IN DIESEM KAPITEL

- Maschinelles Lernen und seine Geschichte überblicken
- Frameworks für maschinelles Lernen vergleichen

Kapitel 1
Maschinelles Lernen und TensorFlow

TensorFlow ist ein leistungsstarkes Framework von Google zum Entwickeln von Programmen für maschinelles Lernen. Bevor wir uns im Detail der praktischen Programmierung von TensorFlow-Modulen widmen, bietet dieses Kapitel zunächst eine schrittweise Einführung in die Thematik. Ein kurzer geschichtlicher Abriss vermittelt Ihnen einen Eindruck der Fortschritte, die beim maschinellen Lernen bereits erzielt wurden. Danach erfahren Sie, wie es zur Entwicklung von TensorFlow und vergleichbaren Frameworks für maschinelles Lernen kam.

Was ist maschinelles Lernen?

Wie es sich gehört, zählt *Terminator* zu meinen absoluten Lieblingsfilmen. Das erste Mal habe ich diesen Science-Fiction-Klassiker auf einer Geburtstagsparty gesehen, als ich dreizehn Jahre alt war. Die Story war mir damals zwar größtenteils noch zu hoch, aber eine Szene hat sich mir eingebrannt: Die Heldin des Films glaubt, in gewohnter Herzlichkeit mit ihrer Mutter zu telefonieren – tatsächlich sitzt jedoch ein bösartiger Roboter aus der Zukunft am anderen Ende der Leitung!

Das Irre daran: Diesem Roboter hatte vorher niemand die Stimme der Mutter oder die passenden Sätze einprogrammiert. Damit seine heimtückische Täuschung nicht auffliegt, musste er also selbstständig die Grammatikregeln der Sprache erlernen, die richtigen Sätze für das Gespräch formulieren und noch dazu die Stimme der Mutter exakt nachahmen. Wenn ein Computer Informationen aus Daten ableitet, ohne vorher präzise Anweisungen erhalten zu haben, bezeichnen wir dies als *maschinelles Lernen*.

Der *Terminator* hat mir meinen ersten Kontakt mit dem maschinellen Lernen beschert. Inzwischen ist dieses Verfahren quasi allgegenwärtig. Mein E-Mail-Dienst weiß, dass Nach-

richten mit dem Betreff »Online-Apotheke« in den Spam-Ordner gehören, Nachrichten mit den »aktuellen Aktionsangeboten« meines örtlichen Supermarkts hingegen nicht. Wenn ich Lust auf einen Kick mit Sportfreunden aus dem städtischen Fußballverein habe, zeigt mir Google Maps im Handumdrehen die kürzeste Route zum Sportplatz. Und wenn ich eine neue Bettlektüre brauche, hat Amazon.de sowieso immer passende Buchvorschläge parat. Hinter all dem steckt keine Zauberei, sondern maschinelles Lernen.

Programme, die sich das maschinelle Lernen zunutze machen, suchen nach Mustern in riesigen Datenmengen. Im Gegensatz zu klassischen Programmen müssen sie dabei Unsicherheiten und Wahrscheinlichkeiten in ihre Berechnungen einbeziehen. Es überrascht also kaum, dass auch die Programmierung von maschinellen Lernprogrammen eine gänzlich andere Herangehensweise erfordert. Entwickler müssen sich mit völlig neuen Konzepten und Datenstrukturen vertraut machen.

Glücklicherweise gibt es inzwischen zahlreiche Frameworks, die den Entwicklungsprozess vereinfachen. TensorFlow erfreut sich derzeit wahrscheinlich der größten Beliebtheit. Dieses Buch zeigt Ihnen, wie Sie die leistungsstarken Funktionen dieses Open-Source-Toolkits von Google richtig einsetzen, um eigene maschinelle Lernprogramme zu entwickeln.

Wenngleich ethische Aspekte nicht Thema dieses Buchs sind, sei an dieser Stelle daran erinnert, dass die Programmierung von bösartigen Robotern moralisch abzulehnen ist. Klar, Sie könnten damit Ihre Lehrer und Professoren mächtig beeindrucken. Und auf im Lebenslauf macht sich so was sicher auch prima. Aber unsere Gesellschaft sieht derlei Ambitionen eher kritisch, und Freunde machen Sie sich damit bestimmt nicht. Falls Sie trotzdem der Versuchung erliegen und unbedingt einen bösartigen Roboter programmieren wollen, nun ja, dann wird Ihnen TensorFlow dieses dubiose Vorhaben ungemein erleichtern.

Geschichte des maschinellen Lernens

Wenn Sie mich fragen, ist maschinelles Lernen das spannendste Themenfeld in der modernen Softwareentwicklung und TensorFlow das am besten geeignete Framework. Um Ihnen zu verdeutlichen, warum TensorFlow eine so herausragende Stellung einnimmt, schildern die folgenden Abschnitte, wie es zu seiner Entstehung kam. Abbildung 1.1 stellt wichtige Ereignisse in der Geschichte des maschinellen Lernens und verwandte Softwareentwicklungen in einer vereinfachten Zeitleiste dar.

Sobald Sie verstehen, warum Forscher und Unternehmen so viel Zeit in die Entwicklung dieser Technologie gesteckt haben, wird Ihnen sicher auch klar, warum es keineswegs Zeitverschwendung ist, wenn auch Sie sich genauer mit TensorFlow befassen.

Statistische Regression

Ebenso wie Mineralölkonzerne sich auf der Suche nach kostbarem Öl mit ihren Bohrern durch Erde und Gestein kämpfen, wühlen sich maschinelle Lernprogramme durch enorme Datenmassen, um wertvolle Informationen und Erkenntnisse zutage zu fördern. Dieser

1894	Francis Galton nutzt statistische Regression zur Analyse vererbter Merkmale
1943	McCulloch und Pitts entwerfen das erste künstliche Neuron
1957	Frank Rosenblatt erfindet das Perzeptron
1963	Vapnik und Chervonenkis entwickeln den Support-Vector-Machine-Algorithmus
1974	Paul Werbos nutzt Backpropagation zum Training eines neuronalen Netzes
1982	John Hopfield stellt das Hopfield-Netz vor
1998	Yann LeCun trainiert ein Faltungsnetz zur Erkennung von Schriftzeichen
2002	Collobert, Kavukcuoglu und Farabet veröffentlichen das Torch-Framework
2006	Netflix verspricht 1 Mio. US-Dollar für Algorithmus für Filmempfehlungen
2014	Ian Goodfellow et al. erfinden Generative Adversarial Networks
2015	François Chollet veröffentlicht Keras für die Entwicklung tiefer neuronaler Netze
2015	Google Brain Team veröffentlicht TensorFlow 1.0

Abbildung 1.1: Fortschritte im maschinellen Lernen werden durch die Wissenschaft und von Unternehmen vorangetrieben.

Prozess wird fachsprachlich als *inferentielle* oder *mathematische Statistik* bezeichnet und war bereits den alten Griechen bekannt. Die für uns interessanten Entwicklungen beginnen jedoch erst im 19. Jahrhundert und wurden von dem Wissenschaftler Francis Galton angestoßen. Wenngleich sein Augenmerk primär der Anthropologie galt, ersann er viele jener Konzepte und Hilfsmittel, die heute von Statistikern und maschinellen Lernprogrammen genutzt werden.

Besonders faszinierte ihn die Vererbung von Merkmalen und Talenten. Beim Beobachten von Hunden stellte er fest, dass die Nachkommen von außergewöhnlich begabten Hunden im Laufe der Zeit tendenziell nur durchschnittliche Merkmale aufwiesen. Er beschrieb dies als *Rückkehr zur Mittelmäßigkeit*. (Die heutige Statistik spricht von der *Rückkehr zur Mitte*.) Galton beobachtete dasselbe Phänomen auch bei Menschen und Platterbsen und verwendete zur Analyse seiner Daten moderne statistische Konzepte wie Normalverteilungen, Korrelationen, Varianzen und Standardabweichungen.

Um den Zusammenhang zwischen der Körpergröße eines Kinds und der durchschnittlichen Körpergröße der Eltern aufzuzeigen, entwickelte Galton eine Methode, mit der sich diejenige Gerade bestimmen lässt, die einer Serie von Datenpunkten am besten entspricht. Abbildung 1.2 veranschaulicht die Vorgehensweise. (Galtons Daten wurden von der University of Alabama bereitgestellt.)

Galtons Methode zur Suche nach passenden Geraden für Datenpunkte wurde als *lineare Regression* bekannt, und der Begriff *Regression* ist heute aus der Statistik nicht mehr wegzudenken. Auch beim maschinellen Lernen spielt die Regression eine wichtige Rolle – Kapitel 6 widmet sich dem Thema im Detail.

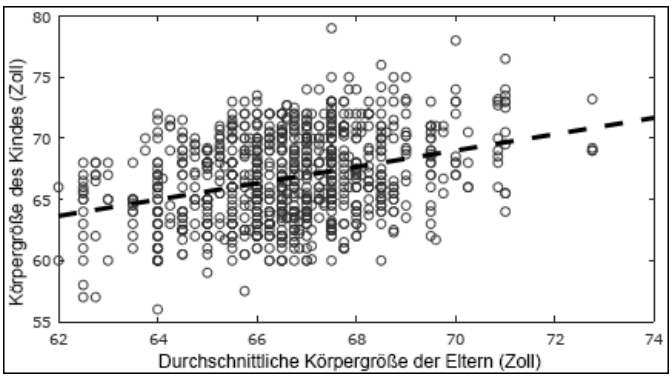

Abbildung 1.2: Die lineare Regression identifiziert einen klaren Trend in einer unübersichtlichen Datenwolke.

Nachahmung des Gehirns

Im Jahr 1905 untersuchte Santiago Ramón y Cajal das Gewebe eines Hühnerhirns und die Vernetzungen zwischen den Zellen, die man später als *Neuronen* bezeichnen würde. Cajals Erkenntnisse stießen bei Wissenschaftlern rund um die Welt auf hohes Interesse, und 1943 entwarfen Warren McCulloch und Walter Pitts ein mathematisches Modell für das Neuron. Mit ihren künstlichen Neuronen ließen sich die geläufigen booleschen Operationen `AND` und `OR` problemlos implementieren.

Im Zuge seiner Statistikforschungen entwickelte der Psychologe Frank Rosenblatt dann ein weiteres Neuronenmodell, das auf den Arbeiten von McCulloch und Pitts aufbaute. Er gab seinem Modell den Namen *Perzeptron* und erstellte durch die Vernetzung mehrerer Perzeptronen in Schichten einen Schaltkreis mit der Fähigkeit zur Erkennung von Bildern. Derartig vernetzte Perzeptronen bezeichnen wir als *neuronale Netze*.

Rosenblatt ließ seinen Vorführungen gewagte Prognosen zu zukünftigen Perzeptronenrechnern folgen. Die Forschungsabteilung der US-Marine war davon so beeindruckt, dass sie bereitwillig Finanzmittel zur Entwicklung eines solchen Neurocomputers zur Verfügung stellte. Die Maschine erhielt den Namen Mark I Perceptron.

Den Perzeptronenrechnern schien also eine glorreiche Zukunft vorherbestimmt – doch 1969 musste das Forschungsfeld einen herben Rückschlag hinnehmen. Marvin Minsky und Seymour Papert präsentierten in ihrem Buch *Perceptrons* (MIT Press) eine sehr kritische Betrachtung von Rosenblatts Ideen. Sie bewiesen auf mathematischem Wege zahlreiche Einschränkungen von zweischichtigen neuronalen Feedforward-Netzen. Beispielsweise seien die Netze nicht in der Lage, nichtlineare Funktionen zu erlernen oder die boolesche exklusive ODER-Verknüpfung (XOR) zu implementieren.

Neuronale Netze haben seit den 1960ern enorme Fortschritte gemacht. Rückblickend lässt sich die Kurzsichtigkeit in der Argumentation von Minsky und Papert kaum leugnen. Damals jedoch warf ihre Arbeit so viele Zweifel auf, dass die US-Marine und zahlreiche weitere große Organisationen das Interesse an neuronalen Netzen vorerst verloren.

Steter Fortschritt

Obwohl sie nunmehr deutlich weniger Unterstützung erhielten, setzten Forscher und Akademiker ihre Arbeiten auf dem Gebiet des maschinellen Lernens fort. Ihr ungebremster Enthusiasmus resultierte in vielen bahnbrechenden Entwicklungen:

- ✔ 1965 führten Ivakhnenko und Lapa mehrschichtige Perzeptronen mit nichtlinearen Aktivierungsfunktionen vor.
- ✔ 1974 nutzte Paul Werbos einen Backpropagation-Algorithmus zum Training eines neuronalen Netzes.
- ✔ 1980 entwickelte Kunihiko Fukushima sein Neocognitron, ein mehrschichtiges neuronales Netz zur Bilderkennung.
- ✔ 1982 entwarf John Hopfield eine Art rekurrentes neuronales Netz – das Hopfield-Netz.
- ✔ 1986 brachten Sejnowski und Rosenberg einem neuronalen Netz namens NETtalk die korrekte Aussprache von Wörtern bei.

Diese Entwicklungen ermöglichten zwar viele neue Anwendungen des maschinellen Lernens, lösten aber kaum Beifallsstürme aus. Das Problem war, dass es den Rechnern schlicht an Geschwindigkeit und Arbeitsspeicher fehlte, um Aufgabenstellungen in der Praxis schnell genug bewältigen zu können. Doch das sollte sich bald ändern.

Revolutionäre Rechenkapazitäten

Anfang der 1990er-Jahre führten rasante Fortschritte in der Halbleiterindustrie zu enormen Verbesserungen bei der Rechenleistung. Forscher machten sich diese neue Leistungsfähigkeit auch beim maschinellen Lernen zunutze. Endlich war es möglich, praxisrelevante maschinelle Lernszenarien durchzuspielen, statt nur simple Modelle zu testen.

Als der Kalte Krieg sich zuspitzte, wuchs das Interesse von Militärstrategen an Computerprogrammen zur automatischen Zielerkennung. Inspiriert durch Fukushimas Neocognitron, konzentrierten sich Forscher auf speziell zur Bilderkennung entwickelte Netze, die sogenannten *konvolutionellen neuronalen Netze* (CNN). Ein wichtiger Schritt nach vorn gelang Yann LeCun, dessen CNN-basierte LeNet5-Architektur im Jahr 1994 erfolgreich handschriftliche Zeichen erkannte.

Doch es ergab sich ein entscheidendes Problem. Forscher legten ihren Programmen zwar ähnliche Theorien zugrunde, schrieben aber jeweils ihren eigenen Code. Sie konnten somit die Ergebnisse ihrer Kollegen nicht reproduzieren und ihren Programmcode untereinander nicht einfach austauschen. Gingen einem Forscher die Finanzmittel aus, verschwand nicht selten auch seine gesamte Programmierarbeit von der Bildfläche.

Ende der 1990er umfasste mein Job die Programmierung von konvolutionellen neuronalen Netzen zur Gesichtserkennung. Während die theoretischen Überlegungen zu neuronalen Netzen mich sehr faszinierten, steigerte ihre praktische Anwendung regelmäßig meinen

Frustpegel. Maschinelle Lernprogramme müssen sorgfältig justiert werden, damit sie akzeptable Ergebnisse liefern. Jede Anpassung des Codes erfordert jedoch einen neuen Trainingsdurchgang, der im Falle eines CNN unter Umständen *mehrere Tage* dauern kann. Obendrein hatte ich nicht genügend Trainingsdaten, um akkurate Ergebnisse sicherstellen zu können.

Wie so viele andere Forscher stand auch ich vor dem Problem, dass die Theorie hinter dem maschinellen Lernen zwar ausgereift war, die Softwareentwicklung aber noch in den Kinderschuhen steckte. Programmierer benötigten dringend Frameworks und Standardbibliotheken, um nicht mehr den kompletten Code selbst schreiben zu müssen. Und trotz aller Bemühungen von Intel erforderte das maschinelle Lernen noch schnellere Prozessoren, die noch größere Datenmengen verarbeiten konnten.

Big Data und Deep Learning

Mit Anbruch des 21. Jahrhunderts nahm der Siegeszug des Internets seinen Lauf, und Datenspeichermedien wurden deutlich preiswerter. Große Unternehmen hatten plötzlich Zugriff auf Terabytes an Daten zu potenziellen Kunden und entwickelten verbesserte Tools zur Auswertung derart riesiger Datenmengen – die *Big-Data-Revolution* war nicht mehr aufzuhalten.

Den Chefs dieser Unternehmen stellte sich nun eine neue knifflige Frage: Wie könnte man aus den Datenschätzen Kapital schlagen? Zu den Prioritäten zählte natürlich die Werbung, denn je mehr Kunden mit den richtigen Anzeigen erreicht werden, desto höher sind die Umsätze eines Unternehmens. Was fehlte, waren klare Regeln, wie man Produkte Kunden zuordnen könnte.

Viele Unternehmen starteten interne Forschungsinitiativen, um geeignete Möglichkeiten zur Auswertung und Nutzung der eigenen Daten zu bestimmen. Doch 2006 schlug Netflix einen anderen Weg ein. Der Streaming-Anbieter stellte einen großen Teil seiner Datenbank ins Internet und rief zur Entwicklung eines Empfehlungssystems auf. Als Preis winkten eine Million US-Dollar. Der Gewinner – Pragmatic Chaos von BellKor – kombinierte mehrere maschinelle Lernalgorithmen und verbesserte das bisherige System von Netflix um zehn Prozent.

Neben Netflix integrierten auch andere bekannte Unternehmen maschinelles Lernen in ihre Produkte. Google beispielsweise bestimmt durch maschinelle Lernalgorithmen, welche Werbeanzeigen durch AdSense auf den Ergebnisseiten seiner Suchmaschine eingeblendet werden. Gemeinsam mit Tesla entwickelte Google zudem selbstfahrende Autos, die dank maschinellem Lernen nicht von der Straße abkommen und sich nahtlos in den Verkehr einfügen.

Rund um die Welt wurden mehr und mehr Unternehmen auf diese neuen Verfahren aufmerksam. Maschinelles Lernen entwickelte sich von einer reinen Science-Fiction-Träumerei zum profitablen Geschäftswerkzeug, und clevere Unternehmer finden immer neue Wege, wie sie maschinelles Lernen gewinnbringend auf ihre Datensammlungen anwenden können.

Auch in Forscherkreisen wurde dieser neue Trend aufmerksam verfolgt. Um das moderne maschinelle Lernen mit seiner enormen Komplexität und dem hohen Aufwand bei der Datenverarbeitung von den früheren, eher simplen und wenig effektiven Ansätzen abzugrenzen, fasste man die neuen Verfahren unter dem Begriff *Deep Learning* zusammen. Kapitel 7 befasst sich genauer mit den technischen Aspekten von Deep Learning.

Frameworks für maschinelles Lernen

Beim Einsatz von maschinellem Lernen in der Praxis spielen Frameworks eine wichtige Rolle. Sie automatisieren zahlreiche Aspekte der Anwendungsentwicklung und ermöglichen es Programmierern, Codesegmente wiederzuverwenden und bewährten Vorgehensweisen zu folgen. Die nächsten Abschnitte stellen fünf der beliebtesten Frameworks vor: Torch, Theano, Caffe, Keras und TensorFlow.

Torch

Torch sicherte sich als erstes Framework für maschinelles Lernen eine treue Anhängerschaft. Die von Ronan Collobert im Jahr 2002 vorgestellte erste Version diente als Toolset für numerische Berechnungen mithilfe multidimensionaler Arrays – den sogenannten *Tensoren* –, die durch reguläre Vektor- und Matrixoperationen verarbeitet werden. Im Laufe der Zeit wurde Torch um Routinen für den Entwurf, das Training und die Bewertung von neuronalen Netzen erweitert.

Wenngleich Torch bei Akademikern und Unternehmen wie IBM und Facebook auf hohes Interesse stieß, konnte sich das Framework nur begrenzt durchsetzen, da die Schnittstelle auf der Skriptsprache Lua basiert. Die anderen hier vorgestellten Frameworks – Theano, Caffe, Keras und TensorFlow – lassen sich über die Programmiersprache Python ansprechen, die als erste Wahl für maschinelles Lernen gilt.

Theano

Im Jahr 2010 stellte eine Forschergruppe der Universität Montreal die Programmbibliothek Theano für numerische Berechnungen vor. Ebenso wie NumPy bietet Theano diverse Python-Routinen für Operationen mit multidimensionalen Arrays. Doch im Gegensatz zu NumPy speichert Theano die Operationen in einer Datenstruktur, die als *Graph* bezeichnet und in hochleistungsfähigen Code übersetzt wird. Theano unterstützt außerdem die *symbolische Differenzierung*, durch die automatisch Ableitungen von Funktionen gefunden werden.

Aufgrund seiner Leistungsfähigkeit und der symbolischen Differenzierung wird Theano von vielen Entwicklern maschineller Lernprogramme als Toolset für numerische Berechnungen bevorzugt. Insbesondere schätzen Entwickler die Möglichkeit zur Ausführung von Graphen auf normalen Prozessoren (CPU) sowie auf speziellen Grafikprozessoren (GPU).

Caffe

Im Rahmen seiner Doktorarbeit an der UC Berkeley entwarf Yangqing Jia ein Framework namens Caffe zum Programmieren von Bilderkennungsprogrammen. Als weitere Entwickler in das Projekt einstiegen, wuchs auch der Funktionsumfang von Caffe. Inzwischen werden unterschiedlichste maschinelle Lernalgorithmen und viele Arten von neuronalen Netzen unterstützt.

Caffe wurde in C++ geschrieben und lässt sich wie Theano per GPU-Ausführung beschleunigen. Durch diese Betonung der Leistungsfähigkeit erfreut sich Caffe bei vielen Akademikern und Entwicklern in der Industrie hoher Beliebtheit. Facebook zeigt sich besonders interessiert und veröffentlichte 2007 eine überarbeitete Version: Caffe2. Diese Version steigert die Leistung von Caffe und ermöglicht die Ausführung von Anwendungen auf Smartphones.

Keras

Während andere Lösungen die Leistungsfähigkeit und den Funktionsumfang in den Vordergrund stellen, legt Keras hohen Wert auf die Modularität und Einfachheit der Entwicklung. François Chollet entwarf Keras als Schnittstelle zu anderen Frameworks für maschinelles Lernen. Viele Entwickler greifen per Keras auf Theano zu und kombinieren so die Einfachheit von Keras mit der Leistung von Theano.

Seine Einfachheit verdankt Keras seiner kleinen API und intuitiv verständlichen Funktionen, mit denen geläufige Aufgaben beim maschinellen Lernen bewältigt werden. Keras eignet sich daher ideal für unerfahrene Anwender, bietet allerdings nur begrenzte Möglichkeiten zur individuellen Anpassung von Operationen.

Chollet veröffentlichte Keras unter der MIT-Lizenz, und Google hat die Schnittstelle in TensorFlow integriert. Aus diesem Grund bevorzugen es viele TensorFlow-Entwickler, ihre neuronalen Netze mit Keras zu programmieren.

TensorFlow

Wie der Titel dieses Buchs verrät, wollen wir uns hier auf TensorFlow konzentrieren. TensorFlow 1.0 wurde 2015 vom Projektteam »Google Brain« vorgestellt. Inzwischen ist bereits Version 1.8 freigegeben. Da das Framework der Apache-2.0-Lizenz unterliegt, dürfen Sie TensorFlow ungehindert verwenden und modifizieren und Ihre Modifikationen weiter verteilen.

Als primäre Schnittstelle kommt Python zum Einsatz; genau wie bei Caffe sind Kernfunktionen von TensorFlow aber zugunsten einer besseren Leistung in C++ geschrieben. Wie Theano speichert auch TensorFlow seine Operationen in einem Graphen, der durch einen Grafikprozessor, ein Remote-System oder ein Netz aus Remote-Systemen verarbeitet wird. Zusätzlich gibt es das sogenannte TensorBoard, durch das Graphen und ihre Operationen visualisiert werden.

Wie die anderen Frameworks unterstützt TensorFlow sowohl die CPU- als auch die GPU-Ausführung. Außerdem können TensorFlow-Anwendungen über die Google Cloud Platform (GCP) ausgeführt werden. Die GCP bietet geballte Rechenleistung zu relativ niedrigen Kosten. Wenn Sie mich fragen, ist dies der wichtigste Vorteil von TensorFlow. In Kapitel 13 widmen wir uns diesem Aspekt im Detail.