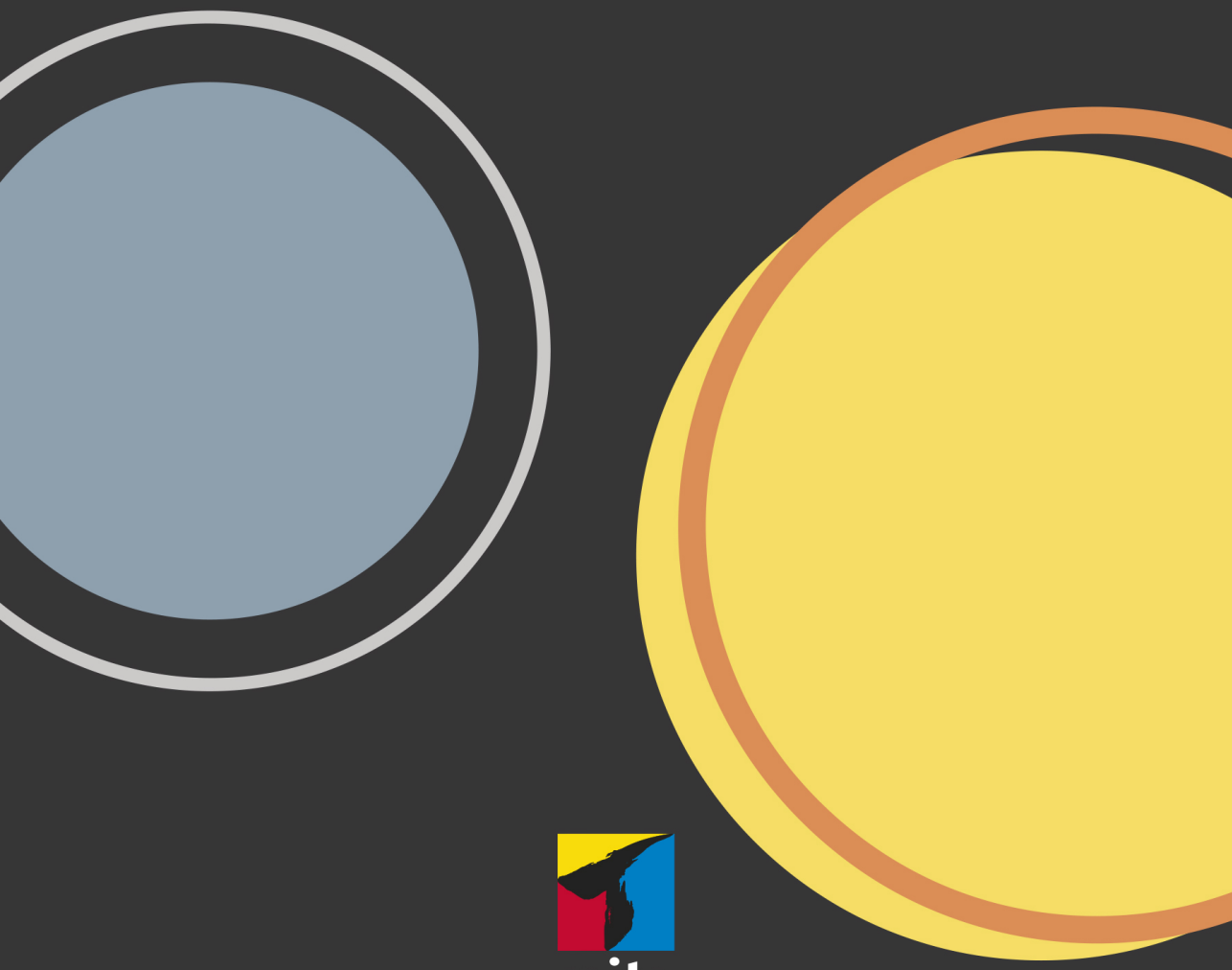
Deutsche Ausgabe des Buches »The Hundred-Page Machine Learning Book«

Andriy Burkov

MACHINE LEARNING KOMPAKT

ALLES, WAS SIE WISSEN MÜSSEN



mitp

»Eine tolle Einführung in Machine Learning von einem erstklassigen Fachmann.«

— **Karolis Urbonas**, Leiter Data Science bei Amazon

»Ich wünschte, es hätte ein solches Buch gegeben, als ich mich als Student der Statistik mit Machine Learning beschäftigt habe.«

— **Chao Han**, Vizepräsident, Leiter Forschung und Entwicklung bei Lucidworks

»Andriys Buch gibt von der ersten Seite an Gas und kommt sofort zum Wesentlichen.«

— **Sujeet Varakhedi**, technischer Leiter bei eBay

»Ein tolles Buch für Entwickler, die in der alltäglichen Arbeit Machine Learning einsetzen wollen, ohne sich ewig damit beschäftigen zu müssen.«

— **Deepak Agarwal**, Vizepräsident KI-Abteilung bei LinkedIn

»Ein ausgezeichneter Einstieg in Machine Learning.«

— **Vincent Pollet**, Forschungsleiter bei Nuance

Andriy Burkov

Machine Learning kompakt

Alles, was Sie wissen müssen

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 978-3-95845-996-0
1. Auflage 2019

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Authorized German translation from the English language edition, entitled
THE HUNDRED-PAGE MACHINE LEARNING BOOK, ISBN 978-1-9995795-0-0

Lektorat: Sabine Schulz, Janina Bahlmann, Lisa Kresse
Sprachkorrektorat: Anna Ueltgesforth
Fachkorrektur: Stefan Niedergriese, Alexander Loth (Autor des Buches *Datenvisualisierung mit Tableau* / mitp)
Covergestaltung: Andriy Burkov / Canva.com
Satz: Dr. Joachim Schlosser, www.schlosser.info

klimaneutral
natureOffice.com | DE-077-529981
gedruckt

Inhaltsverzeichnis

Vorwort zur deutschen Ausgabe

Im Laufe des letzten Jahrhunderts haben wir die Kunst der Computerprogrammierung entwickelt und perfektioniert, um alltägliche Aufgaben zu automatisieren. Herkömmliche Programme basieren jedoch immer noch auf handgefertigten Regeln, sodass die Programmierung selbst zu einer komplizierten und langwierigen Aufgabe wird. Angenommen Sie möchten einen E-Mail-Spam-Filter entwickeln: Bei der herkömmlichen Programmierung müssten Sie Millionen von E-Mails durchlesen, um Regeln abzuleiten, die lästige Junkmails mehr oder weniger zuverlässig herausfiltern. Sicherlich muss es einen effizienteren Weg geben, und darum geht es beim maschinellen Lernen. Im Wesentlichen handelt es sich bei Machine Learning um eine Reihe von Techniken, mit deren Hilfe Computer lernen, aus Daten Vorhersageregeln abzuleiten, und diese automatisch in Programme umwandeln.

Nun leben wir in einem Zeitalter, in dem Machine Learning nicht mehr aus dem Alltag wegzudenken ist. Sei es Gesichtserkennung auf dem iPhone, optische Zeichenerkennung bei der Post, Internetsuchmaschinen, Empfehlungsdienste von digitalen Verkaufsstellen oder der klassische E-Mail-Spamfilter. Während es leichtfällt, diese Liste unendlich weiterzuführen, fällt es mir schwer, an eine moderne Technologie zu denken, welche nicht in irgendeiner Weise Machine Learning integriert.

Für die weitere Integration von Machine Learning in Produkte unseres Alltags ist Machine-Learning-Talent gefragt wie nie zu vor. Die schiere Menge an neuen Jobausschreibungen, die nach Data Scientists oder Machine-Learning-Wissenschaftlern oder -Ingenieuren suchen, ist enorm. Aber auch in verwandten Berufen wie Softwareentwickler oder Informatiker sind Machine-Learning-Kenntnisse mehr und mehr gefragt. Diese sind vor allem für Entwickler wichtig, um erkennen zu können, wo und wann es Sinn macht, Machine Learning in einem Produkt zu verwenden. Aber auch für Nutzer von Produkten, die auf Machine Learning basieren, wird es zunehmend wichtiger, sich mit dieser Technologie auszukennen, um auch ihre Limitierungen zu kennen, und um die eigenen Verhaltensweisen und Er-

wartungen gegebenenfalls an diese neuen Technologien anzupassen. Daher halte ich es für unerlässlich, Ressourcen zu schaffen, die diese Technologien einer breiten Zielgruppe zugänglich machen.

Als Andriy mir von seinem Plan erzählte, ein klar strukturiertes, aber absichtlich relativ kurz gehaltenes Buch als Einführung in das Thema Machine Learning zu verfassen, war ich zunächst skeptisch. Ich fragte mich, ob es tatsächlich möglich sei, alle essentiellen Grundkenntnisse in einer brauchbaren Weise auf weniger als 200 Seiten zu kommunizieren. Als ich einige Zeit später eine Kopie dieses Werks in meinen Händen hielt, musste ich zugeben, dass Andriy dieses Kunststück mit großer Bravour gelungen ist.

Als ich vor ungefähr 4 Jahren das Buch Machine Learning mit Python verfasst habe, war es mitunter das erste, das nicht auf purer akademischer Theorie basierte, sondern auch praktische Code-Beispiele beinhaltete, aber zugleich auch mehr theoretische Tiefe hatte als die herkömmlichen, fast ausschließlich aus Code-Beispielen bestehenden Lehrbücher. Ich habe dieses Lehrbuch mit keinerlei großen Erwartungen geschrieben und war positiv überrascht, dass dieser Spagat aus Theorie und Praxis eine so große Leserbasis mit weltweit über 60.000 verkauften Exemplaren erreicht hat. Obwohl Machine Learning mit Python in der Hinsicht ein großer Erfolg war, Machine Learning einer breiten Zielgruppe zugänglich zu machen, hat es meiner Ansicht nach zwei Schwachpunkte: Mit mehr als 500 Seiten stellt es den Lesern eine hohe Einstiegshürde, und außerdem ist Python nicht für jedermann, da es Dutzende anderer Programmiersprachen gibt, die je nach Anwendungsbereich bevorzugt werden.

Meiner Ansicht nach stellt der bewusste Verzicht von expliziten Code-Beispielen in Andriys Buch einen absoluten Pluspunkt dar. Diese Herangehensweise macht dieses Buch nicht nur zu einer zeitlosen Ressource, da sich evolvierende Code-Bibliotheken stetig verändern und schnell nicht mehr aktuell sind, sondern es hilft auch dabei, sich zunächst auf das Wesentliche zu konzentrieren, bevor Sie sich an praktischen Beispielen in Ihrer bevorzugten Programmiersprache versuchen.

Um mich im Sinne dieses Buches kurz zu halten, möchte ich Sie nicht weiter von Ihrer bevorstehenden und aufregenden Reise in den Bereich des Machine Learnings abhalten. Ich hoffe, Ihnen gefällt dieses Buch mindestens genauso gut wie mir, und ich wünsche Ihnen ein erfreuliches und angenehmes Lernen!

Dr. Sebastian Raschka
Asst. Professor of Statistics, University of Wisconsin-Madison, Autor des Bestsellers ***Machine Learning mit Python*** (mitp-Verlag, deutsche Ausgabe des Buches *Python Machine Learning*)

Vorwort

In den letzten zwanzig Jahren sind die verfügbaren Datenmengen explosionsartig gewachsen, und dementsprechend hat auch das Interesse an statistischen und Machine-Learning-Anwendungen zugenommen. Das hatte tiefgreifende Auswirkungen. Vor zehn Jahren waren meine Kollegen erstaunt, dass mein Wahlfachkurs Statistik für Studenten der Wirtschaftswissenschaften voll belegt war, denn die meisten Wahlveranstaltungen waren schlecht besucht. Heute bieten wir dafür einen Masterstudiengang an, den größten spezialisierten Studiengang der Universität, und die Zahl der Anmeldungen kann sich mit derjenigen für Betriebswirtschaftslehre messen. Das Studienangebot wurde drastisch erhöht, dennoch beklagen sich die Studenten, dass die Kurse überfüllt sind. Aber auch Data-Science- und Machine-Learning-Kurse erfreuen sich außerordentlicher Beliebtheit, denn die Nachfrage nach auf diesen Gebieten ausgebildeten Absolventen hat stark zugenommen.

Dieser Nachfrage liegt eine einfache, aber unbestreitbare Tatsache zugrunde. Machine-Learning-Ansätze haben in den verschiedensten Bereichen für neue Erkenntnisse gesorgt, wie etwa in den Sozialwissenschaften, in der Wirtschaft, in der Biologie und in der Medizin, um nur einige zu nennen. Das hat zu einer enormen Nachfrage nach entsprechend qualifizierten Absolventen geführt. Die Ausbildung der Studenten war dennoch eine Herausforderung, weil der Großteil der verfügbaren Literatur für Akademiker gedacht war und sich auf statistische und theoretische Eigenschaften der Anpassungsalgorithmen oder der resultierenden Schätzer konzentrierte. Forscher und Praktiker, die bei der Implementierung eines vorgegebenen Verfahrens oder bei praktischen Aufgaben nach Unterstützung suchten, wurden kaum berücksichtigt. Die Betroffenen mussten wissen, welche Verfahren auf eine Aufgabe anwendbar sind, und ihre Stärken und Schwächen sowie die erforderlichen Voraussetzungen kennen. Die theoretischen Merkmale oder detaillierte Informationen über den Anpassungsalgorithmus waren von viel geringerer Bedeutung. Beim Verfassen des Buchs »An Introduction to Statistical Learning with R« (ISLR) hatten wir diese Gruppe im Sinn. Die

Begeisterung, mit der es aufgenommen wurde, zeigt, wie groß der Bedarf dafür war.

»Machine Learning kompakt« folgt einem ähnlichem Paradigma. Ebenso wie ISLR verzichtet das Buch auf theoretische Herleitungen und bietet dem Leser stattdessen die wichtigsten Informationen bei der Implementierung der verschiedenen Ansätze. Es ist ein kompaktes Handbuch zum Thema Data Science und ich sage vorher, dass es sowohl für Theoretiker als auch für Praktiker zu einer unverzichtbaren Ressource wird. Die englische Ausgabe ist mit rund 150 Seiten kurz genug, um sie am Stück zu lesen. Aber trotz der Kürze werden alle wichtigen Machine-Learning-Ansätze behandelt: klassische lineare und logistische Regression, moderne Support Vector Machines, Deep Learning, Boosting und Random Forests. Es mangelt auch nicht an Details zu den verschiedenen Ansätzen, und der interessierte Leser kann im Wiki zum Buch weitere Einzelheiten zu einem bestimmten Verfahren finden. Das Buch setzt weder spezielle mathematische oder statistische Kenntnisse noch Programmiererfahrung voraus und sollte somit jedem zugänglich sein, der bereit ist, etwas Zeit in das Erlernen dieser Verfahren zu investieren. Für angehende Doktoranden sollte es Pflichtlektüre sein, denn es wird beim weiteren Lernen als nützliche Referenz dienen. Einige der Algorithmen werden mithilfe von Code-Beispielen in Python veranschaulicht, einer der verbreitetsten Programmiersprachen beim Machine Learning. Ich kann das Buch sowohl Einsteigern, die mehr über Machine Learning erfahren möchten, als auch erfahrenen Praktikern, die ihr Wissen erweitern möchten, nur wärmstens empfehlen.

Gareth James
Professor für Data Science an der University of Southern California, Koautor (mit Witten, Hastie und Tibshirani) des Bestsellers ***An Introduction to Statistical Learning, with Applications in R***

Einleitung

Beginnen wir mit den Fakten: Maschinen lernen nicht. Eine typische »lernende Maschine« sucht nach einer mathematischen Formel, die die gewünschten Ergebnisse liefert, wenn sie auf eine Sammlung von Eingaben, die sogenannten »Trainingsdaten«, angewendet wird. Diese Formel liefert auch für die meisten anderen Eingaben (die sich von den Trainingsdaten unterscheiden) die richtigen Ergebnisse, vorausgesetzt sie entstammen derselben oder einer ähnlichen statistischen Verteilung, der die Trainingsdaten entnommen wurden.

Weshalb ist das kein Lernen? Weil die Ausgabe sehr wahrscheinlich völlig falsch ist, wenn man die Eingabe leicht verändert. So funktioniert das Lernen bei Lebewesen nicht. Wenn man das Spielen eines Videospiels erlernt hat, ist man noch immer ein guter Spieler, wenn der Bildschirm ein klein wenig gedreht wird. Ein Machine-Learning-Algorithmus, der mit geradem Blick auf den Bildschirm trainiert wurde, ist nicht mehr in der Lage, das Spiel auf dem gedrehten Bildschirm zu spielen – es sei denn, er wurde auch darauf trainiert, den gedrehten Bildschirm zu erkennen.

Und wieso heißt es dann »Machine Learning«? Der Grund dafür ist, wie so oft, Marketing: Der Ausdruck wurde 1959 von Arthur Samuel geprägt, einem Vorreiter im Bereich der Computerspiele und der künstlichen Intelligenz, während er bei IBM tätig war. In den 2010er-Jahren versuchte IBM, den Begriff »cognitive computing« zu vermarkten, um sich von der Konkurrenz abzusetzen. Auf ähnliche Weise versuchte IBM in den 1960er-Jahren, den coolen neuen Begriff »Machine Learning« zu nutzen, um sowohl Kunden als auch auch begabte Angestellte anzulocken.

Wie Sie sehen, ist künstliche Intelligenz keine Intelligenz und Machine Learning ist kein Lernen. Machine Learning ist jedoch eine allgemein anerkannte Bezeichnung, die sich auf die Wissenschaft und die Entwicklung von Maschinen bezieht, die in der Lage sind, verschiedene nützliche Aufgaben zu erledigen, ohne dass man sie explizit dafür programmieren muss. Das Wort »Learning« bzw. »Lernen« ist also nicht buchstäblich zu verstehen, sondern eine Analogie zum Lernen von Lebewesen.

Wer sollte dieses Buch lesen?

Das Buch enthält nur diejenigen Teile des seit Ende der 1960-er Jahre entwickelten umfassenden Materials über Machine Learning, das von praktischen Wert ist. Ein Einsteiger ins Machine Learning findet gerade genug Informationen, um ein ausreichendes Verständnis zu erlangen, das es ermöglicht, die richtigen Fragen zu stellen.

Alle, die Machine Learning bereits in der Praxis einsetzen, können das Buch als Sammlung von Handlungsanweisungen nutzen. Das Buch erweist sich ebenfalls als nützlich, wenn man am Anfang eines neuen Projekts Gedanken austauscht und sich die Frage stellt, ob Machine Learning zur Lösung einer gegebenen technischen oder geschäftlichen Aufgabe geeignet ist, und wenn ja, welches Verfahren sich zur Lösung anbietet.

Verwendung des Buchs

Wenn Sie ein Einsteiger ins Machine Learning sind, sollten Sie das Buch von Anfang bis Ende durchlesen. (Es sind nur rund 180 Seiten, das ist also kein großer Aufwand.) Wenn Sie an einem bestimmten Thema, das im Buch behandelt wird, besonders interessiert sind und mehr darüber erfahren möchten, finden Sie in den meisten Abschnitten einen QR-Code.

Wenn Sie einen der QR-Codes mit Ihrem Smartphone scannen, wird Ihnen ein Link zu einer Webseite des englischsprachigen Wikis angezeigt (theMLbook.com), das dieses Buch ergänzt. Sie enthält zusätzliches Material: empfohlene Lektüre, Videos, Fragen und Antworten, Codeschnipsel, Tutorials und vieles mehr. Das Wiki wird kontinuierlich durch Beiträge des Autors und anderer Interessierter ergänzt. Dieses Buch wird also, wie ein guter Wein, immer besser, nachdem Sie es gekauft haben. Scannen Sie den QR-Code, um zum Wiki des Buchs zu gelangen. Für einige Abschnitten gibt es zwar keinen QR-Code, aber dennoch eine dazugehörige Wiki-Seite. Geben Sie in das Suchfeld des Wikis einen Suchbegriff ein, um sie anzuzeigen.

Darüber hinaus können Sie unter www.mitp.de/995 alle im Buch verwendeten Diagramme und Abbildungen in Farbe herunterladen.

Viel Spaß beim Lesen!

Andriy Burkov

Kapitel 1

Einführung

1.1 Was ist Machine Learning?

Machine Learning ist ein Teilgebiet der Informatik, das sich mit der Entwicklung von Algorithmen befasst, die eine Sammlung von Beispielen für ein bestimmtes Phänomen benötigen, um nützlich zu sein. Die Beispiele können natürlichen oder menschlichen Ursprungs oder von anderen Algorithmen erzeugt worden sein.

Machine Learning kann auch als das Lösen einer praktischen Aufgabe definiert werden, indem man 1) eine Datenmenge sammelt und 2) anhand dieser Daten mithilfe von Algorithmen ein statistisches Modell entwickelt. Das statistische Modell kann dann zur Lösung der praktischen Aufgabe eingesetzt werden.

1.2 Arten des Lernens

Es gibt überwachtes, teilüberwachtes, unüberwachtes und bestärkendes Lernen (Reinforcement Learning).

1.2.1 Überwachtes Lernen

Beim **überwachten Lernen**[1] besteht die **Datenmenge** aus einer Sammlung **gekennzeichneter Beispiele** $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. Jedes Element $\mathbf{x}_i$ aus N wird als ein **Merkmalsvektor** bezeichnet. Ein Merkmalsvektor ist ein Vektor, bei dem jede Dimension $j = 1, \ldots, D$ einen Wert enthält, der das Beispiel in irgendeiner Form beschreibt. Dieser Wert heißt **Merkmal** (engl. *feature*) und wird als $x^{(j)}$ notiert. Wenn beispielsweise jedes Beispiel $\mathbf{x}$ in der Sammlung eine Person repräsentiert, dann könnte das erste Merkmal $x^{(1)}$ die Größe in Zentimetern enthalten, das zweite Merkmal $x^{(2)}$ könnte das Gewicht

[1] **Fettgedruckte** Begriffe sind im Index am Ende des Buchs zu finden.

in Kilogramm angeben, das dritte Merkmal $x^{(3)}$ könnte das Geschlecht bezeichnen und so weiter. Bei allen Beispielen in der Datenmenge enthält das Merkmal an der Position j des Merkmalsvektors jeweils die gleiche Art Information. Das heißt: Wenn $x_i^{(2)}$ eines Beispiels $\mathbf{x}_i$ das Gewicht in Kilogramm enthält, dann enthält auch $x_k^{(2)}$ in allen Beispielen $\mathbf{x}_k$, $k = 1, \ldots, N$ das Gewicht in Kilogramm. Die **Kennzeichnung** (engl. *label*) y_i kann entweder ein Element einer endlichen Menge $\{1, 2, \ldots, C\}$ von **Klassen** sein oder eine reellwertige Zahl oder eine komplexere Struktur, wie ein Vektor, eine Matrix oder ein Graph. Sofern nicht anders angegeben, bezeichnet y_i in diesem Buch entweder eine endliche Menge von Klassen oder eine reelle Zahl[2]. Sie können sich eine Klasse wie eine Kategorie vorstellen, der ein Beispiel zugeordnet wird. Wenn die Beispiele E-Mails sind und die Aufgabe darin besteht, Spam zu erkennen, dann gibt es die beiden Klassen $\{Spam, Kein\_Spam\}$.

Das Ziel eines **überwachten Lernalgorithmus** besteht darin, anhand der Datenmenge ein **Modell** zu entwickeln, das einen Merkmalsvektor $\mathbf{x}$ als Eingabe entgegennimmt und Informationen ausgibt, die es ermöglichen, die Kennzeichnung für diesen Merkmalsvektor herzuleiten. Ein Modell, das anhand einer Personendatenmenge erzeugt wurde, könnte beispielsweise einen Merkmalsvektor entgegennehmen, der eine Person beschreibt, und die Wahrscheinlichkeit dafür ausgeben, dass die Person an Krebs leidet.

1.2.2 Unüberwachtes Lernen

Beim **unüberwachten Lernen** besteht die Datenmenge $\{\mathbf{x}_i\}_{i=1}^N$ aus einer Sammlung **ungekennzeichneter Beispiele**. Hier bezeichnet $\mathbf{x}$ wieder einen Merkmalsvektor, und das Ziel eines **unüberwachten Lernalgorithmus** besteht darin, ein **Modell** zu entwickeln, das einen Merkmalsvektor $\mathbf{x}$ als Eingabe entgegennimmt und ihn entweder in einen anderen Vektor umwandelt oder ein Ergebnis liefert, das zur Lösung einer praktischen Aufgabe eingesetzt werden kann. Beim **Clustering** gibt das Modell beispielsweise die Cluster-ID der Merkmalsvektoren der Datenmenge zurück. Bei der **Dimensionsreduktion** ist die Ausgabe des Modells ein Merkmalsvektor, der weniger Merkmale als die Eingabe $\mathbf{x}$ besitzt; bei der **Erkennung von Ausreißern** ist die Ausgabe eine reelle Zahl, die angibt, wie sehr sich $\mathbf{x}$ von einem »typischen« Beispiel aus der Datenmenge unterscheidet.

[2] Eine reelle Zahl ist eine Größe, die einen Abstand auf einer Linie angeben kann. Beispiele: 0, -256.34, 1000, 1000.2.

1.2.3 Teilüberwachtes Lernen

Beim **teilüberwachten Lernen** enthält die Datenmenge sowohl gekennzeichnete als auch ungekennzeichnete Beispiele. Für gewöhnlich ist die Anzahl der ungekennzeichneten Beispiel sehr viel größer als die der gekennzeichneten. Das Ziel eines **teilüberwachten Lernalgorithmus** ist das gleiche wie das eines überwachten Lernalgorithmus. Man hofft darauf, dass die Verwendung vieler ungekennzeichneter Beispiele dem Lernalgorithmus hilft, ein besseres Modell zu finden (man könnte auch sagen zu »berechnen«).

Es erscheint nicht gerade einleuchtend, dass das Lernen vom Hinzufügen weiterer ungekennzeichneter Beispiele profitiert. Wir fügen der Aufgabe anscheinend zusätzliche Unsicherheit hinzu. Allerdings bringt das Hinzufügen weiterer ungekennzeichneter Beispiele auch zusätzliche Informationen über die Aufgabe mit sich: Eine größere Stichprobe spiegelt die Wahrscheinlichkeitsverteilung der mit Labeln gekennzeichneten Daten besser wider. Theoretisch sollte ein Lernalgorithmus diese zusätzlichen Informationen nutzen können.

1.2.4 Reinforcement Learning

Reinforcement Learning (bestärkendes Lernen) ist ein Teilgebiet des Machine Learning, bei dem sich die Maschine in einer Umgebung »aufhält« und in der Lage ist, den **Zustand** dieser Umgebung in Form eines Merkmalsvektors wahrzunehmen. Die Maschine kann in jedem Zustand **Aktionen** ausführen. Verschiedene Aktionen ergeben unterschiedliche **Belohnungen** und können die Maschine in einen anderen Zustand der Umgebung überführen. Ziel eines bestärkenden Lernalgorithmus ist das Erlernen einer **Policy** (Vorschrift).

Eine Policy ist eine Funktion (ähnlich dem Modell beim überwachten Lernen), die den Merkmalsvektor eines Zustands entgegennimmt und eine für diesen Zustand optimale Aktion ausgibt. Die Aktion ist optimal, wenn sie die zu erwartende **durchschnittliche Belohnung** maximiert.

Durch Reinforcement Learning können Aufgaben gelöst werden, bei denen die Entscheidungsfindung sequenziell erfolgt und die ein langfristiges Ziel aufweisen, wie es etwa bei Spielen, Robotik, Ressourcenverwaltung oder Logistik der Fall ist. In

diesem Buch konzentriere ich mich auf einzelne Entscheidungsfindungen, bei denen die Eingaben voneinander unabhängig und die Vorhersagen schon vorhanden sind. Auf Reinforcement Learning werde ich also nicht näher eingehen.

1.3 Funktionsweise des überwachten Lernens

In diesem Abschnitt erkläre ich kurz die Funktionsweise des überwachten Lernens, damit Sie den gesamten Vorgang vor Augen haben, bevor wir uns mit den Einzelheiten befassen. Ich verwende als Beispiel überwachtes Lernen, weil es das in der Praxis am häufigsten eingesetzte Machine-Learning-Verfahren ist.

Am Anfang des überwachten Lernens steht das Zusammenstellen der Daten. Die Daten für überwachtes Lernen bestehen aus einer Sammlung von (Eingabe, Ausgabe)-Paaren. Als Eingabe kann praktisch alles dienen, beispielsweise E-Mails, Bilder oder Messdaten eines Sensors. Die Ausgaben sind für gewöhnlich reelle Zahlen oder Kennzeichnungen (wie etwa »Spam«, »Kein_Spam«, »Hund«, »Katze«, »Maus« usw.). In manchen Fällen sind die Ausgaben Vektoren (z. B. die vier Koordinaten eines Rechtecks um eine Person auf einem Bild), Sequenzen (wie etwa [»Adjektiv«, »Adjektiv«, »Substantiv«] für die Eingabe »großes schickes Auto«) oder besitzen eine andere Struktur.

Nehmen wir an, Sie wollen überwachtes Lernen zur Erkennung von Spam einsetzen. Sie stellen die Daten zusammen, beispielsweise 10.000 E-Mails, die jeweils mit der Kennzeichnung »Spam« oder »Kein_Spam« versehen sind. (Sie können diese Kennzeichnung von Hand hinzufügen oder jemanden für die Erledigung dieser Aufgabe einstellen.) Jetzt müssen die E-Mails in Merkmalsvektoren umgewandelt werden.

Der Datenanalytiker weiß aus Erfahrung, wie man Daten aus der Praxis, wie eine E-Mail, in einen Merkmalsvektor konvertiert. Ein gängiges Verfahren ist das Bag-of-words-Modell (»Beutel-voller-Wörter«). Man verwendet ein Dictionary, das beispielsweise 20.000 alphabetisch sortierte Wörter enthält, und erstellt nach folgendem Verfahren einen Merkmalsvektor:

- Das erste Merkmal ist gleich 1, wenn die E-Mail das Wort »A« enthält, andernfalls ist dieses Merkmal gleich 0;
- das zweite Merkmal ist gleich 1, wenn die E-Mail das Wort »Aaron« enthält, andernfalls ist dieses Merkmal gleich 0;
- ...

- das Merkmal an Position 20.000 ist gleich 1, wenn die E-Mail das Wort »Zulu« enthält, andernfalls ist dieses Merkmal gleich 0.

Dieses Verfahren wird auf alle E-Mails der Datensammlung angewendet. Das ergibt 10.000 Merkmalsvektoren (jeder Vektor besitzt die Dimensionalität 20.000) mit einer Kennzeichnung (»Spam«/»Kein_Spam«).

Jetzt sind maschinenlesbare Eingabedaten verfügbar, aber die Ausgabedaten liegen noch immer in Textform vor. Bei manchen Lernalgorithmen ist es erforderlich, die Kennzeichnungen in Zahlen umzuwandeln, beispielsweise in Werte wie 0 oder 1, um die Kennzeichnung »Kein_Spam« bzw. »Spam« zu repräsentieren. Der von mir verwendete Algorithmus wird als **Support Vector Machine** (SVM) bezeichnet. Die positiven Kennzeichnungen (in diesem Fall »Spam«) müssen den numerischen Wert $+1$ (eins) und die negativen den Wert -1 (minus eins) besitzen.

Jetzt liegen eine **Datenmenge** und ein **Lernalgorithmus** vor, und Sie können den Lernalgorithmus auf die Datenmenge anwenden, um das **Modell** zu erhalten.

Die SVM betrachtet jeden Merkmalsvektor als einen Punkt in einem hochdimensionalen Raum (der in diesem Fall 20.000-dimensional ist). Der Algorithmus bildet alle Merkmalsvektoren in diesen 20.000-dimensionalen Raum ab und zeichnet eine imaginäre 19.999-dimensionale Linie (eine *Hyperebene*), die Beispiele mit positiver Kennzeichnung von Beispielen mit negativer Kennzeichnung trennt. Beim Machine Learning wird diese Grenze, die verschiedenen Klassen zugehörige Beispiele voneinander trennt, als **Entscheidungsgrenze** bezeichnet.

Die Gleichung der Hyperebene ist durch zwei **Parameter** gegeben, nämlich durch einen reellwertigen Vektor $\mathbf{w}$, der von gleicher Dimensionalität wie der Eingabemerkmalsvektor $\mathbf{x}$ ist, und einer reellen Zahl b:

$$\mathbf{wx} - b = 0,$$

wobei der Ausdruck $\mathbf{wx}$ für $w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \ldots + w^{(D)}x^{(D)}$ steht, und D gibt die Anzahl der Dimensionen des Merkmalsvektors $\mathbf{x}$ an.

(Falls einige der Gleichungen Ihnen nicht ganz klar sind: In Kapitel 2 werden wir uns mit den mathematischen und statistischen Konzepten befassen, die zum Verständnis erforderlich sind. Versuchen Sie fürs Erste, ein Gespür dafür zu entwickeln, was hier geschieht. Nach der Lektüre des nächsten Kapitels wird alles sehr viel einleuchtender sein.)

Die vorhergesagte Kennzeichnung eines Eingabemerkmalsvektors $\mathbf{x}$ ist nun wie folgt gegeben:

$$y = \text{sign}(\mathbf{w}\mathbf{x} - b),$$

wobei sign ein mathematischer Operator ist, der beliebige Werte entgegennimmt und $+1$ oder -1 zurückgibt, wenn die Eingabe eine positive bzw. eine negative Zahl ist.

Das Ziel des Lernalgorithmus – in diesem Fall der SVM – ist es, anhand der Datenmenge die optimalen Werte $\mathbf{w}^*$ und b^* für die Parameter $\mathbf{w}$ und b zu finden. Wenn der Lernalgorithmus diese optimalen Werte gefunden hat, ist das **Modell** $f(\mathbf{x})$ wie folgt definiert:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^*\mathbf{x} - b^*)$$

Um mit einem SVM-Modell vorherzusagen, ob eine E-Mail Spam ist oder nicht, müssen Sie also den Text der E-Mail in einen Merkmalsvektor umwandeln, ihn mit $\mathbf{w}^*$ multiplizieren, b^* subtrahieren und auf das Ergebnis die Vorzeichenfunktion anwenden. Das liefert uns die Vorhersage $+1$ für »Spam« und -1 für »Kein_Spam«.

Wie ermittelt die Maschine $\mathbf{w}^*$ und b^*? Sie löst eine Optimierungsaufgabe. Maschinen sind bestens dazu geeignet, Funktionen unter Nebenbedingen zu optimieren.

Und welche Nebenbedingungen müssen hier erfüllt sein? Das Modell soll vor allem die Kennzeichnungen der 10.000 Beispiele richtig vorhersagen. Zu jedem Beispiel $i = 1, \ldots, 10000$ gehört ein Paar $(\mathbf{x}_i, y_i)$, wobei $\mathbf{x}_i$ den Merkmalsvektor des Beispiels i und y_i die Kennzeichnung bezeichnet, die einen der beiden Werte -1 oder $+1$ annimmt. Die Bedingungen lauten somit:

$$\begin{aligned} \mathbf{w}\mathbf{x}_i - b \geq +1, &\quad \text{wenn } y_i = +1, \\ \mathbf{w}\mathbf{x}_i - b \leq -1, &\quad \text{wenn } y_i = -1. \end{aligned}$$

Die Hyperebene, die positive und negative Beispiele voneinander trennt, sollte einen möglichst breiten **Rand** (engl. *margin*) besitzen. Damit ist der geringste Abstand von Beispielen beider Klassen gemeint, der durch die Entscheidungsgrenze definiert ist. Ein großer Rand trägt zu einer besseren **Generalisierungsfähigkeit** bei, also wie gut das Modell zukünftig neue Beispiele klassifizieren kann. Um das zu erreichen, müssen wir die euklidische Norm $\|\mathbf{w}\|$ von $\mathbf{w}$ minimieren, die durch $\sqrt{\sum_{j=1}^{D}(w^{(j)})^2}$ gegeben ist.

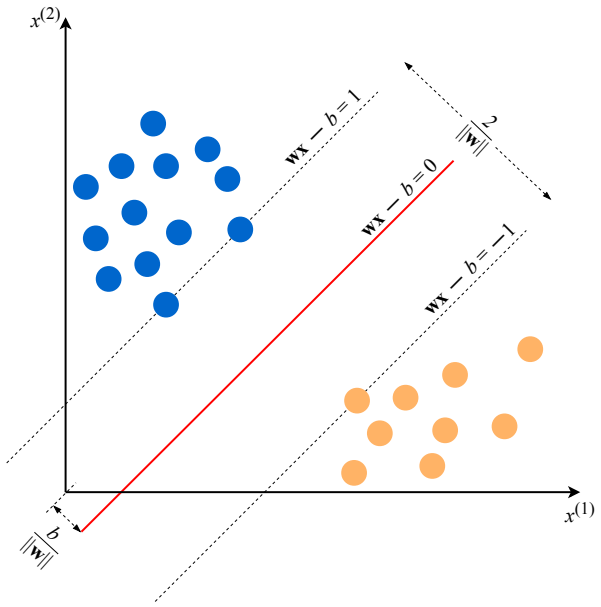


Abb. 1.1: Beispiel für ein SVM-Modell mit zweidimensionalen Merkmalsvektoren

Die von der Maschine zu lösende Optimierungsaufgabe sieht also folgendermaßen aus:

Minimiere $\|\mathbf{w}\|$ *bezüglich* $y_i(\mathbf{w}\mathbf{x}_i - b) \geq 1$ *für* $i = 1, \ldots, N$. Der Ausdruck $y_i(\mathbf{w}\mathbf{x}_i - b) \geq 1$ ist lediglich eine kompakte Schreibweise für die beiden obigen Bedingungen.

Die durch $\mathbf{w}^*$ und b^* gegebene Lösung der Optimierungsaufgabe wird als **statistisches Modell** oder einfach als **Modell** bezeichnet. Der Vorgang des Erstellens eines Modells heißt **Training**.

Bei zweidimensionalen Merkmalsvektoren kann die Aufgabe und ihre Lösung wie in Abbildung 1.1 visualisiert werden. Die blauen und orangefarbenen Kreise repräsentieren positive bzw. negative Beispiele. Die durch $\mathbf{w}\mathbf{x} - b = 0$ gegebene Gerade ist die Entscheidungsgrenze.

Weshalb liefert die Minimierung der Norm von $\mathbf{w}$ den breitesten Rand zwischen den beiden Klassen? Geometrisch betrachtetet definieren die Gleichungen $\mathbf{w}\mathbf{x} - b = 1$ und $\mathbf{w}\mathbf{x} - b = -1$ zwei parallele Hyperebenen, wie in Abbildung 1.1 ersichtlich ist. Der Abstand zwischen diesen beiden Hyperebenen ist durch $\frac{2}{\|\mathbf{w}\|}$ gegeben. Je kleiner also die Norm $\|\mathbf{w}\|$ ist, desto größer ist der Abstand zwischen den beiden Hyperebenen.

So funktionieren Support Vector Machines. Diese Version des Algorithmus nutzt ein sogenannten *lineares Modell*. Es wird als linear bezeichnet, weil die Entscheidungsgrenze eine gerade Linie ist (oder eine flache Ebene oder Hyperebene). SVMs können auch **Kernels** verwenden, die zu einer beliebig nicht-linearen Entscheidungsgrenze führen. In manchen Fällen kann es unmöglich sein, die beiden Punktwolken perfekt voneinander zu trennen, weil die Daten verrauscht oder die Kennzeichnungen fehlerhaft sind, oder weil es **Ausreißer** (engl. *outliers*) gibt (Beispiele, die sich sehr von einem »typischen« Beispiel in der Datenmenge unterscheiden). Eine weitere Version einer SVM kann einen Hyperparameter[3] als Strafterm verwenden, um Fehlklassifikationen von Trainingsbeispielen bestimmter Klassen zu bestrafen. In Kapitel 3 werden wir näher auf den SVM-Algorithmus eingehen.

Sie sollten sich an dieser Stelle Folgendes merken: Ein Lernalgorithmus, der zwecks Klassifikation ein Modell entwickelt, erzeugt implizit oder explizit eine Entscheidungsgrenze. Sie kann gerade, gekrümmt oder anders geformt sein, oder eine Überlagerung verschiedener geometrischer Figuren darstellen. Die Form der Entscheidungsgrenze bestimmt die **Korrektklassifikationsrate** (den Anteil der richtig vorhergesagten Beispiele) des Modells. Die verschiedenen Lernalgorithmen unterscheiden sich durch die Form der Entscheidungsgrenze, bzw. die Art, wie sie auf Basis der Trainingsdaten algorithmisch oder mathematisch berechnet wird.

In der Praxis sind zwei weitere Unterschiede von Lernalgorithmen zu berücksichtigen: die Geschwindigkeit der Modellbildung und die Dauer der Berechnung einer Vorhersage. Bei vielen praktischen Anwendungen wird man einen Lernalgorithmus bevorzugen, der ein ungenaueres Modell schnell erzeugt. Darüber hinaus wird man ungenaueren Modellen den Vorzug geben, die sehr viel schneller Vorhersagen treffen können.

1.4 Weshalb das Modell mit neuen Daten umgehen kann

Weshalb ist ein Machine-Learning-Modell in der Lage, die Kennzeichnung von neuen, völlig unbekannten Daten richtig vorherzusagen? Werfen Sie einen Blick auf Abbildung 1.1. Wenn zwei Klassen sich durch eine Entscheidungsgrenze voneinander trennen lassen, dann befinden sich die Beispiele,

[3] Ein Hyperparameter ist eine Eigenschaft eines Lernalgorithmus, die für gewöhnlich (aber nicht immer) einen numerischen Wert besitzt. Der Wert beeinflusst die Funktionsweise des Algorithmus. Diese Werte werden vom Algorithmus nicht anhand der Daten erlernt, sondern müssen vor der Ausführung des Algorithmus festgelegt werden.

die zu unterschiedlichen Klassen gehören, offensichtlich in zwei verschiedenen Unterräumen, die durch die Entscheidungsgrenze entstehen.

Wenn die für das Training verwendeten Beispiele zufällig ausgewählt werden und voneinander unabhängig sind, dann ist es statistisch gesehen *wahrscheinlicher*, dass neue negative Beispiele im Diagramm nicht allzu weit von anderen negativen Beispielen entfernt sind. Gleiches gilt auch für ein neues positives Beispiel: Es wird *wahrscheinlich* aus der Umgebung anderer positiver Beispiele stammen. In solch einem Fall wird die Entscheidungsgrenze weiterhin positive und negative Beispiele *mit hoher Wahrscheinlichkeit* gut voneinander trennen. In anderen, *weniger wahrscheinlichen Situationen*, wird das Modell fehlerhaft arbeiten, aber da diese Situationen weniger wahrscheinlich sind, wird die Anzahl der Fehler wahrscheinlich kleiner sein als die Anzahl der richtigen Vorhersagen.

Je größer die Menge der Trainingsbeispiele ist, desto unwahrscheinlicher wird es, dass ein neues Beispiel den für das Training verwendeten Beispielen unähnlich ist (und im Diagramm weit von ihnen entfernt ist).

Um die Wahrscheinlichkeit zu minimieren, bei neuen Beispielen Fehler zu begehen, versucht der SVM-Algorithmus, den breitesten Rand zu finden und die Entscheidungsgrenze so anzupassen, dass sie so weit wie möglich von Beispielen beider Klassen entfernt ist.

Wenn Sie mehr über die *Lernfähigkeit* erfahren möchten und den engen Zusammenhang zwischen Fehlern des Modells, der Größe der Trainingsdatenmenge, der Form der mathematischen Gleichung zur Definition des Modells und der Dauer der Entwicklung des Modells besser verstehen möchten, sollten Sie sich über *PAC Learning* informieren. Die PAC-Learning-Theorie (PAC steht für »probably approximately correct«, also »vermutlich ungefähr richtig«) hilft Ihnen dabei, zu analysieren, ob und unter welchen Bedingungen ein Lernalgorithmus »vermutlich« einen »ungefähr richtigen« Klassifizierer erstellen kann.

Index