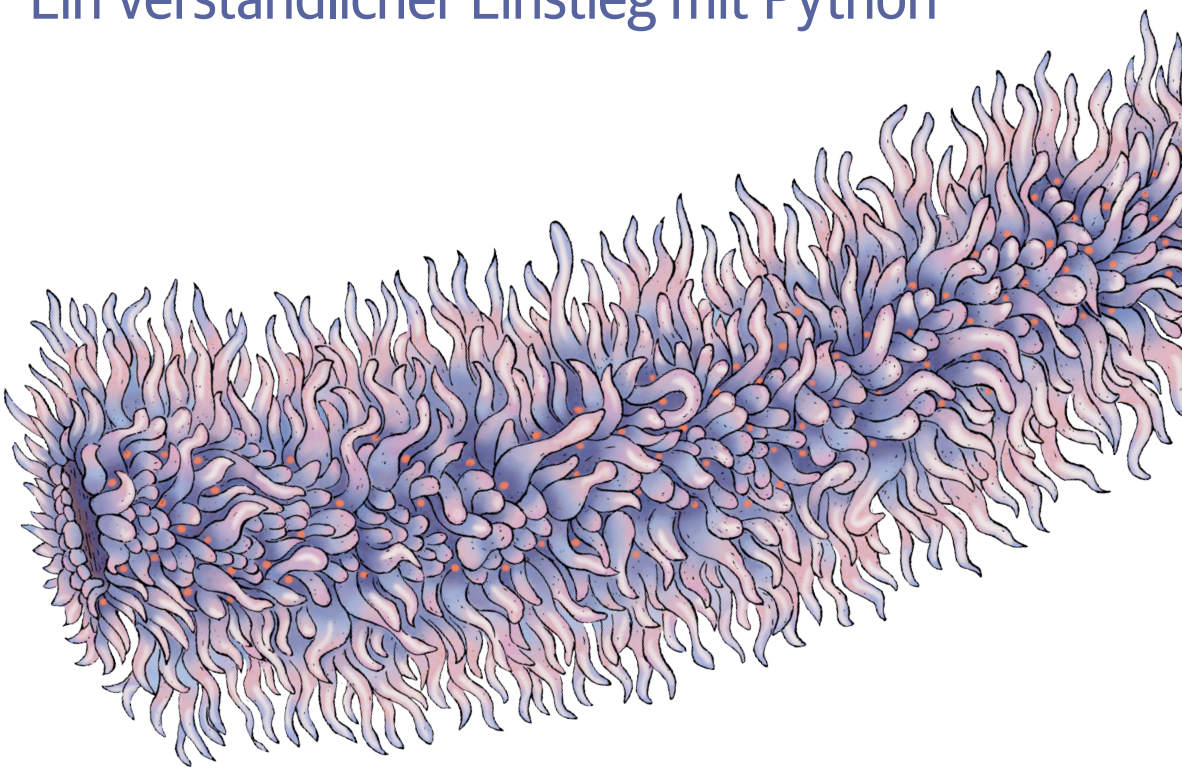
O'REILLY®

2. Auflage
aktualisiert und
erweitert

Neuronale Netze selbst programmieren

Ein verständlicher Einstieg mit Python



Tariq Rashid

Übersetzung von Frank Langenau

2. Auflage

Neuronale Netze selbst programmieren

Ein verständlicher Einstieg mit Python

Tariq Rashid

Deutsche Übersetzung von
Frank Langenau

O'REILLY®

Tariq Rashid

Lektorat: Alexandra Follenius
Übersetzung: Frank Langenau
Copy-Editing: 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 978-3-96009-245-2
PDF 978-3-96010-847-4
ePub 978-3-96010-848-1

2., aktualisierte und erweiterte Auflage 2024

Wieblinger Weg 17
69123 Heidelberg

Title of the English original: Make Your Own Neural Network
ISBN 978-1530826605

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.

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

Inhalt

Einführung

Die Suche nach intelligenten Maschinen

Seit Tausenden von Jahren versuchen Menschen zu verstehen, wie Denkprozesse funktionieren und wie sie sich mit irgendwelchen Maschinen nachbilden lassen – als Denkmaschinen.

Wir geben uns nicht zufrieden mit mechanischen oder elektronischen Maschinen, die uns bei einfachen Arbeiten unter die Arme greifen – Feuersteine, die Feuer anzünden, Flaschenzüge, die schwere Steine heben, und Taschenrechner, die vorgegebene Rechenaufgaben lösen.

Stattdessen möchten wir anspruchsvollere und komplexere Aufgaben automatisieren, beispielsweise ähnliche Fotografien klassifizieren, kranke von gesunden Zellen unterscheiden oder sogar ein gepflegtes Schachspiel erleben. Solche Aufgaben verlangen anscheinend nach menschlicher Intelligenz oder zumindest nach einer geheimnisvollen, tiefer verborgenen Fähigkeit des menschlichen Gehirns, die man in einfachen Maschinen wie Taschenrechnern nicht findet.

Die Idee einer Maschine mit dieser menschenähnlichen Intelligenz ist so verlockend und übermächtig, dass unsere Kultur viele Fantasien und Ängste dazu entwickelt hat – man denke an den äußerst leistungsstarken, doch letztendlich bedrohlichen HAL 9000 aus dem Film *2001: Odyssee im Weltraum* von Stanley Kubrick, die verrückten *Terminator*-Roboter und das sprechende KITT-Auto mit cooler Persönlichkeit aus der klassischen TV-Serie *Knight Rider*.

Als Gary Kasparov, der damals amtierende Schachweltmeister und Großmeister, 1997 durch den IBM-Computer Deep Blue geschlagen wurde, fürchteten wir das Potenzial der Maschinenintelligenz genau so, wie wir diese historische Errungenschaft gefeiert hatten.

Unser Verlangen nach intelligenten Maschinen ist so stark, dass schon so mancher der Versuchung erlegen war, eine solche Maschine vorzutäuschen. Im berüchtigten Schachtürken, einer mechanischen Schachmaschine, war lediglich eine Person im Inneren eines Schranks versteckt!

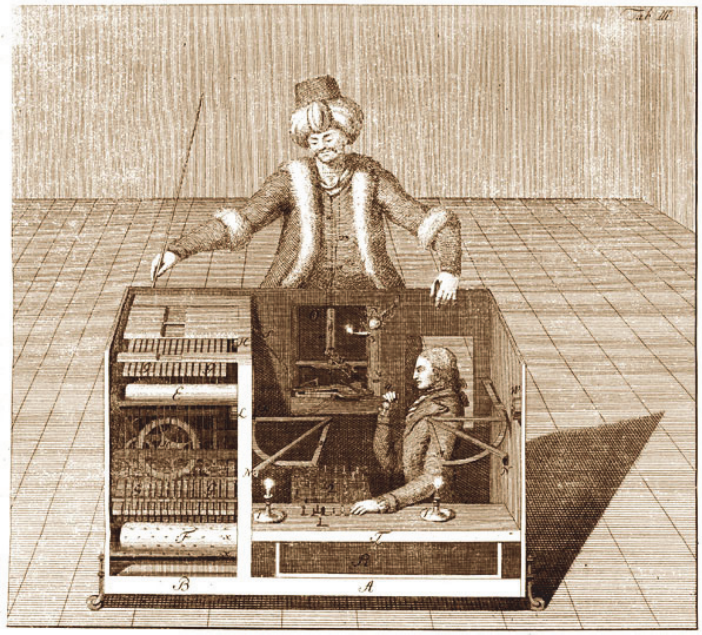


Abbildung E-1: Der Schachtürke – eine »Maschine«, die Schach spielen konnte

Ein neues goldenes Zeitalter – von der Natur inspiriert

Optimismus und Ambitionen in Bezug auf künstliche Intelligenz hatten in den 1950er-Jahren einen Höhenflug, als das Thema formalisiert wurde. Erste Erfolge zeigten sich bei Computern, die einfache Spiele beherrschten und Lehrsätze beweisen konnten. Manche waren davon überzeugt, dass innerhalb eines Jahrzehnts Maschinen erscheinen würden, die das Niveau menschlicher Intelligenz erreichten.

Doch die künstliche Intelligenz erwies sich als harter Brocken, und der Fortschritt stagnierte. Die 1970er-Jahre standen unter schlechten Vorzeichen in Bezug auf künstliche Intelligenz, begleitet von Mittelkürzungen und zurückgehendem Interesse.

Es schien so, als würden Maschinen aus kalter, fest verdrahteter Logik, die mit absoluten Einsen und Nullen arbeiten, niemals in der Lage sein, die nuancenreichen organischen, manchmal unscharfen Denkprozesse von biologischen Gehirnen zu erreichen.

Nach einer Periode ohne wesentliche Fortschritte tauchte eine sehr vielversprechende Idee auf, um die Suche nach Maschinenintelligenz aus ihrer Starre zu befreien. Warum versucht man nicht, künstliche Gehirne zu bauen, indem man kopiert, wie reale biologische Gehirne arbeiten? Reale Gehirne mit Neuronen an-

stelle von Logikgattern, weichere biodynamische Schlussfolgerungen anstelle von kalten, harten, schwarz-weißen, absolutistischen herkömmlichen Algorithmen.

Die Wissenschaftler wurden inspiriert von der scheinbaren Einfachheit eines Bienen- oder Taubengehirns verglichen mit den komplexen Aufgaben, die sie bewältigen konnten. Gehirne, die nur einen Bruchteil eines Gramms wiegen, sind offenbar zu komplizierten Aktionen fähig, zum Beispiel zur Flugsteuerung und Anpassung an den Wind, dazu, Nahrung und Beutetiere zu identifizieren und schnell zu entscheiden, ob gekämpft oder geflüchtet werden muss. Könnten Computer, die jetzt über massive billige Ressourcen verfügten, diese Gehirne nachbilden und verbessern? Eine Biene hat etwa 950.000 Neuronen – könnten heutige Computer mit Speicherkapazitäten im Gigabyte- und Terabyte-Bereich die Bienen übertreffen?

Doch mit den herkömmlichen Konzepten zur Problemlösung können diese Computer trotz der massiven Speicherkapazitäten und superschnellen Prozessoren nicht das erreichen, wozu die relativ winzigen Gehirne von Vögeln und Bienen in der Lage sind.

Neuronale Netze haben sich aus diesem Drang nach biologisch inspiriertem intelligentem Computing herausgebildet – und sind in der Folge zu den leistungsfähigsten und nützlichsten Methoden auf dem Gebiet der künstlichen Intelligenz geworden. Heute ist Google DeepMind zu fantastischen Dingen fähig, beispielsweise von sich aus zu lernen, wie Videospiele gespielt werden, und zum ersten Mal wurde der Weltmeister im unglaublich komplexen Spiel Go geschlagen. Als wesentlicher Bestandteil der Architektur dienen neuronale Netze. Neuronale Netze bilden bereits den Kern vieler Alltagstechnologien – wie der automatischen Nummernschilderkennung und der Decodierung von handschriftlichen Postleitzahlen auf handgeschriebenen Briefen.

Dieses Buch erläutert, was neuronale Netze sind, wie sie funktionieren und wie Sie eigene neuronale Netze erstellen können, die sich für die Erkennung von handgeschriebenen Zeichen trainieren lassen. Eine solche Aufgabe ist mit herkömmlichen Ansätzen der Rechentechnik äußerst schwer zu realisieren.

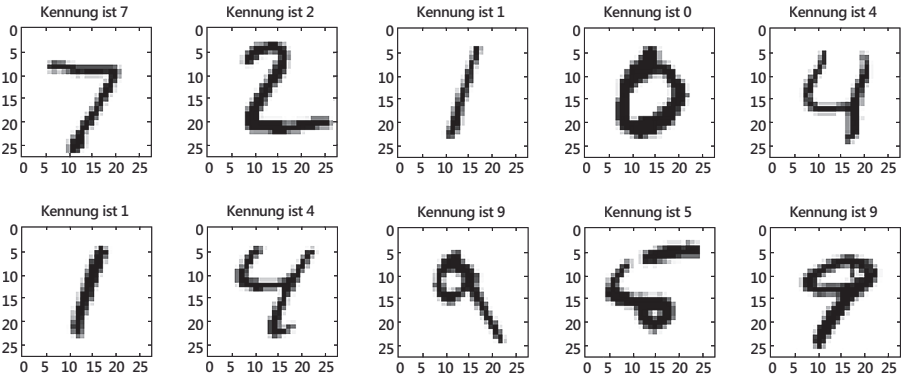


Abbildung E-2: Handgeschriebene Ziffern, die automatisch erkannt werden sollen

An wen richtet sich dieses Buch?

Dieses Buch ist für alle gedacht, die an neuronalen Netzen interessiert sind und eigene Netze erstellen und einsetzen möchten. Es richtet sich an diejenigen, die sich mit den ziemlich einfachen, aber doch spannenden mathematischen Ideen, die den Kern der Arbeitsweise von neuronalen Netzen bilden, auseinandersetzen wollen.

Leserinnen und Leser müssen keine Experten der Mathematik oder Informatik sein. Spezielle Kenntnisse oder mathematische Fähigkeiten, die über die Schulmathematik hinausgehen, sind nicht erforderlich. Um Ihre eigenen neuronalen Netze zu erstellen, kommen Sie mit den vier Grundrechenarten aus. Die schwierigste Operation ist die Gradientenanalyse – doch selbst dieses Konzept wird so erklärt, dass es von den meisten verstanden werden kann.

Interessierten Lesern oder Studentinnen kann dieses Buch als Ausgangspunkt dienen, um spannende Exkursionen in die Welt der künstlichen Intelligenz zu unternehmen. Hat man einmal die Grundprinzipien neuronaler Netze verinnerlicht, kann man die Kernideen auf viele unterschiedliche Probleme anwenden.

Dozentinnen und Dozenten können dieses Buch als leicht verständliche Erläuterung neuronaler Netze und ihrer Implementierung nutzen, um Studentinnen und Studenten zu begeistern und anzuregen, künstliche Intelligenz mit nur wenigen Codezeilen einer Programmiersprache selbst zu erkunden. Die Funktionsfähigkeit des Codes wurde getestet mit einem Raspberry Pi, einem kleinen und preiswerten Computer, der in Schulen und im Studium sehr beliebt ist.

Schön wäre es gewesen, wenn es zu meiner Zeit ein solches Buch gegeben hätte, als ich als Teenager mühevoll herausfinden wollte, wie diese leistungsfähigen und dennoch geheimnisvollen neuronalen Netze funktionieren. Zwar konnte man ihnen in Büchern, Filmen und Zeitschriften begegnen, doch zu jener Zeit konnte ich nur schwierige akademische Artikel finden, die sich an Leute richteten, die schon Mathematiker waren oder sich zumindest in der mathematischen Ausdrucksweise auskannten.

Wenn mir doch nur jemand das Thema hätte so erläutern können, dass es ein einigermaßen neugieriger Schüler verstehen kann! Dieses Buch soll das jetzt nachholen.

Was werden wir tun?

In diesem Buch begeben wir uns auf Tour, um ein neuronales Netz zu erstellen, das handschriftliche Ziffern erkennen kann.

Wir beginnen mit sehr einfachen Vorhersageneuronen und verbessern sie sukzessive, wenn wir an ihre Grenzen stoßen. Dabei legen wir kurze Pausen ein, wenn

erforderliche mathematische Konzepte eingeführt werden, um zu verstehen, wie neuronale Netze lernen und Problemlösungen vorhersagen können.

Auf unserer Tour begegnen wir vielen mathematischen Konzepten. Dazu gehören zum Beispiel Funktionen, einfache lineare Klassifizierer, iterative Verfeinerung, Matrixmultiplikation, Gradientenrechnung, Optimierung nach dem Gradientenverfahren und sogar geometrische Drehungen. Alle diese Konzepte werden aber wirklich leicht verständlich erläutert, und über einfache Schulmathematik hinaus sind weder Vorkenntnisse noch fachliche Erfahrungen erforderlich.

Nach dem erfolgreichen Aufbau unseres ersten neuronalen Netzes entwickeln wir es in verschiedene Richtungen weiter. So verwenden wir Bildverarbeitung, um unser maschinelles Lernen zu verbessern, ohne auf zusätzliche Trainingsdaten zurückzugreifen. Wir werfen sogar einen kurzen Blick in das Gedächtnis unseres neuronalen Netzes, um zu erfahren, ob es etwas von seinen inneren Lernmechanismen preisgibt – etwas, das viele Bücher nicht zeigen!

Außerdem werden wir Python, eine einfache, nützliche und beliebte Programmiersprache, kennenlernen, wenn wir unser neuronales Netz schrittweise aufbauen. Auch hierfür werden keine vorhandenen Programmierkenntnisse vorausgesetzt oder benötigt.

Wie gehen wir vor?

Dieses Buch hat vor allem das Ziel, die Konzepte hinter neuronalen Netzen möglichst vielen Menschen zugänglich zu machen. Das bedeutet, dass wir bei einem neuen Gedanken immer von etwas Bekanntem ausgehen und es in kleinen, einfachen Schritten weiterentwickeln. Auf diese Weise gelangen wir zu einem Punkt, an dem wir interessante und spannende Erkenntnisse über die neuronalen Netze mitnehmen können.

Um den Faden nicht zu verlieren, widerstehen wir der Versuchung, alles zu diskutieren, was nicht unbedingt erforderlich ist, um ein eigenes neuronales Netz zu erstellen. Es wird aber auf interessante Zusammenhänge und Berührungspunkte hingewiesen. Wenn Sie sich näher damit befassen möchten, nur zu!

Dieses Buch betrachtet nicht alle möglichen Optimierungen und Verfeinerungen an neuronalen Netzen. Es gibt viele davon, doch sie würden hier vom eigentlichen Zweck ablenken – nämlich die wesentlichen Ideen in möglichst einfacher und überschaubarer Weise vorzustellen.

Gegliedert ist dieses Buch in vier Kapitel:

- In *Kapitel 1* gehen wir behutsam die mathematischen Konzepte durch, die in einfachen neuronalen Netzen wirken. Wir verzichten zunächst bewusst auf Computerprogramme, um nicht von den Kerngedanken abzulenken.

- In *Kapitel 2* lernen wir gerade so viel Python, um unser eigenes neuronales Netz implementieren zu können. Wir trainieren es darauf, handgeschriebene Ziffern zu erkennen, und wir testen seine Erkennungsleistung.
- In *Kapitel 3* gehen wir weiter, als es für das Verständnis einfacher neuronaler Netze erforderlich ist – just for fun. Wir probieren Ideen aus, um die Leistung unseres neuronalen Netzes weiter zu steigern, und wir werfen auch einen Blick in das Innere eines trainierten Netzes, um zu erkunden, wie das Netz intern lernt und Entscheidungen für die Ausgabe trifft.
- In *Kapitel 4* überführen wir unser neuronales Netz in ein typisches professionelles Szenario, indem wir es mithilfe von PyTorch erstellen, einem in der Branche führenden Framework. Wir sammeln einige praktische Erfahrungen mit der Funktionalität, die uns das Framework PyTorch bietet.

Und keine Angst: Alle hier verwendeten Softwaretools sind *kostenlos* und *Open Source*, sodass Sie dafür nichts bezahlen müssen. Zudem brauchen Sie keinen teuren Computer, um eigene neuronale Netze zu erstellen. Der gesamte Code in diesem Buch ist auf einem sehr preiswerten (ca. 5 Euro teuren) Raspberry Pi Zero getestet worden.

Anmerkung des Autors

Ich hätte mein Ziel verfehlt, wenn ich Ihnen keinen Eindruck davon vermittelt hätte, wie spannend und überraschend Mathematik und Informatik sein können.

Ich hätte versagt, wenn ich Ihnen nicht gezeigt hätte, wie unglaublich leistungsstark Schulmathematik und einfache Computerrezepte sein können – indem man mit eigener künstlicher Intelligenz die Lernfähigkeiten menschlicher Gehirne nachbildet.

Ich hätte versagt, wenn ich Ihnen nicht die Zuversicht und den Wunsch mitgegeben hätte, das unglaublich umfangreiche Gebiet der künstlichen Intelligenz weiter zu erkunden.

Ihr Feedback zu diesem Buch ist mir sehr willkommen. Schreiben Sie dazu den Verlag O'REILLY unter *kommentar@oreilly.de* an.

Den im Buch verwendeten Code finden Sie auf GitHub unter:

https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork

KAPITEL 1

Wie neuronale Netze arbeiten

»Lass dich von all den kleinen Dingen um dich herum inspirieren.«

Leicht für mich – schwer für dich

Computer sind im Grunde nichts weiter als Rechenmaschinen. Arithmetische Aufgaben können sie äußerst schnell ausführen.

Damit sind sie prädestiniert für Aufgaben, die vor allem mit Rechnen zu tun haben – Zahlen addieren, um den Umsatz zu ermitteln, Prozentwerte bilden, um die Umsatzsteuer zu berechnen, Diagramme vorhandener Daten zeichnen usw.

Selbst beim Ansehen von Catch-up-TV oder beim Streamen von Musik hat der Computer nicht viel mehr zu tun, als immer und immer wieder einfache arithmetische Anweisungen auszuführen. Es mag Sie überraschen, doch auch die über das Internet übertragenen Videos, die aus Einsen und Nullen bestehen, werden mit arithmetischen Operationen rekonstruiert, die nicht komplexer sind als die Grundrechenarten, die wir in der Schule gelernt haben.

Zahlen wirklich schnell zu addieren – Tausende oder sogar Millionen pro Sekunde –, ist sicherlich eindrucksvoll, doch das hat nichts mit künstlicher Intelligenz zu tun. Einem Menschen erscheint es vielleicht schwer, schnell große Summen zu bilden, doch ist hierzu kaum Intelligenz erforderlich. Es genügt vollauf, die einfachsten Anweisungen zu befolgen, und genau das ist es, was die Elektronik in einem Computer realisiert.

Drehen wir nun den Spieß um und tauschen wir die Rolle mit dem Computer!

Sehen Sie sich die folgenden Bilder an und versuchen Sie, zu erkennen, was sie enthalten:



Abbildung 1-1: Bilderkennung – einfacher für den Computer oder für den Menschen?

Wir können ein Bild mit menschlichen Gesichtern, einer Katze und einem Baum sehen und erkennen. Praktisch sind wir dazu sehr schnell in der Lage und noch dazu mit einer ziemlich hohen Genauigkeit. Nur in wenigen Fällen liegen wir falsch.

Die recht großen Informationsmengen, die die Bilder enthalten, können wir sehr erfolgreich verarbeiten, um den Bildinhalt zu erfassen. Derartige Aufgaben sind für Computer nicht so einfach lösbar – es ist sogar unglaublich schwierig.

Tabelle 1-1: Wer kann was besonders gut verarbeiten?

Problem	Computer	Mensch
Tausende großer Zahlen schnell multiplizieren	Leicht	Schwer
Gesichter auf einem Foto mit einer Menschenmenge heraussuchen	Schwer	Leicht

Wir ahnen, dass für die Bilderkennung menschliche Intelligenz erforderlich ist – etwas, das Maschinen fehlt, egal wie komplex und leistungsfähig wir sie gebaut haben, weil es eben keine Menschen sind.

Doch es sind genau solche Probleme, die wir dem Computer übertragen möchten – denn Computer arbeiten schnell und werden nicht müde. Um derartige Probleme geht es bei der künstlichen Intelligenz.

Da Computer immer auf Elektronik basieren, besteht die Aufgabe der künstlichen Intelligenz darin, neue Rezepte bzw. *Algorithmen* zu finden, die auf neuartige Weise versuchen, derart schwierigere Probleme zu lösen. Selbst wenn das nicht perfekt gelingt, dann immerhin noch gut genug, um einen Eindruck von einer menschenähnlichen Intelligenz in der Praxis zu geben.

Kernideen

- Manche Aufgaben sind für herkömmliche Computer leicht, für Menschen aber schwer, beispielsweise das Multiplizieren von Millionen Zahlenpaaren.
- Andererseits sind manche Aufgaben für herkömmliche Computer schwer, für Menschen jedoch leicht, beispielsweise das Erkennen von Gesichtern auf einem Foto einer Menschenmenge.

Eine einfache Vorhersagemaschine

Wir beginnen supereinfach und bauen Schritt für Schritt darauf auf.

Stellen Sie sich eine simple Maschine vor, die eine Frage entgegennimmt, etwas »nachdenkt« und eine Antwort ausgibt. Das läuft genau wie im obigen Beispiel ab, in dem wir selbst die Eingaben über die Augen aufnehmen, mit unserem Gehirn die Szene analysieren und daraus ableiten, was die Objekte in dieser Szene bedeuten. Abbildung 1-2 stellt dies schematisch dar.

Abbildung 1-2: Schema einer einfachen Vorhersagemaschine

Computer denken nicht wirklich, sie sind lediglich bessere Taschenrechner. Deshalb wollen wir die Vorgänge mit treffenderen Worten beschreiben (siehe Abbildung 1-3).

Abbildung 1-3: Alternative Beschreibung der Vorhersagemaschine

Ein Computer nimmt eine Eingabe entgegen, führt bestimmte Berechnungen aus und liefert dann eine Ausgabe. Das folgende Beispiel soll das veranschaulichen. Es wird eine Eingabe von »3 x 4« verarbeitet. Das geschieht möglicherweise dadurch, dass die Multiplikation in einen einfacheren Satz von Additionen überführt wird. Die ausgegebene Antwort lautet »12«.

Abbildung 1-4: Beispiel für die Verarbeitung einer Multiplikation

Vielleicht denken Sie jetzt: »Was soll daran beeindruckend sein?« Das stimmt schon. Wir verwenden hier einfache und vertraute Beispiele. Damit veranschaulichen wir die Konzepte, die auf die interessanteren neuronalen Netze angewendet werden, die wir uns später ansehen.

Fahren wir die Komplexität jetzt eine winzige Stufe höher.

Stellen Sie sich eine Maschine vor, die Kilometer in Meilen umrechnet (siehe Abbildung 1-5).

Abbildung 1-5: Umrechnung von Kilometern in Meilen

Nun nehmen wir an, dass wir die Formel für die Umrechnung zwischen Kilometern und Meilen nicht kennen. Wir wissen lediglich, dass die Beziehung zwischen beiden *linear* ist. Wenn man also die Anzahl der Meilen verdoppelt, wird die gleiche Entfernung in Kilometern ebenfalls verdoppelt. Das ist intuitiv verständlich. Das Universum wäre ein seltsamer Ort, sollte dies nicht gelten!

Diese lineare Beziehung zwischen Kilometern und Meilen liefert uns einen Anhaltspunkt über diese geheimnisvolle Berechnung – sie muss die Form haben: `Meilen` = `Kilometer` × `c`, wobei `c` eine Konstante ist. Den Wert dieser Konstanten `c` kennen wir aber noch nicht.

Die einzigen anderen Anhaltspunkte liefern einige Beispiele, die Kilometer und Meilen paarweise angeben. Diese sind wie Beobachtungen der Wirklichkeit, mit denen man wissenschaftliche Theorien überprüft – sie sind Beispiele für die Wahrheit der echten Welt.

Tabelle 1-2: Wertepaare für die Umrechnung zwischen Kilometern und Meilen

Wahrheitsbeispiel	Kilometer	Meilen
1	0	0
2	100	62,137

Was sollten wir tun, um die fehlende Konstante `c` zu ermitteln? Setzen wir einfach einmal einen zufälligen Wert ein und probieren wir es aus! Versuchen wir es mit `c` = 0,5 und schauen wir, was passiert.

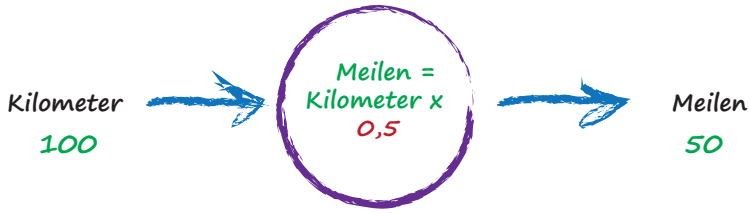


Abbildung 1-6: Zufällig gewählte Konstante c

Hier haben wir `Meilen = Kilometer × c`, wobei `Kilometer` gleich 100 und `c` unsere derzeitige Schätzung 0,5 sind. Damit erhalten wir 50 Meilen.

Nun gut. Das ist gar nicht mal so schlecht unter dem Aspekt, dass wir `c = 0,5` zufällig ausgewählt haben! Doch wir wissen, dass das Ergebnis nicht genau ist, weil das Wahrheitsbeispiel Nummer 2 uns sagt, dass die Antwort 62,137 sein sollte.

Wir liegen um 12,137 daneben. Das ist der *Fehler*, die Differenz zwischen unserer berechneten Antwort und der tatsächlichen Wahrheit aus unserer Beispielliste. Das heißt,

```
Fehler = wahr - berechnet
= 62,137 - 50
= 12,137
```

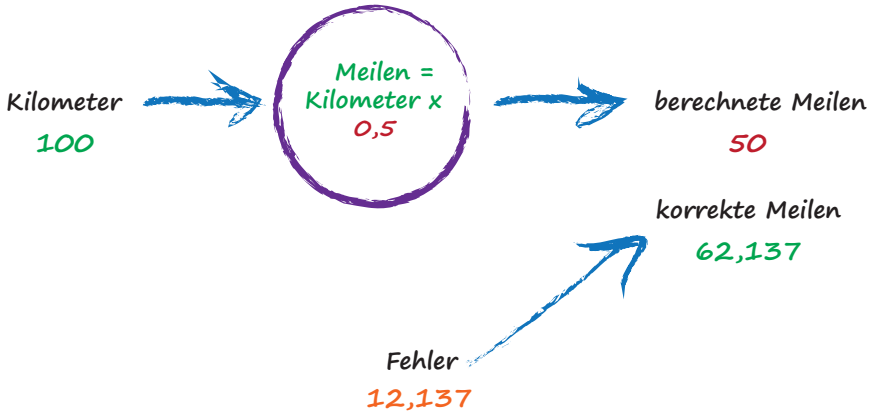


Abbildung 1-7: Der Fehler bei unserer ersten Schätzung

Was kommt als Nächstes? Wir wissen, dass wir falsch liegen und wie groß die Abweichung ist. Anstatt nun aufgrund dieses Fehlers zu verzweifeln, nutzen wir ihn, um zu einer zweiten, besseren Schätzung für `c` zu gelangen.

Sehen Sie sich diesen Fehler noch einmal an. Wir haben 12,137 zu wenig geschätzt. Da die Formel für die Umrechnung von Kilometern in Meilen eine lineare

Beziehung darstellt (`Meilen = Kilometer × c`), wissen wir, dass bei einer Erhöhung von `c` auch der Ausgabewert größer wird.

Wir erhöhen `c` von 0,5 auf 0,6 und sehen uns das neue Ergebnis an. Wenn `c` also auf 0,6 gesetzt ist, erhalten wir `Meilen = Kilometer × c = 100 × 0,6 = 60`. Das ist besser als die vorherige Antwort 50. Zweifellos haben wir einen Fortschritt gemacht!

Der Fehler ist nun mit 2,137 viel kleiner. Es könnte sogar ein Fehler sein, mit dem wir durchaus leben können.

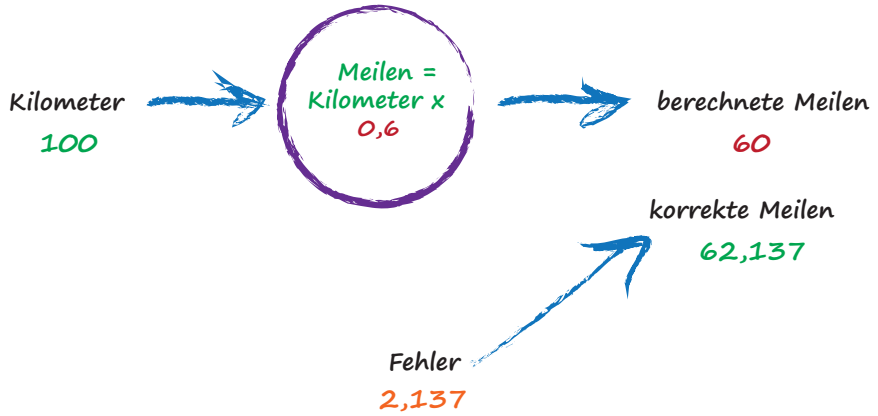


Abbildung 1-8: Die zweite Schätzung ergibt einen kleineren Fehler.

Wichtig ist hier, dass wir uns bei der Entscheidung, um wie viel der Wert von `c` angehoben werden soll, am Fehler orientiert haben. Wir wollten die Ausgabe 50 vergrößern, also haben wir `c` ein wenig erhöht.

Anstatt zu versuchen, den genauen Wert zu ermitteln, um den `c` sich ändern muss, fahren wir mit diesem Verfahren der Verfeinerung von `c` fort. Wenn Sie davon nicht überzeugt sind und meinen, es sei doch einfach genug, die genaue Antwort zu ermitteln, sollten Sie daran denken, dass vielen der interessanteren Probleme keine so einfache Formel zugrunde liegt, die Ausgabe und Eingabe in Beziehung bringt. Deshalb brauchen wir raffiniertere Methoden – wie zum Beispiel neuronale Netze.

Fahren wir also fort. Die Ausgabe von 60 ist immer noch zu klein. Wir schrauben den Wert von `c` ein weiteres Mal nach oben, und zwar von 0,6 auf 0,7.

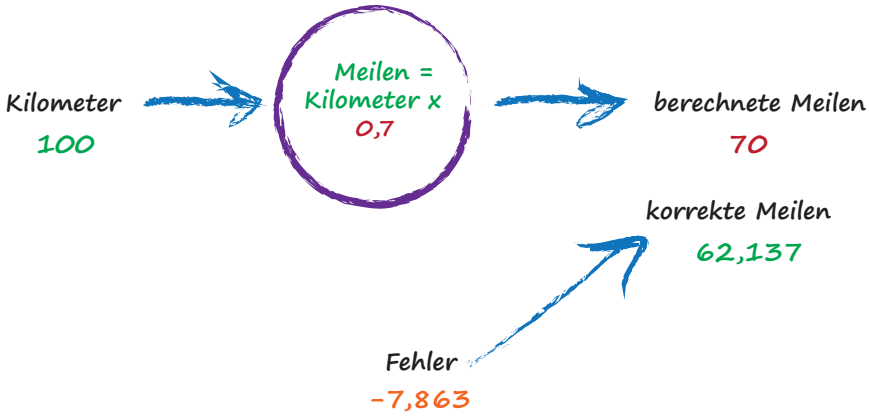


Abbildung 1-9: Die nächste Schätzung liefert einen negativen Fehler.

Oh, nein! Wir sind zu weit gegangen und über die korrekte Antwort hinausgeschossen. Der vorherige Fehler war 2,137, nun liegt er aber bei –7,863. Das Minuszeichen besagt lediglich, dass wir den Zielwert überschritten statt unterschritten haben. Denn der Fehler ergibt sich als `korrekter Wert - berechneter Wert`.

Da nun c = 0,6 besser als c = 0,7 war, könnten wir uns mit dem kleinen Fehler von c = 0,6 zufriedengeben und diese Übung jetzt beenden. Doch wir wollen noch ein wenig weitergehen. Wir könnten c doch auch in kleineren Schritten verändern, beispielsweise von 0,6 auf 0,61.

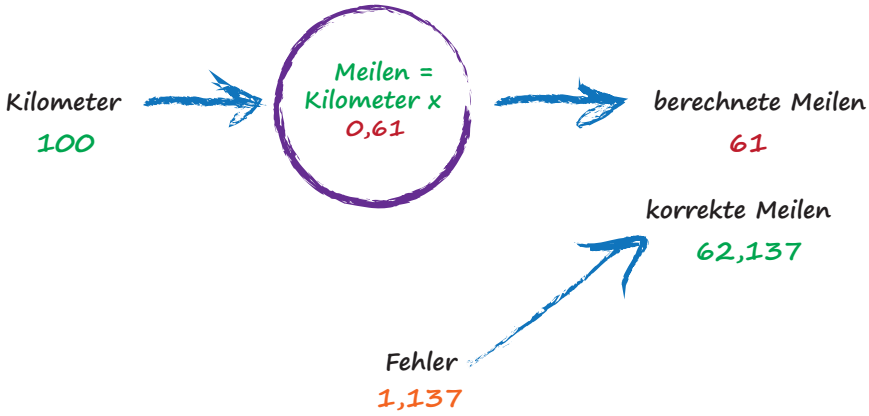


Abbildung 1-10: Die Konstante c wird jetzt in kleineren Schritten verändert.

Das ist schon viel besser als vorher. Wir haben einen Ausgabewert von 61, der nur 1,137 vom exakten Wert 62,137 abweicht.

Dieser letzte Versuch hat uns also gelehrt, dass wir den Wert von c moderat verändern sollten. Wenn die Ausgaben der korrekten Antwort näher kommen – d. h. der Fehler kleiner wird –, sollte man die veränderbare Komponente nicht so stark

anheben. Auf diese Weise lässt sich vermeiden, dass über den richtigen Wert hinausgeschossen wird, wie es weiter oben passiert ist.

Ohne uns nun dadurch ablenken zu lassen, wie denn c im Detail ermittelt wird, bleiben wir beim Konzept der sukzessiven Verfeinerung und wählen als Korrekturwert einen bestimmten Bruchteil des Fehlers. Das ist intuitiv richtig – ein großer Fehler bedeutet, dass eine größere Korrektur erforderlich ist, und ein winziger Fehler heißt, wir brauchen die kleinsten Anpassungsschritte für c.

Ob Sie es glauben oder nicht, wir sind gerade den prinzipiellen Lernprozess in einem neuronalen Netz durchgegangen – wir haben die Maschine trainiert, damit sie immer besser dabei wird, die richtige Antwort zu geben.

Es lohnt sich, kurz innezuhalten und darüber nachzudenken – wir haben ein Problem nicht in einem einzigen Schritt genau gelöst, wie wir es oftmals in der Schulmathematik oder bei wissenschaftlichen Problemen tun. Stattdessen haben wir einen gänzlich anderen Weg eingeschlagen, indem wir eine Antwort ausprobiert und sie wiederholt verbessert haben. Man spricht auch von *iterativ* und meint damit, dass eine Antwort wiederholt Stück für Stück verbessert wird.

Kernideen

- Alle nützlichen Computersysteme übernehmen eine Eingabe und liefern eine Ausgabe, wobei dazwischen bestimmte Berechnungen stattfinden. Neuronale Netze unterscheiden sich hiervon nicht.
- Wenn wir nicht genau wissen, wie etwas funktioniert, können wir versuchen, es mithilfe eines Modells abzuschätzen, dessen Parameter wir anpassen können. Hätten wir nicht gewusst, wie wir Kilometer in Meilen umrechnen, könnten wir eine lineare Funktion mit einem anpassbaren Gradienten als Modell verwenden.
- Eine gute Möglichkeit, dieses Modell zu verfeinern, besteht darin, die Parameter anzupassen, und zwar basierend darauf, wie falsch das Modell verglichen mit bekannten wahren Beispielen ist.

Klassifizieren unterscheidet sich nicht sehr vom Vorhersagen

Die obige einfache Maschine bezeichnen wir als *Prädiktor*, weil sie eine Eingabe übernimmt und eine Vorhersage darüber trifft, wie die Ausgabe sein sollte. Um diese Vorhersage zu verfeinern, haben wir einen internen Parameter angepasst. Dabei haben wir uns an dem Fehler orientiert, der gegenüber dem korrekten Wert bei einem bekannten wahren Beispiel auftritt.

Sehen Sie sich nun die Darstellung in Abbildung 1-11 an, die Messungen der Breite und Länge von Insekten zeigt.

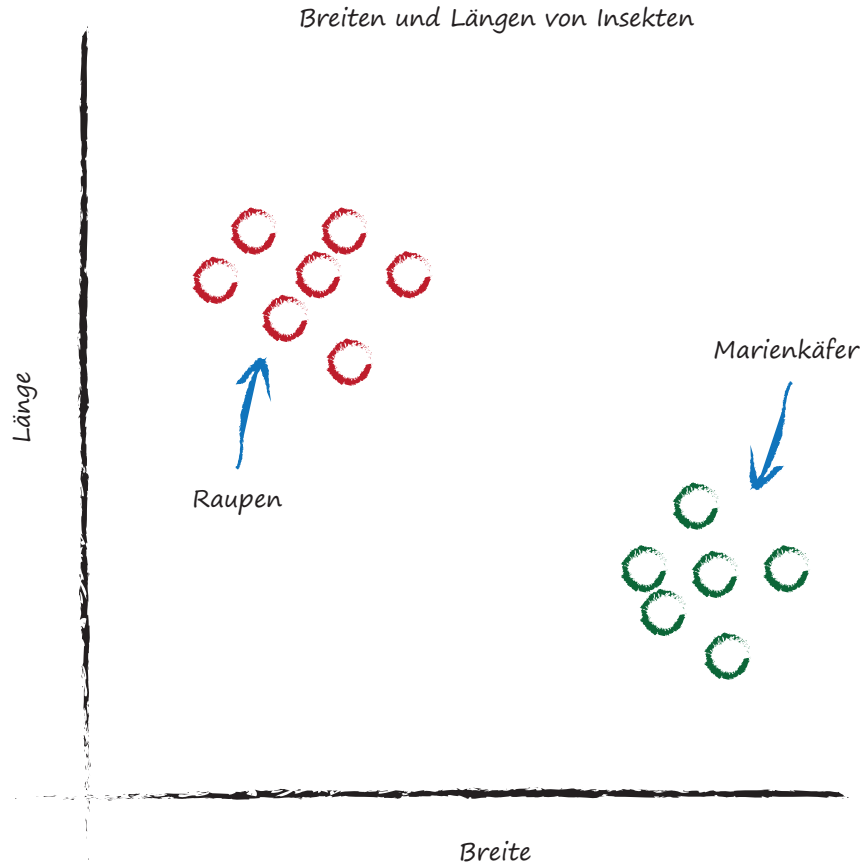


Abbildung 1-11: Diagramm mit Messdaten für die Breite und Länge von Raupen und Marienkäfern

Es lassen sich ganz klar zwei Gruppen ausmachen. Die Raupen sind dünn und lang, die Marienkäfer breit und kurz. (Um das Prinzip zu verdeutlichen, haben wir hier etwas nachgeholfen, Marienkäfer sind natürlich nicht ganz so breit).

Erinnern Sie sich noch an den Prädiktor, der versucht hat, die richtige Anzahl von Meilen für eine gegebene Anzahl von Kilometern herauszufinden? Dieser Prädiktor bestand im Kern aus einer anpassbaren linearen Funktion. Wie Sie wissen, ergeben lineare Funktionen gerade Linien, wenn man ihre Ausgabewerte über den Eingaben als Diagramm darstellt. Der anpassbare Parameter `c` hat den Anstieg dieser Geraden verändert.

Was passiert, wenn wir über diese Punkte eine gerade Linie legen?

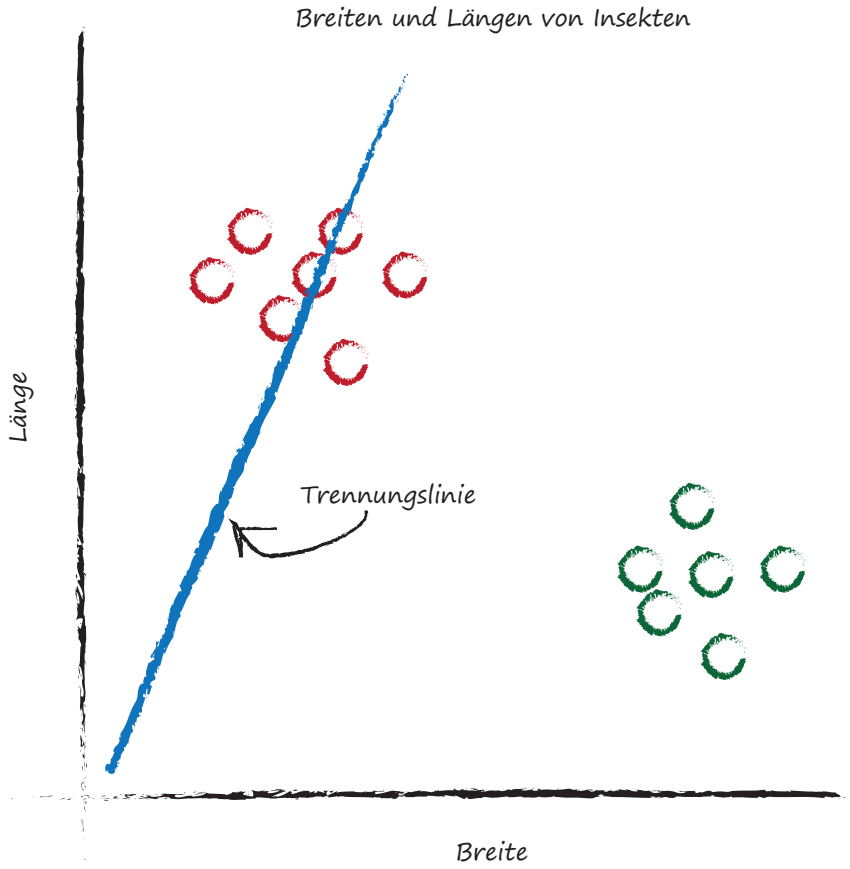


Abbildung 1-12: Trennungslinie durch eine Punktwolke von Messdaten

Die Linie können wir nicht in der gleichen Weise wie zuvor verwenden – um eine Zahl (Kilometer) in eine andere (Meilen) zu konvertieren –, doch vielleicht können wir die Linie nutzen, um verschiedene Arten von Dingen zu trennen.

Wenn die Linie in der obigen Punktdarstellung die Raupen von den Marienkäfern trennen würde, könnte man sie verwenden, um anhand der Messwerte ein unbekanntes Insekt zu *klassifizieren*. Die in Abbildung 1-12 eingezeichnete Linie leistet das noch nicht, da die Hälfte der Raupen auf derselben Seite der Trennungslinie wie die Marienkäfer liegt.

Probieren wir eine andere Linie aus, indem wir wieder den Anstieg der Geraden anpassen, und sehen wir uns die Wirkung an (siehe Abbildung 1-13).

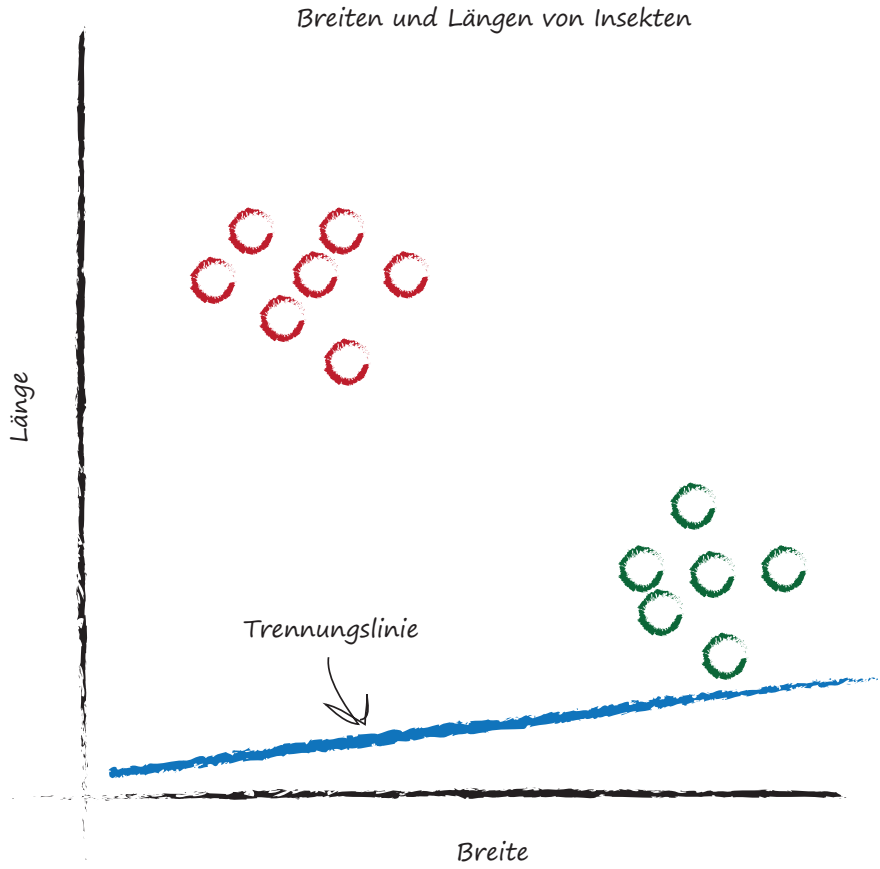


Abbildung 1-13: Trennungslinie mit anderem Anstieg

Dieses Mal ist die Trennungslinie sogar völlig unnütz! Sie trennt die beiden Insektenarten überhaupt nicht.

Abbildung 1-14 zeigt einen weiteren Versuch.