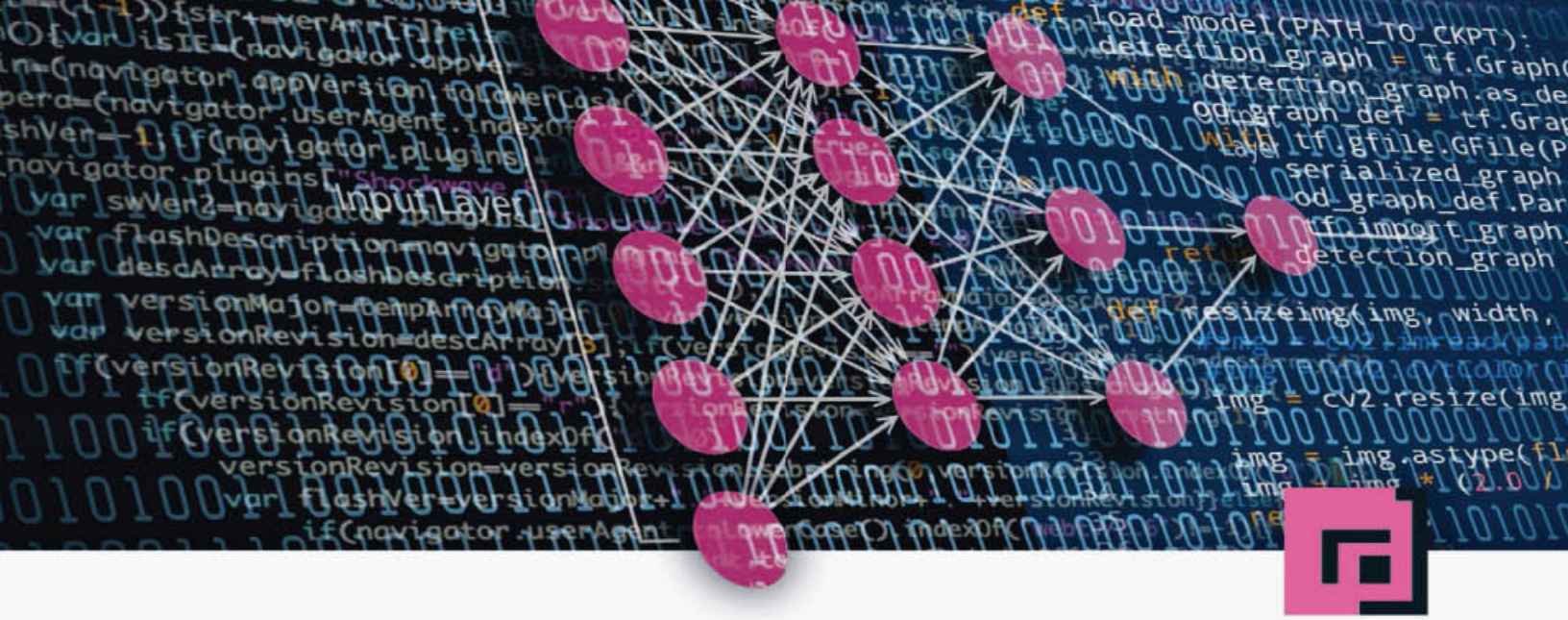


Paolo Perrotta

Machine Learning für Softwareentwickler

Von der Python-Codezeile zur Deep-Learning-Anwendung

dpunkt.verlag

Paolo Perrotta

Machine Learning für Softwareentwickler

Von der Python-Codezeile zur Deep-Learning-Anwendung

Paolo Perrotta

Lektorat: Dr. Michael Barabas
Übersetzung & Satz: G&U Language & Publishing Services GmbH, Flensburg, *www.gundu.com*
Copy-Editing: Claudia Lötschert, *www.richtiger-text.de*
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.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 978-3-86490-787-6
PDF 978-3-96910-025-7
ePub 978-3-96910-026-4
mobi 978-3-96910-027-1

1. Auflage 2020

Wieblinger Weg 17
69123 Heidelberg

Original ISBN: 978-1-68050-660-0

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: hallo@dpunkt.de.

5 4 3 2 1 0

Inhalt

Anhang

Für meine Frau.

Na, wie klingt das, Irene?

Danksagung

Ein besonderes Dankeschön gilt meinen Fachgutachtern: Alessandro Bahgat, Arno Bastenhof, Roberto Bettazzoni, Guido »Zen« Bolognesi, Juan de Bravo, Simone Busoli, Pieter Buteneers, Andrea Cisternino, Sebastian Hennebrüder, Alberto Lumbreras, Russ Olsen, Luca Ongaro, Pierpaolo Pantone, Karol Przystalski, Dan Sheikh, Leonie Sieger, Gal Tsubery, l'ùmarèin pugnàtta di Casalecchio und Giancarlo Valente. Ich schulde euch eine ganze Menge! Ja, unter anderem auch ein Bier!

Vielen Dank auch den großzügigen Lesern, die mir in der Betaphase Kommentare und Errata geschickt haben: Marco Arena, Glen Aultman-Bettridge, Zbynek Bazanowski, Jamis Buck, Charles de Bueger, Leonardo Carotti, Helge Eichhorn, George Ellis, Bruno Girin, Elton Goci, Dave Halliday, Darren Hunt, Peter Lin, Karen Mauney, Bradley Mirly, Vasileios Ntarlagiannis, Volkmar Petschnig, David Pinto, Conlan Rios, Roman Romanchuk, Ionut Simion, Drew Thomas und Jeroen Wenting. Sollten wir uns mal auf einer Konferenz begegnen, dann sprecht mich ruhig an!

Danke auch an Annamaria Di Sebastiano, die mir die Geschichte vom Anfang des ersten Kapitels erzählt hat. Lange nicht mehr gesehen!

Des Weiteren danke ich Marc Schnierle für seine hervorragende Web-App LaTeX4technics[1], mit der ich die Formeln erstellt habe; Kimberly Geswein für die Gestaltung der Schriftart Indie Flower, die ich in den Diagrammen verwendet habe; den Designern der in der Originalausgabe ausgiebig verwendeten Schriftarten DejaVu Sans und DejaVu Sans Mono; und dem Stadtrat von Brisbane für die Veröffentlichung des Fotos mit dem Ameisenigel[2], das in leicht veränderter Form in einem der späteren Kapitel auftaucht.

Dieses Buch ging durch die Hände von drei hervorragenden Lektorinnen. Vielen Dank an Meghan Blanchette und Susan Conant, die mich von den ersten Entwürfen bis zur Mitte des Buchs brachten. Entschuldigt bitte, wenn ich hier noch Unterschiede mache, aber mein besonderer Dank gilt Katherine Dvorak für die unglaubliche Arbeit, die sie in dieses Buch investiert hat, und ihre feste, aber immer geduldige Führung. Du bist eine erstaunliche Lektorin, Katie.

Vielen Dank auch an Dad, Mom, Anna und Susanna sowie an meine erweiterte Familie von Freunden und Verwandten, die weit verstreut rund um die Welt leben. Es ist mir immer eine große Freude, euch um mich zu haben.

Und schließlich danke ich meiner Frau Irene: Du warst da, als ich mit dem Schreiben begann, und hast mich in diesem rauen Sommer sehr unterstützt. Jetzt, da das Buch in den Druck geht, bist du immer noch da, unerschütterlich und unbeirrbar. Ich verdanke dir alles. Dieses Buch ist dir gewidmet.

Wie um alles in der Welt ist so etwas möglich?

Machine Learning kann wie Zauberei wirken. Wie kann ein Computer die Objekte in einem Bild erkennen? Wie kann ein Auto selbstständig fahren?

Diese Leistungen sind nicht nur für Laien verblüffend, sondern auch für viele Softwareentwickler wie Sie und mich. Obwohl ich schon viele Jahre lang Code schrieb, war es mir völlig unklar, wie Machine Learning überhaupt funktionieren konnte. Während ich ein bisschen mit den neuesten Web-Frameworks herumbastelte, schrieben andere Leute faszinierende Software, die wie Science-Fiction wirkte und die ich einfach nicht fassen konnte.

Ich wollte mitmachen. Ich wollte selbst in der Lage sein, solche Dinge zu tun.

Da ich schon wusste, wie man Software schreibt, glaubte ich, ich könnte Machine Learning schnell begreifen. Wie schwer konnte das auch schon sein? Also setzte ich ein zuversichtliches Lächeln auf und widmete mich dem Studium dieses Themas. Mein zuversichtliches Lächeln gefror, als ich gegen ganze Batterien von Wänden anrannte.

Für Entwickler wie mich fühlt sich Machine Learning fremdartig an. Diese Disziplin quillt über von mathematischer Terminologie, akademischen Konventionen

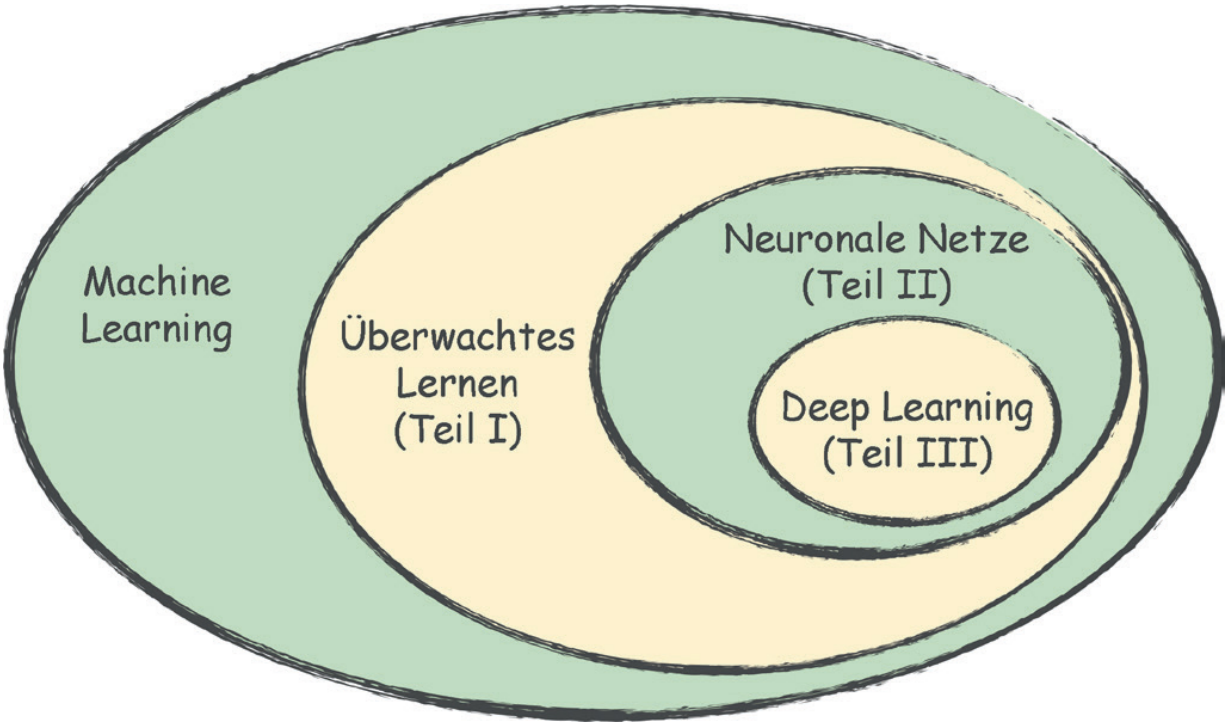
und, mal ganz ehrlich gesagt, schlechtem Code. Anstatt auf Tutorials werden Sie auf Vorträge und Forschungsberichte verwiesen. Für viele von uns ist Machine Learning ebenso einschüchternd wie faszinierend.

Dieses Buch ist das, was ich selbst gut hätte gebrauchen können, als ich mich mit Machine Learning zu befassen begann: eine Einführung, die sich an Entwickler richtet und in unserer Sprache geschrieben ist. Nach der Lektüre werden Sie mit den Grundlagen vertraut und in der Lage sein, Machine-Learning-Programme zu schreiben. Höchstwahrscheinlich werden Sie dann zwar immer noch nicht ihr eigenes selbstfahrendes Auto konstruieren können, aber zumindest werden Sie wissen, wie um alles in der Welt so etwas möglich ist.

Über dieses Buch

Dies ist ein Buch für Entwicklerinnen und Programmierer, die Machine Learning von der Pike auf lernen wollen.

Machine Learning ist ein sehr breit gefächertes Gebiet, das sich in einem einzelnen Buch nicht vollständig abhandeln lässt. Der Schwerpunkt liegt hier daher auf den drei Aspekten des Machine Learning, die heutzutage als die wichtigsten gelten: auf *überwachtem Lernen*, *neuronalen Netzen* und *Deep Learning*. Was es mit diesen Begriffen auf sich hat, werden wir uns im Verlauf dieses Buchs noch ansehen, aber für den Anfang finden Sie hier schon einmal eine Skizze sowie einige Definitionen in Kurzfassung:



- *Überwachtes Lernen* ist die heutzutage am weitesten verbreitete Spielart des Machine Learning. Teil I dieses Buchs, »*Von null auf Bilderkennung*«, gibt eine praktische Einführung in überwachtes Lernen. Schon im zweiten Kapitel werden wir ein minimales ML-Programm schreiben, das wir anschließend Schritt für Schritt erweitern und in ein als *Perzeptron* bezeichnetes Machine-Learning-System verwandeln. Unser Perzeptron ist dabei ein echtes Programm für maschinelles Sehen, das leistungsfähig genug ist, um handgeschriebene Zeichen zu erkennen. Wir gestalten es komplett selbst, ohne dazu ML-Bibliotheken zu Hilfe zu nehmen, damit Sie jede einzelne Zeile des Codes genau verstehen.
- Es gibt viele verschiedene Möglichkeiten, um ein System für überwachtes Lernen zu implementieren. Am häufigsten werden dazu *neuronale Netze* verwendet. Dabei handelt es sich um einen brillanten Algorithmus, der sich an die Verknüpfung der Neuronen in unserem

Gehirn anlehnt. Teil II dieses Buchs ist diesen Netzen gewidmet. Hier bauen wir das Programm aus Teil I zu einem richtiggehenden neuronalen Netz aus. Dazu müssen wir einige Probleme überwinden, aber die Mühe zahlt sich aus: Am Ende haben wir ein neuronales Netz, das weit leistungsfähiger ist als unser ursprüngliches Programm. Auch hier schreiben wir den Code wieder Zeile für Zeile selbst, sodass Sie selbst mit den internen Mechanismen experimentieren können.

- In den letzten Jahren wurden bei neuronalen Netzen erhebliche Fortschritte erzielt, als Forscher bahnbrechende Techniken zu ihrer Konstruktion und Verwendung erfanden. Diese moderne Technologie ist weit leistungsfähiger als die früheren einfachen neuronalen Netze, weshalb sie auch einen eigenen Namen erhalten hat: *Deep Learning*. Das ist auch der Titel von Teil III. Darin schreiben wir unser neuronales Netz mithilfe einer modernen ML-Bibliothek um. Der resultierende Code dient uns dann als Ausgangspunkt zur weiteren Erörterung von Deep Learning. Am Ende schauen wir uns noch einige anspruchsvolle Deep-Learning-Techniken an, womit wir auch die Grundlagen für Ihre weitere Beschäftigung mit diesem Thema legen.

Die Wirklichkeit ist natürlich nicht so sauber geordnet, wie unser Bild anzudeuten scheint. Beispielsweise kommen neuronale Netze nicht nur beim überwachten Lernen, sondern auch auf anderen Gebieten des Machine Learning zum Einsatz. Allerdings bildet dieses Diagramm einen guten Ausgangspunkt, um ein Gefühl für die einzelnen Themen in diesem Buch und ihre Beziehungen zueinander zu gewinnen.

Bevor wir beginnen

Dieses Buch macht Sie nicht über Nacht zu einem Machine-Learning-Profi, aber es kann Ihnen ein anschauliches, praktisches Verständnis dafür vermitteln, wie Machine Learning funktioniert. Ich will Ihnen einen Blick hinter die Kulissen dieser Disziplin gewähren, Ihnen die verborgenen Mechanismen zeigen und den Schleier des Geheimnisvollen lüften. Wenn Sie erst einmal die Grundprinzipien des Machine Learning kennen, ist es für Sie einfacher, selbst weiter nachzuforschen, diese Techniken in Ihre Berufspraxis einfließen zu ließen und vielleicht sogar eine Karriere als ML-Ingenieur einzuschlagen.

Sie müssen kein erfahrener Entwickler sein, um dieses Buch verstehen zu können. Allerdings sollten Sie schon damit vertraut sein, kurze Programme zu schreiben. Wenn Sie Python kennen, dann umso besser, denn das ist die Sprache, die ich hier durchgängig verwende. Wenn nicht, ist das jedoch auch kein Beinbruch. Die Sprache ist sehr benutzerfreundlich, und der Code in diesem Buch ist nicht kompliziert. Zur Einführung können Sie Anhang A, »*Grundlagen von Python*«, lesen, und wenn Sie nicht mehr weiterwissen, finden Sie im Internet weitere Informationen.

Mit Machine Learning ist eine Menge Mathematik verbunden. Ich werde sie nicht vereinfachen, aber so anschaulich darstellen wie möglich. Für das Verständnis dieses Buchs brauchen Sie schon etwas Oberstufenmathematik. Ich gehe davon aus, dass Sie mit kartesischen Koordinatensystemen vertraut sind, wissen, was Achsen und ein Ursprung sind, und etwas mit einem Funktionsgraphen anfangen können. Darüber hinaus brauchen Sie nicht viel mathematisches Fachwissen. Sie können auch versuchen, das Buch zu lesen, wenn Sie glauben, ganz furchtbar schlecht in Mathe zu sein – aber bereiten Sie sich in diesem Fall schon einmal darauf vor, dass es vielleicht doch nicht möglich ist.

Wenn Sie sich dagegen mit linearer Algebra und Analysis gut auskennen, werden Ihnen einige der mathematischen Aspekte wahrscheinlich selbstverständlich erscheinen. In diesem Fall können Sie die Erklärungen, die Sie nicht brauchen, natürlich auch gern überspringen.

Mathematischer Hintergrund

Eine anschauliche Darstellung von mathematischen Dingen ist immer sehr hilfreich, aber manchmal sind auch formalere Erklärungen sinnvoll. Wenn Sie bei der einen oder anderen Formel den Faden verlieren oder wenn Ihnen Mathematik liegt und Sie gern mehr über ein Thema erfahren wollen, dann schauen Sie sich die Kästen mit dem Titel »Mathematischer Hintergrund« an. Darin weise ich Sie auf die Mathematiklektionen der hervorragenden Khan Academy[1] hin. Dort werden Sie etwas Passendes für sich finden, wie gut oder schlecht Ihre Vorkenntnisse in Mathematik auch immer sein mögen.

Um es ganz klar zu sagen: Diese zusätzlichen Lektionen sind optional. Es ist nicht erforderlich, sie sich anzusehen, um dieses Buch verstehen zu können. Sie dienen lediglich dazu, Ihre Kenntnisse über die Mathematik des Machine Learning zu vertiefen.

Machine Learning bringt seine eigene umfangreiche und besondere Terminologie mit. Sie werden daher wahrscheinlich auf neue Wörter und auf neue Bedeutungen bekannter Wörter stoßen. Gehen Sie es ruhig an. Sie müssen sich nicht sofort alles merken. Im Zweifelsfall können Sie immer in Anhang B, »*Wörterbuch des Machine Learning*«, nachschlagen.

Viele der Datenmengen, die ich als Beispiele verwende, bestehen aus Bildern. Mit Machine Learning können Sie jedoch mehr tun, als nur Bilder zu erkennen: Sie können Text analysieren, Musik generieren und sogar Unterhaltungen führen. Allerdings eignet sich gerade die Bilderkennung für anschauliche Beispiele, weshalb ich diese Disziplin im ganzen Buch als Standardanwendung benutze.

Es gibt auch einige Onlineressourcen, die Sie kennen sollten. Eine davon ist die offizielle Webseite[2] zur

Originalausgabe dieses Buchs auf Pragmatic Bookshelf. Von dort können Sie auch den Quellcode der Beispiele herunterladen.

Des Weiteren gibt es zu diesem Buch eine Begleitwebsite namens ProgML[3]. Sie enthält (in englischer Sprache) einige zusätzliche Erklärungen und Einzelheiten, die nicht mehr in dieses Buch passten. An den entsprechenden Stellen im Text habe ich Verweise darauf angegeben.

Verweise auf ProgML sind auf diese Weise gestaltet.

Genug der Vorrede – legen wir nun los mit Teil I!

Teil 1
Von null auf Bilderkennung

Dieser Teil gibt eine Einführung in überwachtes Lernen. Innerhalb von nur zwei Kapiteln werden wir ein erstes Machine-Learning-System programmieren, das wir anschließend Schritt für Schritt weiterentwickeln, bis es leistungsfähig genug ist, um handgeschriebene Ziffern zu erkennen.

Ja, Sie haben richtig gelesen: Auf den nächsten *etwa 100 Seiten* werden wir ein Programm zur Bilderkennung schreiben. Was noch besser ist: Wir werden dazu keine Machine-Learning-Bibliothek verwenden. Abgesehen von einigen Allzweckfunktionen für arithmetische Berechnungen und grafische Darstellung schreiben wir den ganzen Code selbst.

Es ist sehr unwahrscheinlich, dass Sie in Ihrer zukünftigen Karriere jemals Machine-Learning-Algorithmen von Grund auf selbst schreiben müssen. Aber es einmal zu tun, um die Grundlagen richtig zu verstehen, ist von unschätzbarem Wert. Sie werden genau wissen, was jede einzelne Zeile des fertigen Programms tut. Danach wird Machine Learning für Sie nie wieder wie schwarze Magie wirken.

1
Einführung in Machine Learning

Softwareentwickler erzählen sich gern Veteranengeschichten. Sobald ein paar von uns in einer Kneipe zusammensitzen, fragt einer: »An was für Projekten arbeitet ihr gerade?« Dann nicken wir heftig und hören uns die amüsanten und teilweise furchtbaren Geschichten der anderen an.

Bei einem dieser abendlichen Geplänkel Mitte der 90er erzählte mir eine Freundin von dem unmöglichen Auftrag, an dem sie gerade arbeitete. Ihre Vorgesetzten wünschten sich von ihr ein Programm, das Röntgenaufnahmen analysieren und dadurch Krankheiten erkennen konnte, etwa eine Lungenentzündung.

Meine Freundin warnte die Geschäftsleitung vor, dass das ein hoffnungsloses Unterfangen sei, aber man wollte ihr nicht glauben. Wenn ein Radiologe das leisten konnte, so argumentierten die Manager, warum dann nicht auch ein Visual-Basic-Programm? Sie stellten ihr sogar einen Radiologen zur Seite, damit sie lernte, wie er vorging, und dies in Code umsetzen konnte. Diese Erfahrung bestärkte sie jedoch nur in ihrer Meinung, dass Radiologie menschliches Urteilsvermögen und menschliche Intelligenz erforderte.

Wir lachten über die Sinnlosigkeit dieser Aufgabe. Ein paar Monate später wurde das Projekt aufgegeben.

Doch kehren wir nun in die Gegenwart zurück. 2017 veröffentlichte ein Forschungsteam der Stanford University einen Algorithmus, um Lungenentzündung anhand von Röntgenbildern zu erkennen.[1] Er erfüllte nicht nur seine Aufgabe, sondern war sogar zuverlässiger als ein professioneller Radiologe. Das hatte als unmöglich gegolten. Wie hatten die Forscher es geschafft, diesen Code zu schreiben?

Die Antwort lautet: gar nicht. Anstatt Code zu schreiben, setzten sie Machine Learning ein. Sehen wir uns an, was das bedeutet.

Programmierung und Machine Learning im Vergleich

Das folgende Beispiel zeigt den Unterschied zwischen Machine Learning (oder kurz ML) und gewöhnlicher Programmierung. Stellen Sie sich vor, Sie sollen ein Programm erstellen, das Videospiele spielt. Bei der traditionellen Programmierung würden Sie dazu Code wie den folgenden schreiben:

```
enemy = get_nearest_enemy()
if enemy.distance() < 100:
  decelerate()
  if enemy.is_shooting():
    raise_shield()
  else:
    if health() > 0.25:
      shoot()
```

```
        else:
            rotate_away_from(enemy)
    else:
        # ... und noch viel mehr Code
```

Und so weiter. Der Großteil des Codes würde aus einer Riesenmenge von `if... else`-Anweisungen vermischt mit Befehlen wie `shoot()` bestehen.

Moderne Sprachen bieten uns zwar Möglichkeiten, diese hässlichen, verschachtelten `if`-Anweisungen durch angenehmere Konstruktionen wie Polymorphismus, Mustererkennung oder ereignisgestützte Aufrufe zu ersetzen, aber das Grundprinzip der Programmierung bleibt unverändert: Sie sagen dem Computer, wonach er Ausschau halten und was er tun soll. Dabei müssen Sie jede mögliche Bedingung aufführen und jede mögliche Aktion definieren.

Mit dieser Vorgehensweise sind wir weit gekommen, aber sie hat auch einige Nachteile. Erstens dürfen Sie nichts auslassen. Wahrscheinlich können Sie sich Dutzende oder gar Hunderte von besonderen Situationen vorstellen, die Sie in dem Videospielprogramm berücksichtigen müssen. Was geschieht, wenn sich ein Gegner nähert, sich aber ein Power-up zwischen Ihnen und ihm befindet, das Sie vor seinen Schüssen schützen kann? Ein menschlicher Spieler wird eine solche Situation schnell erkennen und zu seinem Vorteil nutzen. Kann ein Programm das auch? Das hängt ganz von dem Programm ab. Wenn Sie diesen Sonderfall beim Schreiben des Codes berücksichtigt haben, dann kann das Programm damit umgehen. Allerdings wissen wir, wie schwer es ist, selbst in so eng umrissenen Aufgabenfeldern wie der Buchhaltung jegliche Sonderfälle

abzudecken. Wenn Sie sämtliche Sonderfälle auf so komplexen Gebieten wie dem Spielen von Videospielen, dem Fahren eines Lkw oder dem Erkennen eines Bilds auflisten wollen, kann ich Ihnen dazu nur viel Glück wünschen.

Es reicht aber nicht nur, all diese Fälle aufzulisten; Sie müssen auch wissen, wie Sie dabei jeweils eine Entscheidung fällen. Das ist die zweite große Einschränkung bei der Programmierung, die in manchen Fachgebieten schon das Aus bedeutet. Betrachten Sie zum Beispiel eine Aufgabe aus dem Bereich des *maschinellen Sehens* wie das bereits erwähnte Erkennen einer Lungenentzündung anhand einer Röntgenaufnahme.

Wir wissen nicht *genau*, wie ein Radiologe eine Lungenentzündung erkennt. Wir haben zwar eine grobe Vorstellung davon, dass er nach undurchsichtigen Bereichen sucht, aber wir wissen nicht, wie sein Gehirn solche undurchsichtigen Bereiche erkennt und bewertet. Manchmal kann ein solcher Experte selbst nicht erklären, wie er zu der Diagnose gekommen ist, sondern nur vage Begründungen geben wie: »Ich weiß aus Erfahrung, dass eine Lungenentzündung nicht so aussieht.« Da wir nicht wissen, wie solche Entscheidungen ablaufen, können wir einen Computer auch nicht anweisen, sie zu fällen. Dieses Problem stellt sich bei allen typisch menschlichen Aufgaben, etwa dem Verkosten von Bier oder dem Verstehen eines Satzes.

Machine Learning dagegen stellt das Prinzip der traditionellen Programmierung auf den Kopf: Der Computer erhält keine *Anweisungen*, sondern *Daten*, und wird aufgefordert, selbst herauszufinden, was er tun soll.