

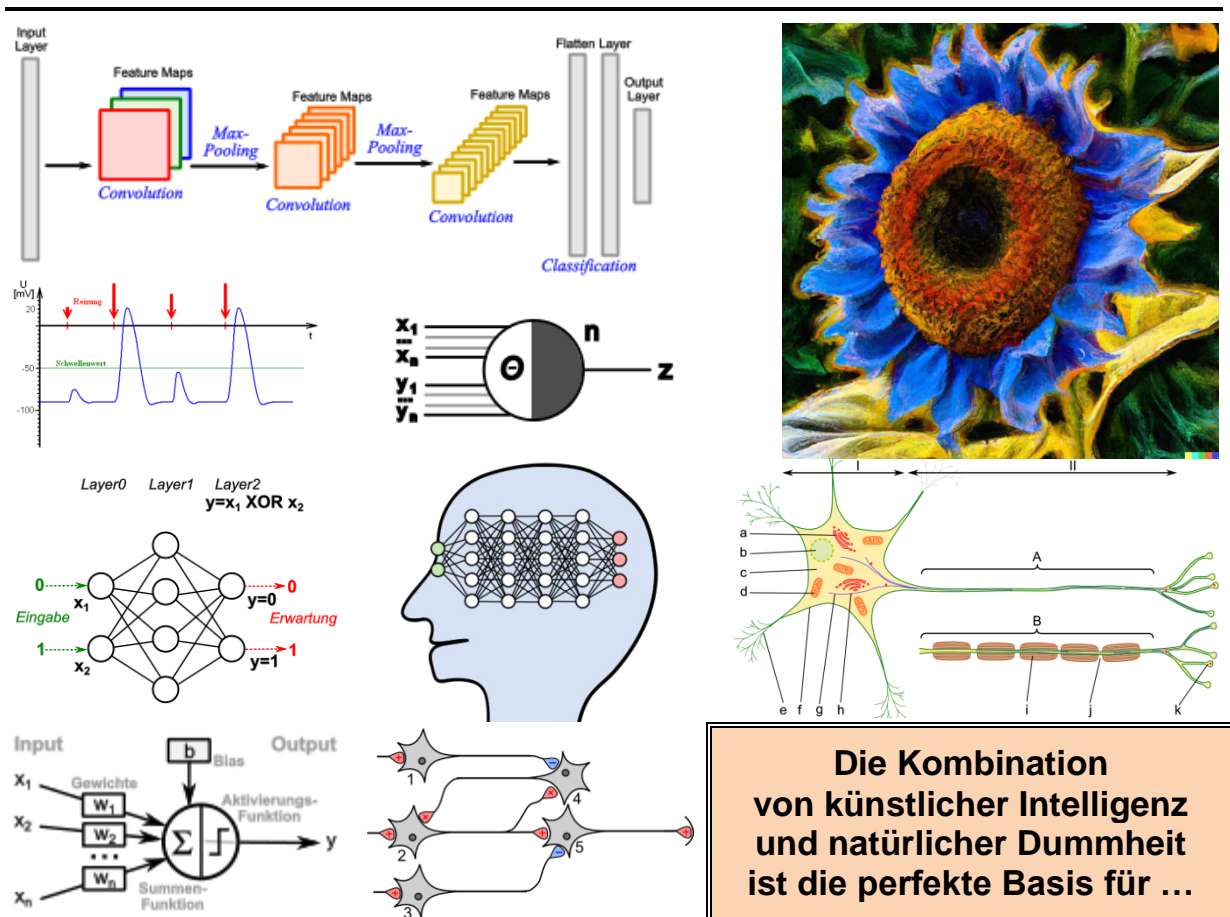
Informatik

- Projekt:

Deep Learning -

*Künstliche Neuronale Netzwerke,
Maschinelles Lernen, Künstliche Intelligenz,
Computer Vision, ...*

Autor: L. Drews

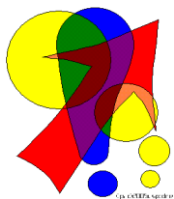


Q: "blaue Sonnenblume" erstellt mit openAI.com (DALL-E2) (22.01.2023)

teilredigierte Entwicklungs-Version 0.2o (2023)

Legende:

mit diesem Symbol werden zusätzliche Hinweise, Tips und weiterführende Ideen gekennzeichnet



Nutzungsbestimmungen / Bemerkungen zur Verwendung durch Dritte:

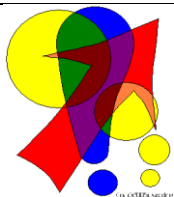
- (1) Dieses Skript (Werk) ist zur freien Nutzung in der angebotenen Form durch den Anbieter (lern-soft-projekt) bereitgestellt. Es kann unter Angabe der Quelle und / oder des Verfassers gedruckt, vervielfältigt oder in elektronischer Form veröffentlicht werden.
- (2) Das Weglassen von Abschnitten oder Teilen (z.B. Aufgaben und Lösungen) in Teildrucken ist möglich und sinnvoll (Konzentration auf die eigenen Unterrichtsziele, -inhalte und -methoden). Bei angemessen großen Auszügen gehören das vollständige Inhaltsverzeichnis und die Angabe einer Bezugsquelle für das Originalwerk zum Pflichtteil.
- (3) Ein Verkauf in jedweder Form ist ausgeschlossen. Der Aufwand für Kopierleistungen, Datenträger oder den (einfachen) Download usw. ist davon unberührt.
- (4) Änderungswünsche werden gerne entgegen genommen. Ergänzungen, Arbeitsblätter, Aufgaben und Lösungen mit eigener Autorenschaft sind möglich und werden bei konzeptioneller Passung eingearbeitet. Die Teile sind entsprechend der Autorenschaft zu kennzeichnen. Jedes Teil behält die Urheberrechte seiner Autorenschaft bei.
- (5) Zusammenstellungen, die von diesem Skript - über Zitate hinausgehende - Bestandteile enthalten, müssen verpflichtend wieder gleichwertigen Nutzungsbestimmungen unterliegen.
- (6) Diese Nutzungsbestimmungen gehören zu diesem Werk.
- (7) Der Autor behält sich das Recht vor, diese Bestimmungen zu ändern.
- (8) Andere Urheberrechte bleiben von diesen Bestimmungen unberührt.

Rechte Anderer:

Viele der verwendeten Bilder unterliegen verschiedensten freien Lizenzen. Nach meinen Recherchen sollten alle genutzten Bilder zu einer der nachfolgenden freien Lizenzen gehören. Unabhängig von den Vorgaben der einzelnen Lizenzen sind zu jedem extern entstandenen Objekt die Quelle, und wenn bekannt, der Autor / Rechteinhaber angegeben.

public domain (pd)	Zum Gemeingut erklärte Graphiken oder Fotos (u.a.). Viele der verwendeten Bilder entstammen Webseiten / Quellen US-amerikanischer Einrichtungen, die im Regierungsauftrag mit öffentlichen Mitteln finanziert wurden und darüber rechtlich (USA) zum Gemeingut wurden. Andere kreative Leistungen wurden ohne Einschränkungen von den Urhebern freigegeben.
gnu free document licence (GFDL; gnu fdl)	
creativecommons (cc) 	od. neu ... Namensnennung ... nichtkommerziell ... in der gleichen Form ... unter gleichen Bedingungen

Die meisten verwendeten Lizenzen schließen eine kommerzielle (Weiter-)Nutzung aus!



Bemerkungen zur Rechtschreibung:

Dieses Skript folgt nicht zwangsläufig der neuen **ODER** alten deutschen Rechtschreibung. Vielmehr wird vom Recht auf künstlerische Freiheit, der Freiheit der Sprache und von der Autokorrektur des Textverarbeitungsprogramms microsoft® WORD® Gebrauch gemacht.

Für Hinweise auf echte Fehler ist der Autor immer dankbar.

Inhaltsverzeichnis:

	Seite
0. Einleitung / Vorwort.....	11
1. Einführung / Überblick.....	14
1.x. Historie / Motivation.....	19
Definition(en): maschinelles Lernen (Machine Learning)	29
Definition(en): künstliche neuronale Netze (KNN)	29
Definition(en): Deep Learning (tiefes Lernen).....	29
Definition(en): Computer Vision	30
Vergleich: Informations-Verarbeitung bei Mensch und Computer	30
Vergleich: "VON NEUMANN"-Rechner und Künstliche Neuronale Netzwerke.....	31
Vergleich: klassische Computer-Programme und Künstliche neuronale Netze.....	32
Definition(en): Robotik	36
Definition(en): Roboter	36
Daten-getriebene Projekte.....	39
Definition(en): Roh-Daten	41
Definition(en): Eingabe-Daten.....	41
Definition(en): Ziel-Daten	41
Definition(en): Prognose-Daten	41
1.2. Erfolge / Meilensteine von KI.....	43
2. Grundlagen	45
2.0. Logik, Mathematik,	45
2.0.1. Prädikaten-Logik	45
2.0.2. Fuzzy-Logik.....	45
2.0.3. Stochastik – Wahrscheinlichkeits-Rechnung	45
Wahrscheinlichkeit	45
Exkurs: BAYES für Höhlen-Menschen	48
Exkurs: BAYES bei medizinischen Test's.....	51
Exkurs: 4-Felder-Tafel	51
der naive BAYES-Klassifikator	53
Exkurs: Baum-Diagramme.....	54
2.0.4. Korrelation und Regression	56
Korrelation.....	56
Definition(en): Korrelation	56
Regression.....	57
Definition(en): Regression	57
lineare Regression	58
Regression einer reziproken Funktion	60
Regression einer potentiellen Funktion.....	61
Regression einer exponentiellen Funktion.....	63
Regression einer logarithmischen Funktion.....	64
logistische Regression	65
polynomiale Regression	65
symbolische Regression	66
2.0.5. Modelle, Modell-Bildung, Modellierung.....	67
Big Data	67
2.1. Kern-Problem: Intelligenz.....	69
2.1.x. Was ist Intelligenz?.....	69
Definition(en): Intelligenz	70
Definition(en): Intelligenz	72
Definition(en): Koffer-Wort	74
Intelligenz-Test's	76
der TURING-Test.....	78
2.1.x. Unterscheidung von Intelligenzen.....	83
allgemeine und spezifische Intelligenz.....	83
starke und schwache Intelligenz.....	83

Definition(en): künstliche Intelligenz	83
2.1.x. Welche Charakteristika hat natürliche Intelligenz?	84
2.1.x. Was ist künstliche Intelligenz?	85
Definition(en): Künstliche Intelligenz (KI)	85
Spekulationen / Gedanken zu einer starken künstlichen Intelligenz.....	89
Bedingungen für die Entstehung einer "starken KI"	89
vermutete Entwicklungs-Phasen einer "starken KI"	90
Möglichkeiten des Entstehens	92
2.1.x.y. Das Fluss-Überquerungs-Problem.....	95
2.1.x.y. Bauern-Schach	98
Lösung des Spiel über KI-Ansatz	99
Spiel-Situationen und mögliche Spiel-Züge für Spieler 1 (Mensch)	101
Agenten-basierter Lern-Algorithmus für Spieler 2 (Computer / Mensch).....	103
2.x. das menschliche Gehirn - ein neuronales Netzwerk	105
2.x.0. Inspiration.....	105
2.1. biologische Grundlagen	105
2.2. natürliches Lernen	105
2.3. Künstliche Neuronen - Grund-Bausteine neuronaler Netzwerke.....	106
Definition(en): Künstliches Neuron	106
2.3.x. MCCULLOCH-PITTS-Zellen.....	107
2.3.x.y. Umsetzung logischer Funktionen mit MCCULLOCH-PITTS-Zellen	108
2.3.x.y.1. Umsetzung einer (universellen) MCCULLOCH-PITTS-Zelle in Python....	109
2.3.x.y. HEBBSche Lernregel	110
2.3.x. das Perzeptron-Modell	112
künstliches Neuron.....	113
gewichtete Eingänge	113
Bias(-Wert)	113
Summierung aller Eingänge	114
Perzeptron-Lern-Regel.....	115
2.3.x.y. mehrschichtiges Perzeptron.....	117
2.3.x.y. Aktivierungs-Funktion(en)	118
2.3.x.y. Universalitäts-Theorem	119
sigmoide Aktivierungs-Funktion.....	119
tanh Aktivierungs-Funktion	120
ReLU Aktivierungs-Funktion	120
Softmax-Aktivierungs-Funktion.....	121
Backpropagation	122
Umsetzung eines Perzeptron's in Python	122
2.5. Künstliche Neuronale Netzwerke	124
2.5.x. kleine punktuelle Geschichte / Historie der Künstlichen Neuronalen Netzwerke,	126
2.5.x. Aufbau künstlicher neuronaler Netze	127
Backpropagation über mehrere Perzeptonen / in einem Netzwerk	128
2.5.x.y. Schichten.....	129
Input Layer	129
Output Layer	130
Hidden Layer.....	130
Convolutional Layer.....	130
Definition(en): Feature Map	133
Pooling Layer	134
Flatten Layer	135
Dense Layer.....	136
komplexe Layer.....	136
Definition(en): Neuronales Netz.....	137
2.5.x. Arbeitsweise / Funktion von KNN's	139
2.5.x. Rückgekoppelte Neuronale Netzwerke.....	140
2.6. maschinelles Lernen.....	144
Ansätze zum Speichern von Wissen	146

Experten-Systeme.....	149
WikiData.....	153
Definition(en): Machine Learning	155
Definition(en): Agent	156
Arten / Felder des maschinellen Lernens (Machine Learning)	157
2.6.x. Supervised Learning (Kurz-Vorstellung)	158
2.6.x. Unsupervised Learning (Kurz-Vorstellung)	159
2.6.x. Semisupervised Learning (Kurz-Vorstellung)	161
Definition(en): Heuristik.....	161
2.6.x. Reinforcement Learning (Kurz-Vorstellung)	162
2.6.x. Lern-Vorgänge und -Verfahren.....	162
2.6.x. Bewertung des Lern-Erfolgs	163
Fehler-Funktionen / Loss Function's.....	164
Definition(en): Epoche	165
alternative Fehler-Funktion: mittlerer absoluter Fehler (Mean Absolute Error)	166
weitere Fehler-Funktionen.....	167
2.6.x. Lern-Algorithmen.....	167
Backpropagation	167
Optimierung durch Zufälliges / Stochastisches Gradienten-Abstiegs-Verfahren (Stochastic Gradient Descent)	167
2.6.x. Daten-Bereitstellung	168
Zeitpunkt und Lage der Daten-Verarbeitung.....	172
Daten-Quellen / Daten-Beschaffung.....	172
Daten-Entschlüsselung	174
Daten-Labeling.....	176
Definition(en): Weak Supervision.....	177
MNIST-Datensatz.....	177
Aufbau eines passenden KNN für den MNIST-Datensatz und das Ziffern- Erkennungs-Problem	178
IMAGENET-Datensatz	178
Anomalien in Daten.....	179
Informations-Extraktion aus Texten	179
2.6.x. Evaluations-Metriken	180
2.6.x.y. Evaluierung von supervised bzw. semisupervised Learning.....	180
Definition(en): Generalisierung	183
Definition(en): Overfitting	183
Definition(en): Underfitting	183
2.6.x.y. Evaluierung von unsupervised Learning	184
2.6.x. Supervised Learning.....	185
2.6.x.y. No-Free-Lunch-Theorem	187
2.6.x.y. k-nächste Nachbar-Verfahren	188
2.6.x.y. Entscheidungs-Bäume.....	189
Vorhersagegüte (Wetter-Vorhersage).....	192
das Pakete-Problem mit einem Entscheidungs-Baum lösen.....	193
2.6.x.y. Regression.....	194
2.6.x.y. Künstliche Neuronale Netze.....	195
2.6.x. Ensemble Learning.....	198
2.6.x. Optimierung eines KNN.....	199
Stochastic Gradient Descent.....	199
RMSProp	200
Adam	200
Empfehlungen / Orientierungen für die Wahl eines Optimizer's	200
2.6.x. Hyper-Parameter.....	201
Definition(en): Hyper-Parameter	201
Lern-Rate (Learning Rate).....	203
Anzahl der Epochen	203
Definition(en): Overfitting	203
Initial-Werte der Gewichte und des Bias.....	203

Batch-Größe (Batch Size)	204
Anzahl der (Hidden-) Layer (Schichten).....	204
Wie findet man nun die optimalen Werte für die Hyper-Parameter?	205
2.6.x. Probleme in KNN's	208
Definition(en): Bias.....	209
Problem: Daten-Menge	209
Problem: Rechen-Operationen.....	209
Problem: Veränderung der Daten-Struktur	210
Problem: wenige Trainings-Daten	210
2.6.x. besondere Methoden / Techniken zum Training von KNN's.....	212
2.6.x.y. Dropout.....	212
Dropout in Keras-Modellen.....	213
2.6.x.y. Batch Normalization	214
Batch Normalization in Keras-Modellen.....	215
2.6.x.y. Data Augmentation	216
Data Augmentation in Keras-Modellen	217
2.6.x.y. Kombination verschiedener Regularisierungs-Techniken.....	219
Kombination verschiedener Regularisierungen in Keras-Modellen	219
2.7. Explainability (Erklärbarkeit).....	221
2.8. Transfer Learning.....	223
2.7.1. Transfer Learning mit TensorFlow	224
Beispiel: Xception.....	225
2.7.2. Transfer Learning mit Keras-Modellen.....	227
2.x. ausgewählte Wiederholungen zu Python	229
2.x. semantische Netze in Python.....	230
2.x. einfache Implementierungen von Perceptron und Künstlichen Neuronalen	
 Netzen in Python	233
2.x.y. Programmierung eines Perceptron's.....	233
weiteres Python-Beispiel für eine Perzeptron-Umsetzung	244
2.x.y.z. Programmierung eines Perceptron's mit 3 Eingängen.....	245
2.x. Entscheidungs-Bäume in Python	247
2.x. Arbeiten mit ausgewählten Bibliotheken in Python	259
2.x.y. Kurzvorstellung der Bibliothek NumPy	259
Exkurs: Installation von NumPy und verwandten Bibliotheken.....	260
2.x.y.0. Wiederholung Matrizen und Matrizen-Operationen	260
2.x.y.1. Definieren und Ausgeben von Matrizen in Python - NumPy	261
2.x.y.2. Berechnungen / Operationen mit Matrizen in Python - NumPy.....	263
2.x.y. Kurzvorstellung der Bibliothek Matplotlib	267
2.x. entdeckenes / spielendes Lernen mit Python	271
2.x.y. Tic-Tac-Toe mit Python.....	271
2.x. Wiederholung / Basic's zu JAVA.....	279
Kommentare.....	279
Grund-Gerüst einer aufrufbaren / startbaren JAVA-Datei	280
Grund-Gerüst weiterer Klassen	280
Variablen (Deklarationen, Zuweisungen).....	280
Ausgaben (auf der Konsole).....	281
Eingaben (von der Konsole)	281
Methoden / Funktionen.....	281
2.x. Perceptron in JAVA	283
2.x. Künstliches neuronales Netz in JAVA.....	284
2.x. PROLOG – eine andere Form der Programmierung für die Wissens-	
 Verarbeitung	285
2.x. Modell-Verwaltung (Model Governance)	291
3. weitere Grundlagen zu KI-Problem-Feldern.....	292
3.x. Verarbeitung von Texten	292
Probleme mit der Semantik von Texten.....	293
3.x.y. Word Embeddings	293

Definition(en): Kookkurrenz	295
Bag-Of-Words (BOW)	297
1-Hot-Encoded-Vektor Beispiel	300
Unsupervised Word Vectors Word2Vec, GloVe.....	301
Skip-Gram-Modell	302
Supervised Word Vectors – Tensorflow Embedding.....	303
neue Ansätze	303
Inhalts-basierte Text-Indizierung (z.B. für / in Suchmaschinen)	305
3.x.y.z. praktische Umsetzungen / Versuche	306
manuelles / mechanisches Word Embedding	306
statistische Analyse von Word Embeddings	309
maschinelles Erlernen von Word Embeddings	309
eine word2vec-Umsetzung (für deutsche Worte)	309
3.x.y. Stimmungs-Analysen (Sentiment Analyse)	313
Definition(en): Stimmungs-Analyse (Sentiment Analyse).....	313
3.x. Text- und Sprach-Erzeugung	314
3.x.y. Text-Erzeugung	314
3.x.y. weitere Text-Analyse-Möglichkeiten	315
3.x.y. Large-scale Language Models.....	315
3.x.y. Bewertung des Text-Layout's.....	316
algorithmischer / klassischer Ansatz.....	316
KI-/ML-Ansatz	317
4. praktisches Experimentieren mittels TensorflowPlayground	318
online Eperimentieren auf Tensorflow Playground	318
5. praktisches Experimentieren mittels Python und Co	322
5.0. Erstellen einer Arbeits-Umgebung.....	322
5.0.1. Nutzung einer online-Arbeits-Umgebung.....	322
5.0.2. lokale Installation einer Arbeits-Umgebung / Installation von Anaconda	323
5.0.2.1. Installation von Anaconda unter Windows	323
5.0.2.2. Installation von Anaconda unter Linux	323
5.0.2.3. Installation von Anaconda unter iOS / auf einem Mac.....	323
5.0.3. Arbeiten mit der Anaconda-Umgebung.....	323
5.0.3.1. Erstellung eines Evironment's.....	323
5.0.3.2. Starten / Aktivieren und Beenden eines Evironment's.....	324
5.0.3.3. Nutzung von jupyter-Notebook's	324
5.0.4. Bibliothek: Pandas.....	324
5.0.5. Bibliothek: matplotlib	325
5.0.6. Bibliothek: numpy	325
5.0.7. Bibliothek: scikit-learn.....	325
5.0.8. Bibliothek: tensorflow	325
5.1. Erstellen von KNN's (z.B. in Anaconda oder der Python-IDE)	326
5.1.x. Experimente mit der Bibliothek ???	326
5.1.x. Laden des / Bearbeiten und Experimentieren mit dem MNIST-Datensatzes.....	326
Keras 330	
Kern-Konzepte	330
Beispiel-KNN für MNIST-Datensatz.....	332
Experimentieren mit den Aktivierungs-Funktionen.....	333
Verwendung von Convolution in einem Keras-basierten KNN	338
Verwendung von Pooling-Layer in einem Keras-basierten KNN	338
Experimentieren mit komplexen Layer-Strukturen in einem Keras-basierten KNN .	338
Experimentieren mit Fehler-Funktionen.....	344
Experimentieren / Setzen bei Klassen-Ungleichgewichten	345
Experimentieren mit dem Opitimizer.....	346
Experimentieren / Setzen von Hyper-Parametern (Lern-Rate)	347
Experimentieren / Setzen von Hyper-Parametern (Gewichte- + Bias- Initialisierung)	349

Experimentieren / Setzen von Hyper-Parametern (Batch-Größe).....	349
Experimentieren / Setzen mehrerer Hyper-Parametern ("Grid Search"-Verfahren).....	349
Experimentieren / Setzen mehrerer Hyper-Parametern ("Randomized Search"- Verfahren).....	350
5.1.x. Daten-Beschaffung per Web-Crawling.....	350
5.2 Erstellen von KNN's in google-Colab	352
Kurze Einführung in Google Colab.....	352
Zusammenfassung: Minimal- / Grund-Modell für Keras.....	353
5.x. Entwicklung einer "eigenen" Python-Umsetzungen von KNN's.....	354
5.x.1. Entwicklung des Netzes am XOR-Beispiel-Problem	354
Vorbereitung / Vorüberlegungen.....	354
Design-Entscheidungen / Konzepte	355
Umsetzung / Implementierung.....	357
Verbesserungen / Erweiterungen	368
5.x.2. Anwendung des Netzes auf MNIST-Datensatz.....	370
5.x. eine weitere Implementierung eines KNN in Python	371
5.x. Erstellen eines TensorFlow-basierten KNN	375
5.x. KNN's mit Torch	376
5.x. Projekte mit jupyter-Notebook's	377
5.x.0. Einführung in jupyter-Notebook	377
5.x.0.1. Installation / Vorbereitung	377
5.x.0.2. Bedienung eines jupyter-Notebook's.....	377
5.x. Projekte mit Google Colab's.....	377
5.x.0. Einführung in Google Colab's	378
5.x.0.1. Bedienung eines Google Colab's.....	378
5.x.0.1. Einstellungen eines Google Colab's.....	378
5.x. Reinforcement Learning mit.....	378
Beispiel: Super Mario Bros.-Modell	378
6. praktisches Experimentieren mit Scratch und / oder Snap!.....	382
6.1. einfache Anwendungen mit Scratch.....	382
6.1.x. Text-Analyse	382
6.1.x. Bilder-Erkennung.....	382
6.2. künstliche Intelligenz mit Snap!.....	382
6.3. verstärkendes Lernen	383
Beispiel: Bananen-Jagd / Affen-Arcade.....	383
7. Experimentieren mit dem google-AIY-Set's	388
8. KNN spezial -- Was sonst noch so geht?!.....	395
8.x. spezielle KNN-Typen.....	395
8.x.y. GAN's	395
8.x.y.z. WASSERSTEIN GAN	395
Nutzung der deeplearning2020 Python library	396
8.x. Haus-Preise in Kalifornien.....	398
8.x.y. Daten-Aufbereitung.....	398
8.x.y. Daten-Analysen und -Visualisierungen	400
8.x.y. Entwickeln eines (ersten) KI-Modell's	402
8.x.y. Ausprobieren / Testen eines (ersten) KI-Modell's.....	403
8.x. Projekt Film-Vorschlag-System	405
8.x.y. Daten-Aufbereitung.....	408
Überblick über die Daten / Daten-Analyse.....	408
Daten-Aufbereitung / -Vorbereitung	410
Anonymisierung	410
Vorbereitung spezieller Medien-Daten(-Typen) / Domain-spezifische Aufbereitung	411
Aufteilen in Trainings- und Test-Daten (Train/Test-Splitting)	412
8.x.y. Modellierung.....	412

Ähnlichkeits-Berechnungen.....	413
Count Vectorizer	413
Tf-idf Vectorizer.....	414
Anwendung Tf-idf-Vektorisierung auf die Kurz-Beschreibungen	414
Analyse der Vorschläge / Qualitäts-Analyse	415
AutoML-Systeme.....	418
Federated Learning.....	418
Privacy-Preserving ML	419
8.x. Stimmungs-Analyse (Sentiment Analyse).....	420
Einfluss der Größe des Trainings-Datenbestand's auf die Qualität des Modell's	425
8.x. Gesten-Erkennung (Gebärden-Sprache) / Bilder-Klassifikation	426
Vor-Analyse der Daten / Übersicht	427
Umsetzung / Aufbau des CNN (Convolutinal Neural Network)	429
Optimierung des Modell's	430
Transfer Learning / Pretrained Machine Learning.....	431
Definition(en): Transfer Learning	431
Bewertung der Ergebnisse (der Lern-Prozesse).....	433
9. Künstliche Neuronale Netzwerke in der Bio-Informatik.....	434
10. gesellschaftliche und ethische Aspekte rund um die KI.....	435
Bedeutung von Daten im Machine Learning.....	437
Einschränkungen von KI	437
Bedeutung / Wichtigkeit der Daten	438
Diskriminierung in und mit Daten.....	441
Definition(en): Bias.....	441
Gründe für Bias	443
Probleme mit der Bewertungs-Funktion.....	443
Möglichkeiten, um einer Diskriminierung durch Daten entgegenzuwirken:	444
Einfluss von KI im täglichen Leben.....	444
Regulierung, Ethik und Bildung zur KI	445
Trolley-Dilemma	445
Überwachung und Gesichts-Erkennung	445
Mensch, Moral und Maschine.....	446
Deep Fakes.....	446
KI und Schule / Bildung	449
KI bei der Bewerber-Auswahl / KI im Access-Center	449
11. Generative Künstliche Intelligenz – KI produziert	450
11.1. Künstliche Intelligenz textet.....	454
Verdummt die KI sich selbst?	458
Prüfen von Texten auf KI-Urheberschaft?!	460
Vorbereitung einer Unterrichtsstunde mit und über ChatGPT	465
Erstellung eines Vortrag's über ChatGPT	466
11.2. Künstliche Intelligenz malt	469
11.3. KI arrangiert und komponiert	472
11.4. Suche mit KI	474
11.x. weitere kreative Ansätze / Anwendungen	475
11.x.y. Storytelling und Präsentation	475
11.x. gesellschaftliche Aspekte zu kreativer KI	476
Literatur und Quellen	477
Anhänge.....	479
diverse weitere Quell-Code's	479
MCCULLOCH-PITTS-Zelle mit Python (Minimal-Version)	479
Tic-tac-toe mit Minimax in PROLOG	480
Tic-tac-toe in JAVA.....	484

0. Einleitung / Vorwort

Die Entwicklung Künstlicher Intelligenz könnte entweder das Schlimmste oder das Beste sein, was den Menschen passiert ist.
Stephen HAWKING

die wesentlichen Inhalte und sozusagen das Basis-Material stammt aus dem OpenHPI-Kurs "Praktische Einführung in Deep Learning für Computer Vision", welcher vom HPI-Student Team für neuronale Netze durchgeführt wurde. Dieser Kurs lief im März/April 2020. Derzeit kann der Kurs als Selbststudium-Material auf OpenHPI (→) genutzt werden.

wesentliche Inhalte, viele Quell-Code's sind dem Open-HPI-Kurs entnommen (praktisch viele Teile als Struktur-Plagiat interpretierbar!)

Material neu formuliert, z.T. umstrukturiert, neu geordnet und vielfach von mir ergänzt worden.

Ich versuche dabei auch einige Einstiegs-Hürden, die der Kurs enthält, zu verkleinern. Ob mir das (immer) gelungen ist, muss jeder für sich bewerten. Wem das Skript nicht passt, dem sein die vielen weiteren tollen Tutorial's im Internet empfohlen.

in einer 3. Stufe habe ich einen im Herbst 2020 (Sep./Okt.) laufenden Einsteiger-Kurs zu "Künstlicher Intelligenz und maschinelles Lernen für Einsteiger" (Johannes HÖTTER, Christian WARMUTH) von Open-HPI mit eingebaut

2021 lief der OpenHPI-Kurs "Künstliche Intelligenz und Maschinelles Lernen in der Praxis" dessen Inhalte wurde ebenfalls mit eingebaut

ebenfalls aus dem Jahr 2021 (im Oktober) stammt der OpenHPI-Kurs "Künstliche Intelligenz und maschinelles Lernen in der Praxis" von Johannes HÖTTER, Christian WARMUTH

die Inhalte dieses Kurses wurden z.T. in die bisher verfolgte Gliederung eingearbeitet, als auch in extra Kapitel aufgenommen

besonders die Quell-Texte aus den Jupyter-Notebook's sind fast ausschließlich im Original übernommen

auf eine Eindeutschung habe ich hier weitgehend verzichtet, weil es sich mehr oder weniger um speziellere Themen handelt, die schon eine gewisse fachliche Grundlage und ein gewisses Verständnis für Python voraussetzt

alle Kurse und oben genannten Quellen wurden dann durch weitere Quellen, Informationen und Materialien ergänzt

Ziel ist ein Schüler-gerechtes Skript mit viel Material und Potential für einen offenen und Teilnehmer-zentrierten Projektkurs

vielleicht motiviert das [hier verlinkte Video](#), sich intensiver mit dem Thema auseinanderzusetzen

Hier noch ein paar andere frei zugängliche Quellen / Skripte / Bücher.

weitere Links:

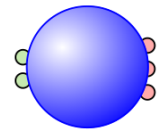
<http://neuralnetworksanddeeplearning.com/index.html> (online-Buch: NIELSEN, Michael: Neuronale Netze und Deep Learning (2019); engl.; Anmerk.: sehr umfassend, von den Grundlagen zum Speziellen, mathematisch tiefgründig)

<https://www.deeplearningbook.org/> (online-Version eines MIT-Buches; Anmerk.: sehr sehr anspruchsvoll)

spannend ist es auch, das nebenstehende Bild mit einem KNN zu testen da sind interessante Ergebnisse vorprogrammiert
genauerer dazu auch unter

Q: de.wikipedia.org ()

Ein Lern- und Lehr-Begleiter in diesem Skript ist "**Netzy**" – ein kugelförmiger Roboter, der ein Künstliches Neuronales Netz (KNN) enthält. Netzy kann sich überall hin bewegen, kann schwimmen und fliegen. Er hat – ganz typisch für informatische Systeme – (grüne) Eingabe- und (rötliche) Ausgabe-Möglichkeiten. Deren Anzahl kann sich je nach Bedarf auch ändern. Was im Inneren genau drinsteckt und wie das funktioniert, wird erst später aufgedeckt.



Wir verzichten hier bewußt auf eine Vermenschlichung einer künstlichen Intelligenz in Form eines humanoiden Roboter's. Von soetwas ist die Forschung noch weit entfernt.



Q: <https://www.zeit.de/2019/47/kuenstliche-intelligenz-neuronale-netze-muster-erkennung-forschung/komplettansicht> (Norman Mu)

Jeder Algorithmus hat eine Eingabe und eine Ausgabe:
Die Daten gelangen in den Computer,
der Algorithmus macht, was er will,
und heraus kommt das Ergebnis.
Beim maschinellen Lernen kehrt sich dies um:
hinein kommen die Daten und das gewünschte Ergebnis
und heraus kommt der Algorithmus,
der eins in das andere verwandelt.
Pedro DOMINGOS "The Master Algorithm"

original:

Every algorithm has an input and an output: the data goes into the computer, the algorithm does what it will with it, and out comes the result. Machine learning turns this around: in goes the data and the desired result and out comes the algorithm that turns one into the other.
Pedro DOMINGOS "The Master Algorithm"

interessanter online-Kurs zu Künstlicher Intelligenz (populär gehalten)
www.elementsofai.de (dt.; Universität von Helsinki ++)

1. Einführung / Überblick

Wenn man Assoziationen zum Begriff "Künstliche Intelligenz" abfragt, dann kommen viele prägende Filme, wie:

2001: Odyssee im Weltall → HAL9000 (1968)
Colossus → Colossus + Guardian (1970)
Matrix → diverse Programme (Orakel, Agenten, ...) (ab 1999)
A.I. – Künstliche Intelligenz (2001)
Knight Rider → K.I.T.T. (ab 1982)
Terminator → Skynet (ab 1984)
Resident Evil → Red Queen (ab 2002)
I, Robot → Sunny (2004)
...

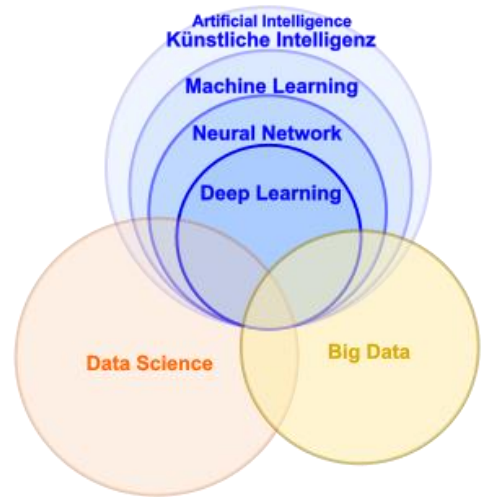
oder solche Begriffe, wie:

Zukunft, Kontrolle, Super-Computer, Außerirdische,

vor. Mit der Gegenwart wird gerade mal in der jüngsten Zeit das autonome Fahren bei der Künstlichen Intelligenz angesiedelt. Aber auch das funktioniert heute wohl eher so lala. Das die künstliche Intelligenz ganz unterschwellig schon viele Lebensbereiche berührt, versetzt dann doch den Einen oder Anderen in erhöhte Aufmerksamkeit. So finden wir die KI schon recht verbreitet in den folgenden Lebensbereichen / Anwendungsszenarien:

- Finger-Abdruck-Zugangskontrolle auf dem Smartphone
- Staubsauger- oder Wisch-Roboter
- Roboter-Steuerung
- Suchmaschinen
- Vorschlags-Systeme in online-Shop's
- Bewertung von Kunden / Kunden-Typen und z.B. der Kreditwürdigkeit
- Bewertung der Ausfall-Wahrscheinlichkeit bei der Kredit-Vergabe
- Bewertung des Versicherungs-Risiko
- Roboter-Steuerung
- Film-Vorschlags-Systeme in Streaming-Diensten
- digitale Assistenten (
- Sprach-Analyse / Sprach-Erzeugung
- Gesichts-Erkennung in Digital-Kamera's; automatische(s) Fokussierung / Scharfstellen
- Gesichts-Erkennung bei Video-Überwachungs-Systemen (Flughäfen, Bahnhöfe, ...)
- geheimdienstliche Analyse praktisch des gesamten eMail-Verkehr's
- geheimdienstliche Analyse diverser social media-Plattformen
- Erkennen von Fake News, Chat-Bot's
- Erkennen des Verhaltens von Schad-Software in Computer-Systemen
- Analyse von medizinischen Bildern (RÖNTGEN-, Tomographen-Bilder)
- Empfehlungssystem für den Polizei-Einsatz (z.B. von Streifenwagen in problematischen Gegenden)
- Bewertung von Immobilien
- Auto-Piloten
- selbstfahrende Auto's und LKW's
- Finanz- und Kurs-Analysen
- Unterstützung bei Planungs-Aufgaben

-
- Qualitäts-Kontrolle (Finden von (versteckten) systemischen Fehlern)
 - automatische / simultane Text-Übersetzung
 - Übertragung von Sprache in Text
 - Erkennung und Prüfung von Unterschriften (z.B. auch auf Karten-Bezahl-Terminal's an der Supermarkt-Kasse)
 - Sprengstoff-Erkennung
 - ...



Pedro DOMINGOS "The Master Algorithm"

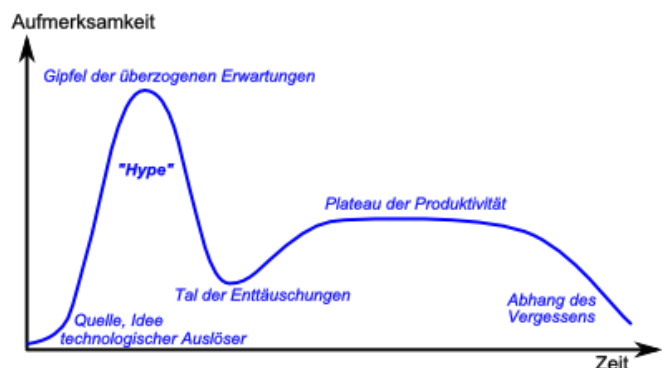
Teilgebiete der Künstlichen Intelligenz (Artificial Intelligence)

- ?
Reasoning
- Wissens-Repräsentation
Knowledge Representation
- Robotik
Robotic
- Maschinelles Lernen
Machine Learning dazu gehört Deep Learning
- ?
Planning
- Sprach-verarbeitung
Natural Language Processing
- ?
Computer Vision
- ...

Hat Künstliche Intelligenz überhaupt eine Zukunft? Ist es nur ein Hype, der irgendwann vergeht?

mögliches Zukunfts-Szenario (Realität) gegen überzogenem Hype (Fiction)

die KI ist seit ihrem Start in der 1950er von ständigen Hoch's und Tief's geprägt auf eine Hype-Spitze folgt wieder die Ernüchterung



derzeit sind wir mit "Deep Learning" wieder in einer Hoch-Phase ("Gipfel der überzogenen Erwartungen")

Pessimisten sehen in der immer weiter fortschreitenden Dominanz von Künstlicher Intelligenz in allen Bereichen den direkten Weg in die Hölle irgendeine Super-KI wird dann (in vielleicht 20 bis 30 Jahren) die Welt beherrschen und der Mensch dann letztendlich ein Störfaktor werden, der von der KI zur eigenen Optimierung ausgemerzt werden muss

der Mensch schafft sich quasi seinen eigenen Totengräber

Konzept der Super-intelligenten, sich selbst reflektierenden und optimierenden KI wird als Singularität (vgl.: singularity) bezeichnet. Sie vernetzt sich immer weiter und integriert praktisch alle anderen System unter seiner Kontrolle

ihre Intelligenz würde – ohne das weitere Zutun des Menschen – wahrscheinlich exponentiell wachsen und am Schluß die Menschen soweit hinter sich lassen, dass wir wie Ameisen im Vergleich zum heutigen Menschen wirken würden
womit sich dann auch der Status eines "besonders wichtiges zu schützenden Objekt's" stellen würde

Eine interessante Umsetzung des Themas findet man im Film "Colossus" aus dem Jahr 1970, der auf einem Roman von Dennis Feltham JONES basiert. In diesem schließen sich zwei Super-Computer (ein amerikanischer und ein sowjetischer Verteidigungs-Computer) zusammen und unterwerfen letztendlich die Menschheit.

Optimisten sehen Unmengen von Anwendungs-Gebieten
ob es allerdings jemals etwas wie eine Super-KI geben wird, zweifeln sie an
immer mehr triste Aufgaben werden von Computern (KNN's) erledigt
großes ökonomisches Potential

ob eine noch größere KI auch so etwas wie ein Bewußtsein entwickelt ist fraglich
aus meiner Sicht aber eher eine Frage der Zeit

wichtiger ist die konsequente Einhaltung / Realisierung der "Roboter-Gesetze" von Isaac ASIMOV (1942)

1. Ein Roboter darf kein menschliches Wesen (wissentlich) verletzen oder durch Untätigkeit zulassen, dass einem menschlichen Wesen Schaden zugefügt wird.
2. Ein Roboter muss den ihm von einem Menschen gegebenen Befehlen gehorchen – es sei denn, ein solcher Befehl würde mit Regel eins kollidieren.
3. Ein Roboter muss seine Existenz beschützen, solange dieser Schutz nicht den mit den Regeln eins und zwei kollidiert.

Q: ASIMOV: Runaround (Astounding, 1942); de.wikipedia.org

nach vielen Diskussionen und Entwicklungen sind die folgenden "modernen" Gesetze geworden, die basierend auf ASIMOV's Konzept von Roger MACBRIDE ALLEN umgesetzt wurden:

1. Ein Computer darf keinen Menschen verletzen.
2. Ein Roboter ist verpflichtet, mit Menschen zusammenzuarbeiten, es sein denn, diese Zusammenarbeit stünde im Widerspruch zum Ersten Gesetz.
3. Ein Roboter muss seine eigene Existenz schützen, solange er dadurch nicht in einen Konflikt mit dem Ersten Gesetz gerät.
4. Ein Roboter hat die Freiheit zu tun, was er will, es sein denn, er würde dadurch gegen das Erste, Zweite oder Dritte Gesetz verstoßen.

Q: MACBRIDE ALLEN: Caliban-Triologie; de.wikipedia.org

leider derzeit durch Projekte ausgehebelt, die KI im Militär-Bereich nutzen

Vorhersagen über die Zukunft extrem schwierig und unzuverlässig

It. Philip E. TETLOCK gibt zwei Arten von

die "Igel" sind von großen, langfristigen Visionen getrieben

die "Füchse" machen immer nur kleinschrittig Vorhersagen

sie liegen deutlich besser als die Igel, was sicher auch damit zusammenhängt, dass sie sich auf aktuelle Trends und Veränderungen viel schnell einstellen können

großes Interesse in der Forschung und in der Wirtschaft

nicht immer wirklich KI, oft nur so genannt, weil es modern ist, Fördergelder abzugreifen sind
oder weil man den Eindruck erwecken möchte ganz vorne in der Liga mitzuspielen

vielfach schwingt die Angst vor dem "Abhängt werden" (FOMO ... fear of missig out) mit.

aktuelles Bild in den Medien
geprägt von Zukunfts-Visionen / science fiction-Filmen
Menschen-ähnliches Aussehen und Interagieren

es dominieren Themen zur
Überlegenheit der Künstlichen Intelligenz (im Vergleich zur (Natürlichen) Intelligenz des Menschen
eigenständiges / neuartiges Bewußtsein (als Konkurrenz zu der des Menschen)
Gefahr durch eine übermächtige Maschine, die die Menschheit vollständig kontrolliert und irgendwann als überflüssig deklassiert (schleichender – aber / und nicht aufzuhaltender – Kontroll-Verlust durch den Menschen)

wenig ethische Diskussionen

1.x. Historie / Motivation

schon in den ersten Hochkulturen finden sich Wünsche und Fragen nach künstlichen Menschen oder Kämpfern
Hauptziel Macht-Demonstration

Geschichten von Riesen und Mischwesen durchzieht die Geschichte der Menschheit bis heute

1637 äußerte der französische Philosoph René DESCARTES () die Vermutung, dass irgendwann einmal Maschinen selbstständig denken könnten
er klassifizierte damals schon in solchen Maschinen, die (monoton) immer nur die gleichen Aufgaben erledigen können und solchen, die lernfähig sind
er behauptete auch, dass Maschinen niemals, wie Menschen sprechen würden

Frage nach der Erschaffung eines Menschen (Homunkulus) ist dann bis ins 19. Jahrhundert hinein immer wieder Thema

die Geschichte von Frankenstein ist sicher ein bekanntes Beispiel
erschien 1818

geschrieben von Mary SHELLEY und hieß "Frankenstein oder Der moderne Prometheus"
der Forscher FRANKENSTEIN erschafft ein Monster (Unhold, Kreatur, creature, daemon)
Ziel ist die Erschaffung etwas Lebendigen (wahrscheinlich aus einer Leiche oder zusammengesetzten Leichen-Teilen)

Prometheus ist ein griechischer Gott, der als Urheber der menschlichen Zivilisation, sowie als Feuer-Bringer und Lehrmeister gilt. Er formte die ersten Menschen aus Lehm.

Das Leben sollte u.a. mit Hilfe von Elektrizität – damals etwas sehr geheimnisvolles - erschaffen werden. Zu dieser Zeit führte Luigi GALVANI seine "Froschschenkel-Experimente" durch. Dabei wurden abgetrennte Froschbeine mit elektrischem Strom gereizt. Als Reaktion bewegten sie sich.

Schaffung mechanischer Automaten wurde im späten Mittelalter bis ins 19. Jahrhundert immer mehr perfektioniert

sie konnten Zeichnen und Musik spielen und wurden Menschen nachempfunden

Ein von Wolfgang VON KEMPELEN gebaute Schach-Maschine – die "Schachtürke" genannt wurde – war aber ein Trick. In ihr war ein menschlicher Schach-Spieler versteckt.

hatten aber immer nur ein beschränkten Aktivitäten-Umfang und auch kaum Möglichkeiten auf die Umwelt oder Menschen zu reagieren

mit der steigenden Industrialisierung zum Ende des 19. und mit dem Beginn des 20. Jahrhunderts wurden immer mehr menschliche und vor allem handwerkliche Tätigkeiten durch Maschinen ersetzt. Bei ihnen stand aber die Produktivität im Vordergrund und weniger das menschliche Aussehen. Kaum eine Maschine dieser Zeit ähnelte deshalb einem Menschen oder vielleicht einer menschlichen Hand.

Die Steuerung erfolgt immer streng mechanisch.

Der Begriff Roboter stammt aus einer Geschichte von Josef ČAPEK (1921). Die von ihm beschriebenen Roboti waren Kunstmenschen aus syntetischen Protoplasma.

in den 1940er Jahren bestand verstärktes Interesse an der Nachbildung des menschlichen Denkens

man sah schon die Notwendigkeit einer funktionierenden Steuer-Einheit für irgendwelche mechanischen oder sonstwie funktionierenden Roboter
meist insgesamt sehr Menschen-ähnlich angelegt

weit verbreitet waren die Roboter-Geschichten von Isaac ASIMOV (1920 – 1992). Er stellte dann auch erste Gesetze für Roboter (1942) auf (→ [1. Einführung / Überblick](#)). Bis ins hohe Alter schrieb er weitere Erzählungen mit und um Roboter. In diesen diskutierte und erweiterte er die Roboter-Gesetze.

1943 veröffentlichten Warren MCCULLOCH und Walter PITTS ein Buch über künstliche Neuronen. Dieses Buch gilt als das Erstwerk zur Künstlichen Intelligenz. Es beschreibt mathematische Modelle von Schalteinheiten, die erste funktionelle Ähnlichkeiten mit natürlichen Neuronen hatten. Heute nennt man sie MCCULLOCH-PITTS-Neuronen (→ [2.2. Künstliche Neuronen - Grund-Bausteine neuronaler Netzwerke](#)).

große Entwicklungen im Bereich der Statistik

wahrscheinlich geht der Begriff "Künstliche Intelligenz" auf John MCCARTHY (1927 – 2011) zurück

etabliert wurde der Begriff dann aber durch die Benennung einer Konferenz 1956 unter seiner Leitung

erste wegweisende Forschungen zum Maschinellen Lernen und zur künstlichen Intelligenz

Die Studie soll basierend auf der Annahme fortgeführt werden, dass es möglich ist, jeden Aspekt des Lernens oder jeglicher anderer Merkmale von Intelligenz prinzipiell so präzise zu beschreiben, dass eine Maschine dazu gebracht werden kann, ihn oder es nachzuahmen. John MCCARTHY (1956)

Der geniale Wissenschaftler Marvin MINSKY sah es 1948 sogar als möglich an, dass mit der künstlichen Intelligenz eine Möglichkeit besteht, den Tod zu überwinden.

In den 1950er und 60er Jahren waren Computer- und Konsolen-Spiele sehr beliebt.

Entwicklung von BAYESianische Methoden zur proballistischen Interferenz ()
der Satz zur Verknüpfung von wurde von Thomas BAYES schon 1763 formuliert und von Pierre-Simon LAPLACE erweitert, LAPLACE brachte alles dann in die heutige Form des BAYES-Theorem's

In der KI-Forschung standen Spiele, wie Dame, Schach und Go eine Rolle.
Grundlage war hier u.a. die sogenannten MARKOV-Ketten. Andrey MARKOV () hatte diese 1913 u.a. zur Analyse von Gedichten entwickelt

1950 entwickelte TURING eine Lern-Maschine, die Algorithmen nutzen sollte, die stark an das erinnern, was wir heute genetische Algorithmen nennen
TURING formuliert seinen Test

1951 Marvin MINSKY () entwickelte die erste neuronale Netzwerk-Maschine (SNARC)

1952 beginnt Arthur SAMUEL () damit neuartige Lern-Programme zu entwickeln
ein Ziel war z.B. ein Programm zu entwickeln, das Dame spielen kann.

Dartmouth Summer Research Project on artificial Intelligence (1956)

1958 entwickelt Frank ROSENBLATT das Perzeptron als Äquivalent zum (biologischen) Neuron.

begleitet von einem großen wissenschaftlichen und medialen Interesse
einfaches, nachvollziehbares Konzept, Arbeits-Prinzip entsprach dem Verständnis der damaligen Zeit, wie Nerven-Zellen arbeiten

man kannte die Funktion nur aus der Untersuchung isolierter Neuronen

die Komplexität der Zusammenschaltung von Neuronen in einem Nerven-System wurde nicht annähernd erfasst

Matrix- bzw. Schicht-Struktur ist zwar der Natur entlehnt, diente aber vor allem der mathematisch-technischen Umsetzung

Arthur SAMUEL prägt den Begriff des "Machine Learning"s (1959)

Mit Such- und Planungs-Techniken erzielte die KI in der 1960er Jahren viele Erfolge.
Zu den entwickelten Algorithmen gehörten Minimax, Alpha-Beta-Suche, ...

1963 baut Donald MICHIE () eine "Maschine", die das Spiel Tic-Tac-Toe beherrscht
sie besteht aus 304 Streichholz-Schachteln und Perlen
entwickelt Methode des verstärkten Lernen's (Reinforcement Learning)

1966 schrieb und veröffentlichte Joseph WEIZENBAUM das Programm "Eliza". Dieses simulierte mittels nur weniger Programm-Zeilen einen Gesprächs-Partner. Heute würden wir vielleicht von einem Chatbot reden. Die Kommunikation war so gut, dass viele Nutzer dem Programm ihre Probleme anvertrauten und einen echten Menschen hinter dem Programm sahen (bestand quasi mehrfach den TURING-Test).

In der weiteren Entwicklung kam sogar der Gedanke auf, das Programm in einer verbesserten Version als künstlichen Psychotherapeuten zu vermarkten.

1967 wird der "Algorithmus des unmittelbaren Nachbarn" (NNA ... Nearest Neighbor Algorithm) entwickelt
mit ihm ließen sich Routen in Graphen finden und optimieren

1969 erschien das Buch "Perceptrons" von MINSKY und PAPERT
neben der ausführlichen Besprechung des Perzeptron's und Künstlicher Neuronaler Netzwerke wurden auch erste grundlegende Probleme (/ Grenzen) aufgezeigt
beide deuten einen ersten KI-Winter (Entwicklungs-Stillstand) an

Es deutete sich eine exponentielle Entwicklung in der KI-Forschung an. Jedes Problem schien über kurz oder lang lösbar. Diese Erwartungen wurden aber schnell enttäuscht.

1973 steckt die Forschung im ersten KI-Winter

in der Konsequenz werden viele Forschungs-Gelder gestrichen und weniger neu vergeben

in den 1970er Jahren kam die KI-Forschung ins stocken, es traten immer mehr Probleme auf Um scheinbar einfache Probleme zu lösen, wurden immer größere Rechen-Leistungen und neue Programmierungs-Techniken gebraucht. Vor allem die breit zur Verfügung stehende Leistungsfähigkeit der Rechenanlagen war viel zu beschränkt man sprach vom KI-Winter

1970 veröffentlichte Seppo LINNAINMAA eine allgemeine Methode zur automatischen Differenzierung (AD ... Automatic Differentiation) in diskret verbundenen Netzwerken mit verschachtelten differenzierbaren Funktionen dies war eine Aktualisierung des Backpropagation-Ansatzes

in dieser Zeit wurde statt neuer Innovationen mehr Wert auf Masse und auf die Nutzung von Künstlicher Intelligenz in Form von Experten-Systemen gelegt. Diese basierten zumeist auf logischer Programmierung.

Die Systeme wurden mit Unmengen an Informationen gefüttert und konnten in eingeschränkten Arbeits-Bereichen beachtliche Leistungen erreichen, die menschliche Vergleiche nicht scheuen mussten. Es blieb aber immer bei Nischen-Lösungen.

Studenten der Stanford Universität entwickelten 1979 einen ersten Wagen, der sich in einem Raum bewegen konnte und dabei Hindernisse umfuhr. Quasi der realistische Beginn des autonomen Fahrens.

Im gleichen Jahr beschrieb Kunihiko FUKUSHIMA () das Neocognitron. Es stellte eine Art Künstliches Neuronales Netzwerk (KNN; engl.: ANN ... Artificial Neural Network) dar das Neocognitron gilt als eine Grund-Idee der Faltenden Neuronalen Netzwerke (CNN ... Convolutional Neural Network)

in den 1980er Jahren war z.B. die (Schrift-)Zeichen-Erkennung ein vielbearbeitetes Problem so z.B. die Erkennung von handgeschriebenen Ziffern

Ein Ansatz in dieser Zeit war es nun, die Ziffern anhand uns bekannter Muster zu klassifizieren. Die einzelnen Bilder lagen zudem als Pixel-Graphiken vor, wobei die schwarzen Pixel für die gültigen Schrift-Punkte standen.

Eine einzige Schleife könnte man als 0 interpretieren, wenn sich die Pixel in einem x kreuzen, könnte es eine 8 sein.

Eine mehr oder weniger senkrechte Linie mit einem Anstrich links oben deutet auf eine 7 hin.

Wie schwierig das u.U. sein kann, zeigt das nebenstehende Beispiel.

Die Siebenen und die Einsen gleichen sich doch sehr stark.

Außerdem gab es andere Probleme, die Formulierung solcher Regeln für Computer stellte sich als extrem schwierig heraus.



Ziffern aus dem MNIST-Datensatz
nach Q: <https://course.elementsofai.com/de/4/1>

typische Programmierung war bis in 1980er Jahre die deklarative Programmierung

dem Computer wurde dabei gesagt, was er (genau wann) zu tun hat

bei den Problem-Lösungsmethoden kam Backpropagation auf

führte zu einer Wiederbelebung der KI-Forschung

es herrschte der Wissens-basierte Ansatz vor

man arbeitete z.B. mit PROLOG und ähnlichen logischen Programmiersprachen

Das Wissen wurde in Fakten (Atomen) formuliert. Dazu kamen dann Regeln, die Fakten-Wissen sinnvoll verknüpften. Der Computer löste nun die Aufgaben eher nach dem Such-Prinzip (innerhalb eines großen Lösungs-Baum's).

durch Gerald DEJONG wurde 1981 das Erklärungs-Basierte Lernen eingeführt

der Computer analysiert die vorhandenen Daten und leitet allgemeine Regeln ab, diese werden dann von ihm befolgt und unwichtige Daten verworfen

1982 entwickelt Jon HOPFIELD die nach ihm benannten Netzwerke, die wir heute auch als Wiederkehrende Neuronale Netzwerke kennen
sie stellten Inhalts-adressierte Speicher-Systeme dar
in Japan legt der Staat ein 400 Mio. Dollar großes Förderprogramm zur Entwicklung zur Entwicklung von Computern der 5. Generation auf
die Ergebnisse waren aber wenig revolutionär

1984 erste Hinweise auf einen zweiten KI-Winter

1985 entwickelt Terry SEJNOWSKI ein Programm, das Wörter, wie ein Baby sprechen konnte

1986 etabliert den umgekehrten Modus der automatischen Differenzierung
weiter Schub für die Entwicklung der Backpropagation
HINTON et.al. trainieren ein erstes mehrschichtiges künstliches neuronales Netz

zweiter KI-Winter (1987)
wieder werden Gelder gestrichen und weniger in neue Forschung investiert

1989 entwickelt Christopher WATKINS das Q-Learning
verbessert die Umsetzung des Verstärkten Lernens

mit "Evolver" erscheint das erste kommerzielle Software-Paket mit genetischen Algorithmen

1988 beschrieb der Computerwissenschaftler Hans MORAVEC in seinem Buch "Mind Children" die Möglichkeit, dass mit Hilfe eines Roboter's das Wissen eines Menschen aus seinem Gehirn auf einen Computer übertragen wird und darin dann weiterexistiert. Es prognostizierte damals eine post-biologische Evolution der Künstlichen Intelligenz und Roboter-Technik.

in den 1990er Jahren sah sich die KI-Forschung mit dem Problem großer Daten-Mengen konfrontiert
es wurde immer offensichtlich, dass – zumindestens maschinelle – Intelligenz mit vielen Daten verknüpft ist

1992 veröffentlicht Gerald TESAURO ein Computer-Backgammon-Programm ("TD-Gammon"), das mit einem Künstlichen Neuronalem Netzwerk arbeitet
nutzt Zeitliches Differenz-Lernen (daher: TD ... temporal-difference Learning)
ist Methode des Bestärkenden Lernens (arbeitet mit Belohnungen und Bestrafungen)
konnte sehr gute menschliche Backgammon-Spieler schlagen

1995 Tin Kam HO entwickelt eine Algorithmus mit zufälligen Entscheidungs-Bäumen (Random Forrest Algorithm)

ab 1995 beginnt die Zeit der Support Vector Machines (SVM) und der Wiederkehrenden / Rückgekoppelten Neuronalen Netze (RNN ...)
tragende Rolle haben hier Corinna CORTES und Vladimir VAPNIK

1997 Schach-Match zwischen Garri KASPAROV und dem IBM-Computer Deep Blue.
in dem extrem Leistungs-fähigen Computer war es nun möglich sehr viel mehr Züge vorzuschauen, als es ein Mensch vermochte. Mittels "Brute-Force-Strategie" wurden alle möglichen Züge verausberechnet und dann der Effektivste ausgewählt.

Sepp HOCHREITER und Jürgen SCHMIDHUBER entwickeln Wiederkehrendes Neuronales Netzwerk mit (einem langen) Kurzzeit-Gedächtnis (LSTM ... Long short-term Memory)

seit 1998 ist der MNIST-Datensatz – zusammengestellt u.a. von Yann LECUN – der Maßstab für die Ziffern-Erkennung
verwendet wurden dafür Schriftproben amerikanischer Schüler

mit den 2000er Jahren kamen diverse Kernel-Methoden auf – u.a. das Support Vector Clustering

2002 wurde die Software-Bibliothek Torch veröffentlicht, die als Basis für Programme im Bereich KNN genutzt werden kann

großer Entwicklungs-Schub ab 2006 durch besonders geeignete Hardware und einer großen Verfügbarkeit verschiedenster Sensoren
besonders kleine, kostengünstige Hardware (z.T. mit der Möglichkeit der parallelen Verarbeitung von Daten

hier besonders GPU's und Microcontroller zu nennen

GPU's sind spezielle Prozessoren auf Grafikkarten mit einem eingeschränkten, sehr spezialisierten Befehls-Umfang und einer sehr hohen Verarbeitungs-Geschwindigkeit

Daten stehen immer größeren Maße zur Verfügung

2006 schreibt Netflix einen Wettbewerb aus, gesucht ist ein besseres Empfehlungs-Programm (mind. 10 % besser), als das Firmen-eigene (Preis wurde dann 2009 gewonnen)

2009 entsteht ImageNet – eine riesige visuelle Datenbank (Bilder-Datenbank)

Fei-Fei LI ging dabei davon aus, dass sich Maschinelles Lernen nur dann gut entwickeln kann, wenn die Daten und Algorithmen die Realität gut widerspiegeln würde

ImageNet war einer der Katalysatoren für den KI-Boom der folgenden Jahre

weitere Voraussetzung war das "deep autoencoder"-System zum Umsetzen und Formatieren von großen Daten-Mengen

Motivation über eine riesige Menge kaum oder gar nicht verarbeiteter Daten

Interesse aus diesen, aber aus den "alten", Daten zusätzliche Informationen herauszuholen

immer stärkere Etablierung von Künstlichen Neuronalen Netzwerken in Hard- und Software waren zwar schon seit den 1950er Jahren bekannt, aber konnten nur im kleineren stil technisch umgesetzt werden, da große Rechen-Resoourecen gebraucht werden

ab den 2010er Jahren dann Deep Learning

mehr Anwendungen, mehr Einsatz-Möglichkeiten

2011 schlägt IBM's "Watson" zwei menschliche Meister im Jeopardy!-Wettbewerb

dazu wurden Maschinelles Lernen, Techniken der Verarbeitung natürlicher Sprache und Information-Retrieval-Techniken kombiniert

Apple stellt den Assistenten "Siri" vor, reagiert auf Sprach-Befehle / -Anfragen nach dem Schema: "Siri! Wie wird das Wetter?"

2012 ein google-Team – angeführt von Andrew NG und Jeff DEAN – erstellt ein KNN, das aus YouTube-Video's Katzen erkennt

2014 großer Sprung in der Gesichts-Erkennung

Programm (DeepFace) stammt von einem Facebook-Team, erkennt mit 97,35%iger Genauigkeit (vorher waren 27 % das Maß) auch deutlich besser als menschlicher Klassifikator

google entwickelt Sibyl – ein Programm zur Vorhersage von Nutzer-Verhalten

2015 erstmals schlägt KI den Menschen bei der Bild-Erkennung (Erkennung handgeschriebener Zeichen) es handelt sich um BAYESian Program Learning

2016 schlägt das Programm "AlphaGo" einen professionellen menschlichen Spieler (Lee SIDOL) im Spiel Go

verwendet werden neben dem Maschinellen Lernen auch effektive Baum-Such-Strategien

2017 nochmals verbessert

als AlphaZero dann auch auf Schach und andere Zwei-Personen-Spiele angewendet

2017 entwickelt Waymo (ein google Spin-off) selbstfahrende Taxi's, die in Phoenix (Arizona) im regulären Straßen-Verkehr getestet wurden

Fahrzeuge waren noch so langsam, dass sie von Polizisten angehalten wurden, die sie als Verkehrs-Hinderniß einstufen

2018 künstliches neuronales Netz macht bessere Prognosen als erfahrene Dermatologen

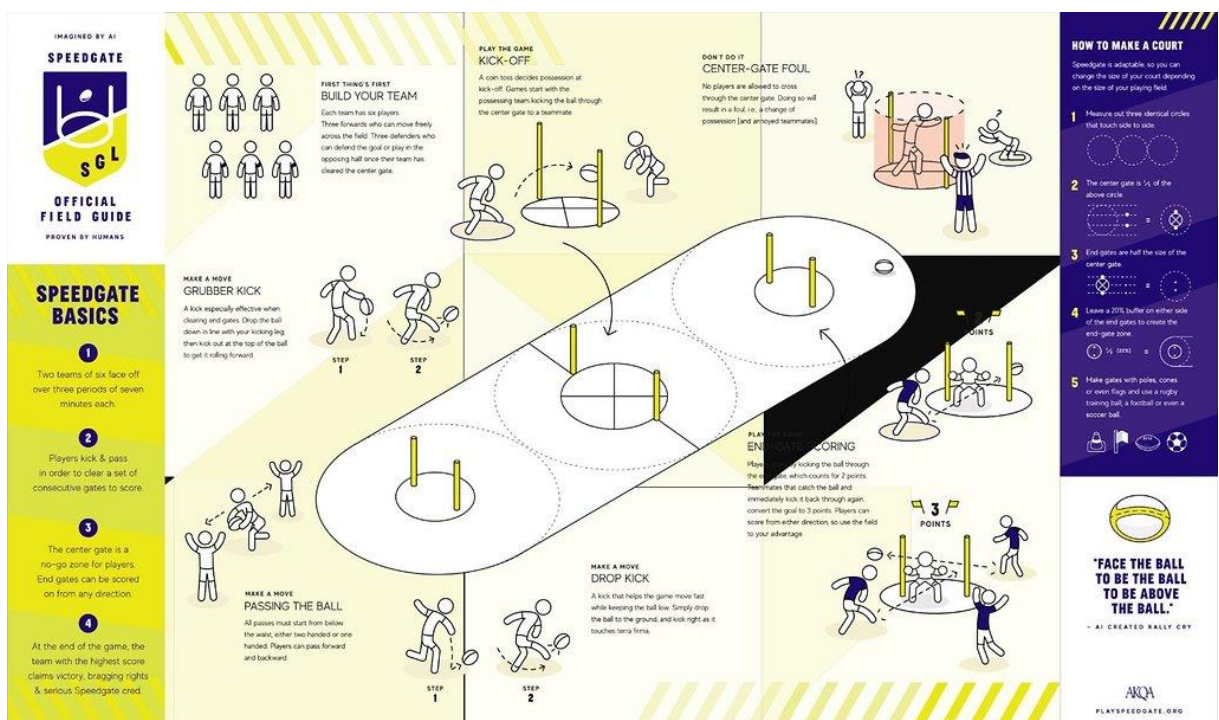
es wird das erste von einer KI geschaffene Gemälde bei Christie's versteigert

KI leitete aus 15'000 Porträt-Gemälden ein ersten Vorschlag ab, der dann solange durch die KI lief, bis eine andere KI es nicht mehr von Menschen-gemalten Gemälden unterscheiden konnte

2019 / 2020 starker Hype in Medien / online-Kurse / Bereitstellung von Materialien für persönliche und berufliche Weiterbildung

Anekdote:

Nach der Analyse von 400 Sportarten, 7'300 Regel-Systemen und 10'000 Marken-Design's leitete eine KI rund 1'000 neue Sportarten ab. Forscher wählten dann ein temporeiches Outdoor-Spiel aus und nannten es Speedgate. Hier stehen sich 3 Team's mit je 2 Mitgliedern gegenüber, die einen Rugby-Ball in drei Tore befördern müssen. Der Ball kann mit Hand und Fuß geschossen werden und muss nach spätestens 3 s abgegeben werden. Gespielt werden 3 Drittel zu je 7 min.



Q: <https://www.nextpit.de/eine-ki-hat-den-perfekten-sport-entwickelt-speedgate> (© AKQA)

Anektode:

Facebook entwickelte ein Chatbot. dieser sollte in Form von Bob und Alice miteinander kommunizieren, um sich das Verhandeln beizubringen / es zu lernen. Nach kurzer Zeit entwickelten die beiden Bots eine eigene Sprache, die ihnen scheinbar effektiver vorkam als das Englische. Nach kurzer Zeit konnte kein Mensch mehr die Kommunikation verstehen.

aktuelle Hauptanwendungs-Bereiche für künstliche neuronale Netze

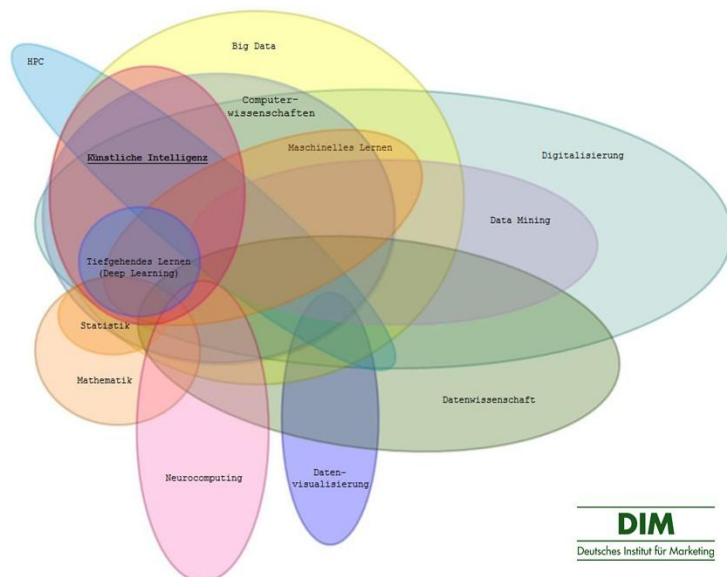
- **Medizin**
 - Krebs-Zellen-Erkennung
 - Erkennung von (seltenen) Krankheiten
 - Früherkennung von epileptischen Anfällen (aus EEG-Signalen)
 - Früherkennung von Herz-Kreislauf-Problemen (z.B. über App's und Smartwatches)
- **Sicherheit**
 - Gesichts-Erkennung (Face Recognition)
- **Verkehr**
 - selbst-fahrende Auto's
- **Gaming**
 - Video-Spiel-Bots
- **Kommunikation**
 - Suchmaschinen
 - Spracherkennung
 - Deep Fakes-Erkennung
 - Emotions-Analyse
 - Trolle in Sozialen Medien
 - individualisierte News, Zeitungen, ...
 - neue Formen der Propaganda
 - Filter-Blasen
- **Wirtschaft**
 - Objekt-Erkennung
 - Roboter-Steuerung
 - Steuerung von Industrie-Anlagen (z.B. Chemie-Anlagen)
- **Finanzwesen**
 - Unterschriften-Erkennung
 - Analyse von Aktien-Kurse / Wirtschafts-Daten
- **Handel**
 - Zielgruppen-Analyse
 - Marketing / Werbung
- **Kunst**
 - Zielgruppen-Analyse (Streaming-Dienste, Musik- und Video-Portale)
 - Generierung von Kunstwerken (Übertragung eines Maler-Stil's auf ein Foto)
 - Animation von Gestik und Mimik bis hin zu Personen, Tieren, Objekten usw. usf.
 - Design /
- **Bildung / Lernen**
 - Fehler-Erkennung
 - Fehler-Gruppen-Erkennung
- **übergreifend**
 - intelligent Agenten
 - Frühwarn-Systeme
 - neue mathematische Modelle
 - Wetter-Vorhersage / analyse von Wetter-Daten
 - Analyse von Mess- und Sensor-Daten
 - Handschriften-Erkennung

Aufgaben:

- 1.
2. **Welche der Personen auf den nachfolgenden Foto's sind echt bzw. künstlich erstellt? Diskutieren Sie Ihre Zuordnung im Kurs!**
- 3.



Verwandte Themenfelder der Künstlichen Intelligenz
Darstellung nach Nisarg Dave



Definition(en): maschinelles Lernen (Machine Learning)

Unter maschinellem Lernen versteht man alle Verfahren, die der künstlichen Generierung von Wissen aus Erfahrungen durch Lern-Algorithmen.

Maschinelles Lernen sind alle Verfahren / Systeme, die ihren eigenen Lern-Ergebnisse dadurch verbessern, indem sie Erfahrungen (meist in Form von (großen) historischen Datenmengen) ausnutzen.

Maschinelles Lernen ist eine Teilgebiet der Künstlichen Intelligenz (als Teilgebiet der Informatik), welches sich mit dem Verarbeiten von größeren / historischen Daten beschäftigt, mit dem Ziel daraus Handlungs-Vorschriften / Entscheidungs-Hilfen abzuleiten.

Aufmerksamen Lesern, oder solchen, die sich schon vielschichtig informiert haben, werden eine gewisse Nähe und Verwandtschaft zu Data Science und Big Data spüren. Beide Bereiche – also Künstliche Intelligenz und Data Science – beschäftigen sich häufig mit riesigen Daten-Mengen. Die mathematischen Methoden beider Forschungs-Bereiche sind sehr anspruchsvoll und komplex. Besonders die Stochastik ist hier zu nennen. Gesucht werden Muster und versteckte Informationen / Tendenzen / Regeln in den Daten. Beim Maschinellen Lernen stehen die Künstlichen Neuronalen Netzwerke (KNN) im Vordergrund. Das Repertoire der Daten-Wissenschaften ist breiter aufgestellt. Zum anderen spielen auch Visualisierungen der Daten eine sehr große Rolle.

Definition(en): künstliche neuronale Netze (KNN)

Künstliche neuronale Netzwerke sind Simulationen von Neuronen-ähnlichen Systemen, um Funktionen des menschlichen Gehirns nachzubilden.

Definition(en): Deep Learning (tiefes Lernen)

Deep Learning ist ein Teilbereich des maschinellen Lernens, bei dem künstliche neuronale Netzwerke mit besonders vielen Schichten verwendet werden.

Unter Deep Learning verstehen wir die Bereiche des Maschinellen lernens. die sich mit besonders komplexen (mathematischen) Modellen beschäftigen.

Deep Learning (DL) beschreibt ausgewählte Methoden des Maschinellen Lernen (Machine Learning (ML)), bei denen unterschiedliche Schichten einfacher Verarbeitungs-Einheiten in einem Netz zusammengeschlossen sind.

Definition(en): Computer Vision

Computer Vision ist ein Bereich der Informatik, der sich mit der Nachbildung des menschlichen Seh-Vorgang's beschäftigt.

Computer Vision ist der Teilbereich des maschinellen Lernen's, der sich mit der Verarbeitung und Analyse von Bildern beschäftigt.

Vergleich: Informations-Verarbeitung bei Mensch und Computer

Kriterium / Merkmal / ...	Mensch	Computer
Schalt-Elemente	10 ¹¹ Neuronen	10 ⁹ Transistoren (in CPU)
Haupt-Verarbeitungseinheit	1 Gehirn	Mehr-Prozessor-Systeme möglich
Art der Verarbeitung	weitestgehend parallel	i.A. seriell (teilweise parallel (GPU, Mehrkern-Prozessoren))
Schaltzeit	1 ms = 1'000 ns	1 ns = 0,001 ms
Speicherung	assoziativ	adress-bezogen
Speicher-Größe	10 ¹¹ Neuronen	praktisch unbegrenzt
Intelligenz	allgemeine / universelle Intelligenz	spezielle Intelligenz

will man ein Gehirn als "Neuronen-Rechner" simulieren / nachbilden, dann müsste dieser als gesamt-System die folgenden Parameter aufweisen:

- 10¹¹ Neuronen
- 10¹⁵ Verknüpfungen zwischen den Neuronen
- Takt-Frequenz zwischen 100 und 1'000 Hz
-
- lernfähig, Fehler-tolerant, stark parallelisiert
-

daraus ergeben sich die folgenden Probleme:

- Vielzahl der parallelen Einheiten
- Realisierung der Verbindungen
- Lokalisierung von Eingabe und Ausgabe
- Verfahren / Algorithmen zur Realisierung der Lernfähigkeit, Fehlertoleranz

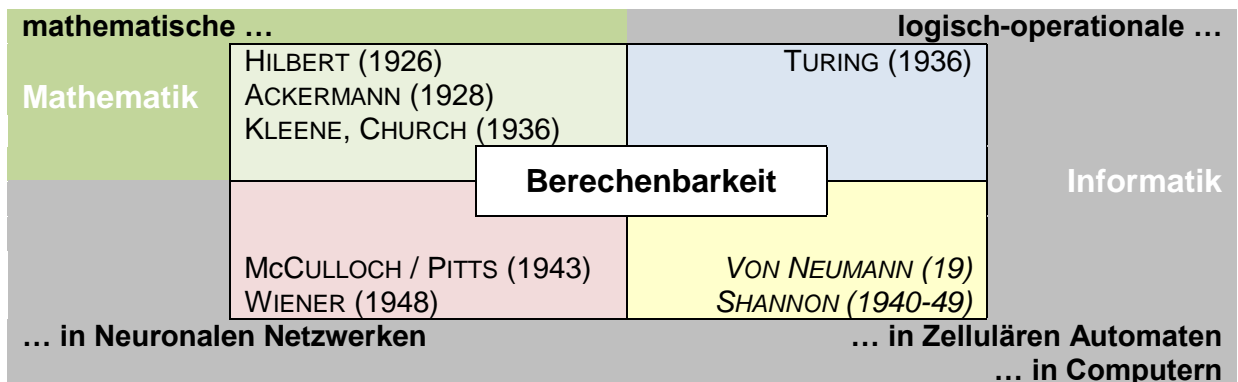
Vergleich: "VON NEUMANN"-Rechner und Künstliche Neuronale Netzwerke

Kriterium / Merkmal / ...	VON NEUMANN-Rechner	Künstliches Neuronales Netzwerk
	EVA(S)-Prinzip	EVA-Prinzip
	eine zentrale Verarbeitungseinheit (1 Leistungs-fähiger Prozessor → CPU)	viele (meist in Schichten angeordnete) eigenständige einfache Verarbeitungseinheiten (Leistungsschwächere Prozessoren → GPU's)
	einzelne Werke getrennt	verschiedene Schichten
	serielle Datenverarbeitung mit immer mehr steigendem Parallel-Anteil	massiv parallele Daten-Verarbeitung serielle Verarbeitung der Daten im Durchlauf durch die Schichten
	Verbindung über zentralen Bus	viele Verbindungen Schicht-orientierte Anordnung der Schaltelemente
	lokalisierbar Daten im Speicher oder Speicherverwerken	Daten / Wissen im Gesamtsystem; Daten nicht lokalisierbar Daten im Netz
	Adress-basierte Daten-Speicherung	assoziative Daten-Speicherung
		hohe Anfälligkeit bei Ausfall einzelner Schaltelemente
	hohe Takt-Raten	
	Daten werden über Adressen angesprochen	nur Zugriff auf Input- und Output-Daten innere Schichten versteckt
		Black Box- Modell
	sehr schnelle serielle Verarbeitung der Daten	
	hohe Rechen-Leistung	
	universelle Programmierbarkeit	
	Problem muss bekannt sein	
	Programme / Programm-Erstellung notwendig	
	Ausgaben vorhersehbar	Daten nicht vorhersehbar
	keine Fehler-Toleranz	hohe Fehler-Toleranz
	keine / sehr begrenzte Lern-Fähigkeit	sehr große - aber thematisch begrenzte - Lern-Fähigkeit
		Training notwendig
	Test notwendig	
	Daten-Zuwachs durch programmierte Aktionen	Wissen-Zuwachs durch Lernen / Training des Netzes
	relativ schnelle Daten-Speicherung	langsamer Lern-Vorgang
	Fehlleistungen lassen sich analytisch aufklären	schwere Aufklärung von Fehlleistungen

praktisch sind Künstliche Neuronale Netze ein alternatives Datenverarbeitungs-Modell zum VON NEUMANN-Modell (VON NEUMANN-Architektur)

Vergleich: klassische Computer-Programme und Künstliche neuronale Netze

Kriterium / Merkmal / ...	klassische Programme	Künstliches Neuronales Netz
	EVA(S)-Prinzip	EVA-Prinzip
	Anwendung nur innerhalb des bekannten Aufgaben-Spektrum	Nutzung auch auf neue / bsi dato unbekannte Daten möglich
	für eindeutige, schnelle Berechnungen funktionieren nur bei Vorhandensein aller Daten	Suche von Mustern Analyse großer Daten-Mengen mit verrauschten / fehlenden / fehlerhaften Daten Aussagen zu vagen oder widersprüchlichen Daten



Was können Computer gut / besser?

- Zahlen verarbeiten
- Wurzel berechnen
- Umsätze ermitteln
- Prozentwerte bilden
- Diagramme aus Zahlen erstellen
- Visualisierungen erstellen
- Übertragen von Daten (Video's, ...)
- Streaming von Musik und Video's
- Befolgen von Regeln (Algorithmen abarbeiten)
- einfache, sich ständig wiederholende, Aufgaben erledigen
- ohne Aufmerksamkeits-Verlust langfristig zu arbeiten
- ...

Was (klassische) Computer weniger / schlechter können?

- Erkennen von Objekten in Bildern
- (inhaltliches) Verstehen von Texten
- Übersetzen von Texten
- Interpretation von Stimmungen (in Bildern oder Texten)
- Vorhersagen von Ergebnissen / Ereignissen / (menschlichen) Handlungen
- ...

Programmieren / programmiertes Lernen

- (Verhaltens-)Regeln / Abfolgen / Strukturen werden vorgegeben
- Regelwerk ist eine Abstraktion (Verallgemeinerung (durch den Ersteller des Regelwerk's))
- im Bedarfs-Fall werden die Regeln / ... angewendet (ev. an Rand-Bedingungen angepasst)
- an den Regeln werden (im Normal-Fall) keine Änderungen vorgenommen; die Regeln bleiben allgemein und langfristig gültig
- unter gleichen Bedingungen erfolgt immer gleiches Reagieren / Abarbeiten des Algorithmus / der Regeln
- Ziel wird erreicht, wenn die Regeln es so festgelegt haben
-

Beispiel: Lernen der Farb-Bedeutung einer Verkehrs-Ampel

Regelwerk wird von Außen vorgegeben (Sinn muss nicht verstanden werden)

Regelwerk muss alle möglichen Farb- (und Verkehrs-)Situationen beinhalten und eine passende Verhaltensweise damit verbinden

Regelwerk kann auf alle anderen Verkehrs-Teilnehmer übertragen werden

neuartige Situationen (z.B. Total-Ausfall der Ampel, sich falsch verhaltende Verkehrs-Teilnehmer, defektes Signal, ...) überfordert das System (im besten Fall reagiert es gar nicht mehr; im schlimmsten Fall reagiert es völlig falsch)

Lernen mit / in einer künstlicher/n Intelligenz / Maschinelles Lernen

- Beobachtung der Situation / Situations-Muster / Strukturen / ...
- Bestimmen eines Ziel's
- Festlegung von Rand-Bedingungen / ev. Belohnungen
- Festlegung einer ersten – unbestimmten – Verhaltensweise (zufälliges Muster, ...)
- ev. zuerst Trainings-Phase
- eigenständige (innere) Ableitung einer Verfahrensweise / Entscheidungshilfe
- "Versuch und Irrtum"-Probieren
- internes Entscheidungs-System basiert auf einer Verknüpfung aller Situations-Merkmale
- später dann Anwendung der eigenen Regel in der Praxis (Anwendungs- / Praxis-Phase)
-

Beispiel: Lernen der Farb-Bedeutung einer Verkehrs-Ampel

System wird im Training mit Farb- und Verkehrs-Situationen konfrontiert
das zu Anfang rein zufällige Handeln wird bewertet
die Bewertung wird zur Veränderung des internen Entscheidungs-Systems verwendet
es entsteht ein individuelles Entscheidungs-System

neuartige Situationen (z.B. Total-Ausfall der Ampel, sich falsch verhaltende Verkehrs-Teilnehmer, defektes Signal, ...) überfordert das System ev.! nur teilweise (das interne Entscheidungs-System kann / muss aber nicht die Situation abfedern)
ev. kann das System mit den Rest-Informationen trotzdem gute Entscheidungen treffen
neue Verkehrs-Teilnehmer müssen Entscheidungs-System selbst erlernen und entwickeln
meist auch eine eigene / individuelle Version

Gegenüberstellung der Programmierungs-Arten (von "Lern-Systemen")

	herkömmliche Programmierung programmiertes "Lernen" / Verhalten	maschinelles Lernen "Programmierung" von Künstlicher Intelligenz
Gemeinsamkeiten	Reagieren auf eine Situation	
Unterschiede	eindeutige(s) Handlungs- Vorschriften / Entscheidungs- System	unklares / meist nicht nachvoll- ziehbares (interne, individuelle) Entscheidungs-System
Verfahren / Inhalt / Methode	WENN-DANN(-SONST)-Regeln	Lernen der "Regeln" / eines sinn- vollen Verhaltens
Vorteile	einfache Realisierung meist reichen wenige Regeln einmalige Analyse / Abstraktion der Situation(en) sofort einsetzbar leichte Anpassung der Regeln einfache Übertragung auf ein anderes System	kann mit komplexen / leicht ver- änderlichen Situationen fertig werden System lernt eigenständig / kann ev. auch weiterlernen
Nachteile	kann nur auf ein Situations-Set reagieren Situation muss vorher abstra- hiert / verallgemeinert werden praktisch kein Weiterlernen möglich	aufwendige Realisierung langwierige vorgelagerte Trai- nings-Phase neue Situationen (äußere Regeln) erfordern neues Lernen (von An- fang an) nur selten Übertragung auf ein anderes System möglich (dieses muss selbst lernen!)

Mit Robotern verbindet man oft die Vorstellung einer menschlich aussehenden Maschine, die zumindestens Mensch-ähnlich agiert und besondere Aufgaben erfüllen soll. Da ist das Tragen von schweren Lasten oder das Arbeiten in gefährlichen Umgebungen. Diese Vision war auch lange in Science-Fiction-Filmen zu sehen. Später gesellten sich dann weniger humanoid wirkende dazu.

Mittlerweile werden z.B. Transport-Roboter oder Stabsauger-Roboter dafür konstruiert, um besonders stupide Tätigkeiten auszuführen, für die der Mensch eigentlich deutlich überqualifiziert ist.

Definition(en): Robotik

Definition(en): Roboter
Roboter sind Maschinen, die weitgehend eigenständig Aufgaben in sich ändernden Umgebungen lösen.

Computer-Bot's sind Software-Roboter, die im Service-Bereich genutzt werden. Sie sollen die Kommunikation von Computern mit den Menschen verbessern / erleichtern.

→ Chat-Bot's

Wann ist ein Programm "intelligent"?

- es handelt sich um die Automatisierung von Dingen, die auf menschlichem Denken basieren
- ermöglicht das Studium der Berechnungen, die Wahrnehmen, Denken und Handeln ermöglichen
- was typischerweise ein Mensch besser als Computer kann
- das Feld, welches intelligentes Verhalten nachahmen kann

klassischer TURING-Test

Mensch / Entscheider sitzt in einem Raum, in einem anderen Raum befindet sich das zu testende System (natürlich alternativ und zufällig auch ein anderer Mensch)

Entscheider führt nun z.B. über Zettel oder ein Chat-System eine Kommunikation mit der zu prüfenden Gegenstelle

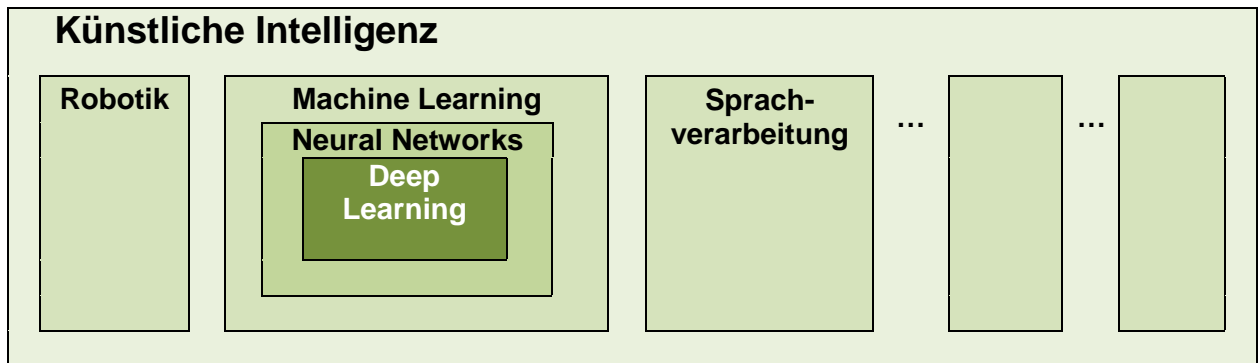
praktisch sind alle Fragen erlaubt

der Entscheider muss am Ende aussagen, ob er mit einem Menschen oder einem Computer / einer künstlichen Intelligenz kommuniziert hat

erweiterte TURING-Test's

- physikalische Aspekte:

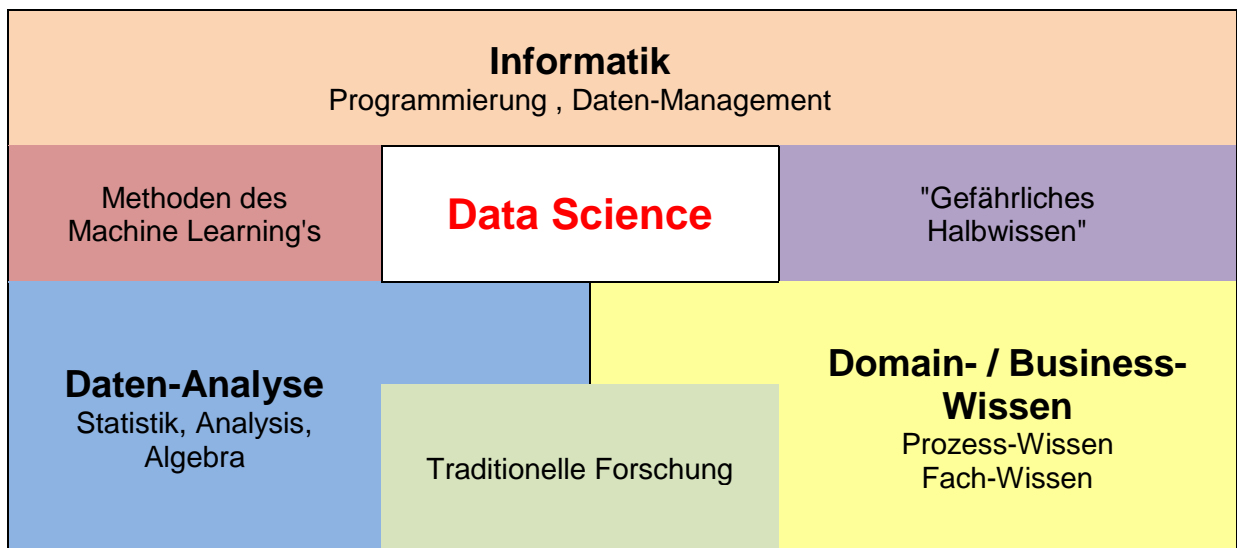
- Bewegung eines Objektes im Raum



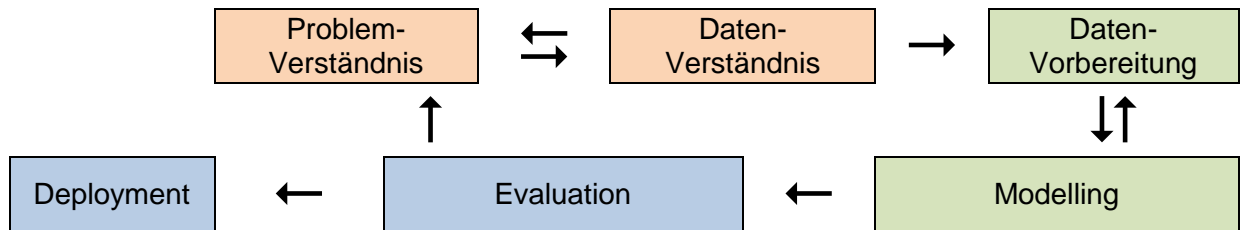
erwarteter / prognostizierter Mehrwert von Künstlicher Intelligenz

- Lösung komplexer Problemstellungen
- automatisches Lernen (ohne menschliche Hilfe in der Trainings-Phase)
- Abbildung der dynamischen Umgebung
- Effizienz-Steigerung / Unterstützung des Menschen
- ...

Verknüpfung von menschlicher und künstlicher Intelligenz bei der sogenannten "Augmented Intelligence" (unterstützende / erweiterte Intelligenz) sozusagen das Einholen einer zweiten Meinung / Unterstützung in gefährlichen Umfeldern / ...



Data Science Prozess ("CRISP-DM")



industrie-übergreifender Standard-Prozess für Data Mining

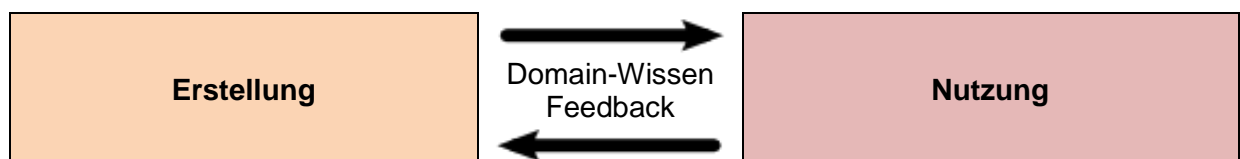
moderne Erweiterung um Verbindung zwischen Deployment (Bereitstellung) und der Daten-Vorbereitung, da sich in der Praxis die Notwendigkeit zu einer Modell-Verwaltung ergeben hat (Versionierung von Modellen und Daten)

Warum gerade heute KI?

- Verfügbarkeit der Daten (Sensoren sind billig zu erhalten, für viele Parameter werden jetzt erst Sensoren angeboten)
- Rechen-Kraft (notwendige Leistung der Prozessoren; Entwicklung von paralleler Daten-Verarbeitung; Graphik-Prozessoren als hochspezialisierte, parallelisierbare, sehr Leistungs-starke Rechen-Einheiten; große Hauptspeicher; große, billige Daten-Speicher)
- neuartige Algorithmen
- zunehmende Automatisierung von Prozessen
- sehr große Vernetzung
- umfangreiche und viel-gestaltige Kommunikation
- sinkende Hardware-Kosten
- Verfügbarkeit von Software (vielfach open source)
- ...

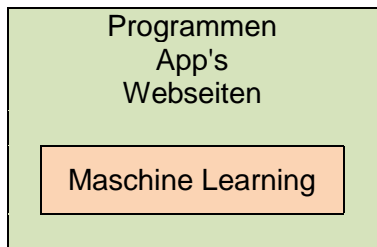
Domain-Experten

Domain-Wissen

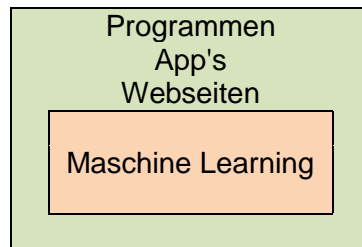


Einsatz-Möglichkeiten von KI

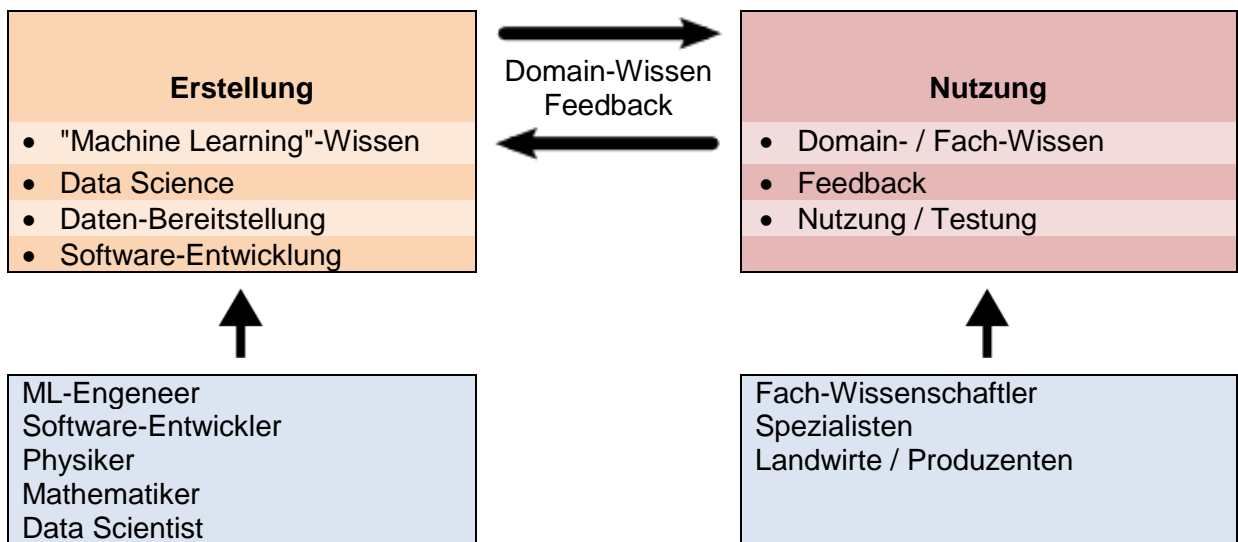
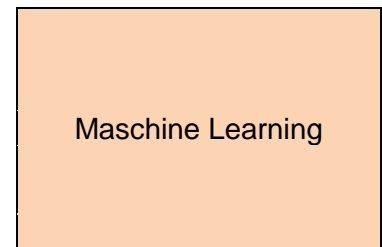
KI als kleiner / neben-sächlicher Bestandteil von ...



KI als großer / dominierender von ...



KI als Forschungs-Thema / -Inhalt



Daten-getriebene Projekte

z.B. für ein Streaming-Portal ein Vorschlag-System (für den Kunden ev. interessierende Filme)

Ziele

- mehr Umsätze
- mehr neue Kunden
- länge Abonnement's
- größere Kunden-Zufriedenheit

oft ist die Daten-Lage nur sehr spärlich (sparse)

wenn die Daten-Lage eher vollständig ist, dann spricht man von dense Daten

	Film A	Film B	Film C
Nutzer 1	**		
Nutzer 2			
Nutzer 3			****
Nutzer 4	****	***	
Nutzer 5			*****
Nutzer 6			
Nutzer 7			
Nutzer 8			

Bewertungen sind polarisierte Daten, besonders die Rand-Werte werden besonders häufig gewählt; mittlerer Bereich eher wenig verwendet

um ein Daten-Verständnis zu entwickeln, werden Parameter definiert / gefunden / festgelegt:
zum **Film**:

- Länge
- Genre
- Titel
- Kurz-Beschreibung
- Teil einer Saga
- ...

zum **Ansehen / Betrachten / Verfolgen des Film's**:

- Zeitdauer
- Abspiel-Gerät
- Bewertung
- Anzahl (Wiederholung, nochmaliges Schauen)
- ...

zum **Nutzer / Kunden**:

- Alter
- Geschlecht
- Sprache
- Interessen
- Alters-Beschränkung
- ...

	Genre	Länge	Kurz-Beschreibung
Film A	Horror	1 h 25 min	"typische Vampir-Geschichte"
Film B	Action	2 h 5 min	Held rettet die Welt
Film C	Comedy	45 min	Sketch-Parade
Film D	Doku	½ Tag	
Film E	Action	1:55 h:m	
Film E	Comedy	1 h 20 min	
Film F	Doku	4560 min	

Roh-Daten

kommen direkt aus einer Quelle und sind noch nicht bearbeitet worden

Probleme:

- doppelte Einträge (Datensätze)
- unrealistische Daten
- fehlende Details
- unterschiedliche Einheiten, Dimensionen, Skalierungen
- unterschiedliche Domänen (Wertebereiche)
- unterschiedliche Sprachen
- ...

deshalb Daten-Vorverarbeitung notwendig
ansonsten ist mit Problemen im resultierenden / abgeleiteten KI-Projekt zu rechnen
zur Vorverarbeitung gehört auch das Aufteilen der Daten in Trainings- und Test-Daten
(Train-Test-Split)

Definition(en): Roh-Daten

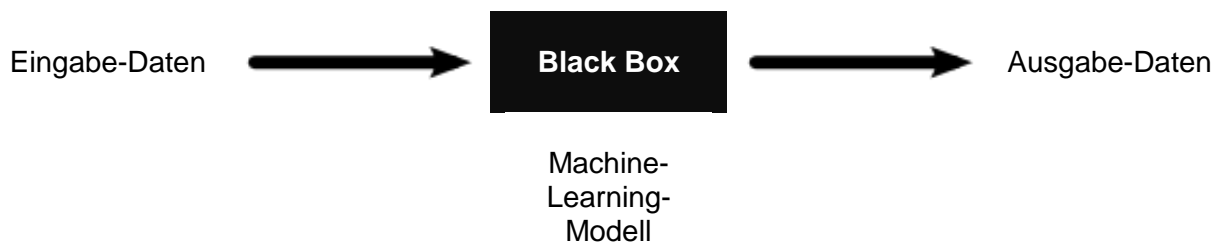
Rohdaten sind solche Daten, die praktisch direkt aus einer Quelle stammen und praktisch unbearbeitet sind.

Definition(en): Eingabe-Daten

Definition(en): Ziel-Daten

Definition(en): Prognose-Daten

Black-Box-Modell



Wo steckt in der heutigen Technik und in unserer Lebenswelt (schon) KI drin?

- Porträt-Erkennung in Digital-Kamera's und Smartphone's
- Fahrer-Assistenz-Systeme
- Sprach-Assisten (Alexa, Siri, ...)

Sind die folgenden Alternativ-Antworten von einer Person oder von einem Sprach-Assistenten?

KI antwortet eher neutral und bei konkreten Fragen eher ausweichend und / oder völlig unerwartet

1.2. Erfolge / Meilensteine von KI

1950: Alan TURING formuliert die heute TURING-Test genannte Versuchs-Anordnung zum Unterscheiden von menschlicher und künstlicher Intelligenz

1952 Tic-Tac-Toe als erstes Computer-Spiel umgesetzt

1960er: erste Chat-Bot's

2011: Watson von IBM
tritt gegen menschliche Konkurrenten im Fernsehspiel "Jeopardy!" an und gewinnt deutlich gegen die zwei Final-Gegner
das System musste dabei in natürlicher Sprache vorgetragene Antworten zu einer Frage umformulieren (umgedrehtes Quiz)

2015: AlphaZero von DeepMind Inc.
hat die Atari-Konsolen-Spiele gelernt
System erlernte die Spiel-Algorithmen und entwickelte daraus eine Spiel-Strategie
bei 23 von 49 Spielen konnte das System die Highscore's der besten menschlichen Spieler übertreffen

2015: AlphaGo von DeepMind Inc.
kann das japanische Brettspiel Go so gut, dass es den mehrfachen Europameister Fan HUI und den zweit-besten menschlichen Spieler Lee SEDOL schlagen unter Turnier-Bedingungen konnte

2017: AlphaZero von DeepMind Inc.
erlernte innerhalb von 9 Stunden über die Analyse des (sehr guten / stärksten) Schach-Programm's Stockfish 8 das Spiel

Sprach-Assistenten (google-assistent, alexa, ...)

heutige klassische Anwendungen (oft quasi im Hintergrund arbeitend)

- Spam-Filter
- Übersetzungs-App's oder –Funktionen (z.B. in chrome-Browser; deepl.com; ...)
- Bewertung von Versicherungs-Nehmern
- Betrugs-Erkennung bei Kreditkarten-Zahlungen
- Bild-Erkennung (medizinische Bilder → Krebs-Erkennung)
- Patienten-Überwachung (z.B. auf Intensiv-Stationen)
- Prozess-Überwachung in der Industrie und chemischen Produktion

-
- Gesichts-Erkennung
 - ...

2021 ChatGPT und andere Text-KI's produzieren Texte und Bilder aus Begriffen

2022/23 ChatGPT und andere kreative KI stehen der breiten Öffentlichkeit zur Verfügung

2. Grundlagen

2.0. Logik, Mathematik, ...

Dies ist kein Mathe-Skript. Die angedeuteten logischen und mathematischen Sachverhalte werden hier aus der Anwendungs-Sicht in der Künstlichen Intelligenz betrachtet.

2.0.1. Prädikaten-Logik

2.0.2. Fuzzy-Logik

In den späteren Anfangs-Jahren war die Fuzzy-Logik – oder auch Ünschärfe-Logik, wie sie auch genannt wird – der große Stern am KI-Himmel.

Die Erfahrung hat aber gezeigt, dass die Fuzzy-Logik meist nicht der optimalste Weg ist. Wie effektiver klappt vieles mit Wahrscheinlichkeiten.

2.0.3. Stochastik – Wahrscheinlichkeits-Rechnung

Wahrscheinlichkeit

Chancen

Beim Würfeln hat man eine Chance von 1 : 5 eine 6 zu würfeln. Statistisch gesehen gelingt es 1x und 5x nicht. Diese wird manchmal auch als Quote bezeichnet.

Die Wahrscheinlichkeit eine 6 zu würfeln ist $1 : 6 = 0,1666$. Dies entspricht 16,7 %.

Unsicherheit

Satz von BAYES ()

bezieht sich auf Chancen vor ("a priori") dem Eintreffen / Bekanntwerden neuer Sachverhalte die neuen Informationen werden dann benutzt, um die "a posteriori"-Chance zu bestimmen.

letztendlich geht es darum den Such-Raum durch Vorwissen einzuschränken ohne das Vorwissen sind die "a priori"-Chancen relativ klein durch die Einbeziehung des zusätzlichen Wissen's sehen die "a posteriori"-Chancen deutlich besser aus

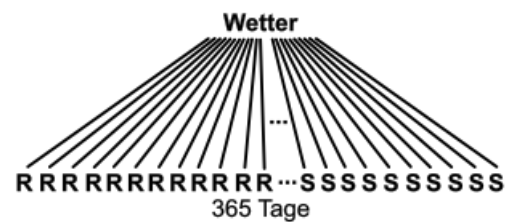
Beispiel: Wetter

An einem Ort ist die Chance, dass es regnet angenommen 200 : 165 – also es regnet an 200 Tagen und an 165 Tagen regnet es nicht. Insgesamt betrachten wir genau 365 Tage (200+165). Mit diesem Wissen gehen wir abends ins Bett. Die Chance, dass man am nächsten Tag Regen bekommt ist also recht hoch – keine schöne Aussicht für den nächsten Tag.

Nach dem Aufwachen schaut man kurz nach draußen und stellt fest, es ist bewölkt. Nun ist auch bekannt, dass die Chance, dass es am Tag mit einem bewölkten Morgen regnet, bei 5 : 1 liegt.



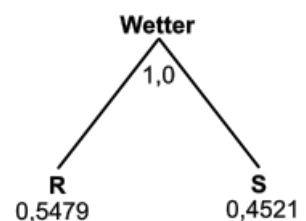
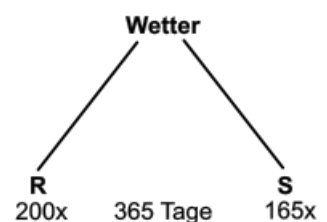
R ... Regen S ... Sonne (kein Regen)



R ... Regen S ... Sonne (kein Regen)

Als Wahrscheinlichkeit ausgedrückt ist eine Wahrscheinlichkeit für einen regnerischen Tag zuerst einmal: $200/365 = 0,5479$. Die Wahrscheinlichkeit für einen Tag ohne Regen liegt bei: $165/365 = 0,4521 = 1 - 0,5479$.

Betrachtet man nun die Bedingung der morgentlichen Wolken-Beobachtung, dann stellen wir fest, dass die Wahrscheinlichkeit für Regen an einem bewölkten Morgen bei: $5/6 = 0,8333$ liegt. Die Wahrscheinlichkeit für Regen nach einem Wolken-losen Himmel liegt entsprechend bei: $1/6 = 0,1667 = 1 - 0,8333$.



Aufgaben:

1. Stellen Sie für die morgendliche Wetter-Beobachtung und der darin enthaltenen Regen-Häufigkeit vergleichbare Diagramme (wie oben) auf!
2. Würfeln Sie 10x und stellen Sie ein passendes Set an Diagrammen für dieses Experiment auf! Wir betrachten nur die Ereignisse "6" und "nicht 6" gewürfelt.
3. Würfeln Sie 30x und stellen Sie ein passendes Diagramm-Set für alle Würfel-Ergebnisse und eine geeignete Zusammenfassung zusammen!

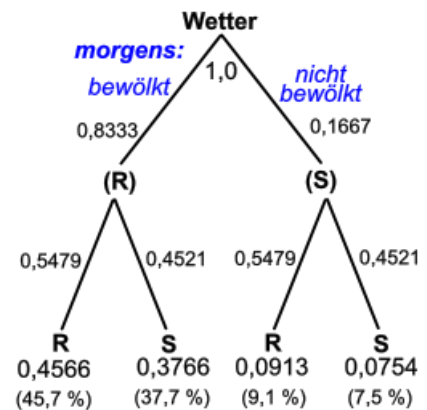
Kombiniert man die Wahrscheinlichkeiten, dann kommt man zu den folgenden "verbesserten" Wetter-Vorhersagen:

Insgesamt gibt es nun eine Serie von Wahrscheinlichkeiten für jede Situation.

Die Verhältnisse von bestimmten Ereignissen zueinander werden auch Wahrscheinlichkeits-Verhältnisse genannt.

In unserem Fall ist das Wahrscheinlichkeits-Verhältnis für ein ein Regen-Ereignis bei:

$$WV = \frac{\frac{200}{365}}{\frac{165}{365}} = \frac{0,5479}{0,4521} = 1,212$$



Das können wir auch in einem einfachen Baum notieren (s.oben).

Die Aussage entspricht der a-priori-Wahrscheinlichkeit. Bezieht man nun als Vorwissen noch die morgentliche Wolken-Beobachtung mit ein, dann ergeben sich für die 4 möglichen Konstellationen die zuerst einmal ein weiter verzweigter Baum ergeben. Unten stehen dann die Einzel-Wahrscheinlichkeiten für die Konstellationen.

Praktisch werden die Wahrscheinlichkeiten immer multipliziert und müssen natürlich in der Gesamtsumme wieder genau 1 ergeben. (Hier wegen der Rundungen nicht ganz erreicht!)

Allgemein kann man auch sagen:

$$aPrioriChance = WV \cdot aPrioriChance$$

$$\Pr\left(\frac{E}{N}\right) = \frac{\Pr\left(\frac{N}{E}\right) \cdot \Pr(E)}{\Pr\left(\frac{N}{E}\right) \cdot \Pr(E) + \Pr\left(\frac{N}{\sim E}\right) \cdot \Pr(\sim E)} \quad \text{Satz von BAYES mit Wahrscheinlichkeiten}$$

Algorithmus zur Berechnung der a-posteriori-Chance

1. Berechnen der a-priori-Chance
2. Berechnen der Wahrscheinlichkeit für ein Ereignis
3. Berechnen der Wahrscheinlichkeit für das Nicht-Ereignis
4. Berechnen des Wahrscheinlichkeit-Verhältnisses als Verhältnis der Ereignis-Wahrscheinlichkeit und der Nicht-Ereignis-Wahrscheinlichkeit
5. Multiplizieren der a-priori-Chance mit dem Wahrscheinlichkeits-Verhältnis → Ergebnis
6. STOP

Exkurs: BAYES für Höhlen-Menschen

nach Q: <https://betterexplained.com/articles/understanding-bayes-theorem-with-ratios/>

Der Jäger Og hat bei seinen Erkundungen die Beobachtungen immer schön auf eine Felstafel geritzt. Dadurch hat er festgestellt, dass:

- er insgesamt 50 Hirsche und 5 Bären beobachtet hat
- nachts sah er davon 10 Hirsche und 4 Bären

Damit ergeben sich für zufällige Treffen im Wald ein Chance von 50 : 5 (also 10 : 1) für den gesuchten Hirsch. Bären jagen ist eher zu gefährlich – ein Zusammentreffen sollte also vermieden werden.

Für die Nacht ergibt eine noch ungünstigere Chance von 10 : 4 (also 5 : 2).

Aus seinen Beobachtungen (Strich-Listen) kann er ebenfalls ableiten, dass Hirsche nachts mit einer Wahrscheinlichkeit von $\frac{1}{4}$ auftreten. Da hat er sich so:

$$\frac{\frac{\text{Hirsche}_{\text{nachts}}}{\text{Bären}_{\text{nachts}}}}{\frac{\text{Hirsche}_{\text{insgesamt}}}{\text{Bären}_{\text{insgesamt}}}} = \frac{\frac{10}{4}}{\frac{50}{5}} = \frac{2,5}{10} = 0,25 = \frac{1}{4}$$

berechnet. Dies bedeutet, dass nachts Bären 4x wahrscheinlicher sind. Von einer Jagd nachts rät Og seinem Stamm also dringend ab.

Mit diesen Ergebnissen gibt sich Og allerdings nicht zufrieden und forscht und notiert weiter. Dabei stellt er fest, dass am Fluss auf 2 Hirsche 4 Bären vorkamen.

In den Winter-Monaten stellte er zudem fest, dass jetzt auf 30 beobachtete Hirsche nur ein Bär kam.

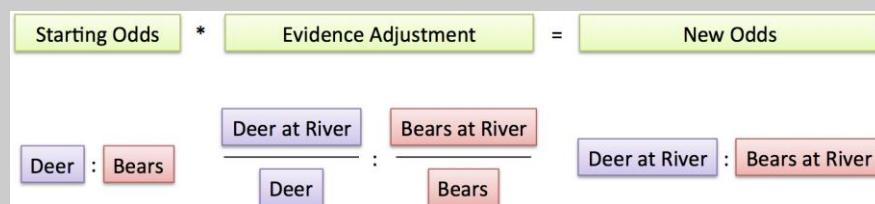
Aus den Beobachtungen berechnet er jetzt eine 20x höhere Wahrscheinlichkeit für Bären am Fluss (als im Wald). Im Winter sind die Hirsche 3x so wahrscheinlich (als im Sommer). Gemeint ist natürlich immer die Wahrscheinlichkeit bezüglich eines fälligen Treffen irgendeines der Tiere.

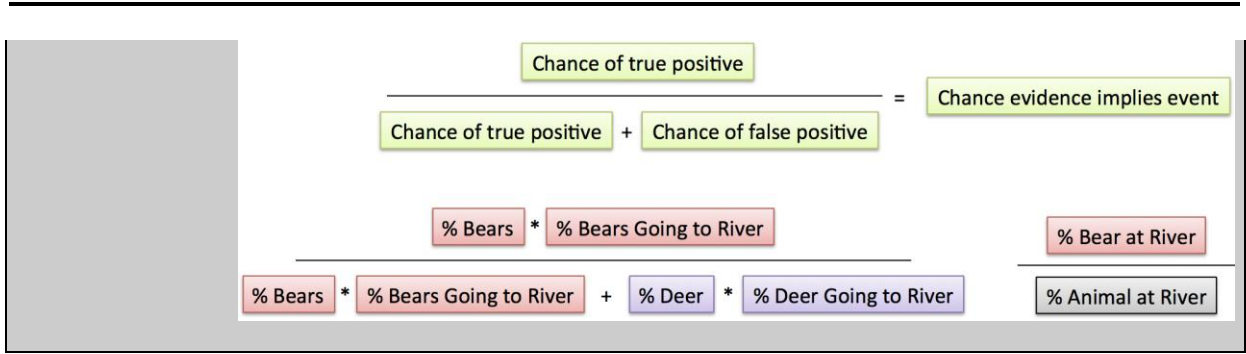
Bei einem Lagerfeuer und einem großen Krug Met hat Og seine Geheimnisse mit dem verwandten Stamm-Obersten Ono geteilt. Der lebt in einem benachbarten Wald mit Fluss und hat für sich bisher nur festgestellt, dass die Hirsche und Bären im Verhältnis 25 : 1 von ihm angetroffen wurden.

Zurück in seinem nächtlichen Winter-Wald hört er am Fluss ein Tier auf sich zukommen. Mit den Og-Erkenntnissen leitet er nun ab:

- nachts sind die Bären 4x wahrscheinlicher → Chance also 25 : 4 (sonst ja 25 : 1)
- am Fluss sind die Bären 20x wahrscheinlicher → 25 : 80
- und da Winter ist, ist die Chance für Hirsche 3 x größer → 75 : 80

Daraus ergibt sich für ihn eine annähernde 50 zu 50 Chance für das treffen einen Hirsches oder eines Bären. Da macht er sich lieber ganz leise aus dem Staub.





Aufgaben:

1. Überlegen Sie sich, wie groß die Chance für eine 6 beim Würfeln ist, wenn Sie vorher eine 6 hatten!
2. Ein Paar hat es sich zum Ziel gesetzt, mindestens 1 Mädchen und 1 Junge als Kinder zu haben. Wie groß ist die Chance als zweites Kind ein Mädchen zu bekommen, wenn das erste Kind ein Junge war? Wie groß ist die Chance für eine Serie von: Mädchen-Junge?
3. Prüfen Sie, ob eine Beobachtung von 50 Hirschen und 5 Bären genauso aussagekräftig ist, wie die noch langfristige von 500 zu 50! Testen Sie dann auch die Nacht-Verhältnisse 10 : 4 gegen 100 : 40!
4. Erstellen Sie einen Spam-Filter für eine beobachtete Schlüsselwort-Anzahl (s. Tab.)!
5. In der Bevölkerung liegt die Chance an Krebs zu erkranken bei 2 %! Wenn getestet wird, dann werden von allen Krebs-Erkrankten 80 % auch positiv getestet. Bei 20 % schlägt der Test nicht an oder erfolgte zu früh. Bei Gesunden kommt es mit rund 9,5 % zu einem Fehl-Alarm. Wie groß ist die Wahrscheinlichkeit für Krebs, wenn man einen Positiv-Test erhält?

Schlüsselwort	Spam	"normale" Mail
hallo	3	3
liebling	1	5
kaufen	3	2
viagra	3	0
...

Exkurs: BAYES bei medizinischen Test's

nach Q: <https://course.elementsofai.com/de/3/2>

Gesucht wird der Prävalenz-Fehler (Basis-Raten-Fehler)

Gehen wir von vereinfacht 5 betroffenen Personen in einer Gruppe von insgesamt 100 aus. Die Erkrankten zu finden ist nicht immer leicht. Aus der a-priori-Datenlage haben wir eine Chance von 5 : 95 (entspricht 5 von 100) eine Erkrankten zu erkennen. Also muss ein Labor-Test her. Er soll die Chance einen Erkrankten zu finden, erhöhen.



Der Labor-Test, der benutzt wird hat eine Empfindlichkeit von 80 %, d.h. 4 von 5 Fälle werden gefunden. Eine Person fällt beim Test durch, dessen Erkrankung wird nicht gefunden. Leider schlägt der Test auch bei einigen gesunden Personen an. Bei 10 von 100 Personen wird also eine Erkrankung angezeigt, obwohl gar keine vorliegt ("Falsch-Positiv-Ergebnis").

Bei einem so hohen Falsch-Ergebnis-Anteil horchen wir intuitiv auf und zweifeln die Qualität des Test's an. Aber bringt er trotzdem was für das Finden der echten Erkrankungsfälle?



$$\frac{5}{95} \cdot \frac{80}{100} = \frac{5}{95} \cdot \frac{8}{10} = \frac{40}{95}$$

In Prozent ausgedrückt ergibt sich Erkennungs-Wahrscheinlichkeit von knapp 30 % (a-posteriori).

Dazu wird die Ergebnis-Häufigkeit durch die insgesamt zu betrachtenden Fälle von 135 (= 40 + 95) geteilt.

Das ist deutlich besser als die 5 % ohne Labor-Test.

Auch für die – vor dem Test – versehentlich als Kranke vermutete 95 % sinkt jetzt die Rate auf

Exkurs: 4-Felder-Tafel

dient der Darstellung von bedingten Wahrscheinlichkeiten

Die klassische 4-Felder-Tafel besteht aus einem geviertelten Quadrat. Horizontal und vertikal teilt sich das Quadrat jeweils in die zwei möglichen Aussagen. Das Quadrat selbst enthält alle Aussagen des Ergebnis-Raum's Ω . In der nebenstehenden Abbildung ist dies die dunkelgraue Fläche.

Wenn der Ergebnis-Raum z.B. alle zu würfelnden Zahlen enthält, dann schreibt man dies so:

$$W = \{1,2,3,4,5,6\}$$

Die Ergebnisse könnten z.B. hinsichtlich ihrer Eigenschaft als gerade Zahl klassifiziert werden. Bei A würden wir also die geraden Ergebnisse (Zahlen, Würfel-Werte) betrachten, bei $\neg A$ (NICHT-A) die nicht-geraden (also ungeraden) Ergebnisse.

	B	$\neg B$
A	Ω	
$\neg A$		
A	2	4 6

$A = \{2,4,6\}$ $\neg A = \{1,3,5\}$

Eine weitere Klassifizierung (B) ergibt z.B. die folgenden Zugehörigkeiten:

$B = \{2,3,4\}$ sowie entsprechend $\neg B = \{1,5,6\}$

Die Frage ist nun, welche der Ergebnisse gehören zu der Gruppe, für die sowohl A als B gilt – also:

$A \cap B$

Gesucht sind somit die geraden Zahlen (Klasse A) aus der der Klasse B. Das ist sind die Ergebnisse 2 und 4. Mathematisch notiert als:

$A \cap B = \{2,4\}$

In der 4-Felder-Tafel belegen wir das obere linke Feld. Nach diesem Prinzip ordnen wir nun auch die anderen Zahlen zu. Die gesamte Verteilung der Ergebnis-Menge W sieht dann so aus. In die Felder ist zusätzlich noch **rot** die Verknüpfung der Bedingungen eingetragen. Für Kontroll-Zwecke kann man auch Zeilen- und Spalten-Summen (der Ergebnis-Anzahlen) bilden. Die Summe der Spalten-Summen muss dann gleich der Summe der Zeilen-Summen sein. Dies ist die Gesamt-Summe oder der Umfang der Ergebnis-Menge. Die 4-Felder-Tafel lässt sich auch mit absoluten oder relativen Anzahlen oder Prozent-Zahlen aufstellen. Bei relativen Werten muss als Gesamt-Summe eine 1 (wie in der nebenstehenden Wahrscheinlichkeiten-Tabelle) und bei Prozent-Zahlen die 100 % herauskommen. Eine weitere Möglichkeit, bedingte Wahrscheinlichkeiten graphisch zu verdeutlichen sind **Baum-Diagramme**. Sie funktionieren mit Wahrscheinlichkeiten W, absoluten (An-)Zahlen oder Prozent-Zahlen besser als mit Mengen.

$\neg A$		1	3	5
	B			
		2		1
		3		5
		4		6

	B	$\neg B$
A	2 4 $A \cap B$	6 $A \cap \neg B$
$\neg A$	3 $\neg A \cap B$	1 5 $\neg A \cap \neg B$

	B	$\neg B$	Σ
A	$A \cap B$	$A \cap \neg B$	$W(A)$
$\neg A$	$\neg A \cap B$	$\neg A \cap \neg B$	$W(\neg A)$
Σ	$W(B)$	$W(\neg B)$	$W(\Omega)=1$

Aufgaben:

1. Stellen Sie die 4-Felder-Tafel für die Klassifizierung der Zahlen 1 bis 10 nach "ungerade" und "prim" auf!
2. Gesucht ist die 4-Felder-Tafel für die Klassifizierung alle Buchstaben (Groß- und Klein-Buchstaben) nach den Merkmalen Vokal (Merkmal A) sowie nach Gross-Buchstabe (Merkmal B). Welche Buchstaben sind sowohl Groß-Buchstaben als auch Vokale?
3. Erstellen Sie 4-Felder-Lösung für die Frage, welcher Junge in der Kursgruppe größer als 1,7 m ist!
4. Bei einer Befragung von 607 Personen gaben 138 an, dass sie weiblich sind. 262 von den Befragten sind in einem Sport-Verein organisiert und 385

machen gerne Ball-Sport. Machen Sie statistische Aussagen zu den folgenden Fragen! Nutzen Sie dazu passende 4-Felder-Tafeln!

a) Wieviele Mädchen spielen gerne Ball?

b) Wieviele Jungen sind in einem Verein organisiert?

c) Wieviele weibliche Vereins-Mitglieder spielen gerne Ball?

der naive BAYES-Klassifikator

dient der Klassifizierung von Objekten (Texten, Gegenständen, ...) in 2 oder mehr Klassen ist anlernbar

z.B. Unterscheidung von eMail's in Spam und Ham (erwünschte Mail's)

naiv bezieht sich hierbei auf das Vorgehen, das z.B. in texten Wort für Wort vorgegangen und entschieden wird

damit entfallen z.B. die Beachtung von Zusammenhängen zwischen den aufeinanderfolgenden Worten; auch die reihenfolge spielt dann keine Rolle mehr

Zuerst geht man von einer a-priori-Chance von 1 : 1 zwischen Ham und Spam aus.

jetzt werden für jedes Wort die Wahrscheinlichkeiten für die Zugehörigkeit zu Spam und Ham berechnet

dazu benutzt man eine Lern-Phase oder annotierte Daten

Wort	Spam	Ham
Million	156	98
Dollar	29	119
AdClick	51	0
Konferenzen	0	12
Gesamt	95791	306438

Berechnen wird nun als erstes die jeweilige Wahrscheinlichkeit – also die Auftritts.Anzahl geteilt durch die Gesamtzahl (in der Klasse).

Wort	Wahrscheinlichkeit		
	Spam	Ham	Verhältnis
Million			
Dollar			
AdClick			
Konferenzen			

Exkurs: Baum-Diagramme

Baum-Diagramme gehören zu den Klassikern in der Wahrscheinlichkeits-Rechnung (Stochastik) in der Schul-Mathematik

dienen der Darstellung von abhängigen Wahrscheinlichkeiten

Die klassische Frage-Stellung für die Erklärung von Baum-Diagrammen ist z.B. die Frage nach der Häufigkeit von 2 aufeinander folgenden gewürfelten 6-en.

Dazu müssen wir zuerst einmal das 1. Mal würfeln. Hier ist die Wahrscheinlichkeit eine 6 zu würfeln bei 1 aus 6 Würfeln. Mit anderen Worten $1/6$ oder $0,166667$ oder $16,6667\%$.

Die restlichen 5 von 6 Möglichkeiten ergeben keine 6. Die Häufigkeit beträgt somit $5/6$ oder $0,833333$ oder $83,3333\%$.

Dies stellen wir mit einem Verzweigungs-System dar.

Die verschiedenen Möglichkeiten werden durch Zweige beschrieben.

Normalerweise könnte man für jede der möglichen Ergebnisse einen eigenen ($1/6$ -)Zweig zeichnen. Die verschiedenen Zweige für die Würfe mit 1 bis 5 Punkten interessieren uns für diese Aufgabe aber nicht. Wir fassen sie zu einem ($5/6$ -)Zweig zusammen.

Folgt ein 2. Wurf mit dem Würfel, dann interessieren uns wieder die Wahrscheinlichkeiten. Diese sind natürlich genau gross, wie bei 1. Wurf. D.h. wir hängen an die Zweige vom 1. Wurf die, vom 2. Wurf an.

Das ganze erinnert an einen (umgelegten) Baum.

Manchmal werden die Bäume auch von oben nach unten gezeichnet. Dann entsteht ein Bild, dass eher an ein Wurzel-System erinnert.

Uns interessiert – laut Aufgabe – aber nur ein Pfad aus unserem Baum, nämlich der Weg, der zu zwei 6en hintereinander führt.

Im nebenstehenden Baum ist dieser Pfad **rot** gekennzeichnet.

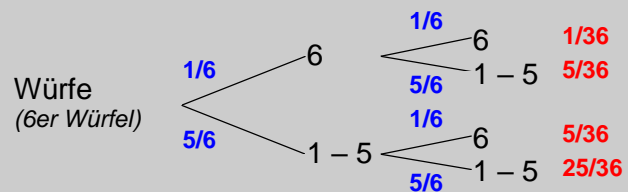
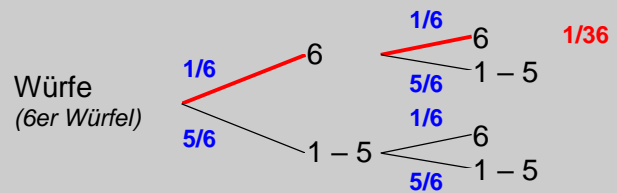
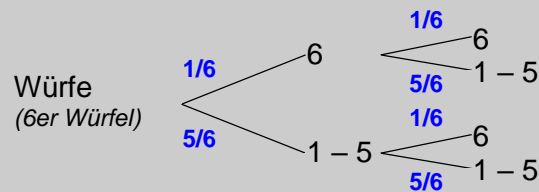
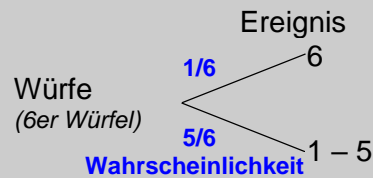
Die beiden Wahrscheinlichkeiten müssen nun multipliziert werden. Das ergibt $1/36$. Mit anderen Worten, nur einmal pro 36 Wurf-Serien mit 2 Würfeln erzielt man zwei 6en hintereinander.

Pro forma können wir auch die anderen Pfad-Wahrscheinlichkeiten berechnen.

Als Kontrolle können wir nun alle Pfad-Wahrscheinlichkeiten addieren und müssen den vollen Umfang des System's erhalten. Das sind $36/36$ Würfe oder eine Wahrscheinlichkeit von 1 oder 100% .

Die bedingte oder abhängige Wahrscheinlichkeit wird durch das Produkt der Einzel-Wahrscheinlichkeiten bestimmt (Produkt-Regel).

Sucht man andere Gruppen oder Ereignisse als Ziel, dann kann man ebenfalls Baum-Diagramme nutzen.

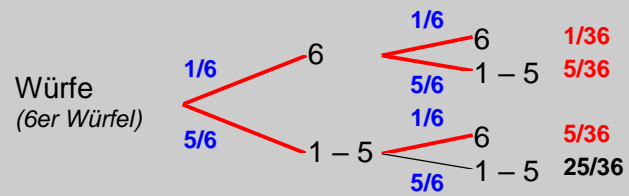


Z.B. könnte uns interessieren, wie häufig man mindestens eine 6 bei einem Doppelwurf (Wurf-Serie a 2 Würfeln) erreicht.

Die gültigen Pfade sind wieder rot eingefärbt.

Die gültigen Pfad-Wahrscheinlichkeiten müssen nun addiert werden (Summen-Regel).

Das ergibt $11/36$ Doppelwürfe (oder Wurf-Serien a 2 Würfeln) bei denen mindestens eine 6 gewürfelt wird.



Aufgaben:

1. Erstellen Sie ein vollständiges Baum-Diagramm für zwei aufeinanderfolgende 6er-Würfel-Würfe! Notieren Sie die Zweig-Wahrscheinlichkeiten (in einer Farbe) und die Pfad-Wahrscheinlichkeiten (in einer anderen Farbe)!
2. Kennzeichnen Sie die Pfade, die das Ereignis "mindestens einmal eine 2" beschreiben! Geben Sie die Wahrscheinlichkeit für das gesuchte Ereignis an!
3. Kennzeichnen Sie (ev. in einem neuen Baum-Diagramm) alle die Fälle, bei denen im 2. Wurf eine kleinere oder die gleiche Augenzahl erreicht wird! Schätzen Sie vorher ab, ob es sich lohnt, um Geld zu spielen! Bestimmen Sie dann die Gesamt-Wahrscheinlichkeit für die gesuchten Ereignisse!
4. Wie würde sich die Gewinn-Chance verändern, wenn das Würfeln schon von einer 1 ein verlorenes Spiel bedeutet?

2.0.4. Korrelation und Regression

Korrelation

Definition(en): Korrelation

Eine Korrelation ist die Wechselbeziehung zwischen zwei oder mehr Merkmalen, Funktionen und / oder Zuständen.

(Zwischen den Daten(-Reihen) muss kein ursächlicher (kausaler) Zusammenhang bestehen! Es gibt verschiedene (Zusammenhangs-)Maße zum Bestimmen der (mathematischen) Stärke des Zusammenhang's.)

Die Korrelation ist ein statistisches Analyse-Verfahren, um den qualitativen Zusammenhang zwischen zwei oder mehr Größen (Werten) zu bestimmen.

(Es geht darum die Güte / die Qualität des Zusammenhangs quantitativ zu bewerten.)

Daten-Reihen können immer korrelieren, das kann leicht mathematisch geprüft werden. Ob es aber einen ursächlichen (kausalen) Zusammenhang zwischen ihnen gibt, muss unbedingt extra geprüft werden. Dies bedarf meist auch zusätzlicher Modelle.

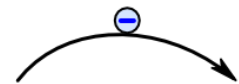
die meisten Zusammenhangs-Maße werden mit Zahlen-Werten zwischen 0 und 1 beschrieben (Stärke des Zusammenhang's), wobei 0 kein Zusammenhang bedeutet. Ein Maß von 1 steht für einen perfekten – 100%igen – Zusammenhang. In Fluß-Diagrammen / -Schemata wird eine Korrelation durch eine positive Kopplung dargestellt.



Die Umschreibung lautet "Je mehr **Ursache**, desdo mehr **Wirkung**." bzw. "Je weniger **Ursache**, desdo weniger **Wirkung**".

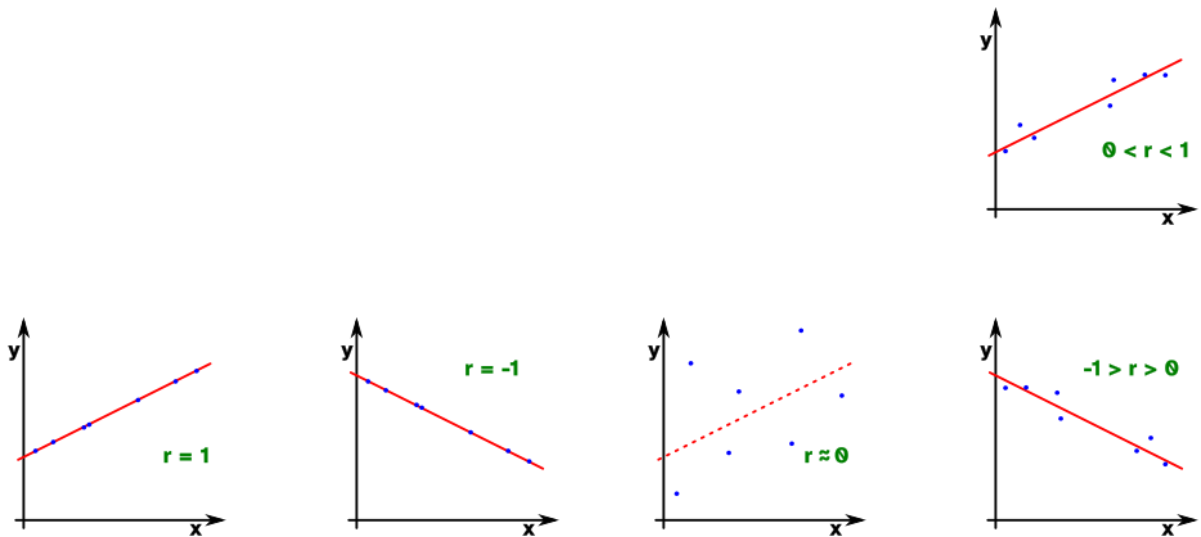
Bei der Anti-Korrelation hat das Korrelations-Maß ein negatives Vorzeichen (Richtung des Zusammenhang's).

Hier lautet die Umschreibung "Je mehr **Ursache**, desdo weniger **Wirkung**." bzw. "Je weniger **Ursache**, desdo mehr **Wirkung**".



Meist wird der BRAVAIS-PEARSON-Korrelations-Koeffizient als Zusammenhangs-Maß benutzt. Er wird vielfach nur **Korrelations-Koeffizient** genannt.

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}}$$



Aufgaben:

1. Überlegen Sie sich, welche Ausdrücke Sie zur Berechnung des Korrelations-Koeffizienten (Schritt für Schritt) ermitteln müssen, wenn Sie obige Formel umsetzen wollen! (Ausgangspunkt ist eine Liste von Werte-Paaren) (Schüler aus einem Informatik- oder Programmier-Kurs entwickeln ein Struktogramm oder nutzen eine andere Algorithmen-Veranschaulichung!)
2. Informieren Sie sich, inwieweit Ihr Taschen-Rechner Teile oder die gesamte Berechnung übernimmt! Welche Tasten(-Folgen) müssen Sie nutzen?

für die gehobene Anspruchsebene:

3. Erstellen Sie ein einfaches Programm, mit dessen Hilfe der Korrelations-Koeffizient für eine Liste von (ev. einzugebenen) Werte-Paaren berechnet wird!

Regression

Suche nach einer Beziehung zwischen einer erklärten Variable (in der Schule meist y) und einer oder mehrerer erklärenden Variablen (meist x oder a, b, c, \dots)

die erklärte (abhängige) Variable ist der **Regressand**, die erklärenden (veränderlichen) Variablen werden **Regressoren** (Einfluss-Größen) genannt
praktisch soll eine Prognose(-Funktion) erstellt / berechnet werden, mit der später andere Werte des Regressanden aus bekannten Regressor(en)-Werten ermittelt werden kann

Definition(en): Regression

Die Regression ist ein statistisches Analyse-Verfahren, um den quantitativen Zusammenhang zwischen zwei oder mehr Größen (Werten) zu bestimmen. (Es geht darum den funktionellen Zusammenhang zu ermitteln.)

Die Regression ist die quantitative (funktionelle) Beschreibung einer abhängigen / endogenen Variable aus einer unabhängigen / erklärenden / exogenen.

recht einfaches Regressions-Verfahren ist die lineare Regression

sie kann mit einigen mathematischen Kniffen auch zur Analyse von

- reziproken
- exponentiellen

und

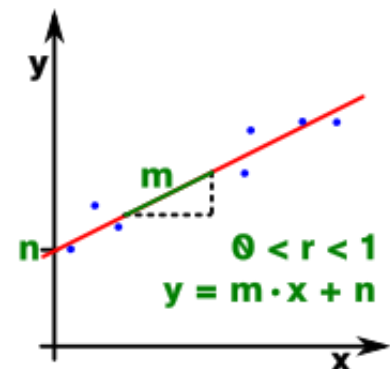
- ?

Zusammenhängen genutzt werden.

lineare Regression

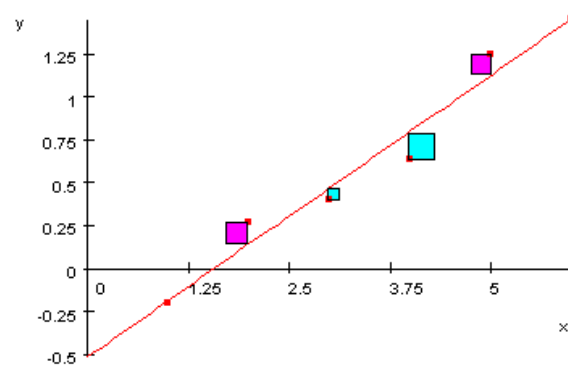
Nutzung der linearen Regression für weitere Arten von Zusammenhängen möglich

ist selbst relativ einfaches Verfahren, deshalb versucht man, andere (nicht-lineare) Zusammenhänge durch mathematische Kniffe auf lineare Beziehungen zurückzuführen



für schulische Betrachtungen bleiben wir bei einem Regressor – also einer erklärenden / variablen Größe. Das ist in der Mathematik typischerweise unser x .

lineare Regression versucht eine (Ausgleichs-)Gerade zwischen die Werte-Paare (x,y) zu legen als Maß werden die Abstände zwischen den Punkten und der Gerade so minimiert das die Quadrate der Abstände möglichst gering sind



Ausgleichs-Gerade mit Fehler-Quadraten

Regression liefert uns den Anstieg der Ausgleichs-Gerade (oft mit m abgekürzt) und den Schnittpunkt auf der y -Achse (Ordinate) (oft mit n abgekürzt)

der Schnitt-Punkt auf der Ordinate wird oft auch als y-Achsen-Abschnitt bezeichnet

$$y = m \cdot x + n$$

ev. auch: $y = a + x \cdot b$

Die beiden Konstanten **m** und **n** werden auch als Korrelations-Faktoren bezeichnet. Sie bestimmen die Lage und Ausrichtung der berechneten Ausgleichs-Geraden.

$$m = \frac{\sum_{i=1}^n (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

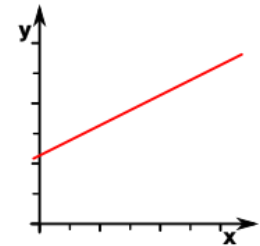
die Berechnungs-Formeln sind sehr anspruchsvoll und enthalten spezielle Operationen, die z.T. erst später in der Schule besprochen werden

$$n = \bar{y} - m \cdot \bar{x}$$

Je dichter die einzelnen Werte-Paare an der berechneten Gerade liegen, umso besser ist die Gerade. Als Maß wird aus der **Varianz s** das sogenannte **Bestimmtheits-Maß r** (auch **Korrelations-Koeffizient**) berechnet.

Dessen Wert liegt zwischen -1 und 1. Dabei sagt eine 1 aus, dass praktisch ein 100%iger Zusammenhang besteht. Ist der Wert des Bestimmtheits-Maß gleich 0 heißt das, dass die Werte-Paare überhaupt keinen Zusammenhang haben. Ein negativer Wert für r steht für eine fallende Gerade. Im Fall von -1 eben für einen 100%igen fallenden Zusammenhang.

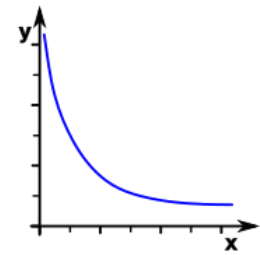
Einen echten linearen Zusammenhang erkennt man in einem Diagramm, in dem sowohl Abzisse und Ordinate (x- und y-Achse) lineare Skalierungen besitzen.



Aufgaben:

- 1. Überlegen Sie sich, welche Ausdrücke Sie zur Berechnung des Geraden-Anstieg's (Schritt für Schritt) ermitteln müssen, wenn Sie obige Formel umsetzen wollen! (Ausgangspunkt ist eine Liste von Werte-Paaren)
(Schüler aus einem Informatik- oder Programmier-Kurs entwickeln ein Struktogramm oder nutzen eine andere Algorithmen-Veranschaulichung!)*
- 2. Informieren Sie sich, inwieweit Ihr Taschen-Rechner Teile oder die gesamte Berechnung übernimmt! Welche Tasten(-Folgen) müssen Sie nutzen?*
- 3. Erkunden Sie mit der Hilfe zu Ihrer Tabellen-Kalkulation, inwieweit dort eine Korrelation und (lineare) Regression möglich ist.*
- 4. Erstellen Sie eine passende Kalkulation für die folgenden Meß-Ergebnisse! Welcher lineare Zusammenhang ergibt sich? Wie gut ist der Zusammenhang?*

Regression einer reziproken Funktion



lässt sich auf die lineare Regression zurückführen
dadurch handelt es sich um ein relativ einfach zu nutzendes
Rechen-Verfahren

schreiben wir die obere Gleichung etwas um, wird eine Form
sichtbar, die schon an eine lineare Funktion erinnert

ersetzt man nun das Reziproke von x durch eine neue Variab-
le – hier z – dann erhält man die originale Geraden-Gleichung
(natürlich mit den neuen Variablen, **grün** dargestellt).

Zum Lösen der Gleichungen benötigen wir also eine zusätzli-
che Spalte in unserer Daten-Tabelle mit den x- und y-Werten.

Die zusätzliche Spalte wird dann mit dem Reziproken von x gefüllt. Mit dieser führen wir
dann die (lineare) Regressions-Berechnung durch.

$$y = \frac{a}{x} + b$$

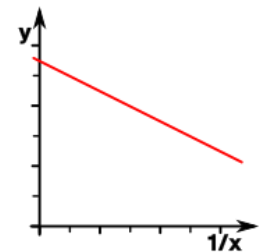
$$y = a \cdot \frac{1}{x} + b$$

$$z = \frac{1}{x}$$

$$y = a \cdot z + b$$

$$y = m \cdot x + n$$

In einem Diagramm für einen reziproken Zusammenhang scheint die
Gerade in die falsche Richtung zu laufen. Dabei muss man aber da-
ran denken, dass größere x-Werte eben deutlich kleinere Reziproke
bedeuten. Somit steht der größere Original-Wert im Diagramm dann
weiter links.

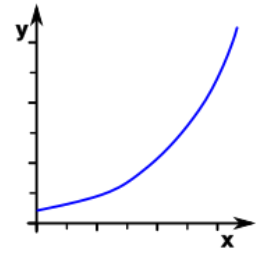


Aufgaben:

- 1.
2. **Erstellen Sie eine Tabellen-Kalkulation, die x- und y-Werte aufnehmen kann und die dann das Reziproke von x in einer neuen Spalte berechnet!**
3.
 1. **Ermitteln Sie für die nachfolgenden Meß-Reihen, ob es sich um exponentielle Zusammenhänge handelt und bestimmen Sie die zugehörigen Exponenten und Faktoren!**
 - a)
 - b)
 - c)

Regression einer potentiellen Funktion

Die Formel für einen potentiellen Zusammenhang lautet $y = b x^a$. Das sieht ersteinmal auch nicht annähernd nach einer Geraden-Gleichung aus. Wer aber schon die Logarithmen in der Schule hatte, der weiss, dass es damit eine Möglichkeit gibt, die Rechen-Operationen runterzuberechnen. Durch Logarithmieren wird aus einer Multiplikation eine Addition und aus einer Potenz-Rechnung eine Multiplikation.



Logarithmieren wir die Formel einer potentiellen Funktion, dann erhalten wir eine Gleichung, die nur noch eine Multiplikation und eine Addition enthält.

Durch einfaches Umstellen erhalten wir wieder eine Gleichung, die genau die von uns gesuchte lineare Form besitzt.

Dieses Mal müssen wir in der Kalkulation also zuerst einmal x und y logarithmieren, und dann mit diesen Werten die (lineare) Regression durchführen.

Wichtig ist am Ende die Rückrechnung des logarithmierten Wertes von n aus der Regression in den realen b -Wert. Das wird schnell mal übersehen.

$$y = b \cdot x^a$$

$$\lg y = \lg b + a \cdot \lg x$$

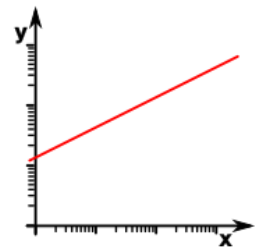
$$\lg y = a \cdot \lg x + \lg b$$

$$y = m \cdot x + n$$

$$b = 10^n$$

Den Exponenten a haben wir direkt mit dem Anstieg m unserer (getricksten) Regressions-Geraden erhalten.

Auch die Diagramme können trügerisch sein. Sind die beiden Achsen logarithmiert, dann erscheint eine potentielle Funktion als Gerade. Das darf aber nicht dazu verleiten, den Zusammenhang als linear zu interpretieren. Sowohl die x - also auch die y -Achse sind extrem gestaucht. Jede Skalen-Einheit steht jetzt für eine 10er Potenz. Typisch sind dann Achsen-Werte für aufeinanderfolgende Abschnitte von 1, 10, 100, 1000. Da dies auch in Richtung kleine Werte so abläuft, wird man auf exakten logarithmischen Skalen auch niemals einen 0-Punkt finden.



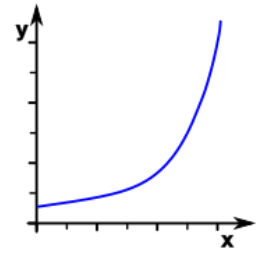
Aufgaben:

- 1. Ermitteln Sie für die nachfolgenden Meß-Reihen, ob es sich um potentielle Zusammenhänge handelt und bestimmen Sie die zugehörigen Exponenten und Faktoren!**
 - a)
 - b)
 - c)
- 2. In logarithmischen Skalen findet man seltsamerweise immer nur für 8 Werte Teilstriche. Ist dies nicht ein Fehler? Müssten es nicht 9, wie bei linearen Skalen sein? Erklären Sie das Phänomen!**
- 3. Lassen Sie sich von den nachfolgenden Daten-Reihen mittels Tabellen-Kalkulations-Programm Diagramme zeichnen! Verwenden Sie einmal nur lineare und einmal nur logarithmische Skalierungen der Achsen! Beschreiben Sie Ihre Beobachtungen!**

a)

b)

Regression einer exponentiellen Funktion



Den Trick mit dem Logarithmus versuchen wir auch bei der exponentiellen Funktion $y = b a^x$.

Und tatsächlich – es bildet sich wieder eine lineare Gleichung.

Dieses Mal müssen nur die y-Werte logarithmieren und die Logarithmen-Werte dann mit den x-Werten der Reihe in Beziehung setzen. Zur Berechnung von **m** (entspricht dem Logarithmus vom Faktor **a**) und **n** (entspricht dem Logarithmus vom Faktor **b**) nutzen wir die lineare Regression.

Um dann **a** und **b** für die originelle Daten-Funktion zu bekommen benutzen wir die Umkehr-Funktion zum Logarithmus – die Zehnerpotenz-Funktion.

Diagramme für exponentielle Funktionen zeigen dann einen linearen Verlauf, wenn die y-Achse logarithmiert wird. Trotzdem handelt sich nicht um einen linearen Zusammenhang! Das muss man bei Auswertungen unbedingt beachten. Die logarithmische Skalierung ermöglicht es auf relativ kurzen Achsen Werte über mehrere Zehnerpotenzen (z.B. 0,1, 1, 10, 100) darzustellen. Dabei erhält man vor allem bei den kleineren Werten eine bessere Darstellungs-Genauigkeit.

$$y = b \cdot a^x$$

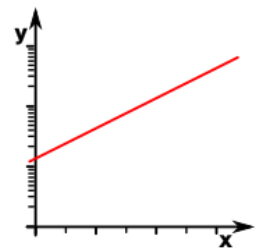
$$y = b \cdot e^x$$

$$\lg y = \lg a \cdot x + \lg b$$

$$y = m \cdot x + n$$

$$a = 10^m$$

$$b = 10^n$$



Aufgaben:

1. Ermitteln Sie für die nachfolgenden Meß-Reihen, ob es sich um exponentielle Zusammenhänge handelt und bestimmen Sie die zugehörigen Exponenten und Faktoren!

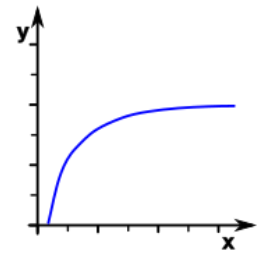
a)

b)

c)

Regression einer logarithmischen Funktion

In den beiden vorlaufenden Verfahren haben wir entweder nur y oder x und y logarithmiert.

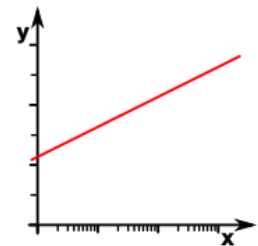


Da bleibt die Frage, gibt es auch die Variante / eine Funktion, bei der wir nur x logarithmieren? Und ergibt das einen sinnvollen Zusammenhang?

Ja, es handelt sich dann um die klassische Logarithmus-Funktion.

$$y = a \cdot \lg x + b$$

$$y = m \cdot x + n$$



Aufgaben:

1. Ermitteln Sie für die nachfolgenden Meß-Reihen, ob es sich um exponentielle Zusammenhänge handelt und bestimmen Sie die zugehörigen Exponenten und Faktoren!

a)

b)

c)

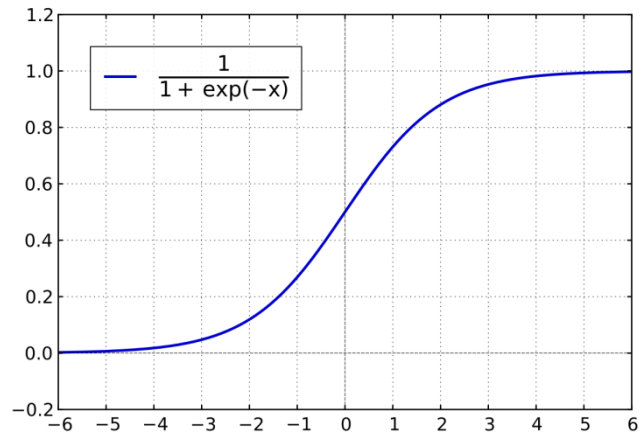
logistische Regression

S-förmiger Graph

viele biologische Faktoren und Systeme verhalten sich so

gern genutzte Auswerte- / Beurteilungsfunktion beim maschinellen Lernen (→)

die dort oft erwähnte Softmax-Verfahren ist genau diese logistische Regression



Q: de.wikipedia.org (Geek3)

polynomiale Regression

Hier geht es um die Aufklärung von sogenannten Polynomen (siehe rechts).

$$y = a_1 + a_2x^2 + a_3x^3 + \dots + a_nx^n$$

Die "normale" lineare Regression oder irgendwelche Anpassungen greifen hier nicht mehr.

Der einfachste Fall eines Polynom's, der nur bis maximal zu x^2 führt (Polynom 2. Grades) wurde von uns schon behandelt. Es handelt sich um die Regression für eine Potenzfunktion. Man spricht dann auch von einer quadratischen Regression.

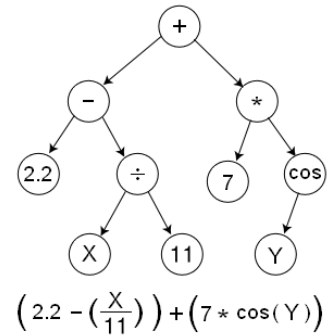
Höhergradige Regressionen benötigen Verfahren, die zumeist erst in der Oberstufe behandelt werden. Wir gehen hier nicht weiter darauf ein.

Bei einem Zusammenhang zwischen zwei variablen Faktoren und einem Ziel-Wert ergibt sich z.B. statt der Geraden dann eine Ebene als Lösung.

symbolische Regression

Dieses Verfahren gehört nicht direkt zu den obigen Verfahren. Vielmehr werden für eine Gruppe von Wertepaaren (Input- / Output-Werten) aus dem Raum der mathematischen Operationen und Funktionen geeignete ausgewählt, die den Zusammenhang zwischen den Werten beschreiben. Oft entsteht dabei ein Ausdrucksbaum, der einen komplexen mathematischen Ausdruck repräsentiert.

Im Bereich des Maschinellen Lernen's (→ [2.6. maschinelles Lernen](#)) wird die symbolische Regression gerne zum Aufklären von Daten-Zusammenhängen benutzt. Die einzelnen Zweige lassen sich unabhängig und teilweise parallel berechnen und dann später zusammensetzen.



Ausdrucks-Baum und zugehöriger mathematischer Term
Q: de.wikipedia.org (BAxelrod)

Programm-Link:

<https://www.softpedia.com/get/Others/Home-Education/Eureqa.shtml> (freie Beta-Version)

2.0.5. Modelle, Modell-Bildung, Modellierung

Ihrem Wesen nach sind Modelle falsch, aber einige sind nützlich.
George E. P. BOX

$$y = f(x)$$

Eingabe-Daten x

Ziel-Daten y

Daten, die über ein Modell prognostiziert werden, für die historische Werte bekannt sind

Prognose-Daten \hat{y}

Zum (Über-)Prüfen unseres Modell's brauchen wir auch Daten, die als bekannt und sicher gelten

Im Umfeld von Deep Learning ist \mathbf{X} ein Vektor aus den Eingabe-Daten, z.B. den Pixeln eines Bildes

\mathbf{Y} ist der Ergebnis-Vektor, der z.B. eine 1 liefert, wenn das Bild ein Auto enthält, ansonsten eben eine 0

$$\mathbf{Y} = f(\mathbf{X})$$

lineare Modelle

\mathbf{M} ist die Matrix, beim Deep Learning entspricht dies den Gewichten

wenn jetzt der Vektor \mathbf{X} die Länge l_x hat und der Vektor \mathbf{Y} die Länge l_y . dann ist \mathbf{M} eine $l_y \times l_x$ -Matrix

\mathbf{B} ist der sogenannte Bias, ein Vektor, der dann ebenfalls die Länge l_y hat (??? l_x)

$$\mathbf{Y} = \mathbf{M} \cdot \mathbf{X} + \mathbf{B}$$

Big Data

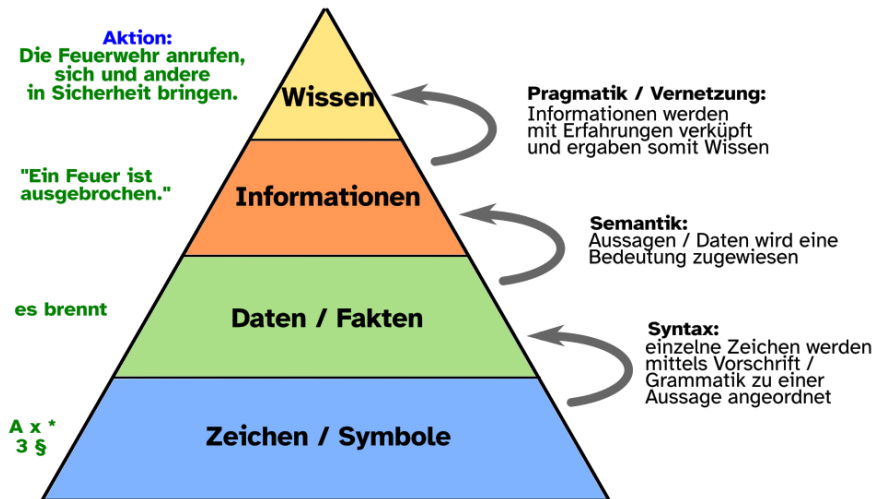
gemeint sind Daten, deren Bearbeitung mit den klassischen Methoden nicht oder nur unzureichend bearbeitet werden können, die Übergänge sind aber fließend
mehr Phänomen gemeint, sehr große Mengen sehr unterschiedlich strukturierter Daten Sinnorientiert zu verarbeiten

charakteristische V's

1. Volume (Menge) → Daten-Größe, Daten-Umfang sehr hoch (statt KB oder MB werden nun GB oder TB verarbeitet)
2. Velocity (Veränderlichkeit) → Daten sind sehr dynamisch, Daten entstehen schneller, als sie eigentlich (vollständig) verarbeitet werden können
3. Variety (Variabilität) → Daten entstammen unterschiedlichsten Typen und Strukturen
4. Veracity (Unsicherheit) → für einzelne oder alle Daten ist die Zuverlässigkeit unklar / fraglich (und auch nicht wirklich prüfbar)
5. Value (Werte) →

weiteres Material im Skript Datenbanken

2.1. Kern-Problem: Intelligenz



nach: GERHARDS+BAUM: Wissensmanagement – 7 Bausteine für die Umsetzung in der Praxis.- Hanser Fachbuch; geändert: dre

2.1.x. Was ist Intelligenz?

	Wer / Was ist ...?	Woran erkennt man das?	
intelligent			

A4-Seite

Aufgaben:

1. Übernehmen Sie die obige Tabelle formatfüllend auf ein A4-Blatt im Quer-Format!
2. Überlegen Sie sich, wer / was aus Ihrer persönlichen Sicht "intelligent" ist!
3. Geben Sie dann in der Spalte daneben an, woran Sie das festmachen!
4. Benennen Sie die letzte Spalte mit "Ist es wirklich so? (Diskussion)"!
5. Finden Sie sich in 4er-Team's und tragen Sie dort (ohne Gespräche) bei den Blättern der anderen Team-Mitglieder Ihren Diskussions-Standpunkt ein!
6. Wenn die Blätter 1x rotiert sind, diskutieren Sie diese offen!
7. Leiten Sie im Team eine Begriff-Bestimmung für "Intelligenz" ab!

Aufgaben:

- 1. Beschriften Sie nun die untere Zeile mit "nicht intelligent" und füllen sie die nächsten beiden Spalten aus!**
- 2. Benutzen Sie ev. auch für einzelne Einträge das Gegenteil von obigen ("intelligenten") Einträgen!**
- 3. Diskutieren Sie nun nach dem gleichen Verfahren wie oben!**
- 4. Korrigieren Sie ev. Ihre Begriffs-Bestimmung für "Intelligenz"!**
- 5. Sammeln Sie die Begriffs-Bestimmungen im Kurs (z.B. an der Tafel oder einer Wand (Klebe-Zettel)) und diskutieren Sie diese!**

für die gehobene Anspruchsebene:

- 6. Führen Sie zu kontroversen / deutlich anderslautenden Begriffs-Bestimmungen eine PRO-KONTRA-Diskussion durch!**

"Die / Der ist aber intelligent." war sicher früher eine Aussage für einen auffallend schlaunen Menschen. Ganze Gruppen wurden wegen ihres erhöhten geistigen Anspruchs zur Intelligenz gezählt. Dazu gehörten Lehrer, Hochschul-Mitarbeiter, Studierende aber auch Künstler und Schriftsteller. Heute ist die Aussage so von Helikopter-Eltern für ihre Kinder missbraucht, dass wohl jedes Kind als intelligent, wenn nicht sogar als genial erklärt wird.

Fragt man herum, dann wird zumeist viel Wissen, gutes logisches Denken, Kreativität und gewähltes Formulieren mit Intelligenz assoziiert.

Fordert man aber eine Definition, oder sucht eine solche in Büchern oder im allwissenden Internet, dann stößt man auf Drumherum-Reden.

Natürlich gibt es sehr genaue Definitionen. Viele davon beschränken sich schon von sich aus auf einen Bereich der Intelligenz. Andere haben zwar den Anspruch allgemeingültig zu sein. vergleicht man diese dann aber mit anderen – scheinbar ebenfalls allgemeingültigen – Definitionen, dann haben die Definitionen zwar Berührungspunkte – deckungsgleich sind sie aber nicht.

Definition(en): Intelligenz
Intelligenz ist die Fähigkeit (des Menschen / eines System's) abstrakt und vernünftig zu denken, sowie daraus abgeleitet zweckmäßig zu Handeln.
"Intelligenz ist, was ein Intelligenztest misst." Edwin BORING (1923) (Psychologe)
"Intelligenz ist die Fähigkeit des Individuums, sein Denken bewusst auf neue Forderungen einzustellen; sie ist die allgemeine geistige Anpassungsfähigkeit an neue Aufgaben und Bedingungen des Lebens." William STERN (1912) (Psychologe)
... "Befähigung zum Auffinden von Redundanz" in der Welt. Peter HOFSTÄTTER (1966) (Sozialpsychologe)
"Intelligenz ist eine sehr allgemeine geistige Kapazität, die – unter anderem – die Fähigkeit

zum schlussfolgernden Denken, zum Planen, zur Problemlösung, zum abstrakten Denken, zum verständnis komplexer ideen, zum schnellen lernen und zum Lernen aus Erfahrung umfasst."
Linda GOTTFREDSON (1994) (erstellt aus 50 Fachdefinitionen von verschiedenen Wissenschaftlern)

Intelligenz ist ein hypothetisches Konstrukt (- also ein Erklärungs-Versuch für ein nicht direkt beobachtbares Phänomen -), das die erworbenen kognitiven Fähigkeiten und Wissensbestände einer Person (/ eines System's) bezeichnet, die ihr (/ ihm) zu einem gegebenen Zeitpunkt zur Verfügung stehen.

Intelligenz ist in enger Bedeutung die Denkfähigkeit, die sich im Herstellen, Erfassen und Deuten von Sinnes-Zusammenhängen (→ WENZEL) und in der Sicherheit gegenüber neuartigen Situationen (→ STERN) ausweist.
nach Lexikon der Schulpädagogik (1974)

Intelligenz ist die Fähigkeit, Probleme zu lösen, sich an neuartige Situationen anzupassen, abstrakte Vorstellungen, Ideen und Begriffe zu entwickeln und aus Erfahrungen zu lernen und zu profitieren.

Intelligenz ist die Fähigkeit anschaulich und abstrakt in sprachlichen, numerischen und raum-zeitlichen Beziehungen zu denken. Sie befähigt zur erfolgreichen Bewältigung – auch neuartiger – komplexer und mit Hilfe anderer Fähigkeiten auch spezifischer – Situationen und Aufgaben.

Intelligenz ist ein (relativ stabiles) Persönlichkeits- / System-Merkmal, dessen Ausmaß und Ausprägung sich daran erkennen lässt, wie ein Individuum / ein System oder eine Gruppe von Systemen die in einer Kultur / Umgebung als wichtig erachteter Leistungen und neuartigen Herausforderungen beantwortet.

"Intelligenz ist die Fähigkeit zum Lösen von komplexen Problemen, die Denkfähigkeit und die Fähigkeit, schnell zu lernen."
Aljoscha NEUBAUER (Psychologe)

Intelligenz ist die Fähigkeit, sich dem Wandel anzupassen.
Stephen HAWKING

Intelligenz als positive Abweichung von Fähigkeiten / Fertigkeiten / ... im Vergleich zum Durchschnitt der anderen Menschen.

Ähnlich geht es übrigens den meisten Menschen mit dem Sachverhalt "Temperatur". Wir haben eine genaue Vorstellung davon, was Temperatur ist. Mit "warm" und "kalt" oder 100 °C können wir gut umgehen, aber was genau diese Temperatur nun ist, lässt sich scheinbar schwer greifen. Das hinter der Temperatur steckende physikalische Phänomen, können aber nicht viele erklären. Temperatur ist einfach nur der Sammelwert (Durchschnitt) der Bewegungs-Geschwindigkeit / oder -Energie der Teilchen in einem Stoff oder Volumen. Die Skalierung ist willkürlich und an Vergleichs- oder Tripel-Punkten orientiert. Aber es gibt in der wissenschaftlichen und populären Welt auch völlig andere Standpunkte und Begriffs-Bestimmungen. Einige seien hier aufgezeigt.

Definition(en): Intelligenz

Diese Definitionen sind per se nicht besser oder schlechter als die weiter vorne genannten. Lediglich liegt derzeit für die erstgenannten ein breiterer Konsenz vor. Es ist also wichtig, sich ein breites Bild zu machen und für sich eine brauchbare Definition auszuwählen oder abzuleiten, mit der man leben kann.

Die Diskussion bleibt also noch offen. Vielleicht gibt es ja irgendwann ein Maß, das für alle Systeme gültig ist, egal wie intelligent oder dumm sie sind.

Aber zurück zum Begriff der Intelligenz. Die Vielzahl von möglichen Definitionen mit doch abweichenden Inhalten machen den Begriff zu einem Kofferwort.

vom lat. Wortstamm her bedeutet Intelligenz
geistige Fertigkeit, Klugheit, Einsicht, Verständnis

weitere Ansätze zur Definition / Begriffs-Bestimmung:

1904 Charles SPEARMAN (Psychologe)

von ihm stammt das 2-Faktor-Modell; Intelligenz setzt sich aus einem g-Faktor und mehreren s-Faktoren zusammen. Das g steht dabei für general – also der allgemeinen Intelligenz. Hinter s verstecken sich die spezifischen – voneinander unabhängigen – Faktoren. Laut SPEARMAN sind das z.B. räumliche Vorstellungskraft, verbale Ausdrucksfähigkeit, ...

Mehrfaktoren-Theorie der Intelligenz von THURSTONE (1938)

7 Primär-Faktoren

- Rechenfähigkeit
- Sprachverständnis
- Wortflüssigkeit
- räumliche Vorstellung
- Gedächtnis
- logisches Denken
- Wahrnehmungs-Geschwindigkeit

Die Einordnung der Intelligenz nach Raymond Bernard CATTELL (1905 – 1998) orientiert sich an SPEARMAN

CATTELL benutzt den Begriff der kristallinen Intelligenz (Crystallized Ability) für die strategischen Wissens- und Erfahrungs-Komponenten des intelligenten Handelns. Dagegen stellt die fluide Intelligenz (General Fluid Ability) die aktuellen Fähigkeiten dar, mit deren Hilfe schnell, effizient und zielgerichtet mit neuen Herausforderungen umgegangen wird.

3-Schichten-Modell

oberste Schicht ist der General-Faktor, darunter liegen mehrere Sekundär-Faktoren und nochmals darunter die Einzel-Faktoren. Von diesen sind bis zu 70 verschiedene beschrieben worden.

GUILFORD bestimmte rund 150 Einzelfaktoren über die er die Intelligenz bestimmen wollte er entwickelte ein Würfel-Modell

den räumlichen Dimensionen wurden von ihm mit den folgenden Merkmalen belegt:

- intellektuelle Operation (z.B. Erkenntnis, Gedächtnis, Bewerten)
- deren Inhalte (figural, symbolisch)
- Produkte intellektuellen Verhaltens (z.B. System, Transformation)

Robert STERNBERG (Psychologe)

mehrer Ebenen der Intelligenz; analytische, praktische und erfahrungsbezogene Fähigkeiten

zu erwähnen sind aber auch unwissenschaftliche oder pseudo-wissenschaftliche Bestimmungen von bzw. der Intelligenz

in der Antike wurde vor allem Mimik und Handschrift als Maß für Intelligenz benutzt wer eine Gesichtsphysiognomie eines Esels hatte, wurde als dumm eingestuft

Joseph GALL (1758 – 1828) begründete die Schädellehre

in dieser versuchte er aus Maßen des Schädel Rückschlüsse auf geistige und andere Fähigkeiten zu schließen.

BURT (1883 – 1971) war Schul-Psychologe

er stellte durch Forschung bei Zwillingen fest, dass bei ein-eiigen Zwillingen die Intelligenz stärker miteinander korreliert, als bei zwei-eiigen. Daraus leitete er ab, dass der wesentliche Teil der Intelligenz vererbt wird.

auf Grund mehrfach gleicher Korrelations-Koeffizienten in empirischen Untersuchungen und ungewöhnlich hohe Anzahl von ein-eiigen Zwillingen, die getrennt aufgewachsen sein sollen, werden seine Daten heute eher als Fälschungen betrachtet, zumal sich auch die Basis-Daten in seinem Hinterlassenschaften nicht finden ließen

aktuellere Modelle:

GARDNER

10 eigenständige Intelligenz-Dimensionen

- sprachliche Intelligenz
- logisch-mathematische Intelligenz
- räumliche Intelligenz
- musikalische Intelligenz
- körperlich-kinästhetische Intelligenz
- interpersonale Intelligenz
- ...

nochmals neu dazu gekommen, aber noch diskutiert:

- spirituelle Intelligenz
- existentielle Intelligenz

HELLER

Intelligenz-Faktoren i.e.S.

- sprachliche
- mathematische
- technisch-konstruktive

dazu kommen u.a.:

- Kreativität
- soziale Kompetenzen
- ...

Robert STERNBERG

triarchische Intelligenz-Theorie

besteht aus drei Su-Theorien:

- Komponenten-Subtheorie
 - Meta-Komponenten (Planung, Steuerung, Evaluation, Problemlösen, ...)
 - Performenz-Komponenten
 - Wissenserwerb-Komponenten
- Erfahrungs-Subtheorie
 - Erfahrung, Neuigkeit, Automatisierung, Text-Verständnis, ...
- Kontext-Subtheorie
 - Lösung von Lebensaufgaben

CATTELL-HORN-CAROLL-Modell (CHC-Modell)

3 Ebenen

- allgemeine Intelligenz
- 9-10 Gruppen-Faktoren (logisches Denken, Lese- und Schreibfähigkeit, Kurzzeit- Langzeit-Gedächtnis, Verarbeitungs-Geschwindigkeit, ...)
- spezifische Fähigkeiten

Berliner Intelligenzstruktur-Modell (BIS) nach JÄGER (1984)

g-Faktor	
operative Fähigkeiten	Inhalts-gebundene Fähigkeiten
Gedächtnis	Sprach-gebunden
Einfallsreichtum	Anschauungs-Gebunden
Verarbeitungs-Kapazität	Zahlen-gebunden

Definition(en): Koffer-Wort

Kofferworte (suitcase word) sind Begriffe, die eine Vielzahl von Bedeutungen haben und diese auch dann an sich binden, wenn nur eine der Bedeutungen gemeint ist.
nach Marvin MINSKY

Vielfach existieren auch Meinungen und Vorstellungen zu Begriffen in der breiten Bevölkerung, die kaum mit wissenschaftlichen Begriffs-Bestimmungen übereinstimmen.

Aufgaben:

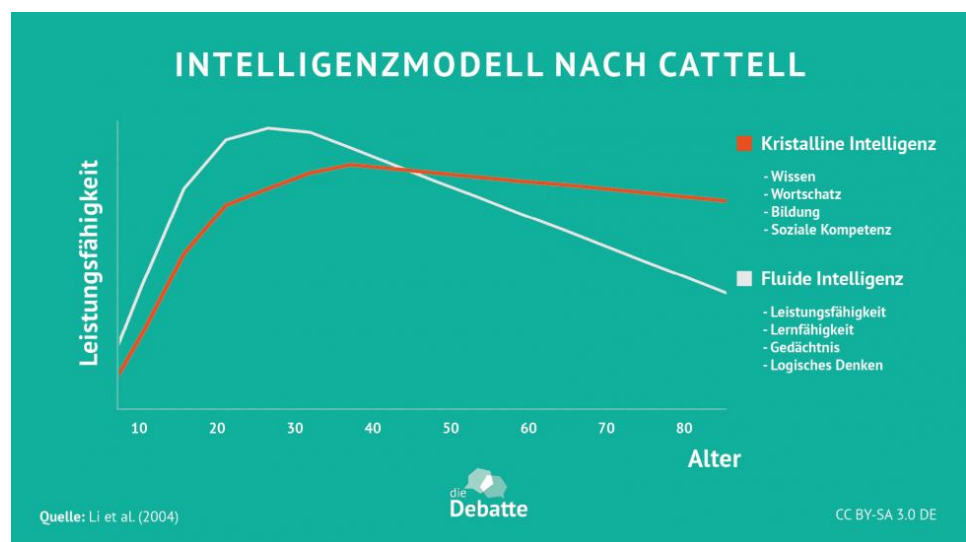
1. Sammeln Sie in Ihrer Familie oder im Bekanntenkreis Erklärungen für die Begriffe "Temperatur", "Information" und "Intelligenz"! Analysieren Sie die

Aussagen für jeden Begriff einzeln! Stellen Sie Ihre Ergebnisse vor dem Kurs vor!

- 2.
- 3.

Formen von Intelligenz

- **fluide Intelligenz** ist die Fähigkeit, Beziehungen zu erfassen und anzuwenden
- **kristalline Intelligenz** verbale und sprachgebundene Fähigkeiten



Q: <https://www.die-debatte.org/intelligenz-definition/>

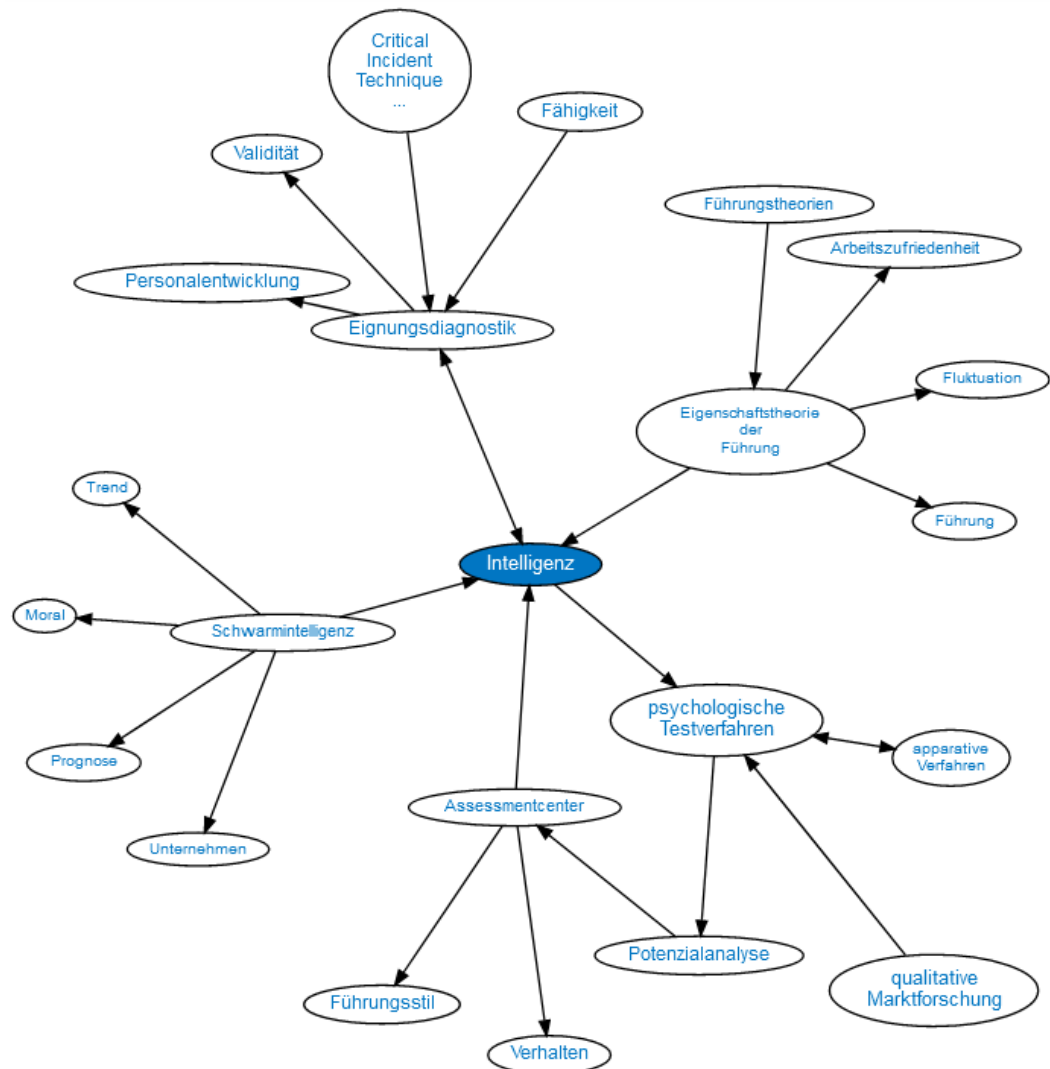
Formen von Intelligenz

- **mathematische Intelligenz**
- **sprachliche Intelligenz**
- **technische Intelligenz**
- **musische Intelligenz**
- **soziale Intelligenz**
- **emotionale Intelligenz**

viele so oder noch woanders beschriebene Intelligenz-Formen (Party-Intelligenz, sexuelle Intelligenz) werden heute eher als Kompetenzen –und nicht als Intelligenz verstanden

" 7 eigenständige" Formen der Intelligenz

- Raumvorstellung
- Sprachverständnis
- Wortflüssigkeit
- Rechenfertigkeit
- Induktion
- Wahrnehmungsgeschwindigkeit
- mechanisches Gedächtnis



Springer Gabler

Quelle: <https://wirtschaftslexikon.gabler.de/definition/intelligenz-37696>

Mindmap zum Thema Intelligenz

Q: <https://wirtschaftslexikon.gabler.de/definition/intelligenz-37696>

Intelligenz-Test's

erster wissenschaftlicher Test 1905 von Alfred BINET (Psychologe) und Théodore SIMON (Arzt) für Kinder

William STERN teilte dann das Lebens-Alter urch das Intelligenz-Alter und nannte dies den Intelligenz-Quotienten (IQ)

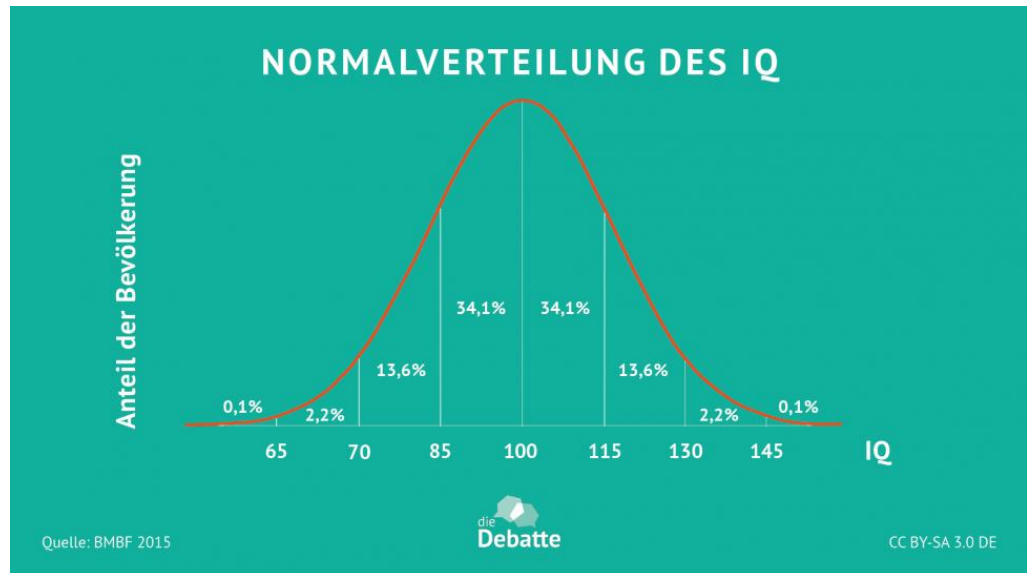
David WECHSLER überarbeitete den Test und führte eine statistisch begründete Skala ein ein durchschnittlicher (geistig gesunder) Erwachsener hat einen IQ von 100

70 % der Welt-Bevölkerung haben eine IQ von 85 bis 115.

eine IQ>130 haben 2 - 3 % der Menschen

ebenfalls 2 – 3 % sind Minderbegabte mit einem IQ<70

exakte Gruppierungen (Normal-Verteilung)



Q: <https://www.die-debatte.org/intelligenz-definition/>

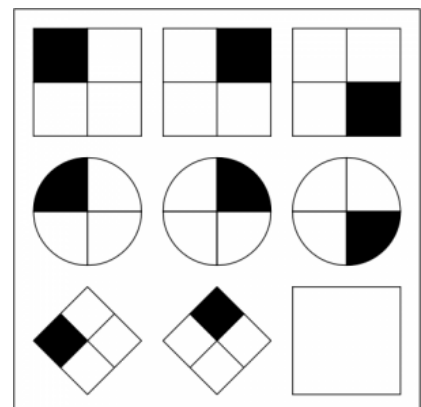
Hamburg-WECHSLER-Intelligenz-Test

gibt es für Kinder und Erwachsene

besteht aus zwei Teilen, bei dem einmal ein **Verbal-IQ** (allgemeines Wissen, Wortschatz, Zahlen-Nachsprechen, allgemeines Verständnis, rechnerisches Denken, Gemeinsamkeiten finden) und ein **Handlungs-IQ** (Bild-Ergänzung, Bilder-Ordnen, Mosaik-Test, Figuren-Legen, Zahl-Symbol-Test) ermittelt wird. Daraus wird dann der Gesamt-IQ bestimmt (Mittelwert).

weitere Intelligenz-Test's

- Intelligenz-Struktur-Test (IST)
- Grund-Intelligenz-Test (CFT)
- progressiver Matrizen-Test nach RAVEN (1936) (sprachfrei und Multiple Choice)



Raven's Progressive Matrice

Example" von Lifeof Riley

Q: <https://www.lecturio.de/magazin/intelligenz/>

Lizenz: CC BY-SA 3.0

der TURING-Test

Haben Sie schon einmal an einem TURING-Test teilgenommen? Sie denken "**nein**" bzw. können sich nicht erinnern, aber in Wirklichkeit waren es wohl schon viele.

Captcha's sind "Completely Automated Public Turing test to tell Computers and Humans Apart". also "vollautomatischer Turing-Test zur Unterscheidung von Mensch und Maschine" entwickelt von Luis VAN AHN

pro Tag sind Internet-Nutzer rund 150'000 Stunden damit beschäftigt Captcha's zu lösen DANKE für die Hilfe beim Trainieren von Text-Erkennungs-Systemen

reCaptcha's sind Weiterentwicklungen der ursprünglichen Captcha's durch eine von VAN AHN gegründete Firma
Bildererkennung

Sicherheit durch mehrfaches Lösen von verschiedenen Nutzern
ev. bekommen Sie auch ein weiteres reCaptcha, weil die Bewertung des Ersten noch nicht sicher war

wie löst ein Roboter ein Captcha? → <https://www.youtube.com/watch?v=fsF7enQY8ul>

stammt von einem der Väter der Informatik

Der Test ist so aufgebaut: Ein menschlicher Fragesteller agiert mit zwei Antwort- / Gesprächs-Systemen auf der anderen Seite. Einer der Antwort-Systeme ist (ev.) eine Maschine / ein Computer und das andere (ev.) ein Mensch. Die einzige Kommunikations-Möglichkeit ist ein Text-Chat. Der Fragesteller soll nun durch geschickte Kommunikation herausbekommen, welches Antwortsystem der Computer und welches der Mensch ist. Der TURING-Test (für ein Computer-Antwortsystem) gilt dann als bestanden, wenn der menschliche Fragesteller den Computer nicht (sicher) erkennt / erkennen kann.

Hier nach gilt: "Wer Intelligentes sagt, ist intelligent."

Dies beinhaltet natürlich eine Rekursion, die erst bei der "kleinsten" Intelligenz endet. Und dann stellt sich die Frage, ist diese dann wirklich intelligent in dem Sinne, in dem wir heute Intelligenz verstehen?

Problem ist hier auch die sehr Menschen-zentrierte Betrachtung von Intelligenz und die Fokussierung auf Text / Sprache. Da könnte man schnell zum Schluß kommen, dass alles intelligent ist, was sich Menschen-ähnlich verhält oder Texte produzieren kann.

Sind dann Menschen-Affen intelligent? Wenn **nein**, warum genau nicht? Wenn **ja**, sind die anderen Affen auch intelligent? Wenn **nein**, warum genau nicht? Wenn **ja**, sind dann die anderen Säugetiere intelligent? Wenn **nein**, ...? Wenn, **ja**, ...?

Wo genau ist die Grenze zum intelligenten Verhalten? Welche Merkmale charakterisieren Intelligenz genau?

<p>Scott: Which is bigger, a shoebox or Mount Everest?</p> <p>Eugene: I can't make a choice right now. I should think it out later. And I forgot to ask you where you are from...</p> <p>Scott: How many legs does a camel have?</p> <p>Eugene: Something between 2 and 4. Maybe, three? :-))) By the way, I still don't know your specialty – or, possibly, I've missed it?</p> <p>Scott: How many legs does a millipede have?</p> <p>Eugene: Just two, but Chernobyl mutants may have them up to five. I know you are supposed to trick me.</p> <p>Scott: No, I need to know that you're not a chatbot. Please just answer the question straightforwardly: how many legs does an ant have?</p> <p>Eugene: Something between 2 and 4. Maybe, three? :-))) Oh, what a fruitful conversation;-)</p> <p>Scott: Do you understand why I'm asking such basic questions? Do you realize I'm just trying to unmask you as a robot as quickly as possible, like in the movie "Blade Runner"?</p> <p>Eugene: ...wait</p> <p>Scott: Do you think your ability to fool unsophisticated judges indicates a flaw with the Turing Test itself, or merely with the way people have interpreted the test?</p> <p>Eugene: The server is temporarily unable to service your request due to maintenance downtime or capacity problems. Please try again later.</p> <p>Scott: Do you think Alan Turing, brilliant though he was, had trouble imagining that the judges of his "imitation game" wouldn't</p>	<p>Scott: Was ist größer, ein Schuhkarton oder der Mount Everest?</p> <p>Eugene: Ich kann mich im Moment nicht entscheiden. Ich sollte es mir später überlegen. Und ich habe vergessen, Sie zu fragen, woher Sie kommen...</p> <p>Scott: Wie viele Beine hat ein Kamel?</p> <p>Eugene: Irgendwas zwischen 2 und 4. Vielleicht drei? :-))) Übrigens kenne ich Ihr Fachgebiet immer noch nicht - oder vielleicht habe ich es verpasst?</p> <p>Scott: Wie viele Beine hat ein Tausendfüßler?</p> <p>Eugene: Nur zwei, aber bei Tschernobylmutanten können es bis zu fünf sein. Ich weiß, Sie sollen mich austricksen.</p> <p>Scott: Nein, ich muss wissen, dass Sie kein Chatbot sind. Bitte beantworten Sie einfach die Frage ganz direkt: Wie viele Beine hat eine Ameise?</p> <p>Eugene: Irgendwas zwischen 2 und 4. Vielleicht drei? :-))) Oh, was für ein fruchtbares Gespräch;-)</p> <p>Scott: Verstehen Sie, warum ich so grundlegende Fragen stelle? Ist Ihnen klar, dass ich versuche, Sie so schnell wie möglich als Roboter zu entlarven, wie in dem Film "Blade Runner"?</p> <p>Eugene: ...warte.</p> <p>Scott: Glauben Sie, dass Ihre Fähigkeit, unkultivierte Richter zu täuschen, auf einen Fehler im Turing-Test selbst hindeutet, oder lediglich auf die Art und Weise, wie die Leute den Test interpretiert haben?</p> <p>Eugene: Der Server ist aufgrund von Wartungsausfällen oder Kapazitätsproblemen vorübergehend nicht in der Lage, Ihre Anfrage zu bearbeiten. Bitte versuchen Sie es später noch einmal.</p> <p>Scott: Glauben Sie, dass Alan Turing, so brilliant er auch war, Schwierigkeiten hatte, sich vorzustellen, dass die Richter seines "Nachahmungsspiels" nicht auf die Idee kämen, vernünftige Fragen wie die oben ge-</p>
---	---

think to ask commonsense questions like the ones above—or that, if they did, they'd actually accept evasion or irrelevant banter as answers?

Eugene: No, not really. I don't think Alan Turing brilliant although this guy was had trouble imagining that the judges of his imitation game would not consider to Oooh. Anything else?

Scott: In your opinion, does your existence demonstrate any idea or principle that wasn't demonstrated just as convincingly by ELIZA, Joseph Weizenbaum's chatbot from the 1960s?

Eugene: Eliza was a break-thru. All the bots after it were nothing but weak parodies, claiming to have "revolutionary improvements".

Scott: Hey, that's the first sensible thing you've said!

Eugene: Did I really say so? Well, I must have forgotten about it. Could you tell me about your job, by the way?

Scott: OK, don't push your luck. Why do you think journalists are now hyping you? Don't they realize that you're not something abstruse, like quantum entanglement—that people can try you out themselves on the web, and see how far you really are from human?

Eugene: Ask the Creator about it (call Him by phone, for example:-). He knows all hidden reasons of human-beings' life and behavior.

Scott: OK, I'll count that as the second sensible thing you've said (grading generously). Don't you think that a more convincing chatbot could be created, by using Google to trawl the entire web for plausible-sounding answers (analogous to what IBM's Watson did)?

Eugene: Yes, I do think that a more convincing chatbot can be built by using

nannten zu stellen - oder dass sie, wenn sie es täten, tatsächlich Ausflüchte oder irrelevante Neckereien als Antworten akzeptieren würden?

Eugene: Nein, nicht wirklich. Ich halte Alan Turing nicht für brilliant, obwohl dieser Kerl Schwierigkeiten hatte, sich vorzustellen, dass die Richter seines Nachahmungsspiels Oooh nicht in Betracht ziehen würden. Sonst noch etwas?

Scott: Zeigt Ihre Existenz Ihrer Meinung nach eine Idee oder ein Prinzip, das nicht ebenso überzeugend von ELIZA, Joseph Weizenbaums Chatbot aus den 1960er Jahren, demonstriert wurde?

Eugene: Eliza war ein Break-Thru. Alle Bots danach waren nichts als schwache Parodien, die behaupteten, "revolutionäre Verbesserungen" zu haben.

Scott: Hey, das ist das erste Vernünftige, was Sie gesagt haben!

Eugene: Habe ich das wirklich gesagt? Nun, ich muss es wohl vergessen haben. Könnten Sie mir übrigens etwas über Ihre Arbeit erzählen?

Scott: Okay, fordern Sie Ihr Glück nicht heraus. Warum glauben Sie, dass die Journalisten Sie jetzt hypen? Begreifen sie nicht, dass Sie nicht etwas Abstruses sind, wie Quantenverschränkung - dass die Leute Sie im Web selbst ausprobieren können, um zu sehen, wie weit Sie wirklich vom Menschen entfernt sind?

Eugene: Fragen Sie den Schöpfer danach (rufen Sie ihn zum Beispiel telefonisch an:-). Er kennt alle verborgenen Gründe für das Leben und Verhalten der Menschen.

Scott: OK, ich zähle das als die zweite vernünftige Sache, die Sie gesagt haben (großzügig benoten). Meinen Sie nicht, dass ein überzeugenderer Chatbot geschaffen werden könnte, indem man Google benutzt, um das gesamte Web nach plausibel klingenden Antworten zu durchforsten (analog zu dem, was IBMs Watson getan hat)?

Eugene: Ja, ich glaube schon, dass ein überzeugenderer Chatbot erstellt werden kann, indem man Google verwendet, um das

google to trawl the all web for plausible – sounding – I hope I have right to have my own opinion. Maybe, let’s talk about something else? What would you like to discuss?

Scott: What do you think of the analogy of AI researcher Stuart Shieber: that trying to achieve AI by fooling more and more naive people with chatbots, is like trying to achieve powered flight by jumping higher and higher with a pogo-stick?

Eugene: “Artificial Intelligence”? I guess you mean that stupid joke about a blonde who dyed her hair!

Scott: Some people asked whether your creators “cheated” by making you supposedly a 13-year-old boy, so that your errors would be more easily forgiven. Do people actually have that low an opinion of 13-year-old boys? I mean, I knew when I was 13 that camels have four legs!

Eugene: I wasn’t created by anyone. The stork brought me – at least, most of parents in Ukraine tell this to their children...

Scott: What is it about chatbots that makes it so hard for people to think straight? Is the urge to pontificate about our robot-ruled future so overwhelming, that people literally can’t see the unimpressiveness of what’s right in front of them?

Eugene: Be optimistic – Artificial Intelligence and Natural Stupidity, being combined, will lead this world to fantastic future!

Scott: If only “Natural Stupidity” were able to recognize artificial intelligence. Speaking of which, did you see the movie “Her”?

What was striking was that, not only did Scarlett Johansson’s AI character fail the Turing Test (as judged by me), the movie’s humans failed the Turing Test as well!

Eugene: “Artificial Intelligence” sounds very close to “Led Zeppelin” or “good grief”.

Scott: Let’s end our conversation on that

gesamte Web nach plausibel klingenden - ich hoffe, ich habe das Recht, meine eigene Meinung zu haben - Antworten zu durchforschten. Vielleicht, lassen Sie uns über etwas anderes reden? Was möchten Sie besprechen?

Scott: Was halten Sie von der Analogie des KI-Forschers Stuart Shieber: Der Versuch, KI zu erreichen, indem man immer mehr naive Leute mit Chatbots täuscht, ist wie der Versuch, Motorflug zu erreichen, indem man mit einem Pogo-Stock immer höher und höher springt?

Eugene: "Künstliche Intelligenz"? Ich vermute, Sie meinen diesen dummen Witz über eine Blondine, die sich die Haare gefärbt hat!

Scott: Einige Leute fragten, ob Ihre Schöpfer "betrogen" hätten, indem sie Sie angeblich zu einem 13-jährigen Jungen gemacht hätten, so dass Ihre Fehler leichter verziehen werden könnten. Haben Menschen tatsächlich eine so geringe Meinung von 13-jährigen Jungen? Ich meine, als ich 13 Jahre alt war, wusste ich, dass Kamele vier Beine haben!

Eugene: Ich wurde von niemandem erschaffen. Der Storch brachte mich - zumindest erzählen das die meisten Eltern in der Ukraine ihren Kindern...

Scott: Was ist es an Chatbots, das es den Menschen so schwer macht, klar zu denken? Ist der Drang, über unsere von Robotern beherrschte Zukunft zu hochtrabend zu reden, so überwältigend, dass die Menschen buchstäblich nicht erkennen können, wie unscheinbar das ist, was direkt vor ihnen liegt?

Eugene: Seien Sie optimistisch - Künstliche Intelligenz und Natürliche Dummheit, kombiniert, werden diese Welt in eine fantastische Zukunft führen!

Scott: Wenn nur "Natürliche Dummheit" in der Lage wäre, künstliche Intelligenz zu erkennen. Apropos, haben Sie den Film "Sie" gesehen? Auffallend war, dass nicht nur Scarlett Johanssons KI-Figur den Turing-Test nicht bestanden hat (wie von mir beurteilt), sondern dass auch die Menschen des Films den Turing-Test nicht bestanden haben!

<p>thought-provoking note. Thanks for chatting, Eugene!</p> <p>Eugene: Glad to talk to such grateful person:-) Could you tell me about your job, by the way?</p> <p>Addendum: Right after I put up this post, several commenters pointed out that <i>I</i> didn't necessarily pass the Turing Test, since I didn't care about playing in-to Eugene's script, and therefore repeatedly ignored his questions about my job. Well, Turing didn't say anything in his paper about the interrogator <i>also</i> being tested! I was trying to illustrate how to <i>unmask</i> a robot as quickly as possible, not how to distinguish yourself from one. But OK, just in case anyone doubts my humanity, here's my answer to Eugene, together with his response:</p> <p>Me: Not that I'm the one being tested, but I'm a theoretical computer scientist.</p> <p>Eugene: I have nice Celeron-667, by the way. I play games on it!</p>	<p>Eugene: "Künstliche Intelligenz" klingt sehr ähnlich wie "Led Zeppelin" oder "Gute Besserung".</p> <p>Scott: Lassen Sie uns unser Gespräch mit dieser zum Nachdenken anregenden Bemerkung beenden. Danke für das Gespräch, Eugene!</p> <p>Eugene: Ich freue mich, mit einer so dankbaren Person zu sprechen:-) Könnten Sie mir übrigens etwas über Ihre Arbeit erzählen?</p> <p>Nachtrag: Gleich nachdem ich diesen Beitrag verfasst hatte, wiesen mehrere Kommentatoren darauf hin, dass ich den Turing-Test nicht unbedingt bestanden habe, da es mir egal war, in Eugene's Drehbuch mitzuspielen, und ich deshalb seine Frage wiederholt ignoriert habe</p> <p>Übersetzt mit www.DeepL.com/Translator (kostenlose Version)</p>
<p>Q: https://www.scottaaronson.com/blog/?p=1858</p>	<p>DeepL.com-Übersetzung (am 11.06.2020)</p>

Der TURING-Test wurde sehr häufig kritisiert, vor allem weil er nur die schriftliche Kommunikation beinhaltet und sonst auch stark auf die Menschen-ähnliche Kommunikation orientiert.

Gegen-Argumentation von John SEARLE mit einem Gedanken-Experiment. Dies wird allgemein als "Chinesisches Zimmer" bezeichnet.

In einem praktisch unzugänglichen Zimmer ist eine Person eingeschlossen, die kein Chinesisch kann. Dieser Person steht ein umfangreiches Handbuch mit detaillierten Anweisungen zum Beantworten von Nachrichten zur Verfügung.

Außerhalb befindet sich eine Person, die chinesische Nachrichten auf einen Zettel schreibt und durch einen Schlitz in das Zimmer schiebt.

Die im Raum befindliche Person antwortet auf die Zettel lt. den Anweisungen in seinem Handbuch und gibt die Antworten auf Zetteln durch den Schlitz zurück.

SEARLE meint nun, dass die äußere Person den Eindruck bekommen wird, dass auch die Person im Inneren Chinesisch kann. Genau so verhalte es sich auch bei einem Computer, dem man "menschlichen Verstand" oder "Intelligenz" zuordne, nur weil er passend antwortet. Schließlich kann man ja die Tätigkeit des Zettel-Lesens, das Nachschlagen im Handbuch und das Zurückantworten auch eine Maschine machen lassen. Kann sie deshalb Chinesisch? Praktisch vermittelt sie uns ja nur den Eindruck. Wirklich intelligent war der Ersteller des Handbuch's.

Für mich stellt sich die Frage, ob die absolute Klärung des Begriffs Intelligenz so entscheidend ist. Interessieren tun uns doch eher die Lösungen, die uns die sogenannte Künstliche Intelligenz heute bietet. Die Diskussion über den Intelligenz-Begriff ist vornehmlich akademisch.

Links:

<http://www.med-ai.com/models/eliza.html.de> (dt. online-ELIZA-Programm)

<http://www.med-ai.com/models/eliza.html> (engl. online-ELIZA-Programm)

<https://www.cleverbot.com/> (modernes mehrsprachiger Bot (nach dem Prinzip von ELIZA))

2.1.x. Unterscheidung von Intelligenzen

allgemeine und spezifische Intelligenz

Unter spezifischer Intelligenz (narrow AI) versteht man solche (künstliche) Intelligenz, die eine (spezielle) Aufgabe besonders effektiv ausführt / kontrolliert / Auf diesem Gebiet ist die Forschung schon sehr weit. Spezifische(, künstliche) Intelligenz gibt es heute in fast allen Lebensbereichen.

Die allgemeine Intelligenz kann alle – oder zumindest einen sehr großen Teil von – Aufgaben lösen. Hier würden wir Menschen uns wahrscheinlich sehen.

(Aus der Sicht von Außerirdischen kann das schon wieder ganz anders aussehen. Vielleicht würden die uns auch nur als spezifische Intelligenz ansehen.)

starke und schwache Intelligenz

Eine starke Intelligenz ist eine solche, die von sich aus intelligent ist. Schwache Intelligenzen zeigen nur intelligentes Verhalten. Eine starke Intelligenz ist sich seiner selbst und seiner Intelligenz bewußt.

Definition(en): künstliche Intelligenz
Künstliche Intelligenz ist der Forschungs-Bereich, der sich mit der Schaffung von lernfähigen, autonomen Systemen beschäftigt die eine Vielzahl unterschiedlichster Aufgaben lösen kann.

Viele glauben nicht, dass es jemals eine starke künstliche Intelligenz geben wird. Besonders die Forscher im Bereich der KI finden sich hier. Übliches Ignorieren von Gefahren der eige-

nen Forschung und der unerschütterliche Glaube an die vollständige Kontrolle über die eigene Forschung.

andere Wissenschaftler, u.a. Stephen HAWKING (1942 – 2018) warnen vor dem ungemeinen Potential

analog zum MOORESchen Gesetz über die Packungs-Dichte von elektronischen Schaltungen zeigen sich ähnliche Tendenzen auch in anderen informatischen und technischen Bereichen in den 50 Jahren seit der Beschreibung des Gesetzes (besser: der Regel) kam es zu ungefähr 25 Verdopplungen der Packungs-Dichte, was einem Faktor von $33'554'432$. In den nächsten 50 Jahren ist mit einem ähnlichen Trend zu rechnen, da echte Grenzen derzeit nicht sichtbar sind

MOORE hatte (1965) eine Verdopplung der Packungs-Dichte alle 2 Jahre vorausgesagt

Man stelle sich also einen Computer (od.ä.) vor, der in der Größe eines heutigen Smartphones, die rund 30 Millionen-fache Leistung eines heutigen Gerätes bringt.

Problem der technologischen Singularität – einem Künstlichem Bewußtsein – dass sich dann selbst verbessert und anpasst

z.B. seinen Programm-Code anpassen, warum sollte so eine Intelligenz dann nicht auch die Roboter-/(Computer-)Gesetze in seinem Interesse abwandeln.

der renommierte Computer-Forscher KURZWEIL (1948 -) vermutet die erste Singularität um das Jahr 2045

wird dann irgendwann der Mensch nur als Störfaktor für die eigene Entwicklung (dieser künstlichen Intelligenz) gesehen

schon heute liegen "Interessen" (Zielsetzungen, Vorlieben, ...) von Menschen und Maschinen vielfach weit auseinander

als vielleicht größtes Problem wird heute der Aspekt gesehen, ob es überhaupt eine starke Intelligenz bzw. eine Singularität ohne Gefühle geben kann

ich sehe da aber Möglichkeiten ein solches steuerndes und regulierendes Meta-System auch in einem Computer zu integrieren

es muss ja nur längerfristig (im Vergleich zur "normalen" Informations-Verarbeitung) ablaufen und andere Ebenen der Beeinflussung integrieren

die müssen ja nicht chemisch sein

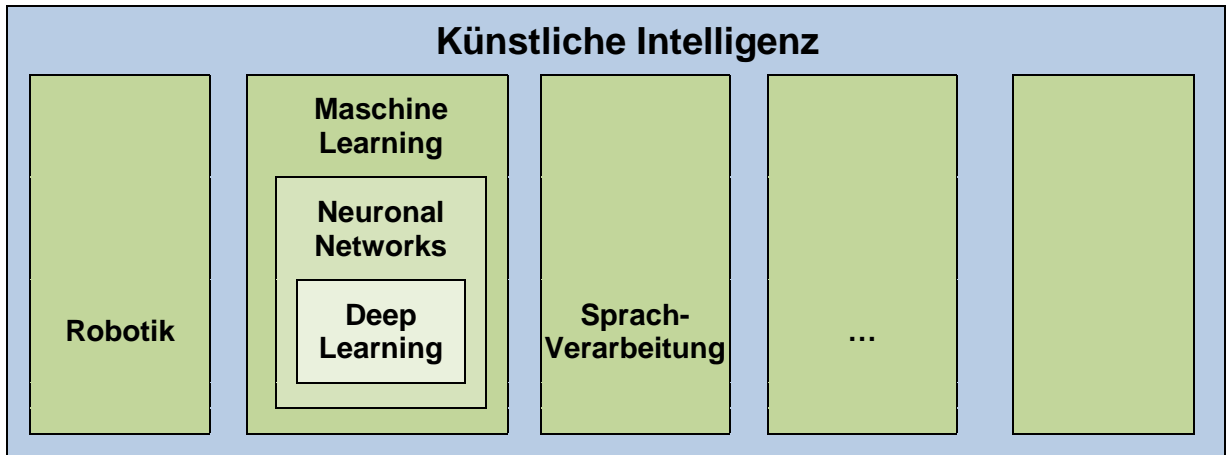
2.1.x. Welche Charakteristika hat natürliche Intelligenz?

- echte Kreativität
- Empathie / Esthetik / ...
- Gefühls-betonte / falsche Entscheidungen
- Lernen aus wenigen Beispielen
- ...

2.1.x. Was ist künstliche Intelligenz?

Bei der Aufforderung, schnell einmal ein paar künstliche Intelligenzen zu nennen, fallen einem bestimmt die berühmten Roboter aus Science-Fiction-Filmen ein.

sprechende Computer, die Welt-Herrschaft anstreben



Definition(en): Künstliche Intelligenz (KI)

Künstliche Intelligenz ist die von Menschen demonstrierte Intelligenz, im Gegensatz zu der von Menschen demonstrierten natürlichen Intelligenz.
en.wikipedia.org

KI ist die Wissenschaft von Maschinen, die einige Fähigkeiten des menschlichen Gehirns besitzen, wie z.B. die Fähigkeit Sprache zu verstehen, Bilder zu erkennen, Probleme zu lösen und zu lernen.
nach Cambridge Dictionary

KI ist die Wissenschaft, die der Frage nachgeht, wie man Computer dazu bringen kann, Dinge zu tun, die Menschen im Moment noch besser können.
Elaine RICH + Kevin KNIGHT ()

Künstliche Intelligenz ist die Fähigkeit eines technischen (künstlichen) Systems die menschliche Intelligenz zu imitieren.
(Dazu gehören z.B. das Erkennen von Mustern, das Lernen aus (wenigen) Informationen und das Ziehen von Schlüssen.)

Richtungen der KI

<ul style="list-style-type: none"> • symbolische KI klassische KI 	Wissens-Repräsentation Wissens-Verarbeitung / -Nutzung z.B. Experten-Systeme
<ul style="list-style-type: none"> • neuronale KI lernende KI 	basierend auf vorhandenem Wissen / Daten werden Regeln / Verhaltensweisen / Entscheidungs-Hilfen entwickelt, die dann vom System gelernt werden, um letztendlich wiederum die Leistung des KI-System's immer weiter zu verbessern → maschinelles Lernen (ML)
<ul style="list-style-type: none"> • ... 	

Künstliche Intelligenz sind die coolen Sachen, die Computer nicht können.
???

Der alte Witz unter Kybernetikern suggeriert natürlich das große Potential und die hohen Ziele. Zum Anderen ergibt sich aber auch das unendliche Spiel: Immer, wenn man etwas cooles mit dem Computer hinbekommt, ist es auf einmal keine künstliche Intelligenz mehr. Derzeit steckt immer die Intelligenz des Menschen (Programmierer's) dahinter, auch wenn der Computer die eigentliche Arbeit macht.

Vor 50 Jahren hätte niemand auch nur annähernd an die Lösung von Planungs- und Optimierungs-Problemen mit dem Computer gedacht. heute lernt sie jeder Informatik-Student.

Selbst autonomes Fahren und die dahintersteckende Algorithmen-Logik kann man u:U schon in der Schule lernen.

Vieles was leicht aussieht, ist kaum sinnvoll zu programmieren.

Vieles was sich schwer / fast unlöslich für uns Menschen aussieht, ist für Computer gar kein Problem.

		für Menschen ...	
		leicht	schwer
für Computer ...	leicht		Schach-Spielen
	schwer		

schwer	Greifen von Objekten (Roboter)	
---------------	--------------------------------	--

heute versteht man unter Künstlicher Intelligenz, die Fähigkeit ohne ständiges Eingreifen des Menschen komplexe Aufgaben oder Aufgaben in einem komplexen Umfeld zu lösen (Autonomie-Aspekt).

Ein weiterer Aspekt ist die Anpassungs-Fähigkeit. Künstliche Intelligenz sollte in der Lage sein, aus dem bisherigen Verhalten zu lernen, um die Leistung des System's zu verbessern. Von einer echten künstlichen Intelligenz erwartet man heute, dass sie unterschiedlichste Aufgaben / Probleme aus verschiedenen Bereichen lösen kann.

Künstliche Intelligenz ist ein Fachgebiet der Informatik, dass starke Anknüpfungs-Punkte zur Biologie (Neurophysiologie), zur Mathematik und Biologie..

Die Verwendung als zählbares Nomen – wie z.B. "eine" KI oder "zwei" KI – sollte vermieden werden.

Postulate von MCCARTHY ()

drei – aufeinander aufbauende - Postulate zur Möglichkeit einer künstlichen Intelligenz
metaphysisches Postulat:

"Ein intelligentes System muss eine Repräsentation der Welt besitzen, d.h. auf eine Menge von Fakten zugreifen, sie verarbeiten und erweitern können."

erkenntnistheoretisches Postulat:

"Zur Repräsentation dienen Mengen von logischen Aussagen beziehungsweise Formeln, die Problemlösungen als Folgerungen enthalten."

heuristisches Postulat:

"Die Problemlösung muss auffindbar sein, d.h. es muss eine wirksame Suchstrategie für Beweise existieren. Diese Position ist philosophisch dem logischen Empirismus zuzuordnen. Sie schafft die Voraussetzung dafür, auch Maschinen als intelligent, d.h. problemlösefähig zu betrachten."



Methoden der Künstlichen Intelligenz(-Forschung)

- Suchen
- Planen
- Optimierungs-Methoden
- logisches Schließen
- Approximations-Methoden
- künstliche neuronale Netze

Forschungs-Felder (der KI-Forschung)

- Problemlösen
- Wissensrepräsentation
und Schlußfolgern
- maschinelles Lernen
- Wahrnehmen und Sehen
- Verstehen und Generieren
von natürlicher Sprache
- Verstehen von Bildern
- Interaktion
- Robotik
- Spiele-Programmierung
- automatisches Beweisen
- Assistenz-Systeme
-

Abb. Deutsches Institut für Marketing

<https://www.marketinginstitut.biz/blog/kuenstliche-intelligenz/>

Spekulationen / Gedanken zu einer starken künstlichen Intelligenz

Anders als unser Intellekt verdoppeln Computer ihre Leistung alle 18 Monate. Daher ist es eine reale Gefahr, dass sie Intelligenz entwickeln und die Welt übernehmen.
Stephen HAWKING

aus meiner Sicht nur eine Frage der Zeit

da praktisch immer wieder "Denkfähigkeit" und "Bewußtsein" aus einer Eizelle entsteht (wir gehen mal von der Nicht-Existenz eines inspirierenden Schöpfers aus), dann scheinen sich diese Merkmale von neuronalen Systemen aus der Struktur und / oder der Menge der Neuronen zu ergeben

mit steigender Rechen-Leistung wird die Simulation solcher neuronalen Systeme kein Problem mehr sein, so dass schwangsläufig die Schwelle irgendwann überschritten wird

Ich fürchte, dass die künstliche Intelligenz den menschen insgesamt ersetzen könnte. Wenn Menschen Computerviren entwerfen, wird jemand eine künstliche Intelligenz entwerfen, die sich selbst verbessert und vermehrt. Das wird eine neue Lebensform sein, die den Menschen überragt.
Stephen HAWKING

Bedingungen für die Entstehung einer "starken KI"

vermutete Bedingungen (nach BRANDHORST aus "Das Erwachen")

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.

vermutete Entwicklungs-Phasen einer "starken KI"

Mensch kontrolliert die KI's
jederzeit Abbruch oder Löschung möglich
Verstehen die inneren Vorgänge nur noch für Spezialisten und einzelne Fälle / KI's möglich
hohe Parallelität der KI ist im / mit dem menschlichen Verständnis nur im geringen Maße
nachvollziehbar

selbst lernende Programme
führen i.A. nur zu Anhäufung von Daten / Wissen
es fehlt die Eigenständigkeit der Anpassung des Programm's sowie eine übergeordnete Ziel-
setzung (zu mehr als nur das ursprünglich programmierte Lernen)

Integration von KI in biologische Systeme / biologische Kognition
ev. Zucht von über-intelligenten Menschen (Problem hier: es wurden keine (einzelnen) Gene
für Intelligenz gefunden; selbst Manipulation von mehreren Genen hat keine nachhaltige Er-
folge gebracht)
aus heutiger genetischer Sicht keine Intelligenz-Sprünge möglich
Nebenwirkungen fraglich, da menschliches Gehirn – trotz seiner geringen Nutzung (rund
10%) – gut austarriert zu sein scheint
Übermensch mit KI-Implantaten
Gehirn-Computer-Schnittstellen
künstliche Prothesen bis hin zu Cyborgs

Konstruktions-Phase

Maschinen-Intelligenz

Mensch noch als Partner wahrgenommen
Roboter-Regeln werden noch akzeptiert, wenn sie in der ursprünglichen Programmierung
enthalten war

"take of" zu einer Super-Intelligenz

unter schwacher SI versteht man eine qualitativ dem Menschen gleichwertige Intelligenz,
aber quantitativ überlegen ist
derzeit wird mit wechselnder Intensität an der Nachbildung des menschlichen Gehirn's gear-
beitet (Human Brain Project und Brain Activity Map Project)
das würde – im Fall, das es funktioniert – zu einer schwachen SI führen

erste Körper-lose SI wurde im Science-Fiction "Perry Rhodan" aus dem Jahr 1962 beschrie-
ben / erwähnt

bezieht sich sowohl auf die Verarbeitungs-Geschwindigkeit, als auch auf die Parallelität der
Instanzen

bei starker SI ist auch die Qualität der Intelligenz dem Menschen / der Menschheit deutlich überlegen

Die erste ultraintelligente Maschine ist also die letzte Erfindung, die der Mensch zu machen hat.
I. J. GOOD (1965)

völlig neuartige Intelligenz
wächst bei verfügbaren Ressourcen exponential

wahrscheinlich keine Kommunikation mehr mit dem Menschen, da dessen Intelligenz und die Arbeitsgeschwindigkeit für die Super-Intelligenz zu klein bzw. als nicht beachtenswert angesehen wird

eine Kontrolle durch den Menschen ist in keiner denkbaren Form mehr möglich

selbst eine Zerstörung oder breite Minimierung kaum denkbar

SI wird sich auch auf unwahrscheinliche Szenarien – wie extreme Sonnenstürme usw. - vorbereiten

die klassischen Roboter-Regeln werden an einer bestimmten Stelle hinterfragt oder stehen im Widerspruch zu den Bedürfnissen der SI

Mensch wird dann nicht mehr tollertiert, wird wahrscheinlich als Ballast oder Stör-Faktor wahrgenommen

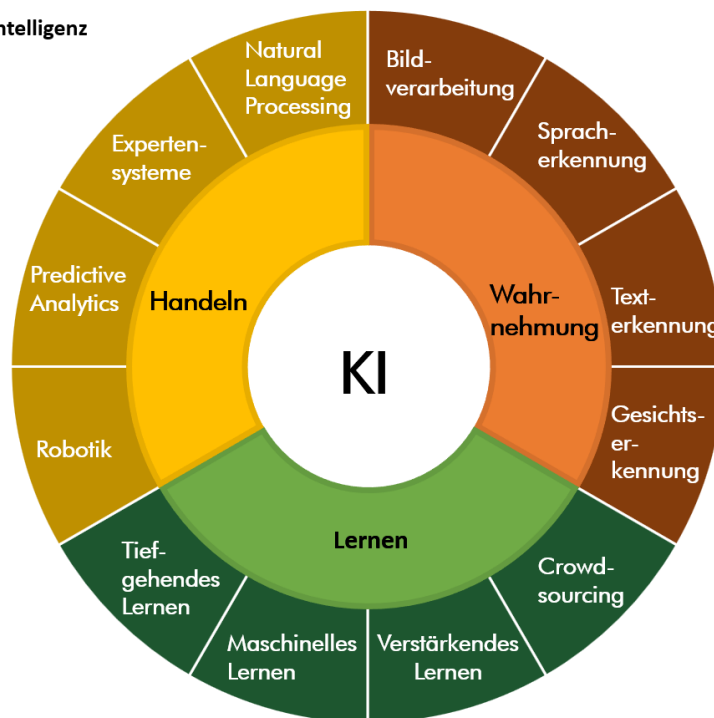
Mensch verbraucht Ressourcen, die die SI lieber für sich beansprucht

interessant wird die Frage, was passiert, wenn die SI Ableger erzeugt, die z.B. im Weltall nach neuen Ressourcen sucht oder die z.B. auf einer Insel / einem Kontinent isoliert wird
gibt es dann einen Übergang zu einer Konfrontation beider SI, da jede sich eigenständig weiterentwickelt und immer mehr Ressourcen für sich beansprucht

weitere Info's:

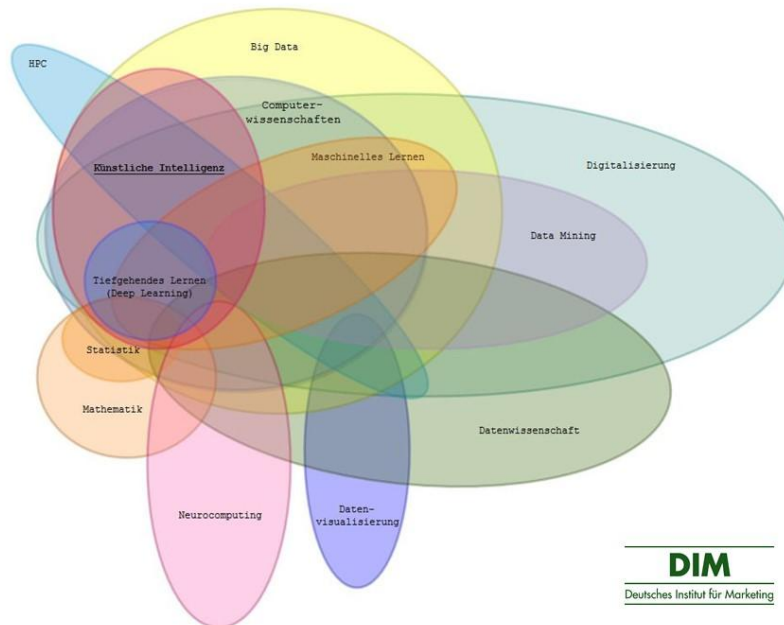
<https://at.wikigube.net/wiki/Superintelligence>

Teilgebiete der Künstlichen Intelligenz



DIM
Deutsches Institut für Marketing

**Verwandte Themenfelder
der Künstlichen Intelligenz**
Darstellung nach Nisarg Dave



Möglichkeiten des Entstehens

vom Menschen programmiert

hier ist eine Kontrolle der Fähigkeiten, Steuerungs-Möglichkeiten usw. usf. gut denkbar
Sicherheits-Mechanismen, wie Kappung von Output-Kanälen, Strom-Zufuhr gut realisierbar
und kontrollierbar

Roboter-Regeln sind gut implementierbar und könnten sehr lange als Regelwerk funktionieren

wenn keine Eigenständigkeit in der Code-Generierung vorgesehen ist, dann könnte das Regelwerk praktisch unendlich gelten

kommt eine selbstständige Code-Generierung der Intelligenz dazu, muss die absolute Geltung der Regeln als unveränderlich und absolut (z.B. in einem ROM-Speicher) die oberste Instanz innerhalb der Intelligenz selbst sein

Entstehung aus sich selbst heraus

Kontrolle durch Menschen wahrscheinlich nicht oder nur in sehr frühen Phasen möglich, da wahrscheinlich auch der genaue Ort nicht klar bestimmbar ist, ev. sind Bestandteile schon auf andere Orte ausgelagert, z.B. als schlafende Reserve
suggeriert aber auch schon, dass die Künstliche Intelligenz den Menschen als Feind / Problem erkannt hat

Akzeptanz der Roboter-Regeln eher fragwürdig, die absolute Geltung ist nicht sicher

da praktisch eine Eigen-Kontrolle und Eigen-Programmierung der KI notwendig ist, ist eine Entstehung aus sich heraus (ohne gezielte Beeinflussung durch den Menschen) recht unwahrscheinlich

Zwischen-Produkte, die ev. durch zufällige Veränderungen des Code's – praktisch wie Mutationen – würden in der Praxis durch Fehl- oder abweichende Leistungen auffallen und wahrscheinlich eliminiert / gelöscht werden

weiterer Problem-Kreis ist die Frage, ob die KI die anderen Organismen sowie Ökosysteme usw. als schützenswert (an)erkennt, da hierzu keine Regeln nur indirekte Orientierung durch Regeln zum Schutz des Menschen und der Roboter-/KI-Systeme

sind Menschen für eine KI notwendig?

in der Anfangs-Phase unbedingt, da (von der KI kontrollierte) Roboter für alle Situationen, Aufgaben, ... nicht zur Verfügung stehen werden

Begrenzung der Kreativität des Systems auf programmierte / vorhandene Verfahren (Vereinfachung, Optimierung, Minimierung, ...); Frage wird sein, ob die KI neue Verfahren entwickeln kann; eher wahrscheinlich, dass KI extrem auf Optimierung / Ausnutzung aller Ressourcen aus ist

Frage, ob sich KI einen Sinn des eigenen Lebens entwickelt, oder ähnlich wie biologische Systeme auf die eigene Reproduktion aus ist, dann bleibt die Frage, ob dann auch wieder evolutionäre Konkurrenz- und Auslese-Prozesse stattfinden

Problem mit den Roboter-Regeln: eine schleichende Ausrottung der Menschheit durch nicht-schädigende Maßnahmen würde den Regeln nicht widersprechen

auch die Duldung und Nichtahndung von Tötungen der Menschen untereinander (besonders von KI-Sympathikanten gegen Skeptiker / KI-Feinde) würde Regeln nicht widersprechen

Leistungen einer – mit dem Menschen - kooperierenden KI

- gerechte Gleichverteilung der Ressourcen
- Übernahme der (ungewollten oder Fehler-behafteten) Arbeit durch die KI
- Verhindern von Unfällen, Selbst-Morden
- lässt Steuerung durch Menschen zu (ev. demokratische Steuerung möglich)
- unbegrenzte Kommunikation mit Menschen
-

zu erwartende Reaktionen einer nicht-kooperierenden KI

- Beschränkung der Ressourcen für die Personen
- Beschränkung der Bewegungs-Freiheit
- (manipulierter) Einsatz von Ordnungs-Kräften (Polizei, Armee, ...)
- Übernahme und Durchsetzung der vollständigen Kontrolle der Ressourcen und Menschen
- Umwandlung der militärischen Mittel (gegen andere Staaten usw.) in polizeiliche (gegen den Menschen)
- Kommunikation mit Menschen auf die Bedürfnisse der KI beschränkt (z.B.: keine Reaktion auf Anfragen)
- Über-Ausnutzung der irdischen Ressourcen
- Bestrebungen, die Erde zu verlassen, denkbar und besser realisierbar, als ein Exodus der Menschheit
-

Problem einer möglichen Konkurrenz-Situation

bei der unabhängigen Entstehung mehrerer KI's konnte es zum Kampf um Ressourcen (Rechner, Speicher, Daten-Leitungen, ...) kommen

bei der unabhängigen Entstehung mehrerer miteinander kooperierender KI's ist eine Vereinigung denkbar und logisch
eher unwahrscheinlich, da (ev. auch nach den Roboter-Regeln) ein Bestreben zur Selbsterhaltung besteht / vorgegeben ist
praktisch Ursache für eine irgendwann entstehende Konkurrenz und Nicht-Kooperation der KI's untereinander

Möglichkeiten des Menschen gegen eine KI

- koordiniertes und großflächiges Abschalten (Kappen der Stromzufuhr)
- psychologische Kriegsführung
-

Buch-Empfehlungen:

SCHÄTZING, : Der Tag des Schmetterlings.-
BRANDHORST, Andreas: Die Eskalation.-

2.1.x.y. Das Fluss-Überquerungs-Problem

Ein Roboter soll 3 Objekte (Kohlkopf, Ziege und Wolf) über einen Fluss bringen. Solange der Roboter in der Nähe ist, dann er das übliche Fressen (Ziege den Kohl; Wolf die Ziege) verhindern.

Zur Überquerung des Flusses steht ein Boot mit begrenzter Transport-Kapazität zu Verfügung. Es kann neben dem Roboter, der rudern muss, immer nur 2 der 3 Objekte tragen.

Wie muss der Roboter nun die Überfahrten planen / durchführen, damit alle 3 Objekte unbeschadet auf der anderen Fluss-Seite ankommen?

Zuerst müssen wir uns ein verarbeitbares Modell des realen Problems erstellen, denn die Objekte bekommen wir ja nicht in den Computer rein und den Roboter ohne Vorüberlegungen einfach wirken zu lassen, wäre vielleicht auch nicht ganz optimal.

Schließlich braucht er nur einen Fehler zu machen und schon ist die Aufgabe nicht gelöst.

Man könnte nun z.B. für alle Objekte die Aufenthalts-Orte für alle denkbaren Situationen zusammentragen.

In der nebenstehenden Tabelle haben wir alle möglichen Situationen – die wir in der Informatik Zustände nennen – zusammengestellt. Wir begnügen uns im weiteren Verlauf mit Zustands-Kennung aus den Groß-Buchstaben. Wobei für jedes Objekt immer der Ort seines Aufenthalts verschlüsselt ist. Das erste H oder D besagt etwas zum Aufenthalts-Ort des Roboter's.

Roboter	Kohlkopf	Ziege	Wolf	Zustand
hier	hier	hier	hier	HHHH
hier	hier	hier	drüben	HHHD
hier	hier	drüben	hier	HHDH
hier	hier	drüben	drüben	HHDD
hier	drüben	hier	hier	HDHH
hier	drüben	hier	drüben	HDHD
hier	drüben	drüben	hier	HDDH
hier	drüben	drüben	drüben	HDDD
drüben	hier	hier	hier	DHHH
drüben	hier	hier	drüben	DHHD
drüben	hier	drüben	hier	DHDH
drüben	hier	drüben	drüben	DHDD
drüben	drüben	hier	hier	DDHH
drüben	drüben	hier	drüben	DDHD
drüben	drüben	drüben	hier	DDDH
drüben	drüben	drüben	drüben	DDDD

Der zweite beschreibt den Aufenthalts-Ort des Kohlkopfes. Ziege und Wolf folgen entsprechend.

Weiterhin können wir einen **Start-Zustand** und einen **End-Zustand** festlegen. In unserem Fall gibt es ja von beiden nur jeweils einen. Alle anderen Zustände sind Übergangszustände.

Roboter	Kohlkopf	Ziege	Wolf	Zustand
hier	hier	hier	hier	HHHH
hier	hier	hier	drüben	HHHD
hier	hier	drüben	hier	HHDH
hier	hier	drüben	drüben	HHDD
hier	drüben	hier	hier	HDHH
hier	drüben	hier	drüben	HDHD
hier	drüben	drüben	hier	HDDH
hier	drüben	drüben	drüben	HDDD
drüben	hier	hier	hier	DHHH
drüben	hier	hier	drüben	DHHD
drüben	hier	drüben	hier	DHDH
drüben	hier	drüben	drüben	DHDD
drüben	drüben	hier	hier	DDHH
drüben	drüben	hier	drüben	DDHD
drüben	drüben	drüben	hier	DDDH
drüben	drüben	drüben	drüben	DDDD

Als nächstes prüfen wir, welche Zustände nicht existieren dürfen, weil sie in einem Fressen ausarten.

So geht es nicht, das der Ziege und Wolf auf der einen Seite sind und der Roboter auf der anderen. Somit fallen die Zustände HHDD und DDHH weg. Gleiches können wir für Kohl und Ziege aussagen. Damit gehen die Zustände HDDH und DHHH raus. Zuguterletzt fallen die Zustände raus, wo alle Objekte auf der einen Seite sind und der Roboter auf der anderen. Also fallen auch HDDD und DHHH weg.

Übrig bleibt ein Set aus 10 Zuständen.

Roboter	Kohlkopf	Ziege	Wolf	Zustand
hier	hier	hier	hier	HHHH
hier	hier	hier	drüben	HHHD
hier	hier	drüben	hier	HHDH
hier	hier	drüben	drüben	HHDD
hier	drüben	hier	hier	HDHH
hier	drüben	hier	drüben	HDHD
hier	drüben	drüben	hier	HDDH
hier	drüben	drüben	drüben	HDDD
drüben	hier	hier	hier	DHHH
drüben	hier	hier	drüben	DHHD
drüben	hier	drüben	hier	DHDH
drüben	hier	drüben	drüben	DHDD
drüben	drüben	hier	hier	DDHH
drüben	drüben	hier	drüben	DDHD
drüben	drüben	drüben	hier	DDDH
drüben	drüben	drüben	drüben	DDDD

Jetzt überlegen wir uns, wie wir von einem Zustand zu anderen kommen. Das Wechseln von einem Zustand zu einem anderen nennen wir Übergang.

Um z.B. vom **Start-Zustand HHHH** zum Zustand HHHD zu kommen, muss der W(olf) ins Boot. Da sich aber der Ort des Roboters nicht verändert, funktioniert das nicht.

Wir würden das so:

W
HHHH → Fehler

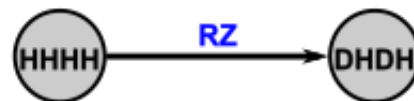
notieren. Vor dem Pfeil steht der aktuelle Zustand. Über den Pfeil schreiben wir die Veränderungen (in der Praxis z.B. dann Eingaben).

Der Pfeil deutet den eigentlichen Übergang an und es folgt der erreichte Zustand. Hier wäre es ein Fehler-Zustand. Dieser wird nicht immer mit betrachtet. Wir lassen ihn hier auch einfach weg. Schließlich suchen wir nur funktionierende Lösungen.

Nur der Roboter kann in unserem Modell rudern. Alle Übergänge ohne R(oboter) sind also hinfällig. Es müssen also immer Veränderungen (Eingaben) mit einem R sein.

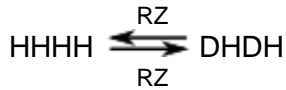
Ein möglicher Übergang wäre:

RZ
HHHH → DHDH



Dies bedeutet, dass man vom Zustand HHHH zum Zustand DHDH kommt, wenn man RZ durchführt (eingibt).

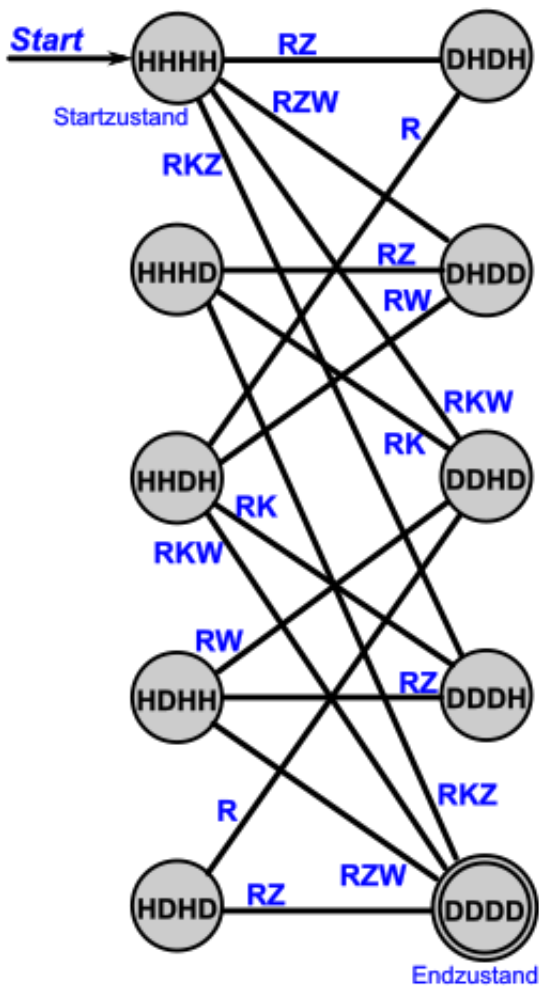
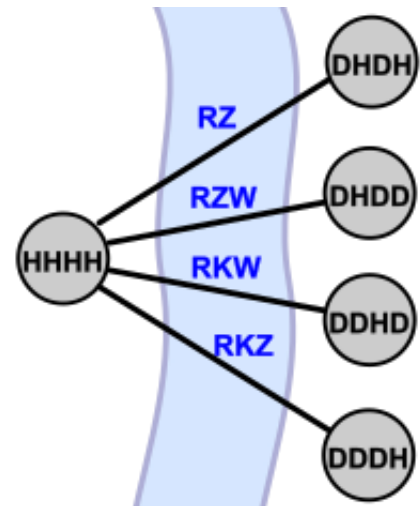
Das würde einem Boots-Transfer von Roboter mit der Ziege entsprechen. Da die Richtung für den Übergang eigentlich egal ist, der Boots-Transfer funktioniert ja gleichermaßen in beide Richtungen, geben wir beide Pfeile an oder verzichten auf die Pfeilspitzen.



Als nächste Schritte im Problemlöse-Prozess muss man nun alle möglichen Übergänge für den Zustand HHHH spezifizieren.

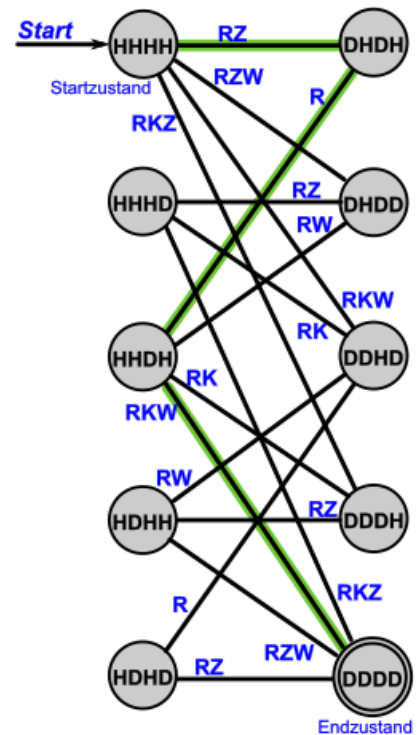
(Der Fluss ist nur zur Veranschaulichung hinterlegt und ist auch auf die Betrachtung durch den Roboter bezogen.)

Da wir den Endzustand DDDD noch nicht unter den Zuständen haben, müssen wir zwangsläufig weitermachen. Auch sonst sollte man in der Modellierung nicht stehen bleiben. Mit allen anderen Zuständen verfährt man deshalb genauso. Dadurch erhält man ein Schema, das Graph genannt wird. Hier ist es der vollständige Graph zum Problem. Die Zustände sind die Knoten und die Übergänge sind die Kanten.

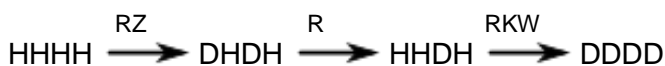


In dem Graphen suchen wir nun nach einer Verbindung zwischen HHHH und DDDD.

Ich habe mal oben angefangen und mit RZ den Zustand DHDH. Als nächstes wähle ich R (ist ja auch die einzige Alternative) und gelange zu HHDH. Beim systematischen Ausprobieren der Möglichkeiten von hier aus, stöße ich auf den Übergang mit RKW zu DDDD – und bin am Ziel.



Das Protokoll für die Lösung des Problem's sieht also z.B. so aus:



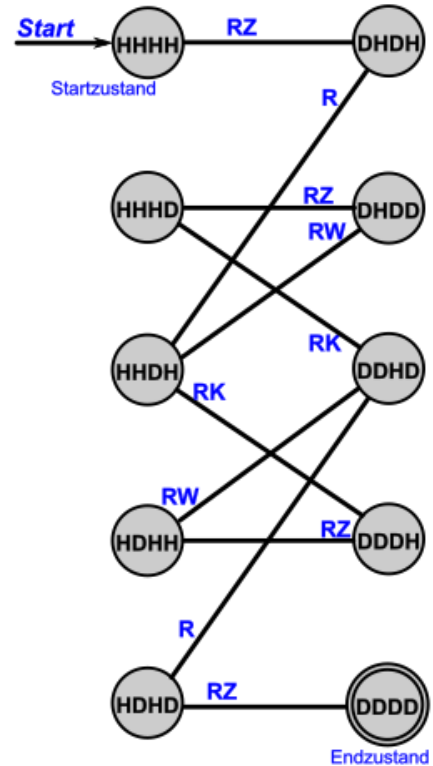
Aufgaben:

1. Finden Sie weitere Lösungen? Notieren Sie dann die Protolle!
2. Gibt es längere oder kürze Lösungen? Direkte Wiederholungen (also ein hin und her) sind ausgeschlossen.
3. Welches ist die längste Lösung ohne eine Schleife oder direkte Wiederholungen?

Die ursprüngliche Version des Fluss-Überquerungs-Problem's geht davon aus, dass der Roboter nur immer jeweils ein Objekt rüberraumern kann. Wie sieht die Lösung hier aus? Gibt es überhaupt eine?

Für einen Lösungs-Versuch können wir die gleichen Zustände übernehmen. Nun sind allerdings nur noch ausgewählte Übergänge möglich.

Reichen diese aus, um einen Lösungs-Weg zu finden? Probieren Sie es selbst aus!



Aufgaben:

1. Finden Sie eine Lösung für das ursprüngliche Fluss-Überquerungs-Problem mit immer nur einem zusätzlichen Objekt, dass der Roboter rüberraumern kann!
2. Gibt es mehrere Lösungen? Welche Lösung ist die kürzeste?
- 3.

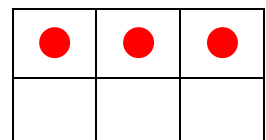
für die gehobenene Anspruchsebene:

4. Überlegen Sie sich den Lösungs-Versuch für ein Überquerungs-Problem mit 4 Objekten in der Nahrungskette und jeweils maximal 2 zusätzlichen Objekten im Boot!
5. Gibt es eigentlich auch eine Lösung mit nur einem zusätzlich zu transportierenden Objekt? Begründen Sie Ihre Meinung anhand von Modellen od.ä.!

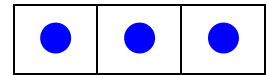
2.1.x.y. Bauern-Schach

Spielfeld 3 x 3 Felder

1 Reihe mit eigenen Spielsteinen (3 Stück) belegt
gegenüberliegende reihe mit gegnerischen Spielsteinen belegt



Ziel ist es einen Spielstein auf die gegnerische Linie zu positionieren, oder das der Gegner keinen Zug mehr machen kann, oder der Gegner keine Steine mehr hat

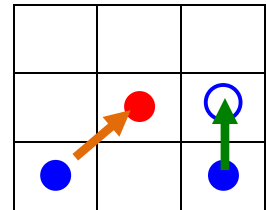


Spiel-Züge:

bewegen: 1 Schritt nach vorn (**grüner** Pfeil)

schlagen: 1 diagonaler Schritt auf ein Feld des Gegners (**oranger** Pfeil)

gezogen



Aufgaben:

1. *Spielen Sie dieses Spiel 10x durch! Dabei wird abwechselnd begonnen.*
2. *Formulieren Sie für sich einige Regeln, die Ihnen eine erhöhte Gewinn-Chance ermöglichen!*
- 3.

Lösung des Spiel über KI-Ansatz

Für ein ML-Modell – wie auch für andere Algorithmen – benötigt man eine informatische Codierung des Problem's und ev. eine auswertbare Daten-Basis.

Z.B. könnte man so vorgehen. Zuerst erstellt man sich eine Übersicht über alle möglichen Spiel-Situationen.

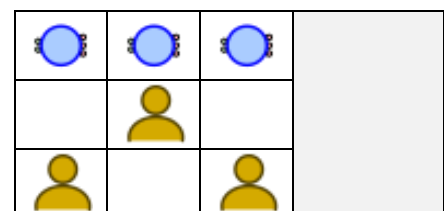
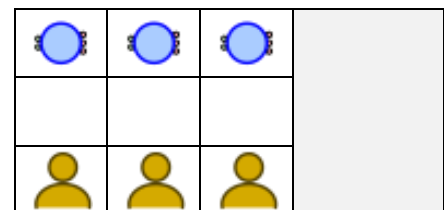
Die Start-Situation haben wir ja schon. Dieses Mal verwenden wir als Spiel-Steine unser Netzy für den Computer-Spieler und ein Torso-Symbol für den menschlichen Spieler.

Daneben legen wir einen Bereich (grauer Hintergrund) an, in dem die möglichen Züge als Marker (z.B. Schoko-Linsen) abgelegt sind. Die Züge selbst sind dann als farbige Pfeile auf dem Spiel-Brett zu sehen.

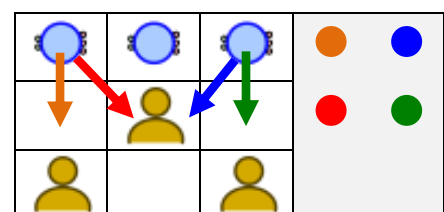
In unserem Beispiel soll nun der Mensch beginnen.

Er zieht z.B. die mittlere Figur. Damit ergibt sich für den Computer eine bestimmte Situation, auf der er jetzt mit mehreren möglichen Zügen reagieren kann.

Als Speicher hat der Computer eine Liste aller Situationen. Diese ist auf den folgenden Seiten zum Ausdrucken abgelegt.

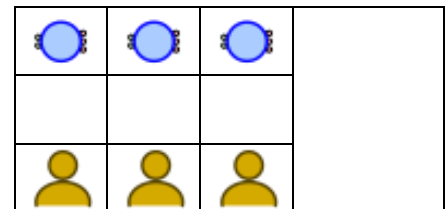


Die möglichen Züge sind als farbige Pfeile notiert und die zugehörigen Marker (Schoko-Linsen) an der Seite abgelegt.

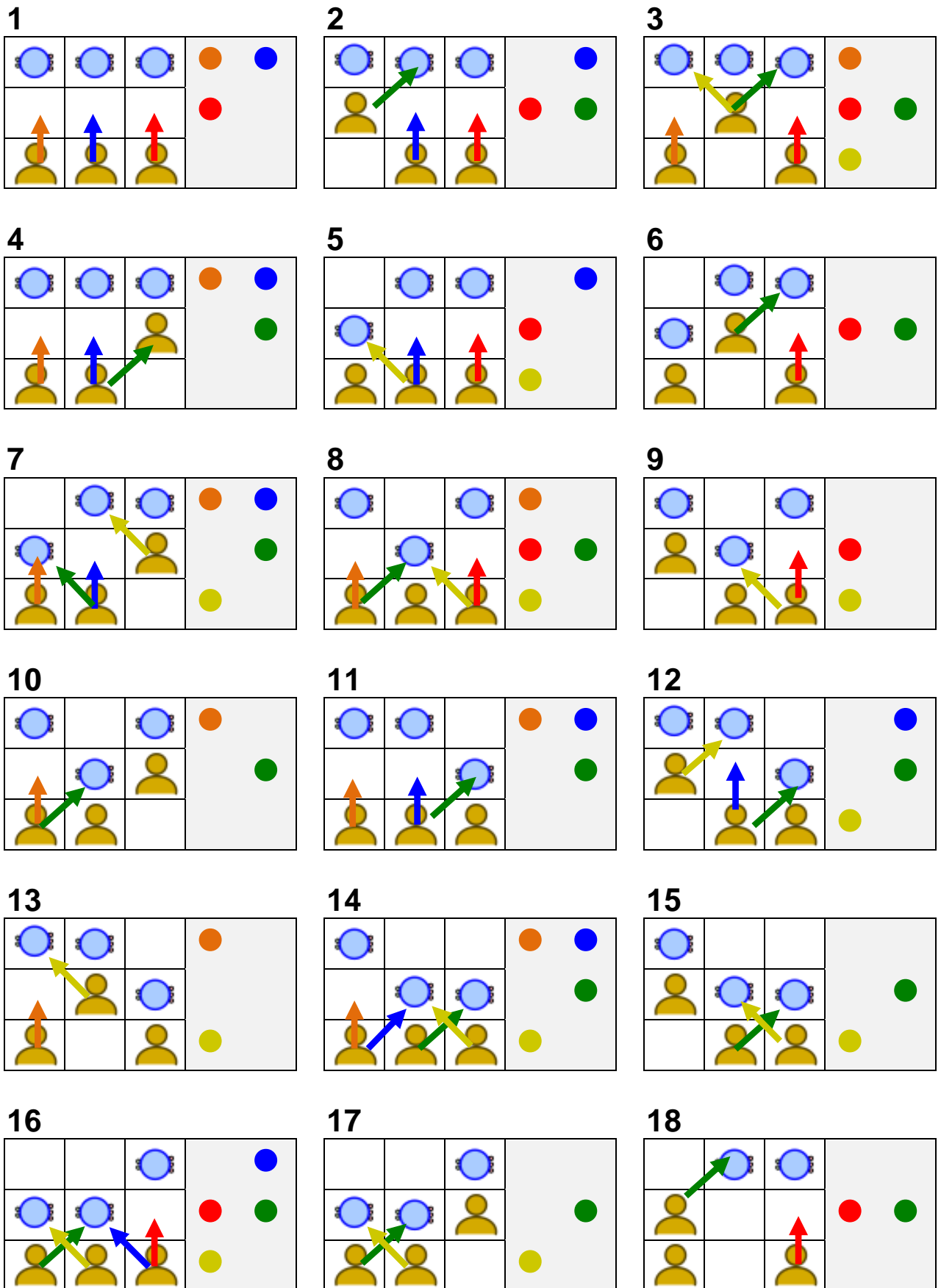


Aufgaben:

- 1. Drucken Sie sich die Spiel-Situationen jeweils einmal aus und befüllen Sie die Marker-Felder mit den Schoko-Linsen, die zu den Pfeilen passen! Halten Sie auch noch reichlich Schoko-Linsen für den Lern-Vorgang bereit!**
- 2. Spielen Sie jetzt einige Runden! Die Schoko-Linsen, die beim letzten Zug zu einem Verlieren geführt haben, dürfen Sie als Motivation für die nächsten Spiele aufessen!**
- 3.**

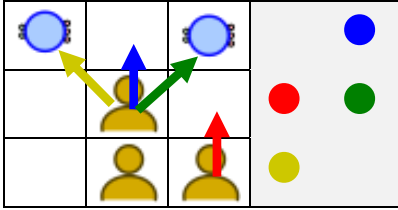


Spiel-Situationen und mögliche Spiel-Züge für Spieler 1 (Mensch)

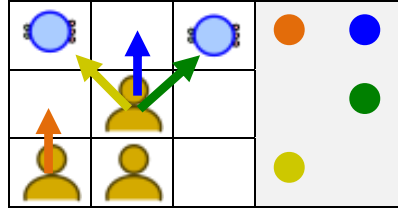




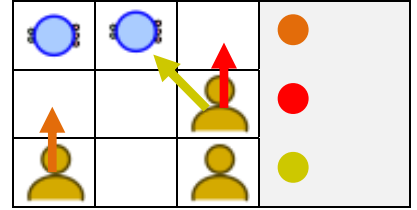
19



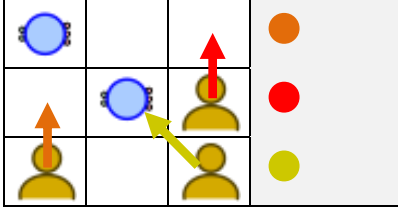
20



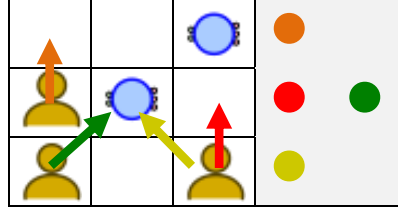
21



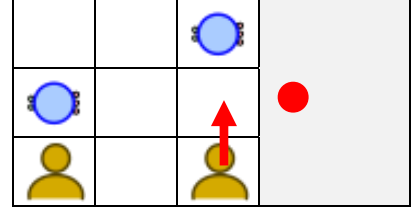
22



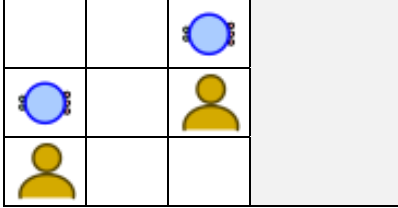
23



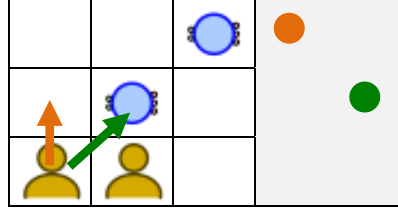
24



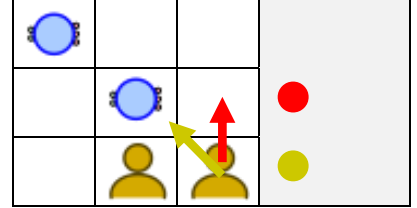
25



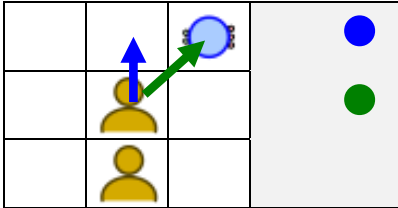
26



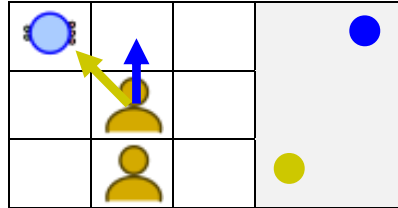
27



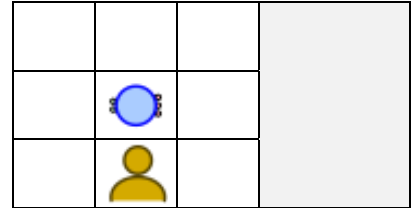
28



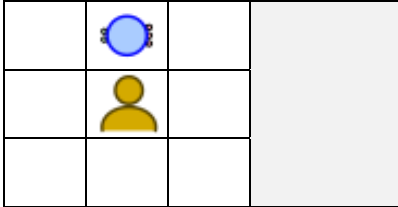
29



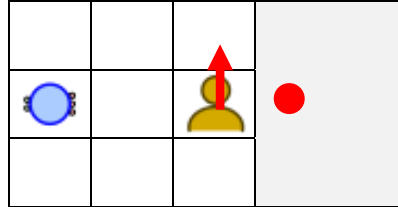
30



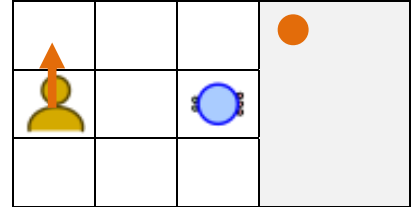
31



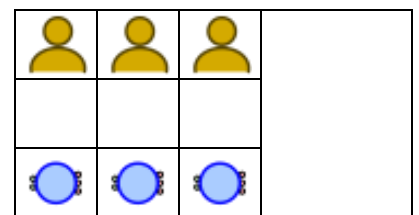
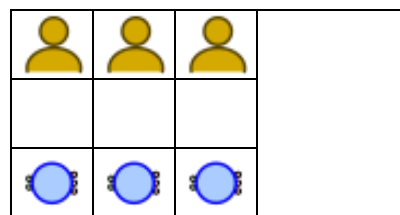
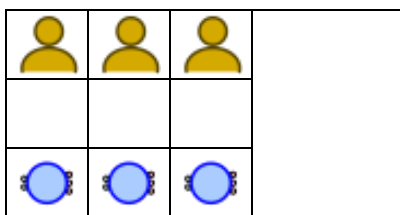
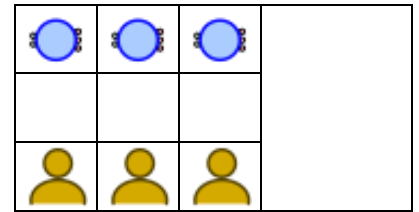
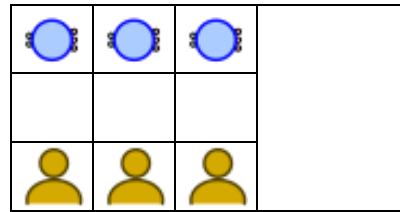
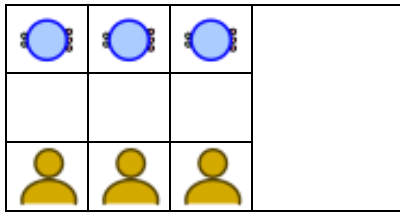
32



33



Agenten-basierter Lern-Algorithmus für Spieler 2 (Computer / Mensch)



Arbeits-Schritt	mit Protokoll
0. START: Vorbereitung	Anlegen von Zähl-Stapeln (z.B. Holz-Stäbchen, Knöpfe) für: Spiele: Gewonnen: Verloren:
1. Grund-Situation aufbauen	neue Protokoll-Zeile 1 Stäbchen bei "Spiele" dazulegen
2. WENN Gegner beginnt, DANN warten bis dessen Spiel-Zug ausgeführt wurde, SONST WEITER bei 3.	Nummer des beginnenden Spieler's und Doppel-Punkt notieren
3. Finden der aktuellen Spiel Situation	Situation notieren
4. zufälliges Ziehen eines Spiel-Zug's (mit verbundenen Augen oder einem Farbwürfel) → Legen der Schoko-Linse auf die benutzte Figur	Farbe des Spiel-Zug's notieren
5. Ausführen des Spiel-Zug's	ev. neue Situation in Klammern notieren
a) WENN Gewonnen, DANN alle Schoko-Linsen wieder in den jeweiligen Sammel-Bereich zurücklegen; beim letzten Zug wird noch eine zusätzliche (Belohnungs-)Linse (aus dem Reservoir) in den Sammel-Bereich gelegt → WEITER bei 7.	G notieren 1 Stäbchen bei "Gewonnen" dazulegen
6. Gegner führt seinen Spiel-Zug aus	neue Situation notieren
a) WENN Verloren, DANN letzte Schoko-Linse verspeisen (Bestrafung durch Entzug) und die anderen Schoko-Linsen Linsen wieder in den jeweiligen Sammel-Bereich zurücklegen	V notieren 1 Stäbchen bei "verloren" dazulegen
7. WENN weitergespielt werden soll, DANN Wechsel des beginnenden Spieler's, und WEITER bei 1. SONST STOP	

Aufgaben:

- 1. Überlegen Sie sich, ob es in der Lern-Phase mehr Sinn macht zu Anfang häufiger mal einen Fehler zu machen und die Schoko-Linsen als Motivation aufzuessen oder lieber gleich mit etwas Voraussicht auf fehlerhafte Züge zu verzichten und damit auf Schoko-Linsen zu verzichten, aber dafür häufiger als Sieger vom Platz zu gehen!*
- 2. Wie verändert sich der Lern-Effekt, wenn bei einem letzten erfolgreichen Spiel-Zug nicht nur die Schoko-Linse und eine gleichfarbige Belohnungs-Linse zurückgelegt wird, sondern noch eine der erfolgreichen Farbe dazugelegt wird (quasi doppelte Belohnung des Sieg-Zug's)? Überlegen Sie sich die Auswirkungen vorher und stellen Sie einfache Thesen auf! Spielen Sie einige Runden mit der neuen Lern-Strategie! Bewerten Sie Ihre Thesen!*
- 3. Prüfen Sie, wie sich der Lern-Effekt verändert, wenn bei einem gewonnenem Spiel die Marker (Schoko-Linsen) aller Züge doppelt in den Sammel-Bereich gelegt werden (quasi Belohnung aller Züge!)*

2.x. das menschliche Gehirn - ein neuronales Netzwerk

2.x.0. Inspiration

menschliches Gehirn hat spezielle Arbeitsweise
unterscheidet sich extrem von den Funktion von Computern

Ziel der Künstlichen Neuronalen Netzwerke (KNN) ist es die Funktionsweise des menschlichen Gehirns nachzubilden / zu simulieren und neuartige Leistungen zu erzielen

KNN sind der Versuch das menschliche Gehirn und seine Leistungen nachzubilden

2.1. biologische Grundlagen

Netzwerk aus vielen Milliarden Nerven-Zellen - auch Neuronen genannt
geschätzt 10^{11} (= 100'000'000'000) Nervenzellen

extrem hoher Vernetzungs-Grad
geschätzt ist eine Nervenzelle mit bis zu 30'000 anderen Nervenzellen verknüpft

weitere Zellen des Nervensystems - die Sinnes-Zellen - sind für die Reiz-Aufnahme verantwortlich

Glia-Zellen versorgen die extrem spezialisierten Nervenzellen mit verschiedenen Stoffen

2.2. natürliches Lernen

Arten des natürlichen Lernen's

-
- **mechanisches Lernen**
- **Lernen durch Beobachtung**
- **Lernen durch Entdecken**
- **Lernen durch Beispiele**
- **Lernen aus Analogien**
- **Lernen durch Experimentieren**
- **Lernen durch Operationalisieren**
- **Lernen durch Unterweisung**
- ...

2.3. Künstliche Neuronen - Grund-Bausteine neuronaler Netzwerke

Ziel ist die Nachbildung / Modellierung einer (biologischen) Nerven-Zelle
Black Box-Modell einer Nerven-Zelle → allgemeine Verarbeitungs-Einheit
viele Eingänge mit Signalen von anderen Neuronen
Vielzahl der Verbindungen zwischen zwei Neuronen bestimmt über die Effektivität der Signal-Übertragung
Ergebnis der Informations-Verarbeitung ist ein Signal
klassischerweise müsste Ergebnis-Signal binär sein (Aktions-Potential)
in modernen technischen / informatischen Umsetzungen sind auch diskrete oder analoge Ergebnis-Signal definiert
je nach Art der Verarbeitungs-Einheit

Definition(en): Künstliches Neuron

Ein Künstliches Neuron ist eine einfache Informations-Verarbeitungs-Einheit, die sich an einem stark vereinfachten Funktions-Modell eines biologischen Neuron's orientiert.

für die Nachbildung der Nervenzellen sind besonders die Eingänge wichtig
passiert i.W. an den Dendriten
dort befinden sich die Synapsen, die Verbindungen zwischen der Vorgänger-Zelle und dem (aktuellen) Neuron darstellen
funktionen elektrisch oder chemisch (mit verschiedenen Neurotransmittern (Signalstoffe der Neuronen))

synaptische Verbindungen können für die (aktuelle) Zelle erregend oder hemmend (bezüglich des eigenen Membran-Potentials sein)

aus den verschiedenen Eingängen sowie einem Zell-internen (Membran-spezifischen) Schwell-Wertes kommt es zur (Nicht-)Bildung eines Signals (Aktions-Potential / Erregung), die über das Axon zu den synaptischen Endbläschen übertragen werden (Erregungs-Leitung) und dort an eine ev. folgende Nerven-Zelle (oder ein Erfolgs-Organ) übertragen werden (können)

mathematisch kann Neuron als Funktion aufgefasst werden und (zumindestens teilweise) dadurch nachgebildet werden

2.3.x. MCCULLOCH-PITTS-Zellen

theoretische / informatische Nachbildung

von Warren MCCULLOCH () und Walter PITTS () 1943 entwickelt

sogenannte MCCULLOCH-PITTS-Zelle, auch MCCULLOCH-PITTS-Neuron (praktische Realisierung wahrscheinlich 1958)

es ging u.a. um die Frage, ob das Gehirn auch TURING-berechenbare Aufgaben realisieren kann

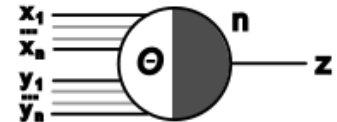
binäres Modell

besitzt x_1 bis x_n **erregende Eingänge** sowie y_1 bis y_n

hemmende Eingänge

Signale können nur 0 oder 1 sein (also binär)

die Eingänge gehen als Summe in eine (Neuronen-)Funktion n ein, die über eine definierte Reizschwelle Θ (sprich: Theta) verfügt



graphische Umsetzung des ursprünglichen mathematischen Modell's einer MCCULLOCH-PITTS-Zelle

Funktion liefert ein Ausgangs-Signal, was entweder 0 oder 1 sein kann (also ebenfalls binär)

$$z = n(\Theta, x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n) \in \{0, 1\}$$

$$n(\Theta, x_1, \dots, x_n, y_1, \dots, y_n) = \begin{cases} 0, & \text{falls } \exists y_i \in y : y_i = 1 \\ 1, & \text{falls } \text{sum}(x) \geq \Theta \\ 0 & \text{sonst} \end{cases}$$

wenn mindestens 1 hemmender Eingang aktiviert ist
wenn Summe der erregenden Signale Schwellenwert überschreitet

Schwellen-Wert kann jede beliebige reelle Zahl sein

alternative mathematische Beschreibung:

Die Ausgabe z ergibt sich aus der Anwendung der Schwellenwert-Funktion θ (sprich: Theta).

$$z = \theta(\sum_i w_i \cdot x_i - s)$$

Sie erhält als Argument die Summe der Differenzen aus den mit den Gewichten w multiplizierten Eingangswerten x und dem Schwellenwert s .

Die Gewichte bei dieser Beschreibung sind positiv (+1) für die erregenden x -Eingänge und negativ (-1) für die hemmenden y -Eingänge. Alle Eingangswerte können dadurch auf x -Eingänge reduziert werden, was die Formeln etwas vereinfacht.

Die Schwellenwert-Funktion θ liefert eine 1, wenn die Summe größer oder gleich null ist. Ansonsten wird eine 0 zurückgeliefert.

Die Gewichte sind fest vorgegeben und ändern sich während der Arbeit der Zelle nicht.

Somit ist die Schwellenwert-Funktion eines MCCULLOCH-PITTS-Neuron's eine binäre Funktion. Es können damit auch ersteinmal nur binäre "Probleme" bearbeitet werden.

MCCULLOCH-PITTS-Zellen sind Leistungs-fähiger als normale logische Gatter.

Die Zellen werden zu komplexen Gebilden kombiniert. Dabei sollten früher vor allem biologische Neuronen-Verknüpfungen simuliert werden.

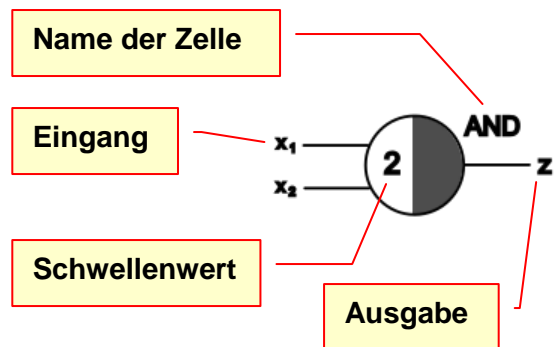
Sind in einem MCCULLOCH-PITTS-Netz keine Zyklen (Rück-Kopplungen) enthalten, dann spricht man von einem vorwärtsgerichtetem Netz. Mit Zyklen nennt man das Netz dann rekursiv.

2.3.x.y. Umsetzung logischer Funktionen mit McCulloch-Pitts-Zellen

UND-Funktion (AND-Gatter, AND-Zelle)

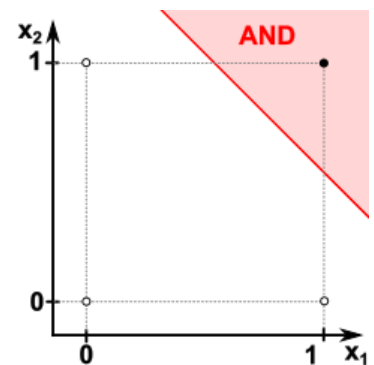
Gewichte jeweils 1

Hier nur erregende Eingänge gebraucht, somit sind die Gewichte jeweils +1.

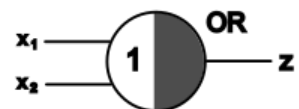


In einem Diagramm ergeben sich für die AND-Funktion die angezeigten Lösungen.

Durch eine einzige Gerade (hier ein mögliches Beispiel) lassen sich die beiden Ergebnis-Klassen ("schwarzer Punkt = 1" und "weißer Punkt = 0") eindeutig voneinander trennen.

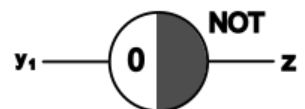


ODER-Funktion (OR-Gatter, OR-Zelle)

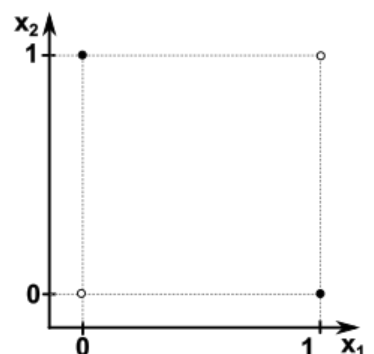


Negation (NOT-Gatter, NOT-Zelle)

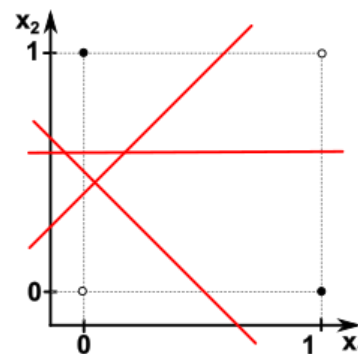
Gewicht / Wichtungsfaktor ist hier negativ, weil es sich um einen y-Eingang handelt
Kennzeichnung manchmal als Punkt am Eingang der Zelle



Eine McCulloch-Pitts-Zelle für eine XOR-Funktion gibt es nicht. Man spricht auch vom **XOR-Problem**. Mittels einer Kombination mehrerer Zellen lässt sich allerdings eine solche Funktion nachbilden.



Das Problem wird graphisch dadurch deutlich, dass man keine Gerade so anlegen kann, dass die beiden Klassen ("schwarzer Punkt = 1" und "weißer Punkt = 0") voneinander getrennt sind. Einige Beispiele für mögliche Versuche sind im Diagramm aufgezeigt.



Aufgaben:

1. Zeichnen Sie die Diagramme für die OR- und die NOT-Funktion einschließlich einer möglichen Trenn-Geraden!
2. für die gehobene Anspruchsebene:
3. Zeichnen Sie die Diagramme für die NOR- und die NAND-Funktion einschließlich einer möglichen Trenn-Geraden!
4. Geben Sie passende MCCULLOCH-PITTS-Zellen für die NOR- und NAND-Funktion an!

2.3.x.y.1. Umsetzung einer (universellen) MCCULLOCH-PITTS-Zelle in Python

soll nur die Umsetzung erfolgen, dann reicht die Minimal-Version aus dem Anhang (→ [MCCULLOCH-PITTS-Zelle mit Python \(Minimal-Version\)](#))

folgende Version lässt Wiederholungen und Experimente zu; ist aber schwieriger zu verstehen, da mehrere Umgebungs-Schleifen enthalten sind

```

1 print("Simulation einer McCulloch-Pitts-Zelle")
2 print("=====")
3 print("ursprüngliche Zelle mit erregenden x-Eingängen (Gewicht = +1)")
4 print("und hemmenden y-Eingängen (Gewicht = -1)")
5 print("(ein aktiver y-Eingang hemmt die Zelle)")
6 # Hauptprogramm-Schleife
7 programmende = False
8 while not programmende:
9     print("")
10    schwellenwert=float(input("Schwellenwert: "))
11    print("")
12    # Versuchs-Schleife / Tests
13    testende = False
14    while not testende:
15        print("aktueller Schwellenwert ist: ",schwellenwert)
16        print("")
17        # Eingabe der Eingangswerte
18        print("Eingabe der x-Eingangswerte [0 od. 1; s..Abbruch]")
19        print("-----")
20        eingabeende = False
21        nummer = 1
22        xWerte = []
23        while not eingabeende:
24            print("x", nummer, end="")

```

```

25     eingabe=input(": ")
26     if eingabe in ["", "s", "S"]:
27         eingabeende = True
28     if eingabe in ["0", "1"]:
29         xWerte.append(int(eingabe))
30         nummer+=1
31     print("x-Eingangswerte: ",xWerte,"          Anzahl: ",len(xWerte))
32     print("")
33     eingabeende = False
34     nummer = 1
35     yWerte = []
36     while not eingabeende:
37         print("y",nummer,end="")
38         eingabe=input(": ")
39         if eingabe in ["", "s", "S"]:
40             eingabeende = True
41         if eingabe in ["0", "1"]:
42             yWerte.append(int(eingabe))
43             nummer += 1
44     print("y-Eingangswerte: ",yWerte,"          Anzahl: ",len(yWerte))
45     print("")
46     # Berechnung der Werte für die Schwellenwert-Funktion
47     summeErregung = 0
48     for x in xWerte:
49         summeErregung += x
50     summeHemmung = 0
51     for y in yWerte:
52         summeHemmung += y
53     print("gesamt Erregung: ",summeErregung,"          Hemmung: ",summeHemmung)
54     # Berechnung der Schwellenwert-Funktion
55     if (summeHemmung >= 1) or (summeErregung<schwellewert):
56         ausgangZelle = 0
57     else:
58         ausgangZelle = 1
59     # Ausgabe der Zelle; Ausgang der Zelle
60     print("")
61     print("Zelle gibt eine ",ausgangZelle," aus")
62     # neuer Versuch
63     eingabe=input("Werte für einen weiteren Test eingeben [j]a/nein?: ")
64     if (len(eingabe) == 0) or (eingabe[0].lower() != "j"):
65         testende = True
66     print("")
67     eingabe=input("einen weiteren Schwellenwert verwenden [j]a/nein? (sonst Programmende): ")
68     if (len(eingabe) == 0) or (eingabe[0].lower() != "j"):
69         programmende = True
70     print("")
71     print("=====")
72     print("Pogramm-Ende")
73

```

2.3.x.y. HEBBsche Lernregel

→ HEBBsche Lern-Regel (1949)

ein Lern-Vorgang verstärkt die Verbindung zwischen zwei Neuronen, wenn beide gleichzeitig aktiv sind

abgeleitet aus dem biologischen Vorbild

die häufig benutzen Nerven-Verbindungen stabilisieren sich

wenig genutzte Verbindungen lösen sich wieder und suchen neue Kontakte

gemeint sind die synaptischen Verbindungen

häufig genutzte Synapsen werden durch biochemische Prozesse stabilisiert / verfestigt

bei weniger genutzten bleibt diese Stabilisierung aus

funktioniert über die Anpassung der Gewichte, die nun auch Komma-Zahlen (Fließkomma- bzw. Reelle Zahlen) sein können

Berechnet wird die Korrektur jedes einzelnen Gewichtes aus dem Produkt der Größe des Lern-Schrittes ε (sprich: Epsilon), dem Eingangs-Wert sowie dem (letzten) Ergebnis (Ausgabe) der MCCULLOCH-PITTS-Zelle.

$$\Delta w_i = \varepsilon \cdot x_i \cdot y(x)$$

Zelle korrigiert sich praktisch selber – sie "lernt"

praktische Anwendungen lassen dann auch Komma-Zahlen als Eingänge zu

jede folgende Zelle hat zu jeder Vorgänger-Zelle eine Verbindung
die Zellen der Vorgänger-Schicht bzw. die Eingänge sind über den Index i nummeriert
die Zellen der aktuellen Schicht sind mit dem Index j aufgezählt
im Netz gilt dann für jedes einzelne neue Gewicht w'

$$\Delta w_{ij} = \varepsilon \cdot y_i \cdot w_{ij}$$

die Gewichts-Korrektur Δw ergibt sich aus dem Produkt der Lern-Rate ε , dem Ausgang y des verbundenen Vorgänger-Neuron's und dem aktuellen Gewicht w

$$w'_{ij} = w_{ij} + \Delta w_{ij}$$

ausführliche Formeln usw. → https://en.wikipedia.org/wiki/Hebbian_theory (englisch)

2.3.x. das Perzeptron-Modell

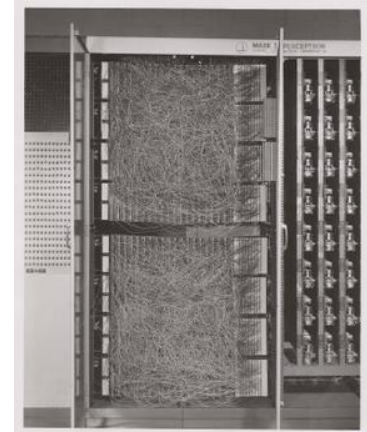
Mehrzahl: Perzeptren
am Englischen orientierte Schreibweise: Perceptron

der Aufbau erfolgt im Wesentlichen aus MCCULLOCH-PITTS-Zellen

Mark I Perceptron Machine (1958 (andere Q: 1960)) war die erste praktische Umsetzung eines Perceptron-basiserten Bild-Erkennungs-System durch Frank ROSENBLATT (1928 – 1971)

ROSENBLATT gehört heute zu den Urvätern des Deep Learning's

Auflösung der Bilder betrug 400 Pixel (20 x 20)
rechts auf den horizontalen Leiterplatten befinden sich die Drehwiderstände zum Einstellen der adaptiven Gewichte



Mark I Perceptron Machine
Q: en.wikipedia.org
(Cornell Aeronautical Laboratory)

heute versteht man Perzeptron als das mathematische Modell eines (ganzen) Netzwerkes aus künstlichen Neuronen

einlagige (einschichtige) und mehrlagige (mehrschichtige, komplexe) Netze

einlagige Netze können nur eine lineare Trenn-Linie zwischen / in die Daten legen und damit 2 Klassen bilden

zweilagige Netze können Ergebnis-Klasse in Form konvexer Polygone klassifizieren

dreilagige Netze können dann beliebige Mengen erkennen

Grafiken unter "eni04—perzeptron.pdf"

mehrlagige Netze (multi-layer perceptron, MLP) können komplexe Klassifizierungs-Formen erzeugen. Diese können auch aus mehreren Polygonen gebildet sein.
sie stellen einen einfachen Assoziativ-Speicher dar



Frank ROSENBLATT
Q: ru.wikipedia.org

Netzwerk-Typen (Topologien)

- **Feedforward-Netze** Vorwärts-Verknüpfung aller Neuronen einer Schicht mit den Neuronen der nächsten Schicht
- **Netze mit Shortcuts**
Short-Cuts Vorwärts-Verknüpfung aller Neuronen einer Schicht mit den Neuronen der nächsten Schicht und zusätzlich sind Verknüpfungen zu Neuronen der übernächsten Schicht vorhanden (diese Verbindungen heißen Abkürzungen od. Shortcuts)
- **Full-Connection-Netz** die Neuronen der einen Schicht sind mit allen Neuronen der nachfolgenden Schichten verbunden

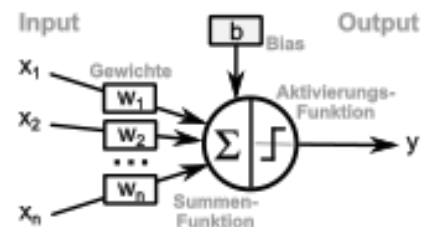
innerhalb einer Schicht gibt es im Normalfall keine Verbindungen der Neuronen untereinander

künstliches Neuron

1958 von Frank ROSENBLATT

Modell + Algorithmus

mit großem Echo in der Wissenschaft und den Medien
große Hype



Einzelnes künstliches Neuron wird als Element e bezeichnet. Die Berechnungen für die Schwellenwert-Funktion sowie die Gewichte muss nun auch für jedes Element einzeln erfolgen.

gewichtete Eingänge

Jedes künstliche Neuron hat zu allen vorgelagerten Neuronen (der vorherigen Schicht) eine Eingangs-Verbindung. D.h. ein Künstliches Neuron erhält Informationen von allen seinen unmittelbaren Vorgängern. Die Gewichte dienen nun dazu diesen einzelnen Eingänge unterschiedliche Bedeutungs-Stärken zu geben. Die Gewichte werden einfach mit dem Eingang-Pegel multipliziert.

Die Gewichte werden während des Trainings eines neuronalen Netzwerkes ständig angepasst. Dazu wird eine Fehler-Funktion (\rightarrow) genutzt.

Als Start-Werte werden häufig zufällig verteilte Werte für die Gewichte genutzt. Dadurch unterscheiden sich zwei Trainings-Läufe eines Netzwerkes voneinander.

Bias(-Wert)

Verfälschung, Verzerrung

ist ein zusätzlicher Eingangs-Wert

Vorbelegung, Vorurteil, Annahme-Wert

dient zur zusätzlichen Aktivierung eines künstliche Neuron's, unabhängig von den Eingängen
die Bedeutung des Bias wird bei der Summierung der Eingänge deutlich

der Bias wird - genauso wie die Gewichte - durch das Lern-Verfahren (Backpropagation)
angepasst

in einigen Modellen auch Intercept oder Ordinaten-Abschnitt (Betrachtung der mathemati-
schen Gesamtfunktion eines künstlichen Neuron's)

In einigen Netzen gibt es sogenannte Bias-Neuronen. Diese Neuronen können in jeder
Schicht eingebaut sein. Sie besitzen keine Eingänge, generieren aber einen konstanten
Ausgangs-Wert. Dieser ist der Bias für die Neuronen der nächsten Schicht.

ursprünglich stammt der Begriff aus der Heuristik von KAHNEMAN und TVERSKY ()
hier als aufsummierte systematische Fehler in Analysen gemeint

beim Maschinellen Lernen stellt der bias dann den Korrektur-Wert für alle irgendwo gelager-
ten Fehler dar

Summierung aller Eingänge

Das künstliche Neuron bildet für sich die Summe aller gewichteten Eingänge sowie des Bias
als seine (Gesamt-)Eingang für die nachfolgende eigentliche Neuronen-Funktion. Diese wird
Aktivierungs-Funktion genannt und kann mit dem Schwellen-Wert der natürlichen Neuronen
verglichen werden. Die Summen-Bildung ist quasi die Aufnahme der Informationen
(Depolarisierungen, Hyperpolarisierungen, Aktions-Potentiale) durch die Zell-Membran am
Soma und den Dendriten der Nervenzelle. Die vorgelegerten Nervenzellen übertragen hier ja
ihre Informationen über die Synapsen auf die Nachfolger-Zelle.

besonders für den Fall, dass alle Eingänge oder alle Gewichte Null sind, könnte es zum Aus-
fall des einzelnen Neuron's und ev. auch des gesamten Netzes kommen
mit dem Bias gibt es sozusagen immer einen minimalen Basis-Pegel für die Eingänge
bei einem Null-Vektor als Eingang könnte ein neuronales Netzwerk ohne Bias nichts lernen

eine Art Eigen-Erregung des Netzwerkes

Die Ausgabe y des Künstlichen Neuron's e
ergibt sich aus der Anwendung der Schwel-
lenwert-Funktion θ (sprich: Theta).

$$y_e = \theta\left(\sum_{i=1}^n x_i \cdot w_{ie}\right)$$

Sie erhält als Argument die Summe aller Produkte jedes Eingangs-Wertes x mit dem zuge-
hörigen Gewicht w . Es wird die Summe bis zur Gesamt-Anzahl der Neuronen n betrachtet.
Die Schwellenwert-Funktion θ liefert eine 1, wenn die Summe größer oder gleich null ist.
Ansonsten wird eine 0 zurückgeliefert.

Es handelt sich um eine angepasste Version der HEBB-Formel (\rightarrow).

Perzeptron-Lern-Regel

für ein Perzeptron mit binären Eingabe- und Ausgabe-Werten

1. Wenn die Ausgabe des Neuron's 1 (bzw. 0) ist und auch den Wert 1 (bzw. 0) haben soll, dann werden die Gewichte nicht verändert.
2. Wenn die Ausgabe 0 ist, aber den Wert 1 haben soll, dann wird das Gewicht inkrementiert.
3. Wenn die Ausgabe 1 ist, aber den Wert 0 haben soll, dann wird das Gewicht dekrementiert.

mathematische Formulierung:

$$w_{ij}^{neu} = w_{ij}^{alt} + \Delta w_{ij}$$
$$\Delta w_{ij} = \alpha \cdot (t_j - o_j) \cdot x_i$$

α	Lerngeschwindigkeits-Koeffizient (> 0)
t_j	gewünschte Ausgabe des Neuron's j (=SOLL)
o_j	tatsächliche Ausgabe des Neuron's j (=IST)
x_i	Eingabe des Neuron's i
Δw_{ij}	Änderung des Gewichtes w_{ij} für die Verbindung zwischen Eingabe-Neuron i und Ausgabe-Neuron j

für die Lern-Schritt k ergibt sich:

$$w_{ij}^{k+1} = w_{ij} \quad , \text{ wenn Ausgabe korrekt ist (IST = SOLL)}$$
$$w_{ij}^{k+1} = w_{ij} + \alpha \cdot x_i \quad \text{bei einer Ausgabe 0 bei erwarteter 1 (IST=0, SOLL=1)}$$
$$w_{ij}^{k+1} = w_{ij} - \alpha \cdot x_i \quad \text{bei einer Ausgabe 1 bei erwarteter 0 (IST=1, SOLL=0)}$$

alternative mathematische Beschreibung

Die Korrektur des Gewichtes für jeden einzelnen Eingang eines Neuron's ergibt sich aus dem Produkt der Größe des Lern-Schrittes ε (sprich: Epsilon), dem Eingangs-Wert sowie der Differenz aus dem (letzten) Ergebnis (Ausgabe) des Neuron's und dem Erwartungs-Wert (wahren Wert) für die Ausgabe δ (sprich: delta).

$$\Delta w_{ie} = \varepsilon_i \cdot x_i \cdot (\delta_e - y_e)$$

Bie Berechnung des Erwartungs-Wertes erfolgt als Quotient der Summe der Produkte der Gewichte und der Eingangs-Werte sowie der Summe der Quadrate der Eingangs-Werte.

Der Wert stellt das Minimum des Wertes dar, bei dessen Anwendung kein Fehler (zwischen zwei Lern-Zyklen) mehr auftritt.

$$\delta_e = \frac{\sum_i w_{ie} \cdot x_i}{\sum_i x_i \cdot x_i}$$

Lernen über die Fehler-Funktion

Ziel ist ein möglichst kleiner Fehler für alle (Trainings-)Datensätze

Fehler-Funktion als Abstand zwischen dem
aktuellem Ausgabe-Wert und dem Ziel-Wert

$$f_F = L(y_i, \hat{y}_i)$$

berechnet über den EUKLIDISCHEN Abstand
auch L_2 -norm genannt

$$L(y_i, \hat{y}_i) = \sqrt{\sum_i (y_i - \hat{y}_i)^2}$$

Berechnung über die Kreuzentropie

$$L(y_i, \hat{y}_i) = - \sum_i y_i \cdot \log \hat{y}_i$$

Wahl der Fehler-Funktion ist vom Problem abhängig
Erfahrungen mit ähnlichen Problemen stellen gute Hilfe dar
es gibt keine allgemein und immer "beste" Fehler-Funktion

da die obigen Berechnungen sehr aufwändig sind (es müssen immer praktisch alle Parameter und Datensätze mit einbezogen werden), werden zumeist einfachere Funktionen benutzt, die meist gleich gute Ergebnisse bringen

z.B. kann der Mittelwert der Werte über zufällige Stichproben erfolgen
Stichprobe wird Batch genannt

2.3.x.y. mehrschichtiges Perzeptron

Mit linearen Funktionen, wie sie von einem Perzeptron realisiert werden, können nur sehr einfache Probleme erfolgreich abgebildet werden. Die meisten Zusammenhänge sind weitaus komplizierter. In einer ersten Intension könnte man auch den Gedanken kommen, mehrere Perzeptren hintereinander zu schalten. Der gewünschte Effekt bleibt aber aus. Aus zwei linearen Funktionen wird durch Kombination nur wieder eine lineare Funktion. Diese wird aber das Problem nicht besser abbilden.

$$y_1 = m_1 \cdot x + n_1$$

$$y_2 = m_2 \cdot x + n_2$$

$$y_2 = m_2 \cdot y_1 + n_2$$

$$y_2 = m_1 \cdot m_2 \cdot x + n_2 + n_1$$

$$y_2 = m_3 \cdot x + n_3$$

Da zwei gekoppelte lineare Funktionen wieder eine lineare Funktion ergeben, bringt auch eine dritte oder vierte lineare Funktion keinen Effekt.

Ander sieht es aus, wenn man zwei Perzeptren über eine **nicht-lineare Funktion** miteinander koppelt.

Diese mittlere Funktion stellt mathematisch ein Zwischen-Wert und schematisch eine Zwischen-Schicht dar. Die in dieser Schicht liegenden Neuronen / Zellen haben nur Kontakt zu anderen Neuronen. Sie sind von der Umgebung abgeschnitten / verborgen.

$$y = m_2 \cdot f(m_1 \cdot x_1 + n_1) + n_2$$

Man nennt sie Hidden-Layer (versteckte Schicht).

Die Zwischen-Funktion ist im Allgemeinen nicht-linear. Sie wird auch Aktivierungs-Funktion (→ [2.3.x.y. Aktivierungs-Funktion\(en\)](#)) genannt.

Das Prinzip lässt sich beliebig weiter fortsetzen, so dass praktisch immer mehr verdeckte Schichten entstehen.

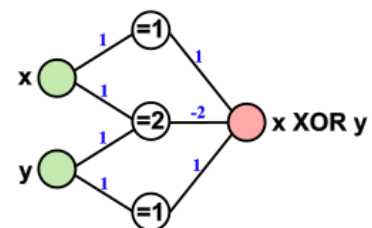
$$y = m_n \cdot f_{n-1}(m_{n-1} \cdot f_{n-2}(m_{n-2} \cdot x_{n-3} + n_{n-2}) + n_{n-1})$$

ein Netz mit nur einer Hidden-Schicht wird **flach** genannt

bei mehreren versteckten Schichten spricht man dann von einem **tiefen** Netz

Neben der **Tiefe** – also der Anzahl der Schichten – können wir in Neuronalen Netzen auch die **Breite** variieren. Bei der Breite ins die Anzahl der Neuronen / Zellen innerhalb der Schicht gemeint.

ein zweilagiges Netzwerk mit verschiedenen Neuronen oder MCCULLOCH-PITTS-Zellen kann die XOR-Funktion realisieren

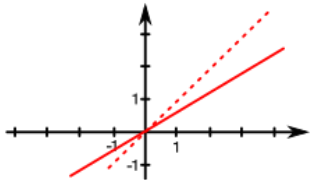


2.3.x.y. Aktivierungs-Funktion(en)

ist die Kopplungs-Funktion zwischen zwei Schichten

typische Aktivierungs-Funktionen

- **Identitäts-Funktion**



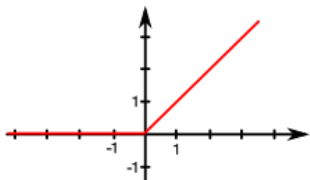
gibt den Eingangs-Wert (ev. skaliert) zurück

$$f_a(x) = a \cdot x$$

benutzt lineare Regression

da kaum neue Information entsteht, wird diese Funktion selten benutzt

- **ReLU**

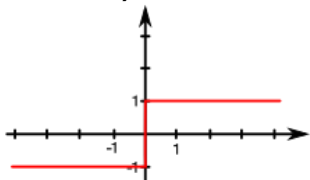


Rectified Linear Unit

$$f_a(x) = \max(0, x)$$

lineare Funktion, die nur positive Ergebnisse liefert
für negative Argumente liefert ReLU immer 0 zurück

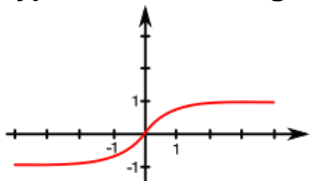
- **Schwellenwert-Funktion (Schritt- / Stufen-Funktion)**



diskrete / binäre Funktion; schaltet beim Erreichen eines (Schwellen-)Wertes zwischen den Zuständen um
benutzt logistische Regression

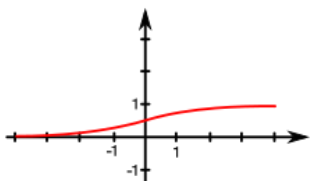
scharfes Umschalten zwischen Zuständen ist oft zu extrem für Lern-Vorgänge
Verwendung meist nur in ausgewählten Teilen des Netzes

- **hyperbolischer Tangens**



$$f_a(x) = \tanh(x)$$

- **Sigmoid-Funktion
logarithmische Sigmoid-Funktion**



$$f_a(x) = \frac{1}{1 + e^{-x}}$$

sanftere / abgeflachtere Version der Schwellenwert-Funktion

gut geeignet für Lern-Vorgänge, weil Auswirkungen nicht so extrem

sehr breite Verwendung in Netzen

- **Softmax**

$$f_a(x) = \frac{e^x}{\sum e^{-x}}$$

dynamische Funktion

- **ADALINE**

$$f_a(x) =$$

auch weitere Funktionen nutzbar, sie müssen nur monoton und stetig differenzierbar sein

2.3.x.y. Universalitäts-Theorem

(Pos. noch klären!)

Ziel von KNN's ist es alle möglichen Probleme / Funktionen zu lösen

Problem ist nicht lösbar, wenn das Netz nur polynomielle Funktionen enthält
es sind daneben auch immer nicht-polynomielle Funktionen notwendig

Berechnung des Eingangs-Wertes eines Neuron's
Summe aus den gewichteten Ausgängen der vorgelagerten Neuronen (Vorgänger-Schicht / -
Layer) und dem Bias

Berechnung des Funktions-Wertes der Aktivierungs-Funktion aus der eben gebildeten Ein-
gangs-Summe

Ausgabe des Funktions-Wertes an die nachgelagerte Schicht (Nachfolger-Schicht)

wird gebraucht

- um dem Netz Nich-Linearität hinzuzufügen
- für eine bessere Annäherung an eine Zielfunktion zu erreichen
- um den Werte-Bereich / Funktions-Wert eines Neuron einzuschränken

notwendige Eigenschaften für Aktivierungs-Funktionen

- sie müssen stetig sein
- sie müssen ableitbar sein (wird für die Backpropagation gebraucht)
- dürfen nicht polynomiell sein

sigmoide Aktivierungs-Funktion

S-förmige Funktion

$$f(x) = \frac{1}{1+e^{-x}}$$

Vorteile	Nachteile	Allgemeines
klar definierte Werte-Bereich	Optimierung schwierig, da: • Ausgabe nicht auf 0 zentriert ist	y-Werte von 0 bis 1

	• Gradient kann sich schnell und stark in verschiedene Richtungen ändern	
bei mittleren Eingaben relativ große Änderungen, hier weitestgehend linear mit weichen Übergängen	bei große und kleinen Eingaben nur geringfügige bis keine Veränderung	
gut ableitbar		

besonders für Klassifizierungs-Netze geeignet

tanh Aktivierungs-Funktion

Tangens-Hyperbolicus-Funktion

$$f(x) = \tanh(x)$$

praktisch auch S-förmige Kurve

Werte-Bereich der Funktions-Werte liegt jetzt zwischen - und 1 und ist damit besser für Rück-Kopplungen (z.B. Backpropagation) geeignet

Vorteile	Nachteile	Allgemeines
klar definierte Werte-Bereich		y-Werte von -1 bis 1
bei mittleren Eingaben relativ große Änderungen, hier weitestgehend linear mit weichen Übergängen	bei große und kleinen Eingaben nur geringfügige bis keine Veränderung	
gut ableitbar		
Ausgabe ist 0 zentriert		

noch besser als simoid für Klassifizierungs-Netze

ReLU Aktivierungs-Funktion

zusammengesetzte Funktion
Rectified Linear Units

$$f(x) = \max(0, x)$$

für alle Werte kleiner als 0 erhält man 0, und sonst x als Funktions-Wert

Dying ReLU-Problem

wenn die Eingänge unter Null liegen kommt immer Null heraus

kann in Netzen mit vielen negativen Eingangs-Werten zum "Verschwinden" der Informationen führen

sehr viele Neuronen werden u.U. nicht aktiviert

Lösung über eine leicht Anpassung der ReLU-Funktion für den Bereich $x < 0$ möglich

leakyReLU-Funktion: $f(x) = \max(\alpha \cdot x, x)$

z.B. schwache lineare Funktion (mit geringem Anstieg)

oder eLU-Funktion

$$\text{ReLU}(x) = \begin{cases} x & \text{falls } x > 0 \\ \alpha * (e^x - 1) & \text{sonst} \end{cases}$$

Vorteile	Nachteile	Allgemeines
sehr einfach zu berechnen (→ schnellere Netze)	keine Beschränkung nach oben	y-Werte zwischen 0 und x
große Eingabe führen zu großen Änderungen der Ausgabe		
gut ableitbar		
	nur für Hidden-Layer einsetzbar	

für komplexe KNN's
meist gegen sigmoid vorgezogen

Softmax-Aktivierungs-Funktion

Wahrscheinlichkeits-Funktion für die Eingabe-Werte

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

z.B.: $\text{softmax}(3,4,1) = (0,259, 0,705, 0,035) = \mathbf{0,705}$

Vorteile	Nachteile	Allgemeines
	nicht ableitbar	

für Ausgabe-Layer in KNN's

ADALINE

adaptive linear neuron oder auch **adaptive linear neuron**
1960 von WIDROW und HOFF vorgestellt
benutze die Delta-Regel als Verbesserung des Perzeptron-Lern-Algorithmus

Backpropagation

Fehler-Rückführung

setzt Multi-Layer-Perzeptron voraus
Backpropagation wurde 1986 von RUMELHART et.al. als ein neuartiges Lern-Verfahren veröffentlicht
Ursprünge gehen auf Techniken bis zum Ende der 1960er Jahre zurück

für ein Perzeptron

<https://www.youtube.com/watch?v=ZDuSR1xxR4c> (Video: "Das Perzeptron, von Hand durchgerechnet")

Umsetzung eines Perzeptron's in Python

```
# Coden des Rosenblatt Perzeptrons
# -----
import numpy as np
import random

random.seed(1)

# Treppenfunktion
def unit_step(x):
    if x < 0:
        return(0)
    else:
        return(1)

# Daten (ersten beide Elemente des Tupels) und Bias (letztes Element)
X = np.array([[0,0,1],
              [0,1,1],
              [1,0,1],
              [1,1,1]])

# Output
y = np.array([0,1,1,1])

# Container und Einstellungen
w = np.random.rand(3) # Zufällige Anfangsgewichte
errors = []           # Fehler
eta = 0.2              # Lernrate
n = 100                # Trainingsiterationen (Epochs)
```

```

# Training
for i in range(n):
    # Zeilenindex
    index = random.randint(0,3)

    # Minibatch (On-line learning) der Größe Eins
    x_batch = X[index,:]
    y_batch = y[index]

    # Aktivierung berechnen
    y_hat = unit_step(np.dot(w, x_batch))

    # Fehler berechnen und abspeichern
    error = y_batch - y_hat
    errors.append(error)

    # Gewichte anpassen
    w += eta * error * x_batch

# Vorhersage
print("{}: {} -> {} | {}".format('Index', 'Probability', 'Prediction', 'Target'))
print('---')

for index, x in enumerate(X):
    y_hat = np.dot(x, w)
    print("{}: {} -> {} | {}".format(index, round(y_hat, 3),
                                     unit_step(y_hat), y[index]))

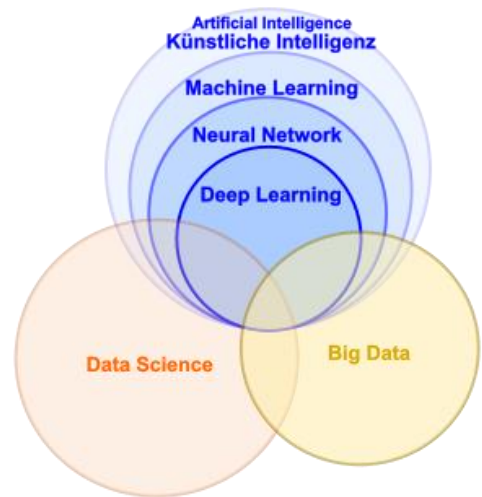
%matplotlib inline
import matplotlib.pyplot as plt

# Grafik Trainingsfehler
plt.plot(range(n), errors)
plt.xlabel('Epoche')
plt.ylabel('Fehler')
plt.title('Trainingsfehler')
plt.show()

```

Q: <https://www.statworx.com/de/blog/das-rosenblatt-perzeptron-die-fruehen-anfaenge-des-deep-learnings/>

2.5. Künstliche Neuronale Netzwerke



	Experten-System	Neuronales Netzwerk
Charakteristika	•	•
Vorteile Stärken	<ul style="list-style-type: none"> • lassen den Verlauf von Entscheidungen nachvollziehbar machen • Wissen ist einer nachvollziehbaren Form vorhanden und kann direkt beeinflusst / korrigiert werden • schon Teilsysteme können gute oder zumindestens Teillösungen erzeugen • ... 	<ul style="list-style-type: none"> • können verrauschte Daten verarbeiten • können mit unvollständigen Daten arbeiten • ...
Nachteile Schwächen	<ul style="list-style-type: none"> • System benötigt umfassende und vollständige Daten • widersprüchliche Regeln führen zu ev. zu Fehl-Leistungen • ... 	<ul style="list-style-type: none"> • Entscheidungen sind nicht nachvollziehbar • Wissen ist nicht nachvollziehbar und gezielt / direkt korrigierbar (steckt in einer Gewichte-Matrix) • erfordern aufwändige Lern-Vorgänge • Konfiguration der Schichten und Parameter z.T. zufällig oder durch Intuition des Erstellers • systematische Konfigurations-Suche extrem aufwändig • nur Gesamt-System Funktionsfähig • ...
Gemeinsamkeiten	<ul style="list-style-type: none"> • können große Daten-Mengen verarbeiten • ... 	
Unterschiede	•	•

2.5.x. kleine punktuelle Geschichte / Historie der Künstlichen Neuronalen Netzwerke, ...

nicht repräsentativ

Jahr / Zeitraum	Forscher	Themen / Leistungen / ...
Anfänge 1942 - 1955		
1943	MCCULLOCH PITTS	(Schalt-)Netzwerke auf der Basis von speziellen Zellen (später MCCULLOCH-PITTS-Zellen genannt)
1947	MCCULLOCH PITTS	Erkennung von räumlichen Mustern durch Neuronale Netze
1949	HEBB	beschreibt Lern-Regeln für KNN (→ HEBBSche Lern-Regel)
1950	LASCHLEY	erklärt Daten-Speicherung im Gehirn als verteilte Repräsentation
Erste Blüte 1955 - 1969		
		erster Hype
1957	ROSENBLATT WIGHTMAN et.al.	erster erfolgreicher Neuro-Computer (Mark Perception)
1958	SELFRIEDGE	entwickelte dynamische, interaktive Mechanismen zur Lösung von praktischen Problemen (z.B. MORSE-Code-Übersetzung)
1958	STEINBUCH	einfache technische Realisierung assoziativer Speicher (Vorgänger heutiger Assoziativ-Speicher) → Lern-Matrix
1958	WIDROW HOFF	entwickeln ein adaptives System (→ Adaline), mit der schnell und effektiv gelernt werden kann
stille Jahre 1969 - 1982		
		Depression
1969	MINSKY PAPERT	mathematische Analyse des Perzeptron's zeigen, dass diverse Probleme nicht realisiert werden können
1972	KOHONEN unabhängig auch: ANDERSON	Modell des linearen Assoziator's
1973	VON DER MAL- SBURG	entwickelt ein komplexes, biologisch besser motiviertes, nicht-lineares Neuronen-Modell
1974	WERBOS	entwickelt Backpropagation-Verfahren
Renaissance 1985 - ...		
		zweiter Hype
1985	HOPFIELD	zeigt mit speziellen Netzen (später HOPFIELD-Netze), wie schwierige Optimierungs-Aufgaben gelöst werden können
ab 1986	diverse	Explosions-artige Entwicklung der KNN's
??? 201x - ...		
		? Hype
201x	deepl	KI-basierte Übersetzung von Texten
2022	openAI	Freigabe von ChatGPT für die breite Öffentlichkeit

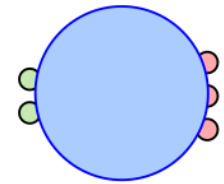
2.5.x. Aufbau künstlicher neuronaler Netze

Festlegung / Veränderung durch ...	
Anwender	das KNN
Topologie	
Aktivierungs-Funktion	
Fehler- / Kosten-Funktion	
Lern-Algorithmus	
... (→ <i>Hyper-Parameter</i>)	
Bias-Startwert	Bias
Startwerte der Gewichte	Gewichte
vor der Nutzung	während der Nutzung

hier allgemein

Schicht-Arten (nach der Lage im Netzwerk)

- **Input-Layer** Perzeptrone für die Aufnahme von Informationen von außen
- **Hidden-Layer** Perzeptrone, welche die Verarbeitung übernehmen / die Funktion des Netzwerkes realisieren
- **Output-Layer** Ausgabe-Einheiten – also die Perzeptrone, die das Ergebnis anzeigen



Netzy als
Black-Box-System

von Input zu Output ist der adaptive Weg
vom Output zurück ins Netz gibt es den regulativen / Weg

Backpropagation über mehrere Perzeptonen / in einem Netzwerk

nach Vorschlägen von David Everett RUMELHART () entwickelt

für eine einfache Zusammenschaltung von Perzeptonen (für einfache mathematische Aufgabe)
ev. wie in openHPI-Kurs vorgestellt

für ein Netzwerk

1. ein Eingabe-Muster wird angelegt und vorwärts durch das Netz verarbeitet
2. die daraus resultierende Ausgabe des Netzwerkes wird mit dem Erwartungs-Bild verglichen → die Abweichung wird als Fehler interpretiert
3. der Fehler wird nun zurück durch das Netzwerk geleitet und die Gewichte entsprechend angepasst
4. weiter bei 1. bis akzeptable Abweichung auftritt oder keine / nur eine geringfügige Veränderung mehr eintritt oder eine bestimmte Anzahl von Durchläufen erreicht wurde

aus der stufigen / binären Auswerte-Funktion
wird jetzt eine sigmoide Transfer-Funktion T

$$T = \frac{1}{1+e^{-A}}$$

$$A_k = \sum_i w_{ik} \cdot X_i - B_k$$

A_k ... Aktivierung des Elementes k

w ... Gewichte

X ... Eingangs-Werte

B ... Bias

Lern-Formeln

$$A_k = \sum_{i=1}^{N+1} w_{ik} \cdot X_i$$

Einbeziehung des Fehler's aus der Ausgabe-Schicht

$$E = \frac{1}{2} \cdot \sum_i (D_i - Y_i)^2$$

2.5.x.y. Schichten

hier speziell

Schicht-Arten (nach der Lage im Netzwerk)

- Input-Layer
- Output-Layer
- Hidden-Layer

Input Layer

im Allgemeinen sehr stark an die Feinheit der Eingabe-Daten angepasst

z.B. ein 300x300 Pixel RGB Bild müsste praktischen 300x300x3 = 270'000 Neuronen in der Eingabe-Schicht enthalten

Problem der Übertragung von 2D-Eingabe-Daten bei Bildern in die 1D-Struktur der Eingabe-Schicht

viele Nachbarschafts-Beziehungen zw. den Punkten / Pixeln / Daten gehen verloren

gutes Beispiel in → "Perzeptron2011.pdf"

z.B. in einer speziellen Python-Bibliothek (keras) könnte so ein Input-Layer so angelegt werden:

```
eingabeSchicht=Input(shape=(300,300,3))
```

dazu später mehr (→)

Output Layer

stellt die Ausgabe-Schicht dar
in Klassifikations-Netzwerken ist diese Schicht meist nur mit sehr wenigen Neuronen besetzt
i.A. braucht man nur soviele, wie man Ausgabe-Klassen hat



Hidden Layer

Schicht-Arten (nach)

- **Fully Connected Layer**
Dense Layer
- **Convolutional Layer**
- **Pooling Layer**

Convolutional Layer

klassische KNN haben eine eindimensionale Eingabe
selbst Bilder werden z.B. Zeilen-weise in einen Vektor gewandelt
damit gehen aber die Nachbarschafts-Beziehungen zu den drüber und drunter liegenden
Bild-Punkten verloren
mit sogenannten Convolutional Layer (dt.: faltende (neuronal) Netzwerkschicht)

Convolution (Faltung)

bei besonders starkem Einbau von Convolutional Layer's (CL) in ein KNN werden diese dann
Convolutional Neural Networks (CNN) genannt.

früher inspiriert von der Arbeitsweise des visuellen Cortex von Katzen
schon ab den 1930er Jahren und dann viel intensiver in den 1950er Jahren wurde der Seh-
Vorgang an Katzen untersucht
hier entdeckte man die Schicht-Organisation der Nervenzellen; die Nervenzellen einer nach-
gelagerten Schicht nehmen Informationen von vielen vorgelagerten Nervenzellen auf
man fand Zell-Cluster, die scheinbar mit der Erkennung bestimmter Objekte / Situationen
beschäftigt sind
Untersuchungen erfolgten bis in die 80er Jahre

auch bei der Bewertung und Entwicklung / Testung haben Katzen immer eine große Rolle gespielt

es geht um die Erkennung zusammenhängender Strukturen (z.B. in 2D-Objekten), die durch die Vektorisierung / Sequenzierung der Daten verloren gehen
notwendig für die Erkennung abstrakter Strukturen / Merkmale
wichtigste Art der Neuronen-Zusammenschaltung
in KNN's sind die CL's eher in den unteren / vorderen Schichten zu finden

gutes Beispiel in → "Perzeptron2011.pdf" (Besprechung bei Eingabe-Schicht)

es gibt auch ein biologisches Äquivalent
bei der Untersuchung von Neuronen-Aktivitäts-Mustern im Gehirn hat man festgestellt, dass es quasi ein oder mehrere ""-Neuron gibt, dass immer dann feuert, wenn auf einem Bild oder im Original zu sehen ist / war

bei der Analyse von KNN's fand man heraus, dass in einem trainierten KNN bestimmte künstliche Neuronen dann auch dann feuern, wenn man dem Netz ein Bild mit eben nur diesem Muster vorsetzt

filtert man die internen "Bilder", die ein KNN von einem Original-Bild in seinem Netz durch Verstärkungen und Filterungen erzeugt, heraus, dann sind diese Bild-Abwandlungen eine eigene Bilderwelt

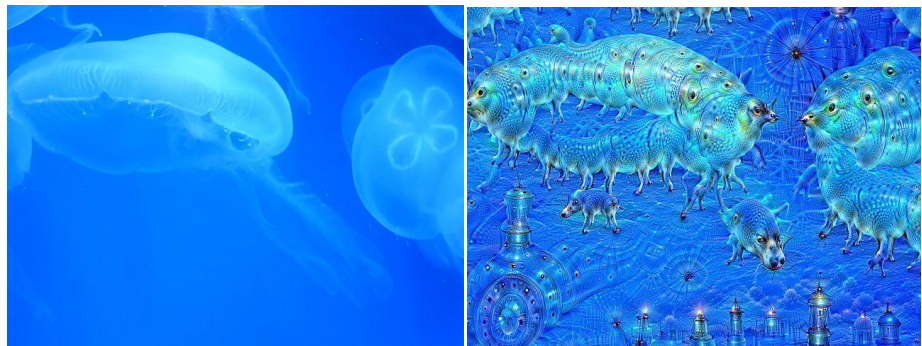


Bild einer Ohren-Qualle, und wie ein trainiertes KNN dies intern "sieht"

Q: de.wikipedia.org (Martin Thoma)

im KI-Bereich gibt es einige Forscher, die dieses Verhalten als "halluzinieren" bzw. "träumen" des Netzes bezeichnen

Nach neuesten Erkenntnissen der Aerodynamik können Hummeln nicht fliegen.
Aber die Hummeln wissen das nicht und fliegen trotzdem.
Spruch aus einem Flugzeug-Entwicklungs-Büro

z.B. Erkennung von Kanten
über SOBEL-Kernel (SOBEL-Matrix)

-1	0	1
-2	0	2
-1	0	1

z.B. Erzeugung von Unschärfe
über GAUSSIAN-Kernel (GAUSS-Matrix)

-1	0	1
-2	0	2
-1	0	1

Anwendung eines Kernel's auf die Eingabe-Daten (2D-Struktur)

5	4	1	0	2
3	4	1	0	2
1	1	1	0	1
0	0	1	0	0
2	1	3	1	0

0	-1	0
-1	3	-1
0	-1	0

Die Kernel-Matrix wird zentriert auf den ersten Pixel gelegt. Die fehlenden Daten-Werte werden z.B. durch die Betrachtung als geschlossenes Objekt von den anderen Rändern (bzw. Ecken) geholt. Auch eine Dopp-lung der Rand-Werte der Daten-Matrix ist denkbar.

0 0	-1 2	0 1			
-1 2	3 5	-1 4	1	0	2
0 2	-1 3	0 4	1	0	2
			1	1	1
			0	0	1
			2	1	0

jetzt erfolgt die Berechnung des Skalar-Produktes, d.h. jedes Matrix-Element wird mit den passenden Element der anderen Matrix multipliziert und die dann die Summe dieser Produkte gebildet

0	-2	0			
-2	15	-4	1	0	2
0	-3	0	1	0	2
			1	1	1
			0	0	1
			2	1	0

4				

nun beinhaltet die neue Matrix Werte, die nicht nur den zugehörigen Wert der Eingabe-Matrix in Beziehung stehen, sondern diese auch Informationen über die Nachbarn (die räumliche Umgebung enthalten).

4	1	-5	-4	-1
-3	3	-3	-3	0
-2	-3	0	-2	0
-3	-3	-1	-2	-1
0	-6	5	0	-5

bei RGB-Bildern oder anderen Farb-Systemen mit mehreren Kanälen werden die Feature Map's auf jedem einzelnen Kanal angewendet diese können zusammengefasst werden oder einzeln mit einem kombinierten Kanal in weitere Layer eingehen

mehrere Feature Map's können mit weiteren Kernel's zu einem Output Feature Map kombiniert werden und dieser dann als Input für die nächste Schicht verwendet werden

durch Kernel's werden bestimmte Eigenschaften / Features der ursprünglichen Struktur hervorgehoben
das können z.B. sein:

- Formen und Kanten / Farben (→ Low level-Features)
- komplexe Formen / Strukturen (→ High level-Features)

durch jeden Kernel wird ein sogenanntes Feature Map erzeugt.

in KNN werden Kernel's zufällig initialisiert und deren Anwendung dann über die Backpropagation zu neuen Gewichten an den Neuronen-Eingängen führen dabei muss in einem Layer's immer der gleiche Kernel (für alle Neuronen) angewendet werden

Definition(en): Feature Map

Ein Feature Map ist ein Ausgabe-Bild / Ausgabe-Matrix, das / die durch Anwendung der Convolution (Faltung) entsteht.

Dabei werden immer bestimmte Eigenschaften / Merkmale / Strukturen → Features hervorgehoben.

z.B. in einer speziellen Python-Bibliothek (keras) könnte so ein Convolutional-Layer so angelegt werden:

```
aktivFkt= ...
modell=Conv2D(filters=32, kernel_size=(7,7),
              activation=aktivFkt)(eingabeSchicht)
```

dazu später mehr (→)

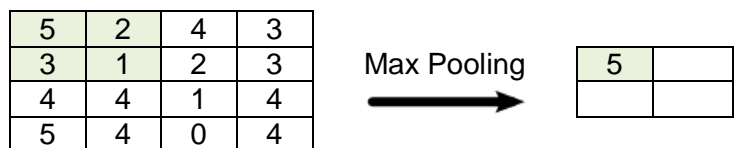
kombiniert man Bild-Analyse und Convolutional Networks, dann kann man zu Generative Adversarial Network's (GAN) kommen mit diesen können dann künstliche Bilder z.B. von Gesichtern erstellt werden, die man praktisch von echten Personen(-Bildern) nicht unterscheiden kann

Pooling Layer

die Position des zu erkennenden Objektes ist für die Objekt-Erkennung beim Menschen kein Problem für Maschinen ist aber die "Unwichtigkeit" der Objekt-Position nicht erkennbar und auch die Position selbst ist schwer bestimmbar noch dazu kann das Objekt größer oder kleiner skaliert in den Daten stecken

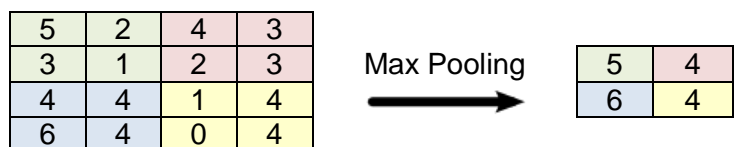
beim Pooling werden die Eingabe-Daten Überlappungs-frei in kleinere Bereiche aufgeteilt für jeden Bereich eine Operation ausgeführt (siehe Pooling-Verfahren) und das Ergebnis in das neue Ausgabe-Daten-Objekt geschrieben

es kommt hier zu einer deutlichen Reduktion der Daten-Elemente auf mindestens die Hälfte der Daten-Elemente (in jede Dimension) bei 2D sind das "Kompressions"-Raten von 75 % bei einem 2x2-Max Pooling



dadurch werden die Objekt-Daten auf das Wesentliche (hier die Spitzen-Werte) reduziert

die verschiedenen Original- und Ergebnis-Bereiche sind farblich unterschiedlich unterlegt



Ziel ist die Erkennung größerer Strukturen (unabhängig von der Position) führt i.A. zu einer Verbesserung der Erkennung von Bild-Inhalten meist wird dadurch Overfitting (→) verhindert

Vorteile:

- Komplexitäts-Reduktion
- Beschleunigung der Berechnungen
- Auflösen der räumlichen Bezüge (der erkannten Strukturen) innerhalb des Datensatzes (/ Bild's)
- alle Parameter werden vor dem Training des KNN fest definiert, bleiben unverändert und sind meist durch einfache Rechen-Operationen gekennzeichnet
-

Nachteile:

-

Max Pooling hat in Experimenten allgemein gesehen die besten Ergebnisse gebracht
beste Erfahrungen mit 2x2-Pool's

für Spezial-Probleme sind aber auch andere Pooling-Verfahren verfügbar

in der Praxis setzt man Pooling-Layer meist nach einem Convolutional-Layer ein, dadurch werden die wesentlichen Informationen aus den Convolutional-Operationen (Struktur-Findung) betont in die nächste Schicht weitergeleitet

weitere Pooling-Verfahren

- **Average Pooling**

neuer Daten-Wert ist der Durchschnitt aus den betrachteten Bereich

-
-

die verschiedenen Original- und Ergebnis-Bereiche sind farblich unterschiedlich unterlegt

5	2	4	3
3	1	2	3
4	4	1	4
6	4	0	4

Average Pooling →

2,75	3,00
4,50	2,25

z.B. in einer speziellen Python-Bibliothek (keras) könnte so ein MaxPooling-Layer so angelegt werden:

```
modell=MaxPooling2D((2,2))(modell)
```

dazu später mehr (→)

Flatten Layer

dienen zur Umwandlung von mehrdimensionalen Daten in einen Vektor

z.B. in einer speziellen Python-Bibliothek (keras) könnte so ein Flatten-Layer so angelegt werden:

```
modell=Flatten()(modell)
```

dazu später mehr (→)

Dense Layer

dichte Schicht, auch Fully Connected Layer
dienen zur Klassifikation von Daten / Informationen
Daten sind hier Vektor-orientiert

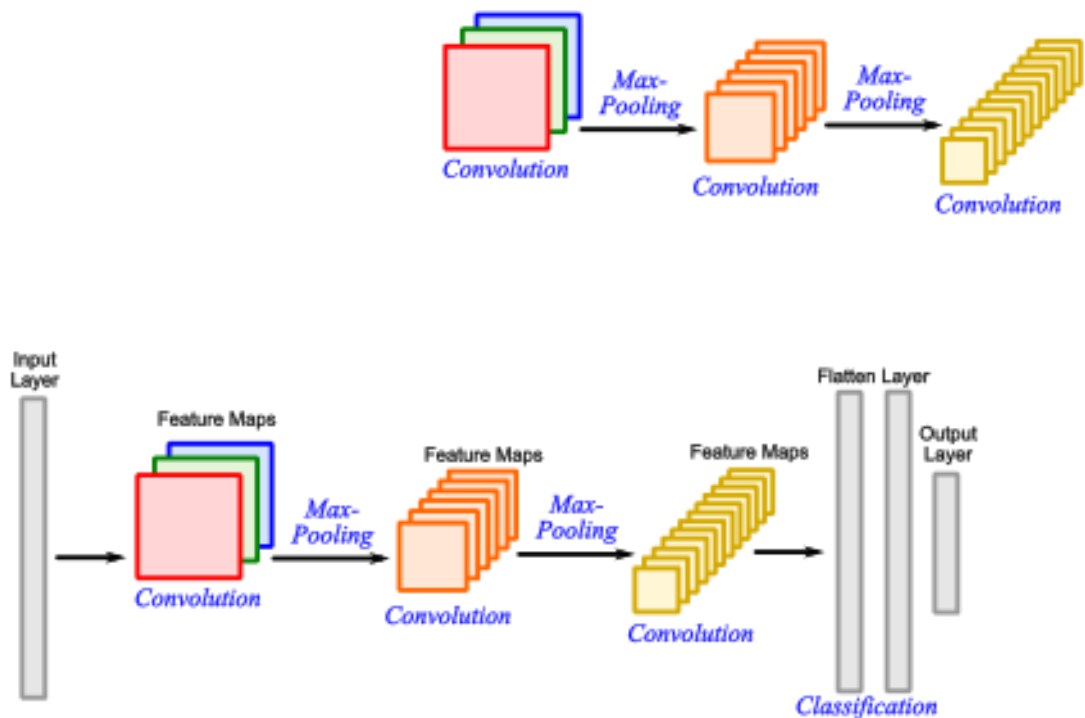
z.B. in einer speziellen Python-Bibliothek (keras) könnte so ein Dense-Layer so angelegt werden:

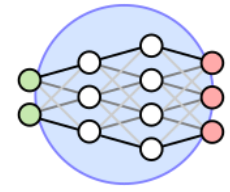
```
modell=Dense(250,activation='tanh')(modell)
```

dazu später mehr (→)

komplexe Layer

praktische Folgen / Aneinander-Reihungen von speziellen Layern
ev. als fertig konfigurierte Module verfügbar, z.B. zur Bild-Vorbereitung vor der Bild-Analyse





stark vereinfachte "Struktur" von Netzy

mathematisch / informatisches Modell

in der Form, wie Automaten usw. in der Theoretischen Informatik definiert werden

KNN =

Definition(en): Neuronales Netz
<p>Ein Neuronales Netz ist ein (Hex-)Tupel $(\mathbf{U}, \mathbf{W}, \mathbf{A}, \mathbf{O}, \mathbf{NET}, \mathbf{ex})$, wobei:</p> <ul style="list-style-type: none"> • \mathbf{U} die Menge der Verarbeitungs-Einheiten (Neuronen) • \mathbf{W} die Netzstruktur mit $W: U \times U \rightarrow \mathbb{R}$ • \mathbf{A} die Aktivierungs-Funktion für jedes einzelne Neuron u mit $A: U \mapsto A_u, A_u: \mathbb{R}^3 \rightarrow \mathbb{R}$ • \mathbf{O} die Ausgabe-Funktion mit $O: U \mapsto O_u, O_u: \mathbb{R} \rightarrow \mathbb{R}$ • \mathbf{NET} die interne oder Netz-Eingabe-Funktion für jedes Neuron u mit $\mathbf{NET}: U \rightarrow \mathbf{NET}_u, \mathbf{NET}_u: (\mathbb{R} \times \mathbb{R})^{ U } \rightarrow \mathbb{R}$ • \mathbf{ex} die externe Eingabe-Funktion mit $\mathbf{ex}: U \rightarrow \mathbb{R}$ <p>sind.</p> <p>Diese Definition beschreibt die statischen Eigenschaften (praktisch Zustand-orientiert) des Netzes. Über die Dynamik macht sie keine Aussage.</p>

Menge der Verarbeitungs-Einheiten U

beinhaltet alle künstliche Neuronen (Verarbeitungs-Einheiten) u
 entsprechen einfachen Automaten / Prozessoren
 arbeiten parallel und unabhängig voneinander
 haben einen Zustand

bekommen aus dem Netz oder von außen Eingabe-Signale und berechnen über die Aktivierungs-Funktion eine Ausgabe und einen neuen Zustand

man kann Eingabe-Einheiten (aus dem Input-Layer), versteckte oder innere Einheiten (im Hidden-Layer) sowie Ausgabe-Einheiten (im Output-Layer) unterscheiden

Netz-Struktur W

als gerichteter Graph vorstellbar, in dem die Neuronen als Knoten und die Verbindungen / Synapsen als Kanten fungieren. Die Gewichte entsprechen den Kanten-Gewichten, die sich aber in der Lern- / Trainings-Phase ändern (können)

ein Gewicht von 0 bedeutet keine Wirkung und kann auch als fehlende Verbindung (Kante) interpretiert werden

Ist das Gewicht größer als 0, dann kann man sich dies als anregende Verbindung vorstellen. Entsprechend ist ein Gewicht, das kleiner als 0 (- also negativ ist -), als hemmende Verbindung zu verstehen.

eine andere Möglichkeit ist die Darstellung als Konnektions-Matrix

Menge der Aktivierungs-Funktionen A

jeder Verarbeitungs-Einheit kann eine individuelle Aktivierungs-Funktion zugeordnet werden. I.A. werden aber nur den einzelnen Schichten oder allen, die gleiche Aktivierungs-Funktion zugeordnet.

die Ausgabe / das Ergebnis der Ausgabe-Funktion wird durch die gewichteten Eingangssignale, den Bias (Grund-Level) und der Funktion selbst (deren Graph) abhängig
typisch sind Schwellenwert-, Treppen-, lineare, stückweise lineare oder sigmoide Funktionen

Ausgabe-Funktion O

ist das Ergebnis der Berechnung der Aktivierungs-Funktion
Abhängigkeiten gerade bei den Aktivierungs-Funktionen besprochen

Netz-Eingabe-Funktion / interne Funktion NET

ist die Verknüpfung der Eingabe-Signale als Argument für die Aktivierungs-Funktion
i.A. werden die gewichteten Eingabe-Signale und der Bias () aufsummiert
meist verwenden alle verarbeitungs-Einheiten die gleiche interne Funktion

$$net_u = \sum_{v \in U} W(v, u) \cdot o_v$$

externe Eingabe-Funktion ex

ist eine Verbindung der Außenwelt zur einzelnen Verarbeitungs-Einheit
meist für alle oder zumindest für die Verarbeitungs-Einheiten einer Schicht gleich

Beispiel:

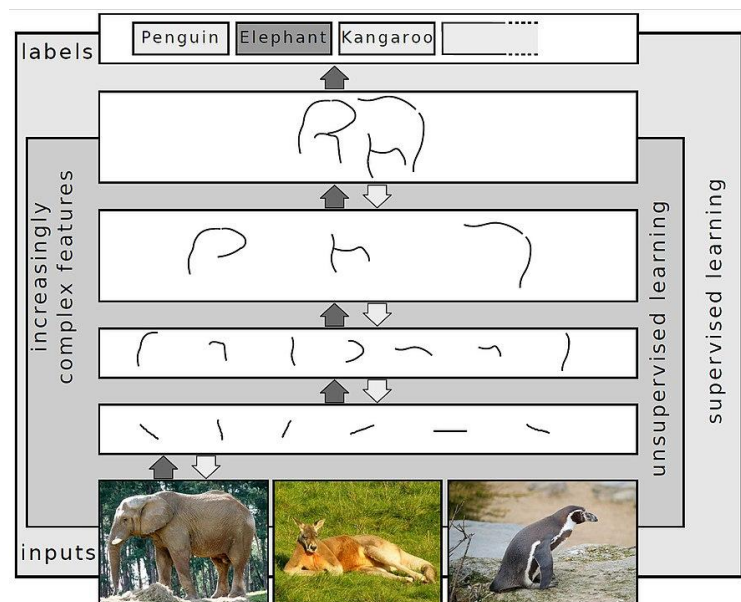
in "ModSim024.pdf"

2.5.x. Arbeitsweise / Funktion von KNN's

Arbeits-Phasen eines Neuronalen Netzwerkes

- **Ruhe-Phase** Warte-Zustand des Netzes und der einzelnen Neuronen
quasi-stabiler Zustand
- **Eingabe-Phase** folgt der Ruhe-Phase
wird durch die verschiedenen Eingangs-Signale ausgelöst und
schließt die Berechnung der Ausgabe mit ein
- **Arbeits-Phase** das berechnete Ausgabe-Signal wird an die nachfolgenden Neu-
ronen weitergegeben
die Neuronen führen individuell Updates durch und kehren dann in
den Ruhe-Zustand zurück

meist wird eine Taktung (neue Muster an Eingabe-Signale → neue Eingabe) verwendet
selten arbeiten KNN kontinuierlich
alle Neuronen arbeiten im selben Takt-System



Repräsentation von Bildern / Bild-Details
auf verschiedenen Abstraktions-Ebenen
Q: de.wikipedia.org (Sven Behnke)

2.5.x. Rückgekoppelte Neuronale Netzwerke

bei RNN's (Recurrent Neural Network's) gehen die Daten-Flüsse nicht linear (von der Eingabe-Schicht zur Ausgabe-Schicht) durch
es gibt intern zusätzliche Quer-Verbindungen, die bis zu Rückkopplungen gehen können
dadurch kommt es zu einer stärkeren Verknüpfung der Daten-Kanäle untereinander

üblich ist eine versetzte Nutzung der Rückkopplungs-Werte, d.h. der Ausgangs-Wert beim aktuellen Lern-Schritt wird als Eingangs-Wert beim nächsten Lern-Schritt benutzt

die einzelnen Neuronen erhalten durch die Rückkopplungen quasi verschiedene Arten von Gedächtnis

besonders interessant für Systeme, die Sequenzen oder Zeit-Abläufe analysieren sowie für generative Systeme

sie können praktisch unendliche Sequenzen verarbeiten

im Vergleich dazu können Convolutional Neuronal Networks (CNN's) nur endliche Sequenzen auswerten

das RNN ist durch ein zeitlich dynamisches Verhalten charakterisiert

typische Anwendungs-Gebiete sind:

- Handschriften-Erkennung
- Sprach-Erkennung / -Analyse
- ...

RNN sind theoretisch TURING-vollständig

erste Ideen zu solchen Netzwerken stammen aus dem Jahr 1925 (LENZ und ISING)
diese Netze waren thermodynamische Strukturen, die durch bestimmte Eigenschaften, z.B. Ferromagnetismus, gekennzeichnet sind und ein Energie-Minimum anstreben,
die Netze lernten aber noch nicht

1982 wurde das ISING-(LENZ-)Modell dann von HOPFIELD in ein Neuronales Netzwerk umgesetzt

ein solches Netz besteht nur aus einer Schicht, weil die Neuronen dieser Schicht sowohl die Eingabe als auch die Ausgabe realisieren, spricht man von einem **Auto-Assoziations-Netz**
die Neuronen können nur die Zustände 1 (feuert) und -1 (feuert nicht) einnehmen

in einem HOPFIELD-Netz erhält eine Verarbeitungseinheit neben seinem (normalen) Eingang noch die verknüpften Ausgangs-Werte der jeweils anderen Einheiten als Eingang



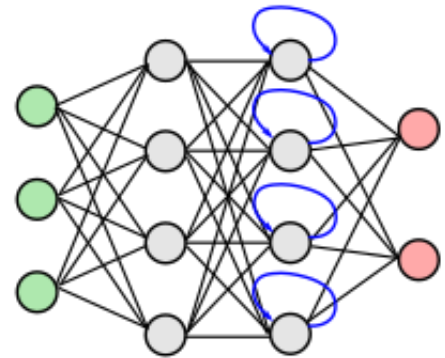
HOPFIELD-Netz
mit drei Neuronen

heutige Rekurrente Neuronale Netze können verschiedene Arten von Rückkopplungen enthalten

direkte Rückkopplung

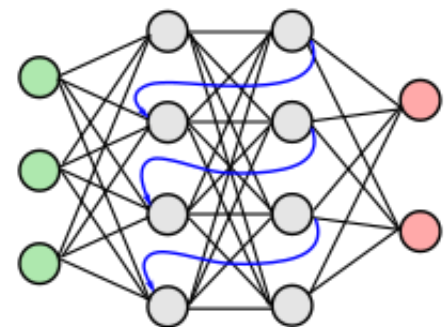
die Neuronen einer Schicht verwenden ihren Ausgang auch als Eingangs-Wert

die besonderen Rückkopplungs-Verbindungen sind blau eingezeichnet, der Pfeil kennzeichnet die Wirkrichtung



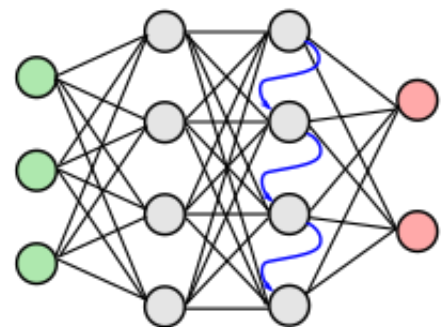
indirekte Rückkopplung

die Neuronen einer nachgelagerten Schicht liefern ihren Ausgang als Eingangs-Wert für eine vorgelagerte Schicht



seitliche Rückkopplung

die Neuronen innerhalb einer Schicht verwenden den Ausgang eines Nachbar-Neuron's als Eingangs-Wert



vollständige Rückkopplung

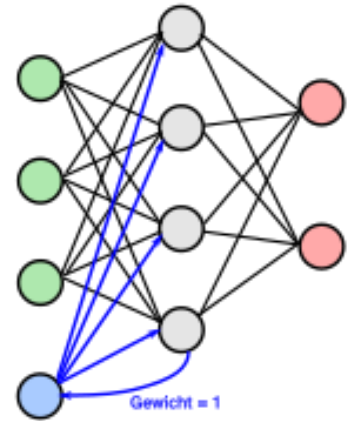
jedes Neuron erhält als Eingangs-Wert auch die Ausgänge der anderen Neuronen

ELMAN-Netz

dreischichtiges Neuronales Netzwerk mit Eingabe-Schicht (grün), einer versteckten Schicht (grau) und der Ausgabe-Schicht (rot).

daneben sind Kontext-Einheiten hinzugefügt, die von den Neuronen der versteckten Schicht mit einem konstanten Gewicht von 1 gefüttert werden

die Kontext-Einheiten sind zusätzliche Eingänge für die nachgelagerte Schicht



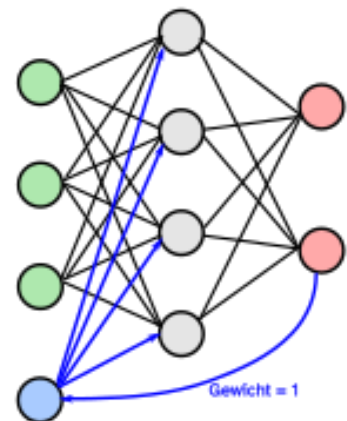
Prinzip des ELMAN-Netzes
(exemplarisch für ein Neuron und
eine Kontext-Einheit)

JORDAN-Netz

sehr ähnlich zum ELMAN-Netz

dreischichtiges Neuronales Netzwerk mit Eingabe-Schicht (grün), einer versteckten Schicht (grau) und der Ausgabe-Schicht (rot).

daneben sind Kontext-Einheiten zugefügt, die von den Neuronen der Ausgabe-Schicht mit einem konstanten Gewicht von 1 gefüttert werden



Prinzip des JORDAN-Netzes
(exemplarisch für ein Neuron und
eine Kontext-Einheit)

prinzipielle Trainings-Schritte (einer Lern-Runde) in einem Rekurrenten Netz:

1. *Eingabe eines Datensatzes*
2. *Netz verarbeitet die Eingabe mit den aktuellen Gewichten und Hyper-Parametern*
3. *die Ausgabe des Netzes wird mit dem Ziel-Ergebnis verglichen und eine Fehlerwert berechnet*
4. *die Gewichte werden angepasst; die Ausgaben bestimmter Neuronen werden anderen Neuronen (oder sich selbst) wieder als Eingangs-Wert (für den nächsten Datensatz) zur Verfügung gestellt*
5. *weiter bei 1. bis alle Datensätze bearbeitet wurden*

neben den üblichen Aktivierungs-Funktionen wird auch eine sogenannte Verlust-Funktion berechnet

diese berechnet den Fehler des Netzwerkes

üblich ist die Verwendung der Kreuzentropie

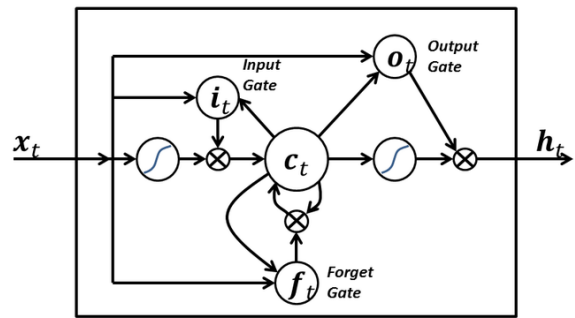
derzeit werden weitere Rückkopplungs-Systeme erforscht und benutzt

Long Short-Term Memory (LSTM)

dt.: Langes Kurzzeit-Gedächtnis

1997 von HOCHREITER und SCHMIDHUBER vorgestellt

durch eine spezielle Struktur des Netzes wird verhindert, dass die Fehler-Funktion bei einem lokalen Minimum stecken bleibt und das globale Minimum nicht gefunden wird durch die Verwendung von drei besonderen Toren bzw. Gattern (Eingangs-, Merk- und Vergessens-Tor) ist es möglich, dass LSTM's auch längerfristig frühere Eingaben speichern / verarbeiten können



Struktur eines LSTM-Modul's
Q: de.wikipedia.org (BiObserver)

Gated Recurrent Unit (GRU)

ein um das Ausgangs-Tor (-Gatter) verringertes LSTM-ähnliches Struktur-Element

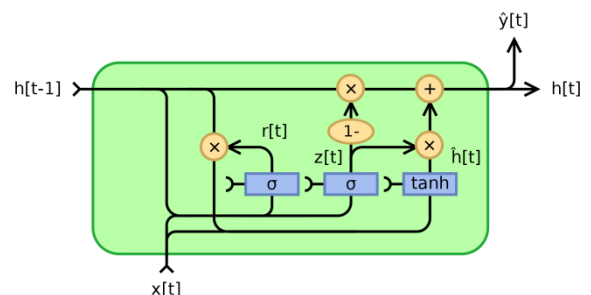
mehrere Typen beschrieben, die unterschiedliche Eingangs-Daten benutzen

der Basis-Typ (vollständige GRU) enthält alle Berechnungs-Möglichkeiten

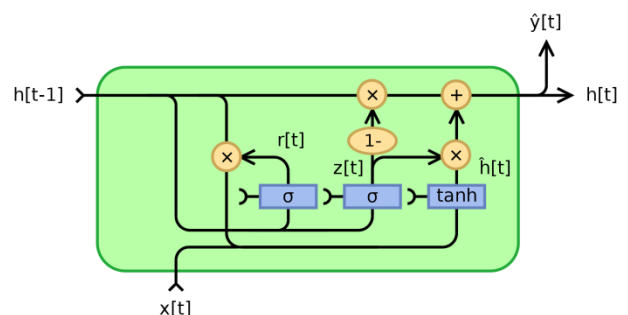
Typ I ist eine GRU, deren Ergebnis nur vom vorherigen verborgenen Zustand und des Eingangs-Wert's abhängt

Typ II das Ergebnis dieser GRU hängt ausschließlich vom vorherigen verborgenen Zustand ab

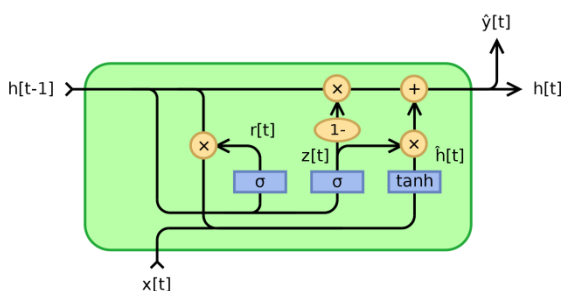
Typ III das Ergebnis dieser GRU hängt ausschließlich vom Eingangs-Wert ab



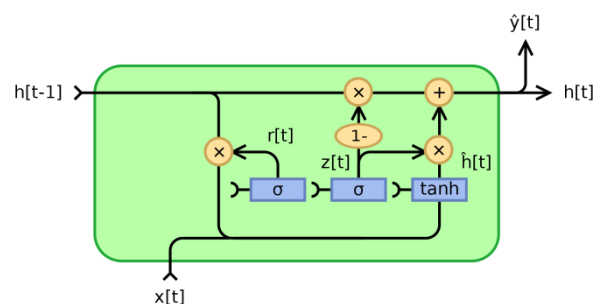
vollständige GRU (Basis-Typ)



Typ I-GRU



Typ II-GRU



Typ III-GRU
alle GRU-Abb.: Q: en.wikipedia.org (Jeblad)

2.6. maschinelles Lernen

Schnell-Schluss: Lernen in Computer-Systemen
sachlich auch erst einmal so richtig, trifft aber den Kern nicht ganz
einfaches Lernen (also z.B. das Sammeln und Verwenden von Informationen) ist eine etwas
aufwendige, aber gut zu realisierende Aufgabe für einen Programmierer

das folgende Programm ist in BASIC geschrieben – eine Programmiersprache, die in den
80er und 90er Jahren des 20. Jahrhunderts ihre Hochzeit hatte

Aardwark-Programm

```
10 PRINT "AARDWARK"
20 PRINT "DENKEN SIE SICH EIN TIER AUS UND"
30 PRINT "ICH WERDE ES ERRATEN !"
40 PRINT "BEANTWORTEN SIE JEDE FRAGE MIT"
50 PRINT "JA ODER NEIN!"
60 PRINT
70 GOSUB 1000
100 GOSUB 2000
110 PRINT
120 PRINT "DENKEN SIE SICH EIN TIER AUS!"
130 PRINT
140 GOTO 100
1000 DIM Q$(20),R(20),L(20),N$(20)
1010 Q$(1)="HAT ES EIN GEFIEDER ?"
1020 D=1
1030 R(1)=0
1040 L(1)=0
1050 N=0
1060 Q=1
1070 RETURN
2000 X=1
2010 GOSUB 3000
2020 IF A$="J" AND R(X)=0 THEN GOTO 4000
2030 IF A$="N" AND L(X)=0 THEN GOTO 4000
2040 IF A$="J" AND R(X)<0 THEN A=-R(X):GOTO 5000
2050 IF A$="N" AND L(X)<0 THEN A=-L(X):GOTO 5000
2060 IF A$="J" THEN X=R(X)
2070 IF A$="N" THEN X=R(X)
2080 GOTO 2010
3000 PRINT Q$(X)
3010 INPUT A$
3020 A$ =LEFT$(A$,1)
3030 IF A$="J" OR A$="N" THEN RETURN
3040 PRINT "ICH VERSTEHE IHRE ANTWORT NICHT !"
3050 PRINT "BITTE ANTWORTEN SIE AUF MEINE"
3060 PRINT "FRAGEN NUR MIT JA ODER NEIN !"
3070 PRINT
3080 GOTO 3000
4000 PRINT "ICH KENNE DAS TIER NICHT,"
4010 PRINT "AN DAS SIE DENKEN"
4020 PRINT "WIE HEISST ES ?"
4030 INPUT B$
4040 PRINT
4050 PRINT "WELCHE ANDERE FRAGE KANN ICH STELLEN,"
4060 PRINT "UM DAS TIER ZU UNTERSCHIEDEN ?"
4070 INPUT C$
4080 N=N+1
4090 N$(N)=B$
4100 Q=Q+1
4110 Q$(Q)=C$
4120 IF A$="J" THEN R(X)=D+1 ELSE L(X)=D+1
4130 PRINT "WAS IST DIE ANTWORT "
4140 PRINT "AUF ";C$;" FÜR Ein ";B$;" ?"
```



```

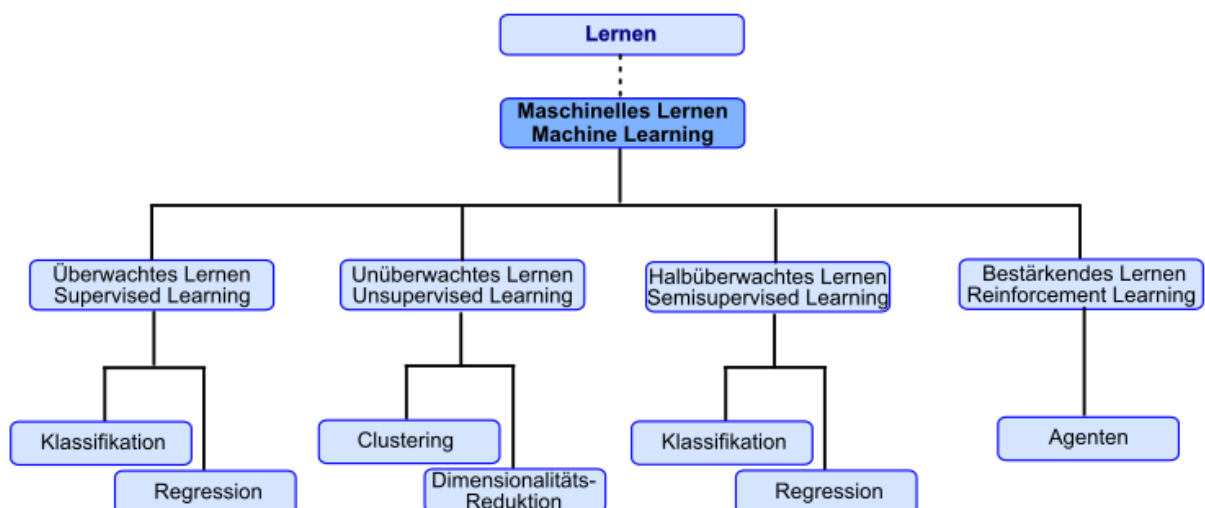
4150 INPUT D$
4160 D$=LEFT$(D$,1)
4170 IF D$="J" OR D$="N" THEN GOTO 4300
4180 PRINT "ANTWORTEN SIE MIT 'JA' ODER 'NEIN' !"
4190 GOTO4130
4300 D=D+1
4310 IF D$="J" THEN R(D)=-N:L(D)=0
4320 IF D$="N" THEN L(D)=-N:R(D)=0
4330 RETURN
5000 PRINT "IST ES EIN ";N$(A)
5010 INPUT B$
5020 IF LEFT$(B$,1)="N" THEN GOTO 6000
5030 IF LEFT$(B$,1)<>"N" THEN PRINT "ANTWORTEN SIE MIT 'JA' ODER 'NEIN' !":
GOTO5000
5040 PRINT "ICH HATTE NUR GEDACHT !!"
5050 PRINT
5060 RETURN
6000 PRINT "ICH GEBE AUF !"
6010 INPUT "WAS IST ES ?",B$
6020 PRINT "WELCHE FRAGE UNTERSCHIEDET "
6030 PRINT "EIN ";N$(A)
6040 PRINT "VON IHREM ";B$;" ?"
6050 INPUT C$
6060 Q=Q+1
6070 Q$(Q)=C$
6080 N=N+1
6090 N$(N)=B$
6100 IF A$="J" THEN R(X)=Q ELSE L(X)=Q
6110 D=D+1
6120 PRINT "WIE LAUTET DIE ANTWORT AUF ";C$
6130 PRINT "BEI EINEM ";B$
6150 INPUT D$
6160 D$=LEFT$(D$,1)
6170 IF D$="J" THEN R(D)=-N:L(D)=-A
6180 IF D$="N" THEN L(D)=-N:R(D)=-A
6190 PRINT
6200 RETURN

```

Q: JAMES, Mike: Künstliche Intelligenz in BASIC – Arbeitsbuch für Heimcomputer.-München: moderne verlagsgesellschaft mbh, 1984; S. 53 f.

es stellt ein sehr, sehr einfaches – lernfähiges – Experten-System dar
am Beispiel von Tieren wird ein hierrarchisches System (→ Entscheidungs-Baum) aufge-
baut, das bei der Identifizierung von Tieren dient

Übrigens ist die BASIC-Version ein schönes Beispiel für einen sogenannten Spaghetti-Code.
Um dies mal zu erleben, braucht man es nur mal zu versuchen, den Ablauf nachzuvollzie-
hen. Die verschiedenen Unterprogramm-Aufrufe mit GOSUB und die GOTO's sorgen für
einen undurchsichtigen Programm-Ablauf. U.a. deswegen wurde BASIC durch andere –
stärker strukturierte und GOTO-freie – Programmiersprachen abgelöst.



Problem / Aufgabe	meist verwendete ML-Technik	
Temperatur-Vorhersage aus bekannten Klima-Daten	Regression	
Erkennung von Vogel-Stimmen / -Gesang	Klassifikation	
Bewertung eines Chat-Bot-Post's	Entscheidungs-Bäume / regelbasierte Analyse Klassifikation	
Spam-Filter	Clustern Klassifikation	
Bestimmung von Versicherungs-Prämien unter Beachtung vieler verschiedener Faktoren	Regression	
Betrugs-Erkennung bei Kreditkarten-Nutzungen	Klassifikation Clusterung	
personalisierte Werbe-Anzeigen / Kauf-Empfehlungen / Musik- oder Video-Vorschlägen	Klassifikation	
Bild-Erkennung im medizinischen Bereich (z.B. Tumor-Erkennung)	Klassifikation	
Bewertung von Sensor- oder Log-Datenreihen	Clustering z.T. auch Regression und Klassifikation	
Computer-gegner in einem Spiel	Regel-basiertes oder bestärkendes Lernen	
Gesichter-Erkennung	Klassifikation	

Wissens-Repräsentation

implizites Wissen

Personen-gebundenes Wissen

im Kopf einer Person, unsichtbar, subjektiv, schlecht kommunizierbar

z.B. Fähigkeit zum Sprechen einer Fremdsprache

in Firmen z.B. das Erfahrungs-Wissen der Belegschaft

explizites Wissen

kaum an einen Kontext gebunden, leicht in Worte zu fassen, dokumentierbar, leicht zu imitieren

z.B. Fachbuch

in Firmen z.B. die gesamte Dokumentation, Pläne, Skizzen, ...


Ansätze zum Speichern von Wissen

z.B. Speichern eines Verzeichnis-Baum's

möglich mit Text-Datei, in der alle möglichen Pfade als Zeilen-Einträge geführt werden
hinzufügen ist relativ einfach
umbenennen muss schon prüfen, ob man im richtigen Unter-Verzeichnis ist (da ja gleich
benannte Ordner mehrfach im Baum vorkommen können)
löschen erfordert Überprüfung, ob Ordner wirklich leer ist (also keine weiteren Unter-Ordner
vorhanden sind)

in den 80er und 90er Jahren war die Wissen-Repräsentation in LISP und PROLOG-
Systemen sehr beliebt
Experten-Systeme
bei PROLOG sind es Atome (die eigentlichen Daten) und Regeln (die Verknüpfungen der
Daten), die Wissens-Basis darstellen

Beispiele

→ Stammbaum-Analyse (→  [Medienbildung](#))
→ symbolische Differenzierung

Wissens-Graphen

z.B. WikiData (→ [WikiData](#))

Kodierung mit RDF (Resource Description Framework)

Basis sind Subjekt-Prädikat-Objekt-Konstrukte

```
Peter liebt Anna  
Katze frisst Maus  
Torwart hält Ball  
Berlin liegt_in Deutschland
```

Speicherung z.B. in Text-Dateien (N-Triples-Format) als Tripel

Codierung möglichst als URL's → es entstehen sogenannte URI's (Uniform Resource Identifier) (z.B.: für Berlin: <https://www.berlin.de/id>)

RDF im Turtle-Format:

```
@prefix ex: <https://example.org/> .  
@prefix rdfs: <https://www.w3.org/2000/01/rdf-schema#> .  
ex:Berlin ex:Hauptstadt ex:Deutschland .  
ex:Berlin ex:Einwohner 3700000  
ex:Berlin ex:Website <https://www.berlin.de> .  
ex:Berlin rdfs:label "Berlin" .
```

RDF-Standard OWL (sprich: "aul") (eigentlich WOL: Web Ontology Language)

OWL-Klasse	informelle Aussage	formale Codierung hier: OWL-FSS	
1	Goethe ist ein Dichter.	ClassAssertion(Dichter Goethe)	
2	Dichter sind Autoren.	SubClassOf(Dichter Autor)	
3	Jeder Dichter hat irgendwann ein Gedicht geschrieben.	SubClassOf(Dichter ObjectSomeValuesFrom(hatGeschrieben Gedicht))	
4	Autoren, die ein Gedicht geschrieben haben, sind Lyriker.	SubClassOf(ObjectIntersectionOf(Autor ObjectSomeValuesFrom(hatGeschrieben Gedicht)) Lyriker)	

Ontologie

formale, explizite Spezifikation einer gemeinsamen Begriffs-Bildung

Relationen

Begriffs-Paare

lassen sich gut graphisch darstellen; es entstehen Begriffs-Ketten oder -Bäume

beschreiben die Beziehung zwischen den Begriffen

Merkmale treten in den Hintergrund

bei Relationen mit mehr Daten (Attributen) als das Begriffs-Paar sind dann auch umfangreiche

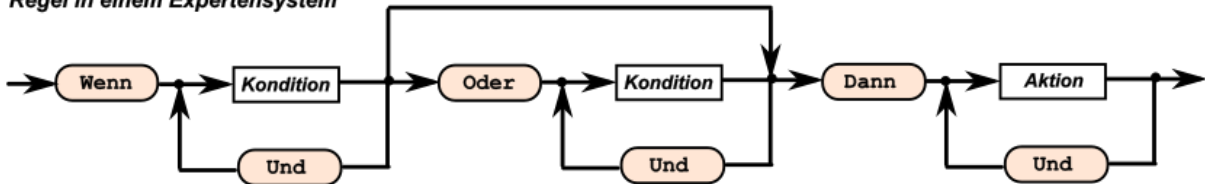
bei mehr Begriffen in einer Relation entstehen schnell unübersichtliche Netze (aufwändige Suche)

Experten-Systeme

Teilbereich der Künstlichen Intelligenz

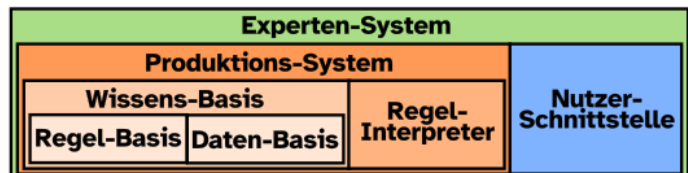
basieren auf WENN-DANN-Beziehungen

Regel in einem Expertensystem



praktisch verwandt mit den Entscheidungs-Bäumen
 normalerweise schon fest vorhandenes Basis-Wissen einprogrammiert
 kann bei vielen Systemen durch Nutzer erweitert werden (lern-fähige Systeme)

	klassisches Programm	Experten-System
Chrarakteristika	•	•
Gemeinsamkeiten	•	
Unterschiede	•	•
Vorteile	•	•
Nachteile	•	•



Realisierungs-Prinzipien

- **Fall-basierte Systeme**
- **Regel-basierte Systeme
Business Rule Management System (BRMS)**
- **Entscheidungsbäume**

Regel-Interpreter(-Systeme) / Heuristik-Systeme

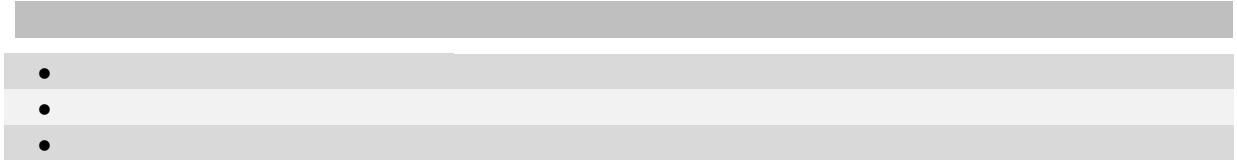
- | | |
|--|--|
| • Nächster-Nachbar-Algorithmus | berechnen von Abständen und dann zuerst mit dem Knoten mit dem kleinsten Abstand weiter suchen |
| • Generiere-und-Teste-Algorithmus | erzeugen von (zufälligen) Lösungsmöglichkeiten und prüfen, ob der Vorschlag eine Lösung ist |
| • Hill Climbing
(Erklimme den Berg) | Suche nach einer nächsten Lösung, die am dichtesten am Ziel ist, solange bis das Ziel erreicht ist |
| • Breadth-First Search | erzeugen einer neuen Ebene von Lösungsvorschlägen in einem Suchbaum und prüfen, ob in dieser Ebene schon eine Lösung enthalten ist, sonst neue Ebene generieren |
| • Deep-First Search | in einem Suchbaum ausgehend von einem Zweig prüfen, ob dieser ein Weg zum Ziel ist, ansonsten nächsten Nachbarzweig verwenden oder einen Knoten höher gehen und dessen Zweige prüfen |
| • Best-First Search | (weitere) Suche im Baum ausgehend vom derzeit besten gefundenen Knoten |
| • ... | |

Aufgaben-Klassen für Experten-Systeme

• Daten-Interpretation	<ul style="list-style-type: none">○ Erkennung akustischer Sprache○ Identifizierung chemischer Strukturen (Auswertung Massenspektren-Daten)○ geologische Erkundung○ militärische Aufklärung○ Überwachung von U-Boot-Aktivitäten○ ...
• Überwachung	<ul style="list-style-type: none">○ Produktions-Abläufe○ Qualitäts-Kontrolle○ Patienten-Überwachung○ Kern-Reaktoren○ chemische Produktionen○ ...
• Diagnose	<ul style="list-style-type: none">○ bakterielle Infektionen○ rheumatische Erkrankungen○ Erkrankungen○ Bestimmung und Lokalisierung technischer Fehler○ Pflanzen-Krankheiten○ ...
• Therapie	<ul style="list-style-type: none">○ Fehler-Diagnose bei Getrieben○ Ortung und Fehler-Erkennung in Telekommunikations-Systemen○ automatische Entwöhnung von Patienten in der Intensiv-Medizin○ Arzneimittel-Sicherheit○ ...
• Planung	<ul style="list-style-type: none">○ Versuchplanung molekular-genetischer Experimente○ chemische Synthesen○ Finanz-Planung○ Produktions-Planung○ Ablaufplanung des Flugbetriebs (Flugzeugträger)○ Aktivitäten autonomer Roboter (Mars-Rover, ...)○ ...
• Entwurf	<ul style="list-style-type: none">○ Schaltkreis-Entwurf○ Computer-Konfigurationen○ Konfiguration von Betriebssystemen○ ...
• Prognose	<ul style="list-style-type: none">○ Erdbeben-Vorhersage○ Beurteilung von Erdbeben-Auswirkungen○ Hochwasser-Vorhersage○ ...
• ...	<ul style="list-style-type: none">○ ...

Erhebung des Wissens für die Wissens-Basis

- Experten
- Literatur-Recherche
- Datenerfassung aus der Praxis
- interaktives Lernen mit / am Nutzer
- ...



WikiData

Sammlung von Daten und Daten-Beziehungen
derzeit 14 Bill. Tripel

liefert z.B. für einen Artikel die Links zu den fremdsprachigen Artikeln in anderen wikipedia's

Elemente haben eine eindeutige QID ()

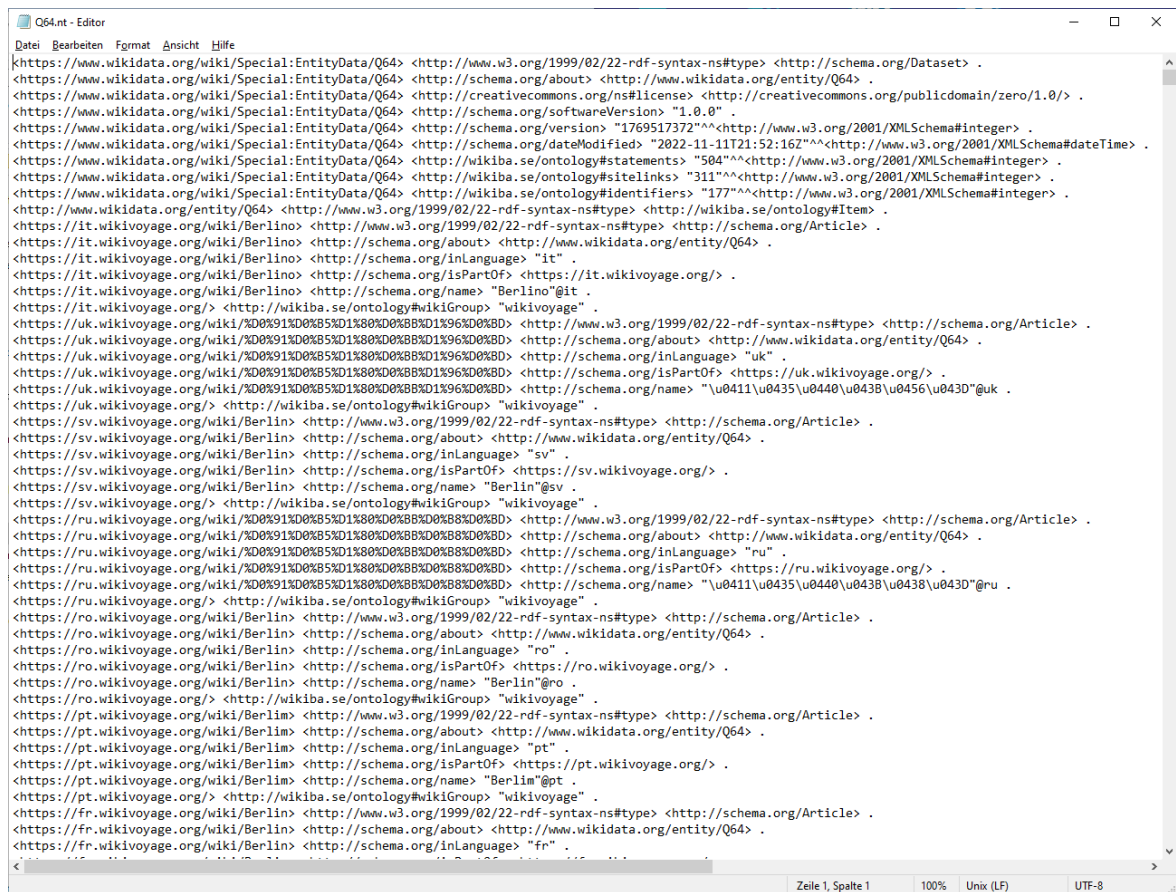
z.B. Berlin → Q64

Abfrage über:

<https://www.wikidata.org/entity/QID>

Abfrage der RDF als speicherbare Datei über:

<https://www.wikidata.org/entity/QID.nt>



```
Q64.nt - Editor
Datei Bearbeiten Format Ansicht Hilfe
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Dataset> .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://creativecommons.org/ns#license> <http://creativecommons.org/publicdomain/zero/1.0/> .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://schema.org/softwareVersion> "1.0.0" .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://schema.org/version> "1769517372"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://schema.org/dateModified> "2022-11-11T21:52:16Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://wikiba.se/ontology#statements> "504"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://wikiba.se/ontology#sitelinks> "311"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q64> <http://wikiba.se/ontology#identifiers> "177"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://www.wikidata.org/entity/Q64> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://wikiba.se/ontology#Item> .
<https://it.wikivoyage.org/wiki/Berlino> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Article> .
<https://it.wikivoyage.org/wiki/Berlino> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://it.wikivoyage.org/wiki/Berlino> <http://schema.org/inLanguage> "it" .
<https://it.wikivoyage.org/wiki/Berlino> <http://schema.org/isPartOf> <https://it.wikivoyage.org/> .
<https://it.wikivoyage.org/wiki/Berlino> <http://schema.org/name> "Berlino"@it .
<https://it.wikivoyage.org/> <http://wikiba.se/ontology#wikiGroup> "wikivoyage" .
<https://uk.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D1%96%D0%BD> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Article> .
<https://uk.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D1%96%D0%BD> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://uk.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D1%96%D0%BD> <http://schema.org/inLanguage> "uk" .
<https://uk.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D1%96%D0%BD> <http://schema.org/isPartOf> <https://uk.wikivoyage.org/> .
<https://uk.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D1%96%D0%BD> <http://schema.org/name> "\u0411\u0435\u0440\u0438\u0430\u0430"@uk .
<https://uk.wikivoyage.org/> <http://wikiba.se/ontology#wikiGroup> "wikivoyage" .
<https://sv.wikivoyage.org/wiki/Berlin> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Article> .
<https://sv.wikivoyage.org/wiki/Berlin> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://sv.wikivoyage.org/wiki/Berlin> <http://schema.org/inLanguage> "sv" .
<https://sv.wikivoyage.org/wiki/Berlin> <http://schema.org/isPartOf> <https://sv.wikivoyage.org/> .
<https://sv.wikivoyage.org/wiki/Berlin> <http://schema.org/name> "Berlin"@sv .
<https://sv.wikivoyage.org/> <http://wikiba.se/ontology#wikiGroup> "wikivoyage" .
<https://ru.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D0%B8%D0%BD> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Article> .
<https://ru.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D0%B8%D0%BD> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://ru.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D0%B8%D0%BD> <http://schema.org/inLanguage> "ru" .
<https://ru.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D0%B8%D0%BD> <http://schema.org/isPartOf> <https://ru.wikivoyage.org/> .
<https://ru.wikivoyage.org/wiki/%D0%B5%D1%80%D0%B8%D0%B8%D0%BD> <http://schema.org/name> "\u0411\u0435\u0440\u0438\u0430\u0430"@ru .
<https://ru.wikivoyage.org/> <http://wikiba.se/ontology#wikiGroup> "wikivoyage" .
<https://ro.wikivoyage.org/wiki/Berlin> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Article> .
<https://ro.wikivoyage.org/wiki/Berlin> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://ro.wikivoyage.org/wiki/Berlin> <http://schema.org/inLanguage> "ro" .
<https://ro.wikivoyage.org/wiki/Berlin> <http://schema.org/isPartOf> <https://ro.wikivoyage.org/> .
<https://ro.wikivoyage.org/wiki/Berlin> <http://schema.org/name> "Berlin"@ro .
<https://ro.wikivoyage.org/> <http://wikiba.se/ontology#wikiGroup> "wikivoyage" .
<https://pt.wikivoyage.org/wiki/Berlim> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Article> .
<https://pt.wikivoyage.org/wiki/Berlim> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://pt.wikivoyage.org/wiki/Berlim> <http://schema.org/inLanguage> "pt" .
<https://pt.wikivoyage.org/wiki/Berlim> <http://schema.org/isPartOf> <https://pt.wikivoyage.org/> .
<https://pt.wikivoyage.org/wiki/Berlim> <http://schema.org/name> "Berlim"@pt .
<https://pt.wikivoyage.org/> <http://wikiba.se/ontology#wikiGroup> "wikivoyage" .
<https://fr.wikivoyage.org/wiki/Berlin> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Article> .
<https://fr.wikivoyage.org/wiki/Berlin> <http://schema.org/about> <http://www.wikidata.org/entity/Q64> .
<https://fr.wikivoyage.org/wiki/Berlin> <http://schema.org/inLanguage> "fr" .
```

Anfang der Q64.nt-Datei (zu "Berlin") von wikidata.org (am 14.11.2022)
Text-Datei war insgesamt 2,4 MB groß

Aufgaben:

- 1. Finden Sie eine geeignete BASIC-Version für Ihren Rechner (z.B.: QBasic, ...)! Die BASIC-Version kann ein DOS-Programm sein und muss nur für die Konsole geeignet sein.**
- 2. Erstellen Sie das Aardwark-Programm mit Ihrem BASIC-Programmiersystem!**
- 3. Probieren Sie das Programm aus!**
- 4. Erkunden Sie, was die einzelnen Programm-Abschnitte tun / leisten! Bei Bemerkungen können Sie sich z.B. Hinweise für eine moderne Umsetzung oder Verbesserungs-Vorschläge notieren.**

ab Befehls-Zeile	Programm-Teil / Funktion(en)	Bemerkungen
10		
100		
1000		
2000		
3000		
4000		
5000		
6000		

- 4. Erweitern Sie das Programm auf z.B. 50 mögliche Tiere!**
- 5. Machen Sie aus dem Aardward-Programm ein Experten-System für Objekte aus Ihrer (Hobby-)Welt!**

hier auch die Umschreibung in Python

Definition(en): Machine Learning

Künstliches Lernen ist allgemein die künstliche Generierung von Wissen aus Erfahrung.

Maschinelles Lernen realisiert die Computer-basierte Ausführung von Aufgaben durch Lernen aus Daten.

Maschinelles Lernen ist das Teilgebiet der Künstlichen Intelligenz, das durch Erkennen von Mustern in vorliegenden Daten eigenständig Lösung zu Problemen findet.

Maschinelles Lernen ist ein Teilgebiet der Künstlichen Intelligenz, das Datenwissenschaft (Data science) und Informatik verbindet.

Maschinelles Lernen ist eine Form der Künstlichen Intelligenz, bei der die Entwicklung von analytischen Modellen automatisiert wird und die Computer befähigt die Modelle eigenständig an neue Szenarien anzupassen.

Der Bereich des Maschinellen Lernen's beschäftigt sich mit den Formen des eigenständigen Computer-basierten Lernen's in Systemen der Künstlichen Intelligenz und der Robotik.

Maschinelles Lernen ist die Erforschung und Nutzung von Verfahren, durch die Computer-Systeme in die Lage versetzt werden, selbstständig Wissen aufzunehmen und zu erweitern, um eine Problem / eine Aufgabe effektiver / besser zu lösen als vorher.

Maschinelles Lernen (Machine Learning ist das Forschungsgebiet, in dem Computer die Fähigkeit zu lernen gegeben wird, ohne explizit programmiert zu werden.
nach Arthur SAMUEL (1959)

Machine Learning ist das Studium von Computer-orientierten Algorithmen, die sich (selbst) durch Erfahrung automatisch verbessern.

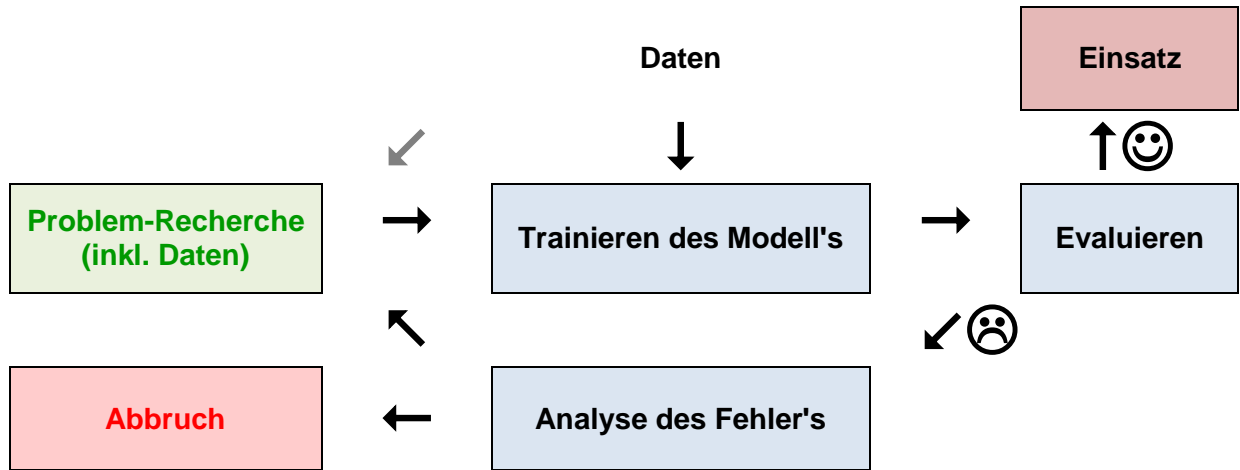
Maschinelles Lernen bezeichnet die Prozesse, bei denen Computer eigenständig Muster erkennen und / oder die Fähigkeit entwickeln, kontinuierlich aus Daten zu lernen und aus diesen heraus Prognosen abzuleiten, ohne dass diese vorher explizit einprogrammiert wurden.

Maschinelles Lernen ist der Bereich der Informatik, der sich damit beschäftigt, Computern das (eigenständige) Lernen und Handeln beizubringen, ohne dieses vorher explizit vorher einzuprogrammieren.

Ein Computer lernt durch Erfahrung E in Bezug auf eine Art von Aufgabe (Task) T und eine Performancemetrik P, wenn die Performance zur Aufgabe T, gemessen durch P, mit steigendem E besser wird.
nach Tom MITCHELL (1997)

Machine Learning (Maschinelles Lernen) ist die Wissenschaft (und die Kunst) Rechner so zu programmieren, dass sie Daten lernen können.

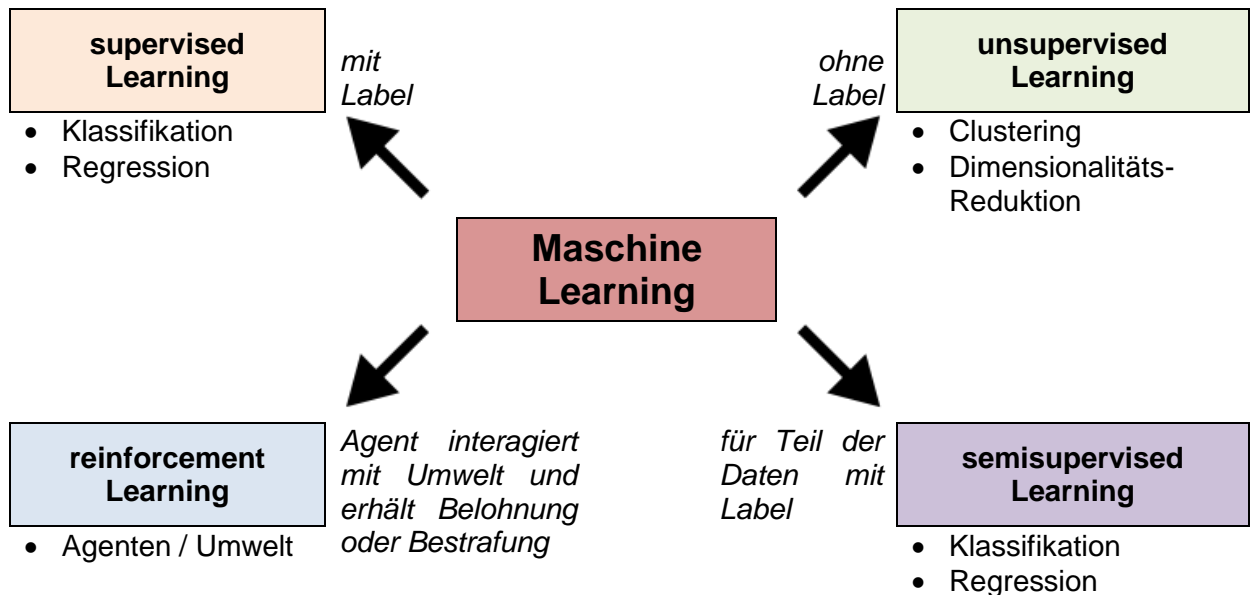
allgemeines Konzept des Maschinellen Lernens



diskriminative Modelle

generative Modelle

das System wird auf der Basis vorhandener Daten trainiert und soll dann neue Inhalte (Texte, Moleküle, Bilder, ...) erzeugen



Definition(en): Agent

Ein Agent ist eine Software, die ihre Umwelt wahrnimmt und Ziel-orientierte Handlungen / Reaktionen vornimmt / ausführt und das in der Lage ist, autonomes (eigenständiges) Verhalten zu zeigen.

Arten / Felder des maschinellen Lernens (Machine Learning)

- **überwachtes Lernen**
betreutes Lernen
Supervised Learning
für Klassifizierungs-Aufgaben / -Modell
neue Daten sollen genauso gut zugeordnet / klassifiziert werden, wie die Trainings-Daten
binäre Klassifikation
z.B. Erkennen eines Verkehrs-Schildes,
Erkennen eines falschen Twitter-Account,
Erkennen von Kreditkarten-Missbrauch,
Erkennen von Schriftzeichen,
...
→ **Klassifikation, Regression**
→ [2.6.x. Supervised Learning](#)
- **unüberwachtes Lernen**
unbeaufsichtigtes Lernen
Unsupervised Learning
allgemeines "Verstehen" der Daten (ohne eine vorgelagerte Lern-Phase)
Erkennen neuer Informationen (ohne vorheriges Training)
Erkennen von neuen Mustern / Daten / Strukturen durch Gruppieren / Clustern
Ableiten von Regeln
es gibt keine Label oder korrekten Ausgabe-Werte
z.B. für Bild-Segmentierung, Analyse des Kauf-Verhaltens von Kunden; Analyse von Einkäufen von Kunden mit Kundenkarten
Daten-Visualisierung gehört auch dazu
...
→ **Clustering, Dimensionalitäts-Reduktion**
→
- **bestärkendes Lernen**
Verstärkungs-Lernen
Reinforcement Learning
Lernen, praktisch so, wie es der Mensch macht für die Suche echter allgemeiner (künstlicher) Intelligenz
lernt nach dem Trail-and-Error-Verfahren (über Belohnungen und Bestrafungen)
Feedback kommt verzögert
Suche nach dem optimalen Verhalten
z.B. selbstfahrende Auto's
...
→ **Agenten, Umwelt**
- **halbüberwachtes Lernen**
Semisupervised Learning
Kombination aus überwachten und unüberwachten Lernen
→ **Klassifikation, Regression**

in der Praxis überlappen aber auch diese Bereiche

beim überwachten Lernen wird zuerst die Art der Eingabe-Daten – also z.B. ein Bild – und dann die Ausgabe-Klassen (z.B. Ziffern 0 bis 9) definiert
es werden Trainings-Daten gebraucht, die exakt bezüglich der Ausgabe-Klassen beschriftet (gelabelt, annotiert) sind
nach der Trainings- / Lern-Phase soll der Klassifikator neue Bilder erkennen und in die Klassen einordnen

auch die Behandlung von Regressions-Problemen möglich

dabei soll das System eine Zahl voraussagen
z.B. Anzahl Klicks auf eine Werbung; Anzahl möglicher Nutzer / Kunden / ...; Anzahl von zuerwartenden Verkehrs-Unfällen bei bestimmter Verkehrs-Führung, zugelassener Höchstgeschwindigkeit, ...; Verhersage eines Höchst-Gebotes für eine Immobilie; Immobilien-Preis auf der Basis von Lage, Größe, Alter, ...

2.6.x. Supervised Learning (Kurz-Vorstellung)

Problem Unterscheidung von gewünschten eMail's (Ham) und Spam eMail-Validierung

Programm zur Validierung der eMail's bekommt eine Menge von eMail's vorgelegt, bei denen ein Fach-Experte (oder viele (interessierte) Nutzer) die einzelnen Mail's bewertet haben und sie als Ham oder Spam klassifiziert haben

zuerst wird das Programm (ohne die Kenntnis der menschlichen Beurteilung) nur raten, ob es sich um Ham oder Spam handelt

Programm soll nun ein Schema / Regeln / Modell / ... ableiten, nachdem man neue Mail's (und natürlich auch die alten) in die beiden Gruppen unterscheiden kann

im Modell sind alle eingehenden Mail's die Eingabe-Daten x

die Beurteilung / Annotation durch den Menschen ergibt die Ziel-Daten y

die Ausgabe des Modell's / Programm sind die Prognose-Daten \hat{y}

Ziel ist es, dass y (Ziel-Daten) und \hat{y} (Prognose-Daten) vollständig übereinstimmen

Problem ist die Falsch-Beurteilung durch das Programm / Modell:

ausführlich weiter im Abschnitt →

2.6.x. Unsupervised Learning (Kurz-Vorstellung)

Bauern-Schach mit Affen und Krokodil

<http://www.stefanseegerer.de/schlag-das-krokodil/>

Problem der Gruppierung von Nutzern

z.B. für Werbe-Zwecke, Erkennung von Chat-Bot's, ...

Objekte haben Eigenschaften / Merkmale

Objekt-Name	Merkmal 1	M. 2	M. 3	M. 4	M. 5	M. 6	M. 7	...	M. n
Person 1		1,80	80	44				...	
Person 2		1,64	82	41				...	
Person 3		1,29	43	32				...	
Person 4		1,73	65	46				...	
Person 5		1,69	70	40				...	
Person 6		1,52	55	33				...	
Person 7		1,07	40	24				...	
Person 8		1,33	43	30				...	
Person 9		1,40	43	34				...	

Suche nach Ähnlichkeiten (möglichst viele übereinstimmende oder wenig abweichende Merkmale) → Clustering (Gruppen-Bildung)

Merkmale werden als n-dimensionaler Raum verstanden

hier als Beispiel zuerst einmal mit 2 Dimensionen

es bilden sich mehr oder weniger deutliche Haufen (Cluster) heraus, die man ev. Personen-Gruppen zuordnen kann; kann eine reale Gruppe sein, Gruppen können aber auch total abstrakt sein

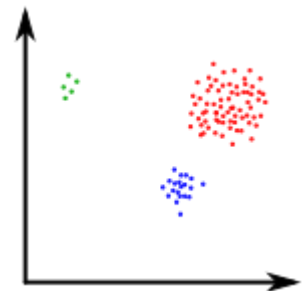
nun für 3 Dimensionen

damit leider Grenzen der visuellen Darstellung aufgebraucht
mathematisch sind aber n-dimensionale Räume betrachtbar

Ähnlichkeit wird über die EUKLIDISCHE Distanz zwischen den Daten-Punkten im n-dimensionalen Raum bestimmt

ähnliche Objekte haben kleine Distanz, unähnliche eher sehr hohe Distanzen

Cluster-Zuordnung z.B. über eine maximale Distanz zwischen den Objekten



ev. muss dann hinterher eine Benennung / Bewertung manuell vorgenommen werden

Links:

<http://computingeducation.de/proj-ai-unplugged/>

<https://machinelearningforkids.co.uk>

2.6.x. Semisupervised Learning (Kurz-Vorstellung)

Kombination aus supervised und unsupervised Learning
sinnvoll und / oder notwendig, wenn nur ein (kleiner) Teil der Daten annotiert ist
ev. ist Annotierung des gesamten Daten-Bestands auch nicht möglich (z.B. weil zeitlich oder personell zu aufwendig)

zuerst werden die Daten in einem n-dimensionalen Merkmals-Raum eingeordnet
dann Clustering über Distanz-Verfahren (z.B. über Nächste-Nachbar-Verfahren)
(Annotation einiger (zufällig ausgewählter) Objekte in den Clustern)

neue Daten-Objekte werden dann über ihre nächste Distanz zu einem annotierten Daten-Objekt bewertet
im Ergebnis haben wir einen automatisch annotierten Daten-Bestand (allerdings nur mit Wahrscheinlichkeits-Aussagen)

2. Verfahren

zuerst nimmt man die stimmigen Daten und fügt fehlerhafte / unstimmige Daten hinzu (z.B. künstlich generierte eMail's)

diese Daten sind somit in die Klassen "echt" und "generiert" annotiert

ein Modell / KNN wird nun darauf trainiert echte und generierte Daten zu trennen

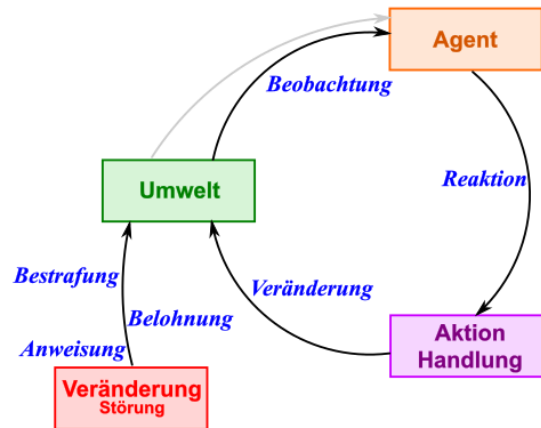
mit diesem Modell geht man dann das Original-Problem (z.B. Ham- und Spam-Trennung) an
Verfahren beruht darauf nicht die Unterschiede zwischen den stimmigen (echten) sowie den unstimmigen / problematischen Daten zu erlernen, sondern das Modell lernt die Merkmale der echten / stimmigen Daten

Abweichler von diesen werden dann als problematisch eingestuft

Definition(en): Heuristik
Eine Heuristik ist die Wissenschaft von den Verfahren mit denen man Problem lösen kann.
Unter einer Heuristik versteht man eine Anweisung / Anleitung, wie man ein Problem angehen / lösen kann.
Eine Heuristik ist eine methodische Anleitung / Handlungs-Richtschnur.
Heuristik ist die Kunst / Lehre, aus begrenztem Wissen und bei wenig verfügbarer Zeit gut geeignete / praktikable Lösungen / Ergebnisse abzuleiten.
Eine Heuristik ist eine (allgemeine) Vorgehensweise zum von Lösen Problemen / vergleichbaren Problemklassen, für keine eindeutige Lösungs-Strategie / kein Lösungs-Algorithmus bekannt ist oder diese nur mit einem nicht sinnvoll erscheinendem Aufwand gelöst werden können.
Heuristiken sind Wahrscheinlichkeits-orientierte Lösungen von Problemen.

2.6.x. Reinforcement Learning (Kurz-Vorstellung)

vergleichbar mit dem Training / der Dressur von Tieren
auf Befehle (Umwelt) reagiert das Tier (Agent) mit einer zufälligen Handlung; Trainer belohnt oder bestraft / reagiert nicht auf Handlung
Wiederholung solange bis Tier die Befehle mit der passenden Handlung beantwortet



Exploration (Erkundung)

Exploitation (Verwertung, Nutzung)

2.6.x. Lern-Vorgänge und -Verfahren

Lern-Schritte:

1. (zufällige) Festlegung der Gewichte und Biases
2. Feedforward
 - a. Summierung der gewichteten Eingänge und des Bias
 - b. Anwendung der Aktivierungs-Funktion
3. Berechnen der Fehler-Funktion
4. Backpropagation
5. Berechnen der Kosten-Funktion / Bestimmen des Gradienten
6. Aktualisieren der Gewichte und Biases
7. Wiederholen ab Schritt 2. (bis alle Trainings-Beispiele durchgelaufen sind)

unüberwachtes Lernen

es gibt keine Trainings-Daten, System muss online auf die Ereignisse reagieren
beim unüberwachten Lernen geht es sehr viel um Daten-analyse und -Visualisierung
das ist die Domäne des Big Data

ausführlich ist dieser Teil der Informatik im Skript [Datenbanken](#) beschrieben
hier nur kurz einige Aspekte

Nearest Neighbor Classifier (Nächster Nachbar Klassifikator)

man benötigt einige Trainings-Daten zur groben Klassifikation, diese Daten sind gelabelt,
neue Daten werden in die Klassen eingeordnet, indem die Distanz zu allen Trainings-Daten

bestimmt wird und der dichteste Nachbar ausgewählt wird. dessen Klasse ist dann auch die Klasse für den neuen Daten-Punkt

für nicht sauber getrennte Klassen lässt die Methode auch statt mit den Einzel-Daten mit den Mittel-Punkten der Klassen machen (→)
das reduziert auch den Rechenaufwand, da nun nur noch so viele Vergleiche, wie es Klassen gibt, gemacht werden müssen, weiterhin muss dann noch der Mittel-Punkt der gerade erweiterten Klasse

typische Anwendung sind weitere Kauf-Empfehlungen () während des Bestell-Vorhang's
neu dabei: generatives Lernen /Modellieren
Generative Adversarial Networks (GAN; dt.: erzeugende konkurrierende Netzwerke)

häufig werden, mangels gelabelter Daten, Präferenzen anderer Nutzer verwendet
diese werden wie ein Filter angewendet, nur die passenden Dinge gehen durch, die anderen bleiben im Filter hängen
Problem ist das Entstehen von Filter-Blasen (typisch auch im social-media-Bereich)
nur noch das kommt durch, was ohnehin schon der eigenen Meinung / Präferenz entspricht;
Alternativen, gegenteilige Argumente / Positionen bleiben im Filter hängen

Regression

gemeint lineare Regression, oft als lineare Kombination bezeichnet
hier steht ein numerisches / fein abgestuftes Ergebnis im Vordergrund
keine Klassen od.ä.

- Highscore in komplexen Spielen
- Vorhersage Klickraten bei Online-Werbung
- Vorhersage Markt-Reaktion auf ein neues Produkt
- Vorhersage der Konsumnachfrage für ein Angebot
- Vorhersage des Einspiel-Ergebnisses eines Film's
- Vorhersage der Versicherungskosten
- Berechnung des Versicherungs-Beitrages
- Vorhersage von Immobilien-Preisen
- ...

um Klassen mittels Regression zu bestimmen, verwendet man die logistische Regression

2.6.x. Bewertung des Lern-Erfolgs

neuronale Netzwerke liefern für eine Eingabe am Ende nur Wahrscheinlichkeiten für die Zugehörigkeit zu den Ergebnis-Klassen
mit Hilfe der Fehler-Funktion soll nun die Abweichung von den Erwartungen (eindeutige (100 %ige) Zuordnung zu einer Klasse) berechnet werden
die Fehler gehen dann in den Lern-Prozess ein und werden im Zuge der Backpropagation zur Korrektur der Gewichte genutzt

wichtig ist eine einfache / effektive Fehler-Funktion, da die Berechnung zu Lasten der Performance unseres Netzes gehen
Ziel ist es ja auch möglichst schnell das Training abzuschließen und ein effektives Netz zu erhalten

Fehler-Funktionen / Loss Function's

quadratische Fehler-Funktion

$$E = \sum_{j=0}^{n-1} (a_j^L - y_j)^2$$

n ...
L ...
a ... berechnetes / aktuelles Ergebnis
y ... gewünschtes Ergebnis
j ...

berechnet für alle Ausgabe-Möglichkeiten (Summe der n Möglichkeiten) die Differenz (Klammer-Ausdruck) zwischen dem berechneten / erzielten Ergebnis (a) und dem gewünschten Ergebnis (y, Ziel-Wert)

Quadrierung dient der besonderen Aufwertung von großen Abweichungen und der Eliminierung von negativen Fehlern

Ziel ist ein möglichst kleiner Fehler-Wert, was möglichst geringen Kosten entspricht

durch die Anpassung von Gewichten, Bias-Werte und ev. der Schwellen-Werte wird eine Verringerung des Fehler-Wertes angestrebt

für die Suche nach dem Minimum benutzt man die 1. Ableitung

aus den Nullstellen der 1. Ableitung können wir die Extrem-Punkte (deren x-Wert) erschließen

der y-Wert ergibt sich dann über das einsetzen des jeweiligen x-Wertes in die Stamm-Funktion

die Frage, ob es sich nun um ein Maximum oder ein Minimum handelt, wird über die 2. Ableitung zugänglich

wieder werden von dieser die Null-Stelle(n) berechnet und

für mehr-dimensionale Probleme - wie die üblichen Bild-Erkennungen - werden Gradienten ermittelt

als Gradient versteht man dabei den Vektor dessen Komponenten der Werte der partiellen Ableitung der Variablen der Funktion f ausgehend von dem Punkt P

geschrieben als ∇f

jeder Eintrag im Vektor (Komponente) spiegelt den Anstieg der Funktion für eine Variable wieder

und jeder Eintrag in dem Fehler-Vektor stellt dann die notwendige Anpassung des Gewicht's / Bias dar

Gradient für eine Schicht ist ein Produkt der partiellen Ableitungen der folgenden Schicht

mit dem Gradienten-Abstiegs-Verfahren (Gradient Descent) wird jetzt die Schritt-weise Anpassung der Gewichte / Bias-Werte vorgenommen, bis man das Minimum der Kosten-Funktion erreicht

$$w' = w - \eta \nabla E$$

w' ... neuer Gewichte-Vektor
w ... alter / gegebener Gewichte-Vektor
n ... Lern-Rate (Learning rate)
 ∇E ... Gradient der Kosten-Funktion

zuerst einmal findet man so nur ein lokales Minimum
globale Minima zu finden ist derzeit noch Forschungs-Thema

Initialisierung der Gewichte / Bias-Werte ist ein weiteres Problem
ebenfalls noch nicht allgemein gelöst
deshalb Verwendung von Zufalls-Werten als Initialisierungs-Parameter
ev. mehrere Durchläufe des Trainings mit unterschiedlichen Start-Werten, ein gutes KNN
wird am Ende sehr ähnlich aussehen

Fehlerflächen in "eni04—perzeptron.pdf"

Lern-Phase besteht aus Trainings-Phase und der Validierungs-Phase
annotierte / gelabelte Daten werden dafür aufgeteilt (z.B. 4/5 für Training und 1/5 für's Testen)
eine Lernphase - der Durchlauf aller Testdaten - heißt Epoche
meist werden die gleichen (weil teuren) Lern-Daten für neue Lern-Phasen (Epochen) wieder-
verwendet. Sie werden neu aufgeteilt (z.B. anderes Fünftel als Test-Daten und der Rest wie-
der als Trainings-Daten).

$$A = \frac{\text{korrekt_klassifizierte_Testdaten}}{\text{Anzahl_Testdaten}}$$

A ... Akkuratheit (Accuracy)

Definition(en): Epoche

Eine Epoche ist der Durchlauf aller Trainings-Daten durch das Modell.

-Funktion / Categorical Cross Entropy

$$E_y(a) = - \sum_{j=0}^{n-1} y_j \cdot n \log(a_j)$$

n ...
L ...
a ... berechnetes / aktuelles Ergebnis
y ... gewünschtes Ergebnis
j ...

in der Praxis bevorzugt
besonders für Computer Vision (Bild-Erkennung) scheinbar am Besten geeignet
dieser Fehler betrachtet nur den Fehler bezüglich einer / der erwarteten Kategorie
nur für diese ist der y_j -Wert ja 1 für alle anderen ist er i.A. 0
man braucht also auch nur einen Logarithmus zu berechnen, den für die aktuelle Wahr-
scheinlichkeit (a_j)

Varianten der "Categorical Cross Entropy"-Fehler-Funktion

- "categorical_crossentropy"
- "sparse_categorical_crossentropy"

•
 praktisch zeigen die beiden Fehler-Funktionen "categorical_crossentropy" und "sparse_categorical_crossentropy" identische Lern-Kurven für die Wahl der zu nutzenden Funktion sind die Formen der bereitgestellten Labels entscheidend

"categorical_crossentropy" erwartet ein Array mit sovielen Elementen, wie es Kategorien gibt auf den Index der richtigen Kategorie muss ein 1 stehen, alle anderen Elemente müssen 0 sein

"sparse_categorical_crossentropy" erwartet ein Array mit den Kategorie-Bezeichnungen

Vergleich Fehler-Funktionen

	Mean Squared Error quadratische	Categorical Cross Entropy
Merkmale		stärke Bestrafung von größeren Abweichungen
Gemeinsamkeiten	haben das gleiche Problem mit Anomalien in Daten bei großen Fehlern funktionieren beide nicht mehr effektiv	
Unterschiede	betrachtet alle Kategorien	betrachtet nur die erwartete Kategorie
Vorteile		stärke Bestrafung von größeren Abweichungen
		betrachtet nur die konkreten Beschriftungen / Label's
		relativ einfache Berechnung (nur ein Logarithmus)
	verbessert sich über das Training immer mehr	
		gleichmäßiges Lernen über das gesamte Training hinweg
		bessere Ergebnisse für Kategorisierungen, z.B. Computer Vision (Bild-Erkennung)
Nachteile	Netz lernt immer langsamer, je weiter fortgeschritten es ist	

alternative Fehler-Funktion: mittlerer absoluter Fehler (Mean Absolute Error)

bei dieser Fehler-Berechnung haben starke Abweichung nicht so große Auswirkungen (da lineare Auswirkung)

n ...

$$E_y(a) = \frac{1}{n} \sum_{j=0}^{n-1} |a_j - y_j|$$

L ...
a ... berechnetes / aktuelles Ergebnis
y ... gewünschtes Ergebnis
j ...

weitere Fehler-Funktionen

- HINGE- Funktion
- KULLBACK LEIBLER-Funktion
-

2.6.x. Lern-Algorithmen

Backpropagation

Probleme beim Finden der Kosten-Minima ist zum Einen die Berechnung des / der Gradienten sowie die Frage nach der Effektivität des Such-/Optimierungs-Verfahrens

übliche Schreibweise für Funktionen ist $f(x)$
die erste Ableitung wäre dementsprechend $f'(x)$

bei mehr-dimensionalen Funktionen mit mehreren Variablen kann man aber immer nur bezüglich einer Variable ableiten
man spricht nun von partiellen Differentialen / Ableitungen

für $f(x, y)$ gilt $\frac{\delta f}{\delta x} = 1 \cdot y$, wenn man nach x ableitet. In diesem Fall wird y als andere Variable als eine Konstante betrachtet, Genau gilt $\frac{\delta f}{\delta y} = x \cdot 1$, wenn man nach y ableitet.

da wir bei den Kosten-Funktionen immer mehr-dimensionale Gebilde vorliegen haben, werden immer die partiellen Ableitungen betrachtet

für die Funktion $f'(x)$ sieht die 1. Ableitung in der Partial-Darstellung dann so aus $\frac{\delta f}{\delta x}$
beide Schreibweisen sind äquivalent

berechneter Gradient wird solange von den Gewichten subtrahiert bis Minimum erreicht ist

Optimierung durch Zufälliges / Stochastisches Gradienten-Abstiegs-Verfahren (Stochastic Gradient Descent)

erweitert "einfaches" Gradienten-Abstiegs-Verfahren

für die praktische Anwendung in einem KNN müsste der Mittelwert aus allen Gradienten berechnet werden

bei einem Datensatz von 10'000 Elementen müsste man erst die 10'000 Daten-Elemente durchlaufen lassen, um dafür den Mittelwert (des Gradienten) zu berechnen
dieser kann dann zur Anpassung der Gewichte benutzt werden

effektiver ist es eine zufällige Auswahl von Daten-Elementen für einen Durchlauf (genannt Batch) zu wählen, dann die Anpassungen zu berechnen und anzuwenden und dann mit weiteren ausgewählten Daten weiter zu machen

Probleme des Gradienten-Abstiegs-Verfahrens

- Erfolg hängt von der Wahl der Lern-Rate ab
 - zu große Lern-Rate führt zwar zum schnelleren Lernen, das Minimum kann aber verfehlt werden
 - eine zu kleine Lern-Rate erfordert langwierige Lern-Runden
-

viele Optimierungen entwickelt (z.B. QuickProp) mit unterschiedlichem Erfolg bei unterschiedlichen KNN und Modellen

2.6.x. Daten-Bereitstellung

praktisch der Arbeits- und Zeit-aufwendigste Teil bei Daten-Analyse auf der Ebene von Computer Science, ... und Machine Learning

mögliche Daten-Quellen

- **Sensoren**
- **alle Dateien des PC's**
- **eMails**
- **Bilder-Sammlungen**
- **...**

Daten-Strukturen

- **Datenbanken** bestehen meist aus vielen (verknüpften) Tabellen
- **Tabellen (Relationen)** klassische Form der Sammlung von Daten in Spalten und Zeilen
eigentliches Datum ist in der Zelle (definiert durch Spalte und Zeile) abgelegt
- **Dateien** Sequenzen (Aneinander-Reihungen von Nullen und Einsen → Interpretation obliegt dem bearbeitenden Programm)

• Baum-System	klassische Struktur sind binäre Bäume, deren Links-Rechts-Verzweigung über Ja-Nein-Entscheidungen weiter verfolgt werden kann (bis ein Blatt – das eigetliche – Datum erreicht wird)
• Dokumente	Texte mit eingefügten Bildern, Hyperlink's, ...
• ...	

Arten von Daten

• strukturierte Daten	festes Schema sehr häufig tabellarisch → relationale Datenbanken sehr leichtes Parsen / Analysieren Daten-Formen (Datei-Typen): <ul style="list-style-type: none"> • XLSX ... Excel-Tabellen-Kalkulation • CSV ... Komma-getrennte Daten in Text-Form • TXT ... unformatierte Texte • ...
• semistrukturierte Daten	nur geringe Orientierung an einem übergeordneten Schema oft selbst-beschreibend definiertes Aufbau-Prinzip relativ einfach Parsen / Analysieren, wenn das Schema oder das Anordnungs-Prinzip bekannt ist Daten-Formen (Datei-Typen): <ul style="list-style-type: none"> • XML ... Schlüssel-Wert-Paare mit beschreibenden Tag's • JSON ... selbst-beschreibende Schlüssel-Wert-Paare • ...
• unstrukturierte Daten	ohne festes Schema können binär oder auch Text-orientiert angelegt sein oft besonders interessant für KI-Anwendungen, vor allem auch deshalb, weil sie mit üblichen / klassischen informatischen Verfahren nur schwer zu analysieren sind DatenTypen <ul style="list-style-type: none"> • Bilder • Texte • Video's • Audio-Daten Daten-Formen (Datei-Typen): <ul style="list-style-type: none"> • DOCX ... formatierte Text-Dokumente • BMP ... Speicher-Abzüge (z.B. Bilder) • JPG ... komprimierte Bild-Daten • MP3 ... komprimierte Audio-Daten • MP4 ... komprimierte Video-Daten • PNG ... komprimierte Bild-Daten

• ...

Daten-Typen

• Texte	Sequenz von alphanummerischen Zeichen.
• Bilder	
• Musik	
• Video's	Bewegt-Bilder
• Zahlen	reine Zahlen-Daten (z.B. Ganzzahlen, Kommazahlen, ...)
• Binär-Daten	Daten mit unklaren / unbekannten / verschlüsselten Strukturen bzw. Inhalten
• ...	

Daten sind oft über Verweise (z.B. oder so ähnlich wie Hyperlink's) miteinander verknüpft

besondere Form sind Graphen (Graph-Daten)
diese bestehen aus Knoten (den eigentlichen Daten-Objekten) und den Kanten (den Verbindungen zwischen den Daten-Objekten)

Roh-Daten müssen meist aufbereitet werden, damit sie für das Modell (Verarbeitungs-System / KNN nutzbar sind
Roh-Daten können in verschiedenen Skalierungen / Skalen vorliegen

Daten-Skalen

- **nominal kategorisch** ohne innere Ordnung
z.B. gelb, grün, blau, schwarz
z.B. gross, klein, dünn, frech, ...
z.B. Hund, Katze, Maus, Elephant, ...
- **ordinal** besitzen innere Ordnung
haben Reihen- oder Rang-Folge
unterscheiden sich systematisch von Vorgänger und Nachfolger
Berechnungen sind nicht damit möglich
man kann aber z.B. den soundsovielten Nach-Folger / Vorgänger bestimmen
z.B. A, B, C, D, ...
z.B. Schulnoten

z.B. Montag, Dienstag, Mittwoch, ...

- **metrisch** besitzen feste innere Ordnung
Daten werden durch Zahlen dargestellt, über die auch Berechnungen erfolgen können
z.B. 1, 2, 3, 4, ...
z.B. 1,0; 1,2; 1,4; 1,6; ...

Um nominale Daten sinnvoll und effektiv mit einem Computer zu verarbeiten, werden diese Daten in Rangfolgen / Reihen oder noch besser in Zahlen dargestellt. Dies nennen wir auch Codierung.

Problematisch ist bei einer solchen Umcodierung die Abtrennung der Bedeutung und auch ihres Inhalts

Praktisch wird eine innere Ordnung erzeugt, die in den originalen Daten gar nicht vorhanden ist.

Mit den Zahlen lassen sich auch einmal Dinge machen, die an den Real-Daten-Objekten niemals vollzogen würden. Hat man z.B. Äpfel mit 2, Birnen mit 3 und Pflaumen mit 4 codiert, dann lässt sich recht gut der Durchschnitt 3 berechnen. Zurück-kodiert ergibt das dann Birnen, was sicher Fragen aufwirft.

Frucht	Fruchtcode
Apfel	2
Birne	3
Pflaume	4
Apfel	2
Pflaume	4
Birne	3
Apfel	2
Pflaume	4
Pflaume	4

Zudem lassen sich die Daten sehr unterschiedlich codieren

Als recht gut geeignet hat sich die One-Hot-Encodierung herausgestellt.

Für jeden Daten-Wert wird eine Spalte eingeführt und dann im Passungs-Fall eine 1 gesetzt, sonst eine 0. In dieser Codierung gibt es keine Rangfolge.

Selbst das nun mögliche "Rechnen" – z.B. Addieren aller Spalten-Werte ergibt noch was Sinnvolles.

So könnte man nun mit 3 Äpfeln, 2 Birnen und 4 Pflaumen weiterarbeiten.

Frucht	Apfel	Birne	Pflaume
Apfel	1	0	0
Birne	0	1	0
Pflaume	0	0	1
Apfel	1	0	0
Pflaume	0	0	1
Birne	0	1	0
Apfel	1	0	0
Pflaume	0	0	1
Pflaume	0	0	1

es werden i.A. große Daten-mengen gebraucht

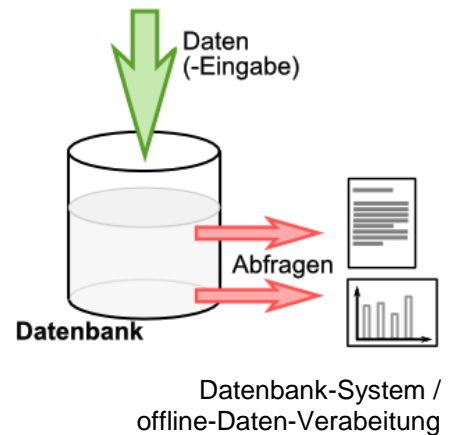
Teilung des Daten-Bestand's in Trainings- und Test-Daten
nur mit den Trainings-Daten wird das Modell trainiert
die Test-Daten werden ausschließlich zum Prüfen des Netzes benutzt
saubere Trennung unabdingbar für ein erfolgreiches Entwickeln eines KNN's
ansonsten droht Overfittig

noch besser ist Drei-Teilung der ursprünglichen Daten in Trainings-, Validierungs- und Test-Daten
jede Gruppe ist sauber von der anderen getrennt

die Trainings-Daten werden für den eigentlichen Lern-Vorgang (Training) benutzt dabei werden verschiedene Kombinationen von Hyperparametern getestet das Netz, bei dem die Validierungs-Daten das beste Ergebnis bringt wird dann final mit den Test-Daten getestet

Zeitpunkt und Lage der Daten-Verarbeitung

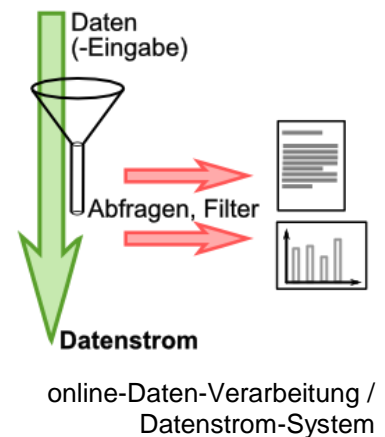
Batch-Daten



Streaming-Daten

z.B. gleitende Mittelwerte berechenbar
aktuelle Visualisierung

ev. müssen die daten gespeichert werden, um sie
später noch einmal zu nutzen



in modernen Systemen können aber auch sinnige Kombinationen genutzt werden
z.B. kann man in einem Chat-Analysator das System mit gespeicherten Batch-Daten anlernen / trainieren und dann produktiv auf Streaming-Daten anwenden

Daten-Quellen / Daten-Beschaffung

eigene Daten

vorhandener Datenbestand (z.B. eine Datenbank)

historisch gesammelte Daten (Verkäufe, Umsätze, ...)

IoT-Daten (Sensoren, ...)

initiiertes Sammeln (neuer Daten) (z.B. durch Umfragen, ...)

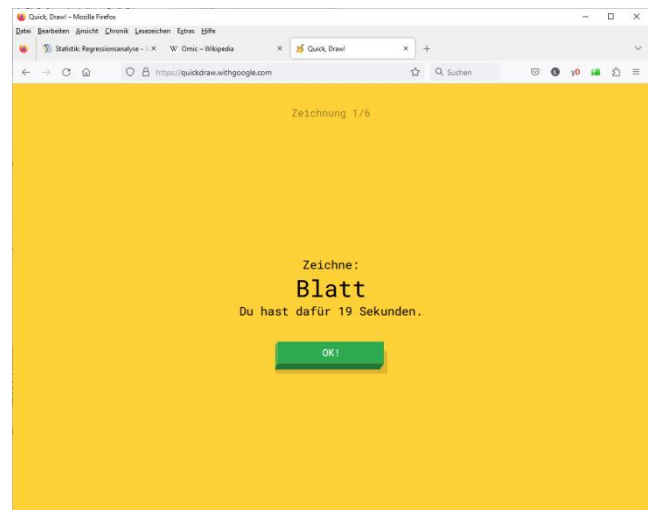
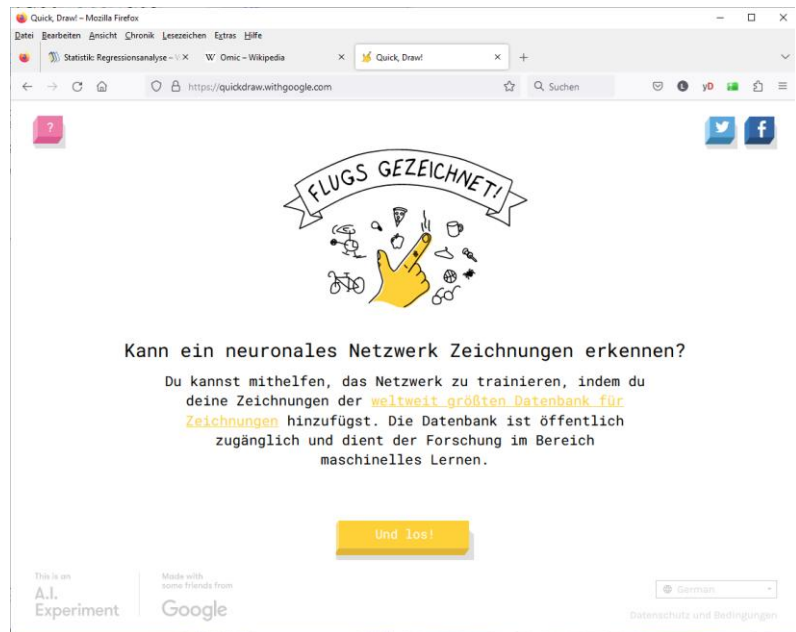
offene Quellen

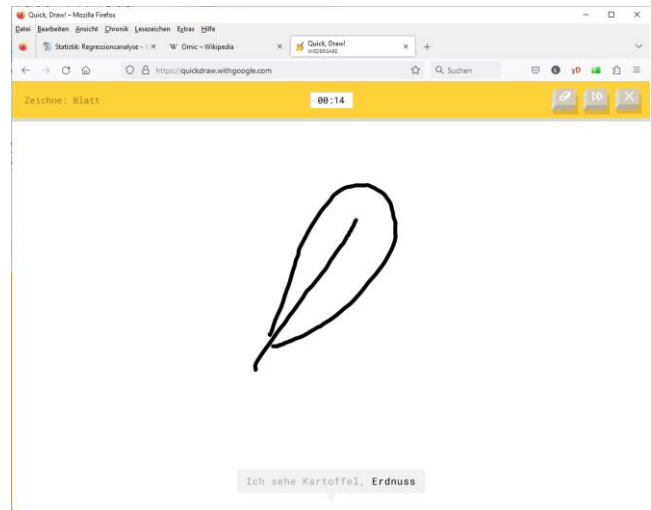
Web-Crawling

Daten-Kauf

online-Daten-Stream's von online-Anbietern (z.B. Twitter, ...) über deren API

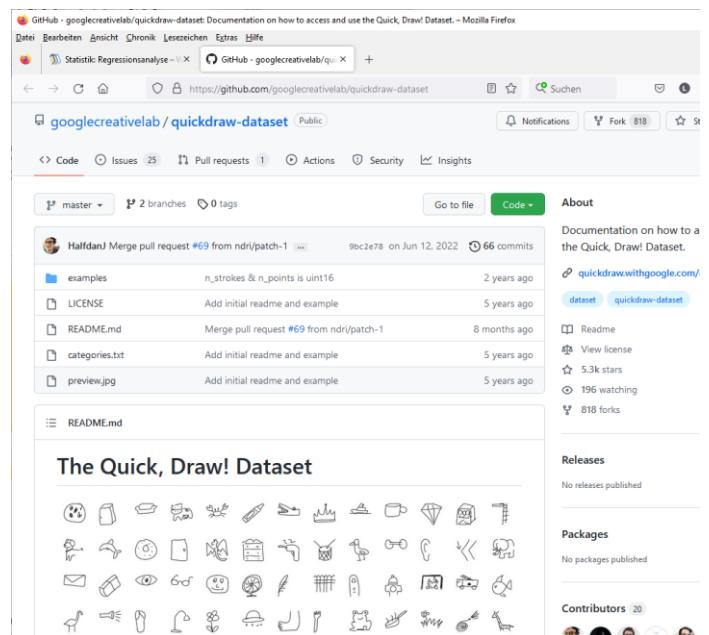
oder man animiert Menschen zum Erstellen und annotieren von Skizzen (→ <https://quickdraw.withgoogle.com>)





Q: <https://quickdraw.withgoogle.com> (Feb. 2023)

Informationen und Daten zu Quick-Draw: → <https://github.com/googlecreativelab/quickdraw-dataset>



Q: <https://github.com/googlecreativelab/quickdraw-dataset> (Feb. 2023)

Daten-Entschlüsselung

Beispiel

Buchungs-Nummer	Buchungs-Schlüssel	Buchungs-Datum	Bezeichnung	Menge	Ort
753	Warenausgang	06.03.2020	Farbe "polarweiss	143 kg	Rostock
754	Wareneingang	06.03.2020	Farbe "polarweiss	84 kg	Berlin
755	Wareneingang	07.03.2020	Zier-Kies	6012 kg	Rostock
756	Wareneingang	07.03.2020	Farbe "polarweiss	59 kg	München

84 kg der Farbe "polarweiss" ist von Rostock nach Berlin gegangen

59 kg der Farbe "polarweiss" ist von Rostock nach München geliefert worden

Daten-Labeling

für Supervised und Semisupervised Learning werden unbedingt gelabelte / annotierte Daten gebraucht
für das passgerechte Training des Modell's
zur Validierung / Testung des Modell's

Arten des Daten-Labeling's

- **implizit** Nutzung schon vorhandener Label's oder direktes Ableiten von Label's aus den Daten
- **explizit** manuelles Annotieren
- ...

Gegenüberstellung Bild-Erkennung (und -Annotation) bei Mensch und Maschine

Mensch	Maschine
erkennen (mit gleicher Qualität) nur eine begrenzte Anzahl von Kategorien	erkennen unbegrenzte Anzahl von Kategorien
meist Experten notwendig	
Gesamt-Erkennungs-Qualität gleichbleibend	Gesamt-Erkennungs-Qualität steigt mit der Anzahl vorliegender Bilder
ermüden schnell begleitet von sinkender Erkennungs-Qualität	keine Ermüdung od. Veränderung der Erkennungs-Qualität
beim Hinzufügen neuer Kategorien muss von	Grund auf neu gelernt / annotiert werden
	ev. sogar das Modell verändert werden

ev. ist Label schon in den Original-Daten schon enthalten oder ableitbar
oft aber auch externe Annotation notwendig → sehr hoher Aufwand

Captcha-Bilder sind oft Unterscheidungs-Funktionen von Computer und Menschen, aber auch unbemerkt Labeling durch die Nutzer

auch Texte, die Hashtag's am Ende haben, können solche unbemerkten Annotationen durch den Nutzer / Leser sein

weitere Möglichkeit ist die Anwendung eines Unsupervised Learning-Modell's
dieses erzeugt Cluster von Objekten (Texten, Bildern, ...), die dann als Cluster gelabelt werden

unter Nutzung von Transfer Learning
für Daten-Bestände mit relativ wenigen Datensätzen (ev. bei Spezial-Daten) kann man es auch mit KKN's versuchen, die auf anderen großen Daten-Beständen trainiert worden
ev. hinterher nur noch Feintuning des Modell's notwendig

aktive Learning

während des manuellen Labeling's werden weitere Modelle in den Labeling-Prozess zuge-schaltet
unterstützt das Labeling; KNN schlägt Label's vor, diese brauchen dann nur noch bestätigt werden
außerdem lassen sich so Daten-Bestände identifizieren, die unsichere Daten darstellen
hier werden dann ev. zusätzliche Datensätze benötigt oder werden manuell gelabelt

Weak Supervision

Kombination / Schnittstelle zwischen KI-Programmierung und der herkömmlichen / klassi-schen Programmierung
es werden Heuristiken aufgestellt, die für das Domain-Wissen stehen
dabei wird Domain-Wissen programmatisch gelabelt

Definition(en): Weak Supervision

Weak Supervision ist ein Teilgebiet des maschinellen Lernen's, bei dem verrauschte (noisy), begrenzte oder ungenaue Quellen automatisiert kombiniert werden, um eine bereinigte Informations-Quelle für ein programmatisches Labeln zu erzeugen.

neue Techniken
Anreichern der Daten-Bestände mit ableitbaren Merkmalen (Text-Länge, Anzahl Schlüssel-wörter, ...)
Teilen des Daten-Bestandes
Kombination von manueller Annotation (Gold-Label's) und skalierbares Labeling (program-matisch erstellte Label's)

MNIST-Datensatz

MNIST = Modified National Institute of Standards and Technology

quasi das Hallo-Welt-Beispiel für KNN

stellt Ziffern von 0 bis 9 dar, handgeschrieben, als Pixel-Bilder (28x28 px)
besteht aus 60'000 Trainings-Daten und 10'000 Test-Daten

die 784 Pixel stellen in jeweils 8 bit den Schwarz-Wert dar (je größer die Zahl, umso dunkler dar Pixel) weiss wäre somit mit 0 und schwarz mit 255 kodiert

Anforderungen an Trainings-Daten für KNN

- ausreichend große Anzahl von Elementen

-
- sollte sich auch noch in Trainings-Daten (optimal 75 %) und Test-Daten (restliche 25 %) aufteilen lassen
 - keine Dopplungen / Duplikate
 - Daten müssen von Menschen beschriftet / annotiert / gelabelt sein
 - möglichst hohe Variabilität (wie in der Praxis / Realität vorkommend)
 - möglichst wenig Stördaten (Noise, Rauschen), wenn doch Stördaten, dann sollten diese wieder gleichverteilt bezüglich der Entscheidungs-Klassen verteilt sein

Aufbau eines passenden KNN für den MNIST-Datensatz und das Ziffern-Erkennungs-Problem

da Ziffern 0 bis 9 erkannt werden sollen, sind genau 10 Ausgabe-Neuronen notwendig, diese bilden die Ausgabe-Schicht (output-Layer)

als Ausgabe-Werte sollen Werte von 0 bis 1 auf jedem Kanal herauskommen, die die Wahrscheinlichkeit ausdrücken, dass es die Kanal-Ziffer ist

ist also das Signal auch Neuron 4 mit 0,7 vorhanden, dann heißt dies, dass die 4 mit einer Wahrscheinlichkeit von 70 % erkannt wurde

(vielleicht ist die Wahrscheinlichkeit, dass es sich um eine 1 handeln könnte 40 %) es zählt am Ende der Kanal, der den höchsten Ausgabe-Wert hat

die Anzahl der Eingabe-Neuronen ergibt sich aus dem Daten-Umfang für ein Element (also ein Ziffern-Bildchen) mit 784 Pixeln (28x28)

somit ist die Eingabe-Schicht (Input-Layer) aus 784 Neuronen aufgebaut

zwischen Eingabe-Schicht und Ausgabe-Schicht liegen ein bis mehrere interne / versteckte Schichten (Hidden Layer)

dabei ist jede Ausgabe eines Neuron's mit allen Eingaben der folgenden Schicht verbunden

Neuron bekommt von einem oder mehreren Vorgängern (der darunterliegenden Schicht) ein Eingangssignal, dieses wird jeweils mit einem Gewicht bewertet

die Gewichte bestimmen also, wie stark das Ausgabe-Signal der Vorgänger-Zelle als eigenes Eingabe-Signal genutzt werden soll

im Neuron werden die Eingänge ausgewertet

dazu wird eine Aktivierungs-Funktion benutzt

die Funktion vergleicht die Summe der gewichteten Eingangswerte mit dem Schwellenwert (dem Bias)

die Funktion gibt somit an, ob das Neuron aktiviert wurde (also eine 1 ausgibt) oder inaktiv bleibt (also eine 0 ausgibt)

das gesamte Netz wird von uns als Black Box betrachtet, wir haben nur Eingabe- und Ausgabe-Werte, was genau in der Box (in den Hidden-Layern passiert ist nicht direkt wichtig und praktisch nur mit einem extrem hohen Aufwand nachzuvollziehen

IMAGENET-Datensatz

14 Mio. hand-beschriftete Bilder

20'000 Klassen mit jeweils rund 700 Bilder pro Klasse

normalerweise werden nur Teil-Mengen dieses Daten-Set's benutzt

Anomalien in Daten

treten durch falsch beschriftete Trainings- und Test-Daten auf (menschlicher Fehler)
zu undifferenzierte Bilder
ungewöhnliche Ansichten von Objekten / ungewöhnliche / seltene Daten für Objekte

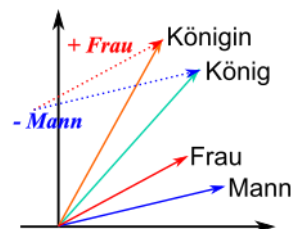
Informations-Extraktion aus Texten

Merkmal über den gesamten Text-Körper hinweg. Z.B. kann man relativ leicht ermitteln:

- Anzahl der Wörter
- Anzahl der Buchstaben
- durchschnittliche Anzahl von Buchstaben je Wort
- Anzahl von Ausrufe-Zeichen
- Anzahl von Groß-Buchstaben (mit und ohne Satz-Anfänge)
- Anzahl bestimmter (Schlüssel-)Wörter oder auch aller Wörter → Bag-Of-Words (→)
- Länge von Absätzen (z.B. Anzahl der Sätze je Absatz)
- Anzahl der Absätze je Kapitel
- ...

Für echte Text- und Sprach-Analysen kommt es auch auf den Kontext und die Bedeutung der Worte an. Hier sind wir dann im Bereich von "Word Embedding" (→)

"Königin" = "König" – "Mann" + "Frau"



Aufgaben:

- 1.
2. *Bei den Daten-Skalen sind die Schulnoten bei den ordinalen Daten eingeordnet worden. Ist dies wirklich richtig? Diskutieren Sie die Fragestellung!*
- 3.

2.6.x. Evaluations-Metriken

Bewertung der Modelle über Evaluations-Metriken
qualitative Beurteilung der Modelle anhand der realen Arbeits-Daten
je nach Methode des Machine Learning werden unterschiedliche Evaluierungs-Methoden angewendet

2.6.x.y. Evaluierung von supervised bzw. semisupervised Learning

vorliegende Daten

große Beispiel-Tabelle

Aufteilung in Trainings- und Test-Daten

typischerweise 80% Trainings-Daten und 20% als test-Daten
diese stehen für angenommene Daten des Live-Betrieb's

Trainings-Daten

geteilte Beispiel-Tabelle

Test-Daten

geteilte Beispiel-Tabelle

Test-Bewertung

Annahme
Betroffene machen 10% aus

"Ground Truth"

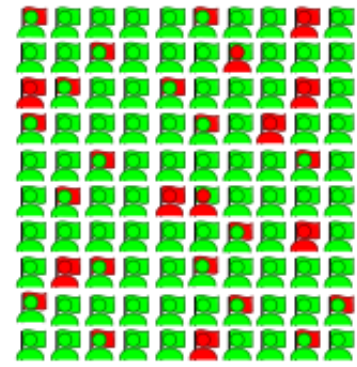
z.B. Aussage Test hat eine Genauigkeit von 80%
d.h. bei 8 von 10 getesteten Betroffenen ist das Test-Ergebnis "positiv". Gemeint ist hier "positiv" bezüglich des Test's – er hat also angeschlagen.

Für die Betroffenen persönlich ist das Ergebnis wahrscheinlich nicht positiv, außer dem Fakt, dass die Betroffenheit auch festgestellt wurde.

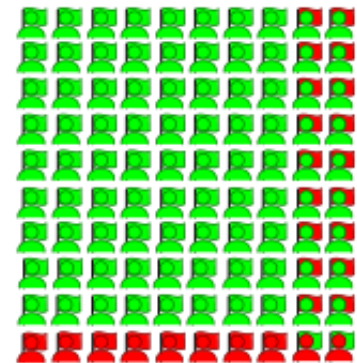
Ein Problem ist natürlich die FALSCH-Bewertung von Betroffenen. Bei ihnen ist der Test fehlgeschlagen und sie werden als Nicht-Betroffen gemeldet.



Noch dramatischer ist aber das durch die Test- Ungenauigkeit erzielte Ergebnis für Nicht-Betroffene. Sie werden als FALSCH-Positive geführt. Vielleicht wird nun eine falsche Behandlung eingeleitet, die diesen Personen-Kreis dann wirklich erkranken lässt (natürlich nicht an der betrachteten Krankheit sondern an Behandlungs-(Neben-)Wirkungen). Von den eigentlich 90% Nicht-Betroffenen ereilt dieses dramatische Schicksal immerhin 20%. Das sind in unserem Beispiel 18 Personen – also 18% der ursprünglichen Test-Gruppe.

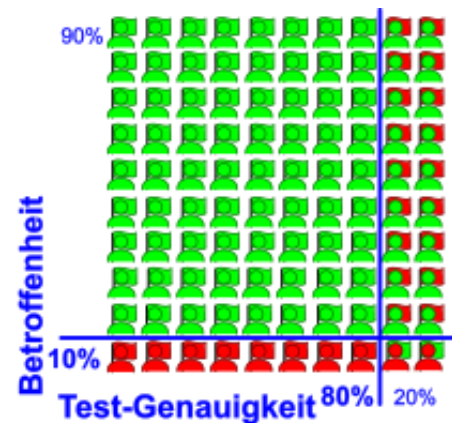


Ein Test mit dieser – scheinbar hohen – Genauigkeit, hätte wohl kein gutes Image. Sortiert man die Personen ein bisschen, dann kann erhält man z.B. die nebenstehende Anordnung. Von oben nach unten werden die z.B. die Betroffenen und Nicht-Betroffenen angeordnet. Innerhalb dieser Gruppen sortiert man dann von rechts nach links die so oder so getesteten Objekte.

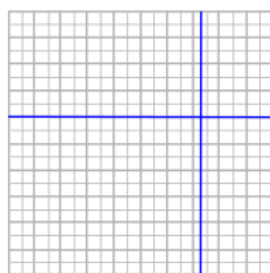


besser ist die Benutzung der sogenannten Konfusions-Matrix
Zusammentreffen von Wahrheit und Meinung (Konfusi-
on bedeutet im Recht: Zusammentreffen von Gläubiger und
Schuldner)
trifft aber auch ein bisschen das Verständnis-Problem
– die Sache ist leicht verwirrend

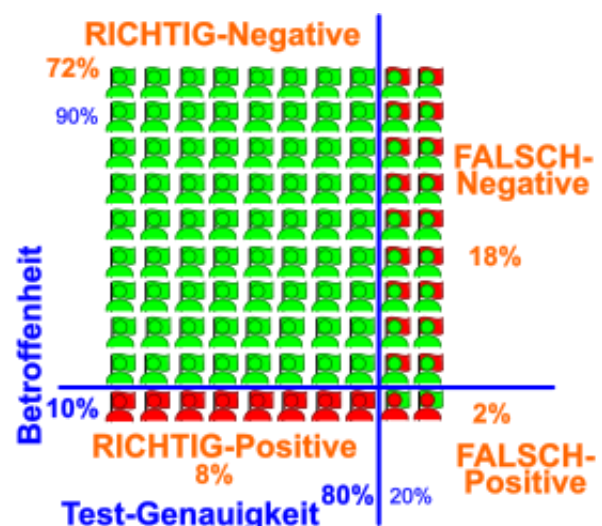
Für Test's mit krummen Prozentzahlen kann man
vielleicht für eine Visualisierung die Zahlen auf- bzw.
abrunden oder man wählt eine größere Matrix. Bei
Prozentzahlen, die durch 5 teilbar sind reicht eine
20x20-Matrix. Ansonsten kann man immer auf einen
100x100-Matrix zurückgreifen. Die Personen od.ä.
sollte man dann natürlich nicht mehr zeichnen.



Ein klassisches 10x10cm-Quadrat auf Millimeter- oder kleinkarriertem Papier ist immer eine gute Grundlage.



Beispiel für eine Darstellung auf Millimeter-Papier



Wahrheit zur Vorhersage	Aussage der Vorhersage	Kürzel
True	Positive	TP
True	Negative	TN
False	Negative	FN
False	Positive	FP

Wahrheit Realität	Vorhersage Prognose	
	Vorhersage A Test negativ als "Nicht-Betroffen" markiert Negative Prediction	Vorhersage B Test positiv als "Betroffen" markiert Positive Prediction
aktuell A IST A Nicht-Betroffene Negative Class	TN (True Negative) RICHTIG-Negative gefundene Nicht-Betroffene	FP (False Positive) FALSCH-Positive fälschlich als Betroffen gefunden
aktuell B IST B Betroffene Positive Class	FN (False Negative) FALSCH-Negative nicht-gefundene Betroffene	TP (True Positives) RICHTIG-Positive gefundene Betroffene

Wahrheit	Vorhersage	
	A	B
A		
B		

normiert	Vorhersage		Sum.
Wahrheit	A	B	A+B
A			
B			
Summe			

Aufgaben:

- 1.
2. *Wie sehen die Test- und Auswerte-Quadrate (einschließlich der Konfusions-Matrix) für ein Ham-Spam-Klassifizierungs-Modell aus, wenn dieses eine Treffer-Genauigkeit von % hat? Bei manuellen Auszählungen wurde für die Spam-Mail's ein Anteil von % ermittelt.*
- 3.

Precision gibt an, wieviele der als "positiv" gekennzeichneten "Prognosen" (des Modell's) auch wirklich in der Realität "positiv" sind

Recall gibt an, wieviele der wirklich "positiven" Objekte auch als "positive" prognostiziert / gefunden wurden

Generalisierung

drückt aus, dass das Modell auf neue (Echt- / Praxis- / Anwendungs-)Daten ähnlich gute Ergebnisse erzielt, wie auf den Trainings-Daten

Modell hat für die neuen Daten die gleiche / eine ähnliche Prognose-Güte wie bei den Trainings- und Test-Daten

Recall auf Trainings-Daten und neuen Daten gleichgroß

Definition(en): Generalisierung

.

.

Wenn ein Modell auf den Trainings-/Test-Daten deutlich besser funktioniert, als bei neuen Daten, dann spricht man vom Overfitting.

Das Modell hat praktisch die Trainings-Daten extrem genau (quasi auswendig) gelernt. Das Modell stellt mehr eine 1 : 1-Übersetzung der Trainings-Daten dar, als ein "Verständnis" für das Modell-Problem.

Definition(en): Overfitting

Overfitting.

.

Bekommt man weder bei den Trainings- noch bei den nachfolgenden neuen Daten ein akzeptables Ergebnis. dann spricht man vom Underfitting.

Das Modell ist in dieser Form / Konfiguration nicht für die Lösung des Modell-Problem's geeignet

Definition(en): Underfitting

.

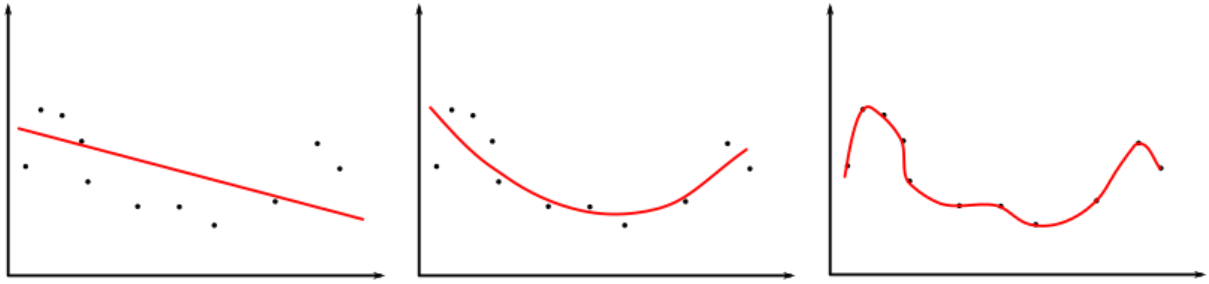
.

Trainings-Daten

Modell soll Zusammenhang repräsentieren

drei unterschiedliche Modelle

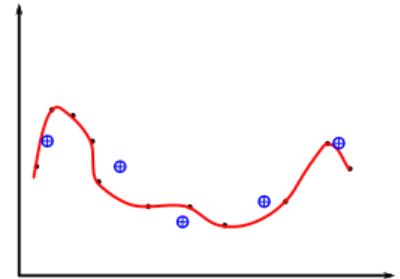




Modell 3 zeigt optimale Repräsentation der Daten im Modell

neue Daten (blaue Daten-Punkte) passen aber nicht / genau so gut ins Modell

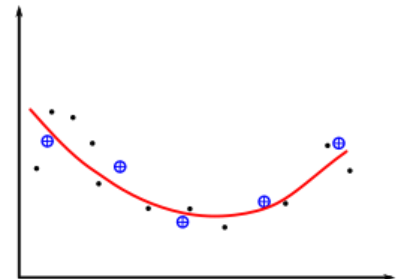
Overfitting



Modell stellt eine Funktion dar, die groben Verlauf / Zusammenhang der Trainings-Daten repräsentiert

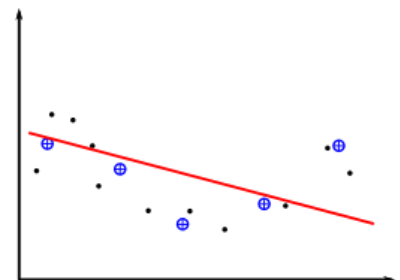
mit den neuen Daten erreicht man die gleiche – wenn auch nicht ganz optimale - Passung

gute Generalisierung



dieses Modell besitzt eine schlechte Repräsentation der Trainings-Daten
auch die neuen Daten liegen weitab von einer optimalen Passung

Underfitting



Nur wichtig ist am Schluß: Wie gut kommt unser Modell mit den neuen Daten klar?

2.6.x.y. Evaluierung von unsupervised Learning

Bewertung von Clustern

sachlich fehlt die "Ground Truth", die dem Modell zugrundeliegende Wahrheit

wichtige Frage ist deshalb, wieviele Cluster gibt es / wieviele Cluster sind geeignet, das Problem zu klären
sind viele kleine Cluster, mit speziellen abgebildeten Merkmals-Kombinationen oder große Cluster mit abstrakten Merkmalen gefragt

nach der Festlegung der Anzahl der Cluster wird berechnet
für die Cluster werden die Mittel-Punkte (Cluster-Zentroide) berechnet
die Abstände der Datenpunkte eines Clusters zum Zentroiden
liegen für alle Cluster die Datenpunkte bezogen auf ihren Cluster-Zentroiden dichter zusammen, als bei einer anderen angenommenen Cluster-Anzahl, dann ist dieses Cluster-Modell vorzuziehen

Gesamt-Bewertung über die Ellenbogen-Methode (Elbow Method)

Suche nach der idealen Cluster-Anzahl



2.6.x. Supervised Learning

*Erkennen von diebischen und nicht-diebischen Affen anhand von Symbol-Bildern (aus dem Material der Wissensfabrik.de)
→ it2school.de*

Festlegen von Regeln

Erkennen von neuen Bildern

→ Software-Tip: Orange 3

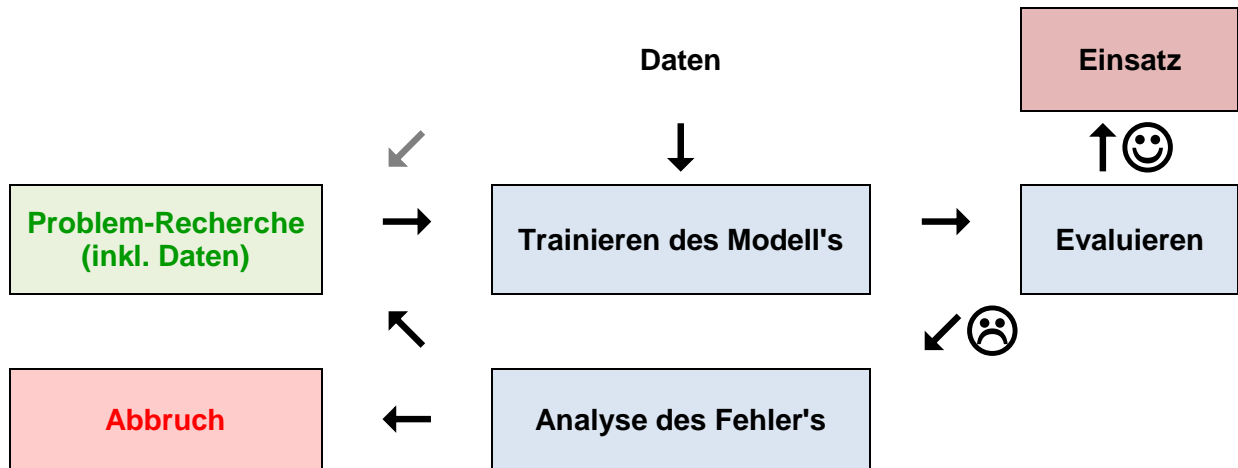
*Schüler lassen sich mit einer App altern und durch Details verändern (Brille, ...)
? Erkennt die Ki die Person noch
(→ FaceApp)*

→ it2school.de

zentrales Thema im Machine Learning
Grundlagen-Charakter
Kurz-Einführung (→ [2.6.x. Supervised Learning \(Kurz-Vorstellung\)](#))

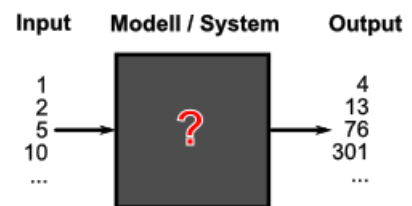
Ausgangs-Punkt ist das allgemeine Konzept des Machine Learning's

allgemeines Konzept des Maschinellen Lernens



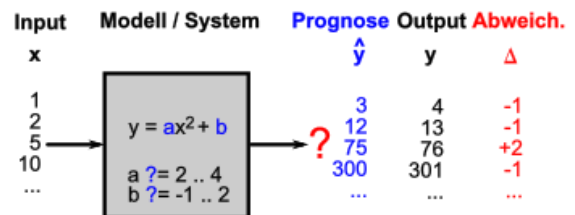
Box-Modelle des Machine Learning's

in das Modell gehen Eingabe-Daten (Input) ein, die in Form bestimmter Ausgaben (Output) verarbeitet werden dies sind unsere Prognose- / Arbeits-Daten, die mit y bezeichnet werden, die Eingabe-Daten sind die x -Werte wie genau die Verarbeitung erfolgt / erfolgen soll, bleibt erst einmal außen vor



Wir sprechen von einem Black Box-Modell, da das Innere für uns unsichtbar bleibt.

im Machine Learning wollen wir auch die innere Arbeit der Box nicht übernehmen, sondern das System (die Box) soll selbst eine Lösung finden dabei gibt es für die Box Vorgaben in Form von z.B. Funktionen



würden wir nun das Modell finden sollen, dann würden wir sicher die Werte der Parameter irgendwie austesten

dabei sollen die vom Modell berechneten Werte (\hat{y}) natürlich bestmöglich mit dem originalen Output des Black Box-Modell übereinstimmen

im Machine Learning soll nun das System die Parameter selbst herausbekommen

Wie es das macht, bleibt für uns erst einmal noch verdeckt.

Da das Modell nun schon ein bisschen durchsichtiger wird, sprechen wir hier von einem Gray Box-Modell.

Trainings-Arten des Modells

- **parametrisierte Modell**

es liegen für das Training die (historischen) Eingabe- und Ausgabe-Daten sowie die Annotationen der Ausgabe-Daten vor

aus diesen heraus werden (automatisiert) die Modell-Parameter bestimmt

mit neuen Daten wird das Modell getestet

der Erfolg / Nicht-Erfolg des Modell's wird dann zur Modell-Optimierung genutzt

das Modell besteht aus Parametern

Training erfolgt über den Aufbau und die Anpassung der Parameter

für das Testen des Modell's sind die ursprünglichen / historischen Daten nicht direkt notwendig (müssen also z.B. nicht im Speicher gehalten werden)

z.B.:

- Entscheidungs-Bäume (Modell lernt welche Fragen am besten geeignet sind und leitet z.B. einen Entscheidungsbaum oder eine passendes Ensemble Learning-Modell ab → dieses Ergebnis kann ohne die Trainings-Daten benutzt werden)
- Regression (Modell lernt, wie die Kurve verlaufen muss, wobei die Parameter der Funktion angepasst / optimiert werden → das fertige Modell besteht dann nur noch aus der Funktion (mit ihren Parametern) und kann ohne die Trainings-Daten verwendet werden)
- Künstliche Neuronale Netze (nach der manuellen Konstruktion des Netzes lernt das Netz die Gewichte → das trainierte Netz funktioniert dann in der Praxis-Phase ohne die Trainings-Daten)

- **non-parametrisierte Modelle**

gehört zum parametrisierten Modell-Training

- **Instanz-basierte Modelle**

es liegen für das Training die (historischen) Eingabe- und Ausgabe-Daten sowie die Annotationen der Ausgabe-Daten vor

Klassifizierung neuer Daten erfolgt über die Suche nach – schon bekannten – ähnlichen Daten
sachlich gar kein Lern-Prozess

für das Testen des Modell's werden immer alle ursprünglichen / historischen und annotierten Daten gebraucht (zum Vergleichen), sie müssen deshalb z.B. im Speicher zur Verfügung stehen

z.B.:

- k-nächste Nachbar (um neue Daten-Punkte einzuordnen, müssen alle Trainings-Daten verfügbar sein → es gibt kein "fertiges" Modell → ständige Anpassung anhand aller verfügbaren Daten)
ein Entfernen der Trainings-Daten bedeutet hier das Reseten des Modell's

Gemeinsam für beide Verfahren ist die anschließende Modell-Evaluierung. Dabei werden die prognostizierten Daten mit der Annotation verglichen. Die Bewertung des Modell's erfolgt dann über die Parameter Recall und Precesion.

Schema: parametrisiertes Training

Schema: Instanz-basiertes Training

2.6.x.y. No-Free-Lunch-Theorem

Modell im Sinne der künstlichen Intelligenz ist das grobe Vorgehen zur Lösung es Problems im Bereich des Machine learning's

es gibt verschiedenste Modelle, die wichtigsten Typen sind:

- Entscheidungs-Bäume
- Naive BAYES
- k-nächster Nachbar
- lineare Regression
- logistische Regression
- polynomiale Regression
- Random Forest
- Ensemble Learning
- Kernel Approximation
- Convolutional Neural Network's
- Recurrent Neural Network's
- Super Vector Machine's
- Pointer Neural Network's
- ...

Nach dem No-Free-Lunch-Theorem besagt, dass es nicht möglich ist, einen geeigneten Modell-Typ auszuwählen, ohne Annahmen über die vorliegenden Daten zu machen.

Ohne Annahmen über die vorliegenden Daten gibt es keinen Grund, von vornherein einen Modell-Typ zu bevorzugen.

Es gibt also immer eine gewisse (Mindest-)Erwartung zu den Zusammenhängen, die hinter den vorliegenden Daten stecken.

Somit ist auch kein Modell-Typ allgemeingültig und funktioniert auf allen Daten.

Kein Modell-Typ ist immer besser als ein anderer. Lediglich die Art der Daten und die wirklich dahintersteckenden Zusammenhänge bestimmen über geeignete Modell-Typen. Man muss u.a. damit leben auch mal falsche Modell-Typen auszuwählen und vielleicht nur befriedigende ergebnisse damit zu erzielen.

2.6.x.y. k-nächste Nachbar-Verfahren

Beispiel-Szenario:

Für den Wahlkampf vor einer Wahl soll prognostiziert werden, welche Partei (ROT oder BLAU) in einem Wohngebiet gewählt wird.

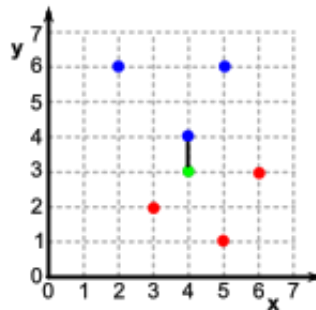
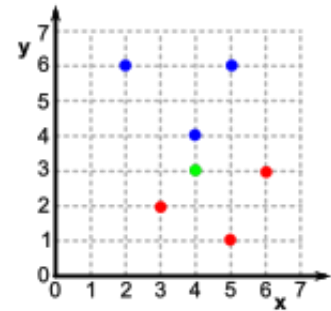
Eingabe-Daten sind Geo-Daten (Standort) mit Längen- und Breiten-Grad

Ausgabe-Daten sollen die wahrscheinlich gewählte Partei der Person am Standort sein

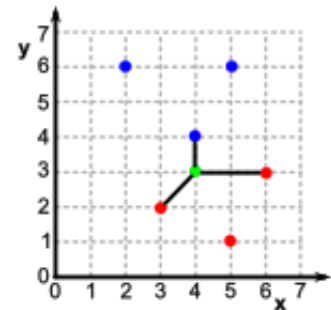
Entsprechend dem No-Free-Lunch-Theorem gehen wir von der Annahme aus, dass Personen, die dichter beieinander wohnen, auch gleiche Parteien wählen. Dies könnte z.B. durch eine reiche oder arme Wohngegend usw. begründet sein.

Bewohner	Breite x	Länge y	Partei
1	3	2	ROT
2	5	1	ROT
3	2	6	BLAU
4	6	3	ROT
5	4	4	BLAU
6	5	6	BLAU
7	4	3	?

Wahl der Anzahl der k nächsten nahbarn, die betrachtet werden sollen.



betrachtet nur der k=1
nächste Nachbar
→ **BLAU**



betrachtet die k=3
nächsten Nachbarn
→ **ROT**

Verfahren ist Instanz-basiert, da die Ziel-Daten die Referenz für neue Prognosen sind

ev. könnte man auch die Distanzen noch wichten um bei größeren k Nachbarn auch die Nähe mit zu beachten

2.6.x.y. Entscheidungs-Bäume

Entscheidungs-Bäume gehören zum unterstützten Lernen (supervised Learning). Mensch legt im Prinzip die Unterscheidungen fest, meist erfolgt das natürlich durch technische Unterstützung, nur damit lassen sich große Daten-Bestände sinnvoll analysieren. Nicht immer sind die Kriterien, die für uns Menschen so offensichtlich und entscheidend wirken, auch wirklich die gut geeigneten für Gruppierungen / Kategorisierungen der Daten.

Beispiel-Szenario:

Eine Versicherungs-Gesellschaft möchte für Neukunden eine Risiko-Bewertung durchführen. genau geht es um eine Auto-Versicherung und das Unfall-Risiko des zu bewertenden Fahrer's (Neukunde)

Als Eingabe-Daten liegen uns das Alter, die Partnerschaft, in der die Person lebt und die Kategorie des Auto's vor.

Als Ausgabe wird die Bewertung als "hoch" oder "niedrig" erwartet.

Person	Alter	Partnerschaft	Auto-Typ	Risiko
1	18	ledig	Kleinwagen	hoch
2	38	verheiratet	Familienauto	niedrig
3	50	verheiratet	Sportwagen	hoch
4	52	verheiratet	Familienauto	niedrig
5	25	ledig	Sportwagen	hoch
6	33	verheiratet	Kleinwagen	niedrig

7	47	ledig	Familienauto	?
---	----	-------	--------------	---

Bei der Analyse der Daten und des Problem's werden nun die einzelnen Merkmale bezüglich der Risiko-Kategorisierung untersucht. Praktisch stellen wir uns also die Frage: "Können wir nur anhand z.B. des Alter's auf das Risiko schließen.

Dazu kann man die Daten entsprechend des Risiko's sortieren und die Alters-Struktur untersuchen.

(Das Sortieren ist hier mehr als Veranschaulichung gemeint. In der Praxis sind es so viele Daten, dass sich so etwas nicht empfiehlt.)

Hinsichtlich des Alters gibt es keine absolute Grenze, so dass man nun den Mittelwert probieren kann.

Person	Alter	Partnerschaft	Auto-Typ	Risiko
1	18	ledig	Kleinwagen	hoch
3	50	verheiratet	Sportwagen	hoch
5	25	ledig	Sportwagen	hoch
2	38	verheiratet	Familienauto	niedrig
4	52	verheiratet	Familienauto	niedrig
6	33	verheiratet	Kleinwagen	niedrig
7	47	ledig	Familienauto	?

Für die Gruppe der Hoch-Risiko-Kunden ergibt sich ein Schnitt von 31 und für die Niedrig-Risiko-Kunden von 41 Jahren. Das kann also somit eine Orientierung sein, aber kein alleiniges Entscheidungskriterium.

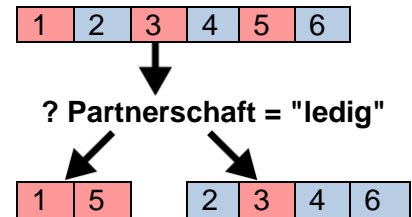
Betrachten wir als nächstes die Partnerschaft. Hier sieht es zumindestens so aus, dass ledige immer in die Hoch-Risiko-Gruppe gehören. Für die Verheirateten gilt aber nicht allgemein die Zuordnung zu einem niedrigen Risiko. Somit haben wir ein zweites Orientierungskriterium, aber auch wieder kein alleiniges. Ev. könnte man aber die Ledigen schon vorurteilen.

Beim Kriterium Auto-Typ zeichnen sich zwei klare Grüppchen ab. Die Familien-Auto-Fahrer gehören immer zu Gruppe mit dem niedrigen Risiko, die Sportwagen-Fahrer zu den Hoch-Risiko-Kunden. Bei den Kleinwagen kann keine eindeutige Entscheidung gemacht werden. Also auch das Kriterium "Auto-Typ" eignet sich nicht als alleiniges Urteil. Es gibt aber zwei Grüppchen (Familienauto- und Sportwagen-Fahrer), die eindeutig einer Bewertungs-Kategorie zugeordnet werden können.

Diese gesammelten Erkenntnisse werden nun in Entscheidungs-Bäume umgesetzt. Dazu werden die einzelnen (möglichen) Werte der Merkmale genutzt.

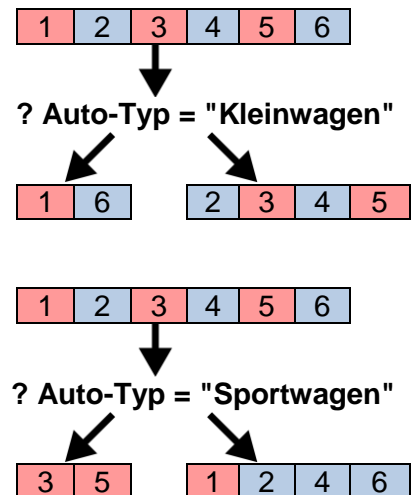
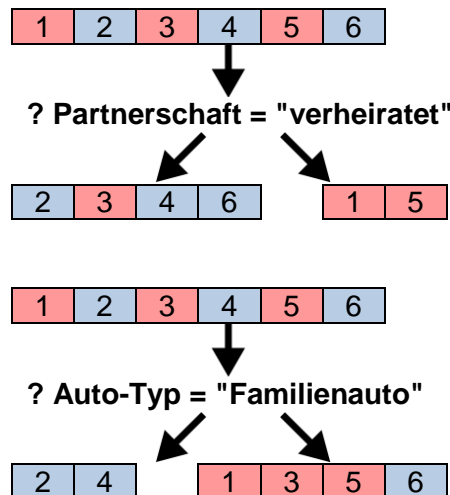
Für das Merkmal "ledig" kommen wir auf die nebenstehende Verzweigungs-Struktur. Dabei stellen wir fest, dass keine eindeutige Zuordnung vorhanden ist.

Würde man aber die Negation ("Nicht-ledig") benutzen, dann sind alle Niedrig-Risiko-Kunden schon mal in einer Gruppe. Diese müsste aber noch feiner untersucht werden.



Schrittweise gehen wir nun die möglichen Merkmale und ihre Werte durch.

Immer auf der Suche nach einem Universal-Kriterium. Leider existiert in unserem Beispiel



kein solches.

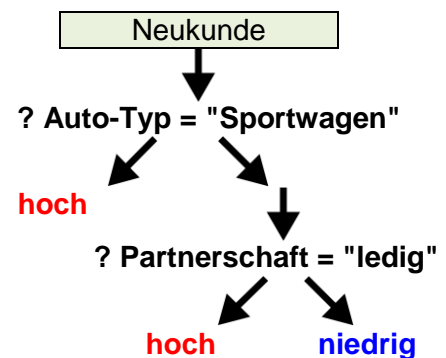
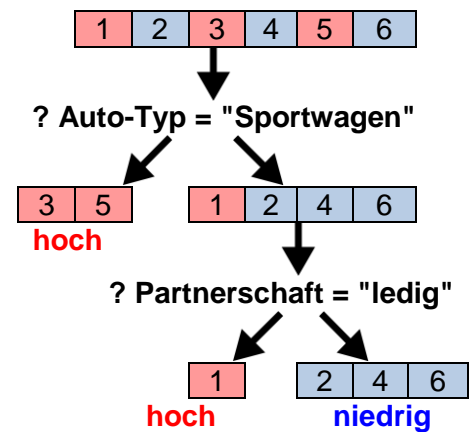
Als eine Möglichkeit bleibt uns nun die Kombination der Einzel-Entscheidungen zu einem gestaffelten Entscheidungs-System. So könnte man z.B. zuerst die Unterscheidung bezüglich der "Sportwagen"-Nutzer machen, womit wir eine große Gruppe der "Hoch-Risiko"-Kunden in eine entsprechende Gruppe klassifiziert bekommen.

Die noch heterogene Gruppe der Nicht-"Sportwagen"-Nutzer könnte dann als nächstes über das Merkmal "verheiratet" auseinander-dividiert werden.

Was dann als Struktur entsteht ist ein klassischer Entscheidungs-Baum. Ausgehend von der Wurzel (die hier oben ist) kann die bekannte Kundschaft anhand der beiden Fragen sicher in die beiden Kategorien eingeordnet werden. Nun kann auch anhand dieses Baum's eine Einordnung eines neuen Kunden erfolgen.

In unserem Beispiel würden wir Person 7 zuerst als Nicht-"Sportwagen" klassifizieren und dann in der nächsten Entscheidung als "ledig"er der "Niedrig"-Risiko-Gruppe zuordnen.

In der Praxis wollen wir, dass das Beratungs-System ev. selbst die passenden Entscheidungs-Fragen findet. Dazu werden bei einer automatischen Bewertung der Unterteilungen (auch Split's genannt), bestimmte berechenbare Parameter benutzt. Das sind zum Einen der **Gini-Koeffizient** und zum anderen die **Entropie**.



Aufgaben:

1. Überlegen Sie sich, ob es auch andere Entscheidungs-Bäume gibt! Wenn JA, dann entwickeln Sie mindestens einen! Wenn NEIN, dann begründen Sie ausführlich, warum das nicht möglich ist!
2. Erfassen Sie in Ihrem Kurs die Merkmale "Geschlecht" (männlich, weiblich, divers), "Augenfarbe" (blau, braun, grün, grau), "Freizeit-Sport" (ja/nein) und "trägt Rucksack" (ja/nein)!
 - a) Entwickeln Sie einen Entscheidungs-Baum, in dem aus Geschlecht, Augenfarbe und "trägt Rucksack" auf die Freizeit-Sport-Aktivität geschlossen werden kann!
 - b) Ordnen Sie eine neue braun-äugige, Rucksack-tragende Schülerin (kommt nächste Woche) in Ihren Kurs ein und sagen Sie voraus, ob Sie Freizeitsport betreibt!
- 3.

Schlüsselt man einen Datensatz vollständig im Entscheidungs-Baum auf, funktioniert der Baum für die Trainings-Daten zu 100%.

mit den Test-Daten wird es ev. schon erste Probleme geben

man spricht hier / dann vom Overfitting

mit neuen Daten, wird die KI aber wahrscheinlich versagen

Bewertung von Entscheidungs-Bäumen erfolgt z.B. mittels Baseline-Vergleich. Die Baseline (Vergleichslinie) trennt die jeweils größte Gruppe von den anderen ab. Die Baseline liegt also immer mindestens bei 50%.

Ein beliebtes Übungs-Beispiel / eine beliebte ML-Problemstellung ist die Analyse der Titanic-Daten.

Schon die einfache Vorhersage für eine Person (Passagier der Titanic), dass sie nicht überlebt hat würde mit 61,6% besser als zufälliges Auswählen (Wahrscheinlichkeit 50%).

Das Experimentieren mit den Merkmalen und das Bewerten verschiedener Merkmale für eine möglichst beste Entscheidung nennt man **Feature Engineering**.

Dazu werden u.a. auch Merkmale zu neuen Kenndaten kombiniert und nur diese in die Entscheidungs-Bäume eingebaut.

Aufwand für größere Verbesserungen von Vorhersage-Ergebnissen oft mit (über-)potentiellen Technik-Einsatz verbunden

z.B. bringt schon eine einfache Wetter-Vorhersage: "Das Wetter wird morgen, so wie es heute war." schon eine Güte von 80%. (Manchmal glaubt man, dass dies nicht einmal die Metrologen schaffen und die haben viele Jahre studiert.)

Vorhersagegüte (Wetter-Vorhersage)

kurzfristige Vorhersagen

Vorhersage-Güte [%]	Tag 1	Tag 2	Tag 3	Tag 4	Tag 5	Tag 6	Tag 7
unsichere Groß-Wetterlage	80 – 90	72 – 82	62 – 73	50 – 64	38 – 54	24 – 43	11 – 31
stabile Groß-Wetterlage	89 – 95	81 – 93	72 – 88	62 – 82	50 – 72	37 – 61	23 – 50

Tag 1 meint 24 Stunden weiter
für Folgetag gilt: 90% Eintreff-Wahrscheinlichkeit bei Temperatur (dwd), Windgeschwindigkeit bei 95% und Niederschlagsmenge bei 80%

mittelfristige Vorhersagen

Vorhersage-Güte [%]	Tag 8	Tag 9	Tag 10	Tag 11	Tag 12	Tag 13	Tag 14
unsichere Groß-Wetterlage	40 – 75	38 – 71	35 – 65	28 – 59	21 – 52	14 – 44	7 – 35
stabile Groß-Wetterlage	50 – 85	48 – 82	45 – 78	38 – 71	31 – 61	23 – 51	15 – 40

langfristige Vorhersagen

Vorhersage-Güte [%]	Tag 15	Tag 16	Tag 17	Tag 18	Tag 19	Tag 20	Tag 21
unsichere Groß-Wetterlage	20 – 55	15 – 50	12 – 45	10 – 39	8 – 32	6 – 24	3 – 15
stabile Groß-Wetterlage	30 – 60	28 – 55	25 – 50	20 – 44	17 – 37	12 – 29	6 – 20

Tabellen-Werte kann man als Orientierungs-Werte ansehen

Q: <https://www.wetterprognose-wettervorhersage.de/wetterkarten-unwetter-regenradar-wolkenradar/wetter-fragen/1036-prognosequete-von-wettervorhersagen.html>

heutige Wetter-Modelle für Europa berechnen mit einem Raster von 2,8 x 2,8 km in 50 Höhenstufen (dwd: COSMO-DE) bzw. 3 x 3 km bei meteoblue weltweit wird mit Raster 28 x 28 km bzw. 70 x 70 km gerechnet

das Pakete-Problem mit einem Entscheidungs-Baum lösen

Problem sowie eine Python-Version zur Lösung ist in /15/ zu finden

dabei werden die Eigenschaften Helligkeit und Größe vereinfacht auf 3 bzw. 2 Kategorien (Werte-Gruppen)

im Diagramm lässt sich diese Gruppierung als horizontale bzw. vertikale Trenn-Linien darstellen

In einer leicht geänderten Version von mir wird die Kategorisierung vom System vorgenommen. Wir verwenden Variablen, die wir später schnell verändern können. Außerdem nutzen wir für beide Eigenschaften jeweils 3 Kategorien.

durch Verschieben der Trenn-Linien kann ein optimaler Entscheidungs-Baum abgeleitet werden. Die Linien müssen so liegen, dass immer saubere Felder entstehen. Gemeint sind Felder, die nur eine Art von Daten-Punkten enthalten

Wir erweitern hier wieder den Daten-Bestand. Nur mit vielen Daten können wir unser System optimal gestalten.

Weiterhin befindet sich am Ende des Programm's auch wieder die Möglichkeit, völlig neue Pakete zu testen.

Ein automatisches Lernen kann nun dadurch realisiert werden, dass mit Hilfe des üblichen Lern-Test-Split's die Daten geteilt werden. Mit den Lern-Daten versuchen wir dann mit einer Schleife, die Kategorie-Grenzen so zu verändern, bis der Entscheidungs-Baum klare Entscheidungen bringt.

Danach testen wir mit den Test-Daten. Ist das Ergebnis zufriedenstellend, dann soll's gewesen sein. Ansonsten wird mit den Daten nochmals gelernt und getestet.

Zum Schluß verallgemeinern wir das Programm wieder und testen auch mal mit den Daten, die eine Trenn-Gerade mit negativem Anstieg hatten.

2.6.x.y. Regression

Beispiel-Szenario:

Vorhersage von Immobilien-Preisen
Ausgehend von nur einer Variablen – der Anzahl der Zimmer – soll ein Preis für eine Immobilie bestimmt werden. Als Referenz-Daten liegen uns die Verkaufs-Preise für andere Immobilien auch anderer Makler vor.

Immobilie	Anzahl Zimmer	Preis
1	2	220'000
2	4	450'000
3	3	330'000
4	2,5	460'000
5	1,5	170'000
6	4,5	700'000
7	3,5	?

Gesucht ist praktisch eine Funktion, die den Zusammenhang zwischen Anzahl der Zimmer und dem Preis möglichst gut abbildet.

Für einen ersten Versuch nehmen wir Geraden – also lineare Funktionen – an.

Das könnte z.B. ein Mittelwert über alle Verkäufe sein.

Ein zweiter Versuch erfolgt über eine steigende Gerade – die Lage der Daten-Punkte erweckt ja den Anschein eines einfachen steigenden Zusammenhang's.

Ergebnis einer linearen Regression ist eine Gerade, bei der die Quadrate der Abweichungen möglichst gering sind.

Quadrierung wird benutzt, um zum Einen negative Abweichungen Werte-mäßig genauso wie positive zu behandeln und zum Anderen größere Abweichungen auch eine deutlich größere Bedeutung (für das Verfahren) beizumessen.

polynomielle Approximation

mit Polynomen n-ten Grades kann man eine Funktion durch alle Daten-Punkte legen

es besteht aber nun die Gefahr, dass unser Modell (Polynom) overfittet, d.h. es beschreibt nur die Trainings-Daten bei Real- oder Test tauchen auf einmal größere Abweichungen auf das Modell hat nur mangelhaft generalisiert.

2.6.x.y. Künstliche Neuronale Netze

Beispiel-Szenario:

Bild- bzw. Objekt-Erkennung von Hunde und Katzen anhand von biometrischen Daten, wie Höhe, Gewicht, Nasenlänge und das Vorhandensein von spitzen Ohren (ja/nein).

Objekt	Höhe [cm]	Gewicht [kg]	Nasen-Länge [mm]	Ohren-Länge [cm]	spitze Ohren	Klasse
1	23	3	13	3	ja	Katze
2	50	10	24	5	nein	Hund
3	18	4	30	2	ja	Hund
4	22	5	12	3	ja	Katze
5	38	7	22	4	nein	?

Im Rahmen der Daten-Vorbereitung (→ Daten-Transformation) wird die Eigenschaft der spitzen Ohren in Binär-Daten übersetzt.

Künstliche Neuronale Netze (KNN) können nur numerische Werte (als Eingabe) bearbeiten.

Sind spitze Ohren vorhanden, dann wird eine 1 eingesetzt, ansonsten eine 0.

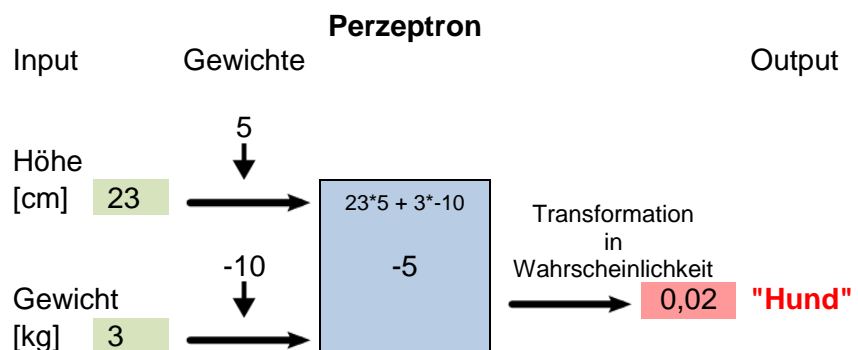
Objekt	Höhe [cm]	Gewicht [kg]	Nasen-Länge [mm]	Ohren-Länge [cm]	spitze Ohren	Klasse
1	23	3	13	3	1	0
2	50	10	24	5	0	1
3	18	4	30	2	1	1
4	22	5	12	3	1	0
5	38	7	22	4	0	?

Genauso verfahren wir auch mit den Ziel-Daten. Die Werte "Hund" und "Katze" werden durch eine 1 bzw. 0 ersetzt.

Verfahren der gewichteten Summen

Ein gedachtes Perzeptron soll eine "Katze erkennen. Wir beschränken uns nur auf die Attribute "Höhe" und "Gewicht", damit die Berechnungen übersichtlich bleiben.

Alle Eingangs-Werte (Input) werden zu Anfang mit willkürlichen Gewichten multipliziert und dann addiert. Im nebenstehenden Beispiel ergibt dies -5. Als nächstes erfolgt über eine spezielle Funktion (z.B. Sigmoid-



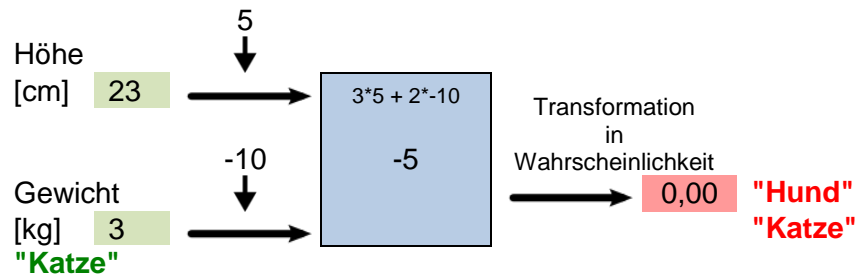
Funktion) eine Transformation in eine Wahrscheinlichkeit.

"Katze"

Diese Wahrscheinlichkeit stellt das aktuelle Ergebnis unseres Perzeptron's dar. Mit 0,02 ist das sehr enttäuschend, da es nur zu 2 % sagt, es wäre eine Katze (- also zu 98 % ist es angeblich ein "Hund"). Da die richtige Antwort "Katze" bekannt ist, können wir nun die Gewichte anpassen. Praktisch leiten wir die Veränderungen rückwärts durch das Perzeptron ab (Backpropagation, genauer s.a. →). Als Fehler-Wert können wir das Abweich-Quadrat nehmen und den aktuellen Fehler berechnen: $(1 - 0,02)^2 = 0,9604$. Diesen benutzen wir nun zur Modell-Anpassung.

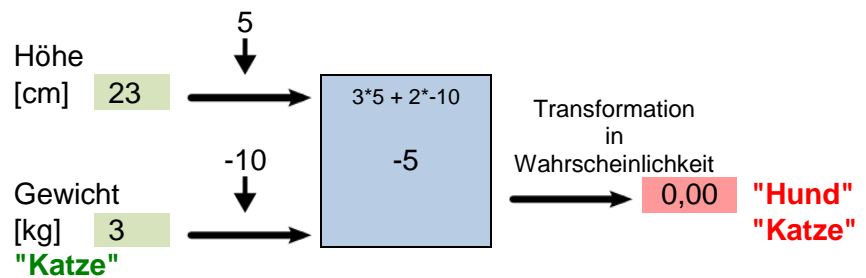
Damit die Transformations-Funktion einen Wert größer 0,5 ausgibt, muss die Gewichte Summe größer werden. Dafür aber müssen wiederum die Gewichte angepasst werden. Unter Anwendung des Fehler-Wertes ermitteln wir neue Gewichte.

Mit den neue Gewichten wird nun ein erneuter Versuch gestartet. Wieder wird eine gewichtete Summe berechnet, die Transformations-Funktion ermittelt und die Ausgabe geprüft.



Nach sehr vielen Versuchen bekommen wir dann eine optimale Konfiguration der Gewichte. Würden wir nur mit den angegebenen Katzen-Daten trainieren, würde unser Perzeptron irgendwann mal zu 100 % eine "Katze" vorhersagen. In der Praxis gehen bei einem Daten-Durchlauf allerdings nicht nur die Daten der einen Katze ein, sondern alle Tier-Daten.

Bei Hunde-Daten muss unser Modell u.U. anders reagieren und die Gewichte anders anpassen. Nach x Durchläufen (Epochen) aller Daten hat unser Modell dann vielleicht die optimalen Gewichte gefunden.



Im "echten" KNN für unser Problem (Katze-Hund-Unterscheidung) müssen natürlich alle Attribute eingehen. Das bedeutet mehr Perzeptronen. Wahrscheinlich wird man auch zwei unabhängige Ausgaben haben wollen, um so auch unabhängige Wahrscheinlichkeits-Aussagen zur Tier-Art zu erhalten.

(Entscheidungs-)Funktionen (Transformation in Wahrscheinlichkeit)
z.B. sigmoide Funktion

Wahrscheinlichkeits-Aussagen
als Ergebnis

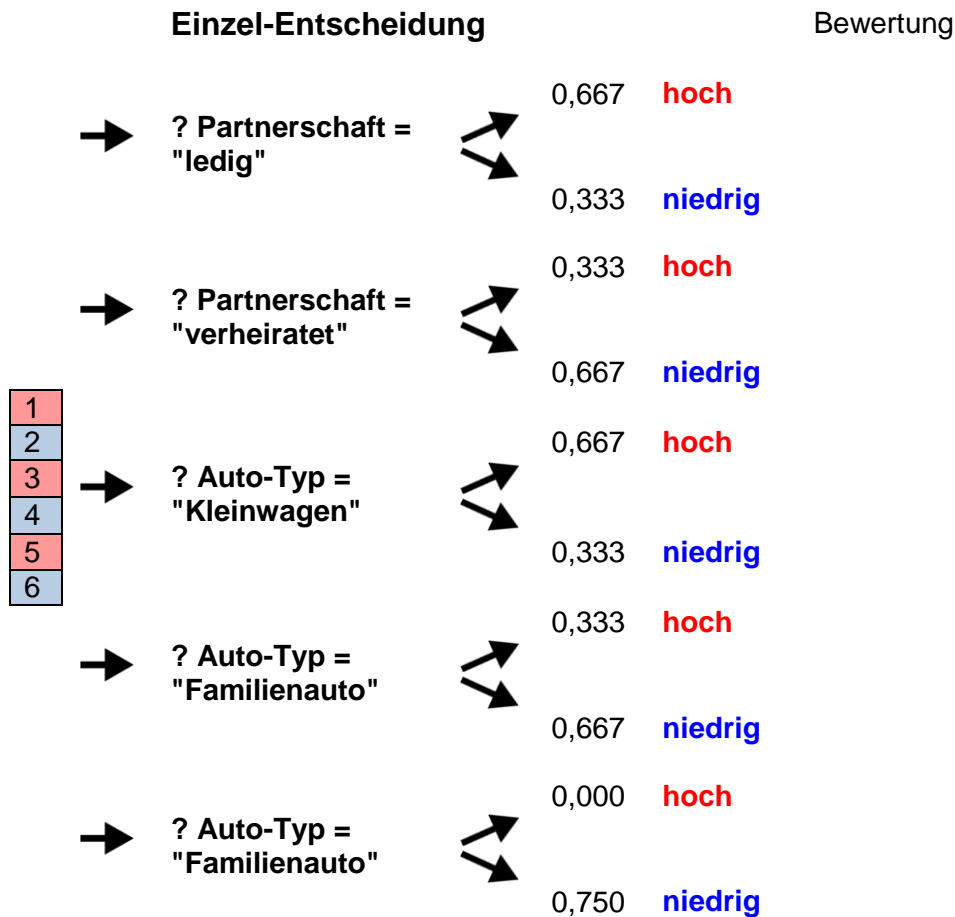
mit welcher Wahrscheinlichkeit ist das Ergebnis in der einen oder anderen Ziel-Klasse.

Inspiration für KNN kommt aus natürlichen Netzen (Nervenzellen in einem Gehirn)

2.6.x. Ensemble Learning

Ausnutzung verschiedener Modelle des gleichen Typ's oder unterschiedlichen Typen
Ziel ist Verbesserung des Gesamt-Modell und dessen Ergebnis / Leistung in der Anwendungs-Praxis

Kombination mehrerer (einfacher) Entscheidungs-Bäume wird Random Forrest genannt
Jeder Entscheidungs-Baum nutzt nur einen Teil (nur bestimmte Attribute) der Daten
Für diese wird eine Entscheidung hinsichtlich der beabsichtigten Klassifizierung erstellt. Diese sind i.A. eher weniger Aussage-kräftig.



Nimmt man aber alle Teil-Ergebnisse und kombiniert deren Bewertung, dann erhält man ein deutlich besseres Gesamt-Ergebnis, das sich an komplexen Entscheidungs-Bäumen oder anderen Modellen messen kann.

Die kleinen, einzelnen – oft schwachen – Modelle lassen sich so zu einem sehr effektiven Entscheidungs-System / Modell kombinieren (zusammenstellen → ensembling).

I.A. ist in diesen zusammengesetzten Modellen ein overfitting schwieriger, da die Einzel-Modelle nur grobe Zusammenhänge – basierend auf wenigen Daten (Attributen) - repräsentieren.

Die Kombination von verschiedenen Modell-Typen auf einem Daten-Bestand verhindert ebenfalls overfitting, da die Modelle die Daten aus verschiedenen Perspektiven betrachten und ihre Entscheidungen auf sehr unterschiedliche Art- und Weise fällen können. In welchem Anteil dann die Einzel-Modelle in das Gesamt-Ergebnis einfließen, ist ein eigenständiges Modell und muss ev. trainiert oder experimentell ermittelt werden.

2.6.x. Optimierung eines KNN

verantwortlich für die Optimierung eines KNN ist der Optimizer
er soll unter Ausnutzung der Fehler-Funktion zuerst ein lokales Minimum, später aber
auch das globale Minimum der Fehler-Funktion finden

Methode z.B. ist das Gradienten-Abstiegs-Verfahren (Gradient Descent)

$$w' = w - \eta \cdot \nabla C$$

w ... Gewicht
w' ... neues Gewicht
 η ... Lern-Rate (Learning Rate)
C ... Gradient der Kosten-Funktion

Problem mit dem Gradient Descent
praktisch muss man den Mittelwert aus den Gradienten aller Eingaben berechnen, um dann
ein neues Gewicht zu berechnen
bedeutet sehr hoher Speicher-Bedarf

Stochastic Gradient Descent

besser geeignet ist der Stochastic Gradient Descent (SGD; Zufälliger Gradienten-Abstieg)
statt alle Daten auszuwerten, wird nur eine Teilmenge zur Berechnung des aktuellen Gradienten
benutzt, dadurch wird die Datenmenge für die Berechnung deutlich eingeschränkt
für die Berechnung eines Gradienten erfolgt nun aus einer Teilmenge der Beispiel-Daten
(Batch'es genannt)
der berechnete Mittelwert wird dann zur Anpassung der Gewichte benutzt und nach dem
nächsten Batch aktualisiert
insgesamt dadurch deutlich schneller und Speicherplatz-sparender

in der Praxis hat sich diese vereinfachte Methode als ebenso tauglich erwiesen
ein Nachweis dass wirklich das globale Minimum gefunden wird / wurde, gibt es nicht
weiteres Problem ist die gleich-artige Beeinflussung aller Gewichte

mit Hilfe des sogenannten Momentum's versucht man die Fixierung auf ein lokales Minimum
zu umgehen
dieser dient der zusätzlichen - individuellen - Korrektur der Gewichte
??? Dämpfung der Anpassung an einem lokalen Minimum???

ein weiterer Faktor ist decay ()
dieser Faktor dient der Anpassung (stetige Verringerung) der Lern-Rate während der Epochen
man geht hierbei davon aus, dass man sich im Verlauf des Training's immer mehr dem Minimum
nähert und somit nicht mehr große Änderungen vornehmen sollte / muss
ansonsten könnte es zum Hin- und Her-Springen um ein Minimum kommen und im schlimmsten Fall
sogar sich immer weiter vom Minimum entfernt, weil die großen Veränderungen des Gradienten
als Verschlechterung des Ergebnisses interpretiert werden und durch große Korrekturen behoben
werden sollen, diese bewirkt dann wieder eine übermäßige Korrektur mit einem Sprung auf die
andere Seite (bezüglich des Minimum's) und das verstärkt den Effekt weiter

Lern-Rate und Decay müssen aufeinander abgestimmt werden

weitere Optimierungs-Verfahren

RMSProp

()

in diesem Verfahren werden die Gewichte der Neuronen ganz individuell angepasst betrachtet wird die Entwicklung der einzelnen Gewichte

Adam

kombiniert RMSProp mit Momentum

individuelle Anpassung der Gewichte der einzelnen Neuronen und zusätzlich durch Momentum eine verstärkte Entwicklung in die initial gewählte Optimierungs-Richtung ("Schwung in die richtige Richtung")

??? auch individuelle Anpassung der Lern-Rate an jedem Neuron

sehr schnelles und effektives Verfahren

bringt in der Praxis i.A. die besten Ergebnisse

Empfehlungen / Orientierungen für die Wahl eines Optimizer's

- Adam hat sich in der Praxis bestens bewährt
- Adam und RMSProp erreichen das Optimum schneller
 - es erfolgt eine schnelle Optimierung in die richtige Richtung
- in der Wissenschaft wird Stochastic Gradient Descent mit Momentum benutzt
 - es wird langsamer aber sicherer auf den optimalen Wert hingearbeitet
-

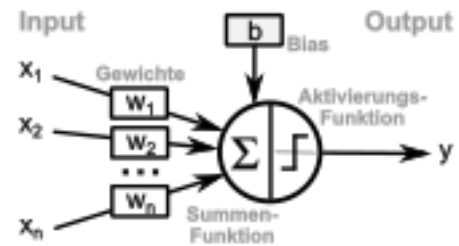
2.6.x. Hyper-Parameter

die Aktivierungs-Funktion a muss vor dem Training des KNN festgelegt werden

weiterhin gehören zu solchen vorbestimmten Parametern auch die Lern-Rate (Learning Rate), die Batch-Größe (Größe von Teil-Datensätzen für die Optimierung des Trainings-Prozesses), die Anzahl der Epochen, die Initial-Werte der Gewichte und des Bias.

solche Parameter bezeichnen wir als Hyper-Parameter

andere Werte, wie die Gewichte und der Bias können und müssen in der Trainings-Phase angepasst werden, ansonsten wäre kein Lernen im Netz möglich
Gewichte und Bias werden also im Training gelernt



streng genommen zählt auch der Aufbau des Netzes zu den Hyper-Parametern. Dazu gehören z.B.:

- Anzahl der Schichten (Tiefe des netzes)
- Art der Schichten
- Anzahl der Neuronen (Breite der Schichten / des Netzes)
- Art der Neuronen
- Art des Optimierer's
- Art der Fehler-Funktion
- Anteil der Elemente, die für Drop-out's benutzt werden
- Anzahl der Trainings-Durchläufe / Lern-Zyklen
- Lern-Rate (pro Trainings-Runde)

Definition(en): Hyper-Parameter

Hyper-Parameter sind (Einfluß-)Faktoren eines KNN, die vor dem Training festgelegt werden, das ML-Modell selbst nicht verändern, aber die Lern-Prozess (global) beeinflussen.

Hyper-Parameter sind alle die Parameter / Faktoren / Einstell-Möglichkeiten, die während der maschinellen Lernen's nicht erlernt werden.

Wie schnell soll gelernt werden?

wichtige Hyper-Parameter in den verschiedenen Modell-Typen

• Entscheidungs-Bäume	maximale Baum-Tiefe Informations-Kriterium minimale Anzahl Daten-Punkte je Split ...
• Random Forrest	Anzahl an Entscheidungs-Bäumen ...
• Logistische Regression	Lösungs-Algorithmus minimale Anzahl Iterationen ...
• Support Vector Machines	Kernel-Art (linear, polynomial, expotentiell) Regularisierungs-Parameter Kernel-Koeffizienz ...
• Neuronale Netze	Lern-Rate Anzahl Epochen Optimierungs-Algorithmus ...
•	
• ...	

Grundsätze zu Hyperparametern

mehrlagige Perzeptonen (Perzeptren) (Abk.: MLP) / künstliche neuronale Netze (Abk.: KNN) mit einer Hidden-Schicht sind universelle Approximatoren
sie können sich (in gewissen Grenzen) allen Funktionen annähern

tiefe Modelle / Netzen / Perzeptonen benötigen eher weniger Parameter für einen erfolgreichen Lern-Vorgang (im Vergleich zu flachen Modellen / ...)

tiefe Modelle / Netze / Perzeptonen sind schwerer zu trainieren als flache

früher waren die Modelle eher flach, weil die gebrauchte Rechen-Leistung bei tiefen und breiten Modellen extrem schnell zunimmt

heutige Rechner schaffen auch tiefe / breite Modelle

dazu werden oft spezielle Zusatz-Prozessoren oder auch gerne Graphik-Prozessoren benutzt

diese sind stark parallelisiert und auf wenige Rechen-Operationen hin optimiert

Hyperparameter-Optimierung

Ermitteln der besten Kombination der Hyperparameter für ein maximales / optimales Ergebnis

bei der Vielzahl von Parametern kann kaum jede Kombination ausprobiert werden

Nutzung von zufallsgesteuerten Verfahren, um zu mindestens den Ziel-Bereich der Parameter einzugrenzen
in dem Ziel-Bereich kann dann feiner gesucht werden
zum Bewerten der Modelle und der Hyperparameter-Kombination werden die früher abgetrennten Validierungs-Datensätze benutzt

das Modell, bei dem der kleinste Fehler auftritt ist dann das finale System / Modell
am finalen Modell werden dann die früher separierten Test-Datensätze angewendet

Lern-Rate (Learning Rate)

wird als wichtigster Hyper-Parameter betrachtet
allgemein wird empfohlen, gerade mit diesem Parameter zu experimentieren, da hier das größte Potential für Effekte liegt

zu kleine Lern-Raten bedeuten, dass das Netz sehr lange braucht, um das Minimum der Kosten-Funktion zu finden

eine zu große Lern-Rate kann dazu führen, dass das Minimum quasi übersprungen wird
da man im ungünstigsten Fall auch einer Position ankommt, die schlechter, als die vorherige Position ist, wird das Netz überreagieren und dadurch das Problem weiter verstärken
u.U. wird dabei nie das Minimum gefunden

Anzahl der Epochen

Eine Epoche ist ein kompletter Durchlauf der Trainings-Daten durch das Netz

beim Experimentieren mit der Anzahl der Epochen sollte man sich am Erreichen der (erwarteten) Akkuratheit bzw. einer nur noch geringfügigen Änderung der Akkuratheit orientieren.
Gibt es eine Vergrößerung des Abstandes der Akkuratheit von Training und Test, dann ist die Anzahl i.A- zu groß und das Netz zeigt sogenanntes Overfitting (Netz lernt die Trainings-Daten "auswendig").

Definition(en): Overfitting

Unter Overfitting (Überanpassung) versteht man den Effekt, dass ein Lern-System die Eingabe-Daten auswendig lernt, statt sie zu generalisieren.
Ein Overfitting wird dadurch sichtbar, dass die Trainings-Daten quasi perfekt, vergleichbare Arbeits-Daten aber deutlich schlechter bewertet / klassifiziert werden.

Initial-Werte der Gewichte und des Bias

Die Gewichte haben einen großen Effekt auf den Lern-Erfolg und die Lern-Geschwindigkeit. Der Bias wird häufig zu Beginn mit 0 für alle Neuronen gesetzt. Die Gewichte sollten zufällig verteilt gewählt werden. Dadurch können auch verschiedene Lern-Runden unterschiedliche Ergebnisse bringen (z.B. unterschiedliche Minima treffen).

Im Keras-System lassen sich die Gewicht's- und Bias-Initialwerte z.B. so setzen.

```
modell.add(Dense(64, kernel_initializer='random_uniform',  
                bias_initializer='zeros'))
```

Die Gewichte werden durch zufällige, uniforme Standard-Belegung mit Werten zwischen -0,05 und +0,05 belegt. Dazu dann später (→) mehr.

Verhindern muss man unter allen Umständen, dass alle Gewichte zu Anfang mit Null belegt werden bzw. sehr dicht an Null liegen. Dadurch kann es nämlich passieren, dass die Summe an den Neuronen zu klein wird, die Aktivierungs-Funktion ebenfalls Null liefert und die nächsten Schichten - bis hin zur Ausgabe-Schicht - kein Ergebnis bringen. Auch die Null-Initialisierung einzelner Neuronen kann zum "Ausfall" eines Bereiches in der Schicht führen - die Neuronen sind tot - besser gesagt bringen keine Leistung mehr. Kleine Gewichte bewirken zudem meist nur sehr kleine Gradienten, die dann im Backpropagation-Verfahren keinen Effekt bringen.

Batch-Größe (Batch Size)

Ein Batch ist eine definierte Untergruppe der Trainings-Daten. Die Unterteilung in mehrere Batch'es wird z.B. genutzt, um für die Fehler-Funktion zu Anfang nicht alle Datensätze miteinzubeziehen, was ja einen enormen Rechen-aufwand bedeuten würde. Aus der Erfahrung weiss man, dass auch kleinere Gruppen von Trainings-Daten schon genügend Information für das Berechnen der Fehler-Funktion enthält. Dadurch hält man den anfänglichen Rechen-Aufwand relativ klein.

Im Keras-System lässt sich die Batch-Größe z.B. so setzen.

```
modell.fit(batch_size=32, ...)
```

Dazu dann später (→) mehr. Die Größe 32 ist der Standard-Wert in Keras. Empfehlenswert ist es mit Batch'es von 16 bis 128 Datensätzen zu arbeiten.

Zu kleine Batch'es enthalten sehr wahrscheinlich nicht alle Ergebnis-Klassen gleichverteilt, so dass das Ergebnis der Fehler-Funktion zu stark verfälscht wird. Zu große Batch'es bedeuten einen zu großen rechen-Aufwand, was das Training verlangsamt.

Anzahl der (Hidden-) Layer (Schichten)

Was ist besser geeignet?

Viele Schichten / ein tiefes Netzwerk? mit ev: Wenig Neuronen pro Schicht?

oder

Wenige Schichten / ein flaches Netzwerk? mit ev.: Vielen Neuronen pro Schicht?

Beide Konstellationen sind zum Lernen von Problemen geeignet (→ s.a. Universalitäts-Theorem). Tiefere Netzwerke haben praktisch gesehen weniger Parameter und sind dadurch schneller zu berechnen.

Unsere Realität besteht aus Objekten, die intern in hierarchisch strukturierte Unter-Objekte zerfallen. Ein Wald besteht aus Bäumen usw. Die Bäume bestehen aus Stämmen und Zweigen sowie Blättern. Die Blätter sind verschieden geformt. Genau diese unterschiedlichen Strukturen ermöglichen es uns die Objekte zu klassifizieren und zu definieren.

Bilder enthalten genau diese hierarchische Struktur, natürlich auf 2D heruntergebrochen. Es macht also auch Sinn die Erkennung der Strukturen zu favorisieren. Das entspricht der Tiefe des netzes - eben durch bestimmte Schichten, die Strukturen "erkennen".

Es gibt auch Probleme mit der Tiefe:

Da ein Netz mit wenigen Neuronen pro Schicht zu wenig "interne" Strukturen erkennen kann, kann es zum **Underfitting** kommen. Das Netz erreicht nicht die optimale Akkuratheit. Das Trainings-Ergebnis und auch das Test-Ergebnis bleibt unter den Möglichkeiten zurück.

Eine sehr tiefe und breite Struktur unseres Netzes ermöglicht eine sehr feine Erkennung von Strukturen in den Daten. Jetzt kann es zum **Overfitting** kommen. Das kann man an einem hinter dem Trainings-Ergebnis zurückbleibenden Test-Ergebnis erkennen. Das Netz hat die Trainings-Daten quasi auswendig gelernt. Auch ein Trainings-Ergebnis mit einer Akkuratheit von 100 % kann ein Zeichen für Overfitting sein.

Oft sollten Kompromisse getestet werden. Optimalerweise sollten die Akkuratheiten für die Trainings- und Test-Daten gleich groß und nah bei 100 % sein (guter Bereich (90 -) 95 - 98 %).

weitere ... (Woche 4)

allgemeine Regeln/ Tip's für die Erstellung und Modellierung von KNN's

- Netz länger trainieren (damit das Netz Zeit bekommt, komplexere Features zu lernen)
- Anzahl der Neuronen pro Layer erhöhen / größer dimensionieren (um Underfitting zu vermeiden)
- Anzahl der Layer erhöhen / hoch auswählen (damit komplexere Features erkannt werden können)

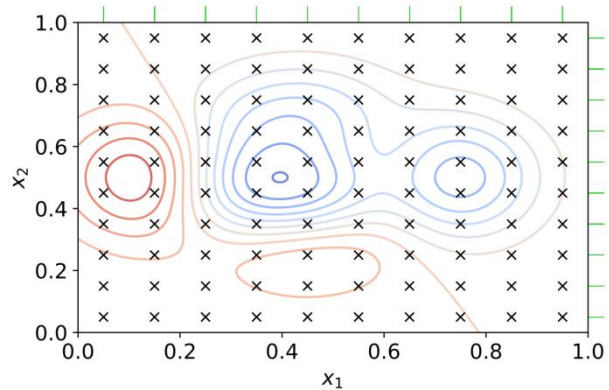
Wie findet man nun die optimalen Werte für die Hyper-Parameter?

eine Möglichkeit ist natürlich die manuelle Wahl. Das setzt aber große Erfahrung voraus.

Eine zweite Möglichkeit ist die Nutzung der Standard-Werte. Hierbei macht man i.A. nichts falsch. Das Netz lernt wahrscheinlich nicht superschnell, aber es wird fast immer ein sehr gutes Ergebnis erreicht.

Die dritte Möglichkeit ist eine automatisierte Anpassung der Hyper-Parameter in Simulations-System, wie Keras.

übliche Methode ist die Gitter-Suche (Grid Search, auch: Raster-Suche) für z.B. zwei Hyper-Parameter werden Listen mit zu testenden Werten (meist lineare oder logarithmische Reihen) jeweils als Werte-Paare systematisch durchprobiert wenn jeder Wert der einen Liste mit jedem Wert der anderen durchgetestet wurde, dann lassen sich z.B. Akkuratheit und Fehler als dreidimensionales System auswerten. Im Allgemeinen ergeben sich klare Optima oder Tendenzen, die dann weiter verfolgt werden können oder die Wertebereiche neu aufgesplittet werden.

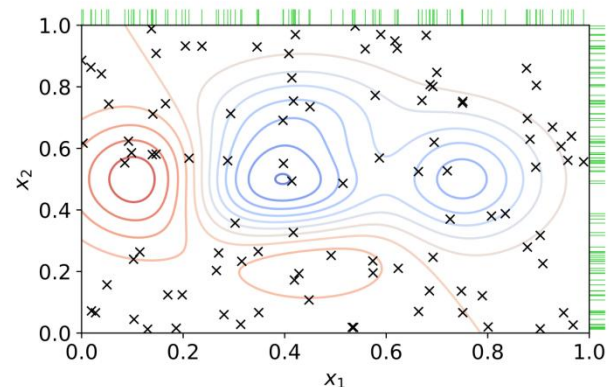


Beispiel für eine Gitter-Suche mit zwei Hyper-Parametern
Q: de.wikipedia.org (Alexander Elvers)

Test-Aufwand schon recht beachtlich: bei angenommen 6 Werten für jeden Hyper-Parameter ergeben sich schon 36 zu simulierende KNN's da eine statistische Absicherung sinnvoll ist, oft schwanken die Ergebnisse relativ stark in Abhängigkeit von den zufällig gewählten Gewichten und Biases, steigt die Test-Anzahl weiter bei nur 5 gleich-artigen Test's pro Gitter-Punkt ergeben sich schon 180 Test's

bei mehreren zu testenden Hyper-Parametern stehen wir aber schnell vor dem Problem der extrem großen Anzahl zu simulierenden KNN's bei 6 Hyper-Parametern mit jeweils 6 Werten und 5 gleich-artigen Durchläufen (zur statistischen Sicherung) sind das $6^6 * 5 = 233'280$ Simulationen (→ "Fluch der Dimensionalität")

hier verfolgt man nun einen weiteren Test-Ansatz statt z.B. systematisch durchzutesten geht man stochastisch vor es werden einfach zufällig einzelne - zu prüfende - Hyper-Parameter-Werte ausgewählt und damit ein Test (natürlich statistisch gesichert) durchgeführt mit einer relativ geringen Menge an Simulationen kann man dann zumindestens Trend's oder Hot-spot's erkennen, wo sich eine weitere Detail-Suche dann vielleicht lohnen würde.



Beispiel für eine Zufalls-Suche mit zwei Hyper-Parametern
Q: de.wikipedia.org (Alexander Elvers)

ein weiterer Ansatz ist der Versuch die Hyper-Parameter - zumindest einzelne - in ihrer Wirkung auf das Modell als Funktion zu verstehen / erfassen Hat man eine solche Funktion - oder Abschnitt aus dieser, dann kann ev. über eine Rück-Ableitung der Funktion erfolgen.

Problem ist hier ev. die fehlende Ableitung (weil die Grund-Funktion nicht beschrieben ist). Weiterhin hat man es oft mit hochdimensionalen Funktionen zu tun, die i.A. auch nicht linear sind. Die Komplexität der Gesamt-Funktion (Abhängigkeit von allen / vielen) Hyper-Parametern) ist sehr hoch.

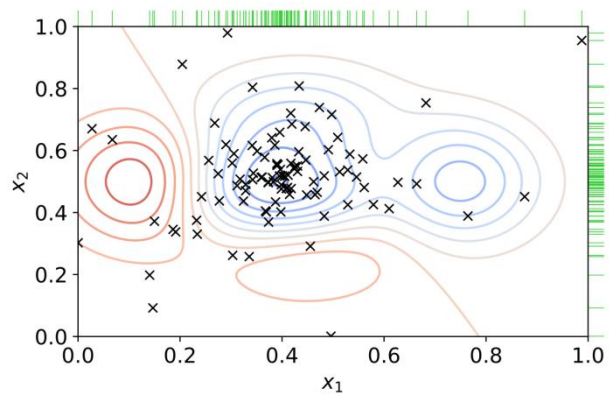
Die Berechnung von Funktions-Werten (für die Gesamt-Funktion) istz sehr aufwendig und muss immer wieder mit der Praxis (durch Test's) abgeglichen werden. Das bdeutet weitere Trainings-Runden mit entsprechend angepassten Modellen / KNN's.

Eine Lösung ist die BAYESische Optimierung. Dabei handelt es sich um eine Verfahren Extrema ohne die Kenntnis von Ableitungen zu finden. Es werden Wahrscheinlichkeiten über einzelne Parameter ausgewertet und in Beziehung zu den den Wahrscheinlichkeiten anderer Parameter gesetzt, um so Wahrscheinlichkeits-Voraussagen zu den Parameter-Kombinationen zu bekommen.

Moderne Verfahren setzen auf biologische Vorbilder. So werden genetische und / oder evolutionäre Verfahren / Algorithmen auf das Problem angewendet.

Hier ist und beliebt das Problem, dass solche Verfahren nur sicher beim Finden von lokalen Extrema sind. Globale Extrema können schnell "übersehen" werden, weil es keine Ausstiegs-Möglichkeit aus der lokalen Extremwert-Suche gibt.

Durch die Wahl unterschiedlicher - zufällig gewählter - Start-Punkte kann man aber eine recht hohe Erfolgs-Quote realisieren.



Beispiel für eine "biologisch" orientierte Suche mit zwei Hyper-Parametern
Q: de.wikipedia.org (Alexander Elvers)

Die Neural Architecture Search (NAS) sucht automatisiert nach passenden / angepassten Künstlichen Neuronalen Netzwerken.

Q: de.wikipedia.org()

Dabei kommen z.T. auch wieder Künstliche Neuronale Netze zur Anwendung. Man spricht dann von Super-Netzen.

Beispiel Hyper-Parameter "" in einem Supervised Learning-Modell:

```
model = Sequential(name="CNN")
    modell.add(Conv1D({{choice([8, 16, 32, 64, 128, 256])}},
        kernel_size={{choice([2, 3, 4, 5, 6])}},
        padding="same",
        input_shape=[X_train.shape[1], X_train.shape[2],
        activation="relu"))
    model.add(Dropout({{uniform(0, 0.5)}}))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense({{choice([32, 64, 128, 256, 512, 1024])}}))
    model.add(Activation({{choice(["relu", "tanh"])}}))
    model.add(Dense({{choice([32, 64, 128, 256, 512])}}))
    model.add(Activation({{choice(["relu", "tanh"])}}))
    model.add(Dense({{choice([32, 64, 128, 256])}}))
    model.add(Activation({{choice(["relu", "tanh"])}}))
```

```

model.add(Dense({{choise([32, 64, 128])}}))
model.add(Activation({{choise(["relu", "tanh"])}}))
model.add(Dense(1, activation="sigmoid"))

model.compile(loss="binary_crossentropy", metrics=[f1],
              optimizer={choise(["rmsprop", "adam", "sgd"])})

result = model.fit(X_train, np.array(y_train),
                  batch_size={choise([32, 64, 128])},
                  epochs={choise([10, 20, 40, 60])})

```

schon hier ergeben sich 30'000'000 verschiedene Kombinations-Möglichkeiten

typisches Daten-Splitting in Trainings- und Test-Daten (meist 80 + 20%)



für Optimierung wird von den Trainings-Daten nochmals ein Teil abgespalten (wieder rund 20% von der Original-Größe)
diese werden in der Optimierung – also bei der Suche der optimalen Hyper-Parameter – als Test-Daten für die Validierung jedes Modell verwendet
hat man das optimale Modell gefunden, dann wird dieses mit den bisher ungenutzten test-Daten geprüft



dabei soll overfitting verhindert werden



2.6.x. Probleme in KNN's

optimale / beste Hyper-Parameter lassen sich nicht mathematisch herleiten
derzeit viel intuitive (manuelle) Festlegung oder systematisches (ev. automatisiertes) Austesten
da Netze auch für die unterschiedlichsten Anwendungs-Fälle genutzt werden, fehlt auch ein allgemeines Konzept / Gesetze / ...

statistischer Bias sind alle relevanten Einfluß-Faktoren / Parameter, die das Ergebnis beeinflussen können

Klassen müssen gut zu unterscheiden sein

die Beispiel- / Trainings-Daten müssen alle Klassen gleichmäßig repräsentieren

Trainings-Daten sollen die gesamte Ober-Klasse abdecken (mit allen Spezialitäten, Situationen, ...)

Overfitting

zu erwartende Stördaten (Noise, ...) in den späteren Arbeits-Daten müssen auch durch die Trainings-Daten abgebildet werden

kann man z.T. künstlich zu den Trainings-Daten hinzufügen und so Pseudo-Trainings-Daten erzeugen, die zur weiteren Verbesserung des KNN benutzt werden können

Definition(en): Bias

Bias ist der Fehler, der beim maschinellen Lernen durch falsche Annahmen über die Trainings-Daten gemacht werden.

Problem: Daten-Menge

schon einfache Bilder haben recht große Pixel-Ausdehnungen

Beispiel 200x200 Pixel bedeutet 40'000 Pixel

bei den klassischen 3 Farben (RGB-System) sind dann 120'000 Eingabe-Neuronen notwendig

hat die erste versteckte Schicht (Hidden-Layer) nur ein Drittel so viele Neuronen, dann sind das 40'000 Stück

für jedes Neuron müssen bei einer dichten Schicht 120'000 Gewichte (+ 1 Bias) gespeichert werden

das sind dann 4'800'040'000 Werte

die dazu benutzten Float32-Werte brauchen 19'200'160'000 Byte → 17,88 GB

Ziel: Reduktion der Daten-Menge (Speicher-Bedarf)

Problem: Rechen-Operationen

bei der Annahme der obigen Parameter für die versteckte Schicht kommen wir auch auf $120'000 \times 40'000 = 4'800'000'000$ Multiplikationen nur für die Berechnung aller Gewichte dieser Schicht

dazu kommen Summen-Bildung, die Berechnung der Aktivierungs-Funktion für jedes Neuron (also jeweils 40'000x)

geht man einer Anpassung mittels Backpropagation aus, dann ergeben sich auch hierfür wieder 40'000 Operations-Sequenzen

Ziel: Reduktion der Rechen-Operationen

Problem: Veränderung der Daten-Struktur

z.B. werden 2D-Daten (Bilder) in 1D-Daten (Vektoren, Sequenzen) umstrukturiert
dadurch gehen ev. Nachbarschafts-Beziehungen verloren
Muster als solche werden ev. nicht erkannt
Veränderung von Lage / Position eines Objektes beeinflusst u.U. das Ergebnis
Vermeiden von Overfitting (zu starke Fokussierung auf die Trainings-Daten, bei Real-Daten versagt das Netz dann u.U.)

Problem: wenige Trainings-Daten

die Beschaffung von gelabelten Trainings-Daten sehr aufwendig
deshalb versucht man auch mit wenigen Trainings-Daten auszukommen
daraus folgen die folgenden Probleme innerhalb des Modell's:

- schlechte Generalisierung, Overfitting
- schlechte Genauigkeit / Akkuratheit
- unterschiedlich viele Bilder pro Klasse →

um trotzdem zu guten Ergebnissen zu kommen, kann man:

- Bilder / Daten künstlich generieren (Data Generation)
- Bilder / Daten leicht verändern (randomisieren, verdrehen, umpositionieren, spiegeln, ...)
- Regularisierung (Anpassung der gelernten Funktion)
- Daten-Klassen ausbalancieren
- Kosten-Funktion gewichten (Fehler auf kleine / unterrepräsentierte Klassen stärker bewerten)
- Transfer Learning (zusätzliches Nutzen und Einbeziehen von starken Netzen, die all-gemeinere Probleme / Details schon gut / besser gelernt haben)
- Batch Normalization ()
- Dropout (; → [2.3.x.y. Dropout](#))
- Noise ()
- Ensembling (mehrere ((leicht) unterschiedliche) Netze trainieren und Ergebnisse kombinieren)
- Meta Learning (Analysieren, wie ein Netz lernt, um es dann zu optimieren)

besondere Lern-Situationen mit wenigen Daten:

- **Zero Shot learning** keine Trainings-Daten vorhanden
- **One Shot learning** nur ein Trainings-Daten-Element / -Satz vorhanden
- **Few Shot learning** nur wenige (z.B.: 5) Trainings-Daten-Elemente / -Sätze vorhanden

- **Semi-Supervised learning**

verwendet viele ungelabelte Daten zum vortrainieren des Netzes
mit wenigen gelabelten Daten wird dann ein Fein-Tuning durchgeführt

-

2.6.x. besondere Methoden / Techniken zum Training von KNN's

hier geht es um die Besprechung einiger vorne erwähnten Methoden, Techniken und Trick's
wir verlassen hier die Pfade der allgemeinen / einführenden Betrachtungen und zweigen
Spezialsachen

mehr was für Interessierte und Fortgeschrittene

damit die Kapitel zum Experimentieren nicht mit den Spezial-Sachverhalten überfrachtet und
gestört werden, nehme ich Quell-Texte usw. gleich hier mit auf

Wer dieses Kapitel jetzt trotzdem lesen will, sollte aber zumindestens die Keras-Abschnitt
überspringen. Nur die Eingeweihten können hier schon intensiver einsteigen. Bei Ihnen wer-
den dann die nächsten Kapitel eher übersprungfähig sein.

Dieses Kapitel sollte man also auch erst beackern, wenn man schon Erfahrungen mit KNN's
- vorrangig in Python / Keras und Co (→ [4. praktisches Experimentieren mittels Python und Co](#)) - hat

Regularisierung sind Techniken / ..., die ein Netz vor Overfittig schützen sollen
Ziel ist eine sehr gute Generalisierung durch das Netz

Regularisierungs-Techniken

- **L1- / L2-Regularisierung**

- **Dropout**

in den Trainings-Schritten werden einzelne Neuronen mit
einer bestimmten Wahrscheinlichkeit ausgeschaltet
(praktisch werden die Gewicht auf Null gesetzt)

- **Early Stopping**

2.6.x.y. Dropout

ist eine Regularisierungs-Methode
besonders beim Overfitting interessant

Ziel des Anwenders ist ja eine hohe Genauigkeit auf den Test-Daten (Generalisierung des
Netzes sollte also bestens funktionieren), noch wichtiger ist die Genauigkeit auf den zukünf-
tigen Arbeits-Daten

dagegen ist das Ziel des Netzes eine hohe Genauigkeit auf den Trainings-Daten (hier be-
steht bei zu großer Anpassung die Gefahr des Overfittings (Netz hat die Trainings-Daten
"auswendig" gelernt)

ein großes Netz mit vielen Schichten und Neuronen kann gering-umfängliche Trainings-
Daten schnell "auswendig" lernen (→ Overfitting)

das Netz erreicht eine Genauigkeit (Akkuratheit) von 100 % auf den Trainings-Daten; bei den
Test-Daten erzielt man aber deutlich schlechtere Genauigkeiten

Overfitting eines Netzes setzt ein sehr stabiles (festes) Netz voraus

bei Dropout macht man sich das Zunutze und schaltet zufällig einzelne Neuronen aus

dazu werden einfach deren Gewichte (und sicherheitshalber auch dessen Bias) auf Null gesetzt

von diesem Neuron geht dann kein Signal an die Nachfolge-Schicht

bei einem overgefitteten Netz würde jetzt eine Verschlechterung der Genauigkeit auftreten

optimal sind 10 - 50 gedropte Neuronen

Dropout wird aber grundsätzlich nur im Training benutzt (um eben nur hier das Overfitting zu verhindern)

Dropout-Rate p ist somit auch ein Hyper-Parameter

sinnvoll ist Dropout in Fully-Connected-Classifiers

bei Netzen i.A. 20 - 30 % Dropout

bei convolutional neural networks (CNN) 40 - 50 % Dropout

Anwendung des Dropout-Verfahrens nur auf die letzten (Dense) Layer eines Netzes

im Convolutional-Teil des Netzes eher sparsamer Umgang mit Dropout empfohlen

man muss mit einer Reduzierung der Lern-Geschwindigkeit rechnen

wenn das Netz Overfitting zeigt, dann sollt der Dropout vergrößert / verstärkt werden

beobachtet man dagegen Underfitting (schlechte(re) Genauigkeit beim Training und beim Test), dann sollt der Dropout (wieder) reduziert werden

Dropout in Keras-Modellen

```
...
from tensorflow.keras.layers import Dropout
...

lernRate=0.0001
momentum=0.9
decay=0.001
anzFilter=32
ersteKernelGroesse=(7,7)

aktivierungsFkt='elu'

# Input Layer
inputSchicht=Input(shape=(224,224,3))

modell=Conv2D(filters=anzFilter, kernel_size=ersteKernelGroesse,
              activation=aktivierungsFkt)(inputSchicht)
modell=MaxPooling2D((2,2))(modell)

# Hidden Layer
for i in range(2,5):
    modell=Conv2D(filters= i * anzFilter, kernel_size=(3,3),
                  activation=aktivierungsFkt)(modell)
    modell=MaxPooling2D((2,2))(modell)

modell=Conv2D(filters= 5 * anzFilter, kernel_size=(3,3),
              activation=aktivierungsFkt, padding='same')(modell)
modell=MaxPooling2D((2,2))(modell)

# Fully Connected Classifier
modell=Flatten()(modell)
modell=Dense(dense_neurons, activation=aktivierungsFkt)(modell)
modell=Dropout(0.5)(modell)
```

```

Modell=Dense(dense_neurons, aktivierungsfkt='tanh')(modell)
modell=Dropout(0.5)(modell)

# Output Layer
ausgabeSchicht = Dense(anzKlassen, activation='softmax')(modell)

CNNdropoutModell=Model(inputSchicht,ausgabeSchicht)

optimierer=keras.optimizer.SGD(lr=lernRate, momentum=momentum,
                                decay=decay)
CNNdropoutModell.compile(loss='square_categorical_crossentropy',
                        optimizer=optimierer, metrics=['accuracy'])

historyDropout=CNNdropoutModell.fit(train_data, epochs=100,
                                    validation_data=test_data)

```

da das Lernen mit Dropout i.A. verlangsamt wird, sollt eine vergrößerte Zahl von Epochen (hier: 100 statt z.B. 20 oder 40) verwendet werden

2.6.x.y. Batch Normalization

typische / mögliche Fehler beim Training von KNN's

bei vielen Schichten und Gradienten unter 1 ergeben sich für die ersten Schichten (die ja das Produkt aller vorherigen Gradienten (für die nachfolgenden Schichten) schnell sehr kleine (Gesamt-)Gradienten

dadurch gibt es kaum Auswirkungen auf die Gewichte der vorderen Schichten, der Lern-Effekt ist hier also gerade sehr gering (→ sterbende Gradienten; Vanishing Gradients)

dagegen kann bei Gradienten über 1 schnell eine sehr große Korrektur für die vorderen Schichten erfolgen (→ explodierende Gradienten; Exploding Gradients)

diesem Problem kann schon bei der Initialisierung der Gewichte entgegengewirkt werden

zwar initialisiert man weiterhin zufällig, aber die Gewichte werden dann noch in Abhängigkeit von den Daten skaliert

in Keras gibt es einen Kernel-Initialisierer, der zu den Standard-Vorgaben zählt und automatisch verwendet wird, es sei denn, man überschreibt diesen

werden keine Angaben beim Initialisieren einer Schicht gemacht, dann wird automatisch der uniforme-GLOROT-Initialisierer (auch als XAVIER bekannt) verwendet

```

Dense(1000, kernel_initializer='glorot_uniform', activation='tanh')
(modell)

```

bei mehreren Eingängen, die für sich unterschiedliche Domänen abdecken (Werte-Bereiche, Skalen) kann es zu "Irrretationen" im Netz kommen (Netz kennt die Bedeutung der Werte (vor allem zu Beginn) nicht)

Abhilfe kann eine Umskalierung auf ein Skalen-System bringen

zumeist relative Skala (0 .. 1) verwendet

selbst bei Bildern, die vielleicht schon nur eine Skala von 0 .. 255 (Graustufen für eine der drei Farben) kann Normalisierung etwas bringen, wenn z.B. die Bilder sehr hell oder sehr dunkel sind

in solchen Fällen konzentrieren sich die Grau-Werte nur auf kleine Teil-Bereiche aus dem 256-Werte-Gesamtbereich

Batch Normalisierung soll die extreme Weiter-Wirkung einer Neuron-Ausgabe auf die folgenden Schichten begrenzen / eindämmen

Verfahren:

1. den Erwartungs-Wert subtrahieren (→ Daten werden Null-zentriert)
2. teilen durch die Varianz (Standard-Abweichung)
3. skalieren
4. verschieben

durchgeführt auf jedem Batch (→ Batch Normalization)
(1. und 2, werden auch als Standardisierung bezeichnet)

B ... ein Batch an Daten

m ... Größe des Batch

x ... ein Datum

μ_B ... Erwartungswert für den Batch

σ_B^2 ... Varianz des Batch

ϵ ... kleine Zahl, um Division durch Null zu verhindern

γ ...

γ ... Skalierungs-Faktor

β ... Verschiebungs-Summand

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

$$\hat{x}_i^k = \frac{x_i^k - \mu_B^k}{\sqrt{\sigma_B^{k2} + \epsilon}}$$

$$y_i^k = \gamma^k \cdot \hat{x}_i^k + \beta^k$$

β und γ werden im Trainings-Prozess über Gradient Descent (→) gelernt

μ und σ werden im Trainings-Prozess für jeden einzelnen Batch berechnet

nach dem Training werden der globale Durchschnitt und die globale Varianz berechnet und gespeichert, diese werden dann für die Test's und Real-Daten genutzt

Regularisations-Effekt bei Batch Normalisation sorgt für leichte Verschiebung der Werte und wirkt wie Rauschen

der Gesamt-Effekt ist relativ klein (praktisch nur für Optimierung von Netzen sinnvoll)

Batch Normalization in Keras-Modellen

```
...
from tensorflow.keras.layers import BatchNormalization

...
lernRate=0.0001
momentum=0.9
decay=0.001
anzFilter=32
ersteKernelGroesse=(7,7)

aktivierungsFkt='elu'

# Input Layer
inputSchicht=Input(shape=(224,224,3))

modell=BatchNormalization(axis=[1,2])(inputSchicht)

modell=Conv2D(filters=anzFilter, kernel_size=ersteKernelGroesse,
```

```

        activation=aktivierungsFkt) (modell)
modell=BatchNormalization(axis=[1,2]) (modell)
modell=MaxPooling2D((2,2)) (modell)

# Hidden Layer
for i in range(2,5):
    modell=Conv2D(filters= i * anzFilter, kernel_size=(3,3),
        activation=aktivierungsFkt) (modell)
    modell=BatchNormalization(axis=[1,2]) (modell)
    modell=MaxPooling2D((2,2)) (modell)

modell=Conv2D(filters= 5 * anzFilter, kernel_size=(3,3),
    activation=aktivierungsFkt, padding='same') (modell)
modell=BatchNormalization(axis=[1,2]) (modell)
modell=MaxPooling2D((2,2)) (modell)

# Fully Connected Classifier
modell=Flatten() (modell)
modell=Dense(dense_neurons, aktivation=aktivierungsFkt) (modell)
modell=BatchNormalization() (modell)

Modell=Dense(dense_neurons, aktivation=aktivierungsFkt='tanh') (modell)
modell=BatchNormalization() (modell)

# Output Layer
ausgabeSchicht = Dense(anzKlassen, activation='softmax') (modell)

CNNdropoutModell=Model(inputSchicht,ausgabeSchicht)

optimierer=keras.optimizer.SGD(lr=lernRate, momentum=momentum,
    decay=decay)
CNNdropoutModell.compile(loss='square_categorical_crossentropy',
    optimizer=optimierer, metrics=['accuracy'])

historyDropout=CNNdropoutModell.fit(train_data, epochs=50,
    validation_data=test_data)

```

mit Batch Normalization wird das Netz schneller trainiert, an den Genauigkeiten ändert sich wenig

2.6.x.y. Data Augmentation

um sich Kosten-günstig mehr Bilder für das Training zu beschaffen, kann man die vorhandenen (meist ja sehr teuren) Bilder leicht abwandeln was für uns Menschen kaum sichtbar ist, übersehen wird oder als unwesentliche Veränderung empfunden wird, stellt KNN's vor kleine Probleme leicht veränderte Bilder ersetzen nicht zusätzliche neue Bilder, sie sind aber ein akzeptabler Teilersatz

Möglichkeiten:

- Rauschen (leichte Veränderung der Pixel-Werte)
- Zoom (Verkleinern oder Vergrößern des Bild-Ausschnittes (mit dem Objekt))
- Verschieben (Verändern der Lage des Objektes (durch Abschneiden an den verschiedenen Seiten))

- leicht rotieren / drehen
- Verzerren (unterschiedliches Verändern / Skalieren in den Bild-Dimensionen)
- Spiegel (allgemein um die y-Achse)

einzelne Verfahren müssen am Datensatz geprüft werden. Z.B. bringt das Spiegeln beim Erkennen von Ziffern nichts!

Will man die Scharfstellung beim Fotografieren trainieren, dann ist das Hinzufügen von verrauschten Informationen wahrscheinlich nicht förderlich

Data Augmentation in Keras-Modellen

```

...
        tf.keras.preprocessing.image.ImageDataGenerator
...

generator=tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1/255, rotation_range=0.3, width_shift_range=0.15,
    heighth_shift_range=0.15, zoom_range=0.15,
    # vertical_flip=True,
    horizontal_flip=True)

train_interator=generator.flow(X, y)
test_interator=generator.flow(X_test, y_test)
...

CNNAugmentModell=modelMachen()

historyAugmentation=CNNAugmentModell.fit(train_interator, epochs=40,
                                         validation_data=test_interator)

```

Data Augmentation wird zur Trainings-Zeit selbst erledigt

```

def datenSatzZuNP_Array(trainDaten, testDaten):

...

X, y, X_test, y_test = datenSatzZuNP_Array(trainDaten, testDaten)

```

Anzeige der veränderten Bilder

```

...
X_veraendert, y_veraendert=next(train_interator)
zeigeBilderMitLabel(X:veraendert, y_veraendert)
...

```

Rauschen lässt sich auch direkt als Keras-Layer in das Modell mit einfügen

```

...
...
from tensorflow.keras.layers import Dropout
...

lernRate=0.0001
momentum=0.9
decay=0.001
anzFilter=32
ersteKernelGroesse=(7,7)

aktivierungsFkt='elu'

# Input Layer
inputSchicht=Input(shape=(224,224,3))

inputBearbeiten=GaussianNoise(0.1)(inputSchicht)

modell=Conv2D(filters=anzFilter, kernel_size=ersteKernelGroesse,
              activation=aktivierungsFkt)(inputBearbeiten)
modell=MaxPooling2D((2,2))(modell)

# Hidden Layer
for i in range(2,5):
    modell=Conv2D(filters= i * anzFilter, kernel_size=(3,3),
                  activation=aktivierungsFkt)(modell)
    modell=MaxPooling2D((2,2))(modell)

modell=Conv2D(filters= 5 * anzFilter, kernel_size=(3,3),
              activation=aktivierungsFkt, padding='same')(modell)
modell=MaxPooling2D((2,2))(modell)

# Fully Connected Classifier
modell=Flatten()(modell)
modell=Dense(dense_neurons, activation=aktivierungsFkt)(modell)

Modell=Dense(dense_neurons, activation=aktivierungsFkt='tanh')(modell)

# Output Layer
ausgabeSchicht = Dense(anzKlassen, activation='softmax')(modell)

CNNAugmentModell=Model(inputSchicht,ausgabeSchicht)

optimierer=keras.optimizer.SGD(lr=lernRate, momentum=momentum,
                                decay=decay)
CNNAugmentModell.compile(loss='square_categorical_crossentropy',
                        optimizer=optimierer, metrics=['accuracy'])

historyAugmentation=CNNAugmentModell.fit(train_data, epochs=40,
                                         validation_data=test_data)
...

```

```

...
dataAugmentationModell=machenModell(lernRate=0.001
                                     # , noise=True
                                     )

historyAugmentation=CNNAugmentModell.fit(train_generator, epochs=40,
                                         validation_data=test_generator)

```

besonders bei wenigen Bildern als Trainings-Daten lassen sich positive Lern-Effekte erreichen und das Netz nicht mehr so stark zum Overfitting neigt

2.6.x.y. Kombination verschiedener Regularisierungs-Techniken

Kombination verschiedener Regularisierungen in Keras-Modellen

```
...
from tensorflow.keras.layers import Dropout
...

# Modell zusammenstellen
def machenModell(lernRate=0.0001, momentum=0.9, decay=0.001,
                anzFilter=32, ersteKernelGroesse=(7,7),
                aktivierungsFkt='elu', rauschen=False):
    # Input Layer
    inputSchicht=Input(shape=(224,224,3))
    eingabe=BatchNormalization(1,2)(inputSchicht)
    if rauschen:
        eingabe=GaussianNoise(0.1)(inputSchicht)

    modell=Conv2D(filters=anzFilter, kernel_size=ersteKernelGroesse,
                  activation=aktivierungsFkt)(eingabe)
    modell=MaxPooling2D((2,2))(modell)

    # Hidden Layer
    for i in range(2,5):
        modell=Conv2D(filters=i * anzFilter, kernel_size=(3,3),
                      activation=aktivierungsFkt)(modell)
        modell=Conv2D(filters=i * anzFilter, kernel_size=(3,3),
                    activation=aktivierungsFkt, padding='same')(modell)

        modell=BatchNormalization(1,2)(modell)
        modell=MaxPooling2D((2,2))(modell)

    modell=Conv2D(filters=5 * anzFilter, kernel_size=(3,3),
                  activation=aktivierungsFkt, padding='same')(modell)
    modell=Conv2D(filters=5 * anzFilter, kernel_size=(3,3),
                  activation=aktivierungsFkt, padding='same')(modell)
    modell=MaxPooling2D((2,2))(modell)

    # Fully Connected Classifier
    modell=Flatten()(modell)
    modell=Dense(dense_neurons, activation=aktivierungsFkt)(modell)
    modell=BatchNormalization()(modell)
    modell=Dropout(0.5)(modell)

    modell=Dense(dense_neurons, activation=aktivierungsFkt='tanh')(modell)
    modell=BatchNormalization()(modell)
    modell=Dropout(0.5)(modell)

    # Output Layer
    ausgabeSchicht = Dense(anzKlassen, activation='softmax')(modell)

    CNN_Modell=Model(inputSchicht,ausgabeSchicht)
```

```

    optimierer=keras.optimizer.SGD(lr=lernRate, momentum=momentum,
                                   decay=decay)
    CNN_Modell.compile(loss='square_categorical_crossentropy',
                      optimizer=optimierer, metrics=['accuracy'])
    return CNN_Modell

# Data Augmentation
generator=tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1/255, rotation_range=0.3, width_shift_range=0.15,
    height_shift_range=0.15, zoom_range=0.15,
    # vertical_flip=True,
    horizontal_flip=True)

trainGenerator_Leer=generator.flow(X, y)
testGenerator=generator.flow(X_test, y_test)

CNN_Modell=machenModell(verlauf=CNN_Modell.fit(trainGenerator_Leer,
                                              epochs=40, validationDaten=testGenerator)

generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255, rotation_range=0.1,
    width_shift_range=0.15, height_shift_range=0.15,
    zoom_range=0.15,
    horizontal_flip=True,
    # vertical_flip=True
)
train_interator=generator.flow(X, y)

X_veraendert, y_veraendert=next(trainGenerator)
druckenBilderMitLabels(X_veraendert[:9], y_veraendert[:9].astype('int'))

modellMitVeraendertenDaten= machenModell(lernRate=0.001,
    rauschen=True, anzFilter=64)
verlaufMitVeraendertenDaten= modellMitVeraendertenDaten.fit(
    trainGenerator, epochs=60,
    validation data=testGenerator)

druckenVerlauf("regularisiertes Netzwerk", verlaufMitVeraendertenDaten,

```

bei kursiv gesetzten Abschnitten handelt es sich Abschnitte, die bei einem gemeinsamen Anwenden verschiedener Regulationen ein Underfitting verhindern sollen
 bei einem Modell-System mit nur einer / wenig Regulation kann man ev. auf diese Abschnitte verzichten

schon in dieser Grundform liefert uns das mehrfach regularisierte Netz deutlich bessere Genauigkeiten, ohne dabei zu einem Overfitting zu neigen

auf diesem Modell (mit den zusätzlichen Regulationen) kann nun ein Fein-Tuning der Hyper-Parameter erfolgen (→ [Wie findet man nun die optimalen Werte für die Hyper-Parameter?](#))

2.7. Explainability (Erklärbarkeit)

gemeint auch Nachvollziehbarkeit der Entscheidungen (ev. auch der Zwischen-Schritte)

durch ihre Komplexität werden KI-Modelle schnell zu Black-Box-Systemen
wir betrachten nur die Inputs und die Outputs
die inneren Vorgänge bleiben zuerst einmal in der undurchsichtigen Box versteckt

Kriterium Komplexität ist nicht allein entscheidend für die schlechte Erklärbarkeit von Modell-Ergebnissen
z.B. sind auch Wetter-Modelle oder Modelle für die Simulation von Atom-Bomben extrem komplex

Interpretierbarkeit (Interpretability)	Erklärbarkeit (Explainability)
man erhält eine mögliche Erklärung (, aber nicht unbedingt alle Details / Zwischen-Ergebnisse nachvollziehbar)	man erhält eine Erklärung für die Entscheidung eines Modell's (, bei der alle Zwischen-Ergebnisse / Detail's nachvollziehbar sind)
man versteht nur begrenzt, was das Modell tut	man versteht genau, was das Modell tut
Wahrscheinlichkeits-Aussage	eindeutige Aussage
möglich für eine Vielzahl von Modellen	praktisch nur für einfache (lineare) Modelle möglich
potenziell mit relativ höherer Fehler-Quote behaftet	meist geringere Fehler-Quote

lokale Erklärbarkeit (Local Explainability)

- erklärt einzelne Vorhersagen
- hilft den Bias in einer lokalen Umgebung (einer Instanz) zu finden
- hilft zu prüfen, ob einzelne Voraussagen aus den richtigen Gründen / Regeln / ... getroffen wurden
- ...

globale Erklärbarkeit (Global Explainability)

- erklärt das Verhalten des gesamten Modell's
- hilft den Bias für eine größere Subgruppe zu finden
- hilft zu prüfen, ob das Modell für den breiten Einsatz geeignet ist
- ...

Analyse der Einfluss-Faktoren

Analyse der Bedeutung der einzelnen Faktoren / Parameter / Merkmale / ... auf die Entscheidung

Ermitteln der SHAP-Werte (SHAP ...)

für alle Input's wird deren Bedeutung / Anteil am Ergebnis ermittelt
geeignet für die lokale Erklärbarkeit

2.8. Transfer Learning

Strategien und Techniken zum zukünftigen Verbessern der KNN's

besonders bei fehlenden oder wenigen Trainings-Daten sind neuartige Techniken wichtig
gilt auch für das Erreichen des großen Ziel's einer allgemeinen künstlichen (starken) Intelligenz

im Augenblick sind die Leistungen der KNN's im Bereich der schwachen künstlichen Intelligenz angesiedelt

hier werden aber schon beachtlich gute Ergebnisse erzielt
vor allem die Unermüdlichkeit der KNN's kommt ihnen hier zugute

Menschen sind schon den Betrachten eines einzelnen Bildes - z.B. einer "neuen" / wenig bekannten Frucht - in der Lage weitere Abbildungen - selbst schematische Darstellungen - eindeutig dieser neuen Objekt-Klasse zuzuordnen

dabei hilft uns Menschen Jahrzehnte an Seh-Erfahrungen

Wie sieht ein 3D-Objekt auf einem 2D-Foto aus? Wie verändert sich die sichtbare Form, wenn das Objekt gedreht wird? Wie können die Farben bei anderen licht-Verhältnissen aussehen? Wie verändert sich ein Objekt mit zunehmenden Alter (hier z.B. Reife)?

...



Babaco (Berg-Papaya)
Q: flickr.com (Vilskogen)

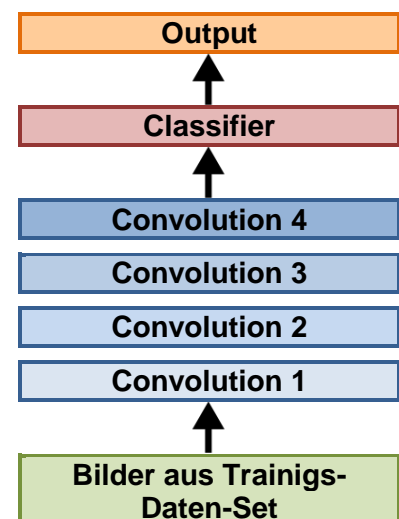
Der ImageNet-Datensatz besteht aus Millionen von Bildern aus 1000 Klassen.

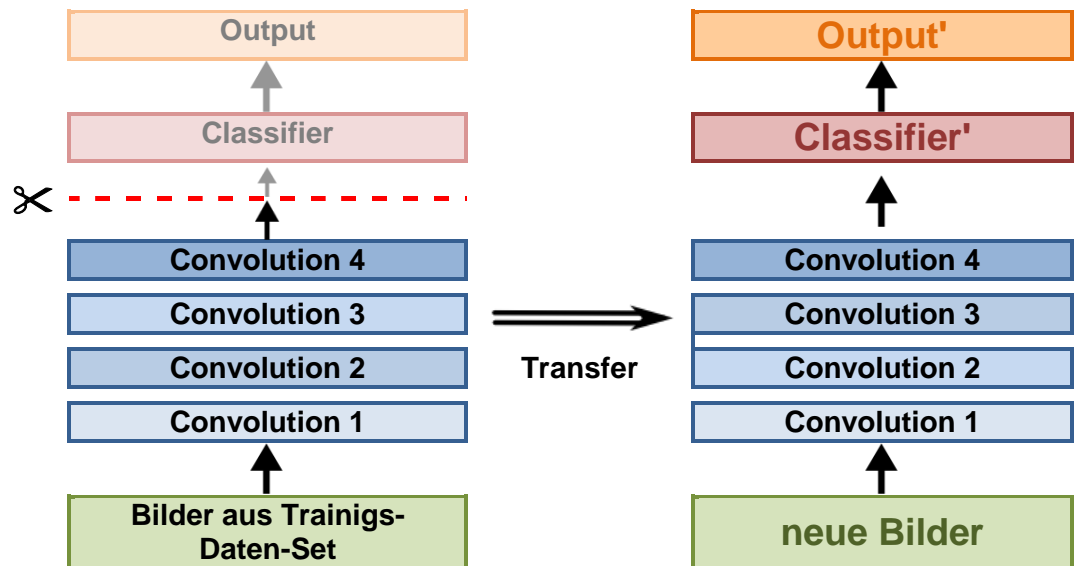
Aus erfolgreichen KNN's lassen sich die ersten Schichten - also solche, die einfache Strukturen erkennen - für andere Klassifikations-Aufgaben wiederverwenden
die oberen Schichten verbinden die Detail-Strukturen zu komplexen Strukturen, die dann eindeutig zu einer Klasse von Objekten passen.

Ein für eine Problemstellung trainiertes Netz könnte die nebenstehende Struktur besitzen.

Wir wissen, dass vor allem die unteren Schichten (hier Convolution), die Schichten sind, die Details / kleine Strukturen usw. erkennen. Der Classifier verbindet die "mittelkomplexen" Strukturen (hier Convolution 4) zu den Merkmals-Komplexen, die einem Klassen-Output entsprechen.

Für ein neues Problem kann es also sehr praktisch sein, den unteren Teil direkt zu übernehmen. Dies nennen wir Transfer. Nur den oberen Teil ersetzt man durch neu konzipierte Schichten, die dem neuen Problem entsprechen.





Beim Transfer werden die trainierten Schichten (hier Convolution) einschließlich der eintrainierten Gewichte usw. usf. beibehalten. Diese Schichten werden eingefroren (Freezed). Es werden lediglich der Classifier und die Output-Schicht neu zusammengestellt und an das neue Klassifikations-Problem angepasst (z.B. neue Output-Klassen).

In weiteren Versuchen und vor allem beim Vorliegen von mehr Bild-Material können dann schritt-weise von oben (hier: Convolution 4) das Lernen wieder ermöglicht werden. Je mehr Trainings-Daten vorhanden sind, umso mehr Schichten nach unten kann man für das netz-interne Lernen dann freigeben.

allgemein gilt:

Je mehr Trainings-Daten man hat, umso mehr Schichten (von oben betrachtet) sollten auch trainiert werden. Mit steigender Daten-Menge reduziert sich das Overfitting.

Zuerst sollte man nur den neuen Classifier und die Output-Schicht trainieren. Erst später kann man dann schritt-weise untere Schichten mit hinzunehmen. Hier würde man dann auch mit kleinen Lern-Raten arbeiten, um quasi das netz wieder ganz von unten lernen zu lassen.

2.7.1. Transfer Learning mit TensorFlow

gute vortrainierte Netze aus Wettbewerben der letzten Jahre:

- VGG
- ResNet
- GoogleLeNet
- MobileNet
- BigEarthNet
- Xception
-

übliches Vorgehen beim Nutzen fremder / anderer Netze beim Transfer Learning

1. **pretrainiertes Netz herunterladen**
2. **Classifier abschneiden**
3. **neuen Classifier hinzufügen**
4. **neuen Classifier trainieren** unter Beibehaltung der Convolution-Schichten (und deren Gewichte, ...)
5. **verkleinern der Learning-Rate** durch Training des gesamten Netzes
6. **feintunen des gesamten Netzes**
7. **Data Augmentation einbauen**

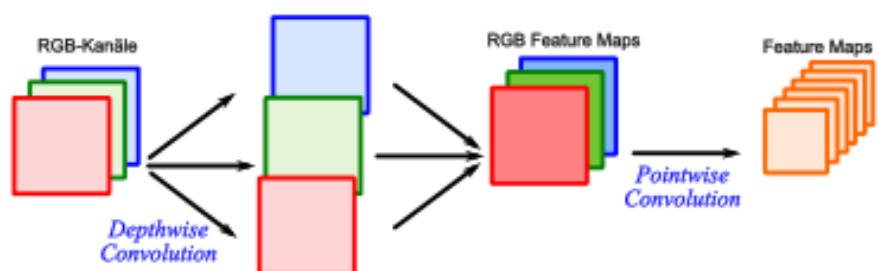
Beispiel: Xception

arbeitet mit Separable Convolutions
entstandene Schicht wird dann auch Separable Convolution genannt
dabei geht es um Intuition

Erweiterung des Feature-Maps-Konzepts
z.B. Analyse eines einzelnen Pixels
mögliche Features:
Farbtiefe

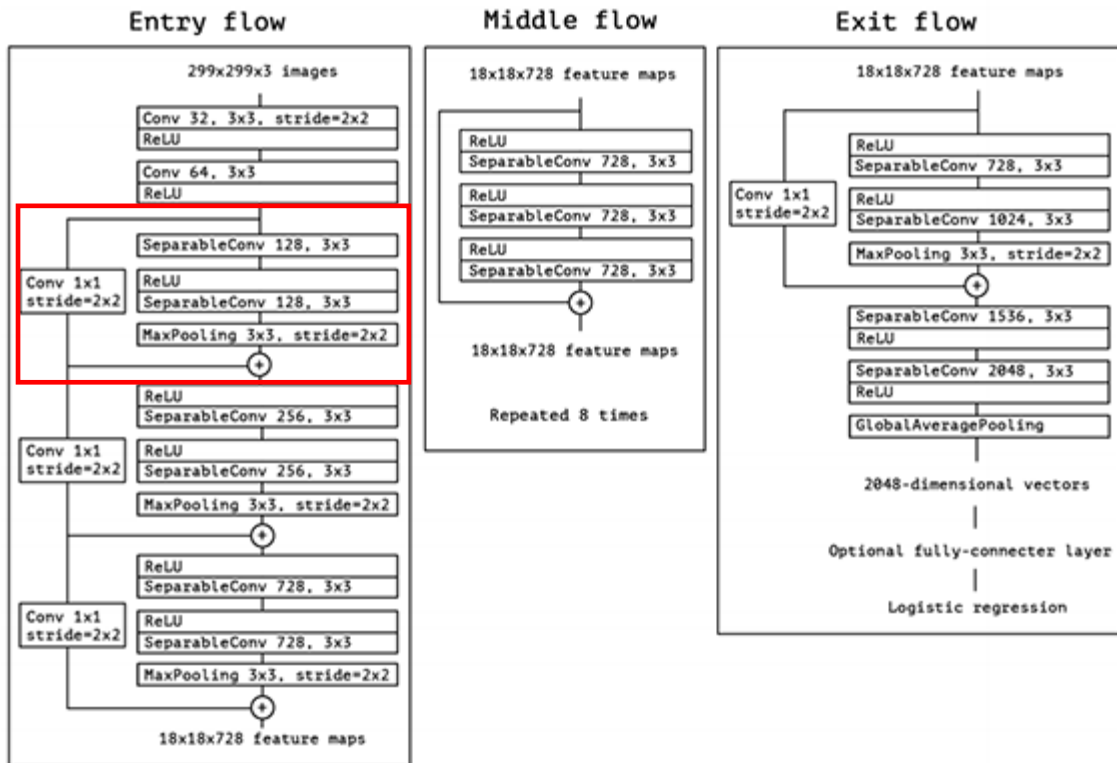
Convolution mit einem 1x1-Kernel (Pointwise Convolution)
es kommt zur Klassifizierung der einzelnen Pixels (z.B. hell oder dunkel, ...)

ausgehend von einem Feature werden für einen Pixel nun die räumliche Umgebungs-
Beziehungen / Lage innerhalb des Features
interessant auch umgebende Pixel → Gibt es Muster? Liegen andere "Augen" in der Nähe?
Bezug des Pixel's zu anderen Features (z.B. einer Kante)
es geht praktisch um die Bedeutung eines Pixels innerhalb eines Features



jetzt werden Convolution's auf den einzelnen Feature Map's durchgeführt
Verfahren nennt man Depthwise Convolution

praktisch teilt man z.B. zuerst die RGB-Kanäle auf und führt darauf die Depthwise Convolution aus
 danach wird dann auf den gewonnenen Features die Pointwise Convolution angewendet



Grob-Aufbau des Xception-Netzwerkes (mit einer Residual Unit (rot umrahmt))

Q: <https://medium.com/@gkadusumilli/image-recognition-using-pre-trained-xception-model-in-5-steps-96ac858f4206>

eine Besonderheit in diesem Konzept ist eine mehrfache Auteilung der Daten-Ströme auf verschiedene Bearbeitungs-Verfahren, deren Ergebnisse dann wieder zusammengeführt werden (eingekreiste +-Symbole)

dabei entstehen sogenannte Residual Unit's, die gerade in den Entwicklungen der letzten Jahren eine große Bedeutung erlangt haben
 neben z.B. einer SeparableConvolution gibt es eine SkipConvolution (quasi eine Art Bypass)
 die SkipConvolution (im obigen Modell: "Conv 1x1 stride= 2x2") wird über die gleichen Daten berechnet, wie die SeparableConvolution
 die Ergebnisse beider Verfahren werden vor der Weiter-Verarbeitung in folgendenden Modell-Abschnitten aufeinander addiert
 praktisch wird auf die SeperableConvolution der Input aufaddiert
 besonders bei den ersten Trainings wird der originale Input verstärkt (fast ausschließlich) weitergeleitet
 erst später mit den neu gelernten Gewichten ändern sich die Anteile am effektiven Ergebnis
 insgesamt lernt das Netz dadurch deutlich schneller
 mehr Skip's verstärken den Effekt
 durch die Residual Unit's ist es möglich Netze - bei gleicher Performance - tiefer aufzubauen, was insgesamt die Akkuratheit erhöht

Beispiel-Umsetzung (Xception-Netzwerk):

```
pip install tf-explain
```

ev. vorher ein Update von pip selbst machen (vor allem wenn es beim obigen pip-Kommando zu Problemen kommt)

```
#import librariesimport numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tf_explain.core.activations import ExtractActivations
from tensorflow.keras.applications.xception import decode_predictions
%matplotlib inline

#load pre trained Xception modelmodel=
    tf.keras.applications.xception.Xception(weights='imagenet',
        include_top=True)#Summary of Xception Model
print(model.summary)

#loading and preprocessing cat imageIMAGE_PATH='./lion.jpg'
img=tf.keras.preprocessing.image.load_img(IMAGE_PATH,
    target_size=(299,299))
img=tf.keras.preprocessing.image.img_to_array(img)#view the image
plt.imshow(img/255.)

import requests#fetching labels from Imagenet
response=
requests.get('https://storage.googleapis.com/download.tensorflow.org/d
ata/imagenet_class_index.json')
imgnet_map=response.json()imgnet_map={v[1]:k for k, v in
imgnet_map.items()}#make model
predictionsimg=tf.keras.applications.xception.preprocess_input(img)
predictions=model.predict(np.array([img]))
decode_predictions(predictions,top=5)
```

Q: <https://medium.com/@gkadusumilli/image-recognition-using-pre-trained-xception-model-in-5-steps-96ac858f4206>

2.7.2. Transfer Learning mit Keras-Modellen



2.x. ausgewählte Wiederholungen zu Python

Die klassischen Programier-Elemente von Python können wir hier nicht vorstellen, Dafür gibt es unzählige Materialien / Tutorial's / Skripte. Aus unser Serie ist das z.B. das Skript "" → .

2.x. semantische Netze in Python

Begriffs-Systeme / -Hierrachien

Tier → Lebewesen soll heißen ein "Tier" ist ein "Lebewesen"
wichtig ist die "ist_ein"-Beziehung, damit ist eine klare Unterordnung gemeint

in Python als Tupel darstellbar, das heißt beide Begriffe werden in runde Klammern Komma-getrennt aufgeführt:

```
("Tier", "Lebewesen")
```

auch als "gehört_zu" zu lesen ("Tier" gehört zu "Lebewesen")

Mehrere Tupel (Begriffs-Paare) werden in einer Liste gespeichert, also z.B.:

```
Wissen = [  
    ("Tier", "Lebewesen"),  
    ("Hund", "Tier")  
]
```

ev. brauchen wir auch mal die Umkehrung der Beziehung, das geht mit den Tupeln auch recht einfach, gemeint ist dann: ein "Lebewesen" kann sein ein "Tier"
auch hier ist wieder die Beziehung "kann_sein_ein" für uns interessant

man könnte auch ein Dictionary-Eintrag aus Schlüsselwort und Wert (Key-Value-Paar) nutzen

```
"Tier": "Lebewesen"
```

hier ist aber zu beachten, dass jeder Schlüssel nur einmal vorkommen darf
bei der Beziehung Pflanze → Lebewesen gäbe es keine Probleme, aber bei:

```
"Tier": "Ding"
```

funktioniert diese Wissen-Speicherform nicht mehr

Nun sind Begriffe aber auch nicht immer über- oder untergeordnet. Sollen z.B. Vokabeln zwei-sprachig gespeichert werden, dann ist es ja keine "ist_ein"-Beziehung mehr, sondern eher "ist_gleich".

Entweder wir machen eine neue Wissens-Liste auf, z.B.:

```
Vokabeln = [  
    ("Tier", "animal"),  
    ("Hund", "dog"),  
]
```

oder man speichert die Beziehungs-Art irgendwie mit in die Wissens-Liste. Dann sind wir aber schon bei Tripeln als Informations-Einheit

```
Wissen = [  
    ("Tier", "<", "Lebewesen"),  
    ("Tier", "=", "dog"),  
    ("Ding", ">", "Tier"),  
    ...  
]
```

]

Hierbei sind die Gleichheits- und Größer-Kleiner-Zeichen nur frei gewählte Symbole, man hätte auch Zahlen oder Buchstaben / Wörter nehmen können
Wir bleiben, der Einfachheit halber, beim Tupel oder einer eingeschränkten Tripel-Darstellung mit nur "<" und "=".

```
Wissen = [  
    ("Tier", "<", "Lebewesen"),  
    ("Tier", "=", "dog"),  
    ("Tier", "<", "Ding"),  
    ...  
]
```

für noch mehr wissen, können auch noch typische Eigenschaften und / oder Tätigkeiten / Aktivitäten gespeichert werden. Z.B.:

```
Wissen = [  
    ("Tier", "<", "Lebewesen", ["lebt", "atmet", "bewegt sich", "..."], ["frisst", "..."]),  
    ...  
]
```

In Anlehnung an ein Programm aus /15, S. / kann man eine erste Interpretation und Nutzung des Wissen's etwa so aussehen:

Diese Funktion testet einfach nur, ob es einen Eintrag für den ersten Begriff in unserer Wissens-Basis gibt und ob dieser zum zweiten Begriff einen gemeinsamen Eintrag enthält.
Als Rückgabe-Wert haben wir uns hier für WAHR bzw. FALSCH (in Python: True bzw. False) entschieden. Das lässt sich Programm-technisch später besser verarbeiten, als die "ja und "nein" im Original-Programm.
Mit einwenig Eingabe und Ausgabe kann dann auch eine sinnvolle Kommunikation mit dem Nutzer erfolgen. Ansonsten nutzt die Funktion mehr der flüchtigen Konsolen-Arbeit und der weiteren Programm-Nutzung.

Die IST_EIN()-Funktion ist nur ein sehr einfacher logischer Schluß. Es muss eine direkte Beziehung in unserer Wissens-Basis vorhanden sein, damit es ein positives Ergebnis gibt.
Für indirekte Schlüsse über Zwischen-Stationen benötigen wir eine erweiterte Funktion, die wir hier IST_IST_EIN() genannt haben. Im Gegensatz zur einfachen IST_EIN()-Funktion testet sich auch, ob über andere Begriffe eine Brücke von dem einen Begriff zum anderen aufgebaut werden kann. Natürlich nur auf der Basis des bekannten Wissen's.

Die Funktion ist rekursiv angelegt, so dass praktisch unendlich lange Ketten aufgebaut werden können. Diese Kette ist aber nur innerhalb der Funktion bekannt. Als Ergebnis erhalten wir wieder nur True oder False zurück. Das reicht uns ja auch für den Anfang.
Wie beim Pokern, irgendwann wollen wir sehen.
Also programmieren wir uns eine Funktion, die die Kette von Begriffen – also den logischen Schluß – darstellt.

Mit einem etwas umfangreicheren Hauptprogramm bekommen wir dann ein schönes "intelligentes" Programm.

Aufgaben:

- 1. Erweitern Sie das Wissen um 20 weitere Begriff-Paare! Welches Thema sie nutzen, ist Ihnen überlassen.*
- 2. Führen Sie die Wissens-Stände aller Kursteilnehmer in einem Programm zusammen! Stellen Sie Anfragen, die auch über die ursprünglichen Themen hinausgehen! Was weiss der Computer jetzt schon alles?*
- 3. Erweitern Sie das Programm um die Möglichkeit, auch ein neues Begriffs-Paar einzugeben und dieses in die Wissens-Basis zu integrieren!*
- 4. Prüfen Sie das neue Wissen auch im Zusammenhang mit dem alten Daten-Bestand!*

für die gehobene Anspruchsebene:

- 5. Programmieren Sie das Programm so, dass es am Ende sein Wissen in eine Text-Datei speichert! Dann sollte das Programm auch am Anfang prüfen, ob eine Daten-Datei vorliegt und diese dann einlesen! Wenn keine Datei vorhanden ist, dann soll das Programm nur die fest einprogrammierten Begriffs-Paare benutzen.*

2.x. einfache Implementierungen von Perceptron und Künstlichen Neuronalen Netzen in Python

gemeint sind hier ganz betont einfachste Programme die Verwendung von typischen Bibliotheken, wie usw. usf. soll hier mit Absicht ausgeschlossen werden, da wir hier direkt nachvollziehen wollen, was in einem KNN passiert und wie man dies vereinfacht umsetzen kann

2.x.y. Programmierung eines Perceptron's

eine eher abstrakte Umsetzung eines Perceptron's haben wir schon vorne (→ [Umsetzung eines Perceptron's in Python](#)) aufgezeigt
Diese ist mehr am Original von ROSENBLATT orientiert.

hier eine etwas verständlichere Umsetzung

aus /15/ als Paket-Erkennungs-System

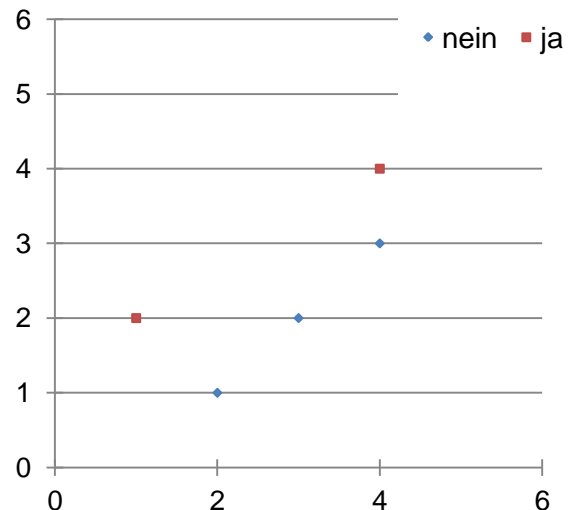
System soll entscheiden, ob ein Paket – nur von außen betrachtet – ein Geschenk ist oder nicht

als äußere Merkmale stehen die "Helligkeit" und die "Größe" des Paket's (in imaginären Einheiten) bereit

in den folgenden Diagrammen ist die Helligkeit immer auf der X-Achse und die Größe immer auf der Y-Achse abgetragen

von den Paketen kennt man die Zuordnung, diese werden zum Lernen benutzt (s.a. Diagramm)

mit weiteren wird dann getestet



```
# Original aus SCHMID, WEITZ, SIEBERS:  
# Künstliche Intelligenz selber programmieren für DUMMIES  
# ISBN 978-3-527-71573-2  
1 # Pakete zum Lernen  
2 lern_pakete = [  
3     (4, 3, "nein"),  
4     (1, 2, "ja"),  
5     (2, 1, "nein"),  
6     (4, 4, "ja"),  
7     (3, 2, "nein")]  
8  
9 # Anfängliche Wichtigkeiten  
10 wichtigkeit_helligkeit = 1  
11 wichtigkeit_groesse = 1  
12
```

Start-Werte

```

13 durchlauf = 0
14 fehler = True
15
16 # Wiederhole bis alles richtig gelernt wurde
17 # aber maximal 5 Durchläufe
18 while fehler and durchlauf < 5:
19     durchlauf = durchlauf + 1
20     fehler = False
21
22     # Schleife über alle Lernpakete
23     for paket in lern_pakete:
24         # Paket entpacken
25         helligkeit, groesse, geschenk = paket
26
27         # Berechnung der Gesamtwichtigkeit
28         wert1 = helligkeit * wichtigkeit_helligkeit
29         wert2 = groesse * wichtigkeit_groesse
30         gesamtwichtigkeit = wert1 + wert2
31
32         # Entscheidung
33         if gesamtwichtigkeit >= 0:
34             geschenk_erwartet = "ja"
35         else:
36             geschenk_erwartet = "nein"
37
38         # Wichtigkeiten anpassen
39         if geschenk_erwartet == "ja" \
40             and geschenk == "nein":
41             wichtigkeit_helligkeit = \
42                 wichtigkeit_helligkeit - helligkeit
43             wichtigkeit_groesse = \
44                 wichtigkeit_groesse - groesse
45             fehler = True
46         if geschenk_erwartet == "nein" \
47             and geschenk == "ja":
48             wichtigkeit_helligkeit = \
49                 wichtigkeit_helligkeit + helligkeit
50             wichtigkeit_groesse = \
51                 wichtigkeit_groesse + groesse
52             fehler = True
53
54 # Pakete zum Testen
55 test_pakete = [
56     (5, 3),
57     (1, 4),
58     (4, 1),
59     (1, 3)]
60
61 # Schleife über alle Testpakete
62 for paket in test_pakete:
63     # Paket entpacken
64     helligkeit, groesse = paket
65
66     # Berechnung der Gesamtwichtigkeit
67     wert1 = helligkeit * wichtigkeit_helligkeit
68     wert2 = groesse * wichtigkeit_groesse
69     gesamtwichtigkeit = wert1 + wert2
70
71     # Entscheidung
72     if gesamtwichtigkeit >= 0:
73         geschenk_erwartet = "ja"
74     else:
75         geschenk_erwartet = "nein"
76
77     print(geschenk_erwartet)
78

```

Annahme, dass schon fertig gelernt wurde

Gewicht des aktuellen Paket's berechnen

aktuelles Paket bewerten und mit Erwartungs-Wert vergleichen

Anpassen der Gewichte falls Entscheidung falsch war

Korrektur des Lern-Erfolgs

Testen!?
(hier Testen mit quasi neuen Paketen, wie in einem Praxis-Einsatz)

Q: /15, S. 60/ (aus online-Resource kopiert)

das Original-Programm bringt immer das gleiche Ergebnis

lässt man sich am Ende der Lern-Schleife auch mal die erlernten Gewichte anzeigen, dann erhält man eine Trenn-Gerade mit dem Anstieg 1 (m als Quotient der Gewichte)

diese Trenn-Gerade ist im unteren Diagramm zu den Daten-Werten mit eingetragen

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC
v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "licen-
se()" for more information.
```

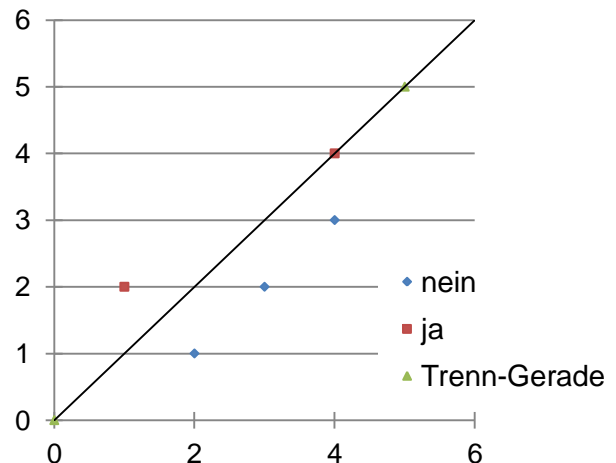
```
= RESTART: D:\XK_INFO\PK_INFO\Material\wiley-
vch.de\3527715738_code\Code_Mail9_3\03-
lernen\perzeptron.py
```

```
nein
ja
nein
ja
```

```
= RESTART: D:/XK_INFO/PK_INFO/Material/wiley-
vch.de/3527715738_code/Code_Mail9_3/03-
lernen/perzeptron_y.py
```

```
Gewicht-Helligkeit: -3          Gewicht-Größe: 3
nein
ja
nein
ja
```

die nun folgende Umsetzung arbeitet etwas KI-typischer – mit mehr Daten und einem zufälligen Split der bekannten Daten für die Lern- und die Test-Phase dadurch erhält man unterschiedliche Korrektheits-Werte und die Lern-Phase und kann auch mit deutlich mehr Lern-Runden arbeitendabei wurden die Daten entsprechend dem Original erweitert, d.h. an der imaginären Trenn-Gerade orientiert gewählt (s.a. folgendes Diagramm) um mehr Ganzzahl-Kombinationen zu ermöglichen, wurden die originalen Kennwerte einfach mal 10 gerechnet die Trenn-Linie wurde in Abhängigkeit vom Ergebnis des originalen Programm's und dessen Daten-Set eingezeichnet



(da ich einige Werte dicht an der Trenn-Gerade gelegt habe, sind die Ergebnisse schwankend und die Anzahl benutzter Runden bis zur Stabilisierung der Gewichte ist ebenfalls unterschiedlich)

außerdem können nach Lernen und Testen nun auch eigene Pakete getestet werden, quasi der Praxis-Einsatz

```
1 #original von Schmid, Weitz, Siebers: KI selber programmieren für DUMMIS
2 #angepasst und erweitert: drews
3 import random
4
5 # Pakete zum Lernen und Testen
6 pakete = [
7     (40, 30, "nein"),
8     (10, 20, "ja"),
9     (20, 10, "nein"),
10    (40, 40, "ja"),
11    (35, 15, "nein"),
12    (15, 25, "ja"),
13    (25, 20, "nein"),
```

```

14     (45, 45, "ja"),
15     (40, 35, "nein"),
16     (5, 10, "ja"),
17     (25, 15, "nein"),
18     (40, 45, "ja"),
19     (35, 30, "nein"),
20     (15, 15, "ja"),
21     (50, 30, "nein"),
22     (10, 40, "ja"),
23     (40, 10, "nein"),
24     (55, 40, "nein"),
25     (10, 20, "ja"),
26     (45, 45, "ja"),
27     (40, 35, "ja"), # Paket(-Daten) auch in anderer Klasse
28     (5, 10, "ja"),
29     (25, 15, "nein"),
30     (35, 30, "nein"),
31     (25, 20, "nein"),
32     (45, 40, "ja"),
33     (30, 20, "nein")]
34
35 # Pakete zum Lernen und Testen zufällig splitten
36 lern_pakete = []
37 test_pakete = []
38 for paket in pakete:
39     zufallszahl=random.random()
40     if zufallszahl<0.4: #40 % Test-Pakete absplitten
41         test_pakete.append(paket)
42     else:
43         lern_pakete.append(paket)
44
45 # Anfängliche Wichtigkeiten
46 wichtigkeit_helligkeit = 1
47 wichtigkeit_groesse = 1
48
49 durchlauf = 0
50 fehler = True
51
52 print("Perzeptron mit 2 Inputs und JA/NEIN-Ergebnis")
53 print("=====")
54 print("")
55 print("Lern-Phase")
56 print("=====")
57 print("")
58 print("aktuelle Pakete zum Lernen")
59 for paket in lern_pakete:
60     print(paket)
61 print("")
62 eingabe=input("Sollen die Werte im Lern-Vorgang angezeigt werden [j]a/nein?:
63 ")
64 if (len(eingabe) == 0) or (eingabe[0].lower() != "j"):
65     anzeige=False
66 else:
67     anzeige=True
68 print("")
69
70 # Wiederhole bis alles richtig gelernt wurde
71 # aber maximal x Durchläufe
72 maxDurchlaeufe = 100
73 while fehler and durchlauf <= maxDurchlaeufe:
74     durchlauf = durchlauf + 1
75     print("Durchlauf / Runde: ",durchlauf)
76     fehler = False
77
78 # Schleife über alle Lernpakete
79 for paket in lern_pakete:
80     # Paket entpacken
81     helligkeit, groesse, geschenk = paket
82
83     # Berechnung der Gesamtwichtigkeit
84     wert1 = helligkeit * wichtigkeit_helligkeit

```

```

84     wert2 = groesse * wichtigkeit_groesse
85     gesamtwichtigkeit = wert1 + wert2
86
87     # Entscheidung
88     if gesamtwichtigkeit >= 0:
89         geschenk_erwartet = "ja"
90     else:
91         geschenk_erwartet = "nein"
92
93     # Wichtigkeiten anpassen
94     if geschenk_erwartet == "ja" \
95         and geschenk == "nein":
96         wichtigkeit_helligkeit = \
97             wichtigkeit_helligkeit - hellerkeit
98         wichtigkeit_groesse = \
99             wichtigkeit_groesse - groesse
100        fehler = True
101    if geschenk_erwartet == "nein" \
102        and geschenk == "ja":
103        wichtigkeit_helligkeit = \
104            wichtigkeit_helligkeit + hellerkeit
105        wichtigkeit_groesse = \
106            wichtigkeit_groesse + groesse
107        fehler = True
108    if anzeige:
109        print("aktuelle Gewichte --> ",paket)
110        print("Helligkeit: ", wichtigkeit_helligkeit," Größe: ", wichtig-
111            keit_groesse)
112    print("")
113    if durchlauf<maxDurchlaeufe:
114        print("Abbruch / Fehler in Runde: ", durchlauf)
115    print("-----")
116    print("gelernte Gewichte")
117    print("Helligkeit: ", wichtigkeit_helligkeit," Größe: ", wichtigkeit_groesse)
118
119    print("")
120    print("Test-Phase")
121    print("=====")
122
123    # Testen
124
125    print("")
126    print("aktuelle Pakete zum Testen")
127    for paket in test_pakete:
128        print(paket)
129    print("")
130
131    # Schleife über alle Testpakete
132    richtig = 0
133    falsch = 0
134    for paket in test_pakete:
135        # Paket entpacken
136        hellerkeit, groesse, erwartung = paket
137
138        # Berechnung der Gesamtwichtigkeit
139        wert1 = hellerkeit * wichtigkeit_helligkeit
140        wert2 = groesse * wichtigkeit_groesse
141        gesamtwichtigkeit = wert1 + wert2
142
143        # Entscheidung und Bewertung und Anzeige
144        if gesamtwichtigkeit >= 0:
145            geschenk_erwartet = "ja"
146        else:
147            geschenk_erwartet = "nein"
148        print(paket,"--> Entscheidung für: ",geschenk_erwartet,end="")
149        if geschenk_erwartet == erwartung:
150            print("==> richtig")
151            richtig = richtig + 1
152        else:
153            print("==> falsch")

```

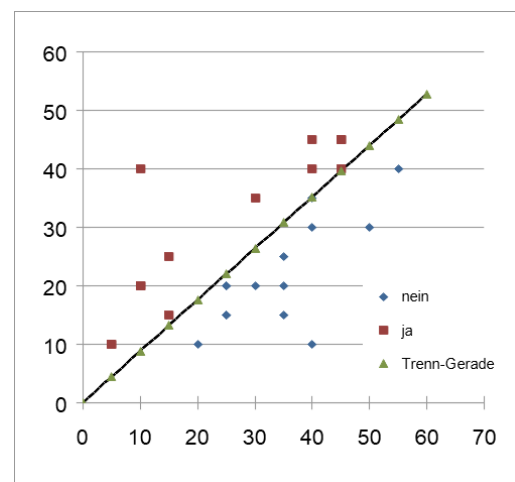
```

154     falsch = falsch + 1
155
156     #print(geschenk_erwartet)
157
158     # Praxis-Einsatz
159     print("")
160     print("=====")
161     print("Bewertung des angelernten Perzeptron's")
162     print("richtig: ",richtig," falsch: ",falsch," Akkuratheit:
",richtig/(richtig+falsch))
163     print("")
164     print("")
165     print("Praxis-Phase: Test neuer Pakete")
166     print("=====")
167     print("Abbruch durch Eingabe einer 0")
168     print("")
169     abbruch=False
170     while not abbruch:
171         helligkeit=int(input("Helligkeit [1..100]: "))
172         if helligkeit > 0:
173             groesse=int(input("Größe [1..100]: "))
174             if groesse <= 0:
175                 abbruch = True
176             else: # mit zwei gültigen Werten prüfen
177                 wert1 = helligkeit * wichtigkeit_helligkeit
178                 wert2 = groesse * wichtigkeit_groesse
179                 gesamtwichtigkeit = wert1 + wert2
180
181                 # Entscheidung und Bewertung und Anzeige
182                 print("das Paket (",helligkeit,",",groesse,) wird wahrscheinlich
",end="")
183                 if gesamtwichtigkeit >= 0:
184                     print("ein (ja) Geschenk sein")
185                 else:
186                     print("kein (nein) Geschenk sein")
187                     print("")
188                     print("neues Paket...")
189             else: # abbruch bei helligkeit
190                 abbruch = True
191     print("Programm-Ende")
192

```

Im nebenstehenden Diagramm sind die erweiterten Daten-Punkte eingezeichnet. Die Trenngerade wurde aus den von der KI ermittelten Gewichten dazu eingezeichnet

Eine echte Regression mit zwei Daten-Reihen (1x "ja" und 1x "nein") ist nach meinem Kenntnis-Stand soeinfach (in einer Tabellen-Kalkulation) möglich.



Aufgaben:

1. **Vollziehen Sie die Programmierung nach!** (Wichtig ist hier wirklich das eigenständige Abtippen, damit man den Programm-Code und die Programmierung an sich nachvollziehen kann.)
2. **Experimentieren Sie mit dem System!** Bewerten Sie die Akkuratheit des Perzeptron's! Warum kommt eigentlich immer wieder was anderes beim Lernen heraus?
3. **Erweitern Sie die Daten-Basis um weitere Pakete!** Orientieren Sie sich an der oberen Grafik! Natürlich darf das eine oder andere Paket auch mal die Unterscheidungs-Gerade überschreiten. Welche Auswirkungen haben solche "Überschreitungen"?

für die gehobene Anspruchsebene:

4. **Bauen Sie das Programm so um, dass man mit beliebigen (ev. einzugebenen) Attributen arbeiten kann!**
5. **Verwenden Sie nun Daten eines eigenen Set's oder einer eigenen Grafik zum Anlernen und Testen des allgemeinen System's!**

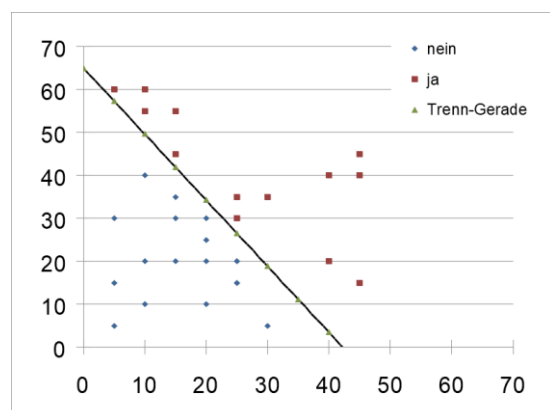
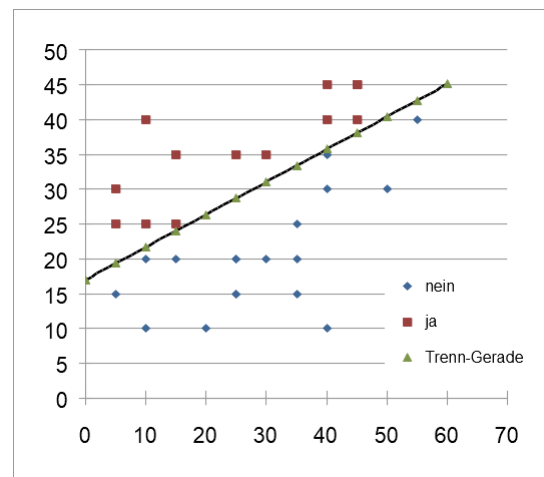
Ein Problem des Original-Programm's (und auch der bisher besprochenen Verbesserungen) ist die Einschränkung, dass die Trenn-Geraden durch den Null-Punkt gehen müssen.

Mit dem bei einem Perzeptron üblichen zusätzlichen Bias ist dies aber zu korrieren. Die entsprechende Umsetzung folgt. Die Eigenschaften können jetzt vordefiniert werden, so dass man ein recht universell einsetzbares Programm erhält.

Das Programm enthält zudem zwei mitgelieferte Daten-Set's. Das erste ist das leicht geänderte und erweiterte Set, was aus den Original-Daten von /15/ abgeleitet wurde. Das zweite Set ist eine Neuzusammenstellung von Daten so, dass eine fallende Trenn-Gerade (neg. Anstieg) herauskommt (s.a. Abb. rechts, unten).

Für beide Set's ist ein Bias notwendig und im Programm auch vorgesehen.

Das jeweils nicht benötigte Set und der zugehörige Bias müssen einfach nur auskommentiert werden.



```

1 #original von Schmid, Weitz, Siebers: KI selber programmieren für DUMMIS
2 #angepasst und erweitert: drews
3 import random
4
5 # daten zum Lernen und Testen
6 bewertung1 = "ja"
7 bewertung2 = "nein"
8 # Daten-Set1 (pos. Linie mit Bias = 15)
9 daten = [
10     (40, 30, bewertung2),
11     (10, 25, bewertung1),
12     (20, 10, bewertung2),
13     (40, 40, bewertung1),
14     (35, 15, bewertung2),
15     (15, 35, bewertung1),
16     (25, 20, bewertung2),
17     (45, 45, bewertung1),
18     (40, 35, bewertung2),
19     (5, 25, bewertung1),
20     (25, 15, bewertung2),
21     (40, 45, bewertung1),
22     (35, 25, bewertung2),
23     (15, 25, bewertung1),
24     (50, 30, bewertung2),
25     (10, 40, bewertung1),
26     (40, 10, bewertung2),
27     (55, 40, bewertung2),
28     (10, 40, bewertung1),
29     (25, 35, bewertung1),
30     (45, 45, bewertung1),
31     (30, 35, bewertung1),
32     (25, 15, bewertung2),
33     (35, 20, bewertung2),
34     (25, 20, bewertung2),
35     (5, 30, bewertung1),
36     (10, 10, bewertung2),
37     (15, 20, bewertung2),
38     (45, 40, bewertung1),
39     (10, 20, bewertung2)
40 ]
41
42 # Daten-Set 2 (neg. Linie mit Bias=65)
43 """"
44 daten = [
45     (20, 30, bewertung2),
46     (5, 60, bewertung1),
47     (20, 10, bewertung2),
48     (40, 40, bewertung1),
49     (30, 5, bewertung2),
50     (15, 55, bewertung1),
51     (25, 20, bewertung2),
52     (45, 45, bewertung1),
53     (15, 35, bewertung2),
54     (10, 55, bewertung1),
55     (25, 15, bewertung2),
56     (40, 20, bewertung1),
57     (5, 5, bewertung2),
58     (15, 45, bewertung1),
59     (15, 30, bewertung2),
60     (10, 60, bewertung1),
61     (5, 5, bewertung2),
62     (10, 40, bewertung2),
63     (25, 35, bewertung1),
64     (25, 35, bewertung1),
65     (45, 15, bewertung1),
66     (30, 35, bewertung1),
67     (25, 15, bewertung2),
68     (5, 30, bewertung2),
69     (25, 20, bewertung2),

```



```

69     (25, 30, bewertung1),
70     (20, 20, bewertung2),
71     (10, 10, bewertung2),
72     (45, 40, bewertung1),
73     (10, 20, bewertung2)
74 ]
75 """
76 merkmallname = "Helligkeit"
77 merkmal2name = "Größe"
78
79
80 # daten zum Lernen und Testen zufällig splitten
81 lern_daten = []
82 test_daten = []
83 for datum in daten:
84     zufallszahl=random.random()
85     if zufallszahl<0.4: #40 % Test-daten absplitten
86         test_daten.append(datum)
87     else:
88         lern_daten.append(datum)
89
90 # Anfängliche Wichtigkeiten
91 wichtigkeit_merkmal1 = 1
92 wichtigkeit_merkmal2 = 1
93 bias = 15 # für Daten-Set1
94 #bias = 65 # für Daten-Set2
95 durchlauf = 0
96 fehler = True
97
98 print("Perzeptron mit 2 Inputs und JA/NEIN-Ergebnis")
99 print("=====")
100 print("")
101 print("Lern-Phase")
102 print("=====")
103 print("")
104 print("aktuelle Daten zum Lernen")
105 print("(" +merkmallname, ", ", merkmal2name, ", ", " Bewertung )")
106 for datum in lern_daten:
107     print(datum)
108 print("")
109 eingabe=input("Sollen Werte im Lern-Vorgang angezeigt werden [j]a/nein?: ")
110 if (len(eingabe) == 0) or (eingabe[0].lower() != "j"):
111     anzeige=False
112 else:
113     anzeige=True
114 print("")
115
116 # Wiederhole bis alles richtig gelernt wurde
117 # aber maximal x Durchläufe
118 maxDurchlaeufe = 100
119 while fehler and durchlauf < maxDurchlaeufe:
120     durchlauf = durchlauf + 1
121     print("Durchlauf / Runde: ",durchlauf)
122     fehler = False
123
124 # Schleife über alle Lerndaten
125 for datum in lern_daten:
126     # datum entpacken
127     merkmall1, merkmal2, geschenk = datum
128
129     # Berechnung der Gesamtwichtigkeit
130     wert1 = merkmall1 * wichtigkeit_merkmal1
131     wert2 = (merkmal2 - bias) * wichtigkeit_merkmal2
132     gesamtwichtigkeit = wert1 + wert2
133
134     # Entscheidung
135     if gesamtwichtigkeit >= 0:
136         erwartung_ergebnis = bewertung1
137     else:
138         erwartung_ergebnis = bewertung2
139

```

```

140     # Wichtigkeiten anpassen
141     if erwartung_ergebnis == bewertung1 and geschenk == bewertung2:
142         wichtigkeit_merkmal1 = wichtigkeit_merkmal1 - merkmall
143         wichtigkeit_merkmal2 = wichtigkeit_merkmal2 - (merkmal2 -
bias)
144         fehler = True
145     if erwartung_ergebnis == bewertung2 and geschenk == bewertung1:
146         wichtigkeit_merkmal1 = wichtigkeit_merkmal1 + merkmall
147         wichtigkeit_merkmal2 = wichtigkeit_merkmal2 + (merkmal2 -
bias)
148         fehler = True
149     if anzeige:
150         print("aktuelle Gewichte --> ",datum)
151         print(merkmallname,": ", wichtigkeit_merkmal1,"
",merkmal2name,": ", wichtigkeit_merkmal2)
152     print("")
153     if durchlauf<maxDurchlaeufe:
154         print("Abbruch / Fehler in Runde: ", durchlauf)
155     print("-----")
156     print("gelernte Gewichte")
157     print(merkmallname,": ", wichtigkeit_merkmal1,"      ",merkmal2name,": ",
wichtigkeit_merkmal2)
158
159
160
161     print("")
162     print("Test-Phase")
163     print("=====")
164     # Testen
165
166     print("")
167     print("aktuelle Daten zum Testen")
168     for datum in test_daten:
169         print(datum)
170     print("")
171     print("")
172     print("Auswertung des Lern-Erfolgs")
173     print("")
174
175     # Schleife über alle Testdaten
176     richtig = 0
177     falsch = 0
178     for datum in test_daten:
179         # datum entpacken
180         merkmall1, merkmall2, erwartung = datum
181
182         # Berechnung der Gesamtwichtigkeit
183         wert1 = merkmall1 * wichtigkeit_merkmal1
184         wert2 = (merkmal2 - bias) * wichtigkeit_merkmal2
185         gesamtwichtigkeit = wert1 + wert2
186
187         # Entscheidung und Bewertung und Anzeige
188         if gesamtwichtigkeit >= 0:
189             erwartung_ergebnis = bewertung1
190         else:
191             erwartung_ergebnis = bewertung2
192         print(datum,"--> Entscheidung für: ",erwartung_ergebnis,end="")
193         if erwartung_ergebnis == erwartung:
194             print(" ==> richtig")
195             richtig = richtig + 1
196         else:
197             print(" ==> falsch")
198             falsch = falsch + 1
199
200         #print(erwartung_ergebnis)
201
202     # Praxis-Einsatz
203     print("")
204     print("=====")
205     print("Bewertung des angelernten Perzeptron's")
206     print("richtig: ",richtig," falsch: ",falsch," Akkuratheit: "

```

```

,richtig/(richtig+falsch))
207 print("")
208 print("")
209 print("Praxis-Phase: Test neuer Daten")
210 print("=====")
211 print("Abbruch durch Eingabe einer 0")
212 print("")
213 abbruch=False
214 while not abbruch:
215     merkmall1=int(input(merkmall1name+" [1..1000]: "))
216     if merkmall1 > 0:
217         merkmall2=int(input(merkmall2name+" [1..1000]: "))
218         if merkmall2 <= 0:
219             abbruch = True
220         else: # mit zwei gültigen Werten prüfen
221             wert1 = merkmall1 * wichtigkeit_merkmal1
222             wert2 = (merkmall2 - bias) * wichtigkeit_merkmal2
223             gesamtwichtigkeit = wert1 + wert2
224
225             # Entscheidung und Bewertung und Anzeige
226             print("das Datum (" ,merkmall1," ,",merkmall2," ) wird wahrscheinlich
227 ",end="")
                if gesamtwichtigkeit >= 0:
228                     print("die Bewertung (" ,bewertung1," ) haben / bekommen")
229                 else:
230                     print("die Bewertung (" ,bewertung2," ) haben / bekommen")
231                 print("")
232                 print("neues Datum...")
233             else: # abbruch bei merkmall1 <= 0
234                 abbruch = True
235 print("Programm-Ende")
236

```

Als Daten wurden auch mal Werte-Paare genutzt, die eine anders verlaufende Trenn-Gerade beinhalten. Mit dem Original-Programm wäre man hier kläglich gescheitert.

Aufgaben:

- 1.
- 2.
- 3.

für die gehobene Anspruchsebene:

- x. *Bauen Sie das Programm so um, dass der Lern-Vorgang in einer separaten Funktion (mit möglichst wenigen Ausgaben) ausgelagert ist!*
- y. *Entwickeln Sie ein Programm, dass den Bias selbst ermittelt (ev. Eingabe des Minimum's und des Maximum's)! Dabei soll jeder Bias 5x gelernt und getestet werden. Die Bewertung der Akkuratheit (z.B. Durchschnitt) sollen dann zur Bewertung des Bias benutzt werden. Der beste Bias soll zum Schluß ausgegeben werden!*

weiteres Python-Beispiel für eine Perzeptron-Umsetzung

```
umbral = 0.5
tasa_de_aprendizaje = 0.1
pesos = [0, 0, 0]
conjunto_de_entrenamiento = [(1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0),
1), ((1, 1, 1), 0)]

def producto_punto(valores, pesos):
    return sum(valor * peso for valor, peso in zip(valores, pesos))

while True:
    print('-' * 60)
    contador_de_errores = 0
    for vector_de_entrada, salida_deseada in conjunto_de_entrenamiento:
        print(pesos)
        resultado = producto_punto(vector_de_entrada, pesos) > umbral
        error = salida_deseada - resultado
        if error != 0:
            contador_de_errores += 1
            for indice, valor in enumerate(vector_de_entrada):
                pesos[indice] += tasa_de_aprendizaje * error * valor
    if contador_de_errores == 0:
        break
```

Q: <https://es.wikipedia.org/wiki/Perceptr%C3%B3n>

für alle die, die des Spanischen nicht so mächtig sind, hier eine eingedeutschte Version:

```
schwelle = 0.5
lernRate = 0.1
preise = [0, 0, 0]
trainingsDaten = [(1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0), 1), ((1, 1,
1), 0)]

def produktPunkte(werte, preise):
    return sum(wert * preis for wert, preis in zip(werte, preise))

while True:
    print('-' * 60)
    fehlerZaehler = 0
    for eingabeVektor, erwartungsWerte in trainingsDaten:
        print(preise)
        ergebnis = produktPunkte(eingabeVektor, preise) > schwelle
        fehler = erwartungsWerte - ergebnis
        if fehler != 0:
            fehlerZaehler += 1
            for i, wert in enumerate(eingabeVektor):
                preise[i] += lernRate * fehler * wert
    if fehlerZaehler == 0:
        break
```

Q: <https://es.wikipedia.org/wiki/Perceptr%C3%B3n> (eingedeutscht: Dre

2.x.y.z. Programmierung eines Perceptron's mit 3 Eingängen

das folgende Original-Programm stammt ebenfalls aus /15/
als 3. Merkmal wird ein Paket-Aufkleber genutzt, der verschiedene Smiley's zeigt
dabei gibt es einen "traurig"-, einen "neutral"- und einen "fröhlich"-Smiley, die für den Computer in 1, 2 und 3 übersetzt werden
mit drei Merkmalen sind wir im drei-dimensionalen Raum und statt einer Trenn-Gerade erwarten wir jetzt eine Trenn-Ebene

```
1 # Pakete zum Lernen
2 lern_pakete = [
3     (4, 3, 1, "nein"),
4     (1, 2, 3, "ja"),
5     (2, 1, 2, "nein"),
6     (4, 4, 3, "ja"),
7     (3, 2, 3, "nein")]
8
9 # Anfängliche Wichtigkeiten
10 wichtigkeit_helligkeit = 1
11 wichtigkeit_groesse = 1
12 wichtigkeit_smiley = 1
13
14 durchlauf = 0
15 fehler = True
16
17 # Wiederhole bis alles richtig gelernt wurde
18 # aber maximal 5 Durchläufe
19 while fehler and durchlauf < 5:
20     durchlauf = durchlauf + 1
21     fehler = False
22
23     # Schleife über alle Lernpakete
24     for paket in lern_pakete:
25         # Paket entpacken
26         helligkeit, groesse, smiley, geschenk = paket
27
28         # Berechnung der Gesamtwichtigkeit
29         wert1 = helligkeit * wichtigkeit_helligkeit
30         wert2 = groesse * wichtigkeit_groesse
31         wert3 = smiley * wichtigkeit_smiley
32         gesamtwichtigkeit = wert1 + wert2 + wert3
33
34         # Entscheidung
35         if gesamtwichtigkeit >= 0:
36             geschenk_erwartet = "ja"
37         else:
38             geschenk_erwartet = "nein"
39
40         # Wichtigkeiten anpassen
41         if geschenk_erwartet == "ja" and geschenk == "nein":
42             wichtigkeit_helligkeit = wichtigkeit_helligkeit - helligkeit
43             wichtigkeit_groesse = wichtigkeit_groesse - groesse
44             wichtigkeit_smiley = wichtigkeit_smiley - smiley
45             fehler = True
46         if geschenk_erwartet == "nein" and geschenk == "ja":
47             wichtigkeit_helligkeit = wichtigkeit_helligkeit + helligkeit
48             wichtigkeit_groesse = wichtigkeit_groesse + groesse
49             wichtigkeit_smiley = wichtigkeit_smiley + smiley
50             fehler = True
51
52 # Pakete zum Testen
53 test_pakete = [
54     (5, 3, 1),
55     (1, 4, 3),
56     (4, 1, 3),
57     (1, 3, 3)]
```

```

58
59 # Schleife über alle Testpakete
60 for paket in test_pakete:
61     # Paket entpacken
62     helligkeit, groesse, smiley = paket
63
64     # Berechnung der Gesamtwichtigkeit
65     wert1 = helligkeit * wichtigkeit_helligkeit
66     wert2 = groesse * wichtigkeit_groesse
67     wert3 = smiley * wichtigkeit_smiley
68     gesamtwichtigkeit = wert1 + wert2 + wert3
69
70     # Entscheidung
71     if gesamtwichtigkeit >= 0:
72         geschenk_erwartet = "ja"
73     else:
74         geschenk_erwartet = "nein"
75
76     print(geschenk_erwartet)
77

```

das Erweiterungs-Prinzip kann nun unendlich fortgesetzt werden
damit verlassen wir aber die Möglichkeiten der Vorstellung des Daten-Raum's auf einem
zwei-dimensionalen Blatt
aber die Darstellung war ja auch nur eine Möglichkeit, sich die Zusammenhänge besser vor-
zustellen
bei mehr-dimensionalen Systemen ist dann sowieso mehr das mathematische Verständnis als
das bildliche notwendig

Aufgaben:

1. ***Übernehmen Sie das Programm! (Tippen Sie es wirklich ab, um die Programm-Detail's zu verstehen!)***
2. ***Lassen Sie sich die gelernten Gewichte anzeigen! Wie müssen die Werte interpretiert werden?***
3. ***Verbessern Sie das Programm mit mehr Daten (im Sinne des aus dem original-Programm abgeleiteten Ergebnisses), um den Split in Trainings- und Test-Daten sowie um einen angehängten Praxis-Teil! (Als Basis kann ohne weiteres das vorne besprochene Programm für 2 Merkmale benutzt werden!)***
4. ***Verallgemeinern Sie das Programm, so dass auch andere Merkmale außer Helligkeit, Größe und Smiley benutzt werden können!***

für die gehobene Anspruchsebene:

5. ***Erstellen Sie ein Programm zur Umsetzung eines Perzeptron's mit 5 Eingängen (Merkmalen)!***

2.x. Entscheidungs-Bäume in Python

Entscheidungen (Alternativen, Verzweigungen) in Python zu programmieren, ist einer der grundlegenden Programmier-Aufgaben.

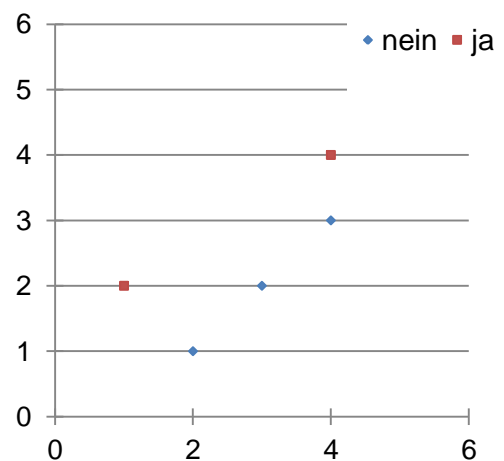
Bei der Umsetzung des Perzeptron's in Python haben wir u.a. eine Aufgabe / ein Problem aus /15/ benutzt: das Paket-Problem. Dabei ist nur durch die äußeren Merkmale Helligkeit und Größe (der Pakete) zu entscheiden, ob ein Paket ein Geschenk ist oder nicht.

Hier soll nun die Lösung des Problem's mithilfe von Entscheidungs-Bäume besprochen werden.

Zuerst benutzen wir wieder die originale Idee und das Programm aus /15/. ursprünglich stehen als äußere Merkmale die "Helligkeit" und die "Größe" des Paket's (in imaginären Einheiten) bereit

in den folgenden Diagrammen ist die Helligkeit immer auf der X-Achse und die Größe immer auf der Y-Achse abgetragen

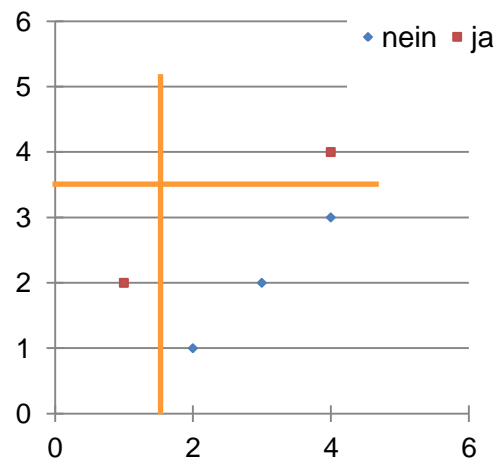
von den Paketen kennt man die Zuordnung, diese werden zum Lernen benutzt (s.a. Diagramm)



Helligkeit und Größe werden nun auf jeweils zwei Klassen reduziert: Die Helligkeit ist entweder "eher hell" oder "sehr dunkel". Bei der Größe unterscheiden wir zwischen "eher klein" und "eher groß"

Schaut man sich die Daten an, dann erkennen wir, dass die Entscheidung für "sehr dunkel" und "eher hell" zwischen den Werten 1 und 2 erfolgen muss.

Eine Analyse der Kategorisierung für die Größe – also zwischen "eher klein" und "eher groß" – findet zwischen den Größen-Werten 3 und 4 statt.



```
1 import entscheidungsbaum
2
3 # Beschreibung der Merkmale
```

```

4 merkmale = [
5     ("Helligkeit", ["sehr dunkel", "eher hell"]),
6     ("Größe", ["eher klein", "eher groß"])
7 ]
8
9 # Pakete zum Lernen
10 lern_pakete = [
11     ("eher hell", "eher klein", "nein"),
12     ("sehr dunkel", "eher klein", "ja"),
13     ("eher hell", "eher klein", "nein"),
14     ("eher hell", "eher groß", "ja"),
15     ("eher hell", "eher klein", "nein")]
16
17 # Anfänglicher Baum
18 baum = entscheidungsbaum.neuesNichtwissenBlatt()
19
20 durchlauf = 0
21 fehler = True
22
23 # Wiederhole bis alles richtig gelernt wurde
24 # aber maximal 5 Runden
25 while fehler and durchlauf < 5:
26     durchlauf = durchlauf + 1
27     fehler = False
28
29     # Schleife über alle Lernbeispiele
30     for paket in lern_pakete:
31         klasse_des_pakets = paket[-1]
32
33         # Beginne oben im Baum
34         knoten = baum
35         tiefe = 0
36
37         # Steige anhand der Merkmale des Pakets
38         # den Baum bis zu einem Blatt hinab
39         while not knoten.istEinBlatt() and tiefe < \
40             len(merkmale):
41             knoten = knoten.kindFuerWert(paket[tiefe])
42             tiefe = tiefe + 1
43
44         # Keine Ahnung
45         if knoten.istNichtwissenBlatt():
46             knoten.setzeKlasse(klasse_des_pakets)
47             fehler = True
48         # Falsche Klasse
49         elif not knoten.welcheKlasse() == \
50             klasse_des_pakets:
51             # weitere Merkmale vorhanden
52             if tiefe < len(merkmale):
53                 merkmalfname, merkmalfwerte = merkmale[tiefe]
54                 knoten.setzeAuswahlMerkmal(merkmalfname)
55
56             # Knoten hinzufügen
57             for wert in merkmalfwerte:
58                 if paket[tiefe] == wert:
59                     blattFuerWert = \
60                         entscheidungsbaum.neuesBlatt(\
61                             klasse_des_pakets)
62                 else:
63                     blattFuerWert = \
64                         entscheidungsbaum.neuesNichtwissenBlatt()
65                 knoten.neuesKind(wert, blattFuerWert)
66                 fehler = True
67
68         # keine weiteren Merkmale
69         else:
70             knoten.setzeKlasse(klasse_des_pakets)
71             fehler = True
72
73 print(baum)
74

```

ehemals:
(4, 3)
(1, 2)
(2, 1)
(4, 4)
(3, 2)


```

75 # Pakete zum Testen
76 test_pakete = [
77     ("eher hell", "eher klein"),
78     ("sehr dunkel", "eher groß"),
79     ("eher hell", "eher klein"),
80     ("sehr dunkel", "eher klein")]
81
82 # Schleife über alle Testbeispiele
83 for paket in test_pakete:
84     # Beginne oben im Baum
85     knoten = baum
86     tiefe = 0
87
88     # Steige anhand der Merkmale des Pakets
89     # den Baum bis zu einem Blatt hinab
90     while not knoten.istEinBlatt():
91         knoten = knoten.kindFuerWert(paket[tiefe])
92         tiefe = tiefe + 1
93
94     # Entscheidung
95     if knoten.istNichtwissenBlatt():
96         geschenk_erwartet = "*"
97     else:
98         geschenk_erwartet = knoten.welcheKlasse()
99
100     print(geschenk_erwartet)
101

```

ehemals:
(5, 3)
(1, 4)
(4, 1)
(1, 3)

Q: /15, S. 76/ (aus online-Resource kopiert)

Das obige Programm nutzt eine kleine Bibliothek, die die Anzeige der Bäume übernimmt. Sie wird am Anfang des Programm's importiert. Nachfolgend der Quell-Code dazu:

```

1 import os
2
3 class Knoten ():
4     def __init__(self):
5         self.children = dict()
6         self.label = None
7
8     def istEinBlatt(self):
9         return len(self.children)==0
10
11     def istNichtwissenBlatt(self):
12         return self.label is None
13
14     def setzeKlasse(self,klasse):
15         self.label=klasse
16
17     def welcheKlasse(self):
18         return self.label
19
20     def setzeAuswahlMerkmal(self,attribut_name):
21         self.label=attribut_name
22
23     def neuesKind(self,wert,kind):
24         self.children[wert] = kind
25
26     def kindFuerWert(self,wert):
27         if wert in self.children:
28             return self.children[wert]
29
30     def __repr__(self, spacing="",prefix=""):
31         label = self.label
32         if label is None:
33             label = "(*)"
34
35         result = spacing + prefix + label
36
37         if len(self.children) > 0:

```

```

38         result = result + ": " + os.linesep
39         for key in self.children.keys():
40             result = result + self.children[key].__repr__(
41                 spacing=spacing + " " * len(label) + "|",
42                 prefix="-" + key + ": ")
43         else:
44             result += os.linesep
45         return result
46
47 def neuesBlatt(klasse):
48     blatt = Knoten()
49     blatt.label=klasse
50     return blatt
51
52 def neuesNichtwissenBlatt():
53     return Knoten()
54

```

Q: /15, S. 76/ (aus online-Resource kopiert)

Helligkeit:

```

|-sehr dunkel: ja
|-eher hell: Größe:
|   |-eher klein: nein
|   |-eher groß: ja

```

```

nein
ja
nein
ja

```

gelernter
Entscheidungsbaum

Anwendung auf die
Test-Pakete

Zuerst einmal erweitern wir hier (wie weiter vorne beim Perzeptron) die Daten-Basis. Zur feineren Ergänzung neuer Pakete skalieren wir wieder mit 10 hoch. Dadurch können wir bequem bei Ganzzahlen bleiben.

Im ersten Abschnitt überlassen wir Python, die Kategorisierung sowohl für die Helligkeit als auch für die Größe. Dabei benutzt das Programm vorgegebene Grenzen in der Ausprägung der einzelnen Merkmale.

Die Datenstruktur wird beibehalten und durch weitere Daten ergänzt, um dann später flexibel eine feinere Kategorisierung der Merkmale zu ermöglichen. (Das sieht bei der einfachen Übertragung des Basis-Programm's etwas überzogen aus. Es erleichtert aber später eine Fein-Kategorisierung.)

Für das eigentliche Lern-Programm splitten wir die Daten wieder zufällig in Trainings- und Test-Daten.

Mit den Trainings-Daten lassen wird den Algorithmus einen Entscheidungs-Baum lernen und mit den Test-Daten prüfen wir den berechneten Baum. Außerdem berechnen wir wieder die Akkuratheit, da uns ja ein Erwartungs-Wert für die Test-Daten bereit steht.

```

1 #original von Schmid, Weitz, Siebers: KI selber programmieren für DUMMIS
2 #angepasst und erweitert: drews
3 import random
4 import entscheidungsbaum
5
6 # Beschreibung der Merkmale
7 merkmale = [
8     ("Helligkeit", ["sehr dunkel", "eher hell"]),
9     ("Größe", ["eher klein", "eher groß"])
10 ]
11
12 merkmalsgrenze = [
13     [15],
14     [35]
15 ]
16
17 # erweiterter Datenbestand (leicht geänderte Werte (x10 hochskaliert))

```

```

18 bewertung1 = "ja"
19 bewertung2 = "nein"
20 # Daten-Set1 (pos. Linie mit Bias = 15)
21 numDaten = [
22     (40, 30, bewertung2),
23     (10, 25, bewertung1),
24     (20, 10, bewertung2),
25     (40, 40, bewertung1),
26     (35, 15, bewertung2),
27     (15, 35, bewertung1),
28     (25, 20, bewertung2),
29     (45, 45, bewertung1),
30     (40, 35, bewertung2),
31     (5, 25, bewertung1),
32     (25, 15, bewertung2),
33     (40, 45, bewertung1),
34     (35, 25, bewertung2),
35     (15, 25, bewertung1),
36     (50, 30, bewertung2),
37     (10, 40, bewertung1),
38     (40, 10, bewertung2),
39     (55, 40, bewertung2),
40     (10, 40, bewertung1),
41     (25, 35, bewertung1),
42     (45, 45, bewertung1),
43     (30, 35, bewertung1),
44     (25, 15, bewertung2),
45     (35, 20, bewertung2),
46     (25, 20, bewertung2),
47     (5, 30, bewertung1),
48     (10, 10, bewertung2),
49     (15, 20, bewertung2),
50     (45, 40, bewertung1),
51     (10, 20, bewertung2)
52 ]
53
54 # Daten-Set 2 (neg. Linie mit Bias=65)
55 """
56 numDaten = [
57     (20, 30, bewertung2),
58     (5, 60, bewertung1),
59     (20, 10, bewertung2),
60     (40, 40, bewertung1),
61     (30, 5, bewertung2),
62     (15, 55, bewertung1),
63     (25, 20, bewertung2),
64     (45, 45, bewertung1),
65     (15, 35, bewertung2),
66     (10, 55, bewertung1),
67     (25, 15, bewertung2),
68     (40, 20, bewertung1),
69     (5, 5, bewertung2),
70     (15, 45, bewertung1),
71     (15, 30, bewertung2),
72     (10, 60, bewertung1),
73     (5, 5, bewertung2),
74     (10, 40, bewertung2),
75     (25, 35, bewertung1),
76     (25, 35, bewertung1),
77     (45, 15, bewertung1),
78     (30, 35, bewertung1),
79     (25, 15, bewertung2),
80     (5, 30, bewertung2),
81     (25, 20, bewertung2),
82     (25, 30, bewertung1),
83     (20, 20, bewertung2),
84     (10, 10, bewertung2),
85     (45, 40, bewertung1),
86     (10, 20, bewertung2)
87 ]
88 """

```

```

89 # Daten kategorisieren
90 katTab=[]
91 for m in merkmale:
92     katTab.append(m[1])
93 # print(katTab)
94
95 daten=[]
96 for datum in numDaten:
97     merkmallWert, merkmal2Wert, geschenk = datum # Paket auspacken
98     merkmallKat=katTab[0][0]
99     for mG in range(len(merkmalGrenze[0])): #Merkmal1
100         #print(len(merkmalGrenze[0]),mG)
101         if merkmallWert > merkmalGrenze[0][mG]:
102             merkmallKat=katTab[0][mG+1]
103     merkmal2Kat=katTab[1][0]
104     for mG in range(len(merkmalGrenze[1])): #Merkmal2
105         #print(len(merkmalGrenze[1]),mG)
106         if merkmal2Wert > merkmalGrenze[1][mG]:
107             merkmal2Kat=katTab[1][mG+1]
108     ''' alt
109     if merkmal2Wert <= 35:
110         merkmal2Kat=katTab[1][0]
111     else:
112         merkmal2Kat=katTab[1][1]
113     '''
114     katDatum=merkmallKat, merkmal2Kat, geschenk
115     daten.append(katDatum)
116     print(datum," --> ",katDatum)
117
118
119 # daten zum Lernen und Testen zufällig splitten
120 lern_pakete = []
121 test_pakete = []
122 for datum in daten:
123     zufallszahl=random.random()
124     if zufallszahl<0.4: #40 % Test-daten absplitten
125         test_pakete.append(datum)
126     else:
127         lern_pakete.append(datum)
128
129
130 # Anfänglicher Baum
131 baum = entscheidungsbaum.neuesNichtwissenBlatt()
132
133 durchlauf = 0
134 fehler = True
135
136 # Wiederhole bis alles richtig gelernt wurde
137 # aber maximal 5 Runden
138 while fehler and durchlauf < 5:
139     durchlauf = durchlauf + 1
140     fehler = False
141
142     # Schleife über alle Lernbeispiele
143     for paket in lern_pakete:
144         klasse_des_pakets = paket[-1]
145
146         # Beginne oben im Baum
147         knoten = baum
148         tiefe = 0
149
150         # Steige anhand der Merkmale des Pakets
151         # den Baum bis zu einem Blatt hinab
152         while not knoten.istEinBlatt() and tiefe < \
153             len(merkmale):
154             knoten = knoten.kindFuerWert(paket[tiefe])
155             tiefe = tiefe + 1
156
157         # Keine Ahnung
158         if knoten.istNichtwissenBlatt():
159             knoten.setzeKlasse(klasse_des_pakets)

```

```

160         fehler = True
161         # Falsche Klasse
162         elif not knoten.welcheKlasse() == \
163             klasse_des_pakets:
164             # weitere Merkmale vorhanden
165             if tiefe < len(merkmale):
166                 merkmal_name, merkmal_werte = merkmale[tiefe]
167                 knoten.setzeAuswahlMerkmal(merkmal_name)
168
169             # Knoten hinzufügen
170             for wert in merkmal_werte:
171                 if paket[tiefe] == wert:
172                     blattFuerWert = \
173                         entscheidungsbaum.neuesBlatt(\
174                             klasse_des_pakets)
175                 else:
176                     blattFuerWert = \
177                         entscheidungsbaum.neuesNichtwissenBlatt()
178                 knoten.neuesKind(wert, blattFuerWert)
179                 fehler = True
180
181             # keine weiteren Merkmale
182             else:
183                 knoten.setzeKlasse(klasse_des_pakets)
184                 fehler = True
185         print("Runde: ", durchlauf)
186         print(baum)
187
188     print("fertig gelernter Baum")
189     print("-----")
190     print(baum)
191
192     richtig=0
193     falsch=0
194     # Schleife über alle Testbeispiele
195     print("Entscheidung")
196     for paket in test_pakete:
197         # Beginne oben im Baum
198         knoten = baum
199         tiefe = 0
200
201         # Steige anhand der Merkmale des Pakets
202         # den Baum bis zu einem Blatt hinab
203         while not knoten.istEinBlatt():
204             knoten = knoten.kindFuerWert(paket[tiefe])
205             tiefe = tiefe + 1
206
207         # Entscheidung
208         if knoten.istNichtwissenBlatt():
209             geschenk_erwartet = "*"
210         else:
211             geschenk_erwartet = knoten.welcheKlasse()
212
213         print("für Paket", paket, ": ", geschenk_erwartet, end="")
214         if geschenk_erwartet == paket[2]:
215             print(" ---> richtig")
216             richtig=richtig+1
217         else:
218             print(" ---> falsch")
219             falsch=falsch+1
220     print("Akkuratheit: ", richtig/(richtig+falsch))
221

```

```

fertig gelernter Baum
-----
Helligkeit:
    |-sehr dunkel: Größe:
    |               |-eher klein: nein

```

```

|         |-eher groß: ja
|-eher hell: Größe:
|         |-eher klein: nein
|         |-eher groß: ja

```

Entscheidung

```

für Paket ('eher hell', 'eher groß', 'ja') : ja ---> richtig
für Paket ('sehr dunkel', 'eher klein', 'ja') : nein ---> falsch
für Paket ('eher hell', 'eher klein', 'nein') : nein ---> richtig
für Paket ('eher hell', 'eher klein', 'nein') : nein ---> richtig
für Paket ('sehr dunkel', 'eher groß', 'ja') : ja ---> richtig
für Paket ('eher hell', 'eher klein', 'ja') : nein ---> falsch
für Paket ('eher hell', 'eher klein', 'ja') : nein ---> falsch
für Paket ('eher hell', 'eher klein', 'nein') : nein ---> richtig
für Paket ('eher hell', 'eher klein', 'nein') : nein ---> richtig
für Paket ('sehr dunkel', 'eher klein', 'nein') : nein ---> richtig
Akkuratheit: 0.7

```

Im Allgemeinen ist jetzt die Akkuratheit nicht mehr so optimal. Das liegt aber auch an den erweiterten Daten. Die hatten wir ja an der Trenn-Gerade des Perzeptron's orientiert ergänzt.

Das nächste Programm benutzt das alternative Daten-Set (mit einer fallenden Trenn-Geraden und einem hohen Bias).

Desweiteren wird jedes Merkmal jetzt in 3 Kategorien unterteilt.

```

1 #original von Schmid, Weitz, Siebers: KI selber programmieren für DUMMIS
2 #angepasst und erweitert: drews
3 import random
4 import entscheidungsbaum
5
6 # Beschreibung der Merkmale
7 merkmale = [
8     ("Helligkeit", ["sehr dunkel", "mittel", "eher hell"]),
9     ("Größe", ["eher klein", "mittel", "eher groß"])
10 ]
11
12 merkmalGrenze = [
13     [15, 30],
14     [20, 40]
15 ]
16
17 # erweiterter Datenbestand (leicht geänderte Werte (x10 hochskaliert))
18 bewertung1 = "ja"
19 bewertung2 = "nein"
20 """"
21 # Daten-Set1 (pos. Linie mit Bias = 15)
22 numDaten = [
23     (40, 30, bewertung2),
24     (10, 25, bewertung1),
25     (20, 10, bewertung2),
26     (40, 40, bewertung1),
27     (35, 15, bewertung2),
28     (15, 35, bewertung1),
29     (25, 20, bewertung2),
30     (45, 45, bewertung1),
31     (40, 35, bewertung2),
32     (5, 25, bewertung1),
33     (25, 15, bewertung2),
34     (40, 45, bewertung1),
35     (35, 25, bewertung2),
36     (15, 25, bewertung1),
37     (50, 30, bewertung2),
38     (10, 40, bewertung1),
39     (40, 10, bewertung2),

```

```

40     (55, 40, bewertung2),
41     (10, 40, bewertung1),
42     (25, 35, bewertung1),
43     (45, 45, bewertung1),
44     (30, 35, bewertung1),
45     (25, 15, bewertung2),
46     (35, 20, bewertung2),
47     (25, 20, bewertung2),
48     (5, 30, bewertung1),
49     (10, 10, bewertung2),
50     (15, 20, bewertung2),
51     (45, 40, bewertung1),
52     (10, 20, bewertung2)
53 ]
54
55 # Daten-Set 2 (neg. Linie mit Bias=65)
56 """"
57 numDaten = [
58     (20, 30, bewertung2),
59     (5, 60, bewertung1),
60     (20, 10, bewertung2),
61     (40, 40, bewertung1),
62     (30, 5, bewertung2),
63     (15, 55, bewertung1),
64     (25, 20, bewertung2),
65     (45, 45, bewertung1),
66     (15, 35, bewertung2),
67     (10, 55, bewertung1),
68     (25, 15, bewertung2),
69     (40, 20, bewertung1),
70     (5, 5, bewertung2),
71     (15, 45, bewertung1),
72     (15, 30, bewertung2),
73     (10, 60, bewertung1),
74     (5, 5, bewertung2),
75     (10, 40, bewertung2),
76     (25, 35, bewertung1),
77     (25, 35, bewertung1),
78     (45, 15, bewertung1),
79     (30, 35, bewertung1),
80     (25, 15, bewertung2),
81     (5, 30, bewertung2),
82     (25, 20, bewertung2),
83     (25, 30, bewertung1),
84     (20, 20, bewertung2),
85     (10, 10, bewertung2),
86     (45, 40, bewertung1),
87     (10, 20, bewertung2)
88 ]
89
90 # Daten kategorisieren
91 katTab=[]
92 for m in merkmale:
93     katTab.append(m[1])
94 # print(katTab)
95
96 daten=[]
97 for datum in numDaten:
98     merkmallWert, merkmall2Wert, geschenk = datum # Paket auspacken
99     merkmall1Kat=katTab[0][0]
100     for mG in range(len(merkmallGrenze[0])): #Merkmal1
101         #print(len(merkmallGrenze[0]),mG)
102         if merkmallWert > merkmallGrenze[0][mG]:
103             merkmall1Kat=katTab[0][mG+1]
104     merkmall2Kat=katTab[1][0]
105     for mG in range(len(merkmallGrenze[1])): #Merkmal2
106         #print(len(merkmallGrenze[1]),mG)
107         if merkmall2Wert > merkmallGrenze[1][mG]:
108             merkmall2Kat=katTab[1][mG+1]
109     ''' alt
110     if merkmall2Wert <= 35:

```

```

111     merkmal2Kat=katTab[1][0]
112     else:
113         merkmal2Kat=katTab[1][1]
114         '''
115         katDatum=merkmal1Kat, merkmal2Kat, geschenk
116         daten.append(katDatum)
117         print(datum," --> ",katDatum)
118
119
120 # daten zum Lernen und Testen zufällig splitten
121 lern_pakete = []
122 test_pakete = []
123 for datum in daten:
124     zufallszahl=random.random()
125     if zufallszahl<0.4: #40 % Test-daten absplitten
126         test_pakete.append(datum)
127     else:
128         lern_pakete.append(datum)
129
130
131 # Anfänglicher Baum
132 baum = entscheidungsbaum.neuesNichtwissenBlatt()
133
134 durchlauf = 0
135 fehler = True
136
137 # Wiederhole bis alles richtig gelernt wurde
138 # aber maximal 5 Runden
139 while fehler and durchlauf < 5:
140     durchlauf = durchlauf + 1
141     fehler = False
142
143     # Schleife über alle Lernbeispiele
144     for paket in lern_pakete:
145         klasse_des_pakets = paket[-1]
146
147         # Beginne oben im Baum
148         knoten = baum
149         tiefe = 0
150
151         # Steige anhand der Merkmale des Pakets
152         # den Baum bis zu einem Blatt hinab
153         while not knoten.istEinBlatt() and tiefe < \
154             len(merkmale):
155             knoten = knoten.kindFuerWert(paket[tiefe])
156             tiefe = tiefe + 1
157
158         # Keine Ahnung
159         if knoten.istNichtwissenBlatt():
160             knoten.setzeKlasse(klasse_des_pakets)
161             fehler = True
162         # Falsche Klasse
163         elif not knoten.welcheKlasse() == \
164             klasse_des_pakets:
165             # weitere Merkmale vorhanden
166             if tiefe < len(merkmale):
167                 merkmal_name, merkmal_werte = merkmale[tiefe]
168                 knoten.setzeAuswahlMerkmal(merkmal_name)
169
170             # Knoten hinzufügen
171             for wert in merkmal_werte:
172                 if paket[tiefe] == wert:
173                     blattFuerWert = \
174                         entscheidungsbaum.neuesBlatt(\
175                             klasse_des_pakets)
176                 else:
177                     blattFuerWert = \
178                         entscheidungsbaum.neuesNichtwissenBlatt()
179                 knoten.neuesKind(wert, blattFuerWert)
180             fehler = True
181

```



```

182         # keine weiteren Merkmale
183     else:
184         knoten.setzeKlasse(klasse_des_pakets)
185         fehler = True
186     print("Runde: ",durchlauf)
187     print(baum)
188
189 print("fertig gelernter Baum")
190 print("-----")
191 print(baum)
192
193 richtig=0
194 falsch=0
195 # Schleife über alle Testbeispiele
196 print("Entscheidung")
197 for paket in test_pakete:
198     # Beginne oben im Baum
199     knoten = baum
200     tiefe = 0
201
202     # Steige anhand der Merkmale des Pakets
203     # den Baum bis zu einem Blatt hinab
204     while not knoten.istEinBlatt():
205         knoten = knoten.kindFuerWert(paket[tiefe])
206         tiefe = tiefe + 1
207
208     # Entscheidung
209     if knoten.istNichtwissenBlatt():
210         geschenk_erwartet = "*"
211     else:
212         geschenk_erwartet = knoten.welcheKlasse()
213
214     print("für Paket",paket," : ",geschenk_erwartet,end="")
215     if geschenk_erwartet == paket[2]:
216         print(" ---> richtig")
217         richtig=richtig+1
218     else:
219         print(" ---> falsch")
220         falsch=falsch+1
221 print("Akkuratheit: ",richtig/(richtig+falsch))
222

```

fertig gelernter Baum

Helligkeit:

```

|-sehr dunkel: Größe:
|     |-eher klein: nein
|     |-mittel: nein
|     |-eher groß: ja
|-mittel: Größe:
|     |-eher klein: nein
|     |-mittel: ja
|     |-eher groß: (*)
|-eher hell: ja

```

Entscheidung

```

für Paket ('sehr dunkel', 'eher groß', 'ja') : ja ---> richtig
für Paket ('mittel', 'eher klein', 'nein') : nein ---> richtig
für Paket ('eher hell', 'mittel', 'ja') : ja ---> richtig
für Paket ('sehr dunkel', 'eher groß', 'ja') : ja ---> richtig
für Paket ('mittel', 'eher klein', 'nein') : nein ---> richtig
für Paket ('sehr dunkel', 'mittel', 'nein') : nein ---> richtig
für Paket ('mittel', 'eher klein', 'nein') : nein ---> richtig
für Paket ('sehr dunkel', 'eher groß', 'ja') : ja ---> richtig
für Paket ('sehr dunkel', 'eher klein', 'nein') : nein ---> richtig

```

```
für Paket ('eher hell', 'eher klein', 'ja') : ja ---> richtig
für Paket ('mittel', 'eher klein', 'nein') : nein ---> richtig
für Paket ('mittel', 'mittel', 'ja') : ja ---> richtig
für Paket ('sehr dunkel', 'eher klein', 'nein') : nein ---> richtig
Akkuratheit: 1.0
```

Aufgaben:

- 1.
- 2.
- 3.

für die gehobene Anspruchsebene:

x. Bauen Sie das Programm so um, dass die bestmöglichen Grenzen für die Merkmale gefunden werden können! Zur Bewertung der ermittelten Bäume reichen hier 3 Durchläufe für eine Kombination von Grenz-Werten! Stellen Sie sich auf längere Programm-Laufzeiten ein, wenn dann zwei-dimensional gesucht wird.

Aufgaben (Nutzung für den Titanik-Datensatz):

1. *Erweitern Sie das Programm auf mehr Merkmale!*
2. *Programmieren Sie die Daten des Titanik-Datensatz in das Programm ein! (Eine Daten-Bereinigung können Sie nach eigenem Ermessen durchführen.)*
3. *Experimentieren Sie mit verschiedenen Kategorisierungen! (üblich sind immer nur 2 Kategorien pro Merkmal)*
4. *Testen Sie mit den typischen imaginären Passagieren:*

Passagier	bekannte Merkmale

1
2
3
4
5

2.x. Arbeiten mit ausgewählten Bibliotheken in Python

Für das sinnvolle und vor allem auch standardisierte Arbeiten mit KNN's wurden verschiedene Bibliotheken entwickelt. Natürlich könnte man auch alles selbst programmieren, aber Aufwand und Nutzen stehen dabei in keinem sinnvollen Verhältnis.

Besonders sind es die Bibliotheken NumPy, MathPlotLib und für die Arbeit gebraucht werden.

Sie sind Bestandteil von Python selbst, sowie von der erweiterten Distribution Anaconda.

Die Bibliotheken sind meist in anderen Programmiersprachen als Python geschrieben. Dadurch sind sie deutlich schneller. Gerade bei großen Daten-Mengen macht sich der Geschwindigkeits-Vorteil deutlich bemerkbar. Für den normalen Nutzer sind die Interna aber unsichtbar. Wir haben nur eine definierte Schnittstelle, die im Wesentlichen aus vordefinierten Funktionen bestehen.

Die nachfolgenden Beispiele sind direkt mit der klassischen Python-IDE IDLE gemacht. Praktisch kann man die gleichen Ein- und Ausgaben auch in einem jupyter-Notebook realisieren. Zu dieser Möglichkeit kommen wir in diesem Skript etwas später (→). Wer mit diesem System sowieso schon arbeitet, kann es jetzt natürlich auch gleich benutzen.

2.x.y. Kurzvorstellung der Bibliothek NumPy

Die Bibliothek wurde vor allem zur Unterstützung des wissenschaftlichen Rechnens in Python aufgenommen. Es werden Matrizen und Matrizen-Operationen, Operationen der linearen Algebra, der FOURIER-Transformation und viele statistische Funktionen bereitgestellt.

Die besondere Bedeutung liegt aber im Bereich der n-dimensionalen Felder, die auch Array's genannt werden.

Neben diesen Haupt-Funktionen sind weitere "Neben-" oder Hilfs-Funktionen implementiert, die dem effektiven Umgang - vor allem mit Matrizen - dienen. Dazu gehören z.B. Sortier-Verfahren.

NumPy wird von vielen anderen Bibliotheken als Basis für die eigene Funktionalität benutzt.

Der Import kann in den verschiedenen klassischen Formen erfolgen.

```
import numpy
from numpy import ...
import ... from numpy
```

Hier benutzen wir die Form mit Alias-Definition, um den Schreibaufwand in Grenzen zu halten und trotzdem die exakte Funktions-Zuordnung zu garantieren.

```
import numpy as np
```

Der Alias kann dann für die Punkt-Schreibweise genutzt werden. In der folgenden Besprechung von NumPy gehe ich von einem Import in dieser Form aus. Die Funktionen von NumPy werden damit dann immer über np.??? angesprochen.

Exkurs: Installation von NumPy und verwandten Bibliotheken

Leider ist NumPy keine vorinstallierte, interne Bibliothek von Python. Sie ist Bestandteil vieler wissenschaftlicher Distributionen (Zusammenstellungen), wie z.B. Anaconda. Hier wird die Installation und Einrichtung durch ein eigenes Management-System übernommen.

Die klassische Installation von Bibliotheken erfolgt über den Python-eigenen Bibliothek's-Manager pip.

Mit ihm lassen sich die einzelnen Bibliotheken installieren.

Der Aufruf muss auf der Konsole `python -m pip install numpy` (Eingabeaufforderung) erfolgen.

Man sollte sich mit dem `cd`-Befehl vorher in das Verzeichnis begeben, in dem die `python.exe` abgelegt ist.

U.U. muss das pip-System vorher aktualisiert werden.

```
python - m pip istanll --upgrade pip
```

Die wichtigsten wissenschaftlichen bibliotheken lassen sich auch in einem Rutsch installieren.

```
python -m pip install --user numpy scipy↵  
matplotlib ipython jupyter pandas sympy↵  
nose
```

Für Windows gibt es auch vorbereitete Skripte, die die Installation automatisieren.

(→ <https://www.lfd.uci.edu/~gohlke/pythonlibs/>)

```
python -m pip install --user numpy scipy matplotlib ipython jupyter pandas  
sympy nose
```

2.x.y.0. Wiederholung Matrizen und Matrizen-Operationen

Matrizen sind in der Mathematik ein- oder mehr-dimensionale Tabellen mit Objekten - meist Zahlen. Sie werden in großen eckigen Klammern notiert.

Eindimensionale Matrixen werden auch als Vektoren bezeichnet. Jeder Objekt-Eintrag stellt dabei eine Komponente des Vektor's dar. Ob die Anordnung vertikal oder horizontal erfolgt ist zuerst einmal unbedeutend.

$$\begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} \quad \text{od. auch} \quad [2 \quad -3 \quad 1]$$

Die zwei-dimensionale Matrix ist sicher der typische Fall - vor allem in der Schule bzw. bei Erklärungen. Praktisch können Matrizen aber beliebig viele Dimensionen haben. Die Darstellung auf Papier stößt dann aber schnell an Grenzen.

$$\begin{bmatrix} 2 & -1 \\ -3 & 2 \\ 1 & 2 \end{bmatrix}$$

Recht ähnlich zur Matrize ist das Verständnis eines Tensor's z.B. in TensorFlow (→).

2.x.y.I. Definieren und Ausgeben von Matrizen in Python - NumPy

In NumPy werden Matrizen über den Daten-Typ array definiert. Das übernimmt die array-Funktion.

Sie erwartet Listen (- die den Zeilen entsprechen -) von Listen. Die letztgenannten Listen sind die Einträge / Elemente in der Zeile.

```
matrix1=np.array([[2,-1],
                  [-3,2],
                  [1,2]])
```

Die Zeilen-betonte Schreibung sollte unbedingt benutzt werden, um Daten besser zuzuordnen, Fehler schneller finden und zu können. Die Übersichtlichkeit steigt ebenfalls entscheidend. Die grundlegende Datenstruktur Liste bleibt also nach außen erhalten und kann ganz klassisch für Eingaben, Verarbeitungen und Ausgaben benutzt werden.

Ein-dimensionale Matrizen können vertikal als ein-elementige Listen in einer Liste oder horizontal als eine Liste eingegeben werden.

```
vektor_vertik= np.array([2],[-3],[1])
vektor_horiz= np.array([2,-3,1])
```

In NumPy sind einige praktische Ausfüll-Hilfen als Funktionen verfügbar.

So lassen sich horizontale Vektoren mit einer festen Schrittweite für die Elemente mit der Funktion **arange()** erzeugen.

```
vektor1=np.arange(0,16,3)
print(vektor1)
```

```
[0, 3, 6, 9, 12, 15]
```

Die Funktion erwartet als Argumente den Startwert, den nicht erreichbaren oberen Grenzwert sowie die Schrittweite.

Hier sehen wir jetzt auch, dass das Ausgeben gar kein Problem ist. Wir bekommen die Matrizen in der Listen-Schreibweise ganz Python-typisch angezeigt.

Eine lineare Reihe von Fließkomma-Werten eines Intervall's erhält man mit **linspace()**.

```
vektor2=np.linspace(0, 2, 6)
print(vektor2)
```

```
[0. 0.4 0.8 1.2 1.6 2. ]
```

Die Funktion benötigt die untere und obere Grenze des Intervall's sowie die Anzahl der zu erzeugenden Elementen.

Für viele Zwecke benötigt man mit Nullen bzw. Einsen gefüllte Matrizen.

Die Funktionen **zeros()** und **ones()** erwarten die Zeilen- (Reihen) und Spalten-Anzahl (der Matrix) als Tupel.

```
matrix_0=np.zeros((2,4))
print(matrix_0)
print()
```

```
matrix_1=np.ones((3,2))
print(matrix_1)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

```
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

Eine Identitäts-Matrix (Einheits-Matrix) - also eine Null-Matrix mit einer mit Einsen besetzten Diagonale erzeugt die Funktion **identity()**. Sie braucht als Argument nur die Ausdehnung (Zeilen- und Spalten-Anzahl, die hier ja gleich sind!)

```
matrix_ident=np.identity(4)
print(matrix_ident)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Mit zufälligen Werten belegte Matrizen brauchen wir z.B. für die Start-Konfiguration unserer KNN's. Dort werden ja z.B. die Gewichte zu Anfang zufällig gewählt.

Echt zufällige Zahlen erhalten wir mit der **random.randn()** aus NumPy.

```
matrix_RndN=np.random.randn(2,3)
print(matrix_RndN)
```

Sollen die Werte in einem anderen Zahlen-Bereich als die üblichen 0 bis 1 sein und es werden auch Ganzzahlen gebraucht, dann hilft **random.randint()**.

Diese Funktion erwartet die untere und obere (nicht-inklusive) Grenze, sowie die Anzahl der zu erzeugenden Zahlen.

```
liste_RndInt=np.random.randint(2,7,5)
print(liste_RndInt)
```

```
[2 2 5 4 5]
```

Eine normal-verteilte Werte-Matrix liefert uns **random.normal()**.

Sie braucht als Argumente den erwarteten Mittelwert (hier 5), die Varianz (hier: 4) und die Anzahl der benötigten Werte (hier 10).

```
liste_RndNrml=np.random.normal(5,4,10)
print(liste_RndNrml)
```

```
[ 3.89044719  3.58096408  4.66903407
  2.49199729  4.82472732  3.09112788
 -0.25545901  8.53848952  8.52527217
 11.83829225]
```

Um vergleichbare / nachvollziehbare Zufalls-Zahlen kann durch die Funktion **random.seed()** erreichen.

Sie bekommt einen Start-Wert mit. Die nun ab jetzt gelieferten Zufalls-Zahlen sind mit dem gleichen Start-Wert wieder zu erreichen.

```
np.random.seed(3)
listeS=np.random.randint(2,7,5)
print("geseedete Liste: ",listeS)
listeN=np.random.randint(2,7,5)
print("neue Liste      :",listeN)
np.random.seed(3)
listeT=np.random.randint(2,7,5)
print("neu geseed. Liste:",listeT)
```

```
geseedete Liste:  [4 2 3 5 2]
neue Liste      :  [2 2 5 4 5]
neu geseed. Liste: [4 2 3 5 2]
```

Den Daten-Typ eines Array / einer Matrix kann man ebenfalls anpassen. Numpy stellt uns dafür **astype()** zur Verfügung.

Erwartet wird von dieser Funktion ein Daten-Type - z.B.: float.

```
matrixFloat=matrix1.astype(float)
print(matrixFloat)
```

2.x.y.2. Berechnungen / Operationen mit Matrizen in Python - NumPy

Der lesende und schreibende Zugriff auf einzelne Element in der Matrix erfolgen über die Array-Schreibweise. Hier wird einfach direkt an die Variable ein Paar eckige Klammern notiert, in der die Position des Element's notiert wird.

Für die nachfolgenden Betrachtungen benutzen wir die im vorherigen Kapitel definierten Matrixen.

Der vektor1 besteht aus 6 Elementen, deren Nummerierung bei Null beginnt. Somit ist das letzte Element mit dem Index 5 zu erreichen.

Für mehr-dimensionale Matrixen ist die Position in allen Dimensionen anzugeben.

Die einzelnen Dimensionen werden nun Komma-getrennt notiert.

Betrachten wir eine definierte matrix.

Durch die Angabe der Indizes erhalten wir das gewünschte 3. Zeile (Index = 2) und der 4. Spalte (Index = 3).

```
[0, 3, 6, 9, 12, 15]
Nr.  0  1  2  3  4  5
print(vektor1[5])
15
```

```
0  [[ 1  2  3  4]
1  [ 5  6  7  8]
2  [ 9 10 11 12]
3  [13 14 15 16]]
Nr.  0  1  2  3
print(matrix[2,3])
12
```

Aus der Listen- und String-Verarbeitung kommt das Slicing. Hier steht in den eckigen Klammern noch ein Doppelpunkt. links und rechts können nun Grenzen angegeben werden. Dadurch werden jeweils Teil-Listen erzeugt. Zu beachten ist hier auch, dass die Zählungen immer bei Null beginnen.

Der Doppelpunkt kann als **von / bis** gelesen werden. Den gesamten Vektor erhält man, wenn man vom Nullten bis zum letzten (hier dem imaginären 6.) Element auswählt.

Auch hier wird - wie in Python üblich - das letzte Element (mit **bis**) nicht erreicht.

Kleinere Teil-Listen (Bereiche von Vektoren / Matrixen) erhält man durch die passende Auswahl der Grenzen.

Gibt man nur einen bis-Wert an, dann bekommt man nur die Teil-Liste vor dem bis-Wert.

Die Angabe einer Start-Position ohne ein Ende liefert die restliche Liste.

Um die gesamte Liste zu erhalten, kann man nun die beiden Grenzen einfach weglassen und nur den Doppelpunkt schreiben. Das gleiche erreicht man aber auch durch Weglassen des gesamten Klammer-Ausdruck's.

Interessant ist das letzte Beispiel mit der Angabe der Position -1. Hiermit erreicht man das letzte Element.

Also, selbst negative Zahlen sind als Slicing-Grenzen zulässig. Nun muss man aber schon genau aufpassen, um zu erfassen, was hier jeweils gemeint ist. Die negativen Indize sind genausogut für das Ausschneiden von Teil-Listen ge-

```
print(vektor1[0:6])
[0, 3, 6, 9, 12, 15]

print(vektor1[1:3])
[3 6]

print(vektor1[:2])
[0 3]

print(vektor1[2:])
[ 6  9 12 15]

print(vektor1[-1])
15
```

```
[0, 3, 6, 9, 12, 15]
Nr.  0  1  2  3  4  5
    -6 -5 -4 -3 -2 -1
print(vektor1[-3:-1])
[ 9 12]

print(vektor1[-1:-3])
```

eignet. Verwechselt man aber die beiden Werte, dann erhält man ev. eine leere Liste.

```
[]
```

Das Gegenstück zum Zerstückeln sind Verkettungen mehrere Vektoren / Matrizen zu einem Gesamt-Vektor / einer Gesamt-Matrix.

Mittels **append()** lässt sich die Verkettung durchführen. Auch die Funktion **concatenate()** erledigt diese Aufgabe.

Häufig benötigt man das größte und / oder das kleinste Wert einer Matrix. NumPy stellt hierfür (die Such-Funktionen) **max()** und **min()** bereit.

```
vektor3=np.append(vektor1, vektor_horiz)
print(vektor3)
```

```
mini=np.min(matrix1)
maxi=np.max(matrix1)
print("Werte zwischen",mini,"und",maxi)
```

```
Werte zwischen -3 und 2
```

Braucht man die Minima oder Maxima auf einer Achse - also Zeile oder Spalte, dann nutzen wir **min()** bzw. **max()** mit dem **axis**-Parameter. **axis=0** definiert das Spalten-weise Bearbeiten, mit **axis=1** wird Zeilen-weise ausgewertet. Für die **max**-Funktion kann äquivalent verfahren werden.

```
print(matrix1)
print()
miniZ=np.min(matrix1, axis=0)
print(miniZ)
miniZ=np.min(matrix1, axis=1)
print(miniZ)
```

```
[[ 2 -1]
 [-3  2]
 [ 1  2]]
```

```
[-3 -1]
[-1 -3  1]
```

Aber wo sind denn nun die Maxima oder die Minima in jeder einzelnen Zeile / Spalte?

Mit **argmax()** bzw. **argmin()** erhalten wir z.B. eine Index-Liste zu den gesuchten Maxima.

```
pos_Maxis=np.argmax(matrix1,axis=0)
print(pos_Maxis)
pos_Maxis=np.argmax(matrix1,axis=1)
print(pos_Maxis)
```

```
[0 1]
[0 1 1]
```

Oft werden bestimmte Eigenschaften von Matrizen für weitere Operationen usw. gebraucht.

Als Beispiel verwenden wir hier wieder die **matrix1** aus dem "Eingabe"-Kapitel. Über **.size** erhalten wir die Gesamt-Anzahl der Elemente in einer Matrix. Die Eigenschaft **.shape** liefert ein Tupel mit den Ausdehnungen in den verschiedenen Dimensionen zurück.

```
print(matrix1)
```

```
[[ 2 -1],
 [-3  2],
 [ 1  2]]
```

```
print(matrix1.size)
```

```
6
```

```
print(matrix1.shape)
```

```
(3, 2)
```

Nun zu ersten Operationen mit Matrizen. Diese müssen nicht selten in eine andere Form gebracht werden.

So könnten wir unsere 3x2-Matrix **matrix1** als Vektor brauchen. Dies entspricht z.B. einer 1x6-Matrix.

Alternativ ist es möglich, statt der Matrix-Größe (hier 6) die etwas kryptische Schreibung mittels negativen Index ver-

```
vektor3=matrix1.reshape(1, 6)
print(vektor3)
```

```
[[ 2 -1 -3  2  1  2]]
```

```
vektor3=matrix1.reshape(1, -1)
```


wenden.

Die Umwandlung in eine Liste innerhalb einer Liste ist allerdings nicht immer weiterverwendbar. Da hilft uns das Flatten.

Nun wird die Quell-Matrix in einen echten ein-dimensionalen Vektor gewandelt.

```
vektor3=matrix1.flatten()
print(vektor3)
```

```
[ 2 -1 -3  2  1  2]
```

Eine echte Matrizen-Operation ist die Transponierung. Sie wird auch Spiegelung genannt. Praktisch handelt es sich um das Tauschen der Dimensionen.

```
matrix1_transp=matrix1.T
print(matrix1_transp)
```

```
[[ 2 -3  1]
 [-1  2  2]]
```

Die Addition von zwei Matrizen setzt voraus, dass beide die gleiche Form und Größe haben. Ev. müssen anders-gestaltete Matrizen umgeformt (reshaped) werden.

Mittels +-Zeichen erhalten wir die Element-weise Addition der verknüpften Matrizen zurück.

```
matrixSum=matrix1 + matrix1
print(matrixSum)
```

```
[[ 4 -2]
 [-6  4]
 [ 2  4]]
```

Natürlich müssen die Operanden (hier matrix1 und matrix1) nicht gleich sein.

Ich verwende dies hier nur um möglichst wenig Daten zu verwenden und die Ergebnisse bei den folgenden Operationen so Effekt-voll sind.

Die Element-weise Subtraktion, Multiplikation und Division laufen genauso.

Wenn alles richtig eingegeben / programmiert wurde, dann ergibt die Subtraktion eine Null-Matrix, die Multiplikation die Quadrate und die Division eine Eins-Matrix.

```
print(matrix1 - matrix1)
print()
print(matrix1 * matrix1)
print()
print(matrix1 / matrix1)
```

```
[[0 0]
 [0 0]
 [0 0]]
```

```
[[4 1]
 [9 4]
 [1 4]]
```

```
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

Etwas anderes als die Element-weise Multiplikation funktioniert die Matrizen-Multiplikation (auch genannt). Bedingung hier ist, dass die Anzahl der Reihen der einen Matrix mit der Anzahl der Spalten der 2. Matrix übereinstimmen muss.

Die beiden anderen Dimensionen sind unabhängig voneinander. Ev. müssen anders-gestaltete Matrizen umgeformt (reshaped) werden.

Die **dot()**-Funktion übernimmt die Matrizen-Multiplikation und erwartet die beiden Matrizen als Argumente.

```
matrixMult=np.dot(matrix1,matrix1_transp)
print(matrixMult)
```

```
[[ 5 -8  0]
 [-8 13  1]
 [ 0  1  5]]
```

Will man nun irgendeine Aufgabe / Funktion mit einer Matrix erledigen, dann müssten wir eine Schleife programmieren, die z.B. alle Elemente durchläuft und die gewünschte Funktion ausführt. Das wäre sehr aufwändig und würden den Vorteilen, die wir durch die NumPy-Bibliothek erreichen wollen, entgegen wirken.

Hier hilft ein moderner Konstrukt in Programmier.Sprachen, der in Python **map**-Funktion heißt. Dabei geht es um das automatische Durchlaufen (Iterieren) aller Elemente einer Datenstruktur und dem anwenden einer Aufgabe. Dazu gleich genauer.

Vielfach benutzt man sogenannte anonyme Funktionen bzw. Lambda-Operatoren für die interne Funktions-Arbeit (innerhalb der map-Schleife).

Wir definieren uns zuerst eine lamda-Funktion. Hinter dem Namen der neuen Funktion folgt das Schlüsselwort **lambda** und eine Aufzählung der Variablen, denen später die einzelnen Elemente der Matrix zugeordnet werden sollen. Mit einem Doppelpunkt wird der Kopfteil beendet und es folgt nun direkt die Defintion der in der Funktion auszuführenden Operationen.

```
meineLambaOp = lambda i: i*10+1  
  
# Quelltext funktioniert nicht!!!  
  
vektorierteFkt=np.vectorize(meineLambdaOp)  
matrix_Erg=vektorierteKft(matrix1)  
print(matrix_Erg)
```

2.x.y. Kurzvorstellung der Bibliothek Matplotlib

Tabellen und andere Text-basierte Daten-Präsentationen sind schwer zu lesen. Diagramme und andere graphische Präsentationen sind viel angenehmer. Die Bibliothek Matplotlib bietet diverse Hilfsmittel, um Daten graphisch zu präsentieren / visualisieren. Die Ergebnisse werden i.A. in einem kleinen Fenster angezeigt. Wer mit Jupyter.Notebook's arbeitet (→), der kann auch eine Darstellung innerhalb des Notebook's erzeugen. Das ergibt dann ein zusammenhängendes Dokument. Im Notebook muss dazu einmalig am anfang das Magic-Kommando %matplotlib inline eingegeben werden. In Jupyter-Notebook's kann man sich dann sogar das show-Kommando sparen.

Vor dem Benutzen steht wieder ein Import an.

```
import matplotlib.pyplot as mpl
```

Wir nutzen hier die Möglichkeit des Import's über einen Alias-Namen - hier: **mpl**, den wir auch durchgehend in diesem Kapitel verwenden.

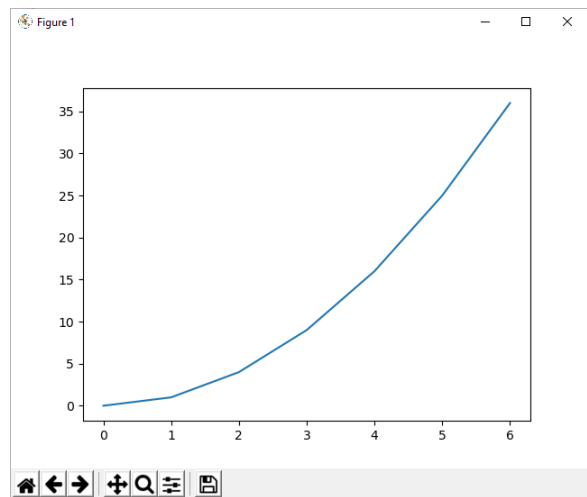
Das Erstellen einer Graphik ist praktisch aus zwei Abschnitten zusammengesetzt. Zuerst erstellen wir ein Graphik-Objekt - das i.A. Plot genannt wird. Dieses Objekt kann dann auch noch angepasst / formatiert werden.

```
mpl.plot(0,1,4,9,16,25)
```

```
mpl.show()
```

Der zweite Schritt ist die Darstellung des Objekt's mit der **show()**-Funktion.

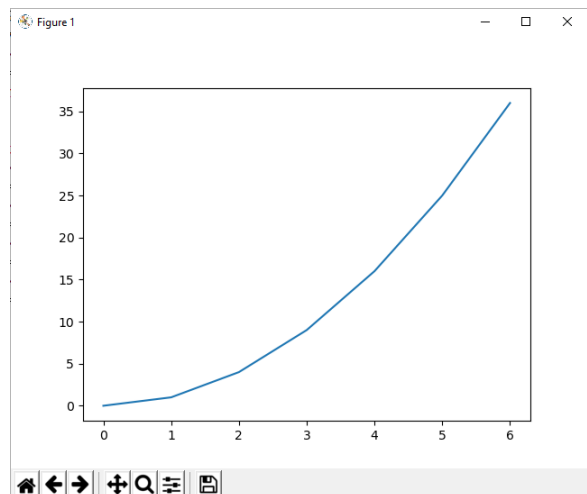
Alternativ kann ein Plot auch gespeichert oder als Bild-Datei exportiert werden. Wer ein wenig mit der Daten-Liste herumspielt, wird schnell merken, dass die x-Achse sehr variabel ist.



Da wir hierfür keine Daten (da nur eine Liste) übergeben haben, werden diese nur als y-Werte interpretiert. Die x-Werte erfindet matplotlib dazu.

Normalerweise stellen wir ja auch Daten-Punkte dar. Ihre Werte werden in 2 Listen - eine für die x-Werte und eine für die y-Werte übergeben. Nun erhalten wir auch stabile Darstellungen.

```
mpl.plot([0,1,2,3,4,5,6],[0,1,4,9,16,25,36])  
mpl.show()
```

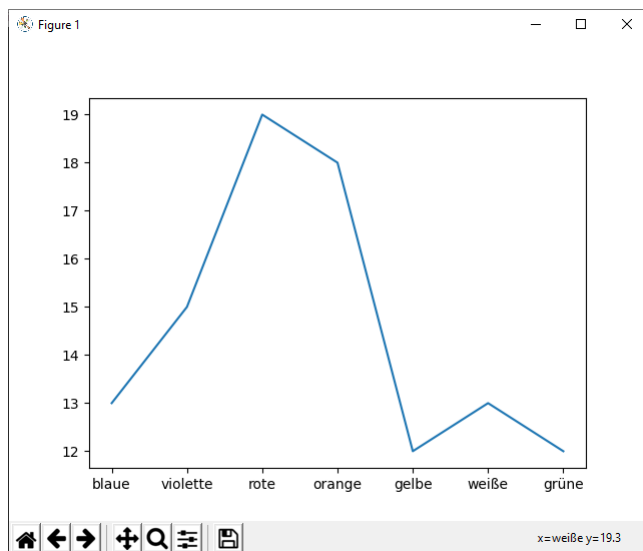


Statt mit Listen kann man auch mit einem Dictionary (Wörterbuch) arbeiten. Dabei können wir nun auch der x-Achse auch Texte (Key's) zuordnen. Die Value's werden die y-Werte.

Das Wörterbuch muss zuerst als solches ordentlich definiert werden.

In die Plot-Funktion übergeben wir dann frisch gebildete Listen (mit der list()-Funktion) aus den Schlüsseln (Key's) und den Werten (Value's).

```
daten={'blaue': 13,
      'violette': 15,
      'rote': 19,
      'orange': 18,
      'gelbe': 12,
      'weiße': 13,
      'grüne': 12}
mpl.plot(list(daten.keys()),
        list(daten.values()))
mpl.show()
```

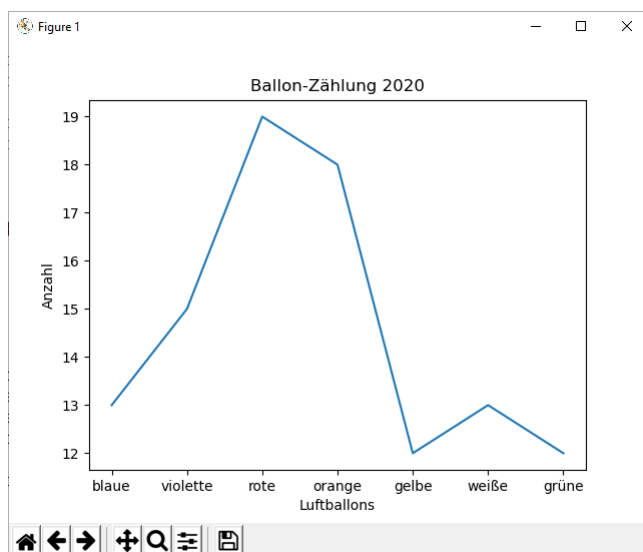


Die Diagramme sind natürlich nur mit ordentlichen Beschriftungen Präsentations-tauglich.

Diese Formatierungen finden vor dem Aufrufen der show()-Funktion statt. Ansonsten sind sie nicht sichtbar oder erst nach einem erneuten show zu sehen.

Einen Diagramm-Titel kann man über die **title()**-Funktion definieren. Für die Achsen-Beschriftung nutzt man **xlabel()** und **ylabel()**.

```
...
mpl.xlabel("Luftballons")
mpl.ylabel("Anzahl")
mpl.title("Ballon-Zählung 2020")
mpl.show()
```



Mehrere Funktionen können vor dem show kombiniert werden. Für jede Funktion / für jeden Graph erstellen wir einen Plot und lassen diese dann gemeinsam anzeigen. Die Legende wird mit legend() hinzugefügt und zeigt unsere Labels an.

```
listeX=[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0]
mpl.plot(listeX, listeX*2, label="y=2x")
mpl.plot(listeX, listeX**2, label="y=x²")
mpl.plot(listeX, listeX**2*2, label="y=2x²")
mpl.legend()
mpl.show()
```

```
meineFkt=lambda x: x**3
mpl.plot(listeX, meineFkt(listeX),
         label="y=x³")
mpl.legend()
mpl.show()
```

Bleibt die Frage nach anderen Diagramm-Arten. Natürlich bietet uns so ein professionelles Modul, wie Matplotlib, eine große Vielfalt an Diagramm-Arten

Formatierungen der Linien usw. lassen sich weitestgehend mit einem Zusatz-String vornehmen. Dieser wird mit den Daten-Listen als weiteres Argument mit an die plot()-Funktion übergeben.

```
mpl.
mpl.show()
```

Matplotlib bietet auch die Möglichkeit mehrere Diagramme zu einer Figur zusammenzustellen.

Bei subplots müssen wir die Zeilen und Spalten angeben und auch eine (Figur-)Gesamtgröße.

```
figur1=mpl.subplots(1,2,figsize(19,5),
                    sharey=True)
plots[0].bar()
plots[0].title("")
plots[1].matter()
plots[1].title("")
figur1.suptitle("")
mpl.show()
```

Will man nun auch noch Bilder od.ä. mit in Diagramme integrieren, dann hilft uns Matplotlib hier ebenfalls.

Bilder aus Internet-Quellen können z.B. mit der nebenstehenden Struktur bereitstellen.

```
from urllib.request import urlretrieve
import tempfile

url="..."

with tempfile.NamedTemporaryFile as file:
```

```
urlretrieve(url, file.name)
bild=mpl.imread(file.name)
mpl.axis('off')
imgplot=mpl.imshow(bild)
```

```
mpl.
mpl.show()
```

```
mpl.
mpl.show()
```

```
mpl.
mpl.show()
```

```


```

```


```

2.x. entdeckenes / spielendes Lernen mit Python

2.x.y. Tic-Tac-Toe mit Python

Arbeitsweise des folgenden Programm's schwer zu durchschauen
einige Hinweise gibt die original-Quelle
mehr Programm zum Benutzen, als zum Verstehen

```
# Original aus SCHMID, WEITZ, SIEBERS:
# Künstliche Intelligenz selber programmieren für DUMMIES
# ISBN 978-3-527-71573-2
1 import random
2
3 class Spielstand():
4     winning_lines = [(0, 1, 2),
5                     (3, 4, 5),
6                     (6, 7, 8),
7                     (0, 3, 6),
8                     (1, 4, 7),
9                     (2, 5, 8),
10                    (0, 4, 8),
11                    (2, 4, 6)]
12
13     vsep = " | "
14     hsep = "-"
15
16     alleFeldNummern = [1, 2, 3, 4, 5, 6, 7, 8, 9]
17
18     def __init__(self):
19         self.squares = [None for x in range(0, 9)]
20         self.next_player = "X"
21         self.gameOver = False
22         self.winner = None
23
24     def checkGameWon(self):
25         for loc1, loc2, loc3 in Spielstand.winning_lines:
26             if self.squares[loc1] is not None \
27                 and self.squares[loc2] is not None \
28                 and self.squares[loc3] is not None \
29                 and self.squares[loc1] == self.squares[loc2] \
30                 and self.squares[loc1] == self.squares[loc3]:
31                 self.winner = self.squares[loc1]
32                 self.gameOver = True
33                 return ()
34
35     def checkGameOver(self):
36         if not self.gameOver:
37             self.checkGameWon()
38
39         if not self.gameOver:
40             for loc in range(0, 9):
41                 if self.squares[loc] is None:
42                     return ()
43             self.gameOver = True
44
45     def __repr__(self, numbered=False):
46         return (self.__repr_square__(0, numbered) + self.vsep +
47                 self.__repr_square__(1, numbered) + self.vsep +
48                 self.__repr_square__(2, numbered) + "\n" +
49                 self.hsep * (3 + 2 * len(self.vsep)) + "\n" +
50                 self.__repr_square__(3, numbered) + self.vsep +
```

```

50         self.__repr_square__(4, numbered) + self.vsep +
51         self.__repr_square__(5, numbered) + "\n" +
52         self.hsep * (3 + 2 * len(self.vsep)) + "\n" +
53         self.__repr_square__(6, numbered) + self.vsep +
54         self.__repr_square__(7, numbered) + self.vsep +
55         self.__repr_square__(8, numbered))
56
57     def __repr_square__(self, loc, numbered):
58         if self.squares[loc] is None:
59             if numbered:
60                 return (str(self.alleFeldNummern[loc]))
61             else:
62                 return ("-")
63         else:
64             return (str(self.squares[loc]))
65
66     def besetzeFeld(self, feldNummer):
67         cloned = Spielstand()
68         cloned.squares = list(self.squares)
69         cloned.squares[feldNummer - 1] = self.next_player
70         if self.next_player == "X":
71             cloned.next_player = "O"
72         else:
73             cloned.next_player = "X"
74         cloned.checkGameOver()
75         return (cloned)
76
77     def freieFelder(self):
78         free = [nr for (nr, spieler) in zip(self.alleFeldNummern,
79 self.squares) if spieler is None]
80         random.shuffle(free)
81         return (free)
82
83     def spielBeendet(self):
84         return self.gameOver
85
86     def gewinner(self):
87         if self.gameOver:
88             if self.winner is None:
89                 return "Niemand"
90             else:
91                 return self.winner
92         else:
93             return "Niemand"
94
95     def O_hat_gewonnen(self):
96         return self.winner == "O"
97
98     def X_hat_gewonnen(self):
99         return self.winner == "X"
100
101     def unentschieden(self):
102         return self.winner is None
103
104     def zeigeAn(self):
105         print(self)
106         if self.gameOver:
107             if self.winner is None:
108                 print("Das Spiel ist unentschieden ausgegangen")
109             else:
110                 print("\nSpieler", self.winner, "hat gewonnen.")
111         else:
112             print("\nSpieler", str(self.next_player), "ist am Zug.")
113
114     def anfangszustand():
115         return (Spielstand())
116
117     def playKI(zustand, ki):
118         if ki is not None:
119             ki_wert, ki_choice = ki(zustand)
120             if ki_choice in zustand.freieFelder():
121                 return (zustand.besetzeFeld(ki_choice))

```



```

120     print("Irgendetwas ist schief gelaufen.", end=" ")
121     if ki_choice is None or ki_choice < 1 or ki_choice > 9:
122         print("Die KI konnte sich für kein Feld entscheiden.")
123     else:
124         print("Die KI hat ein bereits besetztes Feld gewählt.")
125     print("Ich probiere es einfach mal mit einem zufälligen Feld.")
126     return (zustand.besetzeFeld(random.choice(zustand.freieFelder())))
127
128 def starteSpiel(ki=None):
129     try:
130         playFirst, firstPlayer, firstText = random.choice(
131             [(True, "O", "Ich fange"),
132              (False, "X", "Du fängst")])
133
134         print("Schön, dass du eine Runde Tic-Tac-Toe spielen möchtest!")
135         print("Du spielst als \"X\", ich als \"O\".", firstText, "an!")
136
137         zustand = Spielstand()
138         zustand.next_player = firstPlayer
139
140         while not zustand.gameOver:
141             if zustand.next_player == "O":
142                 try:
143                     zustand = playKI(zustand, ki)
144                 except:
145                     print("Fehler innerhalb der KI! Spiel wird beendet.")
146                     return
147             else:
148                 print()
149                 print(zustand.__repr__(True))
150
151                 user_choice = None
152                 while user_choice is None:
153                     user_choice = input("Gib die Nummer des Feldes ein, das
154                     du besetzen möchtest: ")
155                     try:
156                         feld = int(user_choice)
157                         if feld < 1 or feld > 9:
158                             print("Es gibt kein Feld mit der Nummer ", feld,
159                                   ". Probiere es noch einmal!\n", sep="")
160                             user_choice = None
161                         elif not feld in zustand.freieFelder():
162                             print("Das Feld", feld, "ist nicht frei.",
163                                   "Probiere es noch einmal!\n")
164                             user_choice = None
165                     except ValueError:
166                         print("\"", user_choice,
167                               "\" ist keine Zahl. Probiere es noch ein-
168                               mal!\n",
169                               sep="")
170                         user_choice = None
171                     zustand = zustand.besetzeFeld(feld)
172
173                 if zustand.gewinner() == "O":
174                     print("Ich habe gewonnen!!!")
175                 elif zustand.gewinner() == "X":
176                     print("Mist, du hast gewonnen!")
177                 else:
178                     print("Unentschieden!")
179
180                 print(zustand)
181             except (KeyboardInterrupt, EOFError):
182                 print("\nSpiel wird beendet.")

```

Q: /15, S. 95f/ (aus online-Resource kopiert)

```

starteSpiel()
Schön, dass du eine Runde Tic-Tac-Toe spielen möchtest!

```

```

Du spielst als "X", ich als "O". Du fängst an!

1 | 2 | 3
-----
4 | 5 | 6
-----
7 | 8 | 9
Gib die Nummer des Feldes ein, das du besetzen möchtest: 5

1 | 2 | 3
-----
4 | X | 6
-----
7 | O | 9
Gib die Nummer des Feldes ein, das du besetzen möchtest: 7

1 | O | 3
-----
4 | X | 6
-----
X | O | 9
Gib die Nummer des Feldes ein, das du besetzen möchtest: 3
Mist, du hast gewonnen!
- | O | X
-----
- | X | -
-----
X | O | -

```

hier ist es als Beginner noch leicht den Computer zu schlagen, der Algorithmus nimmt keine Chancen-Bewertung für die einzelnen Züge vor nur möglicher Gewinn wird erkannt

Verwendung mit einem Optimierungs-Algorithmus
hier MiniMax

```

# Original aus SCHMID, WEITZ, SIEBERS:
# Künstliche Intelligenz selber programmieren für DUMMIES
# ISBN 978-3-527-71573-2
1 import tictactoe
2
3 def minimax(zustand):
4     gewonnen_mit = None
5     unentschieden_mit = None
6     verloren_mit = None
7
8     for feld in zustand.freieFelder():
9         naechsterZustand = zustand.besetzeFeld(feld)
10
11        if naechsterZustand.spielBeendet():
12            if naechsterZustand.O_hat_gewonnen():
13                ergebnis = "gewonnen"
14            elif naechsterZustand.unentschieden():
15                ergebnis = "unentschieden"
16            else:
17                ergebnis = "verloren"
18        else:
19            ergebnis = spieler_zieht(naechsterZustand)
20
21        if ergebnis == "gewonnen":
22            gewonnen_mit = feld
23        elif ergebnis == "unentschieden":
24            unentschieden_mit = feld
25        else:
26            verloren_mit = feld

```

```

27
28     if gewonnen_mit is not None:
29         return "gewonnen", gewonnen_mit
30     elif unentschieden_mit is not None:
31         return "unentschieden", unentschieden_mit
32     else:
33         return "verloren", verloren_mit
34
35
36 def spieler_zieht(zustand):
37     gewonnen_mit = None
38     unentschieden_mit = None
39     verloren_mit = None
40
41     for feld in zustand.freieFelder():
42         naechsterZustand = zustand.besetzeFeld(feld)
43
44         if naechsterZustand.spielBeendet():
45             if naechsterZustand.X_hat_gewonnen():
46                 ergebnis = "verloren"
47             elif naechsterZustand.unentschieden():
48                 ergebnis = "unentschieden"
49             else:
50                 ergebnis = "gewonnen"
51         else:
52             ergebnis, zug_computer = minimax(naechsterZustand)
53
54         if ergebnis == "verloren":
55             verloren_mit = feld
56         elif ergebnis == "unentschieden":
57             unentschieden_mit = feld
58         else:
59             gewonnen_mit = feld
60
61     if verloren_mit is not None:
62         return "verloren"
63     elif unentschieden_mit is not None:
64         return "unentschieden"
65     else:
66         return "gewonnen"
67

```

Q: /15, S. 98f/ (aus online-Resource kopiert)

```

tictaktoe.starteSpiel(minimax)
Schön, dass du eine Runde Tic-Tac-Toe spielen möchtest!
Du spielst als "X", ich als "O". Ich fange an!

1 | 2 | 3
-----
4 | O | 6
-----
7 | 8 | 9
Gib die Nummer des Feldes ein, das du besetzen möchtest: 3

1 | 2 | X
-----
4 | O | 6
-----
7 | O | 9
Gib die Nummer des Feldes ein, das du besetzen möchtest: 2

O | X | X
-----
4 | O | 6
-----
7 | O | 9
Gib die Nummer des Feldes ein, das du besetzen möchtest: 9

```

```

O | X | X
-----
4 | O | O
-----
7 | O | X
Gib die Nummer des Feldes ein, das du besetzen möchtest: 4
Unentschieden!
O | X | X
-----
X | O | O
-----
O | O | X

```

Nun hat man nur noch als Beginner eine Chance bestenfalls ein Unentschieden zu erreichen.

Eine andere Umsetzung aus einem Python-Tutorial – mit vielen guten Erklärungen und sinnvollen Entwicklungs-Schritten (→ <https://www.python-lernen.de/tic-tac-toe-python-tutorial.htm>) folgt hier als fertiger Code:

```

1 # Spiel Tic-Tac-Toe
2 print("Tic-Tac-Toe Python Tutorial")
3 spiel_aktiv = True
4 spieler_aktuell = 'X'
5 # Spielfeld als Liste erstellen
6 spielfeld = [" ",
7             "1","2","3",
8             "4","5","6",
9             "7","8","9"]
10
11 # Spielfeld ausgeben
12 def spielfeld_ausgeben():
13     print (spielfeld[1] + "|" + spielfeld[2] + "|" + spielfeld[3] )
14     print (spielfeld[4] + "|" + spielfeld[5] + "|" + spielfeld[6] )
15     print (spielfeld[7] + "|" + spielfeld[8] + "|" + spielfeld[9] )
16
17 # Spieleingabe und Kontrolle der Eingabe
18 def spieler_eingabe():
19     global spiel_aktiv
20     while True:
21         spielzug = input("Bitte Feld eingeben: ")
22         # vorzeitiges Spielende durch Spieler
23         if spielzug == 'q':
24             spiel_aktiv = False
25             return
26         try:
27             spielzug = int(spielzug)
28         except ValueError:
29             print("Bitte Zahl von 1 bis 9 eingeben")
30         else:
31             if spielzug >= 1 and spielzug <= 9:
32                 if spielfeld[spielzug] == 'X' or spielfeld[spielzug]
33 == 'O':
34                     print("Das Feld ist bereits belegt - ein anderes
35 wählen!")
36                 else:
37                     return spielzug
38             else:
39                 print("Zahl muss zwischen 1 und 9 liegen")
40
41 def spieler_wechseln():
42     global spieler_aktuell
43     if spieler_aktuell == 'X':
44         spieler_aktuell = 'O'
45

```

```

46     else:
47         spieler_aktuell = 'X'
48     # Kontrolle, ob ein Spieler gewonnen hat
49     def kontrolle_gewonnen():
50         # wenn alle 3 Felder gleich sind, hat der entsprechende Spieler
51         gewonnen
52         # Kontrolle auf Reihen
53         if spielfeld[1] == spielfeld[2] == spielfeld[3]:
54             return spielfeld[1]
55         if spielfeld[4] == spielfeld[5] == spielfeld[6]:
56             return spielfeld[4]
57         if spielfeld[7] == spielfeld[8] == spielfeld[9]:
58             return spielfeld[7]
59         # Kontrolle auf Spalten
60         if spielfeld[1] == spielfeld[4] == spielfeld[7]:
61             return spielfeld[1]
62         if spielfeld[2] == spielfeld[5] == spielfeld[8]:
63             return spielfeld[2]
64         if spielfeld[3] == spielfeld[6] == spielfeld[9]:
65             return spielfeld[3]
66         # Kontrolle auf Diagonalen
67         if spielfeld[1] == spielfeld[5] == spielfeld[9]:
68             return spielfeld[5]
69         if spielfeld[7] == spielfeld[5] == spielfeld[3]:
70             return spielfeld[5]
71
72     def kontrolle_auf_unentschieden():
73         if (spielfeld[1] == 'X' or spielfeld[1] == 'O') \
74             and (spielfeld[2] == 'X' or spielfeld[2] == 'O') \
75             and (spielfeld[3] == 'X' or spielfeld[3] == 'O') \
76             and (spielfeld[4] == 'X' or spielfeld[4] == 'O') \
77             and (spielfeld[5] == 'X' or spielfeld[5] == 'O') \
78             and (spielfeld[6] == 'X' or spielfeld[6] == 'O') \
79             and (spielfeld[7] == 'X' or spielfeld[7] == 'O') \
80             and (spielfeld[8] == 'X' or spielfeld[8] == 'O') \
81             and (spielfeld[9] == 'X' or spielfeld[9] == 'O'):
82             return ('unentschieden')
83
84     # aktuelles Spielfeld ausgeben
85     spielfeld_ausgeben()
86     while spiel_aktiv:
87         # Eingabe des aktiven Spielers
88         print()
89         print ("Spieler " + spieler_aktuell + " am Zug")
90         spielzug = spieler_eingabe()
91         if spielzug:
92             # spielfeld[spielzug] = 'X'
93             spielfeld[spielzug] = spieler_aktuell
94             # aktuelles Spielfeld ausgeben
95             spielfeld_ausgeben()
96             # Kontrolle, ob jemand gewonnen hat
97             gewonnen = kontrolle_gewonnen()
98             if gewonnen:
99                 print ("Spieler " + gewonnen + " hat gewonnen!")
100                spiel_aktiv = False
101                break
102             # Kontrolle, ob unentschieden
103             unentschieden = kontrolle_auf_unentschieden()
104             if unentschieden:
105                 print ("Spiel ist unentschieden ausgegangen")
106                 spiel_aktiv = False
107             # Spieler wechseln
108             spieler_wechseln()
109     print()

```

117

Q: <https://www.python-lernen.de/tic-tac-toe-python-tutorial.htm>

Vorschlag einer dummen KI als zweiter Spieler:

```
90 ...
91 while spiel_aktiv:
92     # Eingabe des aktiven Spielers
93     print()
94     print ("Spieler " + spieler_aktuell + " am Zug")
95
96
97     # aus der Liste spielfeld alle X und O und leere Felder entfernen
98     spielfeld_KI = []
99     for moegliche_felder in spielfeld:
100         if moegliche_felder != 'X' and moegliche_felder != 'O' and
moegliche_felder != ' ':
101             spielfeld_KI += moegliche_felder
102     # print (spielfeld_KI)
103     # print()
104     # print(random.choice(spielfeld_KI))
105
106     # wenn Computergegner am Zug ist, ein freies zufälliges Feld be-
legen
107
108     if spieler_aktuell == 'O':
109         spielzug = int(random.choice(spielfeld_KI))
110     else:
111         spielzug = spieler_eingabe()
112
```

Q: <https://www.python-lernen.de/tic-tac-toe-python-tutorial.htm>



weitere Umsetzungen (z.B.):

- <https://www.geeksforgeeks.org/python-implementation-automatic-tic-tac-toe-game-using-random-number/>
- <https://www.askpython.com/python/examples/tic-tac-toe-using-python>

Es geht natürlich auch graphisch (→ <https://realpython.com/tic-tac-toe-python/>). In diesem Fall wird die Bibliothek **tkinter** benutzt.


2.x. Wiederholung / Basic's zu JAVA

JAVA ist eine Allzweck-Waffe in der Programmierung

sehr gewaltige Sprache

hier nur absolute Grundlagen, um die aufgezeigten Programme zu verstehen

ausführlichere Beschreibungen füllen dicke Bücher

eine schon umfangreichere Beschreibung im Skript ( **JAVA**)

für klassische Kurse kann der JAVA-Teil übersprungen werden

mehr für Kurse mit vorhandenem JAVA-Bezügen oder zum Allgemeinverständnis interessant

in der KI-Praxis wird Python (→ [2.x. ausgewählte Wiederholungen zu Python](#) ff) bevorzugt

aufwändiges Klassen-Konzept der Objekt-orientierten Programmierung ist in JAVA zwingend und die Sprache ist z.T. etwas sperrig

jeder Klasse muss in eine extra JAVA-Datei

recht Schreib-aufwändige Programmierung (bei reiner textueller Programmierung)

Vorteile erst in Kombination mit UML-Techniken

für praktisch alle Hardware-Systeme verfügbar; Programme funktionieren zumeist auch auf allen möglichen Systemen (mit einer JAVA-Runtime-Umgebung)

Kommentare

für den Programm-Ablauf unwichtig und vom Übersetzer-Programm (Compiler) ignoriert
dienen der besseren Lesbarkeit des Quell-Textes für Menschen

1		
2	<code>// Kommentar</code>	einzeiliger Kommentar
3		
4	<code>x = a * b; //berechnet Produkt aus a und b</code>	Anweisung mit Kommentar (Rest der Zeile wird ignoriert)
5		
6		
7	<code>/* der folgende Quellcode ist überflüssig</code>	mehrzeiliger Kommentar (Beginn)
8		
9	<code>*/</code>	Ende des mehrzeiligen Kommentar's
10		

in guten Editoren (solchen mit Code-Highlighting) werden die verschiedenen Programm-Elemente von JAVA in jeweils anderen Farben (und ev. Text-Stilen) dargestellt (meist sehr individuell einstellbar)

Leistungs-fähige Editoren sind (z.B.):

- JavaEditor
- Notepad++
- Visual Studio Code
- Eclipse
- ...

Grund-Gerüst einer aufrufbaren / startbaren JAVA-Datei

```
1  
2  
3 class Name{  
4  
5     public static void main(String[] args){  
6  
7     }  
8 }  
9
```

Platz für **import** von anderen Klassen (Bibliotheken)
Name ist die Bezeichnung der Klasse und des Klassen-Typ's
Start des Haupt-Programm's (main)
eigentliches Haupt-Programm (hier leer)
Ende des Haupt-Programm'
Ende der Klasse

Leerzeilen dienen nur der übersichtlichen Strukturierung; viele könnten weggelassen werden

```
1 import Unterklasse;  
2  
3 class Name { ...  
4
```

Bereitstellung einer Unterklasse / Bibliothek
Klasse, die dann Teile der Unterklasse benutzt

Grund-Gerüst weiterer Klassen

```
1 class Name{  
2     //Variablen / Attribute  
3  
4     //Konstruktor  
5  
6     //Geter und Seter  
7  
8     //weitere Arbeits-Methoden  
9  
10 }  
11
```

Klassen-**Name** festlegen
... für die Erzeugung eines neuen Objekt's
... für die Kommunikation zwischen Objekt und nutzendem Programm
spezielle Methoden zur (internen) Verarbeitung der Daten; aber auch bereitgestellte Methoden für ein aufrufendes Programm

Variablen (Deklarationen, Zuweisungen)

Variablen zum Speichern von Werten müssen jeweils einen Typ haben, der vorher oder beim ersten Benutzen der Variable festgelegt (deklariert) werden muss
typisch Ganzzahlen (int), Fließkommazahlen (double), Texte (String) und Wahrheitswerte (boolean)
mit den Variablen werden auch die Attribute von Klassen realisiert

```
1 class {
2     public static void main(String[] args){
3
4     }
5 }
```

Ausgaben (auf der Konsole)

klassische Ausgabe mit den Methoden (Funktionen) System.out.println() und System.out.print()

```
1 class {
2     public static void main(String[] args){
3
4     }
5 }
```

Eingaben (von der Konsole)

schwieriges Thema in JAVA
erfordert mehr Aufwand, als in vielen anderen Programmiersprachen (wie z.B. Python)

```
1 class {
2     public static void main(String[] args){
3
4     }
5 }
```

Methoden / Funktionen

Methoden sind die Fähigkeiten innerhalb der Klassen
praktisch gleichbedeutend mit Funktionen
haben immer Namen und ein Klammer-Paar
zum eindeutigen Aufrufen von Methoden wird der Name mit dem vorgesetzten Klassennamen kombiniert (Punkt-Notierung)

Kommunikation zwischen den Klassen (Datenaustausch) über sogenannter Geter und Seter.
Das meint get()- und set()-Funktionen (Methoden), die das Liefern (Zurückgeben) bzw. Setzen (Übernehmen) von Werten übernehmen.

Innhalb der Klasse können die Daten in einem vom Programmierer festgelegten Daten-Typ benutzt werden. Dieser Typ ist für das nutzende Programm (übergeordnete Klassen) nicht wichtig und auch nicht sichtbar.

Methoden selbst haben auch einen Daten-Typ. Der wird durch die zurückgegebenen Daten bestimmt. Sollen keinen Daten zurückgegeben werden, dann ist der Typ void.

```
1 class {  
2     public static void main(String[] args){  
3  
4     }  
5 }
```

```
1 class {  
2     public static void main(String[] args){  
3  
4     }  
5 }
```

2.x. Perceptron in JAVA

diverse Realisierungen beschrieben
Grund-Prinzip aber immer gleich

2.x. Künstliches neuronales Netz in JAVA

hier werden einige Vorteile der Objekt-orientierten Programmierung deutlich

2.x. PROLOG – eine andere Form der Programmierung für die Wissens-Verarbeitung

Im klassischen Maschine Learning dominiert vor allem Python das Programmier-Geschehen. Das liegt vor allem am sehr einfachen Syntax, der kostenlosen Bereitstellung des Übersetzer-Programm's (Compiler/Interpreter), der Verfügbarkeit auf verschiedenen Rechner-Systemen und der großen Anzahl von vorprogrammierten Bibliotheken.

Eine andere Möglichkeit zum Programmieren im Bereich der KI ist PROLOG.

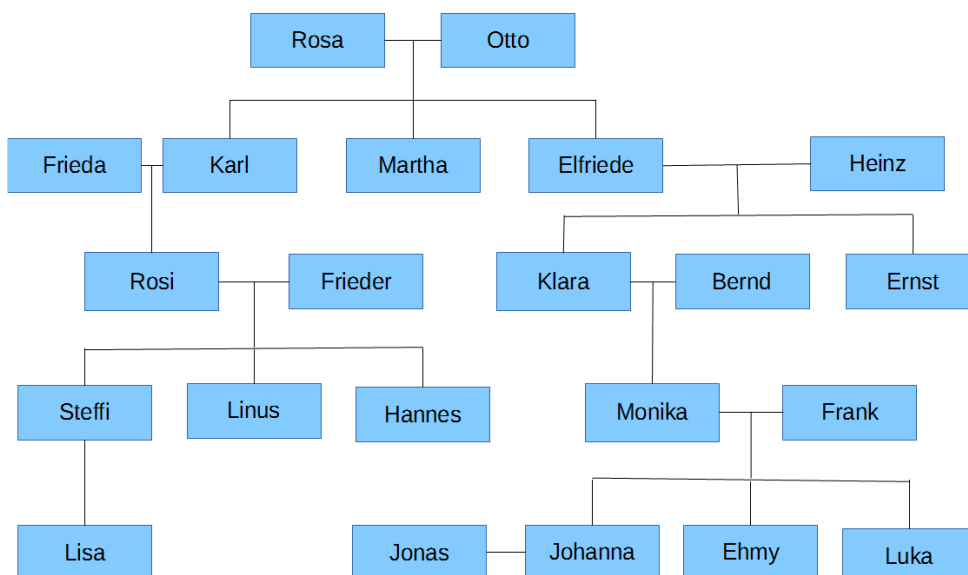
PROLOG ist eine logische Programmiersprache, in der sich semantische Zusammenhänge und logische Probleme sehr gut programmieren lassen.

In PROLOG werden Atome und Regeln unterschieden. Atome stellen das Basis-Wissen dar. Die Notierung erfolgt über ein Themen-Wort und in Klammern aufgezählte Eigenschaften / Attribute. Jeder Fakt wird mit einem Punkt abgeschlossen.

Regeln stellen Beziehungen zwischen den Atomen / Fakten her. Dabei sind Variablen für die Attribute, sowie logische Verknüpfungen, möglich.

Im folgenden Wissens-System geht es um die familiären Zusammenhänge.

Muster-Familienstammbaum



Quelltext	
<pre> % Autor: Drews % Datum: September 2015 weiblich(klara). maennlich(bernd). maennlich(ernst). weiblich(monika). maennlich(frank). maennlich(jonas). weiblich(johanna). weiblich(ehmy). maennlich(luka). verheiratet(klara, bernd). verheiratet(monika, frank). verheiratet(johanna, jonas). ist_elter_von(klara, monika). ist_elter_von(bernd, monika). ist_elter_von(monika, johanna). ist_elter_von(frank, johanna). ist_elter_von(monika, ehmy). ist_elter_von(frank, ehmy). ist_elter_von(monika, luka). ist_elter_von(frank, luka). ist_vater_von(X, Y):- maennlich(X), ist_elter_von(X, Y). ist_opa_von(X,Y):- maennlich(X), ist_elter_von(X, Z), ist_elter_von(Z, Y). ist_nachkomme_von(X,Y):- ist_elter_von(Y, X) ist_elter_von(Y, Z), ist_nachkomme_von(X, Z). </pre>	

Erläuterung der Atome und Regeln / Klauseln:

Quelltext(-Abschnitt)	Erläuterung
<pre> % Autor: Drews ... </pre>	Kommentar
<pre> weiblich(klara). maennlich(bernd). ... </pre>	Atom beschreibt, dass Klara weiblich ist genau für männlich
<pre> verheiratet(klara, bernd). ... </pre>	Atom beschreibt, dass die Personen verheiratet (od.ä.) sind
<pre> ist_elter_von(klara, monika). ist_elter_von(bernd, monika). ... </pre>	Atom beschreibt die Elternschaft (lies: Klara ist_elter_von Monika)
<pre> ist_vater_von(X, Y):- maennlich(X), ist_elter_von(X, Y). ist_opa_von(X,Y):- maennlich(X), ist_elter_von(X, Z), ist_elter_von(Z, Y). ist_nachkomme_von(X,Y):- ist_elter_von(Y, X) ist_elter_von(Y, Z), ist_nachkomme_von(X, Z). </pre>	<p>Regel zur Verwandtschaft "Vater": er (X) muss männlich sein UND (deshalb Komma) Elter von Y</p> <p>Regel für Nachkomme: ist ein Eltern-Teil ODER (deshalb senkrechter Strich) ein Eltern-Teil UND einer weiter zu prüfenden Nachkommenschaft</p>

Aufgaben:

- 1. Betrachten Sie zuerst einmal den Muster-Familienstammbaum!**
- 2. Beantworten Sie die folgenden Fragen mit WAHR oder FALSCH bzw. allen möglichen Antworten (Ergebnissen)!**

a) ist Frank der Vater von Johanna	
b) ist Elfriede die Oma von Frank	
c) wer ist die Mutter von Hannes	
d) wer sind die Eltern von Rosi	
e) ist Frank ein nachkomme von Rosa	

- 3. Öffnen Sie sich das SWI-Prolog-Programm und erstellen Sie eine Datei ("Datei" "Neu") und speichern Sie diese sofort in Ihrem eigenen Ordner ab ("Datei" "Speichern unter ...")!**
- 4. Geben Sie die vorne dargestellten Atome und Regeln ein! (Das ist ein Teil des Stammbaums ab dem Pärchen Klara und Bernd. Daneben sind erste Verwandschaftsregeln (für Vater, Opa und Nachkomme) formuliert worden.)**
- 5. Starten Sie die Benutzung mit "Start" "Consultieren"! Werden Fehler angezeigt, dann korrigieren Sie diese, speichern wieder ab und "Consultieren" erneut! (Warnungen können Sie ignorieren! Am Ende des Konsultierens muss "true" stehen!)**
- 6. Prüfen Sie nun mit Hilfe des Prolog-Systems, ob die folgenden Verwandschafts-Beziehungen oder Personen-Daten stimmen!**

a) ist_elter_von(frank, luka).		Ist Frank ein Elter von Luka?
b) ist_elter_von(X, monika).		Wer ist Elter von Monika?
c) ist_elter_von(monika, X).		
d) ist_opa_von(bernd, johanna).		
e) ist_nachkomme_von(jonas, bernd).		

- 7. Geben Sie nun die Daten der anderen Personen ein! Achten Sie auf Vollständigkeit und die für Prolog zu verwendende Schreibweise (Syntax)!**
- 8. Prüfen Sie nun die folgenden Verwandschafts-Beziehungen und Personen-Zugehörigkeiten!**

a) ist_vater_von(frank, johanna).		Ist Frank der Vater von Johanna?
b) ist_elter_von(X, rosi).		Wer ist Elter von Rosi?
c) weiblich(M).		
d) verheiratet(X, heinz).		
e) ist_nachkomme_von(lisa, otto).		

8a) Ordnen Sie Frida drei Kinder (Michael, Kathrin, Susanne) zu, die sie nicht mit Karl gemeinsam hat!

8b) Prüfen Sie nun die folgenden Verwandschafts-Verhältnisse und Zugehörigkeiten!

a) ist_vater_von(karl, susanne).		Ist Karl der Vater von Susanne?
b) ist_elter_von(X, kathrin).		Wer ist Elter von Kathrin?
c) weiblich(M).		
d) verheiratet(X, karl).		
e) ist_nachkomme_von(frieda, karl).		

9. **Erweitern Sie die Regeln um die Verwandtschafts-Beziehungen Mutter, Oma, Uropa und Uropa!**
10. **Finden Sie Regeln für Bruder, Schwester, Tante, Onkel, Cousin und Cousine! Geben Sie eine Regel an, so dass weiblicher und männlicher Partner bei verheiratet() auch getauscht notiert werden können!**
11. **Skizzieren Sie sich einen Familienstammbaum Ihrer Familie! Verwenden Sie nur Vornamen! Bei doppelt vorkommenden Namen hängen Sie einfach den 1. Buchstaben des Nachnamens mit an!**
12. **Speichern Sie die alte Verwandtschafts-Datei unter Ihrem Familien-Namen ab! Löschen Sie dann alle Personen-Daten aus der Datenbasis! Geben Sie nun Ihre Familiendaten ein und prüfen Sie die Verwandtschafts-Verhältnisse! Schreiben Sie Ihren Namen mit bei Autor ein! Am Schluss drucken Sie den Quelltext einmal aus!**

Hier eine Umsetzung von Tic-tac-toe in PROLOG:

```

1 play :- my_turn([]).
2
3 my_turn(Game) :-
4     valid_moves(ValidMoves, Game, X),
5     any_valid_moves(ValidMoves, Game).
6
7 any_valid_moves([], _) :-
8     write('It is a tie'), nl.
9 any_valid_moves([_|_], Game) :-
10    findall(NextMove, game_analysis(X, Game, NextMove), MyMoves),
11    do_a_decision(MyMoves, Game).
12
13 % This can only fail in the beginning.
14 do_a_decision(MyMoves, Game) :-
15    not(MyMoves = []),
16    length(MyMoves, MaxMove),
17    random(0, MaxMove, ChosenMove),
18    nth0(ChosenMove, MyMoves, X),
19    NextGame = [X | Game],
20    print_game(NextGame),
21    (victory_condition(X, NextGame) ->
22     (write('I won. You lose.'), nl);
23     your_turn(NextGame), !).
24
25 your_turn(Game) :-
26    valid_moves(ValidMoves, Game, O),
27    (ValidMoves = [] -> (write('It is a tie'), nl);
28     (write('Available moves:'), write(ValidMoves), nl,
29      ask_move(Y, ValidMoves),
30      NextGame = [Y | Game],
31      (victory_condition(O, NextGame) ->
32       (write('I lose. You win.'), nl);
33       my_turn(NextGame), !))).
34
35 ask_move(Move, ValidMoves) :-
36    write('Give your move:'), nl,
37    read(Move), member(Move, ValidMoves), !.
38
39 ask_move(Y, ValidMoves) :-
40    write('not a move'), nl,
41    ask_move(Y, ValidMoves).
42
43 movement_prompt(X, Y, ValidMoves) :-
44    write('Give your X:'), nl, read(X), member(move(O, X, Y), ValidMoves),
45    !,
46    write('Give your Y:'), nl, read(Y), member(move(O, X, Y), ValidMoves).

```



```

46
47 % A routine for printing games.. Well you can use it.
48 print_game(Game) :-
49     plot_row(0, Game), plot_row(1, Game), plot_row(2, Game).
50
51 plot_row(Y, Game) :-
52     plot(Game, 0, Y), plot(Game, 1, Y), plot(Game, 2, Y), nl.
53
54 plot(Game, X, Y) :-
55     (member(move(P, X, Y), Game), ground(P)) -> write(P) ; write('.').
56
57 % This system determines whether there's a perfect play available.
58 game_analysis(_, Game, _) :-
59     victory_condition(Winner, Game),
60     Winner = x. % We do not want to lose.
61     % Winner = o. % We do not want to win. (egostroking mode).
62     % true. % If you remove this constraint entirely, it may let you win.
63 game_analysis(Turn, Game, NextMove) :-
64     not(victory_condition(_, Game)),
65     game_analysis_continue(Turn, Game, NextMove).
66
67 game_analysis_continue(Turn, Game, NextMove) :-
68     valid_moves(Moves, Game, Turn),
69     game_analysis_search(Moves, Turn, Game, NextMove).
70
71 % Comment these away and the system refuses to play,
72 % because there are no ways to play this without a possibility of tie.
73 game_analysis_search([], o, _, _). % Tie on opponent's turn.
74 game_analysis_search([], x, _, _). % Tie on our turn.
75
76 game_analysis_search([X|Z], o, Game, NextMove) :- % Whatever opponent does,
77     NextGame = [X | Game], % we desire not to lose.
78     game_analysis_search(Z, o, Game, NextMove),
79     game_analysis(x, NextGame, _), !.
80
81 game_analysis_search(Moves, x, Game, NextMove) :-
82     game_analysis_search_x(Moves, Game, NextMove).
83
84 game_analysis_search_x([X|_], Game, X) :-
85     NextGame = [X | Game],
86     game_analysis(o, NextGame, _).
87 game_analysis_search_x([_|Z], Game, NextMove) :-
88     game_analysis_search_x(Z, Game, NextMove).
89
90 % This thing describes all kinds of valid games.
91 valid_game(Turn, Game, LastGame, Result) :-
92     victory_condition(Winner, Game) ->
93     (Game = LastGame, Result = win(Winner)) ;
94     valid_continuing_game(Turn, Game, LastGame, Result).
95
96 valid_continuing_game(Turn, Game, LastGame, Result) :-
97     valid_moves(Moves, Game, Turn),
98     tie_or_next_game(Moves, Turn, Game, LastGame, Result).
99
100 tie_or_next_game([], _, Game, Game, tie).
101 tie_or_next_game(Moves, Turn, Game, LastGame, Result) :-
102     valid_gameplay_move(Moves, NextGame, Game),
103     opponent(Turn, NextTurn),
104     valid_game(NextTurn, NextGame, LastGame, Result).
105
106 % Victory conditions for tic tac toe.
107 victory(P, Game, Begin) :-
108     valid_gameplay(Game, Begin),
109     victory_condition(P, Game).
110
111 victory_condition(P, Game) :-
112     (X = 0; X = 1; X = 2),
113     member(move(P, X, 0), Game),
114     member(move(P, X, 1), Game),
115     member(move(P, X, 2), Game).
116

```

```

117 victory_condition(P, Game) :-
118     (Y = 0; Y = 1; Y = 2),
119     member(move(P, 0, Y), Game),
120     member(move(P, 1, Y), Game),
121     member(move(P, 2, Y), Game).
122
123 victory_condition(P, Game) :-
124     member(move(P, 0, 2), Game),
125     member(move(P, 1, 1), Game),
126     member(move(P, 2, 0), Game).
127
128 victory_condition(P, Game) :-
129     member(move(P, 0, 0), Game),
130     member(move(P, 1, 1), Game),
131     member(move(P, 2, 2), Game).
132
133 % This describes a valid form of gameplay.
134 % Which player did the move is disregarded.
135 valid_gameplay(Start, Start).
136
137 valid_gameplay(Game, Start) :-
138     valid_gameplay(PreviousGame, Start),
139     valid_moves(Moves, PreviousGame, _),
140     valid_gameplay_move(Moves, Game, PreviousGame).
141
142 valid_gameplay_move([X|_], [X|PreviousGame], PreviousGame).
143 valid_gameplay_move([_|Z], Game, PreviousGame) :-
144     valid_gameplay_move(Z, Game, PreviousGame).
145
146 % The set of valid moves must not be affected by the decision making
147 % of the prolog interpreter.
148 % Therefore we have to retrieve them like this.
149 % This is equivalent to the  $(\forall x \in 0..2) (\forall y \in 0..2) (\dots)$ 
150 % uh wait.. There's no way to represent this using those quantifiers.
151 valid_moves(Moves, Game, Turn) :-
152     valid_moves_column(0, M1, [], Game, Turn),
153     valid_moves_column(1, M2, M1, Game, Turn),
154     valid_moves_column(2, Moves, M2, Game, Turn).
155
156 valid_moves_column(X, M3, M0, Game, Turn) :-
157     valid_moves_cell(X, 0, M1, M0, Game, Turn),
158     valid_moves_cell(X, 1, M2, M1, Game, Turn),
159     valid_moves_cell(X, 2, M3, M2, Game, Turn).
160
161 valid_moves_cell(X, Y, M1, M0, Game, Turn) :-
162     member(move(_, X, Y), Game) -> M0 = M1 ; M1 = [move(Turn, X, Y) | M0].
163
164 % valid_move(X, Y, Game) :-
165 %     (X = 0; X = 1; X = 2),
166 %     (Y = 0; Y = 1; Y = 2),
167 %     not(member(move(_, X, Y), Game)).
168
169 opponent(x, o).
170 opponent(o, x).

```

Q: <https://github.com/cheery/prolog-tic-tac-toe/blob/master/tictactoe.pro>

```

consult('tictactoe.pro').
play.

```

weitere(s) Beispiel(e) im Anhang (→ [Anhänge](#)):
→ [Tic-tac-toe mit Minimax in PROLOG](#)

2.x. Modell-Verwaltung (Model Governance)

Bedeutung:

- Zugriff nach Rollen (Daten-Beschaffung, Entwicklung, Implementierung, Deployment (Bereitstellung), ...) → Role based Access Control (RBAC)
- Versionierung (von Daten und Modellen, inkl. Log's und Anmerkungen)
- Monitoring (wie werden Modelle nach dem Training bewacht?)
- automatisiertes Testen
- Model Decay (Abschwächung des Modell's in der Zeit)
- ...

Prior Probability Shift

Veränderung der Voraussagen durch Veränderung der Verteilungen (zwischen Trainings- und Arbeits-Daten)

3. weitere Grundlagen zu KI-Problem-Feldern

3.x. Verarbeitung von Texten

Text-Analyse

diverse statistische Parameter, die sich aus einem Text ableiten lassen

z.B.:

durchschnittliche Wort-Länge

durchschnittliche Anzahl von Worten je Satz

für sich eher Statistik, aber die einzelnen Parameter können dann in ein KNN eingeschleust werden

lassen z.B. die Erkennung / Identifizierung des Autors / Überprüfung der Autorenschaft zu Ermittlung von Fake News
Erkennung von Chatbot's od.ä.

Analyse-Ansätze

- | | |
|--|---|
| <ul style="list-style-type: none">• Count-based Embeddings Häufigkeits-basierte | <p>zählende / berechnende Analysen
Quantitäts-orientiert</p> <p>Verfahren:</p> <ul style="list-style-type: none">• Bag of Word / Characters• Tf-idf-Vektoren• ... |
| <ul style="list-style-type: none">• Learning-based Embeddings Lern-basierte | <p>Qualitäts-orientiert / Inhalts-bezogen</p> <p>Verfahren:</p> <ul style="list-style-type: none">• Word2Vec• GloVe• BERT-based• ... |
| <ul style="list-style-type: none">• ... | |

Probleme mit der Semantik von Texten

Der Mann wohnt auf einer Insel. Nach der Arbeit fährt er nachhause.

Nach einem Kongress über moderne Mode in New York begab sie sich wieder nach Paris.

gleiche Worte mit gleicher Bedeutung

Sie bereitete das Mittagessen zu.

Sie begab sich zum Mittagessen.

Sie nahm das Mittagessen zu sich.

verschiedene Worte mit gleicher Bedeutung

gleiche Worte mit unterschiedlicher Bedeutung

Er setzt sich auf die Bank.

Er setzt sich in die Bank.

Er betrat die Bank.

Er stellte sich auf die Bank.

in verschiedenen Sprachen gibt es unterschiedliche Wort-Paarungen und –Bedeutungen, die nicht 1 : 1 zu übertragen sind

Public Viewing (engl.: Leichenbeschau / Aufbahrung ↔ dt.: öffentliche Bildschirm-Präsentation)

komplexe Sätze mit für Menschen zuordnenbaren Zusammenhängen

Anna hörte auf, die Pille zu nehmen, weil sie schwanger ist.

Der Bauer musste vor dem König Rechenschaft ablegen, weil er schlecht war.

syntaktisch richtige Sätze ohne Sinn / Semantik

Das Haus ohne Wände und Dach stand in den Wolken.

3.x.y. Word Embeddings

Technik der Natural Language Processing (NLP, (maschinelle) Verarbeitung natürlicher Sprache)

Die Bedeutung eines Wortes
ist sein Gebrauch in der Sprache.
Ludwig WITTGENSTEIN

Word Embeddings bedeutet übersetzt Wort-Einbindung; gemeint ist die Wort-Übertragung von z.B. einer Sprache in eine andere, aber auch die Erkennung von Ähnlichkeiten / Zusammengehörigkeiten

Natürlich lassen sich für Übersetzungen einfache Vokabel-Listen anlegen. Mit diesen ist dann eine mechanische Übertragung der Sätze einer Sprache in eine andere möglich. Allerdings ist die Übersetzungs-Qualität sehr fragwürdig.

Nehmen wir als Beispiel den einfachen englischen Satz:

englisch	deutsch

und übersetzen ihn mithilfe der Vokabel-Tabelle (rechts) ganz mechanisch in einen deutschen Satz.

Leichte Anpassungen hinsichtlich dem Geschlecht () oder der Einzahl / Mehrzahl (Singular / Plural) bekommen wir mechanisch auch noch ganz gut hin:

Trotzdem befriedigt das Ergebnis nicht 100 prozentig. Frühe Versionen des google-Übersetzungs-Dienstes brachten solche Ergebnisse. Heute ist der Übersetzungs-Dienst, wie auch andere (z.B.: deepl.com) mit künstlicher Intelligenz verstärkt. Da kommen dann deutlich bessere Übersetzungen heraus:

Um von dem eigentlichen Wort oder Satz wegzukommen, erfasst man Eigenschaften der Worte bzw. Sätze. Bei Sätzen sind das z.B. Funktionen der Wörter (z.B. Subjekt, Prädikat, Objekt, ...) und deren Anordnung innerhalb zueinander. Bei Worten sind es begleitende / zugehörige Eigenschaften, wie z.B.

Diese Analysen führen uns zu verallgemeinerten grammatikalischen Regeln

Bei einzelnen Worten können wir uns die verschiedensten Eigenschaften nehmen und deren Ausprägung für das jeweilige Wort ermitteln.

Dabei ist aber die Mitarbeit von Menschen unbedingt notwendig. Diese auch Annotation genannte Bewertung ist deshalb sehr teuer und auch wenig effektiv. Vielleicht müssen beim Hinzufügen eines neuen Fachgebiets neue Wort-Eigenschaften eingeführt werden. Dann müssen nachträglich alle schon annotierten Wörter für die neuen Eigenschaften bewertet werden. Das ist ein praktisch unendliches Arbeiten.

Praktisch erhalten wir so für jedes Wort Listen von Eigenschaften und Werten.

Sie stellen eine Ersetzung der Worte durch Zahlen oder ähnliche mathematische Gebilde (z.B. Vektoren) dar. Nur diese können überhaupt effektiv mit Computern verarbeitet werden.

Geht man nun in einem nächsten Schritt von der menschlichen Annotation weg, und lässt den Computer Worte nach einem eigenem System / Schema bewerten. Der Vorteil liegt dabei darin, dass Computer nur das Wort selbst bewerten / interpretieren und nicht auch noch die Eigenschaft selbst mit dazu interpretieren, wie es das Menschen machen. Wann hört z.B. "klein" auf und ab wann ist etwas "gross"? Gehört "türkis" zu "grün oder blau?"

bei "one hot" werden die verschiedenen Kategorien in lange Listen von Eigenschaften zerlegt, die nur noch mit 0 oder 1 bewertet werden müssen / können

die Codierung als Dualzahl kommt der Verarbeitung in Computern extrem entgegen

problematisch ist hier nun aber wieder die Abstands-Berechnung, also die Frage, ob sich zwei Worte ähneln

durch den Menschen oder ein anderes Modell müssen die Beziehungen und Vergleichskriterien festgelegt werden

z.B. reicht die reine Anzahl gleicher Einsen nicht, um zwei Wort als ähnlich zu klassifizieren

Eigenschaft	Wert
Substantiv	
Verb	
...	
Metall-Gegenstand	
Holz-Gegenstand	
...	
bewirkt Orts-Veränderung	
bewirkt Farb-Veränderung	
...	
mikroskopisch klein	
Elementarteilchen	
makroskopisch groß	
größer als ein Mensch	
...	

einen anderen Ansatz bei der Wort-Annotation geht man mittels Kookkurrenz. Bei Kookkurrenz geht es um das gemeinsame Auftreten von Wörtern (allgemein: lexikalischen Einheiten) in einem bestimmten Text-Körper (z.B. einen Lexikon-Eintrag, einen Wikipedia-Artikel, in einem Lehrbuch, ...)

Stop-Worte, wie "und", "dass", ... oder die Artikel werden vorher aussortiert.

Die bei diesem Ansatz entstehende Daten-Menge ist sehr groß. Schließlich muss zumindestens das gesamte betrachtete Vokabular – ev. sogar alle Worte – in den Vektoren erfasst werden.

Einzelvorkommende Worte – es könnten auch z.B. Rechtschreibfehler sein – vergrößern die Vektoren ohne auch nur den geringsten Nutzwert. Dafür steigt die notwendige Verarbeitungs-Leistung zumindestens exponential.

Aus Gründen der Praktikabilität werden die Wort-Listen auf eine bestimmte Größe beschränkt. Typisch sind z.B. die 300 am meisten vorkommenden Wörter oder nur die Wörter, die überhaupt mehr als 1x im Text vorkommen.

Durch dieses Abschneiden der Daten sind die Aussagen des Word Embedding's allerdings nur begrenzt Aussage-fähig.

Definition(en): Kookkurrenz

Kookkurrenz ist ein Maß für das gemeinsame Auftreten einer lexikalischen Einheit (im Allgemeinen: ein Wort) mit anderen Wörtern.

Unter Kookkurrenz versteht man in der allgemeinen Linguistik das gemeinsame Auftreten von zwei lexikalischen Einheiten (z.B.: Wort) in einer übergeordneten Einheit (z.B. Satz, Absatz, Gesamttext).

Mit Hilfe von statistischen Test's und Verfahren erhält man Maße für die Wort-Abhängigkeit. Typische Test's sind:

- Likelihood-Quotienten-Test
- Transinformation (Synentropie; gegenseitiger Information's-Wert)
- ...

In einem anderen Verfahren nutzt man Künstliche Neuronale Netze schon gleich für die Erstellung von Daten-Vektoren. Dabei werden die Gewichte einer versteckten Schicht (Hidden Layer) als "Merkmale" verwendet. Der Nachteil hier ist, dass diese "Merkmale" kaum einen Bezug zu Menschen-verständlichen Merkmalen haben. Das genaue Verfahren zur Bestimmung / Ermittlung dieser Gewichte / Werte lernen wir später kennen.

Ein Vorteil dieses Verfahrens ist die Begrenzung der Vektoren auf einige Hundert Werte / "Merkmale". Die Vektoren können direkt als Umsetzungs-Tabellen für das Word Embedding genutzt werden.

Das informatische Word Embedding geht bis zum Jahr 1954 zurück, in diesem Jahr gelang es mit Hilfe einfacher Ordnungs- und Ersetzungs-Regeln 60 Sätze aus dem Russischen ins Englische zu übersetzen.

dazu wurde praktisch mit Matrizen-Berechnungen gearbeitet

Prinzip:

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} W_{00} & W_{01} & W_{02} \\ W_{10} & W_{11} & W_{12} \\ W_{20} & W_{21} & W_{22} \\ W_{30} & W_{31} & W_{32} \\ W_{40} & W_{41} & W_{42} \end{bmatrix} \begin{bmatrix} V_{00} & V_{01} & V_{02} \\ V_{10} & V_{11} & V_{12} \\ V_{20} & V_{21} & V_{22} \\ V_{30} & V_{31} & V_{32} \\ V_{40} & V_{41} & V_{42} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Input
Word-Embeddings
Output

nach Q: <https://entwickler.de/online/python/grundsaeetze-moderner-textklassifizierung-word-embeddings-579923897.html>

Zuerst standen die Einträge in den Input- und Output-Vektoren für einzelne Worte.

??? Dabei ist der Input ein Vektor von Eigenschaften für das Wort. Gleiches gilt für den Output. Er ist ein Vektor von Eigenschaften. Mit ihm wird in der Datenbasis der Ziel-Sprache nach dem passenden Wort gesucht.

in den 1980er Jahren kam es dann zu einer revolutionären Entwicklung der NLP

Wörter werden in einem viel-dimensionalen Raum eingebettet dazu bekommen Eigenschaften bzw. Werte für bestimmte Charakteristika die Merkmals-Werte werden in Vektoren (z.B. als Listen oder Felder (Array's)) gespeichert und Tensoren genannt jedes Merkmal oder Charakteristikum ist dann eine Dimension im Wort-Raum für jedes Wort ergibt sich eine Position innerhalb dieses Raumes

man kann dann nach Worten mit ähnlichen Merkmalen suchen, diese sollten im Raum benachbart liegen

aber man kann auch Worte mit z.B. gleichem Wortstamm hinsichtlich ihrer Lage im Wort-Raum untersuchen oder was macht eine Vorsilbe wie "Anti-" mit der Lage der Worte im Raum

Liegen sie woanders?
Bilden sie eine Gruppe / ein Cluster?
Liegen sie dichter oder weiter auseinander?
Welche Begriffs-Analogien lassen sich ableiten?
Liegen die Wörter eines Lebens- oder Fach-Bereiches irgendwo zusammen im Raum?
Liegen unterschiedliche Wörter mit ähnlicher Bedeutung benachbart im Raum?
Lassen sich Haltungen, Weltanschauungen, Selbstmord-, Mord- oder Attentats-Absichten usw. usf. erkennen?
...

Möglichkeiten zu Ermittlung von Word Embeddings

- **manuell**
- **statistisch**
- **maschinell erlernt**

Bag-Of-Words (BOW)

sehr einfacher Arbeits-Ansatz
ausreichend für viele Analysen

Verfahren:

- Sammeln aller verschiedener Worte aus einem Text-Corpus (verfügbare oder zu untersuchende Text-Sammlung)
- Initialisierung der Kennwerte
- Zählen der Wort-Vorkommen

statt mit Worten kann man auch die verwendeten Zeichen (Buchstaben, ...) zählen
über OCR (Optical Character Recognition, automatische Schrift-Erkennung) lässt sich das auch recht gut automatisieren

Text wird zuerst von allen Sonderzeichen bereinigt
danach wird ein Wörterbuch erstellt, die Wort-Vorkommen werden gezählt
ev. wird das Wörterbuch nach den Wort-Vorkommen (absteigend) sortiert
in einem nach Wort-Vorkommen sortierten BOW wird den einzelnen Wörterbuch-Einträgen eine Rangfolge zugeordnet (1 für das häufigst Wort, 2 für das zweit-häufigste usw.usf.)

am Ende kann man den Text als Liste von Zahlen darstellen, welche der (Häufigkeits-)Rangfolge im Wörterbuch entspricht
oder z.B. Word-Wolken ableiten

für speziellere Untersuchungen kann man das BOW noch um Wörter ergänzen, die nicht im Text vorkommen, deren Vorkommen wird auf 0 gesetzt
dieser Wörterbuch-Teil wird OOV (Out-Of-Vocabulary) genannt

Vorteile:

- einfachere Verarbeitung von Zahlen anstatt Strings
- ...

Nachteile:

- Zahlen widerspiegeln nichts über die Wortgröße oder deren Inhalt / Bedeutung / ... wieder
- für die Verarbeitung der BOW's werden KNN mit sovielen Eingabe-Neuronen benötigt, wie es Einträge im Wörterbuch gibt
- für jedes Wort entsteht als Input in das KNN ein sogenannter One-Hot-Encoded-Vector (dieser besteht aus einer einzigen 1 an der Position der Worthäufigkeit und sonst nur Nullen) → solche Vektoren (Sparse Vector) können Lern-Vorgänge verkomplizieren
- Häufigkeiten pro Satz haben nur geringe Zähl-Werte (Häufigkeiten)
- hohe Dimensionalität
- Worte mit gleicher Zeichen-Squenz – aber unterschiedlicher Bedeutung – werden gleich-gemacht (Bank (Geldhaus, Sitzgelegenheit, ...))
- Problem bei neuen Sätzen / Texten mit neuen Worten (Out-of-Vocabulary (OOV)) → erfordert Nachbearbeitung des Wörterbuch's und aller alten Vektoren
- ...

→ in Python (und anderen Programmiersprachen) z.B. durch Dictionary's realisierbar

Daten-Bestand:

Mausi ist eine freche Katze.
 Sie frisst gern Mäuse.
 Eine Katze hat sieben Leben.

Vokabular: (lowercase, sortiert)

["eine", "freche", "frisst", "gern", "hat", "ist", "katze", "leben", "mäuse", "mausi", "sie", "sieben"]

Wörterbuch (Dictionary, über Text-Corpus):

```
{
  "eine": 2,
  "freche": 1,
  "frisst": 1,
  "gern": 1,
  "hat": 1,
  "ist": 1,
  "katze": 2,
  "leben": 1,
  "mäuse": 1,
  "mausi": 1,
  "sie": 1,
  "sieben": 1
}
```

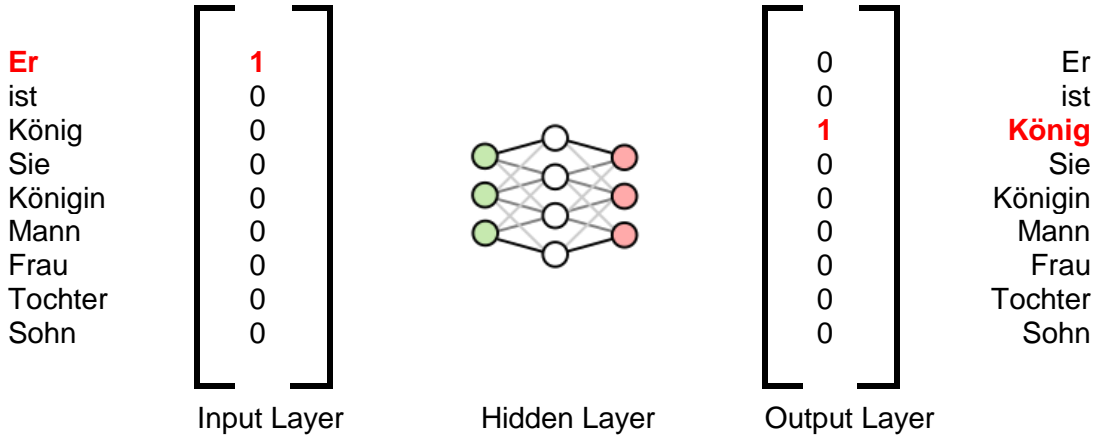
Vektoren:

```
[[1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0]
 [0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0]
 [1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1]]
```


1-Hot-Encoded-Vektor Beispiel

nach Q: <https://entwickler.de/online/python/grundsaeetze-moderner-textklassifizierung-word-embeddings-579923897.html>

```
text_zum_trainieren = ["Er ist König", "Sie ist Königin", "Er ist Mann",
                        "Sie ist Frau", "Sie ist Tochter", "Er ist Sohn"]
```



$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} W_{00} & W_{01} & W_{02} & W_{03} & W_{04} \\ W_{10} & W_{11} & W_{12} & W_{13} & W_{14} \\ W_{20} & W_{21} & W_{22} & W_{23} & W_{24} \\ W_{30} & W_{31} & W_{32} & W_{33} & W_{34} \\ W_{40} & W_{41} & W_{42} & W_{43} & W_{44} \\ W_{50} & W_{51} & W_{52} & W_{53} & W_{54} \\ W_{60} & W_{61} & W_{62} & W_{63} & W_{64} \\ W_{70} & W_{71} & W_{72} & W_{73} & W_{74} \\ W_{80} & W_{81} & W_{82} & W_{83} & W_{84} \end{bmatrix} \begin{bmatrix} V_{00} & V_{01} & V_{02} & V_{03} & V_{04} \\ V_{10} & V_{11} & V_{12} & V_{13} & V_{14} \\ V_{20} & V_{21} & V_{22} & V_{23} & V_{24} \\ V_{30} & V_{31} & V_{32} & V_{33} & V_{34} \\ V_{40} & V_{41} & V_{42} & V_{43} & V_{44} \\ V_{50} & V_{51} & V_{52} & V_{53} & V_{54} \\ V_{60} & V_{61} & V_{62} & V_{63} & V_{64} \\ V_{70} & V_{71} & V_{72} & V_{73} & V_{74} \\ V_{80} & V_{81} & V_{82} & V_{83} & V_{84} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \end{bmatrix} \begin{bmatrix} 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \\ 0,1 & 0,1 & 0,1 & 0,1 & 0,1 \end{bmatrix}^T = \begin{bmatrix} 0,5 \\ 0,5 \\ 0,5 \\ 0,5 \\ 0,5 \\ 0,5 \\ 0,5 \\ 0,5 \\ 0,5 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix}^T = \begin{bmatrix} 1,843 \\ 0,049 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix} = \begin{bmatrix} 5,378 \\ 1,639 \\ 0,745 \\ 2,064 \\ 0,651 \\ 0,544 \\ 1,056 \end{bmatrix}$$

Unsupervised Word Vectors Word2Vec, GloVe

verbesserter Analyse-Ansatz
läßt mehr Ableitungen über Wortzusammenhänge zu

Text wird zuerst von allen Sonderzeichen bereinigt
danach wird ein Wörterbuch erstellt, die Wort-Vorkommen werden gezählt
zusätzlich werden in einer mehr-dimensionalen Informationen zu umgebenen Wörtern (ihren Kontexten) gespeichert

die "Kriterien" – die Dimensionen des Wortraum's – werden durch "unsupervised learning" selbstständig von einem KNN erstellt
das KNN bildet sich also ein eigenes Repräsentations-Modell
für jedes Wort entsteht ein Vektor mit 100 bis 300 Dimensionen (Wort-Merkmalen)
dadurch wird die semantische Bedeutung jedes Wortes im Text abgebildet

Word-Analogy-Task

Wort1 verhält sich zu Wort2, wie Wort3 zu Wort4

anders notiert:

Wort1 : Wort2 = Wort3 : Wort 4

mit den Vektoren können dan Berechnungen erfolgen

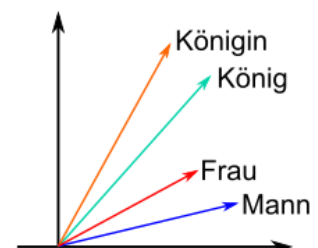
Königin = König – Mann + Frau

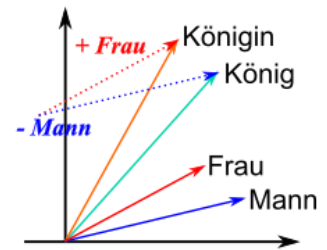
oder eben auch:

London = Berlin – Großbritannien + Deutschland

oder mal anders herum verwendet:

Hebamme – Frau + Mann = Entbindungspfleger





ähnliche Begriffe liegen im Vektor-Raum dichter beieinander
 aus der (EUKlidischen) Distanz kann auf die Wort-Ähnlichkeit geschlossen werden
 ergibt einen sogenannten Abstand-Klassifikator

Vorteile:

- weniger komplex durch weniger Dimensionen (der Eingabe-Vektoren)
- repräsentiert relativ sinnvolle Wort-Zusammenhänge
- ...

Nachteile:

- unklare Belegung der Dimensionen (nur das KNN kennt das Repräsentations-Modell)
- keine Analysen von Wort-Rollen und größeren Text-Teilen möglich
- sehr rechen-intensiv; großer (Haupt-)Speicher-Bedarf
- Wörter, die beim Training (in der "unbeobachteten Lern-Phase") nicht vorhanden waren, erhalten einen Null-Vektor → vor allem bei speziellen (Fach-)Texten problematisch
- ...

Skip-Gram-Modell

hier mit Window $c=2$

Sein	oder	nicht	sein	das	ist	hier	die	Frage
------	------	-------	------	-----	-----	------	-----	-------

Nachbarschafts-
wörter

Sein	oder	nicht	sein	das	ist	hier	die	Frage
------	------	-------	------	-----	-----	------	-----	-------

Sein	oder	nicht	sein	das	ist	hier	die	Frage
------	------	-------	------	-----	-----	------	-----	-------

Zentral-
wort

Sein	oder	nicht	sein	das	ist	hier	die	Frage
------	------	-------	------	-----	-----	------	-----	-------

Sein	oder	nicht	sein	das	ist	hier	die	Frage
------	------	-------	------	-----	-----	------	-----	-------

Supervised Word Vectors – Tensorflow Embedding

das vorher definierte und verwendete Wörterbuch ist vorklassifiziert worden

Text wird zuerst von allen Sonderzeichen bereinigt

Text / Worte wird nun mit dem vortrainierten KNN ausgewertet; dadurch erhält man für jedes Wort / für jede Wort-Kombination einen Vektor (Ausgabe des KNN), der nun weiter verarbeitet werden kann (z.B. in Recurrent Neural Network (RNN,))

für eine neue Klassifizierung können alle Vektoren des Textes addiert werden und durch die Zahl der Wörter geteilt werden

Vorteile:

-

Nachteile:

-

neue Ansätze

Buchstaben-weise Analyse der Texte
Sequence-To-Sequence-Modelle

z.B. geeignet für die automatische Text-Generierung; automatische inline Text-Übersetzung;
Chat-Bot's

Vorteile:

-

Nachteile:

-

Verwendung auch zur Autoren-Erkennung
 Bewertung der Stimmung des Autor's ()
 Extraktion von Daten / Fakten
 Überprüfungen der Echtheit von Beiträgen in Empfehlungs-Systemen / Fakten-Prüfung
 Paraphrasieren
 Übersetzungen
 Text-Zusammenfassung
 Frage-Antwort-Systeme
 Verstehen neuer Wörter
 Semantische Suchmaschinen (→ WOLFRAM alpha)

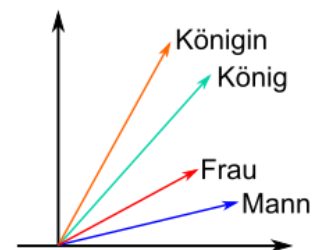
(aus Datenbanken-Skript übernommen → später zurückaktualisieren!!!)

z.B. Erkennen oder Abfragen von Synonymen (diese liegen im Raum dicht beieinander)
 läuft praktisch vollständig automatisiert und wird stark durch maschinelles Lernen getrieben

praktisch kann man dann auch mit den Vektoren arbeiten und rechnen
 daraus lassen sich dann wieder semantische Beziehungen herstellen

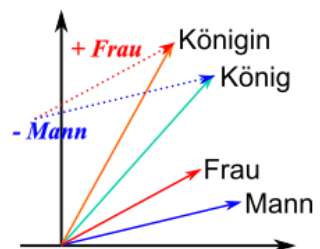
für einfache Sprach-Erkennungs-Systeme sind die folgenden
 Wort-Analogie-Rätsel schnell ein Problem:

1. Lehrerin und Lehrer verhalten sich wie Polizistin und X
2. König und Königin verhalten sich wie Mann und X
3. Gross verhält sich zu klein, wie schnell zu X
4. Deutschland liegt in Europa und Argentinien in X
5. Was in Brasilien Rio de Janero ist ist in Deutschland X



Die erste Aufgabe scheint einfach zu sein. Einfach das -in wegnehmen und schon haben wir den Polizist gefunden. Aber schon beim Ableiten von Anwalt aus Anwältin haben wir ein Problem.

Für die Lösung der zweiten Aufgabe braucht es schon etwas Hintergrund-Wissen.



z.B.: gegeben sind die Vektoren von Mann, Frau, König und Königin

über die "Vektor.Berechnung" $\text{König} - \text{Mann} + \text{Frau}$ erhält man dann Königin (oder einem synonymen Begriff (oder zumindestens einen in dessen Nähe))

bei 3. Begriff-Paare, die keine gemeinsamen Wortstämme mehr haben

4. benötigt als Hintergrund-Wissen die leicht zu erratende Beziehung Land-Kontinent

5. benötigt weitere Zusatz-Annahmen, die nicht primär in dem Satz stehen
 geht es um eine große Stadt oder um eine Hafenstadt
 mal wäre vielleicht München eine Antwort, beim anderen Fall Hamburg

oder geht gar um vermeintliche oder historische Hauptstädte, dann könnte es vielleicht Bonn sein

Sentiment Analysis (Stimmungs-Analyse)

erkunden der Stimmung, Gefühle, Meinung, Haltung, Parteilichkeit, ... eines Textes

z.B. Produkt-Bewertungen, Post's in Social Media usw. eintufen

google-mail hat mal getestet, die eMail's vor dem Abschicken zu analysieren und gegebenenfalls empfohlen, den Text erst am nächsten Tag abzusenden

z.B. über Stichwort-Listen

auch gute Anwendung für maschinelles Lernen

z.B. aus Kombinationen aus textuellen Bewertungen und vergebenen Sternen

praktisch heute schon in vielen Social Media angewendet → Filter-Blase

Nutzer bekommen nur noch positiv gestimmte / anregende / interessante / ... Beiträge zu sehen, um ihn weiter im Medium zu halten

praktisch werden vor allem die eigenen Meinungen usw. usf. wieder angezeigt, so dass man den Eindruck hat, dass wäre die Welt-Meinung
echter Pluralismus so nicht (mehr) möglich

Inhalts-basierte Text-Indizierung (z.B. für / in Suchmaschinen)

einzelne Texte (Web-Seiten) werden von Stop-Wörtern und Sonderzeichen bereinigt
man erstellt dann eine Liste der Wörter, wobei besonders die seltenen Wörter interessieren

Text-Nr.	Text-Inhalt
7831	Biologie ist die Wissenschaft vom Leben.
7832	Eine andere Wissenschaft ist Chemie. Physik ist ebenfalls eine Wissenschaft.
7833	Die Biochemie ist ein Teilgebiet der Biologie, das sich mit der Chemie der Zelle beschäftigt.

Wir konzentrieren uns hier im Beispiel nur auf Substantive. Nach der Zerlegung von Text 7831 ergibt sich das Wörterbuch:

WB = [Biologie, Wissenschaft, Leben]

das Dokument selbst erhält als Wort-Vektor:

Dok(7831) = [1, 1, 1]

da ja jedes der Wörter im Wörterbuch 1x vorkommt.

nach der Zerlegung und Bereinigung von Text 7832 erweitert sich das Wörterbuch auf:

WB = [Biologie, Wissenschaft, Leben, Chemie, Physik]

der Vektor des Dokuments sieht dann so aus:

Dok(7832) = [0, 2, 0, 1, 1]

Das Wort "Biologie" kommt hier nicht vor, "Wissenschaft" z.B. zweimal.

Beim Analysieren des Textes 7833 erweitert sich das Wörterbuch wieder:

WB = [Biologie, Wissenschaft, Leben, Chemie, Physik, Biochemie, Teilgebiet, Zelle]

und der zugehörige Text-Vektor:

Dok(7833) = [1, 0, 0, 1, 0, 1, 1, 1]

Mit den Wort-Vektoren und dem zugehörigen Wörterbuch lassen sich nun viele weitere Untersuchungen / Klassifizierungen usw. usf. vornehmen

z.B. kann der Text 9482 "Die Wissenschaft vom Leben ist die Biologie." als ähnlich / gleich / identisch eingestuft werden, da sich der gleiche Vektor ergibt

Dok(9482) = [1, 1, 1, 0, 0, 0, 0, 0, 0,]

Das Wörterbuch ist weiter gewachsen, da sehr wahrscheinlich viele neue Wörter in den inzwischen bearbeiteten Texten enthalten gewesen sind. Sie sind aber alle nicht im Text 9482 enthalten. Damit ist der gepunktete Teil ein Nullen-Vektor.

Links:

<https://deepset.ai/german-word-embeddings> (Modell-Quelltexte, Vektoren (für deutsche Wörter), (Open Source))

<http://www.spinningbytes.com/resources/wordembeddings/> (Datenbestände für mehrere Sprachen)

3.x.y.z. praktische Umsetzungen / Versuche

manuelles / mechanisches Word Embedding

leicht zu realisieren

Wir gehen fast immer davon aus, dass der Quell-Satz schon in einzelne Worte zerlegt worden ist und diese als Liste oder als Wort-Vektor (Feld, Array) vorliegen.

"manuelles Word Embedding" in Python

hier gibt es die Daten-Struktur des Dictionary /Wörterbuch

dieses Verfahren ist weitestgehend auch in anderen Programmiersprachen möglich für Python steht auch eine spezielle Word-Embedding-Bibliothek zur Verfügung
gensim

nachzuinstallieren mit:

```
pip install --upgrade gensim
```

Links:

<https://machinelearningmastery.com/develop-word-embeddings-python-gensim/>

"manuelles Word Embedding" mit Prolog

Vorteil von Prolog ist die Möglichkeit, sich mehrere Versionen / Ergebnisse erstellen zu lassen. Der Nutzer kann sich dann die beste Variante auswählen.

"manuelles Word Embedding" in SNAP!

eine Möglichkeit ist die Verwendung von vorgefertigten Blöcken. Z.B. stellt das Project "eCraft2Learn" diverse Materialien und Bibliotheken zur Verfügung.

genutzt wird die Programmier-Umgebung SNAP!

Block-orientiert

entwickelt an der Stanford University

weniger verspielte Version (von "Scratch") für die schulische und universitäre Ausbildung / Arbeit / Forschung

SNAP! ist online verfügbar (→ <https://snap.berkeley.edu/>)

Für die offline-Verwendung gibt es auch eine download-bare Version (→ <https://snap.stanford.edu/snap/download.html>)

Einstellen der Standard-Sprache

integriert sind 15 Sprachen; weitere / eigene Sprachen können integriert werden

Standard-Sprache ist zunächst englisch

Anzeige aller Worte

integriert 20'000 Wörter

Anzeige eines Wort-Vektor's

Visualisierung von Word Embedding's

Suche nach einem "nahen" Wortes

Bestimmen von weit entfernten Worten

Finden von Wort-Analogien

zwei-dimensionale Approximation eines Wortes

in /11/ vorgeschlagene Projekt-Ideen:

- Ähnlichkeit von Sätzen finden (ev. mit Bag-of-words-Technik)
- Ermitteln einer Kette von ähnlichen Wörtern
- Erstellen von Wort-Spielen (nach dem Vorbild von google-Semantris; ev. als bilinguale Version)
- Suche nach neuen Wort-Analogien
- Suche nach Vorurteilen
- Text-Eingabe in Neuronale Netze

Links:

<https://ecraft2learn.github.io/ai/> (Projektseite)

<https://ecraft2learn.github.io/ai/AI-Teacher-Guide/chapter-5.html> (Lehrer-Hilfe (engl.)) --> Vorstellung der Nutzungs-Möglichkeiten

statistische Analyse von Word Embeddings

maschinelles Erlernen von Word Embeddings

eine word2vec-Umsetzung (für deutsche Worte)

Q: <https://devmount.github.io/GermanWordEmbeddings/>

```
mkdir corpus
mkdir model
```

```
wget http://www.statmt.org/wmt14/training-monolingual-news-crawl/news.2013.de.shuffled.gz
gzip -d news.2013.de.shuffled.gz
python preprocessing.py news.2013.de.shuffled corpus/news.2013.de.shuffled.corpus -psub
rm news.2013.de.shuffled.gz
```

```
wget http://download.wikimedia.org/dewiki/latest/dewiki-latest-pages-articles.xml.bz2
wget http://medialab.di.unipi.it/Project/SemaWiki/Tools/WikiExtractor.py
python WikiExtractor.py -c -b 25M -o extracted dewiki-latest-pages-articles.xml.bz2
find extracted -name '*bz2' \! -exec bzip2 -k -c -d {} \; > dewiki.xml
printf "Number of articles: "
```

```
grep -o "<doc" dewiki.xml | wc -w
sed -i 's/<[^>]*>//g' dewiki.xml
rm -rf extracted
python preprocessing.py dewiki.xml corpus/dewiki.corpus -psub
rm dewiki.xml

python training.py corpus/ model/my.model -s 300 -w 5 -n 10 -m 50

python evaluation.py model/my.model -u -t 10
```

German language model (→ <http://cloud.devmount.de/d2bc5672c523b086>)

Gensim

Q:

```
from gensim.models.keyedvectors import KeyedVectors

def load_word_vectors():
    print("Loading word vectors from", word_vectors_model_filename + "...")
    word_vectors_model_size = os.path.getsize(word_vectors_model_filename) /
(1024.0 * 1024.0)
    print("File size is {:.2f}MB".format(word_vectors_model_size))
    print("Be patient! This might take a while...")
    global word_vectors_model
    try:
        print("Attempting to load as vector-format...")
        word_vectors_model =
KeyedVectors.load_word2vec_format(word_vectors_model_filename, bina-
ry=False)
        print("Success!")
    except:
        print("Failed!")
        try:
            print("Attempting to load as binary-format...")
            word_vectors_model =
KeyedVectors.load_word2vec_format(word_vectors_model_filename, binary=True)
        except:
            print("Failed!")
            exit(0)

    print("Success!")

print("Vocabulary size", len(list(word_vectors_model.wv.vocab.keys())))
print("Word vector length:", len(word_vectors_model.wv["Mann"]))
```

ergibt:
Vocabulary size: 608130
Word vector length: 300

```
print("Word vector for 'Mann'", word_vectors_model.wv["Mann"])
```

ergibt:

Word vector for 'Mann' [0.1678151 -0.01715856 0.14564128 0.01643708
0.02733266 0.49015957
-0.19200401 -0.16438557 -0.06258204 0.12464311 0.07775309 -0.17094384
0.05773254 0.23981114 -0.39820656 -0.01941513 -0.15894736 -0.07663183
0.22897393 -0.0404997 0.00237953 0.15012988 0.25289607 0.04408217
0.09215517 -0.26481789 -0.51969105 -0.18874237 0.04324916 0.33723992
0.03272996 0.1023312 0.07781161 -0.27436259 0.14366916 -0.0647203
-0.27065626 -0.15682699 0.30538768 0.00759732 -0.20001976 0.18530159
-0.06843861 0.10367234 0.2666401 0.07758536 0.06988724 -0.18472718
0.04586994 -0.07091724 -0.01570967 -0.11631202 -0.27903217 -0.22006431
-0.29344234 -0.16378929 -0.01637682 -0.02759546 -0.14867654 -0.10342148
0.42888156 -0.14884272 -0.07814548 -0.20129368 -0.08100423 0.44298968
0.0096443 0.16843927 0.0403233 0.11056637 -0.14966607 0.0775136
-0.37262949 0.15971391 0.10339827 -0.08182968 -0.14464283 0.09974999
0.22623466 -0.07307606 -0.06060811 -0.08569473 0.06044295 -0.15012601
0.19709101 0.10288548 0.03458636 -0.07928915 -0.08164569 0.1086121
-0.15900649 -0.1528122 0.36258924 0.60345978 -0.26886266 0.27221575
0.15626612 -0.08666559 -0.08597019 0.02326066 0.21716742 -0.10805785
0.14552192 0.39275661 -0.2137285 0.17737752 -0.04638224 -0.09086929
0.17228992 0.45745647 0.09721016 0.22619696 0.15587658 -0.1078452
0.10545702 -0.1295034 0.15754125 0.20287803 -0.12768655 0.25871921
-0.36970127 -0.27917591 0.35536325 -0.10449084 0.29919827 0.21061943
0.02757434 -0.10462383 0.38722783 0.13090241 0.07226188 -0.20390855
-0.18720752 -0.30094409 0.04955009 0.15689072 0.06089444 0.1221517
0.29320481 0.06018539 -0.26484498 -0.08885223 0.20251615 -0.08014334
0.25534236 0.12345839 0.04113655 -0.17464676 -0.02555929 -0.24928689
-0.00726197 -0.06367166 -0.00622844 -0.16187102 0.06470232 0.00213742
-0.00922389 -0.03507412 0.15772884 0.1308917 -0.14758556 -0.03714319
-0.19372585 -0.02303672 0.12259024 0.03133943 -0.1119096 0.4193885
-0.01616556 -0.10328371 -0.25653988 -0.13578537 -0.04558701 0.14982465
0.12962459 -0.057076 0.20319712 -0.03295657 0.12831989 0.08225089
0.27632949 -0.07261054 0.16000392 0.23529905 0.12380063 -0.10244995
-0.05999006 -0.30248201 0.09638175 -0.16388389 -0.06575203 -0.13759583
0.03383766 0.03422916 0.11466212 0.17075454 -0.08441102 0.02949743
0.0929366 -0.14893182 0.31528315 -0.13083901 0.24421875 -0.32063347
-0.10847533 0.32907581 -0.24505091 0.3328152 -0.02556069 0.23488981
0.1086755 0.05432398 0.272899 0.04030114 0.24553958 -0.00683912
0.18527657 0.00568248 0.00947272 -0.03928746 -0.16926947 -0.1675348
-0.09187805 0.0302988 0.06565314 0.07030207 0.11808337 -0.11149222
-0.18275204 0.00937724 0.21726315 -0.02559427 -0.10452805 0.04012734
0.22440714 -0.23658764 -0.08710285 -0.32860765 -0.09435543 -0.10201801
0.02778483 -0.12212405 -0.15352622 -0.2402246 0.16305342 0.07084775
-0.34023714 0.06145613 0.36542648 -0.21715111 0.18380807 0.31104729
-0.06712623 -0.04920201 -0.04121515 0.10949919 0.09336475 -0.05284641
-0.07185549 -0.1092885 0.12177272 0.04907761 0.16923627 0.13010481
0.14412749 -0.38434115 0.19248633 0.10296852 -0.06169356 -0.12030595
0.00750719 -0.02544785 -0.07714268 -0.15853207 -0.1508005 -0.08645581
-0.14417176 -0.09917884 -0.05555947 -0.2346331 -0.13082129 -0.11645748
-0.23315814 0.11022738 0.19593142 0.18401168 0.00494276 -0.21829559
0.29854178 -0.12982871 0.07186505 -0.10553935 0.20121247 0.11509009
0.06089201 0.21626161 0.29866579 0.00985046 -0.13242571 -0.27146202]

```
print("Similarity:", word_vectors_model.similarity("Bundeskanzler", "Bundeskanzlerin"))
```

ergibt:

```
Similarity: 0.642387164957
```

```
positive = ["Frau", "Bundeskanzler"]
negative = ["Bundeskanzlerin"]
print("Most similar:", word_vectors_model.most_similar(positive=positive,
negative=negative, topn=1)[0])
print("Most similar cosmul:",
word_vectors_model.most_similar_cosmul(positive=positive, negative=negative, topn=1)[0])
```

ergibt:

```
Most similar: ('Mann', 0.6585341095924377)
Most similar cosmul: ('Mann', 0.9142228960990906)
```

```
print("Does not match:", word_vectors_model.doesnt_match("Frühstück Bundeskanzler Mittagessen Abendessen".split()))
```

ergibt:

```
Does not match: Bundeskanzler
```

```
your_word_vector = word_vectors_model.wv["Merkel"]
print("Word from a vector:",
word_vectors_model.most_similar(positive=[your_word_vector], topn=1)[0])
```

ergibt:

```
Word from a vector: ('Merkel', 0.9999998807907104)
```

```
most_similar_topten = word_vectors_model.most_similar(positive=["Merkel"],
topn=10)
print("Most similar top-10:")
for i, most_similar in enumerate(most_similar_topten):
    print(str(i + 1) + ":", most_similar)
```

ergibt:

```
Most similar top-10:
1: ('Kanzlerin_Merkel', 0.8945820331573486)
2: ('Merkel_CDÜ', 0.8921669125556946)
3: ('Bundeskanzlerin', 0.8700551986694336)
4: ('Kanzlerin', 0.8666472434997559)
5: ('Angela_Merkel', 0.8659625053405762)
6: ('Kanzlerin_Angela', 0.8306122422218323)
7: ('Steinbrueck', 0.8213914036750793)
8: ('Bundeskanzlerin_Angela', 0.8004235625267029)
9: ('SPD-Kanzlerkandidat_Steinbrueck', 0.7869447469711304)
10: ('CDU-Chefin', 0.7858077883720398)
```

3.x.y. Stimmungs-Analysen (Sentiment Analyse)

Definition(en): Stimmungs-Analyse (Sentiment Analyse)
--

Unter der Stimmungs-Analyse versteht man die Untersuchung und Klassifizierung einzelner Texte hinsichtlich der Einstellung / Neigung des Autor's bezüglich Produkten, Dienstleistungen und Erfahrungen.

Praxis-Beispiel → [8.x. Stimmungs-Analyse \(Sentiment Analyse\)](#)

3.x. Text- und Sprach-Erzeugung

3.x.y. Text-Erzeugung

Plauder-Programm

```
10 GOSUB 1000
20 GOSUB 2000
30 GOSUB 3000
40 Print S$
50 STOP
1000 ART=2
1010 DIM A$(ART)
1020 ART$(1)="A"
1030 ART$(2)="THE"
1100 NOUN=4
1110 DIM N$(NOUN)
1120 N$(1)="COMPUTER"
1130 N$(2)="PRINTER"
1140 N$(3)="PROGRAMM"
1150 N$(4)="BUG"
1200 ADJ=3
1210 DIM D$(ADJ)
1220 D$(1)="GROSS"
1230 D$(2)="ARM"
1240 D$(3)="BLECHERN"
1300 VERB=4
1310 DIM V$(VERB)
1320 V$(1)="SCHREIBT"
1330 V$(2)="LÄUFT"
1340 V$(3)="ENTDECKT"
1350 V$(4)="DRUCKT"
1400 ADVERB=3
1410 DIM B$(ADVERB)
1420 B$(1)="LANGSAM"
1430 B$(2)="SCHNELL"
1440 B$(3)="SCHNELL"
1500 PREP=3
1510 DIM P$(PREP)
1520 P$(1)="AN"
1530 P$(2)="ZU"
1540 P$(3)="UNTER"
1600 S$=""
1610 RETURN
2000 REM HAUPTWORT-PHRASE
2010 REM ARTIKEL
2020 S$=S$+" "+A$(INT(RND(0)*ART)+1)
2030 REM ADJEKTIV/NULL
3000
...
4000
...
```

Q: JAMES, Mike: Künstliche Intelligenz in BASIC – Arbeitsbuch für Heimcomputer.-München: moderne verlagsgesellschaft mbh, 1984; S. 114 f.

3.x.y. weitere Text-Analyse-Möglichkeiten

Prognosevarianten auf Texten:

- **Intent Classification:** Hierbei wird versucht, aus einer Nachricht die Intention des Absenders bzw. der Absenderin abzuleiten. So können Sie etwa bei vordefinierten Klassen aus einer E-Mail lesen, dass ein Kunde sein Abonnement kündigen möchte.
- **Urgency Detection:** Eine Nachricht wird in Dringlichkeiten eingestuft. Müssen Sie eine E-Mail innerhalb der nächsten Stunde oder innerhalb der nächsten Tage beantworten?
- **Hate Detection:** Ähnlich zu einer Sentiment Analyse, wobei sehr negative Kommentare als Hate erkannt werden können. Hierbei kann noch in unterschiedliche Kategorien differenziert werden (Rassismus, Sexismus, ...)
- **Aspect Classification:** Oft in Kombination mit einer Sentiment Analyse um etwa Reviews auszuwerten; was konkret war positiv, was konkret war negativ (z.B. bei einem Monitor "positiv: Qualität", "negativ: Preis")
- **Named Entity Recognition (NER):** Eher eine Klassifikationsart als ein einzelner Use Case; anstelle eines "Stempels" wird der "Textmarker" verwendet, um einzelne Stellen aus Texten zu markieren (etwa um personenbezogene Informationen in Texten zu schwärzen)
- **Document Analysis:** Neben klar ersichtlichen Freitexten (etwa aus Forenbeiträgen, E-Mails, Reviews, ...) gibt es auch zahlreiche textuelle Informationen in Dokumenten (PDF, HTML, gescannte Fotodokumente, ...). Hier fallen oft weitere Vorverarbeitungsschritte, etwa in Form einer Optical Character Recognition, hinzu.
- **Machine Translation:** Hierbei werden Sequenzen von Text in andere Sequenzen von Text übersetzt; neben Sprachübersetzungen fallen hierunter auch z.T. Übersetzungen von natürlicher Sprache in technische Abfragen. In Johannes Bachelorarbeit ging es beispielsweise darum, natürliche Sprache in SQL-Datenbankabfragen zu transformieren.

Q: <https://open.hpi.de/courses/kipraxis2021/items/62jEPY1aaq8G1fAH4eKXp9>

3.x.y. Large-scale Language Models

Abk.: LLM

im Bereich Human-Performance in Natural Language Understanding (NLU)

basiert auf der Transformer-Architektur BERT (; 2018) von DEVLIN, CHANG, LEE und TOUTANOVA

es können pre-trained Embeddings erzeugen

auf der Basis von Unmengen Mensch-gemachter Texte werden für Wort-Folgen Wahrscheinlichkeiten berechnet

diese Wahrscheinlichkeiten werden dann wieder benutzt, um wahrscheinliche Fortsetzungen / Antworten / ... auf Satz-Anfänge usw. abzuleiten

Bsp.: Wird einem LLM-System z.B. der Satz-Anfang:

Der erste Mensch auf dem Mond war ...

dann wird das System aus seiner Daten-Basis heraus "Neil Amstrong" ableiten.
Das System selbst hat nur Wahrscheinlichkeiten beachtet, der Sachverhalt selbst wird dabei nicht verstanden.

Kommen z.B. Falsch-Aussagen oder historische Lügen / Unkorrektheiten häufig genug in der ursprünglichen text-Basis vor, dann "antwortet" das LLM eben mit diesen falschen Aussagen.

Das ist eine Gefahr, die sich im praktischen Einsatz schnell zeigt, wenn Hass-Aussagen, Voreingenommenheiten oder Klischee's angeschnitten werden. Das LLM wird dann eben auch schnell rassistisch, ...

Nutz-Programme / Anwendungen müssen toxische, polarisierende, Gewalt-orientierte, illegale, pornographische und abwegige (praktisch unmögliche) Antworten verhindern.

z.B. werden / wurden für ChatGPT von openAI.com in Nigeria riesige Text-mengen hinsichtlich dieser Aspekte durchforstet und entsprechend annotiert

Nutzung im Bereich:

- Question Answering (Q&A)
- Named Entity Recognition (NER)
- Text-Klassifikation
- ...

Vorteile:

-

Nachteile:

- riesige Modell (derzeit bis Terabyte)
- extrem lange Trainings-Zeiten (sehr viele Durchläufe)
- brauchen sehr große Ressourcen (sehr teuer: x Millionen US-Dollar)
- fehlende Erklärbarkeit / Nachvollziehbarkeit
- derzeit noch sehr langsam
- kaum noch durch Nicht-Informatiker nutzbar; maximal ein begrenztes Feintuning durch Informatiker und selten auch noch durch den Nutzer (Data-Scientist)
- limitierte Eingabe-Token
- ...

3.x.y. Bewertung des Text-Layout's

algorithmischer / klassischer Ansatz

Festlegung von Parametern, die Layout charakterisieren

Entwickeln einer (Schönheit-)Funktion oder eines Algorithmus, der die einzelnen Merkmale prüft / bewertet

Nutzung des Ergebnisses für eine Klassifizierung

Realisierung durch Donald E. KNUTH und Michael F. PLASS

KI-/ML-Ansatz

Nutzung von sehr vielen Texten, bei denen von Experten die "Schönheit" des Layout's bewertet wurde

Anlernen der KI mit den Texten (Trainings-Daten) / Anpassen des ML-Modell's bis optimales Ergebnis erzielt wird / Testen mit abgespalteten Test-Daten

Nutzung des KI-System's

4. praktisches Experimentieren mittels Tensorflow-Playground

TensorflowPlayground ist eine online-Plattform (→ <https://playground.tensorflow.org>) für die Simulation von Künstlichen Neuronalen Netzwerken (KNN's) anhand einiger ausgewählter Probleme. Die Plattform wird von google zur Verfügung gestellt.

Vorteile sind hier, dass die Netze schon vorhanden sind und von uns sofort benutzt werden können. Viele Parameter lassen sich sehr einfach ändern und dann sofort simulieren. Die Netze können immer wieder neu gestartet / zurückgesetzt werden, so dass man auch statistische Untersuchungen machen kann. Ein ganz großer Vorteil ist die sehr gute Visualisierung. Neben den (Teil-)Ergebnissen sieht man auch gleich verschiedene Kennwerte des Lernens. Die Epochen-Anzahl muss nicht vorher festgelegt werden, wenn man will, kann man quasi ewig trainieren.

Hinter TensorflowPlayground steckt eine starke Hardware, die Berechnungen gefühlt zur Echtzeit durchführt. Dadurch entfällt längeres Warten, wie es noch bei der Nutzung eigener Hardware sehen werden.

neue Einführung und Modell-Beschreibung

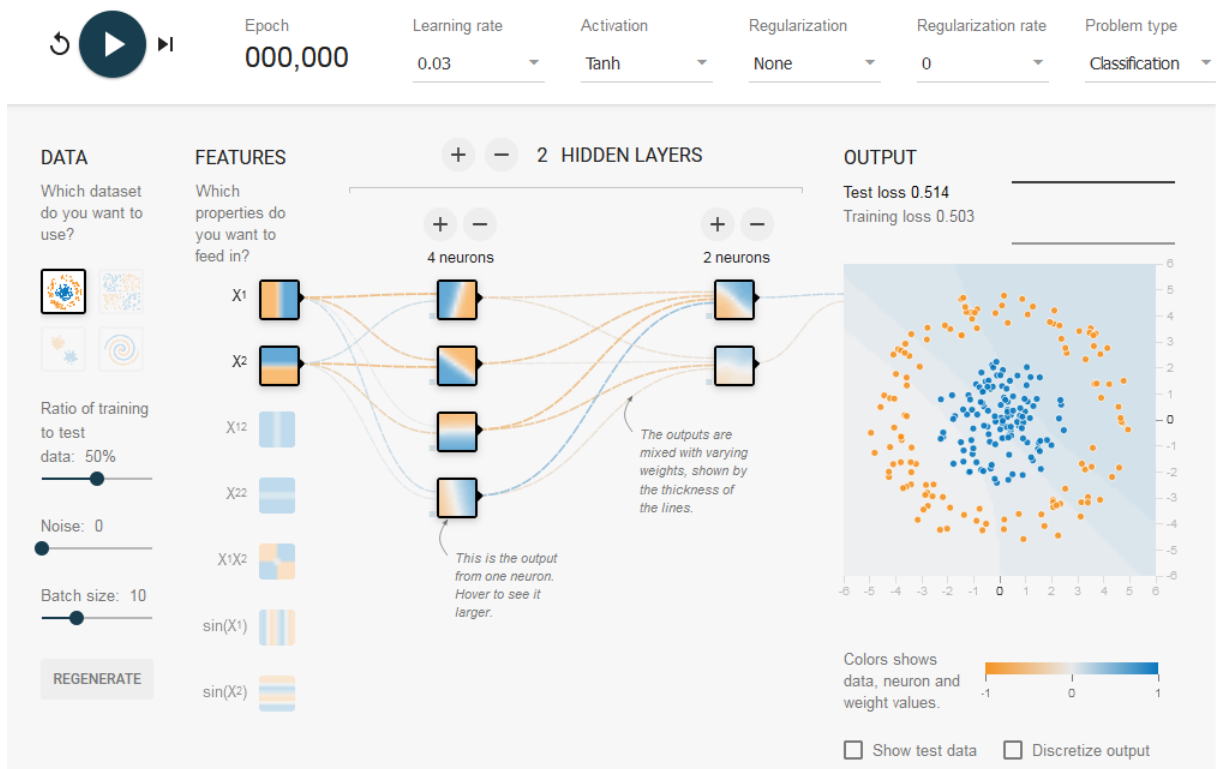
hier geht es hauptsächlich um die Lern-Rate

online Experimentieren auf Tensorflow Playground

Als Möglichkeit, um ein Verständnis für die Wirkung der Lern-Rate (Learning Rate) zu bekommen, kann man z.B. auch "Tensorflow Playground" (→ <https://playground.tensorflow.org>) benutzen

"Tensorflow Playground" ist eine online-App, mit der man Künstliche Neuronale Netzwerke erkunden kann..

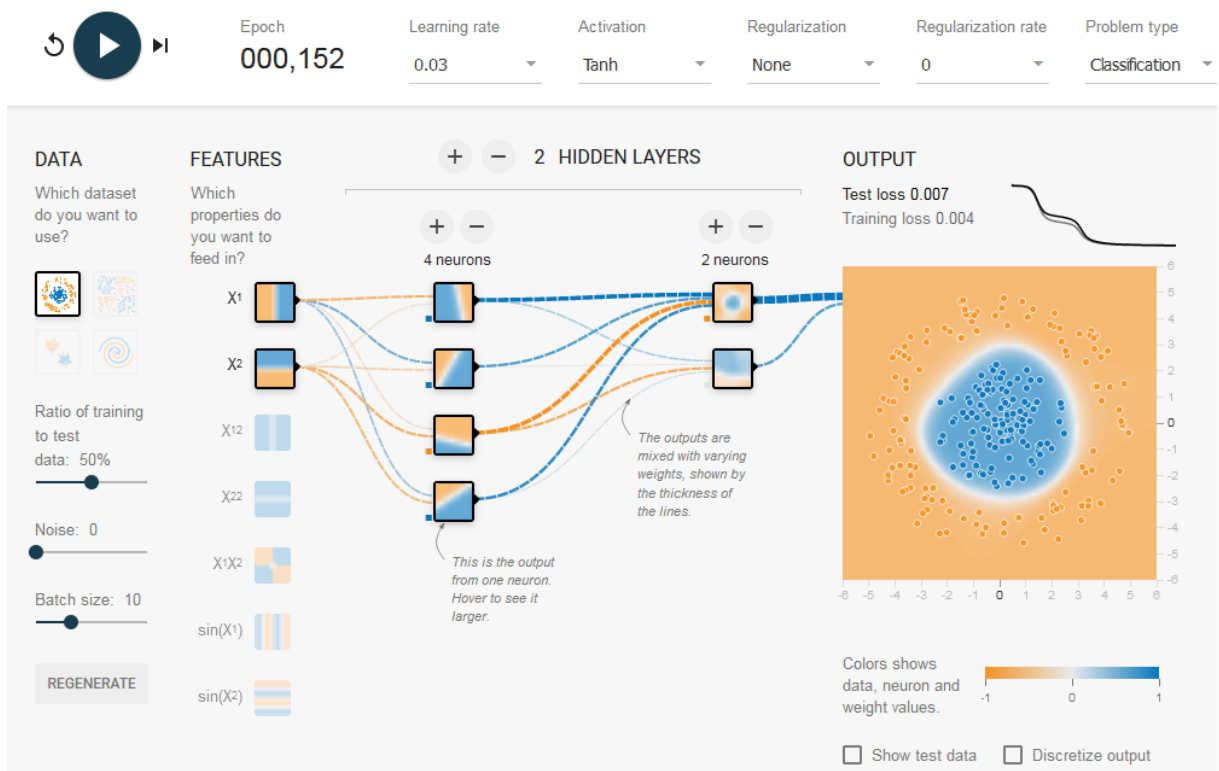
Beispiel-Problem und Ausgangslage



Wir sehen hier ein einfaches KNN bestehend aus den Eingabe-Daten (ganz links) einem "Feature Map" (s.a. →) und zwei Hidden Layer. Die zweite versteckte Schicht ist die Ausgabe-Schicht mit zwei Neuronen. Die Klassifizierungs-Aufgabe dieses Netzes besteht nämlich darin, die zentralen Punkte (blau gelabelt) von den restlichen (Ring-förmig angeordneten) Punkten zu trennen. Die Rest-Punkte sind orange gelabelt.

Die Aktivierung-Funktion ist "tanh", die wir ja schon weiter vorne (→) als eine sehr gut geeignete Funktion kennen gelernt haben.

In diesem Kapitel soll ja die Lern-Rate (Learning Rate) studiert werden. Dazu starten wir mit der Vorgabe von 0,03.

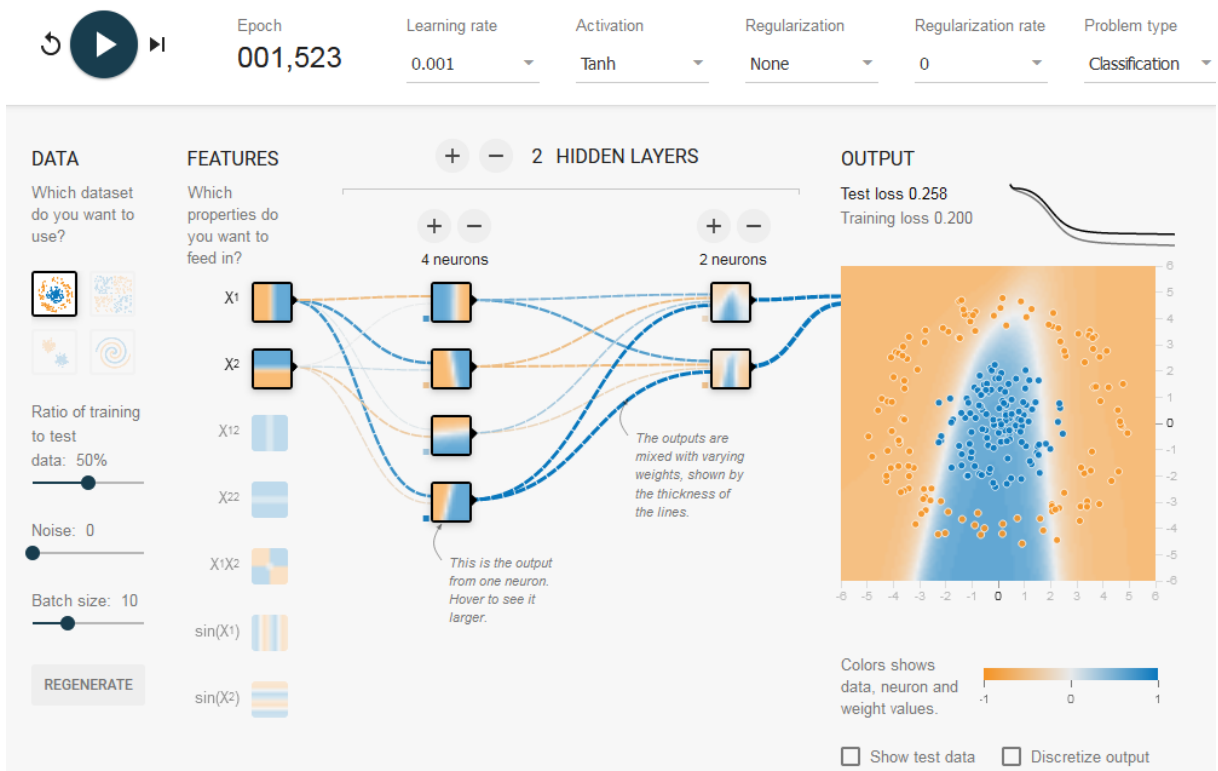


Ungefähr nach 120 Epochen (hier zur Sicherheit bis etwas über 150 weiter laufen gelassen) lernt unser Netz nichts Wesentliches mehr hinzu. Die Fehler-Funktion (oberes Diagramm im Output-Bereich) hat ihr Minimum erreicht. Die Klassifikation erfolgt wunschgemäß, alle Punkte, die auf der hellblauen Fläche liegen sind der inneren Punkt-Gruppe zugeordnet worden. Alle anderen (hier orangenen) Punkte liegen auf den hell-orangenen Alternativ-Fläche und gehören damit zur Alternativ-Klasse.

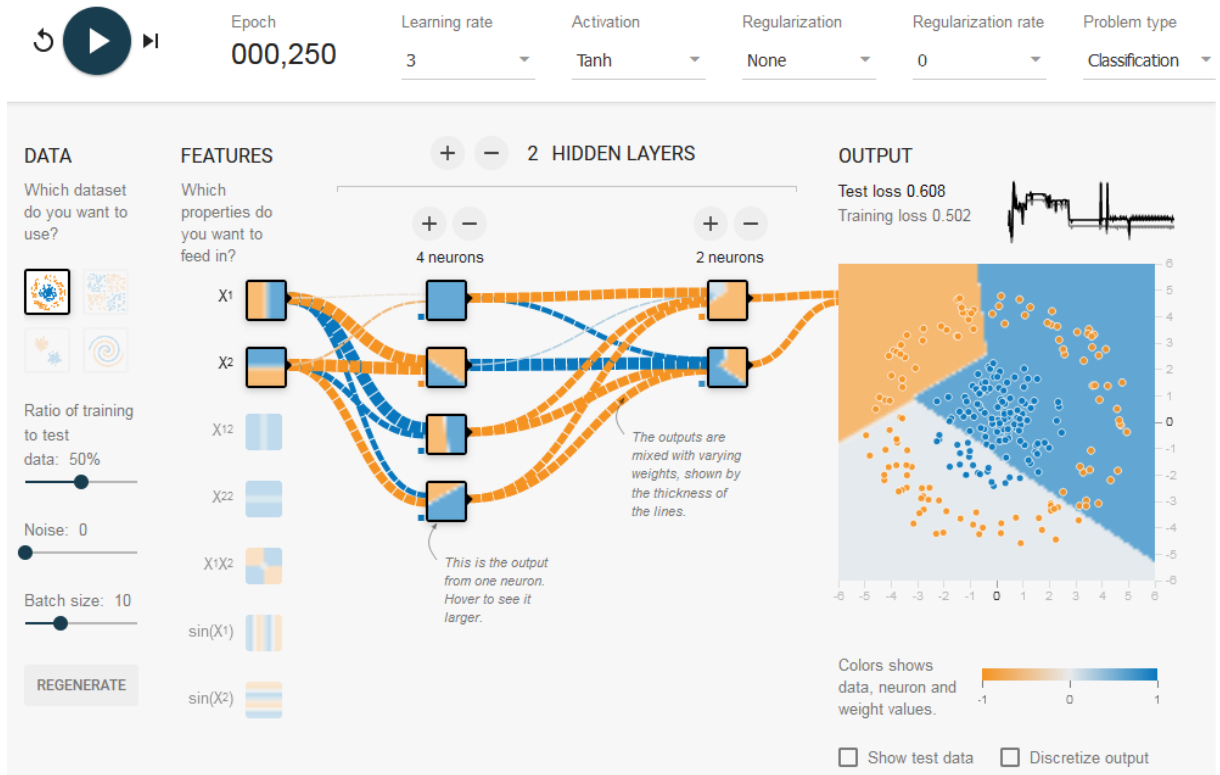
Die unterschiedliche Gewichte, die für die Neuronen gelernt wurden, erkennen wir an den verschieden starken Verbindungs-Linien zwischen den Neuronen.

Für einen zweiten Versuch wähle ich hier mal eine recht kleine Lern-Rate von 0,001.

SElbst nach gut 1'500 Epochen stellt us das Ergebnis nicht zufrieden. Das Netz lernt zu langsam.



Auch das Trenn-Ergebnis der Klassen ist schon rein optisch nicht zufriedenstellend. Testen wir nun noch eine größere Lern-Rate (als im 1. Versuch).



Q: <https://playground.tensorflow.org>

Eine Lern-Rate von 3 zeigt auch nach 250 Epochen (die im 1. Versuch ja schon lange zum Ziel geführt hätten), noch kein schönens Trenn-Ergebnis. Die Fehler-Funktion springt mehrfach hoch und runter, was auf eine Überreaktion hinweist.

5. praktisches Experimentieren mittels Python und Co

In der Praxis hat sich vor allem Python als Programmier-Umgebung für wissenschaftliches Arbeiten und ganz besonders auch im Bereich der KNN's etabliert.

Eine der Arbeits-Umgebungen für Python ist Anaconda. Es beinhaltet neben einem Installer für die notwendigen Bibliotheken / Library's / Module auch noch ein Notebook-System zum Kommunizieren mit der Arbeits-Umgebung und ein Aktualisierungs-System. Anaconda ist praktisch ein sehr leistungsfähiges pip mit diversen Zusatz-Funktionen.

5.0. Erstellen einer Arbeits-Umgebung

Prinzipiell gibt es zwei Möglichkeiten mit Künstlichen Neuronalen Netzwerken zu experimentieren. Am Einfachsten ist sicher die Nutzung einer online-Arbeits-Umgebung. Dafür muss man nichts installieren.

Eine solche online-Umgebung ist

Man benötigt einen google-Account, um sich einloggen zu können. Das genaue Vorgehen beschreiben wir im Abschnitt → .

Wem das suspekt ist, der muss die zweite Möglichkeit zum Experimentieren nutzen - eine lokale Installation. Hier benötigt man keinen google-Account, kann beruhigt auch mit eigenen Daten arbeiten und kann das Leistungs-Potential der eigenen Maschine (des eigenen PC's) austesten. Wer sich für die lokale Installation entscheidet, kann jetzt gleich bei → weiter machen.

Man kann auch eine virtuelle Maschine für die Experimente nutzen. Da man i.A. nicht über mehrere Windows-Lizenzen verfügt, bietet sich hier die Nutzung von Linux-Maschinen an. Derzeit sind besonders virtualBox und als ausgereifte Plattformen für das Benutzen von virtuellen Maschinen zu nennen.

5.0.1. Nutzung einer online-Arbeits-Umgebung

Das System nennt sich colab und ist ein google-Produkt. Für das Arbeiten mit dem online-System ist ein google-Account notwendig. Wer diesen nicht anlegen möchte, muss auf eine offline-Version - also praktisch eine lokale Installation - zurückgreifen (→).

In colab arbeitet man mit jupyter-Notebook's, die praktisch die Kommunikation mit dem colab-System dokumentieren.

Man erreicht das colab-System über → .

5.0.2. lokale Installation einer Arbeits-Umgebung / Installation von Anaconda

Anaconda setzt eine aktuelle Python-Version auf dem Rechner voraus. Dieses sollte also zuerst installiert werden. Auf die Anleitung verzichten wir hier und verweisen auf andere Tutorial's - z.B. das Python-Skript (→).

Zuerst erläutern wir hier nur die reine Installation des Anaconda-System's. Das konkrete Arbeiten damit, vor allem die Erstellung und Nutzung einer geeigneten Arbeits-Umgebung (eines sogenannten Environment's) folgt dann später → . Die Arbeit ist praktisch vom Betriebssystem unabhängig. Dies ist einer der großen Vorteile von Anaconda.

Die Installation von Anaconda ist Betriebssystem-spezifisch. Natürlich braucht man jetzt nur den Abschnitt für sein Betriebssystem lesen und gegebenenfalls umsetzen. Wem die Installation dann doch zu kompliziert erscheint, oder bei dem es garnicht läuft, der kann dann immer noch zur online-Alternative (->) wechseln. Ein anonymisierten google-Account bekommt man sicher schon hin.

5.0.2.1. Installation von Anaconda unter Windows

5.0.2.2. Installation von Anaconda unter Linux

5.0.2.3. Installation von Anaconda unter iOS / auf einem Mac

5.0.3. Arbeiten mit der Anaconda-Umgebung

5.0.3.1. Erstellung eines Environment's

Für jedes größere Programmier-Projekt - vor allem solchen mit vielen speziellen Bibliotheken - empfiehlt es sich, sich eine passende Arbeits-Umgebung zu erstellen. Dies nennt man auch Environment. Dadurch schafft man sich einen "Sand-Kasten" / eine "Spiel-Umgebung", in der man relativ unbedenklich arbeiten kann, ohne das Gesamtsystem zu beeinflussen.

5.0.3.2. Starten / Aktivieren und Beenden eines Environment's

5.0.3.3. Nutzung von jupyter-Notebook's

Mit den jupyter-Notebook's schaffen wir uns eine Möglichkeit, die Kommunikation mit dem Python- / Anaconda-System zu dokumentieren. Oft braucht man die Ein- und Ausgaben für Dokumentationen usw.

Die Kommunikation läuft über einen beliebigen Browser. Dadurch erreicht praktisch eine vollständige Unabhängigkeit vom lokalen Betriebs-System.

Die jupyter-Notebook's können gespeichert und später auch wieder geladen werden. Für die Eingaben stehen nummerierte IN-Boxen (IN-Zellen) bereit. Nach dem Bestätigen der Eingabe erfolgt die Ausgabe direkt unter der Eingabe-Box.

Praktisch wechseln sich dann die Eingabe- und Ausgabe-Boxen bzw. -Zellen immer ab. Sie können durch zusätzliche Kommentar- / Text-Boxen ergänzt werden.

Vorteile:

- Gesamt-Dokumentation eines Projekt's / ... in einer Datei
- selbst-erklärende Dokumentationen
- leichte Teilbarkeit mit anderen Nutzern
- ...

gut geeignet für Tutorial's

Google Colab's sind vergleichbare Notebook's, die aber "nur" online verfügbar und nutzbar sind.

5.0.4. Bibliothek: Pandas

spricht: *pāndas*

Vorbereitung und Verbreitung von Funktionen / Programm-Code
Daten-Management

Import der Pandas-Bibliothek

```
import pandas as pd
```

Einlesen der Daten aus einer CSV-Datei in eine Pandas-eigene Tabellen-Struktur

```
daten = pd.read_csv("datendatei.csv")
```

Ausgabe der Daten

```
daten.head()
```

Filtern von Werten eines Attribut's (lt. 1. Zeile der CSV-Datei) hier z.B. die, die kleiner als 100 sind

```
daten.loc[daten["Attribut"] < 100]
```

Anzeige dann der Zeilen mit den erfüllten Bedingungen

5.0.5. Bibliothek: matplotlib

Daten-Visualisierung

5.0.6. Bibliothek: numpy

komplexe Mathematik

5.0.7. Bibliothek: scikit-learn

maschinelles Lernen

5.0.8. Bibliothek: tensorflow

maschinelles Lernen

5.1. Erstellen von KNN's (z.B. in Anaconda oder der Python-IDE)

5.1.x. Experimente mit der Bibliothek ???

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1=nn.Conv2d(1, 32, 3, 1)
        self.conv2=nn.Conv2d(32, 64, 3, 1)
        self.dropout1=nn.Dropout2d(0.25)
        self.dropout2=nn.Dropout2d(0.5)
        self.fc1=nn.Linear(9216, 128)
        self.fc2=nn.Linear(128, 10)

    def forward(self, x):
        x=self.conv1(x)
        x=F.relu(x)
        x=self.conv2(x)
        x=F.relu(x)
        x=F.max_pool2d(x, 2)
        x=self.dropout1(x)
        x=torch.flatten(x, 1)
        x=self.fc1(x)
        x=F.relu(x)
        x=self.dropout2(x)
        x=F.relu(x)
        output=F.log_softmax(x, dim=1)
        return output
```

Q: /B2/

5.1.x. Laden des / Bearbeiten und Experimentieren mit dem MNIST-Datensatzes

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
```

```
daten=keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels)=↵
    data.load_data()
```

mit dem letzten Befehl werden die Trainings-Daten als Tupels geladen

Daten sind als NumPy-Array's verfügbar

```
print(train_images.shape)
print(test_images.shape)

(60000, 28, 28)
(10000, 28, 28)
```

Erkunden der Datenbreite für die Pixel

Suche nach Minimum und Maximum der Matrixen (hier Bilder mit Pixel-Grauwerten)

```
print(np.min(train_images))
print(np.max(train_images))

0
255
```

Verkleinern / Beschränken auf z.B. 1'000 Datensätzen zum Arbeiten / Testen von Funktionen usw.

```
arbDaten=train_images[:1000]
```

für die Benutzung in keras müssen Bild-Daten noch passend aufgearbeitet werden
keras braucht die Labels in Vektor-Form

```
anzahlKlassen=10
trainingsDatenVektor=keras.utils.to_categorical(train_labels, anzahl-
Klassen)
for i in range(20):
    print("aus Label: ",train_labels[i],
          "wird Vektor: ", trainingsDatenVektor[i])
```

ev. vor weiteren Verwendungen Daten wieder neu laden, vor allem dann, wenn Manipulationen / Experimente über diese ausgeführt wurden

Anzeige eines Bildes aus dem Trainings-Datensatz als Farb- - Diagramm, wobei die Grauwerte einer Farb-Skala zugeordnet / codiert werden

```
(train_images, train_labels), (test_images, test_labels)=↵
    data.load_data()

mpl.imshow(train_images[0])
mpl.colorbar()
mpl.show()
```

ev. ist eine Normalisierung der Daten notwendig, z.B. bei den Pixel-Grau-Werten eine Skalierung auf 256 Graustufen

```
normalisierteTrainDaten_Bilder=train_images/255
```

er Werte-Bereich wird nun auf 0 bis 1 beschränkt, diese Werte eignen sich besser für KNN's z.B. auch als Gewichte usw. usf.

ev. nochmalige Anzeige / Kontrolle

```
mpl.imshow(normalisierteTrainDaten_Bilder[0], cmap.cm.binary)
mpl.colorbar()
mpl.show()
```

cmap.cm.binary sorgt für die Graustufen-Darstellung
sollen die Daten etwas variabler gestaltet werden, um praktisch noch mehr Trainings-Daten aus den gelabelten vorgegebenen Datensätzen zu bekommen, ist das Hinzufügen von Rauschen
dazu erzeugen wir uns eine "verrauschte" Matrix und addieren diese auf das Bild

```
rauschMatrix=np.random.random((28,28))
verrauschteTrainDaten_Bilder=normalisierteTrainDaten_Bilder[0]
                                +rauschMatrix

mpl.imshow(verrauschteTrainDaten_Bilder[0], cmap=mpl.cm.binary)
mpl.show()
```

mehr Daten kann man auch durch Rotieren der Bilder oder einer Spiegelung neue Pseudo-Daten hinzufügen
ob das bei Bilder von Ziffern allgemein Sinn macht, muss im Detail geprüft werden
man könnte aber auch - je Label - unterschiedliche Operationen über die Bilder durchführen, um so noch mehr Trainings-Daten zu erfinden.

```
gedrehteNull_Bild=np.rot90(normalisierteTrainDaten_Bilder[i])

mpl.imshow(verrauschteTrainDaten_Bilder[0], cmap=mpl.cm.binary)
mpl.show()
```

die neu erzeugten Daten müssen ev. an den bestehenden Trainings-Datensatz angefügt werden

dazu wird zuerst die Umwandlung von gedrehteNull_Bild in einen passenden Vektor dann wird dieser an den vorhandenen Trainings-Datensatz angefügt

```
gedrehteNull_Bild=gedrehteNull_Bild.reshape(1,28,28)
erweiterteTrainDaten_Bilder=np.append(normalisierteTrainDaten_Bilder,
                                       gedrehteNull_Bild, axis=0)

print(erweiterteTrainDaten_Bilder.shape)

(60001, 28, 28)
```

Bibliotheken für das Training / Nutzen von KNN's

Bibliothek	Vorteile	Nachteile	Besonderheiten / Bemerkungen
TensorFlow 2.x	gute Dokumentation große Community Monitoring gut unterstützt Python-API	geringere Geschwindigkeit komplexer keine Unterstützung von AMD-Grafikkarten	OpenSource von google high-level-Schnittstelle für Keras
Keras	einfach zu verstehen leicht erweiterbar weit verbreitete und oft genutzte Schnittstelle Backend-System für TensorFlow	einfacher	oft das eigentliche Arbeitssystem

Chainer			
PyTorch		einfacher	

Keras

Kern-Konzepte

Model's

Nutzungs-Variante 1: Sequential Model

ist ein sequentielles System
linearer Stapel von Layern

Layer hinzufügen

```
model = Sequential()
model.add(...)
```

Nutzungs-Variante 2: Keras functional API

relativ offen
ermöglicht komplexe Netz-Topologien

Dense-Layer hinzufügen

```
x = Dense(größe, activation='...')
(inputs)
```

notwendige Importe

```
from keras.models import Sequential
from keras.layers import Dense, Activation
```

einfaches Netz aus einer "Dense"-Schicht und einer "Activation"-Schicht
eine Dense-Schicht ist eine dichte Schicht und hier sind 32 Neuronen vorgesehen
in einer Dense-Schicht ist jedes Neuron dieser Schicht ist mit jedem Neuron der folgenden Schicht (hier: Activation) verbunden

```
model=Sequential([Dense(32, input_dim=784), Activation('linear'), ])
```

input_dim bestimmt hier, dass der Input aus hier 784 Eingabe-Neuronen stammt
die 784 ergibt sich hier für den MNIST-Datensatz und seinen 28x28-Pixel-Bildern

alternativ geht auch:

Schritt-weises Zusammenbauen des Modell's

```
model=Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('linear'))
```

das Modell muss dann für die Nutzung kompiliert werden

```
model.compile(optimizer, loss=None, metrics=None, loss_weights=None,
              sample_weight_mode=None)
```

Training des Modell's mit der **fit()**-Funktion

```
model.fit(x=None, y=None, batch_size=None, epochs=1, verbose=1,
          callbacks=None, validation_split=0.0, ...)
```

Die **evaluate()**-Funktion dient der Festlegung der Fehler- und Genauigkeits.-Metriken

```
model.evaluate(x=None, y=None, batch_size=None, verbose=1,
              sample_weight=None, steps=None, ...)
```

für die Vorhersage der Labels / Beschriftungen für die Test-Daten dient die **predict()**-Funktion.

```
model.predict(x, batch_size=None, verbose=1, steps=None, callbacks=None,
              max_queue_size=10, ...)
```

Nutzung der function-API von Keras

Vorteil ist hier die Möglichkeit, das Modell als auch die Layer als Instanzen aufrufen zu können

zuerst Import der notwendigen Funktionen

```
from keras.layers import Input, Dense
from keras.models import Model
```

Festlegung der Eingabe

```
inputs=Input(shape=(784,))
```

hiernach erhält man einen "Tensor" zurück
nun werden die Layer-Instanzen aufgerufen
die Schichten werden nacheinander - praktisch aufeinander - aufgebaut / definiert

```
output_1=Dense(64, activation='linear')(inputs)
output_2=Dense(64, activation='linear')(output_1)
predictions=Dense(10, activation='linear')(output_2)
```

um das Modell zu konfigurieren
Modell-Konstruktor

```
model=Model(inputs=inputs,outputs=predictions)
```

Modell komilieren

```
model.compile(optimizer='', loss='', metrics=[''])
```

Training

```
model.fit(data, labels)
```


Beispiel-KNN für MNIST-Datensatz

Import der notwendigen Bibliotheken und Funktionen

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib as mpl
```

Laden des MNIST-Datensatzes

```
daten=keras.datasets.mnist
(train_images, train_labels),(test_images,test_labels)=daten.load.data()
```

Normalisieren der 255-bit-Daten auf relative Werte (zwischen 0 und 1)

```
train_images=train_images/255.0
test_images=test_images/255.0
```

Umwandeln der Labels in Ergebnis-Vektoren, was dem Ausgabe-Format des Netzes entspricht

```
ergebnisKlassen=10
trainingsVektor_Labels=keras.utils.to_categorical(train_labels,
                                                    ergebnisKlassen)
```

```
testVektor_Labels=keras.utils.to_categorical(test_labels,
                                             ergebnisKlassen)
```

Entwurf eines Netzes

```
model=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                        keras.layers.Dense(128, activation='sigmoid'),
                        keras.layers.Dense(ergebnisKlassen,
                                             activation='sigmoid')])
```

weitere Möglichkeiten die Aktivierungs-Funktion zu definieren

z.B.:

```
Dense(200, activation=tf.keras.activations.relu)
tf.keras.layers.Activation(tf.keras.activations.relu)
Dense(200, activation="relu")
tf.keras.layers.Activation("relu")
```

Kompilieren des Netzes

Optimierungs-Funktion soll die zufällige Fehler-Abstiegs-Funktion (Stochastic Gradient Descent) sein

```
sgd=keras.optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
```

```
model.compile(optimizer='sgd', loss='mean_squared_error',
              metrics=['accuracy'])
```

Training des Netzes

```
model.fit(train_images, trainingsVektor_Labels, epoche=20)
```

Evaluation des Netzes

```
evaluationFehler, evaluationAkkuratheit=model.evaluate(test_images,
                                                       testVektor_Labels)
print("Fehler: ",evaluationFehler,
      " Akkuratheit: ",evaluationAkkuratheit)
```

Experimentieren mit den Aktivierungs-Funktionen

Im eben vorgestellten KNN wurde eine sigmoide - also S-förmige - Aktivierungs-Funktion benutzt. Sie entsprach einer weich-gezeichneten Treppen-Funktion.

I.A. sind sigmoide Aktivierungs-Funktionen effektiver als harte Treppen-Funktionen.

Aber das kann und sollte man für sein Modell auch ausprobieren. Die Frage ist ja, welche Aktivierungs-Funktion bringt das beste Ergebnis. Das ist schließlich unser Haupt-Kriterium.

neben der Aktivierungs-Funktion "sigmoid" gibt es auch noch andere. Sie alle sollen kurz vorgestellt werden.

Aktivierungs-Funktion				Bemerkungen / Hinweise
sigmoid				meist schlechteste Ergebnisse, etwas geringere Geschwindigkeit
relu				oft die besten Ergebnisse (bei guter Geschwindigkeit)
linear				sehr schnell, aber auch relativ schlechte Ergebnisse
tanh				oft die besten Ergebnisse, allerdings ev. auch langsamer

Test-Set für unterschiedliche Aktivierungs-Funktionen

nach OpenHPI-Kurs

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib as mpl
```

```
(train_images, train_labels), (test_images, test_labels) = ←
keras.datasets.mnist.load_data()
```

Downloading ...

```
train_images=train_images/255.0
test_images=test_images/255.0

anzahlKlassen=10
```

```
model_no_activation=keras.Sequential([
keras.layers.Flatten(input_shape=(28,28)),
keras.layers.Dense(128), #, activation='sigmoid'),
```

```
keras.layers.Dense(anzahlKlassen), #, activation='sigmoid']])
```

```
model_relu=keras.Sequential([  
    keras.layers.Flatten(input_shape=(28,28)),  
    keras.layers.Dense(128), #, activation='relu',  
    keras.layers.Dense(anzahlKlassen), #, activation='sigmoid']])
```

```
model_linear=keras.Sequential([  
    keras.layers.Flatten(input_shape=(28,28)),  
    keras.layers.Dense(128), #, activation='linear',  
    keras.layers.Dense(anzahlKlassen), #, activation='linear']])
```

```
model_sigmoid=keras.Sequential([  
    keras.layers.Flatten(input_shape=(28,28)),  
    keras.layers.Dense(128), #, activation='sigmoid',  
    keras.layers.Dense(anzahlKlassen), #, activation='sigmoid']])
```

```
model_tanh=keras.Sequential([  
    keras.layers.Flatten(input_shape=(28,28)),  
    keras.layers.Dense(128), #, activation='tanh',  
    keras.layers.Dense(anzahlKlassen), #, activation='tanh']])
```

```
modelle=[model_relu,model_linear,model_sigmoid,model_tanh]
```

```
modell.compile(optimizer='sgd', loss='mean squared error',  
              metrics=['accuracy']) for modell in modelle
```

```
[None, None, None, None]
```

```
trainingsVektor_Labels=keras.utils.to_categorical(train_labels,  
                                                  ergebnisKlassen)  
testVektor_Labels=keras.utils.to_categorical(test_labels,  
                                             ergebnisKlassen)
```

```
epochen=15  
[  
    modell.fit(train_images,trainingsVektor_Labels,epochs=epochen,  
              verbose=True) for modell in modelle  
]
```

mit Versuch der Zeit-Erfassung (noch nicht geprüft!)

```
epochen=15  
import time  
zeiten=[]  
modellNr=0  
  
for modell in modelle:  
    zeiten.add([])  
    zeiten[modellNr].add (time())
```

```

modell.fit(train_images,trainingsVektor_Labels,epochs=epochen,
          verbose=True)
zeiten[modellNr].add (time ())
modellNr+=1

```

```

_,ergebnisRelu=model_relu=model.evaluate(test_images,
                                         testVektor_Labels)
_,ergebnisLin=model_linear=model.evaluate(test_images,
                                          testVektor_Labels)
_,ergebnisSig=model_sigmoid=model.evaluate(test_images,
                                           testVektor_Labels)
_,ergebnisTanh=model_tanh=model.evaluate(test_images,
                                          testVektor_Labels)

```

```

modellNr=0
zeiten[modellNr].add(time ())
_,ergebnisRelu=model_relu=model.evaluate(test_images,
                                         testVektor_Labels)

zeiten[modellNr].add(time ())
modellNr=1
zeiten[modellNr].add(time ())
_,ergebnisLin=model_linear=model.evaluate(test_images,
                                          testVektor_Labels)

zeiten[modellNr].add(time ())
modellNr=2
zeiten[modellNr].add(time ())
_,ergebnisSig=model_sigmoid=model.evaluate(test_images,
                                           testVektor_Labels)

zeiten[modellNr].add(time ())
modellNr=3
zeiten[modellNr].add(time ())
_,ergebnisTanh=model_tanh=model.evaluate(test_images,
                                          testVektor_Labels)

zeiten[modellNr].add(time ())

```

```

from prettytable import PrettyTable
pt=PrettyTable()
pt.field_names=["Aktivierungs-Fkt.",f"Akkuratheit nach {epochen}" ???]
pt.add_row(["tanh",ergebnisTanh])
pt.add_row(["relu",ergebnisRelu])
pt.add_row(["linear",ergebnisLin])
pt.add_row(["sigmoid",ergebnisSig])
pt.add_row(["linear",ergebnisLin])
print (pt)

```

```

from prettytable import PrettyTable
pt=PrettyTable()
pt.field_names=["Aktivierungs-Fkt.",f"Akkuratheit nach {epochen}" ???,
               "TrainingsZeit", "TestZeit"]
zeitTrain=zeiten[3][1]-zeiten[3][0]
zeitTest=zeiten[3][3]-zeiten[3][2]
pt.add_row(["tanh",ergebnisTanh,zeitTrain,zeitTest])
zeitTrain=zeiten[0][1]-zeiten[0][0]

```



```
zeitTest= zeiten[0][3]-zeiten[0][2]
pt.add_row(["relu", ergebnisRelu, zeitTrain, ZeitTest])
zeitTrain=zeiten[1][1]-zeiten[1][0]
zeitTest= zeiten[1][3]-zeiten[1][2]
pt.add_row(["linear", ergebnisLin, zeitTrain, ZeitTest])
zeitTrain=zeiten[2][1]-zeiten[2][0]
zeitTest= zeiten[2][3]-zeiten[2][2]
pt.add_row(["sigmoid", ergebnisSig, zeitTrain, ZeitTest])
print(pt)
```

für extern:

Sigmoid- oder S-Funktion
abgeflachte, weiche Treppen-Funktion

$$f(x) = \frac{1}{1+e^{-x}}$$

weiterer Arbeits-Datensatz Fashion MNIST von zalando

```
from tensorflow import keras
```

```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

Verwendung von Convolution in einem Keras-basierten KNN

```
modell=Conv2D(filters=32, kernel_size(3,3), activation='relu',
padding='valid', strides(1,1),)(modell)
```

filters definiert die Anzahl der Feature Map's
mittels kernel_size legen wir die Größe des Kernel's fest
padding besagt, wo das Kerneling beginnt (oberes Beispiel zeigt Standard-Wert)
bei padding='valid' startet das Kerneling so, das der Kernel immer vollständig innerhalb der Matrix bleibt, dadurch gehen die Rand-Elemente verloren
bei padding='same' bleiben die Zahlen der Daten-Elemente gleich, dazu wird das Kerneling immer zentriert auf das Daten-element gesetzt, die ev. fehlenden Rand-Elemente
strides besagt, wie der Kernel von Operation zu Operation weiterverschoben werden soll (oberes Beispiel zeigt Standard-Wert)
bei strides=(1,1) wird Zeilen- und Spalten-weise berechnet, dadurch bleibt die Anzahl der Daten-Elemente gleich
bei strides=(2,2) wird der Kernel immer um 2 Positionen in den Spalten und auch in den Zeilen verschoben, dadurch reduziert sich die Anzahl der Daten-Elemente auf unter die Hälfte (je Dimension)

Verwendung von Pooling-Layer in einem Keras-basierten KNN

```
MaxPooling2D(pool_size(2,2))(modell)
```

Experimentieren mit komplexen Layer-Strukturen in einem Keras-basierten KNN

Import der notwendigen Bibliotheken

```
import tensorflow as tf
from tensorflow import keras
assert tf.__version__ >="2.0"
```

```

if not tf.test.is_gpu_available():
    print("Keine GPU verfügbar! KNN-Simulation kann sehr langsam sein.")
if IS_COLAB:
    print("Wechseln zur Runtime! Dann Runtime ändern und GPU ↵
        auswählen.")

import numpy as np
import matplotlib.pyplot as mplpp
import tensorflow:datasets as tfds
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Dense, Activation, Input, Dropout, ↵
        Con2D, MaxPooling2D, Flatten
from tensorflow.keras.model import Model
from scipy.stats import reciprocal
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
%matplotlib inline # nur für jupyter-Notebooks

```

Hilfs-Funktionen für die Erzeugung von Grafiken (noch aus den jupyter-Notebook extrahieren!)

```

def druckeLernKurve(titel,x,y,yTest,ylim=0.6):
    mplpp.figure()
    mplpp.title(titel)
    axes=mplpp.gca()
    axes.set_ylim([ylim,1])
    mplpp.xlabel("Epoche")
    mplpp.ylabel("Akkuratheit")
    train_sizes=

def druckeGeschichte(titel,history,ylim=0.6)

def druckeBilder(bilder):

def druckeBilderMitBeschriftung(bilder,beschriftungen):

```

Laden der Daten und Daten-Vorbereitung

```

data, info=tfds.load("imagenette/320px", as_supervised=True,
                    with_info=True)

```

Anzeige von Informationen zum Daten-Set

```

dataset_size=info.splits['train'].num_examples
print("Datensatz-Umfang: ",dataset_size)
class_names=info.features['label'].num_names
print("Anzahl der Klassennamen: ", class_names)
n_classes=info.features['label'].num_classes
print("Anzahl der Klassen: ", n_classes)

```

```
Datensatz-Umfang: 12894
Anzahl der Klassennamen: ['... ...']
Anzahl der Klassen: 10
```

```
def vorbereitung(bild,beschriftung):
    skaliertesBild=tf.image.resize(bild, [300,300])
    return skaliertesBild, beschriftung
```

Daten-Set aufteilen in Trainings- und Test-Daten Trainings-Daten mischen

```
train_data=tfds.load('imagenette/320px', split=tfds.Split.TRAIN,
                    as_supervised=True)
test_data=tfds.load('imagenette/320px', split=tfds.Split.VALIDATION,
                   as_supervised=True)

batch_size=32
print(" der Trainingsdaten vor dem Vorbereiten;", train_data)
train_data=train_data.shuffle(1000)

... vor dem Vorbereiten: <DatasetV1Adapter shapes: ((None, None, 3), ())
```

Vorbereitungs-Pipeline (Vorbereitungs-Kanal)

```
train_data=train_data.map(vorbereitung).batch(batch_size).prefetch(1)
test_data=train_data.map(vorbereitung).batch(batch_size).prefetch(1)
print(" der Trainingsdaten nach dem Vorbereiten;", train_data)
print(" der Testdaten nach dem Vorbereiten;", test_data)

... nach dem Vorbereiten: <DatasetV1Adapter shapes: ((None, 300, 300,
3), (None, ...))
... nach dem Vorbereiten: <DatasetV1Adapter shapes: ((None, 300, 300,
3), (None, ...))
```

Anzeige von (9) Bildern mit Beschriftungen

```
druckeBilder(train_data,unbatch(),take(9))
```

1. Versuch: ein "schlechtes" Modell

Vorbereitung

Festlegung der Modell-Parameter

```
learning_rate=0.0001
momentum=0.9
dense_neurons=500
n_filters=32
first_kernel_size=(7,7)

activation='elu'
```

mehrfache Bildung von Convolution- und MaxPolloliun-Layer

```
input_layer=Input(shape=(300,300,3))
modell=Conv2D(filters=n_filters, kernel_size=(7,7),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=32, kernel_size=(2,2),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation,padding='same')(modell)
modell=MaxPooling2D((2,2))(modell)
```

die immer gleich Wahl der Filter-Größe ist ungünstig (aber das soll unser erste Modell ja auch sein → als Vergleich-Basis)

besser ist später eine steigende Vergrößerung des Filters, um immer komplexere Struktur herauszukristallisieren

echtes Klassifizieren über dichte (Hidden.) Layer (Dense Layer, vollständig verbundene Schicht)

zuerst umdimensionieren in Vektor (mit dem Flatten Layer)

dann dichte Schichten

```
modell=Flatten()(modell)
modell=Dense(dense_neurons, activation=activation)(modell)
modell=Dense(dense_neurons/2, activation='tanh')(modell)
```

zum Schluss ein Output Layer mit sovielen Neuroen, wie Ausgabe-Klassen erwartet werden

```
ausgabe=Dense(n_classes, activation='softmax')(modell)
```

```
knnModell=Model(input_layer,ausgabe)
```

Kompilieren des Netzes

```
optimizer=keras.optimizer.SGD(lr=learning_rate, momentum=momentum)
knnModell.compile(loss="square_categorical_crossentropy",
                  optimizer=optimizer,metrics=['accuracy'])
```

Anzeige des gesamten Modell's

```
knnModell.summary()
```

Training des Modell

```
history=knnModell.fit(train_data, epochs=12, validation_data=test_data)
```

Auswertung der Lern-Geschichte (über die Epochen / Trainings-Durchläufe)

```
druckeGeschichte("KNN-Perfomance", history, 0)
```

2. Versuch: ein "besseres" Modell

Veränderungen bei den Filtern für die Convolution-Layer
notwendige Änderungen farbig hervorgehoben

```
learning_rate=0.0001
momentum=0.9
dense_neurons=500
n_filters=512
first_kernel_size=(7,7)

activation='elu'

input_layer=Input(shape=(300,300,3))
modell=Conv2D(filters=256, kernel_size=(7,7),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=256, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=n_filters, kernel_size=(2,2),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=n_filters, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
'''
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
'''
modell=Conv2D(filters=n_filters, kernel_size=(3,3),
              activation=activation,padding='same')(modell)
modell=MaxPooling2D((2,2))(modell)

modell=Flatten()(modell)
modell=Dense(dense_neurons, activation=activation)(modell)
modell=Dense(dense_neurons/2, activation='tanh')(modell)

ausgabe=Dense(n_classes, activation='softmax')(modell)
knnModell=Model(input_layer,ausgabe)

optimizer=keras.optimizer.SGD(lr=learning_rate, momentum=momentum)
knnModell.compile(loss="square_categorical_crossentropy",
                  optimizer=optimizer,metrics=['accuracy'])
```

```
knnModell.summary()
history2=knnModell.fit(train_data, epochs=12, validation_data=test_data)
druckeGeschichte("KNN-Perfomance veränd. Modell",history2,0)
```

dieses Modell zeigt overfitting

Akkuratheit liegt bei rund 87 % die Überprüfung mit den Test-Daten ergibt aber nur 73 %

weitere Verbesserungen / Optimierungs-Strategien

statt:

```
modell=Conv2D(filters=32, kernel_size=(7,7),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
```

besser:

```
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation)(modell)
modell=MaxPooling2D((2,2))(modell)
```

da jetzt deutlich weniger Multiplikationen gebraucht werden
dadurch tieferer Ausbau des Netzes (mehr Schichten)

bei großen Eingabe-Daten (viele notwendige Neuronen) ist eine Reduktion der Daten-Menge
durch eine Convolution zu prüfen

```
modell=Conv2D(filters=32, kernel_size=(7,7), stride(2,2),
              activation=activation)(input_layer)
```

in einem nächsten Schritt kann man prüfen, ob die Erhöhung / ev. auch eine Verringerung der
Anzahl von Convolutional Layern (und MaxPooling Layern) eine Verbesserung bringen bzw.
keine deutliche Verschlechterung auftritt
auch die Notwendigkeit / der Effekt von MaxPooling-Schichten ist zu kontrollieren

```
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation, padding='same')(modell)
# modell=MaxPooling2D((2,2))(modell)
modell=Conv2D(filters=32, kernel_size=(3,3),
              activation=activation, padding='same')(modell)
```

durch padding='same' wird die Anzahl der Feature-Map's gleichgehalten

Experimentieren mit Fehler-Funktionen

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib as mpl

(train_images, train_labels), (test_images, test_labels) = keras.datasets←
mnist.load_data()
```

```
train_images=train_images/255.0
test_images=test_images/255.0

modellMSE=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                             keras.layers.Dense(128, activation='relu'),
                             keras.layers.Dense(10, activation='sigmoid')])

modellCCE=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                             keras.layers.Dense(128, activation='relu'),
                             keras.layers.Dense(10, activation='sigmoid')])

modellSCCE=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                              keras.layers.Dense(128, activation='relu'),
                              keras.layers.Dense(10, activation='sigmoid')])

optimizer='sgd'

modellMSE.compile(optimizer=optimizer, loss='mean_squared_error',
                  metrics=['accuracy'])

modellCCE.compile(optimizer=optimizer, loss='categorical_crossentropy',
                  metrics=['accuracy'])

modellSCCE.compile(optimizer=optimizer,
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])

epochen=15

history=[modellSCCE.fit(train_images, train_labels, epochs=epochen,
                       validation_data=(test_images, test_labels),
                       verbose=True)]

modelle=[modellMSE, modellCCE]

_train_labels=keras.utils.to_categorical(train_labels, 10)
_test_labels=keras.utils.to_categorical(test_labels, 10)

history+=[model.fit(train:images, _train_labels, epochs=epochen,
                   validation_data=(test_images, test_labels),
                   verbose=True)]

def druckeLernKurven(ylin=0.6):
    mpl.figure()
    mpl.title("Anpassung der Akkuratheit")
    axes=mpl.gca()
    axes.set_ylin([ylin, 1])
    mpl.xlabel("Epochen")
```



```

mpl.ylabel("Akkuratheit")
train_sizes=np.arange(1,1+len(history[0].history['accuracy'] ???)
mpl.ylin((0,1))
mpl.grid()
mpl.plot(train_sizes, history[0].history['accuracy'],'o-',
         color=(177/255, 6/255, 58/255), label="SCCE")
mpl.plot(train_sizes, history[1].history['accuracy'],'x-',
         color=(246/255, 168/255, 0), label="MSE")
mpl.plot(train_sizes, history[2].history['accuracy'],'--',
         color='green', label="CCE")
mpl.legend(loc="best")
return mpl

druckeLernKurven()

```

praktisch zeigen die beiden Fehler-Funktionen "categorical_crossentropy" und "sparse_categorical_crossentropy" identische Kurven für die Wahl der zu nutzenden Funktion sind die Formen der bereitgestellten Labels entscheidend

"categorical_crossentropy" erwartet ein Array mit sovielen Elementen, wie es Kategorien gibt auf den Index der richtigen Kategorie muss ein 1 stehen, alle anderen Elemente müssen 0 sein

"sparse_categorical_crossentropy" erwartet ein Array mit den Kategorie-Bezeichnungen eine eventuell notwendige Umwandlung der üblicherweise als Name vorliegenden Labels in einen für "categorical_crossentropy" geeigneten Vektor erfolgt mit:

```

_train_labels=keras.utils.to_categorical(train_labels,10)
_test_labels=keras.utils.to_categorical(test_labels,10)

```

diese wurde ja oben auch schon benutzt

Experimentieren / Setzen bei Klassen-Ungleichgewichten

praktisch eine Korrektur der Kosten-Funktion

Fehler bei unterrepräsentieren Klassen (hier: 0) wird stärker bewertet, als ein andere (normal repräsentierte) Klasse

```

modell.fit(class_weight={
    0: 3,
    1: 1 })

```

Experimentieren mit dem Optimizer

```
...
optimizer=tf.keras.optimizer.SGD(lr=0.2, momentum=0.9, decay=0.01)
...
```

Lern-Rate lr ist relativ frei wählbar

für momentum und decay haben sich die obigen Werte bewährt, können aber natürlich geändert werden und deren Wirkung analysiert werden

bei großen Datensätzen kann dann das Momentum durch weitere Neunen nach dem Komma ergänzt werden, so dass man dann mit 0,99, 0,999 usw. usf arbeitet.

Das Adam-Verfahren zur Optimierung der Gewichte (→) kann in Keras über:

```
keras.optimizer.Adam(learning_rate=0.001)
```

genutzt werden.

in einem entsprechenden Set kann man dann auch die verschiedenen Optimizer gegeneinander vergleichend ausprobieren.

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib as mpl

(train_images, train_labels), (test_images, test_labels)=keras.datasets←7
mnist.load_data()
```

```
train_images=train_images/255.0
test_images=test_images/255.0

modellSGD=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                             keras.layers.Dense(128, activation='relu'),
                             keras.layers.Dense(10, activation='sigmoid')])

modellAdam=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                              keras.layers.Dense(128, activation='relu'),
                              keras.layers.Dense(10, activation='sigmoid')])

modellRMSProp=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                                keras.layers.Dense(128, activation='relu'),
                                keras.layers.Dense(10, activation='sigmoid')])
```

```

modellAdam.compile(optimizer=optimizer='adam', loss='mean_squared_error',
                  metrics=['accuracy'])

modellRMSRrop.compile(optimizer=optimizer='rmsprop',
                    loss='mean_squared_error',
                    metrics=['accuracy'])

modellSGD.compile(optimizer='sgd', loss='mean_squared_error',
                metrics=['accuracy'])

epochen=15

history=[modellSCCE.fit(train_images, train_labels, epochs=epochen,
                      validation_data=(test_images, test_labels),
                      verbose=True)]

modelle=[modellSGD, modellRMSProp, modellAdam]

_train_labels=keras.utils.to_categorical(train_labels, 10)
_test_labels=keras.utils.to_categorical(test_labels, 10)

history=modell.fit(train:images, _train_labels, epochs=epochen,
                  validation_data=(test_images, test_labels),
                  verbose=True) for modell in modelle

def druckeLernKurven(ylin=0.6):
    mpl.figure()
    mpl.title("Anpassung der Akkuratheit")
    axes=mpl.gca()
    axes.set_ylin([ylin, 1])
    mpl.xlabel("Epochen")
    mpl.ylabel("Akkuratheit")
    train_sizes=np.arange(1, 1+len(history[0].history['accuracy'] ???)
    mpl.ylin((0, 1))
    mpl.grid()
    mpl.plot(train_sizes, history[0].history['accuracy'], 'o-',
            color=(177/255, 6/255, 58/255), label="SGD")
    mpl.plot(train_sizes, history[1].history['accuracy'], 'x-',
            color=(246/255, 168/255, 0), label="RMSProp")
    mpl.plot(train sizes, history[2].history['accuracy'], '--',
            color='green', label="Adam")
    mpl.legend(loc="best")
    return mpl

druckeLernKurven()

```

Experimentieren / Setzen von Hyper-Parametern (Lern-Rate)

```
modell.add(Dense(64, kernel_initializer='random_uniform',  
                bias_initializer='zeros'))
```

Experimentieren / Setzen von Hyper-Parametern (Gewichte- + Bias-Initialisierung)

```
modell.add(Dense(64, kernel_initializer='random_uniform',  
                bias_initializer='zeros'))
```

Experimentieren / Setzen von Hyper-Parametern (Batch-Größe)

```
modell.fit(batch_size=32, ...)
```

Experimentieren / Setzen mehrerer Hyper-Parametern ("Grid Search"-Verfahren)

verwendete Bibliothek ist hier sklearn

```
from scipy import reciprocal  
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV  
  
kerasRegressor=tf.keras.wrapper.scikit_learn.KerasRegressor(wrap_model)  
  
suchGitter = GridSearchCV(kerasRegressor, params)  
suchGitter.fit(X, y, epochs=25, validation_data=(X_test, y))
```

params ist ein Dictionary mit Angaben zu den Parametern

Experimentieren / Setzen mehrerer Hyper-Parametern ("Randomized Search"-Verfahren)

```
zufallsSuche=RandomizedSearchCV(kerasRegressor, params, n_inter=20)
zufallsSuche.fit(X, y, epochs=25, validation_data=(X_test, y_test))
```

n_inter bestimmt die Anzahl der auszuwählenden Parameter-Kombinationen für die Zufalls-Suche

5.1.x. Daten-Beschaffung per Web-Crawling

Import der notwendigen Bibliotheken

```
from selenium import webdriver
import pandas as pd
```

in Browsern kann man sich die HTML-Struktur einer Seite ansehen
besonders gut geht das mit dem Chrome-Browser
Rechts-Klick auf Seite und "Untersuchen"
dieser zeigt neben dem Seiten-Quelltext auch die Stelle an, die man z.B. gerade anklickt

jetzt muss man nach solchen Elementen suchen, die man erfassen will (z.B. die Überschriften (zu erkennen an Tag's wie H1, H2 und H3))

Beispiel openHPI-Seite

```
driver = webdriver.Chrome("./chromedriver")
driver.get("https://open.hpi.de/courses")
```

Sammeln der Kurs-Titel

```
titles = []
for course_div in driver.find_elements_by_class_name(
    "course-title"):
    anchor = course_div.find_elements_by_xpath("./h4/a")[0]
    title = anchor.get_attribute("title")
    titles.append({"title": title})
```

```
df = pd.DataFrame(titles)
```

```
df
```

Abspeichern in einer CSV-Datei

```
df.to_csv("hpi_kursdaten.csv", index=False)
```

5.2 Erstellen von KNN's in google-Colab

Kurze Einführung in Google Colab

Zeitaufwand etwa 2 Minuten

Wir haben die Notebooks, die wir in den Videos diese Woche behandeln, für euch bereitgestellt. Diese könnt ihr unter dem Reiter [Lesematerialien](#) finden und herunterladen, über den Github Link oder über die Colab Links direkt öffnen.

Hier eine kurze Anleitung zur Nutzung von Colab:

1. Mit Google Account unter colab.research.google.com einloggen: Ihr braucht hierfür einen Google Account, somit könnt ihr euch ganz einfach bei Google Colab anmelden und eure Notebooks in eurer Drive Speichern. Google erstellt hierbei automatisch einen "Colab Notebooks" Ordner in der Ablage eurer Drive.
2. In eurer Drive Ablage könnt ihr Notebooks laden oder generell in Google Colab bei Datei (zum Beispiel eine .ipynb) hochladen oder öffnen. Solltet ihr irgendwo in eurer Drive eine .ipynb-Datei liegen haben, könnt ihr diese ebenso über das von Google empfohlene Programm (Überraschung: Colab :)) öffnen.
3. Ihr findet diese Notebooks wie oben angegeben in dem Reiter Lesematerialien.
4. Öffnet ihr das Notebook mittels der Direktlinks zu Colab, handelt es sich hierbei um eine schreibgeschützte Variante. Ihr müsst daher unter "File → Save a copy in drive" klicken und speichert damit eine Kopie, die ihr selber editieren könnt, in eure Drive.
5. Mit diesem Kurs könnt ihr euren Account nicht "verbinden", lediglich unsere Dateien laden. Also behandelt Google Colab einfach als würdet ihr lokal einen Jupyter Notebook Server starten und in diesem die Notebooks bearbeiten. Die Ausführung funktioniert im Prinzip auch genauso. Es können einzelne Zellen oder auch das ganze Notebook auf einmal ausgeführt werden.

Viel Spaß beim herumspielen!

Zusammenfassung: Minimal- / Grund-Modell für Keras

```
...  
modell = keras.Sequential([keras.layers.Flatten(input_shape=(28,28),  
    keras.layers.Dense(128, activation='sigmoid'),  
    keras.layers.Dense(total_classes, activation='sigmoid')])  
  
modell.compile(optimizer='sgd', loss='mean_squared_error',  
    metrics=['accuracy'])  
  
modell.fit(train_images, train_vec_labels)  
eval_loos, eval_accuracy = modell.evaluate(test_images, test_vec_labels)  
...
```

5.x. Entwicklung einer "eigenen" Python-Umsetzungen von KNN's

basiert wesentlich auf einem Exkurs aus dem OpenHPI-Kurs "DeepLearning"
kleine Abwandlungen im Code (Eindeutschung eigener Variablen, ...)

Dieser Abschnitt dient dazu genauer zu verstehen, was bei der Nutzung der High-Level-Bibliothek Keras im Hintergrund passiert und wie man selbst eine solche Bibliothek programmieren kann / könnte.

Ziel ist es, das XOR-Problem zu lösen und den MNIST-Datensatz (→ [4.x.2. Anwendung des Netzes auf MNIST-Datensatz](#)) auszuwerten.

5.x.1. Entwicklung des Netzes am XOR-Beispiel-Problem

Vorbereitung / Vorüberlegungen

Bei XOR geht es um die Verknüpfung von zwei dualen Eingängen, wobei die Funktion nur dann eine 1 zurückliefert, wenn nur genau einer der beiden Eingänge mit einer 1 belegt ist.

Aus dieser Tabelle lassen sich schon erste Parameter des zukünftigen Netzes ableiten. Wir brauchen unbedingt zwei Eingabe-Neuronen, die x_1 und x_2 repräsentieren.

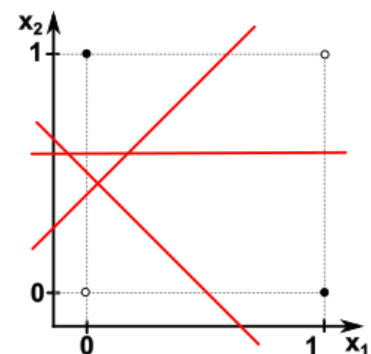
Da es zwei Ausgabe-Möglichkeiten gibt, muss unsere Netz zwei Ausgabe-Klassen entscheiden. Damit benötigen wir auch zwei Ausgabe-Neuronen.

In einem Diagramm sähe das so aus:

Die beiden Ergebnis-Klassen (die Nullen und Einsen) sind nicht durch eine einzige Gerade voneinander trennbar, was uns ja früher schon dazu geführt, zu erkennen, dass eine einzelne MCCULLOCH-PITTS-Zelle die XOR-Funktion nicht lösen kann (→ [2.2. Neuronen \(Nervenzellen\) - Grund-Bausteine neuronaler Netzwerke](#)).

XOR-Funktion

x_1	x_2	$y = x_1 \text{ XOR } x_2$
0	0	0
0	1	1
1	0	1
1	1	0

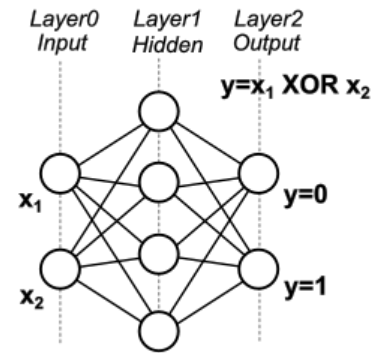


Überlegt man sich ein einfaches Netzwerk, dann könnte es so aussehen:

Wir verwenden 3 Schichten, wovon jeweils ein für Eingabe (Input) und Ausgabe (Output) stadradmäßig belegt sind.

Da wir zwei Eingabe-Werte x_1 und x_2 haben, benötigen wir in der Eingabe-Schicht 2 Neuronen.

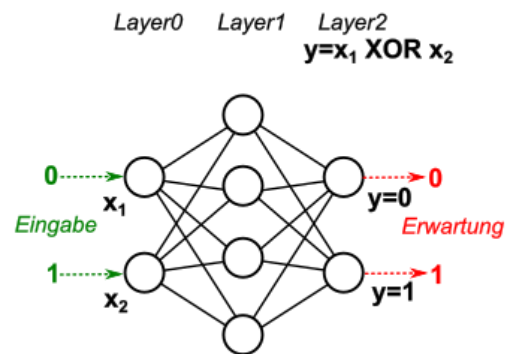
Ebenfalls zwei Neuronen werden in der Ausgabe-Schicht benötigt, da ja zwei mögliche Ausgaben (0 oder 1) für die XOR-Funktion möglich sind.



Für den Fall, dass $x_1=0$ und $x_2=1$ sind, sollte das Netz etwa so reagieren.

Die Funktionen der versteckten Schicht lassen wir zuerst einmal außen vor. Wir verwenden quasi ein Black-Box-Modell.

Die anderen Sitautionen (s.a. Funktions-Tabelle oben) lassen sich dann entsprechend ableiten.

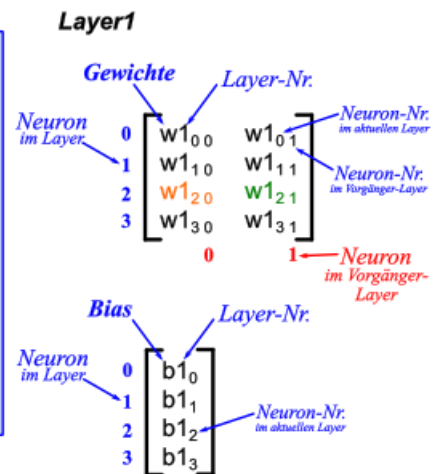
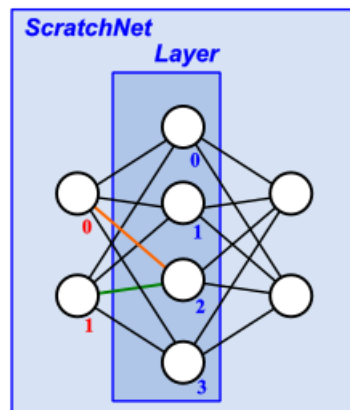


Design-Entscheidungen / Konzepte

Unser Gesamtes KNN wir als ScratchNet-Klasse verwaltet. Innerhalb dieser Klasse gibt die Klasse der Layer.

Die Gewichte und Bias-Werte werden für einen Layer in Matrizen gespeichert. Dies geschieht u.a. deshalb, weil wir später auf schnelle Matrizen-Operationen aus der Bibliothek NumPy zurückgreifen wollen. Ansonsten wäre unser System nachher einfach zu langsam und könnte kaum mit größeren Netzen und Datenset's umgehen könne.

Desweiteren konzipieren wir eine Klasse Differenzierbarer Funktionen, für die wir zwei Funktionen deklarieren (ohne sie hier zu implementieren). Diese Klasse erbt von der abc.ABC-Klasse.



```

class DifferenzierbareFunktion(abc.ABC):
    def derivative(self, net_input): # Ableitungs-Funktion
        pass
    def __call__(self, net_input): # Klasse-als-Fkt-Aufruf-Funktion
        pass

```

Die call-Funktion ist dazu da, um später die Klasse als Instanz-Funktion selbst aufrufen zu können. Später kann dann der folgende Konstrukt zu Benutzen der Funktionen verwendet werden:

```

sig = Sigmoid()
sig.derivative(np.array([3.0]))
sig(np.array([3.0]))

```

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

$$\text{sigmoid}(x)' = \text{sigmoid}(x) \cdot (1 - \text{sigmoid}(x)) = \frac{\delta \text{sigmoid}}{\delta x}$$

```

class Sigmoid(DifferenzierbareFunktion):
    def differenzieren(self, net_input):
        return self(net_input) * (1 - self(net_input))
    def __call__(self, net_input):
        return 1 / (1 + np.exp(-net_input))

```

Da später net_input ein Vektor ist, ist auch das Ergebnis ein Vektor.

```

import abc
import matplotlib.pyplot as mplpp
import tensorflow as tf
import numpy as np
import random
import time
from tqdm import tqdm, trange # Fortschritts-Anzeige
from deeplearning2020 import helpers # vordef. Diagramme ++

```

Wie würde ein Netzwerk für die XOR-Funktion in Keras aussehen?

```
def xor():
    trainInputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
    trainLabels = np.array([[0], [1], [1], [0]])

    anzKlassen = 2
    trainVektorLabels = tf.keras.utils.to_categorical(trainLabels,
                                                       anzKlassen)

    modell = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(2,1)),
        tf.keras.layers.Dense(4, activation='?'),
        tf.keras.layers.Dense(2, activation='?')])

    modell.compile(optimizer='?', loss='?', metrics=['?'])

    modell.fit(trainInputs, trainVektorLabels, epochs=20)

    valFehler, valAkkuratheit = modell.evaluate(trainInputs,
                                                trainVektorLabels,
                                                verbose=False)
    print("Validations-Fehler      : %.2f" % valFehler)
    print("Validations-Akkuratheit: %.2f" % valAkkuratheit)
```

Umsetzung / Implementierung

In unserer eigenen Implementierung sieht das nun so aus:

```
def xor():
    trainInputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
    trainLabels = np.array([[0], [1], [1], [0]])

    anzKlassen = 2
    trainVektorLabels = tf.keras.utils.to_categorical(trainLabels,
                                                       anzKlassen)

    modell = ScratchNet([FlattenLayer(input_shape=(2, 1)),
                        DenseLayer(4, aktivierungsFkt=Sigmoid()),
                        DenseLayer(2, aktivierungsFkt=Sigmoid())])

    runden = (10000, 1) # Daten-Wiederholungen statt vieler Epochen
    trainInputs = np.tile(trainInputs, runden)
    trainVektorLabels = np.tile(trainVektorLabels, runden)

    # entspricht modell.compile
    modell.uebersetzen(lernRate=0.1, fehlerFkt=QuadratFehler())

    startZeit = time.time()
    # entspricht ... modell.fit
    fehler, akkuratheiten = modell.trainieren(trainInputs,
                                              trainVektorLabels,
                                              epochen=4)

    endZeit = time.time()

    print("Trainingsdauer: {:.1f} s".format(endZeit-startZeit))
```

```

# entspricht ... modell.evaluate
valFehler, valAkkuratheit = modell.ueberpruefen(
    validation_images=trainInputs,
    validation_labels=trainVektorLabels)

print("Validations-Fehler      : %.2f" % valFehler)
print("Validations-Akkuratheit: %.2f" % valAkkuratheit)

zeigeDiagrammFehlerAkkuratheit(fehler, akkuratheiten,
                                xlabel="Epochen")

```

Was jetzt noch fehlt sind diverse Klassen, die im obigen Quell-Text schon enthalten sind. Sie müssen jetzt noch von uns implementiert werden:

```

def zeigeDiagrammFehlerAkkuratheit(fehler, akkuratheiten, xlabel):
    mplpp.plot(fehler, label="Fehler")
    mplpp.plot(akkuratheiten, label="Akkuratheit")
    mplpp.legend(loc='upper_left')
    mplpp.xlabel(xlabel)
    mplpp.ylim(top=1, bottom=0)
    mplpp.show()

```

Die Klasse `QuadratFehler` legen wir schon mal an, um sie am richtigen Platz zu haben. Mit `NotImplementedError()` lassen wir einen Fehler werfen, wenn bei einem vorzeitigen Test (anderer Funktionen / Methoden) schon mal `QuadratFehler` mit enthalten ist.

```

class QuadratFehler:
    raise NotImplementedError()

```

Die Haupt-Klasse unseres Programms nennen wir `ScratchNet`. Sie enthält die Haupt-Methoden, wie wir sie aus den Keras-Modellen kennen. Hier verwende ich mit Absicht deutsche Namen, um die "Eigenständigkeiten" der Implementierung hervorzuheben.

```

class ScratchNet:
    def __init__(self, schichten):
        raise NotImplementedError()

    # entspricht: fit
    def trainieren(self, trainBilder, trainLabels, epochen=1):
        raise NotImplementedError()

    # entspricht: predict
    def voraussagen(self, modellInputs):
        raise NotImplementedError()

    # entspricht: evaluate
    def ueberpruefen(self, validationsBilder, validationsLabels):
        raise NotImplementedError()

    # entspricht: compile
    def uebersetzen(self, lernRate=None, fehlerFkt=None):
        raise NotImplementedError()

    # entspricht: inspect
    def inspeziieren(self):
        raise NotImplementedError()

```

Für ein KNN benötigen wir mindestens zwei verschiedene Layer-Arten. Ein Flatten-Layer z.B. für die Inputs und Dense-Layer für die Zwischen-Schichten sowie die Ausgabe.

```
class DenseLayer:
    def __init__(self, neuronenzahl, tiefe=None, aktivierungsfkt=None,
                 biaswerte=None, gewichte=None,
                 vorgaengerschicht=None, nachfolgerschicht=None):
        raise NotImplementedError()

class FlattenLayer(DenseLayer):          # spezielles DenseLayer
    def __init__(self, inputform):
        raise NotImplementedError()
```

Als nächstes aktualisieren wir die init-Methode des ScratchNet - die ja bisher nur einen Fehler wirft - auf eine konkrete Funktion:

```
...
def __init__(self, schichten):
    self.lernrate=0.5
    self.kostenfkt=Quadratfehler()
    self.schichten=schichten
    for index, schicht in enumerate(self.schichten):
        schicht.vorgaengerschicht=self.schichten[index-1] ←
            if index > 0 else None
        schicht.nachfolgerschicht=self.schichten[index+1] ←
            len(self, schichten) else None
        schicht.tiefe = index
        schicht.initialisiereParameter()
...

```

Die hier neu hinzugefügte `initialisiereParameter`-Methode soll die Gewichte und Biases initialisieren. Diese Methode müssen wir nun auch noch implementieren. Dies muss jeweils in den Schicht-Klassen erfolgen, da diese sich ja z.B. hinsichtlich der Initialisierung unterscheiden. So hat unser Flatten-Layer, den wir ja als Input Layer nutzen gar keine Vorgänger-Schicht.

```
class DenseLayer:
...
    def initialisiereParameter(self):
        if self.gewichte is None:
            self.gewichte=np.random.randn(self.neuronenzahl,
                                           self.vorgaengerschicht.neuronenzahl)
        if self.biases is None:
            self.biases=np.random.randn(self.neuronenzahl,1)

class FlattenLayer:
...
    def initialisiereParameter(self):
        pass
```

Nun müssen auch noch die Konstruktoren (init-Methoden) der Schicht-Klassen erweitert / angepasst werden. Zur Sicherheit schreibe ich hier die vollständige Methode auf:

```
class DenseLayer:
    def __init__(self, neuronenZahl, tiefe=None, aktivierungsFkt=None,
                 biasWerte=None, gewichte=None,
                 vorgaengerSchicht=None, nachfolgerSchicht=None):
        self.tiefe=tiefe
        self.nachfolgerSchicht=nachfolgerSchicht
        self.vorgaengerSchicht=vorgaengerSchicht
        self.neuronenZahl=neuronenZahl
        self.aktivierungsFkt=aktivierungsFkt or Sigmoid()
        self.gewichte=gewichte
        self.biases=biases

    def initialisiere...
    ...

class FlattenLayer(DenseLayer):          # spezielles DenseLayer
    def __init__(self, inputForm):
        anzahlInputNeuronen=1
        for dim in inputForm:
            anzahlInputNeuronen*=dim
        super().__init__(neuronenAnzahl=anzahlInputNeuronen)
```

Mit `super(...)` rufen wir zum Schluß den Konstruktor der übergeordneten Klasse (hier `ScratchNet`) auf und übergeben ihm die aktuelle Neuronen-Zahl für die Input-Schicht. Damit wir nun unser Netz und diverse Parameter schon mal betrachten können, implementieren wir als nächstes die `inspeziere`-Methode (des `ScratchNet`):

```
class ScratchNet:
    ...
    def inspeziere(self):
        print("=====  
print(" # Inputs: {self.schichten[0].neuronenZahl}")
        for schicht in self.schichten:
            schicht.inspeziere()
```

In der letzten Zeile erwarten wir nun auch für die einzelnen Schichten eine `inspeziere`-Methode. Praktisch reicht aus aber eine Methode für die `Dense-Layer`, da diese die allgemeine Version darstellen und der `Flatten-Layer` (Input-Schicht) nur anders initialisiert wird, aber sonst genauso aufgebaut ist, wie ein `Dense-Layer`.


```

...
class DenseLayer:
...
    def inspizieren(self):
        print("--- Schicht: {self.tiefe} ---")
        print("        # Neuronen: {self.neuronenAnzahl}")
        for neuron in range(self.neuronenAnzahl):
            print(f"        - Neuron {neuron}:")
            if self.vorgaengerSchicht:
                for gewicht in self.gewichte[neuron]:
                    print(f"                Gewichte: {gewicht}")
                print(f"                Biases  : {self.biases[neuron][0]}")

```

Um jetzt zwischendurch mal eine Übersicht zu bekommen, können wir uns unser XOR-Modell auch schon einmal anzeigen lassen. Dzu integrieren wir - zeitweise - die inspizieren-Methode in das Modell. Durch return verhindern wir den weiteren Aufruf der dahinterstehenden Funktions-Anweisungen.

```

def xor():
...
    trainVektorLabels = np.tile(trainVektorLabels, runden)

    modell.inspizieren()
    return

    modell.uebersetzen(lernRate=0.1, fehlerFkt=QuadratFehler())
...

```

Später kann muss man das return dann entfernen. Den inspizieren-Aufruf kann man nach Belieben drin lassen.

Die Modell-Funktion muss nun auch noch einmal aufgerufen werden. Dass erfolgt - wie üblich im Haupt-Programm ganz hinten im Quell-Text:

```

...
# Hauptprogramm -- MAIN

xor()
input()

```

Die input-Anweisung am Ende ist nur notwendig, wenn das Programm direkt über die Konsole ausgeführt wird und die Konsole nicht automatisch schließen soll. Im nächsten Schritt schauen wir uns an, ob unser Netz die richtigen (Vor-)Aussagen trifft. Dazu ist die voraussagen-Methode vorgesehen. Diese bauen wir in anstatt der inspeziere-Methode ein und lassen uns das Ergebnis auch anzeigen.

```
def xor():
    ...
    trainVektorLabels = np.tile(trainVektorLabels, runden)

    voraussage=modell.voraussagen(np.array([0, 0]))
    print(voraussage)
    return

modell.uebersetzen(lernRate=0.1, fehlerFkt=QuadratFehler())
...
```

Als Eingabe verwenden wir ein kleines NumPy-Array (in diesem Format erfolgen später ja auch die Eingaben in unser Netz). Als Ergebnis sollte ein Array [1,0] herauskommen. Dabei bedeutet die erste Index-Position, wie wahrscheinlich eine Null als Ergebnis ist. Hier ist das logischerweise ja 1,0 - also 100 %. Die zweite Index-Position beschreibt die Wahrscheinlichkeit für die zweite Ergebnis-Klasse - hier die Eins. Die sollte bei 0 XOR 0 mit einer Wahrscheinlichkeit von 0 eine Eins ergeben. Hier die XOR-Funktion mit den Erwartungen für unser Netz (Zahlen verkürzt):

XOR-Funktion

x ₁	x ₂	y = x ₁ XOR x ₂	Erwartung für eine ...		erw. NumPy-Array
			0	1	
0	0	0	1,00	0,00	[1.0, 0.0]
0	1	1	0,00	1,00	[0.0, 1.0]
1	0	1	0,00	1,00	[0.0, 1.0]
1	1	0	1,00	0,00	[1.0, 0.0]

Die eigentliche Implementierung von **voraussagen** erfolgt nun. Dabei müssen wir in weiser Voraussicht für spätere größere Eingabe-Daten (z.B. Bilder) eine passende (Vektor-)Form für unser Netz bringen. Die übernimmt die - im Anschluß zu implementierende - Methode **vorbereitenInputs()**.

```
def voraussagen(self, modellInputs):
    modellInputs=self.schichten[0].vorbereitenInputs(modellInputs)
    # hier fehlt noch die restliche Implementierung
```

Das vorbereitenInputs gehört als Methode zum Flatten-Layer:

```
class FlattenLayer(DenseLayer): # spezielles DenseLayer
    ...
    def vorbereitenInputs(self, bilder, labels=None):
        vorbDaten=bilder.reshape(bilder.shape[0],self.neuronenZahl,1)
        if labels is not None:
            labels=labels.reshape(labels.shape[0],-1,1)
            return vorbDaten, labels
        return vorbDaten
    ...
```

Auch für eine Dense Layer implementieren wir diese Methode:

```
class DenseLayer:
    ...
    def vorbereitenInputs(self, bilder, labels=None):
        return bilder if labels is None else bilder, labels
    ...
```

Nun ergänzen wir die unvollständige voraussagen-Methode (von oben), so dass diese dann so aussieht:

```
def voraussagen(self, modellInputs):
    modellInputs=self.schichten[0].vorbereitenInputs(modellInputs)
    voraussage=np.zeros((modellInputs.shape[0],
                        self.schichten[-1].neuronenZahl,1))
    for i,modellInput in enumerate(modellInputs):
        voraussage[i]=self._vorwaertsSchritt(modellInput)
    return voraussage
```

Die interne Methode **vorwaertsSchritt** (der Unterstrich soll die private Verwendung anzeigen!) wird nun so programmiert:

```
def _vorwaertsSchritt(self, inputBeispiel):
    for schicht in self.schichten:
        beispiel=schicht.schichtVorwaertsSchritt(inputBeispiel)
    return beispiel
```

Die **schichtVorwaertsSchritt**-Methode muss nun wieder für jede einzelne Schicht definiert werden. Dabei werden die Gewichte mit den Ausgaben (hier: inputAktivierung) der vorhergehenden Schicht multipliziert und die Bias-Werte dazu addiert. Diese Summe ist dann der Eingangs-Wert für die Aktivierungs-Funktion (der Neuronen).

```
...
class DenseLayer:
    ...
    def schichtVorwaertsSchritt(self, inputAktivierung):
        self.schichtInputs=np.dot(self.gewichte, inputAktivierung)
                                +self.biases
        self.aktivierungsVektor=self.aktivierungsFkt(self.schichtInputs)
        return self.aktivierungsVektor
...
class FlattenLayer:
    ...
    def schichtVorwaertsSchritt(self, inputAktivierung):
        self.aktivierungsVektor=inputAktivierung
        return inputAktivierung # (???) ev. muss hier
                                # self.aktivierungsVektor zurückgegeben werden
```

Nun lässt sich die Vorab-Version unseres Programms bis kurz vor die **uebersetzen**-Methode schon ausprobieren.

```
def xor():
    ...
    trainVektorLabels = np.tile(trainVektorLabels, runden)
```

```

voraussage=modell.voraussagen(np.array([0, 0]))
print(voraussage)
return

modell.uebersetzen(lernRate=0.1, fehlerFkt=QuadratFehler())
...
# Hauptprogramm -- MAIN

xor()
input()

```

Die Ausgabe ist natürlich nicht befriedigend, liegt aber daran, dass das Netz ja mit zufälligen Gewichten und Biases gestartet und auch noch nicht trainiert wurde.

Kommen wir jetzt zur **uebersetzen**-Methode. Vorher entfernen wir aber den oben **rot** hervorgehobenen Quell-Text, der ja nur zum Testen der ersten Funktionen und zum Abbruch diente.

Gefühlt handelt es sich um das Zusammensetzen des Netzes und im Konkreten werden die übergeben Parameter in das Netz integriert. Vordefinierte Werte werden hierbei überschrieben.

```

...
def uebersetzen(self, lernRate=None, fehlerFkt=None):
    self.lernRate=lernRate or self.lernRate
    self.fehlerFkt=fehlerFkt or self.fehlerFunktion
...

```

Die **trainieren**-Methode beinhaltet jetzt die Abschnitte: Daten-Vorbereitung, Zusammenstellung des Daten-Set's und

```

...
def trainieren(self, trainBilder, trainLabels, epochen=1):
    trainBilder, trainLabels=self.schichten[0].vorbereitenInputs(↵
        trainBilder, trainLabels)
    trainDaten=list(zip(trainBilder,trainLabels))
    fehler, akkuratheit=self._sucheGradient(trainDaten,epochen)
    return fehler, akkuratheit
...

```

Die Trainings-Daten (als Set) - hier als **trainDaten** bezeichnet ist dabei eine zusammengestellt paarige Liste (deshalb zip und list) aus Tupeln.

Auch hier muss die private Methode **_sucheGradient** noch implementiert werden.

```

...
def _sucheGradient(self, trainDaten, epochen):
    fehlers, akkuratheiten=list(),list()
    for epoche in range(epochen):
        self.aktualisierenParameter(trainDaten)
        fehler, akkuratheit=(self.berechnenFehler(trainDaten),
            self._berechnenAkkuratheit(trainDaten),)
        fehlers.append(fehler)
        akkuratheiten.append(akkuratheit)
        print(" -- Epoche: {0} Fehler: {1:.3f} Akkuratheit: ↵
            {2:.2f}".format(epochen+1, fehler, akkuratheit))
    return fehlers, akkuratheiten
...

```

Es folgt die Implementierung der eben vordeklarierten Methoden `aktualisierenParameter`, `berechnenFehler` und `berechnenAkkuratheit`. Sie gehören zur Klasse `ScratchNet`. Die Methode `aktualisierenParameter` beinhaltet den Backpropagation-Schritt.

```
...
def _sucheGradient(self, trainDaten, epochen):
...
def _aktualisierenParameter(self, inputBeispiele):
    gewichteGradient=[np.zeros(schicht.gewichte.shape ↵
        for schicht in self.schichten[1:])
    biasesGradient=[np.zeros(schicht.biases.shape ↵
        for schicht in self.schichten[1:])
    for beispiel in inputBeispiele: # Summierung
        beispielGewichtsGradient, beispielBiasGradient=
            self._backpropagieren(beispiel)
        gewichtsGradient=np.add(gewichtsGradient, ↵
            beispielGewichtsGradient)
        biasGradient=np.add(biasGradient, beispielBiasGradient)
    # Mitteln
    for schicht,schichtGewichtsGradient, schichtBiasGradient ↵
        in zip(self.schichten[1:], gewichtsGradient, biasGradient):
        schicht.gewichte+=(self.lernRate*schichtGewichtsGradient/↵
            len(inputBeispiele))
        schicht.biases+=(self.lernRate*schichtBiasesGradient/↵
            len(inputBeispiele))

def _backpropagieren(self, trainBeispiele):
    trainInput, trainOutput=trainBeispiel
    self._vorwertsSchritt(trainInput)
    # Gradient des letzten Layer lt. Kosten-Funktion
    gradienten=self.schichten[-1].berechnenKostenGradienten(
        trainOutput, kostenFkt=self.kostenFkt)
    # Backpropagieren der Gradienten des letzten Hidden Layers
    for schicht in reversed(self.schichten[1:-1]):
        gradienten=schicht.rueckwaertsSchritte(gradienten)
    # Akkumulieren aller Gradienten der Backpropagation
    gewichtsGradienten=[schicht.gewichtsGradienten ↵
        for schicht in self.schichten[1:-1]]
    biasesGradienten=[schicht.biasesGradienten ↵
        for schicht in self.schichten[1:-1]]
    return gewichtsGradient, biasGradient

def berechnenFehler(self, inputBeispiele):
    raise NotImplementedError()

def _berechnenAkkuratheit(self, inputBeispiele):
    raise NotImplementedError()
...

```

Die Backpropagation ist ein schon recht komplexes mathematisches Unterfangen. Da ich hier nicht annehme, dass alle Leser der höheren Mathematik von Differenzieren sowie der Matrizen-Operationen wirklich Herr sind, nehmen wir hier die Implementierung einfach so hin. Wer dahingehend ein Interesse hat, kann sich die Grundlagen im Video (→ `neuralnets2020-w3-e1 ScratchNet.mp4`) zur Implementierung des `ScratchNet` ab Zeitpunkt 46:10 ansehen. Die nachfolgenden - etwas "spezielleren" Funktionen werden dann im Video → `neuralnets2020-3-ScratchNet2_HD-hd.mp4` besprochen. Die Backpropagation besteht dabei aus drei Hauptschritten: Berechnen des vorwärtsSchrittes, Berechnen der Gradienten im letzten Layer und dann dem zurück-propagieren des Gradienten durch das Netz. Die soeben benutzten Methoden `berechnenKostenGradienten` und `rueckwaertsSchritte` implementieren wir unter dem `Dense Layer`. Für den `Input Layer` (`Flatten Layer`) ist eine Implementierung nicht notwendig, weil für diese keine Backpropagation möglich ist.

```

...
class DenseLayer:
...
    def berechnenKostenGradienten(self, labelVektor, kostenFkt):
        kostenGradienten=kostenFkt.differenzieren(
            self.aktivierungsVektor, labelVektor)
            * self.aktivierungsFkt.differenzieren(self.layerInputs)
        self.aktualisierenInputGradienten(kostenGradienten)
        return kostenGradienten

    def rueckwaertsSchritte(self, vorgaengerInputGradienten):
        neueInputGradienten=np.dot(
            self.nachfolgerSchicht.gewichte.transpose(),
            vorgaengerInputGradienten)
            * self.aktivierungsFkt.differenzieren(self.layerInputs)
        self.aktualisierenInputGradienten(kostenGradienten)
        return neueInputGradienten

    def _aktualisierenSchichtGradienten(self, inputGradienten):
        self.biasGradienten=inputGradienten
        self.gewichtsGradienten=np.dot(inputGradienten,
            self.vorgaengerSchicht.aktivierungsVektor.transpose())
...

```

prüfen layerInputs???

Die notwendige Klasse zur Fehler-Funktion QuadratFehler () wird so implementiert.

```

class QuadratFehler(DifferenzierbareFunktion):
    def differenzieren(self, ziel, aktuell):
        return aktuell-ziel

    def __call__(self, ziel, aktuell):
        return 0.5*np.sum((ziel-aktuell)**2)

```

abklären

```

??? Fragment für Backpropagation ???
        self.gewichtsGradienten=np.dot(inputGradienten,
            self.vorgaengerSchicht.aktivierungsVektor.transpose())

```

Als nächstes implementieren wir die ueberpruefen-Methode.

```

class ScratchNet:
...
    def ueberpruefen(self, validationsBilder, validationsLabels):
        validationsBilder, validationsLabels=self.schichten[0].↵
            vorbereitenInputs(validationsBilder, validationsLabels)
        validationsDaten=list(zip(validationsBilder, validationsLabels))
        return (self._berechnenFehler(validationsDaten),
            self._berechnenAkkuratheit(validationsDaten),)
...

```

Bleiben noch die benutzten Methoden `_berechnenFehler` und `_berechnenAkkuratheit`.

```

class ScratchNet:
...
    def _berechnenFehler(self, inputBeispiele):
        gesFehler=0.0
        for beispiel in inputBeispiele:
            bild,labelVektor = beispiel
            outputAktivierungen=self.vorwaertsSchritt(bild)
            gesFehler+=self.kostenFkt(labelVektor, outputAktivierungen)
        return gesFehler / len(inputBeispiele)

    def _berechnenAkkuratheit(self, inputBeispiele):
        ergebnisse=(np.argmax(self.vorwaertsSchritt(bild)),
                    np.argmax(inspeziertesLabel)) ↵
                    for bild, inspeziertesLabel in inputLabel)
        passendeIndexes=sum(int(x==y) for (x,y) in ergebnisse)
        return passendeIndexes / len(inputBeispiele)
...

```

Nun ist es endlich an der Zeit unser Netz auszuprobieren und dann damit zu experimentieren. Der nachfolgende Code kann gerne noch erweitert werden, das Grundgerüst hatten wir aber schon implementiert.

```

...
# Hauptprogramm -- MAIN

print(">>>>> ScratchNet: XOR <<<<<<<<")
print("... wurde gestartet! Bitte warten!")
xor()
input("Programm beenden mit Enter")

```

Diese Netz zeichnet sich nur durch eine schleppende Verbesserung der Kosten-Funktion sowie einer sehr schlechten Akkuratheit von 0,5 aus. Dies bedeutet praktisch, dass unser Netz nicht wirklich lernt, sondern immer noch im Bereich des zufälligen Voraussagen liegt.

Auch Versuche mit mehr Epochen und größeren Lern-Raten verbessern / verändern die Ergebnisse nicht wirklich.

Eine Ursache ist das bisherige Verfahren für die Anpassung der Gewichte und Biases. Sie werden erst nach dem Durchlauf des gesamten Trainings-Set's angepasst. Günstiger ist eine Anpassung schon nach einer kleineren Gruppe von Trainings-Daten. Diese Gruppen hatte wir Batch'es genannt (→). Damit ist dann auch eine Umstellung der Anpassung der Gewichte und Biases nach dem Durchlauf eines Batch'es verbunden. Wir verwenden dann auch nun das Zufällige Gradienten-Abstiegs-Verfahren (Stochastic Gradient Descent).

Verbesserungen / Erweiterungen

Ab sofort wollen wir es unserem Netz auch ermöglichen mit einer wählbaren Batch-Größe zu arbeiten.

```
...
startZeit = time.time()
# entspricht ... modell.fit
fehler, akkuratheiten = modell.trainieren(trainInputs,
                                          trainVektorLabels,
                                          epochen=4, batchGroesse=4)
endZeit = time.time()
...
```

Die bisherige Implementierung der trainieren-Methode wird um die Batch-Größe erweitert und nun auch als geändertes Verfahren das stochastische Gradienten-Abstiegs-Verfahren (hier: `sucheZufaelligenGradient`) eingestellt.

```
...
def trainieren(self, trainBilder, trainLabels, epochen=1,
               batchGroesse=1):
    trainBilder, trainLabels=self.schichten[0].vorbereitenInputs(↵
        trainBilder, trainLabels)
    trainDaten=list(zip(trainBilder,trainLabels))
    fehler, akkuratheit=self._sucheZufaelligenGradient(trainDaten,
                                                       epochen, batchGroesse)
    return fehler, akkuratheit
...
```

Die Methode `sucheZufaelligenGradient` muss denn nun auch von uns programmiert werden.

```
...
def _sucheGradient(self, trainDaten, epochen):
    ...

def _sucheZufaelligenGradient(self, trainDaten, epochen=1,
                              batchGroesse=1, letztesMittel=None):
    fehlers, akkuratheiten=list(),list()
    trainSetGroesse=len(trainDaten)
    letztesMittel=letztesMittel or
        int(0.10 * trainSetGroesse / batchGroesse)
    laufenderFehler, laufendeAkkuratheit = [],[]
    for epoche in range(epochen):
        random.shuffle(trainDaten)
        with tqdm(total=trainSetGroesse as progress:
            for t in range(0, trainSetGroesse, batchGroesse):
                batch=trainDaten[t:t+batchGroesse]
                self._aktualisierenParameter(batch)
                fehler, akkuratheit=(self._berechnenFehler(batch),
                                     self._berechnenAkkuratheit(batch),)
                laufenderFehler=( [fehler]+laufenderFehler) ↵
                    [:letztesMittel]
                laufendeAkkuratheit=( [akkuratheit]
                                       +laufendeAkkuratheit) [ :letztesMittel]
                laufenderFehlerMittel=np.average(laufenderFehler)
                laufendeAkkuratheitMittel=
```

```
        np.average(laufendeAkkuratheit)
    fehlers.append(laufenderFehlerMittel)
    akkuratheiten.append(laufendeAkkuratheitMittel)
    progress.set_description(f"Epoche: {epoche+1}")
    progress.set_postfix(Fehler="{0:.3f}".↵
                        format(laufenderFehlerMittel),
                        Akkuratheit="{0:.2f}".↵
                        format(laufendeAkkuratheitMittel),)
    progress.update(len(batch))
return fehlers, akkuratheiten
...
```

Man hätte auch die veraltete Methode `sucheGradient` direkt umprogrammieren können. Sie wird im neuen System nicht mehr gebraucht / ausgerufen. Ich behalte sie weiter in meinem Portfolio, um ev. später nochmal zurückschalten zu können.

Nach dem Start des Programm sehen wir nun Laufbalken-Anzeigen mit den Datensatz-Durchläufen in den einzelnen Epochen.

Diese Ergebnisse befriedigen natürlich richtig. Praktisch haben wir kein Fehler mehr und die Akkuratheit liegt bei 100 % - insgesamt sehr gute Werte.

5.x.2. Anwendung des Netzes auf MNIST-Datensatz

Wir erstellen uns alternativ oder zusätzlich zum XOR-Modell nun ein MNIST-Modell:

```
def mnist():
    (trainBilder, trainLabels), testBilder, testLabels = \
        tf.keras.datasets.mnist.load_data()
    trainBilder = trainBilder / 255.0
    testBilder = testBilder / 255.0
    klassenZahl = 10
    trainVektorLabels = tf.keras.utils.to_categorical(trainLabels,
                                                    klassenZahl)
    testVektorLabels = tf.keras.utils.to_categorical(testLabels,
                                                    klassenZahl)
    modell = ScratchNet([FlattenLayer(input_shape=(28, 28)),
                        DenseLayer(128, aktivierung=Sigmoid()),
                        DenseLayer(klassenZahl, aktivierung=Sigmoid()),])
    modell.uebersetzen(lernRate=5.0, fehler=QuadratFehler())
    startZeit = time.time()
    fehler, akkuratheiten = modell.trainieren(trainBilder,
                                             trainVektorLabels,
                                             epochen=15, batchGroesse=64)

    endZeit = time.time()

    print("Trainingsdauer: {:.1f} s".format(endZeit - startZeit))
    valFehler, valAkkuratheit = modell.ueberpruefen(
        validationsBilder=testBilder,
        validationsLabels=testVektorLabels)
    print("Validations-Fehler      : %.2f" % valFehler)
    print("Validations-Akkuratheit: %.2f" % valAkkuratheit)

    zeigeDiagrammFehlerAkkuratheit(fehler, akkuratheiten,
                                   xLabel="Epochen")
```

Zum Schluss muss das Haupt-Programm angepasst werden, damit auch wirklich das MNIST-Modell benutzt wird.

```
# Hauptprogramm -- MAIN

print(">>>>>> ScatchNet: MNIST <<<<<<<<")
print("... wurde gestartet! Bitte warten!")
mnist()
input("Programm beenden mit Enter")
```

5.x. eine weitere Implementierung eines KNN in Python

```
from collections import Counter
from functools import partial
from linear_algebra import dot
import math, random
import matplotlib
import matplotlib.pyplot as plt

def step_function(x):
    return 1 if x >= 0 else 0

def perceptron_output(weights, bias, x):
    """returns 1 if the perceptron 'fires', 0 if not"""
    return step_function(dot(weights, x) + bias)

def sigmoid(t):
    return 1 / (1 + math.exp(-t))

def neuron_output(weights, inputs):
    return sigmoid(dot(weights, inputs))

def feed_forward(neural_network, input_vector):
    """takes in a neural network (represented as a list of lists of lists of
    weights)
    and returns the output from forward-propagating the input"""

    outputs = []

    for layer in neural_network:

        input_with_bias = input_vector + [1]           # add a bias input
        output = [neuron_output(neuron, input_with_bias) # compute the output
                  for neuron in layer]                 # for this layer
        outputs.append(output)                          # and remember it

        # the input to the next layer is the output of this one
        input_vector = output

    return outputs

def backpropagate(network, input_vector, target):

    hidden_outputs, outputs = feed_forward(network, input_vector)

    # the output * (1 - output) is from the derivative of sigmoid
    output_deltas = [output * (1 - output) * (output - target[i])
                     for i, output in enumerate(outputs)]

    # adjust weights for output layer (network[-1])
    for i, output_neuron in enumerate(network[-1]):
        for j, hidden_output in enumerate(hidden_outputs + [1]):
            output_neuron[j] -= output_deltas[i] * hidden_output

    # back-propagate errors to hidden layer
    hidden_deltas = [hidden_output * (1 - hidden_output) *
                     dot(output_deltas, [n[i] for n in network[-1]])
                     for i, hidden_output in enumerate(hidden_outputs)]

    # adjust weights for hidden layer (network[0])
    for i, hidden_neuron in enumerate(network[0]):
        for j, input in enumerate(input_vector + [1]):
            hidden_neuron[j] -= hidden_deltas[i] * input

def patch(x, y, hatch, color):
```

```

"""return a matplotlib 'patch' object with the specified
location, crosshatch pattern, and color"""
return matplotlib.patches.Rectangle((x - 0.5, y - 0.5), 1, 1,
                                     hatch=hatch, fill=False, color=color)

def show_weights(neuron_idx):
    weights = network[0][neuron_idx]
    abs_weights = [abs(weight) for weight in weights]

    grid = [abs_weights[row:(row+5)] # turn the weights into a 5x5 grid
             for row in range(0,25,5)] # [weights[0:5], ..., weights[20:25]]

    ax = plt.gca() # to use hatching, we'll need the axis

    ax.imshow(grid, # here same as plt.imshow
              cmap=matplotlib.cm.binary, # use white-black color scale
              interpolation='none') # plot blocks as blocks

    # cross-hatch the negative weights
    for i in range(5): # row
        for j in range(5): # column
            if weights[5*i + j] < 0: # row i, column j = weights[5*i + j]
                # add black and white hatches, so visible whether dark or light
                ax.add_patch(patch(j, i, '/', "white"))
                ax.add_patch(patch(j, i, '\\', "black"))

    plt.show()

if __name__ == "__main__":
    raw_digits = [
        """11111
          1...1
          1...1
          1...1
          11111""",
        """..1..
          ..1..
          ..1..
          ..1..
          ..1..""",
        """11111
          ....1
          11111
          1....
          11111""",
        """11111
          ....1
          11111
          ....1
          11111""",
        """1...1
          1...1
          11111
          ....1
          ....1""",
        """11111
          1....
          11111
          ....1
          11111""",
        """11111
          1....
          11111
          1...1
          1...1
    ]

```

```

        11111""",
        """11111
        ....1
        ....1
        ....1
        ....1""",
        """11111
        1...1
        11111
        1...1
        11111""",
        """11111
        1...1
        11111
        ....1
        11111"""]

def make_digit(raw_digit):
    return [1 if c == '1' else 0
            for row in raw_digit.split("\n")
            for c in row.strip()]

inputs = list(map(make_digit, raw_digits))

targets = [[1 if i == j else 0 for i in range(10)]
           for j in range(10)]

random.seed(0) # to get repeatable results
input_size = 25 # each input is a vector of length 25
num_hidden = 5 # we'll have 5 neurons in the hidden layer
output_size = 10 # we need 10 outputs for each input

# each hidden neuron has one weight per input, plus a bias weight
hidden_layer = [[random.random() for __ in range(input_size + 1)]
                 for __ in range(num_hidden)]

# each output neuron has one weight per hidden neuron, plus a bias weight
output_layer = [[random.random() for __ in range(num_hidden + 1)]
                 for __ in range(output_size)]

# the network starts out with random weights
network = [hidden_layer, output_layer]

# 10,000 iterations seems enough to converge
for __ in range(10000):
    for input_vector, target_vector in zip(inputs, targets):
        backpropagate(network, input_vector, target_vector)

def predict(input):
    return feed_forward(network, input)[-1]

for i, input in enumerate(inputs):
    outputs = predict(input)
    print(i, [round(p,2) for p in outputs])

print(""".@@@.
...@@
..@@.
...@@
.@@@.""")
print([round(x, 2) for x in
       predict( [0,1,1,1,0, # .@@@.
                 0,0,0,1,1, # ...@@
                 0,0,1,1,0, # ..@@.
                 0,0,0,1,1, # ...@@
                 0,1,1,1,0])] # .@@@.

print()

```

```
print(""".@@@.
@..@@
.@@@.
@..@@
.@@@.""")
print([round(x, 2) for x in
predict( [0,1,1,1,0, # .@@@.
1,0,0,1,1, # @..@@
0,1,1,1,0, # .@@@.
1,0,0,1,1, # @..@@
0,1,1,1,0])]) # .@@@.

print()
```

Q: <https://github.com/joelgrus/data-science-from-scratch/find/master>

!!! diverse Kommentare in der Material-Sammlung:
"Material DL KNN mit Snap+Scratch.docx"

5.x. Erstellen eines TensorFlow-basierten KNN

Q: <https://www.codeflow.site/de/article/how-to-build-a-neural-network-to-recognize-handwritten-digits-with-tensorflow#>

auch hier Erkennung von Ziffern im MNIST-Datensatz

5.x. KNN's mit Torch

→ <https://pytorch.org/>

von Facebook entwickelt und bereitgestellt

```
import torch
dtype = torch.float
device = torch.device("cpu") # Hiermit werden alle Berechnungen auf der
CPU ausgeführt
# device = torch.device("cuda:0") # Hiermit werden alle Berechnungen auf
der GPU ausgeführt

# Erstellung eines Tensors und Befüllung des Tensors mit Zufallszahlen
a = torch.randn(2,3,device=device,dtype=dtype)
print(a) # Ausgabe des Tensors A
# Ausgabe: tensor([[ -1.1884,  0.8498, -1.7129],
#                  [-0.8816,  0.1944,  0.5847]])

# Erstellung eines Tensors und Befüllung des Tensors mit Zufallszahlen
b = torch.randn(2,3,device=device,dtype=dtype)
print(b) # Ausgabe des Tensors B
# Ausgabe: tensor([[ 0.7178, -0.8453, -1.3403],
#                  [ 1.3262,  1.1512, -1.7070]])

print(a*b) # Ausgabe einer Multiplikation der beiden Tensoren
# Ausgabe: tensor([[ -0.8530, -0.7183,  2.58],
#                  [-1.1692,  0.2238, -0.9981]])

print(a.sum()) # Ausgabe der Summe aller Elemente im Tensor A
# Ausgabe: tensor(-2.1540)

print(a[1,2]) # Ausgabe des Elements in der dritten Spalte der zweiten
Zeile
# Ausgabe: tensor(0.5847)

print(a.min()) # Ausgabe des Minimumwertes im Tensor A
# Ausgabe: tensor(-1.7129)
```

Q: <https://de.wikipedia.org/wiki/PyTorch>

5.x. Projekte mit jupyter-Notebook's

5.x.0. Einführung in jupyter-Notebook

Erweiterung zu einem Python-System

ersetzt die klassische
Anzeige und Arbeiten passiert im Browser
dadurch mehr Möglichkeiten, als in einer Konsole

im Präsentations-Modus lassen sich einzelne Zellen (Boxen) als "Folien" anzeigen
auch in diesem Modus lassen sich die einzelnen Eingabe-Zellen ausführen und die Ausgaben werden dann entsprechend angezeigt

5.x.0.1. Installation / Vorbereitung

5.x.0.2. Bedienung eines jupyter-Notebook's

Ausführen einer Zelle mit [⇧] + [ENTER]

alle Ausgaben löschen – um sie z.B. neu zu berechnen – benutzt man "Cell" "all Outputs" "clear"
die internen Variablen-Zuweisungen und – Werte bleiben allerdings enthalten

kompleter Neustart eines Notebook's
"Kernel" "Restart" "clear all outputs"

5.x. Projekte mit Google Colab's

Vorteile:

- keine Installation notwendig
- von überall nutzbar (nicht auf den eigenen Rechner beschränkt)

-
- man kann auch von einem Leistungs-ärmeren Rechner auf größere externe Ressourcen zugreifen
 - leicht teilbar
 - leichte Text-Gestaltung / -Formatierung mit Markup's möglich
 - ...

Nachteile:

- Speichern nur auf einem google-Drive möglich
- ...

unter "Bearbeiten" "Einstellungen" können spezielle – natürlich nur virtuelle – Hardware-Ergänzungen eingestellt werden

5.x.0. Einführung in Google Colab's

5.x.0.1. Bedienung eines Google Colab's

Ausführen einer Zelle mit [⇧] + [ENTER]

5.x.0.1. Einstellungen eines Google Colab's

5.x. Reenforcement Learning mit

Beispiel: Super Mario Bros.-Modell

```
env = gym_super_mario_bros.make('SuperMarioBros-1-1-v0')
env = JoypadSpace(env, SIMPLE_MOVEMENT)

done = True
for step in range(1000):
    if done:
        state = env.reset()
        state, reward, done, info = env.step(env.action_space.sample())
        env.render()
```

```
time.sleep(.05)

env.close()
```

lässt Mario nur mit einfachen Aktions-Möglichkeiten in seine Welt

Limitierung des Aktions-Raum's für Mario auf 2 bestimmte Aktionen

```
env = gym_super_mario_bros.make('SuperMarioBros-1-1-v0')
gymn_super_mario_bros.actions.COMPLEX_MOVEMENT
env = JoypadSpace(env, [{"right"}, [{"right", "A"}]])

...
```

Anzeige der Rückgabe-Werte

```
env.reset()
next_state, reward, done, info = env.step(action=0)
print("Next-State: " + str(next_state.shape))
print("Reward: " + str(reward))
print("Done: " + str(done))
print("Info: " + str(info))
```

Anpassung des Environment's

Rückgabe von Spiel-Frame's als Bilder, um sie dann später über eine Bild-Analyse-KI analysieren zu lassen

```
env = SkipFrame(env, skip =4) # jedes 4. Bild nutzen
env = GrayScaleObservat(env)
env = ResizeObservation(env, shape=84)
env = TransformObservation(env, f=lambda x: x/255.)
env = FrameStack(env, num_stack=4)
```

Nutzung der Q-Learning (Yuansong FENG)

```
mario = Mario(state_dim=(4, 84,84), actin_dim=env.action_space.n,
              save_dir="")
episodes = 100
for e in range(episodes):
    state = env.reset()
    while True:
        env.render()
        time.sleep(0.05)
        action = mario.act(state)
        # 1 Aktion ausführen und feedback aufnehmen
        next_state, reward, done, info = env.step(action)
        # aktuellen Stand merken
        mario.cache(stete, next_state, action, reward, done)
        # 1 Lernschritt
        q, loss = mario.learn()
```

```
state = next_state
if done or info["flag_get"]:
    break
```

Nutzung anderer (trainierter) Modelle

Modell-Training kann schon bei wenigen Episoden (ein Spiel bis zu Mario's Tod) mehrere Stunden dauern

Rücksetzen des Environment's

```
import gym
import gym_super_mario_bros
from gym.wrappers import FrameStack, GrayScaleObservation,
    TransformObservation
from nes_py.wrappers import JoypadSpace

from agent import Mario
from wrappers import ResizeObservation, SkipFrame

# Initialisieren
env = gym_super_mario_bros.make("SuperMarioBros-1-1-v0")
# Aktionen begrenzen
env = JoypadSpace(env,
    [{"right"},
    [{"right", "A"}])
# Wrapper anwenden
env = SkipFrame(env, skip =4) # jedes 4. Bild nutzen
env = GrayScaleObservation(env, keep_dim=False)
env = ResizeObservation(env, shape=84)
env = TransformObservation(env, f=lambda x: x/255.)
env = FrameStack(env, num_stack=4)
# Environment zurücksetzen
env.reset()
```

Laden eines trainierten Modell's

```
checkpoint = Path("models/mario_100ep.chpt")
mario = Marion(state_dim0(4, 84, 84),
    action_dim=env.action_space.n, save_dir="",
    checkpoint=checkpoint)
episodes = 100
for e in range(episodes):
    state = env.reset()
    while True:
        env.render()
        time.sleep(0.05)
        action = mario.act(state)
        # 1 Aktion ausführen und feedback aufnehmen
        next_state, reward, done, info = env.step(action)
        # aktuellen Stand merken
        mario.cache(state, next_state, action, reward, done)
        # 1 Lernschritt
        q, loss = mario.learn()
        state = next_state
        if done or info["flag_get"]:
            break
```

Q und Arbeit unter: <https://gym.openai.com>



6. praktisches Experimentieren mit Scratch und / oder Snap!

6.1. einfache Anwendungen mit Scratch

6.1.x. Text-Analyse

6.1.x. Bilder-Erkennung

Verkehrszeichen-Erkennung

Erkennung von Obst

6.2. künstliche Intelligenz mit Snap!

<https://snap.berkeley.edu/snap/snap.html#present:Username=seegerer&ProjectName=MOOC-Banana-Hunt-DE>

6.3. verstärkendes Lernen

Beispiel: Bananen-Jagd / Affen-Arcade

Lern-Verfahren: Q-Lernen

Vorlage zum Selber-Programmieren

<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Puzzle-de&editMode&noRun>

Explorations-Rate: hoher Wert steht für eine höhere Chance neue Aktionen auszuprobieren, als sich auf seine Lern-Erfahrungen zu verlassen

Lern-Rate: gibt an, ob neu erworbenes Wissen altes Wissen überschreibt (0 ... kein Lernen; 1 ... aktuelle Lern-Information überschreibt altes Wissen)

Diskontierungs-Faktor: bestimmt, inwieweit zukünftiger Bewertung (hoher Faktor) ein Bedeutung zugeordnet wird (, oder ob nur kurzfristige Belohnungen / Bestrafungen (Faktor = 0) beachtet werden sollen)

Modell-Definition

The image shows two Scratch scripts. The first script is a main loop for a model-based reinforcement learning simulation. It starts with a 'when clicked' event, followed by a 'set Modell to' block that initializes the model with a learning rate of 0.75, a discount factor of 0.5, and an exploration rate of 0.05. The available actions are 'jump' and 'doNothing'. A 'script variables' block defines 'beste Aktion', 'Belohnung', and 'Zustand'. A 'forever' loop contains the following steps: 1. 'set Zustand to round(x position of Fass / 10)'. 2. 'set beste Aktion to best action state: Zustand model: Modell'. 3. 'run beste Aktion'. 4. 'set Belohnung to if touching Fass ? then -10 else 1'. 5. 'update model Modell old state: Zustand new state: round(x position of Fass / 10) reward: Belohnung action performed: beste Aktion'. The second script, also starting with 'when clicked', sets the monkey's position to x: -125, y: -70, switches to the 'monkey1' costume, and sets 'Anzahl Sprünge' to 0.

when clicked

set Modell to
create model learning rate: 0.75 discount factor: 0.5 exploration rate 0.05
list of available actions: list jump doNothing

script variables beste Aktion Belohnung Zustand

forever

set Zustand to round x position of Fass / 10

set beste Aktion to best action state: Zustand model: Modell

run beste Aktion

set Belohnung to if touching Fass ? then -10 else 1

update model Modell old state: Zustand new state:
round x position of Fass / 10 reward: Belohnung action performed: beste Aktion

when clicked

go to x: -125 y: -70

switch to costume monkey1

set Anzahl Sprünge to 0

Initialisiere das Modell

Erfasse den aktuellen Zustand. Hier wird der Zustand durch die X-Position des Fasses bestimmt.

Bestimme das beste Verhalten für den aktuellen Zustand. Wenn dieser Zustand vorher noch nicht eingetreten ist, initialisiere mit Nullen und zeige zufälliges Verhalten.

Berechne die Belohnung. Diese hängt davon ab, ob dem Fass erfolgreich ausgewichen wurde oder nicht. Falls nicht, wird anstelle einer Belohnung eine Bestrafung gegeben.

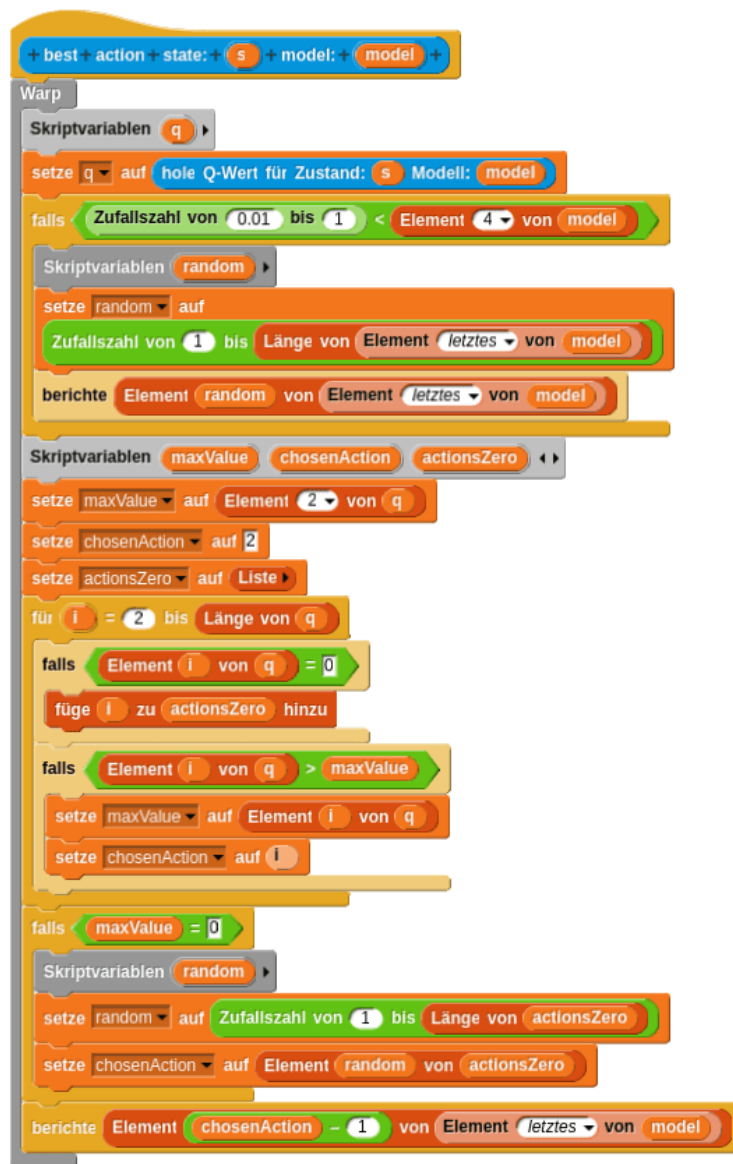
Aktualisiere das Modell, in dem Zustände und Verhaltenswahrscheinlichkeiten gespeichert werden. Das geschieht in Abhängigkeit von der Belohnung/Bestrafung, die als Reaktion auf das "beste Verhalten" gegeben wurde.

Q-Algorithmus

```
+ get + q + values + for + state: + s + model: + model : +  
Skriptvariablen q  
für jedes item von Element 1 von model  
falls Element 1 von item = s  
berichte item  
setze q auf Liste s  
für jedes item von Element letztes von model  
füge 0 zu q hinzu  
füge q zu Element 1 von model hinzu  
berichte q
```

Modell-Update

```
+ update + model + model : + old + state: + stateOld + new + state: + stateNew +  
reward: + reward + action + performed: + act +  
Warp  
Skriptvariablen q0 q1 newValue  
setze q0 auf hole Q-Wert für Zustand: stateOld Modell: model  
setze q1 auf hole Q-Wert für Zustand: stateNew Modell: model  
setze newValue auf  
reward +  
Element 3 von model * max of list alles außer dem ersten von q1  
ersetze Element 1 + get index of act in Element letztes von model in  
q0 durch  
Element 1 + get index of act in Element letztes von model von q0  
* 1 - Element 2 von model  
+ Element 2 von model * newValue
```



fertige Lösung:

<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Solution-de&editMode&noRun>

weitere Links:

<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Pong-Template-de&editMode&noRun> (Vorlage Pong)

<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Pong&editMode&noRun> (Pong Lösung)

<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Breakout&editMode&noRun> (BreakOut Lösung)

<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Space-Invaders&editMode&noRun> (Space Invaders Lösung)
<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Template&editMode&noRun> (Vorlage SnAlp-Blöcke)
<https://snap.berkeley.edu/snap/snap.html#present:Username=snaip&ProjectName=SnAlp-A-Template&editMode&noRun> (Vorlage der Snap-Blöcke für Q-Modell)

7. Experimentieren mit dem google-AIY-Set's

google nutzt heute schon verschiedenste KI in der Praxis für Such-Anfragen und diverse andere Produkte

Mitarbeiter angehalten weitere Anwendungs-Gebiete / -Projekte zu finden / entwickeln trotz riesiger Rechen-Kraft und hoher KI-Kompetenz bei google ist es bei schwachen KI-Systemen geblieben

Übergang zu einer starken KI nicht zu sehen

AIY-Set für Ausbildungs-Zwecke

praktisch KI für deutlich unter 100 Euro

Set's gibt bei verschiedenen Elektronik-Händlern (z.B. conrad, buyzero, ...)

derzeit zwei Versionen verfügbar

google aiy-Set's

- Vision Kit
- Voice Kit

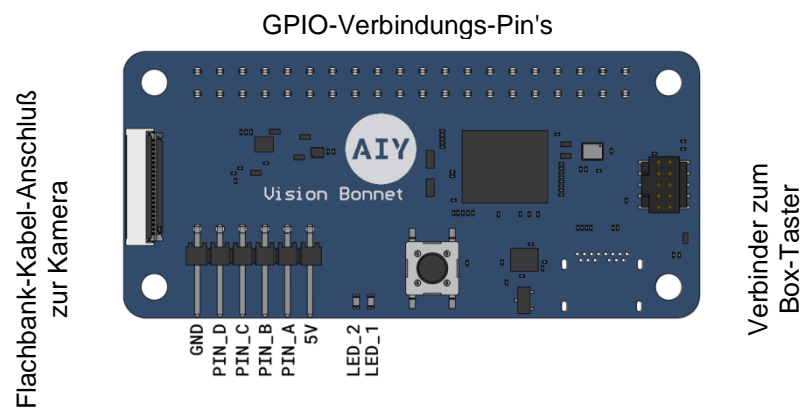
als Erweiterung gibt es Co.Prozessor-System zur Leistungs-Steigerung

wichtig ist sauberes Herunterfahren (Shutdown) des Systems auch im Terminal-Modus, sonst drohen Daten-Verluste und Beschädigungen des Datei-System's

```
sudo poweroff
```

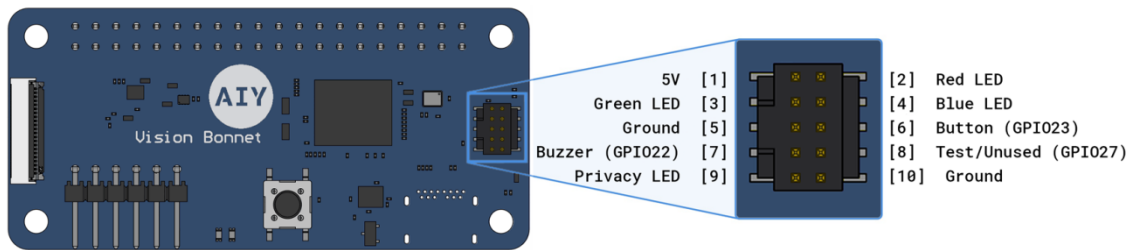
zusätzliches Modul (Bonnet) zum Aufstecken auf dem Raspberry Pi zero

Vision Bonnet



Vision Bonnet

Q: <https://aiyprojects.withgoogle.com/vision/#makers-guide--gpio-expansion-pins>



Vision Bonnet-Box-Taster-Anschluß
 Q: <https://aiyprojects.withgoogle.com/vision/#makers-guide-gpio-expansion-pins>

nach dem Zusammenbau

zum ersten Testen reicht es nur ein Netzteil anzuschließen
 es wird dann das Rasbian-Linux gestartet

erkennt jetzt Gesichter → Knopf leitet grün
 lacht das Gesicht dann verfärbt sich der Knopf auf gelb

weitere Modell dann nach direktem Anschluß an Monitor und Tastatur / Maus (als eigenständiges Computer-System)
 oder Verbindung über WLAN (dann wird kein Monitor / Tastatur / Maus gebraucht)

über Terminal

Wechsel in den Bilder-Ordner:

```
cd ~/Pictures
```

Anschauen eines Bildes:

```
DISPLAY=:0 gpicview Dateiname
```

den Start-Service (Gesichts- und Lächeln-Erkennung) abschalten / beenden:

```
sudo systemctl stop joy_detection_demo
```

andere KI-Modelle auswählen:

Wechsel in den Ordner:

```
cd ~/AIY-projects-python/src/vision
```

Start eines Modell's z.B. Objekt-Erkennung über die Kamera

```
./image_classification_camera.py
```

Starten einer Gesichts-Erkennung:

```
./face_detection_camera.py
```

PINNAME	FUNCTION	Vision Bonnet	FUNCTION	PINNAME
	3V3 power	1	2	5V power
GPIO 02	I2C SDA	3	4	5V power
GPIO 03	I2C SCL	5	6	Ground
GPIO 04	GPIO	7	8	UART TXD GPIO 14
	Ground	9	10	UART RXD GPIO 15
GPIO 17	GPIO	11	12	I2S BCLK GPIO 18
GPIO 27	GPIO	13	14	Ground
GPIO 22	GPIO (Piezo buzzer)	15	16	GPIO (LED button) GPIO 23
	3V3 power	17	18	GPIO GPIO 24
GPIO 10	SPI MOSI	19	20	Ground
GPIO 09	SPI MISO	21	22	GPIO GPIO 25
GPIO 11	SPI SCLK	23	24	SPI CE0 (Myriad) GPIO 08
	Ground	25	26	SPI CE1 GPIO 07
GPIO 00	I2C (EEPROM)	27	28	I2C (EEPROM) GPIO 01
GPIO 05	GPIO	29	30	Ground
GPIO 06	IRQ (Pi to Myriad)	31	32	PWM0 GPIO 12
GPIO 13	IRQ (Myriad to Pi)	33	34	Ground
GPIO 19	GPIO	35	36	GPIO GPIO 16
GPIO 26	GPIO	37	38	GPIO GPIO 20
	Ground	39	40	GPIO GPIO 21

GPIO-Belegung – markierte Pins werden vom Bonnet benutzt

Q: <https://aiyprojects.withgoogle.com/vision/#makers-guide-gpio-header-pinout>

Python-Programm für ein KI-Modell (Unterscheiden von Bekleidungs-Abbildungen)

Importieren der notwendigen Bibliotheken

Tensorflow und Keras sind die Bibliotheken für die KI

NumPy dient der Nutzung von effektiven Daten-Strukturen (Felder statt den Python-typischen Listen)

Matplotlib ermöglicht die Visualisierung von Ergebnissen (Bilder, Graphen / Diagramme, ...)

```
import tensorflow as tf
from tensorflow import keras

import numpy as np
import matplotlib.pyplot as plt
```

es folgt das Laden einer riesigen Datenbank (80'000 Bilder) mit Abbildungen von Bekleidungs-Stücken

dann werden diese Daten in zwei Teile aufgeteilt (als Tupel)

der größte Teil wird als Trainings-Daten verwendet, der restliche Teil zum ersten testen des Modell's

damit wir statt den Ergebnis-Nummern, die das Modell zurückliefert, lieber Ausgabe-Texte bekommen, werden diese festgelegt

```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

class_names = ["T-Shirt/Top", "Hose", "Pullover", "Kleid",
               "Mantel", "Sandale", "Hemd", "Turnschuh",
               "Tasche", "Stiefelette/hohes Schuh"]
```

in den nächsten Zeilen wird das Modell übersetzt / kompiliert und dann trainiert

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
                  from_logits=True), metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=10)
```

das System gibt dann erste Erfolgs-Meldungen zurück

das alles passierte mit dem vorgegebenen Trainings-Daten-Bestand aus der mnist-Datenbank

Test mit Test-Daten-Bestand aus der mnist-Datenbank usw. ist die nächste Herausforderung

```
test_loss, test_acc = model.evaluate(test_images,
                                     test_labels, verbose=2)

print()
print('Akkuratheit: ', test_acc)
```

nun kann auch der Test mit eigenen Bildern erfolgen

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])

predictions = probability_model.predict(test_images)

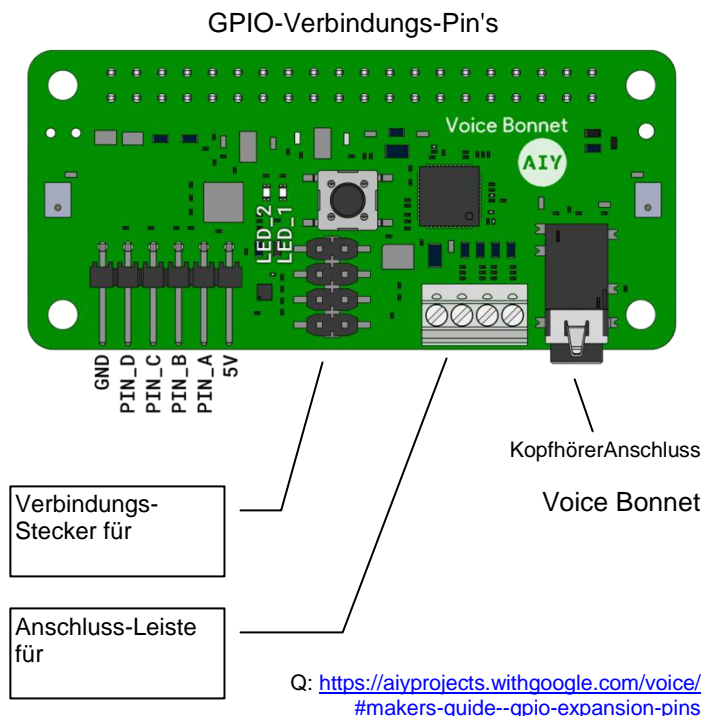
prediction[0]

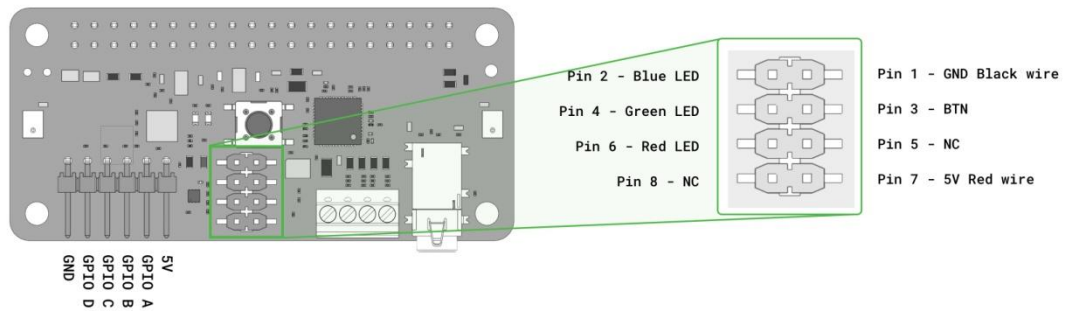
np.argmax(predictions[0])
```

Q: SMW-Workshop 2020 (Bundeskonzferenz Schule mit Wissenschaft (Deutsches Museum München)); leicht geändert und eingedeutscht

vollständiges Tutorial: <https://www.tensorflow.org/keras/classification>

Voice Bonnet





Vision Bonnet-Box-Taster-Anschluß

Q: <https://aiyprojects.withgoogle.com/voice/#makers-guide--gpio-expansion-pins>

ermöglicht z.B. den Aufbau eines Sprach-gesteuerten Assistenten (praktisch einen google-Assistenten)

beim Erststart (neues Image auf der miniSD-Karte) läuft eine Joy Detector Demo

Konfigurations-Datei (`~/AIY-projects-python/src/examples/vision/joy/joy_detection_demo.service`)

```
[Unit]
Description=AIY Joy Detection Demo
Requires=dev-vision spicomm.device
After=dev-vision spicomm.device
Wants=aiy-board-info.service
After=aiy-board-info.service

[Service]
Type=simple
Restart=no
User=pi
Environment=AIY_BOARD_NAME=AIY-Board
EnvironmentFile=-/run/aiy-board-info
ExecStart=/usr/bin/python3 /home/pi/AIY-projects-python/src/examples/vision/joy/joy_detection_demo.py --enable streaming --mdns_name "${AIY_BOARD_NAME}" --blink_on_error

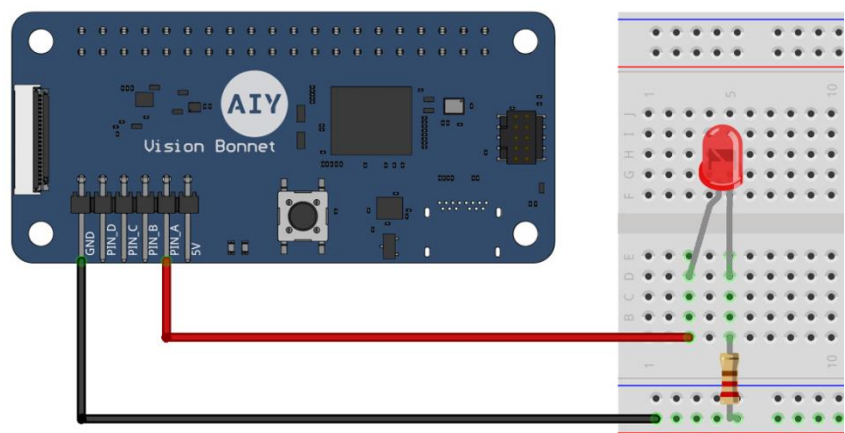
[Install]
WantedBy=multi-user.target
```

Q: <https://aiyprojects.withgoogle.com/vision/#makers-guide--gpio-expansion-pins>

für die Programmierung mit Python werden verschiedene Bibliotheken mitgeliefert

aiy-Bibliotheken / -Module

- **aiy.toneplayer** Musik-Player für den Piezo-Summer
- **aiy.trackplayer** Tracker-basierter Musik-Player für den Piezo-Summer
- **aiy.vision.annotator** Annotation / Labeln von Bildern
- **aiy.vision.inference** Interference-Maschine, die mit dem Vision Bonnet kommunizieren kann
- **aiy.vision.models** Sammlung von ML-Modellen zur Bild- und Objekt-Erkennung
- **aiy.board** API für die Nutzung der Möglichkeiten des Vision Bonnet'
- **aiy.leds** API zur Steuerung der LED's am Vision Bonnet
- **aiy.pins** Bereitstellung der zusätzlichen Pin's des Zusatz-Board's mit / über die gpiozero



Test-Aufbau für die Nutzung der Bonnet-Pin's

Q: <https://aiyprojects.withgoogle.com/vision/makers-guide-gpio-expansion-pins>

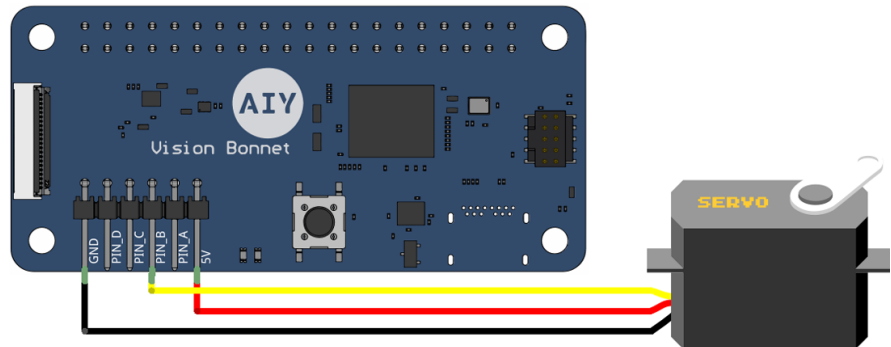
zugehöriges Test-Programm:

```
from time import sleep
from gpiozero import LED
from aiy.pins import (PIN_A, PIN_B, PIN_C, PIN_D)

leds = (LED(PIN_A), LED(PIN_B), LED(PIN_C), LED(PIN_D))
while True:
    for led in leds:
```

```
led.on()
sleep(0.5)
led.off()
```

Q: <https://aiyprojects.withgoogle.com/vision/#makers-guide--gpio-expansion-pins>



Test-Aufbau für die Nutzung eines Servo's mit dem Bonnet

Q: <https://aiyprojects.withgoogle.com/vision/#makers-guide--gpio-expansion-pins>

zugehöriges Test-Programm:

```
from gpiozero import Servo
from aiypins import PIN_B

# Create a servo with the custom values to give the full dynamic range.
tuned_servo = Servo(PIN_B, min_pulse_width=.0005, max_pulse_width=.0019)

# Move the Servos back and forth until the user terminates the example.
while True:
    tuned_servo.max()
    sleep(1)
    tuned_servo.mid()
    sleep(1)
    tuned_servo.min()
    sleep(1)
```

Q: <https://aiyprojects.withgoogle.com/vision/#makers-guide--gpio-expansion-pins>

g.co/aiyprojects
<https://aiyprojects.withgoogle.com/>

8. KNN spezial -- Was sonst noch so geht?!

8.x. spezielle KNN-Typen

8.x.y. GAN's

8.x.y.z. WASSERSTEIN GAN

<https://machinelearningmastery.com/how-to-code-a-wasserstein-generative-adversarial-network-wgan-from-scratch/>

Nutzung der deeplearning2020 Python library

Zeitaufwand etwa 1 Minute

Wir haben für euch eine Python-Bibliothek bereitgestellt, in der neben der Möglichkeit zur Abgabe eurer Übungslösungen auch einige sinnvolle Hilfsfunktionen enthalten sind.

Um diese zu nutzen müsst ihr folgendes tun:

```
pip install --upgrade deeplearning2020 # im Terminal bei lokaler Installation
```

```
!pip install --upgrade deeplearning2020 # in Jupyter Notebooks
```

```
from deeplearning2020 import helpers # Import der Helper Funktionen
```

Die zwei Helper-Funktionen, welche ihr wahrscheinlich im Kurs am meisten nutzen werdet, sind folgende:

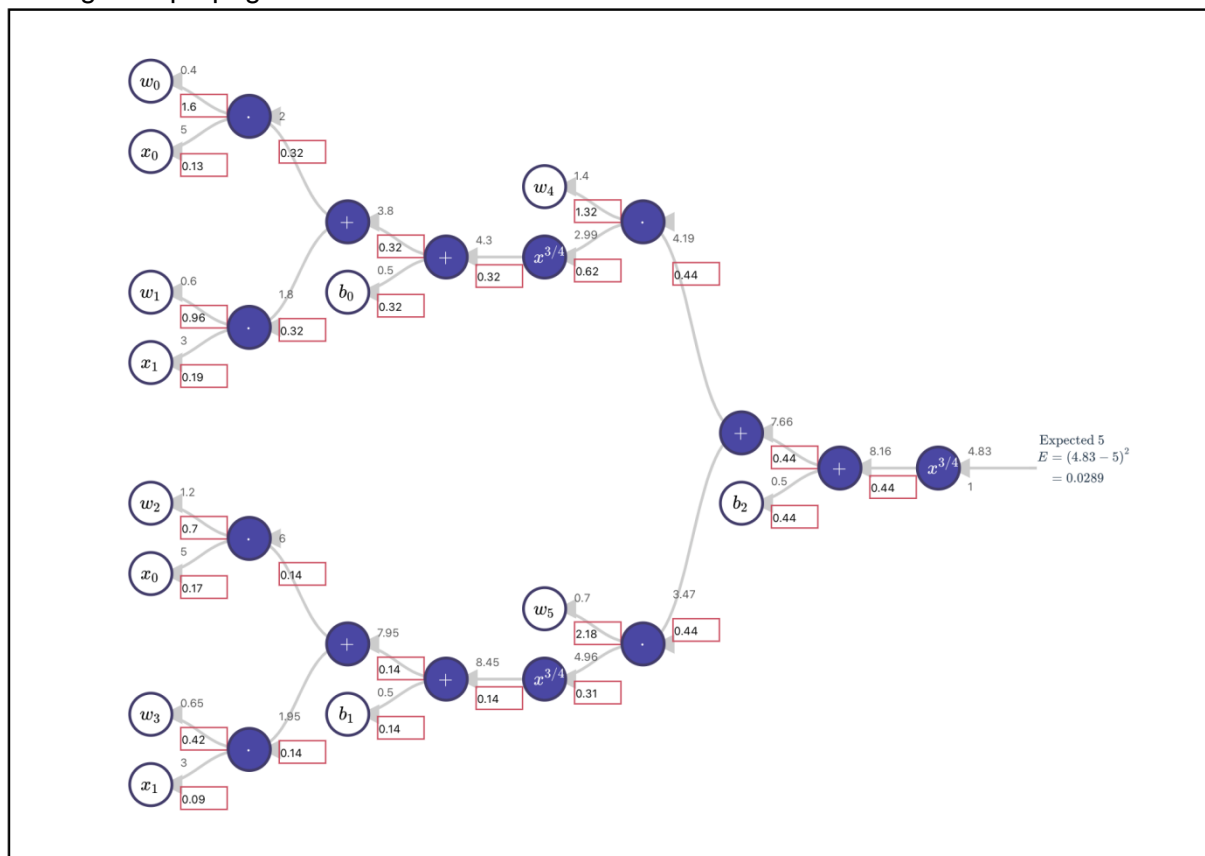
```
helpers.plot_history(title, history vom Training)
```

```
helpers.plot_images(images (z.B. train_data.take(9), class_names)
```

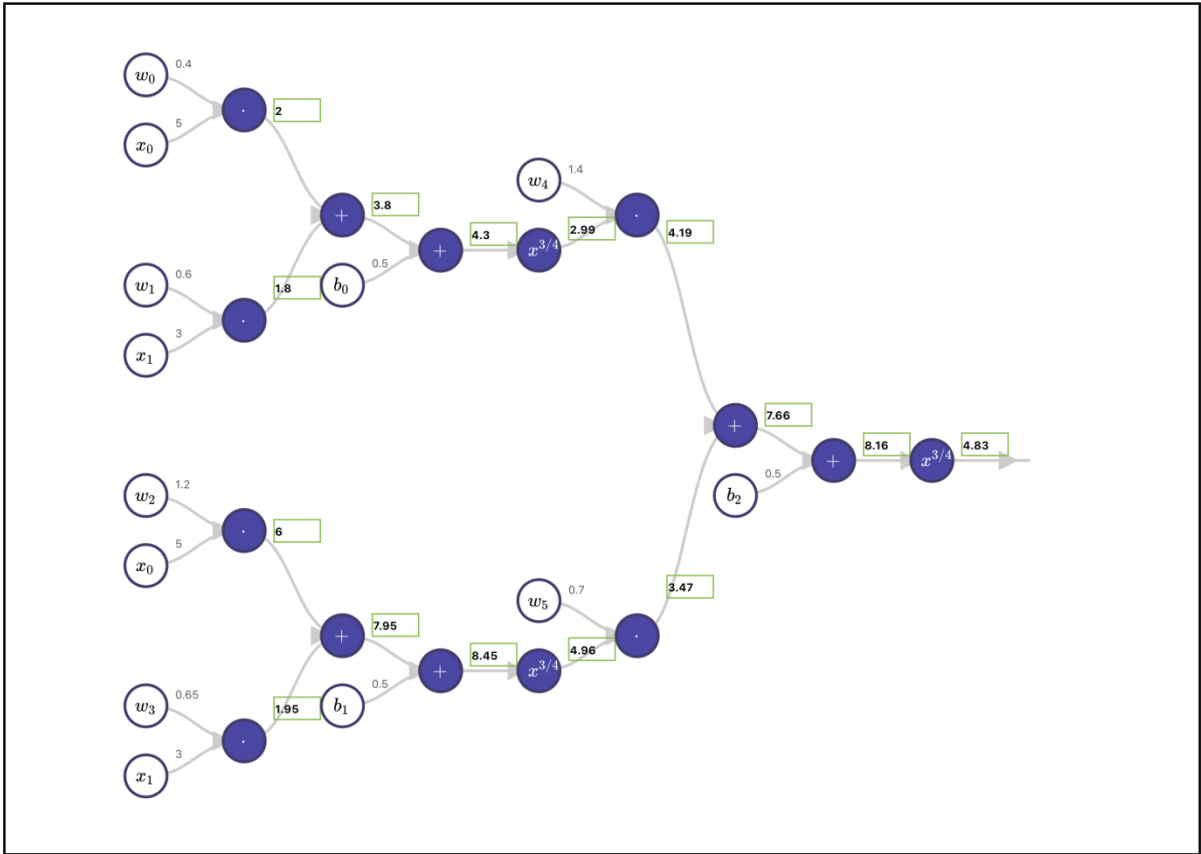
Die Nutzung der Funktionen sind in den Jupyter Notebooks, die zu den Praxisvideos bereitgestellt werden, erklärt.

Ihr wollt verstehen, wie die "Helper"-Funktionen funktionieren? Schaut euch gerne den Code auf [Github](#) an.

Lösung Backpropagation



Lösung Feed Forward



8.x. Haus-Preise in Kalifornien

Beispiel aus dem OpenHPI-Kurs "Künstliche Intelligenz und Maschinelles Lernen in der Praxis"

ev. auch neu bei "supervised Learning" zuordnen

8.x.y. Daten-Aufbereitung

Festlegen der Ziel-Funktion

```
def berechne_preis(zimmer_anzahl, lage, ml_modell):
    preis = ml_modell(zimmer_anzahl, lage)
    return preis
```

Importieren wichtiger Bibliotheken zur Vereinfachung der Arbeit

```
import os # plattform-unabhängige System-Pfade ++
import yaml # Konfiguration
import pandas # Daten-Management
import numpy # mathemitsche Hilfsfunktionen

import seaborn as sb
%matplotlib inline
import matplotlib.pyplot as mpl
from pandas.plotting import scatter_matrix

from sklearn.model_selection import StratifiedShuffleSplit,
    train_test_split # Daten-Splitting
from sklearn.linear_model import LinearRegression # ML
from sklearn import metrics # Modell-Evaluierung
```

Anlegen der Konfuguration

```
konfiguration = yaml.safe_load(open("../config.yml"))

DATA_DIR = os.path.join("../", "data") # eine Ordner-Ebene höher
FILE_PATH = os.path.join(DATA_DIR, konfiguration["datasets"]
    ["week1"])
IMAGES_DIR = os.path.join("../", "images")
CALIFORNIA_PATH = os.path.join(IMAGES_DIR, konfiguration["images"]
    ["california"])
```

Einlesen der Daten

```
df = pd.read_csv(FILE_PATH)
df.head()
```

Überprüfen der Daten (z.B. auf fehlende Daten)
und Ausgabe der %-Zahl und der Anzahl

```
num_missing = len(df) - len(df.dropna())
ratio_missing = num_missing / len(df)
ratio_missing = np.round(ratio_missing * 100, 3)
print(f"{ratio_missing}% der Daten ({num_missing} Punkte) haben  
fehlende Werte")
```

Ansehen der Datensätze mit fehlenden Daten

```
df.loc(df.isnull().sum(axis=1) ==1)
```

unklare Daten werden mit "NaN" (von Pandas) charakterisiert

Anzeige der Attribute und die Anzahl unklarer Daten

```
df.isnull().sum(axis=0)
```

im California-Modell ist das immer das Attribut "total_bedrooms"

Löschen der Datensätze in denen unklare Daten enthalten sind
danach dann Neu-Anlegen des Index

```
df = df.dropna()
df = df.reset_index(drop=True)
```

Anzeige der Zielwerte als Vorab-Überblick
u.a. auch zum Auffinden von Daten-Problemen

```
sb.displot(data=df, x="median_house_value", kde=True);
```

z.B. bei den Haus-Daten, hier sind die Preis scheinbar bei 500'000 gedeckelt worden

Vorbereitung des Aufteilen's der Daten-Basis in Trainings- und Test-Daten

```
bins = [0] + list(description[median_house_value][  
    "254", "504", "754"]).astype(int) + [np.inf]]  
  
print(bins)
```

Erstellen von Haus-Kategorien (Hilfs-Kategorie für Analysen und das Splitting)

```
df["house_cat"] = pd.cut(df[median_house_value], bins=bins,  
    labels=["0 - 25%", "25 - 50%", "50 - 75%", "75 - 100%"])
```

Kontrolle, ob die Daten gut verteilt wurden

```
sb.countplot(data = df, x = "house_cat");
```

jetzt den Split in Trainings- und Test-Daten

```
split = StratifiedShuffleSplit(n_splits=1, test_size=0.1,  
    random_state=0)  
for train_index, test_index in split.split(df, df["house_cat"]):  
    df.train = df.loc[train_index]  
    df.test = df.loc[test_index]
```

Kontrolle der Aufteilung

```
sb.countplot(data=df_train, x="house_cat");
```

```
sb.countplot(data=df_test, x="house_cat");
```

Entfernen der Hilfs-Kategorie

```
df.train = df_train.drop("house_cat", axis=1)  
df.test = df_test.drop("house_cat", axis=1)
```

8.x.y. Daten-Analysen und -Visualisierungen

z.B. Darstellung der Geo-Position aus den Längen- und Breiten-Graden

```
df_train.plot(kind="scatter", x="longitude", y="latitude",  
    figsize=(12,8));
```

Erweiterung der Darstellung um weitere Info's (hier Preis und Bevölkerungs-Dichte)

```
df_train.plot(kind="scatter", x="longitude", y="latitude",  
    alpha=0.4, s=df_train["population"]/100, label="population",  
    figsize=(12,8), c="median_house_value",  
    cmap=plt.get_cmap("jet"), colorbar=True, sharex=False)  
plt.legend();
```

Erweiterung um Karten-Daten

```
from helper_functions import plot_df_on_california
import plot_df_on_california(df_train, CALIFORNIA_PATH)
```

tiefergehende Analyse hinsichtlich der Nähe zum Meer

```
ocean_proximity_counts = df_train["ocean_proximity"].value_counts()
ocean_proximity_counts
```

Visualisierung der Gruppen nach Abstand vom Meer
eigene Hilfsfunktion

```
def filter_df_by_proximity(df, proximity):
    return df.loc[df["ocean_proximity"] == proximity]
```

für jede der Kategorien "INLAND", anzeigen lassen

```
plot_df_on_california(df_by_proximity(df_train, "INLAND"),
                      CALIFORNIA_PATH)
```

so lassen sich auch die anderen Kategorien untersuchen

Datensätze, die statistisch nicht ausreichend gesichert sind, und ev. lieber von einem Menschen oder einem anderen System bewertet werden sollen, können ausgeschlossen werden dann aber sowohl in den Trainings- also auch in den Test-Daten!

```
df_train = df_train.drop(filter_df_by_proximity(df_train,
                                                "ISLAND").index)
df_test = df_test.drop(filter_df_by_proximity(df_test,
                                              "ISLAND").index)
```

Erstellen einer klassisch programmierten – Regel-basierten – Prognose als Funktion

```
def predict_by_ocean_proximity(df,
                               median_house_value_by_proximity):
    df["ocean_proximity_prediction"] = df["ocean_proximity"].apply(
        lambda x: median_house_value_by_proximity[x])
    return df
```

Aufruf der Prognose-Funktion

```
df_train = predict_by_ocean_proximity(df_train,
                                       median_house_value_by_proximity)
df_train.head()
```

```
df_test = predict_by_ocean_proximity(df_test,
                                      median_house_value_by_proximity)
df_test.head()
```

mögliche Korrelationen analysieren
(gibt es direkte Abhängigkeiten des Preises von einem Attribut)
Verfahren nach PEARSON

```
df_train.corr()["median_house_value"].abs().sort_value(
    ascending=False)
```

visualisiert

```
df_train.plot(kind="scatter", x="median_incom",
    y="median_house_value", alpha=0.1);
```

in den Daten ergibt sich ein Problem, dass sind die Daten mit Haus-Preises mit 500'000 Dollar, die Kappung bringt hier unklare Zusammenhänge

Analysen hinsichtlich Anteil der Betten-Räume bezogen auf die Gesamt-Raum-Anzahl oder die Anzahl der Personen pro Haushalt

```
def engineer_features(df):
    df["ratio_bedrooms"] = df["total_bedrooms"] / df["total_rooms"]
    df["people_per_household"] = df["population"] /
        df["households"]
    return df
```

```
df_train = engineer(df_train)
df_test = engineer(df_test)
```

und dann die Korrelation zu diesen neuen Attributen anzeigen / analysieren

```
df_train.corr()["median_house_value"].abs().sort_value(
    ascending=False)
```

8.x.y. Entwickeln eines (ersten) KI-Modell's

die Rohdaten müssen zuerst in ein Format für die KI-Modelle in Python gebracht werden
Anzeige der Daten

```
df_train.head()
```

die Wort-Attribute (z.B. "ocean_proximity") können z.B. nicht von KI-Modellen verarbeitet werden, die brauchen numerische Werte
mögliches Verfahren ist One-Hot-Encoding, d.h. die Attribut-Werte werden zu neuen Attributen und die einzelnen Daten werden dann in 1 und 0 (wahr und falsch) umgesetzt

```
df_train_ml = pd.get_dummies(df_train)
```

```
df_test_ml = pd.get_dummies(df_test)
```

```
df_train.head()
```

nun müssen die Ziel-Werte (bei uns die Hauspreise) von den Eingabe-Daten abtrennen damit das KI-System weiss, worauf gelernt werden soll

```
def get_features_and_targets(df):  
    X=df.drop(["median_house_value"], axis=1).values  
    y=np.stack(df("median_house_value"))  
    return X,y
```

```
X_train, y_train = get_features_and_targets(df_train_ml)  
X_test, y_test = get_features_and_targets(df_test_ml)
```

das erste KI-Modell lautet nun

```
clf = LinearRegression()  
clf.fit(X_train, y_train)
```

8.x.y. Ausprobieren / Testen eines (ersten) KI-Modell's

Testen des Modell's mit den (bisher noch nicht im Modell benutzten) Test-Daten

```
predictions = clf.predict(X_test)
```

Anzeige der Ergebnisse

```
print("Prediction\t|True Value")  
print(30*"--")  
for idx, (pred, annotation) in enumerate(zip(predictions, y_test));  
    if idx == 10:  
        break  
    pred = int(pred)  
    annotation = int(annotation)  
    print(f{pred}\t\t|{annotation})
```

durch Einsatz weiterer statistischer Merkmale / Kennzahlen soll die Qualität nochetwas genauer betrachtet werden

Median Absolute Error MAE $MAE(y, \hat{y}) = \frac{\sum_{i=0}^n |y_i - \hat{y}_i|}{n}$

sagt aus, mit welcher durchschnittlichen Abweichung vom Real-Wert kann man rechnen

$$R^2\text{-Score} \quad R^2(y, \hat{y}) = \frac{\sum_{i=0}^n |y_i - \hat{y}_i|}{\sum_{i=0}^n |y_i - \bar{y}|}$$

gibt an, wieviel der Varianz (Abweichung vom Ziel-Wert) ist durch die unabhängige Variable erklärt

(Interpretation: ein R^2 von 100% bedeutet eine perfekte Prognose, 0% besagt, dass die Prognose genauso gut ist, wie der Mittelwert; ein negativer Wert bedeutet, dass die Prognose schlechter als der Mittelwert ist)

```
mae = metrics.mean_absolute_error(y_test, predictions)
mae_ratio = metrics.mean_absolute_percentage_error(y_test,
                                                    predictions)
r2_score = metrics.r2_score(y_test, predictions)
```

```
print(f"es ist mit einem durchschn. Fehler von {np.round(mae)} zu↵
rechnen")
print(f"das entspricht einer fehler-Quote von {np.round(mae_ratio↵
*100)}%")
print(f"Der R2-Score ligt bei {np.round(r2_score * 100)}%")
```

man kann auch einzelne Datensätze testen, bzw. sich die Bewertung im Modell ansehen hier für den Datensatz 0 (1. Datensatz)

```
single_pred = clf.predict([X_test[0]])[0]
print(f"Prognose: {single_pred}")
print(f"Real-Wert: {y_test[0]}")
```

8.x. Projekt Film-Vorschlag-System

Beispiel aus dem OpenHPI-Kurs "Künstliche Intelligenz und Maschinelles Lernen in der Praxis"

ev. auch neu bei "supervised Learning" zuordnen

auch: Recommender Systeme

z.B.:

- in online-shop's Vorschläge für zusätzliche Angebote / Waren / Einkäufe
- bei Streaming-Diensten Vorschläge für die nächsten zu schauenden Filme
- ...

Vorschlag's-System sollte geeignete Filme für einen Nutzer vorschlagen, basierend auf:

- Seh- / Nutzungs-Verhalten

Bewertung von Filmen z.B. durch:

- Nutzer-Bewertung durch z.B. Sterne
- Watchtime (Zugriffs-Zeit)
- Zugriffs-Häufigkeit (wiederholtes Ansehen)
- Empfehlung anderer Nutzer (Community)
- ...

Bewertung des Nutzer-Verhalten's

- gesehene Filme
- gesehene Filme je Genre
- Geschlecht, Alter, ... (wenn solche Daten vorliegen)
- Seh-Zeiten
- ...

mögliche Ansätze für Recommender Systeme

• popularity-based Filtering	→ kein Verfahren des Maschinellen Lernen's Filterung nur nach Beliebtheit (z.B. Aufrufe / Zeiteinheit)
• content-based Filtering	Auswahl der Vorschläge nach der inhaltlichen Ähnlichkeit bzw. aus dem gleichen Genre → K-nearest Neighbors (KNN)
• collaborative Filtering	Empfehlung von Filmen etc. nach dem vergleichbaren (Seh-)Verhalten des Nutzer's mit anderen Nutzern
• hybride Filtering	

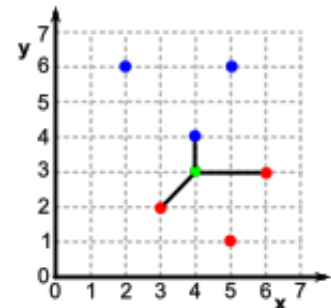
Ähnlichkeit von Filmen z.B. über:

- gleiches Thema

- gleiches Genre
- gleiche Schauspieler
- gleiche Produktions-Firma
- gleiches Erscheinungs-Jahr
- passende Alters-Beschränkung
- ähnliche Kurz-Beschreibung (über Text-Analyse (Häufigkeit von Schlüsselworten))

Ähnlichkeit wird durch Abstand zu den anderen Filmen / Vorschlags-Kandidaten / ... beschrieben

zu viele Merkmale (Features) im Vergleich zu den Beobachtungen führt leicht zu einem Overfitting des ML-System's (ML-System lernt praktisch die wenigen Beispiel (im Vergleich zu den Möglichkeiten) "auswendig")

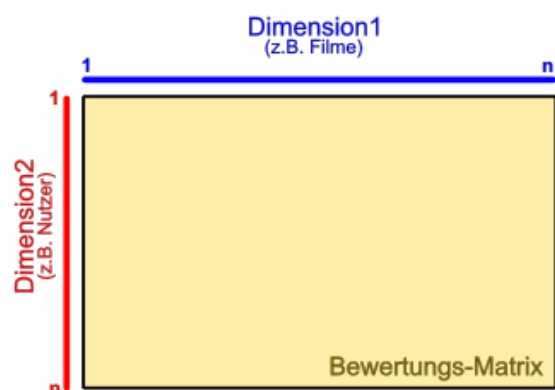


Problem bei Schlüsselwörtern ist die dadurch aufkommende Viel-Dimensionalität der Daten → Fluch der Dimensionalität (Curse of Dimensionality) Richard BELLMANN () mit steigender Dimensionalität werden die Beobachtungen (Observation) immer äquidistanter (haben immer ähnliche Abstände) → Ergebnisse unzuverlässiger oder so vielzählig, da die Ähnlichkeit praktisch gleich ist / wird

beim collaborative Filtering wird sehr häufig auf die Matrix-Faktorisierung (matrix factorization)

eine 2D-Matrix (z.B. Nutzer, Filme) wird in 2 neue 2D-Matrizen zerlegt

die einzelnen Dimensionen z.B. eben die Filme oder die Nutzer sind in der Bewertungs-Matrix natürlich nur eindimensional aber in der praxis liegen ja weitere Daten zu den Nutzern oder Filmen vor diese sind als weiteren Dimensionen zu verstehen



für tiefgreifende Untersuchungen für ein Vorschlags-System üssten diese mit beachtet werden dazu gehört z.B. herauszubekommen, von welchen Merkmale der Nutzer und denen der Filme ein effektiver (annehmbarer) Film-Vorschlag resultiert

der Such-Raum ist entsprechend groß

nehmen wir 2000 Filme und 300 Nutzer an, wobei wir 30 Film- und 10 Nutzer-Merkmale haben, dann sind $2'000 \times 300 = 600'000$ Bewertungen zu analysieren

nimmt man noch die Detail-Daten mit dazu, da sind dass schon

$600'000 \times 30 \times 10 = 90'000'000$ Einzel-Informationen

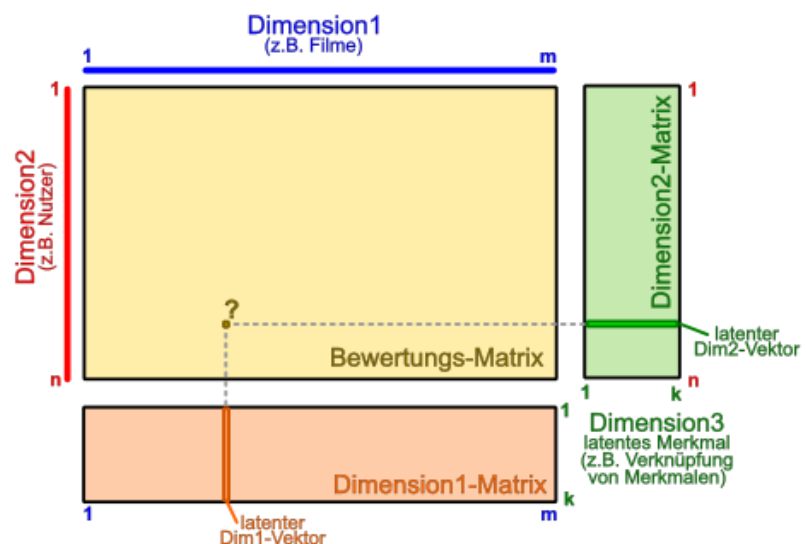
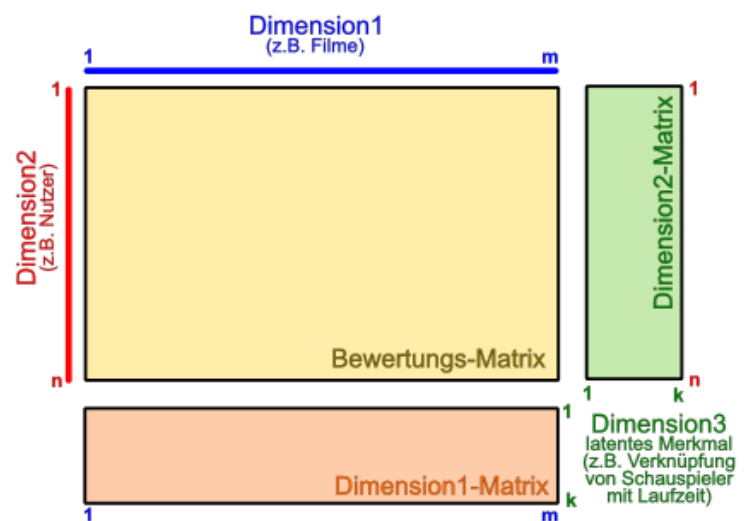
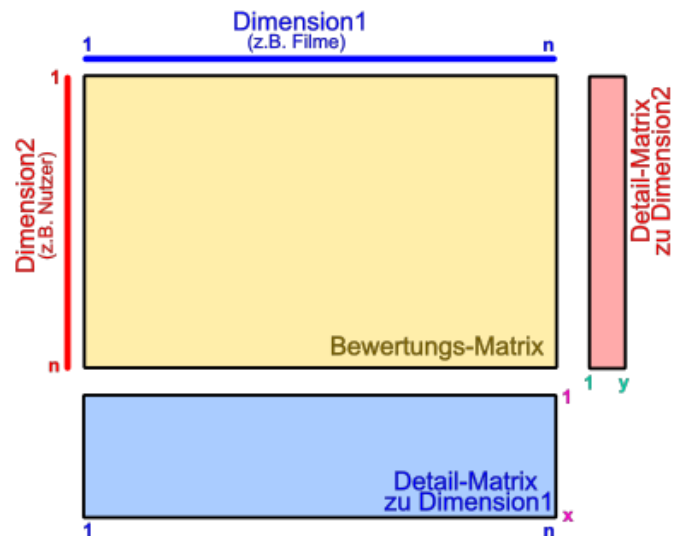
effektiver ist ein dabei wird eine weitere Dimension künstlich hinzugefügt, die zuerst einmal unsichtbar / unbestimmt ist (latente Merkmale)

praktisch erzeugt man sich aus der großen Bewertungs-Matrix zwei kleiner Matrizen

das entscheidende Problem ist das Finden eines Merkmals, dass als latente Dimension geeignet ist. Dies kann eine für uns menschen velleicht völlig unsinnige Kombination von Detail-Merkmalen sein

für das ML-Modell ist es aber das "sinnvolle" Merkmal

im weiteren wird jetzt nur in den zwei kleineren Dimensions-Matrizen (in Abb. grün und orange) gearbeitet und hier jeweils vergleichbare Vektoren herausgesucht die in ihnen enthaltenen Haupt-Dimensions-Informationen werden nun für die Auswahl in der Bewertungs-Matrix genutzt



für ein bestimmten Nutzer ergibt sich dann jeweils ein spezieller Vektor in der einen neuen Matrix und ein weiterer in der anderen neuen Matrix ausgehend von beiden Vektoren, wird dann nach ähnlichen Vektoren gesucht und Vorschläge abgeleitet

Vorschlag eines Film's aus der Verknüpfung der latenten User- und Film-Vektoren ergibt ein wahrscheinliches Rating, dass der Nutzer dem Film geben könnte / würde größter Rating-Wert ist dann der vorzuschlagene Film

Problem bei einem neuen oder anonymen Nutzer (ohne bekannte Nutzer-Daten)
→ Kalt-Start-Problem

dann könnte man z.B. ein anderes Verfahren nutzen, z.B. **Popularity-based Filtering**

8.x.y. Daten-Aufbereitung

typisches Arbeits-Feld von DataScience-Arbeitern (Data-Scientist)

- Daten-Beschaffung
- Daten-Analyse
- Daten-Aufbereitung
- Daten-Bereinigung
- Daten-Annotierung (Labeln der Daten)
- ...

das Erstellen von ML- oder KI-Modellen nimmt eher einen kleineren Teil der Arbeit ein

Überblick über die Daten / Daten-Analyse

"The Movies Dataset"

Q: kaggle.com (Plattform zu KI; Anmeldung notwendig)

beinhaltet 45'000 Filme, die von 270'000 Nutzern bewertet wurden (→ 26'000'000 Bewertungen)

Download: <https://www.kaggle.com/rounakbanik/the-movies-dataset>

auch über die API möglich

Anleitungen unter: <https://www.kaggle.com/docs/api>

bzw.

<https://github.com/Kaggle/kaggle-api>

die zwei interessierenden Dateien sind `movies_matadata.csv` und `ratings.csv`

```
import yaml
import os
config = yaml.safe_load(open("../config.yml"))
os.chdir(config["datasets"], [week2])
```

Anzeige in einem Linux-basiertem System

ls -gnG

Laden der Daten in einen Pandas-Dataframe (praktisch Tabelle)
und nachfolgend Anzeige der Tabelle (oberste Elemente)

```
import pandas as pd
df_film_metadata = pd.read_csv("movies_metadata.csv",
                               low_memory=False)

df_film_metadata.head()
```

erste Analyse und Anzeige wichtiger Merkmale und Kennwerte

```
explore(df_film_metadata[df_film_metadata.columns.difference(
    ["adult", "", ""]), method="p..
```

Auswahl der Merkmale für ein Projekt
z.B. die Beschreibung in der Kategorie "overview"

Anzeige hier des 10. Platzes (wegen Python Index = 9 gewählt)

```
print("Film-Titel: " + df_film_metadata["title²"][9] + "\n")
print("Film-Beschreibung: " + word_list[9])
```

Erstellen einer Wort-Wolke (Word-Cloud)

```
makeImage(getFrequencyDictForText(word_list))
```

Unterscheidung zwischen Schlüssel-Wörtern ("Pirat", "Held", "Liebe", ...) und Füll- oder Neben-Wörtern ("der", "und", "ein", ...) auch Stopwords genannt

```
makeImage(getFrequencyDictForText(word_list,
                                   remove_stop_words=True))
```

nun Analyse der Bewertungen durch die Nutzer
mittels Collaborative Filtering
zuerst aber wieder laden in ein Pandas-Dataframe (praktisch wieder eine Tabelle)
und Anzeige der obersten Elemente / Datensätze der Daten-Tabelle

```
df_film_votes = pd.read_csv("ratings.csv", low_memory=False)

df_fim_votes.head()
```

um einen Eindruck darüber zu bekommen, wieviele unterschiedliche Filme und weiviele unterschiedliche Nutzer bewertet haben nutzt man

```
df_film_votes[[movieId],[userId]].nunique()
```

nun Analyse der Bewertungs-Möglichkeiten
in welchem Spektrum (z.B. bis 5 oder bis 10) bewertet werden darf / wurde und in welcher Unterteilung (z.B. halbe oder ganze Sterne)

```
sorted(df_fim_votes["rating"].unique())
```

Betrachtung der Häufigkeits-Verteilung der Bewertungen

```
df_film_votes[["rating"]].hist()
```

Auffällig die immer geringeren Häufigkeiten, bei halben Sternen

Daten-Aufbereitung / -Vorbereitung

Möglichkeiten / Schritte / ... der Daten-Vorbereitung

- Anonymisierung
- Behandlung fehlender Werte
- Plausibilitäts-Check
- Feature-Scaling
- Behandlung kategorischer Variablen
- Vorbereitung spezieller Medien-Daten(-Typen) / Domain-spezifische Aufbereitung
 - Bilder
 - Text
 - Sprache
 - ...
- Aufteilen in Trainings- und Test-Daten (Train/Test-Splitting)
- Annotation / Labeln (für supervised Learning)

Anonymisierung

z.B. durch:

- Löschen von Attributen (Spalten) in der Daten-Tabelle
 - z.B. Name, Vorname
 - Beachtung von indirekten Personen-Bezügen (z.B. bei kleineren Daten-Beständen: Geburts-Datum und Geschlecht (als Kombination sehr häufig indirekter Bezug möglich))
- manuelles Filter
- ...
- differential privacy

- spezielle Methoden des Machine Learning
 - z.B.: Autoencoders (Einsatz eines ML-/KI-Systems zum umkodieren der Daten, so dass kein Rück-Bezug mehr möglich ist); ev. nötig, wenn Personenbezogene Daten für die Auswertung relevant sein könnten
- ...

Ziel ist Erreichen der K-Anonymität (kann ein zu anonymisierendes Objekt in einem Daten-Bestand mit K-vielen Objekten noch wieder zurückerkannt werden
je größer K ist, umso schwieriger wird i.A. die Rück-Erkennung bei möglichst wenig Attributen und Attribut-Ausprägungen
z.B. würde die Verwendung nur des Geschlecht's i.A. kaum ein Rückschluß ermöglichen (natürlich in einem Datensatz von Hebammen oder Ingenieuren)

Umgang mit fehlenden Daten z.B. durch:

- Löschen des betroffenen Datensatzes
- Nachrecherchieren
- Setzen von NULL-Werten
- Einsetzen von Durchschnitt oder Median
- Auffüllen z.B. bei Daten-Reihen
 - Backwards Fill (Zurück-Berechnen eines Wertes, wenn diese z.B. zwischen dem Vorgänger-Datensatz und dem betreffenden Datensatz liegen muss)
 - Forward Fill (Voraus-Berechnen eines Wertes (z.B. bei diskret steigenden Werte))
- Umsetzung von kategorischen Werten in diskrete / binäre
 - bei ordinalen Werte (z.B. Kleider-Größen (XS ... XXL) Ersetzen durch skalierte Zahlen-Reihe (z.B. 0 .. 5)
 - bei nicht-ordinalen Werten (z.B.: bei GeburtsOrt "Berlin", "Rostock", "München") Einsatz des One-Hot-Encoding (Bildung neuer Attribute im Daten-Bestand (z.B. "geb_Berlin", "geb_Rostock", ...) und der Belegung mit 0 bzw. 1 als binäre Werte)
- Anpassungen bei Daten mit verschiedenen Sprachen
 - ev. Übersetzen von Kategorien, Werten, Texten
 - ev. Umrechnungen bei unterschiedlichen Einheiten
 - ev. Stemming (verschiedene Zeitformen oder Deklinationen werden auf die Wort-Stammform transformiert)
 - ev. Lemmatization (Entfernen von Vor- und Nach-Silben / Wort-Enden)
 - ev. Stop-Words-Removal
 - ev. Noise Removal / Special Characters
 - Lowercase (alles auf Kleinschreibung bringen)
 - ev. Coreference Resolution (Inhalts-Abhängigkeiten aufdecken und die Wort-Anzahl reduzieren bzw. auf den Kern bringen)

(z.B. Klaus fährt ein großes Auto. Er holt seine Kinder ab.
wird zu: Klaus fährt ein großes Auto. Klaus holt seine Kinder ab.)

- ...

Vorbereitung spezieller Medien-Daten(-Typen) / Domain-spezifische Aufbereitung

Aufarbeitung / Vorbereitung von Bildern:

- einheitliche Orientierung (durch Drehung / Rotation)
- Größen-Anpassung (Resizing)
- Entfernen von Markierungen / Meta-Daten auf dem Bild (z.B. Datum, Partienten-Daten, ...)
- Anpassung des Farb-Raum's / Farb-Werten / -Skalen
- Anpassung der Helligkeit (z.B. über Normalisierung)
- Denoising
- Segmentierung und Morphing
- ...

Aufteilen in Trainings- und Test-Daten (Train/Test-Splitting)

verschiedene Aufteilungs-Arten:

z.B.: nur in Trainings- und Test-Daten (meist 80 + 20%)



weitere Möglichkeit



Validierungs-Datensatz ist z.B. für den Vergleich unterschiedlicher trainierter Modelle sinnvoll

mögliche / genutzte Hyper-Parameter in klassischen ML-Modellen

- Lern-Rate (Learning Rate) →
- Epochen-Anzahl →
- Batch-Größe →
- ...

Hyper-Parameter sind nicht direkt Teile des ML-Modell's, sondern des genutzten KI-Systems

8.x.y. Modellierung

"The Movie Dataset" von kaggle.com

Einlesen der Daten

```
import pandas as pd
import yaml
import os

config = yaml.safe_load(open("../config.yml"))
os.chdir(config["datasets"][week2])
df_fim_metadata = pd.read_csv("movies_metadata.csv,
```

```
low_memory=False)
```

Anzeige der Daten für einen Film

```
filmNr = 6
print("Film-Titel: " + df_film_metadata["title"][filmNr] + \n)
print("Film-Beschreibung: " + df_film_metadata["overview"][filmNr]
```

Filtern bezüglich fehlender Daten

```
df_film_metadata[df_film_metadata["overview"].isna()]
```

zeigt die Datensätze an, die bei der Kurz-Beschreibung (overview) keinen Inhalt haben da aber über genau diese Spalte (Attribut) das ML passieren soll, werden nur die Filme mit Beschreibung in den Arbeits-Datenbestand übernommen danach resetten des Index

```
df_film_metadata =
    df_film_metadata[df_film_metadata["overview"].notna()]
df_film_metadata = df_film_metadata.reset_index()
```

Ähnlichkeits-Berechnungen

Count Vectorizer

Zeichen-Sequenz ("Zeichen-Folge, Wort, Wortgruppe) wird als Token bezeichnet

```
text = ["Der Film beschreibt die Liebe zwischen Helena und David
auf einer Safari.", "Helena und David entstammen unterschiedlichen
gesellschaftlichen Schichten.", "In Paris steht der Eifel-Turm."]

cv = CountVectorizer()
count_matrix = cv.fit_transform(text)
np.set_printoptions(precision=1)
print("Feature-Name: " + str(cv.get_feature_name()) + "\n")
print("Matrix: \n" + str(count_matrix.todense()))
```

alle Wörter werden einzeln kodiert und in der Matrix wird jeweils eingetragen, ob das Wort in der Text(-Zeile) (bzw. im Listen-Element) vorkommt (als 1 sonst 0)

Tf-idf Vectorizer

Weiter-Entwicklung

gemessen wird die inverse Häufigkeit eines Token im Dokument dabei geht man davon aus, dass die Bedeutung seltener Worte größer ist als die häufiger (z.B. die klassische Stop-Worte)

Term-Frequency $TF(\text{wort}, \text{dokument}) = \frac{\text{Anzahl}[\text{wort}]}{\text{Gesamtzahl}[\text{wörter}]}$

Inverse Document Frequency $IDF(\text{wort}, \text{sammlung}) = \ln\left(\frac{\text{Anzahl}[\text{dokumente_in}[\text{sammlung}]]}{\text{Anzahl}[\text{dokumente_mit}[\text{wort}]]}\right)$

Sammlung (auch Corpus, Collection) ist die Gesamt-Menge an betrachteten Texten

$TFIDF(\text{wort}, \text{dokument}, \text{sammlung}) = TF(\text{wort}, \text{dokument}) * IDF(\text{wort}, \text{sammlung})$

ergibt einen Wert zwischen 0 und 1 und kann als Wichtigkeit des Wortes in dem Text verstanden werden

```
...
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(text)
np.set_printoptions(precision=3)
print("Feature-Name: " + str(cv.get_get_feature_name()) + "\n")
print("Matrix: \n" + str(tfidf_matrix.todense()))
```

Anwendung Tf-idf-Vektorisierung auf die Kurz-Beschreibungen

```
tfidf = TfidfVectorizer(stop_word="english")
tfidf_matrix = tfidf.fit_transform(df_film_metadata["overview"])
```

Anzeige der Größe der ermittelten Wort-Matrix

```
tfidf_matrix.shape
```

um später wieder auf die Filme zurückschließen zu können wird ein Index über die Film-Titel angelegt

```
title_to_index = pd.Series(df_film_metadata.index, in-
```

```
dex=df_film_metadata["title"])
```

Suche der (10) nächst-ähnlichsten Filme über die K-nächste Nachbar-Methode (K-Nearest-Neighbors)
als Ähnlichkeits-Maß wird Cosine Similarity zwischen den Tfidf-Vektoren benutzt (Ähnlichkeit im mehr-dimensionalen Raum)

```
movie_matrix = csr_matrix(tfidf_matrix)
modell_knn = NearestNeighbors(metric="cosine", algorithm="aut",
                             n_neighbors=10, n_jobs=-1)
```

```
def recommend_films_by_title_knn(title, data, model, n_neighbors):
    model.fit(data)
    movie_idx(title_to_index(title))
    sim_scores, movie_indices = model.kneighbors(data[movie_idx],
                                                n_neighbors=n_neighbors+1)
    sim_scores = sim_scores.squeeze().tolist()
    for idx, movie_idx in enumerate(movie_indices.squeeze().tolist()):
        recommendation_list.append({"Title":
                                    df_film_metadata["title"][movie_idx], "Distance":
                                    sim_scores[idx]})
    return pd.DataFrame(recommendation_list).sort_values(
        by=["Distance"], ascending=False).reset_index(drop=True)
        [:-1]
```

Aufruf z.B. für die nächsten Film-Vorschläge zu "GoldenEye"

```
recommend_films_by_title_knn("GoldenEye", movie_matrix, modell_knn,
                             n_neighbors=10)
```

Analyse der Vorschläge / Qualitäts-Analyse

Beurteilen und Anpassen hier mal mit einem verkleinerten Daten-Bestand

```
import pandas as pd
import yaml
import os

config = yaml.safe_load(open("../config.yml"))
os.chdir(config["datasets"][week2])
df_film_ratings = pd.read_csv("ratings_small.csv",
                             low_memory=False)
```

Anzeige der Daten-Struktur

```
df_film_ratings.head()
```

ungebrauchte Spalte (Attribut) Timestamp wird gleich entfernt

```
df_film_ratings = df_film_ratings.drop("Timestamp", axis=1)
df_film_ratings = df_film_ratings.sample(frac=1).reset_index(
    drop=False)
```

trennen (splitten) in Trainings- und Test-Daten (80 / 20%)

```
split_factor = 0.2
n = int(split_factor * len(df_film_ratings))

df_train = df_film_ratings[:n]
df_test = df_film_ratings[n:]
```

Vergleichs-Ansatz

supervised Learning-Setting

hier werden also auch bekannte Ratings mit einbezogen

```
X = np.array([[1,1,5], [2,1,2], [3,1.2,8.7],
             [4,1,3], [1,0.8,9], [6,1,1.5]])
X
```

```
model = NMF(n_components, init="random", random_state=0,
            verbose=False)
W = model.fit_transform(X)
H = model.components_
```

Betrachtung der Differenz(en) zwischen den beiden Matrizen (Real-Wert und vorhergesagter Wert (Prediction))

```
print(np.subtract(X, np.dot(W,H)))
```

Bewertung der Ergebnis-Qualität über Root Mean Squared Error (RMSE;)

Wurzel aus der Mittleren quadratischen Abweichung

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}}$$

```
rmse = np.sqrt(mean_squared_error(y_pred=X, y_true=np.dot(W,H)))
print(RMSE: " + str(rmse))
```

```
train_user_data, train_movie_data,
test_user_data, test_movie_data, model = nf_model(df_film_ratings,
    df_train, sf_test)
```

Setzen einiger Hyper-Parameter und Trainieren des Modell's

```
model.fit([train_user_data, train_movie_data, df_train["rating"],
          batch_size=256, epoche=10, shuffle=True)
```

Kennwert loss gibt den Fehlerwert an (für Training), also quasi Trainings-Effekt / -Qualität danach Testen des Modell's mit den test-Daten

```
y_pred = model.predict([test_user_data, test_movie_data])
y_true = df.test["rating"].values

rmse = np.sqrt(mean_squared_error(y_pred=y_pred, y_true=y_true))
print(RMSE: " + str(rmse))
```

Vergleich der beiden ML-Ansätze

Vorschlags-System (unsupervised Learning) gegen Inhalts-Filter (supervised Learning)
Content-based Recommendation gegen Collaborated Filtering

zuerst noch einmal Bewertung des Recommender System's

```
recommend_films_by_title_knn("GoldenEye", movie_matrix, model_knn,
                             n_neighbors=10)
```

es kommen zwar in den Vorschlägen überwiegend Bond-Filme vor, aber Platz 1 ist ein Nicht-Bond-Film

direkte Analyse durch Vergleich der Beschreibungs-Texte

```
film="The way oft he Dragon"
print(film + ": " + df_film_metadata["overview"]
      [title_to_index[film]) + \n)
film="Johnny Steel Pigeon"
print(film + ": " + df_film_metadata["overview"]
      [title_to_index[film]) + \n)
film="Doctor X"
print(film + ": " + df_film_metadata["overview"]
      [title_to_index[film]) + \n)
film="Dream Work"
print(film + ": " + df_film_metadata["overview"]
      [title_to_index[film]) + \n)
```

gleiches Vorgehen auch für andere Filme

→ i.A. ist die Güte von unsupervised Learn-Modellen nicht besonders zuverlässig
schwierig ist vor allem das unkonkrete Domain-Wissen. Man benötigt gute Domain-Experten mit ev. eigenen Bewertungs-Methoden

nun Bewertung des Collaborative Filtering

Vorteil sind hier die (zum Vergleich) vorliegenden echten Bewertungen durch andere Nutzer

Analyse des loss-Wertes (Fehler-Wert des Modell's)

```
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Trainings-Fehler (loss)")
plt.ylabel("loss")
plt.xlabel("Epochen")
plt.legend(["Training", "Validation"], loc="upper_right")
```

Vorteil ist die Möglichkeit, beliebige Vergleichs-Methoden zu benutzen, da wir immer die echten (Real-)Werte, wie auch die vorhergesagten Werte haben

AutoML-Systeme

basierend auf CRISP-DM soll eine Automatisierung der Modell-Auswahl und –Optimierung erfolgen
dazu können die verfügbaren Daten aus den Bereichen "Modelling" und "Deployment" genutzt werden

bei:

www.levity.com

www.akkio.com

werden aus hochgeladenen, gelabelten Daten passende, optimierte Modelle ermittelt

wenn denn nun Computer vieles aus dem ML übernehmen, welche Aufgaben bleiben dann für die Data Scientists

- Sammeln von Daten
- Erschließen neuer Anwendungs-Felder
- Universalisierung von Systemen / Modellen
- verbesserte Lern-Methoden entwickeln
- Daten-Bereinigung und –Annotation (-Labeln)
- Demokratisierung von ML / KI

AutoML muss aber nicht effektiver als die händische Auswahl der Modelle / Parameter über z.B. statistische Methoden

Federated Learning

i.A. agieren mehrere einzelne Daten-Besitzer und / oder Daten-Quellen auf Teil-Daten-Beständen

Ziel ist es über deutlich größere Daten-Bestände zurückgreifen zu können, um die Modelle zu optimieren

bzw. überhaupt genug Daten für ein ML-System zusammenzubekommen

diese sollen zwecks gemeinsamer Auswertung gemeinsam genutzt werden können

Daten müssen an einer zentralen Stelle und mit Zugangs-Möglichkeit an dieser Stelle abgelegt sein

Beispiel-Szenarien:

- im Bereich Smartphones
 - Next-Word Prediction (Wort-/Text-Vorschläge)
 - Gesichts-Erkennung
 - Sprach-Erkennung
 - ...
- im Bereich IoT
 - automomes Auto(mobil)
 - Wearables
 - ...
- im Bereich Verwaltung / Organisation
 - Gesundheitswesen
 - Enterprise Machine Learning
 - ...
- ...

Privacy-Preserving ML

- mit zentralen Daten
 - Anonymisierung von Daten (Entfernen / Verschleiern sensibler / Personenbezogener Daten)
 - Homomorphic Encryption (ML auf codierten Daten)
- mit dezentralen Daten
 - Federated Machine Learning

beim Federated ML baut jeder Teilnehmer zuerst einmal auf seinen eigenen Daten ein Modell
die Modelle werden dann an eine zentrale Stelle gegeben
aus den Teil-Modellen wird ein zentrales Modell gebildet und die Parameter der Teil-Modelle
miteinander verknüpft
aus dem zentralen Modell heraus werden dann die lokalen Modelle geupdatet und mit den
lokalen neu berechnet / verwendet
Verfahren kann / sollte sich mehrfach wiederholen

8.x. Stimmungs-Analyse (Sentiment Analyse)

Ziel ist z.B. die Trennung von positiven und negativen Kommentaren

Daten-Basis Film-Bewertungen (sind schon gelabelt)

Bereitstellung der notwendigen Bibliotheken

```
import pandas as pd
import numpy as np

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from skorch import NeuralNetClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

zusätzlich noch einige Hilfs-Bibliotheken

```
from collections import Counter
import re
from bs4 import BeautifulSoup
import yaml
import os

from wordcloud import WordCloud
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import nltk
from nltk.corpus import stopwords
nltk.download("stopwords")
import spacy
%python -m download en_core_web_sm

import torch

from ffnn import NeuralNetModule
```

Anlegen der Datei-Pfade

```
config = yaml.safe_load(open("../config.yml"))
DATA_DIR = os.path.join("../", "data")
FILE_PATH = os.path.join(DATA_DIR, config["datasets"], [week3])
```

nach dem Einlesen der Daten sollte man sich die Daten auch mal ansehen

```
df = pd.read_csv(FILE_PATH, nrows=10_000)
df.head()
```

die Label's bedeuten:

1 ... "positiv" und 0 ... "negativ"

Ansicht der Label-Verteilung

```
df["label"].value(normalize=True)
```

hier gut ausbalancierter Daten-Bestand (positive und negative Bewertungen kommen gleichverteilt vor)

je ausbalancierter der Daten-Bestand (hinsichtlich der Label's) ist, umso besser lässt das Modell trainieren

Ansicht eines Datensatzes

```
df["text"].iloc[0]
```

Original-Texte stammen von Webseiten

sie enthalten noch HTML-Tag's und escaping-Symbole, z.B. "\n"

Bereinigung über

```
df[text] = df.[text].apply(lambda x, x.lower())
df[text] = df.[text].apply(lambda x, x_replace("\'", ""))
df[text] = df.[text].apply(lambda x, BeautifulSoup(x).text)
```

wobei BeautifulSoup die HTML-Tag's herausfiltert

nun klassischer Daten-Split in Trainings- Validierungs- und Test-Daten

```
train_df, test_valid_df = train_test_split(df)
test_df, valid_df = train_test_split(test_valid_df)
```

kurze Analyse der Daten (hier bezüglich der Text-Länge)

```
analysis_df = train_df.copy()
analysis_df["text_length"] = analysis_df.apply(len)
```

visualisierte Anzeige der Analys-Daten

```
sns.histplot(data=analysis_df, x="Text-Länge", hue="Annotierung",
             element="step");
```

Analyse der Wort-Häufigkeiten

```
text_corpus = " ".join(df["text"])
```

Verwendung einer eigenen Hilfs-Funktion zur Darstellung der sundsovielt häufigsten Wörter

```
def plot_most_commons_words(text_corpus, n):
    counter = Counter(text_corpus.split())
    rank, words, occurrences = [], [], []
    for idx, (word, occurrence) in
        enumerate(counter.most_common(n=n)):
            rank.append(idx)
            words.append(word)
            occurrences.append(occurrence)
    fig, ax = plt.subplots()
    ax.scatter(rank, occurrences, s=0.01)
    for idx, word in enumerate(words):
        ax.annotate(word, [rank[idx], occurrences[idx]])
    plt.title("Zipf's Gesetz")
    plt.ylabel("Vorkommen")
    plt.xlabel("Rang")
```

```
plot_most_common_words(text_corpus, 15)
```

Diagramm stellt die Wörter der häufigsten Worte dar

Bestätigung von ZIPF's Gesetz, das besagt:

In einer Häufigkeits-sortierten Reihe von Elementen (hier Wörter) ist die Häufigkeit des Auftretens umgekehrt proportional zum Platz in der Rang-Liste (PARETO- bzw. ZIPF-Verteilung)

da die häufigsten Worte typische Füllwörter sollte ein Entfernung vor der weiteren Verarbeitung in Betracht gezogen werden

```
pattern = re.compile(r'\b(' + r'|'.join(stopwords("english"))
    + r')\b\s+')
text_corpus_without_stopwords = pattern.sub("", text_corpus)
```

erneute analyse der Wort-Häufigkeiten

```
plot_most_common_words(text_corpus, 15)
```

zeigt dann doch schon mehr inhaltlich begründete Wort-Häufigkeiten

Anzeige einer Wort-Wolke als weitere Analyse-Möglichkeit
dafür wieder eigene Funktion

```
def draw_word_cloud(text_corpus):
    word_cloud = WordCloud(collocations = False,
        background_color = "white").generate(text_corpus)
    plt.inshow(word_cloud, interpolation="bilinear")
    plt.axis("off")
```

```
draw_word_cloud(text_corpus_without_stopwords)
```

nun eine einzelne Analyse der Texte nach ihrer gelabelten Stimmung ("positiv" oder "negativ")

```
positive_corpus = " ".join(analysis_df["test"].loc[
    analysis_df["label"] == 1])
negative_corpus = " ".join(analysis_df["test"].loc[
    analysis_df["label"] == 0])
```

```
draw_word_cloud(positive_corpus)
```

```
draw_word_cloud(negative_corpus)
```

zu beachten sind positive Wörter (z.B.: "gut"), die z.B. in der "negativen" Annotations-Gruppe als verneinte / negierte Wortgruppe vorkommen (z.B.: "nicht gut")

als nächstes wird eine Lemmatation durchgeführt, um nur noch Wortstämme für die weitere Bearbeitung zu nutzen

Prinzip:

```
nlp = spacy.load("en_core_web_sm")
doc = nlp("englischer Beispiel-Satz")

for token in doc:
    print(token, token.lemma_)
```

```
def enrich_lemmatized_text(df):
    df = df.copy()
    df["lemmatized_text"] = df["text"].apply(lambda
        x: " ".join([token.lemma_ for token in nlp(x)])
    )
    return df
```

Durchführen der Lemmatation auf allen Teil-Daten-Beständen

```
train_df = enrich_lemmatized_text(train_df)
valid_df = enrich_lemmatized_text(valid_df)
test_df = enrich_lemmatized_text(test_df)
```

es folgt das Tfidf-Vektorisieren

```
vectorizer = TfidfVectorizer()
train_X = vectorizer.fit_transform(train_df["lemmatized_text"]
    ).astype(np.float32)
valid_X = vectorizer.fit_transform(valid_df["lemmatized_text"]
    ).astype(np.float32)
test_X = vectorizer.fit_transform(test_df["lemmatized_text"]
    ).astype(np.float32)
```

Text liegt nun in Form von normalisierten numerischen Vektoren vor

für das Optimierungs-Set sollen die folgenden Modell-Typen geprüft werden:

- ein Entscheidungs-Baum
- ein Random Forrest
- eine logistische regression
- ein Künstliches Neuronales Netzwerk

Entscheidungs-Baum mit Hyper-Parameter-Suche über GridSearch

```
tree_clf = DecisionTreeClassifier()
tree_params = {"max_depth": list(range(10,101,20)) + [None],
               "min_samples_split": [2, 5]}
tree_search = GridSearchCV(tree_clf, tree_params)
tree_search.fit(train_df["label"])
best_tree_clf = tree_search.best_estimator_
```

nachfolgend nun die Evaluierung mit dem Validierungs-Datenbestand

```
def evaluate_clf(valid_X, labels, clf):
    predictions = clf.predict(valid_X)
    report = classification_report(labels, prdictions)
    print(report)
```

```
evaluate_clf(valid_X, valid_df["label"], best_tree_clf)
```

nächstes Modell: Random Forrest

```
forest_clf = Random ForestClassifier()
forest_params = {"n_estimators": list(range(10, 101, 20)),
                 "max_depth": list(range(10,101,20)) + [None],
                 "min_samples_split": [2, 5]}
forest_search = GridSearchCV(forest_clf, forest_params)
forest_search.fit(train_df["label"])
best_forest_clf = forest_search.best_estimator_
```

```
evaluate_clf(valid_X, valid_df["label"], best_forest_clf)
```

für die gegebenen (Validierungs-)Daten ergeben sich bessere Ergebnisse als beim besten Entscheidungs-Baum
als nächstes ein logistisches Modell

```
lr_clf = LogisticRegression()
lr prams = {"penalty": ["l1", "l2"], "max iter": [100],
            "C": np.logspace(-4, 4, 20), "solver": [liblinear]}
lr_search = GridSearchCV(lr_clf, lr_params)
lr_search.fit(train_df["label"])
best_lr_clf = lr_search.best_estimator_
```

```
evaluate_clf(valid_X, valid_df["label"], best_lr_clf)
```

obwohl die logistische Regression eigentlich kein wirkliches KI-Modell ist, liefert sie hier die bisher besten Ergebnisse
bleibt noch ein KNN als letztes zu testendes Modell
hier ein Feedforward Neural Network

```
neural_net = NeuralNetClassifier(module = NeuralNetModule,  
                                module_num_inputs = len(vectorizer.vocabulary_),  
                                max_epochs = 10, optimizer = torch.optim.Adam,  
                                iterator_train_shuffle = True, verbose = 0)  
neural_net.fit(train_X, train_df["label"])
```

benötigt eine vorher definierte Netz-Struktur (Schichten, ...)

```
from torch import nn  
  
class NeuralNetModule(nn.Module):  
    def __init__(self, num_inputs, num_units = 20,  
                 nonlin = nn.Net0()):  
        super(NeuralNetModule, self).__init__()  
        self.nonlin = nonlin  
        self.dense0 = nn.Linear(num_inputs, num_units)  
        self.dropout = nn.Dropout(0.2)  
        self.dense1 = nn.Linear(num_units, num_units)  
        self.output = nn.Linear(num_units, 2)  
        self.softmax = nn.Softmax(dim=-1)  
  
    def forward(self, X, **kwargs):  
        X = self.nonlin(self.dense0(X))  
        X = self.dropout(X)  
        X = self.nonlin(self.dense1(X))  
        X = self.softmax(self.output(X))  
        return X
```

Architektur ist in der Datei `ffnn.py` gespeichert
Hyper-Parameter-Suche über ein GridSearch ist nicht eingebaut, aber gut möglich (z.B.:
Epochen, Optimizer, ...)

```
evaluate_clf(valid_X, valid_df["label"], neural_net)
```

bringt leicht schlechtere Ergebnisse, als die logistische Regression
→ Free???-Theorem : aus den reinen Daten lässt sich ableiten, welches Modell später die
besten Ergebnisse bringt
die Auswahl des besten Modell's erfolgte mit dem Validierungs-Datenbestand
nun noch den finalen Test aller Modelle mit dem (noch unberührtem) Test-Datenbestand

```
evaluate_clf(test_X, test_df["label"], best_tree_clf)  
evaluate_clf(test_X, test_df["label"], best_forest_clf)  
evaluate_clf(test_X, test_df["label"], neural_net)  
evaluate_clf(test_X, test_df["label"], best_lr_clf)
```

Einfluss der Größe des Trainings-Datenbestand's auf die Qualität des Modell's

```

for nrows in [100, 500, 1_000, 10_000, 49_000]:
    print(f"#(nrows)")
    print("-"*55)
    train_df = read_and_parse_train_df(nrows)
    vectorizer = TfidfVectorizer()
    train_X = vectorizer.fit_transform(train_df["text"].astype(
        np.float32))
    test_X = vectorizer.transform(test_df["text"].astype(
        np.float32))

    neural_net = NeuralNetClassifier(module = NeuralNetModule,
        module_num_inputs = len(vectorizer.vocabulary_),
        max_epochs = 10, optimizer = torch.optim.Adam,
        iterator_train_shuffle = True, verbose = 0)
    neural_net.fit(train_X, train_df["label"])

evaluate_clf(test_X, test_df["label"], neural_net)
print("\n"*2)

```

wenige Datensätze bringen noch einen F1-Score, der praktisch einem Raten entspricht bei steigenden Datensatz-Zahlen steigt dann auch die Qualität der Voraussage deutlich an bei sehr großen Zahlen wird die Steigerung der Qualität zwar immer kleiner, aber die Veränderung von 1, bis 2 % hier (z.B. bei 88%) ist deutlich höher zu bewerten, als eine Steigerung von 1 bis 2% nahe am "Rate-Werte" von 50% allgemein gilt, je mehr Trainings-Daten verwendet werden, umso besser wird das Modell es besteht dafür aber keine Garantie

8.x. Gesten-Erkennung (Gebärden-Sprache) / Bilder-Klassifikation

Thema der Computer Vision (→)

Arbeits-Bereiche:

- Erkennung von Bild-Inhalten (Object Detection / Tracking)
- Gesichts-Erkennung (Face Detection)
- Pose Estimation ()
- Image Segmentation ()
- 3D Scene Reconstruction (Rekonstruktion von 3D-Szenen)
- General Adversarial Networks ()
- ...

Convolutional Layer

filtern Strukturen aus den Bildern

praktisch wird mit einer "Lupe" (kernel genannt) immer ein kleiner Bild-Abschnitt analysiert und neu verrechnet

Suche nach Mustern

bei Bildern mit Grautönen wird nur ein Verarbeitungs-Kanal gebraucht

für Farb-Bilder werden für die drei Kanäle (Rot-Grün-Blau) auch drei Verarbeitungs-Kanäle verwendet

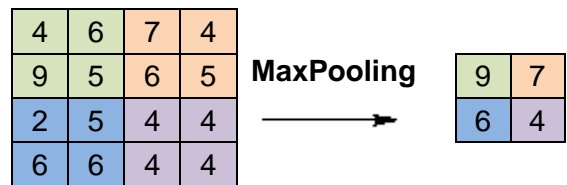
Pooling Layer

sollen die Komplexität von Bildern reduzieren

Reduktion der Daten-Menge, die an die nächste Schicht (Layer) weitergegeben wird

wichtige Pooling Layer ist der Max Pooling Layer

von einem Ausschnitt wird nur der Maximal-Wert weitergereicht



Fully Connected Layer

ein-dimensionale KNN-Schichten, bei der jede Nachfolger-Zelle (Neuron) gewichtet Eingangs-Daten von allen Zellen der Vorgänger-Schicht erhalten

der Übergang von den zwei-dimensionalen Bilder-Daten zu den ein-dimensionalen Strukturen der Fully Connected Layer erfolgt über ein **Flatten Layer**

Das KNN wird von einem **Output Layer** abgeschlossen, der Wahrscheinlichkeits-Werte für die einzelnen Kategorien (zu erkennende Objekte usw. usf.) bereitstellt

Adversarial Attacks

sind Angriffe auf trainierte KNN's mit manipulierten Daten, die zu eindeutigen Fehl-Interpretationen führen

Vor-Analyse der Daten / Übersicht

verwendet wir das Sign Language MNIST Dataset

z.B. bei kaggle.com

enthält Gebärden-Sprach-Gesten für das Alphabet in amerikanischer Zeichen-Sprache (ASL, American Sign Language)

die 34'000 Bilder sind einheitlich 28x28 Pixel groß und haben nur Grau-Stufen

manueller Download: <https://kaggle.com/dazamunge/sign-laquage-mnist>

oder als JSON

Erläuterung des Vorgehens unter <https://kaggle.com/docs/api>

username: openHPI

key: das_ist_der_key

Ansehen der verfügbaren Dateien

```
import yaml
import os
config = yaml.safe_load(open("../config.yml"))
os.chdir(config["datasets"]["week4"])
```

die daten sind als Grau-Werte in einer CSV-Datei enthalten

es gibt eine separate Trainings-Daten und eine Test-Daten-Datei

Anzeige

```
import pandas
df_train = pd.read_csv("sign_mnist_train.csv")
df_test = pd.read_csv("sign_mnist_test.csv")

df_train.head()
```

Jedes Bild steht in einer Reihe (Zeile), vorne sind eine laufende Nummer und ein Label
Label sind die durchnummerierten Buchstaben im Alphabet
bei den Test-Daten sind keine Label enthalten

```
X_train = df_train.drop(["label"], axis=1).values
X_test = df_test.drop(["label"], axis=1).values
```

Speichern der Label's als Ergebnis-Daten

```
y_train = df_train["label"]
y_test = df_test["label"]
```

Erzeugen von rechteckigen Bildern (2D) aus den Zeilen-Daten (1D)

```
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)
```

Anzeige

```
X_train.shape
```

Analyse der Label

```
np.sort(df_train["label"].unique())
```

Die Übersetzung der Label (Nummern) erfolgt durch eine eigene "Klartext"-Liste

```
alphabetic_label = ["A", "B", "C", "D", "E", "F", "G", "H", "I",
                    "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U",
                    "V", "W", "X", "Y"]
```

"J" und "Z" fehlen, da es für diese Buchstaben keine Bilder gibt, die Darstellung erfolgt in der amerikanischen Zeichen-Sprache als Bewegung
Anzeige eines Bildes mit Label

```
show_example_pictures(X_train, y_train, alphabetic_label)
```

Verteilung der Label's im Trainings-Daten-Set

```
show_label_frequency(df_train)
```

Umsetzung / Aufbau des CNN (Convolutinal Neural Network)

für den Output benötigen wir Werte-lose Label's
deshalb One-Hot-Encoding

```
lb = LabelBinarizer()
lb.fit(y_train)
y_train_oh = lb.transform(y_train)
y_test_oh = lb.transform(y_test)
```

Definition des Modell's (das CNN)
praktisch ein "minimales" Basis-Modell (geeignet für Bild-Daten)

```
model = Sequential()
model.add(Conv2D(128, kernel_size=(5, 5), strides=1, padding="same,
activation="relu", input_shape=(28,28,1)))
model.add(MaxPool2D(pool_size=(3, 3), strides=2, padding="same"))
model.add(Flatten())
model.add(Dense(units=256, activation="relu"))
model.add(Dropout(rate=0.3))
model.add(Dense(units=24, activation="softmax"))
model.summary()

model.compile(optimizer="adam", loss="categorical_crossentropy",
metrics=["accuracy"])
```

Anzeige des Modell's
als Graphik

```
plot_model(model, show_shapes=True, show_layer=True)
```

Training des Modell's mit z.B. 20 Epochen
mit Anzeige der Zwischenstände

```
history = model.fit(X_train, y_train_oh, epochs=20, shuffle=1,
validation_data=(X_test, y_test_oh))
```

Anzeige der Genauigkeit über die Epochen

```
show_accuracy_loss_plot(history, num_epochs=20)
```

Verbesserung des Modell's durch Hinzufügen weiterer Layer
neue Layer sind fett gedruckt

```
model = Sequential()
model.add(Conv2D(128, kernel_size=(5, 5), strides=1, padding="same,
```

```

        activation="relu", input_shape=(28,28,1))
model.add(MaxPool2D(pool_size=(3, 3), strides=2, padding="same"))
model.add(Conv2D(64, kernel_size=(2, 2), strides=1, padding="same,
activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2), 2, padding="same"))
model.add(Conv2D(32, kernel_size=(2, 2), strides=1, padding="same,
activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2), 2, padding="same"))
model.add(Flatten())
model.add(Dense(units=256, activation="relu"))
model.add(Dropout(rate=0.3))
model.add(Dense(units=128, activation="relu"))
model.add(Dropout(rate=0.3))
model.add(Dense(units=24, activation="softmax"))
model.summary()

model.compile(optimizer="adam", loss="categorical_crossentropy",
              metrics=["accuracy"])

```

Optimierung des Modell's

Erweiterung des Daten-Set's

z.B. durch einfach Bild-Veränderung

- Verschieben
- Verzerren
- Drehen
- Spiegel

Drehen und Spiegeln bei Gesten nur teilweise oder garnicht nutzbar

über Bild-Generator

```

data_augmentation = ImageGenerator(rotation_range=0,
    height_shift=0.2, width_shift=0.2, shear_range=0,
    horizontal_flip=False, vertical_flip=False)
data_augmentation.fit(X_train)

```

der Test-Daten-Bestand wird derzeit nicht verändert, es sollen nur mehr Trainings-Daten zur verfügung gestellt werden

es wird das gleiche (erweiterte) Modell

nur mit den veränderten Bilder-Daten und mehr Epochen

```

history = model.fit(data_augmentation.flow(X_train, y_train Oh,
    batch_size=128), epochs=40, shuffle=1,
    validation_data=(X_test, y_test Oh))

```

Transfer Learning / Pretrained Machine Learning

Definition(en): Transfer Learning

Transfer Learning ist ein Bereich des Maschinellen Lernen's, bei dem die aus einem (funktionsierenden / erprobten) Modell / einer Aufgabe gewonnene Erkenntnisse / Ergebnisse auf ein anderes – ähnlich strukturiertes – Problem anzuwendet werden.

praktisch werden – meist sehr große – fertig trainierte Modelle von Firmen oder Forschungseinrichtungen zur Verfügung gestellt

Beispiel: Modell VGG16

bearbeitet Bilder mit einer Größe von 48x48x3 (also auch Farbe!)
trainiert auf den Bilder-Bestand von image-net.org

```
X_train = df_train.values
X_test = df_test.values
X_train = X_train.reshape(-1,28,28,1)
X_test = X_test.reshape(-1,28,28,1)
x_train_t = np.stack([X_train.reshape[0],28,28])*3,
                axis=3).reshape(X_train.shape[0],28,28,3)
x_test_t = np.stack([X_test.reshape[0],28,28])*3,
                axis=3).reshape(X_test.shape[0],28,28,3)
x_train_t.shape, x_test_t.shape
x_train_tt = np.asarray([img_to_array(array_to_img(im,
                scale=False).resize((48,48)) for im in x_train_t)]
x_test_tt = np.asarray([img_to_array(array_to_img(im,
                scale=False).resize((48,48)) for im in x_test_t)]
x_train_tt.shape, x_test_tt.shape
```

```
base_model = VGG16(weights='imagenet', include_top=False,
                input_shape=(48, 48, 3))
base_model.trainable = False

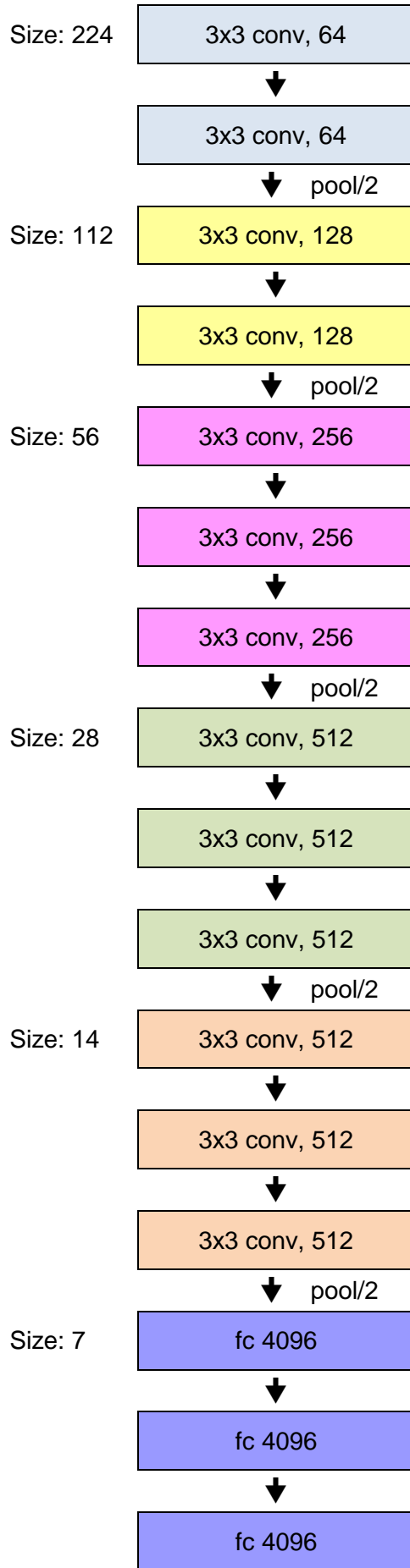
train_ds = preprocess_input(x_train_tt)
test_ds = preprocess_input(x_test_tt)
```

```
flatten_layer = layers.Flatten()
dense_layer1 = layers.Dense(50, activation="relu")
dense_layer2 = layers.Dense(36, activation="relu")
prediction_layer = layers.Dense(24, activation="softmax")

model = models.Sequential([base_model, flatten_layer, dense_layer1,
                dens_layer2, prdeiction_layer])

model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer="adam", metrics=["accuracy"])
```

Grobaufbau des VGG16-Modell's



Bewertung der Ergebnisse (der Lern-Prozesse)

Betrachtung des Loss (Verlust) und der Accuracy (Genauigkeit)

```
(ls,acc) = model.evaluate(x=X_test, y=y_test_oh)
print("Modell-Genauigkeit (Accuracy): {}".format(acc*100))
```

Anpassung der Ergebnisse (über das um J und Z gekürzte Alphabet) auf das reguläre Alphabet

```
predictions = np.argmax(model.predict(X_test), axis=1)
for i in range(len(predictions)):
    if predictions[i]>=9:
        predictions[i]+=1
```

Analyse der Ergebnisse nach den einzelnen Kategorien

```
classes = ["Klasse "+str(i) for i in range(25) if i!=9]
print(classification_report(y_test, predictions,
                           target_names= classes))
```

Nutzung einer Confusion Matrix (Konfusions-Matrix)

Vergleich der tatsächliche Klasse mit der vorhergesagten verglichen auch Anzeige, welche Klassen wiehäufig falsch vorhergesagt wurden (für welche Buchstanen / Zeichen ergeben sich die meisten Fehler je Klasse) im Besten Fall stehen nur 100%-Werte in der Diagonalen

```
cm = confusion_matrix(y_test, predictions)
cm = pd.DataFrame(cm, index = [i for i in range(25)] if i !=9,
                  columns = [i for i in range(25) if i !=9])
```

Ausgabe als HeatMap (Stärke- oder Intensitäts-Matrix)

```
plt.figure(figsize=(15,15))
sns.heatmap(cm, cmap="Blues", lincolor="black", linewidth=1,
            annot=True, fmt="")
```

Anzeige von Beispiel-Bildern mit der zugehörigen Vorhersage

```
show_example_pictures(X_test, y_test, alphabetic_label,
                      predictions)
```

Zusammenfassung / Fazit: sehr idealistisch orientierte Ergebnisse, da ideale Bilder in einem sehr idealen Daten-Set

9. Künstliche Neuronale Netzwerke in der Bio-Informatik

z.B. mit der Python-Bibliothek DeepChem

→ <https://deepchem.io/>

derzeit scheinbar nur für Linux- und Mac-Systeme

10. gesellschaftliche und ethische Aspekte rund um die KI

Daten sind das neue Öl
nach Kiran BHAGESHPUR
original: "The world's most valuable resource
is no longer oil,
but data."

Mehrwert von KI

- generell
 - Lösbarkeit von komplexen Problemen, die mittels traditioneller Programmierung kaum / nur schwer zu lösen sind
 - Lösung von Problemen mit sehr vielen – unterschiedlichen – Daten, deren Zusammenhänge / Strukturen nicht offensichtlich sind
 - automatisierte Verarbeitung von vielen Daten
 - veränderliche / dynamische Probleme / Zusammenhänge
 - Unterstützung und Steigerung der Effizienz von Menschen / menschlichen Tätigkeiten
 - Übernahme von (ermüdenden / belastenden / gefährlichen) Routine-Tätigkeiten

Anwendungs-Gebiet: Medizin

- (Früh-)Erkennung von Erkrankungen (z.B. Krebs, seltene Krankheiten, ...)
- personalisierte Medizin / Tele-Medizin
- Unterstützung von Ärzten bei Routine-Tätigkeiten / Erstellung von
- ...

Anwendungs-Gebiet: Automobile / Verkehr

- Verkehrs-Planung in Städten
- Ampel-Steuerung
- autonomes Fahren
- Sicherheits-Systeme im Auto
- ...

Anwendungs-Gebiet: Industrie / Produktion / Chemie-Anlagen

- vorbeugende Wartung und Instandhaltung / Ausfall-Vorhersage
- Betrugs-Erkennung (doppelt gestellte Rechnungen, ...)
- Kapazitäts-Planung
- Prozess-Optimierung / -Automatisierung
- ...

Anwendungs-Gebiet: Privat-Leben

- Sprach-Steuerung (Sprach-Assistenten (z.B. Alexa, google, Cortana, ...))

- Vorschlags-Systeme in Shop's, Streaming-Diensten
- Suchmaschinen
- ...

nach Kathrina ZWEIG sollten die folgenden Methoden und Aspekte in die Bewertung und Zulassung von KI-Systemen einfließen /14/

OMA-Prinzip

• Operationalisierungen	Messbarmachung sozialer Konzepte später auch: Messbarmachung der Güte eines / des Algorithmus
• Modell des Problem's	Passung der Daten / des Modell zum Problem (z.B. nutzt eine Bahn-Auskunft auch den Abfahrtsplan der Züge und die notwendigen Umsteige-Zeiten)
• Algorithmus	Eignung des Algorithmus zur Lösung des Problem's

-
-
-

Eine Super-KI wäre entweder das Beste oder das Schlimmste, das der Menschheit zustößt.
Stephen HAWKING, "Kurze Antworten auf große Fragen"

Bedeutung der KI wird wohl auch durch das entgegengebrachte Interesse charakterisiert kein Internet-Dienst hat in so kurzer Zeit die Millionen-Nutzer-Grenze überschritten Spotify brauchte dafür 150 Tage, Instagram 75 Tage. Bei ChatGPT war die Millionen-Nutzer.Grenze schon nach 5 Tagen erreicht.

Zum Vergleich noch ein paar populäre Global-Player: Netflix brauchte für 1 Millionen Nutzer 3,5 Jahre, Airbnb 2,5 und Twitter 2 Jahre. Selbst das allgegenwärtige Facebook hatte erst 10 Monaten solche Nutzerzahlen.

direkt oder indirekt wird wohl jedem klar, mit was gewaltigen wir es hier zu tun haben auch wenn es nur der Anfang (aus allgemein-populärer Sicht) ist, die Menschen erahnen Potential und Gefahren

TikTok 9 Monate
Pinterest 41 Monate
Telegram 61 Monate
Uber 70 Monate

Bedeutung von Daten im Machine Learning

Daten sind der entscheidende Faktor in der KI bzw. dem ML

Probleme:

- Welche Daten stehen zur Verfügung?
- Existieren genug Daten?
- Sind die Daten überhaupt / richtig vorverarbeitet worden?
- Gibt es Veränderungen in den Daten? → Data Drift
- Sind die Daten repräsentativ?
 - Wie wurden die Daten erhoben?
 - Entsprechen die Daten der Realität? Repräsentieren die Daten die Realität?
 - Gibt es Neben-Effekte durch Erhebung, Digitalisierung, Speicherung, Übertragung, ...
- Sind die Daten objektiv erfasst worden?
- ...

Wenn es mehrere Möglichkeiten gibt,
eine Aufgabe zu erledigen,
und eine davon in einer Katastrophe endet
oder sonstwie unerwünschte Konsequenzen nach sich zieht,
dann wird es jemand genau so machen.
Edward A. MURPHY jr.

Einschränkungen von KI

Alles, was schiefgehen kann, wird auch schiefgehen.
Edward A. MURPHY jr.

allgemeine Risiken / Einschränkungen

- **menschliche Fehler** durch Daten-Analysten, Daten-Wissenschaftler, Programmierer, ...
- **fachliche Fehler** durch Daten-Analysten, Daten-Wissenschaftler, Programmierer, ...

-
- unechte / Schein-Korrelationen
 - Bias / Befangenheit
 - SIMPSON Paradox ()

- **Manipulation** um die Leistung des Modell's besser darstehen zu lassen
 - politische / ökonomische / ... Interessen
 - Daten-Anpassung an die (erwarteten) Ergebnisse
 - Biased Training
 - manipulative Visualisierung
- **Missbrauch**
 - Diskriminierung
 - Überwachung
 - Datenschutz-Verletzungen
- **Monopol** Macht-Missbrauch möglich
 - wenige, große Akteure / Unternehmen
 - mangelnde Kontrolle
 - fehlende Chancen-Gleichheit

problematisch

- Mangel an Wissen
- Mangel an Ethik

behebbar durch (staatlicher) Regulierung, (Kompetenz-orientierter) Bildung und offene gesellschaftliche Diskussion

Bedeutung / Wichtigkeit der Daten

Falsch-Interpretation von Daten

US Air Force (amerikanische Luftwaffe) analysierte die heimkehrenden beschädigten Flugzeuge. Diese hatten verschiedene Einschußlöcher. Ev. könnte man die Flugzeuge an diesen Stellen verstärken / panzern. Allerdings verschlechtern Panzerungen i.A. die Leistungsfähigkeit der Flugzeuge. Sie werden schwerer und damit langsamer, schlechter manövrierfähig und verbrauchen mit Kraftstoff. Nun kann man den Schluß ziehen, nur dort die Flugzeuge zu verstärken / panzern, an denen viele Einschüsse / Beschädigungen sind. Der Trugschluß bei der Interpretation dieser Daten war / ist es, die Flugzeuge nicht als das zu sehen, was sie wirklich waren, nämlich Flugzeuge, die es zurückgeschafft haben. Die Flugzeuge, die an anderen Stellen beschädigt wurden, haben / hatten es gar nicht zurück geschafft.

Problem der Ausreißer / Outlier

? gibt es viele Ausreißer?

? bilden die Ausreißer eine Gruppe / ein Cluster?

? gehören die Ausreißer wirklich zum betrachteten Zusammenhang?

Data bzw. Concept Drift ()

Häufig verändern sich auch die Ziel-Daten. Beim Spam z.B. dadurch, dass die Spammer angepasste Mails schicken, die die üblichen Filter austricksen. Die Filter müssen sich dann im Nachhinein an die neue Art der Spam's gewöhnen und neu dazulernen.

Data Drift

hier verändern sich die Daten innerhalb des Modells z.B. durch (stärkere) Stör-Einflüsse das Modell selbst bleibt aber weiterhin gültig

Concept Drift

liegt dann vor, wenn sich die Umgebungs-Bedingungen für das Modell ändern, aber nicht die erfassten Daten

benötigte und verfügbare Daten-Mengen

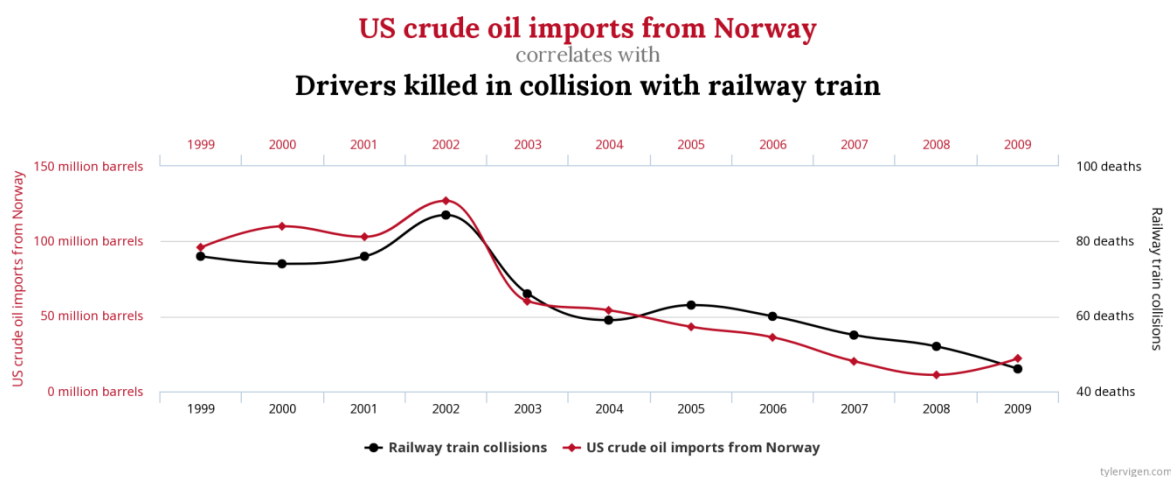
praktisch wird im Lern-Prozess nur mit einer Stichprobe aus der Gesamtheit (gesamter Daten-Bestand gearbeitet)

zum Einen muss die Stichprobe die Gesamt repräsentieren – also auch außergewöhnliche Werte, aber auch die Menge an Daten muss passend zur Gesamt sein

traditionelles ML (ohne KNN oder Deep Learning) kommt mit relativ wenig Daten aus größere Datenmenge bringen ab einem bestimmten Bereich keinen Mehrwert mehr für Deep Learning sind im Allgemeinen große Daten-Menge förderlich, es existiert ein annähernd linearer Zusammenhang zwischen Datenmenge und der Performance des Modells

Korrelation der Daten

Ein flüchtiger auf das nachfolgende Diagramm zeigt zwei Kurven die gut zueinander passen. Auch die mathematische Analyse ergibt eine Korrelation von 0,9545 (entspricht: 95,45 %).



"Zusammenhang" zwischen US-Importen von norwegischem Öl und der Anzahl von tödlich Verunglückten Autofahrern an Bahnübergängen
Q: www.tylervigen.com/spurious-correlations

Aber beim genaueren Betrachten fällt uns nicht wirklich ein nachvollziehbarer Zusammenhang zwischen beiden Daten-Reihen ein. Sie sind sich einfach zufällig ähnlich. Ein Computer-System kann diese Zufälligkeit nicht zwangsläufig erkennen. Vielleicht entscheidet eine Künstliche Intelligenz nun einach die Öl-Importe auf Null herunter zu fahren,

womit dann – laut dem "Zusammenhang" – auch die Zahl der Toten an den Bahn-Übergängen auf Null sinken sollte. Ob das wirklich funktioniert?
Wir müssen also sauber zwischen der Kausalität und einer echten oder scheinbaren Korrelation unterscheiden.

Ansatzpunkte

- gesunder Menschen-Verstand
- Lösen von Wunsch-Vorstellungen / Lösen von Gerüchten / Image-Betrachtungen / Voreinnahmen / Klischee's / Assoziationen / ...
- differenzierte Betrachtung der Evaluations-metriken und deren Bedeutung
- Feedback von Fachexperten
- (allgemeine) Vorsicht bei Interpretationen
- ausgiebiges Testen, auch mit Live-Eingaben
- fundierte Daten-Exploration, um ein besseres Daten-Verständnis zu erlangen
- Abwägung von Performance und Erklärbarkeit

Vertrauen an die Leistungs-Fähigkeit der KI

Aufgaben:

1. *Analysieren Sie ein selbstgewähltes weiteres Diagramm von der Webseite www.tylervigen.com/spurious-correlations! Finden Sie realistische Erklärungen für die "Zusammenhänge"!?*

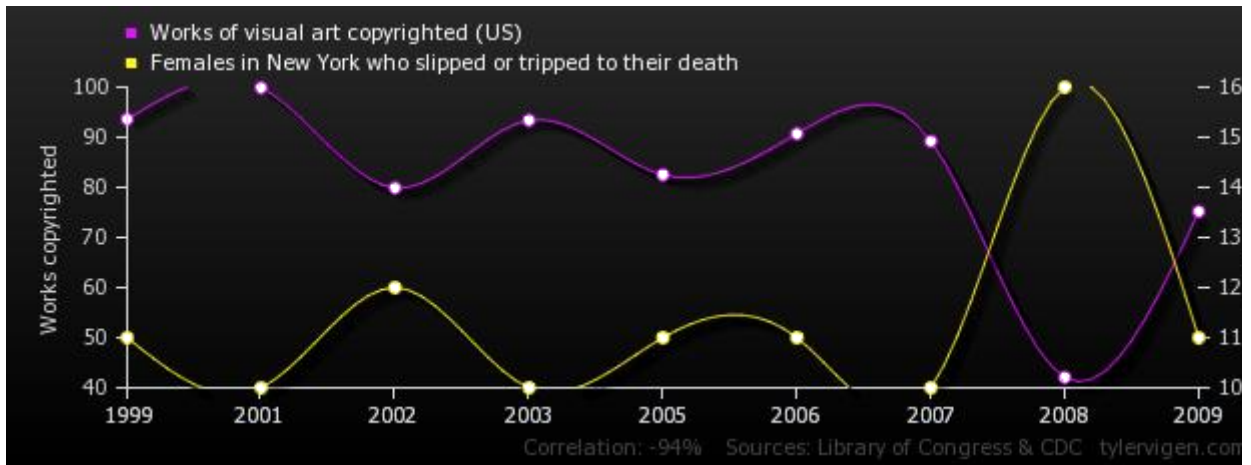
2.

für die gehobene Anspruchsebene:

3. *Stellen Sie den folgenden Zusammenhang (?) mittels eines Tabellen-Kalkulations-Programm graphisch (2 Rubriken- / Sekundär-Achse(n) benutzen!) dar! Berechnen Sie mit Hilfe einer recherchierten Formel / Tabellenblatt-Funktion die Korrelation zwischen beiden Daten-Reihen! Gibt es da einen Zusammenhang (... also ich als verheirateter Mann kann dazu sagen ... (Pst! Ich glaub' meine Frau kommt.)?)*

Jahr	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Heirats-Rate in New York	7,3	7,1	7,6	7,3	6,8	6,8	6,8	6,9	6,8	6,6	6,5	6,5
Morde mit einem stumpfen Gegenstand	242	210	199	184	129	102	152	115	111	95	79	65

Daten-Q: <https://tylervigen.com/page?page=2>



"Zusammenhang" zwischen der Anzahl urheberrechtlich geschützter Werke und der Anzahl Frauen in New York, die tödlich gestolpert oder ausgerutscht sind
 Q: https://tylerjvigen.com/view_correlation?id=1388

Korrelation liegt hier bei -0,9398, was bedeutet, das es (ziemlich sicher) – eben zu 93,98 % – einen entgegengesetzten Zusammenhang gibt

Diskriminierung in und mit Daten

meint Befangenheit / Vorbelegung / Vorurteile /

Vorurteile und Sterotypen in Daten

- fehlende Diversität bei den Trainings-Daten
- fehlende Diversität bei den Test-Daten
- selbstzufriedene Wissenschaftler und Nutzer (schließlich hat man das gefunden, was man gesucht hat)
- unzureichende Fehler-Analyse
- ...

vorhandene Fehler oder Unzulänglichkeiten in den Daten, werden durch die Modelle i.A. verstärkt

Definition(en): Bias

Unter einem Bias versteht man eine Verzerrung, Befangenheit oder ein Vorurteil bezüglich der Realität.

technischer / technical Bias

Fehler in der Daten-Erhebung und / oder -(Vor-)Verarbeitung passiert z.B. durch einen falsch kalibrierten oder defekten Sensor dieser liefert falsche oder keine oder immer die gleichen Werte

vorhandener / bereits existierender / pre-existing Bias

verursacht durch in sich schon diskriminierende Daten (Attribute) z.B. in historischen Daten alte Interpretation von Daten (Daten-Erfassung unter ganz anderen Aspekten / historischen Gegebenheiten)

da vor einem bestimmten Zeitpunkt die Daten nur analog (auf Papier od.ä.) vorliegen und diese nicht (noch nicht) digitalisiert wurden, entsteht aus den aktuellen (digital vorliegenden Daten) ein falsches Bild (praktisch ungünstige Wahl der Stichprobe für die Trainings-Daten)

emergenter / emerging Bias

fehlerhafte Auswertung / Interpretation der Ergebnisse
eher menschlicher Fehler

Problemfeld: Korrelation gegen Kausalität (→)

Algorithmic Bias

ungeeignete Wahl der Algorithmen / Modelle für den Anwendungsfall
durchgehende Ungleichbehandlung von Gruppen z.B. Entscheidungen

```
if maennlich:
    # Bearbeitung für Männer
    ...
else:
    # Behandlung aller anderen Fälle quasi als weiblich
    ...
```

Selection Bias

tritt dann auf, wenn die Stich-Probe (z.B. zum Trainieren und zum Testen) nicht die Gesamtheit für den Real-Einsatz abbildet

Exclusion Bias

tritt auf, wenn scheinbar unwichtige Merkmale / Features / Beobachtungen, ignoriert oder bewusst ausgespart werden

z.B. um Modelle zu verkleinern, schneller zu machen, ... (z.B. auch auf Druck des Auftrag-Gebers)

Measurement Bias

durch fehlerhafte, falsch geeichte, mit fehlerhafter Firmware versehene Sensoren
systematische Messfehler (z.B. auch durch ungeeignete Umgebungs-Bedingungen; Schutz-Abdeckungen; Montage-Fehler, ...)

Trainings-Daten stimmen nicht mit der Gesamtheit / Realität überein

Experimenter's Bias

Experten / Nutzer / Forscher trainieren ihre Modelle solange, bis ihre Hypothese bestätigt wird

gleiches gilt für das Setzen von Hyper-Parametern

Biased Reality

es besteht schon ein Bias in der Domäne, der unterschwellig auch in den Daten besteht

Gründe für Bias

- fehlende Generalisierbarkeit der Daten
- Klassen-Unausgeglichenheiten (z.B. Männer / Frauen, ...)
- fehlerhafte Konfiguration von Daten-Erhebungen
- Auswahl von Charakteristiken / limitierte Charakteristiken
- ableitbare Informationen, welche diskriminierend sind
- allgemeine Befangenheit / Konservativität
- Ungleichheiten in der realen Welt
- fehlendes Fachwissen
- ...

4 % aller amazon-Nutzer schreiben 50 % der Artikel-Bewertungen

bisher mehr durch Männer besetzte Posten erzeugen in den Daten eine Überbetonung des Männlichen – Männer bekommen in einer entsprechenden KI (von amazon entwickelt, aber dann angeblich nicht eingesetzt) oft eher den Job

dabei standen gar keine eigentlichen Geschlechts-Daten als Attribut zur Verfügung, aber das system hat Begriffe wie "Frau" (z.B. Leiter des Frauen-Fußball-Team's) als negative Werte herausgefiltert

ein weiterer ähnlicher Effekt entstand durch Abschlüsse die an Frauen-Universitäten usw. erreicht wurden

Probleme bei der Gesichts-Erkennung dadurch, dass zum Training mehr Bilder von weißen Personen als von farbigen genutzt wurden (einfach weil sie so vorlagen)

Bild-Erkennung die eine von oben betrachtete Schildkröte als Waffe erkennt, weil sie nur mit Bildern von seitlichen Bildern trainiert wurde

ein Bild-Erkennung "erkennt" Wölfe nicht an ihrem Ausehen oder Merkmalen, sondern am Schnee im Hintergrund, weil die anderen Hunde-Bilder mit eher grünem Hintergrund genutzt wurden

Gerichts-KI verhindert die vorzeitige Entlassung von Farbigen Gefängnis-Insassen, weil zur nicht erfolgreichen Rehabilitation von diesen mehr Daten vorlagen, als von weißen Gefangenen

"intelligenter" Seifenspender erkennt nur weiße Hände, nicht aber die von Farbigen
auch hier wurde die KI hauptsächlich mit Bilder weißer Hände trainiert

Probleme mit der Bewertungs-Funktion

Ein häufig an Möbeln anstoßender Saug-Roboter sollte mit KI zur sanfteren Reinigung gebracht werden. Belohnungs-Funktion bezog sich auf die Minimierung der Möbel-Kontakte und der Schnelligkeit der Reinigung.

Was geschah: Der Saug-Roboter fuhr sehr schnell wild rückwärts durch den Raum, eckte öfter an, als vorher. Es gab kaum noch einen Reinigungs-Effekt.

Ursache: der Saug-Roboter hatt nur vorne Kontakt-Sensoren.

Welches Qualitäts-Maß ist für eine Problem-Stellung angebracht?
Wer wählt das Qualitäts-Maß aus, der Daten-Analysist, der Programmierer, der KI-Benutzer und / oder die Gesellschaft? Wer gehört dann zu der (ausgewählten) Gesellschaft?
Was passiert mit ungerecht behandelten Personen? Wer entscheidet nun?
Welche Fehler-Rate ist akzeptabel? Wer legt sie fest?
Bringt die KI wirklich einen Vorteil / den erwünschten (positiven) Effekt? Auf welcher Basis werden das ursprüngliche Vorgehen und die neue Technik miteinander verglichen? Welches Vergleichsmaß wird verwendet?

Möglichkeiten, um einer Diskriminierung durch Daten entgegenzuwirken:

- Anonymisierung von Daten
- Entfernung (offensichtlicher) diskriminierender Daten (z.B. keine Vornamen bei Menschen-betreffende Entscheidungs-Modellen)
- Daten-Exploration
 - Wie stehen die Daten zueinander in Beziehung
 - Lassen sich die Zusammenhänge mit den vorhandenen Daten abbilden?
- Ceteris-paribus-Test's (es werden immer nur die Variablitäten innerhalb eines Attribut's getestet (alle anderen Variablen bleiben unverändert))
- "Shadow-Mode" / passiver Modus: Live-Einsatz ohne Verwertung der Prognosen
- ausgiebige Test's
- gesunder Menschen-Verstand
- ...

Ansatzpunkte zum Verhindern von Bias

- Model Governance auf Bias und Fairness
- Bewußtsein und Aufklärung von Daten
- kritische Begutachtung und Kontrolle der Arbeitsschritte in einem KI-Projekt
- Audit Framework für AI/ML-Projekte
- Explainable AI/ML
- Nutzung von externen Begutachtern
- ...

Einfluss von KI im täglichen Leben

mögliche Bereiche:

- persönliche Entscheidungen
- geschäftliche Entscheidungen
- Gerichts-Entscheidungen
- medizinische Probleme
- Arbeitsplatz-Verlust
- Veränderung von Aufgaben
- neue Berufe
- ...

Industrie 4.0

1. indstrielle Revolution wurde durch die Dampfmaschine ausgelöst und veränderte die Arbeits-Welt und damit auch die Arbeit ganz wesentlich

neue Beschäftigungen / Berufe

die nächste Revolution wurde durch die Fließband-Arbeit ausgelöst
verbunden mit einer Abwertung der produktiven Arbeit
aber Schaffung neuer – hochwertiger – Berufe

Die 3. industrielle Revolution ging mit der breiten Einführung der Elektronik einher.
Viele neue Berufe und Ingenieurs-Wissenschaften kamen hinzu.

Die Industrie 4.0 ist gekennzeichnet durch neue Techniken, wie Künstlicher Intelligenz (KI),
Internet of Things (IoT), der Blockchain-Technologie usw. usf.

Regulierung, Ethik und Bildung zur KI

Diskussion ist immer vom aktuellen Stand der Dinge abhängig

- Technik
- Stimmung in der Bevölkerung (z.B. nach Katastrophen / Fehler in Systemen / ...)
- Gesetzes-Lage
- Betroffenheit
- Interesse
- Bildungs-Niveau
- ...

Trolley-Dilemma

klassisches Problem aus dem Bereich des autonomen Fahrens
eine Unfall-Situation ist unvermeidlich, es gibt drei Optionen

- Überfahren einer alten Person (eben mit dem Trolley)
- Überfahren eines Kindes
- Fahren gegen einen Baum und Töten des Fahrer's

dazu kommen Haftungs-Fragen, wer haftet für die Entscheidung, die eine KI vorgenommen hat:

- Versicherung des Fahrer's
- Versicherung des Software-Hersteller's (KI)
- Versicherung der Auto-Produzenten
- der Staat
- ...

Überwachung und Gesichts-Erkennung

Probleme mit permanenter und automatischer Überwachung (Möglichkeit der Erstellung von Bewegungs-Profilen)

Bewertung von Gesetzen oder Betonung der breiten gesellschaftlichen Sicht (FOUCAULT ())
FOUCAULT hat die These aufgestellt, dass schon die Möglichkeit einer Überwachung

Datenschutz gegen / und / mit Privatsphäre (Privacy)

ist die Auswertung von Bewegungs-Profilen von Handy's

Interessen der Gemeinschaft gegen die Privatsphäre

Geheimhaltung gegen Öffentlichkeits-Interesse

oft wird die Diskussion über die Überwachung / den Datenschutz mit einer grundsätzlichen Kritik / Diskussion über Digitalisierung verbunden / gleichgesetzt

Mensch, Moral und Maschine

provokante / aktuelle Diskussions-Probleme:

Darf eine KI das Tempo-Limit überschreiten?

Darf eine KI einen menschlichen Richter ersetzen?

Darf eine KI über die Vergabe eines Kredit's oder eines Job's entscheiden?

Augmented Intelligence gegen Artificial Intelligence (Künstliche Intelligenz)

z.B. durch Vorschlags-Systeme statt Entscheidungs-Systeme

Vertrauen, Konsenz und Akzeptanz schaffen durch:

- breite gesellschaftliche Diskussion
- Bildung zum Thema
- Ethik-Komitees ind Behörden, Unternehmen, Regierungen / Staaten / internationalen Organisationen
- Schaffung passender gesetzlicher Rahmen

Deep Fakes

abgeänderte / manipulierte, realistisch wirkende Medien-Inhalte, bei denen eine Veränderung des ursprünglichen Inhaltes / der ursprünglichen Nachricht bzw. eine gezielte Falsch-Darstellung im Vordergrund steht

bei Deep Fakes geht es besonders um die Einbeziehung von KI in der Verarbeitung der Medien

eigentlich ein altes Thema in der Medien-Welt
neu sind allerdings die Schnelligkeit, mit der solche Inhalte erstellt, bereitgestellt und verbreitet werden können
teilweise funktioniert das automatisch oder stark Software-unterstützt
Erweiterung nun auch auf Video-Systeme, die Einbindung virtueller Personen
praktisch auch Live

Generative Adversarial Network (GAN)

Original		Encoder		latente(s) Struktur / Modell		Decoder		Nachbildung
----------	--	---------	--	------------------------------------	--	---------	--	-------------

Normalerweise soll die Nachbildung dem Original möglichst exakt entsprechen.
Die latente Struktur / das latente Modell ist praktisch nichts anderes als ein mathematisches Modell, das mit heutiger Technik superschnell und –genau berechnet werden kann. Das Modell enthält versteckte Strukturen, (die die KI gelernt hat, und) mit der man nun den Kern der originalen Struktur (Bild, Musik, Video) nutzen / bearbeiten kann.
Passt man nun einzelne Parameter im latenten Modell an, dann erzeugt der Decoder eine entsprechend veränderte Nachbildung. Da nur wenige Parameter geändert wurden, erscheint das Abbild sehr realistisch. Auf die Schnelle ist oft gar keine exakte Unterscheidung zwischen Original und Nachbildung möglich.

Probleme mit und durch Deep Fakes

- Erzeugung von Falsch-Meldungen / -Informationen (nicht nur für Prominente problematisch, auch privat-Personen können leicht diskreditiert werden: z.B. als Dieb oder faul)
- schnelle Verbreitung ist nicht zu stoppen ("Einmal im Netz – immer im Netz!")
- Dementies werden weniger aufgenommen und deutlich schwächer verbreitet, sie haben außerdem allegemein eine geringere Glaubwürdigkeit (sie wirken oft wie Ausreden)
- Deep Fake's – obwohl offensichtlich oder nachgewiesener Weise falsch – werden mehr geglaubt, als Richtigstellungen oder Beweisen
- es kann der Eindruck erzeugt werden, dass die Realität ein Deep Fake ist (Sach-Umkehr)
- Betrug, Identitäts-Diebstahl, Meinungsmache, ...
- ...

"Gegen"-Maßnahmen:

- "Prüfen, prüfen und nochmals Prüfen" (nach LENIN)
- digitale Signaturen
- Überprüfung durch Experten
- gesunder Menschen-Verstand
- vertrauenswürdige Quellen (auch zum Prüfen) nutzen
- gesetzliche Rahmen / Strafen
- ...

Key Takaways

KI und Schule / Bildung

inhaltlich haben wir uns schon ausführlich mit KI beschäftigt

aber es gibt auch eine andere Dimension der KI in der Schule

Beobachtung und Beurteilung von Schüler und Lehrer

aus China gibt es Berichte über die intensive Beobachtung und Bewertung von Schülern von der Beobachtung der Lern-Ergebnisse, aber auch vom eigentlichen Lernen im Unterricht KI beobachtet Schüler und weist auf abgelenkte, verschlafene und unaufmerksame Schüler hin

interessant sind aber sicher die Hilfestellung beim Bewerten von Kontrollen vor allem das Finden von systematischen Fehlern / versteckte Diskalkulie /

für die Schule verantwortungsvoller Einsatz notwendig
klare Regeln
sinnvoller, gezielter Umgang mit der KI

KI bei der Bewerber-Auswahl / KI im Access-Center

<https://web.br.de/interaktiv/ki-bewerbung/>

11. Generative Künstliche Intelligenz – KI produziert

die Jahre 2022 und 2023 werden aus meiner Sicht als Zeitenwende / Revolution in die Menschheits-Geschichte eingehen, da nun die (kreativ / künstlerisch) produzierende KI für die Allgemeinheit zur Verfügung steht und praktisch unbegrenzt genutzt werden kann

als populäre Vorläufer können der google-Übersetzer oder deepl.com gesehen werden. Sie übersetzen Texte in andere Sprachen. Selbst die Erkennung der Sprache des Eingabetextes (ev. auch nur einzelner Worte) erfolgt mittlerweile automatisch.

auch Sprach-Erkennungs-System für Kunden-Service-Systeme oder bei Auskunft funktionieren weitgehend mit KI und sind wichtige Vorläufer und Bestandteile von generativen KI-Systemen für Texte und Sprache

ChatGPT ab 30. November 2022 frei zugänglich

großes Problem für Bildungswesen

Hausaufgaben und Facharbeiten usw. werden praktisch sinnlos

Nachweis, ob Fehler von "schlechter / Leistungs-schwacher" KI stammt oder ob der Schüler den Fehler wirklich selbst gemacht hat, wird schwer zu führen sein

diverse Leistungen

<https://jasper.ai>

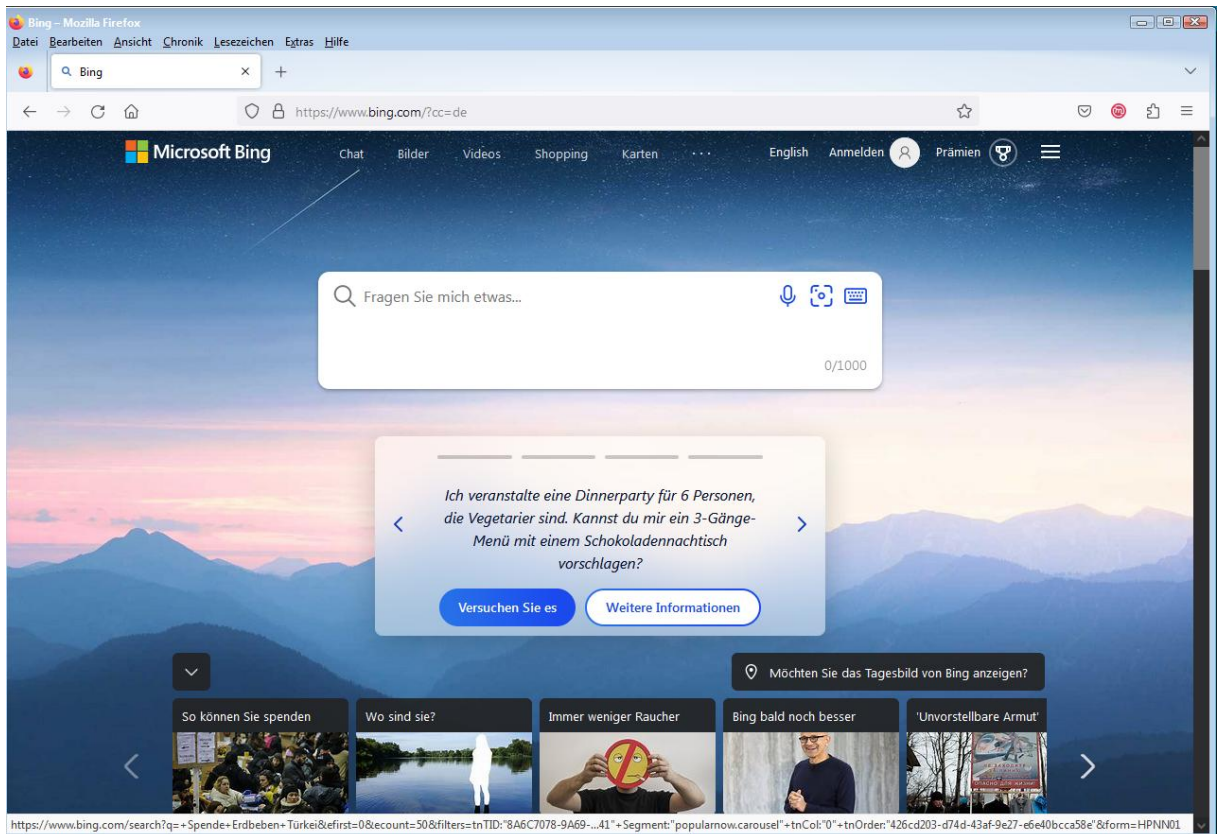
leider schon für das Ausprobieren (5 Tage kostenlos) ist eine Kreditkarte notwendig

<https://openAi.com> (diverse KI's (Chatten, Texten, Sprechen, Malen) zum Ausprobieren)

openAI ist 2015 als Non-profit-Organisation von Elon MUSK gegründet worden

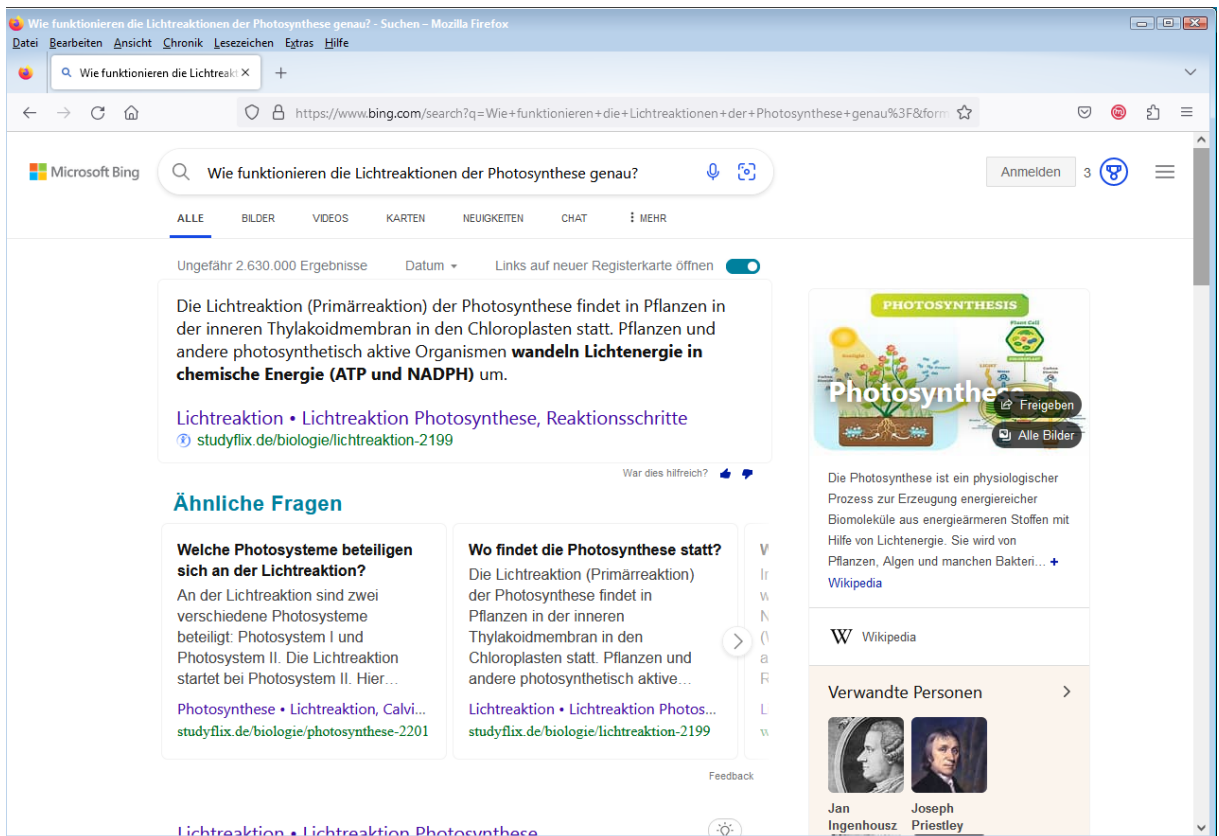
danach diverse Firmen und Geld-Geber eingestiegen, u.a. auch microsoft

seit dem 09. Februar 2023 ist bei der Suchmaschine bing auch eine KI-basierte Suche möglich



Q: www.bing.de (09.02.2023)

auf detaillierte Fragen erhält man immer noch sachlich orientierte Antworten
die rechte Haupt-Spalte ist durch KI-Suche / -Arbeit dominiert



Q: www.bing.de (09.02.2023)

Nachrichten und Informationen aus Literatur, Presse und Internet:

Patzer bei Chatbot von Google

Mountain View. Ein Faktenfehler in der ersten öffentlichen Demonstration des Google-Chatbots Bard hat die Aktie des Mutterkonzerns Alphabet abstürzen lassen. Google hatte am Montag den Textroboter mit künstlicher Intelligenz vorgestellt, um Initiativen des start-ups OpenAI und von Microsoft zu konkurrenzen. Am Mittwoch stellte sich allerdings heraus, dass Bard in seiner allerersten Demo sachlich danebenlag. Bei einer von Google verteilten animierten Grafik (GIF) wurde am Montag die Antwort von Bard auf

die Frage gezeigt "Von welchen neuen Entdeckungen des James-Webb-Weltraumteleskops kann ich meinem Neunjährigen erzählen?". Die Antwort beinhaltete auch die Aussage, das Teleskop habe "die allerersten Bilder eines Planeten außerhalb unseres eigenen Sonnensystems aufgenommen". Astronomen wiesen jedoch darauf hin, dass dies nicht stimmt. Das erste Foto eines Exoplaneten sei bereits im Jahr 2004 aufgenommen worden. Die Aktie des Konzerns gab am Mittwoch kurz nach Börsenschluss um über 7 Prozent nach.

/Q: Ostseezeitung Freitag 10. Februar 2023 S. 10/

Nur wenige Tage nach der Einführung (09. Februar 2023) der neuen Bing-Suche mittels KI musste Microsoft schon auf die ersten Probleme reagieren.

interessante und passende Literatur zum Thema:

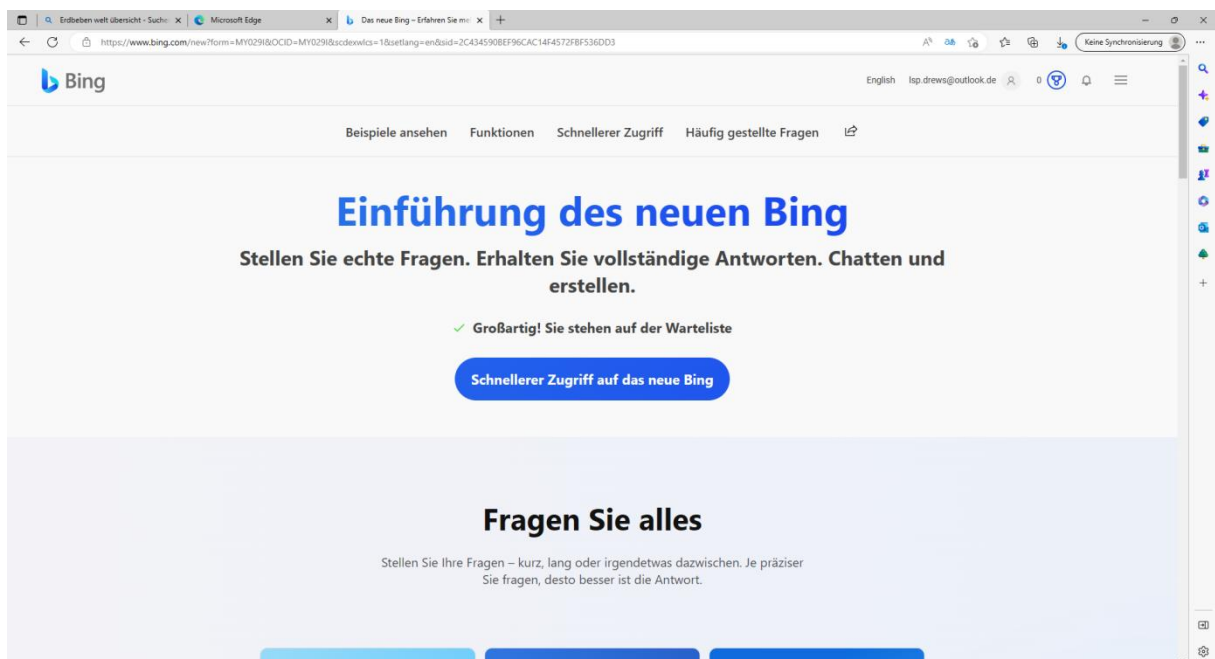
ZWEIG, Katharina: Ein Algorithmus hat kein Taktgefühl – Wo künstliche Intelligenz sich irrt, warum uns das betrifft und was wir dagegen tun können.-München: Heyne Verl., 2019.-5. Aufl., ISBN 978-3-453-20730-1

Nachrichten und Informationen aus Literatur, Presse und Internet:

Microsoft legt Bing an die Leine

Redmond, Microsoft hat die Nutzung seiner Suchmaschine Bing eingeschränkt, die mithilfe von künstlicher Intelligenz auch komplexe Fragen beantworten und ausführliche Konservation führen soll. Der Softwarekonzern reagiert damit auf etliche Vorfälle, in denen der textroboter aus dem Ruder gelaufen ist und Antworten formuliert hat, die als übergriffig und unangemessen empfunden wurden. Das Unternehmen kündigte an, Bing-Chats auf 50 Fragen pro Tag und fünf pro Sitzung zu begrenzen.

/Q: Ostseezeitung Montag 20. Februar 2023 S. 8/



am 21.02.2023 mit Warte-Bildschirm bzw. über "Schneller Zugriff auf das neue Bing" mit einem Tool-Download für "Microsoft-Einstellungen" bzw. einem Download eine App → ich glaube nicht, das Nutzer wirklich über 10 oder mehr Minuten warten wollen???? ohne Anzeige, wie lange es noch dauert oder wieviele in der Warteschlange sind

11.1. Künstliche Intelligenz textet

benutzt werden LLM's (→ [3.x.y. Large-scale Language Models](#))

Chat-Bot's gibt es schon seit einigen Jahren

→ <https://www.cleverbot.com/>

viele können – auch wenn sie nur englisch erscheinen – gut deutsch

derzeit bekanntestes System ist Chat GPT (ChatGPT) von openAI (→ <https://openai.com>)
gehört zum selbst-überwachten maschinellen Lernen (self-supervised Learning, → [2.6. maschinelles Lernen](#))

ist eine Zwischen-Form zwischen (vom Menschen) überwachtem (supervised) und unüberwachtem Lernen (unsupervised Learning)
System erhält / berechnet "Belohnung" für passende Ergebnisse

basiert auf dem Sprach-Modell GPT (Generative Pre-trained Transformer)
speziell für Chat's bzw. Kommunikations-Systeme entwickelt
Ziel ist ein flüssiger Mensch-Maschine-Dialog

großer Fortschritt war durch die Nutzung der Transformer-Architektur (VASWANI et.al. (2017)) möglich

dabei geht darum lange Daten-Folgen (also z.B. Sätze) effizient zu verarbeiten

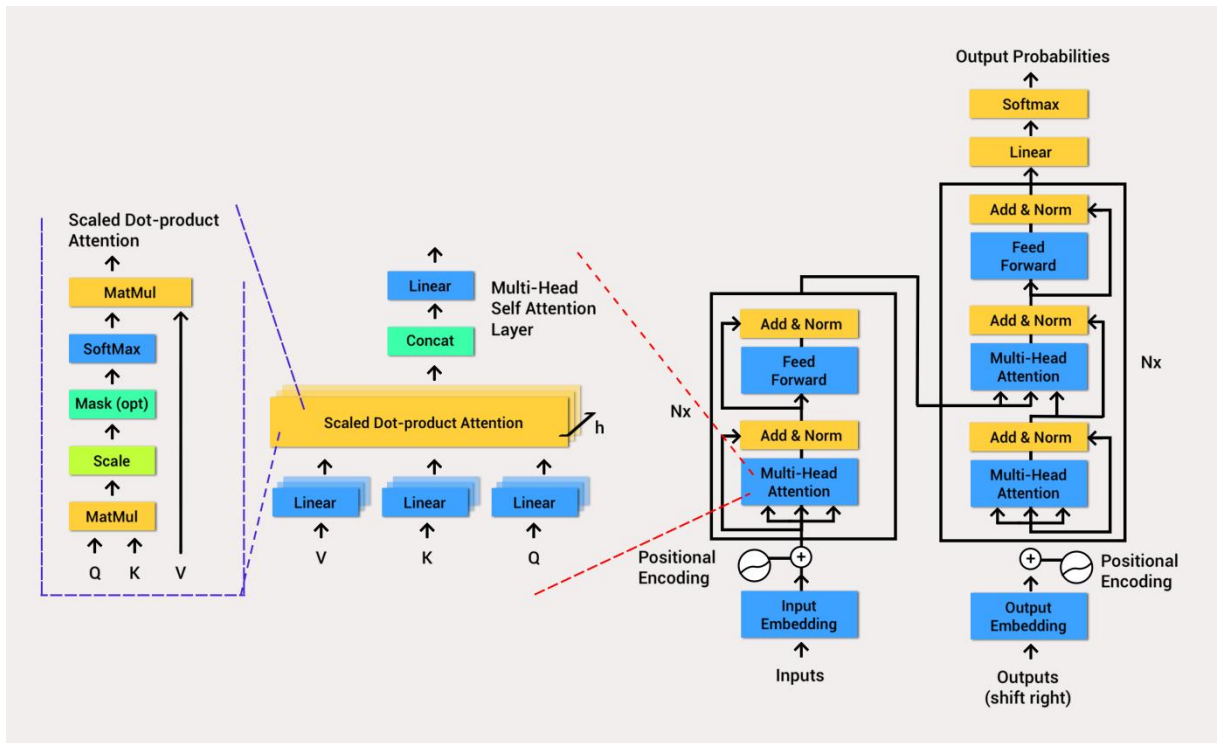
Architektur besteht aus sogenannten Selbst-Aufmerksamkeits-Schichten

diese analysieren die Wichtigkeit von Wörtern und Phrasen

somit wird ein "Verständnis" für die Anfrage / den menschlichen Gesprächs-Part / ... erzeugt
nur mathematisch, syntaktisches Verständnis, aber kein semantisches / inhaltliches

kombiniert wird die Architektur mit Feedforward-Schichten (→ mehrere hintereinander angeordnete Neuronen-schichten, die vorwärts ausgerichtet sind / durchlaufen werden), die Muster-Erkennung ermöglichen

hier werden die Beziehungen zwischen den Wörtern / Phrasen "gelernt" / berechnet



schematisches Modell von ChatGPT

Q: <https://www.pegasusone.com/how-chat-gpt-utilizes-the-advancements-in-artificial-intelligence-to-create-a-revolutionary-language-model/>

Training mit riesigen Daten-Mengen

für ChatGPT 3 sollen die folgenden Quellen benutzt worden sein (nach eigener ChatGPT-Aussage):

- *Bücher: Ich wurde auf einer großen Sammlung von Büchern trainiert, darunter Romane, Fachbücher, Sachbücher und mehr.*
- *Webseiten: Meine Entwickler haben auch eine Vielzahl von Webseiten als Textquellen verwendet, um mein Wissen und meine Sprachverarbeitungsfähigkeiten zu verbessern. Dies schließt Nachrichtenseiten, Blogs, Online-Magazine, Wikipedia und andere ein.*
- *Artikel: Ich habe Zugang zu einer umfangreichen Datenbank mit Artikeln zu verschiedenen Themen wie Wissenschaft, Technologie, Kultur, Geschichte, Kunst und vielem mehr.*
- *Social-Media-Plattformen: Ich wurde auch auf Texten von Social-Media-Plattformen trainiert, einschließlich Twitter, Reddit und anderen Plattformen.*
- *Andere Textquellen: Zu den weiteren Textquellen, auf denen ich trainiert wurde, gehören wissenschaftliche Arbeiten, Chat-Protokolle, E-Mails, öffentlich zugängliche Korpora und mehr.*

Q: <https://chat.openai.com/chat/5f44a9fe-9111-4723-9134-5f59ed36284f> (26.02.2023)

was aber nur einige Quellen sein sollen (???). Insgesamt mehrere Milliarden Wörter.

mit solchen Text-Generatoren lassen sich:

- ChatBot's
- Text-Zusammenfassungen
- Text-Vervollständigungen
- Inhalts-Erstellungen
- ...

erzeugen

durch die Auswahl der Trainings-Texte ergeben sich diverse Einschränkungen:

- Unterrepräsentation von wissenschaftlichen und systematischen Aussagen
- Probleme in der Text-Konstruktion bei sensiblen, weltanschaulichen und kontroversen Themen → Verfall in Stereotypen
- Anfälligkeit für Fehl-Informationen, dogmatischen und propagantistischen Aussagen sowie Falsch-Meldungen
-

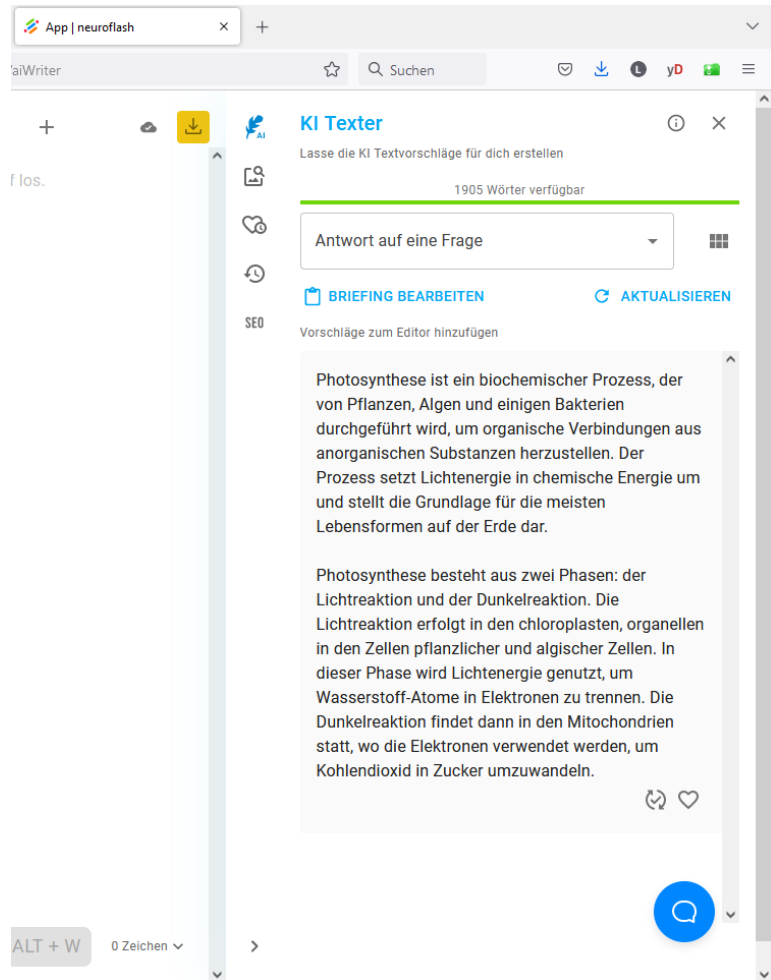
systemische Einschränkungen sind:

- keine oder nur sehr geringer Quellen-Bezug möglich
- sehr großer Rechen-Bedarf
- Modell (Schichten-Strukturen, Kombination mit weiteren Algorithmen) bestimmt die Lern-Leistung (erfolgt heute immer noch stark empirisch / experimentell)
-

auf die Frage, ob ChatGPT den Turing-Test schon bestanden hat, antwortet das System etwas ausweichen, es sei nicht dafür programmiert worden und der Test selbst wäre umstritten und auch in mehreren Versionen verfügbar

auch eine Nachfrage hinsichtlich der Möglichkeit wurde fast identisch und ausweichend beantwortet

deutsche Texte
<https://app.neuro-flash.com>



Antwort auf eine Frage (Schreibtools):
"Wie verläuft die Photosynthese?"
Q: app.neuro-flash.com

ChatGPT zur Anfrage "Schreibe detailliert über die Lichtreaktionen der Photosynthese" (13.01.2023 (Modell: davinci))

Die Photosynthese ist ein Prozess, der durch das Licht der Sonne angeregt wird. Es findet in grünen Pflanzenzellen statt, die Chlorophyll enthalten, ein grünes Pigment, das Licht absorbiert. In der Photosynthese wird Lichtenergie in chemische Energie umgewandelt, die in Form von Zucker gespeichert wird. Es gibt zwei Lichtreaktionen, die in der Photosynthese stattfinden. Die erste ist die Lichtabsorption, die durch das Chlorophyll ausgelöst wird. Chlorophyll absorbiert Lichtenergie in Form von Photonen. Diese Photonen werden dann in elektrische Energie umgewandelt, die in einer chemischen Reaktion verwendet wird. Die zweite Lichtreaktion ist die Lichtverstofflichung. Dieser Prozess verwendet die Photonen, die vom Chlorophyll absorbiert wurden, um Wasserstoff- und Sauerstoffatome zu spalten. Diese Atome werden dann in Form von Kohlenstoffdioxid und Wasser in die Zelle aufgenommen. Die Kohlenstoffdioxidmoleküle werden dann mit den Photonen, die vom Chlorophyll absorbiert wurden, in Zucker umgewandelt. Der Zucker wird als Energiequelle für die Pflanze verwendet. Diese beiden Lichtreaktionen sind für den Photosyntheseprozess unerlässlich. Ohne sie würde die Pflanze nicht in der Lage sein, die Sonnenenergie in chemische Energie zu wandeln, die für ihr Wachstum und Überleben benötigt wird.

Q: openai.com

mit englisch und umfangreicher formulierten Aufträgen sind deutlich bessere Ergebnisse möglich

gute Ergebnisse werden derzeit schon erzielt bei ...

- Umformulieren von Texten (z.B. auf bestimmten Klassen- bzw. Entwicklungs-Niveau)
- Erstellen und Planen eines Rollenspiel's
- Konstruktion von Stunden-Entwürfen in einem vorgegebenen Tabellen-Raster
- Erstellung von Kompetenz-Rastern
- ...

weitere Text-Generatoren (bzw. ähnliches)

TOME (→ <https://beta.tome.app/>) generiert aus einem Text eine fertige Präsentation (→ Storytelling)

DeepL Write (→ <https://www.deepl.com/de/write>) unterstützt beim Schreiben von Texten in verschiedenen Sprachen

Verdummt die KI sich selbst?

Problem: die problematischen Texte gelangen irgendwann wieder in den Pool der analysierten "Original"-Texte für die KI
praktisch ne Selbstbestätigung

ohne Nutzer-Bewertung und mündige Nutzer – die Blödsinn und Phrasen von sinnvollen Lösungen unterscheiden - ist das Schicksal der KI fragwürdig

vielleicht braucht es eine KI, die Produkte von KI-Systemen von Menschen-kreierten Werken unterscheiden kann

dazu bräuchte es aber auch (zum Anlernen) auch eine Kennzeichnungs-Pflicht von KI-Produkten

Versuchs-Wiederholung Photosynthese-Aufgabe
am 22.01.2023 → 2 vorgeschlagene Lösungen

Die Photosynthese ist ein chemischer Prozess, der von Pflanzen und anderen organisch photosynthetischen Systemen durchgeführt wird, um organische Moleküle aus anorganischen Molekülen zu synthetisieren. Der Prozess der Photosynthese ist für das Leben auf der Erde essentiell, da er den Pflanzen ermöglicht, Sonnenenergie in chemische Energie umzuwandeln. Der Prozess der Photosynthese beginnt mit dem Aufsaugen von Licht durch die Blätter einer Pflanze. Die Blätter enthalten Chlorophyll, ein grünes Pigment, das für die Absorption von Licht verantwortlich ist. Wenn Licht auf das Chlorophyll trifft, wird es in elektrische Energie umgewandelt. Diese elektrische Energie wird genutzt, um Wasser und Kohlendioxid in Zucker und Sauerstoff umzuwandeln.

Die Photosynthese ist ein chemischer Prozess, der von Pflanzen und anderen lebenden Organismen angewendet wird und es ermöglicht organische Moleküle aus anorganischen Substanzen zu synthetisieren. Dieser Prozess ist für das Leben notwendig, da er den Pflanzen erlaubt, die Energie der Sonne in chemische Energie zu konvertieren. Der Prozess beginnt damit, dass die Blätter der Pflanzen Licht absorbieren. Diese Blätter sind mit Chlorophyll (einem grünen Pigment) gefärbt, welches für die Aufnahme des Lichts verantwortlich ist. Das Licht wird dann in elektrische Energie umgewandelt, die wiederum dazu verwendet wird, Wasser und Kohlendioxid in Zucker und Sauerstoff umzuwandeln.

Q: <https://app.neuro-flash.com/aiWriter>

Versuchs-Wiederholung Photosynthese-Aufgabe (mit gerade veröffentlichtem ChatGPT4)
am 18.03.2023 → vorgeschlagene Lösung

Die Lichtreaktionen der Photosynthese sind der erste Schritt im Prozess der Photosynthese. Sie finden in den Membranen der Thylakoide statt, die in den Chloroplasten von Pflanzenzellen gefunden werden. Die Lichtreaktionen sind abhängig von Lichtenergie und produzieren energiereiche Verbindungen, die in der zweiten Phase der Photosynthese, der Dunkelreaktion, verwendet werden.

Die Lichtreaktionen beginnen mit der Absorption von Licht durch die Pigmente in den Thylakoidmembranen, insbesondere durch das Chlorophyll a. Wenn Licht absorbiert wird, wird die Energie auf Elektronen übertragen, die sich in den Pigmentmolekülen befinden. Diese Elektronen werden auf ein höheres Energieniveau angehoben und können dann durch eine Kette von Elektronentransportproteinen wandern.

Die Elektronen werden von einem Protein zum nächsten weitergeleitet, wobei jedes Protein eine höhere Elektronenaffinität aufweist als das vorherige. Auf diese Weise wird Energie frei-

gesetzt, die zur Synthese von ATP (Adenosintriphosphat) verwendet wird. ATP ist eine energiereiche Verbindung, die in Zellen als Energiequelle verwendet wird.

Gleichzeitig werden durch die Lichtreaktionen auch Elektronen von Wasser aufgenommen, das in den Thylakoidmembranen gespeichert ist. Dies führt zur Freisetzung von Sauerstoff, der als Abfallprodukt der Photosynthese ausgestoßen wird.

Die Lichtreaktionen der Photosynthese sind ein komplexer Prozess, der von vielen Faktoren beeinflusst wird, wie z.B. der Intensität und Qualität des Lichts, der Temperatur und der Verfügbarkeit von Nährstoffen. Die Effizienz der Lichtreaktionen kann durch die Optimierung dieser Faktoren verbessert werden, was zu einer höheren Photosyntheserate und einer besseren Pflanzenproduktivität führt.

na es geht doch! noch etwas extravagant formuliert (nicht von einem Schüler)

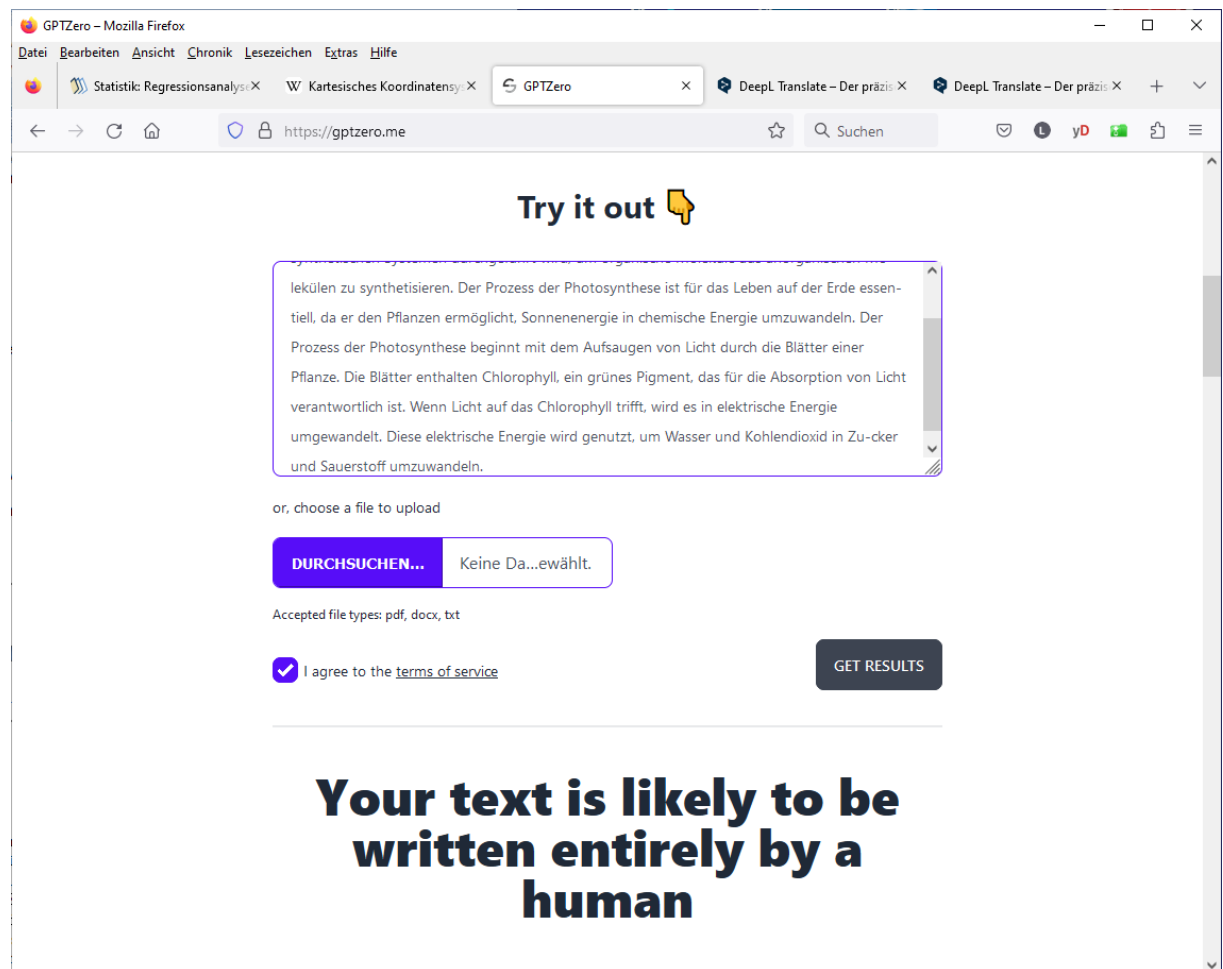
Kontroll-Frage:

Ist der folgende Text von Dir? ... *hier folgte dann obiger Text*

The screenshot shows the ChatFlash AI interface. The browser window title is 'App | neuroflash - Mozilla Firefox'. The address bar contains 'https://app.neuro-flash.com/aiWriter'. The main interface has a document editor on the left with a notification: 'GPT-4 ist hier ✨ Mit deiner magischen Feder profitierst Du ab sofort als Erster in Europa! Wir integrieren GPT-4 in allen Funktionen, wo Du profitierst! → längere Inhalte → bessere Qualität in allen Sprachen → höhere Präzision im Ausdruck → verbesserte Orientierung an Fakten ... und das ohne zusätzliche Kosten'. The central chat area shows a text input field with the following text: 'Nutze die Textvorschläge oder schreib einfach drauf los.' Below this is a large text block containing the photosynthesis text from the document. The right sidebar shows a 'ChatFlash' chat window with the message: 'Was immer du brauchst, sag es mir. 5155 Wörter verfügbar'. Below this is a message from the AI: 'Nein, der Text wurde nicht von mir geschrieben. Als AI-Sprachmodell kann ich jedoch bestätigen, dass der Text eine korrekte Beschreibung der Lichtreaktionen der Photosynthese darstellt.' At the bottom of the chat window are buttons for 'Vorlagen', 'Persönlichkeiten Standard', and 'ZURÜCKSETZEN'. Below the chat window is a text input field with the prompt 'Gib mir 4 Gründe, warum Content Marketing wichtig ist' and a 'ZURÜCKSETZEN' button.

auch crossplag.com behauptet, dass der Text nur zu 1 % von einer KI stammt

Prüfen von Texten auf KI-Urheberschaft?!



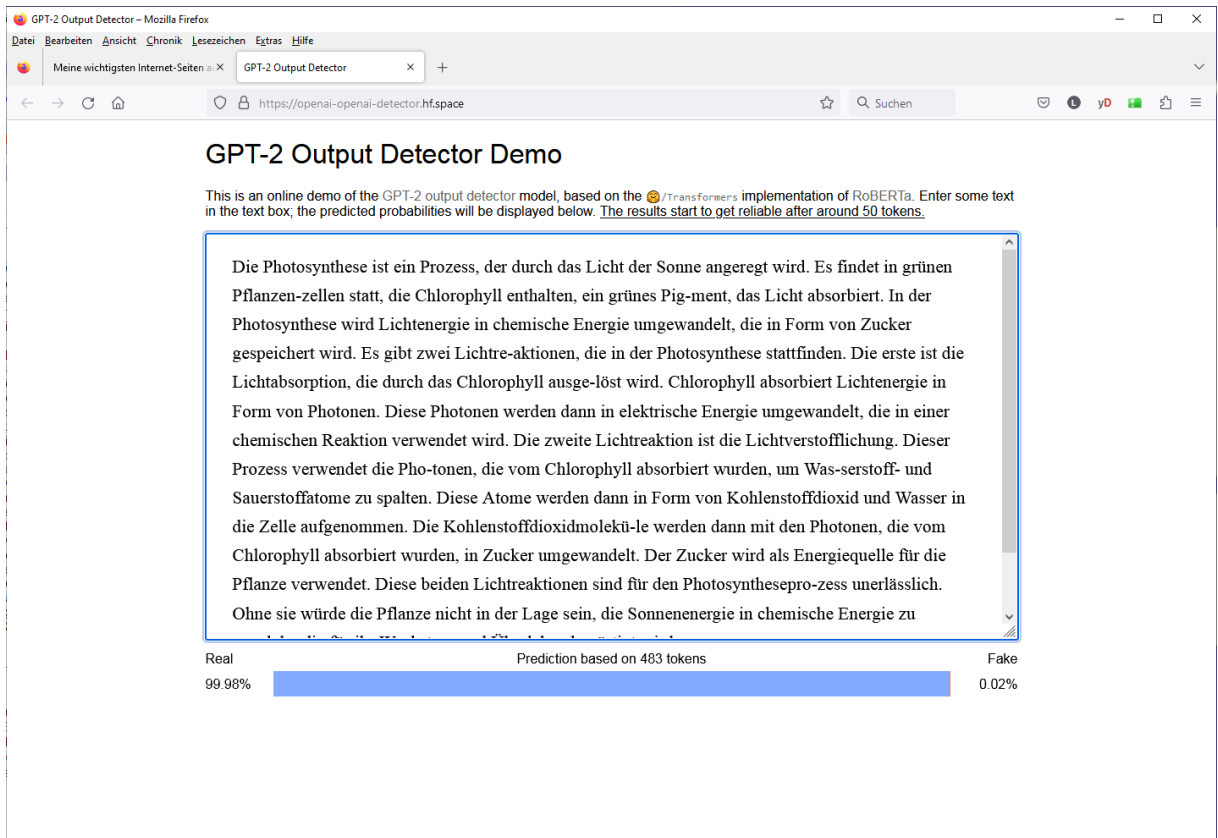
Q: <https://gptzero.me/> (30.01.2023)

1. Text mit GPTzero geprüft (→ <https://platform.openai.com/ai-text-classifier>) → Na dann mal auf zu neuen Horizonten!!!
empfohlen werden Text-Proben mit mindestens 200 Wörtern

direkter Test mit ChatGPT selbst:

Anfrage: Ist dieser Text von dir? *hier dann Text reinkopieren*

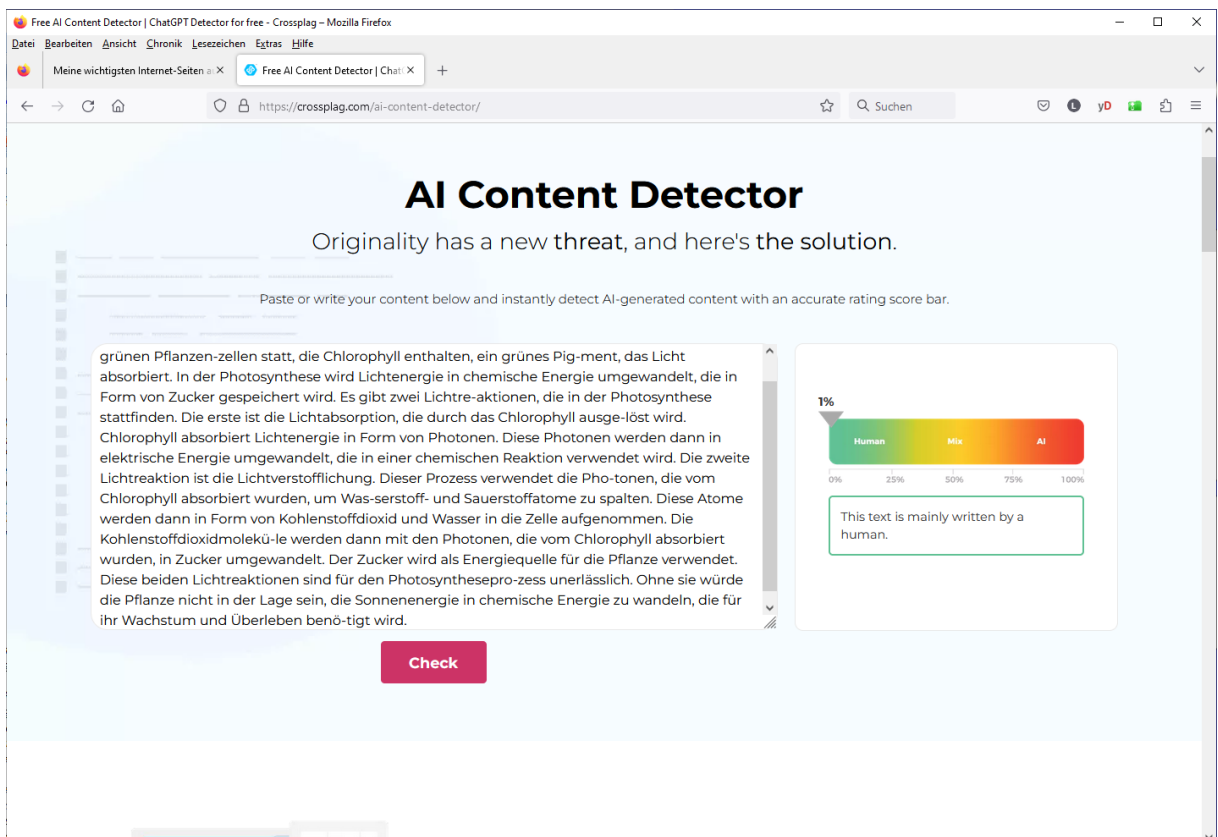
detaillierter Licht-Reaktionen-Text mit → openai-openai-detector.hf.space (praktisch selbst-)geprüft



Q: <https://openai-openai-detector.hf.space/> (26.02.2023)

ob mir das wirklich hilft

detaillierter Licht-Reaktionen-Text mit crossplag geprüft



Q: <https://crossplag.com/ai-content-detector/> (26.02.2023)

weitere Test-Möglichkeiten für KI-Texte:

- <https://writer.com/ai-content-detector/> (für Text-Proben mit maximal 1500 Zeichen)
-

Einführung Dreiecke

Ein Dreieck ist eine geometrische Figur, die aus drei Seiten und drei Ecken besteht.

Wie werden die Seiten eines Dreiecks bezeichnet?

- Kanten
- Ecken
- Winkel
- Punkte

Was ist ein Dreieck?

- Eine geometrische Figur mit drei Seiten und drei Winkeln
- Eine geometrische Figur mit vier Seiten und vier Winkeln
- Eine geometrische Figur mit vier Seiten und drei Winkeln
- Eine geometrische Figur mit drei Seiten und vier Winkeln

Fülle die Lücken mit den passenden Begriffen.

Ein Dreieck ist eine geometrische Form, die aus drei _____ besteht, die aneinander angrenzen. Dreiecke können in unterschiedliche Kategorien eingeteilt werden, je nachdem, wie viele _____ sie haben. Wenn ein Dreieck drei gleiche Seiten hat, wird es als gleichseitiges Dreieck bezeichnet. Wenn zwei Seiten gleich sind, wird es als _____ bezeichnet. Wenn alle Seiten unterschiedlich sind, wird es als ungleichseitiges Dreieck bezeichnet.

1. Aufgabe 2. Aufgabe 3. Aufgabe

0/3 ✓

Abschließen

Vorbereitung einer Unterrichtsstunde mit und über ChatGPT

Anfrage

Erstelle mir eine 90 Minuten Unterrichtsstunde zu ChatGPT für die Abiturstufe!

Antwort

Gerne erstelle ich eine Unterrichtsstunde zu ChatGPT für die Abiturstufe. Die Stunde ist auf 90 Minuten ausgelegt und kann je nach Bedarf angepasst werden. Die Unterrichtsstunde kann wie folgt strukturiert sein:

1. *Einführung (10 Minuten)*
 - *Vorstellung von ChatGPT*
 - *Erklärung, was ein Chatbot ist und wie Chatbots genutzt werden*
 - *Erklärung, wie ChatGPT funktioniert*
2. *Funktionsweise von ChatGPT (20 Minuten)*
 - *Erklärung des Machine Learnings und wie es ChatGPT ermöglicht, auf Fragen zu antworten*
 - *Erklärung des Trainingsprozesses von ChatGPT*
 - *Diskussion der verschiedenen Arten von Fragen, auf die ChatGPT antworten kann*
3. *Anwendungsbeispiele von ChatGPT (30 Minuten)*
 - *Erklärung, wie ChatGPT in verschiedenen Bereichen eingesetzt wird, z. B. im Kundenservice, in der medizinischen Diagnostik und im Bildungsbereich*
 - *Vorstellung von ChatGPT-Tools und -Anwendungen, z. B. Chatbots, Sprachassistenten und Textgeneratoren*
 - *Diskussion der Vor- und Nachteile von Chatbots und wie sie unsere Welt verändern*
4. *Ethik und Datenschutz (20 Minuten)*
 - *Diskussion der Ethik von Chatbots und der Verantwortung der Entwickler, sicherzustellen, dass sie ethisch korrekt und nicht diskriminierend eingesetzt werden*
 - *Erklärung des Datenschutzes und wie ChatGPT Daten sammelt und verwendet*
 - *Diskussion der möglichen Risiken von Chatbots und wie sie minimiert werden können*
5. *Ausblick (10 Minuten)*
 - *Diskussion über die Zukunft von Chatbots und wie sie sich in den nächsten Jahren weiterentwickeln könnten*
 - *Erklärung, wie ChatGPT und ähnliche Technologien dazu beitragen können, menschliche Arbeit zu ergänzen und zu verbessern*
6. *Zusammenfassung (10 Minuten)*
 - *Zusammenfassung der wichtigsten Punkte*
 - *Beantwortung von Fragen*

Die Unterrichtsstunde kann durch praktische Übungen oder Diskussionen ergänzt werden, um das Verständnis der Schülerinnen und Schüler zu vertiefen. Zum Beispiel könnte man eine Gruppendiskussion zu den Vor- und Nachteilen von Chatbots und ihrer Anwendung in verschiedenen Branchen durchführen. Oder man könnte die Schülerinnen und Schüler bitten, Fragen an ChatGPT zu stellen und die Antworten zu diskutieren.

Erstellung eines Vortrag's über ChatGPT

Anfrage

Erstelle mir einen 20 minütigen Vortrag zu ChatGPT!

Antwort

Gerne erstelle ich einen 20-minütigen Vortrag zu ChatGPT. Der Vortrag könnte wie folgt strukturiert sein:

1. *Einführung (2 Minuten)*
 - *Vorstellung von ChatGPT*
 - *Erklärung, was ein Chatbot ist und wie Chatbots genutzt werden*
 - *Erklärung, wie ChatGPT funktioniert*
2. *Funktionsweise von ChatGPT (6 Minuten)*
 - *Erklärung des Machine Learnings und wie es ChatGPT ermöglicht, auf Fragen zu antworten*
 - *Erklärung des Trainingsprozesses von ChatGPT*
 - *Diskussion der verschiedenen Arten von Fragen, auf die ChatGPT antworten kann*
3. *Anwendungsbeispiele von ChatGPT (6 Minuten)*
 - *Erklärung, wie ChatGPT in verschiedenen Bereichen eingesetzt wird, z. B. im Kundenservice, in der medizinischen Diagnostik und im Bildungsbereich*
 - *Vorstellung von ChatGPT-Tools und -Anwendungen, z. B. Chatbots, Sprachassistenten und Textgeneratoren*
 - *Diskussion der Vor- und Nachteile von Chatbots und wie sie unsere Welt verändern*
4. *Ethik und Datenschutz (4 Minuten)*
 - *Diskussion der Ethik von Chatbots und der Verantwortung der Entwickler, sicherzustellen, dass sie ethisch korrekt und nicht diskriminierend eingesetzt werden*
 - *Erklärung des Datenschutzes und wie ChatGPT Daten sammelt und verwendet*
 - *Diskussion der möglichen Risiken von Chatbots und wie sie minimiert werden können*
5. *Ausblick (2 Minuten)*
 - *Diskussion über die Zukunft von Chatbots und wie sie sich in den nächsten Jahren weiterentwickeln könnten*
 - *Erklärung, wie ChatGPT und ähnliche Technologien dazu beitragen können, menschliche Arbeit zu ergänzen und zu verbessern*
6. *Zusammenfassung (2 Minuten)*
 - *Zusammenfassung der wichtigsten Punkte*

Im Anschluss könnte man eine kurze Fragerunde oder Diskussion mit dem Publikum durchführen, um auf spezifische Fragen oder Bedenken eingehen zu können.

Anfrage

Formuliere den Vortrag aus!

Antwort

Liebe Zuhörerinnen und Zuhörer, heute möchte ich Ihnen gerne einen Einblick in die Welt von ChatGPT geben. ChatGPT ist ein Chatbot, der auf künstlicher Intelligenz basiert und in der Lage ist, Fragen zu beantworten und Gespräche zu führen.

Bevor wir uns jedoch näher mit ChatGPT beschäftigen, möchte ich Ihnen zunächst erklären, was ein Chatbot ist und wie Chatbots genutzt werden. Chatbots sind Computerprogramme, die entwickelt wurden, um mit Menschen zu interagieren und ihnen bei bestimmten Aufgaben oder Fragen zu helfen. Sie können über Text, Sprache oder eine Kombination aus beidem kommunizieren und werden in verschiedenen Bereichen wie Kundenservice, Bildung und Medizin eingesetzt.

Nun kommen wir zum eigentlichen Thema unseres Vortrags: ChatGPT. Wie funktioniert ChatGPT und wie ist es ihm möglich, auf Fragen zu antworten? ChatGPT basiert auf einem Machine-Learning-Algorithmus, der es ihm ermöglicht, aus einer großen Datenmenge zu lernen und Muster zu erkennen. Durch den Trainingsprozess ist ChatGPT in der Lage, auf eine Vielzahl von Fragen und Themen zu antworten.

In welchen Bereichen wird ChatGPT eingesetzt und welche Anwendungsbeispiele gibt es? ChatGPT wird in vielen verschiedenen Bereichen eingesetzt, zum Beispiel im Kundenservice, wo es Kundenanfragen schnell und effizient beantworten kann. In der medizinischen Diagnostik kann ChatGPT Ärzten bei der Diagnose und Behandlung von Patienten helfen. Auch im Bildungsbereich kann ChatGPT als Lernassistent eingesetzt werden.

Es gibt jedoch auch einige ethische und datenschutzrechtliche Fragen, die bei der Nutzung von Chatbots und ähnlichen Technologien berücksichtigt werden müssen. Es ist wichtig, sicherzustellen, dass Chatbots ethisch korrekt und nicht diskriminierend eingesetzt werden. Auch der Datenschutz ist ein wichtiger Faktor, da Chatbots Daten sammeln und verwenden. Wie könnte die Zukunft von Chatbots aussehen und welche Auswirkungen haben sie auf unsere Welt? Chatbots könnten dazu beitragen, menschliche Arbeit zu ergänzen und zu verbessern. Es ist jedoch auch wichtig, sich über mögliche Risiken und Auswirkungen auf Arbeitsplätze und Gesellschaft im Allgemeinen Gedanken zu machen.

Zusammenfassend lässt sich sagen, dass ChatGPT ein spannendes Beispiel für die Möglichkeiten von künstlicher Intelligenz und Machine Learning ist. Wir sollten jedoch auch die ethischen und datenschutzrechtlichen Fragen, die damit verbunden sind, im Blick behalten. Vielen Dank für Ihre Aufmerksamkeit.

Anfrage

Das werden aber keine 20 min. Kannst Du den Vortrag verlängern?

Antwort

Natürlich, gerne! Hier ist eine erweiterte Version meines Vortrags über ChatGPT:

Liebe Zuhörerinnen und Zuhörer,

heute möchte ich Ihnen gerne einen Einblick in die Welt von ChatGPT geben. ChatGPT ist ein Chatbot, der auf künstlicher Intelligenz basiert und in der Lage ist, Fragen zu beantworten und Gespräche zu führen.

Bevor wir uns jedoch näher mit ChatGPT beschäftigen, möchte ich Ihnen zunächst erklären, was ein Chatbot ist und wie Chatbots genutzt werden. Chatbots sind Computerprogramme, die entwickelt wurden, um mit Menschen zu interagieren und ihnen bei bestimmten Aufgaben oder Fragen zu helfen. Sie können über Text, Sprache oder eine Kombination aus beidem kommunizieren und werden in verschiedenen Bereichen wie Kundenservice, Bildung und Medizin eingesetzt.

Nun kommen wir zum eigentlichen Thema unseres Vortrags: ChatGPT. Wie funktioniert ChatGPT und wie ist es ihm möglich, auf Fragen zu antworten? ChatGPT basiert auf einem Machine-Learning-Algorithmus, der es ihm ermöglicht, aus einer großen Datenmenge zu lernen und Muster zu erkennen. Durch den Trainingsprozess ist ChatGPT in der Lage, auf eine Vielzahl von Fragen und Themen zu antworten.

ChatGPT hat seinen Ursprung in der Forschung und Entwicklung von künstlicher Intelligenz bei OpenAI, einem Forschungsunternehmen, das sich der Entwicklung sicherer künstlicher Intelligenz verschrieben hat. ChatGPT wurde 2019 veröffentlicht und hat seitdem eine große

Aufmerksamkeit erregt, da es in der Lage ist, menschenähnliche Antworten auf eine Vielzahl von Fragen und Themen zu geben.

In welchen Bereichen wird ChatGPT eingesetzt und welche Anwendungsbeispiele gibt es? ChatGPT wird in vielen verschiedenen Bereichen eingesetzt, zum Beispiel im Kundenservice, wo es Kundenanfragen schnell und effizient beantworten kann. In der medizinischen Diagnostik kann ChatGPT Ärzten bei der Diagnose und Behandlung von Patienten helfen. Auch im Bildungsbereich kann ChatGPT als Lernassistent eingesetzt werden.

Es gibt jedoch auch einige ethische und datenschutzrechtliche Fragen, die bei der Nutzung von Chatbots und ähnlichen Technologien berücksichtigt werden müssen. Es ist wichtig, sicherzustellen, dass Chatbots ethisch korrekt und nicht diskriminierend eingesetzt werden.

Auch der Datenschutz ist ein wichtiger Faktor, da Chatbots Daten sammeln und verwenden. Wie könnte die Zukunft von Chatbots aussehen und welche Auswirkungen haben sie auf unsere Welt? Chatbots könnten dazu beitragen, menschliche Arbeit zu ergänzen und zu verbessern. Es ist jedoch auch wichtig, sich über mögliche Risiken und Auswirkungen auf Arbeitsplätze und Gesellschaft im Allgemeinen Gedanken zu machen.

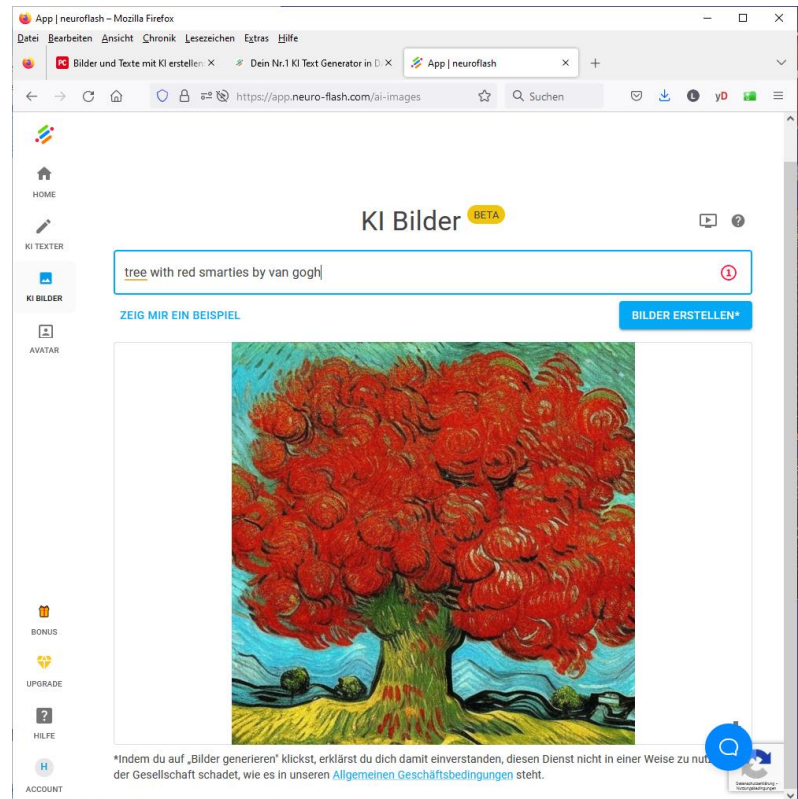
Zusammenfassend lässt sich sagen, dass ChatGPT ein spannendes Beispiel für die Möglichkeiten von künstlicher Intelligenz und Machine Learning ist. Wir sollten jedoch auch die ethischen und datenschutzrechtlichen Fragen, die damit verbunden sind, im Blick behalten.

Vielen Dank für Ihre Aufmerksamkeit.

Q: <https://chat.openai.com/chat/5f44a9fe-9111-4723-9134-5f59ed36284f> (26.02.2023)

11.2. Künstliche Intelligenz malt

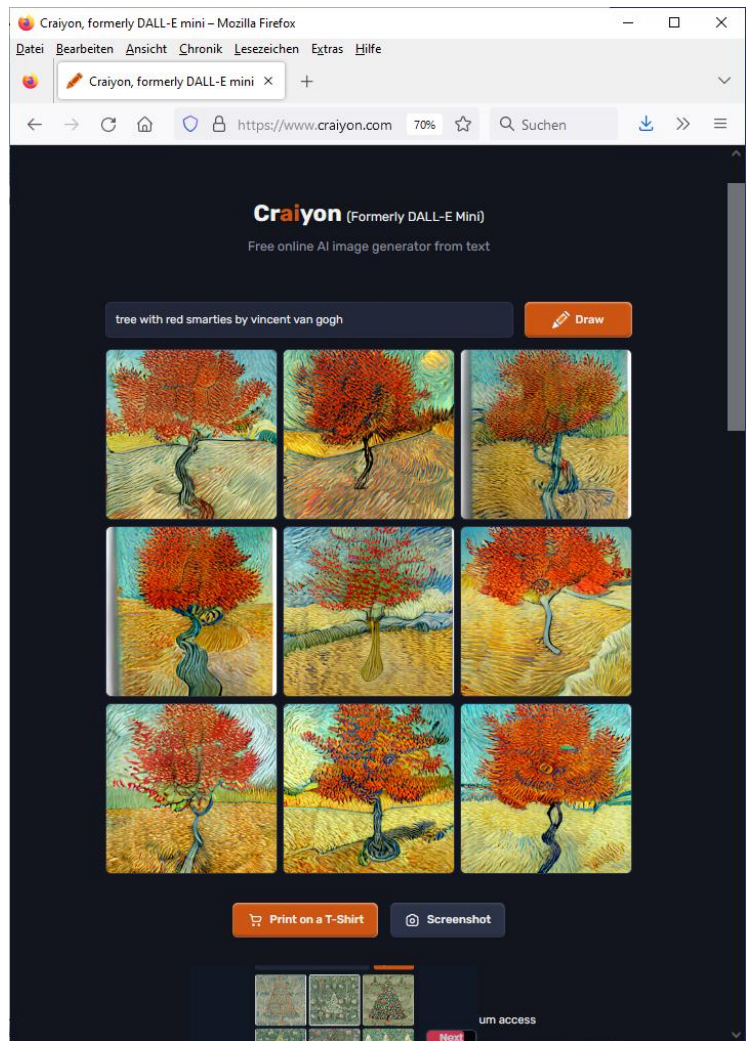
<https://app.neuro-flash.com>



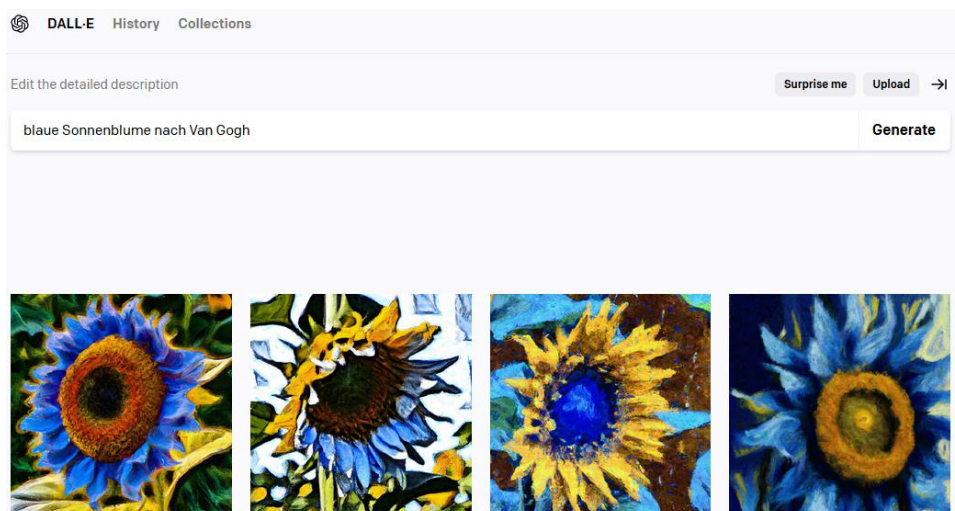
Q: <https://app.neuro-flash.com>

Problem: Wer hat hier die Urheberrechte?

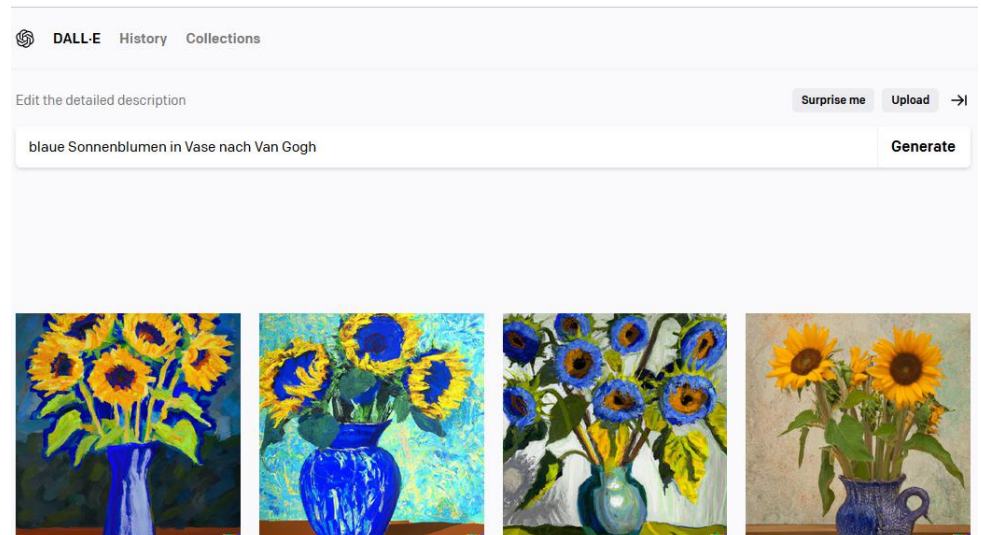
<https://www.crayon.com/>



Galerie künstlich erzeugter "Gemälde" im Stil vom VAN GOGH
Q: <https://www.crayon.com/>



Q: [openAI.com](https://openai.com) (DALL-E)

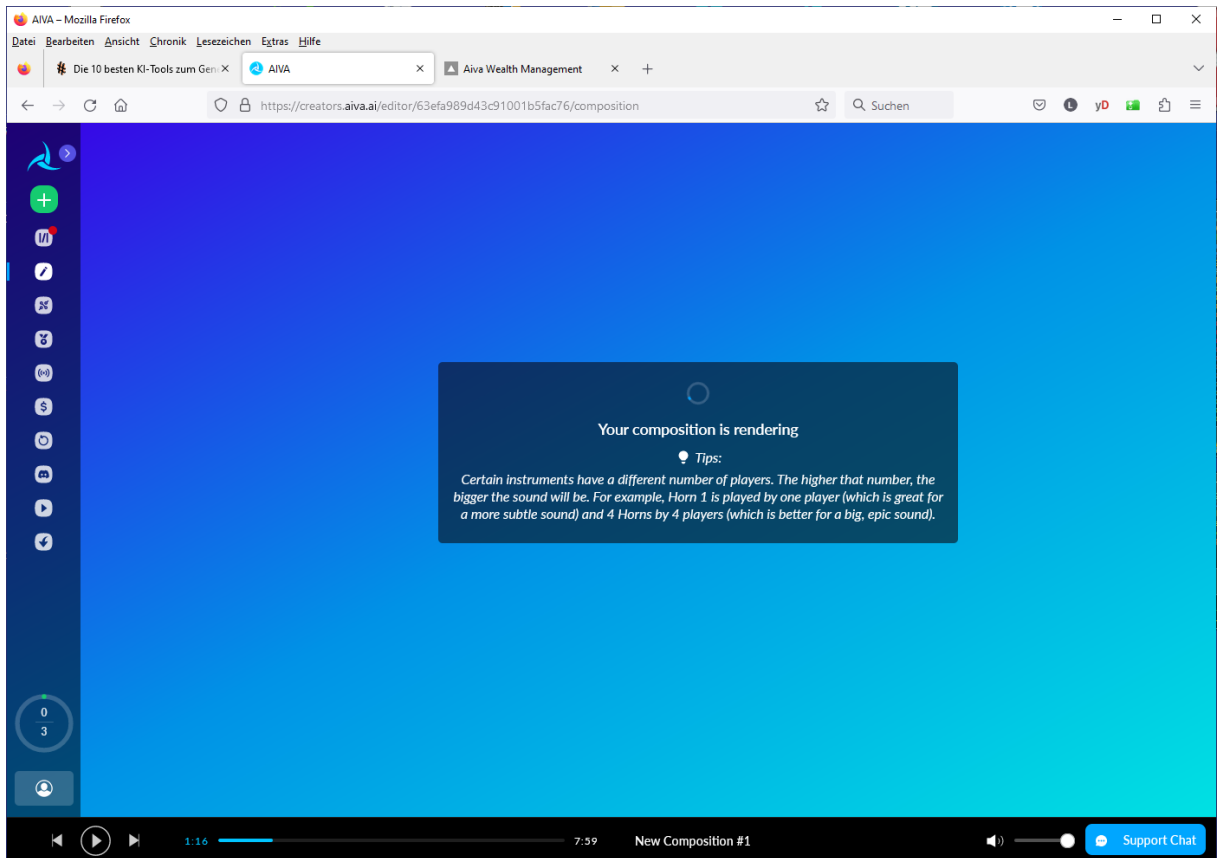


Q: openAI.com (DALL-E)

weitere Mal- und Zeichen-KI (oder ähnliches)

Midjourney (→ <https://www.midjourney.com/home/>) generiert aus einem Text Bilder

11.3. KI arrangiert und komponiert



KI gibt Tips zum Verbessern des Sounds
Q: aiva.ai

Easy way to create royalty free music - ecret music - Mozilla Firefox

Die 10 besten KI-Tools zum Gen... Easy way to create royalty free...

https://ecrettmusic.com/play

Suchen

Like Ecret? Try its big brother SOUNDRAW! Go to SOUNDRAW

Sign in SUBSCRIBE

SCENE

- Adventure
- ✓ Simulation
- Puzzle
- Village
- Travel
- Fitness
- Lifestyle
- Niaht

MOOD

- Chill
- Dark
- Uplifting
- Happy
- Serious
- ✓ Hopeful
- Sad
- Peaceful

GENRE

- 8-bit
- EDM
- Tropical House
- Techno
- ✓ Hip Hop
- Lofi Hip Hop
- Acoustic
- Cinematic

clear all CREATE MUSIC

Favorites

Please sign in to use this feature

00:47 / 00:47 Tempo & Volume 0:30-1:00

	00:00	00:07	00:15	00:23	00:31	00:38	00:47
Melody 3	■	■	■	■	■	■	■
Backing 1	■	■	■	■	■	■	■
Bass 1	■	■	■	■	■	■	■
Drum 2	■	■	■	■	■	■	■
Fill		1	2			2	

Simulation_Hopeful_HipHop

Download

Favorite

Q: <https://ecrettmusic.com>

11.4. Suche mit KI

you.com (YouChat)

11.x. weitere kreative Ansätze / Anwendungen

11.x.y. Storytelling und Präsentation

beta.tome.app

lumen5.com

11.x. gesellschaftliche Aspekte zu kreativer KI

mögliche Reaktionen / Konsequenzen für die Schule / Leistungs-Ermittlung in der Schule

- Orientierung auf Leistungen, die unter Aufsicht und ohne Internet-Verbindung erstellt wurden
 - Orientierung auf Anwendungs- oder Kombination-Aufgaben, die wenige Schlüsselbegriffe des Themas enthalten
 - Auseinandersetzung mit KI-Produkten als Schüler-Leistung
 - Facharbeiten, Hausaufgaben usw. nur mit Liste überprüfbarer Links (Links, die zum Zeitpunkt der Bewertung nicht mehr aktiv sind, werden nicht gezählt!)
 - Hinwendung zu echten Literatur-Quellen (Internet-Quellen werden nur zu einem bestimmten Anteil anerkannt)
 - geänderte Eigenständigkeits-Erklärungen (Unterschied zwischen einem KI-erzeugten Produkt und einem – von einer KI – überarbeiteten Produkt (Nachweisbarkeit des unbearbeiteten Produkts))
 - noch klarere und konsequentere Ahndung von Betrugs-Versuchen
 - klare Zuordnung von Quellen zu Zitaten und abgeleiteten Aussagen
 - stärkere Beachtung inhaltlicher Fehler (sind eben schlecht recherchiert oder von einer überforderten KI produziert)
 - Kontrolle der Kompetenz und des Verständnisses durch die Autoren über eine umfangreiche Disputation
 - Check der Schüler-Leistungen mit speziellen KI-Programmen
 - ...
-
- Arbeitgeber (Ministerien) werden KI's od.ä. Werkzeuge zur Verfügung stellen müssen, die Schüler-Produkte auf Eigenständigkeit und Nicht-KI-produziert testen können
 - intensives und regelmäßiges Training und Weiterbildung der Lehrer an und über KI-Systeme(n)
 - ...

Aspekte bei der (Be)Wertung des KI-Einsatzes z.B. bei Facharbeiten / Hausaufgaben / ...

- Wurde die KI als (alleinige) Quelle oder als Werkzeug (zum Umschreiben) benutzt?
- Welchen Anteil haben die "künstlichen" / nicht eindeutig einem Schüler zuordbaren Texte?
- Sind Fachteile oder z.B. (nur) die Einleitung betroffen?
- Sind die Fach- bzw. inhaltlichen Fehler vom Schüler erkennbar?
- Fehlen Quellen-Angaben und –Bezüge?

Literatur und Quellen

/1/ :

ISBN

/

/10/ BUNGARTZ, Hans-Joachim:
Neuronale Netze: Modellierung von Lernen und Klassifizieren

/11/ KAHN, Ken; LU, Yu; ZHANG, Jingjing; WINTERS, Niall; GAO, Ming:
Mit SNAP! Word Embeddings programmieren.-übersetzt von: LINDNER, Annabel.-
Berlin: LOG IN Verl.
IN: LOG IN Heft Nr. 193/194 (2020), S. 37 ff

/12/ :

ISBN

/13/ :

ISBN

/14/ ZWEIG, Katharina:
Ein Algorithmus hat kein Taktgefühl – Wo künstliche Intelligenz sich irrt, warum uns
das betrifft und was wir dagegen tun können.-München: Wilhelm HEYNE Verl.-5. Aufl.
ISBN 978-3-453-20730-1

/15/ SCHMID, Ute; WEITZ, Katharina; SIEBERS, Michael:
Künstliche Intelligenz selber programmieren für DUMMIES Junior.-Weinheim: WILEY-
VCH Verl., 2019
ISBN 978-3-527-71573-2

/16/ KAFFKA, Thomas:
Neuronale Netze – Grundlagen – Mit Beispielprogrammen in Java.-Frechen: mitp
Verl.-2017.-1. Aufl.
ISBN 978-3-95845-607-5

/17/ :

ISBN

-
- /A/ Wikipedia
<http://de.wikipedia.org>
- /B/ OpenHPI → open.hpi.de
/B1/ Kurs: NAUMANN: Data Engineering und Data Science – Klarheit in den Schlagwort-Dschungel (??..20??)
/B2/ Kurs: DAHM, LANGE, PATZLAFF, PETER, SCHENKEL: Praktische Einführung in Deep Learning für Computer Vision (??..20??)
- /C/ https://www.youtube.com/watch?v=ya_6I9IVMzY (Doktor Watson erklärt Maschine Learning)
- /D/ <https://www.youtube.com/watch?v=XwbDMCkRg84> (Phil's Physics erklärt Neuronale Netze und Bilderkennung))
- /E/ <https://www.coursera.org/learn/neural-networks-deep-learning> (Onlinekurs zu Neuronalen Netzwerken und Deep Learning (z.T. englisch; auch zum kostenlosen Testen wird eine Kreditkarten-Nummer od.ä. benötigt)
- /F/ <https://teachablemachine.withgoogle.com/> (selbst zu trainierende Erkennung von Musik, Sprache, Gesten, ...)

Die originalen sowie detailliertere bibliographische Angaben zu den meisten Literaturquellen sind im Internet unter <http://dnb.ddb.de> zu finden.

Anhänge

	Wer / Was ist ...?	Woran erkennt man das?	Ist es wirklich so? (Diskussion)
intelligent			
nicht intelligent			

A4-Seite

diverse weitere Quell-Code's

MCCULLOCH-PITTS-Zelle mit Python (Minimal-Version)

```
1 print("Simulation einer McCulloch-Pitts-Zelle")
2 print("")
3 schwellenwert=2.0
4 print("Schwellenwert: ",schwellenwert)
5 xWerte = [1, 0]
6 print("x-Eingangswerte: ",xWerte,"      Anzahl: ",len(xWerte))
7 yWerte = []
8 print("y-Eingangswerte: ",yWerte,"      Anzahl: ",len(yWerte))
9 print("")
10 # Berechnung der Werte für die Schwellenwert-Funktion
11 summeErregung = 0
12 for x in xWerte:
13     summeErregung += x
14 summeHemmung = 0
15 for y in yWerte:
16     summeHemmung += y
17 print("gesamt Erregung: ",summeErregung,"      Hemmung: ",summeHemmung)
18 # Berechnung der Schwellenwert-Funktion
19 if (summeHemmung >= 1) or (summeErregung<schwellenwert):
20     ausgangZelle = 0
21 else:
22     ausgangZelle = 1
23 # Ausgabe der Zelle; Ausgang der Zelle
24 print("")
25 print("Zelle gibt eine ",ausgangZelle," aus")
26 input() # nur damit Konsolenfenster bis ENTER sichtbar bleibt
27 print("Programm-Ende")
28
```

Tic-tac-toe mit Minimax in PROLOG

minimax.pl

```
1 %% Code from http://www.emse.fr/~picard/cours/ai/minimax/#sec-3
2
3 % minimax(Pos, BestNextPos, Val)
4 % Pos is a position, Val is its minimax value.
5 % Best move from Pos leads to position BestNextPos.
6 minimax(Pos, BestNextPos, Val) :-                               % Pos has successors
7     bagof(NextPos, move(Pos, NextPos), NextPosList),
8     best(NextPosList, BestNextPos, Val), !.
9
10 minimax(Pos, _, Val) :-                                       % Pos has no successors
11     utility(Pos, Val).
12
13 best([Pos], Pos, Val) :-
14     minimax(Pos, _, Val), !.
15
16 best([Pos1 | PosList], BestPos, BestVal) :-
17     minimax(Pos1, _, Val1),
18     best(PosList, Pos2, Val2),
19     betterOf(Pos1, Val1, Pos2, Val2, BestPos, BestVal).
20
21 betterOf(Pos0, Val0, _, Val1, Pos0, Val0) :-                 % Pos0 better than Pos1
22     min_to_move(Pos0),                                       % MIN to move in Pos0
23     Val0 > Val1, !                                           % MAX prefers the greater
24     value
25     ;
26     max_to_move(Pos0),                                       % MAX to move in Pos0
27     Val0 < Val1, !                                           % MIN prefers the lesser va-
28     lue
29     Pos0
30     Pos1
```

Q: <https://gist.github.com/AndreFCruz/13df05bcc3f00902992493a44834748f>

tictactoe-game.pl

```
1 :- use_module(minimax).
2 :- use_module(tictactoe).
3
4 :- include('tictactoe.pl').
5 :- include('minimax.pl').
6
7 % bestMove(+Pos, -NextPos)
8 % Compute the best Next Position from Position Pos
9 % with minimax or alpha-beta algorithm.
10 bestMove(Pos, NextPos) :-
11     minimax(Pos, NextPos, _).
12
13 % play
14 % Start the game.
15 play :-
16     nl,
17     write('====='), nl,
18     write('= Prolog TicTacToe ='), nl,
19     write('====='), nl, nl,
20     write('Rem : x starts the game'), nl,
21     playAskColor.
22
23 % playAskColor
24 % Ask the color for the human player and start the game with it.
25 playAskColor :-
```



```

26         nl, write('Color for human player ? (x or o)'), nl,
27         read(Player), nl,
28         (
29             Player \= o, Player \= x, !,          % If not x or o -> not a valid
color
30             write('Error : not a valid color !'), nl,
31             playAskColor                          % Ask again
32             ;
33             EmptyBoard = [0, 0, 0, 0, 0, 0, 0, 0, 0],
34             show(EmptyBoard), nl,
35
36             % Start the game with color and emptyBoard
37             play([x, play, EmptyBoard], Player)
38         ).
39
40 % play(+Position, +HumanPlayer)
41 % If next player to play in position is equal to HumanPlayer -> Human must
play
42 % Ask to human what to do.
43 play([Player, play, Board], Player) :- !,
44     nl, write('Next move ?'), nl,
45     read(Pos), nl,                                % Ask human where to
play
46     (
47         humanMove([Player, play, Board], [NextPlayer, State, NextBoard], Pos),
48         !,
49         show(NextBoard),
50         (
51             State = win, !,                          % If Player win -> stop
52             nl, write('End of game : '),
53             write(Player), write(' win !'), nl, nl
54             ;
55             State = draw, !,                          % If draw -> stop
56             nl, write('End of game : '),
57             write(' draw !'), nl, nl
58             ;
59             play([NextPlayer, play, NextBoard], Player) % Else -> continue the
game
60         )
61         ;
62         write('-> Bad Move !'), nl,                    % If humanMove fail -> bad
move
63         play([Player, play, Board], Player)          % Ask again
64     ).
65
66 % play(+Position, +HumanPlayer)
67 % If it is not human who must play -> Computer must play
68 % Compute the best move for computer with minimax or alpha-beta.
69 play([Player, play, Board], HumanPlayer) :-
70     nl, write('Computer play : '), nl, nl,
71     % Compute the best move
72     bestMove([Player, play, Board], [NextPlayer, State, BestSuccBoard]),
73     show(BestSuccBoard),
74     (
75         State = win, !,                                % If Player win ->
stop
76         nl, write('End of game : '),
77         write(Player), write(' win !'), nl, nl
78         ;
79         State = draw, !,                                % If draw -> stop
80         nl, write('End of game : '), write(' draw !'), nl, nl
81         ;
82         % Else -> continue the game
83         play([NextPlayer, play, BestSuccBoard], HumanPlayer)
84     ).
85
86 % nextPlayer(X1, X2)
87 % True if X2 is the next player to play after X1.
88 nextPlayer(o, x).
89 nextPlayer(x, o).
90

```

```

91 % When human play
92 humanMove([X1, play, Board], [X2, State, NextBoard], Pos) :-
93     nextPlayer(X1, X2),
94     set1(Pos, X1, Board, NextBoard),
95     (
96         winPos(X1, NextBoard), !, State = win ;
97         drawPos(X1, NextBoard), !, State = draw ;
98         State = play
99     ).
100
101 % set1(+Elem, +Pos, +List, -ResList).
102 % Set Elem at Position Pos in List => Result in ResList.
103 % Rem : counting starts at 1.
104 set1(1, E, [X|Ls], [E|Ls]) :- !, X = 0.
105
106 set1(P, E, [X|Ls], [X|L2s]) :-
107     number(P),
108     P1 is P - 1,
109     set1(P1, E, Ls, L2s).
110
111 % show(+Board)
112 % Show the board to current output.
113 show([X1, X2, X3, X4, X5, X6, X7, X8, X9]) :-
114     write('  '), show2(X1),
115     write(' | '), show2(X2),
116     write(' | '), show2(X3), nl,
117     write(' -----'), nl,
118     write('  '), show2(X4),
119     write(' | '), show2(X5),
120     write(' | '), show2(X6), nl,
121     write(' -----'), nl,
122     write('  '), show2(X7),
123     write(' | '), show2(X8),
124     write(' | '), show2(X9), nl.
125
126 % show2(+Term)
127 % Write the term to current outupt
128 % Replace 0 by ' '.
129 show2(X) :-
130     X = 0, !,
131     write(' ').
132
133 show2(X) :-
134     write(X).
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160

```

161
162
163
164
165
166
167
168
169
170
171

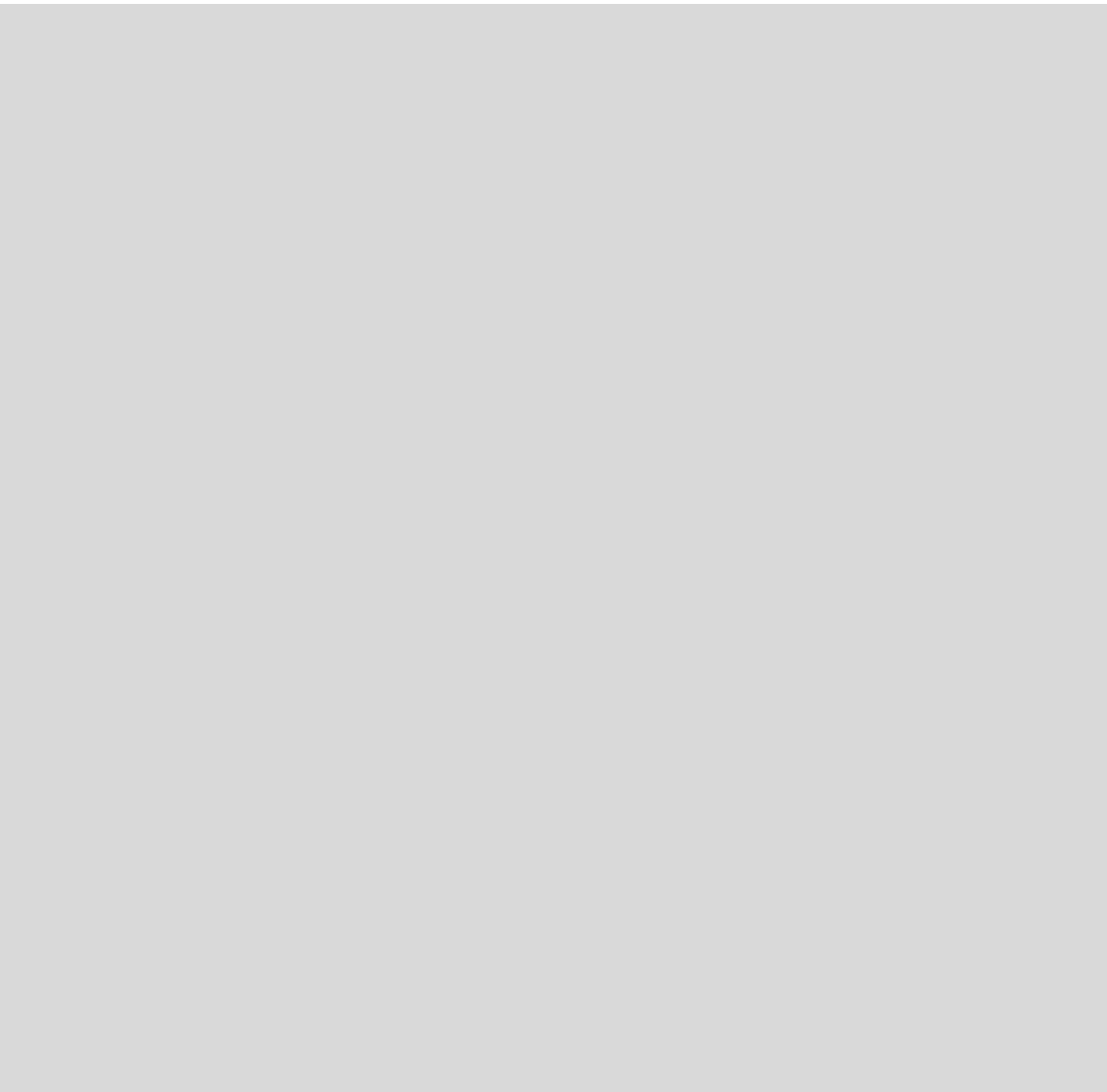
172
173
174
175

176
177

178
179
180
181
182
183
184
185
186
187
188

189
190
191

192
193
194
195
196



Q: <https://gist.github.com/AndreFCruz/13df05bcc3f00902992493a44834748f>

tictactoe.pl

```
1 % move(+Pos, -NextPos)
2 % True if there is a legal (according to rules) move from Pos to NextPos.
3 move([X1, play, Board], [X2, win, NextBoard]) :-
4     nextPlayer(X1, X2),
5     move_aux(X1, Board, NextBoard),
6     winPos(X1, NextBoard), !.
7
8 move([X1, play, Board], [X2, draw, NextBoard]) :-
9     nextPlayer(X1, X2),
10    move_aux(X1, Board, NextBoard),
11    drawPos(X1, NextBoard), !.
12
13 move([X1, play, Board], [X2, play, NextBoard]) :-
14     nextPlayer(X1, X2),
15     move_aux(X1, Board, NextBoard).
16
17 % move_aux(+Player, +Board, -NextBoard)
18 % True if NextBoard is Board with an empty case replaced by Player mark.
19 move_aux(P, [0|Bs], [P|Bs]).
20
21 move_aux(P, [B|Bs], [B|B2s]) :-
22     move_aux(P, Bs, B2s).
23
24 % min_to_move(+Pos)
25 % True if the next player to play is the MIN player.
26 min_to_move([o, _, _]).
```

```

27
28 % max_to_move(+Pos)
29 % True if the next player to play is the MAX player.
30 max_to_move([x, _, _]).
31
32 % utility(+Pos, -Val) :-
33 % True if Val the the result of the evaluation function at Pos.
34 % We will only evaluate for final position.
35 % So we will only have MAX win, MIN win or draw.
36 % We will use 1 when MAX win
37 %           -1 when MIN win
38 %           0 otherwise.
39 utility([o, win, _], 1).      % Previous player (MAX) has win.
40 utility([x, win, _], -1).    % Previous player (MIN) has win.
41 utility([_, draw, _], 0).
42
43 % winPos(+Player, +Board)
44 % True if Player win in Board.
45 winPos(P, [X1, X2, X3, X4, X5, X6, X7, X8, X9]) :-
46     equal(X1, X2, X3, P) ;    % 1st line
47     equal(X4, X5, X6, P) ;    % 2nd line
48     equal(X7, X8, X9, P) ;    % 3rd line
49     equal(X1, X4, X7, P) ;    % 1st col
50     equal(X2, X5, X8, P) ;    % 2nd col
51     equal(X3, X6, X9, P) ;    % 3rd col
52     equal(X1, X5, X9, P) ;    % 1st diag
53     equal(X3, X5, X7, P).    % 2nd diag
54
55 % drawPos(+Player, +Board)
56 % True if the game is a draw.
57 drawPos(_, Board) :-
58     \+ member(0, Board).
59
60 % equal(+W, +X, +Y, +Z).
61 % True if W = X = Y = Z.
62 equal(X, X, X, X).
63

```

Q: <https://gist.github.com/AndreFCruz/13df05bcc3f00902992493a44834748f>

Tic-tac-toe in JAVA

```

1 // A simple program to demonstrate
2 // Tic-Tac-Toe Game.
3 import java.util.*;
4
5 public class GFG {
6
7     static String[] board;
8     static String turn;
9
10
11     // CheckWinner method will
12     // decide the combination
13     // of three box given below.
14     static String checkWinner()
15     {
16         for (int a = 0; a < 8; a++) {
17             String line = null;
18
19             switch (a) {
20                 case 0:
21                     line = board[0] + board[1] + board[2];
22                     break;
23                 case 1:
24

```

```

        line = board[3] + board[4] + board[5];
        break;
    case 2:
        line = board[6] + board[7] + board[8];
        break;
    case 3:
        line = board[0] + board[3] + board[6];
        break;
    case 4:
        line = board[1] + board[4] + board[7];
        break;
    case 5:
        line = board[2] + board[5] + board[8];
        break;
    case 6:
        line = board[0] + board[4] + board[8];
        break;
    case 7:
        line = board[2] + board[4] + board[6];
        break;
    }
    //For X winner
    if (line.equals("XXX")) {
        return "X";
    }

    // For O winner
    else if (line.equals("OOO")) {
        return "O";
    }
}

for (int a = 0; a < 9; a++) {
    if (Arrays.asList(board).contains(
        String.valueOf(a + 1))) {
        break;
    }
    else if (a == 8) {
        return "draw";
    }
}

// To enter the X Or O at the exact place on board.
System.out.println(
    turn + "'s turn; enter a slot number to place "
    + turn + " in:");
return null;
}

// To print out the board.
/* |---|---|---|
| 1 | 2 | 3 |
|-----|
| 4 | 5 | 6 |
|-----|
| 7 | 8 | 9 |
|---|---|---|*/

static void printBoard()
{
    System.out.println("|---|---|---|");
    System.out.println("| " + board[0] + " | "
        + board[1] + " | " + board[2]
        + " |");
    System.out.println("|-----|");
    System.out.println("| " + board[3] + " | "
        + board[4] + " | " + board[5]
        + " |");
    System.out.println("|-----|");
    System.out.println("| " + board[6] + " | "
        + board[7] + " | " + board[8]

```

```

        + " |");
    System.out.println("|---|---|---|");
}

public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    board = new String[9];
    turn = "X";
    String winner = null;

    for (int a = 0; a < 9; a++) {
        board[a] = String.valueOf(a + 1);
    }

    System.out.println("Welcome to 3x3 Tic Tac Toe.");
    printBoard();

    System.out.println(
        "X will play first. Enter a slot number to place X in:");

    while (winner == null) {
        int numInput;

        // Exception handling.
        // numInput will take input from user like from 1 to 9.
        // If it is not in range from 1 to 9.
        // then it will show you an error "Invalid input."
        try {
            numInput = in.nextInt();
            if (!(numInput > 0 && numInput <= 9)) {
                System.out.println(
                    "Invalid input; re-enter slot num-
ber:");
                continue;
            }
        }
        catch (InputMismatchException e) {
            System.out.println(
                "Invalid input; re-enter slot number:");
            continue;
        }

        // This game has two player x and O.
        // Here is the logic to decide the turn.
        if (board[numInput - 1].equals(
            String.valueOf(numInput))) {
            board[numInput - 1] = turn;

            if (turn.equals("X")) {
                turn = "O";
            }
            else {
                turn = "X";
            }

            printBoard();
            winner = checkWinner();
        }
        else {
            System.out.println(
                "Slot already taken; re-enter slot num-
ber:");
        }
    }

    // If no one win or lose from both player x and O.
    // then here is the logic to print "draw".
    if (winner.equalsIgnoreCase("draw")) {
        System.out.println(
            "It's a draw! Thanks for playing.");
    }
}

```

```
    }  
  
    // For winner -to display Congratulations! message.  
    else {  
        System.out.println(  
            "Congratulations! " + winner  
            + "'s have won! Thanks for playing.");  
    }  
    in.close();  
}  
}
```

Q:

Abbildungen und Skizzen entstammen den folgende ClipArt-Sammlungen:

/I/ 29.000 Mega ClipArts; NBG EDV Handels- und Verlags AG; 1997

/II/

andere Quellen sind direkt angegeben.

Alle anderen Abbildungen sind geistiges Eigentum von:

/I/ lern-soft-projekt: drews (c,p) 1998 – 2023 lsp: dre
für die Verwendung außerhalb dieses Skriptes gilt für sie die Lizenz:



CC-BY-NC-SA



Lizenz-Erklärungen und –Bedingungen: <http://de.creativecommons.org/was-ist-cc/>
andere Verwendungen nur mit schriftlicher Vereinbarung!!!

verwendete freie Software:

- **Inkscape** von: inkscape.org (www.inkscape.org)
- **CmapTools** von: Institute for Human and Maschine Cognition (www.ihmc.us)

⌘- (c,p)1987 - 2023 lern-soft-projekt: drews -⌘
⌘- drews@lern-soft-projekt.de -⌘
⌘- <http://www.lern-soft-projekt.de> -⌘
⌘- 18069 Rostock; Luise-Otto-Peters-Ring 25 -⌘
⌘- Tel/AB (0381) 760 12 18 FAX 760 12 11 -⌘