O'REILLY®

3. Auflage
Aktualisiert
und erweitert

Praxiseinstieg Machine Learning

mit Scikit-Learn, Keras und TensorFlow

Konzepte, Tools und Techniken
für intelligente Systeme

powered by
jupyter

Aurélien Géron
Übersetzung von Kristian Rother
und Thomas Demmig

3. AUFLAGE

Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow

Konzepte, Tools und Techniken für intelligente Systeme

Aurélien Géron

Deutsche Übersetzung von Kristian Rother & Thomas Demmig

O'REILLY®

Aurélien Géron

Lektorat: Alexandra Follenius
Übersetzung: Kristian Rother, Thomas Demmig
Korrektorat: Sibylle Feldmann, *www.richtiger-text.de*
Satz: III-satz, *www.drei-satz.de*
Herstellung: Stefanie Weidner
Umschlaggestaltung: Karen Montgomery, Michael Oréal, *www.oreal.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

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.

ISBN:
Print 978-3-96009-212-4
PDF 978-3-96010-760-6
ePub 978-3-96010-761-3
mobi 978-3-96010-762-0

3., aktualisierte und erweiterte Auflage 2023

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde mit mineralölfreien Farben auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie. Hergestellt in Deutschland.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *kommentar@oreilly.de*.

5 4 3 2 1 0

Inhalt

Vorwort **17**

Teil I Die Grundlagen des Machine Learning

1 Die Machine-Learning-Umgebung **31**
Was ist Machine Learning? 32
Warum wird Machine Learning verwendet? 33
Anwendungsbeispiel 36
Unterschiedliche Machine-Learning-Systeme 37
Trainingsüberwachung 38
Batch- und Online-Learning 46
Instanzbasiertes versus modellbasiertes Lernen 49
Die wichtigsten Herausforderungen beim Machine Learning 55
Unzureichende Menge an Trainingsdaten 55
Nicht repräsentative Trainingsdaten 56
Minderwertige Daten 58
Irrelevante Merkmale 58
Overfitting der Trainingsdaten 59
Underfitting der Trainingsdaten 61
Zusammenfassung 62
Testen und Validieren 62
Hyperparameter anpassen und Modellauswahl 63
Datendiskrepanz 64
Übungen 66

2 Ein Machine-Learning-Projekt von A bis Z **69**
Der Umgang mit realen Daten 69
Betrachte das Gesamtbild 71
Die Aufgabe abstecken 71

Wähle ein Qualitätsmaß aus 73
Überprüfe die Annahmen 76
Beschaffe die Daten 76
Die Codebeispiele mit Google Colab ausführen 76
Ihre Codeänderungen und Daten sichern 79
Interaktivität – mächtig, aber gefährlich 80
Code im Buch versus Notebook-Code 80
Die Daten herunterladen 81
Wirf einen kurzen Blick auf die Datenstruktur 82
Erstelle einen Testdatensatz 86
Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen . . . 91
Visualisieren geografischer Daten 91
Suche nach Korrelationen 93
Experimentieren mit Kombinationen von Merkmalen 96
Bereite die Daten für Machine-Learning-Algorithmen vor 97
Aufbereiten der Daten 98
Bearbeiten von Text und kategorischen Merkmalen 101
Skalieren und Transformieren von Merkmalen 105
Eigene Transformer 109
Pipelines zur Transformation 113
Wähle ein Modell aus und trainiere es 118
Trainieren und auswerten auf dem Trainingsdatensatz 118
Bessere Auswertung mittels Kreuzvalidierung 120
Optimiere das Modell 122
Gittersuche 122
Zufällige Suche 124
Ensemble-Methoden 125
Analysiere die besten Modelle und ihre Fehler 126
Evaluiere das System auf dem Testdatensatz 127
Nimm das System in Betrieb, überwache und warte es 128
Probieren Sie es aus! 131
Übungen 132

3 Klassifikation 135
MNIST 135
Trainieren eines binären Klassifikators 138
Qualitätsmaße 139
Messen der Genauigkeit über Kreuzvalidierung 139
Konfusionsmatrix 140
Relevanz und Sensitivität 142
Die Wechselbeziehung zwischen Relevanz und Sensitivität 143
Die ROC-Kurve 147

Klassifikatoren mit mehreren Kategorien 151
Fehleranalyse 154
Klassifikation mit mehreren Labels 158
Klassifikation mit mehreren Ausgaben 160
Übungen 161

4 Trainieren von Modellen 163
Lineare Regression 164
Die Normalengleichung 166
Komplexität der Berechnung 169
Das Gradientenverfahren 170
Batch-Gradientenverfahren 173
Stochastisches Gradientenverfahren 176
Mini-Batch-Gradientenverfahren 179
Polynomielle Regression 181
Lernkurven 183
Regularisierte lineare Modelle 187
Ridge-Regression 187
Lasso-Regression 189
Elastic-Net-Regression 192
Early Stopping 193
Logistische Regression 194
Abschätzen von Wahrscheinlichkeiten 195
Trainieren und Kostenfunktion 196
Entscheidungsgrenzen 197
Softmax-Regression 201
Übungen 204

5 Support Vector Machines 207
Lineare Klassifikation mit SVMs 207
Soft-Margin-Klassifikation 208
Nichtlineare SVM-Klassifikation 210
Polynomieller Kernel 212
Ähnlichkeitsbasierte Merkmale 213
Der gaußsche RBF-Kernel 213
SVM-Klassen und Komplexität der Berechnung 215
SVM-Regression 216
Hinter den Kulissen linearer SVM-Klassifikatoren 218
Das duale Problem 221
Kernel-SVM 222
Übungen 225

6 Entscheidungsbäume 227
Trainieren und Visualisieren eines Entscheidungsbaums 227
Vorhersagen treffen 229
Schätzen von Wahrscheinlichkeiten für Kategorien 231
Der CART-Trainingsalgorithmus 232
Komplexität der Berechnung 233
Gini-Unreinheit oder Entropie? 233
Hyperparameter zur Regularisierung 234
Regression 236
Empfindlichkeit für die Ausrichtung der Achsen 238
Entscheidungsbäume haben eine größere Varianz 239
Übungen 240

7 Ensemble Learning und Random Forests 243
Abstimmverfahren unter Klassifikatoren 244
Bagging und Pasting 247
Bagging und Pasting in Scikit-Learn 249
Out-of-Bag-Evaluation 250
Zufällige Patches und Subräume 251
Random Forests 252
Extra-Trees 253
Wichtigkeit von Merkmalen 253
Boosting 255
AdaBoost 255
Gradient Boosting 258
Histogrammbasiertes Gradient Boosting 262
Stacking 263
Übungen 266

8 Dimensionsreduktion 269
Der Fluch der Dimensionalität 270
Die wichtigsten Ansätze zur Dimensionsreduktion 271
Projektion 271
Manifold Learning 273
Hauptkomponentenzerlegung (PCA) 275
Erhalten der Varianz 276
Hauptkomponenten 276
Die Projektion auf d Dimensionen 278
Verwenden von Scikit-Learn 278
Der Anteil erklärter Varianz 279
Auswählen der richtigen Anzahl Dimensionen 279

PCA als Komprimierungsverfahren 281
Randomisierte PCA. 282
Inkrementelle PCA 282
Zufallsprojektion 284
LLE 286
Weitere Techniken zur Dimensionsreduktion 289
Übungen 290

9 Techniken des unüberwachten Lernens 293
Clustering-Algorithmen: *k*-Means und DBSCAN 294
k-Means 297
Grenzen von *k*-Means 307
Bildsegmentierung per Clustering. 308
Clustering für teilüberwachtes Lernen einsetzen 310
DBSCAN. 313
Andere Clustering-Algorithmen 316
Gaußsche Mischverteilung 318
Anomalieerkennung mit gaußschen Mischverteilungsmodellen . . . 322
Die Anzahl an Clustern auswählen. 324
Bayessche gaußsche Mischverteilungsmodelle 326
Andere Algorithmen zur Anomalie- und Novelty-Erkennung 327
Übungen 329

Teil II Neuronale Netze und Deep Learning

10 Einführung in künstliche neuronale Netze mit Keras 333
Von biologischen zu künstlichen Neuronen 334
Biologische Neuronen. 335
Logische Berechnungen mit Neuronen. 337
Das Perzeptron 338
Mehrschichtiges Perzeptron und Backpropagation 342
Regressions-MLPs. 347
Klassifikations-MLPs 349
MLPs mit Keras implementieren 350
Einen Bildklassifikator mit der Sequential API erstellen. 351
Ein Regressions-MLP mit der Sequential API erstellen. 362
Komplexe Modelle mit der Functional API bauen 363
Dynamische Modelle mit der Subclassing API bauen. 369
Ein Modell sichern und wiederherstellen 371
Callbacks. 372
TensorBoard zur Visualisierung verwenden 373

Feinabstimmung der Hyperparameter eines neuronalen Netzes 378
Anzahl verborgener Schichten 383
Anzahl Neuronen pro verborgene Schicht 384
Lernrate, Batchgröße und andere Hyperparameter 385
Übungen 387

11 Trainieren von Deep-Learning-Netzen 391
Das Problem schwindender/explodierender Gradienten 392
Initialisierung nach Glorot und He 393
Bessere Aktivierungsfunktionen 395
Batchnormalisierung 401
Gradient Clipping 407
Wiederverwenden vortrainierter Schichten 408
Transfer Learning mit Keras 410
Unüberwachtes Vortrainieren 412
Vortrainieren anhand einer Hilfsaufgabe 413
Schnellere Optimierer 414
Momentum 414
Beschleunigter Gradient nach Nesterov 416
AdaGrad 417
RMSProp 419
Adam 419
Scheduling der Lernrate 423
Vermeiden von Overfitting durch Regularisierung 428
ℓ_1- und ℓ_2-Regularisierung 428
Drop-out 429
Monte-Carlo-(MC-)Drop-out 432
Max-Norm-Regularisierung 435
Zusammenfassung und praktische Tipps 436
Übungen 438

12 Eigene Modelle und Training mit TensorFlow 441
Ein kurzer Überblick über TensorFlow 441
TensorFlow wie NumPy einsetzen 445
Tensoren und Operationen 445
Tensoren und NumPy 447
Typumwandlung 447
Variablen 448
Andere Datenstrukturen 449
Modelle und Trainingsalgorithmen anpassen 450
Eigene Verlustfunktion 450
Modelle mit eigenen Komponenten sichern und laden 451

Eigene Aktivierungsfunktionen, Initialisierer, Regularisierer und Constraints . . . 453
Eigene Metriken . . . 454
Eigene Schichten . . . 457
Eigene Modelle . . . 460
Verlustfunktionen und Metriken auf Modellinterna basieren lassen . . . 462
Gradienten per Autodiff berechnen . . . 464
Eigene Trainingsschleifen . . . 468
Funktionen und Graphen in TensorFlow . . . 471
AutoGraph und Tracing . . . 474
Regeln für TF Functions . . . 475
Übungen . . . 477

13 Daten mit TensorFlow laden und vorverarbeiten . . . 479
Die tf.data-API . . . 480
Transformationen verketten . . . 481
Daten durchmischen . . . 483
Daten vorverarbeiten . . . 486
Alles zusammenbringen . . . 487
Prefetching . . . 488
Datasets mit tf.keras verwenden . . . 490
Das TFRecord-Format . . . 492
Komprimierte TFRecord-Dateien . . . 492
Eine kurze Einführung in Protocol Buffer . . . 493
TensorFlow-Protobufs . . . 494
Examples laden und parsen . . . 496
Listen von Listen mit dem SequenceExample-Protobuf verarbeiten . . . 497
Vorverarbeitungsschichten von Keras . . . 498
Die Normalization-Schicht . . . 499
Die Discretization-Schicht . . . 501
Die CategoryEncoding-Schicht . . . 502
Die StringLookup-Schicht . . . 503
Die Hashing-Schicht . . . 504
Kategorische Merkmale mit Embeddings codieren . . . 505
Vorverarbeitung von Text . . . 509
Vortrainierte Sprachmodellkomponenten verwenden . . . 511
Vorverarbeitungsschichten für Bilder . . . 512
Das TensorFlow-Datasets-Projekt . . . 513
Übungen . . . 515

14 Deep Computer Vision mit Convolutional Neural Networks 519
Der Aufbau des visuellen Cortex 520
Convolutional Layers 521
Filter 523
Stapeln mehrerer Feature Maps 524
Convolutional Layer mit Keras implementieren 526
Speicherbedarf 530
Pooling Layers 531
Pooling Layer mit Keras implementieren 533
Architekturen von CNNs 535
LeNet-5 538
AlexNet 539
GoogLeNet 542
VGGNet 545
ResNet 545
Xception 549
SENet 550
Weitere erwähnenswerte Architektur 552
Die richtige CNN-Architektur wählen 554
Ein ResNet-34-CNN mit Keras implementieren 555
Vortrainierte Modelle aus Keras einsetzen 556
Vortrainierte Modelle für das Transfer Learning 558
Klassifikation und Lokalisierung 561
Objekterkennung 563
Fully Convolutional Networks 565
You Only Look Once 567
Objektverfolgung 570
Semantische Segmentierung 572
Übungen 575

15 Verarbeiten von Sequenzen mit RNNs und CNNs 577
Rekurrente Neuronen und Schichten 578
Gedächtniszellen 580
Ein- und Ausgabesequenzen 581
RNNs trainieren 582
Eine Zeitserie vorhersagen 583
Die ARMA-Modellfamilie 588
Die Daten für Machine-Learning-Modelle vorbereiten 591
Vorhersage mit einem linearen Modell 595
Vorhersage mit einem einfachen RNN 595
Vorhersage mit einem Deep RNN 597
Multivariate Zeitserien vorhersagen 598

Mehrere Zeitschritte vorhersagen 600
Mit einem Sequence-to-Sequence-Modell vorhersagen 602
Arbeit mit langen Sequenzen 605
Gegen instabile Gradienten kämpfen 605
Das Problem des Kurzzeitgedächtnisses 608
Übungen 616

16 Verarbeitung natürlicher Sprache mit RNNs und Attention 619
Shakespearesche Texte mit einem Character-RNN erzeugen 620
Den Trainingsdatensatz erstellen 621
Das Char-RNN-Modell bauen und trainieren 623
Einen gefälschten Shakespeare-Text erzeugen 625
Zustandsbehaftetes RNN 626
Sentimentanalyse 629
Maskieren 632
Vortrainierte Embeddings wiederverwenden 635
Ein Encoder-Decoder-Netzwerk für die neuronale maschinelle Übersetzung 637
Bidirektionale RNNs 643
Beam Search 645
Attention-Mechanismen 647
Attention Is All You Need: die Transformer-Architektur 651
Eine Lawine an Transformer-Modellen 662
Vision Transformers 666
Die Transformers-Bibliothek von Hugging Face 672
Übungen 676

17 Autoencoder, GANs und Diffusionsmodelle 679
Effiziente Repräsentation von Daten 681
Hauptkomponentenzerlegung mit einem untervollständigen linearen Autoencoder 683
Stacked Autoencoder 684
Einen Stacked Autoencoder mit Keras implementieren 685
Visualisieren der Rekonstruktionen 686
Den Fashion-MNIST-Datensatz visualisieren 687
Unüberwachtes Vortrainieren mit Stacked Autoencoder 688
Kopplung von Gewichten 689
Trainieren mehrerer Autoencoder nacheinander 690
Convolutional Autoencoder 691
Denoising Autoencoders 692
Sparse Autoencoders 694
Variational Autoencoders 697
Fashion-MNIST-Bilder erzeugen 701

Generative Adversarial Networks . . . 702
Schwierigkeiten beim Trainieren von GANs . . . 706
Deep Convolutional GANs . . . 708
Progressive wachsende GANs . . . 712
StyleGANs . . . 714
Diffusionsmodelle . . . 717
Übungen . . . 724

18 Reinforcement Learning . . . 727
Lernen zum Optimieren von Belohnungen . . . 728
Suche nach Policies . . . 729
Einführung in OpenAI Gym . . . 731
Neuronale Netze als Policies . . . 735
Auswerten von Aktionen: das Credit-Assignment-Problem . . . 737
Policy-Gradienten . . . 739
Markov-Entscheidungsprozesse . . . 743
Temporal Difference Learning . . . 748
Q-Learning . . . 749
Erkundungspolicies . . . 750
Approximatives Q-Learning und Deep-Q-Learning . . . 751
Deep-Q-Learning implementieren . . . 752
Deep-Q-Learning-Varianten . . . 757
Feste Q-Wert-Ziele . . . 757
Double DQN . . . 758
Priorisiertes Experience Replay . . . 759
Dueling DQN . . . 760
Überblick über beliebte RL-Algorithmen . . . 761
Übungen . . . 764

19 TensorFlow-Modelle skalierbar trainieren und deployen . . . 767
Ein TensorFlow-Modell ausführen . . . 768
TensorFlow Serving verwenden . . . 769
Einen Vorhersageservice auf Vertex AI erstellen . . . 777
Batch-Vorhersagejobs auf Vertex AI ausführen . . . 785
Ein Modell auf ein Mobile oder Embedded Device deployen . . . 787
Ein Modell auf einer Webseite laufen lassen . . . 791
Mit GPUs die Berechnungen beschleunigen . . . 793
Sich eine eigene GPU zulegen . . . 794
Das GPU-RAM verwalten . . . 796
Operationen und Variablen auf Devices verteilen . . . 798
Paralleles Ausführen auf mehreren Devices . . . 800

Modelle auf mehreren Devices trainieren 803
Parallelisierte Modelle 803
Parallelisierte Daten 805
Mit der Distribution Strategies API auf mehreren Devices trainieren 812
Ein Modell in einem TensorFlow-Cluster trainieren 813
Große Trainingsjobs auf Vertex AI ausführen 817
Hyperparameter auf Vertex AI optimieren 819
Hyperparameter mit Keras Tuner auf Vertex AI optimieren 822
Übungen 823
Vielen Dank! 823

A Checkliste für Machine-Learning-Projekte 825

B Autodiff 831

C Spezielle Datenstrukturen 839

D TensorFlow-Graphen 847

Index 857

Vorwort

Der Machine-Learning-Tsunami

Im Jahr 2006 erschien ein Artikel (*https://homl.info/136*) von Geoffrey Hinton et al.[1], in dem vorgestellt wurde, wie sich ein neuronales Netz zum Erkennen handgeschriebener Ziffern mit ausgezeichneter Genauigkeit (> 98 %) trainieren lässt. Ein Deep Neural Network ist ein (sehr) vereinfachtes Modell unseres zerebralen Kortex, und es besteht aus einer Folge von Schichten mit künstlichen Neuronen. Die Autoren nannten diese Technik »Deep Learning«. Zu dieser Zeit wurde das Trainieren eines Deep-Learning-Netzes im Allgemeinen als unmöglich angesehen,[2] und die meisten Forscher hatten die Idee in den 1990ern aufgegeben. Dieser Artikel ließ das Interesse der wissenschaftlichen Gemeinde wieder aufleben, und schon nach kurzer Zeit zeigten weitere Artikel, dass Deep Learning nicht nur möglich war, sondern umwerfende Dinge vollbringen konnte, zu denen kein anderes Machine-Learning-(ML-)Verfahren auch nur annähernd in der Lage war (mithilfe enormer Rechenleistung und riesiger Datenmengen). Dieser Enthusiasmus breitete sich schnell auf weitere Teilgebiete des Machine Learning aus.

Zehn Jahre später hat Machine Learning ganze Industriezweige erobert: Es ist zu einem Herzstück heutiger Spitzentechnologien geworden und dient dem Ranking von Suchergebnissen im Web, kümmert sich um die Spracherkennung Ihres Smartphones, gibt Empfehlungen für Videos und steuert vielleicht sogar Ihr Auto.

1 Geoffrey Hinton et al., »A Fast Learning Algorithm for Deep Belief Nets«, *Neural Computation* 18 (2006): 1527–1554.

2 Obwohl die Konvolutionsnetze von Yann Lecun bei der Bilderkennung seit den 1990ern gut funktioniert hatten, auch wenn sie nicht allgemein anwendbar waren.

Machine Learning in Ihren Projekten

Deshalb interessieren Sie sich natürlich auch für Machine Learning und möchten an der Party teilnehmen!

Womöglich möchten Sie Ihrem selbst gebauten Roboter einen eigenen Denkapparat geben? Ihn Gesichter erkennen lassen? Oder ihn lernen lassen, herumzulaufen?

Oder vielleicht besitzt Ihr Unternehmen Unmengen an Daten (Logdateien, Finanzdaten, Produktionsdaten, Sensordaten, Hotline-Statistiken, Personalstatistiken und so weiter), und Sie könnten vermutlich einige verborgene Schätze heben, wenn Sie nur wüssten, wo Sie danach suchen müssten. Mit Machine Learning können Sie das Folgende (und noch viel mehr (*https://homl.info/usecases*)) erreichen:

- Kundensegmente finden und für jede Gruppe die beste Marketingstrategie entwickeln.
- Jedem Kunden anhand des Kaufverhaltens ähnlicher Kunden Produktempfehlungen geben.
- Betrügerische Transaktionen mit hoher Wahrscheinlichkeit erkennen.
- Den Unternehmensgewinn im nächsten Jahr vorhersagen.

Was immer der Grund ist, Sie haben beschlossen, Machine Learning zu erlernen und in Ihren Projekten umzusetzen. Eine ausgezeichnete Idee!

Ziel und Ansatz

Dieses Buch geht davon aus, dass Sie noch so gut wie nichts über Machine Learning wissen. Unser Ziel ist es, Ihnen die Grundbegriffe, ein Grundverständnis und die Werkzeuge an die Hand zu geben, mit denen Sie Programme zum *Lernen aus Daten* entwickeln können.

Wir werden eine Vielzahl von Techniken besprechen, von den einfachsten und am häufigsten eingesetzten (wie der linearen Regression) bis zu einigen Deep-Learning-Verfahren, die regelmäßig Wettbewerbe gewinnen. Wir werden dazu für den Produktionsbetrieb geschaffene Python-Frameworks verwenden:

- Scikit-Learn (*http://scikit-learn.org/*) ist sehr einfach zu verwenden, enthält aber effiziente Implementierungen vieler Machine-Learning-Algorithmen. Damit ist es ein großartiger Ausgangspunkt, um Machine Learning zu erlernen. Scikit-Learn wurde von David Cournapeau im Jahr 2007 erstellt und wird mittlerweile von einem Forschungsteam am Nationalen Forschungsinstitut für Informatik und Automatisierung (INRIA) in Frankreich betreut.
- TensorFlow (*http://tensorflow.org/*) ist eine komplexere Bibliothek für verteiltes Rechnen. Mit ihr können Sie sehr große neuronale Netze effizient trainieren und ausführen, indem Sie die Berechnungen auf bis zu Hunderte von Servern mit mehreren GPUs (*Graphics Processing Units*) verlagern. TensorFlow

(TF) wurde von Google entwickelt und läuft in vielen umfangreichen Machine-Learning-Anwendungen. Die Bibliothek wurde im November 2015 als Open Source veröffentlicht, im September 2019 erschien Version 2.0.

- Keras (*https://keras.io/*) ist eine High-Level-Deep-Learning-API, die das Trainieren und Ausführen neuronaler Netze sehr einfach macht. Sie kommt zusammen mit TensorFlow und greift für alle rechenintensiven Aufgaben darauf zurück.

Dieses Buch verfolgt einen praxisorientierten Ansatz, bei dem Sie ein intuitives Verständnis von Machine Learning entwickeln, indem Sie sich mit konkreten Beispielen und ein klein wenig Theorie beschäftigen.

Auch wenn Sie dieses Buch lesen können, ohne Ihren Laptop in die Hand zu nehmen, empfehlen wir Ihnen, mit den Codebeispielen herumzuexperimentieren.

Codebeispiele

Alle Codebeispiele in diesem Buch sind Open Source und stehen online unter *https://github.com/ageron/handson-mls3* als Jupyter Notebooks zur Verfügung. Dabei handelt es sich um interaktive Dokumente mit Texten, Bildern und ausführbaren Codeabschnitten (in unserem Fall Python). Am einfachsten beginnen Sie damit, diese Notebooks mit Google Colab auszuführen. Das ist ein kostenloser Service, der es Ihnen ermöglicht, beliebige Jupyter Notebooks direkt online laufen zu lassen, ohne etwas auf Ihrem Rechner installieren zu müssen. Sie brauchen nur einen Webbrowser und einen Google-Account.

In diesem Buch gehe ich davon aus, dass Sie Google Colab nutzen, aber ich habe die Notebooks auch auf anderen Onlineplattformen wie Kaggle oder Binder getestet, sodass Sie sie dort ebenfalls einsetzen können, wenn Ihnen das lieber ist. Alternativ können Sie die erforderlichen Bibliotheken und Tools (oder das Docker-Image für dieses Buch) direkt auf Ihrem Computer installieren und die Notebooks dort laufen lassen. Die Anweisungen dazu finden Sie unter *https://homl.info/install*.

Verwenden von Codebeispielen

Dieses Buch ist dazu da, Ihnen beim Erledigen Ihrer Arbeit zu helfen. Wollen Sie über die Codebeispiele hinaus zusätzliche Inhalte in einem Umfang verwenden, der die Regeln eines fairen Einsatzes verletzen würde (zum Beispiel der Verkauf oder Vertrieb von Inhalten aus O'Reilly-Büchern oder das Einbinden eines beträchtlichen Teils dieses Buchs in die Dokumentation Ihres Produkts), wenden Sie sich bitte für eine Erlaubnis an *kommentar@oreilly.de*.

Wir freuen uns über Zitate, verlangen diese aber nicht. Ein Zitat enthält Titel, Autor, Verlag und ISBN. Beispiel: »*Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow* von Aurélien Géron, O'Reilly 2023, ISBN 978-3-96009-212-4.«

Voraussetzungen

Dieses Buch geht davon aus, dass Sie ein wenig Programmiererfahrung mit Python haben. Sollten Sie Python noch nicht kennen, ist *http://learnpython.org/* ein ausgezeichneter Ausgangspunkt. Auch das offizielle Tutorial auf python.org (*https://docs.python.org/3/tutorial/*) ist recht gut.

Es setzt ebenfalls voraus, dass Sie mit den wichtigsten wissenschaftlichen Bibliotheken in Python vertraut sind, insbesondere mit NumPy (*http://numpy.org/*), Pandas (*http://pandas.pydata.org/*) und Matplotlib (*http://matplotlib.org/*). Falls Sie diese Bibliotheken noch nie verwendet haben, ist das nicht schlimm – ihr Einsatz lässt sich leicht erlernen, und ich habe für jede von ihnen ein Tutorial erstellt. Diese finden Sie online unter *https://homl.info/tutorials*.

Wenn Sie vollständig verstehen wollen, wie die Algorithmen zum Machine Learning funktionieren (und nicht nur, wie man sie einsetzt), sollten Sie ein Grundverständnis von einigen mathematischen Konzepten haben – insbesondere von linearer Algebra. Sie sollten wissen, was Vektoren und Matrizen sind und wie Sie einige einfache Operationen mit ihnen ausführen, zum Beispiel das Addieren von Vektoren oder das Transponieren und Multiplizieren von Matrizen. Sollten Sie eine kurze Einführung in die lineare Algebra benötigen (sie ist wirklich kein Hexenwerk), habe ich unter *https://homl.info/tutorials* ein Tutorial bereitgestellt. Sie finden dort ebenfalls eines zur Differenzialrechnung, was hilfreich sein kann, um zu verstehen, wie neuronale Netze trainiert werden, aber für das grobe Verstehen der wichtigsten Konzepte ist es nicht unbedingt von entscheidender Bedeutung. Dieses Buch greift gelegentlich auch auf andere mathematische Konzepte zurück, beispielsweise auf Exponentialfunktionen und Logarithmen, ein bisschen Wahrscheinlichkeitstheorie und ein paar Statistikaspekte, aber alles nicht zu ausgefallen. Brauchen Sie dazu Hilfe, schauen Sie mal bei *https://khanacademy.org* vorbei, wo Sie online viele ausgezeichnete und kostenlose Mathematikkurse finden.

Wegweiser durch dieses Buch

Dieses Buch ist in zwei Teile aufgeteilt. Teil I behandelt folgende Themen:

- Was ist Machine Learning? Welche Aufgaben lassen sich damit lösen? Was sind die wichtigsten Kategorien und Grundbegriffe von Machine-Learning-Systemen?
- Die Schritte in einem typischen Machine-Learning-Projekt.

- Lernen durch Anpassen eines Modells an Daten.
- Optimieren einer Kostenfunktion.
- Bearbeiten, Säubern und Vorbereiten von Daten.
- Merkmale auswählen und entwickeln.
- Ein Modell auswählen und dessen Hyperparameter über Kreuzvalidierung optimieren.
- Die Herausforderungen beim Machine Learning, insbesondere Underfitting und Overfitting (das Gleichgewicht zwischen Bias und Varianz).
- Die verbreitetsten Lernalgorithmen: lineare und polynomielle Regression, logistische Regression, *k*-nächste Nachbarn, Support Vector Machines, Entscheidungsbäume, Random Forests und Ensemble-Methoden.
- Dimensionsreduktion der Trainingsdaten, um dem »Fluch der Dimensionalität« etwas entgegenzusetzen.
- Andere Techniken des unüberwachten Lernens, unter anderem Clustering, Dichteabschätzung und Anomalieerkennung.

Teil II widmet sich diesen Themen:

- Was sind neuronale Netze? Wofür sind sie geeignet?
- Erstellen und Trainieren neuronaler Netze mit TensorFlow und Keras.
- Die wichtigsten Architekturen neuronaler Netze: Feed-Forward-Netze für Tabellendaten, Convolutional Neural Networks zur Bilderkennung, rekurrente Netze und Long-Short-Term-Memory-(LSTM-)Netze zur Sequenzverarbeitung, Encoder/Decoder und Transformer für die Sprachverarbeitung (und mehr!), Autoencoder sowie Generative Adversarial Networks (GANs) und Diffusionsmodelle zum generativen Lernen.
- Techniken zum Trainieren von Deep-Learning-Netzen.
- Wie man einen Agenten erstellt (zum Beispiel einen Bot in einem Spiel), der durch Versuch und Irrtum gute Strategien erlernt und dabei Reinforcement Learning einsetzt.
- Effizientes Laden und Vorverarbeiten großer Datenmengen.
- Trainieren und Deployen von TensorFlow-Modellen im großen Maßstab.

Der erste Teil baut vor allem auf Scikit-Learn auf, der zweite Teil verwendet TensorFlow.

Springen Sie nicht zu schnell ins tiefe Wasser: Auch wenn Deep Learning zweifelsohne eines der aufregendsten Teilgebiete des Machine Learning ist, sollten Sie zuerst Erfahrungen mit den Grundlagen sammeln. Außerdem lassen sich die meisten Aufgabenstellungen recht gut mit einfacheren Techniken wie Random Forests und Ensemble-Methoden lösen (die in Teil I besprochen werden). Deep Learning ist am besten für komplexe Aufgaben

wie Bilderkennung, Spracherkennung und Sprachverarbeitung geeignet. Sie müssen aber genug Daten, Rechenleistung und Geduld haben (sofern Sie nicht ein vortrainiertes neuronales Netz nutzen können, wie Sie noch sehen werden).

Änderungen zwischen der ersten und der zweiten Auflage

Haben Sie schon die erste Auflage gelesen, finden Sie hier die wichtigsten Unterschiede zwischen der ersten und zweiten Auflage:

- Der gesamte Code wurde von TensorFlow 1.x auf TensorFlow 2.x gehoben, und ich habe einen Großteil des Low-Level-Codes mit TensorFlow (Graphen, Sessions, Feature Columns, Estimators und so weiter) durch viel einfacheren Keras-Code ersetzt.
- In der zweiten Auflage wurde die Data-API zum Laden und Vorverarbeiten großer Datasets aufgenommen, die Distribution Strategies API zum Trainieren und Deployen von TF-Modellen im großen Maßstab, TF Serving und die Google Cloud API Platform, mit denen Modelle in Produkte umgewandelt werden können, sowie (zumindest angerissen) TF Transform, TFLite, TF Addons/Seq2Seq, TensorFlow.js und TF Agents.
- Es wurden auch viele zusätzliche ML-Themen hinzugenommen, unter anderem ein neues Kapitel zu unüberwachtem Lernen, Computer-Vision-Techniken zur Objekterkennung und zur semantischen Segmentierung, außerdem das Verarbeiten von Sequenzen durch Convolutional Neural Networks (CNNs), die Verarbeitung natürlicher Sprache (Natural Language Processing, NLP) mit rekurrenten neuronalen Netzen (RNNs), CNNs und Transformer, GANs und vieles mehr.

Auf *https://homl.info/changes2* finden Sie mehr Details.

Änderungen zwischen der zweiten und der dritten Auflage

Haben Sie die zweite Auflage gelesen, finden Sie hier die wichtigsten Änderungen zwischen zweiter und dritter Auflage:

- Der gesamte Code wurde an die neuesten Bibliotheksversionen angepasst. Insbesondere wurden in dieser dritten Auflage viele neue Ergänzungen von Scikit-Learn berücksichtigt (zum Beispiel das Feature Name Tracking, histogrammbasiertes Gradient Boosting, Label Propagation und mehr). Zudem wurden die Bibliothek *Keras Tuner* für das Tuning von Hyperparametern, die

Transformers-Bibliothek von Hugging Face für die Verarbeitung natürlicher Sprache sowie die neuen Vorverarbeitungsschichten und Data Augmentation Layer mit aufgenommen.

- Es wurden diverse Vision-Modelle hinzugefügt (ResNeXt, DenseNet, MobileNet, CSPNet und EfficientNet) und dazu Entscheidungshilfen für das Wählen des richtigen Modells.
- In Kapitel 15 werden nun statt generierter Zeitserien die *Chicago Bus and Rail Ridership*-Daten analysiert, und das ARMA-Modell mit seinen Varianten wird vorgestellt.
- Kapitel 16 zur Verarbeitung natürlicher Sprache baut nun ein Modell zum Übersetzen aus dem Englischen ins Spanische, wobei zuerst ein Encoder-Decoder-RNN und dann ein Transformer-Modell zum Einsatz kommen. Das Kapitel behandelt zudem Sprachmodelle wie Switch Transformers, DistilBERT, T5 und PaLM (mit einem Chain-of-Thought Prompting), außerdem stellt es Vision Transformers (ViTs) vor und gibt einen Überblick über ein paar auf Transformern basierende visuelle Modelle, wie zum Beispiel Data-Efficient Image Transformers (DeiTs), Perceivers und DINO. Zudem gibt es einen kurzen Hinweis auf ein paar große, multimodale Modelle – unter anderem CLIP, DALL·E, Flamingo und GATO.
- In Kapitel 17 zu generativem Lernen werden nun Diffusionsmodelle vorgestellt, und es wird gezeigt, wie Sie ein Denoising Diffusion Probabilistic Model (DDPM) von Grund auf implementieren.
- Kapitel 19 ist von der Google Cloud AI Platform nach Google Vertex AI umgezogen und nutzt jetzt verteilte Keras Tuner für die Hyperparametersuche im großen Maßstab. Es führt nun TensorFlow.js-Code ein, mit dem Sie online experimentieren können. Zudem werden zusätzliche verteilte Trainingstechniken vorgestellt, unter anderem PipeDream und Pathways.
- Um all die neuen Inhalte unterbringen zu können, stehen manche Abschnitte nur noch online zur Verfügung, unter anderem die Installationsanweisungen, die Hauptkomponentenzerlegung (PCA), mathematische Details zu bayesschen gaußschen Mischverteilungsmodellen, TF Agents und die früheren Anhänge A (Lösungen zu den Übungsaufgaben), C (die Mathematik von Support Vector Machines) und E (weitere Architekturen für neuronale Netze). Diese drei früheren Anhänge finden Sie auf der deutschen Website zum Buch unter *https://dpunkt.de/produkt/praxiseinstieg-machine-learning-mit-scikit-learn-keras-und-tensorflow-2/* auf der Registerkarte *Zusatzmaterial*.

Auf *https://homl.info/changes3* finden Sie zusätzliche Informationen zu den Änderungen.

Ressourcen im Netz

Es gibt viele ausgezeichnete Ressourcen, mit deren Hilfe sich Machine Learning erlernen lässt. Der ML-Kurs auf Coursera (*https://homl.info/ngcourse*) von Andrew Ng ist faszinierend, auch wenn er einen beträchtlichen Zeitaufwand bedeutet.

Darüber hinaus finden Sie viele interessante Webseiten über Machine Learning, darunter natürlich den ausgezeichneten User Guide (*https://homl.info/skdoc*) von Scikit-Learn. Auch Dataquest (*https://www.dataquest.io/*), das sehr ansprechende Tutorials und ML-Blogs bietet, sowie die auf Quora (*https://homl.info/1*) aufgeführten ML-Blogs könnten Ihnen gefallen.

Natürlich bieten darüber hinaus viele andere Bücher eine Einführung in Machine Learning, insbesondere:

- Joel Grus, *Einführung in Data Science: Grundprinzipien der Datenanalyse mit Python (https://oreilly.de/produkt/einfuehrung-in-data-science-2/)* (O'Reilly, 2. Auflage). Dieses Buch stellt die Grundlagen von Machine Learning vor und implementiert die wichtigsten Algorithmen in reinem Python (von Grund auf).
- Stephen Marsland, *Machine Learning: An Algorithmic Perspective, 2nd Edition* (Chapman & Hall). Dieses Buch ist eine großartige Einführung in Machine Learning, die viele Themen ausführlich behandelt. Es enthält Codebeispiele in Python (ebenfalls von Grund auf, aber mit NumPy).
- Sebastian Raschka, *Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-learn: Das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics* (mitp Professional, 3. Auflage). Eine weitere ausgezeichnete Einführung in Machine Learning. Dieses Buch konzentriert sich auf Open-Source-Bibliotheken in Python (Pylearn 2 und Theano).
- François Chollet, *Deep Learning with Python* (Manning, 2nd Edition). Ein sehr praxisnahes Buch, das klar und präzise viele Themen behandelt – wie Sie es vom Autor der ausgezeichneten Keras-Bibliothek erwarten können. Es zieht Codebeispiele der mathematischen Theorie vor.
- Andriy Burkov, *Machine Learning kompakt: Alles, was Sie wissen müssen* (mitp Professional). Dieses sehr kurze Buch behandelt ein beeindruckendes Themenspektrum gut verständlich, scheut dabei aber nicht vor mathematischen Gleichungen zurück.
- Yaser S. Abu-Mostafa, Malik Magdon-Ismail und Hsuan-Tien Lin, *Learning from Data* (AMLBook). Als eher theoretische Abhandlung von ML enthält dieses Buch sehr tiefgehende Erkenntnisse, insbesondere zum Gleichgewicht zwischen Bias und Varianz (siehe Kapitel 4).
- Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach, 4th Edition* (Pearson). Dieses ausgezeichnete (und umfangreiche) Buch deckt

eine unglaubliche Stoffmenge ab, darunter Machine Learning. Es hilft dabei, ML in einem breiteren Kontext zu betrachten.

- Jeremy Howard and Sylvain Gugger, *Deep Learning for Coders with fastai and PyTorch* (O'Reilly). Eine wunderbar klare und praxisnahe Einführung in Deep Learning mithilfe der fastai- und PyTorch-Bibliotheken.

Eine gute Möglichkeit zum Lernen sind schließlich Webseiten mit ML-Wettbewerben wie Kaggle.com (*https://kaggle.com/*). Dort können Sie Ihre Fähigkeiten an echten Aufgaben üben und Hilfe und Tipps von einigen der besten ML-Profis erhalten.

In diesem Buch verwendete Konventionen

Die folgenden typografischen Konventionen werden in diesem Buch verwendet:

Kursiv
: Kennzeichnet neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateiendungen.

`Konstante Zeichenbreite`
: Wird für Programmlistings und für Programmelemente in Textabschnitten wie Namen von Variablen und Funktionen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter verwendet.

`Konstante Zeichenbreite, fett`
: Kennzeichnet Befehle oder anderen Text, den der Nutzer wörtlich eingeben sollte.

`Konstante Zeichenbreite, kursiv`
: Kennzeichnet Text, den der Nutzer je nach Kontext durch entsprechende Werte ersetzen sollte.

Dieses Symbol steht für einen Tipp oder eine Empfehlung.

Dieses Symbol steht für einen allgemeinen Hinweis.

Dieses Symbol steht für eine Warnung oder erhöhte Aufmerksamkeit.

Danksagungen

In meinen wildesten Träumen hätte ich mir niemals vorgestellt, dass die erste und zweite Auflage dieses Buchs solch eine Verbreitung finden würde. Ich habe so viele Nachrichten von Lesern erhalten – oft mit Fragen, manche mit freundlichen Hinweisen auf Fehler und die meisten mit ermutigenden Worten. Ich kann gar nicht sagen, wie dankbar ich all diesen Lesern für ihre Unterstützung bin. Vielen, vielen Dank! Scheuen Sie sich nicht, sich auf GitHub (*https://homl.info/issues3*) zu melden, wenn Sie Fehler in den Codebeispielen finden (oder einfach etwas fragen wollen) oder um auf Fehler im Text aufmerksam (*https://homl.info/errata3*) zu machen. Manche Leser haben mir auch geschrieben, wie dieses Buch ihnen dabei geholfen hat, ihren ersten Job zu bekommen oder ein konkretes Problem zu lösen, an dem sie gearbeitet haben. Ich finde ein solches Feedback unglaublich motivierend. Hat Ihnen dieses Buch geholfen, würde ich mich freuen, wenn Sie mir Ihre Geschichte erzählen würden – entweder privat (zum Beispiel über LinkedIn (*https://linkedin.com/in/aurelien-geron/*)) oder öffentlich (beispielsweise in einem Tweet an @aureliengeron oder über ein Amazon-Review (*https://homl.info/amazon3*)).

Ein großer Dank geht ebenfalls an all die wundervollen Menschen, die ihre Zeit und Expertise angeboten haben, um diese dritte Auflage zu begutachten, Fehler zu korrigieren und unzählige Vorschläge zu machen. Diese Auflage ist dank ihnen so viel besser geworden: Olzhas Akpambetov, George Bonner, François Chollet, Siddha Gangju, Sam Goodman, Matt Harrison, Sasha Sobran, Lewis Tunstall, Leandro von Werra und mein lieber Bruder Sylvain. Ihr seid toll!

Ebenfalls dankbar bin ich den vielen Menschen, die mich auf meinem Weg unterstützt haben, indem sie meine Fragen beantworteten, Verbesserungen vorschlugen und zum Code auf GitHub beitrugen – insbesondere Yannick Assogba, Ian Beauregard, Ulf Bissbort, Rick Chao, Peretz Cohen, Kyle Gallatin, Hannes Hapke, Victor Khaustov, Soonson Kwon, Eric Lebigot, Jason Mayes, Laurence Moroney, Sara Robinson, Joaquín Ruales und Yuefeng Zhou.

Dieses Buch würde ohne die fantastischen Mitarbeiterinnen und Mitarbeiter von O'Reilly nicht existieren, insbesondere nicht ohne Nicole Taché, die mir sehr hilfreiches Feedback gab und immer aufmunternd, unterstützend und hilfreich war – eine bessere Lektorin könnte ich mir nicht vorstellen. Ein großer Dank geht auch an Michele Cronin, die mich durch die letzten Kapitel begleitet und mich über die Ziellinie gebracht hat. Vielen Dank an das ganze Produktionsteam, insbesondere an Elizabeth Kelly und Kristen Brown. Ebenfalls vielen Dank an Kim Cofer für das Redigieren und an Johnny O'Toole, der sich um die Beziehung zu Amazon kümmerte und viele meiner Fragen beantwortet hat. Danke an Kate Dullea für das großartige Verbessern meiner Illustrationen. Danke an Marie Beaugurea, Ben Lorica, Mike Loukides und Laurel Ruma dafür, dass sie an dieses Projekt geglaubt und seinen Rahmen definiert haben. Danke an Matt Hacker und alle anderen im Atlas-Team, die all meine technischen Fragen in Bezug auf Formatierung, AsciiDoc,

MathML und LaTeX beantworten konnten, und an Nick Adams, Rebecca Demarest, Rachel Head, Judith McConville, Helen Monroe, Karen Montgomery, Rachel Roumeliotis und alle anderen bei O'Reilly, die zu diesem Buch beigetragen haben.

Ich werde nie all die wunderbaren Menschen vergessen, die mir bei der ersten und zweiten Auflage dieses Buchs geholfen haben: Freunde, Kollegen und Experten – einschließlich vieler Mitglieder des TensorFlow-Teams. Die Liste ist lang: Olzhas Akpambetov, Karmel Allison, Martin Andrews, David Andrzejewski, Paige Bailey, Lukas Biewald, Eugene Brevdo, William Chargin, François Chollet, Clément Courbet, Robert Crowe, Mark Daoust, Daniel »Wolff« Dobson, Julien Dubois, Mathias Kende, Daniel Kitachewsky, Nick Felt, Bruce Fontaine, Justin Francis, Goldie Gadde, Irene Giannoumis, Ingrid von Glehn, Vincent Guilbeau, Sandeep Gupta, Priya Gupta, Kevin Haas, Eddy Hung, Konstantinos Katsiapis, Viacheslav Kovalevskyi, Jon Krohn, Allen Lavoie, Karim Matrah, Grégoire Mesnil, Clemens Mewald, Dan Moldovan, Dominic Monn, Sean Morgan, Tom O'Malley, James Pack, Alexander Pak, Haesun Park, Alexandre Passos, Ankur Patel, Josh Patterson, André Susano Pinto, Anthony Platanios, Anosh Raj, Oscar Ramirez, Anna Revinskaya, Saurabh Saxena, Salim Sémaoune, Ryan Sepassi, Vitor Sessak, Jiri Simsa, Iain Smears, Xiaodan Song, Christina Sorokin, Michel Tessier, Wiktor Tomczak, Dustin Tran, Todd Wang, Pete Warden, Rich Washington, Martin Wicke, Edd Wilder-James, Sam Witteveen, Jason Zaman, Yuefeng Zhou und mein Bruder Sylvain.

Schließlich bin ich meiner geliebten Frau Emmanuelle und unseren drei wunderbaren Kindern Alexandre, Rémi und Gabrielle unendlich dafür dankbar, dass sie mich zur Arbeit an diesem Buch ermutigt haben. Ihre unstillbare Neugier war unbezahlbar: Indem ich einige der schwierigsten Konzepte in diesem Buch meiner Frau und meinen Kindern erklärt habe, konnte ich meine Gedanken ordnen und viele Teile direkt verbessern. Und sie haben mir sogar Kekse und Kaffee vorbeigebracht. Was kann man sich noch mehr wünschen?

TEIL I

Die Grundlagen des Machine Learning