O'REILLY®

Deep learning i modelowanie generatywne

Jak nauczyć komputer malowania, pisania, komponowania i grania

Helion

David Foster

Tytuł oryginału: Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-7283-2

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/deelmg
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Modelowanie generatywne

Ten rozdział zawiera ogólne wprowadzenie do dziedziny modelowania generatywnego. Najpierw opowiem, co to znaczy, że model jest *generatywny*, a potem pokażę, w jaki sposób różni się on od badanego dokładniej *modelowania dyskryminatywnego*. Następnie zaprezentuję framework oraz podstawowe zagadnienia matematyczne, które pozwolą ustrukturyzować nasze ogólne podejście do problemów wymagających rozwiązań generatywnych.

Po omówieniu tych zagadnień zbudujemy pierwszy przykład modelu generatywnego o charakterze probabilistycznym (naiwny model Bayesa). Przekonamy się, że pozwoli to nam generować nowe przykłady, wykraczające poza szkoleniowy zbiór danych, a ponadto pokaże powody, dla których ten typ modelu może się nie sprawdzić, gdy zwiększy się rozmiar i złożoność przestrzeni możliwych kreacji.

Czym jest modelowanie generatywne?

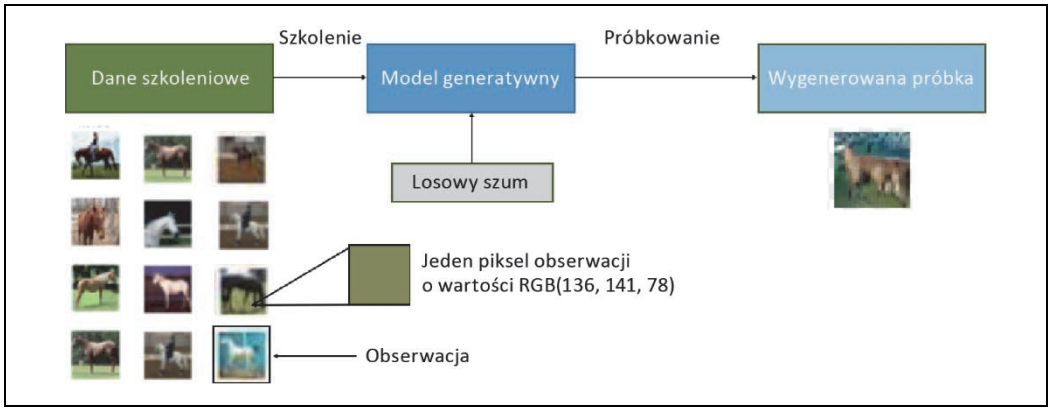
Modelowanie generatywne, ogólnie rzecz biorąc, można zdefiniować następująco:

> Model generatywny opisuje sposób generowania zestawu danych z punktu widzenia modelu probabilistycznego. Poprzez próbkowanie z tego modelu jesteśmy w stanie wygenerować nowe dane.

Załóżmy, że mamy zestaw danych zawierający zdjęcia koni. Możemy dążyć do zbudowania modelu zdolnego do wygenerowania nowego zdjęcia konia, który nigdy nie istniał, ale wciąż wygląda na realnego, ponieważ model nauczył się ogólnych zasad rządzących wyglądem konia. Jest to rodzaj problemu, który można rozwiązać za pomocą modelowania generatywnego. Podsumowanie typowego procesu modelowania generatywnego pokazałem na rysunku 1.1.

Przede wszystkim potrzebujemy zestawu danych składającego się z wielu próbek podmiotu, który staramy się wygenerować. Ten zestaw próbek to *dane szkoleniowe*, a jeden egzemplarz takiego punktu danych to *obserwacja*.

Każda obserwacja składa się z wielu *cech* (ang. *feature*) — w przypadku problemu generowania obrazu cechami są zwykle poszczególne wartości pikseli. Naszym celem jest zbudowanie modelu pozwalającego generować nowe zestawy cech, które wyglądają tak, jakby zostały stworzone przy użyciu tych samych reguł co w danych oryginalnych. Koncepcyjnie, w przypadku generowania obrazu, jest to niezwykle trudne zadanie, biorąc pod uwagę ogromną liczbę możliwości przypisania wartości poszczególnych pikseli oraz stosunkowo niewielką liczbę takich układów tworzących wizerunek podmiotu, który próbujemy symulować.



Rysunek 1.1. Proces modelowania generatywnego

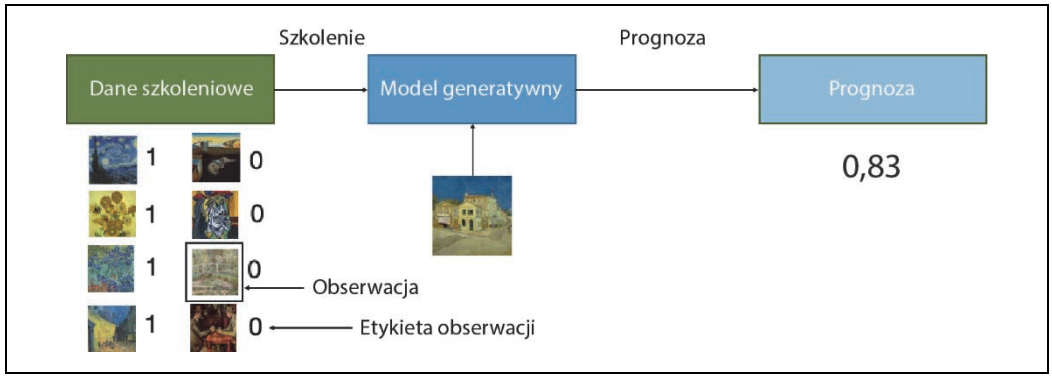
Model generatywny musi również być *probabilistyczny*, a nie *deterministyczny*. Jeżeli nasz model jest jedynie ustalonym obliczeniem — na przykład polega na wyznaczeniu wartości średniej każdego piksela w zbiorze danych — nie jest generatywny, ponieważ daje za każdym razem taki sam wynik. Model musi zawierać element *stochastyczny* (losowy), który ma wpływ na poszczególne generowane przez niego próbki.

Innymi słowy, możemy sobie wyobrazić, że istnieje jakiś nieznany rozkład probabilistyczny, który wyjaśnia, dlaczego niektóre obrazy mogą się znaleźć w zbiorze szkoleniowym, a inne nie. Naszym zadaniem jest zbudowanie modelu, który jak najbardziej naśladuje ten rozkład, a następnie pobieranie z niego próbek w celu wygenerowania nowych, odmiennych obserwacji, które wyglądają tak, jakby mogły być uwzględnione w pierwotnym zestawie treningowym.

Modelowanie generatywne a dyskryminatywne

Aby naprawdę zrozumieć, co próbuje osiągnąć modelowanie generatywne i dlaczego jest to ważne, warto porównać je do innego modelowania, mianowicie *modelowania dyskryminatywnego*. Większość problemów uczenia maszynowego omawianych w literaturze najprawdopodobniej ma charakter dyskryminatywny. Aby zrozumieć różnice pomiędzy poszczególnymi rodzajami modelowania, przyjrzyjmy się przykładowi.

Załóżmy, że mamy zestaw danych opisujących obrazy. Niektóre z nich zostały namalowane przez Van Gogha, a inne przez innych artystów. Mając wystarczająco dużo danych, możemy wytrenować model dyskryminatywny pozwalający odgadnąć, czy dany obraz został namalowany przez Van Gogha. Nasz model nauczyłby się, że pewne zestawienia kolorów, kształtów i tekstur dają wyższe prawdopodobieństwo namalowania przez holenderskiego mistrza. W przypadku obrazów wykazujących takie cechy model będzie odpowiednio podnosił wagę swoich prognoz. Proces modelowania dyskryminatywnego został przedstawiony na rysunku 1.2 — zwróćmy uwagę na różnice względem procesu modelowania generatywnego pokazanego na rysunku 1.1.



Rysunek 1.2. Proces modelowania dyskryminatywnego

Jedną z kluczowych różnic pomiędzy modelem generatywnym a dyskryminatywnym jest to, że podczas realizacji modelowania dyskryminatywnego każda obserwacja w danych szkoleniowych ma *etykietę*. W przypadku problemu klasyfikacji binarnej, takiego jak przykładowy dyskryminator artysty, obrazy Van Gogha będą oznaczone etykietą 1, natomiast obrazy innych artystów etykietą 0. Następnie nasz model będzie się uczył rozróżniać pomiędzy tymi dwiema grupami i wyznaczał prawdopodobieństwo, że nowa obserwacja ma etykietę 1 — to znaczy, że obraz został namalowany przez Van Gogha.

Z tego powodu modelowanie dyskryminatywne jest synonimem *uczenia nadzorowanego* lub uczenia się funkcji, która odwzorowuje wejście na wyjścia z użyciem znakowanego zestawu danych. Modelowanie generatywne jest zwykle wykonywane z wykorzystaniem nieoznakowanego zbioru danych (czyli jako forma uczenia nienadzorowanego), choć może być również stosowane do oznaczonego zbioru danych w celu zapoznania się ze sposobami generowania obserwacji z każdej odrębnej klasy.

Przyjrzyjmy się notacji matematycznej, która opisuje różnicę pomiędzy modelowaniem generatywnym a dyskryminatywnym.

> *Modelowanie dyskryminatywne* szacuje wartość $p(y|\boldsymbol{x})$ — prawdopodobieństwo, że danej obserwacji $\boldsymbol{x}$ zostanie przypisana etykieta y.

> *Modelowanie generatywne* szacuje $p(\boldsymbol{x})$ — prawdopodobieństwo zaobserwowania obserwacji $\boldsymbol{x}$.
>
> Jeśli zestaw danych jest oznaczony, możemy również zbudować model generatywny, który oszacowuje rozkład $p(\boldsymbol{x}|y)$.

Innymi słowy, modelowanie dyskryminatywne próbuje oszacować prawdopodobieństwo, że obserwacja $\boldsymbol{x}$ należy do kategorii y. W modelowaniu generatywnym nie jest wykonywane znakowanie obserwacji. Zamiast tego model generatywny próbuje oszacować prawdopodobieństwo tego, że obserwacja w ogóle się pojawi.

Kluczową kwestią jest to, że nawet gdybyśmy byli w stanie zbudować idealny model dyskryminatywny do identyfikacji obrazów Van Gogha, to nadal model ów nie miałby pojęcia, jak stworzyć obraz, który wygląda jak obraz namalowany przez Van Gogha. Potrafiłby on jedynie określać

prawdopodobieństwo dla istniejących obrazów, ponieważ do tego został przeszkolony. Zamiast tego trzeba wyszkolić model generatywny potrafiący wygenerować zbiór pikseli, które mają dużą szansę przynależności do oryginalnego zbioru szkoleniowego.

Postępy w uczeniu maszynowym

Aby zrozumieć, dlaczego modelowanie generatywne można uznać za kolejną granicę dla uczenia maszynowego, trzeba najpierw przyjrzeć się, dlaczego w ciągu ostatnich dwóch dekad modelowanie dyskryminatywne było siłą napędową postępu w większości metod uczenia maszynowego zarówno w środowisku akademickim, jak i w branży.

Z punktu widzenia środowiska akademickiego postępy w modelowaniu dyskryminatywnym z pewnością łatwiej monitorować. Aby ustalić aktualnie najlepszą w swojej klasie metodę, można zmierzyć wskaźniki wydajności względem typowych zadań klasyfikacyjnych. Modele generatywne są często trudniejsze do oceny, zwłaszcza wtedy, gdy jakość wyników jest w dużej mierze subiektywna. Z tego powodu w ostatnich latach duży nacisk kładzie się na szkolenie modeli dyskryminatywnych. Dąży się do tego, aby dla różnych zadań klasyfikacji tekstu lub obrazu osiągnąć wydajność dorównującą wydajności ludzi lub ją przewyższającą.

Przykładowo w przypadku klasyfikacji obrazów kluczowy przełom nastąpił w 2012 roku, kiedy to zespół kierowany przez Geoffa Hintona z Uniwersytetu w Toronto za projekt głębokiej konwolucyjnej sieci neuronowej wygrał konkurs ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Konkurs polega na klasyfikowaniu obrazów do jednej z tysiąca kategorii i służy jako punkt odniesienia do porównywania najnowszych technik. Model korzystający z uczenia głębokiego miał współczynnik błędu 16%. Był to wynik znacząco lepszy w porównaniu z modelem, który zajął kolejne miejsce — uzyskał on stopę błędów równą zaledwie 26,2%. Wywołało to szybki rozwój technik uczenia głębokiego. W efekcie w kolejnych latach odnotowano jeszcze większy spadek stopy błędów. Zwycięzca z roku 2015 po raz pierwszy uzyskał wydajność przewyższającą wydajność człowieka — stopę błędów na poziomie 4% — a obecnie najnowszy model charakteryzuje się wskaźnikiem błędu wynoszącym zaledwie 2%. W opinii wielu osób problem, który był przedmiotem konkursu, można uznać za rozwiązany.

Oprócz tego, że dla środowiska akademickiego publikowanie wymiernych efektów w przypadku modelowania dyskryminatywnego jest łatwiejsze, z historycznego punktu widzenia modelowanie dyskryminatywne jest łatwiejsze do zastosowania w odniesieniu do problemów biznesowych niż modelowanie generatywne. Ogólnie rzecz biorąc, w środowisku biznesowym nie przejmujemy się sposobem, w jaki wygenerowano dane. Interesuje nas natomiast sposób, w *jaki* powinniśmy skategoryzować nową próbkę lub oszacować jej wartość. Na przykład:

- Jeśli dowódca sił zbrojnych analizuje zdjęcie satelitarne, to interesuje go jedynie prawdopodobieństwo, że przedstawia ono wrogie jednostki, a nie prawdopodobieństwo, że takie zdjęcie w ogóle się pojawi.
- Menedżer ds. relacji z klientami będzie zainteresowany tylko tym, czy wydźwięk przychodzącej wiadomości e-mail jest pozytywny czy negatywny. Nie interesuje go natomiast model generatywny zdolny do stworzenia przykładów nieistniejących wiadomości e-mail od klientów.
- Lekarz będzie chciał znać prawdopodobieństwo, że dany obraz siatkówki oznacza jaskrę. Nie interesuje go natomiast model, który potrafi generować nowe zdjęcia dna oka.

Ponieważ większość rozwiązań pożądanych przez środowiska biznesowe należy do dziedziny modelowania dyskryminatywnego, powstały liczne narzędzia MLaaS (ang. *Machine Learning as a Service* — uczenie maszynowe jako usługa), które mają na celu dostarczanie aplikacji modelowania dyskryminatywnego w branży. W dużej mierze ich działanie sprowadza się do automatyzacji procesów budowania, weryfikowania i monitorowania, które są powszechne dla niemal wszystkich zadań modelowania dyskryminatywnego.

Powstanie modelowania generatywnego

Chociaż największe jak dotąd postępy w dziedzinie uczenia maszynowego zawdzięczamy modelowaniu dyskryminatywnemu, to w ciągu ostatnich trzech do pięciu lat wiele ciekawych osiągnięć w dziedzinie wynika z nowelizacji aplikacji uczenia głębokiego w kierunku zadań modelowania generatywnego.

W szczególności w ostatnim czasie można było zaobserwować większe zainteresowanie mediów projektami modelowania generatywnego. Opisywano między innymi projekt StyleGAN firmy NVIDIA[1], w którym udało się tworzyć hiperrealistyczne zdjęcia ludzkich twarzy, a model językowy GPT-2 opracowany przez OpenAI[2] potrafi dokończyć akapit tekstu na podstawie krótkiego akapitu wprowadzającego.

Wyraźny postęp, jaki osiągnięto w zakresie generowania obrazu twarzy od 2014 roku, został zaprezentowany na rysunku 1.3[3]. Istnieją wyraźne zastosowania tych osiągnięć w takich branżach jak projektowanie gier i kinematografia. Z pewnością można się również spodziewać usprawnień w zakresie automatycznego generowania muzyki. Co prawda nie ma pewności, czy któregoś dnia przeczytamy artykuły lub powieści napisane przez modele generatywne, ale ostatnie postępy w tej dziedzinie są oszałamiające i niewątpliwie sugerowanie, że pewnego dnia tak może się stać, nie jest niedorzecznością. Choć jest to ekscytujące, budzi również wątpliwości natury etycznej związane z rozpowszechnianiem fałszywych treści w internecie i oznacza, że jeszcze trudniejsze może być zaufanie temu, co widzimy i czytamy za pośrednictwem kanałów komunikacji publicznej.

Rysunek 1.3. Tworzenie wizerunku twarzy przy użyciu modelowania generatywnego uległo w ciągu ostatnich czterech lat znacznej poprawie jakości[4]

[1] Tero Karras, Samuli Laine i Timo Aila, *A Style-Based Generator Architecture for Generative Adversarial Networks*, 12 grudnia 2018, *https://arxiv.org/abs/1812.04948*.

[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*, luty 2018, *https://www.eff.org/files/2018/02/20/malicious_ai_report_final.pdf*.

[4] Źródło: Brundage et al., 2018.

Oprócz praktycznych zastosowań modelowania generatywnego (z których wiele jeszcze nie odkryto) istnieją trzy ważniejsze powody, dla których modelowanie generatywne można uznać za klucz do odblokowania znacznie bardziej wyrafinowanej formy sztucznej inteligencji, która wykracza poza to, co można osiągnąć wyłącznie za pomocą modelowania dyskryminatywnego.

Po pierwsze, z czysto teoretycznego punktu widzenia, nie powinniśmy zadowalać się samym dążeniem do mistrzostwa w kategoryzacji danych. Powinniśmy również dążyć do pełniejszego zrozumienia sposobów, w jakie te dane zostały wygenerowane. To niewątpliwie trudniejszy problem do rozwiązania. Powodem jest duża liczba wymiarów przestrzeni możliwych wyników i stosunkowo niewielka liczba kreacji, które można sklasyfikować jako należące do zestawu danych. Jednak, jak się przekonamy, wiele z tych samych technik, które napędzały rozwój modelowania dyskryminatywnego — na przykład uczenie głębokie — można również wykorzystać dla modeli generatywnych.

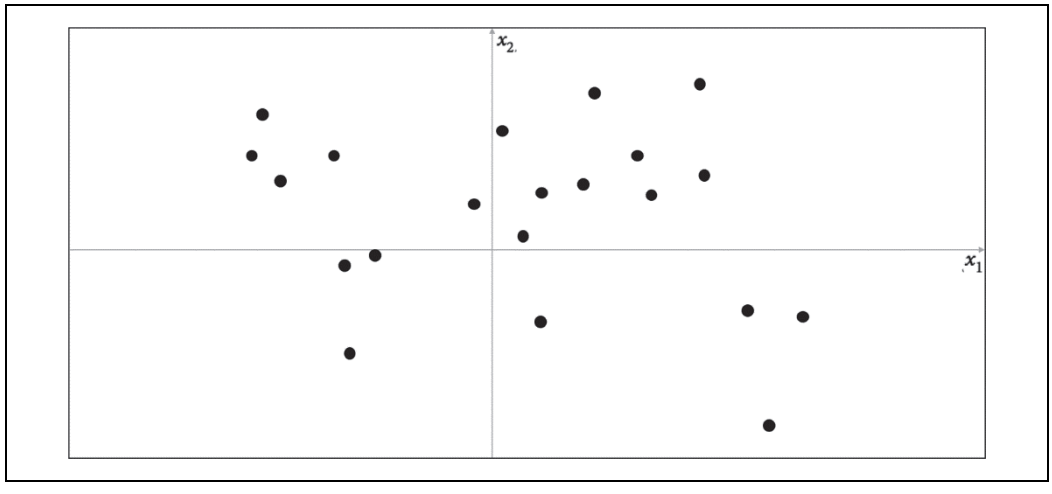
Po drugie, jest wysoce prawdopodobne, że modelowanie generatywne będzie mieć kluczowe znaczenie dla rozwoju przyszłych prac w innych dziedzinach uczenia maszynowego — takich jak na przykład w uczenie przez wzmacnianie (ang. *reinforcenment learning*; badanie agentów uczących zmierzające do optymalizacji celu w środowisku metodą prób i błędów). Na przykład moglibyśmy zastosować techniki uczenia przez wzmacnianie, aby wytrenować robota do spacerowania po określonym terenie. Ogólne podejście mogłoby polegać na zbudowaniu symulacji komputerowej terenu, a następnie przeprowadzeniu wielu eksperymentów, w których agent próbuje różnych strategii. Z biegiem czasu agent uczyłby się, które strategie są skuteczniejsze od innych, i w ten sposób osiągałby stopniowe postępy. Typowym problemem z tym podejściem jest to, że cechy fizyczne środowiska często są bardzo skomplikowane i musiałyby być obliczane w każdym kroku czasowym po to, by przekazać informacje z powrotem do agenta i zdecydować o jego następnym ruchu. Gdyby jednak agent był w stanie symulować swoje otoczenie za pośrednictwem modelu generatywnego, to nie trzeba by było testować strategii w symulacji komputerowej lub w realnym świecie, ale mógłby się jej nauczyć we własnym wyimaginowanym środowisku. Wykorzystanie tego pomysłu pokażę w rozdziale 8. Zajmiemy się w nim szkoleniem samochodu w taki sposób, by pokonywał trasę możliwie jak najszybciej, pozwalając mu uczyć się bezpośrednio z własnego, wyobrażonego środowiska.

Abyśmy mogli stwierdzić, że udało nam się zbudować maszynę, która zyskała formę inteligencji porównywalnej do tej, jaką ma człowiek, to z pewnością częścią rozwiązania powinno być modelowanie generatywne. Jednym z najlepszych przykładów modelu generatywnego w naturalnym świecie jesteś Ty, drogi Czytelniku tej książki. Poświęć chwilę, aby zastanowić się nad tym, jakim niezwykłym modelem generatywnym jesteś. Potrafisz zamknąć oczy i wyobrazić sobie, jak wygląda słoń pod każdym możliwym kątem. Potrafisz sobie wyobrazić szereg różnych prawdopodobnych zakończeń swojego ulubionego programu telewizyjnego oraz zaplanować nadchodzący tydzień w efekcie przemyślenia różnych wariantów przyszłości i podjęcia właściwych działań. Współczesna neurobiologia sugeruje, że nasza percepcja rzeczywistości nie jest wysoce skomplikowanym modelem dyskryminatywnym działającym na podstawie danych wejściowych odbieranych przez nasze zmysły w celu prognozowania tego, co przeżywamy, lecz raczej modelem generatywnym, który od urodzenia jest szkolony w kierunku generowania symulacji swojego otoczenia, które jak najdokładniej pasuje do przyszłości. Niektóre teorie sugerują nawet, że wyjście z tego modelu generatywnego jest tym, co bezpośrednio postrzegamy jako rzeczywistość. Oczywiście głębokie zrozumienie sposobu, w jaki możemy budować maszyny, by nabyły te umiejętności, będzie mieć kluczowe znaczenie dla naszego dalszego zrozumienia funkcjonowania mózgu i ogólnie sztucznej inteligencji.

Mając to na uwadze, zacznijmy naszą podróż do fascynującego świata modelowania generatywnego. Na początek przyjrzymy się najprostszym przykładom modeli generatywnych i niektórym pojęciom, które pomogą nam przeanalizować bardziej złożone architektury opisane w dalszej części tej książki.

Framework modelowania generatywnego

Zacznijmy od zagrania w grę modelowania generatywnego zaledwie w dwóch wymiarach. Na rysunku 1.4 wybrałem regułę wykorzystaną do wygenerowania zbioru punktów **X**. Nazwijmy tę regułę p_{data}. Twoim zadaniem jest wybranie w przestrzeni innego punktu $\boldsymbol{x} = (x_1, x_2)$, który wygląda tak, jakby został wygenerowany przez tę samą regułę.



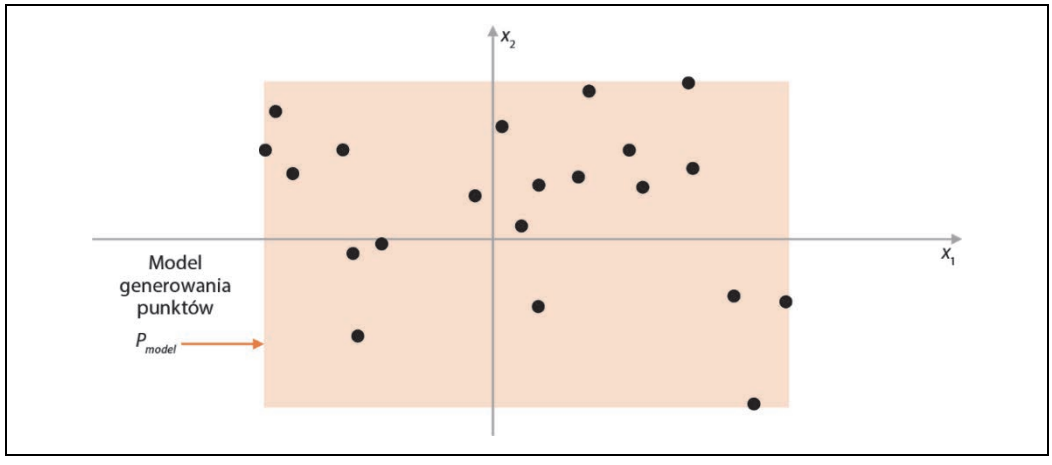
Rysunek 1.4. Zbiór punktów w dwóch wymiarach wygenerowany przez nieznaną regułę p_{data}

W jaki sposób postąpiłeś? Prawdopodobnie wykorzystałeś swoją wiedzę o istniejących punktach danych w celu skonstruowania mentalnego modelu p_{model} zawierającego współrzędne w przestrzeni, gdzie prawdopodobnie powinien znaleźć się punkt. W związku z tym p_{model} jest oszacowaniem p_{data}. Być może zdecydowałeś, że p_{model} powinien wyglądać tak jak na rysunku 1.5 — jest prostokątnym oknem, w którym mogą znajdować się punkty, oraz obszarem na zewnątrz okna, gdzie nie ma szans na znalezienie żadnych punktów. Aby wygenerować nową obserwację, można po prostu wybrać losowy punkt należący do okna lub — bardziej formalnie — próbkę z dystrybucji p_{model}. Gratulacje! Właśnie opracowałeś swój pierwszy model generatywny!

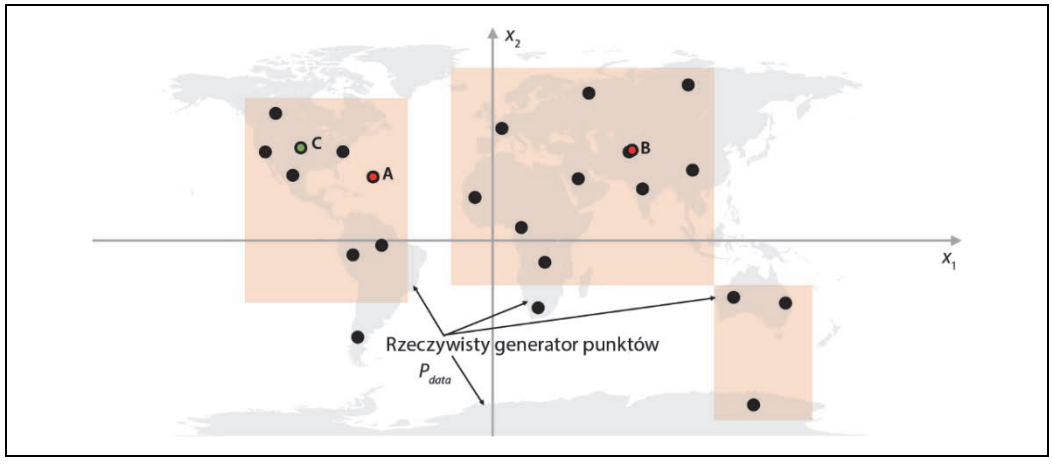
Chociaż nie jest to najbardziej złożony przykład, możemy go wykorzystać, aby zrozumieć cel, jaki stara się osiągnąć modelowanie generatywne. Opisany poniżej framework przedstawia nasze motywacje.

Spróbujmy teraz ujawnić prawdziwy rozkład generujący dane — p_{data} — i ocenić, w jaki sposób framework pasuje do tego przykładu.

Jak można zobaczyć na rysunku 1.6, reguła generowania danych jest po prostu równomiernym rozkładem punktów należących do lądowych części świata — żaden z punktów nie leży w morzu.



Rysunek 1.5. Różowy prostokąt, p_{model}, to oszacowanie rzeczywistego rozkładu generującego dane p_{data}



Rysunek 1.6. Różowe pole, p_{model}, to oszacowanie rzeczywistego rozkładu generującego dane p_{data}(szary obszar)

Framework modelowania generatywnego

- Mamy zestaw danych naszych obserwacji $\boldsymbol{X}$.
- Zakładamy, że obserwacje zostały wygenerowane według jakiegoś nieznanego rozkładu p_{data}.
- Model generatywny p_{model} próbuje naśladować rozkład p_{data}. Jeśli osiągniemy ten cel, będziemy mogli pobierać próbki z modelu p_{model}, aby wygenerować obserwacje wyglądające tak, jakby zostały pobrane z p_{data}.
- Będziemy zadowoleni z modelu p_{model}, jeżeli:
  - Reguła 1: będzie w stanie generować próbki, które wyglądają tak, jakby zostały pobrane z p_{data}.
  - Reguła 2: będzie w stanie generować próbki, które w pewien sposób różnią się od obserwacji zamieszczonych w $\boldsymbol{X}$. Innymi słowy, model nie powinien jedynie powielać tego, co zostało zaobserwowane.

Oczywiście nasz model p_{model} jest olbrzymim uproszczeniem rozkładu p_{data}. Punkty A, B i C przedstawiają trzy obserwacje wygenerowane przez p_{model} z różnym skutkiem:

- **Punkt A** łamie regułę nr 1 frameworka modelowania generatywnego — nie wygląda tak, jakby został wygenerowany przez p_{data}, ponieważ jest pośrodku morza.
- **Punkt B** jest tak blisko punktu należącego do rzeczywistego zestawu danych, że nie powinien nas zaskoczyć fakt, że nasz model potrafi wygenerować taki punkt. Gdyby wszystkie próbki wygenerowane przez model były takie jak punkt B, byłoby to naruszeniem reguły 2. frameworka modelowania generatywnego.
- **Punkt C** można uznać za sukces, ponieważ mógł on być wygenerowany przez p_{data} i jest odpowiednio różny od dowolnego innego punktu należącego do oryginalnego zestawu danych.

Dziedzina modelowania generatywnego jest zróżnicowana, a definicja problemu może przyjąć wiele różnych form. Jednak w większości scenariuszy framework modelowania generatywnego dobrze oddaje sposób, w jaki powinniśmy myśleć o rozwiązywaniu problemów.

Spróbujmy teraz zbudować nasz pierwszy nietrywialny przykład modelu generatywnego.

Probabilistyczne modele generatywne

Nie martw się, jeśli nigdy nie uczyłeś się rachunku prawdopodobieństwa. Do zbudowania i uruchomienia wielu modeli uczenia głębokiego, które zobaczymy w dalszej części tej książki, nie jest istotne gruntowne zrozumienie teorii statystycznej. Jednak aby uzyskać pełne uznanie historii zadania, które próbujemy rozwiązać, warto spróbować zbudować model generatywny, który nie bazuje na uczeniu głębokim, a zamiast tego opiera się wyłącznie na teorii prawdopodobieństwa. W ten sposób będziesz miał podstawy pozwalające zrozumieć wszystkie modele generatywne, niezależnie od tego, czy bazują one na uczeniu głębokim, czy nie, z tego samego, probabilistycznego punktu widzenia.

Jeśli już masz gruntowną wiedzę dotyczącą rachunku prawdopodobieństwa, to świetnie — duża część następnego podrozdziału może być dla Ciebie znana. W rozdziale zamieściłem jednak zabawny przykład. Uważaj, aby go nie przegapić!

Najpierw powinniśmy zdefiniować cztery podstawowe terminy: *przestrzeń próbek*, *funkcja gęstości*, *modelowanie parametryczne* i *szacowanie maksymalnej wiarygodności*.

Przestrzeń próbek

Przestrzeń próbek to kompletny zestaw wszystkich wartości, które może przyjmować obserwacja $\boldsymbol{x}$.

W naszym poprzednim przykładzie przestrzeń próbek składa się z wszystkich punktów o szerokości i długości geograficznej x = (x_1, x_2) na mapie świata.

Przykładowym punktem w przestrzeni próbek jest x = (40,7306, –73,9352) — Nowy Jork.

Funkcja gęstości prawdopodobieństwa

Funkcja gęstości prawdopodobieństwa (lub po prostu *funkcja gęstości*), $p(\boldsymbol{x})$, to funkcja, która odwzorowuje punkt x należący do przestrzeni próbek na liczbę z przedziału od 0 do 1. Suma[5] funkcji gęstości dla wszystkich punktów w przestrzeni próbek musi być równa 1, zatem funkcja gęstości jest ściśle zdefiniowanym rozkładem prawdopodobieństwa[6].

W przykładzie mapy świata funkcja gęstości naszego modelu ma wartość 0 dla punktów poza różowym polem oraz ma stałą wartość wewnątrz tego pola.

Chociaż istnieje tylko jedna rzeczywista funkcja gęstości p_{data}, która mogła wygenerować obserwowalny zestaw danych, istnieje nieskończenie wiele funkcji gęstości p_{model}, które możemy wykorzystać do oszacowania funkcji p_{data}. Aby uporządkować nasze podejście w celu znalezienia odpowiedniej funkcji $p_{model}(\boldsymbol{X})$, możemy użyć techniki znanej jako *modelowanie parametryczne*.

Modelowanie parametryczne

Model parametryczny, $p_\theta(x)$, to rodzina funkcji gęstości, którą można opisać za pomocą skończonej liczby parametrów, θ.

Przykładem modelu parametrycznego jest rodzina wszystkich możliwych pól, jakie można narysować na rysunku 1.5. W tym przypadku mamy cztery parametry: współrzędne lewego dolnego (θ_1, θ_2) i prawego górnego (θ_3, θ_4) rogu pola.

Tak więc każda funkcja gęstości $p_\theta(x)$ w tym modelu parametrycznym (tzn. każde pole) może być w unikatowy sposób opisana za pomocą czterech liczb: $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$.

Wiarygodność

Wiarygodność $\mathcal{L}(\theta|\boldsymbol{x})$ zbioru parametrów θ jest funkcją, która mierzy wiarygodność θ na podstawie pewnego zaobserwowanego punktu $\boldsymbol{x}$.

Można ją zdefiniować w następujący sposób:

$$\mathcal{L}(\theta|X) = \mathrm{p}\theta(\mathrm{x})$$

Oznacza to, że wiarygodność θ pewnego zaobserwowanego punktu można zdefiniować jako wartość funkcji gęstości parametryzowanej przez θ w punkcie $\boldsymbol{x}$.

Jeśli mamy cały zestaw danych $\boldsymbol{X}$ niezależnych obserwacji, to możemy napisać:

$$\mathcal{L}(\theta|X) = \prod_{x \epsilon X} p_\theta(x)$$

[5] Lub całka, jeżeli przestrzeń próbek jest ciągła.

[6] Jeśli przestrzeń próbek jest dyskretna, p(x) jest po prostu prawdopodobieństwem zaobserwowania punktu x.

Ponieważ wyznaczenie tego iloczynu może być dość trudne obliczeniowo, często zamiast niego używamy wiarygodności logarytmicznej ℓ:

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

Istnieją statystyczne powody, dla których wiarygodność jest zdefiniowana w ten sposób, ale dla nas taka definicja jest wystarczająca do tego, aby zrozumieć, dlaczego, intuicyjnie, ma to sens. Definiujemy wiarygodność zbioru parametrów θ jako równą prawdopodobieństwu zaobserwowania danych w modelu sparametryzowanym przez θ.

W przykładzie mapy świata różowe pole, które pokrywałoby tylko lewą połowę mapy, miałoby wiarygodność 0 — nie mogłaby ona wygenerować zestawu danych złożonego z punktów, które obserwowaliśmy w prawej połowie mapy. Różowe pole z rysunku 1.5 ma dodatnią wiarygodność, ponieważ funkcja gęstości w tym modelu jest dodatnia dla wszystkich punktów danych.

Dlatego intuicyjnie ma sens, że celem modelowania parametrycznego powinno być znalezienie dla zestawu parametrów optymalnej wartości $\hat{\theta}$, która maksymalizuje wiarygodność zaobserwowania zestawu danych X. Ta technika jest dość sensownie nazywana *szacowaniem maksymalnej wiarygodności*.

Szacowanie maksymalnej wiarygodności

Szacowanie maksymalnej wiarygodności jest techniką, która pozwala oszacować $\hat{\theta}$ — zestaw parametrów θ w funkcji gęstości, $p_\theta(x)$, które w najbardziej prawdopodobny sposób pozwalają opisać pewne obserwowane dane $\boldsymbol{X}$.

Bardziej formalnie:

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

Wartość $\hat{\theta}$ jest nazywana również *oszacowaniem maksymalnej wiarygodności* (ang. *maximum likelihood estimate* — MLE).

Mamy teraz całą niezbędną terminologię, aby zacząć opisywać sposób, w jaki można zbudować probabilistyczny model generatywny.

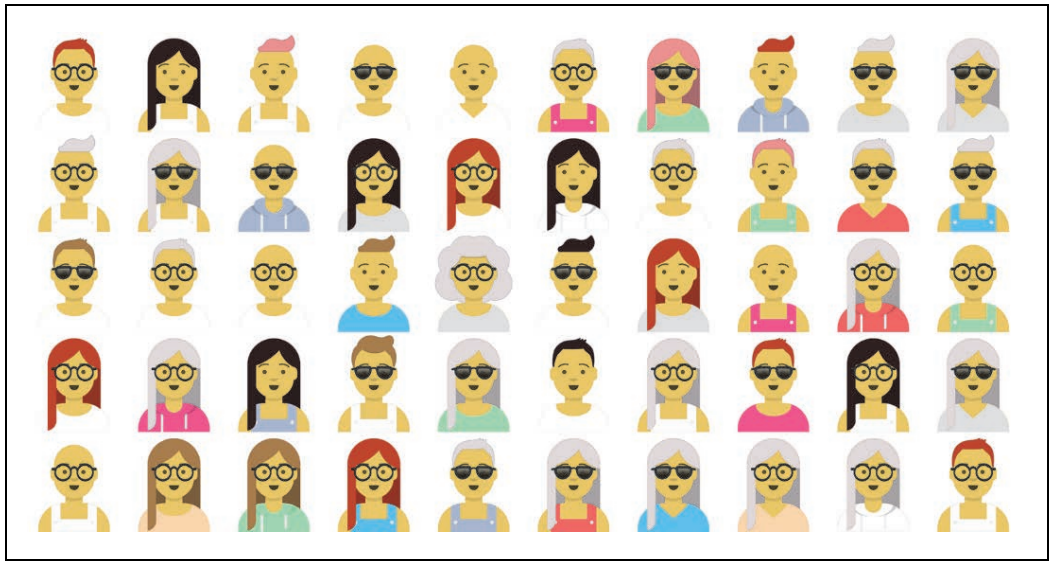
Większość rozdziałów w tej książce będzie zawierać krótkie opowiadanie, które pomaga opisać konkretną technikę. W tym rozdziale wyruszymy w podróż na planetę Zlemia, gdzie czeka na nas pierwsze zadanie związane z modelowaniem generatywnym…

Witaj, Zlemio!

Jest rok 2047. Ku Twojej radości zostałeś powołany na stanowisko Szefa Mody Planety Zlemia. Twoim jedynym obowiązkiem jest tworzenie nowych i ekscytujących trendów w modzie dla mieszkańców planety.

Zlemianie są dość konserwatywni, jeśli chodzi o modę, więc Twoim zadaniem jest generowanie nowych stylów, które są podobne do tych, które już istnieją na planecie, choć nie są identyczne.

W dniu Twojego przybycia zaprezentowano Ci zestaw danych zawierający 50 obserwacji mody Zlemian (rysunek 1.7) i zlecono zadanie znalezienia w ciągu dnia 10 nowych stylów, które masz przedstawić Zlemiańskiej Komisji Mody. W procesie tworzenia swoich dzieł możesz eksperymentować z fryzurami, kolorem włosów, okularami, rodzajem odzieży i jej kolorem.



Rysunek 1.7. Próbki wizerunków 50 Zlemian[7]

Ponieważ czujesz się inżynierem danych, w celu rozwiązania problemu postanowiłeś zastosować model generatywny. Po krótkiej wizycie w Intergalaktycznej Bibliotece wypożyczyłeś książkę pod tytułem *Generatywne uczenie głębokie* i zacząłeś czytać…

Ciąg dalszy nastąpi…

Twój pierwszy probabilistyczny model generatywny

Spróbujmy przyjrzeć się bliżej zestawowi danych Zlemian. Składa się on z N = 50 obserwacji stylów mody obecnie obserwowanych na planecie. Każdą obserwację można opisać za pomocą pięciu cech (*typDodatków*, *kolorOdzieży*, *rodzajOdzieży*, *kolorWłosów*, *rodzajFryzury*), tak jak pokazano w tabeli 1.1.

[7] Źródło obrazów: *https://getavataaars.com*.

Tabela 1.1. Pierwsze 10 obserwacji w zestawie danych twarzy Zlemian

id_twarzy	typDodatków	kolorOdzieży	typOdzieży	kolorWłosów	typFryzury
0	Okrągłe	Biały	KoszulkaZOkrągłym ↪Dekoltem	Rudy	WłosyKrótkieProste
1	Okrągłe	Biały	Ogrodniczki	Platynowy	WłosyKrótkieNaJeża
2	Przeciwsłoneczne	Biały	KoszulkaZOkrągłym ↪Dekoltem	Blond	WłosyKrótkieProste
3	Okrągłe	Biały	KoszulkaZOkrągłym ↪Dekoltem	Rudy	WłosyDługieProste
4	Okrągłe	Biały	Ogrodniczki	Platynowy	BrakWłosów
5	Brak	Biały	Ogrodniczki	Czarny	WłosyDługieProste
6	Przeciwsłoneczne	Biały	Ogrodniczki	Platynowy	WłosyDługieProste
7	Okrągłe	Biały	KoszulkaZOkrągłym ↪Dekoltem	Platynowy	WłosyKrótkieProste
8	Okrągłe	Różowy	BluzaZKapturem	Platynowy	WłosyDługieProste
9	Okrągłe	Pastelowy Pomarańcz	KoszulkaZOkrągłym ↪Dekoltem	Blond	WłosyDługieProste

Oto możliwe wartości dla każdej cechy:

- 7 różnych fryzur (`typFryzury`):
  - `BrakWłosów`, `WłosyDługieKok`, `WłosyDługieKręcone`, `WłosyDługieProste`, `WłosyKrótkieFalowane`, `WłosyKrótkieProste`, `WłosyKrótkieNaJeża`
- 6 różnych kolorów włosów (`kolorWłosów`):
  - `Czarne`, `Blond`, `Brązowe`, `PastelowoRóżowe`, `Rude`, `Platynowe`
- 3 różne rodzaje okularów (`typDodatków`):
  - `Czarne`, `Okrągłe`, `Przeciwsłoneczne`
- 4 różne rodzaje odzieży (`typOdzieży`):
  - `BluzaZKapturem`, `Ogrodniczki`, `KoszulkaZOkrągłymDekoltem`, `SweterekWSerek`
- 8 różnych kolorów odzieży (`kolorOdzieży`):
  - `Czarny`, `Niebieski01`, `Szary01`, `PastelowaZieleń`, `PastelowyPomarańcz`, `Różowy`, `Czerwony`, `Biały`

Istnieją 7×6×3×4×8 = 4032 kombinacje tych cech, więc w przestrzeni próbek istnieją 4032 punkty.

Możemy sobie wyobrazić, że nasz zestaw danych został wygenerowany przez pewien rozkład p_{data}, który faworyzuje niektóre wartości cech. Na przykład na ilustracji przedstawionej na rysunku 1.7 możemy zobaczyć, że popularnym wyborem wydają się białe ubrania, podobnie jak platynowe włosy i koszulki z okrągłym dekoltem.

Problem polega na tym, że nie znamy jawnie rozkładu p_{data} — jesteśmy zmuszeni pracować z próbką obserwacji $\boldsymbol{X}$ wygenerowaną przez p_{data}. Celem modelowania generatywnego jest wykorzystanie tych obserwacji po to, by zbudować model p_{model}, który dokładnie naśladuje obserwacje generowane przez p_{data}.

Aby to osiągnąć, możemy po prostu, na podstawie danych, które zaobserwowaliśmy, przypisać prawdopodobieństwo do każdej kombinacji cech. Tak więc ten model parametryczny miałby d = 4031 parametrów — po jednym dla każdego punktu w przestrzeni próbek minus jeden, ponieważ wartość ostatniego parametru będzie wymuszona tak, aby suma prawdopodobieństw wynosiła 1. A zatem parametry modelu, które staramy się oszacować, to $(\theta_1, \ldots, \theta_{4031})$.

Ta konkretna klasa modelu parametrycznego jest znana jako *rozkład wielomianowy*, a oszacowanie maksymalnej wiarygodności to:

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

gdzie n_j oznacza liczbę razy, kiedy w zestawie danych zaobserwowano kombinację j, a $\boldsymbol{N}$ = 50 to całkowita liczba obserwacji.

Innymi słowy, oszacowanie dla każdego parametru to po prostu proporcja liczby obserwacji kombinacji w zestawie danych względem wszystkich obserwacji.

Na przykład poniższa kombinacja (nazwijmy ją kombinacją 1) pojawia się w zbiorze danych dwukrotnie:

- `(WłosyDługieProste, Rudy, Okrągłe, KoszulkaZOkrągłymDekoltem, Biały)`

Dlatego:

$$\hat{\theta}_1 = 2/50 = 0.04$$

Kolejny przykład (nazwijmy gokombinacją 2) nie pojawia się w zbiorze danych wcale:

- `(WłosyDługieProste, Rudy, Okrągłe, KoszulkaZOkrągłymDekoltem, Niebieski01)`

Dlatego:

$$\hat{\theta}_2 = 0/50 = 0$$

W ten sposób możemy obliczyć wszystkie wartości $\hat{\theta}$j, aby określić rozkład przestrzeni próbek. Ponieważ możemy tworzyć próbki z tego rozkładu, lista potencjalnie mogłaby być nazwana modelem generatywnym. Jednak z jednego ważnego powodu nie jest to model generatywny: mianowicie nigdy nie pozwala wygenerować niczego, czego wcześniej nie zaobserwowano — dla dowolnej kombinacji, której nie było w oryginalnym zestawie danych $\boldsymbol{X}$, zachodzi $\hat{\theta}$j = 0.

Aby rozwiązać ten problem, możemy przypisać dodatkową wartość *pseudocount* równą 1 do każdej możliwej kombinacji cech. Technika ta jest znana jako *wygładzanie addytywne*. W tym modelu wartość MLE dla parametrów można określić następująco:

$$\hat{\theta}_j = \frac{n_j + 1}{N + d}$$

Teraz każda kombinacja charakteryzuje się niezerowym prawdopodobieństwem pojawienia się w próbce z modelu — dotyczy to również tych próbek, których nie było w zbiorze oryginalnym. Jednak ten model nadal nie jest zadowalającym modelem generatywnym, ponieważ prawdopodobieństwo zaobserwowania punktu, którego nie było w oryginalnym zbiorze danych, jest stałe. Gdybyśmy próbowali zastosować taki model do generowania obrazów Picassa, model przypisywałby taką samą wagę losowej kolekcji kolorowych pikseli jak replice obrazu Picassa, która tylko trochę różni się od rzeczywistego dzieła.

W idealnej sytuacji chcielibyśmy, aby model generatywny przypisywał wyższą wagę tym obszarom przestrzeni próbek, które uzna za bardziej prawdopodobne na podstawie pewnej struktury wywnioskowanej z danych, a nie żeby przypisywał wagi probabilistyczne jedynie tym punktom, które występują w zbiorze danych.

Aby osiągnąć ten cel, trzeba zastosować inny model parametryczny.

Naiwny model Bayesa

Naiwny model parametryczny Bayesa wykorzystuje proste założenie, które znacznie zmniejsza liczbę parametrów do oszacowania.

Przyjmujemy naiwne założenie, że każda cecha x_j jest niezależna od każdej innej cechy x_k[8]. Odnosząc to do zbioru danych Zlemian, zakładamy, że na przykład wybór koloru włosów nie ma wpływu na wybór typu odzieży, a typ okularów, który ktoś nosi, nie ma wpływu na fryzurę. Bardziej formalnie dla wszystkich cech x_j, x_k zachodzi:

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

Warunek ten jest znany jako *naiwne założenie Bayesa*. Aby zastosować to założenie, najpierw należy skorzystać z reguły łańcuchowej prawdopodobieństwa i zapisać funkcję gęstości jako iloczyn prawdopodobieństw warunkowych:

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

gdzie K oznacza całkowitą liczbę cech (5 w przypadku zestawu danych Zlemian).

Aby uprościć ostatni wiersz, możemy teraz zastosować naiwne założenie Bayesa:

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

[8] Dla danej zmiennej odpowiedzi y naiwne założenie Bayesa mówi, że istnieje *warunkowa* niezależność pomiędzy każdą parą cech x_j, x_k dla danego y.

To jest naiwny model Bayesa. Problem sprowadza się do oszacowania parametrów $\theta_{kl} = p(x_k = l)$ dla każdej cechy osobno i pomnożenia ich przez siebie w celu wyznaczenia prawdopodobieństwa dla każdej możliwej kombinacji.

Ile parametrów trzeba teraz oszacować? Dla każdej cechy trzeba oszacować parametr dla każdej wartości, którą cecha może przyjąć. Dlatego w przypadku zestawu danych Zlemian ten model jest definiowany jedynie przez 7 + 6 + 3 + 4 + 8 – 5 = 23 parametry[9].

Oszacowania maksymalnej wiarygodności $\widehat{\theta_{kl}}$ są następujące:

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

gdzie $\widehat{\theta_{kl}}$ oznacza liczbę razy, jaką dla danego zestawu danych cecha k przyjmuje wartość l, natomiast N=50 oznacza łączną liczbę obserwacji.

Wyliczone parametry dla zestawu danych Zlemian zostały pokazane w tabeli 1.2.

Tabela 1.2. Wartości MLE dla parametrów w naiwnym modelu Bayesa

typFryzury	n	$\widehat{\theta}$
BrakWłosów	7	0,14
WłosyDługieKok	0	0,00
WłosyDługieKręcone	1	0,02
WłosyDługieProste	23	0,46
WłosyKrótkieFalowane	1	0,02
WłosyKrótkieProste	11	0,22
WłosyKrótkieNaJeża	7	0,14
Razem	50	1,00

kolorWłosów	n	$\widehat{\theta}$
Czarny	7	0,14
Blond	6	0,12
Brązowy	2	0,04
Pastelowo-Różowy	3	0,06
Rudy	8	0,16
Platynowy	24	0,48
Razem	50	1,00

kolorOdzieży	n	$\widehat{\theta}$
Czarny	0	0,00
Niebieski01	4	0,08
Siwy01	10	0,20
PastelowaZieleń	5	0,10
PastelowyPomarańcz	2	0,04
Różowy	4	0,08
Czerwony	3	0,06
Biały	22	0,44
Razem	50	1,00

typDodatków	n	θ
Brak	11	0,22
Okrągłe	22	0,44
Przeciwsłoneczne	17	0,34
Razem	50	1,00

typOdzieży	n	θ
BluzaZKapturem	7	0,14
Ogrodniczki	18	0,36
KoszulkaZOkrągłymDekoltem	19	0,38
SweterekWSerek	6	0,12
Razem	50	1,00

Aby obliczyć prawdopodobieństwo generowania przez model pewnej obserwacji x, po prostu mnożymy przez siebie prawdopodobieństwa poszczególnych cech. Na przykład:

```
p(WłosyDługieProste, Rudy, Okrągłe, KoszulkaZOkrągłymDekoltem, Biały)
= p(WłosyDługieProste) × p(Rudy) × p(Okrągłe) × p(KoszulkaZOkrągłymDekoltem) × p(Biały)
= 0.46 × 0.16 × 0.44 × 0.38 × 0.44
= 0.0054
```

[9] –5 wynika z faktu, że ostatni parametr dla każdej funkcji powinien zapewnić, aby suma parametrów dla tej cechy wynosiła 1.

Zauważ, że ta kombinacja nie pojawia się w oryginalnym zestawie danych, ale nasz model nadal przydziela jej niezerowe prawdopodobieństwo, więc wciąż może zostać wygenerowana. Ponadto istnieje większe prawdopodobieństwo jej wygenerowania niż powiedzmy kombinacji (`WłosyDługieProste`, `Rudy`, `Okrągłe`, `SweterekWSerek`, `Biały`), ponieważ białe ubranie pojawia się w zbiorze danych częściej niż niebieskie.

Dlatego naiwny model Bayesa jest w stanie nauczyć się pewnej struktury danych i wykorzystać ją do wygenerowania nowych próbek, które nie występowały w wyjściowym zbiorze danych. Model oszacował prawdopodobieństwo zaobserwowania wartości każdej cechy niezależnie, zatem zgodnie z założeniem naiwnego modelu Bayesa możemy pomnożyć te prawdopodobieństwa w celu zbudowania kompletnej funkcji gęstości $p_\theta(x)$.

Dziesięć obserwacji pochodzących z modelu przedstawiono na rysunku 1.8.

Rysunek 1.8. Dziesięć nowych stylów Zlemian wygenerowanych za pomocą naiwnego modelu Bayesa

W przypadku tego prostego problemu założenie naiwnego modelu Bayesa, że każda cecha jest niezależna od wszystkich pozostałych cech, jest uzasadnione i dlatego generuje dobry model generatywny.

Teraz przyjrzyjmy się, co się stanie, gdy to założenie się nie sprawdzi.

Witaj, Zlemio! Kontynuacja

Gdy patrzysz na 10 nowych tworów wygenerowanych przez naiwny model Bayesa, czujesz dumę. Ciesząc się sukcesem, zwróciłeś uwagę na dylemat mody na innej planecie, ale tym razem problem nie jest taki prosty.

Na adekwatnie nazwanej planecie Piksel zestaw danych, który otrzymałeś, nie składa się z pięciu ogólnych cech, które widzieliśmy na Zlemi (`kolorWłosów`, `typDodatków` itd.), lecz zawiera jedynie wartości 32×32 piksele, które tworzą każde zdjęcie. Każda obserwacja zatem ma teraz 32×32 = 1024 cechy, a każda cecha może mieć dowolną z 256 wartości (pojedyncze kolory w palecie).

Obrazy z nowego zbioru danych zostały pokazane na rysunku 1.9, a próbki wartości pikseli dla pierwszych 10 obserwacji zostały zestawione w tabeli 1.3.

Rysunek 1.9. Moda na planecie Piksel

Tabela 1.3. Wartości pikseli 458 – 467 dla pierwszych 10 obserwacji na planecie Piksel

id_twarzy	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

Zdecydowałeś, że chcesz wypróbować sprawdzony naiwny model Bayesa raz jeszcze — tym razem wytrenowany na zestawie danych pikseli. Model oszacuje parametry maksymalnej wiarygodności, które zarządzają rozkładem kolorów poszczególnych pikseli. Dzięki temu będziesz w stanie generować próbki z tego rozkładu i w ten sposób tworzyć nowe obserwacje. Gdy jednak spróbowałeś to zrobić, stało się jasne, że coś poszło bardzo źle.

Zamiast nowych stylów mody model wygenerował 10 bardzo podobnych zdjęć, które nie mają wyraźnie rozróżnialnych dodatków, włosów ani koloru odzieży (rysunek 1.10). Dlaczego tak się stało?

Rysunek 1.10. Dziesięć nowych stylów na planecie Piksel wygenerowanych przez naiwny model Bayesa

Wyzwania modelowania generatywnego

Po pierwsze, ze względu na to, że naiwny model Bayesa próbkuje piksele niezależnie, nie ma możliwości, by się dowiedzieć, że dwa sąsiadujące ze sobą piksele prawdopodobnie mają bardzo podobny odcień, ponieważ na przykład są częścią tego samego elementu odzieży. Model potrafi generować piksele w kolorze twarzy i ust, gdyż wszystkie te piksele w zbiorze szkoleniowym mają w każdej obserwacji w przybliżeniu ten sam odcień. Z kolei w przypadku T-shirtów każdy piksel jest próbkowany losowo z różnych kolorów w zbiorze treningowym, bez względu na kolory, które były próbkowane z sąsiednich pikseli. Ponadto brak jest mechanizmu, który powodowałby formowanie z pikseli wokół oczu okrągłych kształtów okularów lub na przykład interpretowałby wzorce fal reprezentujących konkretny typ fryzury na podstawie czerwonych pikseli w górnej części obrazu.

Po drugie, w przestrzeni próbek istnieje obecnie ogromna liczba możliwych obserwacji. Tylko niewielka część z nich to rozpoznawalne twarze, a jeszcze mniejszy podzbiór stanowią twarze, które są zgodne z regułami mody na planecie Piksel. Dlatego jeśli naiwny model Bayesa pracuje bezpośrednio z wysoce skorelowanymi wartościami pikseli, szanse znalezienia satysfakcjonujących kombinacji wartości są bardzo małe.

Podsumowując, na planecie Zlemia poszczególne cechy były niezależne, a przestrzeń próbek stosunkowo niewielka, więc naiwny model Bayesa działał dobrze. Z kolei na planecie Piksel założenie, że każda wartość piksela jest niezależna od każdej innej wartości piksela, jest wyraźnie niesłuszne. Wartości pikseli są ze sobą silnie skorelowane, a przestrzeń próbek jest ogromna, więc znalezienie prawidłowej twarzy przez niezależne próbkowanie pikseli jest prawie niemożliwe. To wyjaśnia, dlaczego nie można oczekiwać, aby naiwny model Bayesa dobrze działał na surowych danych obrazów.

Ten przykład pokazuje dwa główne wyzwania, które musi pokonać model generatywny, aby osiągnąć sukces.

Wyzwania modelu generatywnego

- Model musi sobie radzić z wysokim stopniem warunkowej zależności pomiędzy cechami.
- Model musi potrafić znaleźć niewielką część satysfakcjonujących, być może wygenerowanych obserwacji w dużej, wielowymiarowej przestrzeni próbek.

Kluczem do pokonania obu tych wyzwań jest uczenie głębokie.

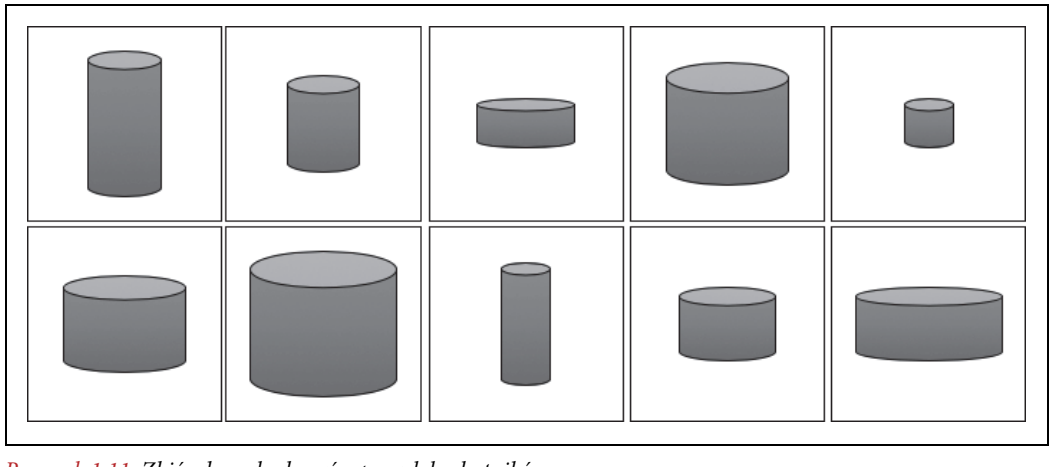
Potrzebujemy modelu, który potrafi wywnioskować odpowiednią strukturę danych. Nie chcemy nakazywać mu, by z góry przyjmował jakieś założenie. To jest dokładnie ten obszar, w którym idealnie sprawdza się uczenie głębokie. Jest to także jeden z głównych powodów, dla których technika ta stała się motorem najnowszych postępów w modelowaniu generatywnym.

Fakt, że uczenie głębokie potrafi tworzyć swoje własne cechy w niskowymiarowej przestrzeni, oznacza, iż jest ono formą *uczenia reprezentacji*. Zanim zajmiemy się w następnym rozdziale uczeniem głębokim, ważne jest przyswojenie sobie najważniejszych pojęć dotyczących uczenia reprezentacji.

Uczenie reprezentacji

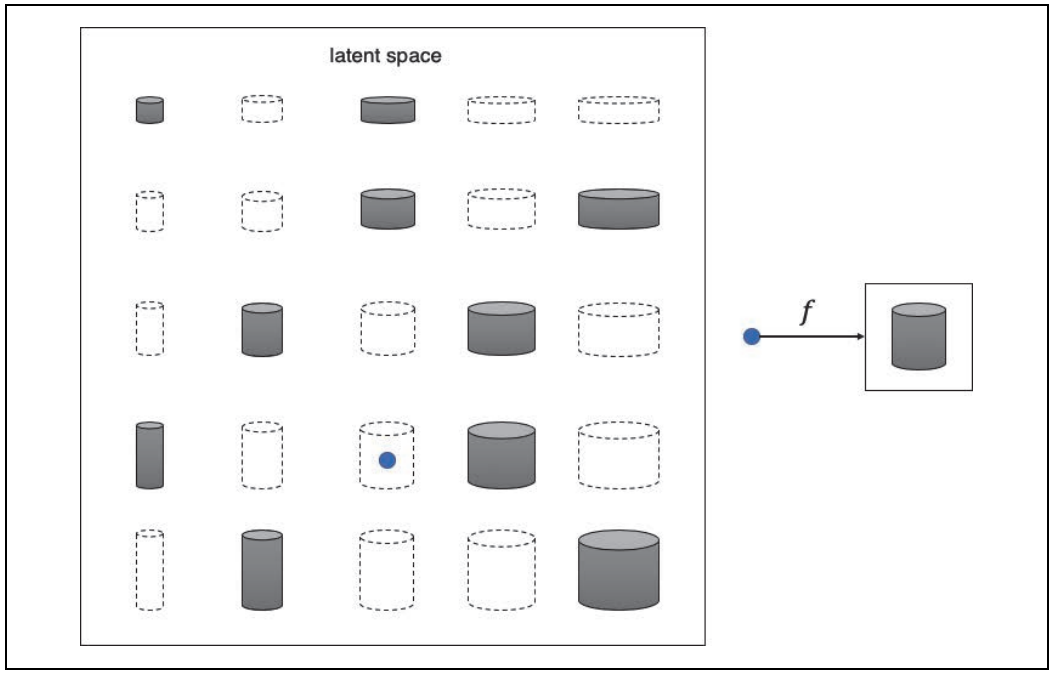
Główną koncepcją uczenia reprezentacji jest rezygnacja z prób bezpośredniego modelowania wielowymiarowej przestrzeni próbek. Zamiast tego dąży się do opisania każdej obserwacji w zbiorze treningowym przy użyciu pewnej niskowymiarowej *przestrzeni ukrytej* (ang. *latent space*), a następnie wywnioskowania funkcji mapowania, która potrafi przekształcić punkt w przestrzeni ukrytej na punkt we właściwej domenie. Innymi słowy, każdy punkt w przestrzeni ukrytej jest reprezentacją pewnego wielowymiarowego obrazu.

Co to oznacza w praktyce? Załóżmy, że mamy szkoleniowy zestaw danych składający się z obrazów puszek herbatników w skali szarości (rysunek 1.11).



Rysunek 1.11. Zbiór danych obrazów puszek herbatników

Dla nas jest oczywiste, że istnieją dwie cechy, które pozwalają jednoznacznie określić każdą z tych puszek: wysokość i szerokość. Mając dane wysokość i szerokość, możemy narysować prawidłową puszkę nawet wtedy, gdy nie było obrazu w zbiorze treningowym. Dla maszyny jednak nie jest to równie łatwe — najpierw musi ona ustalić, że wysokość i szerokość są dwoma wymiarami przestrzeni ukrytej, które najlepiej opisują ten zestaw danych, a następnie wywnioskować funkcję mapującą *f*, która potrafi pobrać punkt z tej przestrzeni ukrytej i zmapować go na obraz puszki herbatników w skali szarości. Wynikową przestrzeń ukrytą puszek herbatników i proces generowania przedstawia rysunek 1.12.



Rysunek 1.12. Ukryta przestrzeń puszek herbatników i funkcja f, która mapuje punkt w przestrzeni ukrytej na oryginalną domenę obrazu

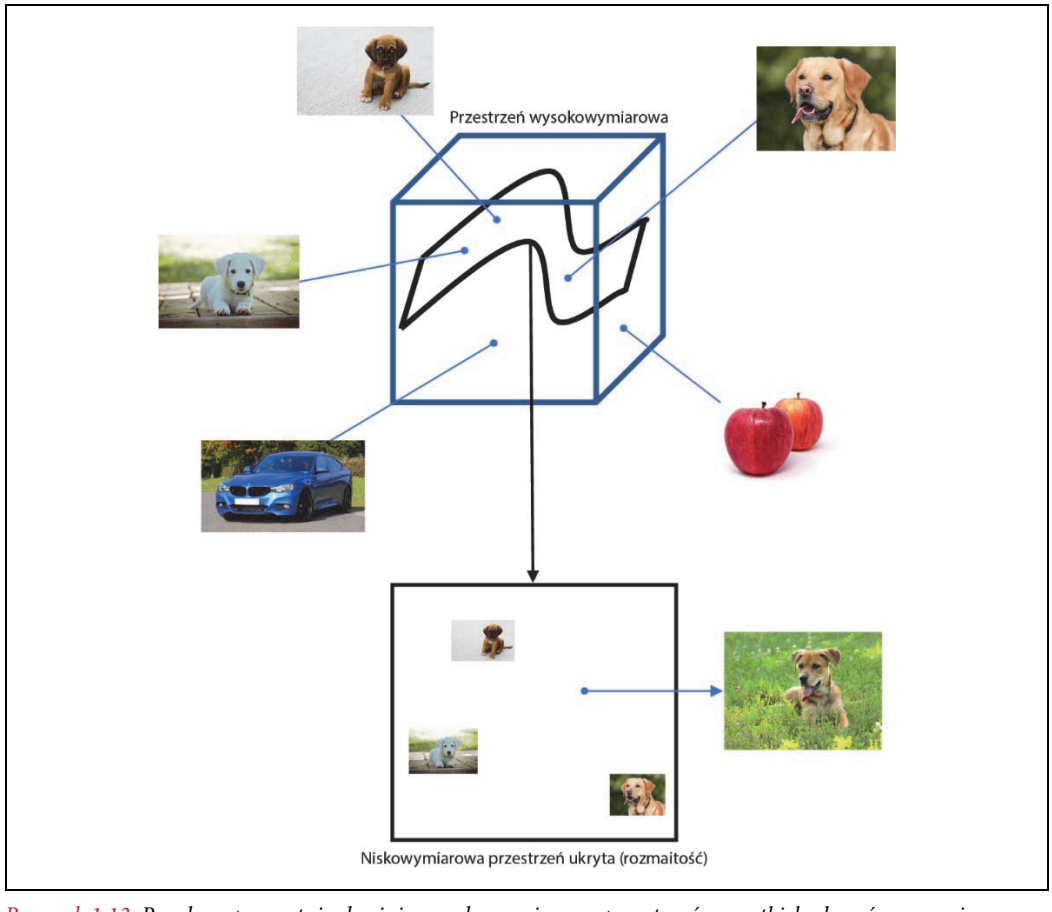
Uczenie głębokie daje nam wiele możliwości nauczenia się często bardzo złożonych funkcji mapowania *f*. Niektóre z najważniejszych technik przeanalizujemy w dalszych rozdziałach tej książki. Na razie wystarczy ogólnie zrozumieć, co stara się osiągnąć uczenie reprezentacji.

Jedną z zalet korzystania z uczenia reprezentacji jest możliwość wykonywania operacji w łatwiejszej do zarządzania przestrzeni ukrytej, która wpływa na wysokopoziomowe właściwości obrazu. Sposób dostosowania cieniowania każdego piksela tak, aby dana puszka herbatników wyglądała na *wyższą*, nie jest oczywisty. Jednak w przestrzeni ukrytej sprowadza się to do dodania 1 do wymiaru *wysokość*, a następnie zastosowania funkcji mapowania, by powrócić do dziedziny obrazu. Przykład tego zachowania zobaczymy w następnym rozdziale. Zastosujemy go jednak nie do puszek herbatników, lecz do twarzy.

Uczenie reprezentacji przychodzi ludziom tak naturalnie, że trudno nam przestać myśleć, jak niezwykłe jest to, iż możemy to robić zupełnie bez wysiłku. Załóżmy, że chcesz opisać swój wygląd komuś, kto patrzy na ciebie w tłumie ludzi i nie wie, jak wyglądasz. Nie zaczniesz od podania koloru

pierwszego piksela włosów, następnie piksela drugiego, potem trzeciego itd. Zamiast tego przyjmiesz rozsądne założenie, że druga osoba ma ogólne pojęcie o tym, jak wygląda przeciętny człowiek, po czym zmodyfikujesz tę bazę cechami opisującymi grupy pikseli, na przykład *mam bardzo jasne włosy* lub *noszę okulary*. Wystarczy nie więcej niż 10 takich zdań, aby rozmówca potrafił zmapować opis na piksele i wygenerować w swojej głowie obraz Ciebie. Obraz nie będzie idealny, ale będzie wystarczająco podobny do Twojego rzeczywistego wyglądu, aby było możliwe znalezienie Cię wśród setek innych ludzi, nawet jeśli osoba, która Cię szuka, nigdy wcześniej Cię nie widziała.

Należy pamiętać, że uczenie reprezentacji nie polega wyłącznie na przypisywaniu wartości do danego zbioru cech, na przykład odcienia blond włosów, wzrostu itp., dla pewnego obrazu. Siła uczenia reprezentacji polega na tym, że technika ta faktycznie uczy, które cechy są najważniejsze dla opisania określonej obserwacji oraz jak wygenerować te cechy na podstawie surowych danych. W języku matematyki uczenie reprezentacji próbuje znaleźć wysoce nieliniową *rozmaitość* (ang. *manifold*), na której bazują dane, a następnie ustalić wymiary wymagane do pełnego opisania tej przestrzeni. Zostało to pokazane na rysunku 1.13.



Rysunek 1.13. Ramka reprezentuje skrajnie wysokowymiarową przestrzeń wszystkich obrazów; uczenie reprezentacji próbuje znaleźć mniej wymiarową podprzestrzeń ukrytą lub rozmaitość, na której bazują poszczególne rodzaje obrazów (na przykład rozmaitość psów)

Podsumowując, nauka reprezentacji ustanawia najbardziej istotne wysokopoziomowe cechy opisujące sposób, w jaki są wyświetlane grupy pikseli. Dzięki temu istnieje wysokie prawdopodobieństwo, że każdy punkt w przestrzeni ukrytej jest reprezentacją dobrze uformowanego obrazu. Poprzez dostrajanie wartości cech w przestrzeni ukrytej możemy utworzyć nową reprezentację, która po zmapowaniu na oryginalną domenę obrazu daje znacznie większą szansę na realny wygląd niż wtedy, gdybyśmy próbowali bezpośrednio korzystać z pojedynczych pikseli.

Po zaprezentowaniu podstaw uczenia reprezentacji, które stanowi rdzeń wielu przykładów generatywnego uczenia głębokiego w tej książce, pozostaje skonfigurowanie środowiska, dzięki któremu będziemy mogli rozpocząć budowanie własnych modeli generatywnych uczenia głębokiego.

Konfiguracja środowiska

W tej książce zamieściłem wiele działających przykładów budowania modeli, które będę omawiać w tekście.

Aby uzyskać dostęp do tych przykładów, trzeba sklonować repozytorium Git towarzyszące tej książce. Git jest systemem kontroli wersji open source, który umożliwia skopiowanie kodu do lokalnego komputera. Dzięki temu można uruchamiać notatniki zarówno na własnym komputerze, jak i w środowisku chmury. Jeśli jeszcze nie zainstalowałeś tego środowiska, postępuj zgodnie z instrukcjami właściwymi dla danego systemu operacyjnego (*http://bit.ly/2MUrvN1*). Aby sklonować repozytorium dla tej książki, przejdź do folderu, w którym chcesz zapisać pliki, i wpisz w terminalu następujące polecenie:

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

Aby mieć najbardziej aktualną wersję, pamiętaj o uruchomieniu następującego polecenia:

```
git pull
```

Teraz powinieneś zobaczyć pliki znajdujące się w folderze na Twoim komputerze.

Następnie trzeba skonfigurować środowisko wirtualne. Jest to po prostu folder, w którym zainstalujesz świeżą kopię Pythona wraz ze wszystkimi pakietami, które będziemy wykorzystywać w tej książce. W ten sposób będziesz mieć pewność, że na Twoją systemową wersję Pythona nie będą miały wpływu żadne z bibliotek, które zainstalujesz.

Jeśli używasz środowiska Anaconda, możesz utworzyć środowisko wirtualne w następujący sposób:

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

Jeśli nie, możesz zainstalować narzędzia `virtualenv` i `virtualenvwrapper` za pomocą następującego polecenia[10]:

```
pip installvirtualenvvirtualenvwrapper
```

Należy również dodać do skryptu startowego powłoki (np. *bash_profile*) następujące wiersze:

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

[10] Dokładne instrukcje instalacji narzędzia `virtualenvwrapper` znajdziesz w dokumentacji (*http://bit.ly/2x8LPQ4*).

❶ Lokalizacja, w której będzie zapisane środowisko wirtualne.

❷ Domyślna wersja Pythona do wykorzystania podczas tworzenia wirtualnego środowiska — zadbaj o to, by wskazywała Pythona 3, a nie Pythona 2.

❸ Ponowne załadowanie skryptu inicjującego `virtualenvwrapper`.

Aby utworzyć wirtualne środowisko o nazwie `generative`, wpisz w terminalu następujące polecenie:

```
mkvirtualenvgenerative
```

Będziesz wiedzieć, że jesteś w środowisku wirtualnym, ponieważ terminal wyświetli przed symbolem zachęty ciąg `(generative)`.

Teraz możesz zainstalować wszystkie pakiety, których będziemy używać w tej książce, za pomocą następującego polecenia:

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

W tej książce będziemy używać Pythona 3. Plik *requirements.txt* zawiera nazwy i numery wersji wszystkich pakietów potrzebnych do uruchomienia przykładów.

Aby sprawdzić, czy wszystko działa zgodnie z oczekiwaniami, będąc w wirtualnym środowisku, wpisz w terminalu polecenie `python`, a następnie spróbuj zaimportować pakiet Keras (biblioteka uczenia głębokiego, z której będziemy często korzystać w tej książce). Powinieneś zobaczyć środowisko Pythona 3 oraz wyświetloną przez pakiet Keras informację, że wykorzystywany jest backend-TensorFlow (rysunek 1.14).

```
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'
```

Rysunek 1.14. Konfiguracja środowiska

Na koniec sprawdź, czy w swoim komputerze skonfigurowałeś dostęp do środowiska wirtualnego za pośrednictwem notatników Jupyter. Jupyter to środowisko umożliwiające interaktywne tworzenie kodu w Pythonie za pośrednictwem przeglądarki. To doskonały sposób na rozwijanie nowych pomysłów i współdzielenie kodu. Z wykorzystaniem notatników Jupyter powstała większość przykładów zamieszczonych w tej książce. Aby zapewnić dostęp do środowiska wirtualnego w środowisku Jupyter, uruchom w terminalu, wewnątrz środowiska wirtualnego, następujące polecenie:

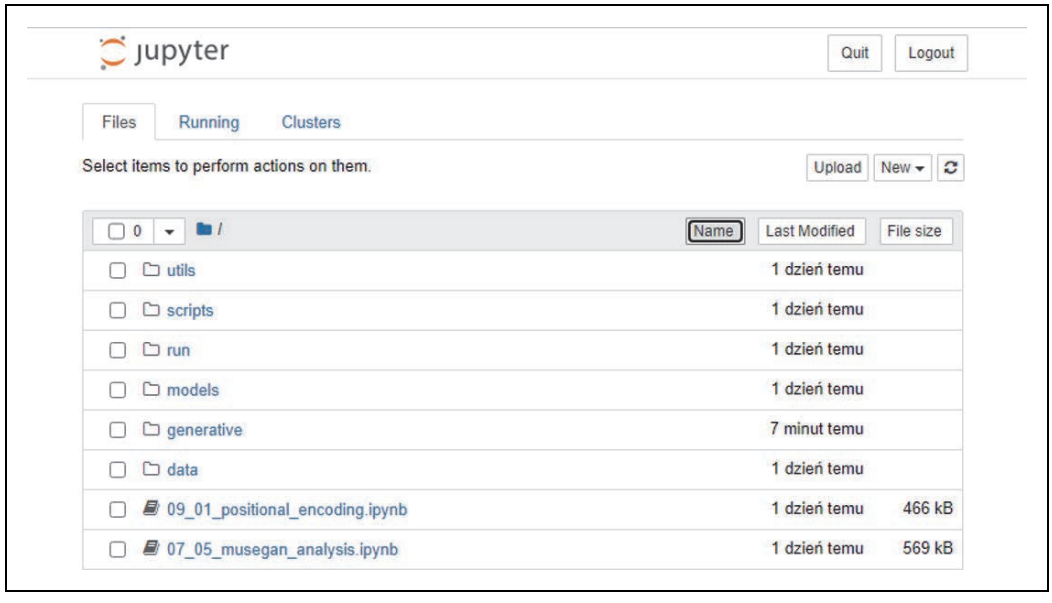
```
python -m ipykernel install --user --name generative ❶
```

❶ Daje dostęp do środowiska wirtualnego, które skonfigurowałeś (`generative`) wewnątrz notatników Jupyter.

Aby sprawdzić, czy wszystko zostało prawidłowo zainstalowane, przejdź w terminalu do folderu, w którym sklonowałeś repozytorium tej książki, a następnie wpisz polecenie:

```
jupyter notebook
```

W przeglądarce powinno otworzyć się okno podobne do pokazanego na rysunku 1.15. Kliknij notatnik, który chcesz uruchomić, a następnie z rozwijanego menu *Kernel/Changekernel* wybierz środowisko wirtualne `generative`.



Rysunek 1.15. Środowisko Jupyter notebook

Teraz jesteś gotowy, by rozpocząć budowę generatywnych głębokich sieci neuronowych.

Podsumowanie

Ten rozdział zawiera wprowadzenie do dziedziny modelowania generatywnego — ważnej gałęzi uczenia maszynowego, która uzupełnia częściej opisywane w literaturze modelowanie dyskryminatywne. W pierwszym prostym przykładzie modelu generatywnego wykorzystaliśmy naiwne założenie Bayesa, aby stworzyć rozkład prawdopodobieństwa zdolnego do reprezentowania wewnętrznej struktury danych i generowania przykładów spoza zbioru szkoleniowego. Pokazałem też, że proste modele zawodzą w sytuacji, gdy złożoność zadania generatywnego rośnie, oraz omówiłem ogólne wyzwania związane z modelowaniem generatywnym. Na koniec przyjrzeliśmy się uczeniu reprezentacji — ważnej koncepcji, która stanowi trzon wielu modeli generatywnych.

W rozdziale 2. zaczniesz poznawać uczenie głębokie oraz zobaczysz, jak korzystać z biblioteki Keras, aby budować modele zdolne do wykonywania zadań modelowania dyskryminatywnego. To da Ci niezbędne podstawy do rozwiązywania generatywnych problemów uczenia głębokiego w dalszych rozdziałach.

Czy potrafisz stworzyć... twórcę?

Techniki uczenia głębokiego rozwijają się w imponującym tempie, a sieci neuronowe znajdują zastosowanie w przeróżnych branżach. Coraz częściej komputer wykonuje zadania, które do niedawna były zarezerwowane dla człowieka. Dobrym przykładem jest tworzenie dzieł sztuki: ostatnie postępy w dziedzinie modelowania generatywnego sprawiają, że maszyny tworzą oryginalne obrazy w określonym stylu, piszą spójne akapity tekstu, komponują przyjemną w odbiorze muzykę i generują prawdopodobne scenariusze zdarzeń. Ta „generatywna rewolucja" już się zaczęła, a jej efekty przekraczają najśmielsze wyobrażenia.

Książka jest praktycznym przewodnikiem przeznaczonym dla inżynierów uczenia maszynowego i analityków danych. W jasny i przystępny sposób omówiono w niej zasadnicze zagadnienia teorii modelowania generatywnego, a następnie zaprezentowano techniki stosowane do budowy modeli generatywnych, włącznie z ogólnym opisem uczenia głębokiego, wariacyjnych autoenkoderów i generatywnych sieci antagonistycznych (GAN). Na tej podstawie — z wykorzystaniem biblioteki Keras — pokazano wewnętrzne funkcjonowanie każdej z tych technik, łącznie z najbardziej nowatorskimi architekturami. Opisano krok po kroku sposoby rozwiązywania takich twórczych zadań jak malowanie, pisanie i komponowanie muzyki, a także zastosowania modelowania generatywnego do optymalizacji strategii grania w gry (modele World).

W książce między innymi:

- działanie autoenkoderów wariacyjnych
- tworzenie sieci GAN, w tym CycleGAN i MuseGAN
- rekurencyjne modele generatywne do tworzenia tekstu oraz mechanizmy uwagi
- modele generatywne w środowiskach uczenia przez wzmacnianie
- architektura Transformer (BERT, GPT-2) oraz modele generowania obrazu

David Foster jest współzałożycielem Applied Data Science i ekspertem w dziedzinie inżynierii danych. Wygrał kilka międzynarodowych konkursów związanych z uczeniem maszynowym. Jest aktywnym członkiem społeczności internetowych skupionych wokół nauki o danych.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-7283-2

9 788328 372832

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 67,00 zł