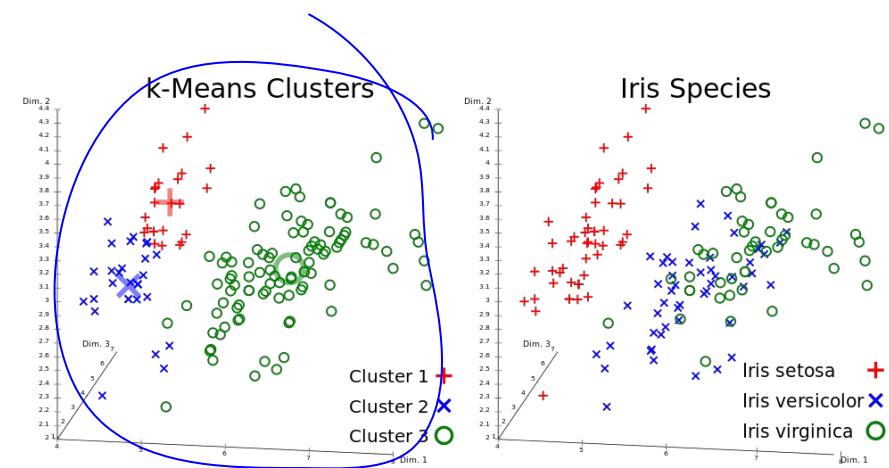
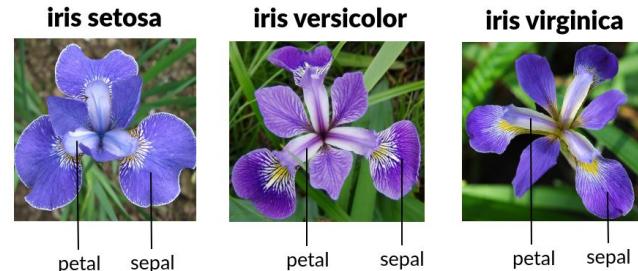


NVIDIA Drive Labs

Clustering

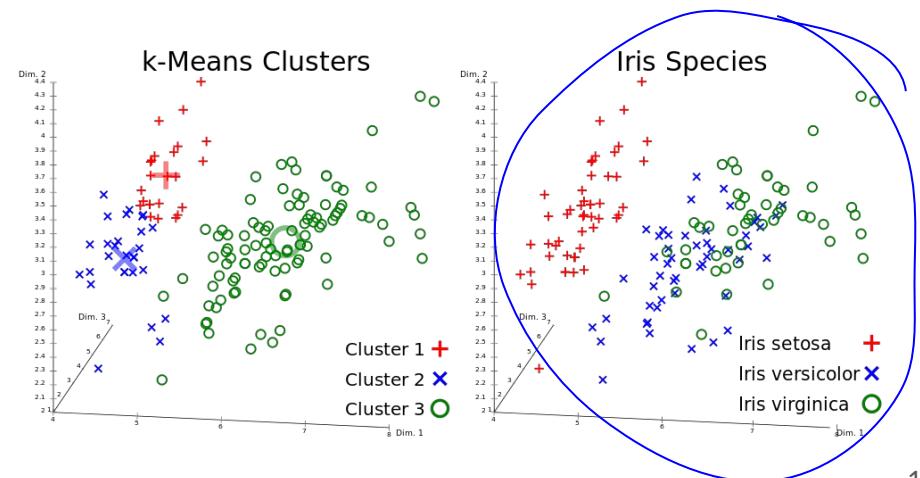
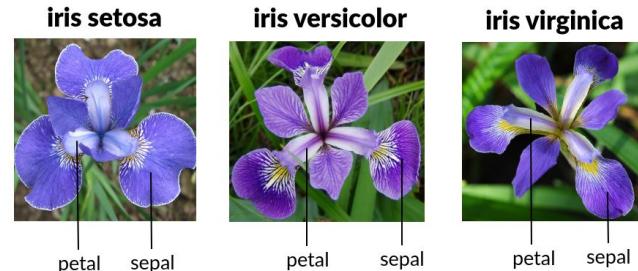
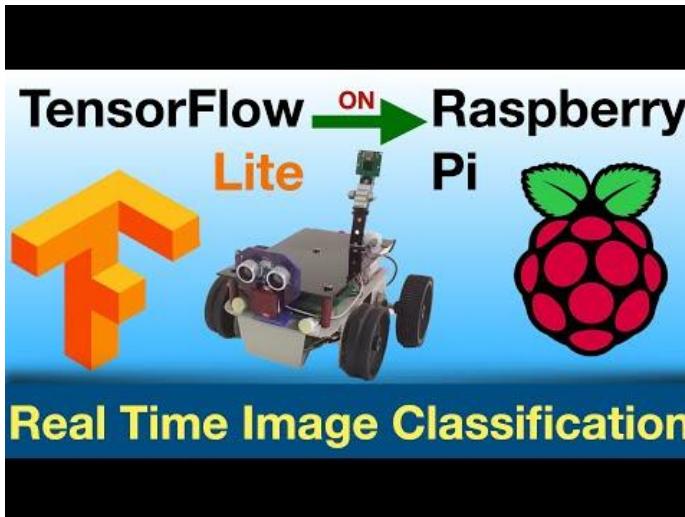
Einteilen einer Datenmenge in zusammenhängenden Untermengen auf Basis ihrer unterliegenden Eigenschaften.

*DBSCAN als Alternative



Klassifizierung

Zuweisen eines beschreibenden Labels zu einem Datenpunkt auf basis von bekannten Datenpunkten.

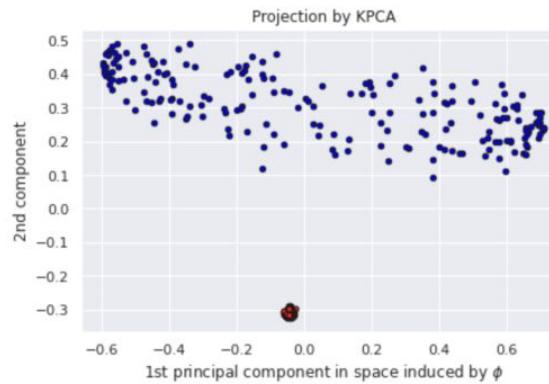
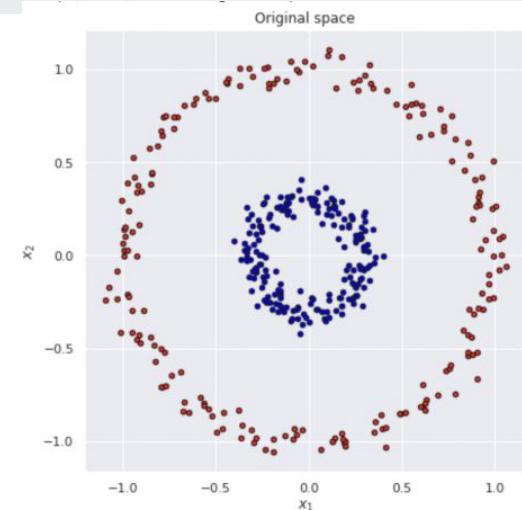


Dimensionsreduktion

Reduktion der Anzahl der benötigten Variablen um einen Datensatz zu beschreiben.

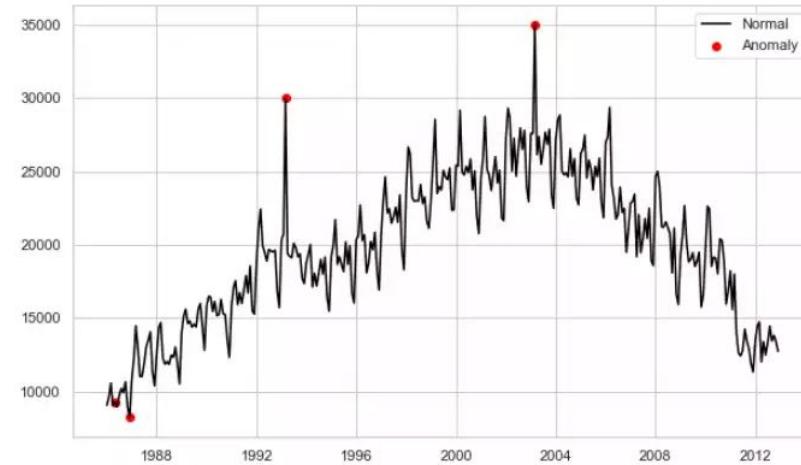
Gleichzeitig versucht man, die Qualität der Daten möglichst gleich zu halten.

- Erleichtert andere Aufgaben, wie Klassifizierung
- Ermöglicht Visualisierung, wenn auf wenigen Dimensionen reduziert wird ($2-x$)



Anomaliedetektion

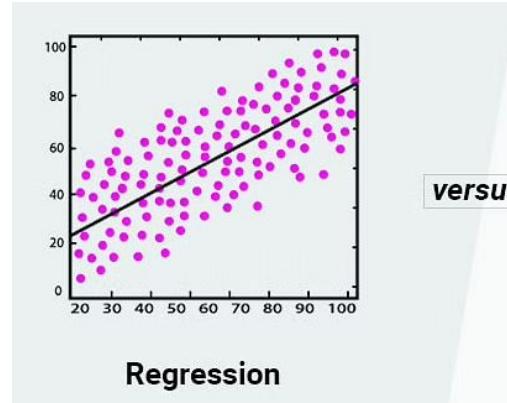
Erkennen irregularer Signalen innerhalb eines Trägersignals



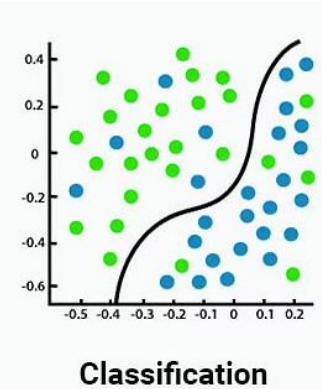
Regression

eine Technik zur Untersuchung der Beziehung zwischen unabhängigen Variablen oder Merkmalen und einer abhängigen Variablen oder einem Ergebnis.

Es wird als Methode zur **prädiktiven** Modellierung beim maschinellen Lernen verwendet, bei der ein Algorithmus verwendet wird, um kontinuierliche Ergebnisse **vorherzusagen**.



Regression



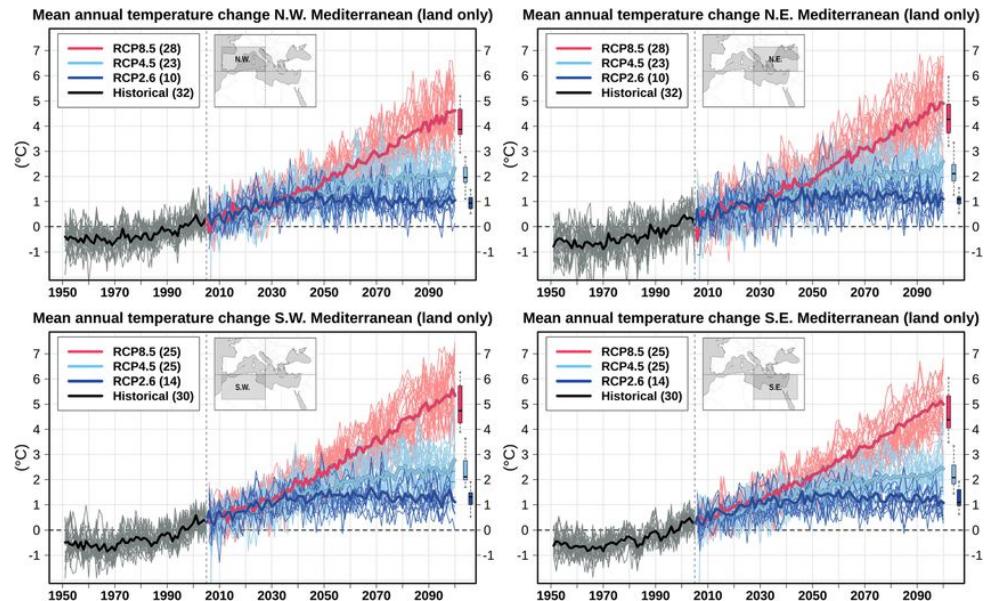
Classification

Klassifizierung: sage Klassen vorher
("Tomate oder Apfel?")

Regression: sage eine Zahl vorher
("")

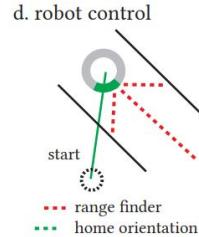
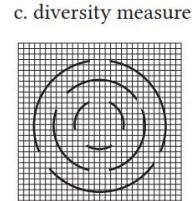
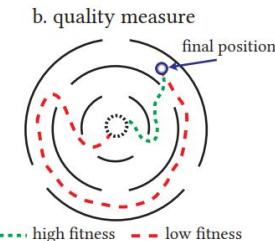
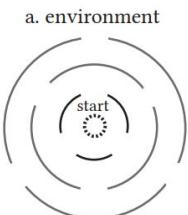
Regression: Zeitreihenvorhersage

Vorhersage von zukünftigen Ereignissen auf Basis von historischen (diskreten) Daten.



Navigation

- Kartografierung
- Lokalisierung
- SLAM: Simultaneous Localization and Mapping
- Pfadplanung (kürzester/effizientester Weg von A-B)
- Kollisionsdetektion
- Kollisionsvermeidung (lokale Pfadplanung)



Hagg, A., Zaefferer, M., Stork, J., & Gaier, A. (2019, July). Prediction of neural network performance by phenotypic modeling. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 1576-1582).



SLAM. Courtesy: Universität Stuttgart.

https://www.ifp.uni-stuttgart.de/en/research/photogrammetric_computer_vision/SLAM/

Kontroll/Steuerung



Carnegie Mellon University Robotics Institute

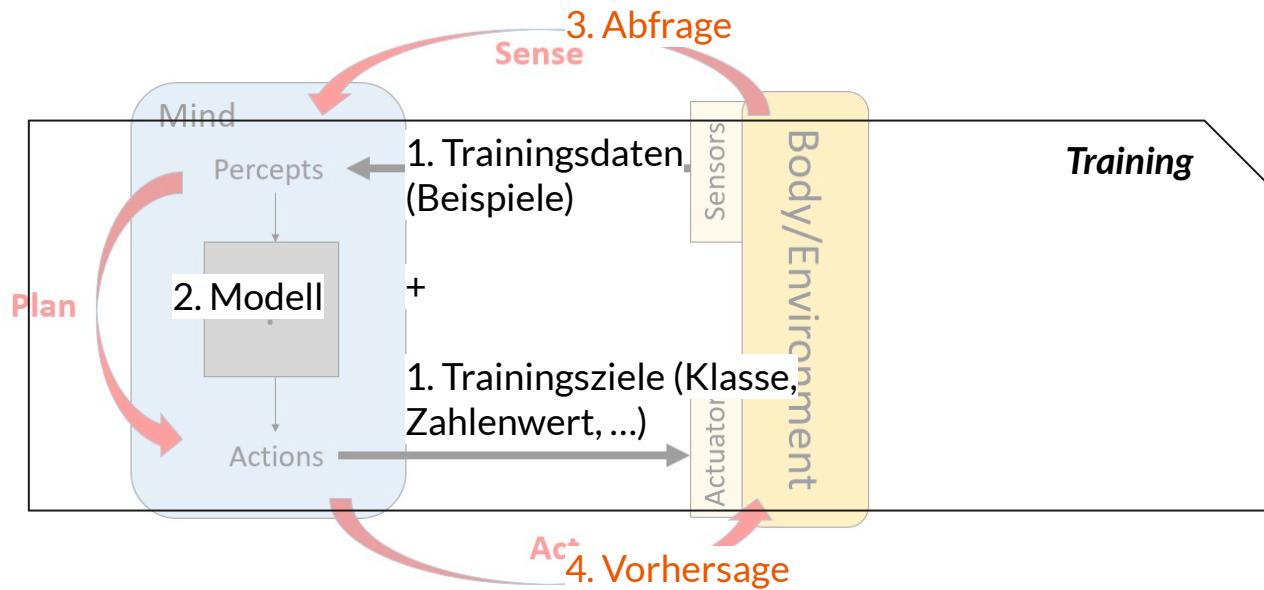


INRIA Nancy, France (Mouret et al)

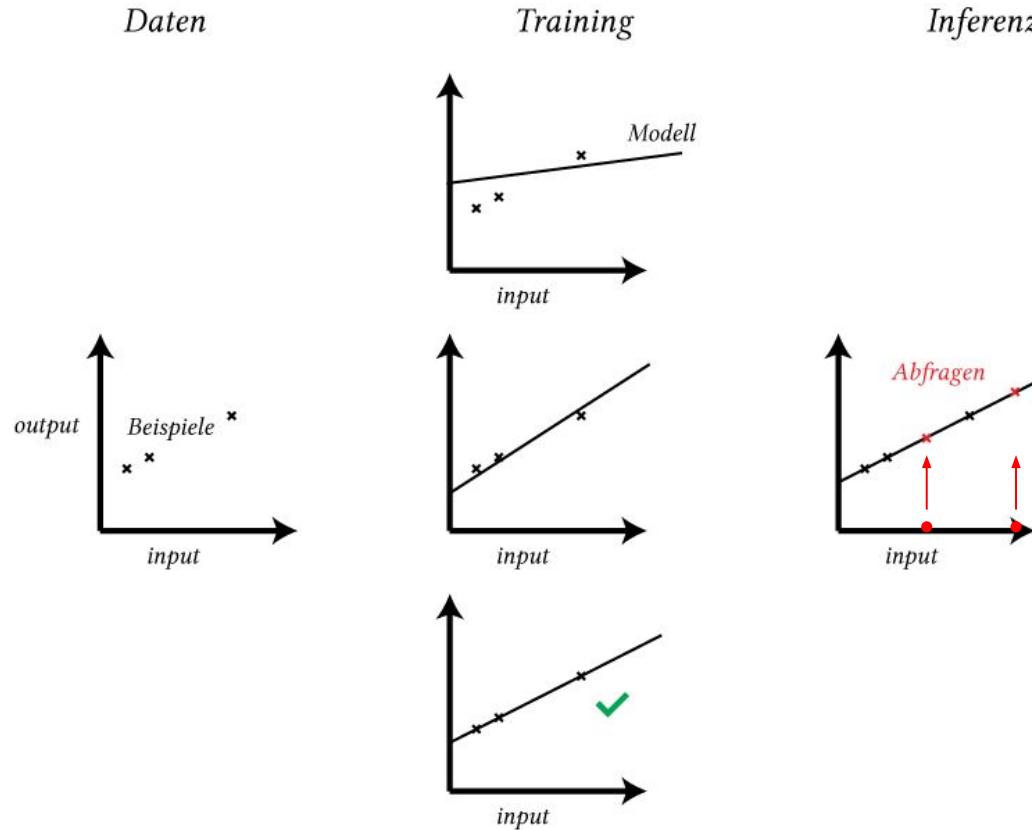
Ansätze

Ansatz	Anwendung	Beispiel
Supervised Learning: Daten und Labels (Input und Output) sind bekannt	Klassifikation, Regression	Zuordnung Namen zu neuen Fotos lernen, auf Basis vieler Profilfotos und Namen bei Facebook
Unsupervised Learning: Daten ohne Labels/Output: kann ich Struktur erkennen?	Segmentierung, Clustering, Anomaliedetektion	Fahrradfahrer in Kameradaten erkennen für autonomes Fahren
Reinforcement Learning: Agent lernt selber eine Policy um Belohnung zu maximieren	Nur Belohnungsmechanismus vorhanden	Kontrollstrategie für Roboter finden
Semi-supervised Learning Daten haben teilweise Labels	Mix aus Supervised und Unsupervised	

Supervised Learning



Supervised Learning



Ansätze

Künstliche Neuronale Netzwerke

- Perceptron
- Multilayer Perceptron
- Recurrent NN
- Convolutional NN
- Deep Learning

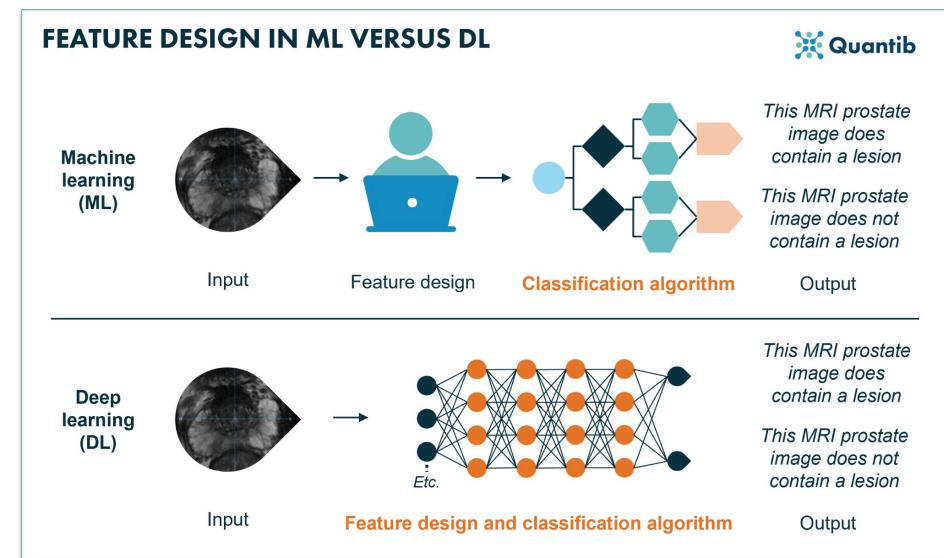
Decision Trees

Lineare Regression

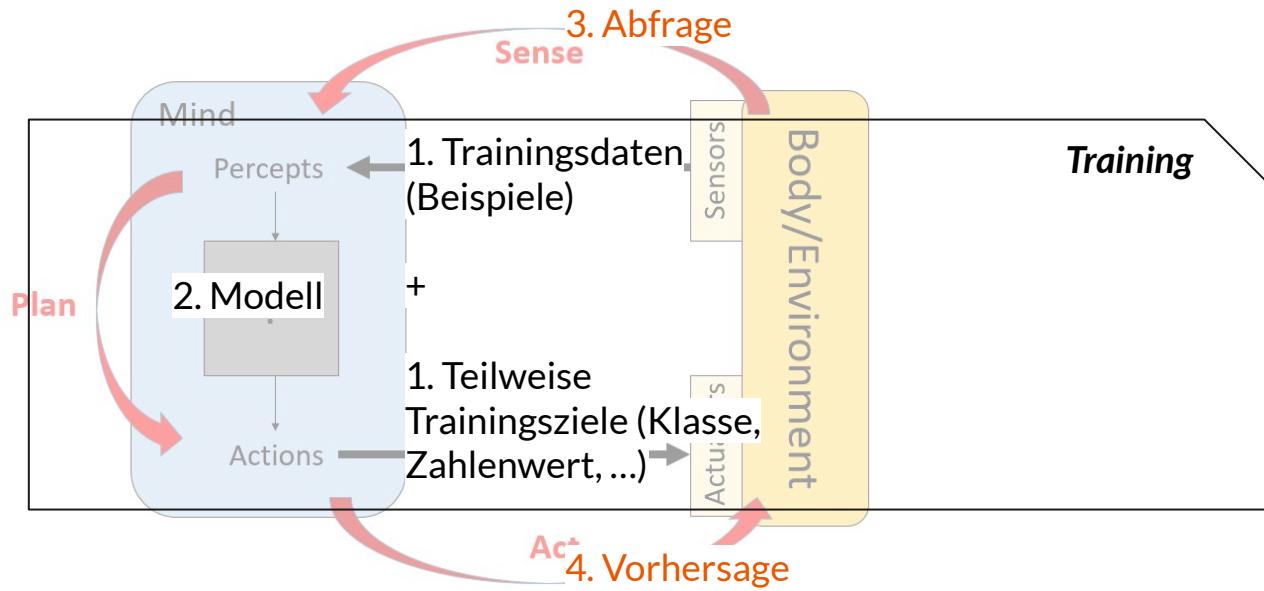
Random Forest

Support-vector Machines

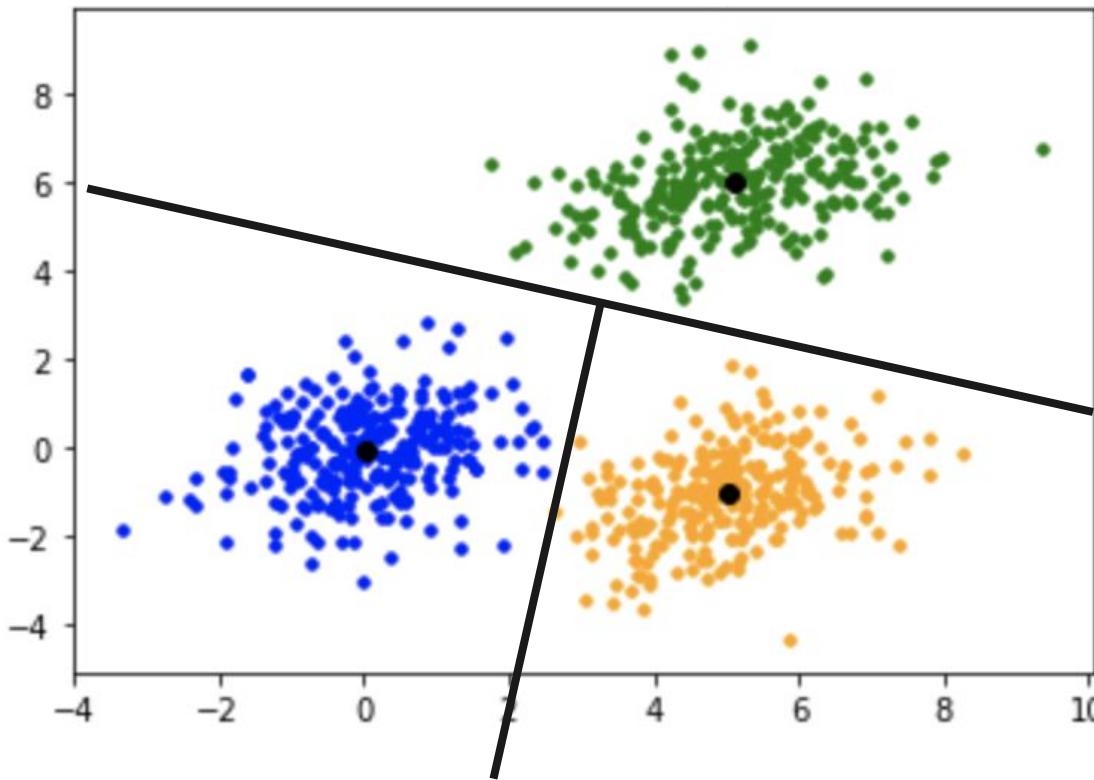
Bayesian Networks



Unsupervised Learning



Unsupervised Learning



Ansätze

k-Means

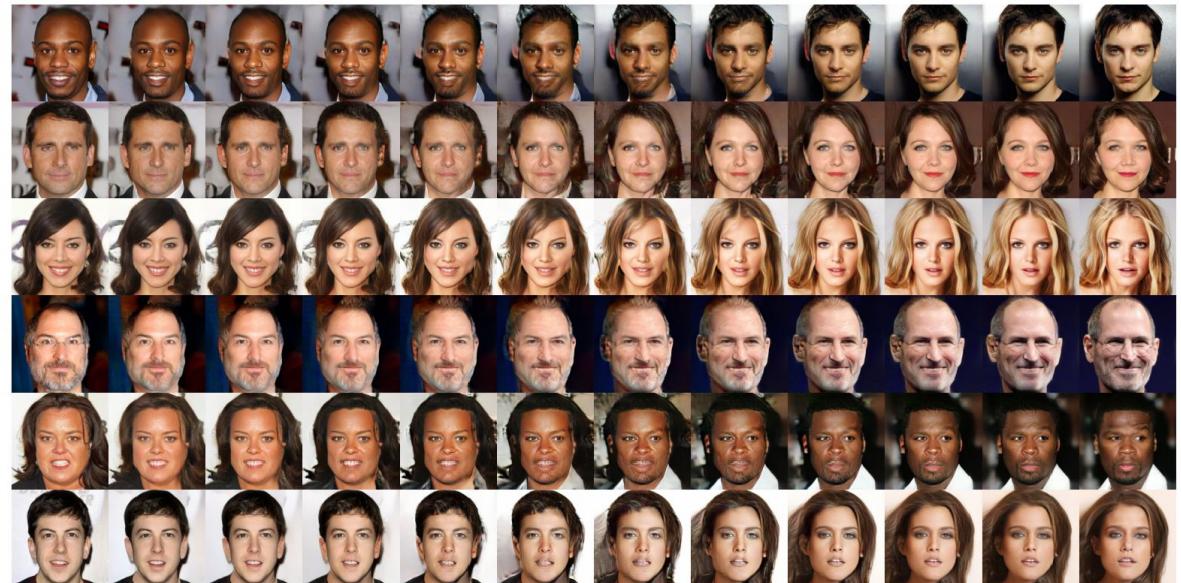
DBSCAN

Variational Autoencoder

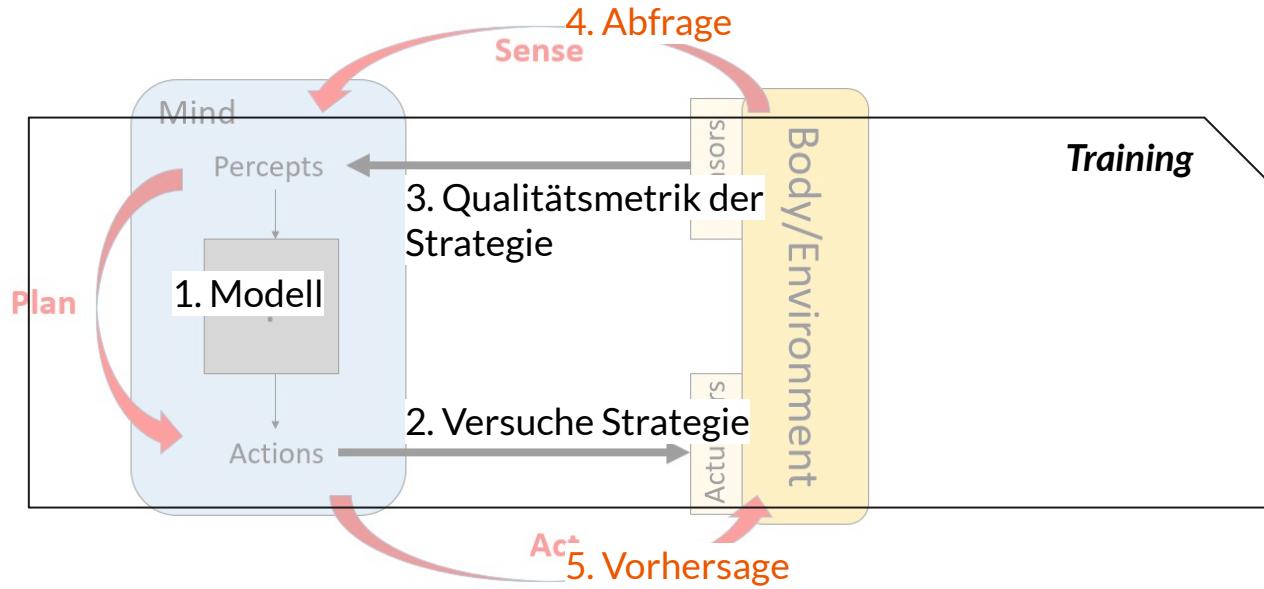
Generative Adversarial Networks

Principal Component Analysis

*GAN/VAE kodieren sehr hochdimensionale
Datensätze in glatte, niedrigdimensionale latente
Räume.*

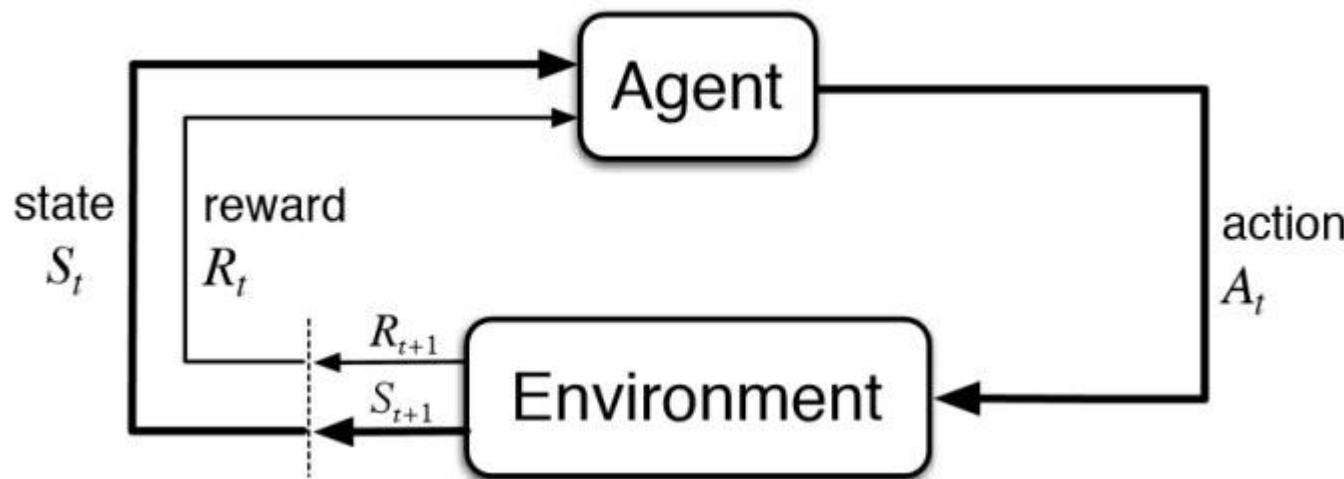


Reinforcement Learning



Reinforcement Learning

Übliche Terminologie und Denkmodell



Ansätze

Monte Carlo

Q-learning

Trust Region Policy Optimization

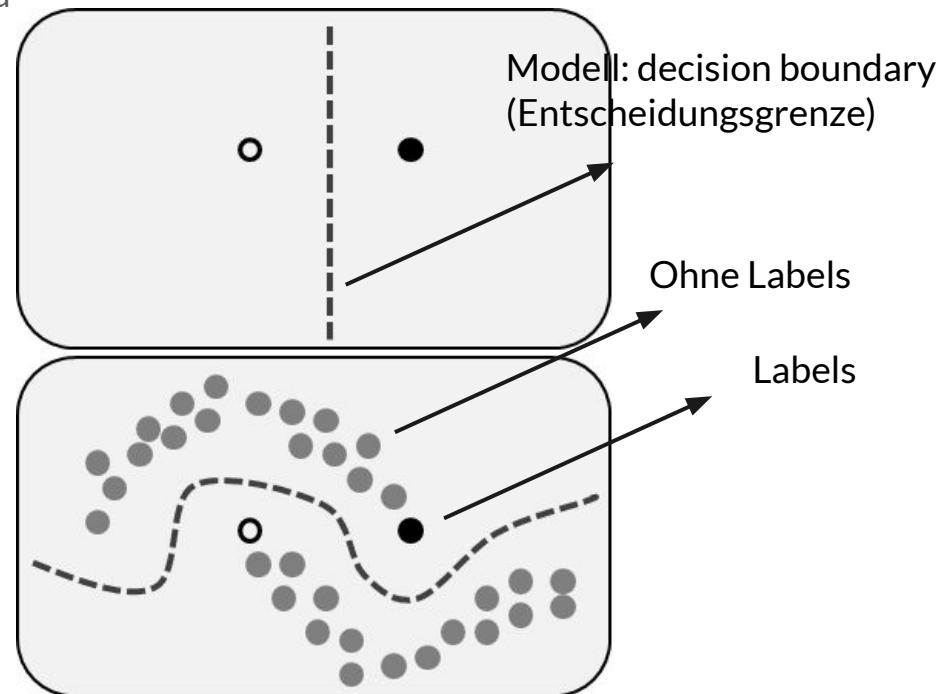
...

OpenAI won 7,215 games and lost just 42 times to human players, equating to a 99.4% overall winrate.



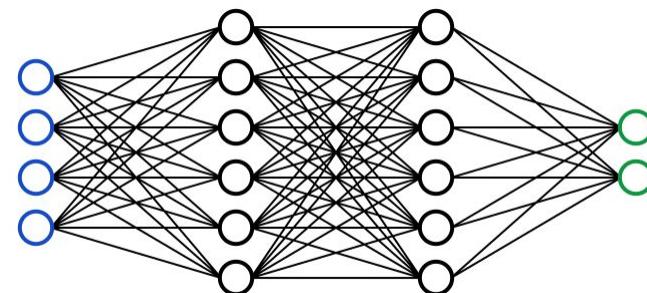
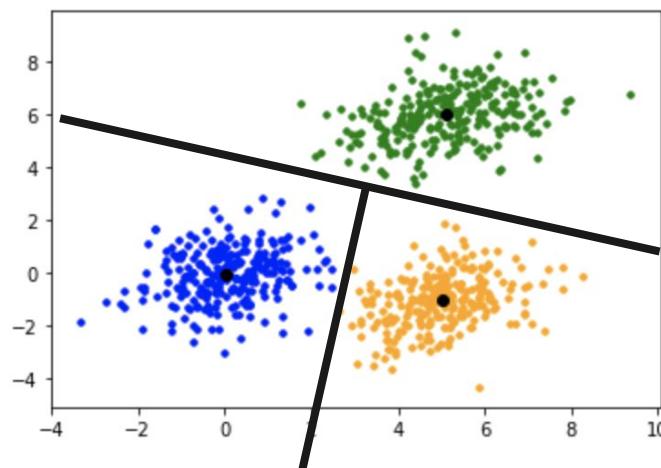
Semi-supervised Learning

Es ist oft nicht möglich oder mit sehr viel Aufwand verbunden, alle Daten zu labeln.



Anmerkungen

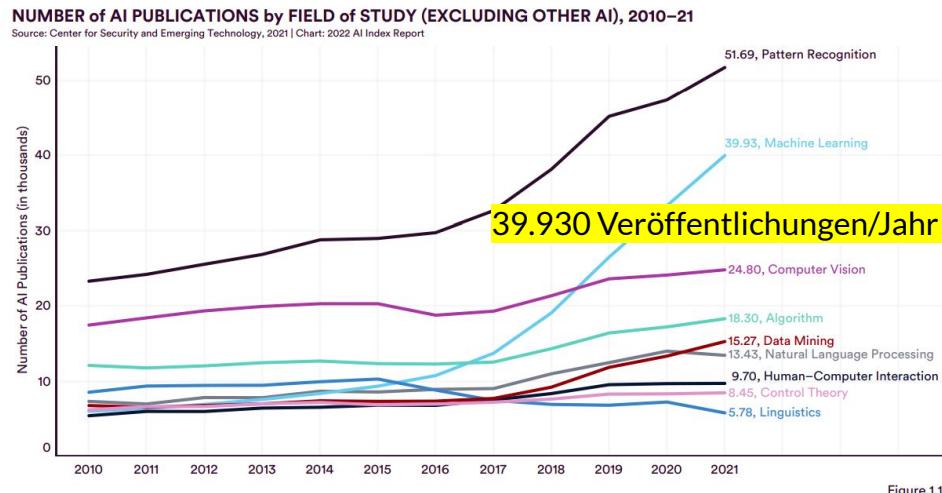
Wir werden in diesem Projekt vor allem Methoden des **Unsupervised** und **Supervised** Learning verwenden (wie z.B. k-means und neuronale Netzwerke)



Anmerkungen

Machine Learning ist ein gigantisches Feld

Arbeiten mit Machine Learning heißt oft, sich selbstständig in Themen einzuarbeiten.



aus: 2022 AI Index Report

So auch in diesem Projekt. Ich gebe euch Beispiele und ein paar kleine Übungen, damit ihr einen Anfang machen könnt.

Aber danach arbeitet ihr euch selbstständig ein. Ich bin da um zu helfen!

Projektentwicklung

Projektplanung

1. Problemstellung

- a. Welche Anwendung?
- b. -> Welches Lernproblem möchte ich lösen
- c. Was für ein Problem ist es? (Supervised,...)
- d. Formulierung Ziel (Minimierung MSE-Testdaten)

2. Ansatz

- a. Welche Methoden kommen infrage?
 - i. Menge der Trainingsdaten
 - ii. Effizienz Inferenz auf Raspi
- b. Selektionskriterien Modell aufstellen
- c. Modell selektieren (mit Argumentation)
- d. Trainingsmethode selektieren

3. Training

- a. Welche Trainingsansätze sind üblich?
- b. Was brauche ich dafür?

4. Evaluation

- a. Evaluationsmetriken (Lossfunktion, MSE, Accuracy/Precision/Recall,...)

Weitere Fragen:

- Ist das Projekt **machbar**? Wir haben nur 3 Wochen Zeit!
- Brauche ich **extra Hardware** und woher bekomme ich diese?
- Kann ich das Projektziel in Teilzielen aufteilen, damit ich, wenn etwas nicht schnell genug klappt, **Ausweichmöglichkeiten** habe?

Prototypgetriebene Projekte

Stufe 1: Attrappe (Ende Woche 1)

- Schnittstelle ist definiert (Input und Output, Datentyp)
- Komponenten und Funktionen enthalten fake Ausgaben

Struktur so schnell wie möglich aufbauen, auch wenn noch nicht alle Komponenten da sind. Stufe 1 ist eine Referenz für das weitere Arbeiten.

Stufe 2: Prototyp (Ende Woche 2)

“Funktioniert, aber nicht immer, gibt noch Fehler”

- Komponenten und Funktionen implementiert
- (evt. Training ML-Modell abgeschlossen)

Stufe 3: Finales Produkt (W3)

Anforderungen sind erfüllt
Produkt kann ausgeliefert werden

- Evaluation Funktionalität
- Nachbesserung
- (evt. bestes ML-Modell gewählt)

Entwicklungsumgebung: Raspberry Pi + OS, Bash, Git, Python, pip & Colab

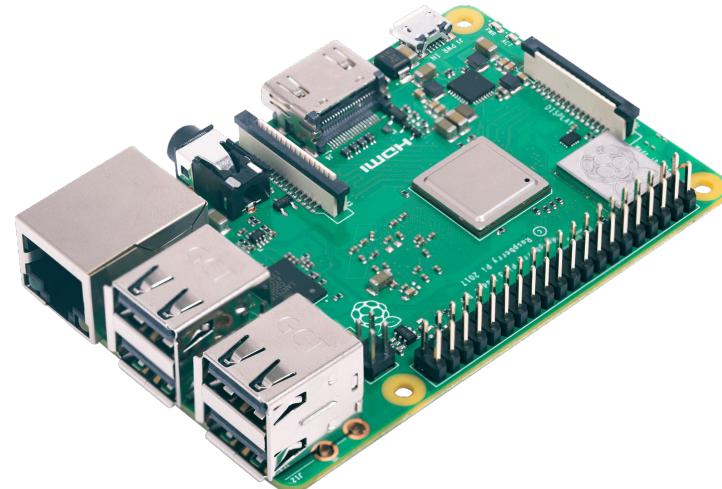
Effizienz

In diesem Projekt untersucht ihr, wie ihr ML-Modelle auf dem Raspberry PI verwendet.
Was schaffen wir mit nur 5W und 1GB RAM?

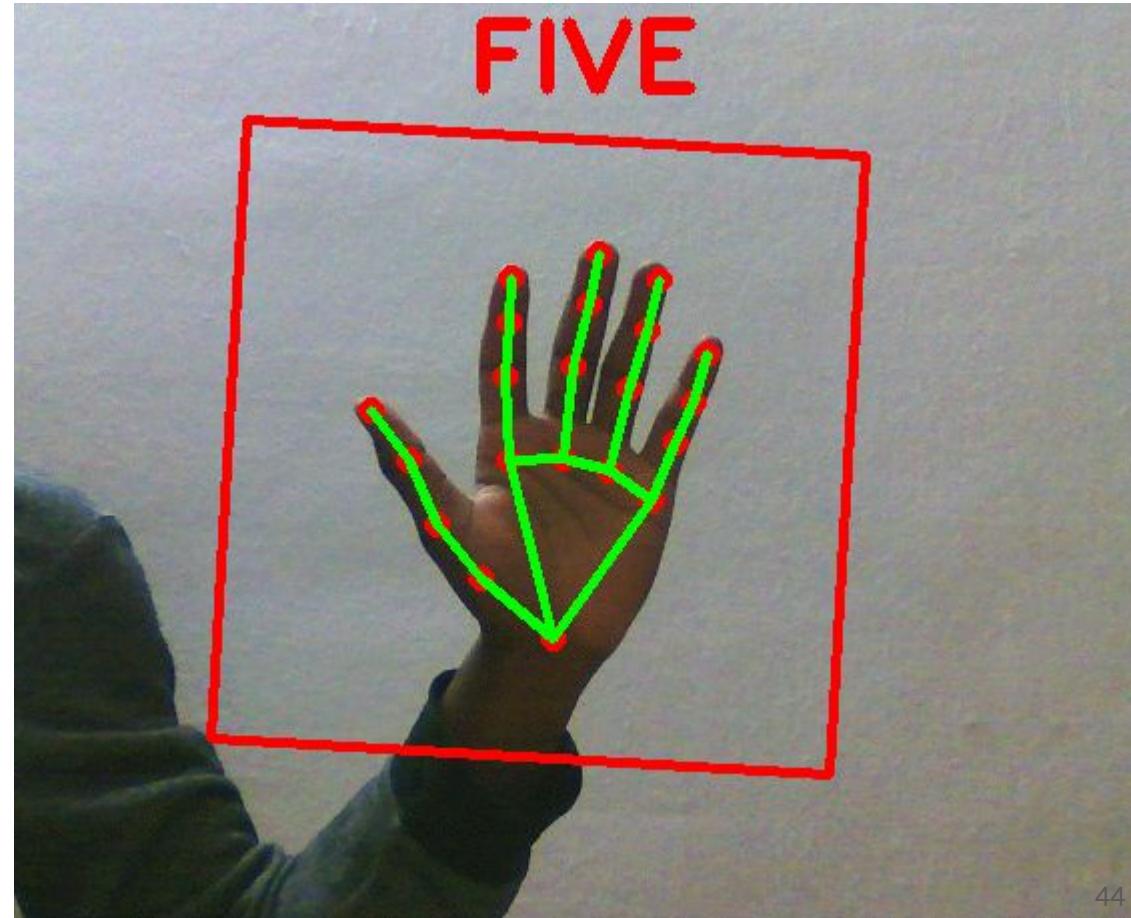
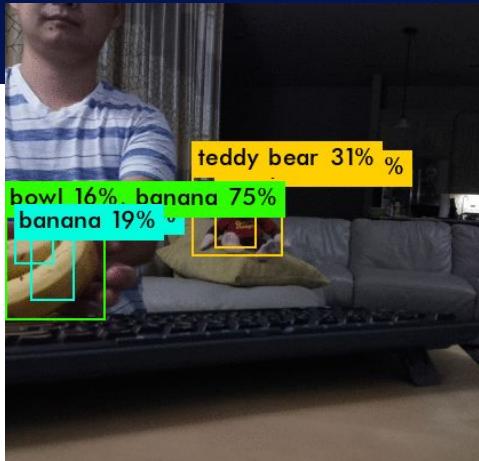
Es geht hierbei um Inferenz, nicht das Training

Was spielt hierbei eine Rolle?

- Frequenz der Abfragen
- Modellgröße/Komplexität
 - RAM
 - CPU/GPU flops
- Training wird meistens ausgelagert!
 - Auf ein schnelleres System
 - z.B. Google Colab



Beispiele Machine Learning mit Raspberry





Raspberry PI ist für das Training oft nicht geeignet (zu wenig Arbeitsspeicher), deshalb:

- Vortrainierte Modelle verwenden
- Training auf Rechner/Laptop
- Training auf Google Colab (Rechnen in the cloud)

Das Raspberry Pi ist also eher für **Inferenz** geeignet!

Das ist üblich in der Industrie: die Modelle werden auf Großrechner trainiert und können dann auf dedizierter Hardware verwendet werden.

Tools

- Raspberry OS stellt bereit:
 - CPU/GPU)
 - Storage
 - Treiber Input-Hardware (Kamera)
- Bash
 - Konfiguration von System
 - Aufruf von Python und Anwendungen
 - Scripts schreiben, die "Dinge tun"
- Python
 - Aufbereitung von Daten
 - Training
 - Inferenz
- Softwarebibliotheken
 - Scikit-Learn, Pytorch, usw.
- Pip/venv/virtualenv
 - Bibliotheksumgebung
 - Ermöglicht es, leicht aktive Bibliotheken auszutauschen
- Git/Github/Gitlab
 - speichern und verwalten von Code und Scripts und Konfigurationsdaten
 - remote/cloud

Bash Kommandozeilenscripting

- Verbreiteste Scriptsprache auf Linux OS
- Sehr mächtig: man kann hiermit das OS konfigurieren, steuern, aber auch kaputt machen
- Don't try this at home: **rm -rf ***: forced removal of files and directories (rekursiv)

Beispielaufruf eines Pythonprogramms mit Bash

```
python run_classifier.py image.png
```

- > Benutze den Python Interpreter um das Script
- > "run_classifier.py" auszuführen mit einer weiteren
- > Parameterübergabe der Zeichenabfolge "image.png"

Git Versionsmanagement

Git ist eine Anwendung mit der wir einfach unseren Code und Scripts managen können

- Diese können als repository bei Github oder einer Gitlab-Instanz gespeichert werden.
- Jede Änderung kann nachverfolgt werden
- Code kann geteilt werden mit anderen Programmierern und Anwendern
- **Hilfreich, wenn das Training auf einem anderen Rechner erfolgt**
- Auf Github erlangt man außerdem Sichtbarkeit, wenn das Repository als “public” gekennzeichnet wird. Repositories können von anderen Entwicklern gefunden werden.

```
git clone git@github.com:alexander-hagg/lpl-test.git  
cd lpl-test  
touch testfile.md  
git add testfile.md  
git commit -m "updated a test file"  
git push
```

```
> Lokale Kopie des Repositories erstellen  
> In Repository wechseln  
> Leere Testdatei erstellen  
> Testdatei zum Commit hinzufügen  
> Commit erstellen mit Beschreibung  
> Repository (remote) aktualisieren
```

Python Programmiersprache

[https://projects.raspberrypi.org/en/projects?software\[\]](https://projects.raspberrypi.org/en/projects?software[])=python
<https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>

```
from time import sleep          # Importiere Zeitmodul
from picamera import PiCamera    # Importiere Kameramodul des Raspberrys

camera = PiCamera()              # Instantiiere Kameralaufzeit
camera.resolution = (1024, 768)   # Konfiguriere Auflösung
camera.start_preview()           # Starte Kamera
# Camera warm-up time
sleep(2)                         # Warte, bis Kamera gestartet ist
camera.capture('foo.jpg')         # Speichere Standbild von Kamera zu Datei foo.jpg
```

pip/venv

Ohne eigene Projektumgebung (also für das ganze System gültig)

`pip install <paketname>` > Installiere externe Bibliothek/Paket

In eigener venv Projektumgebung

`python3 -m venv klassifikation` > Erstelle pip/venv Bibliotheksumgebung

`cd klassifikation` > Wechsle in den Ordner der Umgebung

`source bin/activate` > Aktiviere Umgebung

`pip install <paketname>` > Installiere externe Bibliothek/Paket für “klassifikation”

<https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/>

Google Colab

Erlaubt es, Code remote auf Hardware von Google auszuführen und auch hier Modelle zu trainieren. Es stehen GPUs zur Verfügung.

Für umme.

Grundlage sind Jupyter Notebooks, eine Pythonumgebung fürs Web

<https://jupyter.org/try-jupyter/retr o/notebooks/?path=notebooks/Intro.ipynb>

(google: "try jupyter notebook")

The screenshot shows the Google Colab interface running on Google Drive. The main window displays a Jupyter Notebook cell with Python code for importing libraries and initializing a torch model. The Google Drive sidebar is open, showing options like 'New folder', 'File upload', and 'Folder upload'. A dropdown menu is open, listing Google Docs, Sheets, Slides, Forms, and more. The sidebar also shows storage usage (67.85 GB of 100 GB used) and a 'Buy storage' button. A 'Low Power' status indicator is visible in the top right corner of the sidebar.

```
# Check the GPU type
!nvidia-smi

# Imports
from contextlib import contextmanager
from copy import deepcopy
import math

from IPython import display
from matplotlib import pyplot as plt
import torch
from torch import optim, nn
from torch.nn import functional as F
from torch.utils import data
from torchvision import datasets, transforms, utils
from torchvision.transforms import functional as TF
from tqdm.notebook import tqdm, trange

# Utilities
```

Raspberry PI

Code für diesen Kurs: git@github.com:alexander-hagg/lpl-examples.git

Dokumentation Raspberry Betriebssystem:

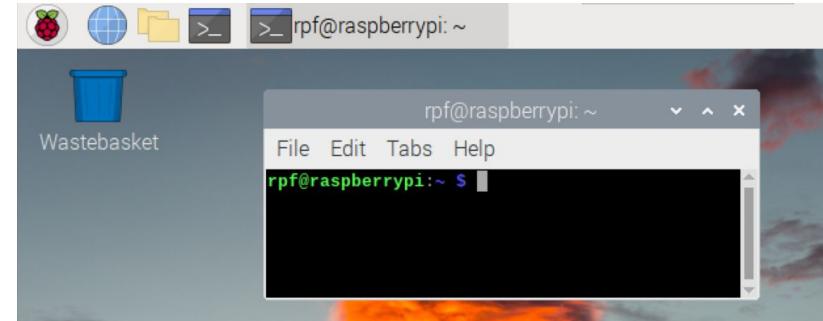
<https://www.raspberrypi.com/documentation/computers/os.html>

Aufgabe Montagnachmittag

Vorbereitung:

- Bilde Projektgruppen aus 2 oder 3 Personen
- gebe mir Bescheid, wer zusammenarbeitet

- Schließe das Raspberry an und fahre es hoch
- Öffnet Terminal
- Meldet euch bei Github an und öffnet ein (privates) Repository für diesen Nachmittag
 - Hier könnt ihr den Code sichern und teilen
 - Für das Projekt öffnet ihr später ein weiteres (privates) Repository
 - Wenn ihr wollt, könnt ihr die Repositories aber auf “public” stellen, wenn ihr den Code veröffentlichen wollt.
- Konfiguriere eine venv Umgebung, wo ihr später die notwendige Bibliotheken mit “conda install” installieren könnt.



Aufgabe Montagnachmittag

Aufgabe 1:

- Passe das Pythonprogramm U01_Kameradaten an, damit 10 aufeinanderfolgende Frames von der Kamera lokal abspeichert werden
- Committe den Code mit Git (git gui) und lade hoch zum Repository (git push)

Aufgabe 2:

- Führe U02_Segmentierung auf einem Bild aus und untersucht das Ergebnis
- Passe das Pythonprogramm U02_Segmentierung an, damit es Bilder direkt von der Kamera holt und diese segmentiert. Untersuche, wie viele Frames per second möglich sind.

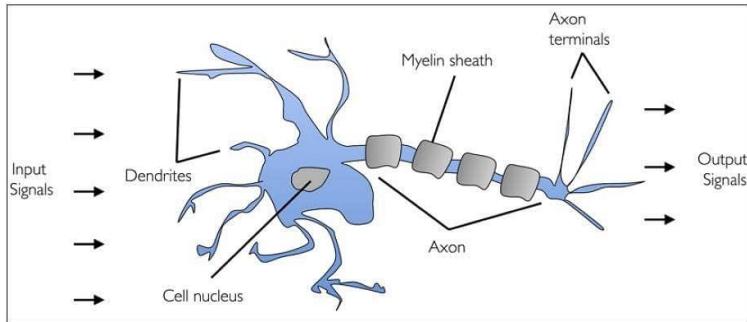
Projekt:

- Entwickle in der Gruppe erste Ideen, was ihr machen wollt im Projekt.
- Um 16 Uhr sprechen wir die erste Ideen durch.

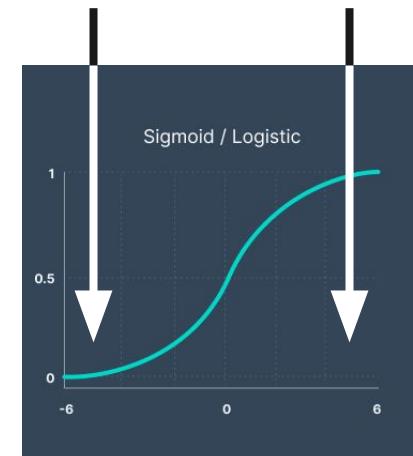
Neural Networks und Training

Das Perceptron

Biologie



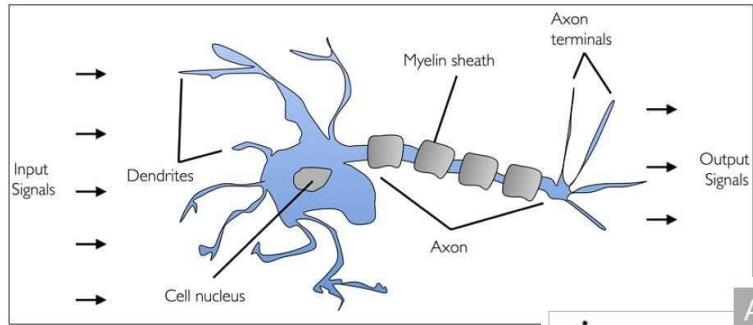
Keine Reaktion



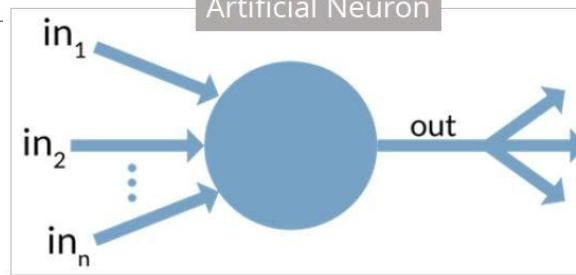
Erhöhte
Aktivität,
Warnungssignal,
Adrenalin-
ausschüttung

Das Perceptron

Biologie

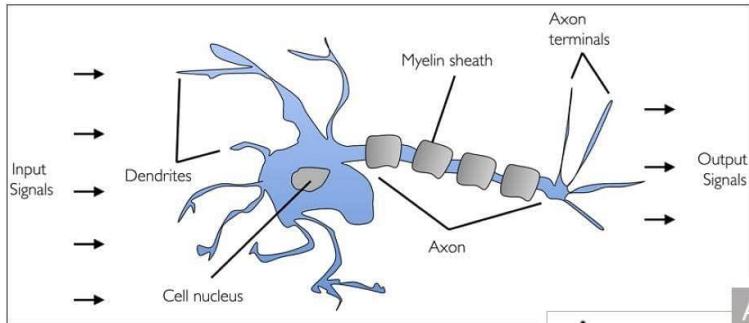


Modell

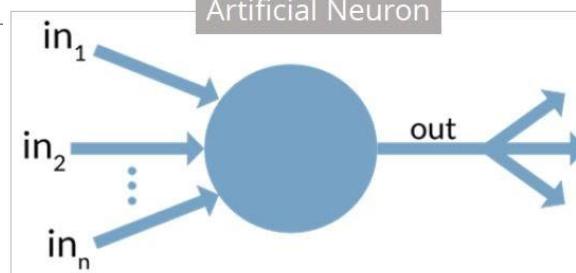


Das Perceptron

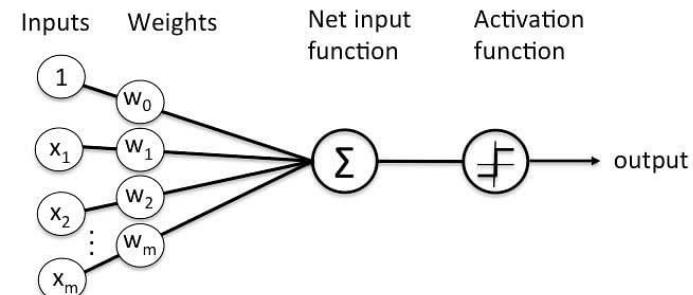
Biologie



Modell



Umsetzung



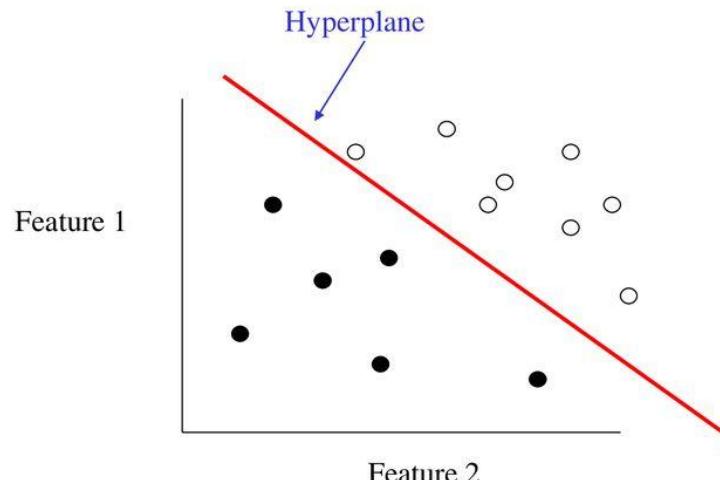
Das Perceptron

Inputs 1 (constant), x_1, x_2

Gewichte w_0, w_1, w_2

Aktivierungsfunktion: Step (0 oder 1)

-> Ein Perceptron kann Daten nur dann unterscheiden, wenn sie linear separierbar sind.



In 2D:

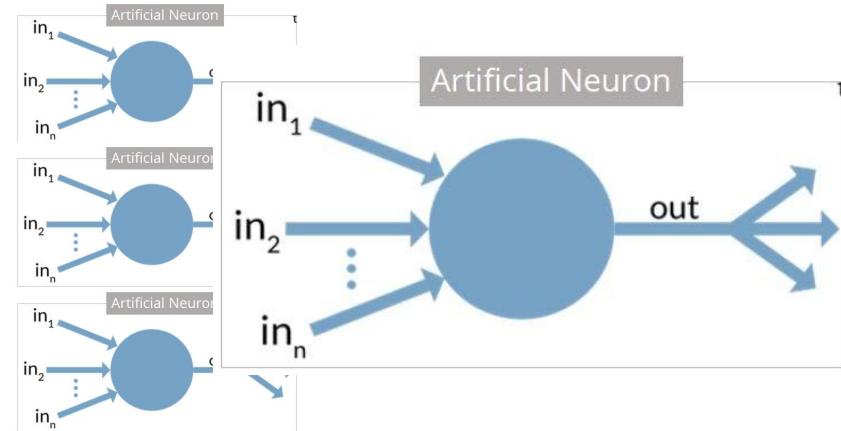
$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$

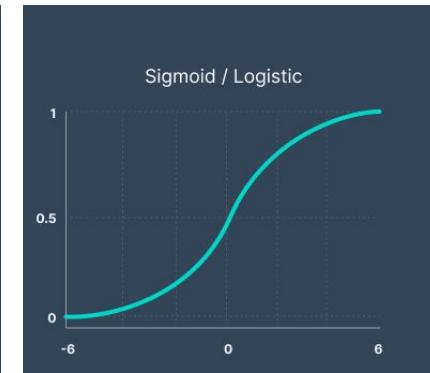
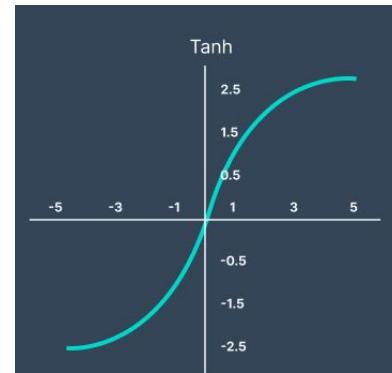
Neuronale Netzwerke: Basics

Wenn die Daten nicht linear separierbar sind, was machen wir dann?

-> mehrere Perceptrons vereinen in ein
Multilayer Perceptron



-> Aktivierungsfunktion austauschen



Neuronale Netzwerke: Basics

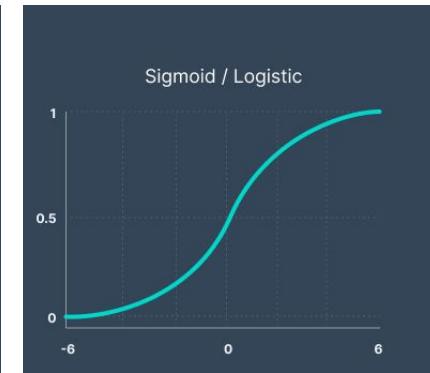
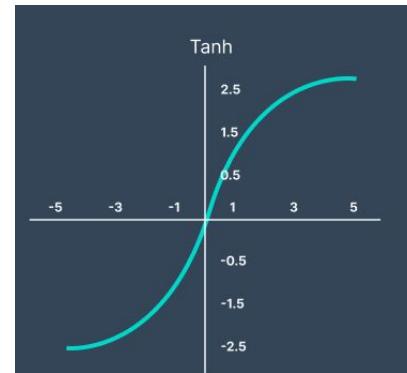
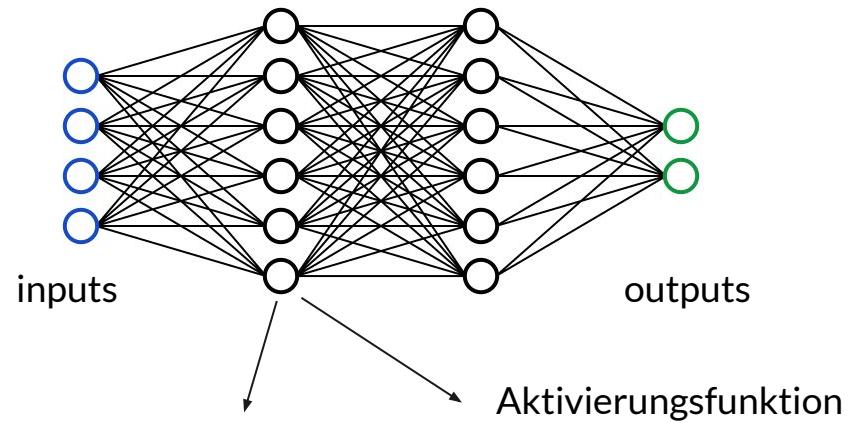
Neuronales Netz (neural network)

= ein mathematisches Modell des biologischen Gehirns

= Ein Graph, der eine Anzahl von Eingaben (inputs) mit Ausgaben (outputs) verbindet.

= verkörpert eine komplexe mathematische Funktion

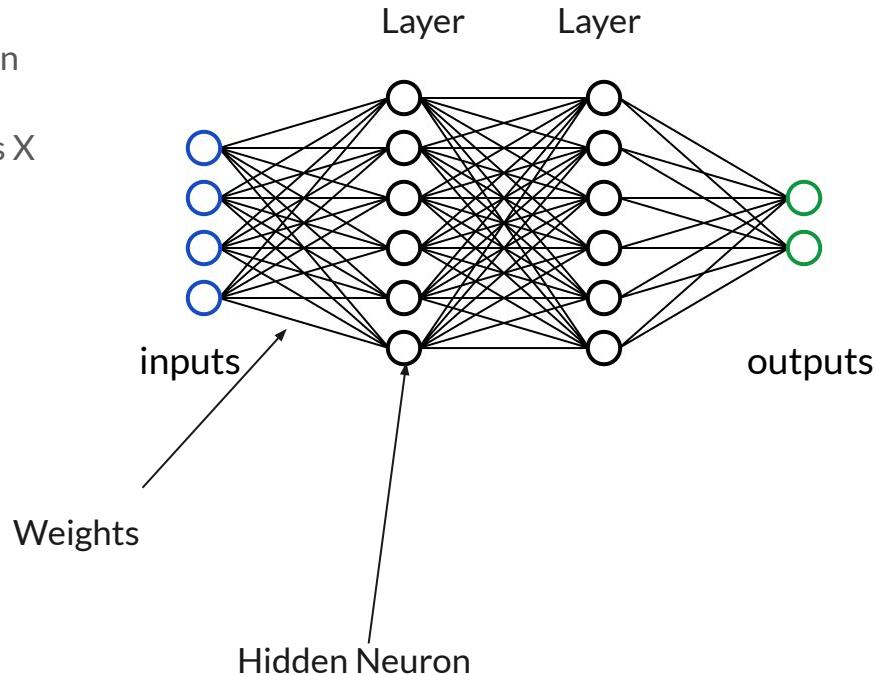
= ein Modell für eine uns unbekannte Funktion, die also nicht vordefiniert werden kann, sondern aus Beispieldaten (Trainingsdaten) abgeleitet wird



Neuronale Netzwerke: Basics

Training: Bestimmung der Gewichte der Kanten

Das Netzwerk wird so trainiert, dass bei Inputs X die Outputs Y (Labels) rauskommen



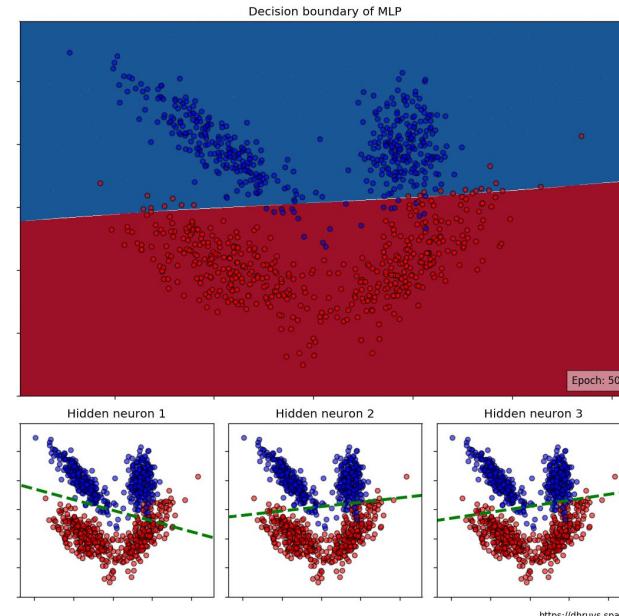
Training am Beispiel neuronaler Netze

Idee.

1. starte mit **willkürlichen** Gewichten
2. Leite die Trainingsinputs durch das Netz
3. Berechne was am Ende rauskommt
4. Berechne einen **Fehler** auf Basis der Trainingsoutputs
5. Nutze den Fehler um die Gewichte zu **justieren**
6. Wenn der Fehler **klein genug** ist, bist du fertig
7. Sonst: fange wieder bei 2 an

Wie justieren wir die Gewichte?

-> Backpropagation

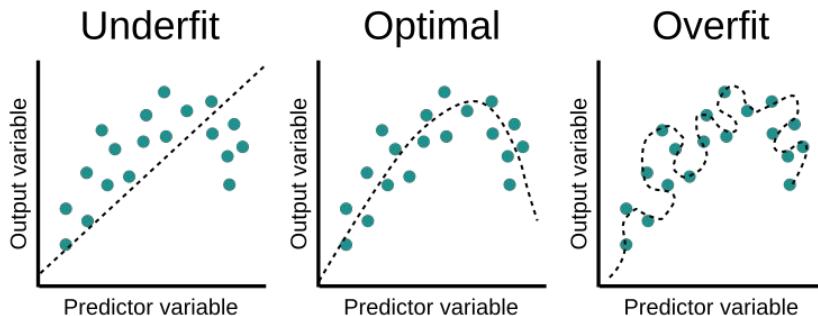


Trainingsverfahren

1. Datenaufbereitung
 - > Split der Daten in Training/Validierung/Test
 - > Normalisierung der Daten
2. Modellselektion und -Konfiguration
3. Training auf Basis der Trainingsdaten
4. Abbruch Training auf Basis Validierung
5. Test des Modells auf Testdaten
 - > Backpropagationmethode
 - > Validierungsdaten *nur* hierfür verwenden
 - > Genauigkeit Modell *nur* auf Basis Testdaten

Datenaufbereitung: Train/Validation/Test Split

Overfittingproblem



Trainingsdaten werden für das Backpropagationtraining verwendet

Fehler der Vorhersage auf Validierungsdaten wird dafür verwendet, das Training abzubrechen (stop condition)

Testdaten werden verwendet, um Modelle miteinander zu vergleichen

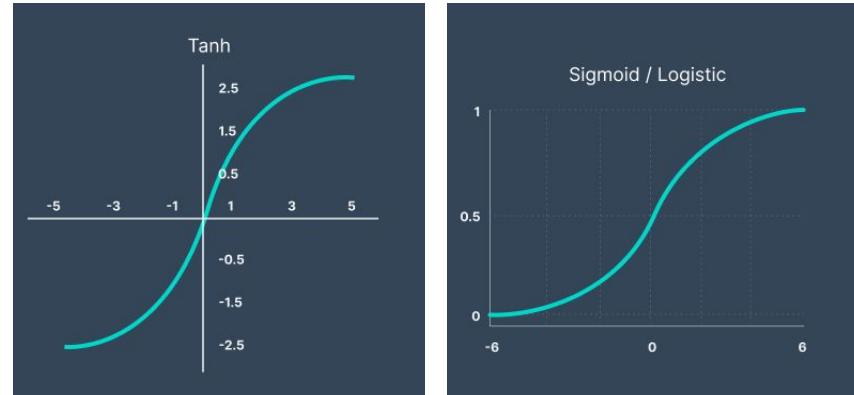
Split: zum Beispiel 80%/10%/10% oder 70%/15%/15%

Datenaufbereitung: Normalisierung

Wertebereichproblem

Beispiel Pixelkategorisierung: hell oder dunkel?

- Schwarzweißbild: 1 Inputkanal
- 0-255, 255 ist hell. 0 ist dunkel
- Braucht Normalisierung, in Abhängigkeit von verwendeter Aktivierungsfunktion!
 - Meistens gemittelt auf 0
 - Varianz der Daten auf 1



Datenaufbereitung: Feature Engineering

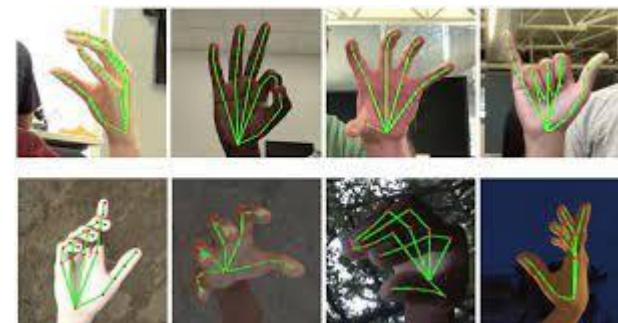
Dimensionalitätsproblem

Größter Unterschied "klassisches" Machine Learning und Deep Learning.

Hochdimensionale Daten manuell auf wenigen aussagekräftigen Features herunterbrechen.

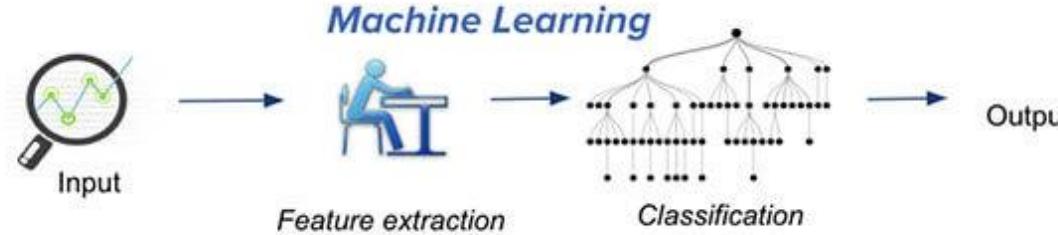
Beispiel Gesture Recognition:

Statt Videodaten direkt zu modellieren, zuerst segmentieren, die größte sich bewegende Region extrahieren und auf ein einfaches Skelettmodell zu mappen

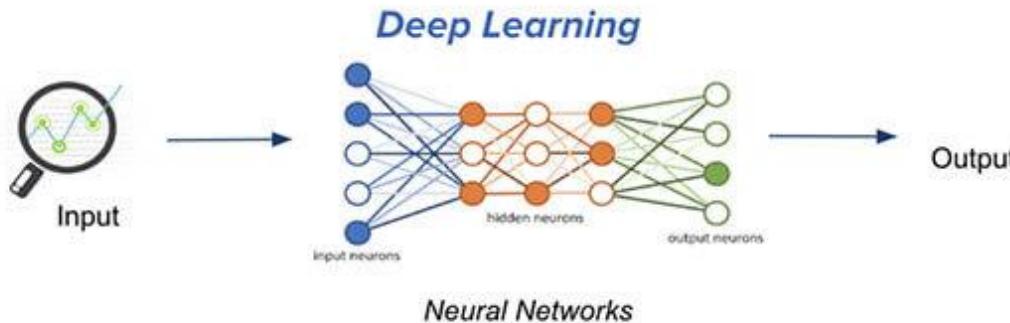


Datenaufbereitung: Feature Engineering

Aber Deep Learning braucht große Datenmengen!



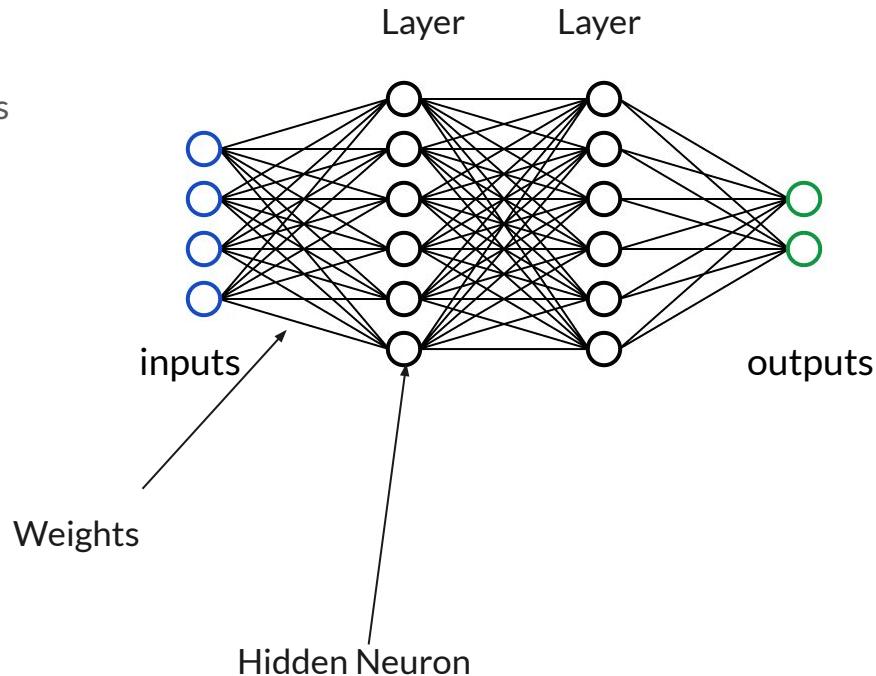
Traditional machine learning uses hand-crafted features, which is tedious and costly to develop.



Deep learning learns hierarchical representation from the data itself, and scales with more data.

Modellselektion und Konfiguration

- Welches ML-Modell wird verwendet?
- Parametrisierung (z.B. Anzahl der Layers und Hidden Neurons, verwendete Aktivierungsfunktion)



Beispiel: Vergleiche 2 Modellierungsmethoden

1. Für beide Modelle A und B:
 2. Wiederhole Training 10 mal
(Backpropagation auf den Trainingsfehler)
 3. wähle jeweils Modell mit geringstem Fehler
über den Validierungssatz
- Vergleichsmetriken
- MAE: mean absolute error
 - MSE: mean square error

Berechne Fehler über Testsatz für Modell A und B

Aufgaben Dienstagnachmittag

Aufgabe 3: Klassifikation mit vortrainiertem
Modell

<https://carpentries-incubator.github.io/machine-learning-novice-sklearn/06-neural-networks/index.html>

Training nicht auf dem Raspberry