KAPITEL 1

Generative Modellierung

Dieses Kapitel bietet eine allgemeine Einführung in die generative Modellierung. Zunächst verdeutlichen wir, was es bedeutet, wenn ein Modell *generativ* ist, und stellen die Unterschiede zu den weiter verbreiteten *diskriminativen* Modellen heraus. Im Anschluss werden der Rahmen und die mathematischen Kernideen vorgestellt, mit denen wir unseren allgemeinen Ansatz für Fragestellungen, die eine generative Lösung erfordern, strukturieren können.

Auf dieser Grundlage entwickeln wir anschließend unser erstes Beispiel für ein wahrscheinlichkeitsbasiertes generatives Modell (Naive Bayes). Wir werden feststellen, dass dadurch neue Beispiele erzeugt werden, die nicht unserem Trainingsdatensatz angehören. Zugleich untersuchen wir die Gründe dafür, dass diese Art von Modellierung mit zunehmender Größe und Komplexität möglicher Gestaltungsvarianten fehlschlagen kann.

Was ist generative Modellierung?

Ein generatives Modell kann im Wesentlichen wie folgt definiert werden:

> Ein generatives Modell beschreibt, wie ein Datensatz im Rahmen eines Wahrscheinlichkeitsmodells erzeugt wird. Die aus diesem Modell gewonnenen Stichproben ermöglichen uns, neue Daten zu generieren.

Angenommen, wir hätten einen Datensatz mit Bildern von Pferden. Wir möchten ein Modell entwickeln, das ein neues Bild von einem Pferd erzeugen kann, das nie existiert hat, aber trotzdem echt aussieht, weil das Modell die allgemeinen Prinzipien gelernt hat, die das Aussehen eines Pferdes bestimmen. Exakt das ist die Art von Aufgabenstellung, die sich mithilfe der generativen Modellierung lösen lässt. Abbildung 1-1 veranschaulicht den generativen Modellierungsprozess an unserem Beispiel.

Zuerst benötigen wir einen Datensatz, der aus zahlreichen Beispielen der zu erzeugenden Entität besteht. Wir sprechen in diesem Zusammenhang vom *Trainingsdatensatz*, bei dem jeder dieser Datenpunkte als *Beobachtung* bezeichnet wird.

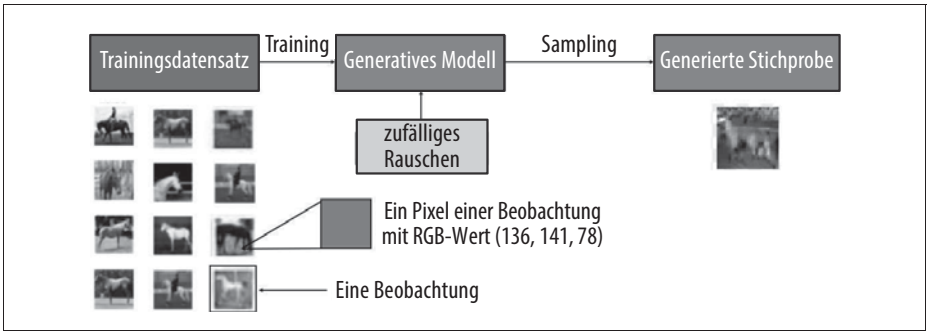

Abbildung 1-1: Der generative Modellierungsprozess

Jede Beobachtung besteht aus vielen *Merkmalen* – bei einer Bilderzeugungsaufgabe sind die Merkmale üblicherweise die jeweiligen Werte der einzelnen Pixel. Unser Ziel ist es, ein Modell zu erstellen, das neue Merkmale erzeugen kann, die so aussehen, als wären sie nach den gleichen Regeln wie die Originaldaten erstellt worden. Rein konzeptionell ist dies für die Bilderzeugung eine unglaublich schwierige Aufgabe – bedenkt man, wie vielfältig die einzelnen Pixelwerte zugeordnet werden können und wie vergleichsweise gering dagegen die Anzahl der Anordnungen ist, die ein Bild der Entität erzeugen, die wir reproduzieren wollen.

Ein generatives Modell sollte zudem *probabilistisch* und nicht *deterministisch* sein. Wenn unser Modell nur eine fest vorgegebene Berechnung umfasst, wie z. B. die Berechnung des Durchschnittswerts jedes Pixels im Datensatz, ist es nicht generativ, da das Modell jedes Mal die gleiche Ausgabe erzeugt. Folglich muss das Modell ein *stochastisches* (zufälliges) Element beinhalten, das die von ihm erzeugten Ausgaben beeinflusst.

Wir können uns das so vorstellen: Es gibt eine unbekannte Wahrscheinlichkeitsverteilung, die erklärt, warum einige Bilder wahrscheinlich im Trainingsdatensatz zu finden sind und andere Bilder hingegen nicht. Unsere Aufgabe ist es, ein Modell zu entwickeln, das diese Verteilung so genau wie möglich nachahmt, um daraus neue, einzigartige Beobachtungen zu generieren, die so aussehen, als würden sie dem ursprünglichen Trainingsdatensatz entstammen.

Vergleich der generativen und diskriminativen Modellierung

Um genau zu verstehen, was generative Modellierung leisten soll und warum sie wichtig ist, ergibt es Sinn, sie mit ihrem Gegenstück, der *diskriminativen Modellierung*, zu vergleichen. Wenn Sie sich in Ihrem Studium mit maschinellem Lernen befasst haben, werden die meisten Fragestellungen, mit denen Sie konfrontiert wurden, höchstwahrscheinlich diskriminativer Natur gewesen sein. Um den Unterschied zu verstehen, schauen wir uns ein Beispiel an.

Angenommen, wir hätten einen Datensatz von mehreren Gemälden. Einige sind von van Gogh und einige von anderen Künstlern. Mit genügend Daten könnten wir ein diskriminatives Modell trainieren, um vorherzusagen, ob ein bestimmtes Bild tatsächlich von van Gogh gemalt wurde. Unser Modell würde lernen, dass bestimmte Farben, Formen und Texturen eher darauf hinweisen, dass ein Gemälde von diesem niederländischen Meister stammt. Für Gemälde mit diesen Merkmalen würde das Modell seine Vorhersage entsprechend höher bewerten. Abbildung 1-2 zeigt den diskriminativen Modellierungsprozess – achten Sie insbesondere darauf, wie er sich von dem in Abbildung 1-1 dargestellten generativen Modellierungsprozess unterscheidet.

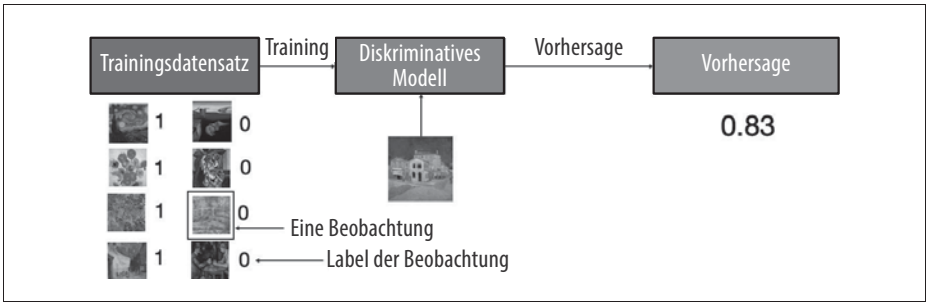

Abbildung 1-2: Der diskriminative Modellierungsprozess

Ein wesentlicher Unterschied besteht darin, dass bei der diskriminativen Modellierung jede Beobachtung in den Trainingsdaten mit einem *Label* versehen ist. Im Fall unserer binären Klassifikationsaufgabe wären Gemälde von van Gogh mit einer *1* und alle anderen mit einer *0* gekennzeichnet. Unser Modell lernt dann, beide Gruppen zu unterscheiden, und gibt die Wahrscheinlichkeit aus, dass eine neue Beobachtung das Label *1* trägt – d. h., dass sie von van Gogh gemalt wurde.

Deshalb ist diskriminative Modellierung gleichbedeutend mit *überwachtem Lernen* beziehungsweise dem Erlernen einer Funktion, die eine Eingabe mithilfe eines mit Labeln versehenen Datensatzes auf eine Ausgabe abbildet. Für die generative Modellierung bedarf es für gewöhnlich nur ungelabelter Datensätze (als Form des unüberwachten Lernens). Sie kann jedoch auch auf einen mit Labeln versehenen Datensatz angewendet werden, um zu lernen, wie man Beobachtungen aus den einzelnen Kategorien erzeugt.

Werfen wir einen Blick auf einige mathematische Notationen, um den Unterschied zwischen generativer und diskriminativer Modellierung zu verdeutlichen.

> *Diskriminative Modellierung* schätzt $p(y|\mathbf{x})$ – die Wahrscheinlichkeit des Labels y, gegeben der Beobachtung $\mathbf{x}$.

Generative Modellierung schätzt $p(\mathbf{x})$ – die Wahrscheinlichkeit, Beobachtung $\mathbf{x}$ zu beobachten.

Sofern der Datensatz mit Labeln versehen ist, können wir auch ein generatives Modell entwickeln, das die Verteilung $p(\mathbf{x}|y)$ schätzt.

Anders ausgedrückt, versucht die diskriminative Modellierung, die Wahrscheinlichkeit abzuschätzen, dass eine Beobachtung $\mathbf{x}$ der Kategorie y angehört. In der generativen Modellierung ist es nicht von Bedeutung, Beobachtungen mit Labeln zu versehen. Stattdessen zielt sie darauf ab, die Wahrscheinlichkeit abzuschätzen, dass die Beobachtung überhaupt auftritt.

Der entscheidende Punkt ist: Selbst wenn wir in der Lage wären, ein perfektes diskriminatives Modell zur Identifizierung von Gemälden von van Gogh zu entwickeln, hätte das Modell immer noch keine Ahnung, wie es ein Gemälde erstellt, das wie eines von van Gogh aussieht. Es kann nur Wahrscheinlichkeiten für bereits existente Bilder ausgeben, da es ausschließlich dazu trainiert wurde. Stattdessen müssten wir ein generatives Modell trainieren, das Pixelgruppen ermitteln kann, die eine hohe Wahrscheinlichkeit besitzen, zum ursprünglichen Trainingsdatensatz zu gehören.

Fortschritte im maschinellen Lernen

Um zu verstehen, warum die generative Modellierung als die kommende Herausforderung für das maschinelle Lernen angesehen werden kann, gilt es zu ergründen, warum die diskriminative Modellierung in den letzten zwei Jahrzehnten die treibende Kraft für die meisten Fortschritte in der Methodik des maschinellen Lernens war, sowohl in der Wissenschaft als auch in der Industrie.

Aus akademischer Sicht ist der Fortschritt der diskriminativen Modellierung sicherlich leichter zu überblicken. Mithilfe von Leistungsmetriken können wir für eine Reihe anspruchsvoller Klassifikationsaufgaben messen, welche Methode derzeit am besten abschneidet. Generative Modelle sind oft schwieriger zu bewerten, insbesondere wenn die Qualität der Ergebnisse weitgehend subjektiv ist. Daher wurde in den letzten Jahren vermehrt Wert auf die Entwicklung diskriminativer Modelle gelegt, um menschliche oder gar übermenschliche Leistungen bei einer Vielzahl von Bild- oder Textklassifikationsaufgaben zu erreichen.

Im Jahr 2012 gelang beispielsweise in der Klassifikation von Bildern der entscheidende Durchbruch, als ein Team um Geoff Hinton von der Universität Toronto mit einem tiefen neuronalen Konvolutionsnetz die *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) gewann. Der Wettbewerb umfasst die Klassifikation von Bildern in eine von 1.000 Kategorien und dient als Benchmark für den Vergleich neuester Methoden. Das Deep-Learning-Modell hatte eine Fehlerquote

von 16% – eine massive Verbesserung gegenüber dem nächstbesten Modell, das lediglich eine Fehlerquote von 26,2% erreichte. Dies löste einen Deep-Learning-Boom aus, der dazu führte, dass die Fehlerquote Jahr für Jahr noch weiter zurückging. Der Gewinner des Jahres 2015 erreichte mit einer Fehlerquote von 4% erstmals eine übermenschliche Leistung, und das aktuell modernste Modell weist eine Fehlerquote von nur 2% auf. Für viele erscheint die Aufgabenstellung nun als gelöst.

Abgesehen von der Tatsache, dass es einfacher ist, messbare Ergebnisse in einem akademischen Umfeld zu veröffentlichen, war die diskriminative im Vergleich zur generativen Modellierung in der Vergangenheit leichter auf Geschäftsprobleme anzuwenden. Im Regelfall interessiert es uns in einem Geschäftsumfeld nicht, wie die Daten generiert wurden. Stattdessen wollen wir wissen, wie ein neues Beispiel kategorisiert oder bewertet werden kann, zum Beispiel:

- Bei einem Satellitenbild würde sich ein Verteidigungsfunktionär nur um die Wahrscheinlichkeit kümmern, dass es feindliche Einheiten enthält, und nicht, wie wahrscheinlich es ist, dass dieses spezielle Bild erscheint.
- Ein Kundenbetreuer wäre nur daran interessiert, zu wissen, ob die Wahrnehmung einer eingehenden E-Mail positiv oder negativ ist, und würde aus einem generativen Modell, das Beispiele für noch nicht existierende Kunden-E-Mails ausgeben könnte, keinen großen Nutzen ziehen.
- Ein Arzt möchte wissen, mit welcher Wahrscheinlichkeit ein bestimmtes Netzhautbild auf Grünen Star hinweist, und nicht auf ein Modell zurückgreifen, das neue Bilder von der Rückseite eines Auges erzeugen kann.

Da die meisten von Unternehmen benötigten Lösungen dem Gebiet der diskriminativen Modellierung entstammen, ist die Anzahl der *Machine-Learning-as-a-Service*-Tools (MLaaS) gestiegen. Sie zielen darauf ab, den Einsatz diskriminativer Modellierung in der Industrie zu kommerzialisieren, indem sie die Erstellungs-, Validierungs- und Überwachungsprozesse, die in fast allen diskriminativen Modellierungsaufgaben vorkommen, weitgehend automatisieren.

Der Vormarsch der generativen Modellierung

Obwohl die diskriminative Modellierung bisher der größte Impulsgeber für Entwicklungen auf dem Gebiet des maschinellen Lernens war, sind in den letzten drei bis fünf Jahren viele der bedeutendsten Fortschritte durch neue Anwendungen des Deep Learning in generativen Modellierungsaufgaben erzielt worden.

Die mediale Aufmerksamkeit stieg insbesondere durch generative Modellierungsprojekte wie StyleGAN von NVIDIA,[1] das in der Lage ist, hyperrealistische Bilder von menschlichen Gesichtern zu erzeugen, und durch das Sprachmodell GPT-2

1 Tero Karras, Samuli Laine und Timo Aila, »A Style-Based Generator Architecture for Generative Adversarial Networks«, 12. Dezember 2018, *https://arxiv.org/abs/1812.04948*.

von OpenAI[2], das eine Textpassage auf Basis eines kurzen einleitenden Absatzes vervollständigen kann.

Abbildung 1-3 zeigt die bemerkenswerten Fortschritte, die seit dem Jahr 2014 bei der Erzeugung von Gesichtsbildern bereits erzielt wurden.[3] Hier gibt es sicherlich nützliche Anwendungen für Branchen wie etwa die Spielegestaltung oder die Filmproduktion. Verbesserungen in der automatischen Erzeugung von Musik werden sicherlich auch in diesen Bereichen immer mehr Beachtung finden. Es wird sich zeigen, ob wir eines Tages Nachrichtenartikel oder Romane lesen werden, die von einem generativen Modell geschrieben wurden. Die jüngsten Fortschritte in diesem Bereich sind enorm. Deshalb ist es sicherlich nicht vermessen, zu behaupten, dass dies eines Tages durchaus der Fall sein könnte. Aber auch wenn es reizvoll erscheint, so wirft es durchaus ethische Fragen rund um die Verbreitung von gefälschten Inhalten im Internet auf, und es könnte immer schwieriger werden, dem zu vertrauen, was wir in den öffentlichen Kommunikationskanälen wahrnehmen und lesen.

Abbildung 1-3: Die Gesichtsbilderzeugung mittels generativer Modellierung hat sich in den letzten vier Jahren deutlich verbessert.[4]

Neben den praktischen Einsatzmöglichkeiten der generativen Modellierung (von denen viele noch unentdeckt sind) gibt es drei wesentliche Gründe dafür, dass sie als Schlüssel zur Erschließung einer weitaus anspruchsvolleren Form der künstlichen Intelligenz angesehen werden kann, die über das hinausgeht, was die diskriminative Modellierung zu leisten vermag.

Erstens sollten wir uns rein theoretisch nicht damit zufriedengeben, dass wir uns lediglich bei der Kategorisierung von Daten auszeichnen können, sondern auch ein umfassenderes Verständnis dafür anstreben, wie die Daten überhaupt generiert wurden. Das ist zweifelsohne ein schwerer zu lösendes Problem, da etliche Ausgaben denkbar sind, jedoch nur wenige davon als zum Datensatz gehörend eingestuft werden würden. Wie wir sehen werden, können viele der Methoden, die bereits die

2 Alec Radford et al., »Language Models Are Unsupervised Multitask Learners«, 2019, *https://paperswithcode.com/paper/language-models-are-unsupervised-multitask*.

3 Miles Brundage et al., »The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation«, Februar 2018, *https://www.eff.org/files/2018/02/20/malicious_ai_report_final.pdf*.

4 Quelle: Brundage et al., 2018.

Entwicklung in der diskriminativen Modellierung vorangetrieben haben, wie z. B. Deep Learning, auch von generativen Modellen genutzt werden.

Zweitens ist es sehr wahrscheinlich, dass die generative Modellierung künftig auch bei Entwicklungen in anderen Bereichen des maschinellen Lernens von zentraler Bedeutung sein wird, wie z. B. beim Reinforcement Learning (der Analyse von Agenten, die in einer bestimmten Umgebung mittels Trial-and-Error ihr Ziel optimieren). Beispielsweise könnten wir das Reinforcement Learning nutzen, um einen Roboter so zu trainieren, dass er über ein bestimmtes Gelände läuft. Der Grundansatz besteht darin, eine Computersimulation des Geländes zu erstellen und dann viele Experimente durchzuführen, bei denen der Agent verschiedene Strategien ausprobiert. Im Laufe der Zeit würde der Agent lernen, welche Strategien erfolgreicher als andere sind, und sich sukzessiv verbessern. Ein typisches Problem bei diesem Ansatz ist, dass die physikalischen Gegebenheiten der Umgebung oft sehr komplex sind und bei jedem Teilschritt berechnet werden müssten, um die Informationen an den Agenten zurückzugeben, auf deren Basis er dann über seinen nächsten Zug entscheiden kann. Wenn der Agent jedoch in der Lage wäre, seine Umgebung durch ein generatives Modell zu simulieren, müsste er die Strategie nicht in der Computersimulation oder in der realen Welt testen, sondern könnte stattdessen in seiner eigenen *imaginären* Umgebung lernen. In Kapitel 8 setzen wir diese Idee in die Praxis um. Wir trainieren ein Auto darauf, so schnell wie möglich auf einer Strecke zu fahren, indem es selbst aus seiner eigenen halluzinierten Umgebung lernen kann.

Nicht zuletzt muss die generative Modellierung sicherlich Teil der Lösung sein, um eine Maschine bauen zu können, die eine Form der Intelligenz erlangt, die der eines Menschen gleicht. Eines der besten Beispiele für ein real existierendes generatives Modell ist die Person, die dieses Buch liest – Sie! Nehmen Sie sich einen Moment Zeit, um zu überlegen, was für ein unglaubliches generatives Modell Sie sind. Sie vermögen die Augen zu schließen und sich vorzustellen, wie ein Elefant aus jedem möglichen Blickwinkel aussehen würde. Sie können sich eine Reihe von unterschiedlichen plausiblen Ausgängen Ihrer Lieblingsserie vorstellen. Und Sie können Ihre Woche im Voraus planen, indem Sie verschiedene Zukunftsszenarien vor Ihrem geistigen Auge durcharbeiten und sich für entsprechende Handlungen entscheiden. Die aktuelle neurowissenschaftliche Theorie legt nahe, dass unsere Wahrnehmung der Realität kein hochkomplexes, auf unseren Sinneseindrücken basierendes diskriminatives Modell ist, um Vorhersagen über das, was wir erleben, zu treffen. Vielmehr gleicht sie einem generativen Modell, das von Geburt an trainiert wird, um Simulationen unserer Umgebung zu liefern, die der Zukunft entsprechen. Einige Theorien legen sogar nahe, dass die Ergebnisse dieses generativen Modells das sind, was wir direkt als Realität wahrnehmen. Offensichtlich ist ein umfassendes Verständnis dafür, wie wir Maschinen bauen können, die diese Fähigkeit erwerben, von zentraler Bedeutung für unser weiteres Verständnis der Funktionsweise des Gehirns und der allgemeinen künstlichen Intelligenz.

Beginnen wir vor diesem Hintergrund unsere Reise in die spannende Welt der generativen Modellierung. Zunächst schauen wir uns die einfachsten Beispiele für generative Modelle und einige der Ideen an, die uns dabei helfen, die komplexeren Architekturen, denen wir später im Buch begegnen werden, durchleuchten zu können.

Das generative Modellierungskonzept

Spielen wir als Erstes ein kleines Spiel mit einem generativen Modell, das nur zwei Dimensionen besitzt. Hierzu habe ich eine Regel gewählt, mit der die Punkte **X** in Abbildung 1-4 erzeugt wurden. Nennen wir diese Regel p_{data}. Ihre Aufgabe besteht darin, einen anderen Punkt $\mathbf{x} = (x_1, x_2)$ in diesem Raum zu wählen, der so erscheint, als wäre er von der gleichen Regel erzeugt worden.

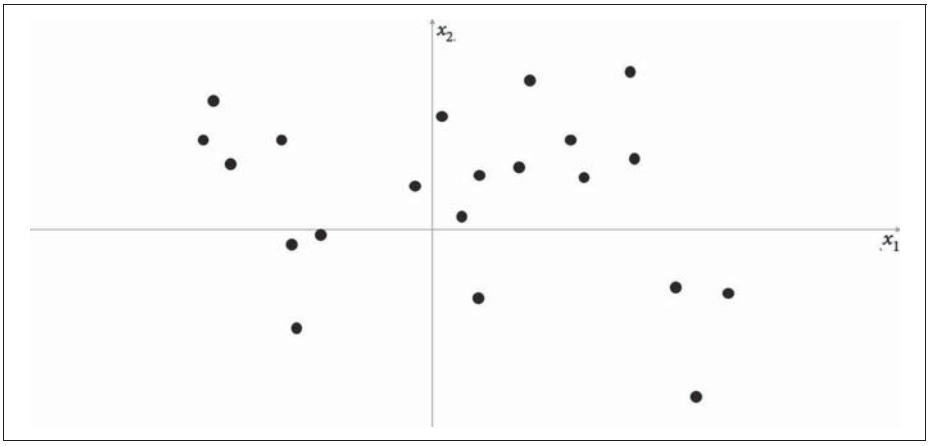

Abbildung 1-4: Eine Punktmenge im zweidimensionalen Raum, die durch die unbekannte Regel p_{data} erzeugt wurde

Für welchen Punkt haben Sie sich entschieden? Wahrscheinlich haben Sie Ihre Kenntnisse über die vorhandenen Datenpunkte genutzt, um vor Ihrem geistigen Auge ein Modell, p_{model}, zu konstruieren, mit dem Sie abschätzen, an welchen Stellen die Punkte eher zu finden sind. In diesem Zusammenhang stellt p_{model} eine *Schätzung* von p_{data} dar. Vielleicht haben Sie entschieden, dass p_{model} wie Abbildung 1-5 aussehen sollte – eine rechteckige Fläche, in der Punkte aufzufinden sind, und ein Bereich außerhalb dieser Fläche, in dem keine Wahrscheinlichkeit besteht, einen Punkt zu finden. Um eine neue Beobachtung zu erzeugen, können Sie einfach einen zufälligen Punkt innerhalb der Fläche wählen – oder, formeller ausgedrückt, aus der Verteilung p_{model} eine Stichprobe ziehen beziehungsweise *samplen*. Herzlichen Glückwunsch, Sie haben gerade Ihr erstes generatives Modell entwickelt!

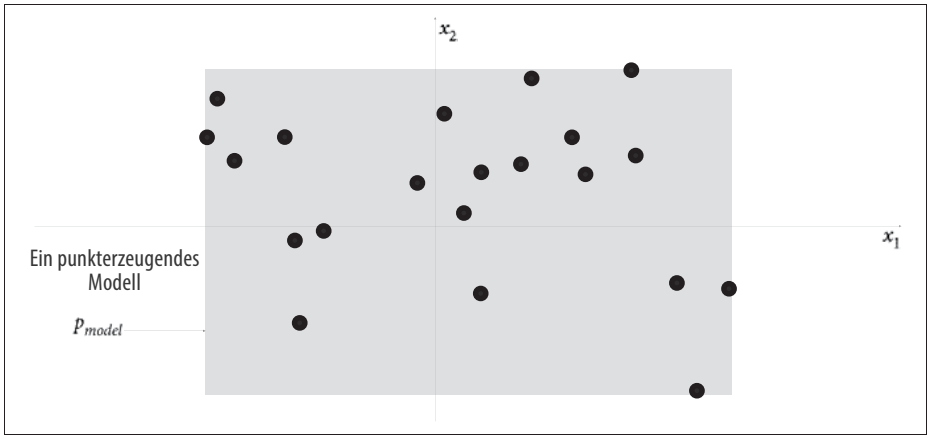

Abbildung 1-5: Die orangefarbene Fläche, p_{model}, ist eine Schätzung der wahren zugrunde liegenden datengenerierenden Verteilung, p_{data}.

Wenn dies auch nicht das komplexeste Beispiel gewesen sein mag, so haben wir dennoch einen ersten Einblick darin gewinnen können, worauf generative Modellierung abzielt. Fassen wir die Grundprinzipien noch einmal zusammen.

Die Grundprinzipien der generativen Modellierung

- Wir haben einen Datensatz an Beobachtungen **X**.
- Wir gehen davon aus, dass die Beobachtungen auf der Grundlage einer unbekannten Verteilung, p_{data}, generiert wurden.
- Ein generatives Modell p_{model} versucht, p_{data} nachzubilden. Wenn wir dies erreichen, können wir aus p_{model} Stichproben ziehen, um Beobachtungen zu erzeugen, die auch p_{data} entstammen könnten.
- Uns überzeugt p_{model}, falls folgende Regeln gelten:
  - Regel 1: Es kann Beispiele erzeugen, die ebenfalls aus p_{data} stammen könnten.
  - Regel 2: Es kann Beispiele generieren, die sich in ausreichendem Maße von den Beobachtungen in **X** unterscheiden. Mit anderen Worten, das Modell sollte nicht einfach Dinge reproduzieren, die es zuvor gesehen hat.

Ergründen wir nun die wahre datengenerierende Verteilung p_{data} und übertragen wir darauf die Grundprinzipien.

Wie wir in Abbildung 1-6 sehen können, ist der zugrunde liegende Datengenerierungsprozess schlicht eine Gleichverteilung der Punkte über die Kontinente hinweg, wobei kein Punkt im Meer zu finden ist.

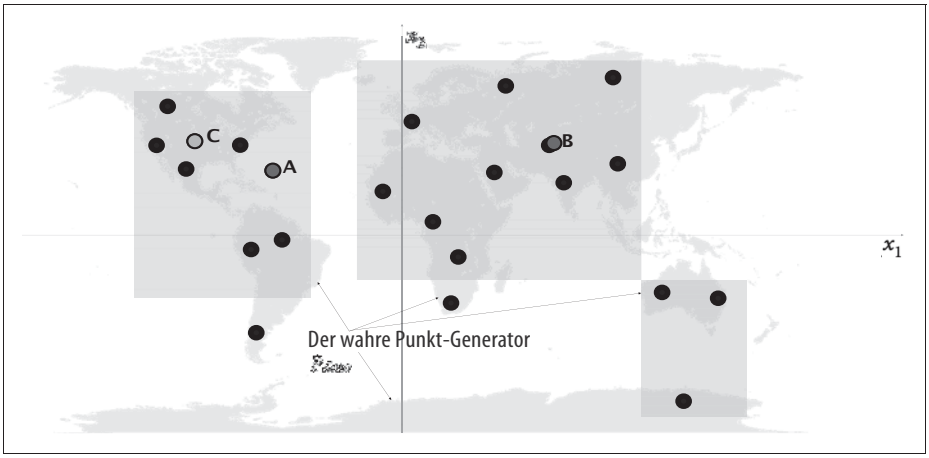

Abbildung 1-6: Die orangefarbenen Flächen, p_{model}, stellen eine Schätzung der tatsächlichen datenerzeugenden Verteilung, p_{data}, dar (die grauen Flächen).

Offensichtlich ist unser Modell p_{model} eine zu starke Vereinfachung von p_{data}. Die Punkte A, B und C heben drei Beobachtungen hervor, die von p_{model} mit unterschiedlichem Erfolg erzeugt wurden:

- **Punkt A** verstößt gegen Regel 1 – er scheint nicht von p_{data} generiert worden zu sein, da er sich mitten auf dem Meer befindet.
- **Punkt B** ist so nah an einem Punkt im Datensatz, dass wir nicht damit zufrieden sein sollten, wenn unser Modell einen solchen Punkt erzeugt. Wenn alle vom Modell erzeugten Beispiele so wären, würde dies gegen Regel 2 verstoßen.
- **Punkt C** kann als Erfolg gewertet werden, da er von p_{data} generiert worden sein könnte und sich in ausreichendem Maße von jedem Punkt im ursprünglichen Datensatz unterscheidet.

Das Anwendungsfeld der generativen Modellierung ist vielfältig, und es sind unterschiedlichste Aufgabenstellungen denkbar. In den meisten Anwendungsfällen bieten die Grundprinzipien der generativen Modellierung jedoch einen Leitfaden dazu, wie wir im Großen und Ganzen über die Lösung der Aufgabe nachdenken sollten.

Befassen wir uns nun mit einem weniger trivialen Beispiel für ein generatives Modell.

Wahrscheinlichkeitsbasierte generative Modelle

Eines vorweg – auch wenn Sie nie Statistik studiert haben, sollten Sie sich keine Sorgen machen! Um die meisten Deep-Learning-Modelle, die wir später in diesem Buch sehen werden, zu entwickeln und zu verwenden, ist es nicht notwendig, ein

ausgeprägtes Wissen über die dahinterstehende statistische Theorie zu besitzen. Um jedoch eine umfassende Einschätzung der Entwicklungen zu gewinnen, lohnt es sich, ein generatives Modell zu entwickeln, das nicht auf Methoden des Deep Learning basiert, sondern ausschließlich auf der Wahrscheinlichkeitstheorie. Dieser Ansatz vermittelt Ihnen die Grundlagen, um alle generativen Modelle, ob basierend auf Deep Learning oder nicht, aus der gleichen wahrscheinlichkeitstheoretischen Sichtweise verstehen zu können.

Wenn Sie bereits über profunde statistische Kenntnisse verfügen, ist das großartig, und ein Großteil des folgenden Abschnitts dürfte Ihnen geläufig sein. Es sei jedoch auf ein lustiges Beispiel in der Mitte dieses Kapitels hingewiesen, das sich zu lesen lohnt!

Als Erstes definieren wir vier Grundbegriffe: *Stichprobenraum*, *Wahrscheinlichkeitsdichtefunktion*, *parametrische Modellierung* und *Maximum-Likelihood-Schätzung*.

Der Stichprobenraum

Der *Stichprobenraum* ist der gesamte Wertebereich, den eine Beobachtung $\mathbf{x}$ annehmen kann.

In unserem vorherigen Beispiel besteht der Stichprobenraum aus allen auf der Weltkarte möglichen Kombinationen aus Breiten- und Längengrad $\mathbf{x} = (x_1, x_2)$.

$\mathbf{x} = (40{,}7306°, -73{,}9352°)$ ist z. B. ein Punkt im Stichprobenraum (New York City).

Die Wahrscheinlichkeitsdichtefunktion

Eine *Wahrscheinlichkeitsdichtefunktion* (oder *Dichtefunktion*), $p(\mathbf{x})$, ist eine Funktion, die einen Punkt $\mathbf{x}$ des Stichprobenraums auf einen Wertebereich zwischen 0 und 1 abbildet. Die Summe[5] der Dichtefunktion aller Punkte im Stichprobenraum muss 1 betragen, damit die Wahrscheinlichkeitsverteilung eindeutig definiert ist.[6]

Im unserem Weltkartenbeispiel ist die Dichtefunktion unseres Modells außerhalb der orangefarbenen Bereiche 0 und innerhalb konstant.

Obwohl es nur eine wahre Dichtefunktion p_{data} gibt, von der angenommen wird, dass sie den beobachtbaren Datensatz erzeugt hat, gibt es unendlich viele Dichte-

5 Beziehungsweise das Integral, wenn der Stichprobenraum kontinuierlich ist.

6 Wenn der Stichprobenraum diskret ist, entspricht $p(\mathbf{x})$ der dem Beobachtungspunkt $\mathbf{x}$ zuzuordnenden Wahrscheinlichkeit.

funktionen p_{model}, mit denen wir p_{data} schätzen können. Um unseren Ansatz zur Suche nach einem geeigneten $p_{model}(\mathbf{X})$ zu strukturieren, können wir eine Technik verwenden, die als *parametrische Modellierung* bekannt ist.

Parametrische Modellierung

Ein *parametrisches Modell*, $p_\theta(\mathbf{x})$, stellt eine Gruppe von Dichtefunktionen dar, die mit einer endlichen Anzahl an Parametern, θ, beschrieben werden können.

Die Gruppe aller möglichen Flächen, die man in Abbildung 1-5 zeichnen kann, stellen ein Beispiel eines parametrischen Modells dar. In diesem Fall gibt es vier Parameter: die Koordinaten der linken unteren (θ_1, θ_2) und der rechten oberen (θ_3, θ_4) Ecke der Fläche.

In diesem parametrischen Modell kann somit jede Dichtefunktion $p_\theta(\mathbf{x})$ (d. h. jede Fläche) eindeutig durch vier Zahlen, $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$, dargestellt werden.

Die Wahrscheinlichkeitsfunktion

Die *Wahrscheinlichkeitsfunktion* oder auch *Likelihood* $\mathscr{L}(\theta|\mathbf{x})$ eines Parametersatzes θ ist eine Funktion, die die Wahrscheinlichkeit von θ unter Berücksichtigung eines beobachteten Punkts $\mathbf{x}$ misst.

Sie ist wie folgt definiert:

$$\mathscr{L}(\theta|\mathbf{x}) = p_\theta(\mathbf{x})$$

Das bedeutet, dass die Wahrscheinlichkeit von θ bei einem Beobachtungspunkt $\mathbf{x}$ durch den Wert der durch θ parametrisierten Dichtefunktion an dem Punkt $\mathbf{x}$ definiert ist.

Wenn wir einen kompletten Datensatz $\mathbf{X}$ unabhängiger Beobachtungen haben, dann können wir formulieren:

$$\mathscr{L}(\theta|\mathbf{X}) = \prod_{\mathbf{x} \in \mathbf{X}} p_\theta(\mathbf{x})$$

Da die Berechnung dieses Produkts ziemlich rechenintensiv sein kann, verwenden wir stattdessen oft die *Log-Likelihood* ℓ:

$$\ell(\theta|\mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} \log p_\theta(\mathbf{x})$$

Es gibt statistische Gründe dafür, dass die Wahrscheinlichkeit auf diese Weise definiert wird. Zu unseren Zwecken genügt es, die Idee dahinter nachvollziehen zu können. Wir definieren einfach die Wahrscheinlichkeit, dass ein Parametersatz θ gleich der Wahrscheinlichkeit ist, die Daten unter dem von θ parametrierten Modell zu erfassen.

Im Weltkartenbeispiel hätte eine orangefarbene Fläche, die nur die linke Hälfte der Karte bedeckt, eine Wahrscheinlichkeit von 0 – der Datensatz hätte unmöglich durch sie generiert werden können, da sich auch Punkte auf der rechten Hälfte der Karte befinden. In Abbildung 1-5 besitzt die orangefarbene Fläche eine positive Wahrscheinlichkeit, da bei diesem Modell die Dichtefunktion aller Datenpunkte positiv ist.

Folglich liegt der Schwerpunkt der parametrischen Modellierung darin, den optimalen Wert $\hat{\theta}$ für den Parametersatz zu finden – also den Wert, der die Wahrscheinlichkeit maximiert, Datensatz $\mathbf{X}$ zu beobachten. Aus diesem Grund wird die Methode als *Maximum-Likelihood-Schätzung* bezeichnet.

Maximum-Likelihood-Schätzung

Die *Maximum-Likelihood-Schätzung* ermöglicht uns, $\hat{\theta}$ zu schätzen, also den Parametersatz θ einer Dichtefunktion $p_\theta(\mathbf{x})$, der am wahrscheinlichsten die beobachteten Daten $\mathbf{X}$ erklärt.

Formaler:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\ \mathscr{L}(\theta|\mathbf{X})$$

$\hat{\theta}$ wird auch als *Maximum-Likelihood-Schätzer* bezeichnet (MLS).

Nun sind Ihnen alle Begriffe geläufig, mit denen wir ein probabilistisches generatives Modell beschreiben können.

Die meisten Kapitel in diesem Buch werden eine kurze Geschichte enthalten, die dabei helfen soll, die jeweils vorliegende Methode zu umschreiben. In diesem Kapitel beginnen wir mit einer Reise zum Planeten Wrodl, wo unsere erste generative Modellierungsaufgabe wartet ...

Hallo Wrodl!

Wir schreiben das Jahr 2047, und Sie freuen sich, dass Sie zum neuen Chief Fashion Officer (CFO) des Planeten Wrodl ernannt wurden. Als CFO liegt es in Ihrer alleinigen Verantwortung, neue und aufregende Modetrends zu schaffen, die die Bewohner des Planeten begeistern.

Die Wrodler sind bekannt dafür, dass sie in Sachen Mode ziemlich speziell sind. Es liegt also an Ihnen, neue Styles zu erschaffen, die denen ähnlich sind, die es bereits auf dem Planeten gibt – sie sollen aber nicht identisch sein.

Bei Ihrer Ankunft erhalten Sie einen Datensatz mit 50 Beobachtungen der Wrodler-Mode (siehe Abbildung 1-7) und erfahren, dass Sie einen Tag Zeit haben, um zehn neue Modelle zu entwickeln, die Sie der Modepolizei zur Begutachtung vorlegen. Sie dürfen mit Frisur, Haarfarbe, Brille, Kleidungstyp und Kleidungsfarbe experimentieren, um Ihre Meisterwerke zu kreieren.

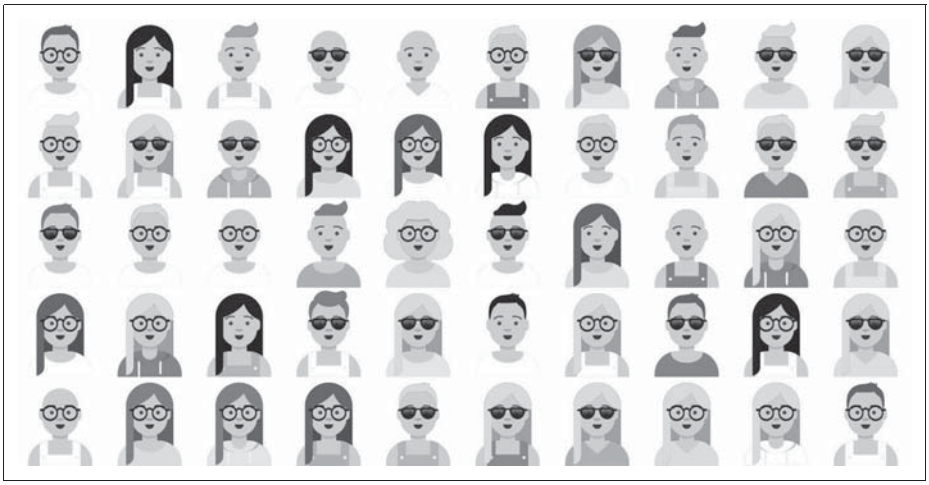

Abbildung 1-7: Porträts von 50 Wrodlern[7]

Da Sie ein begeisterter Datenwissenschaftler sind, entscheiden Sie sich zur Lösung des Problems kurzerhand für den Einsatz eines generativen Modells. Nach einem kurzen Besuch der intergalaktischen Bibliothek nehmen Sie das Buch *Generatives Deep Learning* in die Hand und beginnen zu lesen ...

Fortsetzung folgt …

Ihr erstes wahrscheinlichkeitsbasiertes generatives Modell

Werfen wir einen genaueren Blick auf den Wrodl-Datensatz. Er besteht aus $N = 50$ Beobachtungen von Kleidungsstilen, die derzeit auf dem Planeten verbreitet sind. Wie in Tabelle 1-1 aufgeführt, kann jede Beobachtung durch fünf Merkmale (*accessoirArt*, *kleidungFarbe*, *kleidungArt*, *haarFarbe*, *oberteilArt*) beschrieben werden.

Tabelle 1-1: Die ersten zehn Beobachtungen im Wrodler-Porträtdatensatz

portraet_id	accessoirArt	kleidungFarbe	kleidungArt	haarFarbe	oberteilArt
0	Rund	Weiß	ShirtSchaufelausschnitt	Rot	KurzhaarGlatt
1	Rund	Weiß	Kittel	SilberGrau	KurzhaarKräuselig

7 Quelle: *https://getavataaars.com*

Tabelle 1-1: Die ersten zehn Beobachtungen im Wrodler-Porträtdatensatz (Fortsetzung)

portraet_id	accessoirArt	kleidungFarbe	kleidungArt	haarFarbe	oberteilArt
2	Sonnenbrille	Weiß	ShirtSchaufelausschnitt	Blond	KurzhaarGlatt
3	Rund	Weiß	ShirtSchaufelausschnitt	Rot	Langhaar
4	Rund	Weiß	Kittel	SilberGrau	KeinHaar
5	OhneBrille	Weiß	Kittel	Schwarz	Langhaar
6	Sonnenbrille	Weiß	Kittel	SilberGrau	Langhaar
7	Rund	Weiß	ShirtSchaufelausschnitt	SilberGrau	KurzhaarGlatt
8	Rund	Rosa	Kapuzenpullover	SilberGrau	Langhaar
9	Rund	PastellOrange	ShirtSchaufelausschnitt	Blond	Langhaar

Die Merkmale können folgende Ausprägungen annehmen:

- Sieben verschiedene Frisuren (*oberteilArt*):
  - *KeinHaar*, *LanghaarDutt*, *LanghaarLockig*, *Langhaar*, *KurzhaarLockig*, *KurzhaarGlatt*, *KurzhaarKräuselig*
- Sechs verschiedene Haarfarben (*haarFarbe*):
  - *Schwarz*, *Blond*, *Braun*, *PastellRosa*, *Rot*, *SilberGrau*
- Drei unterschiedliche Brillentypen (*accessoirArt*):
  - *OhneBrille*, *RundeBrille*, *Sonnenbrille*
- Vier verschiedene Kleidungsstücke (*kleidungArt*):
  - *Kapuzenpullover*, *Kittel*, *HemdSchaufelausschnitt*, *HemdVausschnitt*
- Acht verschiedene Konfektionsfarben (*kleidungFarbe*):
  - *Schwarz*, *Blau*, *Grau*, *PastellGrün*, *PastellOrange*, *Rosa*, *Rot*, *Weiß*

Es gibt $7 \times 6 \times 3 \times 4 \times 8 = 4.032$ verschiedene Kombinationen für die Merkmale. Dementsprechend gibt es im Stichprobenraum 4.032 Datenpunkte.

Unser Datensatz scheint von einer zugrunde liegenden Verteilung p_{data} erzeugt worden zu sein, die einige Merkmalswerte gegenüber anderen bevorzugt. Zum Beispiel kann man anhand der Porträts in Abbildung 1-7 erkennen, dass weiße Kleidungsstücke häufiger getragen werden, ebenso silbergraue Haare und T-Shirts mit Rundhalsausschnitt.

Das Problem ist, dass wir p_{data} nicht explizit kennen – wir müssen mit der uns zur Verfügung stehenden Stichprobe an Beobachtungen $\mathbf{X}$ auskommen, die durch p_{data} erzeugt wurde. Das Ziel der generativen Modellierung ist es, aus diesen Beobachtungen ein p_{Modell} zu entwickeln, das die von p_{data} erzeugten Beobachtungen akkurat nachbilden kann.

Um dies zu erreichen, könnten wir einfach – basierend auf unserer Stichprobe – jeder möglichen Merkmalskombination eine Wahrscheinlichkeit zuordnen. Unser parametrisches Modell hätte dann $d = 4.031$ Parameter – einen für jede Beobach-

tung im Stichprobenraum, abzüglich eines Parameters, da der Wert des letzten Parameters zwangsläufig so gewählt werden müsste, dass die Gesamtsumme 1 beträgt. Als Notation der zu schätzenden Modellparameter wählen wir $(\theta_1, \ldots, \theta_{4031})$.

Im vorliegenden Fall wird unser parametrisches Modell durch eine *multinomiale Verteilung* beschrieben. Der Maximum-Likelihood-Schätzer $\widehat{\theta_j}$ jedes Parameters ist entsprechend durch folgende Gleichung gegeben:

$$\widehat{\theta_j} = \frac{n_j}{N}$$

wobei n_j der Anzahl der Beobachtungen der Merkmalskombination j in unserem Datensatz und $N = 50$ der Gesamtanzahl der Beobachtungen entspricht.

Somit gleicht der Schätzwert jedes Parameters der entsprechenden Merkmalskombination schlicht der jeweiligen relativen Häufigkeit im Datensatz.

Beispielsweise taucht die folgende Merkmalskombination (bezeichnen wir sie als Kombination 1) zweimal im Datensatz auf:

- (*Langhaar, Rot, RundeBrille, HemdSchaufelausschnitt, Weiß*)

Deshalb ergibt sich:

$$\widehat{\theta_1} = 2/50 = 0{,}04$$

Dagegen erscheint die folgende Merkmalskombination (nennen wir sie Kombination 2) überhaupt nicht im Datensatz:

- (*Langhaar, Rot, RundeBrille, HemdSchaufelausschnitt, Blau*)

Hierfür ergibt sich:

$$\widehat{\theta_2} = 0/50 = 0$$

Auf diese Weise können wir alle $\widehat{\theta_j}$-Werte berechnen und so eine Verteilung mittels unseres Stichprobenraums definieren. Da wir aus dieser Verteilung wiederum Stichproben ziehen können, könnte unser Ansatz durchaus als generatives Modell fungieren. Allerdings scheitert er an einem entscheidenden Punkt: Das Modell könnte nie etwas erzeugen, was es nicht bereits gesehen hat, da für jede Kombination, die nicht im ursprünglichen Datensatz $\mathbf{X}$ war, $\widehat{\theta_j} = 0$ gilt.

Um diesem Umstand Rechnung zu tragen, könnten wir jeder möglichen Merkmalskombination einen zusätzlichen *Pseudozählwert* von 1 zuweisen. Diese Methode wird als *additive Glättung* bezeichnet. In dem Fall wäre der MLS für die Parameter:

$$\widehat{\theta_j} = \frac{n_j + 1}{N + d}$$

Nun besitzt jede einzelne Kombination eine Ziehungswahrscheinlichkeit ungleich null – auch wenn sie nicht im ursprünglichen Datensatz enthalten war. Dieser Ansatz bietet jedoch noch immer kein zufriedenstellendes generatives Modell, da die Wahrscheinlichkeit, eine Kombination zu beobachten, die nicht im ursprünglichen Datensatz enthalten ist, lediglich eine Konstante ist. Würden wir versuchen, ein solches Modell zur Erzeugung von Picasso-Gemälden zu verwenden, würde es einer zufälligen Zusammenstellung bunter Pixel ebenso viel Gewicht beimessen wie einer Nachbildung eines Picasso-Gemäldes, das sich nur geringfügig von einem echten Gemälde unterscheidet.

Idealerweise soll unser generatives Modell eine inhärente Struktur aus den Daten lernen. Dementsprechend sollte es Bereiche des Stichprobenraums, die seiner Einschätzung nach wahrscheinlicher sind, stärker gewichten, statt lediglich Gewicht auf die Punkte zu legen, die tatsächlich im Datensatz vorhanden sind.

Um das zu erreichen, benötigen wir ein anderes parametrisches Modell.

Das naive Bayes-Modell

Das naive Bayes-Modell bedient sich einer einfachen Annahme, die die Anzahl der Parameter, die geschätzt werden müssen, drastisch reduziert.

Es wird die *naive* Annahme getroffen, dass jedes Merkmal x_j von jedem anderen Merkmal x_k *unabhängig* ist.[8] In Bezug auf den Wrodl-Datensatz gehen wir davon aus, dass die Wahl der Haarfarbe keinen Einfluss auf die Wahl des Kleidungstyps hat und dass die Art der Brille, die jemand trägt, zum Beispiel keinen Einfluss auf seine Frisur hat. Formeller ausgedrückt, gilt für alle Merkmale x_j, x_k:

$$p\left(x_j \middle| x_k\right) = p\left(x_j\right)$$

Diese Gleichung drückt die *Naive-Bayes*-Annahme aus. Um sie anzuwenden, nutzen wir für die einzelnen Wahrscheinlichkeiten jeweils die Kettenregel, um die Dichtefunktion als Produkt bedingter Wahrscheinlichkeiten zu formulieren:

$$\begin{aligned} p(\mathbf{x}) &= p(x_1, \ldots, x_K) \\ &= p(x_2, \ldots, x_K | x_1) p(x_1) \\ &= p(x_3, \ldots, x_K | x_1, x_2) p(x_2 \mid x_1) p(x_1) \\ &= \prod_{k=1}^{K} p(x_k | x_1, \ldots, x_{k-1}) \end{aligned}$$

wobei K die Anzahl der Merkmale ist (d.h. 5 in unserem Wrodl-Beispiel).

8 Wenn eine Antwortgröße y vorhanden ist, besagt die Annahme von Naive Bayes, dass es eine *bedingte* Unabhängigkeit zwischen jedem Paar von Merkmalen x_j, x_k, gegeben y, gibt.

Wenden wir nun die Naive-Bayes-Annahme an, um die letzte Zeile weiter zu vereinfachen:

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

Das ist das naive Bayes-Modell. Die Aufgabe reduziert sich darauf, die Parameter $\theta_{kl} = p(x_k = l)$ für jedes Merkmal einzeln zu schätzen und zu multiplizieren, um die Wahrscheinlichkeiten jeder möglichen Kombination zu finden.

Wie viele Parameter müssen wir nun schätzen? Für jedes Merkmal müssen wir je einen Parameter für jede Merkmalsausprägung schätzen. Daher ist das Modell im Wrodl-Beispiel nur durch 7 + 6 + 3 + 4 + 8 – 5 = 23 Parameter definiert.[9]

Die Maximum-Likelihood-Schätzer $\widehat{\theta_{kl}}$ ergeben sich wie folgt:

$$\widehat{\theta_{kl}} = \frac{n_{kl}}{N}$$

wobei n_{kl} die Häufigkeit angibt, mit der das Merkmal k die Ausprägung l im Datensatz annimmt und $N = 50$ die Gesamtanzahl der Beobachtungen.

Tabelle 1-2 weist die berechneten Parameter für den Wrodl-Datensatz aus.

Tabelle 1-2: Die MLS der Parameter im naiven Bayes-Modell

oberteilArt	n	$\hat{\theta}$	haarFarbe	n	$\hat{\theta}$	kleidungFarbe	n	$\hat{\theta}$
KeinHaar	7	0,14	Schwarz	7	0,14	Schwarz	0	0,00
LanghaarDutt	0	0,00	Blond	6	0,12	Blau	4	0,08
LanghaarLockig	1	0,02	Braun	2	0,04	Grau	10	0,20
Langhaar	23	0,46	PastellRosa	3	0,06	PastellGrün	5	0,10
KurzhaarLockig	1	0,02	Rot	8	0,16	PastellOrange	2	0,04
KurzhaarGlatt	11	0,22	SilberGrau	24	0,48	Rosa	4	0,08
KurzhaarKräuselig	7	0,14	*Insgesamt*	*50*	*1,00*	Rot	3	0,06
Insgesamt	*50*	*1,00*				Weiß	22	0,44
						Insgesamt	*50*	*1,00*

9 Die -5 ist darauf zurückzuführen, dass jeweils der letzte Parameter jedes Merkmals gegeben ist, da die Summe der Parameter für ein Merkmal 1 ergeben muss.

Tabelle 1-2: Die MLS der Parameter im naiven Bayes-Modell (Fortsetzung)

accessoirArt	n	$\hat{\theta}$	kleidungArt	n	$\hat{\theta}$
OhneBrille	11	0,22	Kapuzenpullover	7	0,14
RundeBrille	22	0,44	Kittel	18	0,36
Sonnenbrille	17	0,34	HemdSchaufelausschnitt	19	0,38
Insgesamt	*50*	*1,00*	HemdVausschnitt	6	0,12
			Insgesamt	*50*	*1,00*

Um die Wahrscheinlichkeit zu berechnen, mit der das Modell eine Beobachtung **x** erzeugt, multiplizieren wir einfach die einzelnen Merkmalswahrscheinlichkeiten, beispielsweise:

$$
\begin{aligned}
&p(\textit{Langhaar, Rot, RundeBrille, HemdSchaufelausschnitt, Weiß}) \\
&= p(\textit{Langhaar}) \times p(\textit{Rot}) \times p(\textit{RundeBrille}) \times p(\textit{HemdSchaufelausschnitt}) \times p(\textit{Weiß}) \\
&= 0{,}46 \times 0{,}16 \times 0{,}44 \times 0{,}38 \times 0{,}44 \\
&= 0{,}0054
\end{aligned}
$$

Beachten Sie, dass diese Kombination nicht im ursprünglichen Datensatz enthalten ist, unser Modell ihr aber dennoch eine Wahrscheinlichkeit ungleich null zuweist. Deshalb kann sie überhaupt von unserem Modell erzeugt werden. Außerdem besitzt sie eine höhere Wahrscheinlichkeit, erzeugt zu werden, als z. B. *(Langhaar, Rot, RundeBrille, HemdSchaufelausschnitt, Blau)*, da weiße Kleidungsstücke im Datensatz häufiger vorkommen als blaue.

Ein naives Bayes-Modell ist also in der Lage, eine Struktur aus den vorliegenden Daten zu erlernen und neue Beispiele zu erzeugen, die im ursprünglichen Datensatz nicht enthalten waren. Dabei schätzt es die Wahrscheinlichkeiten unter der Annahme, dass die Merkmalsausprägungen unabhängig voneinander auftreten. Auf diese Weise können die Einzelwahrscheinlichkeiten miteinander multipliziert werden, um die vollständige Dichtefunktion, $p_\theta(\mathbf{x})$, zu erhalten.

In Abbildung 1-8 sind zehn vom Modell erstellte Porträts zu sehen.

Abbildung 1-8: Zehn neue Wrodler-Stile, die vom naiven Bayes-Modell erzeugt wurden

Bei diesem einfachen Anwendungsfall erscheint die Annahme des naiven Bayes-Modells, dass jedes Merkmal unabhängig von jedem anderen ist, durchaus plausibel, und es erweist sich daher auch als ein geeignetes generatives Modell.

Sehen wir uns nun an, was passiert, wenn diese Annahme verletzt wird.

Hallo Wrodl! – Fortsetzung

Mit Stolz blicken Sie auf die zehn neuen Kreationen, die von Ihrem naiven Bayes-Modell erzeugt wurden. Von Erfolg getrieben, widmen Sie Ihre Aufmerksamkeit nun dem Mode-Dilemma eines anderen Planeten – aber diesmal ist die Aufgabenstellung nicht ganz so einfach.

Auf dem zutreffend benannten Planeten Pixel besteht der Ihnen vorliegende Datensatz nicht mehr aus den fünf übergeordneten Merkmalen, die Sie auf dem Planeten Wrodl kennengelernt haben (*haarFarbe, accessoirArt,* etc.). Hier stehen Ihnen lediglich die Werte der 32 × 32 Pixel, die zusammen ein Bild ergeben, zur Verfügung. Jede Beobachtung hat somit 32 × 32 = 1.024 Merkmale, und jedes Merkmal kann einen von 256 Werten annehmen (aller verfügbaren Farben in der Farbpalette).

Die im neuen Datensatz enthaltenen Modestile können Sie in Abbildung 1-9 überblicken. Tabelle 1-3 bietet Ihnen einen anschaulichen Auszug ausgewählter Pixelwerte der ersten zehn Beobachtungen.

Abbildung 1-9: Mode auf dem Planeten Pixel

Tabelle 1-3: Die Pixelwerte 458 bis 467 der ersten zehn Beobachtungen auf dem Planeten Pixel

face_id	px_458	px_459	px_460	px_461	px_462	px_463	px_464	px_465	px_466	px_467
0	49	14	14	19	7	5	5	12	19	14
1	43	10	10	17	9	3	3	18	17	10
2	37	12	12	14	11	4	4	6	14	12
3	54	9	9	14	10	4	4	16	14	9
4	2	2	5	2	4	4	4	4	2	5
5	44	15	15	21	14	3	3	4	21	15
6	12	9	2	31	16	3	3	16	31	2
7	36	9	9	13	11	4	4	12	13	9
8	54	11	11	16	10	4	4	19	16	11
9	49	17	17	19	12	6	6	22	19	17

Sie beschließen, Ihr bewährtes naives Bayes-Modell nun an dem Pixeldatensatz auszuprobieren. Die Farbverteilung der Pixel wird dabei durch die Maximum-Likelihood-Parameter bestimmt. Sobald diese durch das Modell geschätzt wurden, ist es imstande, neue Beobachtungen auf Basis der resultierenden Verteilung zu erzeugen. Nur leider zeigt sich deutlich, dass dabei etwas nicht funktioniert.

Anstatt neuartige Kleidungsstile zu erzeugen, gibt das Modell zehn sehr ähnliche Bilder aus, die keine erkennbaren Accessoires oder klare Haarpartien oder Kleidungsstücke erkennen lassen (siehe Abbildung 1-10). Warum ist das so?

Abbildung 1-10: Zehn neue Planeten-Pixel-Stile, die durch das naive Bayes-Modell erzeugt wurden

Die Herausforderungen der generativen Modellierung

Erstens ist das naive Bayes-Modell nicht dazu imstande, zu erkennen, dass sich zwei benachbarte Pixel mit großer Wahrscheinlichkeit im Farbton sehr ähnlich sind, da sie beispielsweise Teil desselben Kleidungsstücks sind. Dies ist darauf zurückzuführ-

ren, dass die Pixel unabhängig voneinander ausgewertet werden. Das Modell ist zwar in der Lage, die Gesichtsfarbe und den Mund zu erzeugen, da die Pixel an dieser Stelle bei jeder Beobachtung im Trainingsdatensatz ungefähr den gleichen Farbton besitzen. Bei den Pixeln, die die T-Shirts definieren, wird jedoch jedes Pixel nach dem Zufallsprinzip aus einer Vielzahl von verschiedenen Farben im Trainingsdatensatz gesamplet – ohne dabei Rücksicht auf die bei benachbarten Pixeln gesampleten Farben zu nehmen. Zudem versagt es dabei, kreisförmige Brillenformen für Pixel in Augennähe oder rote Pixel in der Nähe des oberen Bildrands, die ein Wellenmuster bilden, das beispielsweise eine bestimmte Frisur abbildet, zu erzeugen.

Zweitens umfasst der Stichprobenraum in diesem Beispiel eine immens große Anzahl möglicher Beobachtungen. Nur ein sehr geringer Teil davon repräsentiert erkennbare Gesichter und ein noch geringerer auch Gesichter, die den Modevorstellungen des Planeten Pixel entsprechen. Wenn unser naives Bayes-Modell also direkt mit den hoch korrelierten Pixelwerten arbeitet, ist die Wahrscheinlichkeit, dass es eine überzeugende Farbkombination findet, äußerst gering.

Abschließend können wir festhalten, dass auf dem Planeten Wrodl die einzelnen Merkmale voneinander unabhängig sind und der Stichprobenraum relativ klein ist, weshalb das naive Bayes-Modell gut funktioniert. Auf dem Planeten Pixel ist die Annahme, dass jeder Pixelwert unabhängig von jedem anderen Pixelwert ist, offensichtlich nicht gültig. Die Pixelwerte sind hoch korreliert, und der Stichprobenraum ist groß, sodass es durch die unabhängige Auswertung der Pixel fast unmöglich ist, ein passendes Gesicht zu finden. Deshalb kann von naiven Bayes-Modellen auch nicht erwartet werden, dass sie bei Rohbilddaten gut funktionieren.

Dieses Beispiel verdeutlicht die beiden wichtigsten Herausforderungen, die ein erfolgreiches generatives Modell bewältigen muss.

Die Herausforderungen der generativen Modellierung

- Wie kann das Modell mit dem hohen Maß bedingter Abhängigkeit zwischen den Merkmalen umgehen?
- Wie findet das Modell in einem hochdimensionalen Stichprobenraum den (winzigen) Anteil an erzeugten Beobachtungen, die zufriedenstellend sind?

Deep Learning ist der Schlüssel zur Lösung dieser beiden Herausforderungen.

Anstatt bereits Annahmen im Voraus treffen zu müssen, benötigen wir ein Modell, das die relevanten Strukturen aus den Daten selbst ableiten kann. Genau hierbei zeichnet sich das Deep Learning aus. Das ist auch einer der Hauptgründe dafür, dass es für die wichtigsten Fortschritte der letzten Jahre in der generativen Modellierung verantwortlich war.

Die Tatsache, dass Deep Learning seine eigenen Merkmale in einem niederdimensionalen Raum finden kann, bedeutet eine Form des *Representation Learning*. Es ist wichtig, die Schlüsselkonzepte des Representation Learning zu verstehen, bevor wir Deep Learning im nächsten Kapitel detaillierter behandeln.

Representation Learning

Die Grundidee beim Lernen von Repräsentationen besteht darin, nicht den hochdimensionalen Stichprobenraum direkt zu modellieren, sondern jede Beobachtung im Trainingsdatensatz mittels eines niederdimensionalen *latenten* Raums zu erfassen. Das Modell muss dann eine Mapping-Funktion lernen, die einen Punkt im latenten Raum nehmen und ihn auf einen Punkt in dem ursprünglichen Raum abbilden kann. Mit anderen Worten: Jeder Punkt im latenten Raum ist die *Darstellung* eines hochdimensionalen Bilds.

Was bedeutet das in der Praxis? Nehmen wir an, wir hätten ein Trainingsdatensatz bestehend aus Graustufenbildern von Keksdosen (siehe Abbildung 1-11).

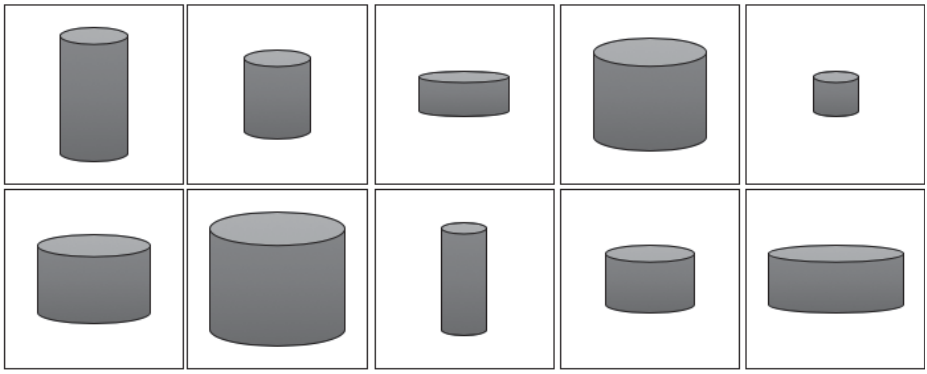

Abbildung 1-11: Der Keksdosen-Datensatz

Für uns ist es offensichtlich, dass es zwei Merkmale gibt, durch die eine Dose eindeutig definiert wird: die Höhe und die Breite der Dose. Bei einer vorgegebenen Höhe und Breite könnten wir die entsprechende Dose zeichnen, auch wenn ihr Bild nicht im Trainingsdatensatz enthalten war. Für eine Maschine ist dies jedoch nicht so einfach – sie müsste zunächst feststellen, dass Höhe und Breite die beiden latenten Raumdimensionen sind, die diesen Datensatz am besten beschreiben, und dann die Mapping-Funktion *f* lernen, die wiederum einen Punkt aus dem latenten Raum auf ein graustufiges Keksdosenbild abbilden kann. Der resultierende latente Raum von Keksdosen und der Erzeugungsprozess sind in Abbildung 1-12 abgebildet.

Deep Learning gibt uns die Möglichkeit, die oft sehr komplexe Mapping-Funktion *f* auf vielfältige Weise zu lernen. Wir werden einige der wichtigsten Methoden in späteren Kapiteln dieses Buchs aufgreifen. Im Moment genügt es, nachzuvollziehen, was durch das Lernen von Repräsentationen bezweckt wird.

Einer der Vorteile des Representation Learning ist, dass wir Operationen innerhalb des überschaubareren latenten Raums durchführen können, die sich auf die Eigenschaften des Bilds auf höchster Ebene auswirken. Es ist nicht offensichtlich, wie man die Schattierung jedes einzelnen Pixels anpassen muss, um ein bestimmtes Keksdosenbild *größer* zu machen. Im latenten Raum muss man lediglich eine 1 zur latenten Dimension *Höhe* hinzuaddieren und dann die Mapping-Funktion anwenden, um zur ursprünglichen Domäne zurückzukehren. Im nächsten Kapitel sehen wir das in Aktion, jedoch nicht für unser Keksdosenbeispiel, sondern anhand von Gesichtern.

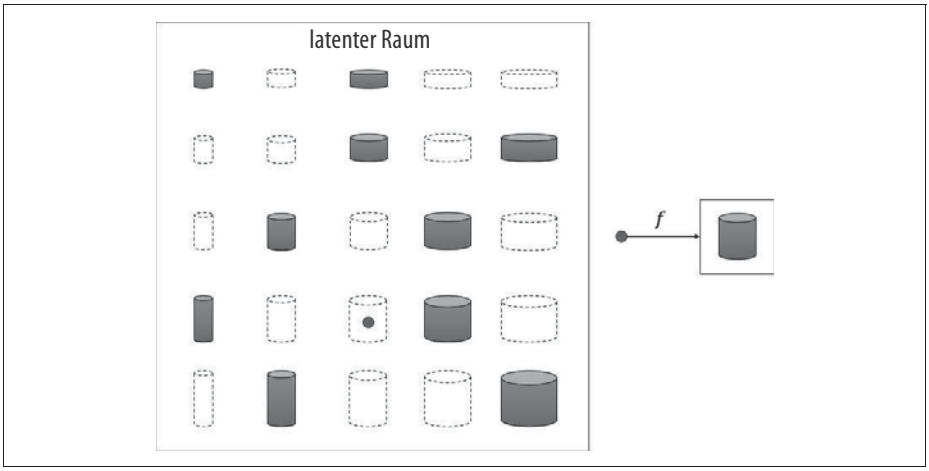

Abbildung 1-12: Der latente Keksdosenraum und die Funktion f, die einen Punkt im latenten Raum auf die ursprüngliche Bilddomäne abbildet

Das Lernen von Repräsentationen ist für uns Menschen so selbstverständlich, dass Sie vielleicht noch nie darüber nachgedacht haben, wie erstaunlich es ist, dass wir es so mühelos können. Angenommen, Sie wollten jemandem Ihr Aussehen beschreiben, der nach Ihnen in einer Menschenmenge sucht und nicht weiß, wie Sie aussehen. Sie würden nicht damit beginnen, die Farbe von Pixel 1 Ihres Haars anzugeben, dann Pixel 2, dann Pixel 3 etc. Stattdessen würden Sie die vernünftige Vermutung anstellen, dass die andere Person eine allgemeine Vorstellung davon hat, wie ein Mensch für gewöhnlich aussieht, und diese Grundvorstellung dann mit weiteren Merkmalen ergänzen, die Pixelgruppen beschreiben, wie z.B. »Ich habe hellblonde Haare« oder »Ich trage eine Brille«. Mit nicht mehr als zehn dieser Aussagen wäre die Person in der Lage, die Beschreibung wieder in Pixeln abzubilden, um ein Bild von Ihnen in ihrem Kopf zu erhalten. Das Bild wäre nicht perfekt, aber es gäbe eine ungefähre Ähnlichkeit mit Ihrem tatsächlichen Aussehen, sodass sie Sie unter möglicherweise Hunderten von anderen Menschen finden könnte, auch wenn Sie sich noch nie zuvor gesehen haben.

Beachten Sie, dass das Representation Learning bei einem bestimmten Bild nicht nur vorgegebenen Merkmalen wie *Blondheit der Haare*, *Höhe* etc. Werte zuweist.

Die eigentliche Stärke des Lernens von Repräsentationen besteht darin, dass es lernt, welche Merkmale am wichtigsten sind, um die gegebenen Beobachtungen zu beschreiben, und wie man diese Merkmale aus den Rohdaten gewinnt. Mathematisch gesehen, versucht das Modell, die hochgradig nicht lineare *Mannigfaltigkeit*, auf der die Daten liegen, zu finden und die Dimensionen zu bestimmen, die erforderlich sind, um diesen Raum vollständig zu beschreiben. Abbildung 1-13 verdeutlicht diese Grundidee.

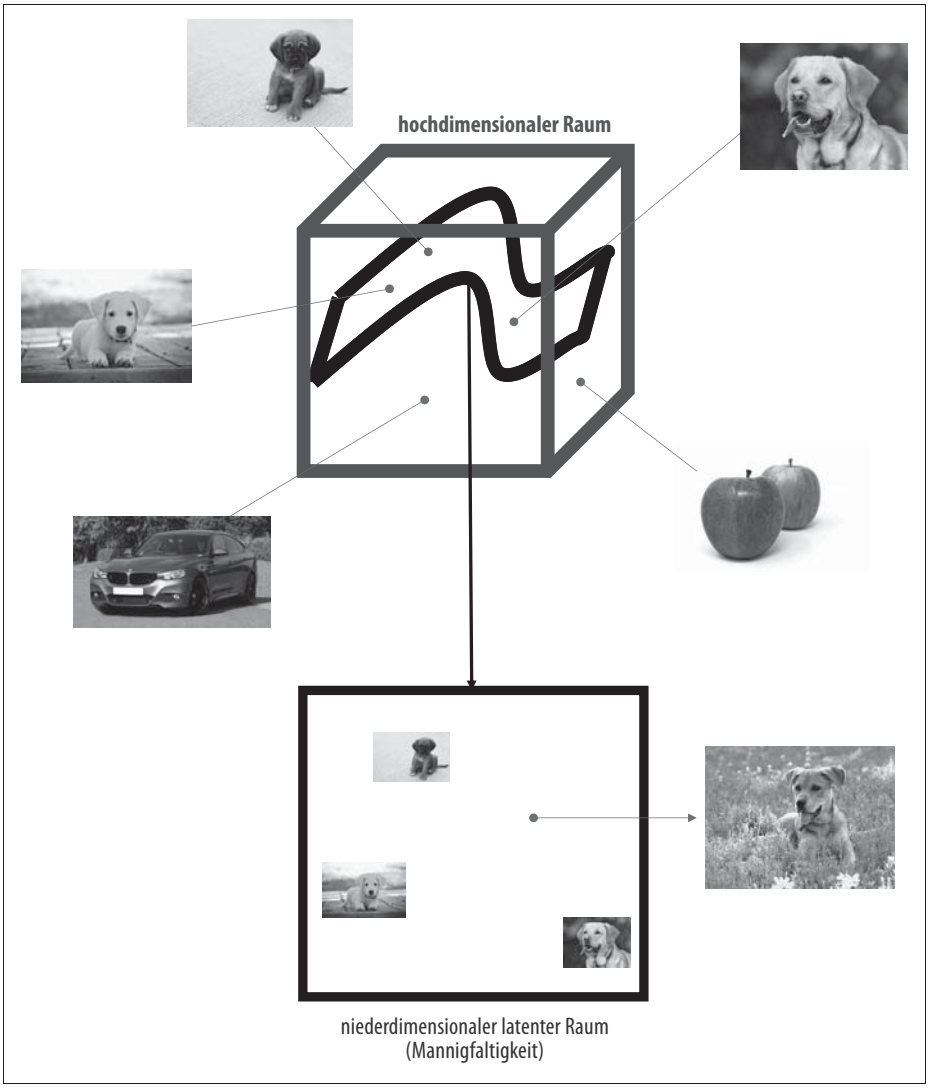

Abbildung 1-13: Der Würfel repräsentiert den extrem hochdimensionalen Raum aller Bilder; das Representation Learning versucht, den niederdimensionalen latenten Unterraum oder die Mannigfaltigkeit zu finden, in dem bestimmte Bildtypen liegen (z. B. die Hunde-Mannigfaltigkeit).

Zusammenfassend lässt sich sagen, dass das Representation Learning die relevantesten Merkmale auf übergeordneter Ebene ermittelt, die Auskunft darüber geben, wie Pixelgruppen dargestellt werden sollten, damit es wahrscheinlich ist, dass die einzelnen Punkte des latenten Raums ein authentisch aussehendes Bild wiedergeben. Indem wir die Merkmalswerte im latenten Raum anpassen, können wir neuartige Abbildungen erzeugen, die, zurück in die ursprüngliche Bilddomäne projiziert, eine bedeutend höhere Wahrscheinlichkeit besitzen, *echt* auszusehen, als wenn direkt mit den einzelnen Rohpixeln gearbeitet worden wäre.

Nachdem Sie nun ein Grundverständnis für das Representation Learning, das das Grundgerüst vieler der generativen Deep-Learning-Beispiele in diesem Buch bildet, erworben haben, bleibt nur noch, Ihre Arbeitsumgebung so zu gestalten, dass Sie mit dem Bau eigener generativer Deep-Learning-Modelle beginnen können.

Einrichten Ihrer Arbeitsumgebung

In diesem Buch gibt es zahlreiche funktionierende Beispielprogramme, die Ihnen nachvollziehbar nahebringen, wie man die im Text behandelten Modelle entwickelt.

Um Zugriff auf diese Beispiele zu erhalten, müssen Sie das Git-Repository klonen, das diesem Buch als Begleitmaterial dient. Git ist ein Open-Source-Versionskontrollsystem, das es Ihnen ermöglicht, den Code lokal zu kopieren, sodass Sie die Notebooks auf Ihrem eigenen Computer oder vielleicht in einer Cloud-basierten Umgebung ausführen können. Möglicherweise haben Sie diese bereits installiert. Wenn nicht, folgen Sie den für Ihr Betriebssystem relevanten Anweisungen (*http://bit.ly/2MUrvN1*).

Um das Repository für dieses Buch zu klonen, navigieren Sie mit dem Befehl `cd` zu dem Ordner, in dem Sie die Dateien speichern möchten, und geben Sie das Folgende in Ihre Eingabekonsole ein:

```
git clone https://github.com/davidADSP/GDL_code.git
```

Stellen Sie stets sicher, dass Sie die aktuellste Version der Codebasis haben, indem Sie den folgenden Befehl ausführen:

```
git pull
```

Sie sollten nun in der Lage sein, die Dateien in einem Ordner auf Ihrem Computer zu sehen.

Als Nächstes müssen Sie eine virtuelle Umgebung einrichten. Dies ist einfach ein Ordner, in den Sie neue Kopien von Python und all den Paketen, die wir in diesem Buch verwenden, installieren. Auf diese Weise stellen Sie sicher, dass Ihre Systemversion von Python durch keine der von uns verwendeten Bibliotheken beeinträchtigt wird.

Sollten Sie die Anaconda-Distribution nutzen, können Sie wie folgt eine virtuelle Umgebung einrichten:

```
conda create -n generative python=3.6 ipykernel
```

Wenn nicht, installieren Sie `virtualenv` und `virtualenvwrapper` mit dem Befehl:[10]

```
pip install virtualenv virtualenvwrapper
```

Sie müssen auch die folgenden Zeilen Ihrem Kommandozeilenstartskript hinzufügen (z. B. *.bash_profile*):

```
export WORKON_HOME=$HOME/.virtualenvs ❶
export VIRTUALENVWRAPPER_PYTHON=/usr/local/bin/python3 ❷
source /usr/local/bin/virtualenvwrapper.sh ❸
```

❶ Der Speicherort, an dem Ihre virtuellen Umgebungen gespeichert werden.

❷ Die standardmäßige Version von Python, die beim Erstellen einer virtuellen Umgebung verwendet wird – stellen Sie sicher, dass diese auf Python 3 und nicht auf Python 2 verweist.

❸ Lädt das Initialisierungsskript `virtualenvwrapper` neu.

Um eine virtuelle Umgebung mit dem Namen *generative* zu erstellen, geben Sie einfach Folgendes in Ihr Terminal ein:

```
mkvirtualenv generative
```

Sie befinden sich innerhalb der virtuellen Umgebung, wenn Ihr Terminal am Anfang der Eingabeaufforderung (`generative`) anzeigt.

Jetzt können Sie alle Pakete, die wir in diesem Buch einsetzen, mit dem folgenden Befehl installieren:

```
pip install -r requirements.txt
```

In diesem Buch verwenden wir Python 3. Die Datei *requirements.txt* enthält die Namen und Versionsnummern aller Pakete, die Sie für die Ausführung der Beispiele benötigen.

Um zu überprüfen, ob alles wie erwartet funktioniert, geben Sie aus Ihrer virtuellen Umgebung heraus in Ihr Terminal `python` ein und versuchen dann, Keras zu importieren (eine Deep-Learning-Bibliothek, die wir in diesem Buch ausgiebig nutzen werden). Sie sollten eine Python-3-Eingabeaufforderung sehen und wie in Abbildung 1-14 eine Statusmeldung dazu, dass Keras das TensorFlow-Backend verwendet.

10 Eine vollständige Anweisung zur Installation von `virtualenvwrapper` finden Sie in der Dokumentation (*http://bit.ly/2x8LPQ4*).

```
Python 3.6.5 (default, Oct  6 2018, 09:49:35)
[GCC 4.2.1 Compatible Apple LLVM 10.0.0 (clang-1000.11.45.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>> keras.__version__
'2.2.4'
```

Abbildung 1-14: Einrichten Ihrer Arbeitsumgebung

Zu guter Letzt müssen Sie sicherstellen, dass auf Ihrem Computer der Zugriff auf Ihre virtuelle Umgebung über Jupyter-Notebooks eingerichtet ist. Jupyter bietet die Möglichkeit, interaktiv in Ihrem Browser in Python zu programmieren oder auch neue Ideen zu entwickeln und seinen Code zu teilen. Die meisten Beispiele in diesem Buch sind in Jupyter-Notebooks geschrieben.

Führen Sie dazu den folgenden Befehl von Ihrem Terminal innerhalb Ihrer virtuellen Umgebung aus:

```
python -m ipykernel install --user --name generative ❶
```

❶ Dadurch erlangen Sie in Jupyter-Notebooks Zugriff auf die virtuelle Umgebung, die Sie gerade eingerichtet haben (`generative`).

Um zu überprüfen, ob die Installation einwandfrei verlief, navigieren Sie in Ihrem Terminal zu dem Ordner, in dem Sie das Buch-Repository geklont haben. Geben Sie ein:

```
jupyter notebook
```

In Ihrem Browser sollte sich ein Fenster öffnen, das einen Bildschirm ähnlich wie in Abbildung 1-15 zeigt. Klicken Sie auf das Notebook, das Sie ausführen möchten, und wählen Sie aus der Drop-down-Liste *Kernel → Change kernel* die virtuelle Umgebung `generative` aus.

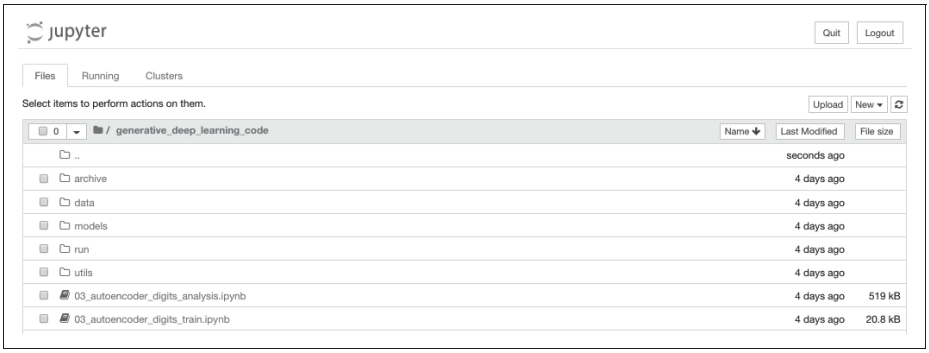

Abbildung 1-15: Jupyter-Notebook

Sie sind nun startklar für die Entwicklung generativer tiefer neuronaler Netze.

Zusammenfassung

Dieses Kapitel führte in das Gebiet der generativen Modellierung ein, ein wichtiges Teilgebiet des maschinellen Lernens, das die verbreitetere diskriminative Modellierung ergänzt. Unser erstes Basisbeispiel eines generativen Modells nutzte die Naive-Bayes-Annahme, um eine Wahrscheinlichkeitsverteilung zu erzeugen, die in der Lage war, die inhärente Struktur der Daten abzubilden und Beispiele außerhalb des Trainingsdatensatzes zu generieren. Wir haben auch gesehen, wie solche Basismodelle mit zunehmender Komplexität der generativen Aufgabe scheitern können, und haben die allgemeinen Herausforderungen der generativen Modellierung analysiert. Schließlich haben wir einen ersten Blick auf das Representation Learning geworfen – ein wichtiges Konzept, das den Kern vieler generativer Modelle bildet.

In Kapitel 2 werden wir mit der Einführung in das Deep Learning beginnen und herausfinden, wie man mit Keras Modelle erstellt, die diskriminative Modellierungsaufgaben bewältigen können. Dadurch werden wir uns die notwendigen Grundlagen aneignen, die wir in späteren Kapiteln für das generative Deep Learning benötigen.