



mitp

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

Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Liebe Leserinnen und Leser,

dieses E-Book, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Mit dem Kauf räumen wir Ihnen das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Jede Verwertung außerhalb dieser Grenzen ist ohne unsere Zustimmung unzulässig und strafbar. Das gilt besonders für Vervielfältigungen, Übersetzungen sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Je nachdem wo Sie Ihr E-Book gekauft haben, kann dieser Shop das E-Book vor Missbrauch durch ein digitales Rechtemanagement schützen. Häufig erfolgt dies in Form eines nicht sichtbaren digitalen Wasserzeichens, das dann individuell pro Nutzer signiert ist. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Beim Kauf des E-Books in unserem Verlagsshop ist Ihr E-Book DRM-frei.

Viele Grüße und viel Spaß beim Lesen,

Ihr mitp-Verlagsteam



Neuerscheinungen, Praxistipps, Gratiskapitel,
Einblicke in den Verlagsalltag –
gibt es alles bei uns auf Instagram und Facebook



[instagram.com/mitp_verlag](https://www.instagram.com/mitp_verlag)



[facebook.com/mitp.verlag](https://www.facebook.com/mitp.verlag)

Sebastian Raschka
Vahid Mirjalili

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

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

3., aktualisierte und erweiterte Auflage

Übersetzung aus dem Amerikanischen
von Knut Lorenzen



Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Bei der Herstellung des Werkes haben wir uns zukunftsbewusst für umweltverträgliche und wiederverwertbare Materialien entschieden.

Der Inhalt ist auf elementar chlorfreiem Papier gedruckt.

ISBN 978-3-7475-0213-6

3. Auflage 2021

www.mitp.de

E-Mail: mitp-verlag@sigloch.de

Telefon: +49 7953 / 7189 - 079

Telefax: +49 7953 / 7189 - 082

Copyright © Packt Publishing 2019

First published in the English language under the title 'Python Machine Learning – Third Edition – (9781789955750)'

© 2021 mitp Verlags GmbH & Co. KG, Frechen

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.



Lektorat: Janina Bahlmann

Fachkorrektur: Dr. Friedhelm Schwenker

Sprachkorrektur: Petra Heubach-Erdmann

Coverbild: Shantanu N. Zagade, © Packt Publishing

Satz: III-satz, Husby, www.drei-satz.de

Druck: Plump Druck & Medien GmbH, Rheinbreitbach

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 Vorhersagemodells	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

2	Lernalgorithmen für die Klassifikation trainieren.	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
3	Machine-Learning-Klassifikatoren mit scikit-learn verwenden	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	Gut geeignete Trainingsdatenmengen: Datenvorverarbeitung	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 Merkmalen	157
4.6	Beurteilung der Bedeutung von Merkmalen mit Random Forests.	164
4.7	Zusammenfassung	167
5	Datenkomprimierung durch Dimensionsreduktion	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 Merkmalsunterraum	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
6	Bewährte Verfahren zur Modellbewertung und Hyperparameter-Optimierung	213
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	Kombination verschiedener Modelle für das Ensemble Learning. . .	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	Machine Learning zur Analyse von Stimmungslagen nutzen.	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
9	Einbettung eines Machine-Learning-Modells in eine Webanwendung	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
10	Vorhersage stetiger Zielvariablen durch Regressionsanalyse.	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
11	Verwendung von Daten ohne Label: Clusteranalyse.	377
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

12	Implementierung eines künstlichen neuronalen Netzes	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
13	Parallelisierung des Trainings neuronaler Netze mit TensorFlow ..	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 tf.data 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 tensorflow_datasets-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	Die Funktionsweise von TensorFlow im Detail	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	Merkmalspalten	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	Bildklassifikation mit Deep Convolutional Neural Networks	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 TensorFlows 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	Modellierung sequenzieller Daten durch rekurrente neuronale Netze	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	Synthetisieren neuer Daten mit Generative Adversarial Networks	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 Diskriminator-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
18	Entscheidungsfindung in komplexen Umgebungen per Reinforcement Learning	701
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
	Stichwortverzeichnis	751



Über die Autoren

Sebastian Raschka erlangte seinen Dokortitel 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 Dokortitel 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 Taban 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 Fotograf, 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 Erlernete 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 vorherzusagen 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 TensorFlow 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.

