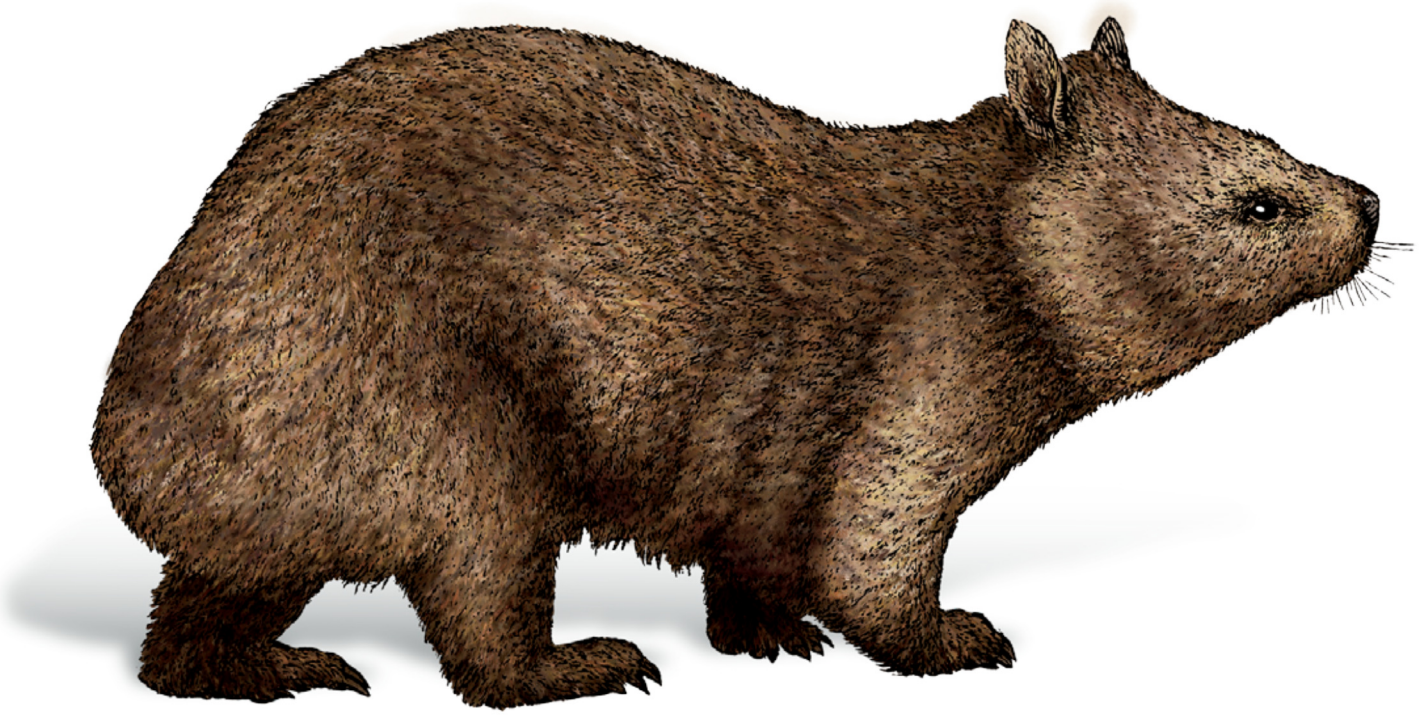
O'REILLY®

Praktyczne uczenie nienadzorowane

przy użyciu języka Python

Jak budować użytkowe rozwiązania uczenia maszynowego na podstawie nieoznakowanych danych.



Ankur A. Patel

Praktyczne uczenie nienadzorowane przy użyciu języka Python

Jak budować użytkowe rozwiązania uczenia maszynowego na podstawie nieoznakowanych danych.

Ankur A. Patel

przekład: Jakub Niedźwiedź

APN Promise
Warszawa 2020

O'REILLY®

Praktyczne uczenie nienadzorowane przy użyciu języka Python

Authorized translation of English edition of
Hands-On Unsupervised Learning Using Python
ISBN 978-1-492-03564-0

APN PROMISE SA, ul. Domaniewska 44a, 02-672 Warszawa
tel. +48 22 35 51 600, fax +48 22 35 51 699
e-mail: mspress@promise.pl

ISBN: 978-83-7541-426-4

Projekt okładki: Karen Montgomery
Ilustracje: Rebecca Demarest

Przekład: Jakub Niedźwiedź
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

Spis treści

Część II. Uczenie nienadzorowane przy użyciu Scikit-Learn

Część III. Uczenie nienadzorowane przy użyciu TensorFlow i Keras

Wstęp

Krótka historia uczenia maszynowego

Uczenie maszynowe jest dziedziną sztucznej inteligencji, w której komputery uczą się na podstawie danych – zwykle w celu poprawy swojej wydajności w jakimś wąsko określonym zadaniu – bez programowania ich bezpośrednio. Termin *uczenie maszynowe* został wprowadzony już w 1959 roku (przez Arthura Samuela, który był legendą w dziedzinie sztucznej inteligencji), ale w XX wieku niewiele było ważnych komercyjnych sukcesów w zakresie sztucznej inteligencji. Dziedzina ta pozostawała niszowym obszarem badawczym dla naukowców pracujących na uniwersytetach.

We wczesnych latach 60. XX wieku wielu członków społeczności zajmującej się sztuczną inteligencją było zbyt wielkimi optymistami co do jej przyszłości. Badacze z tego okresu, tacy jak Herbert Simon i Marvin Minsky, twierdzili, że sztuczna inteligencja osiągnie ludzki poziom w przeciągu kilku dziesięcioleci:[1]

> W ciągu dwudziestu lat maszyny będą w stanie wykonywać dowolną pracę, jaką może wykonać człowiek.
>
> *Herbert Simon, 1965*

> Za trzy do ośmiu lat będziemy mieli maszynę z ogólną inteligencją zbliżoną do przeciętnej istoty ludzkiej.
>
> *Marvin Minsky, 1970*

Zaślepieni swoim optymizmem, badacze skupiali się na projektach związanych z tak zwaną *silną sztuczną inteligencją* albo *ogólną sztuczną inteligencją* (*AGI – artificial general intelligence*), próbując zbudować moduły sztucznej inteligencji będące w stanie rozwiązywać problemy, reprezentować wiedzę, uczyć się i planować, przetwarzać język naturalny, postrzegać otoczenie i sterować poruszaniem. Ten optymizm pomógł przyciągnąć znaczące fundusze do tej rozwijającej się dziedziny, np. ze strony Departamentu Obrony Stanów Zjednoczonych, ale problemy, z którymi zmagali się ci badacze, były zbyt ambitne i ostatecznie skazane na porażkę.

1 Takie opinie zainspirowały Stanleya Kubricka w 1968 roku do stworzenia sztucznej inteligencji HAL 9000 w filmie *2001: Odyseja kosmiczna*.

Badania nad sztuczną inteligencją rzadko przekładały się na przemysłowe zastosowania i co jakiś czas następowały okresy wyhamowania tych badań zwane zimami sztucznej inteligencji. Podczas tych zim (analogia wywodziła się z popularnego w tym okresie określenia „zimna wojna") zainteresowanie sztuczną inteligencją i napływ funduszy malały. Od czasu do czasu pojawiały się okresy wzmożonego zainteresowania sztuczną inteligencją, ale nie były długotrwałe. We wczesnych latach 90. XX wieku zainteresowanie sztuczną inteligencją (i fundusze na nią) osiągnęło minimalny poziom.

Sztuczna inteligencja wróciła, ale dlaczego teraz?

W ostatnich dwóch dekadach sztuczna inteligencja pojawiła się ponownie – najpierw jako czysto akademicki obszar zainteresowań, a teraz już jako w pełni rozwinięta dziedzina przyciągająca najtęższe umysły na uniwersytetach i w korporacjach.

Trzy krytyczne elementy stoją za tym powrotem: przełom w algorytmach uczenia maszynowego, dostępność dużych ilości danych i niezwykle szybkie komputery.

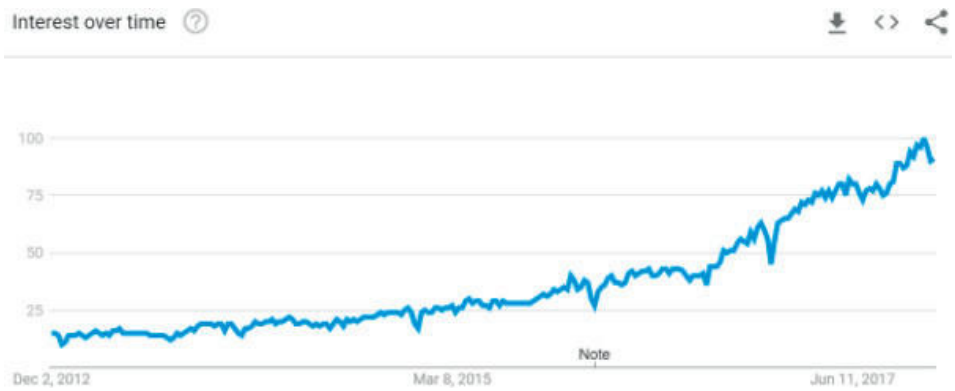
Przede wszystkim, zamiast skupiać się na nadmiernie ambitnych projektach związanych z silną sztuczną inteligencją, badacze zwrócili swoją uwagę na wąsko zdefiniowane podproblemy silnej sztucznej inteligencji, znane też jako *słaba sztuczna inteligencja* lub *wąska sztuczna inteligencja*. To skupienie na naprawianiu rozwiązań dotyczących wąsko zdefiniowanych zadań doprowadziło do przełomów algorytmicznych, które wytyczyły drogę do udanych aplikacji komercyjnych. Dla wielu z tych algorytmów – często opracowywanych początkowo na uniwersytetach lub w prywatnych laboratoriach badawczych – szybko udostępniono otwarty kod źródłowy, co przyspieszyło przyjęcie tych technologii przez przemysł.

Po drugie, wiele organizacji skupiło się na przechwytywaniu danych, a koszty przechowywania danych znacznie spadły, dzięki postępowi w technikach przechowywania danych cyfrowych. Dzięki Internetowi wiele danych również stało się szeroko i publicznie dostępnych na niewidzianą wcześniej skalę.

Po trzecie, komputery stały się znacznie potężniejsze i dostępne w chmurze obliczeniowej, pozwalając badaczom sztucznej inteligencji łatwo i tanio skalować swoją infrastrukturę informatyczną zgodnie z wymaganiami bez ponoszenia z góry ogromnych wydatków inwestycyjnych na sprzęt.

Pojawienie się stosowanej sztucznej inteligencji

Te trzy elementy umożliwiły zaadaptowanie rozwiązań sztucznej inteligencji przez przemysł, pomagając w przyciąganiu z każdym rokiem coraz wyższych poziomów funduszy i zainteresowania. Sztuczna inteligencja nie jest już tylko teoretyczną dziedziną zainteresowań, ale całkiem dobrze rozwiniętym polem praktycznych zastosowań. Rysunek W-1 pokazuje wykres z Google Trends, wskazujący na wzrost zainteresowania uczeniem maszynowym w ostatnich pięciu latach.



Rysunek W-1. *Zainteresowanie uczeniem maszynowym w ostatnim czasie*

Sztuczna inteligencja jest teraz postrzegana jako przełomowa technologia horyzontalna powiązana z rozwojem komputerów i smartfonów, która będzie miała istotny wpływ na każdą dziedzinę gospodarki w najbliższej dekadzie[2].

Do udanych zastosowań komercyjnych wykorzystujących uczenie maszynowe należą między innymi: optyczne rozpoznawanie znaków, filtrowanie spamu w poczcie elektronicznej, klasyfikacja obrazów, komputerowe rozpoznawanie obrazów, rozpoznawanie mowy, tłumaczenie maszynowe, grupowanie danych, generowanie danych syntetycznych, wykrywanie anomalii, zapobieganie cyberprzestępczości, wykrywanie oszustw związanych z kartami kredytowymi, przewidywanie szeregów czasowych, przetwarzanie języka naturalnego, gry planszowe i gry wideo, klasyfikowanie dokumentów, systemy rekomendacyjne, wyszukiwarki, robotyka, reklama internetowa, analiza nastrojów, sekwencjonowanie DNA, analiza rynków finansowych, wyszukiwanie informacji, zautomatyzowane odpowiadanie na pytania, podejmowanie decyzji w ochronie zdrowia.

Główne kamienie milowe w stosowanej sztucznej inteligencji w ostatnich 20 latach

Przedstawione tutaj kamienie milowe pomogły przejść sztucznej inteligencji z obszaru głównie akademickiego tematu rozważań do głównego nurtu technologicznego.

- 1997: Deep Blue, bot sztucznej inteligencji, który był opracowywany od połowy lat 80. XX wieku, pokonuje szachowego mistrza świata Garriego Kasparowa w publicznie nagłośnionym meczu szachowym.
- 2004: DARPA inicjuje DARPA Grand Challenge, coroczne zawody autonomicznych pojazdów przeprowadzane na pustyni. W 2005 roku Uniwersytet Stanforda zdobywa główną nagrodę. W 2007 roku Uniwersytet Carnegie Mellon przeprowadza tę próbę w przestrzeni miejskiej. W 2009 roku Google buduje swój samochód bezzałogowy.

2 Według McKinsey Global Institute ponad połowa wszystkich działań zawodowych, za które ludzie otrzymują wynagrodzenie, może zostać zautomatyzowana do 2055 roku.

W 2015 roku wielu gigantów technologicznych (w tym Tesla, Alphabet's Waymo i Uber) uruchomiło programy mające na celu stworzenie technologii pojazdów autonomicznych.

- 2006: Geoffrey Hinton z Uniwersytetu w Toronto przedstawia szybko uczący się algorytm do szkolenia wielowarstwowych sieci neuronowych, zapoczątkowując rewolucję uczenia głębokiego (deep learning).
- 2006: Netflix uruchamia konkurs Netflix Prize z nagrodą w wysokości miliona dolarów, zachęcając uczestniczące zespoły do wykorzystania uczenia maszynowego w celu poprawienia dokładności swojego systemu rekomendacji o co najmniej 10%. Jeden z uczestniczących zespołów zdobył tę nagrodę w 2009 roku.
- 2007: Sztuczna inteligencja odnosi nadludzkie wyniki w grze w warcaby rozwiązanej przez zespół z Uniwersytetu Alberta.
- 2010: ImageNet uruchamia coroczne zawody – ImageNet Large Scale Visual Recognition Challenge (ILSVRC) – w których zespoły wykorzystują algorytmy uczenia maszynowego w celu poprawnego wykrywania i klasyfikowania obiektów z dużego zestawu danych z obrazami. Przyciąga to uwagę zarówno środowisk uniwersyteckich, jak i gigantów technologicznych. Wskaźnik błędu klasyfikacji spadł z 25% w 2011 roku do jedynie kilku procent w 2015 roku, dzięki znaczącym postępom w głębokich splotowych sieciach neuronowych. Prowadzi to do komercyjnych zastosowań komputerowego rozpoznawania obiektów.
- 2010: Microsoft wypuszcza urządzenie Kinect dla konsoli Xbox 360. Opracowany przez zespół widzenia komputerowego w dziale Microsoft Research, Kinect może śledzić ruch ludzkiego ciała i wykorzystywać go w grach.
- 2010: Siri, jedna z pierwszych cyfrowych asystentek głosowych, zostaje przejęta przez firmę Apple i włączona do telefonu iPhone 4S w październiku 2011 roku. Później zostaje wbudowana w całą gamę produktów firmy Apple. Wspomagana przez splotowe sieci neuronowe i sieci neuronowe długiej pamięci krótkoterminowej, Siri obsługuje rozpoznawanie mowy i przetwarzanie języka naturalnego. Firmy Amazon, Microsoft i Google włączają się do tego wyścigu, wypuszczając swoich asystentów: Alexa (2014), Cortana (2014) oraz Google Assistant (2016).
- 2011: IBM Watson, agent sztucznej inteligencji odpowiadający na pytania, opracowany przez zespół kierowany przez Davida Ferrucciego pokonuje wcześniejszych zwycięzców teleturnieju *Jeopardy!* (w Polsce znany jako *Va banque*): Brada Ruttera i Kena Jenningsa. IBM Watson jest teraz wykorzystywany w kilku branżach, między innymi w handlu detalicznym i ochronie zdrowia.
- 2012: Zespół Google Brain pod przewodnictwem Andrew Ng i Jeffa Deana szkoli sieć neuronową w rozpoznawaniu kotów przy przeglądaniu nieoznakowanych obrazów pochodzących z filmów w serwisie YouTube.
- 2013: Google zwycięża w organizowanym przez DARPA konkursie Robotics Challenge, obejmującym próby, w których półautonomiczne boty wykonywały w trudnych warunkach złożone zadania, takie jak kierowanie samochodem,

chodzenie po rumowisku, usuwanie gruzu z zablokowanego wejścia, otwieranie drzwi i wchodzenie po drabinie.

- 2014: Facebook publikuje swoje prace nad DeepFace, systemem opartym na sieciach neuronowych, który może identyfikować twarze z dokładnością 97%. Jest to poziom zbliżony do ludzkiego i stanowi poprawę o ponad 27% w stosunku do wcześniejszych systemów.
- 2015: Sztuczna inteligencja pojawia się powszechnie w środkach masowego przekazu na całym świecie.
- 2015: Google DeepMind's AlphaGo pokonuje światowej klasy zawodnika Fan Hui w grze w Go. W 2016 AlphaGo pokonuje Lee Sedola, a w roku 2017 pokonuje Ke Jie. W 2017 roku nowa wersja o nazwie AlphaGo Zero pokonuje poprzednią wersję AlphaGo wynikiem 100 do 0. AlphaGo Zero posługuje się technikami uczenia nienadzorowanego i opanowuje grę w Go, po prostu grając ze sobą.
- 2016: Google wypuszcza przeróbkę swojego narzędzia tłumaczącego Google Translate, zastępując dotychczasowy system tłumaczenia oparty na frazach, systemem tłumaczenia opartym na sieciach neuronowych, wykorzystując uczenie głębokie, ograniczając błędy tłumaczenia o ponad 87% i zbliżając się do niemal ludzkiego poziomu dokładności.
- 2017: Libratus, opracowany przez Uniwersytet Carnegie Mellon, wygrywa turniej Texas Hold'em (w wariancie „head-to-head no-limit").
- 2017: Bot przeszkolony przez OpenAI pokonuje profesjonalnego gracza w turnieju Dota 2.

Od wąskiej sztucznej inteligencji do ogólnej sztucznej inteligencji

Oczywiście te sukcesy w stosowaniu sztucznej inteligencji w wąsko zdefiniowanych problemach stanowią jedynie punkt wyjścia. Istnieje rosnące przekonanie w środowisku sztucznej inteligencji, że łącząc kilka systemów słabej sztucznej inteligencji, możemy stopniowo rozwijać silną sztuczną inteligencję. Taki agent silnej sztucznej inteligencji albo ogólnej sztucznej inteligencji będzie miał możliwości zbliżone do ludzkich w wielu szeroko zdefiniowanych zadaniach.

Niektórzy badacze przewidują, że wkrótce po tym, jak sztuczna inteligencja osiągnie poziom człowieka, ta silna sztuczna inteligencja zacznie przewyższać ludzką inteligencję i osiągnie poziom tak zwanej *superinteligencji*. Szacunki dotyczące momentu osiągnięcia takiej superinteligencji wahają się od 15 lat do nawet 100 lat od chwili obecnej, ale większość badaczy uważa, że sztuczna inteligencja rozwinie się na tyle, aby osiągnąć ten poziom za kilka generacji. Czy to kolejne rozdmuchane nadzieje (jakie widzieliśmy w poprzednich cyklach rozwoju sztucznej inteligencji), czy teraz jest inaczej?

Tylko czas pokaże.

Cel i podejście

Większość udanych aplikacji komercyjnych do tej pory – w obszarach, takich jak widzenie komputerowe, rozpoznawanie mowy, tłumaczenie maszynowe i przetwarzanie języka naturalnego – była związana z uczeniem nadzorowanym, wykorzystując oznakowane zbiory danych. Jednakże większość danych na świecie jest *nieoznakowana*.

W tej książce zajmiemy się dziedziną *uczenia nienadzorowanego* (które jest gałęzią uczenia maszynowego wykorzystywaną do znajdowania ukrytych wzorców) i uczeniem się podstawowej struktury danych nieoznakowanych. Według wielu ekspertów branżowych, takich jak Yann LeCun, Dyrektor badań nad sztuczną inteligencją w firmie Facebook oraz profesor NYU, uczenie nienadzorowane jest kolejną granicą do pokonania w rozwoju sztucznej inteligencji i może stanowić klucz do osiągnięcia ogólnej sztucznej inteligencji. Z tego i wielu innych powodów uczenie nienadzorowane jest obecnie jednym z najpopularniejszych zagadnień w dziedzinie sztucznej inteligencji.

Celem tej książki jest nakreślenie pojęć i narzędzi niezbędnych do rozwinięcia intuicji koniecznej do zastosowania tej technologii w codziennych problemach, nad którymi pracujemy. Innymi słowy, jest to praktyczna książka, która pomoże nam budować rzeczywiste systemy. Zbadamy również, jak skutecznie oznakowywać nieoznakowane zestawy danych, aby przekształcać problemy związane z uczeniem nienadzorowanym na pół-nadzorowane.

Ta książka stosuje praktyczne podejście, wprowadzając nieco teorii, ale skupiając się głównie na zastosowaniu technik uczenia nienadzorowanego do rozwiązywania rzeczywistych problemów. Zestawy danych i kod są dostępne w formacie Jupyter Notebooks w serwisie GitHub (*http://bit.ly/2Gd4v7e*).

Uzbrojeni w zrozumienie najważniejszych pojęć i praktyczne doświadczenie uzyskane z tej książki, będziemy w stanie stosować uczenie nienadzorowane wobec dużych zbiorów danych nieoznakowanych, aby odkrywać ukryte wzorce, uzyskiwać głębszy wgląd w zależności biznesowe, wykrywać anomalie, tworzyć grupy w oparciu o podobieństwa, przeprowadzać automatyczne wybieranie cech, generować syntetyczne zestawy danych i nie tylko.

Wymagania wstępne

Ta książka zakłada, że czytelnik ma pewne doświadczenie w programowaniu w języku Python, w tym znajomość NumPy i Pandas.

Więcej informacji na temat języka Python można znaleźć na oficjalnej witrynie języka Python (*https://www.python.org/*). Więcej na temat Jupyter Notebook można znaleźć na oficjalnej witrynie Jupyter (*http://jupyter.org/index.html*). Jako powtórkę z zagadnień dotyczących rachunku całkowego, algebry liniowej, prawdopodobieństwa i statystyki polecam przeczytanie części I podręcznika *Deep Learning* (*http://www.deeplearningbook.org/*), którego autorami są Ian Goodfellow i Yoshua Bengio. Jako powtórkę ogólnych zagadnień uczenia maszynowego polecam przeczytanie pracy *The Elements of Statistical Learning* (*https://stanford.io/2Tju4al*).

Mapa drogowa

Ta książka jest podzielona na cztery części, omawiające następujące tematy:

Część I, Podstawy uczenia nienadzorowanego
: Różnice pomiędzy uczeniem nadzorowanym i nienadzorowanym, przegląd popularnych algorytmów nadzorowanych i nienadzorowanych oraz kompleksowy projekt wykorzystujący uczenie maszynowe.

Część II, Uczenie nienadzorowane przy użyciu scikit-learn
: Redukcja wymiarowości, wykrywanie anomalii oraz analiza skupień (grupowanie) i segmentacja grup.

Więcej informacji na temat pojęć omawianych w częściach I oraz II można znaleźć w dokumentacji scikit-learn (*https://scikit-learn.org/stable/modules/classes.html*).

Część III, Uczenie nienadzorowane przy użyciu TensorFlow i Keras
: Uczenie reprezentacji i automatyczne wyodrębnianie cech, autokodery i uczenie pół-nadzorowane.

Głębokie uczenie nienadzorowane przy użyciu TensorFlow i Keras
: Ograniczone automaty Boltzmanna, sieci głębokiego przekonania i generujące sieci antagonistyczne.

Konwencje wykorzystywane w tej książce

W tej książce używane są następujące konwencje typograficzne:

Kursywa
: Wskazuje nowe pojęcia, adresy URL, adresy e-mail, nazwy plików i rozszerzenia plików.

`Stała szerokość liter`
: Używana w przykładach programów, a także w treści akapitów przy odwołaniach do elementów programów, takich jak nazwy zmiennych lub funkcji, bazy danych, typy danych, zmienne środowiskowe, instrukcje i słowa kluczowe.

`Stała szerokość liter i pogrubienie`
: Pokazuje polecenia lub inny tekst, który powinien być dosłownie wpisany przez użytkownika.

`Stała szerokość liter i kursywa`
: Pokazuje tekst, który powinien być zastąpiony wartościami podanymi przez użytkownika lub wynikającymi z kontekstu.

Ten element oznacza wskazówkę lub sugestię.

Ten element oznacza ogólną uwagę.

Ten element oznacza ostrzeżenie.

Korzystanie z przykładów kodu

Materiały dodatkowe (przykłady kodu, itd.) są dostępne do pobrania z serwisu GitHub (*http://bit.ly/2Gd4v7e*).

Ta książka ma pomóc czytelnikom w realizacji własnych projektów. Jeśli jakiś kod przykładowy jest oferowany w tej książce, można z niego korzystać w swoich programach i dokumentacji. Nie trzeba uzyskiwać od nas pozwolenia, o ile nie kopiuje się znacznej ilości kodu. Na przykład napisanie programu, który wykorzystuje kilka fragmentów kodu z tej książki, nie wymaga zezwolenia. Sprzedawanie lub dystrybucja dysku CD-ROM z przykładami kodu z książek wydawnictwa O'Reilly wymaga pozwolenia. Cytowanie tej książki i przykładów kodu w odpowiedzi na czyjeś pytanie nie wymaga pozwolenia. Włączenie znaczącej ilości kodu przykładowego z tej książki do dokumentacji swojego produktu wymaga pozwolenia.

Doceniamy powołanie się na źródło, ale go nie wymagamy. Zwykle zawiera ono tytuł, autora, wydawcę i numer ISBN książki. Na przykład: „Praktyczne uczenie nienadzorowane przy użyciu języka Python, Ankur A. Patel (O'Reilly / APN Promise). Copyright 2019 Ankur A. Patel, 978-83-7541-426-4".

Jeśli ktoś ma wątpliwości, czy użycie przykładów kodu z tej książki wymaga dodatkowego zezwolenia, może się z nami skontaktować pod adresem *permissions@oreilly.com*.

Nauka z O'Reilly przez Internet

Od ponad 40 lat O'Reilly Media (*http://oreilly.com*) zapewnia szkolenia technologiczne i biznesowe pomagające firmom w osiąganiu sukcesów.

Nasza unikalna sieć ekspertów i innowatorów dzieli się swoją wiedzą i doświadczeniem poprzez książki, artykuły, konferencje oraz platformę nauczania internetowego. Internetowa platforma O'Reilly's daje dostęp na żądanie do szkoleń prowadzonych na żywo, dogłębnych kursów, interaktywnych środowisk kodowania i ogromnej kolekcji materiałów tekstowych i wideo z wydawnictwa O'Reilly i ponad 200 innych wydawców. Więcej informacji można uzyskać pod adresem *http://oreilly.com*.

Jak się z nami skontaktować

Komentarze i pytania dotyczące tej książki należy kierować na adres wydawcy:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (w USA lub Kanadzie)

707-829-0515 (połączenie międzynarodowe lub lokalne)

707-829-0104 (fax)

Mamy stronę WWW dla tej książki, gdzie umieszczamy erratę, przykłady i wszelkie dodatkowe informacje. Można uzyskać dostęp do tej strony poprzez adres *http://bit.ly/unsupervised-learning*.

Aby przesłać komentarz lub zadać pytanie techniczne dotyczące tej książki, należy wysłać wiadomość pod adres *bookquestions@oreilly.com*.

Więcej informacji na temat naszych książek, kursów i konferencji można znaleźć na naszej witrynie WWW pod adresem *http://www.oreilly.com*.

Można nas znaleźć na Facebooku pod adresem: *http://facebook.com/oreilly*

Można nas śledzić w serwisie Twitter: *http://twitter.com/oreillymedia*

Można nas oglądać na YouTube: *http://www.youtube.com/oreillymedia*

Część I

Podstawy uczenia nienadzorowanego

Na początek przyjrzyjmy się obecnemu ekosystemowi uczenia maszynowego i gdzie jest w nim miejsce na uczenie nienadzorowane. Zbudujemy też od podstaw projekt związany z uczeniem maszynowym, obejmujący podstawy, takie jak przygotowywanie środowiska programowania, pozyskanie i przygotowanie danych, badanie danych, wybieranie algorytmów uczenia maszynowego i funkcji kosztów oraz ocena wyników.

Rozdział 1

Uczenie nienadzorowane w ekosystemie uczenia maszynowego

> W większości uczenie się przez ludzi i zwierzęta jest uczeniem nienadzorowanym. Gdyby inteligencja była tortem, uczenie nienadzorowane byłoby samym tortem, uczenie nadzorowane byłoby lukrem na torcie, a uczenie wzmacniane byłoby wisienką na torcie. Wiemy, jak zrobić lukier i wisienkę, ale nie wiemy, jak zrobić tort. Musimy rozwiązać problem z uczeniem nienadzorowanym, zanim będziemy mogli pomyśleć o dotarciu do prawdziwej sztucznej inteligencji.
>
> *Yann LeCun*

W tym rozdziale zbadamy różnicę pomiędzy systemem opartym na zasadach a uczeniem maszynowym, różnicę pomiędzy uczeniem nadzorowanym a uczeniem nienadzorowanym oraz ich mocne i słabe strony.

Omówimy też wiele popularnych algorytmów uczenia nadzorowanego oraz uczenia nienadzorowanego oraz krótko zbadamy, jak się ma do tego uczenie pół-nadzorowane oraz uczenie wzmacniane.

Podstawowa terminologia związana z uczeniem maszynowym

Zanim zagłębimy się w różne typy uczenia maszynowego, przyjrzyjmy się prostemu i powszechnie używanemu przykładowi uczenia maszynowego, aby ułatwić zrozumienie wprowadzanych pojęć: filtrowi spamu w poczcie elektronicznej. Musimy zbudować prosty program, który przyjmuje wiadomości e-mail i poprawnie klasyfikuje je jako „spam" lub „nie spam". Jest to prosty problem związany z klasyfikowaniem.

Przypomnijmy nieco terminologii związanej z uczeniem maszynowym: *zmiennymi wejściowymi* dla tego problemu są teksty wiadomości e-mail. Te zmienne wejściowe są też znane jako *cechy* albo *prognostyki*, albo *zmienne niezależne*. *Zmienną wyjściową* – tym, co próbujemy przewidzieć – jest *oznakowanie* „spam" albo „nie spam". Jest ona też znana jako *zmienna docelowa*, *zmienna zależna* lub *zmienna odpowiedzi* (albo *klasa*, gdyż jest to problem związany z klasyfikowaniem).

Zestaw przykładów, na których szkolona jest sztuczna inteligencja, jest zwany *zestawem szkoleniowym*, a każdy przykład jest zwany *przykładem szkoleniowym* lub *próbką*. Podczas szkolenia sztuczna inteligencja próbuje minimalizować swoją *funkcję kosztu* albo *wskaźnik błędu,* lub też z bardziej pozytywnego punktu widzenia – maksymalizować swoją *funkcję wartości* – w tym przypadku współczynnik prawidłowo sklasyfikowanych wiadomości e-mail. Sztuczna inteligencja aktywnie się optymalizuje podczas szkolenia w kierunku osiągnięcia minimalnego wskaźnika błędu. Wskaźnik błędu jest obliczany przez porównywanie oznakowania przewidzianego przez sztuczną inteligencję z prawdziwym oznakowaniem.

Najbardziej zależy nam jednak na tym, jak dobrze sztuczna inteligencja uogólnia swoje szkolenie przy klasyfikowaniu nigdy wcześniej niewidzianych wiadomości e-mail. To będzie prawdziwy test dla sztucznej inteligencji: czy może poprawnie klasyfikować wiadomości e-mail, których nigdy wcześniej nie widziała, wykorzystując to, czego nauczyła się poprzez szkolenie na przykładach w zestawie szkoleniowym? Ten *błąd uogólnienia* lub *błąd poza próbką* jest główną miarą używaną do oceny rozwiązań uczenia maszynowego.

Taki zestaw nigdy wcześniej niewidzianych przykładów jest znany jako *zestaw testowy* albo *zestaw odłożony* (ponieważ dane te nie są wykorzystywane podczas szkolenia). Jeśli będziemy mieli kilka zestawów odłożonych (być może do oceny naszego błędu uogólnienia podczas szkolenia), możemy mieć pośrednie zestawy odłożone, które wykorzystujemy do oceny postępów przed ostatecznym zestawem testowym; te pośrednie zestawy odłożone są nazywane *zestawami weryfikacyjnymi*.

Zbierając to wszystko razem, sztuczna inteligencja szkoli się na danych szkoleniowych (*doświadczenie*), aby poprawić swój wskaźnik błędu (*wydajność*) w ocenianiu spamu (*zadanie*), a kryterium końcowego sukcesu jest to, jak dobrze jej doświadczenie może zostać uogólnione dla nowych, nigdy wcześniej niewidzianych danych (*błąd uogólnienia*).

System oparty na zasadach a uczenie maszynowe

Wykorzystując podejście oparte na zasadach, możemy zaprojektować filtr spamu z jawnymi zasadami wychwytującymi spam, na przykład takimi, które oceniają jako spam wiadomości e-mail zawierające „u" zamiast „you", „4" zamiast „for", „KUP TERAZ", itd. Ten system byłby jednak trudny w utrzymaniu, gdyż zachowanie rozsyłających spam będzie się zmieniać w czasie, żeby unikać tych zasad. Gdybyśmy korzystali z systemu opartego na zasadach, musielibyśmy często dostosowywać ręcznie te zasady, aby je aktualizować. Byłoby to też bardzo kosztowne do skonfigurowania – pomyślmy o wszystkich zasadach, które musielibyśmy stworzyć, aby był to dobrze działający system.

Zamiast podejścia opartego na zasadach możemy wykorzystać uczenie maszynowe, aby przeszkolić je na danych zawierających wiadomości e-mail i automatycznie stworzyć reguły poprawnie oznaczające niechciane wiadomości e-mail jako spam. Ten system oparty na uczeniu maszynowym mógłby się też automatycznie dostosowywać w czasie. Ten system byłby znacznie tańszy do przeszkolenia i utrzymania.

W tym prostym przykładzie z wiadomościami e-mail możliwe byłoby dla nas ręczne opracowanie zasad, ale w przypadku wielu problemów ręczne opracowanie zasad nie jest w ogóle możliwe. Weźmy na przykład pod uwagę samochód autonomiczny – wyobraźmy sobie zaprojektowanie zasad, jak samochód powinien się zachować w każdym pojedynczym przypadku, jaki kiedykolwiek napotka. Jest to nierozwiązywalny problem, o ile samochód nie będzie mógł się uczyć i dostosowywać swojego zachowania w oparciu o swoje doświadczenie.

Moglibyśmy też użyć systemów uczenia maszynowego jako narzędzia do eksploracji lub odkrywania danych, aby zyskać głębszy wgląd w problem, który próbujemy rozwiązać. W przykładzie z filtrem spamu moglibyśmy uczyć się, które słowa lub wyrażenia najczęściej prowadzą do oceny wiadomości jako spam i rozpoznawać nowo pojawiające się wzorce złośliwych wiadomości.

Uczenie nadzorowane a nienadzorowane

Dziedzina uczenia maszynowego ma dwie główne gałęzie – *uczenie nadzorowane* i *uczenie nienadzorowane* – i wiele podgałęzi pomiędzy tymi dwoma przypadkami.

W uczeniu nadzorowanym agent sztucznej inteligencji ma dostęp do oznakowań, które może wykorzystywać do poprawienia swojej wydajności w danym zadaniu. W problemie z filtrem spamu mamy zbiór danych z wiadomościami e-mail, obejmujący wszystkie teksty we wszystkich wiadomościach e-mail. Wiemy też, które z tych wiadomości e-mail są spamem, a które nie (są to tak zwane *oznakowania*). Oznakowania te są bardzo cenne, pomagając sztucznej inteligencji opartej na uczeniu nadzorowanym oddzielać spam od pozostałych wiadomości e-mail.

W uczeniu nienadzorowanym oznakowania nie są dostępne. W związku z tym zadanie agenta sztucznej inteligencji nie jest dobrze zdefiniowane i nie można tak wyraźnie mierzyć jego wydajności. Rozważmy problem filtra spamu – tym razem bez oznakowań. Agent sztucznej inteligencji będzie teraz próbował zrozumieć podstawową strukturę wiadomości e-mail, rozdzielając bazę danych z wiadomościami e-mail na różne grupy, tak aby wiadomości e-mail w danej grupie były podobne do siebie, ale inne od wiadomości e-mail w innych grupach.

Ten problem uczenia nienadzorowanego jest mniej jasno określony niż problem uczenia nadzorowanego i trudniejszy do rozwiązania przez agenta sztucznej inteligencji. Jeśli zostanie dobrze przeprowadzony, rozwiązanie będzie potężniejsze.

Oto, dlaczego tak jest: sztuczna inteligencja w przypadku uczenia nienadzorowanego może znaleźć kilka grup, które później oznaczy jako „spam", ale może też znaleźć grupy, które później oznaczy jako „ważne" lub zaklasyfikuje do kategorii „rodzina", „praca",

„wiadomości", „zakupy", itd. Innymi słowy, ponieważ problem nie ma ściśle zdefiniowanego zadania, agent sztucznej inteligencji może znaleźć interesujące wzorce wykraczające poza nasz początkowy zakres poszukiwań.

Co więcej, ten system nienadzorowany jest lepszy od systemu nadzorowanego w znajdowaniu nowych wzorców w przyszłych danych, co sprawia, że rozwiązanie nienadzorowane staje się bardziej elastyczne. W tym tkwi siła uczenia nienadzorowanego.

Mocne i słabe strony uczenia nadzorowanego

Uczenie nadzorowane doskonale sprawdza się przy optymalizowaniu wydajności dla dobrze zdefiniowanych zadań z dużą liczbą oznakowań. Rozważmy na przykład bardzo duży zbiór danych z obrazami obiektów, gdzie każdy obraz jest oznaczony etykietą. Jeśli zestaw danych będzie wystarczająco duży i przeszkolimy odpowiednie algorytmy uczenia maszynowego (tzn. splotowe sieci neuronowe) na wystarczająco wydajnych komputerach, będziemy mogli zbudować bardzo dobry system klasyfikacji oparty na uczeniu nadzorowanym.

Ponieważ sztuczna inteligencja oparta na uczeniu nadzorowanym szkoli się na danych, będziemy w stanie zmierzyć jej wydajność (poprzez funkcję kosztu), porównując przewidziane przez nią oznakowanie obrazu z prawdziwym oznakowaniem obrazu, które mamy w zbiorze danych. Sztuczna inteligencja spróbuje jawnie zminimalizować tę funkcję kosztu, tak aby jej błąd dla nigdy wcześniej niewidzianych obrazów (z zestawu odłożonego) był możliwie jak najmniejszy.

Właśnie dlatego oznakowania są tak potężne – pomagają kierować agentem sztucznej inteligencji, zapewniając mu możliwość pomiaru błędu. Sztuczna inteligencja wykorzystuje miarę błędu do poprawienia swojej wydajności w czasie. Bez takich oznakowań sztuczna inteligencja nie wie, jak jest skuteczna (lub nie) w poprawnym klasyfikowaniu obrazów.

Jednakże koszty ręcznego oznakowania zbioru danych z obrazami są wysokie. Nawet najbardziej zadbane zbiory danych z obrazami będą miały jedynie tysiące oznakowań. Stanowi to problem, ponieważ systemy uczenia nadzorowanego będą bardzo dobre w klasyfikowaniu obrazów obiektów, dla których mają oznakowania, ale słabe w klasyfikowaniu obrazów obiektów, dla których nie ma oznakowań.

Systemy uczenia nadzorowanego są potężne, ale mają ograniczone możliwości uogólniania swojej wiedzy ponad oznakowania, na których zostały przeszkolone. Ponieważ większość światowych danych jest nieoznakowana, w przypadku uczenia nadzorowanego zdolność sztucznej inteligencji do poszerzania swojej wydajności na nigdy wcześniej niewidziane przypadki jest dość ograniczona.

Innymi słowy, uczenie nadzorowane jest świetne w rozwiązywaniu problemów wąskiej sztucznej inteligencji, ale nie tak dobre w rozwiązywaniu bardziej ambitnych, mniej wyraźnie zdefiniowanych problemów silnej sztucznej inteligencji.

Mocne i słabe strony uczenia nienadzorowanego

Uczenie nadzorowane bije uczenie nienadzorowane na głowę w wąsko zdefiniowanych zadaniach, dla których mamy dobrze zdefiniowane wzorce, które nie zmieniają się zbytnio w czasie, oraz wystarczająco duże, łatwo dostępne, oznakowane zbiory danych.

Jednak w przypadku problemów, w których wzorce nie są znane lub stale się zmieniają albo dla których nie mamy wystarczająco dużych oznakowanych zbiorów danych, lepiej sprawdza się uczenie nienadzorowane.

Zamiast kierować się oznakowaniem, uczenie nienadzorowane działa poprzez uczenie się podstawowej struktury danych, które są używane do szkolenia. Robi to, próbując przedstawić dane, na których się szkoli przy pomocy zestawu parametrów, który jest znacząco mniejszy od liczby przykładów dostępnych w zestawie danych. Przeprowadzając takie uczenie reprezentacji, uczenie nienadzorowane jest w stanie identyfikować różne wzorce w zestawie danych.

W przykładzie z zestawem danych obrazów (tym razem bez oznakowań) sztuczna inteligencja oparta na uczeniu nienadzorowanym może być w stanie identyfikować i grupować obrazy w oparciu o to, jak bardzo są podobne do siebie i jak różne są od pozostałych. Na przykład wszystkie obrazy, które wyglądają jak krzesła, będą zgrupowane ze sobą, a wszystkie obrazy, które wyglądają jak psy, będą zgrupowane ze sobą, itd.

Oczywiście sztuczna inteligencja oparta na uczeniu nienadzorowanym nie potrafi sama oznakować tych grup jako „krzesła" lub „psy", ale gdy już podobne obrazy są pogrupowane razem, ludzie mają dużo prostsze zadanie przy oznakowywaniu ich etykietami. Zamiast ręcznego oznakowywania milionów obrazów, ludzie mogą ręcznie oznakować wszystkie wyróżnione grupy, a te oznakowania zostaną zastosowane wobec wszystkich elementów wewnątrz danej grupy.

Po wstępnym szkoleniu, jeśli sztuczna inteligencja oparta na uczeniu nienadzorowanym znajdzie obrazy, które nie należą do żadnej z oznakowanych grup, sztuczna inteligencja utworzy osobne grupy dla niesklasyfikowanych dotąd obrazów, dając człowiekowi możliwość oznakowania nowych, nieoznakowanych dotychczas grup obrazów.

Uczenie nienadzorowane sprawia, że niemożliwe wcześniej do rozwiązania problemy stają się bardziej rozwiązywalne i jest dużo bardziej elastyczne w znajdowaniu ukrytych wzorców zarówno w danych historycznych, które są dostępne podczas szkolenia, jak i w przyszłych danych. Co więcej, mamy teraz metodę sztucznej inteligencji dla ogromnych zbiorów nieoznakowanych danych, które istnieją na świecie.

Choć uczenie nienadzorowane jest mniej biegłe od uczenia nadzorowanego w rozwiązywaniu konkretnych, wąsko zdefiniowanych problemów, to lepiej radzi sobie z bardziej otwartymi problemami silnej sztucznej inteligencji i uogólnianiem tej wiedzy.

Równie ważne jest to, że uczenie nienadzorowane może rozwiązywać wiele typowych problemów napotykanych przez analityków danych podczas budowania rozwiązań wykorzystujących uczenie maszynowe.

Używanie uczenia nienadzorowanego do poprawy rozwiązań wykorzystujących uczenie maszynowe

Ostatnie sukcesy w dziedzinie uczenia maszynowego były spowodowane dostępnością dużych ilości danych, postępami w zakresie sprzętu komputerowego i zasobów chmurowych oraz przełomowymi osiągnięciami w algorytmach uczenia maszynowego. Sukcesy te dotyczyły głównie problemów wąskiej sztucznej inteligencji, takich jak klasyfikacja obrazów, widzenie komputerowe, rozpoznawanie mowy, przetwarzanie języka naturalnego i tłumaczenie maszynowe.

Aby rozwiązywać bardziej ambitne problemy sztucznej inteligencji, musimy odblokować wartość uczenia nienadzorowanego. Przyjrzyjmy się najczęstszym wyzwaniom, przed którymi stają analitycy danych przy budowaniu rozwiązań i jak może w tym pomóc uczenie nienadzorowane.

Niewystarczająco oznakowane dane

> Myślę, że sztuczna inteligencja jest zbliżona do budowania statku rakietowego. Potrzebujemy ogromnego silnika i dużej ilości paliwa. Jeśli będziemy mieli wielki silnik i niewielką ilość paliwa, nie dostaniemy się na orbitę. Jeśli będziemy mieli mały silnik i mnóstwo paliwa, nie będziemy mogli nawet wystartować. Do zbudowania rakiety potrzebujemy ogromnego silnika i mnóstwo paliwa.
>
> *Andrew Ng*

Gdyby uczenie maszynowe porównać do statku rakietowego, dane byłyby paliwem – bez dużej ilości danych statek rakietowy nie poleci. Nie wszystkie dane są sobie równe. Aby korzystać z algorytmów nadzorowanych, potrzebujemy dużej ilości oznakowanych danych, których wygenerowanie jest trudne i kosztowne[1].

W przypadku uczenia nienadzorowanego możemy automatycznie oznakowywać nieoznakowane przykłady. Działałoby to następująco: rozdzielalibyśmy wszystkie przykłady na grupy, a następnie stosowali oznakowania z przykładów oznakowanych do nieoznakowanych przypadków w tej samej grupie (klastrze). Nieoznakowane przykłady otrzymywałyby oznakowanie z przykładów oznakowanych, do których są najbardziej podobne. Analizą skupień (dzieleniem na takie grupy) zajmiemy się w rozdziale 5.

Nadmierne dopasowanie

Jeśli algorytm uczenia maszynowego nauczy się zbyt złożonej funkcji w oparciu o dane szkoleniowe, może działać bardzo słabo na nigdy wcześniej niewidzianych przykładach z zestawów odłożonych, takich jak zestaw weryfikacyjny lub zestaw testowy. W tym

1 Istnieją firmy, takie jak Figure Eight, które świadczą takie usługi znakowania przez ludzi dużych ilości danych.

przypadku algorytm nadmiernie dopasowuje dane szkoleniowe – wyciągając zbyt wiele z szumu obecnego w danych – co daje bardzo duży błąd uogólnienia. Innymi słowy, algorytm zapamiętuje dane szkoleniowe, zamiast uczyć się, jak uogólniać zdobytą na ich podstawie wiedzę[2].

Aby rozwiązać ten problem, możemy wprowadzić uczenie nienadzorowane jako *regularyzator*. *Regularyzacja* jest procesem stosowanym w celu zmniejszenia złożoności algorytmu uczenia maszynowego, pomagając mu przechwytywać prawdziwe informacje w danych bez dostosowywania się zbytnio do szumu. Wstępne szkolenie nienadzorowane jest jedną z takich form regularyzacji. Zamiast podawania pierwotnych danych wejściowych bezpośrednio do algorytmu uczenia nadzorowanego, możemy podawać nowe przedstawienie pierwotnych danych wejściowych, które wygenerujemy.

To nowe przedstawienie oddaje istotę oryginalnych danych – prawdziwą wewnętrzną strukturę – tracąc po drodze część mniej reprezentatywnego szumu. Gdy podamy to nowe przedstawienie algorytmowi uczenia nadzorowanego, otrzyma on mniej szumu, przez który musiałby przebrnąć i przechwyci więcej sygnału (istotnych danych), poprawiając swój błąd uogólnienia. Zajmiemy się wyodrębnianiem cech w rozdziale 7.

Przekleństwo wymiarowości

Nawet przy obecnych postępach w zakresie mocy obliczeniowej, algorytmom uczenia maszynowego trudno jest obsługiwać ogromne zbiory danych. Ogólnie rzecz biorąc, dodawanie dalszych elementów nie jest zbyt problematyczne, ponieważ możemy zrównoleglić wykonywane operacje, wykorzystując nowoczesne rozwiązania typu map-reduce, takie jak Spark. Im więcej jednak mamy cech, tym szkolenie staje się trudniejsze.

W przestrzeni o bardzo dużej liczbie wymiarów algorytmy nadzorowane muszą nauczyć się, jak oddzielać punkty i budować aproksymację funkcji, aby podejmować dobre decyzje. Gdy cechy są bardzo liczne, to wyszukiwanie staje się bardzo kosztowne zarówno z punktu widzenia czasu, jak i obliczeń. W niektórych przypadkach znalezienie dobrego rozwiązania wystarczająco szybko może być niemożliwe.

Ten problem jest znany jako *przekleństwo wymiarowości*, a uczenie nienadzorowane jest dobrze przystosowane do radzenia sobie z nim. Dzięki redukcji wymiarowości możemy znaleźć najbardziej istotne cechy w oryginalnym zestawie cech, zmniejszyć liczbę wymiarów do łatwiejszej do ogarnięcia, tracąc przy tym bardzo mało istotnych informacji, a następnie zastosować algorytmy nadzorowane, aby skuteczniej przeprowadzać wyszukiwanie dobrej aproksymacji funkcji. Redukcję wymiarowości będziemy omawiać w rozdziale 3.

2 Zbyt małe dopasowanie jest innym problemem, który może wystąpić przy budowaniu aplikacji uczenia maszynowego, ale jest łatwiejsze do rozwiązania. Zbyt małe dopasowanie występuje, ponieważ model jest zbyt prosty – algorytm nie może zbudować wystarczająco złożonego przybliżenia funkcji, aby podejmować dobre decyzje związane z danym zadaniem. Aby to rozwiązać, możemy zwiększyć rozmiary algorytmu (dodać więcej parametrów, przeprowadzić więcej iteracji szkolenia, itd.) albo zastosować bardziej skomplikowany algorytm uczenia maszynowego.

Konstruowanie cech

Konstruowanie cech jest jednym z najważniejszych zadań wykonywanych przez analityków danych. Bez odpowiednich cech algorytm uczenia maszynowego nie będzie w stanie rozdzielić punktów w przestrzeni wystarczająco dobrze, aby podejmować dobre decyzje na nigdy wcześniej niewidzianych przykładach. Jednakże konstruowanie cech jest zwykle bardzo pracochłonne; wymaga udziału ludzi, którzy kreatywnie zaprojektują ręcznie odpowiednie typy cech. Zamiast tego możemy wykorzystać uczenie reprezentacji z algorytmów uczenia nienadzorowanego, aby automatycznie nauczyć się odpowiednich typów reprezentacji cech i pomóc sobie w rozwiązaniu tego zadania. Zajmiemy się automatycznym wyodrębnianiem cech w rozdziale 7.

Elementy odstające

Jakość danych jest również bardzo ważna. Jeśli algorytmy uczenia maszynowego zostaną przeszkolone na rzadkich elementach odstających od reszty, które mogą zniekształcać ogólne dane, to ich błędy uogólnienia będą mniejsze, niż gdyby ignorowały takie elementy lub zajmowały się nimi oddzielnie. W przypadku uczenia nienadzorowanego możemy przeprowadzać wykrywanie elementów odstających przy użyciu redukcji wymiarowości i tworzyć rozwiązanie specjalne dla elementów odstających i oddzielne rozwiązanie dla normalnych danych. Zbudujemy system wykrywania anomalii w rozdziale 4.

Odchylenie danych

Modele uczenia maszynowego muszą też zwracać uwagę na odchylenie danych. Jeśli dane, na bazie których model tworzy prognozy, różnią się statystycznie od danych, na których model został przeszkolony, to konieczne może być kolejne przeszkolenie modelu na danych bardziej reprezentatywnych dla danych bieżących. Jeśli model nie zostanie ponownie przeszkolony albo nie rozpozna odchylenia danych, ucierpi na tym jakość przewidywania na bieżących danych.

Poprzez budowanie rozkładów prawdopodobieństwa przy użyciu uczenia nienadzorowanego możemy ocenić, jak różne są bieżące dane od danych w zestawie szkoleniowym – jeśli ta różnica jest wystarczająco duża, możemy automatycznie wywoływać ponowne przeszkolenie. Ocenianiem danych pod tym kątem zajmiemy się w rozdziale 12.

Bliższe spojrzenie na algorytmy nadzorowane

Zanim zagłębimy się w systemy uczenia nienadzorowanego, przyjrzymy się algorytmom uczenia nadzorowanego i jak one działają. Pomoże nam to ustalić miejsce uczenia nienadzorowanego w ekosystemie uczenia maszynowego.

W uczeniu nadzorowanym istnieją dwa główne typy problemów: *klasyfikacja* i *regresja*. W przypadku klasyfikacji sztuczna inteligencja musi poprawnie zaklasyfikować elementy do jednej z dwóch lub więcej klas. Jeśli istnieją tylko dwie klasy, problem taki jest

nazywany *klasyfikacją binarną*. Jeśli istnieją trzy lub więcej klasy, problem jest nazywany *klasyfikacją wieloklasową*.

Problemy klasyfikacyjne znane są też jako problemy *dyskretnego* przewidywania, ponieważ każda klasa stanowi dyskretną grupę. Problemy klasyfikacyjne mogą być też określane jako problemy *jakościowe* lub *kategoryczne*.

W przypadku regresji sztuczna inteligencja musi przewidywać zmienną *ciągłą* a nie dyskretną. Problemy związane z regresją mogą być też określane jako problemy *ilościowe*.

Uczenie nadzorowane obejmuje całą gamę algorytmów, od bardzo prostych do bardzo złożonych, ale wszystkie one mają na celu minimalizowanie jakiejś funkcji kosztów albo poziomu błędu (albo maksymalizowanie funkcji wartości) związanych z oznakowaniami, które mamy dla zestawu danych.

Jak wspominano wcześniej, najbardziej zależy nam na tym, jak dobrze dane rozwiązanie uczenia maszynowego uogólnia się do nigdy wcześniej niewidzianych przypadków. Wybór algorytmu uczenia nadzorowanego jest bardzo ważny przy minimalizowaniu tego błędu uogólnienia.

Aby uzyskać możliwie najmniejszy błąd uogólnienia, złożoność modelu algorytmicznego powinna odpowiadać złożoności prawdziwej funkcji leżącej u podstaw danych. Nie wiemy, jaka faktycznie jest ta prawdziwa funkcja. Gdybyśmy to wiedzieli, nie musielibyśmy korzystać z uczenia maszynowego do tworzenia modelu – po prostu użylibyśmy tej funkcji do znalezienia prawidłowej odpowiedzi. Ponieważ jednak nie wiemy, jaka jest ta prawdziwa funkcja, wybieramy algorytm uczenia maszynowego, aby przetestować hipotezy i znaleźć model, który najlepiej przybliża tę prawdziwą funkcję (tzn. ma najniższy możliwy błąd uogólnienia).

Jeśli to, co algorytm modeluje, jest mniej skomplikowane niż prawdziwa funkcja, mamy *zbyt małe dopasowanie* danych. W takim przypadku moglibyśmy poprawić błąd uogólnienia przez wybranie algorytmu, który może modelować bardziej złożoną funkcję. Jeśli jednak algorytm tworzy zbyt skomplikowany model, to *nadmiernie dopasowaliśmy* dane szkoleniowe i będziemy mieli słabą wydajność dla nigdy wcześniej niewidzianych przypadków, zwiększając nasz błąd uogólnienia.

Innymi słowy, wybranie bardziej złożonych algorytmów zamiast prostszych nie zawsze jest właściwym wyborem – czasami prostsze jest lepsze. Każdy algorytm ma swój własny zestaw mocnych i słabych stron oraz założeń, a wiedza o tym, czego użyć w przypadku określonych danych i problemu, który próbujemy rozwiązać, jest bardzo ważna dla opanowania uczenia maszynowego.

W pozostałej części tego rozdziału opiszemy kilka najczęstszych algorytmów nadzorowanych (w tym pewne rzeczywiste aplikacje), zanim zrobimy to samo dla algorytmów nienadzorowanych[3].

3 Ta lista nie jest wyczerpująca, ale zawiera najczęściej używane algorytmy uczenia maszynowego.

Metody liniowe

Najbardziej podstawowe algorytmy uczenia nadzorowanego modelują prostą zależność liniową pomiędzy cechami wejściowymi a zmienną wyjściową, którą chcemy przewidywać.

Regresja liniowa

Najprostszym ze wszystkich algorytmów jest *regresja liniowa*, która wykorzystuje model zakładający liniową relację pomiędzy zmiennymi wejściowymi (x), a pojedynczą zmienną wyjściową (y). Jeśli prawdziwa relacja pomiędzy wejściami a wyjściem jest liniowa, a zmienne wejściowe nie są wysoce skorelowane (sytuacja znana jako *kolinearność*), regresja liniowa może być odpowiednim wyborem. Jeśli prawdziwa relacja jest bardziej złożona lub nieliniowa, regresja liniowa będzie zbyt mało dopasowywać dane[4].

Ponieważ jest to tak proste, interpretowanie relacji modelowanej przez ten algorytm jest również bardzo proste. *Łatwość interpretacji* jest bardzo ważnym czynnikiem dla stosowanego uczenia maszynowego, ponieważ rozwiązania muszą być rozumiane i wdrażane przez techniczne i nietechniczne osoby w danej branży. Bez możliwości łatwej interpretacji rozwiązania stają się niezgłębionymi czarnymi skrzynkami.

Mocne strony
: Regresja liniowa jest prosta, łatwa do interpretowania i trudna do nadmiernego dopasowania, ponieważ nie może modelować nadmiernie złożonych relacji. Jest doskonałym wyborem, gdy podstawowa relacja pomiędzy zmiennymi wejściowymi i wyjściowymi jest liniowa.

Słabe strony
: Regresja liniowa będzie zbyt mało dopasowywać dane, gdy relacja pomiędzy zmiennymi wejściowymi i wyjściowymi jest nieliniowa.

Zastosowania
: Ponieważ rzeczywista podstawowa zależność pomiędzy wagą i wzrostem człowieka jest liniowa, regresja liniowa doskonale nadaje się do przewidywania wagi na podstawie wzrostu jako zmiennej wejściowej lub odwrotnie do przewidywania wzrostu na podstawie wagi jako zmiennej wejściowej.

Regresja logistyczna

Najprostszym algorytmem klasyfikacji jest *regresja logistyczna*, która też jest metodą liniową, ale prognozy są przekształcane przy użyciu funkcji logistycznej. Danymi wyjściowymi dla tej transformacji są *prawdopodobieństwa klas* – innymi słowy, prawdopodobieństwa, że dany przypadek należy do różnych klas, gdzie suma prawdopodobieństw dla każdego

4 Mogą istnieć inne potencjalne problemy, które mogą sprawiać, że regresja liniowa będzie słabym wyborem, w tym elementy odstające, korelacja błędów i niestała wariancja błędów.

przypadku sumuje się do jedności. Każdy przypadek jest następnie przypisywany do klasy, dla której ma on największe prawdopodobieństwo przynależności.

Mocne strony
: Podobnie do regresji liniowej, regresja logistyczna jest prosta i łatwa do interpretowania. Gdy klasy, które próbujemy przewidzieć, nie nachodzą na siebie i są liniowo rozdzielne, regresja logistyczna jest doskonałym wyborem.

Słabe strony
: Gdy klasy nie są liniowo rozdzielne, regresja logistyczna zakończy się niepowodzeniem.

Zastosowania
: Gdy klasy w większości nie nachodzą na siebie – na przykład wzrost małych dzieci w porównaniu do wzrostu dorosłych – regresja logistyczna będzie dobrze działać.

Metody oparte na sąsiedztwie

Inną grupą bardzo prostych algorytmów są metody oparte na sąsiedztwie. Metody oparte na sąsiedztwie są *leniwymi uczniami*, ponieważ uczą się, jak oznakowywać nowe punkty w oparciu o bliskość nowych punktów do istniejących oznakowanych punktów. W przeciwieństwie do regresji liniowej lub regresji logistycznej, modele oparte na sąsiedztwie nie uczą się określonego modelu przewidywania oznakowań dla nowych punktów; modele te raczej przewidują oznakowania dla nowych punktów w oparciu wyłącznie o odległość nowych punktów do istniejących wcześniej oznakowanych punktów. Leniwe uczenie jest też nazywane *uczeniem opartym na przykładach* albo *metodami nieparametrycznymi*.

K najbliższych sąsiadów

Najczęstszą metodą opartą na sąsiedztwie jest metoda *k najbliższych sąsiadów* (*KNN – k-nearest neighbours*). Aby oznakować każdy nowy punkt, algorytm KNN patrzy na liczbę *k* (gdzie *k* jest wartością całkowitą) najbliższych oznakowanych punktów i wykorzystuje je do głosowania nad tym, jak oznakować nowy punkt. Domyślnie KNN wykorzystuje odległość euklidesową do mierzenia tego, co jest najbliżej.

Wybór liczby *k* jest bardzo ważny. Jeśli *k* zostanie ustawione na bardzo małą wartość, algorytm stanie się bardzo elastyczny, tworząc wysoce zniuansowane granice i potencjalnie nadmiernie dopasowując dane. Jeśli *k* zostanie ustawione na bardzo dużą wartość, algorytm KNN stanie się nieelastyczny, tworząc bardzo sztywną granicę i potencjalnie zbyt mało dopasowując dane.

Mocne strony
: W przeciwieństwie do metod liniowych, algorytm KNN jest bardzo elastyczny i biegły w poznawaniu bardziej złożonych, nieliniowych relacji. Algorytm KNN pozostaje przy tym prosty i łatwy do interpretowania.

Słabe strony

KNN słabo sobie radzi, gdy liczba obserwacji i cech rośnie. Algorytm KNN staje się nieefektywny pod względem obliczeniowym w takiej zatłoczonej, wysokowymiarowej przestrzeni, ponieważ musi obliczać odległości od nowego punktu do wielu pobliskich oznakowanych punktów, aby przewidywać oznakowania. Nie może polegać na wydajnym modelu z ograniczoną liczbą parametrów, aby dokonać koniecznych prognoz. Algorytm KNN jest też bardzo czuły na wybór liczby *k*. Gdy *k* będzie zbyt niskie, KNN może nadmiernie dopasowywać dane, a gdy *k* będzie zbyt wysokie, KNN może zbyt mało dopasowywać dane.

Zastosowania

Algorytm KNN jest często używany w systemach polecających, takich jak te do przewidywania zainteresowania filmami (Netflix), muzyką (Spotify), znajomymi (Facebook), zdjęciami (Instagram), wyszukiwaniami (Google) i zakupami (Amazon). Na przykład KNN może przewidywać, co będzie się podobać użytkownikowi na podstawie tego, co lubią podobni użytkownicy (znane jako *filtrowanie kolektywne*) albo co użytkownik lubił w przeszłości (znane jako *filtrowanie oparte na zawartości*).

Metody oparte na drzewach

Zamiast korzystać z metody liniowej, możemy kazać sztucznej inteligencji zbudować *drzewo decyzyjne*, w którym wszystkie przypadki zostaną *rozdzielone* lub *rozwarstwione* na wiele regionów w oparciu o posiadane oznakowania. Po zakończeniu tego podziału każdy region odpowiada określonej klasie oznakowania (dla problemów klasyfikacyjnych) lub zakresowi przewidywanych wartości (dla problemów związanych z regresją). Ten proces jest podobny do automatycznego budowania zasad przez sztuczną inteligencję z wyraźnym celem podejmowania lepszych decyzji lub przewidywań.

Pojedyncze drzewo decyzyjne

Najprostszą metodą opartą na drzewie jest *pojedyncze drzewo decyzyjne*, w którym sztuczna inteligencja przechodzi raz przez dane szkoleniowe, tworzy zasady podziału danych w oparciu o oznakowania i wykorzystuje to drzewo do przewidywania dla danych z nigdy wcześniej niewidzianych zestawów weryfikacyjnych lub testowych. Jednakże pojedyncze drzewo decyzyjne zwykle słabo radzi sobie z uogólnianiem tego, czego nauczyło się podczas szkolenia, do nigdy wcześniej niewidzianych przypadków, ponieważ zwykle nadmiernie dopasowuje dane szkoleniowe podczas swojej jedynej iteracji szkolenia.

Grupowanie typu bootstrap

Aby poprawić pojedyncze drzewo decyzyjne, możemy wprowadzić *grupowanie typu bootstrap* (*bootstrap aggregating* albo w skrócie *bagging*), w którym bierzemy *wiele losowych próbek wystąpień* z danych szkoleniowych, tworzymy drzewo decyzyjne dla każdej próbki, a następnie przewidujemy dane wyjściowe dla każdego wystąpienia, uśredniając prognozy

każdego z tych drzew. Wykorzystując *losowy dobór* próbek i uśrednianie wyników z wielu drzew – podejście, które jest też znane jako *metoda zespołowa* (*ensemble method*) – grupowanie typu bootstrap będzie radzić sobie z nadmiernym dopasowywaniem danych wynikającym z pojedynczego drzewa decyzyjnego.

Losowe lasy

Możemy jeszcze bardziej poprawić nadmierne dopasowanie danych przez próbkowanie nie tylko wystąpień, ale też prognostyk. W przypadku *losowych lasów* bierzemy wiele losowych próbek wystąpień z danych szkoleniowych, tak jak w przypadku grupowania typu bootstrap, ale przy każdym rozwidleniu w drzewie decyzyjnym dokonujemy wyboru na podstawie nie wszystkich prognostyk, ale raczej na podstawie *losowej próbki prognostyk*. Liczba prognostyk, które bierzemy pod uwagę dla każdego rozwidlenia, jest zwykle pierwiastkiem kwadratowym całkowitej liczby prognostyk.

Pobierając próbki prognostyk w ten sposób, algorytm losowych lasów tworzy drzewa, które są jeszcze mniej skorelowane ze sobą (w porównaniu z drzewami w przypadku grupowania typu bootstrap), zmniejszając nadmierne dopasowanie danych i poprawiając błąd uogólnienia.

Boosting

Inne podejście, znane jako *boosting*, jest używane do tworzenia wielu drzew, podobnie jak w przypadku grupowania typu bootstrap, ale do *budowania drzew sekwencyjnie*, wykorzystując to, co sztuczna inteligencja dowiedziała się z poprzedniego drzewa, żeby poprawić wyniki przy kolejnym drzewie. Każde drzewo jest dosyć płytkie z zaledwie kilkoma rozwidleniami, a uczenie odbywa się powoli, drzewo po drzewie. Spośród wszystkich metod opartych na drzewach *automaty wzmacniania gradientowego* (*GBMs – gradient boosting machines*) należą do najlepszych i są często używane przez zwycięzców konkursów z dziedziny uczenia maszynowego[5].

Mocne strony
: Metody oparte na drzewach należą do najskuteczniejszych algorytmów uczenia nadzorowanego w przypadku problemów związanych z przewidywaniem. Metody te są w stanie uchwycić złożone relacje w danych, ucząc się wielu prostych zasad, po jednej na raz. Mogą też obsługiwać brakujące dane i cechy kategoryczne.

Słabe strony
: Metody oparte na drzewach są trudne do interpretowania, zwłaszcza jeśli potrzeba wielu zasad do dobrego przewidywania. Wydajność również staje się problemem, gdy liczba cech wzrasta.

5 Więcej informacji na temat stosowania wzmocnienia gradientowego w konkursach z dziedziny uczenia maszynowego można znaleźć w blogu Bena Gormana (*http://bit.ly/2S1C8Qy*).

Zastosowania

Wzmacnianie gradientowe i losowe lasy świetnie nadają się w przypadku problemów związanych z przewidywaniem.

Maszyny wektorów nośnych

Zamiast budować drzewa do rozdzielania danych, możemy wykorzystywać algorytmy do tworzenia hiperpłaszczyzn w przestrzeni do oddzielania danych, kierując się posiadanymi oznakowaniami. To podejście jest znane jako *maszyny wektorów nośnych* (*SVMs – suport vector machines*). Maszyny SVM pozwalają na pewne naruszenia tego rozdzielania – nie wszystkie punkty w obszarze hiperprzestrzeni muszą mieć takie samo oznakowanie – ale odległość pomiędzy punktami definiującymi granice określonego oznakowania a punktami definiującymi granice innego oznakowania powinny być możliwie maksymalizowane. Granice nie muszą też być liniowe – możemy wykorzystywać nieliniowe elementy do bardziej elastycznego rozdzielania danych.

Sieci neuronowe

Możemy poznawać reprezentacje danych, korzystając z sieci neuronowych, które składają się z warstwy wejściowej, kilku warstw ukrytych i warstwy wyjściowej[6]. Warstwa wejściowa wykorzystuje cechy, a warstwa wyjściowa stara się dopasować zmienną odpowiedzi. Warstwy ukryte stanowią zagnieżdżoną hierarchię konceptów – każda warstwa (albo koncept) próbuje zrozumieć, w jaki sposób poprzednia warstwa odnosi się do warstwy wyjściowej.

Wykorzystując tę hierarchię konceptów, sieć neuronowa jest w stanie uczyć się skomplikowanych konceptów, budując je z kilku prostszych. Sieci neuronowe stanowią jedno z najpotężniejszych podejść do aproksymacji funkcji, ale są podatne na nadmierne dopasowywanie danych i są trudne do interpretowania – te braki będziemy badać bardziej szczegółowo w dalszej części książki.

Bliższe spojrzenie na algorytmy nienadzorowane

Teraz zwrócimy swoją uwagę na problemy, w których nie mamy oznakowań. Zamiast próbować prognozować, algorytmy uczenia nienadzorowanego próbują uczyć się podstawowej struktury danych.

6 Więcej na temat sieci neuronowych można dowiedzieć się z książki *Deep Learning* (*http://www.deeplearningbook.org/*), Ian Goodfellow, Yoshua Bengio i Aaron Courville (MIT Press).

Redukcja wymiarowości

Rodzina algorytmów znanych jako *algorytmy redukcji wymiarowości* rzutuje pierwotne, wielowymiarowe dane wejściowe do przestrzeni o mniejszej liczbie wymiarów, odfiltrowując mniej istotne cechy i zachowując jak najwięcej interesujących. Redukcja wymiarowości pozwala sztucznej inteligencji opartej na uczeniu nienadzorowanym skuteczniej identyfikować wzorce i rozwiązywać problemy o dużej skali i kosztowne obliczeniowo (często związane z obrazami, wideo, mową i tekstem).

Rzutowanie liniowe

Istnieją dwie główne gałęzie redukcji wymiarowości – rzutowanie liniowe i nieliniowa redukcja wymiarowości. Zaczniemy od rzutowania liniowego.

Analiza głównych składowych Jednym z podejść do poznania wewnętrznej struktury danych jest określenie, które cechy z pełnego zestawu cech są najważniejsze w wyjaśnianiu zmienności pomiędzy wystąpieniami danych. Nie wszystkie cechy są sobie równe – w przypadku niektórych cech wartości w zestawie danych nie różnią się znacznie i cechy te są mniej przydatne w analizie zestawu danych. W przypadku innych cech wartości mogą się znacznie różnić – te cechy warto zbadać bardziej szczegółowo, ponieważ będą one lepiej pomagać projektowanemu modelowi rozdzielać dane.

W *analizie głównych składowych* (*PCA – principal component analysis*) algorytm znajduje niskowymiarowe przedstawienie danych, zachowując jak największe zróżnicowanie. Liczba pozostałych wymiarów jest znacznie mniejsza niż liczba wymiarów w pełnym zestawie (tzn. całkowita liczba cech). Tracimy część zróżnicowania, przechodząc do tej niskowymiarowej przestrzeni, ale łatwiej jest zidentyfikować podstawową strukturę danych, co pozwala nam skuteczniej przeprowadzać zadania, takie jak analiza skupień.

Istnieje kilka wariantów algorytmu PCA, które zbadamy w dalszej części książki. Obejmują one takie warianty, jak *przyrostowa analiza PCA* (*incremental PCA*), warianty nieliniowe, takie jak *rdzeniowa analiza PCA* (*kernel PCA*) i warianty rozrzedzone, takie jak *rzadka analiza PCA* (*sparse PCA*).

Rozkład według wartości osobliwych Innym podejściem do poznania podstawowej struktury danych jest zmniejszenie rzędu oryginalnej macierzy cech do mniejszego rzędu, tak aby oryginalna macierz mogła zostać odtworzona przy użyciu liniowej kombinacji pewnych wektorów w macierzy mniejszego rzędu. Jest to znane jako *rozkład według wartości osobliwych* (*SVD – singular value decomposition*). Aby wygenerować macierz mniejszego rzędu, rozkład SVD zachowuje wektory oryginalnej macierzy, które zawierają najwięcej informacji (tzn. mają największą wartość osobliwą). Macierz mniejszego rzędu zachowuje najważniejsze elementy oryginalnej przestrzeni cech.

Losowe rzutowanie Podobny algorytm redukcji wymiarowości polega na rzutowaniu punktów z wysokowymiarowej przestrzeni do przestrzeni o znacznie mniejszej liczbie wymiarów w taki sposób, żeby zachowana została skala odległości między punktami.

Aby to osiągnąć, możemy skorzystać albo z *losowej macierzy Gaussa*, albo z *losowej macierzy rzadkiej*.

Uczenie rozmaitościowe

Zarówno analiza PCA, jak i losowe rzutowanie opierają się na rzutowaniu danych liniowo z przestrzeni o dużej liczbie wymiarów do przestrzeni o mniejszej liczbie wymiarów. Zamiast rzutowania liniowego, być może lepiej przeprowadzić nieliniowe przekształcenie danych – jest to znane jako *uczenie rozmaitościowe* (*manifold learning*) albo *nieliniowa redukcja wymiarowości* (*nonlinear dimensionality reduction*).

Isomap *Mapowanie izometryczne* (*Isomap*) jest jednym z typów algorytmów uczenia rozmaitościowego. Algorytm ten poznaje wewnętrzną geometrię danych, szacując *odległość geodezyjną* albo *zakrzywioną* pomiędzy każdym punktem i jego sąsiadami, zamiast odległości euklidesowej. Isomap wykorzystuje to, aby następnie osadzić oryginalną wysokowymiarową przestrzeń w przestrzeni niskowymiarowej.

Stochastyczne osadzanie sąsiadów z t-rozkładem (t-SNE) Inny algorytm nieliniowej redukcji wymiarowości – znany jako *stochastyczne osadzanie sąsiadów z t-rozkładem* (*t-SNE – t-distributed stochastic neighbor embedding*) – osadza wysokowymiarowe dane w przestrzeni o jedynie dwóch lub trzech wymiarach, pozwalając na wizualizację przekształconych danych. W tej przestrzeni dwu- lub trójwymiarowej podobne wystąpienia są modelowane bliżej siebie, a niepodobne wystąpienia są modelowane dalej od siebie.

Uczenie słownikowe Podejście znane jako *uczenie słownikowe* polega na poznawaniu rzadkiej reprezentacji bazowych danych. Te elementy reprezentatywne są prostymi, binarnymi wektorami (zerami i jedynkami), a każde wystąpienie w zestawie danych może być zrekonstruowane jako suma ważona elementów reprezentatywnych. Macierz (znana jako *słownik*), którą generuje ten algorytm uczenia nienadzorowanego, jest głównie wypełniona zerami z niewielką liczbą niezerowych wag.

Tworząc taki słownik, algorytm ten jest w stanie skutecznie identyfikować najbardziej istotne elementy reprezentatywne oryginalnej przestrzeni cech – są to te, które mają najwięcej niezerowych wag. Elementy reprezentatywne, które są mniej ważne, będą miały mniej niezerowych wag. Podobnie jak w przypadku analizy PCA, uczenie słownikowe doskonale nadaje się do poznawania podstawowej struktury danych, co będzie pomocne w rozdzielaniu danych oraz identyfikowaniu interesujących wzorców.

Analiza niezależnych składowych

Jednym z typowych problemów z nieoznakowanymi danymi jest to, że istnieje wiele niezależnych sygnałów osadzonych razem w cechach, które otrzymujemy. Korzystając z *analizy niezależnych składowych* (*ICA – independent component analysis*), możemy rozdzielić te zmieszane sygnały na poszczególne składowe. Po zakończeniu rozdzielenia możemy zrekonstruować dowolną z oryginalnych cech, sumując jakąś kombinację poszczególnych składników, które wygenerowaliśmy. Analiza ICA jest często wykorzystywana w zadaniach

przetwarzania sygnałów (na przykład w celu zidentyfikowania poszczególnych głosów w nagraniu audio z zatłoczonej kawiarni).

Utajona alokacja Dirichleta

Uczenie nienadzorowane może też objaśniać zestaw danych, ucząc się, dlaczego pewne części zestawu danych są podobne do siebie. Wynika to z poznania nieobserwowanych elementów w zestawie danych – podejście to jest znane jako *utajona alokacja Dirichleta* (*LDA – latent Dirichlet allocation*). Rozważmy na przykład dokument tekstowy zawierający wiele słów. Te słowa w dokumencie nie są czysto losowe; raczej wykazują jakąś strukturę.

Ta struktura może być modelowana przez nieobserwowane elementy znane jako tematy. Po szkoleniu algorytm LDA jest w stanie objaśnić dany dokument przy pomocy małego zestawu tematów, gdzie dla każdego tematu istnieje niewielki zestaw często używanych słów. Jest to ukryta struktura, którą algorytm LDA jest w stanie uchwycić, pomagając nam lepiej objaśnić wcześniej nieustrukturyzowany zasób tekstowy.

Redukcja wymiarowości zmniejsza oryginalny zestaw cech do mniejszego zestawu jedynie najważniejszych cech. Od teraz możemy uruchamiać inne algorytmy uczenia nienadzorowanego na tym mniejszym zestawie cech, aby znajdować interesujące wzorce w danych (zobacz następny podrozdział na temat analizy skupień) albo (jeśli mamy oznakowania) możemy przyspieszyć cykl szkoleniowy algorytmów uczenia nadzorowanego dostarczając tę mniejszą macierz cech, zamiast korzystać z oryginalnej macierzy cech.

Analiza skupień

Po zredukowaniu zestawu oryginalnych cech do mniejszego, łatwiejszego w zarządzaniu zestawu, możemy znaleźć interesujące wzorce, grupując ze sobą podobne wystąpienia danych. Jest to zwane grupowaniem lub analizą skupień i może być przeprowadzone przy użyciu różnych algorytmów uczenia nienadzorowanego i wykorzystywane w rzeczywistych zastosowaniach, takich jak segmentacja rynku.

k-średnich

Aby dobrze grupować dane, musimy zidentyfikować oddzielne grupy, tak aby wystąpienia w danej grupie były podobne do siebie, ale różne od wystąpień w innych grupach. Jednym z takich algorytmów jest *grupowanie przez k-średnich* (*k-means clustering*). Za pomocą tego algorytmu określamy liczbę żądanych grup (klastrów) *k*, a algorytm przypisze każde wystąpienie dokładnie do jednej z tych *k* grup. Optymalizuje to grupowanie, minimalizując *zmienność wewnątrz grupy* (nazywaną również *bezwładnością*), tak aby suma zmienności wewnątrz grupy we wszystkich *k* grupach była jak najmniejsza.

Aby przyspieszyć ten proces grupowania, *k-średnich* losowo przypisuje każdą obserwację do jednej z *k* grup, a następnie zaczyna ponowne przypisywanie tych obserwacji, żeby zminimalizować odległość euklidesową pomiędzy każdą obserwacją a punktem środkowym jej grupy, czyli *centroidem*. W rezultacie różne przebiegi algorytmu *k-średnich* – każdy z losowym początkiem – będą skutkować nieco innymi przypisaniami obserwacji do grup. Spośród tych różnych przebiegów możemy wybrać ten, który ma najlepszą separację, zdefiniowaną jako najniższa całkowita suma zmienności wewnątrz grup dla wszystkich *k* grup[7].

Grupowanie hierarchiczne

Alternatywne podejście do grupowania – które nie wymaga od nas wstępnego przywiązania się do określonej liczby grup – jest znane jako *grupowanie hierarchiczne* (*hierarchical clustering*). Jedna z wersji grupowania hierarchicznego o nazwie *grupowanie aglomeracyjne* (*agglomerative clustering*) wykorzystuje metodę grupowania opartą na drzewach i buduje tak zwany *dendrogram*. Dendrogram może być przedstawiony graficznie jako odwrócone drzewo, w którym liście są u dołu, a pień jest u góry.

Liście na samym dole są poszczególnymi wystąpieniami w zestawie danych. Grupowanie hierarchiczne następnie łączy liście razem – gdy przechodzimy w pionie w górę odwróconego drzewa – w oparciu o to, jak podobne są one do siebie. Wystąpienia (lub grupy wystąpień), które są najbardziej podobne do siebie, są łączone wcześniej, natomiast wystąpienia, które nie są tak podobne, są łączone później. W tym iteracyjnym procesie wszystkie wystąpienia są ostatecznie połączone razem, tworząc pojedynczy pień drzewa.

Taki pionowy obraz jest bardzo pomocny. Po zakończeniu pracy algorytmu grupowania hierarchicznego możemy przejrzeć dendrogram i określić, gdzie chcemy uciąć drzewo – im niżej dokonamy cięcia, tym więcej poszczególnych gałęzi dostaniemy (tzn. więcej grup). Jeśli chcemy mniej grup, możemy wykonać cięcie wyżej na dendrogramie, bliżej pojedynczego pnia u samej góry tego odwróconego drzewa. Umieszczenie tego cięcia jest podobne do wybrania liczby *k* grup w algorytmie *grupowania przez k-średnich*[8].

DBSCAN

Jeszcze bardziej zaawansowany algorytm grupowania (oparty na gęstości punktów) jest znany jako *DBSCAN* (*density-based spatial clustering of applications with noise – oparte na gęstości przestrzenne grupowanie aplikacji z szumem*). Biorąc pod uwagę wszystkie wystąpienia, jakie mamy w przestrzeni, DBSCAN zgrupuje te, które są ściśle upakowane ze sobą, gdzie bliskość jest zdefiniowana jako minimalna liczba wystąpień, które muszą

7 Istnieją szybsze warianty grupowania przez *k*-średnich, takie jak wersja *k*-średnich dla minimalnych partii, którą omówimy w dalszej części książki.

8 Grupowanie hierarchiczne wykorzystuje domyślnie odległość euklidesową, ale może też korzystać z innych miar, takich jak odległość oparta na korelacji, którą zbadamy dokładniej w dalszej części książki.