

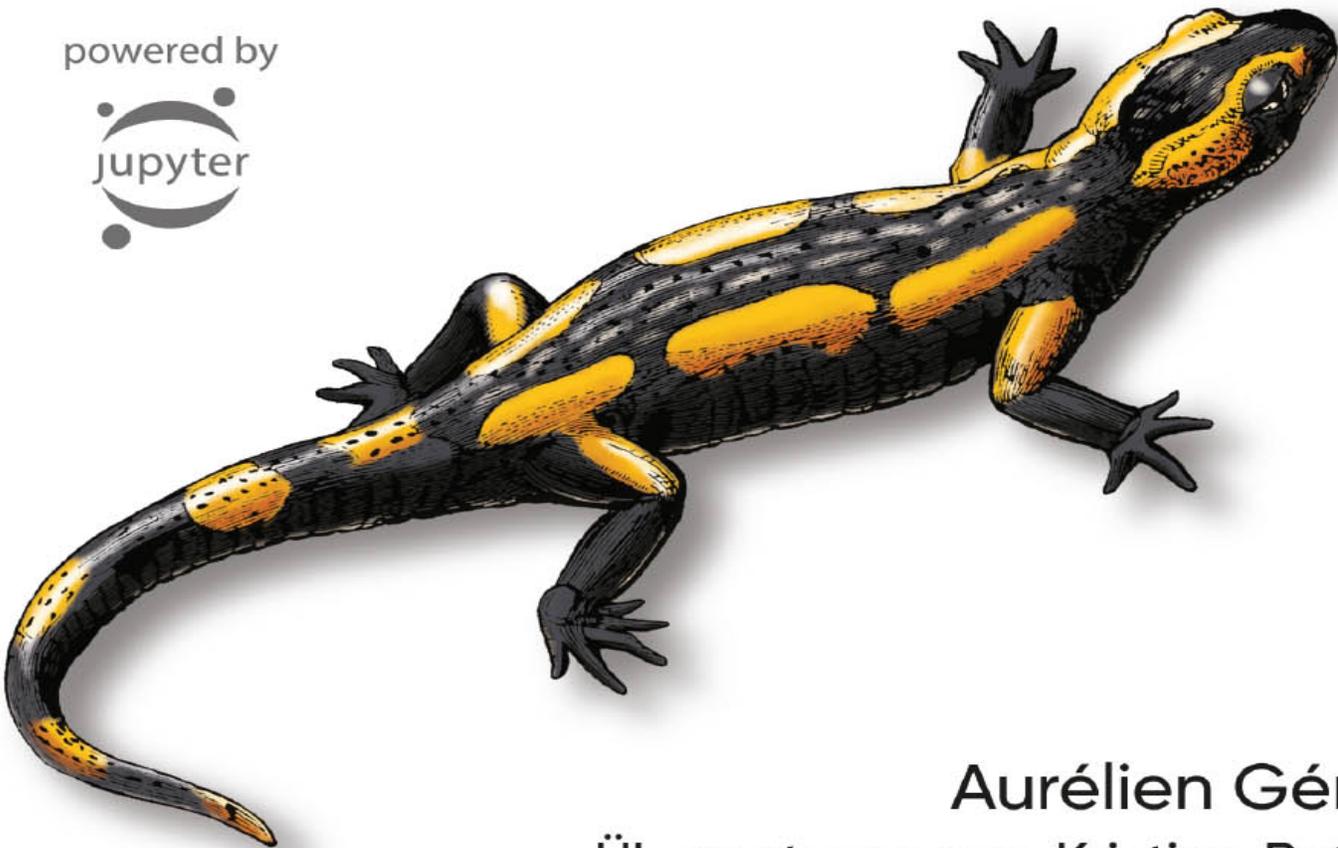
O'REILLY®

2. Auflage
Aktuell zu
TensorFlow 2

Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow

Konzepte, Tools und Techniken
für intelligente Systeme

powered by



Aurélien Géron
Übersetzung von Kristian Rother
und Thomas Demmig

Papier
plus⁺
PDF.

Zu diesem Buch - sowie zu vielen weiteren O'Reilly-Büchern - können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei oreilly.plus⁺:

www.oreilly.plus

2. 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

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 978-3-96009-124-0

PDF 978-3-96010-339-4

ePub 978-3-96010-340-0

mobi 978-3-96010-341-7

2. Auflage

Translation Copyright für die deutschsprachige Ausgabe © 2020 dpunkt.verlag
GmbH

Wieblinger Weg 17

69123 Heidelberg

Authorized German translation of the English edition of *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*, ISBN 9781492032649 © 2019 Kiwisoft S.A.S. This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.



Hinweis:

Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:

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

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag noch Übersetzer können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Vorwort

Teil I Die Grundlagen des Machine Learning

1 Die Machine-Learning-Umgebung

Was ist Machine Learning?

Warum wird Machine Learning verwendet?

Anwendungsbeispiel

Unterschiedliche Machine-Learning-Systeme

Überwachtes/unüberwachtes Lernen

Batch- und Online-Learning

Instanzbasiertes versus modellbasiertes Lernen

Die wichtigsten Herausforderungen beim Machine Learning

Unzureichende Menge an Trainingsdaten

Nicht repräsentative Trainingsdaten

Minderwertige Daten

Irrelevante Merkmale

Overfitting der Trainingsdaten

Underfitting der Trainingsdaten

Zusammenfassung

Testen und Validieren

Hyperparameter anpassen und Modellauswahl

Datendiskrepanz

Übungen

2 Ein Machine-Learning-Projekt von A bis Z

Der Umgang mit realen Daten

Betrachte das Gesamtbild

- Die Aufgabe abstecken

- Wähle ein Qualitätsmaß aus

- Überprüfe die Annahmen

Beschaffe die Daten

- Erstelle eine Arbeitsumgebung

- Die Daten herunterladen

- Wirf einen kurzen Blick auf die Datenstruktur

- Erstelle einen Testdatensatz

Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

- Visualisieren geografischer Daten

- Suche nach Korrelationen

- Experimentieren mit Kombinationen von Merkmalen

Bereite die Daten für Machine-Learning-Algorithmen vor

- Aufbereiten der Daten

- Bearbeiten von Text und kategorischen Merkmalen

- Eigene Transformer

- Skalieren von Merkmalen

- Pipelines zur Transformation

Wähle ein Modell aus und trainiere es

- Trainieren und Auswerten auf dem

- Trainingsdatensatz

- Bessere Auswertung mittels Kreuzvalidierung

Optimiere das Modell

- Gittersuche

- Zufällige Suche

- Ensemble-Methoden

- Analysiere die besten Modelle und ihre Fehler

- Evaluieren das System auf dem Testdatensatz

Nimm das System in Betrieb, überwache und warte es

Probieren Sie es aus!

Übungen

3 Klassifikation

MNIST

Trainieren eines binären Klassifikators

Qualitätsmaße

- Messen der Genauigkeit über Kreuzvalidierung

- Konfusionsmatrix

- Relevanz und Sensitivität

- Die Wechselbeziehung zwischen Relevanz und

- Sensitivität

- Die ROC-Kurve

Klassifikatoren mit mehreren Kategorien

Fehleranalyse

Klassifikation mit mehreren Labels

Klassifikation mit mehreren Ausgaben

Übungen

4 Trainieren von Modellen

Lineare Regression

- Die Normalengleichung

- Komplexität der Berechnung

Das Gradientenverfahren

- Batch-Gradientenverfahren

- Stochastisches Gradientenverfahren

- Mini-Batch-Gradientenverfahren

Polynomielle Regression

Lernkurven

Regularisierte lineare Modelle

- Ridge-Regression

- Lasso-Regression

- Elastic Net

- Early Stopping

Logistische Regression

- Abschätzen von Wahrscheinlichkeiten

- Trainieren und Kostenfunktion

- Entscheidungsgrenzen

- Softmax-Regression

Übungen

5 Support Vector Machines

Lineare Klassifikation mit SVMs

Soft-Margin-Klassifikation

Nichtlineare SVM-Klassifikation

Polynomieller Kernel

Ähnlichkeitsbasierte Merkmale

Der gaußsche RBF-Kernel

Komplexität der Berechnung

SVM-Regression

Hinter den Kulissen

Entscheidungsfunktion und Vorhersagen

Zielfunktionen beim Trainieren

Quadratische Programme

Das duale Problem

Kernel-SVM

Online-SVMs

Übungen

6 Entscheidungsbäume

Trainieren und Visualisieren eines Entscheidungsbaums

Vorhersagen treffen

Schätzen von Wahrscheinlichkeiten für Kategorien

Der CART-Trainingsalgorithmus

Komplexität der Berechnung

Gini-Unreinheit oder Entropie?

Hyperparameter zur Regularisierung

Regression

Instabilität

Übungen

7 Ensemble Learning und Random Forests

Abstimmverfahren unter Klassifikatoren

Bagging und Pasting

Bagging und Pasting in Scikit-Learn

- Out-of-Bag-Evaluation
- Zufällige Patches und Subräume
- Random Forests
 - Extra-Trees
 - Wichtigkeit von Merkmalen
- Boosting
 - AdaBoost
 - Gradient Boosting
- Stacking
- Übungen

8 Dimensionsreduktion

- Der Fluch der Dimensionalität
- Die wichtigsten Ansätze zur Dimensionsreduktion
 - Projektion
 - Manifold Learning
- Hauptkomponentenzerlegung (PCA)
 - Erhalten der Varianz
 - Hauptkomponenten
 - Die Projektion auf d Dimensionen
 - Verwenden von Scikit-Learn
 - Der Anteil erklärter Varianz
 - Auswählen der richtigen Anzahl Dimensionen
 - PCA als Komprimierungsverfahren
 - Randomisierte PCA
 - Inkrementelle PCA
- Kernel-PCA
 - Auswahl eines Kernels und Optimierung der Hyperparameter
- LLE
- Weitere Techniken zur Dimensionsreduktion
- Übungen

9 Techniken des unüberwachten Lernens

- Clustering
 - K-Means

- Grenzen von K-Means
- Bildsegmentierung per Clustering
- Vorverarbeitung per Clustering
- Clustering für teilüberwachtes Lernen einsetzen
- DBSCAN
- Andere Clustering-Algorithmen
- Gaußsche Mischverteilung
 - Anomalieerkennung mit gaußschen Mischverteilungsmodellen
 - Die Anzahl an Clustern auswählen
 - Bayessche gaußsche Mischverteilungsmodelle
 - Andere Algorithmen zur Anomalie- und Novelty-Erkennung
- Übungen

Teil II Neuronale Netze und Deep Learning

10 Einführung in künstliche neuronale Netze mit Keras

- Von biologischen zu künstlichen Neuronen
 - Biologische Neuronen
 - Logische Berechnungen mit Neuronen
 - Das Perzeptron
 - Mehrschichtiges Perzeptron und Backpropagation
 - Regressions-MLPs
 - Klassifikations-MLPs
- MLPs mit Keras implementieren
 - TensorFlow 2 installieren
 - Einen Bildklassifikator mit der Sequential API erstellen
 - Ein Regressions-MLP mit der Sequential API erstellen
 - Komplexe Modelle mit der Functional API bauen
 - Dynamische Modelle mit der Subclassing API bauen
 - Ein Modell sichern und wiederherstellen

- Callbacks
- TensorBoard zur Visualisierung verwenden
- Feinabstimmung der Hyperparameter eines neuronalen Netzes
 - Anzahl verborgener Schichten
 - Anzahl Neuronen pro verborgene Schicht
 - Lernrate, Batchgröße und andere Hyperparameter
- Übungen

11 Trainieren von Deep-Learning-Netzen

- Das Problem schwindender/explodierender Gradienten
- Initialisierung nach Glorot und He
- Nicht sättigende Aktivierungsfunktionen
- Batchnormalisierung
- Gradient Clipping
- Wiederverwenden vortrainierter Schichten
- Transfer Learning mit Keras
- Unüberwachtes Vortrainieren
- Vortrainieren anhand einer Hilfsaufgabe
- Schnellere Optimierer
 - Momentum Optimization
 - Beschleunigter Gradient nach Nesterov
 - AdaGrad
 - RMSProp
 - Adam-Optimierung
 - Scheduling der Lernrate
- Vermeiden von Overfitting durch Regularisierung
 - ℓ_1 - und ℓ_2 -Regularisierung
 - Drop-out
 - Monte-Carlo-(MC-)-Drop-out
 - Max-Norm-Regularisierung
- Zusammenfassung und praktische Tipps
- Übungen

12 Eigene Modelle und Training mit TensorFlow

- Ein kurzer Überblick über TensorFlow

- TensorFlow wie NumPy einsetzen
 - Tensoren und Operationen
 - Tensoren und NumPy
 - Typumwandlung
 - Variablen
 - Andere Datenstrukturen
- Modelle und Trainingsalgorithmen anpassen
 - Eigene Verlustfunktion
 - Modelle mit eigenen Komponenten sichern und laden
 - Eigene Aktivierungsfunktionen, Initialisierer, Regularisierer und Constraints
 - Eigene Metriken
 - Eigene Schichten
 - Eigene Modelle
 - Verlustfunktionen und Metriken auf Modell-Internas basieren lassen
 - Gradienten per Autodiff berechnen
 - Eigene Trainingsschleifen
- Funktionen und Graphen in TensorFlow
 - AutoGraph und Tracing
 - Regeln für TF Functions
- Übungen

13 Daten mit TensorFlow laden und vorverarbeiten

- Die Data-API
 - Transformationen verketteten
 - Daten durchmischen
 - Daten vorverarbeiten
 - Alles zusammenbringen
 - Prefetching
 - Datasets mit tf.keras verwenden
- Das TFRecord-Format
 - Komprimierte TFRecord-Dateien
 - Eine kurze Einführung in Protocol Buffer
 - TensorFlow-Protobufs

- Examples laden und parsen
- Listen von Listen mit dem SequenceExample-Protobuf verarbeiten
- Die Eingabemerkmale vorverarbeiten
 - Kategorische Merkmale mit One-Hot-Vektoren codieren
 - Kategorische Merkmale mit Embeddings codieren
- Vorverarbeitungsschichten von Keras
- TF Transform
- Das TensorFlow-Datasets-(TFDS-)Projekt
- Übungen

14 Deep Computer Vision mit Convolutional Neural Networks

- Der Aufbau des visuellen Cortex
- Convolutional Layers
 - Filter
 - Stapeln mehrerer Feature Maps
 - Implementierung in TensorFlow
 - Speicherbedarf
- Pooling Layers
 - Implementierung in TensorFlow
- Architekturen von CNNs
 - LeNet-5
 - AlexNet
 - GoogLeNet
 - VGGNet
 - ResNet
 - Xception
 - SENet
- Ein ResNet-34-CNN mit Keras implementieren
- Vortrainierte Modelle aus Keras einsetzen
- Vortrainierte Modelle für das Transfer Learning
- Klassifikation und Lokalisierung
- Objekterkennung
 - Fully Convolutional Networks

You Only Look Once (YOLO)
Semantische Segmentierung
Übungen

15 Verarbeiten von Sequenzen mit RNNs und CNNs

Rekurrente Neuronen und Schichten

Gedächtniszellen

Ein- und Ausgabesequenzen

RNNs trainieren

Eine Zeitserie vorhersagen

Grundlegende Metriken

Ein einfaches RNN implementieren

Deep RNNs

Mehrere Zeitschritte vorhersagen

Arbeit mit langen Sequenzen

Gegen instabile Gradienten kämpfen

Das Problem des Kurzzeitgedächtnisses

Übungen

16 Natürliche Sprachverarbeitung mit RNNs und Attention

Shakespearesche Texte mit einem Character-RNN erzeugen

Den Trainingsdatensatz erstellen

Wie ein sequenzieller Datensatz aufgeteilt wird

Den sequenziellen Datensatz in mehrere Fenster unterteilen

Das Char-RNN-Modell bauen und trainieren

Das Char-RNN-Modell verwenden

Einen gefälschten Shakespeare-Text erzeugen

Zustandsbehaftetes RNN

Sentimentanalyse

Maskieren

Vortrainierte Embeddings wiederverwenden

Ein Encoder-Decoder-Netzwerk für die neuronale maschinelle Übersetzung

- Bidirektionale RNNs
- Beam Search
- Attention-Mechanismen
 - Visuelle Attention
 - Attention Is All You Need: Die Transformer-Architektur
- Aktuelle Entwicklungen bei Sprachmodellen
- Übungen

17 Representation Learning und Generative Learning mit Autoencodern und GANs

- Effiziente Repräsentation von Daten
- Hauptkomponentenzerlegung mit einem unternvollständigen linearen Autoencoder
- Stacked Autoencoder
 - Einen Stacked Autoencoder mit Keras implementieren
 - Visualisieren der Rekonstruktionen
 - Den Fashion-MNIST-Datensatz visualisieren
 - Unüberwachtes Vortrainieren mit Stacked Autoencoder
 - Kopplung von Gewichten
 - Trainieren mehrerer Autoencoder nacheinander
- Convolutional Autoencoder
- Rekurrente Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Variational Autoencoder
 - Fashion-MNIST-Bilder erzeugen
- Generative Adversarial Networks
 - Schwierigkeiten beim Trainieren von GANs
 - Deep Convolutional GANs
 - Progressive wachsende GANs
 - StyleGANs
- Übungen

18 Reinforcement Learning

Lernen zum Optimieren von Belohnungen

Suche nach Policies

Einführung in OpenAI Gym

Neuronale Netze als Policies

Auswerten von Aktionen: Das Credit-Assignment-Problem

Policy-Gradienten

Markov-Entscheidungsprozesse

Temporal Difference Learning

Q-Learning

- Erkundungspolicies

- Approximatives Q-Learning und Deep-Q-Learning

Deep-Q-Learning implementieren

Deep-Q-Learning-Varianten

- Feste Q-Wert-Ziele

- Double DQN

- Priorisiertes Experience Replay

- Dueling DQN

Die TF-Agents-Bibliothek

- TF-Agents installieren

- TF-Agents-Umgebungen

- Umgebungsspezifikationen

- Umgebungswrapper und Atari-Vorverarbeitung

- Trainingsarchitektur

- Deep-Q-Netz erstellen

- DQN-Agenten erstellen

- Replay Buffer und Beobachter erstellen

- Trainingsmetriken erstellen

- Collect-Fahrer erstellen

- Dataset erstellen

- Trainingsschleife erstellen

Überblick über beliebte RL-Algorithmen

Übungen

19 TensorFlow-Modelle skalierbar trainieren und deployen

Ein TensorFlow-Modell ausführen

- TensorFlow Serving verwenden

- Einen Vorhersageservice auf der GCP AI Platform erstellen

- Den Vorhersageservice verwenden

Ein Modell auf ein Mobile oder Embedded Device deployen

Mit GPUs die Berechnungen beschleunigen

- Sich eine eigene GPU zulegen

- Eine mit GPU ausgestattete virtuelle Maschine einsetzen

- Colaboratory

- Das GPU-RAM verwalten

- Operationen und Variablen auf Devices verteilen

- Paralleles Ausführen auf mehreren Devices

Modelle auf mehreren Devices trainieren

- Parallelisierte Modelle

- Parallelisierte Daten

- Mit der Distribution Strategies API auf mehreren Devices trainieren

- Ein Modell in einem TensorFlow-Cluster trainieren

- Große Trainingsjobs auf der Google Cloud AI Platform ausführen

- Black Box Hyperparameter Tuning auf der AI Platform

Übungen

Vielen Dank!

A Lösungen zu den Übungsaufgaben

B Checkliste für Machine-Learning-Projekte

C Das duale Problem bei SVMs

D Autodiff

E Weitere verbreitete Architekturen neuronaler Netze

F Spezielle Datenstrukturen

G TensorFlow-Graphen

Index

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 schlägt den Weltmeister im Brettspiel Go. Über kurz oder lang wird ML vermutlich auch Ihr Auto steuern.

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 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, beispielsweise:

- 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.

Anstatt eigene Übungsversionen jedes Algorithmus zu entwickeln, werden wir 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.
- 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 großflächigen Machine-Learning-Anwendungen. Die Bibliothek wurde im November 2015 als Open Source veröffentlicht.
- Keras (<https://keras.io/>) ist eine High-Level-Deep-Learning-API, die das Trainieren und Ausführen neuronaler Netze sehr einfach macht. Sie kann auf TensorFlow, Theano oder Microsoft Cognitive Toolkit (früher bekannt als CNTK) aufsetzen. TensorFlow bringt seine eigene Implementierung dieser API namens *tf.keras* mit, die einige der

fortgeschritteneren TensorFlow-Features unterstützt (zum Beispiel die Möglichkeit, Daten effizient zu laden).

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 als Jupyter-Notebooks unter <https://github.com/ageron/handson-ml2> verfügbaren Codebeispielen herumzuexperimentieren.

Voraussetzungen

Dieses Buch geht davon aus, dass Sie ein wenig Programmiererfahrung mit Python haben und 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/>).

Wenn Sie sich dafür interessieren, was hinter den Kulissen passiert, sollten Sie ein Grundverständnis von Oberstufenmathematik haben (Analysis, lineare Algebra, Wahrscheinlichkeiten und Statistik).

Sollten Sie Python noch nicht kennen, ist <http://learnpython.org/> ein ausgezeichneter Ausgangspunkt. Das offizielle Tutorial auf [python.org](https://docs.python.org/3/tutorial/) (<https://docs.python.org/3/tutorial/>) ist ebenfalls recht gut.

Falls Sie Jupyter noch nie verwendet haben, führt Sie [Kapitel 2](#) durch die Installation und die Grundlagen: Es ist ein leistungsfähiges Werkzeug in Ihrem Werkzeugkasten.

Und für den Fall, dass Sie mit den wissenschaftlichen Bibliotheken für Python nicht vertraut sind, beinhalten die

mitgelieferten Jupyter-Notebooks einige Tutorials. Es gibt dort auch ein kurzes Mathematiktutorial über lineare Algebra.

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? Welches 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, Autoencoder und Generative Adversarial Networks (GANs) 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, vorausgesetzt, Sie haben genug Daten und Geduld.

Änderungen in der zweiten Auflage

Diese zweite Auflage hat sechs zentrale Ziele:

1. Die Behandlung zusätzlicher ML-Themen: weitere Techniken zum unüberwachten Lernen (unter anderem Clustering, Anomalieerkennung, Dichteabschätzung und Mischmodelle), zusätzliche Techniken zum Trainieren von Deep Networks (einschließlich sich selbst normalisierender Netze), weitere Techniken der Bilderkennung (unter anderem Xception, SEnet, Objekterkennung mit YOLO und semantische Segmentierung mit R-CNN), das Verarbeiten von Sequenzen mit Convolutional Neural Networks (CNNs, einschließlich WaveNet), Verarbeitung natürlicher Sprache mit rekurrenten neuronalen Netzen (RNNs), CNNs und Transformers, GANs.
2. Zusätzliche Bibliotheken und APIs (Keras, die Data-API, TF-Agents für das Reinforcement Learning) sowie das Trainieren und Deployen von TF-Modellen im großen Maßstab mithilfe der Distribution Strategies API, TF Serving und Google Cloud AI Platform; zudem eine kurze Vorstellung von TF Transform, TFLite, TF Addons/Seq2Seq und TensorFlow.js.
3. Vorstellen einiger der neuesten Forschungsergebnisse aus dem Deep Learning.
4. Migrieren aller TensorFlow-Kapitel nach TensorFlow 2 und Verwenden der TensorFlow-Implementierung der Keras-API (tf.keras), wann immer das möglich ist.
5. Aktualisieren der Codebeispiele auf die neuesten Versionen von Scikit-Learn, NumPy, pandas, Matplotlib und anderer Bibliotheken.

6. Anpassen einiger Abschnitte zum besseren Verständnis und Beheben von Fehlern dank sehr vieler Rückmeldungen von Lesern.

Es wurden ein paar Kapitel hinzugefügt, andere wurden umgeschrieben, und ein paar wurden neu angeordnet. Unter <https://homl.info/changes2> finden Sie detailliertere Angaben darüber, was sich in der zweiten Auflage geändert hat.

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 (in Monaten).

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. Schließlich sind auf der Deep-Learning-Website (<http://deeplearning.net/>) Ressourcen aufgezählt, mit denen Sie mehr lernen können.

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

- Joel Grus, *Einführung in Data Science: Grundprinzipien der Datenanalyse mit Python* (<https://www.oreilly.de/buecher/13335/9783960091233-einf%C3%BChrung-in-data-science.html>) (O'Reilly). Dieses Buch stellt die Grundlagen von Machine Learning vor und implementiert die

wichtigsten Algorithmen in reinem Python (von null auf).

- Stephen Marsland, *Machine Learning: An Algorithmic Perspective* (Chapman & Hall). Dieses Buch ist eine großartige Einführung in Machine Learning, die viele Themen ausführlich behandelt. Es enthält Codebeispiele in Python (ebenfalls von null auf, aber mit NumPy).
- Sebastian Raschka, *Machine Learning mit Python und Scikit-Learn und TensorFlow: Das umfassende Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning* (mitp Professional). 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 mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek* (mitp Professional). 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-Ismael 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, 3rd 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.

Eine gute Möglichkeit zum Lernen sind schließlich Webseiten mit ML-Wettbewerben wie [Kaggle.com](https://www.kaggle.com) (<https://www.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.

Codebeispiele

Es gibt eine Reihe von Jupyter-Notebooks voll mit zusätzlichem Material, wie Codebeispielen und Übungen, die zum Herunterladen unter <https://github.com/ageron/handson-ml2> bereitstehen.

Einige der Codebeispiele im Buch lassen sich wiederholende Abschnitte oder Details weg, die offensichtlich sind oder nichts mit Machine Learning zu tun haben. Das sorgt dafür, dass Sie sich auf die wichtigen Teile des Codes konzentrieren können, und spart Platz, um mehr Themen unterzubringen. Wollen Sie sich die vollständigen Codebeispiele betrachten, finden Sie diese in den Jupyter-Notebooks.

Geben die Codebeispiele etwas aus, wird dies mit Python-Prompts (>>> und ...) wie in einer Python-Shell dargestellt,