

Sebastian  
Raschka

Vahid  
Mirjalili

3.,  
aktualisierte  
und erweiterte  
Auflage

# Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-learn

Das umfassende **Praxis-Handbuch**  
für Data Science, Deep Learning und Predictive Analytics

# Inhaltsverzeichnis

Über die Autoren .....	17
Über die Korrektoren .....	19
Über den Fachkorrektor der deutschen Ausgabe .....	20
Einleitung .....	21
Einstieg in Machine Learning .....	21
Zum Inhalt des Buches .....	23
Was Sie benötigen .....	26
Codebeispiele herunterladen .....	26
Konventionen im Buch .....	26
1 Wie Computer aus Daten lernen können .....	29
1.1 Intelligente Maschinen, die Daten in Wissen verwandeln .....	29
1.2 Die drei Arten des Machine Learnings .....	30
1.2.1 Mit überwachtem Lernen Vorhersagen treffen .....	31
1.2.2 Interaktive Aufgaben durch Reinforcement Learning lösen .....	34
1.2.3 Durch unüberwachtes Lernen verborgene Strukturen erkennen .....	35
1.3 Grundlegende Terminologie und Notation .....	36
1.3.1 Im Buch verwendete Notation und Konventionen .....	37
1.3.2 Terminologie .....	38
1.4 Entwicklung eines Systems für das Machine Learning .....	39
1.4.1 Vorverarbeitung: Daten in Form bringen .....	40
1.4.2 Trainieren und Auswählen eines Vorhersagmodells .....	40
1.4.3 Bewertung von Modellen und Vorhersage anhand unbekannter Dateninstanzen .....	41
1.5 Machine Learning mit Python .....	42
1.5.1 Python und Python-Pakete installieren .....	42
1.5.2 Verwendung der Python-Distribution Anaconda .....	43
1.5.3 Pakete für wissenschaftliches Rechnen, Data Science und Machine Learning .....	43
1.6 Zusammenfassung .....	44

<b>2</b>	<b>Lernalgorithmen für die Klassifikation trainieren . . . . .</b>	45
2.1	Künstliche Neuronen: Ein kurzer Blick auf die Anfänge des Machine Learnings . . . . .	45
2.1.1	Formale Definition eines künstlichen Neurons . . . . .	46
2.1.2	Die Perzeptron-Lernregel . . . . .	48
2.2	Implementierung eines Perzeptron-Lernalgorithmus in Python . . . . .	51
2.2.1	Eine objektorientierte Perzeptron-API . . . . .	51
2.2.2	Trainieren eines Perzeptron-Modells mit der Iris-Datensammlung . . . . .	55
2.3	Adaptive lineare Neuronen und die Konvergenz des Lernens . . . . .	61
2.3.1	Straffunktionen mit dem Gradientenabstiegsverfahren minimieren . . . . .	62
2.3.2	Implementierung eines adaptiven linearen Neurons in Python . . . . .	64
2.3.3	Verbesserung des Gradientenabstiegsverfahrens durch Merkmalstandardisierung . . . . .	69
2.3.4	Large Scale Machine Learning und stochastisches Gradientenabstiegsverfahren . . . . .	71
2.4	Zusammenfassung . . . . .	77
<b>3</b>	<b>Machine-Learning-Klassifikatoren mit scikit-learn verwenden . . . . .</b>	79
3.1	Auswahl eines Klassifikationsalgorithmus . . . . .	79
3.2	Erste Schritte mit scikit-learn: Trainieren eines Perzeptrons . . . . .	80
3.3	Klassenwahrscheinlichkeiten durch logistische Regression modellieren . . . . .	86
3.3.1	Logistische Regression und bedingte Wahrscheinlichkeiten . . . . .	87
3.3.2	Gewichte der logistischen Straffunktion ermitteln . . . . .	91
3.3.3	Konvertieren einer Adaline-Implementierung in einen Algorithmus für eine logistische Regression . . . . .	93
3.3.4	Trainieren eines logistischen Regressionsmodells mit scikit-learn . . . . .	98
3.3.5	Überanpassung durch Regularisierung verhindern . . . . .	101
3.4	Maximum-Margin-Klassifikation mit Support Vector Machines . . . . .	104
3.4.1	Maximierung des Randbereichs . . . . .	105
3.4.2	Handhabung des nicht linear trennbaren Falls mit Schlupfvariablen . . . . .	106
3.4.3	Alternative Implementierungen in scikit-learn . . . . .	108

3.5	Nichtlineare Aufgaben mit einer Kernel-SVM lösen . . . . .	109
3.5.1	Kernel-Methoden für linear nicht trennbare Daten. . . . .	109
3.5.2	Mit dem Kernel-Trick Hyperebenen in höherdimensionalen Räumen finden. . . . .	111
3.6	Lernen mit Entscheidungsbäumen . . . . .	115
3.6.1	Maximierung des Informationsgewinns: Daten ausreizen . . . . .	116
3.6.2	Konstruktion eines Entscheidungsbaums . . . . .	120
3.6.3	Mehrere Entscheidungsbäume zu einem Random Forest kombinieren . . . . .	124
3.7	k-Nearest-Neighbors: Ein Lazy-Learning-Algorithmus. . . . .	127
3.8	Zusammenfassung . . . . .	130
4	<b>Gut geeignete Trainingsdatenmengen: Datenvorverarbeitung</b> . . . . .	133
4.1	Umgang mit fehlenden Daten . . . . .	133
4.1.1	Fehlende Werte in Tabellendaten . . . . .	134
4.1.2	Instanzen oder Merkmale mit fehlenden Daten entfernen. . . . .	135
4.1.3	Fehlende Werte ergänzen . . . . .	136
4.1.4	Die Schätzer-API von scikit-learn . . . . .	137
4.2	Handhabung kategorialer Daten . . . . .	138
4.2.1	Codierung kategorialer Daten mit pandas . . . . .	139
4.2.2	Zuweisung von ordinalen Merkmalen . . . . .	139
4.2.3	Codierung der Klassenbezeichnungen. . . . .	140
4.2.4	One-hot-Codierung der nominalen Merkmale. . . . .	141
4.3	Aufteilung einer Datensammlung in Trainings- und Testdaten . . . . .	145
4.4	Anpassung der Merkmale. . . . .	148
4.5	Auswahl aussagekräftiger Merkmale. . . . .	150
4.5.1	L1- und L2-Regularisierung als Straffunktionen . . . . .	151
4.5.2	Geometrische Interpretation der L2-Regularisierung . . . . .	151
4.5.3	Dünn besetzte Lösungen mit L1-Regularisierung . . . . .	153
4.5.4	Algorithmen zur sequenziellen Auswahl von Merkmälern . . . . .	157
4.6	Beurteilung der Bedeutung von Merkmalen mit Random Forests. . . . .	164
4.7	Zusammenfassung . . . . .	167
5	<b>Datenkomprimierung durch Dimensionsreduktion</b> . . . . .	169
5.1	Unüberwachte Dimensionsreduktion durch	

Hauptkomponentenanalyse . . . . .	169
5.1.1 Schritte bei der Hauptkomponentenanalyse . . . . .	170
5.1.2 Schrittweise Extraktion der Hauptkomponenten . . . . .	171
5.1.3 Totale Varianz und erklärte Varianz . . . . .	174
5.1.4 Merkmalstransformation . . . . .	176
5.1.5 Hauptkomponentenanalyse mit scikit-learn . . . . .	179
5.2 Überwachte Datenkomprimierung durch lineare Diskriminanzanalyse . . . . .	183
5.2.1 Hauptkomponentenanalyse kontra lineare Diskriminanzanalyse . . . . .	183
5.2.2 Die interne Funktionsweise der linearen Diskriminanzanalyse . . . . .	184
5.2.3 Berechnung der Streumatrizen . . . . .	185
5.2.4 Auswahl linearer Diskriminanten für den neuen Merkmalsraum . . . . .	187
5.2.5 Projektion in den neuen Merkmalsraum . . . . .	190
5.2.6 LDA mit scikit-learn . . . . .	191
5.3 Kernel-Hauptkomponentenanalyse für nichtlineare Zuordnungen verwenden . . . . .	193
5.3.1 Kernel-Funktionen und der Kernel-Trick . . . . .	194
5.3.2 Implementierung einer Kernel-Hauptkomponentenanalyse in Python . . . . .	198
5.3.3 Projizieren neuer Datenpunkte . . . . .	206
5.3.4 Kernel-Hauptkomponentenanalyse mit scikit-learn . . . . .	210
5.4 Zusammenfassung . . . . .	211
<b>6 Bewährte Verfahren zur Modellbewertung und Hyperparameter-Optimierung . . . . .</b>	<b>213</b>
6.1 Arbeitsabläufe mit Pipelines optimieren . . . . .	213
6.1.1 Die Wisconsin-Brustkrebs-Datensammlung . . . . .	213
6.1.2 Transformer und Schätzer in einer Pipeline kombinieren . . . . .	215
6.2 Beurteilung des Modells durch k-fache Kreuzvalidierung . . . . .	217
6.2.1 Holdout-Methode . . . . .	218
6.2.2 k-fache Kreuzvalidierung . . . . .	219
6.3 Algorithmen mit Lern- und Validierungskurven debuggen . . . . .	223
6.3.1 Probleme mit Bias und Varianz anhand von Lernkurven erkennen . . . . .	224

6.3.2	Überanpassung und Unteranpassung anhand von Validierungskurven erkennen .....	227
6.4	Feinabstimmung eines Lernmodells durch Grid Search .....	229
6.4.1	Optimierung der Hyperparameter durch Grid Search .....	230
6.4.2	Algorithmenauswahl durch verschachtelte Kreuzvalidierung .....	232
6.5	Verschiedene Kriterien zur Leistungsbewertung .....	234
6.5.1	Interpretation einer Verwechslungsmatrix .....	234
6.5.2	Optimierung der Genauigkeit und der Trefferquote eines Klassifikationsmodells .....	236
6.5.3	Receiver-Operating-Characteristic-Diagramme .....	238
6.5.4	Bewertungskriterien für Mehrklassen-Klassifikationen .....	241
6.6	Handhabung unausgewogener Klassenverteilung .....	242
6.7	Zusammenfassung .....	245
7	<b>Kombination verschiedener Modelle für das Ensemble Learning . . . . .</b>	247
7.1	Ensemble Learning .....	247
7.2	Klassifikatoren durch Mehrheitsentscheidung kombinieren .....	251
7.2.1	Implementierung eines einfachen Mehrheitsentscheidungs-Klassifikators .....	251
7.2.2	Vorhersagen nach dem Mehrheitsentscheidungsprinzip treffen .....	258
7.3	Bewertung und Abstimmung des Klassifikator-Ensembles .....	261
7.4	Bagging: Klassifikator-Ensembles anhand von Bootstrap-Stichproben entwickeln .....	268
7.4.1	Bagging kurz zusammengefasst .....	269
7.4.2	Klassifikation der Wein-Datensammlung durch Bagging . . . . .	270
7.5	Schwache Klassifikatoren durch adaptives Boosting verbessern .....	274
7.5.1	Funktionsweise des Boostings .....	274
7.5.2	AdaBoost mit scikit-learn anwenden .....	278
7.6	Zusammenfassung .....	282
8	<b>Machine Learning zur Analyse von Stimmungslagen nutzen . . . . .</b>	283
8.1	Die IMDb-Filmdatenbank .....	283
8.1.1	Herunterladen der Datensammlung .....	284
8.1.2	Vorverarbeiten der Filmbewertungsdaten .....	284
8.2	Das Bag-of-words-Modell .....	286
8.2.1	Wörter in Merkmalsvektoren umwandeln .....	287
8.2.2	Beurteilung der Wortrelevanz durch das Tf-idf-Maß .....	289

8.2.3	Textdaten bereinigen . . . . .	291
8.2.4	Dokumente in Tokens zerlegen. . . . .	293
8.3	Ein logistisches Regressionsmodell für die Dokumentklassifikation trainieren . . . . .	295
8.4	Verarbeitung großer Datenmengen: Online-Algorithmen und Out-of-Core Learning. . . . .	298
8.5	Topic Modeling mit latenter Dirichlet-Allokation . . . . .	302
8.5.1	Aufteilung von Texten mit der LDA . . . . .	303
8.5.2	LDA mit scikit-learn . . . . .	303
8.6	Zusammenfassung . . . . .	307
<b>9</b>	<b>Einbettung eines Machine-Learning-Modells in eine Webanwendung</b> . . . . .	309
9.1	Serialisierung angepasster Schätzer mit scikit-learn. . . . .	309
9.2	Einrichtung einer SQLite-Datenbank zum Speichern von Daten . . . . .	313
9.3	Entwicklung einer Webanwendung mit Flask. . . . .	315
9.3.1	Die erste Webanwendung mit Flask . . . . .	316
9.3.2	Formularvalidierung und -ausgabe . . . . .	318
9.4	Der Filmbewertungsklassifikator als Webanwendung . . . . .	324
9.4.1	Dateien und Ordner – die Verzeichnisstruktur . . . . .	326
9.4.2	Implementierung der Hauptanwendung app.py . . . . .	326
9.4.3	Einrichtung des Bewertungsformulars . . . . .	329
9.4.4	Eine Vorlage für die Ergebnisseite erstellen . . . . .	330
9.5	Einrichtung der Webanwendung auf einem öffentlich zugänglichen Webserver . . . . .	333
9.5.1	Erstellen eines Benutzerkontos bei PythonAnywhere . . . . .	333
9.5.2	Hochladen der Filmbewertungsanwendung . . . . .	334
9.5.3	Updaten des Filmbewertungsklassifikators . . . . .	335
9.6	Zusammenfassung . . . . .	338
<b>10</b>	<b>Vorhersage stetiger Zielvariablen durch Regressionsanalyse</b> . . . . .	339
10.1	Lineare Regression. . . . .	339
10.1.1	Ein einfaches lineares Regressionsmodell . . . . .	340
10.1.2	Multiple lineare Regression . . . . .	341
10.2	Die Boston-Housing-Datensammlung. . . . .	342
10.2.1	Einlesen der Datenmenge in einen DataFrame . . . . .	342
10.2.2	Visualisierung der wichtigen Eigenschaften einer Datenmenge . . . . .	344

10.2.3	Zusammenhänge anhand der Korrelationsmatrix erkennen . . . . .	346
10.3	Implementierung eines linearen Regressionsmodells mit der Methode der kleinsten Quadrate . . . . .	348
10.3.1	Berechnung der Regressionsparameter mit dem Gradientenabstiegsverfahren. . . . .	349
10.3.2	Schätzung der Koeffizienten eines Regressionsmodells mit scikit-learn . . . . .	353
10.4	Anpassung eines robusten Regressionsmodells mit dem RANSAC-Algorithmus . . . . .	355
10.5	Bewertung der Leistung linearer Regressionsmodelle . . . . .	358
10.6	Regularisierungsverfahren für die Regression einsetzen. . . . .	361
10.7	Polynomiale Regression: Umwandeln einer linearen Regression in eine Kurve . . . . .	363
10.7.1	Hinzufügen polynomialer Terme mit scikit-learn . . . . .	364
10.7.2	Modellierung nichtlinearer Zusammenhänge in der Boston-Housing-Datensammlung . . . . .	366
10.8	Handhabung nichtlinearer Beziehungen mit Random Forests. . . . .	369
10.8.1	Entscheidungsbaum-Regression. . . . .	370
10.8.2	Random-Forest-Regression . . . . .	371
10.9	Zusammenfassung . . . . .	375
<b>11</b>	<b>Verwendung von Daten ohne Label: Clusteranalyse. . . . .</b>	<b>377</b>
11.1	Gruppierung von Objekten nach Ähnlichkeit mit dem k-Means-Algorithmus . . . . .	377
11.1.1	k-Means-Clustering mit scikit-learn . . . . .	378
11.1.2	Der k-Means++-Algorithmus. . . . .	383
11.1.3	»Crisp« und »soft« Clustering. . . . .	384
11.1.4	Die optimale Anzahl der Cluster mit dem Ellenbogenkriterium ermitteln . . . . .	386
11.1.5	Quantifizierung der Clustering-Güte mit Silhouettendiagrammen . . . . .	388
11.2	Cluster als hierarchischen Baum organisieren . . . . .	393
11.2.1	Gruppierung von Clustern . . . . .	393
11.2.2	Hierarchisches Clustering mittels einer Distanzmatrix . . . . .	395
11.2.3	Dendrogramme und Heatmaps verknüpfen . . . . .	399
11.2.4	Agglomeratives Clustering mit scikit-learn . . . . .	401
11.3	Bereiche hoher Dichte mit DBSCAN ermitteln . . . . .	402
11.4	Zusammenfassung . . . . .	407

<b>12</b>	<b>Implementierung eines künstlichen neuronalen Netzes . . . . .</b>	409
12.1	Modellierung komplexer Funktionen mit künstlichen neuronalen Netzen . . . . .	409
12.1.1	Einschichtige neuronale Netze . . . . .	411
12.1.2	Mehrschichtige neuronale Netzarchitektur . . . . .	413
12.1.3	Aktivierung eines neuronalen Netzes durch Vorwärtspropagation . . . . .	416
12.2	Klassifikation handgeschriebener Ziffern . . . . .	418
12.2.1	Die MNIST-Datensammlung. . . . .	419
12.2.2	Implementierung eines mehrschichtigen Perzeptrons . . . . .	426
12.3	Trainieren eines künstlichen neuronalen Netzes . . . . .	438
12.3.1	Berechnung der logistischen Straffunktion . . . . .	439
12.3.2	Ein Gespür für die Backpropagation entwickeln . . . . .	441
12.3.3	Trainieren neuronaler Netze durch Backpropagation . . . . .	443
12.4	Konvergenz in neuronalen Netzen. . . . .	447
12.5	Abschließende Bemerkungen zur Implementierung neuronaler Netze . . . . .	448
12.6	Zusammenfassung . . . . .	448
<b>13</b>	<b>Parallelisierung des Trainings neuronaler Netze mit TensorFlow . . . . .</b>	451
13.1	TensorFlow und Trainingsleistung . . . . .	451
13.1.1	Herausforderungen . . . . .	451
13.1.2	Was genau ist TensorFlow? . . . . .	453
13.1.3	TensorFlow erlernen . . . . .	454
13.2	Erste Schritte mit TensorFlow . . . . .	455
13.2.1	TensorFlow installieren . . . . .	455
13.2.2	Tensoren in TensorFlow erstellen. . . . .	456
13.2.3	Datentyp und Format eines Tensors ändern . . . . .	457
13.2.4	Anwendung mathematischer Operationen auf Tensoren . . . . .	458
13.2.5	Tensoren aufteilen, stapeln und verknüpfen . . . . .	460
13.3	Eingabe-Pipelines mit <code>tf.data</code> erstellen – die Dataset-API von TensorFlow . . . . .	462
13.3.1	Ein TensorFlow-Dataset anhand vorhandener Tensoren erstellen . . . . .	462
13.3.2	Zwei Tensoren zu einer Datenmenge vereinen . . . . .	463
13.3.3	Durchmischen, Batch erstellen und wiederholen . . . . .	465
13.3.4	Erstellen einer Datenmenge anhand lokal gespeicherter Dateien . . . . .	468
13.3.5	Zugriff auf die Datenmengen der <code>tensorflow_datasets</code> -Bibliothek . . . . .	472

13.4	Entwicklung eines NN-Modells mit TensorFlow .....	478
13.4.1	Die Keras-API (tf.keras) von TensorFlow .....	478
13.4.2	Entwicklung eines linearen Regressionsmodells .....	479
13.4.3	Trainieren des Modells mit den Methoden .compile() und .fit() .....	484
13.4.4	Entwicklung eines mehrschichtigen Perzeptrons zur Klassifikation der Iris-Datensammlung .....	485
13.4.5	Bewertung des trainierten Modells mit der Testdatenmenge .....	490
13.4.6	Das trainierte Modell speichern und einlesen .....	490
13.5	Auswahl der Aktivierungsfunktionen mehrschichtiger neuronaler Netze .....	491
13.5.1	Die logistische Funktion kurz zusammengefasst .....	492
13.5.2	Wahrscheinlichkeiten bei der Mehrklassen-Klassifikation mit der softmax-Funktion schätzen .....	494
13.5.3	Verbreiterung des Ausgabespektrums mittels Tangens hyperbolicus .....	495
13.5.4	Aktivierung mittels ReLU .....	498
13.6	Zusammenfassung .....	499
14	<b>Die Funktionsweise von TensorFlow im Detail .....</b>	501
14.1	Grundlegende Merkmale von TensorFlow .....	502
14.2	TensorFlows Berechnungsgraphen: Migration nach TensorFlow v2 .....	503
14.2.1	Funktionsweise von Berechnungsgraphen .....	503
14.2.2	Erstellen eines Graphen in TensorFlow v1.x .....	504
14.2.3	Migration eines Graphen nach TensorFlow v2 .....	505
14.2.4	Eingabedaten einlesen mit TensorFlow v1.x .....	506
14.2.5	Eingabedaten einlesen mit TensorFlow v2 .....	506
14.2.6	Beschleunigung von Berechnungen mit Funktionsdekoratoren .....	507
14.3	TensorFlows Variablenobjekte zum Speichern und Aktualisieren von Modellparametern .....	509
14.4	Gradientenberechnung durch automatisches Differenzieren und GradientTape .....	514
14.4.1	Berechnung der Gradienten der Verlustfunktion bezüglich trainierbarer Variablen .....	514
14.4.2	Berechnung der Gradienten bezüglich nicht trainierbarer Tensoren .....	516

14.4.3	Ressourcen für mehrfache Gradientenberechnung erhalten . . . . .	516
14.5	Vereinfachung der Implementierung gebräuchlicher Architekturen mit der Keras-API . . . . .	517
14.5.1	Lösen einer XOR-Klassifikationsaufgabe . . . . .	521
14.5.2	Flexiblere Modellerstellung mit Keras' funktionaler API . . . . .	526
14.5.3	Modelle mit Keras' »Model«-Klasse implementieren . . . . .	528
14.5.4	Benutzerdefinierte Keras-Schichten . . . . .	529
14.6	TensorFlows Schätzer . . . . .	533
14.6.1	Merkmalsspalten . . . . .	534
14.6.2	Machine Learning mit vorgefertigten Schätzern . . . . .	538
14.6.3	Klassifikation handgeschriebener Ziffern mit Schätzern . . . . .	543
14.6.4	Benutzerdefinierte Schätzer anhand eines Keras-Modells erstellen . . . . .	545
14.7	Zusammenfassung . . . . .	548
15	<b>Bildklassifikation mit Deep Convolutional Neural Networks</b> . . . . .	549
15.1	Bausteine von Convolutional Neural Networks . . . . .	549
15.1.1	CNNs und Merkmalshierarchie . . . . .	550
15.1.2	Diskrete Faltungen . . . . .	552
15.1.3	Subsampling . . . . .	561
15.2	Implementierung eines CNNs . . . . .	563
15.2.1	Verwendung mehrerer Eingabe- oder Farbkanäle . . . . .	563
15.2.2	Regularisierung eines neuronalen Netzes mit Dropout . . . . .	566
15.2.3	Verlustfunktionen für Klassifikationen . . . . .	570
15.3	Implementierung eines tiefen CNNs mit TensorFlow . . . . .	572
15.3.1	Die mehrschichtige CNN-Architektur . . . . .	573
15.3.2	Einlesen und Vorverarbeiten der Daten . . . . .	574
15.3.3	Implementierung eines CNNs mit TensorFlow Keras-API . . . . .	575
15.4	Klassifikation des Geschlechts anhand von Porträtfotos mit einem CNN . . . . .	582
15.4.1	Einlesen der CelebA-Datenmenge . . . . .	582
15.4.2	Bildtransformation und Datenaugmentation . . . . .	583
15.4.3	Training eines CNN-Klassifikators . . . . .	590
15.5	Zusammenfassung . . . . .	596
16	<b>Modellierung sequenzieller Daten durch rekurrente neuronale Netze</b> . . . . .	597
16.1	Sequenzielle Daten . . . . .	597

16.1.1	Modellierung sequenzieller Daten: Die Reihenfolge ist von Bedeutung .....	598
16.1.2	Repräsentierung von Sequenzen .....	598
16.1.3	Verschiedene Kategorien der Sequenzmodellierung.....	599
16.2	Sequenzmodellierung mit RNNs .....	601
16.2.1	Struktur und Ablauf eines RNNs .....	601
16.2.2	Aktivierungen eines RNNs berechnen .....	603
16.2.3	Rückkopplung mit der verdeckten Schicht oder der Ausgabeschicht.....	606
16.2.4	Probleme bei der Erkennung weitreichender Interaktionen .....	609
16.2.5	LSTM-Speicherzellen.....	610
16.3	Implementierung von RNNs zur Sequenzmodellierung mit TensorFlow.....	612
16.3.1	Projekt 1: Vorhersage der Stimmungslage von IMDb-Filmbewertungen .....	612
16.3.2	Projekt 2: Sprachmodellierung durch Zeichen mit TensorFlow .....	629
16.4	Sprache mit dem Transformer-Modell verstehen.....	642
16.4.1	Der Mechanismus der Selbst-Aufmerksamkeit.....	643
16.4.2	Multi-Head-Attention und Transformer-Block .....	646
16.5	Zusammenfassung .....	647
17	<b>Synthetisieren neuer Daten mit Generative Adversarial Networks .....</b>	649
17.1	Einführung in GANs.....	649
17.1.1	Autoencoder .....	650
17.1.2	Generative Modelle zum Synthetisieren neuer Daten.....	652
17.1.3	Mit GANs neue Beispiele erzeugen .....	654
17.1.4	Die Verlustfunktion des Generator- und Diskriminatoren-Netzes in einem GAN-Modell .....	655
17.2	Ein GAN von Grund auf implementieren.....	658
17.2.1	GAN-Modelle mit Google Colab trainieren .....	658
17.2.2	Implementierung der Generator- und Diskriminator-Netze.....	661
17.2.3	Definition der Trainingsdatenmenge .....	665
17.2.4	Trainieren des GAN-Modells.....	667
17.3	Verbesserung der Qualität synthetisierter Bilder durch Convolutional GAN und Wasserstein-GAN .....	676

17.3.1	Transponierte Faltung . . . . .	677
17.3.2	Batchnormierung . . . . .	678
17.3.3	Implementierung des Generators und des Diskriminators . . . . .	681
17.3.4	Maße für den Unterschied zwischen zwei Verteilungen . . . . .	688
17.3.5	Verwendung der EM-Distanz in der Praxis . . . . .	691
17.3.6	Strafterm . . . . .	692
17.3.7	Implementierung von WGAN-GP zum Trainieren des DCGAN-Modells . . . . .	693
17.3.8	Zusammenbrechen des Verfahrens . . . . .	697
17.4	Weitere GAN-Anwendungen . . . . .	699
17.5	Zusammenfassung . . . . .	700
<b>18</b>	<b>Entscheidungsfindung in komplexen Umgebungen per Reinforcement Learning . . . . .</b>	<b>701</b>
18.1	Einführung: Aus Erfahrung lernen . . . . .	702
18.1.1	Reinforcement Learning . . . . .	702
18.1.2	Definition der Agent-Umgebung-Schnittstelle für ein Reinforcement-Learning-System . . . . .	704
18.2	Theoretische Grundlagen des RLs . . . . .	705
18.2.1	Markov-Entscheidungsprozesse . . . . .	705
18.2.2	Mathematische Formulierung von Markov-Entscheidungsprozessen . . . . .	706
18.2.3	RL-Terminologie: Return, Policy und Wertfunktion . . . . .	710
18.2.4	Dynamische Programmierung und Bellman-Gleichung . . . . .	714
18.3	Reinforcement-Learning-Algorithmen . . . . .	715
18.3.1	Dynamische Programmierung . . . . .	715
18.3.2	Reinforcement Learning mit Monte-Carlo-Algorithmen . . . . .	718
18.3.3	Temporal Difference Learning . . . . .	720
18.4	Implementierung eines RL-Algorithmus . . . . .	723
18.4.1	OpenAI Gym . . . . .	723
18.4.2	Lösung der Grid-World-Aufgabe mit Q-Learning . . . . .	734
18.4.3	Ein Blick auf Deep Q-Learning . . . . .	739
18.5	Zusammenfassung und Schlusswort . . . . .	747
	<b>Stichwortverzeichnis . . . . .</b>	<b>751</b>

# Über die Autoren

**Sebastian Raschka** erlangte seinen Doktortitel an der Michigan State University. Er befasst sich vornehmlich mit Fragen der Berechnung biologischer Phänomene und des Machine Learnings. Im Sommer 2018 wurde er Assistant Professor für Statistik an der University of Wisconsin-Madison. Bei seiner Forschungsarbeit geht es insbesondere um die Entwicklung neuer Deep-Learning-Architekturen zum Lösen von Aufgaben im Fachgebiet Biometrie.

Er verfügt über jahrelange Erfahrung in der Python-Programmierung und leitete mehrere Seminare über praktische Data-Science-Anwendungen, Machine Learning und Deep Learning, unter anderem eine Einführung in Machine Learning auf der SciPy-Konferenz, der maßgeblichen Veranstaltung für wissenschaftliche Anwendungen in Python.

Er ist Autor des viel verkauften Buches *Python Machine Learning*, das 2016 mit dem Preis ACM Computing Reviews Best of ausgezeichnet und in viele Sprachen übersetzt wurde, unter anderem Deutsch, Koreanisch, Chinesisch, Japanisch, Russisch, Polnisch und Italienisch.

In seiner Freizeit leistet Sebastian aktiv Beiträge zu Open-Source-Projekten und die von ihm implementierten Verfahren werden erfolgreich in Mustererkennungswettbewerben wie z.B. Kaggle eingesetzt.

---

Ich möchte diese Gelegenheit nutzen, der großartigen Python-Community und den Entwicklern der Open-Source-Pakete meinen Dank auszusprechen, die mir dabei geholfen haben, die perfekte Umgebung für wissenschaftliche Forschung und Data Science einzurichten. Außerdem möchte ich meinen Eltern danken, die mich bei all meinen beruflichen Zielen, die ich so leidenschaftlich verfolgt habe, stets ermutigt und unterstützt haben.

Mein besonderer Dank gilt den Hauptentwicklern von scikit-learn und TensorFlow. Als jemand, der selbst aktiv an diesem Projekt beteiligt war, hatte ich das Vergnügen, mit tollen Leuten zusammenarbeiten zu dürfen, die sich nicht nur mit Machine Learning und Deep Learning auskennen, sondern auch hervorragende Programmierer sind.

---

**Vahid Mirjalili** erlangte seinen Doktortitel als Maschinenbauingenieur an der Michigan State University mit einer Arbeit über neue Verfahren für Computersimulationen molekularer Strukturen. Er interessiert sich leidenschaftlich für Machine Learning und trat dem iProBe-Lab der Michigan State University bei, wo er Anwendungen des Machine Learnings in verschiedenen Computer-Vision-Projekten (»maschinelles Sehen«) erforschte. Nach mehreren produktiven Jahren am iProBe-Lab und in der Forschung ist Vahid Mirjalili seit Kurzem beim Unternehmen 3M als Forscher tätig, wo er seine Kenntnisse einsetzen kann, um moderne Machine-Learning- und Deep-Learning-Verfahren auf Aufgabenstellungen aus der Praxis anzuwenden.

---

Ich möchte meiner Frau Tában Eslami danken, die mich während meiner Laufbahn stets unterstützt und ermutigt hat. Besonderer Dank gebührt meinen Mentoren Nikolai Priezjev, Michael Feig und Arun Ross, die mich während des Doktorats unterstützt haben, sowie meinen Professoren Vishnu Boddeti, Leslie Kuhn und Xiaoming Liu, die mich so vieles gelehrt haben und mich ermutigten, meiner Leidenschaft zu folgen.

---

# Über die Korrektoren

**Raghav Bali** ist als leitender Data Scientist bei einem der weltweit größten Unternehmen im Gesundheitswesen tätig. Er erforscht und entwickelt für im Gesundheits- oder Versicherungswesen tätige Unternehmen Lösungen, die auf Machine Learning, Deep Learning und der Verarbeitung natürlicher Sprache beruhen. Davor hat er sich bei Intel damit befasst, datengetriebene IT-Lösungen zu ermöglichen, die Deep Learning, die Verarbeitung natürlicher Sprache und klassische statistische Verfahren nutzen. Bei American Express war er auch im Finanzwesen tätig und hat digitale Lösungen für die Kundenbindung entwickelt.

Raghav Bali hat mehrere Bücher bei bedeutenden Verlagen veröffentlicht. Das letzte befasst sich mit den in der Erforschung des Transfer Learnings jüngst erzielten Fortschritten.

Raghav Bali verfügt über einen Master in Informationstechnologie des International Institute of Information Technology (Bangalore). Er liest gerne und ist ein Fotonnarr, wenn er nicht gerade damit beschäftigt ist, Aufgaben zu lösen.

**Motaz Saad** hat einen Doktor in Informatik der University of Lorraine. Er liebt Daten und mag es sehr, mit ihnen zu experimentieren. Er beschäftigt sich seit mehr als zehn Jahren mit der Verarbeitung natürlicher Sprache, Computerlinguistik, Data Science und Machine Learning. Derzeit ist er als Assistant Professor am Fachbereich Informationstechnologie der Islamischen Universität Gaza tätig.

# Über den Fachkorrektor der deutschen Ausgabe

Friedhelm Schwenker ist Privatdozent für Informatik (Fachgebiet: Machine Learning) an der Universität Ulm. Er hat im Bereich der Angewandten Mathematik promoviert und ist seit vielen Jahren im Bereich Machine Learning in Forschung und Lehre tätig. Seine Forschungsgebiete sind Pattern Recognition, Data Mining und Machine Learning mit Schwerpunkt Neuronale Netze. In jüngster Zeit befasst er sich auch mit Anwendungen des Machine Learning im Affective Computing. Er ist Editor von 19 Proceedingsbänden und Special Issues sowie Autor von 200+ Journal- und Konferenzartikeln.

# Einleitung

Aus den Nachrichten und den sozialen Medien ist Ihnen vermutlich bekannt, dass das Machine Learning zu einer der spannendsten Technologien der heutigen Zeit geworden ist. Große Unternehmen wie Google, Facebook, Apple, Amazon, IBM und viele andere investieren aus gutem Grund kräftig in die Erforschung des Machine Learnings und dessen Anwendung. Auch wenn man manchmal den Eindruck bekommt, dass »Machine Learning« als leeres Schlagwort gebraucht wird, handelt es sich doch zweifellos nicht um eine Modeerscheinung. Dieses spannende Fachgebiet eröffnet viele neue Möglichkeiten und ist im Alltag schon nicht mehr wegzudenken. Denken Sie an die virtuellen Assistenten von Smartphones, Produktempfehlungen für Kunden in Onlineshops, das Verhindern von Kreditkartenbetrug, Spamfilter in E-Mail-Programmen oder die Erkennung und Diagnose von Krankheitssymptomen – die Liste ließe sich beliebig lang fortsetzen.

## Einstieg in Machine Learning

Wenn Sie zu einem Praktiker des Machine Learnings und einem besseren Problemlöser werden möchten oder vielleicht sogar eine Laufbahn in der Erforschung des Machine Learnings anstreben, dann ist dies das richtige Buch für Sie. Für einen Neuling können die dem Machine Learning zugrunde liegenden theoretischen Konzepte zunächst einmal erdrückend wirken. In den vergangenen Jahren sind aber viele praxisorientierte Bücher mit leistungsfähigen Lernalgorithmen erschienen, die Ihnen den Start erleichtern.

## Theorie und Praxis

Die Verwendung praxisorientierter Codebeispiele dient einem wichtigen Zweck: Konkrete Beispiele verdeutlichen die allgemeinen Konzepte, indem das Erlernte unmittelbar in die Tat umgesetzt wird. Allerdings darf man dabei nicht vergessen, dass mit großer Macht auch immer große Verantwortung einhergeht! Neben der unmittelbaren Erfahrung, Machine Learning mithilfe der Programmiersprache Python und auf Python beruhenden Lernbibliotheken in die Tat umzusetzen, stellt das Buch auch die den Machine-Learning-Algorithmen zugrunde liegenden mathematischen Konzepte vor, die für den erfolgreichen Einsatz von Machine Learning unverzichtbar sind. Das Buch ist also kein rein praktisch orientiertes Werk, sondern ein Buch, das die erforderlichen Details der Konzepte des Machine Learnings

erörtert, die Funktionsweise von Lernalgorithmen und ihre Verwendung verständlich, aber dennoch informativ erklärt und – was noch wichtiger ist – das zeigt, wie man die häufigsten Fehler vermeidet.

## Warum Python?

Bevor wir uns eingehender mit Machine Learning befassen, müssen wir die wichtigste Frage beantworten: Warum Python? Die Antwort ist ganz einfach: Python ist leistungsfähig, aber dennoch sehr leicht erlernbar. Python ist auf dem Gebiet der Data Science zur verbreitetsten Programmiersprache geworden, weil sie es uns ermöglicht, die lästigen Aspekte des Programmierens zu vergessen, und eine Umgebung bereitstellt, in der wir unsere Ideen schnell umsetzen und Konzepte direkt zur Anwendung bringen können.

## Erkundung des Fachgebiets Machine Learning

Wenn Sie bei Google Scholar den Suchbegriff *machine learning* eingeben, erhalten Sie als Resultat eine riesige Zahl (ca. 3.250.000) von Treffern. Nun können wir in diesem Buch natürlich nicht sämtliche Einzelheiten der in den letzten 60 Jahren entwickelten Algorithmen und Anwendungen erörtern. Wir werden uns jedoch auf eine spannende Tour begeben, die alle wichtigen Themen und Konzepte umfasst, damit Sie eine gründliche Einführung erhalten. Sollte Ihr Wissensdurst auch nach der Lektüre noch nicht gestillt sein, steht Ihnen eine Vielzahl weiterer hilfreicher Ressourcen zur Verfügung, die Sie nutzen können, um die entscheidenden Fortschritte auf diesem Fachgebiet zu verfolgen.

Wir, die Autoren, können aus eigener Erfahrung sagen, dass wir durch die Beschäftigung mit dem Machine Learning zu besseren Wissenschaftlern, Denkern und Problemlösern geworden sind. In diesem Buch möchten wir unsere diesbezüglichen Erkenntnisse mit Ihnen teilen. Wissen wird durch Lernen erworben, das wiederum einen gewissen Eifer erfordert, und erst Übung macht den sprichwörtlichen Meister.

Der vor Ihnen liegende Weg ist manchmal nicht ganz einfach, und einige der Themenbereiche sind deutlich schwieriger als andere, aber wir hoffen dennoch, dass Sie die Gelegenheit nutzen und sich auf den Lohn der Mühe konzentrieren. Im weiteren Verlauf des Buches werden Sie Ihrem Repertoire eine ganze Reihe leistungsfähiger Techniken hinzufügen können, die dabei helfen, auch die schwierigsten Aufgaben auf datengesteuerte Weise zu bewältigen.

## An wen richtet sich das Buch?

Falls Sie sich schon ausführlich mit der Theorie des Machine Learnings beschäftigt haben, zeigt Ihnen dieses Buch, wie Sie Ihre Kenntnisse in die Praxis umsetzen können. Wenn Sie bereits entsprechende Techniken eingesetzt haben, aber

deren Funktionsweise besser verstehen möchten, kommen Sie hier ebenfalls auf Ihre Kosten.

Und wenn Ihnen das Thema Machine Learning noch völlig neu ist, haben Sie umso mehr Grund, sich zu freuen, denn ich kann Ihnen versprechen, dass dieses Verfahren Ihre Denkweise über Ihre in Zukunft zu lösenden Aufgaben verändern wird – und ich möchte Ihnen zeigen, wie Sie Problemstellungen in Angriff nehmen, indem Sie die den Daten innewohnende Kraft freisetzen. Wenn Sie herausfinden möchten, wie Sie Python verwenden können, um die entscheidenden Fragen zu Ihren Daten zu beantworten, greifen Sie einfach zu diesem Buch. Ob Sie völliger Neuling sind oder Ihre Kenntnisse der Data Science vertiefen möchten: Dieses Buch ist eine unentbehrliche Informationsquelle und unbedingt lesenswert.

## Zum Inhalt des Buches

*Kapitel 1, Wie Computer aus Daten lernen können*, führt Sie in die wichtigsten Teilbereiche des Machine Learnings ein, mit denen sich verschiedene Probleme in Angriff nehmen lassen. Darüber hinaus werden die grundlegenden Schritte beim Entwurf eines typischen Machine-Learning-Modells erörtert, auf die wir in den nachfolgenden Kapiteln zurückgreifen.

*Kapitel 2, Lernalgorithmen für die Klassifikation trainieren*, geht zurück zu den Anfängen des Machine Learnings und stellt binäre Perzeptron-Klassifizierer und adaptive lineare Neuronen vor. Dieses Kapitel ist eine behutsame Einführung in die Grundlagen der Klassifikation von Mustern und konzentriert sich auf das Zusammenspiel von Optimierungsalgorithmen und Machine Learning.

*Kapitel 3, Machine-Learning-Klassifikatoren mit scikit-learn verwenden*, beschreibt die wichtigsten Klassifikationsalgorithmen des Machine Learnings und stellt praktische Beispiele vor. Dabei kommt eine der beliebtesten und verständlichsten Open-Source-Bibliotheken für Machine Learning zum Einsatz: scikit-learn.

*Kapitel 4, Gut geeignete Trainingsdatenmengen: Datenvorverarbeitung*, erläutert die Handhabung der gängigsten Probleme unverarbeiteter Datenmengen, wie z.B. fehlende Daten. Außerdem werden verschiedene Ansätze zur Ermittlung der informativsten Merkmale einer Datenmenge vorgestellt. Des Weiteren erfahren Sie, wie sich Variablen unterschiedlichen Typs als geeignete Eingabe für Lernalgorithmen einsetzen lassen.

*Kapitel 5, Datenkomprimierung durch Dimensionsreduktion*, beschreibt ein wichtiges Verfahren zur Reduzierung der Merkmalsanzahl eines Datenbestands durch Aufteilung in kleinere Mengen unter Beibehaltung eines Großteils der nützlichsten und charakteristischsten Informationen. Hier wird der Standardansatz zur Dimensionsreduktion durch die Analyse der Hauptkomponenten erläutert und mit überwachten und nichtlinearen Transformationsverfahren verglichen.

*Kapitel 6, Bewährte Verfahren zur Modellbewertung und Hyperparameter-Optimierung*, erörtert die Einschätzung der Aussagekraft von Vorhersagemodellen. Darüber hinaus kommen verschiedene Bewertungskriterien der Modelle sowie Verfahren zur Feinabstimmung der Lernalgorithmen zur Sprache.

*Kapitel 7, Kombination verschiedener Modelle für das Ensemble Learning*, führt Sie in die verschiedenen Konzepte zur effektiven Kombination diverser Lernalgorithmen ein. Sie erfahren, wie Sie Ensembles einrichten, um die Schwächen einzelner Klassifizierer zu überwinden, was genauere und verlässlichere Vorhersagen liefert.

*Kapitel 8, Machine Learning zur Analyse von Stimmungslagen nutzen*, erläutert die grundlegenden Schritte zur Transformierung von Textdaten in eine für Lernalgorithmen sinnvolle Form, um so die Meinung von Menschen anhand der von ihnen verfassten Texte vorherzusagen.

*Kapitel 9, Einbettung eines Machine-Learning-Modells in eine Webanwendung*, führt vor, wie Sie das Lernmodell des vorangehenden Kapitels Schritt für Schritt in eine Webanwendung einbetten können.

*Kapitel 10, Vorhersage stetiger Zielvariablen durch Regressionsanalyse*, erörtert grundlegende Verfahren zur Modellierung linearer Beziehungen zwischen Zielvariablen und Regressanden, um auch stetige Werte vorhersagen zu können. Nach der Vorstellung der linearen Modelle kommen auch Polynom-Regression und baumbasierte Ansätze zur Sprache.

*Kapitel 11, Verwendung von Daten ohne Label: Clusteranalyse*, konzentriert sich auf einen anderen Teilbereich des Machine Learnings, nämlich auf das unüberwachte Lernen. Wir werden drei unterschiedlichen Familien von Clustering-Algorithmen zugehörige Verfahren anwenden, um Objektgruppen aufzuspüren, die einen gewissen Ähnlichkeitsgrad aufweisen.

*Kapitel 12, Implementierung eines künstlichen neuronalen Netzes*, erweitert das in Kapitel 2 vorgestellte Konzept der Gradient-basierten Optimierung, um leistungsfähige, mehrschichtige neuronale Netze in Python zu erstellen, die auf dem verbreiteten Backpropagation-Algorithmus beruhen.

*Kapitel 13, Parallelisierung des Trainings neuronaler Netze mit TensorFlow*, baut auf den in den vorausgehenden Kapiteln erworbenen Kenntnissen auf, um Ihnen einen praxisorientierten Leitfaden für ein effizienteres Training neuronaler Netze (NN) an die Hand zu geben. Der Schwerpunkt dieses Kapitels liegt dabei auf TensorFlow 2.0, einer quelloffenen Python-Bibliothek, die die Verwendung mehrerer Kerne moderner Grafikprozessoren (GPUs) ermöglicht und die es gestattet, mithilfe von Bausteinen der benutzerfreundlichen Keras-API tiefe NN zu erstellen.

*Kapitel 14, Die Funktionsweise von TensorFlow im Detail*, stellt die fortgeschritteneren Konzepte und Funktionalitäten von TensorFlow 2.0 vor. TensorFlow ist eine äußerst umfassende und ausgeklügelte Bibliothek. Dieses Kapitel betrachtet die grundle-

genden Konzepte des Kompilierens von Code zu statischen Graphen zwecks schnellerer Berechnung und der Definition trainierbarer Modellparameter. Darüber hinaus kommen Themen wie das Trainieren tiefer NN mithilfe von TensorFlows Keras-API sowie die vorgefertigten Schätzer zur Sprache.

*Kapitel 15, Bildklassifikation mit Deep Convolutional Neural Networks*, stellt neuronale Netzarchitekturen vor, die bei maschinellem Sehen und der Bilderkennung aufgrund der gegenüber klassischen Ansätzen überlegenen Leistung zu einem neuen Standard geworden sind, nämlich konvolutionale neuronale Netze (*Convolutional Neural Networks, CNN*). Dieses Kapitel zeigt, wie man Faltungsschichten als Merkmalsextraktoren zur Klassifikation von Bildern verwenden kann.

*Kapitel 16, Modellierung sequenzieller Daten durch rekurrente neuronale Netze*, stellt eine weitere verbreitete neuronale Netzarchitektur für Deep Learning vor, die besonders gut für die Verarbeitung von Text, anderen sequenziellen Daten und Zeitreihen geeignet ist. In diesem Kapitel werden wir verschiedene rekurrente neuronale Netzarchitekturen auf Textdaten anwenden. Als Aufwärmübung betrachten wir zunächst eine Stimmungsanalyse von Filmbewertungen. Anschließend wird erörtert, wie ein rekurrentes NN anhand der Informationen aus Büchern völlig neue Texte erzeugen kann.

*Kapitel 17, Synthetisieren neuer Daten mit Generative Adversarial Networks*, stellt eine verbreitete Form eines NN vor, das dazu verwendet werden kann, neue, realistisch wirkende Bilder zu erzeugen. Das Kapitel enthält zunächst eine kurze Einführung in Autoencoder, einen bestimmten Typ eines NN, das zur Datenkomprimierung verwendet werden kann. Anschließend wird erläutert, wie man den Decoder-Teil eines Autoencoders mit einem zweiten NN kombiniert, das zwischen echten und erzeugten Bildern unterscheiden kann. Indem Sie zwei NN miteinander wetteifern lassen, werden Sie ein GAN (Generative Adversarial Network) implementieren, das neue Bilder von scheinbar handgeschriebenen Ziffern erzeugen kann. Nachdem die grundlegenden Konzepte von GAN vorgestellt wurden, endet das Kapitel mit einer Beschreibung von Verfahren, die das Training von GAN stabilisieren können, wie beispielsweise die Verwendung der Wasserstein-Metrik als Distanzmaß.

*Kapitel 18, Entscheidungsfindung in komplexen Umgebungen per Reinforcement Learning*, beschreibt ein Teilgebiet des Machine Learnings, das typischerweise beim Trainieren von Robotern und anderen autonomen System zum Einsatz kommt. Das Kapitel enthält zunächst eine Einführung in Reinforcement Learning (RL), damit Ihnen die Interaktionen von Agenten und Umgebungen, Belohnungssysteme und das Konzept, aus Erfahrungen zu lernen, vertraut sind. Das Kapitel stellt die beiden Hauptkategorien des RL vor, nämlich modellbasierte und modellfreie RL-Systeme. Nachdem Sie grundlegende Ansätze für Algorithmen kennengelernt haben, wie Monte-Carlo-Verfahren und Temporal-Difference-Algorithmen, werden Sie einen Agenten implementieren und trainieren, der sich mithilfe eines Q-Learning-Algo-

rithmus in einer Grid-World-Umgebung bewegt. Abschließend wird ein Deep-Q-Learning-Algorithmus vorgestellt, der eine Variante des Q-Learnings unter Verwendung tiefer NN ist.

## Was Sie benötigen

Zum Ausführen der Codebeispiele ist die Python-Version 3.7.0 oder neuer auf macOS, Linux oder Microsoft Windows erforderlich. Wir werden häufig von Python-Bibliotheken Gebrauch machen, die für wissenschaftliche Berechnungen unverzichtbar sind, z.B. von SciPy, NumPy, scikit-learn, Matplotlib und pandas.

Im ersten Kapitel finden Sie Hinweise und Tipps zur Einrichtung Ihrer Python-Umgebung und dieser elementaren Bibliotheken. In den verschiedenen Kapiteln werden wir dann der Python-Umgebung weitere Bibliotheken hinzufügen: die NLTK-Bibliothek für die Verarbeitung natürlicher Sprache (Kapitel 8), das Web-Framework Flask (Kapitel 9) und schließlich TensorFlow, um neuronale Netze effizient auf GPUs zu trainieren (Kapitel 13 bis 18).

## Codebeispiele herunterladen

Die Codebeispiele können Sie auf GitHub unter <https://github.com/rasbt/python-machine-learningbook-3rd-edition> oder über die Verlagsseite <http://www.mitp.de/0213> herunterladen. Dort sind auch farbige Abbildungen zu finden.

## Konventionen im Buch

In diesem Buch werden verschiedene Textarten verwendet, um zwischen Informationen unterschiedlicher Art zu unterscheiden. Nachstehend finden Sie einige Beispiele und deren Bedeutungen.

Schlüsselwörter oder Code werden im Fließtext wie folgt dargestellt:

»Ein bereits installiertes Paket kann mit der Option `--upgrade` aktualisiert werden.«

Codeblöcke sehen so aus:

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> y = df.iloc[0:100, 4].values
>>> y = np.where(y == 'Iris-setosa', -1, 1)
>>> X = df.iloc[0:100, [0, 2]].values
>>> plt.scatter(X[:50, 0], X[:50, 1],
...                 color='red', marker='x', label='setosa')
```

```
>>> plt.scatter(X[50:100, 0], X[50:100, 1],  
...                 color='blue', marker='o', label='versicolor')  
>>> plt.xlabel('Länge des Kelchblatts')  
>>> plt.ylabel('Länge des Blütenblatts')  
>>> plt.legend(loc='upper left')  
>>> plt.show()
```

Benutzereingaben oder Ausgaben auf der Kommandozeile werden in nicht proportionaler Schrift gedruckt:

```
> dot -Tpng tree.dot -o tree.png
```

*Neue Ausdrücke* und *wichtige Begriffe* werden kursiv gedruckt. Auf dem Bildschirm auswählbare oder anklickbare Bezeichnungen, wie z.B. Menüpunkte oder Schaltflächen, werden in der Schriftart KAPITÄLCHEN gedruckt: »Nach einem Klick auf die Schaltfläche ABBRECHEN in der unteren rechten Ecke wird der Vorgang abgebrochen.«

### Hinweis

Warnungen oder Hinweise erscheinen in einem Kasten wie diesem.

### Tipp

Und so werden Tipps und Tricks dargestellt.

# Wie Computer aus Daten lernen können

Unserer Ansicht nach ist *das Machine Learning (maschinelles Lernen)*, die Anwendung und Wissenschaft von Algorithmen, die den Sinn von Daten erkennen können, das spannendste Forschungsfeld der Informatik! Wir leben in einem Zeitalter, in dem Daten im Überfluss vorhanden sind – und mit den selbstlernenden Algorithmen des Machine Learnings können wir diese Daten in Wissen verwandeln. Dank der vielen in den letzten Jahren entwickelten Open-Source-Bibliotheken ist jetzt der richtige Zeitpunkt gekommen, um sich eingehend mit dem Thema Machine Learning zu befassen und zu erfahren, wie leistungsfähige Algorithmen dafür eingesetzt werden können, Muster in den Daten zu erkennen und Vorhersagen über zukünftige Ereignisse zu treffen.

In diesem Kapitel werden wir die grundlegenden Konzepte und verschiedene Arten des Machine Learnings erörtern. Mit einer Einführung in die relevante Terminologie schaffen wir die Grundlage dafür, Verfahren des Machine Learnings erfolgreich zum Lösen von Aufgaben in der Praxis einzusetzen.

Dieses Kapitel hat folgende Themen zum Inhalt:

- Allgemeine Konzepte des Machine Learnings
- Die drei Arten des Machine Learnings und grundlegende Begriffe
- Die Bausteine des erfolgreichen Designs von Lernsystemen
- Installation von Python und Einrichtung einer für die Analyse von Daten und Machine Learning geeigneten Umgebung

## 1.1 Intelligente Maschinen, die Daten in Wissen verwandeln

In diesem Zeitalter der modernen Technologie steht eine Ressource im Überfluss zur Verfügung: große Mengen von strukturierten und unstrukturierten Daten. In der zweiten Hälfte des 20. Jahrhunderts hat sich das Machine Learning als eine Teildisziplin der *Artificial Intelligence* (künstliche Intelligenz, KI) herausgebildet, bei der es um die Entwicklung selbstlernender Algorithmen geht, die Erkenntnisse aus Daten extrahieren, um bestimmte Vorhersagen treffen zu können. Das Erfordernis menschlichen Eingreifens zur manuellen Ableitung von Regeln und

der Entwicklung von Modellen anhand der Analyse großer Datenmengen erübrigt sich damit mehr und mehr, denn das Machine-Learning-Verfahren bietet eine effiziente Alternative zur Erfassung des in den Daten enthaltenen Wissens – die zudem die auf diesen Daten basierende Entscheidungsfindung sowie die Aussagekraft von Vorhersagemodellen zusehends verbessert.

Dieses Verfahren wird nicht nur in der Forschung immer wichtiger, es spielt auch im Alltag eine zunehmend größere Rolle: Dank des Machine Learnings erfreuen wir uns stabiler E-Mail-Spamfilter, praktischer Text- und Spracherkennungssoftware, verlässlicher Suchmaschinen, kaum zu schlagender Schachcomputer und hoffentlich bald auch sicherer selbstfahrender Autos. Auch bei medizinischen Anwendungen hat es bemerkenswerte Fortschritte gegeben. So haben Forscher beispielsweise demonstriert, dass Deep-Learning-Modelle Hautkrebs fast so gut wie Menschen erkennen können (<https://www.nature.com/articles/nature21056>). Bei DeepMind haben Forscher kürzlich einen weiteren Meilenstein erreicht. Sie konnten mithilfe von Deep Learning die dreidimensionale Struktur von Proteinen vorhersagen und haben dabei erstmals bessere Ergebnisse als mit physikalischen Ansätzen erzielt (<https://deepmind.com/blog/alphafold/>).

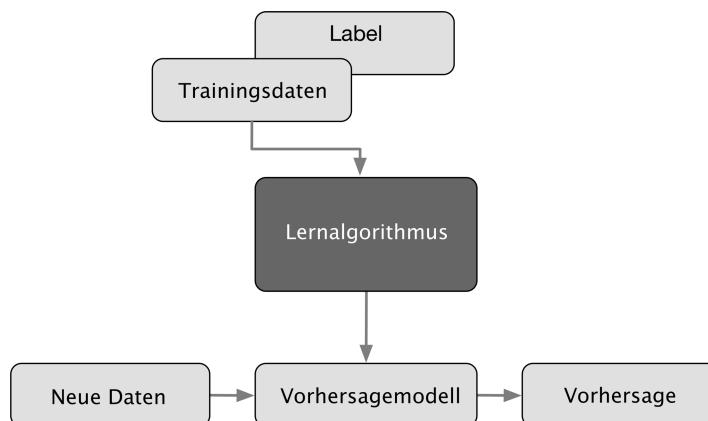
## 1.2 Die drei Arten des Machine Learnings

In diesem Abschnitt werden wir die drei verschiedenen Gattungen des Machine Learnings betrachten: *überwachtes Lernen*, *unüberwachtes Lernen* und *Reinforcement Learning*. Sie werden erfahren, welche grundlegenden Unterschiede es zwischen diesen drei Varianten gibt und anhand von Beispielen allmählich ein Gespür dafür entwickeln, auf welche praktischen Aufgabenstellungen sie sich anwenden lassen:

Überwachtes Lernen	<ul style="list-style-type: none"><li>› Daten mit Label</li><li>› Direktes Feedback</li><li>› Ergebnis/Zukunft vorhersagen</li></ul>
Unüberwachtes Lernen	<ul style="list-style-type: none"><li>› Keine Kennzeichnung/Ziele</li><li>› Kein Feedback</li><li>› Verborgene Strukturen in den Daten finden</li></ul>
Reinforcement Learning	<ul style="list-style-type: none"><li>› Entscheidungsvorgang</li><li>› Belohnungssystem</li><li>› Aktionen erlernen</li></ul>

### 1.2.1 Mit überwachtem Lernen Vorhersagen treffen

Das Hauptziel des überwachten Lernens ist, ein Modell anhand mit Labels gekennzeichneter *Trainingsdaten* zu erlernen, um so Voraussagen über unbekannte oder zukünftige Daten treffen zu können. Der Begriff »überwacht« bezieht sich hier auf Trainingsdaten (Eingabedaten), die bereits mit den bekannten erwünschten Ausgabewerten (Bezeichnungen/Labels) gekennzeichnet sind.



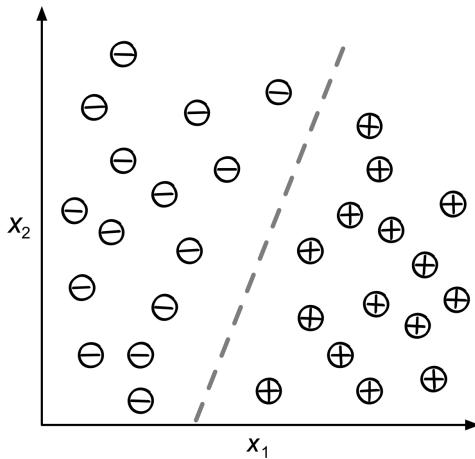
Betrachten wir als Beispiel das Filtern von E-Mail-Spam. Wir können einen überwachten Lernalgorithmus mit einer Sammlung von als Spam oder Nicht-Spam gekennzeichneten E-Mails »trainieren«, um dann vorherzusagen, zu welcher dieser Klassen eine neue E-Mail gehört. Eine solche Einteilung in bestimmte Klassen wird als *Klassifikation* bezeichnet. Eine weitere Unterkategorie des überwachten Lernens ist die *Regression*, bei der die Ausgabewerte im Gegensatz zur Klassifikation stetig sind.

#### Klassifikation: Vorhersage der Klassenbezeichnungen

Die Klassifikation ist eine Unterkategorie des überwachten Lernens, die es zum Ziel hat, anhand vorhergehender Beobachtungen die kategorialen Klassen neuer Instanzen vorherzusagen. Die Bezeichnungen dieser Klassen sind eindeutige, ungeordnete Werte, die als Gruppenzugehörigkeit der Instanzen aufgefasst werden können. Die soeben erwähnte E-Mail-Spamerkennung stellt ein typisches Beispiel für eine *binäre Klassifikation* dar, denn der Algorithmus lernt Regeln, um zwischen zwei möglichen Klassen zu unterscheiden: Spam oder Nicht-Spam.

Die folgende Abbildung illustriert das Konzept einer binären Klassifikation, die mit 30 Beispielen trainiert wird, von denen 15 als *negative Klasse* (Minuszeichen) und weitere 15 als *positive Klasse* (Pluszeichen) gekennzeichnet sind. Die Datenmenge ist in diesem Szenario zweidimensional: Jedem Beispiel sind die beiden

Werte  $x_1$  und  $x_2$  zugeordnet. Nun können wir dem überwachten Lernalgorithmus eine Regel beibringen: Die durch eine gestrichelte Linie dargestellte Grenze trennt die beiden Klassen voneinander und ermöglicht es, neue Daten anhand der Werte von  $x_1$  und  $x_2$  einer der beiden Klassen zuzuordnen.



Die Anzahl der Klassenbezeichnungen muss allerdings nicht auf zwei beschränkt sein. Das von einem überwachten Lernalgorithmus erlernte Vorhersagemodell kann einer neuen, noch nicht mit Label gekennzeichneten Instanz jede Bezeichnung zuordnen, die in den Trainingsdaten vorkommt.

Ein typisches Beispiel für solch eine *Multiklassen-Klassifikation* ist die Handschrifterkennung. Hier könnten wir eine Trainingsdatenmenge zusammenstellen, die aus mehreren handgeschriebenen Beispielen aller Buchstaben des Alphabets besteht. Die Buchstaben (»A«, »B«, »C« usw.) repräsentieren die verschiedenen Kategorien oder Klassenbezeichnungen, die wir vorhersagen möchten. Wenn dann ein Anwender über ein Eingabegerät einen neuen Buchstaben angibt, wäre unser Vorhersagemodell in der Lage, diesen mit einer gewissen Zuverlässigkeit zu erkennen. Das System wäre allerdings nicht imstande, irgendeine der Ziffern von null bis neun zu erkennen, sofern diese nicht ebenfalls Bestandteil der Trainingsdaten waren.

## Regression: Vorhersage stetiger Ergebnisse

Im vorangegangenen Abschnitt haben wir festgestellt, dass es die Aufgabe einer Klassifikation ist, Instanzen kategoriale, ungeordnete Klassenbezeichnungen zuzuordnen. Ein zweiter Typ des überwachten Lernens ist die Vorhersage stetiger Ergebnisse, die auch als *Regressionsanalyse* bezeichnet wird. Hierbei sind verschiedene unabhängige oder *erklärende* Variablen sowie eine stetige Zielvariable (Ergebnis) vorgegeben und wir versuchen, eine Beziehung zwischen diesen Variablen zu finden, um Ergebnisse vorhersagen zu können.

Beachten Sie hier, dass die erklärenden Variablen beim Machine Learning oft als »Merkmale« oder »Features« und die Ergebnisse als »Zielvariablen« bezeichnet werden. Wir werden diese Begriffe ebenfalls verwenden.

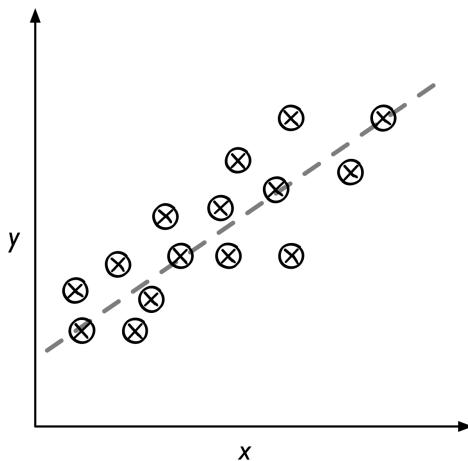
Nehmen wir beispielsweise an, dass wir die von Schülern bei einer Matheprüfung erreichten Punktzahlen prognostizieren möchten. Sofern es einen Zusammenhang zwischen der mit dem Üben für die Prüfung verbrachten Zeit und den erzielten Punktzahlen gibt, könnten wir daraus Trainingsdaten für ein Modell herleiten, das anhand der aufgewendeten Übungszeit die Punktzahlen von Schülern voraussagt, die die Prüfung in Zukunft ebenfalls abzulegen beabsichtigen.

### Tipp: Regression zur Mitte

Der Begriff *Regression* wurde schon 1886 von Francis Galton in einem Artikel mit dem Titel *Regression Towards Mediocrity in Hereditary Stature* geprägt. Galton beschrieb darin das Phänomen, dass sich bei der Bevölkerung die mittlere Abweichung von der durchschnittlichen Körpergröße im Laufe der Zeit nicht vergrößert.

Er beobachtete, dass die Körpergröße der Eltern nicht an die Kinder vererbt wird, vielmehr nähert sich die Größe der Kinder dem Durchschnittswert an.

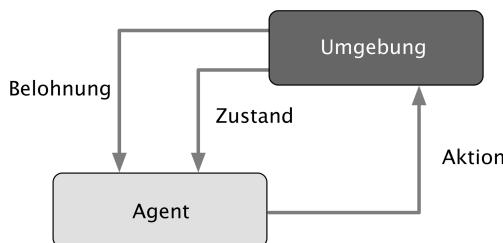
Die folgende Abbildung illustriert das Konzept der *linearen Regression*. Bei vorgegebener unabhängiger Variablen  $x$  und abhängiger Variablen  $y$  passen wir eine Gerade so an die Daten an, dass ein Maß für den Abstand der Geraden von den Beispielwerten (üblicherweise der Mittelwert der quadrierten Differenzen) minimal wird. Nun können wir den aus den Daten ermittelten Schnittpunkt mit der  $y$ -Achse sowie die Steigung der Geraden verwenden, um das Ergebnis für neue Werte vorherzusagen.



## 1.2.2 Interaktive Aufgaben durch Reinforcement Learning lösen

Die dritte Variante des Machine Learnings ist das Reinforcement Learning. Hier besteht die Zielsetzung darin, ein System zu entwickeln (den *Agenten*), das seine Leistung durch Interaktionen mit seiner *Umgebung* verbessert. Zu den Informationen über den aktuellen Zustand der Umgebung gehört typischerweise ein sogenanntes *Belohnungssignal*, daher ist das Reinforcement Learning in gewisser Weise mit dem überwachten Lernen verwandt. Allerdings handelt es sich bei diesem Feedback nicht um die korrekte Klassenbezeichnung oder den richtigen Wert, sondern um eine Bewertung dafür, wie gut die Aktion war, dies wird durch eine *Belohnungsfunktion* festgelegt. Der Agent kann so über Interaktionen mit seiner Umgebung durch Reinforcement Learning erkennen, welche Aktionen besonders gut belohnt werden. Das kann durch schlichtes Ausprobieren (Versuch und Irrtum) oder durch bewusste Planung geschehen.

Ein schönes Beispiel für Reinforcement Learning ist ein Schachcomputer. Hier bewertet der Agent nach einer Reihe von Zügen die Stellung auf dem Schachbrett (die Umgebung), und die Belohnung kann am Ende des Spiels als *Sieg* oder *Niederlage* definiert werden.



Es gibt eine Vielzahl verschiedener Unterarten des Reinforcement Learnings. Im Allgemeinen versucht der Agent jedoch, die Belohnung durch eine Reihe von Interaktionen mit der Umgebung zu maximieren. Jedem Zustand kann eine positive (oder negative) Belohnung zugeordnet werden, und diese Belohnung kann dadurch definiert werden, dass ein Gesamtziel erreicht wird, wie z.B. das Gewinnen oder das Verlieren einer Schachpartie. Beim Schachspiel kann etwa das Ergebnis eines jeden Spielzugs als ein anderer Zustand der Umgebung aufgefasst werden.

Um beim Schach zu bleiben: Stellen Sie sich das Erreichen bestimmter Stellungen auf dem Schachbrett als positives Ereignis vor, das es wahrscheinlicher macht, das Spiel zu gewinnen – beispielsweise das Schlagen einer gegnerischen Spielfigur oder das Bedrohen der Dame. Andere Stellungen wiederum werden als negativ erachtet, beispielsweise wenn eine der eigenen Spielfiguren beim nächsten Zug geschlagen werden kann. Nun erfolgt die Belohnung (positive beim Gewinnen und negative beim Verlieren) beim Schach natürlich erst am Ende des Spiels. Darüber hinaus hängt die Belohnung auch davon ab, wie der Gegner spielt. Er könnte beispielsweise die Dame opfern, das Spiel aber trotzdem gewinnen. Das Reinforce-

ment Learning versucht, eine Reihe von Aktionen zu erlernen, die die Belohnung insgesamt maximieren – entweder durch eine sofortige Belohnung nach einem Zug, oder aber durch eine *verzögerte* Belohnung.

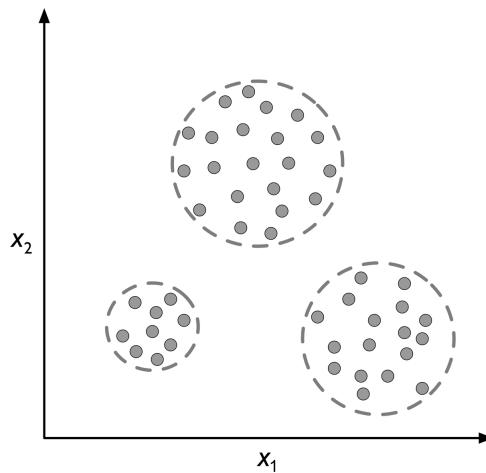
### 1.2.3 Durch unüberwachtes Lernen verborgene Strukturen erkennen

Beim überwachten Lernen ist die richtige Antwort beim Trainieren des Modells bereits im Vorhinein bekannt, und beim Reinforcement Learning definieren wir eine Bewertung oder *Belohnung* für bestimmte Aktionen des Agenten. Beim unüberwachten Lernen hingegen haben wir es mit nicht mit Label gekennzeichneten Daten oder mit Daten unbekannter Struktur zu tun. Durch die beim unüberwachten Lernen eingesetzten Verfahren sind wir in der Lage, die Struktur der Daten zu erkunden, um sinnvolle Informationen daraus zu extrahieren, ohne dass es Hinweise auf eine Zielvariable oder eine Belohnungsfunktion gibt.

#### Bestimmung von Untergruppen durch Clustering

*Clustering* ist ein exploratives Datenanalyseverfahren, das es uns gestattet, Informationen in sinnvolle Untergruppen (*Cluster*) aufzuteilen, ohne vorherige Kenntnisse über die Gruppenzugehörigkeit dieser Informationen zu besitzen. Jeder bei der Analyse auftretende Cluster definiert eine Gruppe von Objekten, die bestimmte Eigenschaften gemeinsam haben, sich aber von Objekten in anderen Gruppen hinreichend unterscheiden. Deshalb wird das Clustering manchmal auch als *unüberwachte Klassifikation* bezeichnet. Es ist ausgezeichnet geeignet, um Informationen zu strukturieren und sinnvolle Beziehungen zwischen den Daten abzuleiten. Beispielsweise ermöglicht es Marketingfachleuten, Kunden anhand ihrer Interessen in Gruppen einzuführen, um gezielte Kampagnen zu entwickeln.

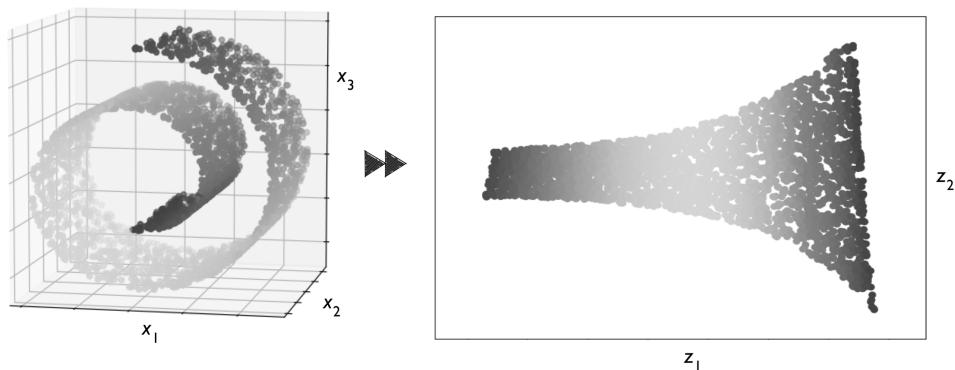
Die folgende Abbildung illustriert, wie man das Clustering-Verfahren zur Organisation nicht mit Label gekennzeichneter Daten in drei verschiedene Gruppen verwenden kann, die jeweils ähnliche Merkmale  $x_1$  und  $x_2$  aufweisen.



## Datenkomprimierung durch Dimensionsreduktion

Die *Dimensionsreduktion* ist eine weitere Teildisziplin des unüberwachten Lernens. Wir haben es des Öfteren mit Daten hoher Dimensionalität zu tun (jede Beobachtung besteht aus einer Vielzahl von Messwerten), was aufgrund der für die Lernalgorithmen geltenden Beschränkungen von Speicherplatz und Rechenleistung eine Herausforderung darstellen kann. Bei der Vorverarbeitung von Merkmalen wird häufig eine unüberwachte Dimensionsreduktion eingesetzt, um die Daten von sogenanntem »Rauschen« zu befreien. Dies kann allerdings zu einer Abschwächung der Aussagekraft bestimmter Vorhersagealgorithmen führen. Die Daten werden in kleinere Unterräume geringerer Dimensionalität aufgeteilt, wobei der Großteil der relevanten Informationen erhalten bleibt.

In manchen Fällen ist die Dimensionsreduktion auch für die Visualisierung der Daten nützlich. Beispielsweise können hochdimensionale Merkmalsmengen auf ein-, zwei- oder dreidimensionale Merkmalsräume projiziert werden, um sie als 3-D- oder 2-D-Streudiagramme bzw. -Histogramme darzustellen. Die Abbildung zeigt ein Beispiel, in dem eine nichtlineare Dimensionsreduktion auf eine 3-D-Punktmenge in Form einer Biskuitrolle angewendet wurde, um sie in einen zweidimensionalen Merkmalsraum zu transformieren.



## 1.3 Grundlegende Terminologie und Notation

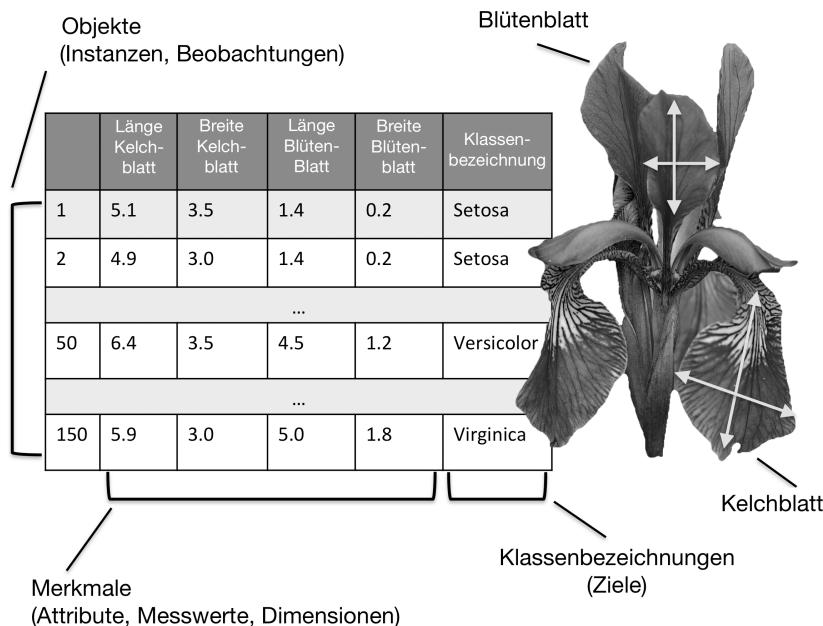
Nachdem wir nun die drei Arten des Machine Learnings – überwachtes und unüberwachtes Lernen sowie Reinforcement Learning – erörtert haben, werden wir als Nächstes die grundlegenden Begriffe klären, die in den folgenden Kapiteln Verwendung finden. Der folgende Abschnitt erläutert die Begriffe, die für die Beschreibung der verschiedenen Aspekte einer Datenmenge verwendet werden, sowie die mathematische Notation, die zur präzisen Beschreibung zum Einsatz kommt.

Da Machine Learning ein so umfassendes und interdisziplinäres Fachgebiet ist, werden Sie früher oder später mit Sicherheit vielen verschiedenen Begriffen begegnen, die ein und dasselbe Konzept beschreiben. Im zweiten der nachfolgenden Abschnitte sind die am häufigsten verwendeten Begriffe aufgeführt, die sich in der

Literatur zum Thema Machine Learning finden. Sie erweisen sich bei der weiteren Lektüre vielleicht als nützlich.

### 1.3.1 Im Buch verwendete Notation und Konventionen

Die folgende Abbildung zeigt einen Auszug der *Iris-Datensammlung*, einem klassischen Beispiel für den Bereich des Machine Learnings. Dabei handelt es sich um Messdaten von 150 Schwertlilien dreier verschiedener Arten: *Iris setosa*, *Iris versicolor* und *Iris virginica*. Jedes der Blumenexemplare wird in dieser Datensammlung durch eine Zeile repräsentiert. In den einzelnen Spalten stehen die in Zentimetern angegebenen Messdaten, die wir auch als *Merkmale* der Datenmenge bezeichnen.



Um die Notation und Implementierung einfach, aber dennoch effizient zu halten, nutzen wir die Grundlagen der *linearen Algebra*. In den nachfolgenden Kapiteln verwenden wir die Matrizen- und Vektornotation zur Beschreibung der Daten. Wir folgen der üblichen Konvention, dass jedes Objekt durch eine Zeile in der Merkmalsmatrix  $X$  repräsentiert und jedes Merkmal als eigene Spalte gespeichert wird.

Die Iris-Datensammlung besteht aus 150 Datensätzen mit jeweils vier Merkmalen und kann somit als  $150 \times 4$ -Matrix  $X \in \mathbb{R}^{150 \times 4}$  geschrieben werden:

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

Ab jetzt verwenden wir ein hochgestelltes  $i$  und ein tiefgestelltes  $j$ , um auf das  $i$ -te Trainingsobjekt bzw. die  $j$ -te Dimension der Trainingsdatenmenge zu verweisen.

Wir notieren Vektoren ( $\mathbf{x} \in \mathbb{R}^{n \times 1}$ ) als fett gedruckte Kleinbuchstaben und Matrizen ( $\mathbf{X} \in \mathbb{R}^{n \times m}$ ) als fett gedruckte Großbuchstaben. Um auf einzelne Elemente eines Vektors oder einer Matrix zu verweisen, werden kursive Buchstaben benutzt ( $x^{(n)}$  bzw.  $x_{(m)}^{(n)}$ ).

Beispielsweise verweist  $x_1^{150}$  auf die erste Dimension des Blumenexemplars 150, die Länge des Kelchblatts. Jede Zeile der Merkmalsmatrix repräsentiert ein Blumenexemplar und kann als vierdimensionaler Zeilenvektor  $\mathbf{x}^{(i)} \in \mathbb{R}^{1 \times 4}$  geschrieben werden, z.B.:

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$$

Jede Merkmalsdimension ist ein 150-dimensional Spaltenvektor  $\mathbf{x}_j \in \mathbb{R}^{150 \times 1}$ :

$$\mathbf{x}_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{bmatrix}$$

Die Zielvariablen (hier die Klassenbezeichnungen) werden ebenfalls als 150-dimensionale Spaltenvektoren notiert:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(150)} \end{bmatrix} \quad (y \in \{\text{Setosa, Versicolor, Virginica}\})$$

### 1.3.2 Terminologie

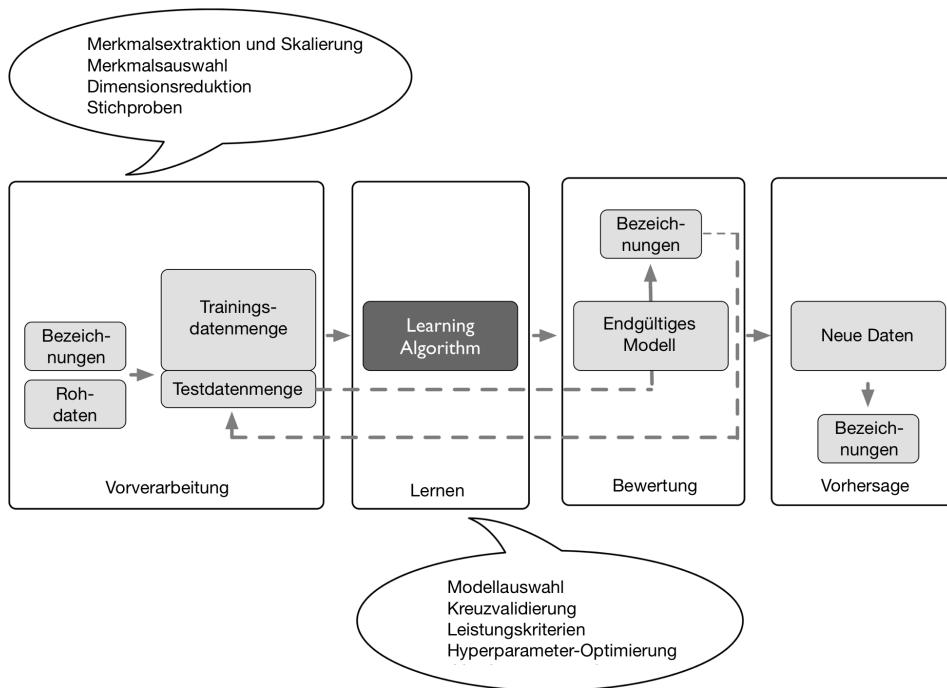
Wie erwähnt ist Machine Learning ein weites Feld und ausgeprägt interdisziplinär, denn es bringt Forscher aus den verschiedensten Fachgebieten zusammen. Viele Konzepte und Begriffe wurden neu entdeckt oder umdefiniert oder sind Ihnen vielleicht schon unter einem anderen Namen bekannt. In der folgenden Liste finden Sie eine Auswahl gebräuchlicher Begriffe und ihrer Synonyme, die sich bei der Lektüre dieses Buches oder anderer Bücher über Machine Learning vielleicht als nützlich erweisen.

- **Trainingsbeispiel:** Eine Zeile in einer Tabelle, die eine Datenmenge repräsentiert. Der Begriff ist gleichbedeutend mit Beobachtung, Datensatz, Instanz oder Stichprobe (mit einer Stichprobe sind für gewöhnlich mehrere Trainingsbeispiele gemeint).

- Training: Anpassung des Modells; bei parametrischen Modellen entspricht das einer Parameterschätzung.
- Merkmal, kurz  $x$ : Eine Spalte in einer Datentabelle oder einer Datenmatrix. Synonyme: Feature, unabhängige Variable, Eingabe, Attribut oder Kovariate.
- Ziel, kurz  $y$ : Synonyme: Ergebnis, Ausgabe, Antwort-/AusgabevARIABLE, abhängige Variable oder (Klassen-)Label oder (Klassen-)Bezeichnung
- Verlustfunktion: Kurz Verlust, wird manchmal als Synonym für Kostenfunktion bzw. Straffunktion gebraucht. Mitunter wird sie auch als Error- oder Fehlerfunktion bezeichnet (nicht zu verwechseln mit der gaußschen Fehlerfunktion). Der Begriff »Verlust« bezieht sich auf den Verlust, der für einen einzelnen Datenpunkt ermittelt wird. Die Kosten hingegen sind ein Maß für den Verlust der gesamten Datenmenge (gemittelt oder summiert).

## 1.4 Entwicklung eines Systems für das Machine Learning

In den vorangegangenen Abschnitten haben wir die grundlegenden Konzepte des Machine Learnings und die drei verschiedenen Arten des Lernens erörtert. In diesem Abschnitt befassen wir uns mit weiteren wichtigen Bestandteilen eines Systems für dieses Verfahren, die den Lernalgorithmus begleiten. Das folgende Diagramm zeigt den typischen Ablauf, der beim Machine Learning in Vorhersagemodellen zum Einsatz kommt, die wir in den folgenden Abschnitten betrachten werden.



### 1.4.1 Vorverarbeitung: Daten in Form bringen

Rohdaten liegen nur selten in einer für die optimale Leistung eines Lernalgorithmus erforderlichen Form vor, deshalb ist die **Vorverarbeitung** der Daten bei jedem Lernalgorithmus von entscheidender Bedeutung.

Im Fall der Iris-Datensammlung aus dem vorangegangenen Abschnitt könnten die Rohdaten beispielsweise als eine Reihe von Fotos der Blumenexemplare vorliegen, denen wir sinnvolle Merkmale entnehmen möchten. Das könnten etwa Grundfarbe und Tönung sowie Höhe, Länge und Breite der Pflanzen sein.

Bei vielen Lernalgorithmen ist es außerdem erforderlich, dass die ausgewählten Merkmale irgendwie normiert sind (hier müssten die Pflanzen im selben Maßstab dargestellt sein), um ein optimales Ergebnis zu erzielen. Dies wird oftmals dadurch erreicht, dass die ausgewählten Merkmale auf ein Intervall  $[0, 1]$  oder eine Standardnormalverteilung (Mittelwert 0 und Standardabweichung 1) abgebildet werden, wie Sie in den nachfolgenden Kapiteln noch sehen werden.

Manche der ausgewählten Merkmale könnten hochgradig korreliert und daher in gewissem Maße redundant sein. In diesen Fällen sind Verfahren zur Dimensionsreduktion nützlich, um die Merkmale auf einen Merkmalsraum geringer Dimensionalität abzubilden. Die Dimensionsreduktion des Merkmalraums hat die Vorteile, dass weniger Speicherplatz benötigt wird und der Lernalgorithmus erheblich schneller arbeitet. In manchen Fällen kann eine Dimensionsreduktion auch die Vorhersagekraft eines Modells verbessern, nämlich wenn die Datenmenge eine große Anzahl irrelevanter Merkmale (Rauschen) aufweist, das heißt, dass sie ein niedriges Signal-zu-Rausch-Verhältnis besitzt.

Um festzustellen, ob ein Lernalgorithmus nicht nur die Trainingsdaten ordentlich verarbeitet, sondern auch mit neuen Daten gut zurechtkommt, ist es sinnvoll, den Datenbestand nach dem Zufallsprinzip in separate Trainings- und Testdatenmengen aufzuteilen: Zum Trainieren und Optimieren des Lernmodells verwenden wir die Trainingsdatenmenge, während wir die Testdatenmenge bis zum Schluss zurückhalten, um das endgültige Modell bewerten zu können.

### 1.4.2 Trainieren und Auswählen eines Vorhersagemodells

Wie Sie in den nachfolgenden Kapiteln noch sehen werden, sind viele verschiedene Lernalgorithmen entwickelt worden, mit denen die unterschiedlichsten Aufgabenstellungen erledigt werden können. An dieser Stelle ist es allerdings wichtig festzuhalten, dass das Lernen nicht umsonst zu haben ist – so in etwa könnte man David Wolperts berühmte »No Free Lunch«-Theoreme zusammenfassen (*The Lack of A Priori Distinctions Between Learning Algorithms*, D.H. Wolpert 1996; *No Free Lunch Theorems for Optimization*, D.H. Wolpert und W.G. Macready, 1997). Noch besser lässt sich dieses Konzept anhand eines berühmten Zitats veranschaulichen:

»Wenn das einzige verfügbare Werkzeug ein Hammer ist, dürfte es verlockend sein, alles wie einen Nagel zu behandeln.« (Abraham Maslow, 1966). Beispielsweise sind alle Klassifikationsalgorithmen in irgendeiner Weise voreingenommen und kein Klassifikationsmodell ist anderen überlegen, wenn man nicht bestimmte Annahmen über die Aufgabenstellung macht. In der Praxis ist es daher von entscheidender Bedeutung, wenigstens eine Handvoll verschiedener Algorithmen zu vergleichen, um das am besten funktionierende Modell zu trainieren und auszuwählen. Aber um Vergleiche zwischen verschiedenen Modellen anstellen zu können, müssen zunächst einmal Bewertungskriterien festgelegt werden. Ein gebräuchliches Kriterium ist die *Korrektklassifikationsrate* (kurz: Klassifikationsrate, engl. *accuracy*, ACC) des Modells, die als Anteil der korrekten Klassifikationen definiert ist.

Nun stellt sich natürlich die Frage: *Wie kann man wissen, welches Modell mit den Testdaten und den »echten« Daten gut funktioniert, wenn man sie nicht bei der Auswahl des Modells verwendet, sondern bis zur Bewertung des endgültigen Modells zurückhält?* Um das mit dieser Frage verbundene Problem zu lösen, können verschiedene Kreuzvalidierungsverfahren eingesetzt werden, bei denen die Trainingsdatenmenge weiter in Trainings- und *Validierungsteilmengen* aufgeteilt wird, um die *Generalisierungsfähigkeit* des Modells abzuschätzen. Und schließlich dürfen wir auch nicht erwarten, dass die Standardparameter der Lernalgorithmen verschiedener Softwarebibliotheken für unsere spezielle Aufgabenstellung optimiert sind. Daher werden wir in den noch folgenden Kapiteln häufig Gebrauch von Verfahren zur *Hyperparameter-Optimierung* machen, um die Leistung unseres Modells feiner abzustimmen.

Man kann sich diese Hyperparameter als Parameter vorstellen, die nicht anhand der Daten ermittelt werden, sondern als Einstellungsmöglichkeiten zur Verbesserung der Leistung, was sehr viel klarer werden wird, wenn wir uns in den folgenden Kapiteln einige dazu passende Beispiele ansehen.

### 1.4.3 Bewertung von Modellen und Vorhersage anhand unbekannter Dateninstanzen

Nach der Auswahl eines an die Trainingsdaten angepassten Modells können wir die Testdatenmenge verwenden, um zu ermitteln, wie gut es mit diesen unbekannten Daten bei der Schätzung des Generalisierungsfehlers zurechtkommt. Sofern die Leistung des Modells zufriedenstellend ausfällt, können wir es verwenden, um anhand neuer, zukünftiger Daten Vorhersagen zu treffen. Hier muss angemerkt werden, dass die Parameter der vorhin erwähnten Verfahren (wie die Skalierung der Merkmalsdarstellungen oder die Dimensionsreduktion) ausschließlich anhand der Trainingsdatenmenge ermittelt werden. Dieselben Parameter werden später auch auf die Testdatenmenge und neue Daten angewendet – ansonsten könnte die bei den Testdaten gemessene Leistung zu optimistisch sein.

## 1.5 Machine Learning mit Python

Im Bereich Data Science ist Python eine der beliebtesten Programmiersprachen, daher gibt es eine Vielzahl nützlicher Bibliotheken, die von der sehr aktiven Python-Community entwickelt wurden.

Die Performance von Interpretersprachen wie Python ist derjenigen von komplizierten Programmiersprachen zwar unterlegen, es gibt allerdings Erweiterungsbibliotheken wie NumPy und SciPy, die auf maschinennahen Fortran- und C-Implementierungen beruhen, um schnelle vektorisierte Berechnungen mit mehrdimensionalen Arrays auszuführen.

Bei Aufgabenstellungen des Machine Learnings werden wir zumeist auf *scikit-learn* zurückgreifen, eine weitverbreitete und leicht verständliche Open-Source-Bibliothek für Machine Learning. In den nachfolgenden Kapiteln, in denen wir uns auf ein Teilgebiet des Machine Learnings namens *Deep Learning* konzentrieren, werden wir die neueste Version der TensorFlow-Bibliothek verwenden, die darauf spezialisiert ist, sogenannte tiefe neuronale Netze zu trainieren, indem sie die Rechenleistung von Grafikprozessoren nutzt.

### 1.5.1 Python und Python-Pakete installieren

Python ist für die drei wichtigsten Betriebssysteme Microsoft Windows, macOS und Linux verfügbar. Das Installationsprogramm und die Dokumentation stehen auf der offiziellen Website unter <https://www.python.org> zum Herunterladen bereit.

Dieses Buch setzt mindestens die Python-Version 3.7.0 voraus, es empfiehlt sich jedoch, immer die neueste verfügbare Python-3-Version zu verwenden. Einige der Codebeispiele sind möglicherweise auch mit Python-Versionen ab 2.7.0 kompatibel, aber da Python 2.7 seit 2019 offiziell nicht mehr unterstützt wird, was auch für die meisten Open-Source-Bibliotheken zutrifft (<https://python3statement.org>), raten wir dringend dazu, Python 3.7 oder neuer zu verwenden.

Die zusätzlichen Pakete, die wir im Buch benutzen werden, können mit *pip* installiert werden. Dieses Installationsprogramm gehört seit der Python-Version 3.3 zur Standardbibliothek. Weitere Informationen über *pip* finden Sie unter <https://docs.python.org/3/installing/index.html>.

Nach erfolgreicher Python-Installation können Sie mit *pip* wie folgt weitere Python-Pakete installieren:

```
pip install Paketname
```

Bereits installierte Pakete können mit der Option *--upgrade* aktualisiert werden:

```
pip install Paketname --upgrade
```

### 1.5.2 Verwendung der Python-Distribution Anaconda

Von Continuum Analytics gibt es eine sehr empfehlenswerte alternative Python-Distribution für wissenschaftliches Rechnen namens *Anaconda*. Hierbei handelt es sich um eine – auch für den kommerziellen Gebrauch – kostenlose Python-Distribution, die alle wichtigen Python-Pakete für Data Science, Mathematik und Engineering in einem einzigen, benutzerfreundlichen und plattformunabhängigen Paket bündelt. Das Installationsprogramm können Sie unter <https://docs.anaconda.com/anaconda/install/> herunterladen. Eine Kurzanleitung ist unter <https://docs.anaconda.com/anaconda/user-guide/getting-started/> verfügbar.

Nach der Installation von Anaconda können Python-Pakete mit dem folgenden Befehl installiert werden:

```
conda install Paketname
```

Bereits vorhandene Pakete werden so aktualisiert:

```
conda update Paketname
```

### 1.5.3 Pakete für wissenschaftliches Rechnen, Data Science und Machine Learning

Im weiteren Verlauf des Buches werden wir vornehmlich mehrdimensionale Arrays von NumPy verwenden, um Daten zu speichern und zu verarbeiten. Gelegentlich kommt auch *pandas* zum Einsatz, eine auf NumPy beruhende Bibliothek, die erweiterte Funktionen für die noch komfortablere Verarbeitung von Tabellen-daten bereitstellt. Zur Ergänzung des Lernerlebnisses werden wir darüber hinaus die sehr anpassungsfähige Matplotlib-Bibliothek einsetzen, die für die Visualisie- rung und das intuitive Verständnis quantitativer Daten oft äußerst nützlich ist.

Die Versionsnummern der im Buch verwendeten Python-Pakete sind nachste-hend aufgeführt. Vergewissern Sie sich, dass Ihre installierten Pakete mindestens diesen Versionsnummern entsprechen, damit gewährleistet ist, dass die Codebei-spiele korrekt ausgeführt werden.

- NumPy 1.17.4
- SciPy 1.3.1
- scikit-learn 0.22.0
- matplotlib 3.1.0
- pandas 0.25.3

## 1.6 Zusammenfassung

In diesem Kapitel haben wir einen ganz allgemeinen Blick auf das Thema Machine Learning geworfen und uns mit dem Gesamtbild sowie den grundlegenden Konzepten vertraut gemacht, die wir in den folgenden Kapiteln eingehender betrachten werden. Sie haben erfahren, dass überwachtes Lernen aus zwei wichtigen Teilgebieten besteht: Klassifikation und Regression. Klassifikationsmodelle ermöglichen es, Objekte bekannten Klassen zuzuordnen und wir können die Regressionsanalyse nutzen, um stetige Werte einer Zielvariablen vorherzusagen. Das unüberwachte Lernen bietet nicht nur praktische Verfahren zum Auffinden von Strukturen in nicht mit Label gekennzeichneten Daten, es kann darüber hinaus bei der Vorverarbeitung auch zur Datenkomprimierung eingesetzt werden.

Wir haben uns kurz die typische Vorgehensweise bei der Anwendung des Machine Learnings auf Problemstellungen angesehen, die bei der weiteren Erörterung und für praktische Beispiele in den folgenden Kapiteln als Grundlage dient. Darüber hinaus haben wir unsere Python-Umgebung eingerichtet und die erforderlichen Pakete aktualisiert und sind nun bereit, uns Machine Learning in Aktion anzusehen.

Im weiteren Verlauf des Buches werden wir neben dem Machine Learning selbst verschiedene Verfahren zur Vorverarbeitung von Daten vorstellen, die dabei helfen, mit verschiedenen Lernalgorithmen die beste Leistung zu erzielen. Wir werden uns im gesamten Buch ziemlich ausführlich mit Klassifikationsalgorithmen befassen, aber auch einige Verfahren der Regressionsanalyse und des Clusterings betrachten.

Vor uns liegt eine interessante Tour, auf der viele leistungsfähige Verfahren des weiten Felds des Machine Learning zur Sprache kommen. Wir gehen jedoch schrittweise vor und bauen auf das in den einzelnen Kapiteln allmählich erworbene Wissen auf. Im nächsten Kapitel beginnt diese Tour mit der Implementierung einer der ersten Lernalgorithmen zum Zweck der Klassifikation, die uns auf das Kapitel 3 (*Machine-Learning-Klassifikatoren mit scikit-learn verwenden*) vorbereitet, in dem wir die scikit-learn-Bibliothek nutzen werden, um erweiterte Lernalgorithmen zu erörtern.

# Stichwortverzeichnis

1-zu-n-Beziehung 600  
\_\_init\_\_()-Funktion 528  
\_Konvention (Unterstrich) 51

## A

Abrundungsfunktion 556  
AdaBoost 274  
AdaBoost-Algorithmus 276  
AdaBoost-Klassifikator-Ensemble 278  
Adaline 61  
Adaline-Algorithmus 411  
Adam-Optimierer 578  
Adversarial Examples 516  
Agent 34, 702, 704  
Agglomeratives Clustering 393, 401  
Ähnlichkeitsfunktion 112  
Aktionswertfunktion 712  
Aktivierungsfunktion 412  
    sigmoide 491  
    Tangens hyperbolicus 492  
Anaconda 43  
Antwortausfälle 133  
API  
    funktionale 527  
Appleby, Austin 300  
Approximationstheorem 525  
Architektur  
    speichern 490  
as\_dataset()-Methode 473  
Aufgabe  
    episodische 709  
    fortlaufende 709  
Aufteilung einer Datenmenge 486  
Ausdruck  
    regulärer 293  
Ausgabespektrum 496  
Ausreißer 355  
Autoencoder 650  
AutoGraph 507  
Autoregression 638  
Average Linkage 394  
Average-Pooling-Schicht 576  
Axon 46

## B

Backpropagation 441  
Backpropagation-Through-Time 609  
Bagging 268  
Bag-of-words-Modell 286  
Batchnormierung 678  
Bayes-Klassifikator 298  
Bellman, Richard 706  
Bellman-Gleichung 714  
Belohnung 713  
Belohnungsfunktion 34, 702  
Belohnungssignal 34, 704  
Bengio, Yoshua 550  
Benutzerdefinierte Schicht 529  
Berechnungsgraph 454, 503  
Beschneiden 584  
Bestimmtheitsmaß 360  
Bewertungsfunktion 654  
Bias 101, 102  
Bias-Einheit 47, 414  
Bias-Varianz-Dilemma 102  
Bibliotheken 51  
Bidirectional-Wrapper 623  
Bildtransformation 584  
Binäre Kreuzentropie 570  
Binärklassifikation 31  
bincount-Funktion 82  
Binomialkoeffizient 249  
Binomialverteilung 249  
BoostedTreeRegressor 542  
Boosting 274  
Bootstrapping 720  
Boston-Housing-Datensammlung 342  
Breiman, Leo 270, 275  
build\_model()-Funktion 635  
builder()-Funktion 473

## C

call()-Methode 480, 528  
CartPole-Umgebung 724, 725  
Cascading Style Sheets 321  
cast()-Funktion 457  
CelebA-Datenset 473, 582

- Chancenverhältnis 87  
close-Methode 314  
Cluster  
  Trägheit 380  
Clusteranalyse 377  
Clustering 35  
  agglomeratives 393, 401  
  divisives 393  
  graphenbasiertes 406  
  hierarchisches 393  
  prototypbasiertes 378  
  spektrales 406  
Clustering-Güte 386, 388  
CNN  
  Bausteine 549  
  Implementierung 563  
  Implementierung mit Keras-API 575  
  in Keras erstellen 576  
  mehrschichtige Architektur 573  
  Schicht konfigurieren 575  
Codierung der Klassenbezeichnung 140  
commit-Methode 314  
Complete Linkage 393  
concat()-Funktion 461  
connect-Methode 314  
Convolutional Neural Network *siehe* CNN  
corrcor-Funktion 347  
Counter-Klasse 614, 615  
CountVectorizer-Klasse 287  
CSV-Datei 134, 285  
CUDA 453  
cumsum-Funktion 175  
cursor-Methode 314  
CycleGAN 700
- D**  
DataFrame-Objekt 55  
Dataset 462  
DatasetsBuilder 486  
Daten  
  fehlende 133  
  kategoriale 138  
  sequenzielle 597  
Datenanalyse  
  explorative 344  
Datenaugmentation 582, 583  
Datenbank 313  
Datenkomprimierung 36  
Datenprojektion 206  
Datenvorverarbeitung 40  
DB Browser for SQLite) 315  
DBSCAN 402  
DCGAN (Deep Convolutional GAN) 676
- DecisionTreeRegressor 370  
Decoder-Netz 650  
Deep Learning 409  
Deep Q-Learning  
  Implementierung 742  
DeepFace 410  
Deep-Neural-Network (DNN) 409  
DeepSpeech 410  
Dekonvolution 677  
Dekorator 317, 507  
Dendriten 46  
Dendrogramm 393, 399  
Denoising Autoencoder 652  
Deserialisierung  
  Sicherheit 311  
Dichte 402  
Differenzierbarkeit 62  
Differenzieren  
  automatisches 442, 514, 515  
Dimensionsreduktion 36, 157  
Diskontierungsfaktor 710  
Diskriminator 655, 661  
Distanzmaß  
  euklidisches 379  
Distanzmatrix 394  
Divisives Clustering 393  
DNNClassifier 543  
Dokumenthäufigkeit  
  inverse 289  
download\_and\_prepare()-Methode 473  
DQN (Deep Q-Network) 739  
dropna-Methode 135  
Dropout 568  
Dropout-Klasse 576  
Dummy-Merkmal 142  
dump-Methode 310  
Dünn besetzter Merkmalsvektor 287  
Dynamische Programmierung 706, 715  
Dynamische Typisierung 508
- E**  
Earth-Mover's-Metrik 688  
Eigenvektor 173  
Eigenwert 173  
Eigenwertzerlegung 174  
Einbettung 618  
Eindimensionale Faltung 552  
Eingabekanal 563  
Elastic-Net-Verfahren 362  
Ellenbogenkriterium 387  
Emoticon 292  
Encoder-Netz 650  
Ensemble 124  
  Fehlerquote 250

Ensemble Learning 247  
 Ensemble-Methode 247  
 Entropie 116, 370  
 Entscheidungsbaum 115  
 Entscheidungsbaum-Klassifikator 233  
 Entscheidungsbaum-Regression 370  
 Entscheidungsbereich 60, 84  
 Entscheidungsfunktion 46  
 Entscheidungsgrenze 526  
 Episode 702  
 Episodische Aufgabe 709  
 Epoche 50  
 $\epsilon$ -Greedy-Policy 720  
 Erklärende Variable 340  
 Erklärte Variable 340  
 Euklidische Distanz 379  
 Euklidische Metrik 128  
 evaluate()-Methode 541  
 execute-Methode 314  
 Expectation-Maximization-Algorithmus (EM-Algorithmus) 305  
 Experience Replay 699  
 Exploitation 705  
 Exploration 705  
 Explorative Datenanalyse (EDA) 344

**F**

F1-Maß 237  
 Fakultätsfunktion 250  
 Falsch-Positiv-Rate 236  
 Faltung
 

- diskrete 552
- eindimensionale 552
- transponierte 676
- Zero-Padding 554
- zweidimensionale 557

 Faltungsschicht 551, 562  
 Farbkanal 563  
 FCM-Algorithmus 385  
 Feedforward-Netz 413  
 Fehlende Daten 133  
 Fehlerquote 249  
 Fehlerrate 236  
 Fehlklassifikation 58  
 Fehlklassifikationsrate 83  
 Feld
 

- rezeptives 551

 Filter 552  
 fit\_transform-Methode 141  
 fit-Methode 54, 137, 484  
 Flask 315, 316  
 floor-Funktion 556  
 Fluch der Dimensionalität 130, 161, 406

Formularvalidierung 318, 320  
 Fortlaufende Aufgabe 709  
 Freund, Yoav 274  
 Friedman, Jerome 281  
 Full-Padding 555  
 Funktion
 

- konvexe 62
- logistische 88, 492

 Funktionale API 527  
 Funktionsapproximation 739  
 Funktionsdekorator 507  
 Fuzziness-Koeffizient 385  
 Fuzzy-C-Mean-Algorithmus (FCM) 384

## G

Galton, Francis 33  
 GAN (Generative Adversarial Network) 25, 649  
 Ganzzahlen als Klassenbezeichnungen 81  
 Gate (LSTM) 611  
 Gaußklammer 556  
 Gaußscher Kernel 112  
 Generalisierungsfehler 218  
 Generatives Modell 652  
 Generator 654, 661  
 get\_dummies-Methode 143  
 Gewichtung 252  
 Gewichtungsvektor 49  
 Gini-Koeffizient 116  
 Global Interpreter Lock (GIL) 452  
 Global-Average-Pooling 590  
 Glorot-Initialisierung 511  
 Google Colab 658  
 Google Translate 410  
 GPI (Generalized Policy Iteration) 717  
 Gradient Boosting 281  
 Gradientenabstiegsverfahren 62
 

- als Batch-Verarbeitung 64
- stochastisches 71

 Gradientenberechnung 517  
 Gradienten-Clipping 610  
 GradientTape 514  
 Grafikprozessor (GPU) 452  
 Graphenbasiertes Clustering 406  
 GraphViz 121  
 Graustufenkanal 563  
 Greedy-Algorithmus 158  
 Grid Search 230  
 Grid-World
 

- Implementierung 732

 Grid-World-Umgebung 726  
 Gruppierung
 

- nach Ähnlichkeit 377

**H**

- Halbmondform 200, 403
- HashingVectorizer 300
- Hauptkomponentenanalyse 169, 179
- Heatmap 347, 399
- Heaviside-Funktion 46, 412
- Helligkeit 584
- Hierarchisches Clustering 393
- Hinton, Geoffrey 550
- Hochreiter, Sepp 610
- Hoff, Tedd 61
- Holdout-Methode 218
- Hundedressur 703
- Hyperebene 111
- Hyperparameter 41, 67, 218, 230, 414

**I**

- IMDb-Datensammlung 284
- Infimum 688
- Informationsgewinn 116, 370
- Informationstheorie 689
- Inlier 355
- Interpolationsverfahren 136
- Inverse Dokumenthäufigkeit 289
- inverse\_transform-Methode 141
- Iris-Datensammlung 37, 80
- isnull-Methode 134

**J**

- Jensen-Shannon-Divergenz 689
- Jinja2 320
- Jupyter Notebook 658

**K**

- Kante
  - rekurrente 602
- Kapazität 522, 567
- Kategoriale Daten 138
- Kategoriale Kreuzentropie 570
- Keras 478
- Kernel 112
  - gaußscher 197
  - polynomialer 197
  - Tangens hyperbolicus 197
- Kernel-Funktion 112, 194
- Kernel-Hauptkomponentenanalyse 193
- Kernel-Methode 110
- KernelPCA-Klasse 210
- Kernel-SVM 109
- Kernel-Trick 112, 194
- Kernobjekt 402
- Kettenregel (Ableitung) 442, 514
- Klasse 31
- Klassenverteilung
  - unausgewogene 242
- Klassifikation 31
  - binäre 31, 46
  - unüberwachte 35
- Klassifikationsfehler 116
- Klassifikationsgüte 234
- Klassifikationsrate 236
- Klassifikator
  - Fehlerarten 234
  - schwacher 274
- Klassifikator-Attribute 266
- k-Means++-Algorithmus 383
- k-Means-Algorithmus 377
- KMeans-Klasse 380
- k-Nearest-Neighbors-Algorithmus 127
- KNN *siehe* k-Nearest-Neighbors-Algorithmus
- Kollinearität 143, 188
- Konfusionsmatrix 234
- Kontrast 584
- Konvention (Unterstrich) 51
- Konvergenz 447
- Konvexe Funktion 62
- Konzentrischer Kreis 203
- Korrektklassifikationsrate 41, 83
- Korrelationskoeffizient 346
- Korrelationsmatrix 346, 347
- Kovarianz 173
- Kovarianzmatrix 173, 346
- Kovarianzverschiebung 680
- Kreis
  - konzentrisch 203
- Kreuzentropie 570
- Kreuzvalidierung
  - 5x2- 233
  - k-fache 218, 219
  - stratifizierte k-fache 221
  - verschachtelte 232
- Kullback-Leibler-Divergenz 689

**L**

- L1-Regularisierung 151, 153
- L2-Regularisierung 102, 151
- L2-Strafterm 152
- LabelEncoder-Klasse 141
- LabelEncoder-Objekt 214
- Lancaster-Stemmer 294
- LASSO 362
- LatentDirichletAllocation-Klasse 303
- Latente Dirichlet-Allokation 303
- Latenter Vektor 650
- Lazy-Learning-Algorithmus 127

- LDA *siehe* Lineare Diskriminanzanalyse  
 LDA-Klasse 191  
 Leaky-ReLU-Aktivierungsfunktion 662  
 Leave-One-Out-Kreuzvalidierung (LOO-Kreuzvalidierung) 221  
 LeCun, Yann 550  
 Leerraum 293  
 Lemmatisierung 294  
 Lernen  
     instanzbasiertes 127  
     überwachtes 30  
     unüberwachtes 30, 35, 377  
     verstärkendes 30  
 Lernen durch Interaktion 702  
 Lernkurve 225, 523  
 Lernrate 48, 67  
     adaptive 72  
 LIBLINEAR-Bibliothek 108  
 LIBSVM-Bibliothek 108  
 linalg.eig-Funktion 173  
 Lineare Diskriminanzanalyse (LDA) 183  
 Lineare Regression 33, 340  
 Lineare Trennbarkeit 50, 61  
 Linkage-Matrix 396  
 Lipschitzstetigkeit 691  
 load()-Funktion 424, 476  
 LogisticRegression-Klasse 98  
 Logistische Funktion 88, 492  
 Logistische Regression 87  
 logit-Funktion 88  
 Logits 638  
 Log-Likelihood-Funktion 91  
 Lösch-Gate 611  
 LSTM (Long Short-Term Memory) 610, 621  
     Ausgabe-Gate 611  
     Eingabe-Gate 611  
     Lösch-Gate 611  
     Speicherzelle 610  
     Zellzustand 610
- M**
- Machine Learning  
     Anwendungen 30  
 Makro-Mittelwertbildung 241  
 Manhattan-Metrik 130  
 map-Methode 140  
 Margin 105  
 Markov-Eigenschaft 715  
 Markov-Entscheidungsprozess 705  
 Markov-Prozess  
     Kanten 708  
     Knoten 708
- Maslow, Abraham 41  
 Matplotlib 43  
 Matrix-Vektor-Multiplikation 66  
 Max-Pooling 561  
 Max-Pooling-Schicht 576  
 McCulloch, Warren 45, 409  
 mean-Methode 70  
 Mean-Pooling 561  
 Median der absoluten Abweichungen (MAD)  
     356  
 Medianwert 137  
 Medoid 378  
 Mehrheit  
     absolute 247  
     relative 248  
 Mehrheitsentscheidung 128, 247  
 Mehrheitsentscheidungs-Klassifikator 251  
 Mehrklassen Klassifikation 241  
 merge\_mode 623  
 Merkmal 37, 39  
     Auswahl 150, 157  
     Bedeutung 164  
     Extrahierung 157  
     Extraktion 169, 550  
     Interpretierbarkeit 166  
     nominales 138, 534  
     ordinales 138, 534  
     Streuung 680  
 Merkmalsanpassung 699  
 Merkmalshierarchie 550  
 Merkmalskarte 551, 565  
 Merkmalsspalte 534  
 Merkmalsstandardisierung 148  
 Merkmalstransformation 176  
 Merkmalsvektor  
     dünn besetzter 286  
 Methode der kleinsten Quadrate 348  
 metric-Parameter 130  
 metrics-Modul 83, 237  
 Microframework 316  
 Mikro-Mittelwertbildung 241  
 Minibatch-Learning 72, 447  
 Minkowski-Metrik 130  
 Min-Max-Skalierung 148  
 Mit Zurücklegen 124  
 Mittelwert-Imputation 136  
 Mittelwertvektor 185  
 Mittlere quadratische Abweichung 360  
 MLxtend-Bibliothek 344, 523  
 MNIST-Datensammlung 419  
 Modalwert 249  
 model\_selection-Modul 81  
 Model-Klasse 528

- M**
- Modell
    - generatives 652
    - speichern und einlesen 490
    - wiederherstellen 541
  - Modellauswahl 218
  - Modellbasiertes Verfahren 707
  - Modellbewertung 219
  - Modellkapazität 522
  - Monte-Carlo-Verfahren 707
  - Multi-Head-Attention (MHA) 646
  - Multiklassen-Klassifikation 32
  - multiprocessing-Bibliothek 452
- N**
- NaN (Not a Number) 133
  - Natural Language Processing (NLP) 283
  - Natural Language Toolkit (NLTK) 294
  - Negative Klasse 31
  - Netflix-Preis 281
  - Netzeingabe 46
  - Neuron 45
    - adaptives lineares 45
  - Neuronales Netz 409
    - Konvergenz 447
  - N-Gramm 288
  - NHWC-Format 576
  - NLP *siehe* Natural Language Processing
  - NoisyLinear-Schicht 529
  - Nominales Merkmal 138, 534
  - Normalgleichung 355
  - Normierung 40, 148
  - n-zu-1-Beziehung 600
  - n-zu-n-Beziehung 600
- O**
- Off-Policy-TD-Steuerung 723
  - Ohne Zurücklegen 124
  - One-hot-Codierung 142, 415
  - OneHotEncoder-Klasse 142
  - One-vs.-All (OvA) 55, 415
  - Online Learning 72, 109
  - OpenAI Gym 723
  - OpenCL 453
  - Opinion Mining 283
  - Ordinales Merkmal 138, 534
  - Out-of-Core Learning 298
- P**
- Padding 553, 555
  - pandas 43
  - partial\_fit-Methode 73
  - PCA (Principal Component Analysis) *siehe* Hauptkomponentenanalyse
  - PCA-Klasse 179
  - permutation-Methode 76
  - Perzeptron 45, 50, 80
    - mehrschichtiges 413
  - Perzeptron-Lernregel 48
  - pickle-Modul 309
  - pip (Installationsprogramm) 42
  - Pipeline 213, 215
  - Pitts, Walter 45, 409
  - Pix2Pix-Algorithmus 700
  - plot\_decision\_regions-Methode 85
  - Policy 712
    - Policy-Bewertung 716
    - Policy-Iteration 717
    - Policy-Verbesserung 717
  - Polymorphie 508
  - Polynomiale Regression 363
  - Pooling-Größe 561
  - Pooling-Schicht 551
  - Porter-Stemmer-Algorithmus 294
  - Porträtfoto 650
  - Positive Klasse 31
  - Powers, David M. W. 237
  - predict\_proba-Methode 100
  - predict-Methode 54, 83, 138, 541
  - preprocessing-Modul 82
  - Problem des explosionsartig wachsenden Gradienten 609
  - Problem des verschwindenden Gradienten 414, 498, 609
  - Programmierung
    - dynamische 706, 715
  - Projektionsmatrix 176
  - Pruning 116
  - PyDotPlus-Bibliothek 122
  - Pyglet-Bibliothek 727
  - PyPrind 284
  - PythonAnywhere 333
  - Python-Pakete 42
- Q**
- Q-Learning 723
    - Implementierung 734
- R**
- Randobjekt 402
  - Random Forest 124, 371
  - Random-Forest-Klassifikator 126, 164
  - Random-Forest-Regression 369, 371
  - Randomisierte Suche 231
  - Rang eines Tensors 454
  - RANSAC-Algorithmus 355
  - Rastersuche *siehe* Grid Search

- Rauschobjekt 402  
 Rauschunterdrückung 652  
 RBF-Kernel 112  
 read\_csv-Funktion 134  
 Receiver-Operating-Characteristic-Diagramme (ROC-Diagramme) 238  
 Regex-Bibliothek 292  
 Regressand 340  
 Regression 31
  - Gradientenabstiegsverfahren 96
  - lineare 33, 340
  - logistische 87
  - multinomiale logistische 87
  - multiple lineare 341
  - polynomiale 363
 Regressionsanalyse 32, 339  
 Regressionsgerade 340  
 Regressions-Hyperebene 358  
 Regressionsmodell
  - Leistung 358
  - lineares 479
  - multiples 358
  - Random Forest 372
 Regressor 340  
 Regulärer Ausdruck 292, 293  
 Regularisierung 102, 361, 566
  - Dropout 568
 Regularisierungsparameter 102, 266  
 Regularisierungsstärke 229  
 Regularisierungsterm 439  
 Reinforcement Learning 701  
 Rekurrente Kante 602  
 Rekurrentes neuronales Netz *siehe* RNN  
 ReLU (Rektifizierte Lineareinheit) 498  
 Replay Buffer 740  
 Resampling 244  
 reset()-Methode 725  
 reshape()-Funktion 457  
 Residualdiagramm 358  
 Residuale Verbindung 646  
 Residuum 340  
 Return 710, 713  
 Rezeptives Feld 551  
 Richtig-Positiv-Rate 236  
 Ridge-Regression 362  
 RNN (Rekurrentes neuronales Netz) 597
  - Aktivierung 603
  - Architektur 602
  - Datenaufbereitung 613
  - Datenbereinigung 630
  - Implementierung mit TensorFlow 612
  - Modell erstellen 620
  - Zeitschritt 603
 RobustScaler 150  
 ROC AUC 238  
 ROC-Kurve 238  
 Ronacher, Armin 315  
 Rosenblatt, Frank 46
- S**
- Same-Padding 555  
 Sättigung 656  
 savez-Funktion 424  
 SBS-Algorithmus 158  
 scatterplotmatrix-Funktion 344  
 Schapire, Robert 274  
 Schätzer 138, 215, 216, 533  
 Schätzer-API 137  
 Schicht
  - benutzerdefinierte 529
 Schichtnormierung 646  
 Schlupfvariable 106  
 Schmidhuber, Jürgen 610  
 Schwellenwert 46  
 Schwellenwertfunktion 412  
 scikit-learn-API 80  
 score-Methode 84  
 Selbst-Aufmerksamkeit 643  
 select-Befehl 314  
 SelectFromModel-Objekt 166  
 Sentimentanalyse 283  
 Sequential Backwards Selection *siehe* SBS-Algorithmus  
 Sequential-Klasse 487, 576  
 Sequenz
  - Eigenschaften 598
  - Repräsentierung 599
 Sequenzielle Daten 597  
 Sequenzmodellierung 599
  - Kategorien 600
 Serialisierung 309  
 SGDClassifier-Klasse 109  
 shuffle-Methode 73  
 Sigmoidfunktion 88, 491  
 Signal 552  
 Silhouettenanalyse 388  
 Silhouettendiagramm 389  
 Silhouettenkoeffizient 388  
 SimpleImputer-Klasse 136  
 SimpleRNN 621  
 Single Linkage 393  
 Skalarprodukt 54  
 Skaleninvarianz 148  
 Skalierungsfaktor 640  
 skip()-Funktion 486  
 Snowball-Stemmer 294

Soft-Margin-Klassifikation 106  
 softmax-Funktion 494  
 Softmax-Regression 87  
 Spektrales Clustering 406  
 Spiegelung 584  
 split()-Funktion 460  
 Sprachmodellierung 629  
 Sprungfunktion 46  
 SQLite 313  
 sqlite3-Bibliothek 313  
 squeeze()-Funktion 457  
 stack()-Funktion 461  
 Stacking 267  
 Stammformreduktion 293  
 Standard-GAN 662  
 Standardisierung 69, 82, 148  
 StandardScaler-Klasse 82  
 std-Methode 70  
 Stemming 293  
 Stetigkeit 339  
 Stimmungsanalyse 283, 612  
 Stopwort 295  
 Straffunktion 62, 91, 92, 412
 

- Fehleroberfläche 442
- logistische 439

 Strafterm 152  
 stream\_docs-Funktion 300  
 Streudiagrammmatrix 344  
 Streumatrix 186  
 StringIO-Funktion 134  
 Suche
 

- randomisierte 231

 summary()-Methode 481  
 sum-Methode 134  
 Support Vector Machine 104  
 Support Vector Machine (SVM) 374  
 Supremum 688  
 SVM *siehe* Support Vector Machine  
 SVM-Pipeline 231  
 Szegedy, Christian 516

## T

take()-Funktion 486  
 Tangens hyperbolicus 492, 495  
 Temporal Difference 707  
 Temporal Difference Learning 720  
 Tensor

- Datentyp 457
- erstellen 456
- Format 457
- Norm 459
- transponieren 458
- umformen 458

TensorFlow
 

- Berechnungsgraph 454, 503
- GPU-Unterstützung 455
- Grundlagen 453
- Installation 455
- Knoten 504
- lineares Regressionsmodell 479
- Merkmale 502
- Operation 504
- Sitzung 505
- Trainingsleistung 451
- Variable 509

 tensorflowdatasets-Bibliothek 472  
 Testdatenmenge 218  
 Text
 

- aufteilen 303
- Textbereinigung 291, 292
- Texterzeugung 633, 637
- Textfeld 320
- tf.image 469
- tf.io 469
- tf.keras 478
- Tf-idf-Maß 289
- Token 286
- Tokenisierung 293
- Tokenizer-Klasse 614
- TokenTextEncoder 615
- Tonalität 283
- Topic Modeling 302
- Totale Variation 688
- train\_test\_split-Funktion 81, 146
- train()-Methode 541
- Training 39
- Trainingsbeispiel 38
- Trainingsdaten 31
- Trainingsdatenmenge 218
- Transformer 215, 216
- Transformer-Architektur 642
- Transformer-Block 646
- Transformer-Klasse 137
- transform-Methode 137, 138
- Transponierte 47
- Transponierte Faltung 676
- transpose()-Funktion 457
- Trefferquote 236
- Trennbarkeit
  - lineare 61
- Typisierung
  - dynamische 508

## U

Überanpassung 84, 101, 150, 567  
 Übergangsquintupel 741  
 Übergangswahrscheinlichkeit 707  
 Überwachtes Lernen 30

Umgebung 34, 704  
 Unteranpassung 101, 567  
 Unterklassenbildung 528  
 Unterscheidbarkeit 189  
 Unüberwachtes Lernen 30, 377

**V**

VAE (Variational Autoencoder) 653  
 Validierungsdatenmenge 41, 218  
 Validierungskurve 227  
 Validierungsmenge 161  
 Variable  
     erklärende 32, 340  
     erklärte 340  
 Variable-Objekt 509  
 Varianz 101, 102  
     erklärte 174  
 Varianzreduktion 370  
 Variation  
     totale 688  
 Vaswani, Ashish 642  
 Vektor  
     latenter 650  
 Vektorisierung 54  
 Verbindung  
     residuale 646  
 Verfahren  
     modellbasiertes 707  
 Verlustfunktion 39, 520  
 Verne, Jules 630  
 Verstärkendes Lernen 30  
 Verwechslungsmatrix 234  
 Vokabular 286  
 Vorhersagemodell 40  
 Vorkommenshäufigkeit 288, 289  
 Vorverarbeitung 40, 148  
 Vorwärtspropagation 416  
 Vorzeichenfunktion 249

**W**

Wahrscheinlichkeit  
     bedingte 88

Wahrscheinlichkeitsverteilung 707  
 Ward Linkage 394  
 Webanwendung 316  
 Wein-Datensatz 145  
 Wertfunktion 655, 712, 714  
 Wertiteration 718  
 Wertvorhersage 721  
 Whitespace 293  
 Widrow, Bernard 61  
 Widrow-Hoff-Regel 61  
 Winograd-Algorithmus 561  
 Wisconsin-Brustkrebs-Datensammlung 214  
 Wolpert, David 40, 268  
 word2vec 302  
 Wortrelevanz 289  
 Wortsequenz 613  
 WTForms-Bibliothek 318

**X**

Xavier-Initialisierung 511  
 XOR-Datensammlung 110  
 XOR-Klassifikation 521, 546

**Z**

Zeitreihe 598  
 Zeitstempel 314  
 Zentroid *siehe* Zentrum  
 Zentrum 378  
 Zero-Padding 553  
 Ziel 39  
 Zielfunktion 62  
 Zielvariable 32, 340  
 Zufallsvariable 706  
 Zufallszahlengenerator 73  
 Zugehörigkeitsgrad 385  
 Zuordnungsfunktion 111  
 Zurücklegen (mit/ohne) 124  
 Zustandsraum 739  
 Zustandsübergangswahrscheinlichkeit 708  
 Zweidimensionale Faltung 557