

• Douglas McIlwraith, Haralambos Marmanis, Dmitry Babenko •

# INTELIGENTNA SIEĆ

## Algorytmy przyszłości

WYDANIE II

Helion

Tytuł oryginału: Algorithms of the Intelligent Web, 2nd Edition

Tłumaczenie: Tomasz Walczak

Projekt okładki: Studio Gravite / Olsztyń; Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

ISBN: 978-83-283-3251-5

Original edition copyright © 2016 by Manning Publications Co. All rights reserved.

Polish edition copyright © 2017 by HELION SA. All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiekolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicielami.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION  
ul. Kościuszki 1c, 44-100 GLIWICE  
tel. 32 231 22 19, 32 230 98 63  
e-mail: [helion@helion.pl](mailto: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/intsi2\\_ebook](http://helion.pl/user/opinie/intsi2_ebook)  
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:  
<ftp://ftp.helion.pl/przyklady/intsi2.zip>

- [Poleć książkę na Facebook.com](#)
- [Kup w wersji papierowej](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

*Tę ksiązkę dedykuję Elly, moj petit pois*

— D.M.



# *Spis treści*

---

*Przedmowa* 9

*Wprowadzenie* 11

*Podziękowania* 13

*O książce* 15

## **Rozdział 1. Budowanie aplikacji na potrzeby inteligentnej sieci 19**

1.1. Inteligentny algorytm w akcji — Google Now 21

1.2. Cykl życia inteligentnych algorytmów 23

1.3. Inne przykłady inteligentnych algorytmów 24

1.4. Czym inteligentne aplikacje nie są 25

*1.4.1. Inteligentne algorytmy nie są myślącymi maszynami do uniwersalnych zastosowań* 25

*1.4.2. Inteligentne algorytmy nie zastąpią ludzi* 25

*1.4.3. Inteligentne algorytmy nie są odkrywane przez przypadek* 26

1.5. Klasy inteligentnych algorytmów 26

*1.5.1. Sztuczna inteligencja* 27

*1.5.2. Uczenie maszynowe* 28

*1.5.3. Analityka predykcyjna* 29

1.6. Ocena działania inteligentnych algorytmów 30

*1.6.1. Ocena inteligencji* 30

*1.6.2. Ocena predykcji* 31

1.7. Ważne uwagi na temat inteligentnych algorytmów 33

*1.7.1. Dane nie są wiarygodne* 34

*1.7.2. Wnioskowanie wymaga czasu* 34

*1.7.3. Wielkość ma znaczenie!* 34

*1.7.4. Różne algorytmy skalują się w odmienny sposób* 35

*1.7.5. Nie wszystko jest gwoździem!* 35

*1.7.6. Dane to nie wszystko* 35

*1.7.7. Czas treningu może się zmieniać* 36

*1.7.8. Celem jest generalizacja* 36

*1.7.9. Ludzka intuicja nie zawsze się sprawdza* 36

*1.7.10. Pomyśl o zaprojektowaniu nowych cech* 36

*1.7.11. Poznaj wiele różnych modeli* 36

*1.7.12. Korelacja nie oznacza związku przyczynowo-skutkowego* 37

1.8. Podsumowanie 37

## Rozdział 2. Wydobywanie struktury z danych — klastrowanie i transformacja danych 39

- 2.1. Dane, struktura, błąd systematyczny i szum 41
- 2.2. „Przekleństwo wymiarów” 44
- 2.3. Algorytm k-srednich 45
  - 2.3.1. K-srednie w praktyce 49
- 2.4. Gaussowski model mieszany 52
  - 2.4.1. Czym jest rozkład Gaussa? 52
  - 2.4.2. Maksymalizacja wartości oczekiwanej i rozkład Gaussa 55
  - 2.4.3. Gaussowski model mieszany 55
  - 2.4.4. Przykład uczenia z użyciem gaussowskiego modelu mieszaneego 57
- 2.5. Zależności między k-srednimi i algorytmem GMM 59
- 2.6. Transformacje osi danych 60
  - 2.6.1. Wektory własne i wartości własne 61
  - 2.6.2. Analiza głównych składowych 61
  - 2.6.3. Przykład zastosowania analizy głównych składowych 63
- 2.7. Podsumowanie 65

## Rozdział 3. Rekomendowanie odpowiednich treści 67

- 3.1. Wprowadzenie — internetowy sklep z filmami 68
- 3.2. Odległość i podobieństwo 69
  - 3.2.1. Więcej o odległości i podobieństwie 73
  - 3.2.2. Który wzór na podobieństwo jest najlepszy? 75
- 3.3. Jak działają systemy rekomendacji? 76
- 3.4. Filtrowanie kolaboratywne według użytkowników 77
- 3.5. Rekomendacje według modelu z wykorzystaniem rozkładu SVD 82
  - 3.5.1. Rozkład SVD 83
  - 3.5.2. Rekomendacje z użyciem rozkładu SVD — wybór filmów dla danego użytkownika 84
  - 3.5.3. Rekomendacje z wykorzystaniem rozkładu SVD — określanie użytkowników, których może zainteresować dany film 90
- 3.6. Konkurs Netflix Prize 93
- 3.7. Ocenianie systemu rekomendacji 94
- 3.8. Podsumowanie 96

## Rozdział 4. Klasyfikowanie — umieszczanie elementów tam, gdzie ich miejsce 97

- 4.1. Do czego potrzebna jest klasyfikacja? 98
- 4.2. Przegląd klasyfikatorów 101
  - 4.2.1. Strukturalne algorytmy klasyfikacji 102
  - 4.2.2. Statystyczne algorytmy klasyfikacji 104
  - 4.2.3. Cykl życia klasyfikatora 105
- 4.3. Wykrywanie oszustw za pomocą regresji logistycznej 106
  - 4.3.1. Wprowadzenie do regresji liniowej 106
  - 4.3.2. Od regresji liniowej do logistycznej 108
  - 4.3.3. Implementowanie wykrywania oszustw 111

- 4.4. Czy wyniki są wiarygodne? 119
- 4.5. Klasyfikowanie w bardzo dużych zbiorach danych 122
- 4.6. Podsumowanie 124

## Rozdział 5. Studium przypadku — prognozowanie kliknięć w reklamie internetowej 127

- 5.1. Historia i informacje wstępne 128
- 5.2. Giełda 130
  - 5.2.1. Dopasowywanie plików cookie 130
  - 5.2.2. Oferty 131
  - 5.2.3. Powiadomienie o wygranej (lub przegranej) w licytacji 132
  - 5.2.4. Umieszczanie reklamy 132
  - 5.2.5. Monitorowanie reklam 132
- 5.3. Czym jest agent? 133
  - 5.3.1. Wymagania stawiane agentowi 133
- 5.4. Czym jest system podejmowania decyzji? 134
  - 5.4.1. Informacje o użytkowniku 135
  - 5.4.2. Informacje o przestrzeni reklamowej 135
  - 5.4.3. Informacje o kontekście 135
  - 5.4.4. Przygotowywanie danych 135
  - 5.4.5. Model dla systemu podejmowania decyzji 136
  - 5.4.6. Odwzorowywanie prognozowanego współczynnika kliknięć na oferowaną kwotę 136
  - 5.4.7. Inżynieria cech 137
  - 5.4.8. Trening modelu 137
- 5.5. Predykcja kliknięć za pomocą biblioteki Vowpal Wabbit 138
  - 5.5.1. Format danych używany w VW 138
  - 5.5.2. Przygotowywanie zbioru danych 141
  - 5.5.3. Testowanie modelu 146
  - 5.5.4. Kalibrowanie modelu 148
- 5.6. Komplikacje związane z budowaniem systemu podejmowania decyzji 150
- 5.7. Przyszłość prognozowania zdarzeń w czasie rzeczywistym 150
- 5.8. Podsumowanie 151

## Rozdział 6. Uczenie głębokie i sieci neuronowe 153

- 6.1. Intuicyjne omówienie uczenia głębokiego 154
- 6.2. Sieci neuronowe 155
- 6.3. Perceptron 156
  - 6.3.1. Trening 158
  - 6.3.2. Trening perceptronu z użyciem pakietu scikit-learn 160
  - 6.3.3. Geometryczna interpretacja działania perceptronu dla dwóch wejść 162
- 6.4. Perceptrony wielowarstwowe 164
  - 6.4.1. Trening z wykorzystaniem propagacji wstecznej 167
  - 6.4.2. Funkcje aktywacji 168
  - 6.4.3. Intuicyjne wyjaśnienie propagacji wstecznej 169
  - 6.4.4. Teoria propagacji wstecznej 170
  - 6.4.5. Wielowarstwowe sieci neuronowe w pakiecie scikit-learn 172
  - 6.4.6. Perceptron wielowarstwowy po zakończeniu nauki 174

6.5. Zwiększenie głębokości — od wielowarstwowych sieci neuronowych do uczenia głębokiego	175
6.5.1. <i>Ograniczone maszyny Boltzmana</i>	176
6.5.2. <i>Maszyny BRBM</i>	177
6.5.3. <i>Maszyny RBM w praktyce</i>	180
6.6. Podsumowanie	183

## Rozdział 7. Dokonywanie właściwego wyboru 185

7.1. Testy A/B	187
7.1.1. <i>Teoria</i>	187
7.1.2. <i>Kod</i>	190
7.1.3. <i>Adekwatność testów A/B</i>	191
7.2. Wieloręki bandyta	192
7.2.1. <i>Strategie stosowane w problemie wielorękiego bandyty</i>	192
7.3. Strategia bayesowska w praktyce	197
7.4. Testy A/B a strategia bayesowska	207
7.5. Rozwinięcia eksperymentu z wielorękim bandytą	208
7.5.1. <i>Bandyci kontekstowi</i>	209
7.5.2. <i>Problem bandytów z przeciwnikiem</i>	210
7.6. Podsumowanie	210

## Rozdział 8. Przyszłość inteligentnej sieci 213

8.1. Przyszłe zastosowania inteligentnej sieci	214
8.1.1. <i>Internet rzeczy</i>	214
8.1.2. <i>Opieka zdrowotna w domu</i>	215
8.1.3. <i>Autonomiczne samochody</i>	215
8.1.4. <i>Spersonalizowane fizyczne reklamy</i>	216
8.1.5. <i>Sieć semantyczna</i>	216
8.2. Społeczne implikacje rozwoju inteligentnej sieci	217

## Dodatek. Pobieranie danych z sieci WWW 219

Przykład — wyświetlanie reklam w internecie	220
Dane dostępne w kontekście reklamy internetowej	220
Rejestrowanie danych — naiwne rozwiązanie	221
Zarządzanie zbieraniem danych w dużej skali	222
Poznaj system Kafka	224
Replikacja w systemie Kafka	226
Grupy konsumentów, równoważenie i kolejność	232
Łączenie wszystkich elementów	233
Ocena systemu Kafka — rejestrowanie danych w dużej skali	236
Wzorce projektowe w systemie Kafka	238
Łączenie systemów Kafka i Storm	238
Łączenie systemów Kafka i Hadoop	240
Skorowidz	243

# Przedmowa

---

Sieć WWW to infrastruktura wykorzystywana przez oparte na internecie społeczeństwo informacyjne. Jest to podstawowe narzędzie, z którego miliardy osób korzystają do interakcji w internecie. Postęp przemysłowy dokonuje się dzięki rozwijaniu usług informacyjnych w internecie. Obecnie dzięki dojrzałym technologiom przetwarzania w chmurze i komunikacji bezprzewodowej sieć WWW staje się nie tylko narzędziem do publikowania i konsumowania informacji, ale też platformą, w której można rozwijać i wdrażać usługi informacyjne, a także udostępniać je miliardom użytkowników w dowolnym miejscu i czasie. Duże zbiory danych (ang. *big data*) zapewniają bogate materiały do budowania wszechstronnych usług, a także umożliwiają wbudowanie inteligencji w usługi i ulepszenie dzięki temu wrażeń z użytkowania usług w sieci WWW. Inteligentne usługi sprawiają, że sieć WWW zmienia nasze życie. Pomaga nam znaleźć odpowiednią restaurację, zaplanować wymarzone wakacje, kupić niemal dowolny produkt i tworzyć społeczności mające najróżniejsze cele. Uzyskanie tej inteligencji jest możliwe dzięki analizie danych wygenerowanych w wyniku interakcji użytkowników z zawartością sieci WWW. Rozwijanie inteligencji w sieci jest więc jednym z podstawowych aspektów nowoczesnej nauki o danych.

Mam wielką przyjemność zaprezentować Ci tę doskonałą książkę, *Inteligentna sieć. Algorytmy przyszłości*, zaktualizowaną przez młodego, ale bardzo doświadczonego badacza danych, dr. Douglasa McIlwraitha. Ta pozycja ma pokazać istotę inteligentnych aplikacji sieciowych — algorytmy zapewniające inteligencję. To ambitny cel. Zaskoczyło mnie to, że Doug zdołał kompleksowo przedstawić ten obszerny temat w przystępny języku na mniej niż 250 stronach.

W tej książce opisane są najpopularniejsze techniki o szerokim spektrum zastosowań. Znajdziesz tu zwięzły opis algorytmów i ich matematycznych podstaw oraz kod w Pythonie. Lektura tej książki była dla mnie prawdziwą przyjemnością. Mam nadzieję, że Tobie też się ona spodoba. Ważniejsze jest jednak to, że po zakończeniu lektury powinieneś zyskać umiejętności i wiedzę pozwalające zwiększyć inteligencję sieci WWW.

**Yike Guo**  
Profesor i dyrektor  
Data Science Institute,  
Imperial College, Londyn



# *Wprowadzenie*

---

Mamy szczęście pracować w jednym z najbardziej ekscytujących obszarów technologii naszych czasów. W ciągu zaledwie kilku lat przeszliśmy od rodzącego się internetu do kompletnie rozwiniętej sieci WWW. Obecnie każdy z nas nosi w kieszeni potężne narzędzie do komunikacji i może błyskawicznie znaleźć odpowiedź na prawie dowolne pytanie.

Rozwijanie inteligentnych algorytmów, które pozwolą zrozumieć i wykorzystać dostępne informacje, odegrało niemałą rolę w powstaniu nowego paradygmatu w sieci. Naszą uwagę zwraca piękno symetrii: w coraz większym stopniu polegamy na inteligentnych algorytmach, które pomagają nam zarządzać poczynaniami w internecie i poza nim, a dzięki temu uzyskujemy więcej danych i lepszy wgląd w to, jak trenować i testować te algorytmy. Ledwie kilka lat temu sieci neuronowe straciły popularność w kręgach naukowych. Jednak obecnie, wraz z pojawianiem się dużych i dostępnych zbiorów danych, ponownie udało się pokazać ich przydatność.

*Prawie* doszliśmy do momentu, w którym będziemy mogli mówić do telefonu, a ten będzie przewidywał nasze potrzeby, umawiał nas na spotkania i komunikował się z odpowiednimi osobami. W nieco bardziej odległej przyszłości pojawią się autonomiczne samochody i rzeczywistość wirtualna. Wszystkie te technologie wzięły się z zastosowania nauk komputerowych do rozwiązywania praktycznych problemów. Inteligentne algorytmy są i będą częścią tego procesu.

Jednak pierwsze kroki w świecie uczenia maszynowego i nauki o danych mogą okazać się trudne. Jest to dziedzina w dużym stopniu zależna od matematyki i statystyki, w której poleganie na samej intuicji często prowadzi do problemów! Chcieliśmy zaktualizować tę książkę, aby uwzględnić postępy poczynione od czasu napisania pierwszego wydania, a także pomóc osobom nowym w omawianej dziedzinie. Na stronach tej książki znajdziesz przystępne przykłady i praktyczne rozwiązania w postaci kodu. Tam, gdzie było to możliwe, staraliśmy się większą uwagę przykładać do podstawowych zasad danej techniki, a nie do kwestii matematycznych. Niełatwo jest zachować równowagę między tymi aspektami, jednak mamy nadzieję, że dobrze sobie z tym poradziliśmy.

Książka zawiera osiem rozdziałów. Każdy z nich dotyczy ważnego obszaru inteligentnej sieci w kontekście algorytmicznym. Książka kończy się dodatkiem, w którym inteligentna sieć jest opisana z perspektywy przetwarzania danych. Dodatek zamieściliśmy, aby praktycy z omawianej dziedziny mogli docenić, jak ważne (i trudne) jest wydajne przenoszenie generowanych z dużą szybkością danych w systemie.



# *Podziękowania*

---

Dziękujemy wszystkim z wydawnictwa Manning, dzięki którym powstanie tej książki było możliwe: wydawcy Marjanowi Bace'owi, a także pracownikom z działów redakcyjnego i produkcyjnego, w tym Janet Vail, Kevinowi Sullivanowi, Tiffany Taylor, Dottie Marisco, Lindzie Rectenwald i wielu innym osobom pracującym na zapleczu.

Jesteśmy wdzięczni wszystkim, którzy recenzowali książkę na różnych etapach jej powstawania. Oto te osoby: Nii A-Okine, Tobias Bürger, Marius Butuc, Carlton Gibson, John Guthrie, Pieter Gyselinck, PeterJohn Hampton, Dike Kalu, Seth Liddy, Radha Ranjan Madhav, Kostas Passadis, Peter Rabinovitch, Srdjan Santic, Dennis Sellinger, dr Joseph Wang i Michael Williams. Wiemy, ile czasu i wysiłku potrzeba na dokładne przeczytanie takiego tekstu, dlatego dziękujemy Wam wszystkim. Otrzymaliśmy od Was bezcenne informacje uwzględnione w książce.

Specjalne podziękowania należą się Davidowi Fombelli Pombalowi za doskonałą recenzję techniczną, dr. Michaelowi Davy'emu za recenzję rozdziału 3. i Matthew Swordowi za recenzję rozdziału 7. Dziękujemy Wam wszystkim.

W tej książce korzystamy z wielu systemów, bibliotek i pakietów opracowanych przez inne osoby. Społeczność programistów, badaczy z dziedziny nauki o danych i ekspertów od uczenia maszynowego robi wiele dobrego. Jesteśmy wdzięczni wszystkim członkom tej społeczności.

Gdy toczyliśmy wstępne rozmowy na temat aktualizacji książki *Inteligentna sieć. Algorytmy przyszłości*, pamiętam, że myślałem wtedy tak: „Pierwsze wydanie było świetne. Ile pracy może wymagać opracowanie drugiego?”. Okazuje się, że wiele. Zmiany w omawianej dziedzinie następują tak szybko i pojawiło się tak wiele ciekawych rozwiązań, którymi chciałem się podzielić, że musiałem bardzo starannie przemyśleć, co zostawić w książce, co usunąć, co zaktualizować i co dodać. Dlatego prace nad nią zajęły znacznie więcej czasu, niż oczekiwałem. Mam jednak to szczęście, że otacza mnie wielu wspaniałych ludzi, którzy są wobec mnie cierpliwi oraz wspierają mnie i motywują, kiedy tego potrzebuje.

Przede wszystkim chcę podziękować mojej narzeczonej, Elly. Kochanie, Twoja miłość, cierpliwość i wsparcie to bardzo cenne stałe w moim życiu. Jestem przekonany, że bez nich ta książka nigdy by nie powstała. Kocham Cię.

Po drugie, dziękuję moim rodzicom i krewnym, którzy stale rozbudzali moją ciekawość i wspierali mnie w trudnych momentach. Mam nadzieję, że ta książka Wam się spodoba i że wiecie, jak wdzięczny jestem Wam za opiekę i troskę.

Po trzecie, chciałbym wspomnieć o znajomych i współpracownikach, których jest zbyt wielu, aby wymieniać ich z nazwiska. Miałem szczęście pracować i bawić się z wyjątkowymi ludźmi. Dzięki temu każdy mój dzień był szczęśliwy. Jestem wdzięczny Wam wszystkim.

Jestem też wdzięczny redaktorom, Jeffowi Bleielowi i Jennifer Stout. To Wy sprawiłyście, że książkę udało się doprowadzić do obecnego poziomu. Jennifer, Twoje pozytywne nastawienie i energia zachęcały mnie do wysiłku pod koniec prac, za co składam Ci podziękowania.

**Douglas McIlwraith**

Dziękuję moim rodzicom, Evie i Alexandrowi. To oni rozbudzili we mnie odpowiedni poziom ciekawości i pasji do nauki, który motywuje mnie do pisania i prowadzenia badań do późna w nocy. Mój dług wobec Was jest zbyt duży, aby mógł go spłacić w ciągu jednego życia.

Z całego serca dziękuję mojej kochanej żonie, Aurorze, i trzem synom, Nikosowi, Lukasowi i Albertowi, którzy są największą dumą i radością mojego życia. Zawsze będę wdzięczny za Waszą miłość, cierpliwość i wyrozumiałość. Nieskończona ciekawość moich dzieci stale inspiruje mnie do badań nad uczeniem się. Wielkie podziękowania należą się też moim teściom, Cuchi i Jose, moim siostrrom, Marii i Katerinie, a także moim najlepszym przyjaciołom, Michaelowi i Antoniemu, za ich ciągłe zachęty i bezwarunkowe wsparcie.

Zaniedbaniem byłoby nie wspomnieć o przyjmującym różną postać wsparciu ze strony dr. Amilcara Avendaño i dr Marii Balerdi, którzy nauczyli mnie wiele o kardiologii i sfinansowali moje wczesne prace z zakresu uczenia się. Dziękuję również profesorowi Leonowi Cooperowi i wielu innym wspaniałym ludziom z Brown University, których zapalił do badania funkcjonowania mózgu zainspirował osoby takie jak ja i był źródłem moich prac z dziedziny intelligentnych aplikacji.

Chcę wspomnieć też o moich obecnych i byłych współpracownikach. Są to: Ajay Bhadari, Kavita Kantkar, Alexander Petrov, Kishore Kirdat i liczne inne osoby, które wspierały wszystkie inicjatywy związane z inteligencją. Mogę poświęcić Wam tylko kilka wierszy, jednak jestem Wam wszystkim bardzo wdzięczny.

**Haralambos Marmanis**

Przede wszystkim dziękuję mojej kochanej żonie, Elenie.

Ponadto chcę podziękować moim obecnym i byłym współpracownikom, którzy wywarli wpływ na moje życie zawodowe i byli dla mnie inspiracją. Oto oni: Konstantin Bobovich, Paul A. Dennis, Keith Lawless i Kevin Bedell.

**Dmitry Babenko**

# O książce

---

Książka *Inteligentna sieć — algorytmy* została napisana po to, aby zapewnić schemat projektowania i tworzenia inteligentnych algorytmów. W książce wykorzystaliśmy wiele z różnych dziedzin informatyki (takich jak uczenie maszynowe i sztuczna inteligencja), ale napisaliśmy ją z myślą o praktykach. Jest to pewnego rodzaju „książka kucharska”, oferująca nowicjuszom w omawianym obszarze praktyczne i sprawdzone przykłady, które każdy może zmodyfikować na własne potrzeby.

## **Kto powinien przeczytać tę książkę?**

Ta książka jest skierowana do osób, które uczą się pisać inteligentne algorytmy, ale mają solidne podstawy w zakresie programowania oraz matematyki i statystyki. Tam, gdzie było to możliwe, staraliśmy się pisać książkę w taki sposób, abyś nie musiał skupiać się na kwestiach matematycznych. W zamian chcieliśmy pokazać na ogólnym poziomie możliwości zastosowania poszczególnych podejść. Oczywiście jeśli jesteś zainteresowany matematyką, zachęcamy do zapoznania się z matematycznymi szczegółami. Optymalnie czytelnicy książki powinni mieć przynajmniej podstawową znajomość języka programowania i ukończyć kurs matematyki z pierwszego roku studiów.

## **Plan książki**

Ta książka zawiera osiem rozdziałów i jeden dodatek:

- Rozdział 1. to wprowadzenie do inteligentnych algorytmów i opis ich wybranych najważniejszych cech. Obejmuje też zasady, którymi będziesz się kierował w dalszych rozdziałach książki.
- Rozdział 2. zawiera omówienie struktury danych. Przede wszystkim przedstawiamy tu przestrzeń cech, a także maksymalizacji wartości oczekiwanej i wektorów własnych.
- Rozdział 3. obejmuje wprowadzenie do systemów rekomendacji. Przedstawiamy tu techniki filtrowania kolaboratywnego i omawiamy konkurs Netflix Prize.
- W rozdziale 4. znajdziesz opis technik kategoryzowania i wprowadzenie do regresji logistycznej. Zobaczysz tu, jak wykorzystać regresję logistyczną do wykrywania oszustw.
- W rozdziale 5. przedstawiamy studium przypadku dotyczące przewidywania kliknięć na potrzeby reklamy internetowej. Wyjaśniamy, jak reklama internetowa działa na zapiszu, a także prezentujemy praktyczny przykład przewidywania kliknięć z wykorzystaniem publicznie dostępnego zbioru danych o kliknięciach.

- Rozdział 6. poświęcony jest głębokiemu uczeniu i sieciom neuronowym. Przedstawiamy tu krótki przegląd sieci neuronowych od ich skromnych początków, a także omawiamy najnowsze dokonania w zakresie uczenia się z wykorzystaniem głębokich sieci.
- W rozdziale 7. wyjaśniamy, jak podejmowane są właściwe decyzje. Omawiamy istotność statystyczną testów A/B, a także kilka sposobów uczenia na żywo na przykładzie problemu wielorękiego bandyty.
- Rozdział 8. zawiera omówienie perspektyw inteligentnej sieci.
- W dodatku wyjaśniamy, jak przetwarzać strumień szybko generowanych zdań i budować wykorzystujące je inteligentne algorytmy. Omawiamy tu kilka wzorców projektowych dotyczących przetwarzania logów sieciowych i przedstawiamy kilka ważnych pułapek, których należy unikać.

Większość rozdziałów można czytać niezależnie od pozostałych, jednak rozdział 5. to studium przypadku wymagające wiedzy z zakresu opisanej w rozdziale 4. regresji logistycznej.

## ***Materiały do pobrania***

Kod i dane potrzebne do uruchomienia przykładów z książki można pobrać z witryny wydawnictwa Manning (<https://www.manning.com/books/algorithms-of-the-intelligent-web-second-edition>) i z serwisu GitHub (<https://github.com/dougmclurraith/aiw-second-edition>). Polską wersję znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/intsi2.zip>). Wyjątkiem jest zbiór danych Criteo Display Challenge Dataset, który z powodu jego wielkości trzeba pobrać bezpośrednio z witryny firmy Criteo (instrukcje zawiera rozdział 5.).

Cały kod został przetestowany w systemie Ubuntu 14.04.2 z użyciem języka Python 2.7.10. Należy uwzględnić różne zależności opisane w pliku z wymaganiami w materiałach do pobrania. W tym pliku znajdziesz instrukcje pozwalające zapewnić zgodność używanego środowiska z przykładowym kodem z tej książki.

## ***Konwencje stosowane w kodzie***

W książce znajdują się liczne przykłady. Kod źródłowy w listingach i elementy kodu poza listingami są zapisane czcionką o stałej szerokości, co pozwala odróżnić je od zwykłego tekstu. W niektórych miejscach podzieliliśmy wiersze i zmieniliśmy wcięcia, aby uwzględnić szerokość strony. Gdy to nie wystarczyło, wiersze kończą się symbolem kontynuacji (→). Ponadto w listingach często pomijamy komentarze z kodu źródłowego, jeśli kod jest objaśniony w tekście. Z niektórymi listingami powiązane są opisy wyjaśniające ważne zagadnienia.

## Konwencje matematyczne

W tekście znajdziesz wiele równań powiązanych z kodem i przedstawianymi zagadnieniami. Stosujemy w nich standardowe konwencje. Macierze są reprezentowane za pomocą wielkich pogrubionych liter (na przykład **M**). Małe pogrubione litery (na przykład **v**) reprezentują wektory. Wartości skalarne są zapisywane kursywą za pomocą małych liter (na przykład *v*).

## O autorach

Dr Douglas McIlwraith pierwszy dyplom uzyskał w dziedzinie informatyki w Cambridge, a później otrzymał tytuł doktora na uczelni Imperial College w Londynie. Jest ekspertem w dziedzinie uczenia maszynowego i obecnie pracuje jako analityk danych w londyńskiej agencji reklamowej. Prowadził badania w dziedzinach systemów rozproszonych, przetwarzania bez granic (ang. *ubiquitous computing*), wszechobecnych czujników (ang. *pervasive sensing*), robotyki i zabezpieczeń. Ekscytuje go obserwowanie, jak technologia wywiera pozytywny wpływ na życie ludzi.

Dr Haralambos Marmannis jest pionierem w obszarze stosowania technik uczenia maszynowego w rozwiązańach przemysłowych. Ma 25 lat doświadczenia w rozwijaniu profesjonalnego oprogramowania.

Dmitry Babenko projektował i budował różne aplikacje oraz platformy infrastrukturalne dla firm z branż: bankowej, ubezpieczeniowej, zarządzania łańcuchem dostaw i analityki biznesowej. Uzyskał tytuł magistra informatyki Białoruskiego Uniwersytetu Informatyki i Radioelektroniki.



# 1

## *Budowanie aplikacji na potrzeby inteligentnej sieci*

### **Zawartość rozdziału:**

- Dostrzeganie inteligencji w sieci
- Typy inteligentnych algorytmów
- Ocena inteligentnych algorytmów

Określenie „inteligentna sieć” dla różnych osób znaczy coś innego. Dla niektórych związane jest z ewolucją sieci WWW w kierunku reagującego na interakcje i przydatnego bytu, który potrafi uczyć się od użytkowników i reagować na ich zachowania. Dla innych znaczy wkroczenie sieci WWW w wiele aspektów naszego życia. Dla mnie inteligentna sieć jest daleka od pierwszej wersji Skynetu, w którym komputery przejmują władzę w dystopiancznej przyszłości. Jest natomiast związana z projektowaniem i implementowaniem w naturalny sposób reagujących aplikacji, dzięki którym wrażenia z użytkowania internetu są w polliczalny sposób lepsze. Prawdopodobnie każdy z Czytelników zetknął się w wielu sytuacjach z inteligencją maszynową. W tym rozdziale przedstawiamy przykłady, które ułatwią Ci dostrzeżenie jej w przyszłości. To z kolei pomoże Ci zrozumieć, co tak naprawdę dzieje się na zapleczu, gdy wchodzisz w interakcje z intelligentnymi aplikacjami.

Skoro wiesz już, że ta książka nie dotyczy pisania bytów, które spróbuja przejąć kontrolę nad światem, warto wspomnieć też o innych rzeczach pominiętych na jej stronach. Przede wszystkim jest to książka skoncentrowana na technologiach używanych na zapleczu. Nie przeczytasz tu o atrakcyjnych interaktywnych wizualizacjach

lub platformach. Te tematy poznasz dzięki świetnym publikacjom Scotta Murraya<sup>1</sup>, Davida McCandlessa<sup>2</sup> i Edwarda Tuftego<sup>3</sup>. Tu nie mamy miejsca na omówienie tego zagadnienia w stopniu, na jaki zasługuje. Z tej książki nie nauczysz się też statystyki. Jednak aby jak najlepiej wykorzystać zawartość tej pozycji, powinieneś znać przyjmniej podstawy tej dziedziny i ukończyć kurs statystyki.

Nie jest to też książka o nauce o danych. Dostępnych jest wiele tytułów pomocnych dla praktyków nauki o danych. Mamy nadzieję, że także ta książka będzie dla nich przydatna, jednak w jej rozdziałach znajdziesz mało szczegółów na temat tego, jak być naukowcem z tej dziedziny. Jej omówienie znajdziesz w teksthach Joel Grusa<sup>4</sup> oraz Fostera Provosta i Toma Fawcetta<sup>5</sup>.

Ponadto ta książka nie jest szczegółowym omówieniem projektowania algorytmów. Często pomijamy szczegóły projektowania algorytmów i przedstawiamy intuicyjny opis zamiast wnikania w szczegóły. Pozwala to omówić więcej zagadnień, choć może działać się to kosztem precyzji. Każdy rozdział możesz traktować jak trop prowadzący przez ważne aspekty danego podejścia i pozwalający dotrzeć do zasobów zawierających więcej szczegółów.

Choć wiele przykładów z tej książki jest napisanych z użyciem pakietu scikit-learn (<http://scikit-learn.org>), nie jest to pozycja poświęcona temu narzędziu. Ten pakiet to tylko narzędzie pozwalające zademonstrować prezentowane w tekście techniki. Każdy przykład opatrzyliśmy przynajmniej ogólnym objaśnieniem tego, dlaczego dany algorytm działa. W niektórych sytuacjach przedstawiamy więcej szczegółów, jednak w wielu miejscach powinieneś kontynuować poszukiwania poza tą książką.

O czym więc jest ta pozycja? Omawiamy tu narzędzia związane z całym procesem funkcjonowania nowoczesnych intelligentnych algorytmów. Opisujemy informacje zbierane na temat przeciętnych użytkowników sieci, które mogą być przetwarzane w przydatne strumienie pozwalające prognozować zachowania tych osób (oraz modyfikować prognozy w reakcji na zmiany zachowań użytkowników). To oznacza, że często odchodzimy od modelu typowego dla książek o podstawach algorytmów i dajemy Ci przedsmak (!) wszystkich ważnych aspektów intelligentnych algorytmów.

Omawiamy nawet (w dodatku) technologię publikuj-subskrybuj, która umożliwia porządkowanie dużych ilości danych w trakcie zbierania. Choć nie jest to temat do książki poświęconej ściśle nauce o danych lub algorytmom, uważamy, że należy opisać go w pozycji dotyczącej intelligentnej sieci. Nie oznacza to, że ignorujemy naukę o danych lub algorytmy — w żadnym razie! Omawiamy tu większość najważniejszych algorytmów używanych przez czołowych graczy w dziedzinie intelligentnych algorytmów.

---

<sup>1</sup> Scott Murray, *Interactive Data Visualization for the Web* (O'Reilly, 2013).

<sup>2</sup> David McCandless, *Information Is Beautiful* (HarperCollins, 2010).

<sup>3</sup> Edward Tufte, *The Visual Display of Quantitative Information* (Graphics Press USA, 2001).

<sup>4</sup> Joel Grus, *Data Science From Scratch: First Principles with Python* (O'Reilly, 2015).

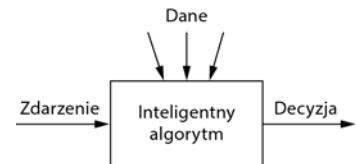
<sup>5</sup> Foster Provost i Tom Fawcett, *Data Science for Business* (O'Reilly Media, 2013).

Tam, gdzie to możliwe, wskazujemy znane przykłady zastosowania technik w praktyce. Dzięki temu będziesz mógł porównać swoją wiedzę z działaniem systemów — i bez wątpienia zrobić tym wrażenie na znajomych!

Wybiegamy jednak za daleko w przyszłość. W tym rozdziale przedstawiamy kilka przykładów zastosowania inteligentnych algorytmów, które powinieneś natychmiast rozpoznać. Opisujemy, czego inteligentne algorytmy nie potrafią, a następnie przedstawiamy taksonomię omawianego obszaru, z którą będziesz mógł wiązać poznawane zagadnienia. W końcowej części prezentujemy szereg metod oceny inteligentnych algorytmów i przedstawiamy kilka przydatnych informacji.

Już słyszymy, jak pytasz: „Czym jest inteligentny algorytm?”. Na potrzeby tej książki za *inteligentny* uznajemy każdy algorytm, który wykorzystuje dane do modyfikacji swojego działania. Pamiętaj, że gdy wchodzisz w interakcję z algorytmem, komunikujesz się wyłącznie z zestawem określonych reguł. Inteligentne algorytmy różnią się od innych tym, że mogą zmieniać swoje działanie w trakcie pracy.

Często użytkownik ma wrażenie, że taki algorytm jest inteligentny. Rysunek 1.1 przedstawia takie algorytmy. Widać tu, że inteligentny algorytm reaguje na zachodzące w środowisku zdarzenia i podejmuje decyzje. Zbierając dane (mogą nimi być też same zdarzenia) z kontekstu, w którym działa, algorytm ewoluuje — w tym sensie, że decyzje nie zależą deterministycznie od samego zdarzenia. Inteligentny algorytm w różnych momentach może podejmować odmienne decyzje w zależności od zebranych danych.



Rysunek 1.1. Ogólny obraz pracy inteligentnego algorytmu. Taki algorytm przejawia inteligencję, ponieważ podejmuje decyzje na podstawie zebranych danych

## 1.1. Inteligentny algorytm w akcji — Google Now

Aby zilustrować proces pokazany na rysunku, postaramy się przeprowadzić analizę aplikacji Google Now. Warto wspomnieć, że jej szczegóły są chronione przez firmę Google, dlatego wykorzystujemy nasze doświadczenie do pokazania, jak algorytm z tej aplikacji może działać na zapleczu.

Użytkownicy urządzeń z systemem Android zapewne natychmiast rozpoznają ten produkt, a dla osób korzystających z systemu iOS mamy informację, że Google Now to odpowiedź Google'a na program Siri. Google reklamuje go za pomocą hasła: „Odpowiednie informacje we właściwym czasie”. Google Now to aplikacja potrafiąca wykorzystać różne źródła informacji i powiadamić użytkowników o pobliskich restauracjach, wydarzeniach, korkach i podobnych rzeczach, które uzna za interesujące dla danej osoby. Aby pokazać, czym jest inteligentny algorytm, posłużymy się konkretnym przykładem z aplikacji Google Now. Gdy wykryje ona korek na standardowej drodze użytkownika do pracy, wyświetla określone informacje przed wyjściem danej osoby z domu. Świetnie! Jak jednak jest to możliwe?

Zaczniemy od zrozumienia, co się tu dzieje. Aplikacja zna lokalizację użytkownika dzięki modułowi GPS i zarejestrowanym stacjom łączności bezprzewodowej. Dlatego aplikacja zawsze wie, gdzie użytkownik się znajduje (z dość dużą dokładnością).

W kontekście rysunku 1.1 jest to jeden z aspektów danych używanych do zmiany działania algorytmu. Teraz potrzebny jest tylko niewielki krok, by ustalić lokalizację domu i pracy. Odbywa się to dzięki wykorzystaniu *uprzedniej wiedzy*, która została wbudowana w algorytm, zanim zaczął on uczyć się na podstawie danych. Tu uprzednia wiedza może mieć postać następujących reguł:

- Lokalizacja najczęściej wykrywana w nocy to dom.
- Lokalizacja najczęściej wykrywana w ciągu dnia to praca.
- Ludzie (w większości) prawie każdego dnia jadą do pracy, a następnie z powrotem do domu.

Choć ten przykład nie jest idealny, dobrze ilustruje pewną kwestię: w społeczeństwie używane są pojęcia „pracy”, „domu” i „dojazdów”, a na podstawie danych i *modelu* można wyciągać wnioski. Tu można ustalić prawdopodobną lokalizację domu i pracy wraz z prawdopodobnymi trasami dojazdu. Używamy tu określenia *prawdopodobne*, ponieważ w wielu modelach uwzględniane jest prawdopodobieństwo określonych decyzji.

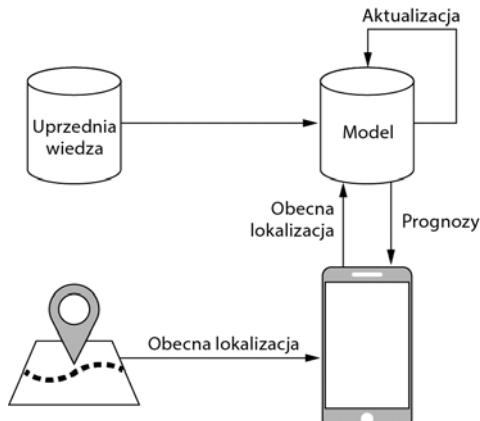
Gdy użytkownik kupuje nowy telefon lub rejestruje nowe konto w usługach firmy Google, Google Now potrzebuje czasu na wyciągnięcie wniosków. Podobnie dzieje się, gdy użytkownik zmienia mieszkanie lub pracę. Aplikacja musi wtedy nauczyć się nowych lokalizacji. Szybkość reagowania modelu na zmiany to *szybkość uczenia się*.

Nadal jednak brakuje informacji, aby móc wyświetlać adekwatne informacje dotyczące tras dojazdu (aby podejmować *decyzje* na podstawie *zdarzeń*). Ostatni fragment układanki wymaga prognozowania, kiedy użytkownik zamierza opuścić jedną lokalizację i wyruszyć w drogę do drugiej. Podobnie jak wcześniej można utworzyć model i określić godzinę dojazdów, a następnie aktualizować ją, aby odzwierciedlić zmiany wzorców zachowania. W przyszłości można określić prawdopodobieństwo, z jakim użytkownik znajduje się w danej lokalizacji i szkuje się do drogi. Jeśli to prawdopodobieństwo przekroczy poziom progowy, Google Now może sprawdzić informacje o korkach i przekazać je w powiadomieniu użytkownikowi.

Ten konkretny aspekt aplikacji Google Now jest dość skomplikowany i prawdopodobnie pracuje nad nim specjalny zespół. Jednak łatwo jest zauważyc, że schemat działania tego narzędzia oparty jest na inteligentnym algorytmie. Aplikacja wykorzystuje *dane* na temat dojazdów, aby zrozumieć zwyczaje użytkownika i przygotować dla niego spersonalizowane podpowiedzi (*decyzje*) na podstawie obecnej lokalizacji (*zdarzenie*). Rysunek 1.2 przedstawia ten proces w formie graficznej.

Warto zauważyc, że produkt Google Now prawdopodobnie wykorzystuje na zapleczu cały pakiet inteligentnych algorytmów. Przeprowadzają one przeszukiwanie tekstowe kalendarza Google, starając się zrozumieć plan dnia użytkownika, a modele badania zainteresowań próbują ustalić, które wyniki wyszukiwania są przydatne i czy należy oznaczyć nowe treści jako interesujące.

Programista inteligentnych algorytmów musi korzystać ze swoich umiejętności do budowania nowych rozwiązań na podstawie złożonych wymagań i starannie identyfikować wszystkie podzadania, które można wykonać za pomocą istniejącej klasy takich

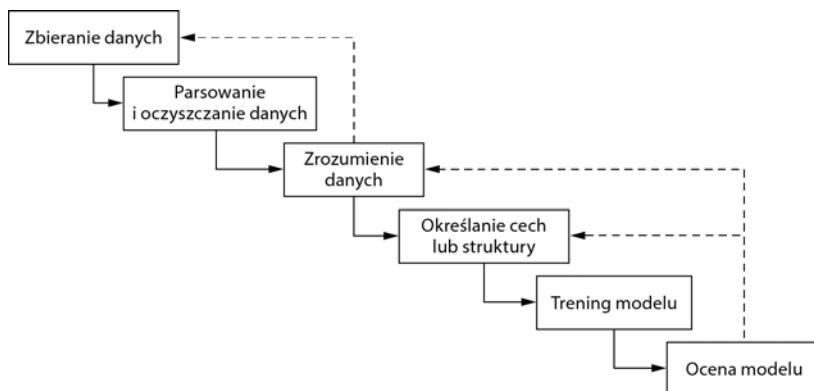


Rysunek 1.2. Graficzne ujęcie jednego z aspektów projektu Google Now. Aby aplikacja Google Now mogła prognozować przyszłe lokalizacje, używa modelu uwzględniającego przeszłe lokalizacje i obecną pozycję. Uprednia wiedza pozwala wbudować wcześniej znane informacje w system

algorytmów. Każde tworzone rozwiązanie powinno być oparte na dokonaniach z omawianej dziedziny i zbudowane na ich podstawie. Wiele takich dokonań omawiamy w tej książce. Wprowadziliśmy tu kilka ważnych pojęć wyróżnionych kursywą. Używamy ich w dalszych rozdziałach przy szczegółowym omawianiu poszczególnych algorytmów.

## 1.2. Cykl życia inteligentnych algorytmów

W poprzednim podrozdziale opisaliśmy, że inteligentny algorytm składa się z czarnej skrzynki, przyjmuje dane i generuje prognozy na podstawie zdarzeń. Przedstawiliśmy konkretny przykład z firmy Google w postaci projektu Google Now. Może się zastanawiasz, jak projektanci inteligentnych algorytmów dochodzą do rozwiązań. Istnieje ogólny cykl życia zaadaptowany z książki *Computational Information Design* Bena Fry’ego<sup>6</sup>. Możesz posługiwać się tym cyklem w trakcie projektowania własnych rozwiązań. Cykl ten pokazany jest na rysunku 1.3.



Rysunek 1.3. Cykl życia inteligentnego algorytmu

<sup>6</sup> Ben Fry, praca doktorska, *Computational Information Design* (MIT, 2004).

Gdy projektujesz inteligentne algorytmy, najpierw musisz pomyśleć o zbieraniu danych (na tym koncentrujemy się w dodatku), a następnie zająć się ich parsowaniem i oczyszczaniem, ponieważ ich format jest niewłaściwy. Następnie trzeba zrozumieć dane, co można uzyskać dzięki ich eksploracji i wizualizacji. Następnie możesz przedstawić dane w odpowiednich formatach (omawiamy to w rozdziale 2.). Na tym etapie jesteś gotowy do rozpoczęcia treningu modelu i oceny zdolności predykcyjnej utworzonego rozwiązania. W rozdziałach od 3. do 7. omawiamy różne modele, którymi możesz się posługiwać. Po zakończeniu każdego etapu możesz cofnąć się do wcześniejszych kroków. Najczęściej używane ścieżki powrotne są pokazane za pomocą przerywanych linii na rysunku 1.3.

### **1.3. Inne przykłady inteligentnych algorytmów**

Przyjrzyjmy się innym aplikacjom z ostatniej dekady, w których też używana jest inteligencja oparta na algorytmach. Punktem zwrotnym w historii sieci WWW było pojawienie się wyszukiwarek. Jednak duża część możliwości sieci WWW pozostawała niewykorzystana aż do 1998 roku, kiedy to zaczęto analizować odsyłacze w kontekście wyszukiwania. Od tego czasu w niecałe 20 lat firma Google rozwinięła się od startupu do lidera w branży technologicznej. Początkowo rozwój tej firmy wynikał z sukcesu wyszukiwania opartego na odsyłaczach, a później — z licznych nowych i innowacyjnych aplikacji z obszaru usług mobilnych i działających w chmurze.

Jednak świat inteligentnych aplikacji sieciowych nie ogranicza się do wyszukiwarek. Amazon był jednym z pierwszych sklepów internetowych, w których użytkownikom prezentowano rekomendacje oparte na wzorcach zakupowych. Możliwe, że znasz tę funkcję. Założmy, że kupujesz książkę na temat platformy JavaServer Faces i inną o Pythonie. Gdy tylko dodasz je do koszyka zakupów, Amazon zaproponuje dodatkowe pozycje powiązane z tymi, które już wybrałeś. Możliwe, że będą to książki dotyczące AJAX-a lub Ruby on Rails. Ponadto gdy ponownie otworzysz witrynę Amazonu, może ona zarekomendować te same lub inne powiązane produkty. Inną inteligentną aplikacją sieciową jest Netflix — największy na świecie internetowy serwis strumieniowania filmów. Oferuje on ponad 53 milionom subskrybentów dostęp do stale zmieniającej się biblioteki filmów i seriali, które można natychmiast obejrzeć za pomocą technologii strumieniowania.

Sukces Netfliksa wynika po części z zapewnienia użytkownikom łatwego sposobu wyboru filmów z bogatej kolekcji. Odpowiada za to system rekomendacji Cinematch. Jego zadanie polega na prognozowaniu, czy użytkownikowi spodoba się dany film. Uwzględniane jest przy tym to, czy danej osobie spodobały się inne filmy. Jest to następny świetny przykład inteligentnej aplikacji sieciowej. Zdolność predykcyjna algorytmu Cinematch jest dla Netfliksa tak ważna, że w październiku 2006 roku firma ogłosiła konkurs na jego usprawnienie z główną nagrodą miliona dolarów. We wrześniu 2009 roku nagroda została przyznana zespołowi BellKor's Pragmatic Chaos. W rozdziale 3. omawiamy algorytmy potrzebne do budowania systemów rekomendacji takich jak Cinematch oraz opisujemy zwycięski projekt.

Wykorzystywanie opinii społeczności do generowania inteligentnych prognoz nie ogranicza się do rekomendacji książek lub filmów. Firma PredictWallStreet rejestruje prognozy użytkowników dotyczące określonych akcji lub indeksów, aby wykryć trendy w opiniach graczy giełdowych i przewidzieć wartość danych papierów wartościowych. Nie zalecamy podjęcia wszystkich oszczędności i rozpoczęcia inwestowania zgodnie z prognozami tej firmy, jest to jednak następny przykład kreatywnego zastosowania w praktyce technik omawianych w tej książce.

## **1.4. Czym inteligentne aplikacje nie są**

Ponieważ w sieci WWW działa tak wiele inteligentnych algorytmów, łatwo jest dojść do wniosku, że odpowiednia liczba inżynierów zdoła opracować lub zautomatyzować dowolny proces. Niech jednak powszechność takich rozwiązań Cię nie zwiedzie.

*Każda odpowiednio zaawansowana technologia jest nieodróżnialna od magii.*

— Arthur C. Clarke

Po zapoznaniu się z aplikacją Google Now możesz mieć skłonność do podejrzewania bardziej rozbudowanych aplikacji o większą inteligencję. Jednak w rzeczywistości takie aplikacje łączą zestaw uczących się algorytmów w celu zapewnienia przydatnego rozwiązania, ograniczonego jednak do konkretnych problemów. Dlatego nie czuj się przytłoczony złożonością zadania, ale zadaj sobie pytanie: „Które aspekty problemu są możliwe do wyuczenia i zamodelowania?”. Dopiero potem będziesz mógł opracować rozwiązania przejawiające inteligencję. W dalszych podrozdziałach omawiamy najczęściej przyjmowane błędne założenia dotyczące inteligentnych algorytmów.

### **1.4.1. Inteligentne algorytmy nie są myślącymi maszynami do uniwersalnych zastosowań**

Na początku tego rozdziału wspomnieliśmy, że nie jest to książka poświęcona budowaniu czujących istot. Omawiamy tu budowanie algorytmów, które potrafią dostosować swoje działanie na podstawie otrzymanych danych. Zgodnie z naszym doświadczeniem projekty biznesowe, które najczęściej kończą się niepowodzeniem, to te z tak rozbudowanymi celami jak rozwiązywanie całego problemu sztucznej inteligencji! Zaczniź od czegoś prostego i rozwijaj to, stale oceniąc aplikację do momentu, w którym uznasz, że pierwotny problem został rozwiązany.

### **1.4.2. Inteligentne algorytmy nie zastąpią ludzi**

Inteligentne algorytmy świetnie radzą sobie z uczeniem się konkretnego zagadnienia na podstawie odpowiednich danych. Nie są jednak dobre w uczeniu się nowych zagadnień wykraczających poza to, jak zostały zaprogramowane. Dlatego inteligentne algorytmy i rozwiązania trzeba opracowywać i łączyć w staranny sposób, aby zapewnić zadowalające efekty.

Ludzie natomiast są doskonałymi uniwersalnymi maszynami obliczeniowymi. Potrafią łatwo zrozumieć nowe koncepcje i wykorzystać wiedzę z jednej dziedziny w innej.

Possiadają różne serwomechanizmy (!) i można ich programować w wielu różnych językach (!!). Błędem jest myśleć, że można łatwo napisać oparte na kodzie rozwiązania wykonyujące pozornie proste ludzkie czynności.

Wiele wymagających udziału człowieka procesów w firmach i organizacjach na pożór wydaje się prostych, jednak zwykle wynika to z tego, że kompletna specyfikacja danego procesu jest nieznana. Dalsze analizy zwykle prowadzą do wykrycia rozbudowanej komunikacji z użyciem różnych kanałów i często także koniecznością uwzględnienia sprzecznych celów. Inteligentne algorytmy słabo sobie radzą w takich scenariuszach i wymagają uproszczenia oraz sformalizowania procesu.

Warto przedstawić prostą, ale trafną analogię do automatyzacji linii montażowych pojazdów silnikowych. W porównaniu z początkami automatyzacji w tej dziedzinie (gdzie pionierem był Henry Ford) obecnie można całkowicie zautomatyzować kroki procesu produkcji z użyciem robotów. Nie zostało to uzyskane, jak mógłby to sobie wyobrażać Henry Ford, dzięki zbudowaniu uniwersalnych humanoidalnych robotów, które zastąpiły pracowników. Automatyzacja była możliwa dzięki abstrakcyjnemu przedstawieniu linii montażowej i rygorystycznemu sformalizowaniu procesu. To z kolei doprowadziło do ścisłego zdefiniowania podzadań, które *można* było rozwiązać dzięki robotyzacji. Choć teoretycznie możliwe jest opracowanie zautomatyzowanych procesów uczenia na potrzeby ręcznie wykonywanych optymalizacji, wymagałoby to podobnego przeprojektowania i formalizacji zadań.

#### **1.4.3. Inteligentne algorytmy nie są odkrywane przez przypadek**

Najlepsze inteligentne algorytmy są często wynikiem wykorzystania prostych abstrakcji i mechanizmów. Takie algorytmy wyglądają na skomplikowane, ponieważ uczą się i zmieniają, ale mechanizmy, na których są oparte, są proste. Natomiast inteligentne algorytmy niskiej jakości często są oparte na wielu warstwach skomplikowanych reguł dodawanych niezależnie od siebie w celu rozwiązyania konkretnych sytuacji. Ujmijmy to tak: zawsze zaczynaj od możliwie najprostszego modelu. Następnie staraj się stopniowo uzyskiwać lepsze wyniki, wbudowując w rozwiązanie dodatkowe inteligentne aspekty. Reguła **KISS** (ang. *keep it simple, stupid*, czyli nie komplikuj, głupku) jest Twoim przyjacielem i niezmienną zasadą inżynierii oprogramowania.

### **1.5. Klasa inteligentnych algorytmów**

Może pamiętaś, że posłużyliśmy się nazwą *inteligentny algorytm* do opisu dowolnego algorytmu, który może modyfikować swoje działanie na podstawie danych. W tej książce to bardzo ogólne określenie ma obejmować wszystkie aspekty inteligencji i uczenia się. Gdy zajrzesz do innych pozycji, prawdopodobnie natrafisz na odmienne określenia, które po części się pokrywają. Oto one: *uczenie maszynowe* (ang. *machine learning* — **ML**), *analityka predykcyjna* (ang. *predictive analytics* — **PA**) i *sztuczna inteligencja* (ang. *artificial intelligence* — **AI**). Rysunek 1.4 przedstawia zależności między tymi dziedzinami.

Choć we wszystkich trzech dziedzinach występują algorytmy wykorzystujące dane do modyfikowania działania, w każdym z tych obszarów kluczowe są inne aspekty. W następnych podpunktach omawiamy po kolejno każdy z tych obszarów, aby zapewnić Ci wiedzę niezbędną do powiązania tych dziedzin.

### 1.5.1. Sztuczna inteligencja

Sztuczna inteligencja, powszechnie znana pod akronimem AI, powstała jako dziedzina informatyki około 1950 roku. Początkowo badacze sztucznej inteligencji mieli ambitne plany i chcieli opracować maszyny myślące podobnie jak ludzie<sup>7</sup>. Z czasem, gdy ustalono pełen zakres prac potrzebnych do zasymulowania inteligencji, cele badaczy stały się bardziej praktyczne i konkretne. Obecnie stosowanych jest wiele definicji sztucznej inteligencji. Na przykład Stuart Russell i Peter Norvig opisują ją tak: „Dziedzina badań nad agentami, które przyjmują bodźce ze środowiska i wykonują działania”<sup>8</sup>, natomiast John McCarthy stosuje następującą definicję: „Dziedzina nauki i inżynierii związana z budowaniem inteligentnych maszyn, a zwłaszcza inteligentnych programów komputerowych”<sup>9</sup>. McCarthy dodaj też, że: „Inteligencja jest obliczeniowym aspektem możliwości realizowania celów w świecie”.

W większości omówień sztuczna inteligencja wiązana jest z badaniami nad agentami (oprogramowaniem i maszynami), które mają zestaw opcji do wyboru i muszą zrealizować konkretne cele. Badania dotyczą określonych dziedzin problemowych (na przykład gier w go<sup>10</sup>, szachy<sup>11</sup> i Jeopardy!<sup>12</sup>) i w takich ograniczonych środowiskach efekty są często znakomite. Na przykład komputer Deep Blue firmy IBM w 1997 roku pokonał Garriego Kasparowa w szachy, a w 2011 roku komputer Watson tej samej firmy wygrał pierwszą nagrodę miliona dolarów w amerykańskim teleturturnieju Jeopardy! Niestety, nieliczne algorytmy dobrze radzą sobie w wymyślonej przez Alana Turinga grze w naśladowanie<sup>13</sup> (nazywanej też testem Turinga), uznawanej przez większość autorów za standardowy test na inteligencję. W tej grze chodzi o to, aby sędzia nie potrafił wykryć, że jego rozmówca jest maszyną (eliminowane są przy tym wskazówki wizualne



Rysunek 1.4. Taksonomia inteligentnych algorytmów

<sup>7</sup> Herbert Simon, *The Shape of Automation for Men and Management* (Harper & Row, 1965).

<sup>8</sup> Stuart Russell i Peter Norvig, *Artificial Intelligence: A Modern Approach* (Prentice Hall, 1994).

<sup>9</sup> John McCarthy, *What Is Artificial Intelligence?* (Stanford University, 2007), <http://www-formal.stanford.edu/jmc/whatisai>.

<sup>10</sup> Bruno Bouzy i Tristan Cazenave, Computer Go: An AI Oriented Survey, „Artificial Intelligence” (Elsevier) 132, nr 1 (2001): 39 – 103.

<sup>11</sup> Murray Campbell, A.J. Hoane i Feng-hsiung Hsu, Deep Blue, „Artificial Intelligence” (Elsevier) 134, nr 1 (2002): 57 – 83.

<sup>12</sup> D. Ferrucci i współpracownicy, Building Watson: An Overview of the DeepQA Project, „AI Magazine” 31, nr 3 (2010).

<sup>13</sup> Alan Turing, Computing Machinery and Intelligence, „Mind” 59, nr 236 (1950): 433 – 60.

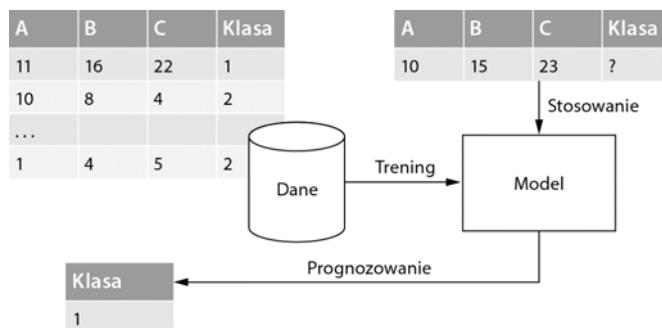
i dźwiękowe — komunikacja odbywa się wyłącznie za pomocą wpisywanego tekstu). To zadanie jest znacznie trudniejsze, ponieważ maszyna musi mieć obszerną wiedzę z wielu obszarów. Sędzia nie jest tu ograniczony, jeśli chodzi o pytania, jakie może zadawać.

### 1.5.2. Uczenie maszynowe

Uczenie maszynowe związane jest ze zdolnością oprogramowania do generalizowania na podstawie wcześniejszych doświadczeń. Ważne jest to, że te generalizacje mają pozwalać na udzielanie odpowiedzi na pytania dotyczące zarówno wcześniej zebranych danych, jak i nowych informacji. Niektóre techniki uczenia polegają na tworzeniu możliwych do wyjaśnienia modeli — nawet laik może prześledzić proces generalizowania. Przykładami są tu drzewa decyzyjne i, w bardziej ogólnym ujęciu, dowolne metody uczenia oparte na regułach. Jednak inne algorytmy nie są równie transparentne dla ludzi. Do tej kategorii należą sieci neuronowe i maszyny **SVM** (ang. *support vector machines*).

Możesz uznać, że zakres uczenia maszynowego znacznie różni się od tematyki sztucznej inteligencji. W sztucznej inteligencji istotne są *agenty*, które mają realizować *cele* (podobnie jak agent będący człowiekiem działa na ogólnym poziomie w swoim środowisku), natomiast w uczeniu maszynowym ważne są uczenie się i generalizacja (bardziej przypomina to wewnętrzne funkcjonowanie ludzi). W uczeniu maszynowym rozwiązywane są takie problemy jak klasyfikowanie (rozpoznawanie klas na podstawie danych) i regresja (prognozowanie jednego wyniku na podstawie innego).

Na ogólnym poziomie praktycy z dziedziny uczenia maszynowego używają *danych treningowych* do opracowania *modelu*. Ten model generalizuje w pewien sposób (zależny od używanego modelu) relacje między danymi, aby możliwe było generowanie prognoz na temat nienapotkanych wcześniej danych. Ilustruje to rysunek 1.5. W tym przykładzie dane obejmują trzy *cechy*: A, B i C. Cechy to aspekty danych. Jeśli klasy to „kobiety” i „mężczyźni”, a zadanie polega na podziale grupy na te klasy, można wykorzystać cechy takie jak wzrost, waga i numer buta.



Rysunek 1.5. Przepływ danych w uczeniu maszynowym. Dane służą do treningu modelu, który można następnie zastosować do nowych danych. Tu schemat ilustruje klasyfikowanie. Schematy obrazujące klastrowanie i regresję wyglądają podobnie

W danych treningowych relacje między *cechami* i *klasami* są znane. W modelu należy ująć te relacje. Po zakończeniu treningu model można zastosować do nowych danych, których klasa jest nieznana.

### 1.5.3. Analityka predykcyjna

Analityka predykcyjna nie jest tak szeroko opisywana w literaturze akademickiej jak sztuczna inteligencja i uczenie maszynowe. Jednak wraz z dojrzewaniem architektur przetwarzania dużych zbiorów danych i rosnącym apetytem na operacyjne wykorzystanie danych i uzyskanie dzięki nim dodatkowej wartości dziedzina ta zyskuje na popularności. Na potrzeby tej książki używamy przedstawionej poniżej definicji, rozbudowującej definicję *analityki* ze słownika oksfordzkiego. Dodany został fragment wyróżniony kursywą:

Analityka predykcyjna: systematyczna obliczeniowa analiza danych lub statystyk *w celu tworzenia modeli predykcyjnych*.

Możesz zadać pytanie: „Jak różni się to od uczenia maszynowego, które też dotyczy predykcji?”. To dobre pytanie. Ogólnie techniki uczenia maszynowego mają pomóc zrozumieć i zgeneralizować strukturę oraz relacje w zbiorze danych. W analityce predykcyjnej ważne jest generowanie ocen, rankingów i predykcji dotyczących przyszłych danych i trendów, często w środowisku biznesowym lub operacyjnym. Choć to porównanie może wydawać się niejasne, warto zauważać, że omawiane tu klasy inteligentnych algorytmów w dużym stopniu się pokrywają oraz nie są konkretne i ściśle.

Co ciekawe, analitycy zwykle nie tworzą rozwiązań z obszaru analityki predykcyjnej. W analityce predykcyjnej ważne jest tworzenie modeli, które na podstawie informacji potrafią reagować szybko i wydajnie, generując przydatne dane wyjściowe prognozujące przyszłe zjawiska. Takie systemy często są tworzone przez inżynierów oprogramowania i naukowców zajmujących się danymi. Sytuację dodatkowo komplikuje to, że w modelach analityki predykcyjnej używane są czasem techniki uczenia maszynowego i sztucznej inteligencji!

### PRZYKŁADY Z OBSZARU ANALITYKI PREDYKCYJNEJ

Aby pomóc Ci intuicyjnie zrozumieć ten obszar, przedstawiamy kilka przykładowych rozwiązań z dziedziny analityki predykcyjnej. Pierwszy pochodzi ze świata reklamy internetowej. Uważni użytkownicy internetu zapewne zauważali, że reklamy często „podążają za nimi”, gdy przeglądają różne witryny. Jeśli wcześniej oglądałeś buty na stronie sklepu Nike, na innych stronach często zobaczysz reklamy tego właśnie obuwia! Jest to tak zwany *retargeting*. Za każdym razem, gdy wczytywana jest strona z reklamami, wiele różnych jednostek podejmuje setki decyzji, chcąc wyświetlić Ci określone reklamy. System wymiany reklamy działa w ten sposób, że każda jednostka podaje cenę, jaką jest gotowa zapłacić za wyświetlenie Ci reklamy. Oferent najwyższej kwoty wygrywa prawo do pokazania reklamy. Ponieważ cały proces musi zachodzić w ciągu milisekund, cenę ustala inteligentny algorytm starający się przewidzieć lub ocenić wartość danego użytkownika. To rozwiązanie z obszaru analityki predykcyjnej podejmuje decyzje na podstawie wcześniejszych zachowań użytkownika i daje korzyści w porównaniu z losowym doborem odbiorców reklam. Do tego przykładu wróćmy w rozdziale 5., gdzie zobaczysz, jak problem ten rozwiązało wiele firm reklamujących się w internecie.

Drugi przykład pochodzi z dziedziny kredytów konsumpcyjnych. Za każdym razem, gdy starasz się o kredyt na karcie stałego klienta, karcie kredytowej, u dostawcy telefonii komórkowej lub o kredyt hipoteczny, sprzedawca naraża się na pewne ryzyko, a także może odnieść określone korzyści. Aby zrównoważyć te aspekty, sprzedawcy chcą wiedzieć, że udzielają kredytów zwłaszcza godnym zaufania osobom, a odmawiają klientom, którzy z większym prawdopodobieństwem nie będą spłacać dłużu. W praktyce podejmowanie takich decyzji zlecane jest wyspecjalizowanym agencjom ratingowym, które za opłatą przekazują sprzedawcy informacje o zdolności kredytowej danej osoby. Ocena jest generowana przez rozwiązywanie z obszaru analityki predykcyjnej na podstawie danych historycznych dla danej populacji. Ta ocena to liczba wysoce skorelowana z ryzykiem dotyczącym danej osoby. Im wyższy wynik, tym bardziej godny zaufania jest klient i tym mniejsze ryzyko niespłacenia kredytu. Zauważ, że to podejście daje tylko przewagę statystyczną, ponieważ osoby o wysokiej ocenie kredytowej też mogą zaprzestać spłaty (choć — jeśli model działa — zdarza się to rzadziej niż w przypadku klientów o niższej ocenie).

## 1.6. Ocena działania inteligentnych algorytmów

Do tej pory pisaliśmy o ogólnych klasach inteligentnych algorytmów i przedstawiliśmy kilka przykładów. Jak jednak praktyk z tej dziedziny może ocenić swój algorytm? Ma to duże znaczenie i to z kilku przyczyn. Po pierwsze, bez obiektywnej oceny nie da się śledzić wyników i ustalić, czy modyfikacje ulepszyły rozwiązanie. Po drugie, jeśli nie da się mierzyć wyników, trudno jest uzasadnić sens stosowania rozwiązania. W kontekście biznesowym menedżerowie i technologowie zawsze będą starali się uwzględnić zyski i koszty, a możliwość solidnej oceny rozwiązania pomaga zachować je w środowisku produkcyjnym.

Dalej wracamy do poszczególnych klas inteligentnych algorytmów i omawiamy strategie ich oceny. Choć poruszamy tu kwestię oceny inteligencji, ten podrozdział dotyczy głównie oceny predykcji (związanych z uczeniem maszynowym i analityką predykcyjną). Ocena i definiowanie inteligencji to obszerne tematy, które zasługują na odrębną książkę. Dlatego zamiast omawiać je w tym miejscu, odsyłamy Czytelników do książek Lindy Gottfredson<sup>14</sup>, Jamesa Flynna<sup>15</sup> i Alana Turinga<sup>16</sup>.

### 1.6.1. Ocena inteligencji

Wcześniej wspomnialiśmy o teście Turinga. Czasem jednak trzeba ocenić systemy, którym stawiane są mniej ambitne cele — na przykład inteligentne systemy grające w szachy lub biorące udział w teleturnieju Jeopardy! Takie systemy nie potrafią naśla-

<sup>14</sup> Linda S. Gottfredson, *Mainstream Science on Intelligence: An Editorial with 52 Signatories, History, and Bibliography*, „Wall Street Journal”, 13 grudnia 1994.

<sup>15</sup> James R. Flynn, *What Is Intelligence? Beyond the Flynn Effect* (Cambridge University Press, 2009).

<sup>16</sup> Alan Turing, *Computing Machinery and Intelligence*.

dować człowieka, ale świetnie radzą sobie w pojedynczych zadaniach. Zaproponowany przez Sandeepa Rajaniego sposób oceny sztucznej inteligencji<sup>17</sup> obejmuje cztery poziomy:

- *Optymalny* — niemożliwe jest uzyskanie lepszych wyników.
- *Zdecydowanie lepszy od człowieka* — działanie lepsze niż jakiegokolwiek człowieka.
- *Lepszy od człowieka* — działanie lepsze niż większości ludzi.
- *Gorszy od człowieka* — działanie gorsze niż większości ludzi.

Na przykład obecny poziom sztucznej inteligencji pozwala tworzyć systemy optymalne w grze w kółko i krzyżyk, lepsze od człowieka (lub nawet zdecydowanie lepsze od człowieka) w szachach i gorsze od człowieka w tłumaczeniu tekstów w językach naturalnych.

### **1.6.2. Ocena predykcji**

Choć sztuczna inteligencja jest interesująca, większość rozwiązań z tej książki nie dotyczy tej dziedziny. Dlatego należy poszukać bardziej adekwatnych sposobów oceny. Pamiętaj, że w uczeniu maszynowym i analityce predykcyjnej celem jest generowanie prognoz na podstawie danych oraz relacji między cechami i docelowymi wartościami (klasami). Istnieje więc konkretny schemat oceny i można zastosować statystykę do formalnego pomiaru wyników.

W tabeli 1.1 przedstawiony jest (skrajnie uproszczony) zbiór danych, który posłuży do zilustrowania pomiaru skuteczności predyktora. Cechy danych są opisane za pomocą liter alfabetu, a obok przedstawione są wartości logiczne oznaczające rzeczywiste i prognozowany wynik. Przymij, że predykcje zostały wygenerowane na podstawie zbioru danych testowych, których pierwotny model nie znał. W zbiorze danych testowych rzeczywiste wyniki były ukryte, dlatego model musiał ustalić dane wyjściowe wyłącznie na podstawie cech.

**Tabela 1.1.** Przykładowy zbiór danych używany do przedstawienia sposobu oceny intelligentnego algorytmu

A	B	...	Rzeczywisty wynik	Predykcja
10	4	...	Prawda	Fałsz
20	7	...	Prawda	Prawda
5	6	...	Fałsz	Fałsz
1	2	...	Fałsz	Prawda

Od razu widać, że do oceny działania tego klasyfikatora można zastosować kilka prostych miar. Przede wszystkim można badać współczynnik predykcji prawdziwie pozytywnych (ang. *true positive rate* — TPR), czyli łączną liczbę predykcji prawdziwie pozytywnych podzieloną przez łączną liczbę wartości pozytywnych w całym zbiorze danych. Miarę tę nazywa się czasem *czułością* (ang. *sensitivity* lub *recall*). Zauważ jednak, że jest to tylko połowa obrazu! Jeśli zbudujesz klasyfikator, który zawsze generuje

<sup>17</sup> Sandeep Rajani, *Artificial Intelligence — Man or Machine*, „International Journal of Information Technology and Knowledge Management” 4, nr 1 (2011): 173 – 76.

pozytywną prognozę (niezależnie od cech w danych), uzyskasz bardzo wysoką czułość. Dlatego tę miarę trzeba analizować razem z innym wskaźnikiem — *swoistością* (ang. *specificity*; inaczej współczynnik predykcji prawdziwie negatywnych, ang. *true negative rate* — **TNR**). Określa on liczbę predykcji prawdziwie negatywnych podzieloną przez łączną liczbę wartości negatywnych w całym zbiorze. Idealny klasyfikator uzyskuje TPR i TNR na poziomie 100%.

Niestety, większość klasyfikatorów jest daleka od doskonałości, dlatego ich skuteczność trzeba oceniać na podstawie błędów. Współczynnik predykcji fałszywie pozytywnych (ang. *false positive rate* — **FPR**) to 1 minus TNR, natomiast współczynnik predykcji fałszywie negatywnych (ang. *false negative rate* — **FNR**) to 1 minus TPR. Są to tak zwane *błędy pierwszego rodzaju* i *błędy drugiego rodzaju*. W tabeli 1.2 przedstawione są relacje między opisanymi miarami.

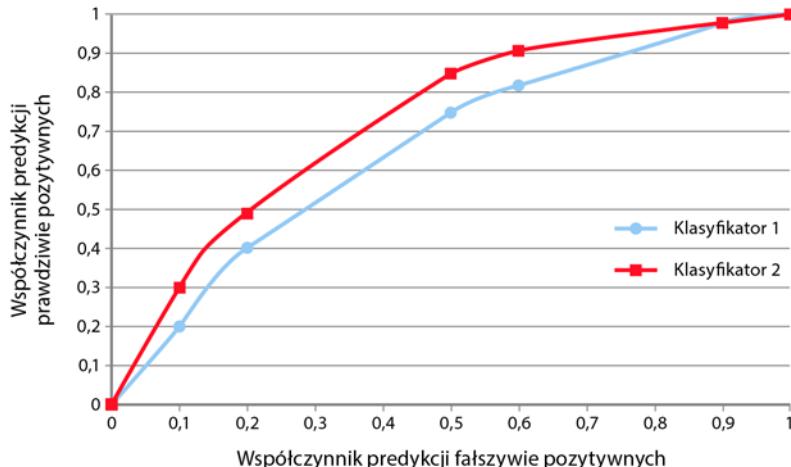
**Tabela 1.2.** Miary skuteczności używane do oceny inteligentnych algorytmów

Miara	Obliczenia
Współczynnik predykcji prawdziwie pozytywnych (TPR)	Predykcje prawdziwie pozytywne / rzeczywiste wyniki pozytywne
Współczynnik predykcji prawdziwie negatywnych (TNR)	Predykcje prawdziwie negatywne / rzeczywiste wyniki negatywne
Współczynnik predykcji fałszywie pozytywnych (FPR)	1 – współczynnik predykcji prawdziwie negatywnych
Współczynnik predykcji fałszywie negatywnych (FNR)	1 – współczynnik predykcji prawdziwie pozytywnych

Aby utrwały te informacje, zastosuj miary z tabeli 1.2 do zbioru danych z tabeli 1.1. Jeśli starannie zastosujesz definicje, stwierdzisz, że TPR i TNR są równe 1/2, z czego wynika, że wartości FPR i FNR są takie same.

Założymy teraz, że algorytm zawiera wewnętrzny mechanizm dostosowujący jego działania — na przykład „pokrętło” dostosowujące czułość algorytmu. Gdy pokrętło jest ustawione na zero, algorytm ma bardzo niską czułość i klasyfikuje wszystkie przypadki jako fałszywe ( $TNR = 1$ ,  $TPR = 0$ ). Natomiast po ustawieniu pokrętła na 11 algorytm uznaje wszystkie przypadki za prawdziwe ( $TNR = 0$ ,  $TPR = 1$ ). Oczywiście żadna z tych sytuacji nie jest pożądana. Optymalny jest klasyfikator, który generuje wynik pozytywny tylko dla pozytywnych elementów i wynik negatywny tylko dla negatywnych elementów ( $TNR = 1$ ,  $TPR = 1$ ). Taki idealny mechanizm jest możliwy wyłącznie dla skrajnie uproszczonych problemów. Możesz jednak zobaczyć, jak rozwiązanie działa dla różnych ustawień pokrętła, i wykorzystać wyniki do oceny algorytmu (zobacz rysunek 1.6).

Na rysunku 1.6 pokazane są krzywe **ROC** (ang. *receiver operating characteristic*) dla dwóch fikcyjnych klasyfikatorów. Te krzywe ilustrują miary TPR i FPR dla różnych ustawień wyimaginowanego pokrętła, które zmienia parametry klasyfikatorów. Wcześniej wspomnieliśmy, że dla idealnego klasyfikatora  $TPR = 0$  i  $TNR = 0$  ( $FPR = 0$ ). Dlatego modele zbliżone do lewego górnego rogu wykresu są teoretycznie lepsze, ponieważ lepiej radzą sobie z wyodrębnianiem klas pozytywnej i negatywnej. Na pokazanym wykresie klasyfikator 2. jest skuteczniejszy i należy go wybrać zamiast klasyfi-



**Rysunek 1.6.** Krzywe ROC dwóch fikcyjnych klasyfikatorów. Po zmodyfikowaniu parametru algorytmu widoczna jest zmiana w jego działaniu dla określonego zbioru danych. Im bliżej krzywa znajduje się lewego górnego rogu, tym klasyfikator jest bliższy ideału, ponieważ  $TPR = 1$  i  $TNR = 1$  ( $TNR = 1 - FPR$ )

katora 1. Inny sposób oceny skuteczności to obliczenie powierzchni pod krzywą ROC (jest to powierzchnia AUC — od ang. *area under the curve*, czyli powierzchnia pod krzywą). Im większa jest powierzchnia AUC, tym wyższa skuteczność modelu.

Do tej pory wszystko wygląda sensownie. Wiedz jednak, że w rozwiązańach z obszaru analityki predykcyjnej sytuacja nie zawsze jest tak prosta. Pomyśl na przykład o ocenach kredytowych. Gdy wydasz ocenę, nie zawsze możesz prześledzić, jak dany klient będzie się zachowywał w przyszłości. Ponadto w takich scenariuszach nie próbujesz udzielić odpowiedzi; celem jest określenie wartości skorelowanej z docelową zmienną (na przykład z wiarygodnością kredytobiorcy). Choć w takich sytuacjach można posłużyć się krzywą ROC, czasem trzeba pokonać kilka dodatkowych przeszkód.

## 1.7. Ważne uwagi na temat inteligentnych algorytmów

Do tej pory omówiliśmy już wiele wprowadzającego materiału. Na tym etapie powinieneś dobrze (choć na ogólnym poziomie) rozumieć inteligentne algorytmy i wiedzieć, jak się nimi posługiwać. Zapewne jesteś niecierplwy i zmotywowany, aby przejść do szczegółów. Nie zawiedziemy Cię. Każdy następny rozdział jest bogaty w nowy i wartościowy kod. Jednak zanim rozpocznesz podróż po ekscytyującym i (dla bardziej cynicznych programistów) atrakcyjnym finansowo świecie inteligentnych aplikacji, powinieneś zapoznać się z listą przydatnych informacji. Liczne z nich zapozyczylismy z doskonałej i przystępnej pracy Pedro Domingosa<sup>18</sup>. Te informacje będą pomocne w czasie lektury tej książki, a także później, w trakcie pracy w dziedzinie inteligentnych algorytmów.

<sup>18</sup> Pedro Domingos, *A Few Useful Things to Know About Machine Learning*, „Communications of the ACM” 55, nr 10 (2012): 78 – 87.

### 1.7.1. Dane nie są wiarygodne

Dane z wielu powodów mogą być niewiarygodne. To dlatego zawsze powinieneś sprawdzać, czy używane dane są godne zaufania. Dopiero potem możesz zacząć zastanawiać się nad rozwiązyaniem problemu za pomocą inteligentnych algorytmów. Nawet inteligentni ludzie dochodzą zwykle do błędnych wniosków, jeśli posługują się nieprawidłowymi danymi. Poniżej znajduje się przydatna, choć niepełna lista źródeł problemów z danymi:

- Dane dostępne w trakcie rozwijania rozwiązania mogą być niereprezentatywne dla danych ze środowiska produkcyjnego. Założmy, że chcesz dzielić użytkowników sieci społecznościowej według wzrostu na grupy: „wysocy”, „przeciętni” i „niscy”. Jeśli najniższa osoba w Twoim zespole programistycznym ma 184 centymetry wzrostu, ryzykujesz tym, że nazwiesz kogoś niskim, ponieważ mierzy „tylko” 184 centymetry.
- W danych mogą występować braki. W rzeczywistości, jeśli dane nie są sztucznie generowane, prawie na pewno będą niepełne. Obsługa braku wartości to skomplikowane zadanie. Zwykle albo pozostawia się lukę w wartościach, albo zapelnia ją domyślnymi lub obliczonymi wartościami. Oba podejścia mogą prowadzić do niestabilnych rozwiązań.
- Dane mogą się zmieniać. Ktoś może zmodyfikować schemat bazy danych lub zmienić znaczenie przechowywanych w niej danych.
- Dane mogą być nieznormalizowane. Założmy, że analizujesz wagę grupy osób. Aby można było dojść do znaczących wniosków na podstawie wagi, jednostki miary dla wszystkich osób powinny być takie same. W całym zbiorze trzeba używać albo funtów, albo kilogramów, a nie jednej jednostki dla części osób i drugiej dla pozostałych.
- Dane mogą być niedostosowane do algorytmicznego podejścia, jakie planujesz zastosować. Dane przyjmują różne postacie i formy, którym odpowiadają *typy danych*. Niektóre zbiory danych są liczbowe, inne nie. Niektóre zbiory danych można porządkować, inne tego nie umożliwiają. Niektóre liczbowe zbiory danych są nieciągłe (na przykład liczba osób w pomieszczeniu), natomiast inne — ciągłe (na przykład temperatura lub ciśnienie atmosferyczne).

### 1.7.2. Wnioskowanie wymaga czasu

Obliczanie rozwiązania zajmuje czas, a szybkość reagowania aplikacji może być krytyczna dla odniesienia przez firmę finansowego sukcesu. Nie powinieneś przyjmować, że wszystkie algorytmy dla wszystkich zbiorów danych będą działały na tyle szybko, by aplikacja mogła udzielić odpowiedzi w ustalonym limicie czasu. Należy przetestować szybkość algorytmu z uwzględnieniem charakterystyki działania aplikacji.

### 1.7.3. Wielkość ma znaczenie!

W kontekście inteligentnych aplikacji wielkość ma znaczenie! Wielkość danych jest istotna w dwóch aspektach. Pierwszy związany jest ze wspomnianym wcześniej czasem reagowania. Drugi dotyczy możliwości uzyskania wartościowych wyników dla dużych

zbiorów danych. Możliwe, że aplikacja potrafi generować doskonałe rekomendacje filmów lub muzyki dla około 100 użytkowników, ale dla grup około 100 000 osób zwraca mało przydatne wyniki.

Ponadto, co związane jest z „przekleństwem wymiarów” (zobacz rozdział 2.), przekazanie większej ilości danych do prostego algorytmu często daje znacznie lepsze wyniki niż budowanie bardziej skomplikowanego klasyfikatora. Gdy przyjrzyisz się dużym korporacjom (takim jak Google), które wykorzystują bardzo duże ilości danych, powinieneś docenić zarówno umiejętność obsługi dużych zbiorów danych treningowych, jak i złożoność oraz zaawansowanie rozwiązań klasyfikujących.

#### **1.7.4. Różne algorytmy skalują się w odmienny sposób**

Nie zakładaj, że możliwe jest skalowanie intelligentnej aplikacji przez proste dodanie kolejnych maszyn. W ogóle nie powinieneś przyjmować, że rozwiązanie jest skalowalne. Niektóre algorytmy się skalują, natomiast inne nie. Założymy, że wśród miliardów tytułów chcesz znaleźć grupy artykułów informacyjnych z podobnymi nagłówkami. Nie wszystkie algorytmy klastrowania mogą działać równolegle. Powinieneś uwzględnić skalowalność na etapie projektowania aplikacji. W niektórych sytuacjach możliwy jest podział danych i zastosowanie intelligentnego algorytmu równolegle do mniejszych zbiorów danych. Algorytmy wybrane w trakcie projektowania mają czasem wersje równoległe (współbieżne), jednak powinieneś sprawdzić to już na początku prac, ponieważ na podstawie użytych algorytmów tworzona jest rozbudowana infrastruktura i logika biznesowa.

#### **1.7.5. Nie wszystko jest gwoździem!**

Możliwe, że zetknęłeś się już z twierdzeniem: „Jeśli masz tylko młotek, wszystko wygląda jak gwoździe”. Oznacza to, że nie da się za pomocą tego samego algorytmu rozwiązać wszystkich problemów wymagających intelligentnych aplikacji.

Intelligentne aplikacje są jak każde inne oprogramowanie — mają określony obszar zastosowań i pewne ograniczenia. Koniecznie starannie przetestuj swoje ulubione rozwiązanie w nowych obszarach. Ponadto zalecamy, aby każdy problem analizować z nowej perspektywy. Inne algorytmy mogą rozwiązywać określone problemy w wydajniejszy lub bardziej dogodny sposób.

#### **1.7.6. Dane to nie wszystko**

Algorytmy uczenia maszynowego nie działają w magiczny sposób i wymagają indukcji, aby wyjść poza dane treningowe i móc przetwarzać nieznane wcześniej informacje. Jeśli masz już bogatą wiedzę na temat zależności w danych, dobrą reprezentacją mogą być modele graficzne, pozwalające łatwo przedstawić uprzednią wiedzę<sup>19</sup>. Staranne przemyślenie tego, co już wiadomo w danej dziedzinie (z uwzględnieniem danych), pomaga budować skuteczne klasyfikatory.

---

<sup>19</sup>Judea Pearl, *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann Publishers, 1988).

### **1.7.7. Czas treningu może się zmieniać**

W niektórych zastosowaniach czas generowania rozwiązania może się znacznie zmieniać przy tylko niewielkiej zmianie parametrów. Zwykle użytkownicy oczekują, że po zmianie parametrów problemu nadal będzie można go rozwiązać w tym samym czasie. Gdy używasz metody zwracającej odległość między dwoma lokalizacjami geograficznymi na Ziemi, spodziewasz się, że uzyskasz wynik w tym samym czasie niezależnie od uwzględnianych lokalizacji. Jednak nie we wszystkich problemach jest to prawda. Pozornie niewinna zmiana w danych może prowadzić do znacznej zmiany czasu generowania wyniku. Bywa, że ten czas zmienia się z sekund na godziny!

### **1.7.8. Celem jest generalizacja**

Jedną z pułapek, w którą praktycy z obszaru uczenia maszynowego wpadają najczęściej, jest koncentracja na procesie pracy i zapomnienie o ostatecznym celu, a jest nim generalizacja analizowanego zjawiska. W fazie testów niezbędne jest stosowanie metod, które pozwalają ocenić ogólność rozwiązania (pomijanie danych testowych w czasie treningu, walidacja krzyżowa itd.). Nic jednak nie zastąpi używania od początku odpowiedniego zbioru danych! Jeśli próbujesz uogólnić proces o milionie atrybutów za pomocą kilkuset przykładów testowych, skupianie się na osiągnięciu wysokiej trafności nie ma żadnego sensu.

### **1.7.9. Ludzka intuicja nie zawsze się sprawdza**

Gdy przestrzeń cech rośnie, następuje eksplozja kombinatoryczna, jeśli chodzi o liczbę możliwych wartości wejściowych. Dlatego już dla umiarkowania rozbudowanego zbioru cech możesz zobaczyć tylko bardzo niewielki ulamek wszystkich możliwych danych wejściowych. Bardziej problematyczne jest to, że wraz z rosnącą liczbą cech ludzka intuicja zaczyna zawodzić. Na przykład większość masy w wielowymiarowym rozkładzie normalnym znajduje się nie blisko średniej, ale w „otoczone” wokół niej<sup>20</sup>. Budowanie prostego klasyfikatora dla niewielkiej liczby wymiarów jest łatwe, jednak przy większej liczbie wymiarów trudno jest zrozumieć zależności w danych.

### **1.7.10. Pomyśl o zaprojektowaniu nowych cech**

Prawdopodobnie zatknąłeś się ze stwierdzeniem: „śmieci na wejściu, śmieci na wyjściu”. Ma ono duże znaczenie w trakcie budowania rozwiązań z obszaru uczenia maszynowego. Bardzo istotne jest tu zrozumienie dziedziny problemowej, a ustalenie zestawu cech uwidaczniającego badane zjawisko może mieć duży wpływ na trafność i ogólność klasyfikatora. Nie wystarczy przekazać klasyfikatorowi wszystkich dostępnych danych i liczyć na cud.

### **1.7.11. Poznaj wiele różnych modeli**

Coraz popularniejsze stają się zestawy modeli, ponieważ pozwalają ograniczyć zmienność w procesie klasyfikacji kosztem tylko niewielkiego błędu systematycznego. W konkursie Netflix Prize pierwsze i drugie miejsce zajęły zestawy warstwowe (w których dane

---

<sup>20</sup>Domingos, A Few Useful Things to Know About Machine Learning.

wyjściowe każdego klasyfikatora są przekazywane do klasyfikatora wyższego poziomu) obejmujące ponad 100 jednostek uczących się. Wiele osób uważa, że dzięki takim technikom w przyszłości klasyfikatory będą skuteczniejsze, jednak to podejście powoduje utworzenie nowej warstwy pośredniej, którą osoba niebędąca ekspertem musi poznać, aby zrozumieć funkcjonowanie systemu.

### **1.7.12. Korelacja nie oznacza związku przyczynowo-skutkowego**

Warto przypomnieć tę często przytaczaną kwestię. Można też zilustrować ją za pomocą żartobliwego eksperymentu myślowego: „Globalne ocieplenie, trzęsienia ziemi, huragany i inne katastrofy naturalne są bezpośrednim efektem zmniejszania się liczby piratów od XIX wieku”<sup>21</sup>. To, że dwie zmienne są skorelowane, nie oznacza ich połączenia związkiem przyczynowo-skutkowym. Często istnieje trzecia (a nawet czwarta lub piąta!) nieobserwowały zmienne, która wpływa na pozostałe. Korelację należy traktować jak oznakę możliwej przyczynowości zasługującą na dalsze analizy.

## **1.8. Podsumowanie**

- Przedstawiliśmy tu bardzo ogólny obraz inteligentnych algorytmów i zaprezentowaliśmy wiele przykładowych rozwiązań używanych w praktyce.
- Inteligentny algorytm cechuje się tym, że potrafi modyfikować swoje działanie na podstawie otrzymywanych danych.
- Pokazaliśmy kilka antywzorców projektowych dotyczących inteligentnych algorytmów. Mamy nadzieję, że posłużą one jako ostrzeżenie dla praktyków z tej dziedziny.
- Wyjaśniliśmy, że inteligentne algorytmy można ogólnie podzielić na trzy kategorie: sztuczna inteligencja, uczenie maszynowe i analityka predykcyjna. W tej książce najczęściej miejsca poświęcamy dwóm ostatnim z tych klas. Dlatego jeśli tylko pobiernie przejrzałeś poświęcone im fragmenty, powinieneś ponownie się z nimi zapoznać, aby zapewnić sobie solidną podstawową wiedzę.
- Przedstawiliśmy wybrane podstawowe miary takie jak krzywa ROC. Obszar pod krzywą ROC często jest używany do oceny względnej skuteczności modeli. Pamiętaj jednak, że istnieje wiele sposobów oceny skuteczności. Tu pokazaliśmy tylko podstawowe z nich.
- Zaprezentowaliśmy też przydatne informacje zdobyte przez społeczność pracującą nad inteligentnymi algorytmami. Będą one nieocenionym kompasem w trakcie podróży po tym obszarze.

---

<sup>21</sup> Bobby Henderson, *An Open Letter to Kansas School Board*, Verganza (1 lipca 2005), <http://www.venganza.org/about/open-letter>.





# *Wydobywanie struktury z danych — klastrowanie i transformacja danych*

---

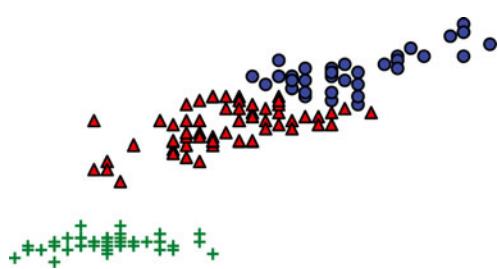
## **Zawartość rozdziału:**

- Cechy i przestrzeń cech
- Maksymalizacja wartości oczekiwanej — sposób trenowania algorytmów
- Transformacja osi danych w celu lepszego reprezentowania danych

W poprzednim rozdziale zapoznałeś się z pojęciem „inteligentne algorytmy”. Od obecnego rozdziału koncentrujemy się na działaniu algorytmów z obszarów uczenia maszynowego i analityki predykcyjnej. Jeśli kiedyś zastanawiałeś się, jakie rodzaje algorytmów są dostępne i jak one działają, dowiesz się tego z tych rozdziałów.

Ten rozdział jest poświęcony strukturze danych, czyli określaniu, czy w używanym zbiorze danych występują opisujące go określone wzorce i reguły. Oto przykład: dostępny jest zbiór danych o populacji z nazwami stanowisk pracy, wiekiem i wynagrodzeniami. Czy występują w nim ogólne reguły lub wzorce pozwalające uprościć dane? Na przykład czy wyższy wiek koreluje z wyższym wynagrodzeniem? Czy duży procent bogactwa jest skumulowany w niewielkim procencie populacji? Po znalezieniu takich uogólnień można je bezpośrednio przedstawić, aby albo pokazać dowód na występowanie

wzorca, albo zapisać zbiór danych w mniejszym, bardziej zwięzłym pliku. Te dwa przykłady są tematem tego rozdziału, a rysunek 2.1 przedstawia wizualną reprezentację uogólnienia danych.



**Rysunek 2.1.** Wizualizowanie struktury w danych. Wartości na osiach  $x$  i  $y$  są tu dowolne i nieistotne. W danych widoczne są trzy różne klasy reprezentowane za pomocą odmiennych kolorów. Widac tu, że wartości  $x$  i  $y$  występują w klastrach skupionych w określonych obszarach wykresu odpowiadających poszczególnym klasom. Widac też, że niezależnie od klasy wyższe wartości  $x$  korelują z wyższymi wartościami  $y$ . Każda z tych właściwości określa strukturę danych i jest opisana w dalszej części rozdziału

### **UWAGA DLA CZYTELNIKÓW WERSJI DRUKOWANEJ — KOLOROWE RYSUNKI**

Wiele rysunków z tej książki najlepiej jest oglądać w kolorze (<ftp://ftp.helion.pl/przyklady/intsci2.zip>).

Omawianie tematu rozpoczynamy od objaśnienia podstawowych pojęć i zdefiniowania znaczenia struktury danych. Omawiamy też błąd systematyczny i szum, które mogą w nieoczekiwany sposób wpływać na zebrane dane. Znajdziesz tu także omówienie „przekleństwa wymiarów” (ang. *curse of dimensionality*) i przestrzeni cech. Te informacje pomogą w analizie relacji między liczbą cech w danych, liczbą punktów danych i zjawiskiem, które próbujemy uchwycić za pomocą inteligentnego algorytmu.

Dalej omawiamy konkretną metodę ujmowania struktury, *klastrowanie*, polegającą na grupowaniu punktów danych na podstawie podobieństwa między nimi. Odbywa się to podobnie jak kolorowanie danych na rysunku 2.1. Klastrowanie ma wiele zastosowań, ponieważ użytkownicy często chcą sprawdzać, które elementy są podobne do siebie. Firma chce na przykład wykrywać podobnych klientów w nadziei, że będą dokonywać zakupów w podobny sposób. Może to być przydatne do identyfikowania najlepszych klientów i rozszerzania bazy użytkowników o podobne im osoby. Niezależnie od przyczyn klastrowania można je przeprowadzać na wiele sposobów. Tu koncentrujemy się na metodzie *k-średnich* (ang. *k-means*; inna nazwa to algorytm centroidów). Polega ona na podziale danych wejściowych na  $k$  różnych obszarów (klastrów). Choć teoretycznie jest to proste zadanie, algorytm k-średnich jest często używany w praktyce i prezentujemy tu cały kod potrzebny do opracowania własnej implementacji z użyciem pakietu scikit-learn. W ramach omawiania algorytmu k-średnich pokazujemy, jak określone są klastry za pomocą iteracyjnego algorytmu przeprowadzania treningu — *maksymalizacji wartości oczekiwanej* (ang. *expectation maximization — EM*). Do tej ważnej techniki wrócimy w ramach omawiania drugiej metody klastrowania z użyciem gaussowskiego modelu mieszanego (ang. *Gaussian mixture model — GMM*).

Używanie algorytmu GMM do klastrowania można traktować jak rozwinięcie algorytmu k-średnich. Za pomocą obu technik można uzyskać dokładnie te same wyniki, jeśli w algorytmie GMM zastosowane zostaną określone ograniczenia. Szczegółowo omawiamy to zagadnienie w dalszych punktach. Na razie możesz uznać, że algorytm

GMM to bardziej ekspresywny model, w którym przyjmowane i ujmowane są dodatkowe założenia dotyczące rozkładu punktów danych. Do trenowania tego modelu też używany jest algorytm EM. Dalej badamy i omawiamy, w jaki sposób i dlaczego te dwie wersje algorytmu EM różnią się między sobą.

W ostatnim podrozdziale omawiamy ważną metodę badania struktury danych i redukowania łącznej liczby cech w zbiorze danych bez poświęcania dostępnych w danych informacji. To zagadnienie związane jest ze zrozumieniem współzmienności zmiennych w zbiorze danych. Na przykład na rysunku 2.1 niezależnie od klasy danych wartości x zwiększą się wraz ze wzrostem wartości y. Omawiana tu metoda to *analiza głównych składowych* (ang. *principal component analysis — PCA*). Używana samodzielnie pozwala ona wykryć główne kierunki zmienności w danych i pomaga zrozumieć, które cechy są istotne, a które nie. Można ją też stosować do odwzorowania danych na mniejszą przestrzeń cech z tylko niewielką utratą informacji. Dlatego metoda ta może być używana do wstępnego przetwarzania w algorytmach klastrowania (lub innych inteligentnych algorytmach). Często pozytywnie wpływa to na czas treningu bez zmniejszania skuteczności algorytmu. Teraz nie przedłużamy już wstępu i wkraczamy do świata danych i struktury.

## 2.1. Dane, struktura, błąd systematyczny i szum

W kontekście tej książki *dane* to dowolne informacje, które można zbierać i przechowywać. Jednak nie wszystkie dane są sobie równe. Na potrzeby inteligentnych algorytmów w sieci WWW dane są zbierane w celu prognozowania zachowań użytkowników w internecie. Warto w tym kontekście wy tłumaczyć pojęcia *struktura, błąd systematyczny i szum*.

Na ogólnym poziomie struktura oznacza niejawne pogrupowanie, uporządkowanie, wzorce lub korelację w danych. Na przykład jeśli zbierzesz wzrost wszystkich mieszkańców Wielkiej Brytanii, prawdopodobnie wykryjesz dwa odrębne zbiory określające wzrost kobiet i mężczyzn. Błąd systematyczny może się pojawić w danych, jeśli osoby rejestrujące dane konsekwentnie podają zaniżone wartości, natomiast szum występuje, gdy te same osoby są niestaranne i nieprecyzyjne przy podawaniu wzrostu.

Aby lepiej zbadać te zagadnienia, przyjrzymy się prawdziwym danym — *zbiorowi danych Iris*. Ten często używany zbiór danych pochodzi z 1936 roku i obejmuje 150 próbek danych dotyczących trzech gatunków kwiatów. Każda próbka obejmuje cztery pomiary (cechy). Zacznijmy od przyjrzenia się danym (zobacz listing 2.1).

**Listing 2.1. Przeglądanie zbioru danych Iris za pomocą pakietu scikit-learn  
(w interaktywnej powłoce)**

```
>>> import numpy as np
>>> from sklearn import datasets
>>>
>>> iris = datasets.load_iris()
>>> np.array(zip(iris.data, iris.target))
array([[array([ 5.1,  3.5,  1.4,  0.2]), 0],
       [array([ 4.9,  3. ,  1.4,  0.2]), 0],
```

← Importowanie zbiorów danych z pakietu scikit-learn.

← Wczytywanie zbioru danych Iris.

← Tworzenie tablicy z danymi i docelowymi wartościami oraz wyświetlanie pierwszych 10 wierszy.

← Wyświetlana jest zawartość pierwszych 10 wierszy z tablicy.

```
[array([ 4.7,  3.2,  1.3,  0.2]), 0],
[array([ 4.6,  3.1,  1.5,  0.2]), 0],
[array([ 5. ,  3.6,  1.4,  0.2]), 0],
[array([ 5.4,  3.9,  1.7,  0.4]), 0],
[array([ 4.6,  3.4,  1.4,  0.3]), 0],
[array([ 5. ,  3.4,  1.5,  0.2]), 0],
[array([ 4.4,  2.9,  1.4,  0.2]), 0],
[array([ 4.9,  3.1,  1.5,  0.1]), 0]], dtype=object)
```

Na listingu 2.1 pokazana jest interaktywna sesja w Pythonie z użyciem pakietu scikit-learn pozwalającą przejrzeć zbiór danych Iris. Po uruchomieniu powłoki Pythona zimportowaliśmy moduł NumPy i pakiet ze zbiorami danych, a następnie wczytaliśmy zbiór danych Iris. Za pomocą metody `zip` utworzyliśmy dwuwymiarową tablicę, w której elementy `iris.data` znajdują się obok elementów `iris.target`, a następnie pobraliśmy pierwszych 10 elementów tablicy.

Czym jednak są elementy `iris.data` i `iris.target`? I co utworzyliśmy? Obiekt `iris` jest typu `sklearn.datasets.base.Bunch`. Jest to obiekt przypominający słownik z zestawem atrybutów (szczegółowe informacje znajdziesz w dokumentacji pakietu scikit-learn). Element `iris.data` to dwuwymiarowa tablica wartości cech, a `iris.target` to jednowymiarowa tablica klas. W każdym wierszu z listingu 2.1 wyświetlana jest tablica atrybutów (*cech*) kwiatu wraz z jego klasą. Co oznaczają te wartości cech i klasy? Każda cecha określa długość i szerokość płatków oraz działek kielicha, natomiast klasa określa gatunek kwiatu. Na listingu 2.2 znajduje się streszczenie tych informacji. Zauważ, że cechy są tu opisane w tej samej kolejności, w jakiej występują na listingu 2.1.

**Listing 2.2. Przeglądanie zbioru danych Iris z użyciem pakietu scikit-learn  
(w interaktywnej powłoce) — ciąg dalszy**

```
>>> print(iris.DESCR)
Iris Plants Database
```

Notes

-----

Data Set Characteristics:

```
:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
    - sepal length in cm
    - sepal width in cm
    - petal length in cm
    - petal width in cm
    - class:
        - Iris-Setosa
        - Iris-Versicolour
        - Iris-Virginica
```

:Summary Statistics:

```
===== ===== ===== ===== ===== =====
      Min   Max   Mean     SD  Class Correlation
===== ===== ===== ===== ===== =====
sepal length:  4.3   7.9   5.84   0.83   0.7826
sepal width:  2.0   4.4   3.05   0.43  -0.4194
```

```
petal length:  1.0  6.9   3.76   1.76   0.9490 (high!)
```

```
petal width:   0.1  2.5   1.20   0.76   0.9565 (high!)
```

```
=====
```

```
:Missing Attribute Values: None
```

```
:Class Distribution: 33.3% for each of 3 classes.
```

```
:Creator: R.A. Fisher
```

```
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
```

```
:Date: July, 1988
```

Za pomocą atrybutu `DESCR` możesz pobrać i wyświetlić w konsoli informacje na temat zbioru danych Iris. Widać tu, że w każdym wierszu typu `iris.data` znajdują się cechy oznaczające długość działki (w centymetrach), szerokość działki (w centymetrach), długość płatka (w centymetrach) i szerokość płatka (w centymetrach) w tej właśnie kolejności. Klasy kwiatów, określone w elemencie `iris.target`, to: Iris-Setosa, Iris-Versicolour i Iris-Virginica. Gdy wróciś do listingu 2.1, zobaczysz, że klasy te są kodowane za pomocą liczb całkowitych. Za pomocą atrybutu `target_names` można łatwo określić, które nazwy odpowiadają poszczególnym liczbom. Atrybut ten przechowuje wartości w kolejności zgodnej z kodującymi je liczbami. Poniższy fragment pokazuje, że 0 odpowiada gatunkowi setosa, 1 to versicolour, a 2 to virginica:

```
>>> iris.target_names
```

```
array(['setosa', 'versicolour', 'virginica'],
```

```
      dtype='|S10')
```

Choć używamy prostego zbioru danych, zagadnienia, które tu poznasz, są uniwersalne dla algorytmów z dziedziny uczenia maszynowego. Dalej w rozdziale zobaczysz, jak za pomocą uczenia maszynowego określić strukturę danych i jak stosować do tego różne techniki.

Aby zilustrować znaczenie struktury, spróbujmy przeprowadzić eksperyment myślowy na zbiorze danych Iris. Możliwe, że wszystkie irisy Virginica są większe od pozostałych — długość i szerokość ich płatków oraz działyki kielicha są wyraźnie większe niż u innych kwiatów. Możliwe też, że któryś z gatunków charakteryzuje się długimi i wąskimi płatkami, natomiast pozostałe mają płatki krótkie i zaokrąglone. Na tym etapie struktura i organizacja danych nie są jeszcze znane. Możesz jednak chcieć zastosować automatyczne techniki do odkrycia tych informacji. To jeszcze nie wszystko — zadaniem dla zautomatyzowanego algorytmu może być wykrycie struktury i ustalenie podgrup lub klastrów oraz przypisanie wszystkich punktów danych do jednego z tych klastrów. Ten proces to *klastrowanie*.

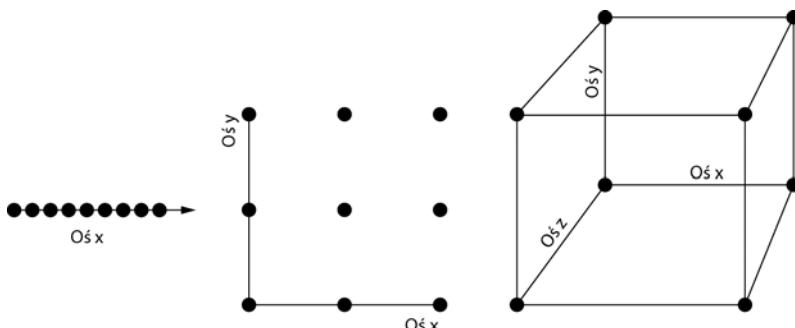
Wcześniej wspomnieliśmy, że nie wszystkie dane są sobie równe. Nawet jeśli założymy, że dostępne pomiary dobrze charakteryzują kwiaty i ujmują unikalowość poszczególnych gatunków, należy zwrócić uwagę na kilka kwestii. Na przykład co zrobić, jeśli dane zostały zebrane na początku okresu kwitnienia i kwiaty nie były rozwinięte do pełnej wielkości? W takiej sytuacji pomiary w systematyczny sposób różnią się od danych dla ogólnej populacji. Występuje wtedy *błąd systematyczny*. Klastry wygenerowane na podstawie takich danych mogą być odpowiednie dla używanego zbioru, ale z powodu różnic względem reszty populacji nie umożliwiają generalizowania.

Ponadto za zbieranie danych mogły odpowiadać różne osoby, które zapewne prowadziły pomiary w odmienny sposób i z różną starannością. To zwiększa poziom losowości w danych, czyli *szum*. Czasem może on powodować, że struktura lub wzorce w danych będą niewidoczne. Aby zmaksymalizować ogólność i skuteczność, w trakcie zbierania danych należy minimalizować błąd systematyczny i szum.

## 2.2. „Przekleństwo wymiarów”

Używany na razie zbiór danych obejmuje tylko cztery cechy. Może się zastanawiasz, dlaczego by nie zebrać setek lub tysięcy pomiarów na temat każdego kwiatu i pozwolić algorytmowi na wykonanie zadania. Nawet jeśli pominiemy praktyczne trudności ze zbieraniem tak wielu pomiarów, występują dwa podstawowe problemy z rejestrowaniem licznych wymiarów, mające istotny wpływ na wykrywanie struktury. Pierwszy problem polega na tym, że duża liczba wymiarów zwiększa przestrzeń, w jakiej rozproszone są punkty danych. Jeśli zachowasz stałą liczbę punktów danych i zwiększasz liczbę atrybutów używanych do ich opisu, gęstość punktów w przestrzeni będzie maleć wykładniczo. Dlatego możesz przez długi czas analizować przestrzeń danych bez możliwości wykrycia w nich zależności trafniejszych od innych możliwości.

Drugi podstawowy problem ma przerażającą nazwę — *przekleństwo wymiarów*. Dotyczy on tego, że dla *dowolnego* zbioru punktów o dużej liczbie wymiarów i *dowolnej* miary odległości między tymi punktami dystans między punktami jest mniej więcej taki sam! Aby zaobserwować ten ważny efekt, warto przyjrzeć się prostemu przykładowi z rysunku 2.2.



Rysunek 2.2. „Przekleństwo wymiarów” — każdy punkt znajduje się w podobnej odległości od pozostałych

Jeśli przyjrzesz się rysunkowi 2.2 od lewej do prawej, zobaczyś, że liczba wymiarów rośnie za każdym razem o jeden. Na początku osiem punktów jest rozłożonych równomiernie w jednym wymiarze (na osi x), na przykład między wartościami 0 i 1. Wtedy minimalna odległość, jaką trzeba pokonać z danego punktu do momentu natrafienia na inny, wynosi  $\min(D) = 0,125$ , natomiast odległość maksymalna to  $\max(D) = 1$ . Stosunek wartości  $\min(D)$  do  $\max(D)$  to 0,125. W przestrzeni dwuwymiarowej osiem punktów też jest rozmiieszczonych równomiernie, jednak teraz  $\min(D) = 0,5$ , a  $\max(D) = 1,414$

(jest to odległość wzdułg głównej przekątnej). Stosunek  $\min(D)$  do  $\max(D)$  wynosi więc 0,354. W przestrzeni trójwymiarowej  $\min(D) = 1$ ,  $\max(D) = 1,732$ , a stosunek  $\min(D)$  do  $\max(D)$  wynosi 0,577. Wraz z rosnącą liczbą wymiarów stosunek odległości minimalnej do maksymalnej dąży do 1. To oznacza, że niezależnie od tego, który kierunek obierzesz i jaką odległość zechcesz zmierzyć, wynik będzie taki sam!

W praktyce oznacza to, że im więcej atrybutów danych zbierasz, tym większa jest przestrzeń możliwych punktów i trudniej jest ustalić podobieństwo między nimi. W tym rozdziale analizujemy strukturę danych. Możesz sądzić, że aby wykryć w danych wzorce i struktury, warto zebrać wiele atrybutów na temat badanego zjawiska. Rzeczywiście tak jest, ale uważaj — liczba punktów danych potrzebnych do generalizowania wzorców rośnie znacznie szybciej niż liczba atrybutów używanych do opisu danego zjawiska. W praktyce trzeba zdecydować się na kompromis. Musisz rejestrować atrybuty dobrze opisujące zjawisko, ale nie w takiej liczbie, by uniemożliwiło to zebranie danych w ilości potrzebnej do generalizacji lub wykrycia wzorców w danej przestrzeni.

## 2.3. Algorytm k-średnich

Algorytm k-średnich dąży do podziału danych na k odrębnych klastrów charakteryzujących się średnimi przypisanych im elementów. Ten efekt często uzyskuje się za pomocą algorytmu Lloyda<sup>1</sup>, który pozwala znaleźć rozwiązanie w iteracyjny sposób. Algorytm ten jest tak popularny, że często to właśnie jego nazywa się *algorytmem k-średnich* lub *centroidów*. Zwykle kiedy ktoś mówi, że stosuje algorytm k-średnich, ma na myśli to, że używa algorytmu Lloyda do rozwiązania problemu k-średnich. Przyjrzyj się teraz prostemu przykładowi zastosowania tego algorytmu w tylko jednym wymiarze. Zastanów się nad osią liczbową z rysunku 2.3.

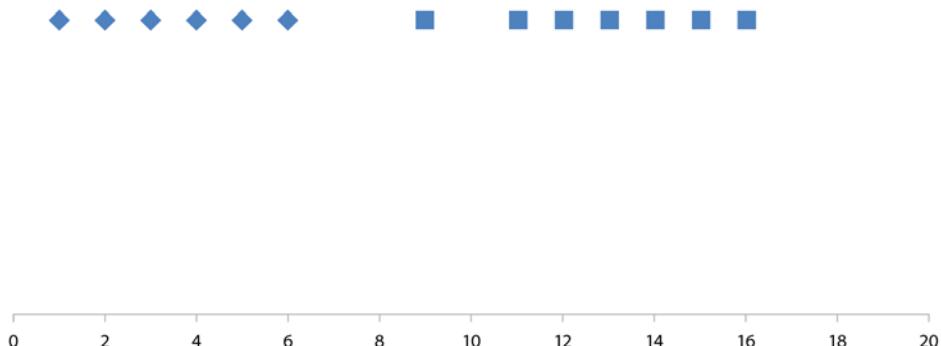
Przedstawiony jest tu zbiór danych obejmujący 13 punktów. Widać, że w danych występuje pewna struktura. Punkty z lewej strony są skupione w przedziale od 0 do 6, natomiast punkty z prawej łączą się w klaster w przedziale od 9 do 16. Nie wiemy, czy są to właściwe klastry, ale intuicyjnie można zaryzykować stwierdzenie, że oba zestawy punktów różnią się na tyle, że można je uznać za odrębne grupy.

Na tym właśnie polega zadanie wszystkich algorytmów klastrowania. Mają one tworzyć grupy punktów danych, które wyznaczają klastry lub struktury w danych. Omówimy ten przykład dokładniej przed przejściem do zastosowania algorytmu k-średnich do rzeczywistych danych. Warto jednak wspomnieć, że ten prosty jednowymiarowy przykład jest wystarczająco ogólny. W praktyce skomplikowane wzorce i struktury są szukane w danych o większej liczbie wymiarów, co oczywiście naraża użytkownika na „przekleństwo wymiarów”.

Algorytm Lloyda iteracyjnie próbuje ustalić średnie dla klastrów ze zbioru danych. Algorytm przyjmuje jako parametr liczbę  $k$ , określającą, ile klastrów ma uzyskać. Może też przyjmować  $k$  szacunkowych średnich dla tych klastrów. Są to wyjściowe średnie

---

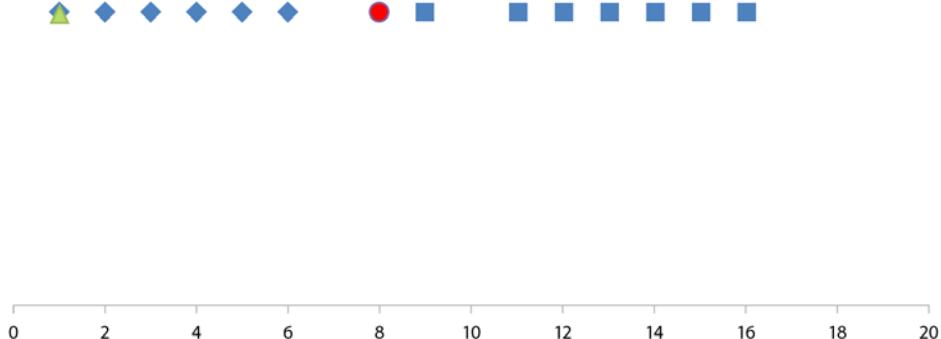
<sup>1</sup> Stuart P. Lloyd, *Least Squares Quantization in PCM*, „IEEE Transactions on Information Theory” 28 (1982): 129 – 137.



**Rysunek 2.3.** Oś liczbową, na której występują dwa możliwe klastry w przestrzeni jednowymiarowej. Elementy oznaczają, że dana liczba znajduje się w zbiorze danych. Punkty danych reprezentowane przez jednakowe figury ilustrują możliwy intuicyjny podział danych na klastry

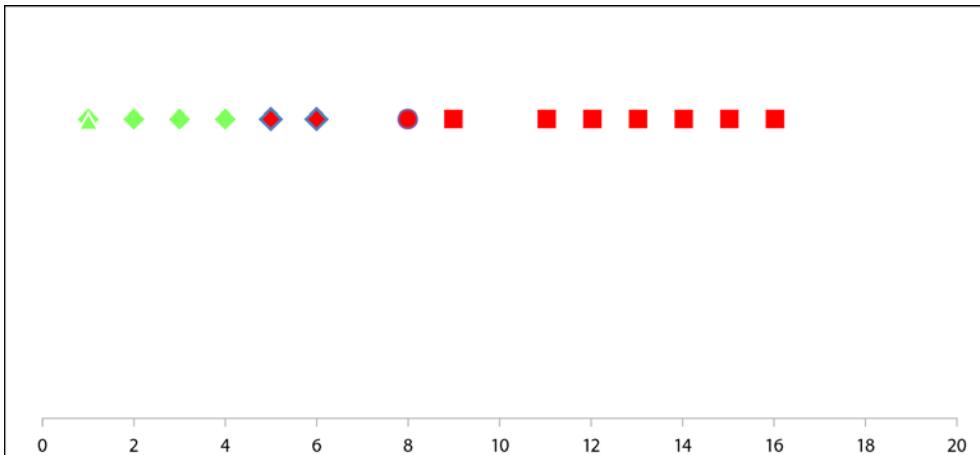
klastrów i nie trzeba się przejmować tym, że na razie są nieprawidłowe. Algorytm iteracyjnie aktualizuje je, aby otrzymać możliwie najlepsze rozwiązanie. Wróćmy teraz do danych z rysunku 2.3 i zobaczymy przykładowy przebieg algorytmu z ich użyciem.

Najpierw założymy, że na razie klastry ani ich średnie nie są znane (jest to uzasadnione założenie w algorytmach klastrowania). Wszystko, co jest dostępne, to 13 punktów danych. Przyjmijmy też, że mamy dobrą intuicję i uważamy, że w danych występują dwa klastry, dlatego zaczynamy od  $k = 2$ . Algorytm działa w następujący sposób: najpierw wyznacza różne od siebie szacunki k średnich. Ustalmy je losowo na 1 (dla  $k_1$ ) i 8 (dla  $k_2$ ). Na rysunku 2.4 są one oznaczone trójkątem i kółkiem. Te wartości to tylko przypuszczenia, które będą później aktualizowane.



**Rysunek 2.4.** Początkowe średnie k klastrów. Średnia klastra  $k_1$  jest oznaczona trójkątem, a średnia klastra  $k_2$  — kółkiem

Teraz każdy punkt ze zbioru danych należy przydzielić do najbliższego mu klastra. W ten sposób wszystkie punkty mniejsze niż 4,5 trafiają do klastra  $k_1$ , a wszystkie punkty większe niż 4,5 zostaną przypisane do klastra  $k_2$ . Ilustruje to rysunek 2.5, na którym punkty danych z klastra  $k_1$  są wyróżnione kolorem zielonym (w drukowanej wersji książki są to romby), natomiast punkty z klastra  $k_2$  są czerwone (są to kwadraty).

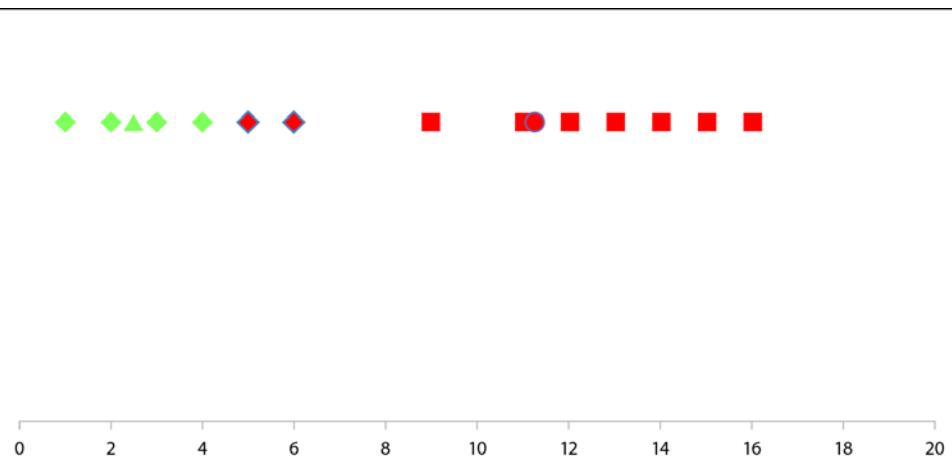


Rysunek 2.5. Początkowy przydział punktów danych do klastrów. Punkty danych przypisane do klastra  $k_1$  są zielone (mają kształt rombów), a punkty z klastra  $k_2$  są czerwone (i są w kształcie kwadratów)

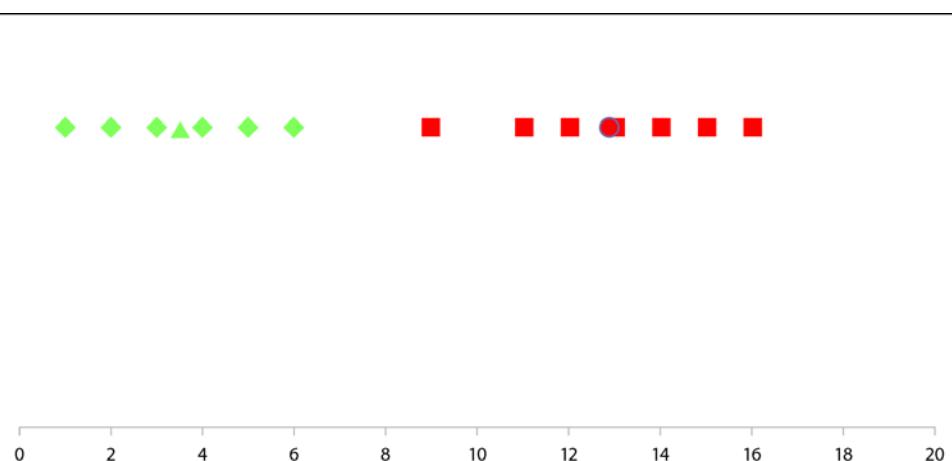
Możesz uznać, że to dobry punkt wyjścia, ale możliwe jest lepsze rozwiązywanie. Jeśli wróćisz do danych, zobaczysz, że punkty poniżej 7 powinny należeć do lewego klastra,  $k_1$ , a punkty powyżej 7 powinny znajdować się w prawym klastrze,  $k_2$ . Aby centroidy klastrów lepiej pasowały do danych, należy obliczyć nowe średnie na podstawie wartości z klastra. Oznacza to, że nowa średnia klastra  $k_1$  to średnia zielonych punktów danych, a nowa średnia klastra  $k_2$  to średnia punktów czerwonych. Nowe średnie wynoszą więc 2,5 ( $k_1$ ) i 11,2 ( $k_2$ ). Zaktualizowane średnie ilustruje rysunek 2.6. Średnie obu klastrów zostały „przyciągnięte” przez przypisane do nich punkty.

Przejdzmy teraz do drugiej iteracji. Tak jak wcześniej każdy punkt danych należy przypisać do klastra o najbliższym punktowi centroidzie. Wszystkie punkty do mniej więcej 6,8 są przydzielane do klastra  $k_1$ , a wszystkie punkty powyżej tej liczby trafiają do klastra  $k_2$ . Jeśli wykonasz te same operacje co wcześniej i obliczysz nowe średnie dla klastrów, otrzymasz centroidy 3,5 i 12,8. Na tym etapie dalsze iteracje nie zmienią już przypisania punktów danych do klastrów, dlatego średnie też pozostaną takie same. Można stwierdzić, że algorytm *osiągnął konwergencję*<sup>2</sup>. Ostateczne przypisanie elementów przedstawia rysunek 2.7.

<sup>2</sup> To samo określenie jest używane w kontekście uczenia dowolnego algorytmu. Gdy funkcja kosztów osiąga wartość minimalną, mówi się, że algorytm treningowy osiągnął konwergencję. W tej książce zwykle polegamy na tym, że biblioteki z pakietu scikit-learn zadbały o osiągnięcie konwergencji w trakcie treningu.



**Rysunek 2.6.** Zaktualizowane średnie dla elementów przypisanych do klastrów. Centroidy są aktualizowane, aby lepiej reprezentowały punkty danych przydzielone do klastrów



**Rysunek 2.7.** Ostateczne przypisanie punktów danych do klastrów. Końcowe centroidy klastrów są przedstawione za pomocą trójkąta i kółka. Punkty danych mniejsze niż 6,8 zostały przydzielone do klastra k<sub>1</sub>, a większe punkty danych znalazły się w klastrze k<sub>2</sub>. Dodatkowe iteracje algorytmu k-średnich nie zmienią podziału na klastry

Zauważ, że ostateczne przypisanie daje dokładnie te średnie, które uzyskalibyśmy w wyniku ręcznego klastrowania (wizualnego określenia klastrów i obliczenia ich średnich). Tak więc algorytm automatycznie ustalił inteligentny podział danych na klastry z tylko minimalnym udziałem programisty.

Warto jednak zauważyć, że choć można udowodnić, iż algorytm osiąga konwersję dla odległości euklidesowej<sup>3</sup>, nie da się zagwarantować uzyskania globalnego minimum. Dlatego ostateczny podział nie zawsze jest optymalny. Algorytm może być

<sup>3</sup> Leon Bottou i Yoshua Bengio, *Convergence Properties of the K-Means Algorithms*, „Advances in Neural Information Processing Systems” 7 (1994).

wrażliwy na początkowe warunki i wykrywać różne średnie klastrów (odpowiadające minimum lokalnym). Dlatego w aplikacji można zastosować różne punkty początkowe i przeprowadzić trening kilkakrotnie. Jedną z technik przybliżających rozwiązań do globalnych minimów jest użycie średniej dla każdej wynikowej grupy centroidów jako początkowych centroidów w końcowym przebiegu. Może to zapewnić lepszą pozycję wyjściową do osiągnięcia optymalnego podziału na klastry, nie można jednak tego zagwarantować.

Na listingu 2.3 przedstawiony jest pseudokod ilustrujący opisane kroki. W pierwszym kroku należy zainicjować centroidy (jest ich  $k$  — tyle, ile zdefiniowano przy uruchamianiu algorytmu). Dalej kod uruchamia pętlę, z której wychodzi po osiągnięciu konwergencji. W poprzednim przykładzie intuicyjnie mogliśmy dostrzec konwergencję, ponieważ centroidy klastrów i przypisane do nich punkty danych przestały się zmieniać. Jednak w praktyce nie zawsze tak się dzieje. Częściej sprawdzane są zmiany średnich. Jeśli zmiany są mniejsze od określonego progu, można bezpiecznie przyjąć, że zmienia się przypisanie tylko niewielkiej liczby punktów, tak więc algorytm osiągnął konwergencję.

#### Listing 2.3. Pseudokod algorytmu k-średnich — maksymalizacja wartości oczekiwanej

Inicjowanie centroidów

while(nie nastąpiła konwergencja centroidów):

**Krok związany z wartością oczekiwana.**

Dla każdego punktu danych przypisz etykietę oznaczającą najbliższy centroid. ←

Oblicz centroidy na podstawie przypisanych do nich punktów danych. ←

**Krok związany z maksymalizacją.**

W kroku związanym z wartością oczekiwana do każdego punktu danych przypisywany jest centroid oddalony od tego punktu o najmniejszą odległość euklidesową. W kroku związanym z maksymalizacją ponownie obliczane są centroidy na podstawie przypisanych do nich punktów danych. Każdy wymiar z przestrzeni cech jest uśredniany w celu uzyskania nowego centroidu.

To tworzy pętlę: dane są przypisywane do klastra, klaster jest dostosowywany do powiązanych z nim danych, dane ponownie są przydzielane do klastra itd. Konwergencja zachodzi, gdy podział danych na klastry przestaje się zmieniać. Opisaliśmy tu określoną implementację algorytmu Lloyda i, bardziej ogólnie, klasy algorytmów *maksymalizacji wartości oczekiwanej*. Dalej dla wygody posługujemy się nazwą algorytm k-średnich. Do maksymalizacji wartości oczekiwanej wracamy w dalszych punktach tego rozdziału.

### **2.3.1. K-średnie w praktyce**

Zobacz teraz, jak algorytm k-średnich działa dla zbioru danych o większej liczbie wymiarów — dla przedstawionego wcześniej zbioru danych Iris. Kod z listingu 2.4 wczytuje zbiór danych Iris i uruchamia implementację algorytmu k-średnich z pakietu scikit-learn. Kod wyświetla na ekranie uzyskane klastry danych, po czym kończy pracę.

Ten prosty przykładowy kod przyjmuje zbiór danych Iris i używa algorytmu k-średnich do sprawdzenia, czy można wykryć wewnętrzne wzorce w danych, jeśli nie istnieją rzeczywiste informacje określające gatunki kwiatów. Kod ustala więc, czy algorytm

## 50 Rozdział 2. Wydobywanie struktury z danych — klastrowanie i transformacja danych

### Listing 2.4. Algorytm k-średnich w praktyce

```
from sklearn.cluster import KMeans
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data
km = KMeans(n_clusters=3)
km.fit(X)

print(km.labels_)
```

potrafi wykryć strukturę dzielącą zbiór danych Iris na trzy różne klastry. Spróbuj uruchomić ten kod. Zobaczysz, że algorytm rzeczywiście rozdziela dane na trzy klastry! To świetna wiadomość, jednak więcej informacji uzyskalibyś, gdybyś mógł przyjrzeć się klastrom razem z danymi. Listing 2.5 ilustruje kod wizualizujący wyniki.

### Listing 2.5. Wizualizowanie danych wyjściowych algorytmu k-średnich

```
from sklearn.cluster import KMeans
from sklearn import datasets
from itertools import cycle, combinations
import matplotlib.pyplot as pl

iris = datasets.load_iris()
km = KMeans(n_clusters=3)
km.fit(iris.data)

predictions = km.predict(iris.data)

colors = cycle('rgb')
labels = ["Klaster 1", "Klaster 2", "Klaster 3"]
targets = range(len(labels))

feature_index=range(len(iris.feature_names))
feature_names=iris.feature_names
combs=combinations(feature_index,2)

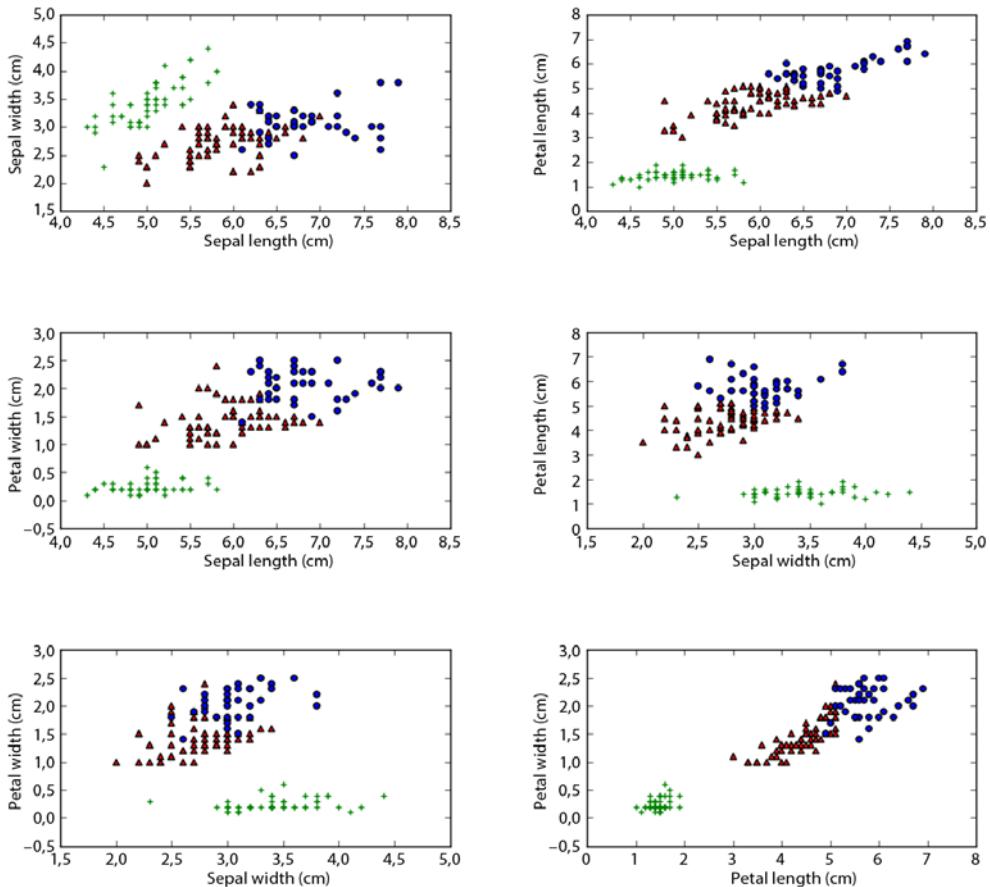
f,axarr=pl.subplots(3,2)
axarr.flat=axarr.flat

for comb, axflat in zip(combs,axarr.flat):
    for target, color, label in zip(targets,colors,labels):
        feature_index_x=comb[0]
        feature_index_y=comb[1]

        axflat.scatter(iris.data[predictions==target,feature_index_x],
                      iris.data[predictions==target,feature_index_y],c=color,label=label)
        axflat.set_xlabel(feature_names[feature_index_x])
        axflat.set_ylabel(feature_names[feature_index_y])

f.tight_layout()
pl.show()
```

Kod z tego listingu generuje wszystkie dwuwymiarowe kombinacje ze zbioru cech irysów. Ilustruje to rysunek 2.8. Kolory i kształty punktów danych oznaczają klastry (wykryte za pomocą algorytmu k-średnich), do których określone punkty należą.



**Rysunek 2.8.** Klastrowanie zbioru danych Iris metodą k-średnich. Wykresy to wynik uruchomienia kodu z listingu 2.5. Zauważ, że k-średnie wyznaczają centroidy klastrów w przestrzeni  $n$ -wymiarowej, gdzie  $n$  to liczba cech (w zbiorze danych Iris wynosi ona cztery). Na potrzeby wizualizacji wyświetliśmy wszystkie kombinacje cech

Na tym rysunku zaprezentowane są wszystkie kombinacje cech irysów, co pozwala ocenić jakość klastrowania. Zauważ, że każdy punkt jest przypisany do klastra o najbliższym temu punktowi centroidzie w przestrzeni czterowymiarowej. Na wszystkich pokazanych dwuwymiarowych wykresach kolory są używane w spójny sposób.

Z krótkiego przeglądu wykresów wynika, że najlepszym sposobem wizualizacji tych danych jest wykres długości i szerokości płatków (prawy dolny). Wydaje się, że te dwie zmienne są wysoce skorelowane. Dlatego uwzględnienie tylko jednej z tych cech może pozwolić na podział danych na trzy klastry. Jeśli rzeczywiście tak jest, jeśli jedna cecha wystarcza do podziału kwiatów, to czy potrzebne są wszystkie cztery? To doskonale pytanie!

Krótką odpowiedź jest taka, że prawdopodobnie nie potrzebujemy wszystkich cech. Jednak w trakcie zbierania danych nie było to wiadome. Znając wyniki, wiadomo, że można było uwzględnić inne dane. Jednak w trakcie ich zbierania nie znaliśmy wyników.

W podrozdziale 2.6 wracamy do tego problemu i proponujemy rozwiązanie w postaci analizy głównych składowych. Ta technika pozwala wygenerować dla danych nowe osie, których kierunki pozwalają uzyskać maksymalną zmienność w danych. Na razie nie jest to jednak istotne. Możesz traktować tę technikę jak sposób na przekształcenie wszystkich cech w jedną lub dwie zaprojektowane tak, by jak najlepiej opisywać dane.

Najpierw jednak wróćmy do tematu tworzenia modeli. Chcemy zaprezentować Ci inny algorytm podziału oparty na modelowaniu. Ta metoda, w odróżnieniu od k-średnich, wykorzystuje rozkład do utworzenia modelu danych. Ponieważ parametry rozkładu trzeba dopiero poznać, model tego rodzaju to *model parametryczny*. Dalej wyjaśniamy, jak różni się on od modelu nieparametrycznego uzyskanego za pomocą k-średnich i w jakich warunkach oba algorytmy dają analogiczne wyniki. Analizowany tu algorytm wykorzystuje rozkład Gaussa i jest nazywany *gaussowskim modelem mieszanym* (ang. *Gaussian mixture model* — GMM).

## 2.4. Gaussowski model mieszany

W poprzednim podrozdziale dokonaliśmy podziału na klastry za pomocą k-średnich. Jest to jedna z najczęściej używanych technik klastrowania. Ta nieparametryczna metoda wykrywania struktury ma bardzo przydatne cechy, dzięki którym idealnie sprawdza się w wielu sytuacjach. W tym podrozdziale omawiamy metodę parametryczną, w której używany jest rozkład Gaussa. Ta metoda to gaussowski model mieszany. W obu podejściach uczenie odbywa się z użyciem algorytmu maksymalizacji wartości oczekiwanej, dlatego w dalszych punktach szczegółowo go opisujemy.

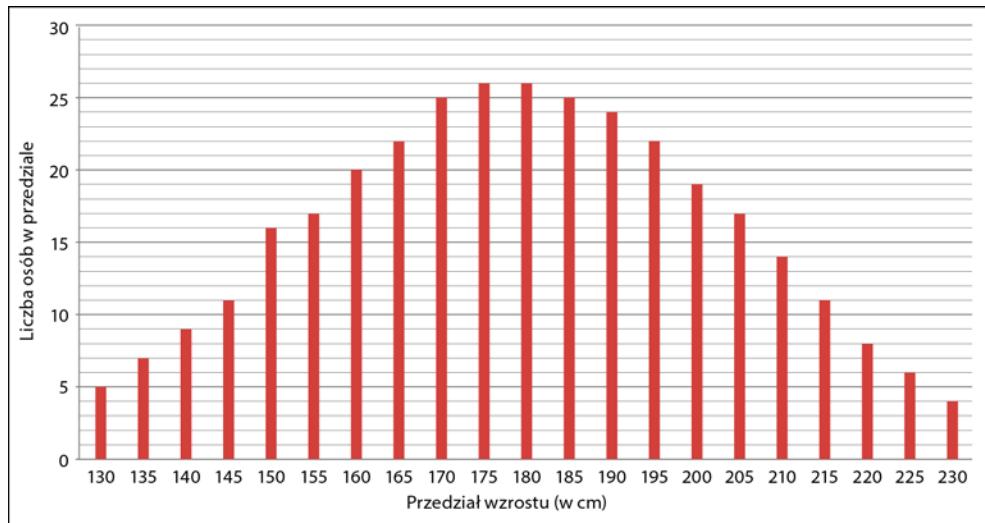
W niektórych sytuacjach technikę k-średnich można wykorzystać do opisu GMM i na odwrót. W k-średnich punkt danych jest bezpośrednio przydzielany do klastra o najbliższym centroidem. Przyjmuję się tu założenie, że skale klastrów są zblizone oraz że w przestrzeni cech nie występują bardzo długie i krótkie klastry (kowariancja cech w klastrach jest podobna). To dlatego przed zastosowaniem k-średnich często warto przeprowadzić normalizację danych. W GMM wspomniane ograniczenia nie występują. Dzieje się tak, ponieważ ta technika modeluje kowariancję cech dla każdego klastra. Nie martw się, jeśli wydaje Ci się to skomplikowane. W dalszych punktach szczegółowo omawiamy te zagadnienia.

### 2.4.1. Czym jest rozkład Gaussa?

Możliwe, że słyszaleś o rozkładzie Gaussa (rozkładzie normalnym). Czym dokładnie jest taki rozkład? Dalej przedstawiamy jego definicję matematyczną, jednak w kategoriach jakościowych można wytlumaczyć, że jest to rozkład występujący naturalnie i bardzo często.

Posłużmy się prostym przykładem. Jeśli pobierzesz losową dużą próbkę osób z populacji, zmierzysz ich wzrost i utworzysz wykres z liczbą osób z określonych przedziałów wzrostu, otrzymasz histogram wyglądający mniej więcej tak jak na

rysunku 2.9. Ten przykładowy i całkowicie fikcyjny zbiór danych pokazuje, że wśród 334 zmierzonych osób najczęściej odnotowany przedział obejmuje 2,5 cm po obu stronach od punktu 180 cm.



Rysunek 2.9. Histogram rozkładu Gaussa. Tu dotyczy on wzrostu 334 fikcyjnych osób. Modalny (najczęściej występujący) przedział ma środek w punkcie 180 cm

Wygląda to na rozkład Gaussa. Jednak aby to sprawdzić, należy zastosować wzór definiujący ten rozkład i metodą prób i błędów zbadać, czy dane są z nim zgodne. Gęstość prawdopodobieństwa rozkładu Gaussa jest określana następującym wzorem:

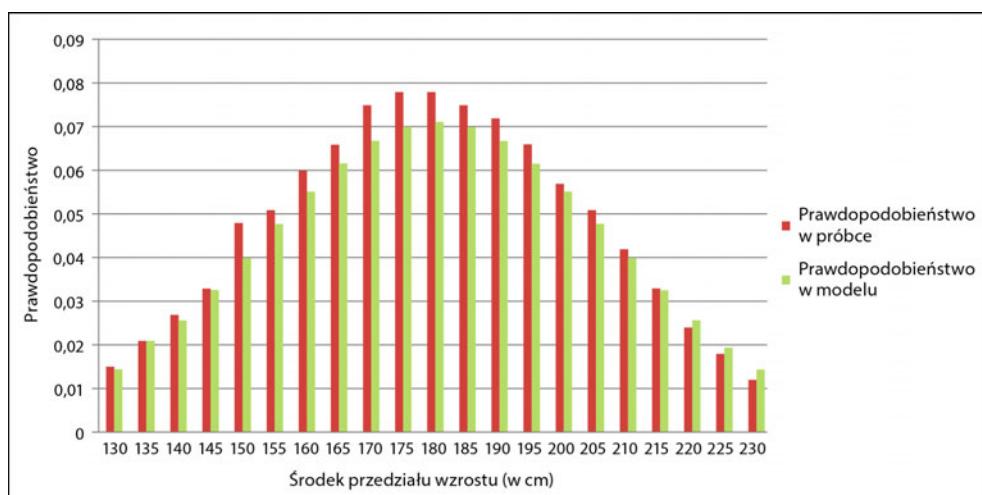
$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Występują tu dwa parametry, które algorytm musi ustalić, aby dopasować równanie do danych. Są to: średnia określana przez parametr  $\mu$  i odchylenie standardowe określane przez parametr  $\sigma$ . Średnia to miara środka rozkładu. Tu można przyjąć, że wynosi ona około 180 cm. Odchylenie standardowe to miara rozkładu danych względem środka. W rozkładzie Gaussa 95% danych zawsze znajduje się w granicach dwóch odchyleń standardowych od średniej. Można więc przyjąć, że ten parametr wynosi tu od 25 do 30, ponieważ większość zebranych danych znajduje się w przedziale od 120 do 240 cm.

Pamiętaj, że przedstawione równanie określa funkcję gęstości prawdopodobieństwa. Jeśli podstawisz wartość za  $x$ , to dla określonego zbioru parametrów funkcja zwróci gęstość prawdopodobieństwa. Zanim jednak przejdziesz do implementowania rozwiązania, warto wspomnieć o jednej kwestii. Rozkład prawdopodobieństwa jest znormalizowany w taki sposób, że obszar pod krzywą jest równy 1. Jest to konieczne do zapewnienia, że zwracane wyniki znajdują się w przedziale dopuszczalnych wartości prawdopodobieństwa. Tu oznacza to tyle, że jeśli chcesz wyznaczyć prawdopodobieństwo wystąpienia wartości z danego przedziału, musisz obliczyć obszar pod krzywą między

dwoma skrajnymi wartościami. Zamiast obliczać ten obszar bezpośrednio, można łatwo uzyskać ten sam wynik, odejmując wyniki kumulatywnej funkcji gęstości dla wartości  $x$  definiujących dany przedział. Jest to dopuszczalne, ponieważ kumulatywna funkcja gęstości dla danej wartości  $x$  zwraca prawdopodobieństwo wystąpienia wartości mniejszej od  $x$  lub jej równej.

Wróćmy teraz do pierwotnego przykładu, aby sprawdzić, czy możliwe jest nieformalne oszacowanie parametrów dla zebranych danych. Założymy, że chcemy zdefiniować przedziały danych w kategoriach prawdopodobieństwa. Każdemu przedziałowi odpowiada prawdopodobieństwo, że przy losowym wyborze jednej z 334 osób znajdzie się ona w danym przedziale. Pożądany efekt można uzyskać, dzieląc liczby z osi y przez liczbę zbadanych osób (334), aby otrzymać prawdopodobieństwo. Te dane ilustrują czerwone słupki, widoczne po lewej stronie w każdej parze na rysunku 2.10. Jeśli teraz zastosujesz kumulatywną funkcję gęstości z parametrami  $\mu = 180$  i  $\sigma = 28$  do obliczenia prawdopodobieństwa wystąpienia wartości z tych przedziałów, będziesz mógł wizualnie zbadać trafność modelu.



Rysunek 2.10. Fikcyjne dane o wzroście 334 osób. Czerwonym kolorem (po lewej stronie każdej pary) pokazane jest prawdopodobieństwo wystąpienia wybranej osoby w określonym przedziale w próbce. Kolorem zielonym (po prawej stronie par) oznaczone jest prawdopodobieństwo oparte na modelu gaussowskim dla parametrów  $\mu = 180$  i  $\sigma = 28$

Szybkie spojrzenie na rysunek 2.10 pokazuje, że początkowe szacunki 180 dla średniej i 28 dla odchylenia standardowego nie są zle. Odchylenie standardowe wydaje się sensowne, choć można je trochę zmniejszyć. Oczywiście możemy ręcznie dostosowywać parametry, aby dopasować je do danych, ale znacznie lepiej będzie zrobić to algorytmicznie. Taki proces to *trening modelu*. Można w nim wykorzystać algorytm maksymalizacji wartości oczekiwanej, co pokazujemy w następnym punkcie.

Trzeba tu dokonać ważnego rozróżnienia na rozkład w próbce i rozkład w populacji. Zakładamy tu, że dane dotyczące 334 użytkowników są reprezentatywne dla całej populacji. Przymajemy też, że dane mają rozkład Gaussa i że próbka została uzyskana

na ich podstawie. Dlatego najlepszym założeniem jest oszacowanie, że dane mają właśnie ten rozkład, jednak należy pamiętać, że jest to tylko oszacowanie! Opiera się ono na założeniu, że wraz ze zbieraniem coraz większej liczby próbek rozkład wysokości będzie zbliżał się do rozkładu Gaussa. Zawsze jednak pozostaje element niepewności. Celem treningu modelu jest minimalizacja tej niepewności przy określonych założeniach dotyczących modelu.

#### **2.4.2. Maksymalizacja wartości oczekiwanej i rozkład Gaussa**

W poprzednim punkcie przedstawiliśmy intuicyjny sposób dopasowywania modelu gaussowskiego do danego zbioru. W tym punkcie prezentujemy to podejście w bardziej formalnym ujęciu, tak aby możliwe było algorytmiczne dopasowywanie modelu do danych. W tym celu używane wcześniej intuicyjnie podejście zastosujemy w bardziej ścisły sposób. Przy określaniu, czy dopasowanie modelu do danych jest dobre, sprawdzaliśmy, czy prawdopodobieństwo w próbce jest zgodne z prawdopodobieństwem w modelu. Następnie można zmienić model, tak aby nowy lepiej pasował do prawdopodobieństwa w próbce. Ten proces należy powtarzać do momentu, w którym prawdopodobieństwa są bardzo zbliżone. Wtedy proces można zatrzymać i uznać, że model zakończył trening.

Teraz dokładnie to samo zrobimy za pomocą algorytmu. Jednak sprawdzanie dopasowania musi opierać się na czymś innym niż ocena wizualna. Używane tu podejście polega na ustaleniu prawdopodobieństwa, że dany model wygeneruje dostępne dane. Obliczana jest więc wartość oczekiwana danych na podstawie modelu. Następnie algorytm spróbuje zmaksymalizować wartość oczekiwana, modyfikując parametry modelu ( $\mu$  i  $\sigma$ ). Proces można iteracyjnie powtarzać do momentu, w którym w każdym cyklu parametry zmieniają się o bardzo niewielką wartość. Zwróci uwagę na podobieństwo tej techniki do algorytmu k-średnich. W przykładzie z k-średnimi na etapie maksymalizacji aktualizowane były tylko centroidy, natomiast w modelu gaussowskim aktualizowane są dwa parametry: średnia i wariancja rozkładu.

#### **2.4.3. Gaussowski model mieszany**

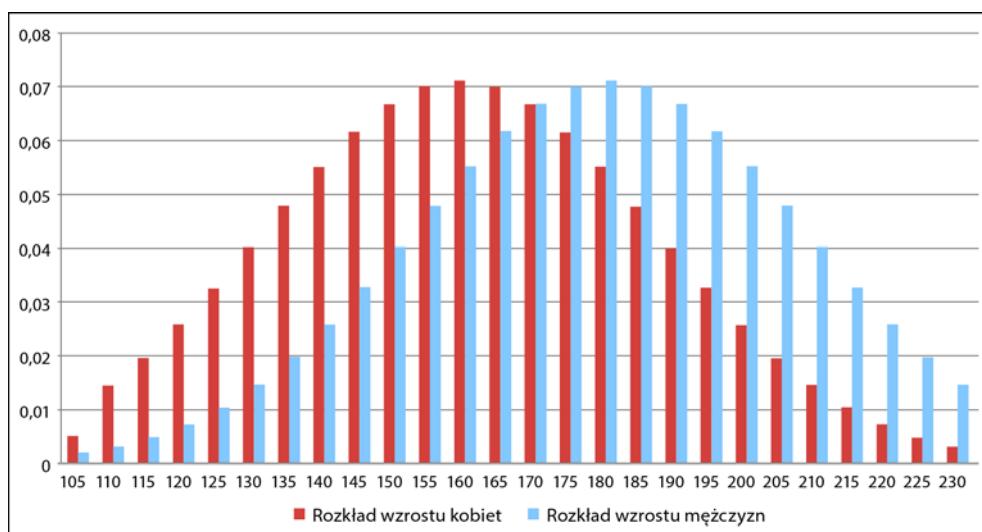
Skoro wiesz już, czym jest rozkład Gaussa, pora przejść do omówienia gaussowskiego modelu mieszanego. Jest to proste rozwinięcie modelu gaussowskiego, w którym do utworzenia modelu rozkładu danych używany jest zestaw rozkładów Gaussa. Zilustrujmy to za pomocą konkretnego przykładu. Założymy, że zamiast losowo mierzyć wzrost osób, chcemy uwzględnić w modelu wzrost mężczyzn i kobiet.

Jeśli przyjmiemy, że dana osoba jest albo mężczyzną, albo kobietą, przedstawiony wcześniej rozkład Gaussa to wynik pobrania próbek z dwóch odrębnych i nachodzących na siebie rozkładów Gaussa. Dlatego zamiast modelować przestrzeń za pomocą jednego rozkładu Gaussa, możemy zastosować dwa takie rozkłady (lub większą ich liczbę):

$$p(x) = \sum_{i=1}^K \phi_i \frac{1}{\sqrt{2\sigma_i^2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$$

To równanie jest bardzo podobne do równania przedstawionego wcześniej w tym rozdziale. Różni się od niego tylko kilkoma szczegółami. Przede wszystkim zwróć uwagę na to, że zamiast jednego rozkładu Gaussa do wyznaczania prawdopodobieństwa używanych jest K odrębnych rozkładów. Każdy z nich ma własne parametry  $\mu$  i  $\sigma$ . Ponadto pojawił się parametr  $\varphi_i$ , określający wagę każdego rozkładu. Wagi muszą być dodatnie, a ich suma musi wynosić 1. To gwarantuje, że równanie przedstawia funkcję gęstości prawdopodobieństwa. Dlatego po scaleniu funkcji w przestrzeni danych wejściowych wynik będzie równy 1<sup>4</sup>.

Wróćmy do omawianego przykładu. Wzrost kobiet może być określony rozkładem Gaussa z niższą średnią niż dla mężczyzn. W ten sposób uzyskujemy funkcję tworzącą opartą na dwóch rozkładach (zobacz rysunek 2.11).



Rysunek 2.11. Rozkład prawdopodobieństwa wzrostu dla kobiet i mężczyzn. Zauważ, że musi być tu znana płeć poszczególnych osób. Na przykład jeśli wiadomo, że dana osoba jest kobietą, prawdopodobieństwo, że jej wzrost znajduje się w danym przedziale, można sprawdzić na osi y

Posługiwaniem się prawdopodobieństwem określonym na osi y na rysunku 2.11 wymaga znajomości płci poszczególnych osób. Jednak takie dane nie zawsze są dostępne (możliwe, że nie zostały zapisane w trakcie pomiarów). Wtedy trzeba ustalić nie tylko parametry każdego rozkładu, ale też podzielić próbkę na podstawie płci ( $\varphi_i$ ). Wartość oczekiwana można obliczyć, mnożąc prawdopodobieństwo, że dana osoba to mężczyzna, przez prawdopodobieństwo wystąpienia wzrostu z danego przedziału dla mężczyzn, oraz dodając do tego wynik podobnej operacji wykonanej na rozkładzie wzrostu kobiet.

Zauważ, że choć ten proces jest bardziej skomplikowany, do treningu modelu można wykorzystać opisaną wcześniej technikę. Gdy określasz wartość oczekiwana (prawdopodobieństwo, że dane zostaną wygenerowane za pomocą modelu mieszanego), możesz

<sup>4</sup> Inaczej niż w poprzednim wzorze, gdzie parametry były jawnie podane po lewej stronie równania, tu parametry zostały pominięte (niejawnie zakładamy, że są określone).

wykorzystać równanie przedstawione na początku punktu 2.4.3. Potrzebna jest tylko strategia aktualizowania parametrów w celu maksymalizacji wartości oczekiwanej. Zastosujmy implementację z pakietu scikit-learn do zbioru danych i zobaczymy, w jaki sposób obliczać zaktualizowane parametry.

#### 2.4.4. Przykład uczenia z użyciem gaussowskiego modelu mieszaneego

W pokazanym wcześniej prostym przykładzie używane były jednowymiarowe rozkłady Gaussa (dane obejmowały tylko jedną cechę). Jednak rozkład Gaussa nie musi być jednowymiarowy. Stosunkowo łatwo można zastąpić średnią wektorem, a wariancję — macierzą kowariancji. Pozwala to tworzyć  $n$ -wymiarowe rozkłady Gaussa uwzględniające dowolną liczbę cech. Aby zademonstrować tę możliwość w praktyce, przejdźmy od przykładu ze wzrostem ponownie do przykładu ze zbiorzem danych Iris, obejmującym w sumie cztery cechy. W następnych punktach omawiamy implementację gaussowskiego modelu mieszaneego z użyciem pakietu scikit-learn (zobacz listing 2.6) oraz wizualizację uzyskanych klastrów.

##### Listing 2.6. Klastrowanie zbioru danych Iris z użyciem gaussowskiego modelu mieszanego

```
from sklearn.mixture import GMM
from sklearn import datasets
from itertools import cycle, combinations
import matplotlib as mpl
import matplotlib.pyplot as pl
import numpy as np.

# Metoda make_ellipses pochodzi ze strony: http://scikit-
# learn.org/stable/auto_examples/mixture/plot_gmm_classifier.html#example-
# mixture-plot-gmm-classifier-py
# Autor: Ron Weiss <ronweiss@gmail.com>, Gael Varoquaux
# Licencja: trzyklauzulowa licencja BSD

def make_ellipses(gmm, ax, x, y):
    for n, color in enumerate('rgb'):
        row_idx = np.array([x,y])
        col_idx = np.array([x,y])
        v, w = np.linalg.eigh(gmm._get_covars()[n][row_idx[:,None],col_idx])
        u = w[0] / np.linalg.norm(w[0])
        angle = np.arctan2(u[1], u[0])
        angle = 180 * angle / np.pi # Przekształcanie na stopnie
        v *= 9
        ell = mpl.patches.Ellipse(gmm.means_[n, [x,y]], v[0], v[1],
                                   180 + angle, color=color)
        ell.set_clip_box(ax.bbox)
        ell.set_alpha(0.5)
        ax.add_artist(ell)

iris = datasets.load_iris()

gmm = GMM(n_components=3, covariance_type='full', n_iter=20)
gmm.fit(iris.data)
```

```

predictions = gmm.predict(iris.data)

colors = cycle('rgb')
labels = ["Klaster 1", "Klaster 2", "Klaster 3"]
targets = range(len(labels))

feature_index=range(len(iris.feature_names))
feature_names=iris.feature_names
combs=combinations(feature_index,2)

f,axarr=plt.subplots(3,2)
axarr_flat=axarr.flat

for comb, axflat in zip(combs,axarr_flat):
    for target, color, label in zip(targets,colors,labels):
        feature_index_x=comb[0]
        feature_index_y=comb[1]
        axflat.scatter(iris.data[predictions==target,feature_index_x],
                      iris.data[predictions==target,feature_index_y],c=color,label=label)
        axflat.set_xlabel(feature_names[feature_index_x])
        axflat.set_ylabel(feature_names[feature_index_y])
        make_ellipses(gmm,axflat,feature_index_x,feature_index_y)

plt.tight_layout()
plt.show()

```

Większość kodu z listingu 2.6 jest identyczna z kodem z listingu 2.5, gdzie wizualizacja dotyczyła danych wyjściowych z algorytmu k-średnich. Oto istotne różnice:

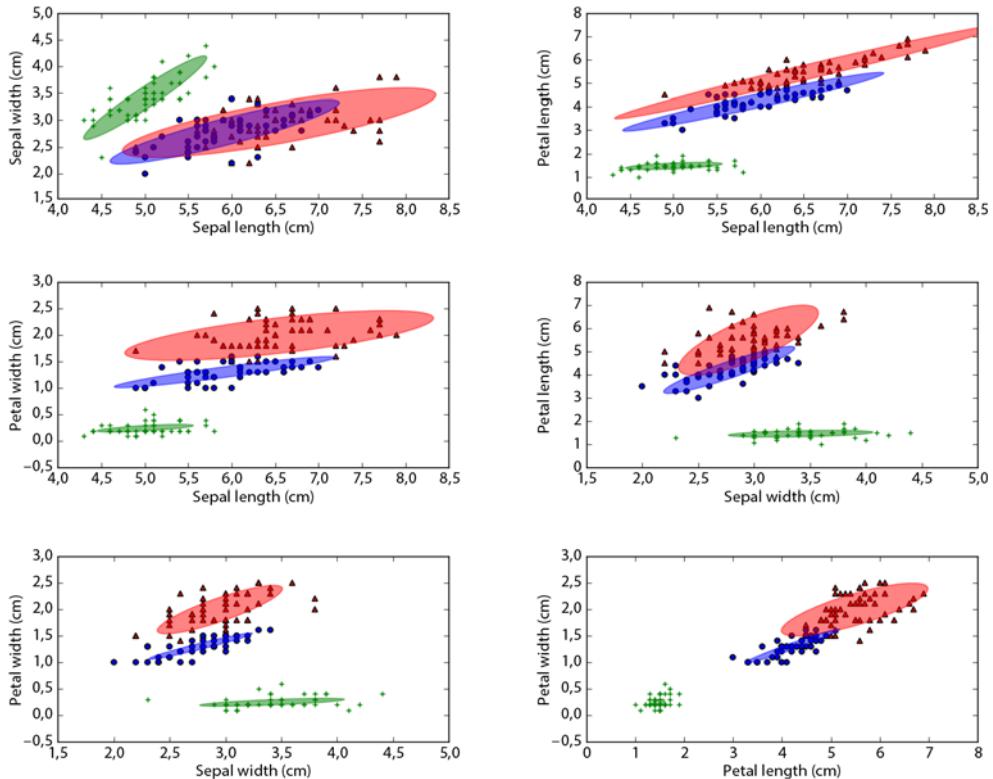
- Używany jest algorytm `sklearn.mixture.GMM` zamiast `sklearn.cluster.KMeans`.
- Tu definiowana i używana jest metoda `make_ellipses`.

Zauważ, że inicjowanie algorytmu odbywa się prawie tak samo jak wcześniej. Używane są tylko dodatkowe parametry. Przyjrzyjmy się im. Aby zainicjować algorytm GMM, należy przekazać następujące parametry:

- `n_components` — określa liczbę rozkładów Gaussa w modelu mieszanym. W poprzednim przykładzie używane były dwa modele.
- `covariance_type` — określa właściwości macierzy kowariancji, a tym samym kształt rozkładów Gaussa. Graficzną interpretację różnic znajdziesz w dokumentacji na stronie <http://scikit-learn.org/stable/modules/mixture.html>.
- `n_iter` — określa liczbę iteracji maksymalizowania wartości oczekiwanej.

Oba fragmenty kodu są zaskakująco podobne do siebie. Jedyne dodatkowe decyzje, jakie trzeba podjąć, dotyczą relacji między wariancjami (kowariancjami) wpływającymi na kształt wynikowych klastrów.

Reszta kodu to metoda `make_ellipses`. Jest to metoda pomocnicza pomagająca przedstawić dane wyjściowe algorytmu GMM w graficznej postaci. Pochodzi ona z dokumentacji pakietu scikit-learn, dlatego możesz rozpoznać pochodzący stamtąd kod. Przed przejściem do wyjaśnień przyjrzyj się pokazanym na rysunku 2.12 danym wyjściowym z listingu 2.6.



**Rysunek 2.12.** Dane wyjściowe z listingu 2.6. Każdy panel wyświetla klastry z czterowymiarowego gaussowskiego modelu mieszanego dla zbioru danych Iris odwzorowane na dwie osie. Kolorowe zbiory reprezentują te klastry w dwóch wymiarach

Na rysunku pokazane są dane wyjściowe z czterowymiarowego gaussowskiego modelu mieszanego. Trzy czterowymiarowe klastry są rzutowane na każdą dwuwymiarową kombinację, co daje dwuwymiarową reprezentację wynikowych klastrów uzyskaną za pomocą metody `make_ellipses`.

Metoda `make_ellipses`<sup>5</sup> na poziomie koncepcyjnym jest prosta. Jej argumenty to obiekt `gmm` (model po treningu), osie wykresu (związane z jednym z sześciu diagramów z rysunku 2.12) oraz dwa indeksy, `x` i `y`, powiązane z parametrami, według których kod ma zredukować czterowymiarowy wykres. Metoda nie zwraca wartości, ale operuje na obiekcie z osiami i rysuje elipsy na wykresie.

## 2.5. Zależności między k-srednimi i algorytmem GMM

Algorytm k-srednich można przedstawić jako specjalny przypadek gaussowskiego modelu mieszanego. Ogólnie gaussowski model mieszany jest bardziej wszechstronny, ponieważ przynależność punktów danych do klastra zależy od kształtu klastra, a nie tylko od odległości od punktów.

<sup>5</sup> Ron Weiss i Gael Varoquaux, *GMM classification*, „scikit-learn”, <http://mng.bz/uKPu>.

### Uwagi na temat metody make\_ellipses

Metoda `make_ellipses` pochodzi z modułu `plot_gmm_classifier` opracowanego przez Rona Weissa i Guela Varoquaza na potrzeby pakietu `scikit-learn`. Dzięki przełożeniu macierzy kowariancji na rysowane wykresy dwuwymiarowe można wykryć pierwszy i drugi kierunek z maksymalną zmiennością oraz oszacować jej zakres. Uzyskane dane pozwalają określić i narysować elipsy odpowiadające kształtom ustalonych rozkładów Gaussa. Otrzymane kierunki i zakresy to wektory własne i wartości własne. Ich szczegółowe omówienie zawiera podrozdział 2.6.

Kształt  $n$ -wymiarowego rozkładu Gaussa zależy od kowariancji między wymiarami w każdym klastrze. Jeśli na obliczane macierze kowariancji narzucone zostaną pewne ograniczenia, algorytmy GMM i k-średnich mogą dać te same wyniki.

Jeśli macierze kowariancji dla klastrów są powiązane ze sobą (to znaczy, że muszą być takie same) i kowariancje wzduż przekątnej są identyczne, a wszystkie pozostałe elementy macierzy są równe zero, otrzymywane są sferyczne klastry o tej samej wielkości i tym samym kształcie. W takim podejściu każdy punkt zawsze należy do klastra o najbliższej średniej. Sprawdź to sam, aby się przekonać!

Na trening gaussowskiego modelu mieszanego za pomocą maksymalizacji wartości oczekiwanej mogą wpływać warunki początkowe (to samo dotyczy k-średnich). Jeśli porównasz algorytmy GMM i k-średnich, zauważysz, że algorytm GMM wymaga kilku dodatkowych warunków początkowych. Trzeba podać nie tylko początkowe centroidy, ale też początkowe macierze kowariancji i wagi rozkładów. Jedna z kilku strategii<sup>6</sup> polega na uruchomieniu algorytmu k-średnich i wykorzystaniu wynikowych centroidów do ustalenia początkowych warunków dla algorytmu GMM.

Widać więc, że nie ma w tym żadnej magii. Oba algorytmy działają podobnie. Główną różnicą między nimi jest złożoność modelu. Ogólnie wszystkie algorytmy klastrowania działają według podobnego wzorca. Na podstawie zbioru danych można wytrenować model w taki sposób, aby model stał się uogólnieniem danych (a najlepiej także procesu prowadzącego do ich powstawania). Trening zwykle odbywa się iteracyjnie, co pokazaliśmy we wcześniejszych przykładach. Zakończenie treningu ma miejsce, gdy nie da się poprawić parametrów w celu lepszego dopasowania modelu do danych.

## 2.6. Transformacje osi danych

Do tego miejsca koncentrowaliśmy się na klastrowaniu danych w pierwotnej przestrzeni cech. A może by tak zmienić przestrzeń cech na bardziej odpowiednią, na przykład z mniejszą liczbą cech, które lepiej opisują strukturę danych?

Jest to możliwe dzięki technice *analizy głównych składowych*. Polega ona na przekształcaniu osi danych w taki sposób, by jako nową podstawę dla danych zastosować kierunki o maksymalnej wariancji (zamiast  $y = 0$  i  $x = 0$ , określających standardowe osie  $x$  i  $y$ ). Te kierunki są wyznaczane przez wektory własne danych, a poziom wariancji

<sup>6</sup> Johannes Blomer i Kathrin Bujna, *Simple Methods for Initializing the EM Algorithm for Gaussian Mixture Models*, 2013, <http://arxiv.org/pdf/1312.5946.pdf>.

dla tych kierunków jest określony za pomocą powiązanych wartości własnych. W dalszych punktach szczegółowo omawiamy te pojęcia, a następnie prezentujemy przykład zastosowania analizy głównych składowych do zbioru danych Iris. Zobaczysz, jak zmniejszyły liczbę cech z czterech do dwóch bez utraty możliwości skutecznego klastrowania zbioru danych Iris w nowej przestrzeni cech.

### **2.6.1. Wektory własne i wartości własne**

Wektory własne i wartości własne<sup>7</sup> to cechy macierzy kwadratowych (o tej samej liczbie wierszy i kolumn). Opisują one macierz, przy czym określają jej specjalne i subtelnne aspekty. W tym punkcie przedstawiamy intuicyjne objaśnienie tych pojęć, a dalej omawiamy ich przydatność w kontekście zbiorów danych. Matematycznie wektory własne i wartości własne są zdefiniowane tak:

$$\mathbf{Cv} = \lambda v$$

W tym wzorze  $\mathbf{C}$  to macierz kwadratowa (o tej samej liczbie wierszy i kolumn),  $v$  to wektor własny, a  $\lambda$  to wartość własna powiązana z tym wektorem. Może się to wydawać niejasne, jednak po zapoznaniu się z przykładem zobaczysz, dlaczego ten wzór jest tak ważny.

Załóżmy, że macierz  $\mathbf{C}$  odpowiada transformacji przechylenia w przestrzeni dwuwymiarowej i dlatego jest macierzą o wymiarach  $2 \times 2$ . Gdy określony jest dwuwymiarowy zbiór danych,  $\mathbf{C}$  można zastosować do każdego punktu danych, aby utworzyć nowy przekształcony (przechylony) zbiór.

Trzeba więc rozwiązać równanie. Czy można ustalić wektory (kierunki) takie, że po zastosowaniu macierzy  $\mathbf{C}$  pozostały one niezmienione z wyjątkiem różnicę w wielkości określonej przez  $\lambda$ ? Odpowiedź na to pytanie jest pozytywna. Co jednak to oznacza i dlaczego jest to ważne?

Po uważnym zastanowieniu się nad opisanym procesem dochodzimy do wniosku, że zapewnia on skrótną, zwięzłą postać transformacji przechylenia. Zestaw wektorów własnych opisuje kierunki, na które nie wpływa zastosowanie przechylenia. Czego jednak reprezentacją jest wartość własna? Określa ona, jaki wpływ ma przechylenie na dany kierunek. Wyznacza więc zakres operacji. Wartości własne mierzą zatem znaczenie danego kierunku. Pozwala to opisać większość wpływu macierzy przechylenia za pomocą samych wektorów własnych o największych wartościach własnych.

### **2.6.2. Analiza głównych składowych**

W analizie głównych składowych<sup>8</sup> w specyficzny sposób przeprowadzana jest dekompozycja wektorów własnych i wartości własnych w celu uzyskania *głównych składowych* zbioru danych. Gdy używany jest zbiór danych o  $n$  cechach, można wyznaczyć macierz kowariancji o postaci pokazanej poniżej.

---

<sup>7</sup> K.F. Riley, M.P. Hobson i S.J. Bence, *Mathematical Methods for Physics and Engineering* (Cambridge University Press, 2006).

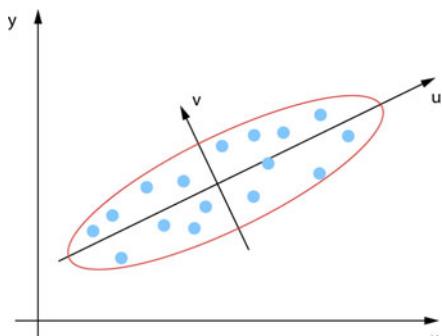
<sup>8</sup> Jonathon Shlens, *A Tutorial on Principal Component Analysis*, archiwum arXiv: 1404.1100, 2014.

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ x_{n1} & \dots & \dots & x_{nn} \end{bmatrix}$$

Ta macierz kowariancji opisuje wariancję par wszystkich cech ze zbioru danych. Element  $x_i x_j$  określa kowariancję cech  $i$  oraz  $j$ . Najlepiej opisać tę macierz jako miarę kształtu i wielkości wariancji w zbiorze danych. Zwróciłeś uwagę na kształt tej macierzy? Ponieważ obliczana jest wariancja każdej cechy względem wszystkich pozostałych, ta macierz zawsze jest kwadratowa. Liczba wierszy jest w niej taka sama jak liczba kolumn. Tu obie te wartości są równe  $n$ , czyli liczbie cech w zbiorze danych.

Zauważłeś coś jeszcze w tej macierzy? Ślusznie, jest ona symetryczna! Ponieważ  $x_i x_j = x_j x_i$ , można przeprowadzić transpozycję macierzy (obrócić ją względem przekątnej), a pozostałe ona taka sama. Zapamiętaj te dwie cechy, ponieważ wkrótce się przydadzą.

Wróćmy do tego, co macierz kowariancji reprezentuje. Pamiętaj, że jest ona miarą kształtu i wielkości wariancji w zbiorze danych. Co uzyskasz, jeśli obliczysz wektory własne i wartości własne macierzy kowariancji? Macierz kowariancji można zrozumieć przez relację (opartą na rozkładzie Choleskiego<sup>9</sup>) do transformacji, która przyjmuje losową próbkę z danych o rozkładzie Gaussa (na przykład z macierzą kowariancji, w której  $x_{ij} = 1$ ,  $i = j$ , a pozostałe elementy to zera) i zwraca dane o rozkładzie takim samym jak w pierwotnym zbiorze danych. Wektory własne macierzy kowariancji reprezentują wektory niezmienione przez taką transformację, nazywane *głównymi składowymi* zbioru danych. Ale to jeszcze nie wszystko. Ponieważ macierz kowariancji jest symetryczna, różne od siebie wektory własne tej macierzy są ortogonalne. Ilustruje to rysunek 2.13.



Rysunek 2.13. Dwie główne składowe dwuwymiarowego zbioru cech. Wektor własny  $u$  ma największą wartość własną, ponieważ wyznacza kierunek o największej zmienności. Wektor  $v$  ma mniejszą wartość własną. Te wektory są ortogonalne (kąt między nimi wynosi 90 stopni). Gdy liczba wymiarów jest większa, iloczyn skalarny dowolnej pary wektorów własnych jest równy zero

Zaletą takiej dekompozycji jest to, że każdy punkt danych można przedstawić jako liniową kombinację wektorów własnych (dzięki ich ortogonalności). Może to pozwolić na kompresję danych wielowymiarowych, ponieważ dla każdego punktu danych wystarczy zapisać odległość między istotnymi wektorami własnymi. Często większość wariancji

<sup>9</sup> Claude Brezinski, *The Life and Work of Andre Cholesky*, „Numer. Algorithms” 43 (2006): 279 – 288.

jest reprezentowana przez pierwsze dwa lub trzy wektory własne. Dlatego punkty danych o dowolnie wielu wymiarach można przedstawić za pomocą wektora z dwoma lub trzema wymiarami.

### 2.6.3. Przykład zastosowania analizy głównych składowych

Gdy już rozumiesz wektory własne, wartości własne i analizę głównych składowych, możesz zabrać się do pracy i zastosować te zagadnienia do danych. Tak jak wcześniej posługujemy się zbiorem danych Iris do zilustrowania obsługi tych zagadnień w pakiecie scikit-learn. Przyjrzyj się listingowi 2.7.

**Listing 2.7. Analiza głównych składowych w zbiorze danych Iris**

```
import numpy as np
import matplotlib.pyplot as pl

from sklearn import decomposition
from sklearn import datasets
from itertools import cycle

iris = datasets.load_iris()
X = iris.data
Y = iris.target

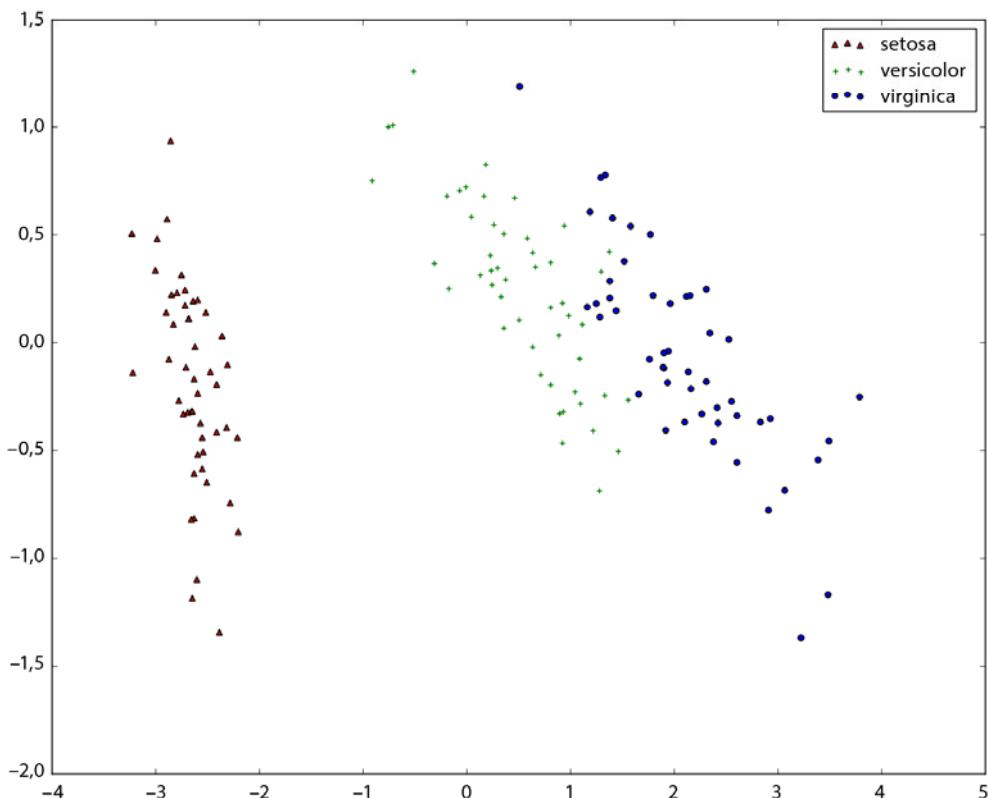
targets = range(len(iris.target_names))
colors = cycle('rgb')

pca = decomposition.PCA(n_components=2)
pca.fit(X)           ← 1 Inicjowanie rozkładu za pomocą analizy głównych składowych z uwzględnieniem dwóch komponentów
X = pca.transform(X) ← 2 Dopasowywanie danych: rozwiązywanie równania na rozkład z wektorem własnym
for target,color in zip(targets.colors):
    pl.scatter(X[Y==target,0],          ← 3 Przekształcanie danych na postać podstawową. Ponieważ
        X[Y==target,1],                ← zachowywane są tylko dwa komponenty, ta postać będzie
        label=iris.target_names[target],c=color)   ← 4 dla każdego gatunku irysów rysuje
                                                    ← (w innym kolorze) obiekty o odpowiednich
                                                    ← współrzędnych na nowej osi

pl.legend()
pl.show()
```

Transformacja zbioru danych Iris za pomocą analizy głównych składowych daje wynik pokazany na rysunku 2.14. Kod wczytuje ten zbiór danych w standardowy sposób, a następnie inicjuje dekompozycję z użyciem analizy głównych składowych z dwiema składowymi. Na tym etapie dekompozycja nie jest jeszcze przeprowadzana (przede wszystkim nie określono, jakich danych ma dotyczyć). Na razie kod przygotowuje obiekty Pythona do następnego etapu, czyli wywołania metody **fit** ②, w której następuje przekształcenie osi danych.

W tym jednym wierszu obliczana jest macierz kowariancji zbioru danych. Dla zbioru danych Iris otrzymywana jest macierz kwadratowa o wymiarach  $4 \times 4$ , dla której znajdują się wektory własne. Wspomniany wiersz wykonuje „ciężką robotę” i znajduje



Rysunek 2.14. Dekompozycja zbioru danych Iris z użyciem dwóch pierwszych wektorów własnych. Oś x przedstawia kierunek, wzduż którego wariancja danych jest maksymalna. Jest to pierwszy wektor własny, a jego wartość własna jest maksymalna w porównaniu z innymi wynikami dla tych danych. Oś y reprezentuje drugi wektor własny, o drugiej największej wartości własnej w wynikach. Punkty na tym wykresie punktowym zostały przeniesione z pierwotnych danych na nowe osie na podstawie dwóch pierwszych wektorów własnych

w zbiorze danych kierunki wyznaczające maksymalną wariancję. Następny wiersz generuje dane przekształcone według nowej osi danych ③. Pierwotne dane są na tym etapie przedstawione za pomocą kombinacji dwóch wektorów własnych o największych wartościach własnych (ponieważ w ① ograniczono liczbę składowych do dwóch). Teraz dane można wyświetlić ④, co daje efekt pokazany na rysunku 2.14. Powinno być jasne, że oś x ( $y = 0$ ) i oś y ( $x = 0$ ) określają tu kierunki (z czterowymiarowego zbioru danych Iris) o najwyższej i drugiej wariancji. Odległości między punktami z wykresu odpowiadają odległości wzduż tych kierunków w pierwotnych danych. Przekształciliśmy więc czterowymiarowy zbiór danych w dwuwymiarowy, zachowując w nowej przestrzeni większość pierwotnej wariancji.

Analiza głównych składowych to bardzo przydatne narzędzie, pomagające wyznaczyć efektywniejszą reprezentację danych i zmimimalizować przy tym utratę mocy dyskryminacyjnej. Choć tu technikę tę zastosowaliśmy do stosunkowo prostego zbioru danych, można z niej korzystać (i często się to robi) do wstępnego przetwarzania rzeczywistych danych — jako wstęp do rozwiązywania problemów z obszaru klasyfikacji i regresji,

którymi zajmujemy się w dalszej części książki. Pamiętaj jednak, że złożoność rośnie szybciej niż liczba cech w zbiorze danych, dlatego wyeliminowanie nadmiarowych i nieistotnych cech jeszcze przed analizą głównych składowych może okazać się przydatne.

## 2.7. Podsumowanie

- Błąd systematyczny to powtarzające się odchylenie dotyczące wszystkich elementów zbioru danych. Szum to losowe odchylenia wokół rzeczywistych mierzonych wartości.
- Przestrzeń cech to przestrzeń obejmująca wszystkie wektory cech dla danej liczby cech. Opisaliśmy „przekleństwo wymiarów” — im więcej cech danego zjawiska jest rejestrowanych, tym więcej punktów danych potrzeba do jego generalizacji. Praktycy z dziedziny uczenia maszynowego muszą starannie dobierać liczbę cech. Jeśli będzie ich za dużo, niemożliwe stanie się zebranie wystarczającej liczby punktów danych do zgeneralizowania badanego zjawiska. Natomiast za mała liczba cech nie pozwoli odpowiednio ująć tego zjawiska.
- Dokładnie przyjrzyliśmy się strukturze danych i szczegółowo opisaliśmy algorytm k-średnich i gaussowski model mieszany. Oba te algorytmy dobrze sprawdzają się w praktyce. Omówiliśmy też zależności między tymi technikami. W uczeniu maszynowym wiele algorytmów w określonych warunkach daje wyniki analogiczne do innych metod. Tak jest też w przypadku tych dwóch algorytmów.
- Maksymalizacja wartości oczekiwanej to ogólna klasa algorytmów, które można stosować do iteracyjnego dopasowywania danych do modelu. Najpierw ustalone jest prawdopodobieństwo, że model wygenerował rzeczywiste dane. Potem model jest modyfikowany w niewielkim stopniu, aby zwiększyć to prawdopodobieństwo. W efekcie model uczy się pierwotnego rozkładu danych z próbek używanych w trakcie treningu i staje się przybliżeniem tego rozkładu.
- Algorytm jest tylko tak dobry jak używane w nim dane! Jeśli cechy zarejestrowane na temat zjawiska nie są wystarczająco opisowe, żadna ilość uczenia maszynowego nie pozwoli uzyskać zadowalających wyników. To na człowieku spoczywa obowiązek wyeliminowania nieistotnych i nadmiarowych cech oraz zminimalizowania wielkości zbioru danych bez poświęcania przy tym mocy dyskryminacyjnej. Często pozwala to uzyskać lepsze wyniki dla tego samego zbioru danych. W końcowej części rozdziału omówiliśmy analizę głównych składowych, która pomaga przekształcić dużą liczbę cech na ich mniejszy zbiór bez istotnego zmniejszania pierwotnej wariancji danych.



# *Rekomendowanie odpowiednich treści*

## **Zawartość rozdziału:**

- Systemy rekomendacji oparte na użytkownikach, produktach i treści
- Wyszukiwanie rekomendacji dotyczących znajomych, artykułów i wiadomości
- Generowanie rekomendacji w witrynach podobnych do Netfliksa

W dzisiejszym świecie jesteśmy przytłoczeni możliwościami wyboru. W niemal każdym aspekcie życia dostępnych jest mnóstwo opcji. Codziennie musimy podejmować decyzje: od samochodów po systemy kina domowego, od księcia lub księżniczki z bajki po prawnika i księgowego, od książek i gazet po wiki i blogi. Ponadto nieustannie jesteśmy bombardowani informacjami (które czasem okazują się dezinformacjami). W takiej sytuacji przydatne są rekomendacje — zwłaszcza wtedy, gdy są one dostosowane do odbiorcy.

W branży wywierania wpływu na wybory konsumentów nikt nie jest zainteresowany dobrymi wynikami bardziej niż agencje reklamowe. Ich racją bytu jest przekonanie Cię, że naprawdę *potrzebujesz* produktu X lub usługi Y. Jeśli nie interesują Cię produkty takie jak X ani usługi takie jak Y, agencja tylko zmarnuje zasoby, a Ciebie zirytuje. Jest to problem typowy dla emisji reklam w tradycyjnych mediach (na billboardach, w telewizji i w radiu). Celem emisji reklamy jest zmiana preferencji odbiorcy w wyniku ciągłego powtarzania tego samego przekazu. Inne, przyjemniejsze i skuteczniejsze

podejście polega na emisji reklam zgodnych z preferencjami odbiorcy. Ma to prowadzić do wyboru produktu zgodnego z osobistymi pragnieniami klienta. To podejście to wyróżnik świata intelligentnej reklamy internetowej.

W tym rozdziale opisujemy wszystko, co musisz wiedzieć na temat budowania systemu rekomendacji. Poznasz tu metodę *filtrowania kolaboratywnego* i systemów rekomendacji opartych na treści. W rozdziale posługujemy się przykładem rekomendowania filmów w internetowym sklepie i uogólniamy ten mechanizm, tak aby proponowane rozwiązania można było stosować w różnych warunkach.

Internetowy sklep z filmami to prosty, ale konkretny i szczegółowy przykład, umożliwiający łatwe zrozumienie wszystkich podstawowych zagadnień z obszaru tworzenia systemów rekomendacji. Szczegółowo analizujemy tu definicję podobieństwa między użytkownikami i prezentujemy mechanizmy, które umożliwiają wykorzystanie miary podobieństwa do przedstawiania rekomendacji. Objaśniamy też bardziej skomplikowaną technikę, *rozkład według wartości osobiściwych* (ang. *singular value decomposition — SVD*), która łączy użytkowników i grupy filmów na podstawie pośrednich zależności wykrytych w danych. Po zakończeniu lektury tego rozdziału będziesz potrafił generować rekomendacje z użyciem dużego zbioru danych i zrozumiesz różne mechanizmy, które to umożliwiają.

Po omówieniu wszystkich podstawowych zagadnień na przykładzie internetowego sklepu z filmami przejdziemy do znacznie ciekawszych kwestii i przedstawimy bardziej skomplikowane przypadki. Zaprezentujemy systemy rekomendacji odgrywające bardzo istotną rolę w internetowej branży filmowej, w księgarniach internetowych i w innych sklepach internetowych.

### **3.1. Wprowadzenie – internetowy sklep z filmami**

Założymy, że jesteś właścicielem internetowego serwisu, który sprzedaje filmy lub zarabia na ich pobraniach i strumieniowaniu. Zarejestrowany użytkownik po zalogowaniu się do aplikacji może przeglądać zwiastuny dostępnych filmów. Jeśli któryś z nich mu się spodoba, użytkownik może dodać film do koszyka i dokonać zakupu później, gdy zakończy przeglądanie zasobów sklepu. Gdy użytkownik dokona zakupu lub otworzy stronę Twojej hipotetycznej aplikacji, warto zaproponować mu także inne filmy. Dostępne są miliony filmów i seriali z różnych gatunków. Niektóre osoby są wyczulone na programy i filmy, których nie lubią, dlatego gdy wyświetlasz użytkownikowi propozycje, powinieneś wybierać preferowane przez niego gatunki i unikać nielubianych. Nie martw się, jeśli wydaje się to trudne. Systemy rekomendacji są po to, aby pomóc Ci kierować do użytkowników odpowiednie treści.

System rekomendacji analizuje wybory, których użytkownik dokonał w przeszłości, i szacuje stopień, w jakim danej osobie może spodobać się określony niewidziany jeszcze film. Na tej podstawie można ustalić, jakiego rodzaju treści użytkownik lubi i w jakim stopniu. W tym celu można porównać *podobieństwo* jego preferencji z cechami danego gatunku. Jeśli chcesz opracować bardziej kreatywne rozwiązanie, możesz pomóc użytkownikom tworzyć w witrynie sieci społecznościowe na podstawie *podobieństwa* ich

gustów w zakresie filmów i programów. Widoczne staje się, że kluczowym aspektem systemów rekomendacji jest możliwość zdefiniowania stopnia podobieństwa dwóch użytkowników lub produktów (albo większej ich liczby). Na podstawie tego podobieństwa można później prezentować rekomendacje.

## 3.2. Odległość i podobieństwo

Wybierzmy dane i rozpoczęźmy szczegółową analizę tych zagadnień. W dalszych punktach będziemy uwzględniać produkty, użytkowników i oceny. W systemach rekommendacji *podobieństwo* to miara pozwalająca ustalić, jak zbliżone do siebie są dwa produkty. Przypomina to ustalanie geograficznej bliskości dwóch miast na podstawie fizycznej odległości między nimi. Odległość i podobieństwo różnią się tylko kontekstem odniesienia i używaną przestrzenią współrzędnych. Zauważ, że dwa miasta mogą być geograficznie „podobne”, ponieważ odległość między nimi w przestrzeni fizycznej jest niewielka, natomiast mogą znacznie różnić się kulturowo, ponieważ odległość między nimi mierzona w kategoriach zainteresowań i zwyczajów populacji jest bardzo duża. Te same obliczenia odległości można wykonać w różnych przestrzeniach, aby uzyskać różne miary podobieństwa.

Dla dwóch miast można posłużyć się długością i szerokością geograficzną, aby obliczyć ich bliskość geograficzną. Oceny filmów możesz traktować jak współrzędne w przestrzeni produktów lub użytkowników. Przyjrzyjmy się tym zagadnieniom w praktyce. Założymy, że wybierasz trzech użytkowników, po czym łączysz ich z listą filmów (produktów) i ich hipotetycznymi ocenami. Zwykle są to oceny od 1 do 5 (włącznie). Dwóch pierwszych użytkowników (Feliks i Cecylia) przyznało same oceny 4 i 5. Te osoby naprawdę lubią wszystkie uwzględniane filmy. Jednak oceny trzeciej osoby (Katarzyny) znajdują się w przedziale od 1 do 3. Można więc uznać, że dwóch pierwszych użytkowników jest podobnych do siebie i różnych od trzeciej osoby. Po wczytaniu przykładowych danych za pomocą skryptu otrzymujemy użytkowników, filmy i oceny z tabeli 3.1.

Zacznijmy od przyjrzenia się podobieństwom między użytkownikami na podstawie ich ocen różnych filmów. Kod przedstawiony jest na listingu 3.1. Używana jest tu skrócona i zmodyfikowana wersja zbioru danych MovieLens, która obejmuje 11 różnych filmów z tabeli 3.1. Plik z danymi znajdziesz w materiałach powiązanych z książką.

Uwzględniane są tu dwie definicje podobieństwa, wybierane na podstawie różnych wartości trzeciego argumentu metody `similarity` zastosowanej na listingu 3.1.

Szczegóły implementacji omówimy dalej. Najpierw przyjrzyj się rysunkowi 3.1. Widoczne są na nim wyniki uzyskane w efekcie porównania trzech użytkowników tylko ze względu na oceny. Dobrze widać tu, że preferencje filmowe Feliksa są dużo bardziej zbliżone do preferencji Cecyllii niż Katarzyny.

Podobieństwo między użytkownikami nie zależy od kolejności przekazywania argumentów do metody `similarity`. Podobieństwo Feliksa do niego samego jest równe 1,0, co uznajemy za maksymalną wartość podobieństwa między dwoma jednostkami. Właściwości podobieństwa wynikają z tego, że wiele miar jest opartych na odległości

**Tabela 3.1.** Oceny użytkowników pokazują, że Feliks i Cecylia są bardziej zgodni niż Feliks i Katarzyna. Używane są tu (za pozwoleniem) zmodyfikowane dane z bazy MovieLens<sup>a</sup>

Użytkownik	Film	Ocena
0: Feliks	0: Toy Story	5
	1: Jumanji	4
	2: Jeszcze bardziej zgryźliwi tetrycy	5
	3: Czekając na miłość	4
	4: Ojciec panny młodej II	5
	5: Gorączka	4
	6: Sabrina	5
	0: Toy Story	5
	2: Jeszcze bardziej zgryźliwi tetrycy	5
	4: Ojciec panny młodej II	4
1: Cecylia	5: Gorączka	5
	7: Tom i Huck	5
	8: Nagła śmierć	4
	9: GoldenEye	5
	0: Toy Story	1
	2: Jeszcze bardziej zgryźliwi tetrycy	2
	3: Czekając na miłość	2
	6: Sabrina	3
	9: GoldenEye	2
	10: Prezydent — Miłość w Białym Domu	1
2: Katarzyna		

<sup>a</sup> GroupLens Research, zbiór danych MovieLens, <http://grouplens.org/datasets/movielens/>.

**Listing 3.1. Obliczanie podobieństwa między użytkownikami na podstawie ocen filmów**

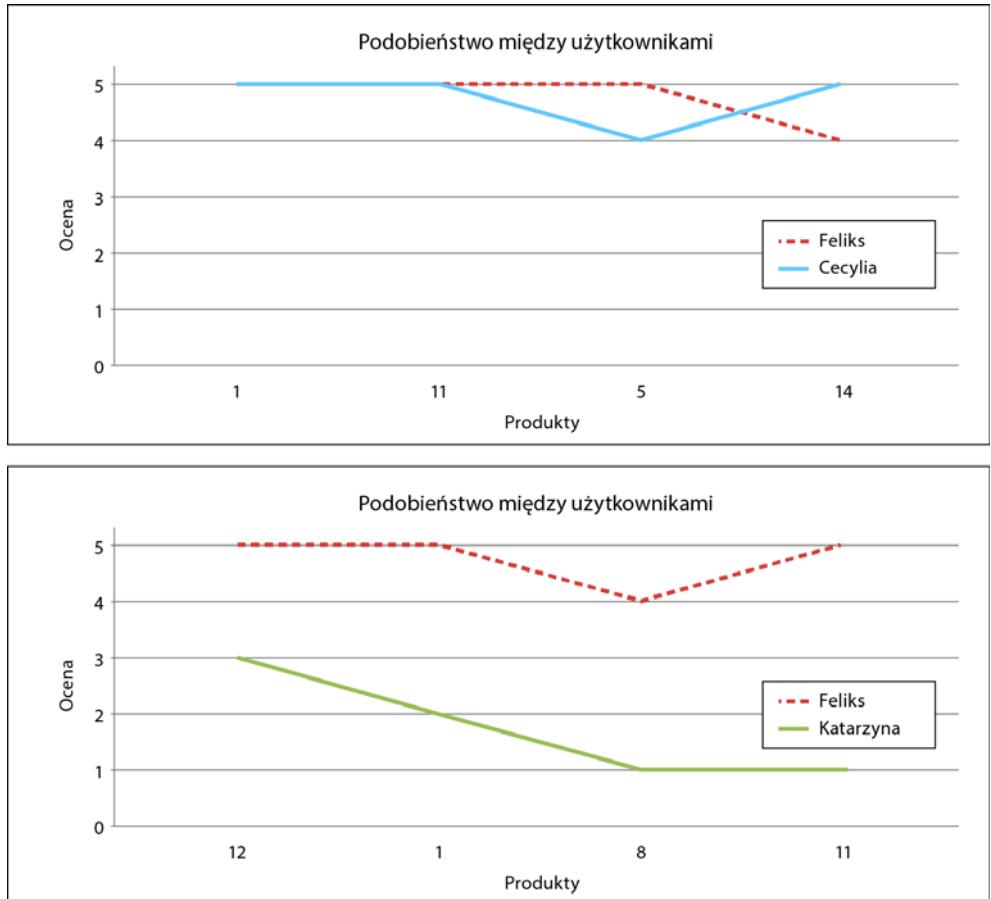
```
dat_file = 'ratings-11.dat'
item_file = 'movies-11.dat'

names = ['Feliks', 'Cecylia', 'Katarzyna']

userdict = read_user_data_from_ratings(dat_file) ← Wczytywanie danych za pomocą
itemdict = read_item_data(item_file)           metody pomocniczej.

similarity(0,1,sim_type=0) | Podobieństwo między Feliksem a Cecylią z uwzględnieniem obu miar.
similarity(0,1,sim_type=1)
similarity(0,2,sim_type=0)
similarity(1,2,sim_type=0)
similarity(2,1,sim_type=0)
similarity(0,0,sim_type=0) | Podobieństwo między Feliksem a nim samym z uwzględnieniem obu miar.
similarity(0,0,sim_type=1)
```

i przypomina omawianą w liceum odległość geometryczną między dwoma punktami w przestrzeni. Ogólnie odległości matematyczne cechują się następującymi czterema ważnymi właściwościami:



Rysunek 3.1. Podobieństwo między dwójką użytkowników można zmierzyć, oceniając, w jakim stopniu (jeśli w ogóle) pokrywają się odpowiadające im linie na pokazanych wykresach. Feliks i Cecylia (u góry) są bardziej podobni do siebie niż Feliks i Katarzyna (u dołu)

- Każda odległość jest równa zero lub większa. W większości sytuacji podobieństwo (podobnie jak odległość) jest ograniczane do wartości nieujemnych. Tu podobieństwo jest ograniczane do przedziału [0,1].
- Odległość między dwoma punktami (na przykład A i B) wynosi zero wtedy i tylko wtedy, gdy A i B to te same punkty. W implementacji metody similarity w omawianym przykładzie ta właściwość jest odzwierciedlana w ten sposób, że jeśli dwóch użytkowników podał dokładnie te same oceny, podobieństwo między tymi osobami jest równe 1,0. Przekonasz się o tym, gdy uruchomisz kod z listingu 3.1, gdzie ten sam użytkownik jest podany dwukrotnie w celu pokazania, że podobieństwo jego z nim samym to 1,0. Oczywiście możesz dodać czwartego użytkownika i udowodnić, że podobieństwo między nim i inną osobą będzie równe 1,0, jeśli obie osoby tak samo ocenili te same filmy.

- Trzecia właściwość dotyczy symetrii — odległość między A i B jest taka sama jak między B i A. To oznacza, że jeśli preferencje Katarzyny są podobne do gustów Cecylii, ta sama relacja w tym samym stopniu zachodzi także w drugą stronę. Często pożądane jest, by miara podobieństwa też cechowała się symetrią ze względu na przekazywane argumenty.
- Czwarta właściwość odległości matematycznych to nierówność trójkąta, związana z odległościami między trzema punktami. W ujęciu matematycznym: jeśli  $d(A,B)$  oznacza odległość między punktami A i B, to zgodnie z nierównością trójkąta  $d(A,B) \leq d(A,C) + d(C,B)$  dla trzeciego punktu, C. Gdy przyjrzysz się danym wyjściowym z listingu 3.1, zobaczyś, że Feliks jest podobny do Cecylii na poziomie 0,391, Cecylia do Katarzyny na poziomie 0,002, a Feliks do Katarzyny na poziomie 0,006, co jest wartością mniejszą niż suma dwóch pierwszych podobieństw. Jednak ta właściwość nie dla wszystkich podobieństw jest zachowana.

Rozluźnienie czwartej podstawowej właściwości odległości jest dopuszczalne w trakcie pracy z podobieństwami. Uwzględnianie właściwości odległości w ramach pracy z podobieństwami nie jest konieczne. Zawsze jednak powinieneś dbać o to, by uwzględniane podstawy matematyczne były zgodne ze zdrowym rozsądkiem. Williamowi Jamesowi przypisuje się pochodzący sprzed ponad stu lat kontrprzykład dla reguły nierówności trójkąta<sup>1</sup>: „Płomień jest podobny do księżyca, ponieważ obie te rzeczy świecą. Księżyca jest podobny do piłki, ponieważ obie te rzeczy są okrągłe. Jednak — niezgodnie z zasadą nierówności trójkąta — płomień nie przypomina piłki”. Ciekawe omówienie podobieństw w kontekście nauk poznawczych znajdziesz w książce *Classification and Cognition* W.K. Estesa<sup>2</sup>.

Na rysunku 3.1 pokazana jest wizualna reprezentacja podobieństwa w postaci wykresów ocen trzech przykładowych użytkowników. Im bliższe sobie są wykresy ocen, tym bardziej podobni są użytkownicy. Im większe różnice między liniami, tym podobieństwo jest mniejsze. W górnej części rysunku widać, że krzywe reprezentujące oceny Feliksa i Cecylii znajdują się blisko siebie, co ilustruje podobieństwo między tymi osobami. Na dolnym rysunku znajdują się oceny Feliksa i Katarzyny. Krzywe są tu znacznie oddalone od siebie, co jest zgodne z niskim poziomem podobieństwa uzyskanym za pomocą obliczeń.

Wykresy ocen z rysunku 3.1 dobrze obrazują odwrotny stosunek między odlegością i podobieństwem. Im większa jest odległość między krzywymi, tym mniejsze podobieństwo między użytkownikami. Im mniejsza odległość dzieli krzywe, tym większe jest podobieństwo użytkowników. Z następnego punktu dowiesz się, że wyznaczanie podobieństwa często obejmuje określanie pewnego rodzaju odległości, choć nie zawsze tak się dzieje. Pojęcie „odległość” jest bardziej powszechnne. Pojęcia „odległość” i „podobieństwo” to specjalne przypadki ogólnej koncepcji miary.

<sup>1</sup> William James, *Principles of Psychology* (Holt, 1890).

<sup>2</sup> W.K. Estes, *Classification and Cognition* (Oxford University Press, 1994).

### 3.2.1. Więcej o odległości i podobieństwie

Przyjrzyj się teraz kodowi, który pomógł ustalić podobieństwo między użytkownikami. Zobacz, w jaki sposób jest ono obliczane. Kod z listingu 3.2 przedstawia metodę `similarity`. Ta metoda przyjmuje trzy argumenty: identyfikatory dwóch użytkowników, których chcemy porównywać, i uwzględniany rodzaj podobieństwa.

**Listing 3.2. Obliczanie podobieństwa między dwoma użytkownikami**

```
def similarity(user_id_a,user_id_b,sim_type=0):
    user_a_tuple_list = userdict[user_id_a].get_items()
    user_b_tuple_list = userdict[user_id_b].get_items()
    common_items=0
    sim = 0.0
    for t1 in user_a_tuple_list:
        for t2 in user_b_tuple_list:
            if (t1[0] == t2[0]):
                common_items += 1
                sim += math.pow(t1[1]-t2[1],2)
    if common_items>0:
        sim = math.sqrt(sim/common_items)
        sim = 1.0 - math.tanh(sim)
        if sim_type==1:
            max_common = min(len(user_a_tuple_list),len(user_b_tuple_list))
            sim = sim * common_items / max_common
    print "Podobieństwo między użytkownikami".
    ↪names[user_id_a],"i",names[user_id_b],"wynosi", sim
    return sim ← Jeśli nie występują wspólne oceny, zwracana jest wartość 0.
```

W kodzie znajdują się dwa wzory na podobieństwo, co pokazuje, że podobieństwo to stosunkowo płynna i modyfikowalna miara. Przyjrzyj się podstawowym krokom w obliczeniach z tych wzorów. Najpierw określone są różnice między ocenami wszystkich filmów, które zostały ocenione przez obie osoby. Te różnice są podnoszone do kwadratu i dodawane. Pierwiastek kwadratowy z tej sumy to *odległość euklidesowa*, przy czym ten wynik nie wystarcza do określenia miary podobieństwa:

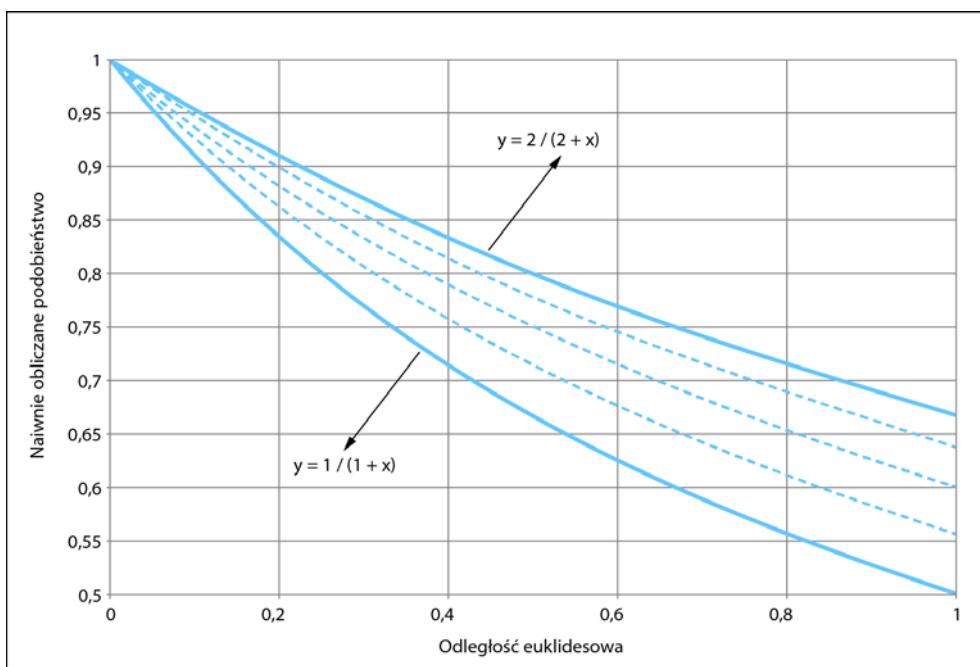
$$d_{a,b} = \sqrt{\sum_i (ocena_{a,i} - ocena_{b,i})^2}$$

To równanie to definicja odległości euklidesowej ( $d_{a,b}$ ) między użytkownikami  $a$  i  $b$ . Jest ona równa pierwiastkowi kwadratowemu z sumy kwadratów różnic między podanymi przez te osoby ocenami tych samych filmów. Wyrażenie  $ocena_{a,i}$  oznacza ocenę, jaką użytkownik  $a$  przyznał filmowi  $i$ .

Wcześniej wspomnieliśmy, że odległość i podobieństwo są w pewnym sensie swoimi odwrotnościami. Im mniejsza jest odległość euklidesowa, tym bardziej dwóch użytkowników jest do siebie podobnych. Pierwsza, naiwna próba ustalenia podobieństwa to dodanie jeden do odległości euklidesowej i obliczenie odwrotności. To sprawi, że dla dużych odległości uzyskane zostaną małe wyniki i na odwrót. Gwarantuje to też, że dla odległości zero podobieństwo będzie równe 1,0.

Może się wydawać, że odwrócenie odległości (po dodaniu stałej jeden) może zadziałać. Jednak to na pozór nieszkodliwe przekształcenie ma wady. Jeśli dwóch użytkowników obejrzało tylko jeden film, oceniony przez jedną osobę na 1 i drugą na 4, suma kwadratów różnic wyniesie 9,0. Wtedy naiwna próba obliczenia podobieństwa na podstawie odległości euklidesowej da wartość 0,25. Ten sam poziom podobieństwa można uzyskać także w innych scenariuszach. Jeśli dwóch użytkowników obejrzało trzy filmy i w sumie oceny różniły się o  $\sqrt{3}$ , podobieństwo według omawianej naiwnej miary też wyniesie 0,25. Intuicyjnie jednak wydaje się, że podobieństwo między tą drugą parą osób powinno być większe niż w przypadku użytkowników, którzy obejrzeliby jeden film i przyznali mu oceny różniące się o trzy jednostki (przy pięciu możliwych).

Te naiwne obliczenia prowadzą do nadmiernego ograniczenia poziomu podobieństwa dla niewielkich odległości (ponieważ dodawana jest wartość jeden), co w mniejszym stopniu wpływa na duże odległości (znacznie większe niż 1,0). Co się stanie, gdy dodamy inną wartość? Ogólna postać naiwnych obliczeń podobieństwa to  $y = \beta / (\beta + x)$ , gdzie  $\beta$  to wolny parametr, a  $x$  to odległość euklidesowa. Rysunek 3.2 pokazuje, jak kształtuje się obliczane w naiwny sposób podobieństwo dla parametru  $\beta$  o różnych wartościach z przedziału od 1,0 do 2,0.



Rysunek 3.2. Krzywe określające naiwnie obliczane podobieństwo jako funkcje odległości euklidesowej

Pamiętając o wadach naiwnych obliczeń podobieństwa, przyjrzyj się pierwszej (domyślnej) definicji podobieństwa między dwoma użytkownikami podanej na listingu 3.2 (dla `sim_type=0`). Jeśli użytkownicy ocenili te same filmy, należy podzielić sumę kwadratów różnic przez liczbę wspólnie ocenionych filmów, wyciągnąć z tego pierwiastek

kwadratowy i przekazać wynik do specjalnej funkcji. Tą funkcją jest *tangens hiperboliczny*. Wartość tangensa hiperbolicznego jest odejmowana od 1,0, tak aby wartość podobieństwa wynosiła od 0,0 do 1,0, gdzie 0,0 oznacza brak podobieństwa, a 1,0 — największe możliwe podobieństwo. Gotowe! Otrzymaliśmy pierwszą definicję podobieństwa użytkowników opartą na ich ocenach.

Druga definicja podobieństwa zastosowana na listingu 3.2 (dla `sim_type=1`) to usprawioniona wersja pierwszego wzoru. W nowej wersji uwzględniany jest stosunek liczby wspólnych filmów do liczby wszystkich możliwych wspólnych filmów. Jest to intuicyjnie uzasadnione. Jeśli ja obejrzałem 30 filmów, a Ty — 20, liczba możliwych wspólnych filmów wynosi 20. Założymy, że oceniliśmy tylko pięć tych samych filmów. Wprawdzie nasze oceny są zbliżone, ale dlaczego liczba wspólnych filmów jest tak niska? Czy nie należy odzwierciedlić tego w ocenie podobieństwa? Właśnie ten aspekt uwzględniany jest w drugim wzorze na podobieństwo. Chodzi o to, by stopień, w jakim dwie osoby oglądają te same filmy, miał wpływ na ocenę podobieństwa entuzjastów kina.

### **3.2.2. Który wzór na podobieństwo jest najlepszy?**

Zapewne wiesz już, że możesz stosować wiele wzorów do określania podobieństwa między dwoma użytkownikami (lub produktami). Oprócz dwóch wzorów przedstawionych wcześniej możesz posłużyć się miarą *podobieństwa Jaccarda*, zdefiniowaną jako stosunek części wspólnej do sumy zbiorów produktów (lub w przypadku podobieństwa między produktami stosunek części wspólnej do sumy zbiorów użytkowników). Tak więc podobieństwo Jaccarda między dwoma zbiorami, A i B, to  $\frac{\text{część wspólna}(A, B)}{\text{suma}(A, B)}$ .

Oczywiście naturalne jest pytanie: „Który wzór na podobieństwo jest najlepszy?”. Odpowiedź, jak zwykle, brzmi: „To zależy”. Tu zależy to od danych. W jednym z niewielu zakrojonych na szeroką skalę porównań miar podobieństwa<sup>3</sup> prosta odległość euklidesowa dała najlepsze empiryczne wyniki spośród siedmiu badanych miar (mimo że inne wzory były bardziej wyrafinowane i intuicyjnie oczekiwano, że dadzą lepsze wyniki). W badaniach uwzględniono 1 279 266 kliknięć w rekomendacje w sieci społecznościowej Orkut w okresie od 22 września do 21 października 2004 roku.

Nie zalecamy tu losowego wyboru miar podobieństwa. Jeśli jednak się spieszysz, rozważ zastosowanie najpierw odległości euklidesowej lub podobieństwa Jaccarda. Te miary powinny zapewnić Ci dobre wyniki. Spróbuj zrozumieć naturę danych i ustalić, co oznacza to, że dwie osoby (lub dwa przedmioty) są podobne. Jeśli nie zrozumiesz powodów, dla których dana miara podobieństwa (określony wzór) jest dobra lub zła, narazisz się na problemy. Aby lepiej zrozumieć to zagadnienie, zastanów się nad często przyjmowanym błędym założeniem: „Najkrótsza ścieżka między dwoma punktami to łącząca je linia prosta”. To stwierdzenie jest prawdziwe tylko w tak zwanych *geometriach płaskich* (na przykład dla boiska piłkarskiego). Aby się o tym przekonać, porównaj drogę biegnącą prosto przez wysokie i wąskie wzgórze z drogą wokół podstawy wzgórza. W wielu sytuacjach prosta droga nie będzie wcale najkrótszą.

---

<sup>3</sup> Ellen Spertus, Mehran Sahami i Orkut Buyukkokten, *Evaluating Similarity Measures: A Large-Scale Study in the Orkut Social Network*, „Proceedings of KDD” 2005.

Oto podsumowanie — jedną z podstaw systemów rekomendacji jest możliwość pomiaru podobieństwa między dowolnymi dwoma użytkownikami i podobieństwa między dwoma produktami. Przedstawiliśmy tu kilka mier podobieństwa, które możesz od razu zacząć stosować, i na przykładzie sklepu z filmami pokazaliśmy typową strukturę danych przetwarzanych w trakcie generowania rekomendacji. Teraz przechodzimy do analizy typów systemów rekomendacji i sposobów ich działania.

### **3.3. Jak działają systemy rekomendacji?**

Skoro już dobrze rozumiesz, czym jest podobieństwo między dwoma użytkownikami i dwoma produktami, hora przejść do omówienia systemów rekomendacji. Są dwa ogólne typy takich systemów. Systemy z pierwszej kategorii wykorzystują głównie analizę treści (powiązanych z produktami, użytkownikami lub i jednym, i drugim). Główne aspekty tego podejścia to zbieranie i analizowanie informacji związanych z użytkownikami i produktami. Te informacje mogą być udostępniane albo przez oprogramowanie, albo przez zewnętrzne źródła. System może rejestrować informacje o użytkownikach *bezpośrednio*, na podstawie odpowiedzi z prezentowanych kwestionariuszy, lub *pośrednio*, w wyniku analizy profilu użytkownika, nawyków czytania wiadomości, e-maili, blogów itd. Nie omawiamy tu szczegółowo systemów rekomendacji z tej kategorii, ponieważ są one w dużym stopniu projektowane ręcznie i specyficzne dla aplikacji.

Druga kategoria to systemy z *filtrowaniem kolaboratywnym* (ang. *collaborative filtering* — CF). Pierwsza generacja takich rozwiązań pojawiła się w eksperymentalnym systemie pocztowym opracowanym około 1992 roku w laboratorium PARC (ang. *Palo Alto Research Center*) firmy Xerox<sup>4</sup>. Filtrowanie kolaboratywne wykorzystuje ślady, które użytkownik pozostawia w trakcie interakcji z oprogramowaniem. Zwykle tymi śladami są oceny wystawione przez użytkownika — na przykład omawiane w poprzednim podrozdziale oceny filmów. Filtrowanie kolaboratywne nie jest ograniczone do zmiennych jednowymiarowych i nieciągłych. Główną cechą tej techniki jest to, że polega na wcześniejszych zachowaniach, a nie na treści poszczególnych elementów z analizowanego zbioru. Generowanie rekomendacji za pomocą filtrowania kolaboratywnego nie wymaga wiedzy z określonej dziedziny ani wstępnego zbierania i analizowania danych. To podejście można podzielić na trzy dalsze podtypy uwzględniające produkty, użytkowników i model.

W filtrowaniu kolaboratywnym według produktów najważniejsze jest podobieństwo między produktami. W przykładzie z rekomendacjami filmów należałoby zbudować macierz filmów, w której wartości określają podobieństwo między nimi. Następnie można polecać użytkownikowi filmy podobne do tych, które już obejrzał lub wysoko ocenił. W filtrowaniu kolaboratywnym według użytkowników istotne jest podobieństwo między użytkownikami. W tym scenariuszu użytkownikom rekomendowane są treści oglądane lub lubiane przez podobne osoby. W podejściu opartym na modelu podo-

---

<sup>4</sup> David Goldberg i współpracownicy, *Using Collaborative Filtering to Weave an Information Tapestry*, „Communications of the ACM” 35, nr 12 (grudzień 1992), <http://mng.bz/eZpM>.

bieństwo między produktami lub użytkownikami nie jest bezpośrednio wyznaczane. Stosowany jest model, który ujmuje interakcje między użytkownikami i produktami.

W dalszych punktach prezentujemy filtrowanie kolaboratywne według użytkowników. Implementację podejścia opartego na produktach pozostawiamy jako ćwiczenie dla Czytelników. W podrozdziale 3.5 przejdziemy do bezpośredniego implementowania filtrowania kolaboratywnego według modelu, wykorzystując rozkład SVD.

#### Filtrowanie kolaboratywne według produktów i według użytkowników

W zależności od danych obie te techniki filtrowania kolaboratywnego mogą mieć różny profil działania. W procesie rekomendowania filmów zwykle występuje większa liczba użytkowników niż filmów. Dlatego macierz podobieństwa użytkowników jest znacznie większa niż dla produktów i kosztowniejsza do wyznaczenia. Gdy wybierzesz dwóch użytkowników, którzy ocenili dużą liczbę filmów, nie masz gwarancji, że filmy te się pokrywają. Każdy z użytkowników prawdopodobnie obejrzał tylko niewielki podzbiór dostępnych filmów. Obliczanie macierzy podobieństwa produktów jest mniej kosztowne, ponieważ jest ona mniejsza. Jeśli sprawdzisz dwa filmy o dużej liczbie ocen, może się okazać (zależy to od danych), że oceniło je wielu tych samych użytkowników, ponieważ liczba ocen jednego filmu jest prawdopodobnie większa niż liczba filmów obejrzanych przez jednego użytkownika. Z tego wynika, że macierz podobieństwa między użytkownikami jest trudniejsza do obliczenia od macierzy podobieństwa między produktami.

Choć w kompletnych systemach rekomendacji filmów macierz podobieństwa między użytkownikami jest trudniejsza do wyznaczenia, przedstawiamy tu implementację filtrowania kolaboratywnego według użytkowników. W praktyce podejście stosowane w obu wersjach filtrowania jest identyczne, a podobieństwo między użytkownikami oparte na ocenionych przez nich filmach jest łatwiejsze do zrozumienia niż podobieństwo między produktami zależne od użytkowników, którzy je ocenili.

### 3.4. Filtrowanie kolaboratywne według użytkowników

Oto przysłowie z czasów starożytnej Grecji (różne jego wersje występują w prawie każdej kulturze świata): „Pokaż mi swoich przyjaciół, a powiem ci, kim jesteś”. Filtrowanie kolaboratywne oparte na grupach podobnych użytkowników jest algorytmicznym ucielesnieniem tego przysłownia. Aby ustalić ocenę, jaką konkretny użytkownik mógłby przyznać danemu produktowi, należy sprawdzić oceny przyznanego temu produktowi przez podobnych użytkowników (możemy nazwać ich przyjaciółmi lub sąsiadami). Następnie wystarczy pomnożyć oceny przyjaciół z uwzględnieniem wag i dodać wyniki. To proste. Na listingu 3.3 znajduje się kod generujący rekomendacje na podstawie podobieństwa między użytkownikami. Używane są tu te same dane z tabeli 3.1, którymi posłużyliśmy się do zilustrowania różnych miar podobieństwa.

#### Listing 3.3. Rekomendacje z użyciem filtrowania kolaboratywnego według użytkowników

```
data = Data()  
format = {'col':0, 'row':1, 'value':2, 'ids': 'int'}  
data.load(dat_file, sep='::', format=format)  
  
similarity_matrix = SimilarityMatrix()  
recommend(0,10) ← Zwraca pierwszych 10 rekomendacji dla użytkownika 0 (Feliksa).
```

recommend(1,10) ← Zwraca pierwszych 10 rekomendacji dla użytkownika 1 (Cecylii).  
 recommend(2,10) ← Zwraca pierwszych 10 rekomendacji dla użytkownika 2 (Katarzyny).

Na listingu 3.3 znajduje się wysokopoziomowy kod do filtrowania kolaboratywnego według użytkowników. Program za pomocą metody `load` wczytuje dane do obiektu typu `recsys.datamodel.data`. Następnie używana jest metoda `recommend` do wygenerowania rekomendacji filmów, które mogą spodobać się użytkownikowi. Ta metoda prognozuje oceny, jakie użytkownik mógłby przyznać określonym filmom. Uzyskane wyniki są sortowane i program zwraca pierwszych 10 filmów. Metoda `recommend` prognozuje oceny tylko tych produktów, które nie zostały jeszcze ocenione przez daną osobę. W używanym zbiorze danych jest mniej niż 10 filmów nieocenionych przez poszczególnych użytkowników, dlatego program nigdy nie dochodzi do tego limitu. W wynikach możesz natopiąć wartość „Brak”. Jest ona przyznawana tylko wtedy, gdy użytkownik nie ocenił danego filmu i nie zrobiła tego także żadna inna osoba ze zbioru danych. Ta wartość nie reprezentuje więc zerowej oceny, ale jej brak.

Aby uzyskać przykładowe wyniki, możesz uruchomić kod. Zobaczysz, że Cecylii rekomendowane są filmy *Jumanji*, *Czekając na miłość* i *Sabrina*. Wynika to przede wszystkim z tego, że Feliks ocenił te filmy. Cecylia jest podobna do Feliksa (o czym przekonałeś się wcześniej), a Feliks przyznał tym filmom wysokie oceny. Wygląda na to, że system rekomendacji działa poprawnie. Zauważ, że każdemu użytkownikowi rekomendowane są tylko te filmy, których dana osoba nie widziała i nie oceniła, a rekomendacje są generowane zgodnie z gustami podobnych użytkowników.

W jaki sposób metoda `recommend` dochodzi do wniosków? Jak potrafi ustalić użytkowników podobnych do danej osoby (przyjaciół)? Jak potrafi zarekomendować filmy z listy tych, których użytkownik jeszcze nie widział? Prześledźmy podstawowe kroki z tej metody, aby zrozumieć, co się w niej dzieje. Systemy rekomendacji wykorzystujące filtrowanie kolaboratywne działają w dwóch krokach. Najpierw obliczają podobieństwo między użytkownikami lub produktami. Następnie wykorzystują średnią ważoną do wyznaczenia oceny, jaką użytkownik mógłby przyznać niewidzianym jeszcze produktom. Przyjrzyj się teraz metodzie `recommend` (listing 3.4), aby zobaczyć, jak ten proces wygląda w praktyce.

#### **Listing 3.4. Metoda recommend**

```
def recommend(user_id, top_n):
    #[item,value),(item1, value1)...]
    recommendations = []
    for i in itemdict.keys():
        if (int(i) not in items_reviewed(int(user_id), userdict)):
            recommendations.append((i,predict_rating(user_id, i)))
    recommendations.sort(key=lambda t: t[1], reverse=True)
    return recommendations[:top_n]
```

Na tym listingu widać, że metoda `recommend` jest stosunkowo prosta i w dużym stopniu bazuje na metodzie `predict_rating`. Metoda `recommend` przyjmuje identyfikator `user_id` i liczbę początkowych rekomendacji, które ma zwrócić. Następnie iteracyjnie sprawdza

każdy produkt ze słownika. Jeśli użytkownik jeszcze nie ocenił tego produktu, wywoływana jest metoda predict\_rating, a wynik zostaje dodany do listy rekomendacji. Ta lista jest potem sortowana i przekazywana z powrotem do jednostki wywołującej. Na listingu 3.5 pokazany jest kod metody predict\_rating.

**Listing 3.5. Prognozowanie ocen wystawianych przez użytkowników**

```
def predict_rating(user_id, item_id):
    estimated_rating = None;
    similarity_sum = 0;
    weighted_rating_sum = 0;

    if (int(item_id) in items_reviewed(user_id,userdict)):
        return get_score_item_reviewed(user_id,item_id,userdict)
    else:
        for u in userdict.keys():
            if (int(item_id) in items_reviewed(u,userdict)):
                item_rating = get_score_item_reviewed(u,item_id,userdict)
                user_similarity =
                    ↳similarity_matrix.get_user_similarity(user_id,u)
                weighted_rating = user_similarity * item_rating
                weighted_rating_sum += weighted_rating
                similarity_sum += user_similarity

    if (similarity_sum > 0.0):
        estimated_rating = weighted_rating_sum / similarity_sum

    return estimated_rating
```

Na listingu widać, że metoda predict\_rating przyjmuje dwa parametry: identyfikator użytkownika i identyfikator produktu, a zwraca prognozowaną ocenę, jaką użytkownik mógłby przyznać danemu produktowi.

Analizując kod wiersz po wierszu, można zobaczyć, że jeśli użytkownik już ocenił dany produkt, kod nie prognozuje oceny, tylko zwraca przyznaną ocenę. Jeżeli dana osoba jeszcze nie oceniła produktu, wykonywane są kroki filtrowania kolaboratywnego według użytkowników. Metoda sprawdza wszystkich użytkowników z systemu pod kątem tego, czy ocenili analizowany produkt. Jeśli użytkownik go ocenił, pobierana jest ocena oraz poziom podobieństwa danego użytkownika z osobą, dla której generowane są prognozy. Następnie obliczana jest ważona ocena (podobieństwo obu użytkowników razy przyznana ocena). Aby ustalić wartość uwzględniającą wszystkich użytkowników z systemu, należy obliczyć sumę wartości weighted\_ratings i sumę podobieństw. Ta druga pozwala znormalizować wynik, tak aby znajdował się w przedziale [0,5]. Jeśli w zbiorze danych występują użytkownicy podobni do osoby, dla której generowane są rekomendacje, i ci użytkownicy ocenili sprawdzany produkt, za pomocą przedstawionego kodu będzie można oszacować ocenę, jaką może wydać ta osoba.

Choć ten kod świetnie wykonuje postawione mu zadanie, zauważ, że występują w nim problemy z wydajnością, które warto zbadać. Gdy przyjrzyisz się listingom 3.4 i 3.5 razem, zobaczyisz, że wykonywane są pętle uwzględniające wszystkie produkty i wszystkich użytkowników z systemu. Warto zastanowić się nad innymi strukturami

danych, które pozwolilyby tego uniknąć. Można rozważyć na przykład zapisywanie użytkowników w słowniku z kluczami w postaci ocenionych produktów (wtedy iteracyjnie przetwarzani będą tylko użytkownicy, którzy ocenili dany produkt, a nie — jak ma to miejsce na listingu 3.5 — wszystkie osoby). Jest to tylko jedno z wielu usprawnień wydajności, które można zastosować w systemach produkcyjnych z filtrowaniem kolaboratywnym według użytkowników.

Do tej pory zaprezentowaliśmy ogólny zarys systemu filtrowania kolaboratywnego według użytkowników. Brakuje jednak pewnych szczegółów kodu z listingu 3.5 — obliczeń macierzy podobieństwa. Na listingu 3.6 pokazana jest klasa odpowiedzialna za takie obliczenia.

#### Listing 3.6. Klasa SimilarityMatrix

```
class SimilarityMatrix:

    similarity_matrix = None

    def __init__(self):
        self.build() ← Wywołuje metodę build
                           w trakcie tworzenia obiektu.

    def build(self):
        self.similarity_matrix = np.empty((len(userdict), len(userdict),))

        for u in range(0, len(userdict)):
            for v in range(u+1, len(userdict)):
                rcm = RatingCountMatrix(int(u), int(v))
                if(rcm.get_agreement_count() > 0):
                    self.similarity_matrix[u][v] = ↗ Oblicza wartości tylko dla w większych
                                                 niż aktualna wartość u (używana jest
                                                 macierz trójkątna górną).
                    self.similarity_matrix[v][u] = ↗ Jeśli użytkownicy się zgadzają
                                                 za pomocą klasy
                                                 RatingCountMatrix zwracane
                                                 jest podobieństwo między nimi.
                    self.similarity_matrix[u][v] = ↗ Podobieństwo między użytkownikiem
                                                 a nim samym wynosi 1.
                else:
                    self.similarity_matrix[u][v] = 0
                    self.similarity_matrix[v][u] = 0

    def get_user_similarity(self, user_id1, user_id2):
        return ↗self.similarity_matrix[min(user_id1, user_id2), max(user_id1, user_id2)] ← Z powodu użycia macierzy trójkątnej górnej trzeba wykonać dodatkowe zadania,
                                         by uzyskać podobieństwo. Użytkownik o mniejszym identyfikatorze musi być podany
                                         w wierszu macierzy, a druga osoba — w jej kolumnie. W przeciwnym razie zwrócone
                                         podobieństwo wyniesie zero (nie jest ono obliczane przez metodę build()).
```

Klasa `SimilarityMatrix` odpowiada za przechowywanie, obliczanie i udostępnianie macierzy określającej podobieństwo między użytkownikami. W momencie tworzenia obiektu tej klasy wywoływana jest metoda `build`, która oblicza podobieństwo między użytkownikami ze słownika `userdict`. Warto zwrócić tu uwagę na dwie ważne kwestie. Po pierwsze, używana jest macierz trójkątna górną. W podrozdziale 3.2 wspomnieliśmy, że uzasadnione może być zapewnianie symetrii podobieństw. Wtedy jeśli Katarzyna jest podobna do Feliksa, Feliks jest podobny do Katarzyny. Użyta tu implementacja ma tę właściwość, dlatego można przedstawić macierz podobieństwa w postaci trójkątnej górnej. Nie ma sensu zapisywać podobieństwa między a i b, jeśli dostępne jest już

podobieństwo między b i a. To podejście pozwala zmniejszyć o połowę macierz podobieństwa i zakres obliczeń potrzebnych do jej uzyskania.

Drugą wartą uwagi kwestią jest zastosowanie klasy pomocniczej RatingCountMatrix. Jej definicję znajdziesz na listingu 3.7.

#### Listing 3.7. Klasa RatingCountMatrix

```
class RatingCountMatrix:
    user_id_a = None
    user_id_b = None
    matrix = None

    def __init__(self, user_id_a, user_id_b):
        num_rating_values = max([x[0] for x in data])
        self.user_id_a = user_id_a
        self.user_id_b = user_id_b
        self.matrix = np.empty((num_rating_values, num_rating_values,))
        self.matrix[:] = 0
        self.calculate_matrix(user_id_a, user_id_b)

    def get_shape(self):
        a = self.matrix.shape
        return a

    def get_matrix(self):
        return self.matrix

    def calculate_matrix(self, user_id_a, user_id_b):
        for item in items_reviewed(user_id_a, userdict):
            if int(item) in items_reviewed(user_id_b, userdict):
                i = get_score_item_reviewed(user_id_a, item, userdict)-1
                j = get_score_item_reviewed(user_id_b, item, userdict)-1
                self.matrix[i][j] +=1

    def get_total_count(self):
        return self.matrix.sum()

    def get_agreement_count(self):
        return np.trace(self.matrix) # Suma wzduż przekątnej
```

Klasa RatingCountMatrix służy do obliczania podobieństwa między użytkownikami na podstawie produktów ocenionych przez obie osoby i poziomu zgodności ocen. Widać tu, że nie wystarczy, by obie osoby ocenili ten sam film; użytkownicy muszą się też mniej więcej zgadzać w ocenach. W klasie RatingCountMatrix jest to reprezentowane za pomocą macierzy określającej kowariancję ocen. Jeśli użytkownicy zawsze oceniają obejrzane filmy w podobny sposób, przekątna macierzy (elementy  $M_{ij}$ , gdzie  $i = j$ ) będzie zapełniona wysokimi wartościami. W przypadku osób, które mają różne opinie, wysokie wartości będą znajdowały się daleko od przekątnej (będą to elementy  $M_{ij}$ , gdzie  $i = \min(ocena), j = \max(ocena)$  lub  $i = \max(ocena), j = \min(ocena)$ ).

Obliczanie podobieństwa na podstawie takich macierzy odbywa się intuicyjnie, co widać na listingu 3.6. Można uwzględnić łączną liczbę zgodnych ocen podzieloną przez łączną liczbę wszystkich wspólnych ocen. Ponownie zwróci uwagę na to, że wydajność

obliczeń jest wysoce nieoptymalna, ponieważ obliczenia w klasie RatingCountMatrix trzeba powtórzyć za każdym razem, gdy określany jest poziom podobieństwa między dwoma użytkownikami. W tym przykładzie jest to akceptowalne, jednak w większych systemach taka wydajność jest nie do przyjęcia.

Do tej pory koncentrowaliśmy się na systemie filtrowania kolaboratywnego według użytkowników. Nie powinieneś mieć trudności z wyobrażeniem sobie, jak odwrócić ten system (tak aby użytkownicy byli traktowani jak produkty, a produkty — jak użytkownicy). Zamiast prezentować kompletne omówienie takiego odwróconego rozwiązania, opracowanie systemu filtrowania kolaboratywnego według produktów pozostawiamy jako ćwiczenie dla Czytelników. Teraz przejdziemy do innej wersji filtrowania kolaboratywnego — według modelu.

Używany tu model wykorzystuje *rozkład według wartości osobliwych* (rozkład SVD). Techniki oparte na modelu dają duże możliwości. W tym przykładzie pozwalają zrozumieć głębsze relacje między użytkownikami i filmami. Przy stosowaniu rozkładu SVD model obejmuje trzecią, nieobserwowną przestrzeń — *przestrzeń zmiennych ukrytych*. Pozwala to uwzględnić fakt, że występuje pewien nieobserwowny powód, dla którego użytkownicy dokonują określonych wyborów (tą przyczyną może być na przykład gustowanie w filmach określonego gatunku). Zamiast ignorować ten powód (jak dzieje się to w filtrowaniu kolaboratywnym według użytkowników i produktów), w tym podejściu próbujemy go uchwycić.

### **3.5. Rekomendacje według modelu z wykorzystaniem rozkładu SVD**

W poprzednim przykładzie grupowaliśmy użytkowników w taki sposób, by najbardziej podobni użytkownicy wpływali na prognozy ocen danej osoby. W tym podejściu (podobnie jak przy filtrowaniu kolaboratywnym według produktów) nie przyjmujemy żadnych założeń co do zależności między użytkownikami i produktami. Na tym polega główna różnica między filtrowaniem kolaboratywnym według użytkowników i produktów a filtrowaniem kolaboratywnym według modelu.

W podejściu opartym na modelu uwzględniane są dodatkowe założenia (model). Gdy używany jest rozkład SVD, zakładamy, że między produktami a użytkownikami występują powiązania w przestrzeni zmiennych ukrytych, niewidocznej w samych danych użytkowników lub produktów. Nie martw się, jeśli wydaje Ci się to niejasne. Oznacza to tylko tyle, że użytkownicy są odwzorowywani na produkty, a produkty na użytkowników w innej, trzeciej przestrzeni. Możesz uznać, że jest to przestrzeń gatunków filmów, choć technicznie nie jest to precyzyjne określenie. Jeśli algorytm ustali, że użytkownik w równym stopniu lubi dramaty, romanse i filmy akcji, a dany obraz należy do jednej z tych kategorii, ten obraz może zostać zarekomendowany określonej osobie (pod warunkiem, że jeszcze go nie widziała). W praktyce przestrzeń zmiennych ukrytych nie jest opisana jako przestrzeń gatunków, tylko jest wykrywana automatycznie przez algorytm. To powinno pomóc Ci zrozumieć, jak działa mechanizm rekomendacji oparty na rozkładzie SVD. Jeśli jednak ten wstęp nie był pomocny, nie martw się — w tym podrozdziale szczegółowo omawiamy to podejście.

### 3.5.1. Rozkład SVD

Rozkład SVD to technika rozkładu macierzy na czynniki. *Rozkład na czynniki* polega na podziale danej jednostki w taki sposób, że po pomnożeniu uzyskanych elementów otrzymywana jest pierwotna jednostka. Rozkład liczb na czynniki jest stosunkowo prosty (10 można na przykład rozłożyć na  $2 \times 5$ ), jednak mnożenie macierzy trudniej jest przeprowadzić w myślach.

Z encyklopedii Wolfram MathWorld<sup>5</sup> można się dowiedzieć, że macierz  $\mathbf{A}$  o  $m$  wierszach i  $n$  kolumnach, gdzie  $m > n$ , można zapisać w następującej postaci:

$$\mathbf{A} = \mathbf{UDV}^T$$

W tym wzorze  $\mathbf{U}$  ma wymiary  $m \times m$ ,  $\mathbf{D} — m \times n$ , a  $\mathbf{V} — n \times n$ .  $\mathbf{D}$  zawiera tylko elementy wzdłuż przekątnej, a  $\mathbf{U}$  i  $\mathbf{V}$  obejmują ortogonalne kolumny ( $\mathbf{U}^T \mathbf{U} = \mathbf{I} = \mathbf{V}^T \mathbf{V}$ ). Ta ostatnia cecha jest ważna, ponieważ sprawia, że kierunki w przestrzeni zmiennych ukrytych też są ortogonalne. Oznacza to, że nie można ich przedstawić za pomocą liniowych kombinacji innych kierunków. Dalej wróćmy do tego zagadnienia.

Możesz się zastanawiać, co to wszystko ma wspólnego z systemami rekomendacji. Okazuje się, że całkiem dużo. Założymy, że wymiary macierzy  $\mathbf{A}$  to  $m \times n$ , gdzie  $m$  to liczba użytkowników, a  $n$  to liczba filmów. Wartość w macierzy oznacza ocenę, jaką użytkownik  $m$  przyznał filmowi  $n$ . Co się stanie, jeśli uda się rozłożyć tę macierz na macierz  $\mathbf{U}$  o wymiarach  $m \times r$ , która zawiera informacje o stosunku użytkowników do elementów z nowej, mniejszej przestrzeni o wielkości  $r$ ? Podobnie można utworzyć macierz  $\mathbf{V}^T$  o wymiarach  $r \times n$ , określającą powiązanie filmów z poszczególnymi wymiarami z przestrzeni  $r$ . Należy też spróbować ustalić, jak ważny jest każdy wymiar z przestrzeni  $r$  ze względu na prognozowanie ocen przyznawanych produktom przez użytkowników. Uzyskane wyniki to *wartości osobliwe* rozłożonej na czynniki macierzy, umieszczane na przekątnej macierzy  $\mathbf{D}$  w porządku nierośnącym z odpowiednio uporządkowanymi kolumnami i wierszami z macierzy użytkowników ( $\mathbf{U}$ ) i produktów ( $\mathbf{V}^T$ ).

Zauważ, że w tym przykładzie przestrzeń zmiennych ukrytych ma rozmiar  $r$ . Możesz zdecydować się zachować tylko pierwszych  $k$  wartości osobliwych, aby ująć najważniejsze wymiary z przestrzeni zmiennych ukrytych w zwięzły sposób. Ten proces przypomina redukcję wymiarów za pomocą dekompozycji z użyciem wartości własnych, który opisaliśmy w rozdziale 2. To nie przypadek! Jeśli interesuje Cię ta tematyka, w tekście Gerbrandsa<sup>6</sup> znajdziesz więcej informacji na temat związków między tymi zagadnieniami.

Zastanów się nad ortogonalnością przestrzeni zmiennych ukrytych w macierzach  $\mathbf{U}$  i  $\mathbf{V}$ . Jest ona zrozumiała. Zastanów się na przykład nad macierzą  $\mathbf{U}$ . Jeśli wybierzesz dwie dowolne kolumny, iloczyn skalarny odpowiadających im wektorów będzie równy zero. Cechy ukryte są tu wybierane w taki sposób, aby były możliwie odległe od siebie ze względu na widzów (są wtedy maksymalnie opisowe). Możesz też zauważać, że sensowny jest wybór cech, które są mocno oddalone od siebie w przestrzeni użytkowników.

<sup>5</sup> Singular Value Decomposition, „Wolfram MathWorld”, 2015, <http://mng.bz/TxyY>.

<sup>6</sup> Jan J. Gerbrands, *On the Relationships between SVD, KLT and PCA*, Conference on Pattern Recognition (1980).

Jeśli miłośnicy fantastyki naukowej nigdy nie oglądają komedii romantycznych i na odwrót, ta cecha jest przydatna w trakcie generowania rekomendacji. Wiadomo również, że także dwie dowolne kolumny macierzy  $V$  (jak i dwa dowolne wiersze macierzy  $V^T$ ) są ortogonalne. Cechy ukryte są więc dobierane tak, by były możliwe oddalone od siebie ze względu na przynależność filmów do grup. Wróćmy do niedoskonalej analogii z gatunkami filmów. Gatunki w przestrzeni filmów są znacznie oddalone od siebie. Na przykład filmy z gatunku fantastyki naukowej zwykle nie są komedią. Warto przypomnieć, że wykrywane cechy nie są gatunkami (którymi posługujemy się w celach demonstracyjnych), tylko innymi właściwościami ujmującymi zależność między filmami i widzami.

Wróćmy do przykładu. Po rozłożeniu macierzy  $A$  na czynniki bardzo łatwo jest dostrzec, jak obliczać poszczególne oceny. Sprawdzana jest pozycja użytkownika w  $r$ -wymiarowej przestrzeni zmiennych ukrytych (przypomina to ustalanie procentowego zainteresowania każdym z  $r$  różnych ortogonalnych gatunków), po czym kod mnoży każde  $r$  przez jego względne znaczenie. W ten sposób uzyskujemy wektor o długości  $r$ . W ostatnim kroku współrzędne z  $r$  wymiarów są badane dla produktu, którego ocenę kod ma przewidzieć (przypomina to ustalanie procentowego powiązania produktu z  $r$  różnymi ortogonalnymi gatunkami), po czym obliczany jest iloczyn skalarny. Otrzymujemy w ten sposób liczbę, która jest prognozowaną oceną danego filmu przez określonego użytkownika. Zauważ, że choć posługujemy się tu gatunkami, robimy to tylko w celach ilustracyjnych. Choć czasem po treningu możliwe jest przypisanie sensu wymiarom, w rzeczywistości nie mają one jawnie określonego znaczenia.

Trening systemu rekomendacji polega tu na rozkładzie macierzy na czynniki. Rekomendacje można następnie sprawdzać bezpośrednio w rozłożonej na czynniki macierzy, zgodnie z opisem z wcześniejszego akapitu. Jeśli wydaje Ci się to zaskakujące, zauważ, że rozkład na czynniki ma pozwolić znaleźć powiązania między użytkownikami, filmami i przestrzenią zmiennych ukrytych, którą według algorytmu należy uwzględnić. To rozwiązanie próbuje przeprowadzić sensowny rozkład i zminimalizować poziom błędów, aby uzyskać model podstawowych relacji. Jeśli się nad tym uważnie zastanowisz, zauważysz, że rekomendacje są zależne od błędu w rozkładzie na czynniki. Omawiamy tę kwestię szczegółowo w następnym punkcie, gdzie zapoznasz się z systemem rekomendacji opartym na rozkładzie SVD.

### **3.5.2. Rekomendacje z użyciem rozkładu SVD – wybór filmów dla danego użytkownika**

W tym punkcie posłużymy się rozkładem SVD do wygenerowania rekomendacji filmów dla użytkowników. Tak jak wcześniej używany jest tu zbiór danych MovieLens. Jednak tym razem nie wybieramy tylko kilku filmów i użytkowników, ale wykorzystujemy wszystkie dostępne dane. Warto przypomnieć, że zbiór danych MovieLens zawiera liczne oceny filmów dokonane przez użytkowników. Wykorzystamy możliwości rozkładu SVD do uzyskania macierzy przestrzeni zmiennych ukrytych, co pozwoli wykryć ewentualne zależności między użytkownikami na podstawie filmów, które zostały przez nich wysoko ocenione.

Przedstawiamy tu także nowy pakiet inny niż scikit-learn. Jest to pakiet python-reccsys<sup>7</sup>. Jest on prosty, a przy tym zapewnia wystarczająco duże możliwości — pozwala pokazać, jak generować rekomendacje za pomocą rozkładu SVD. Dlatego w tym punkcie używamy właśnie tego pakietu. Spójrz na listing 3.8. Przed uruchomieniem przykładowego kodu zainstaluj wspomniany pakiet.

#### **Listing 3.8. Generowanie rekomendacji za pomocą rozkładu SVD**

```
svd = SVD()
recsys.algorithm.VERBOSE = True ←———— Powoduje wyświetlanie na ekranie informacji o postępie

dat_file = './ml-1m/ratings.dat' ←———— Oceny przyznane przez użytkowników
item_file = './ml-1m/movies.dat' ←———— Informacje o filmach

data = Data()
data.load(dat_file, sep='::',
          format={'col':0, 'row':1, 'value':2, 'ids': int})

items_full = read_item_data(item_file)
user_full = read_user_data_from_ratings(dat_file)

svd.set_data(data)
```

Ten listing przedstawia na ogólnym poziomie, jak użyć pakietu recsys do uzyskania rozkładu SVD dla dostępnego bezpłatnie zbioru danych MovieLens ml-1m (powinieneś go pobrać). Ten zbiór danych obejmuje milion ocen przyznanych przez użytkowników. Wykorzystamy je wszystkie do zbudowania systemu rekomendacji. Najpierw należy utworzyć obiekt svd oraz wczytać oceny przyznane filmom przez użytkowników i dodatkowe informacje o filmach (gatunek, rok produkcji itd.). Choć w naszym modelu te dodatkowe dane nie są używane, można je wykorzystać do określenia, na ile sensowne są zwarcane rekomendacje. W ostatnim wierszu do obiektu svd przekazywany jest tylko plik z danymi (o określonym formacie).

Tak jak w poprzednim przykładzie na razie uczenie jeszcze się nie zaczęło. Dane zostały przekazane do obiektu, który będzie odpowiadał za uczenie, ale same obliczenia nie zostały przeprowadzone. Listing 3.9 przedstawia kod potrzebny do treningu modelu, a dokładniej — do rozkładu na czynniki macierzy utworzonej za pomocą metody `load_klasy Date`.

#### **Listing 3.9. Rozkład macierzy na czynniki — używanie rozkładu SVD do generowania rekomendacji**

```
k = 100
svd.compute(k=k, min_values=10,
            pre_normalize=None, mean_center=True, post_normalize=True)

films = svd.recommend(10, only_unknowns=True, is_row=False)
```

---

<sup>7</sup> ACM RecSys Wiki, <http://www.recsyswiki.com/wiki/Python-reccsys>.

Wywoływana jest tu metoda `svd.compute` z pięcioma parametrami. Sterują one przebiegiem rozkładu macierzy na czynniki. Aby omówienie było kompletne, przytaczamy tu dokumentację tej metody<sup>8</sup> i objaśniamy wpływ poszczególnych opcji na rozkład:

- `min_values` — pozwala usunąć wiersze lub kolumny obejmujące mniej niż `min_values` wartości niezerowych.
- `pre_normalize` — normalizuje wejściową macierz. Możliwe wartości tej opcji to:
  - `tfidf` (to wartość domyślna) — macierz jest traktowana jak wykaz pojęć w dokumentach (ang. *terms-by-documents*). Dlatego ważne jest, jak dane są wczytywane. Za pomocą parametru `format` metody `svd.load_data()` możesz określić kolejność pól w pliku wejściowym.
  - `rows` — przeskaliwuje wiersze wejściowej macierzy względem jednostkowej odległości euklidesowej.
  - `cols` — przeskaliwuje kolumny wejściowej macierzy względem jednostkowej odległości euklidesowej.
  - `all` — przeskaliwuje wiersze i kolumny wejściowej macierzy, dzieląc je przez pierwiastek kwadratowy ich normy euklidesowej.
- `mean_center` — centrowanie macierzy wejściowej (metodą odejmowania średnich).
- `post_normalize` — normalizuje każdy wiersz macierzy **UD** do postaci wektora jednostkowego. Dlatego podobieństwo między wierszami (mierzzone odlegością kosinusową) przyjmuje wartości z przedziału  $[-1,0 \dots 1,0]$ .
- `savefile` — plik wyjściowy, w którym zapisywane są transformacje oparte na rozkładzie SVD (wektory **U**, **D** i  $\mathbf{V}^T$ ).

Widać tu, że większość opcji służy do kontrolowania wstępnego lub końcowego przetwarzania danych. Za pomocą pierwszego parametru można usunąć filmy i użytkowników o mniej niż 100 ocenach. To powinno znacznie zmniejszyć wielkość macierzy i pozwolić skoncentrować się tylko na tych filmach lub użytkownikach, dla których dostępne są dane wystarczające do uzyskania rekomendacji. Drugi i trzeci parametr umożliwiają normalizację i centrowanie danych. Normalizacja jest ważna, ponieważ zapewnia gwarancje dotyczące wariancji danych. Centrowanie zmienia punkt odniesienia dla tej wariancji. Oba parametry pomagają zapewnić stabilność algorytmu wykonyjącego rozkład SVD. Tu stosowana jest wykorzystująca jeden wektor metoda Lanczosa<sup>9</sup> wywoływana za pomocą biblioteki SVDLIBC<sup>10</sup>.

Wróćmy do przykładu i wygenerujmy rekomendacje za pomocą omawianego modelu. W ostatnim wierszu listingu 3.9 wywoływana jest metoda `svd.recommend()`, do której przekazywane są trzy parametry. Pierwszy parametr określa użytkownika, dla którego generowane są rekomendacje (tu wybraliśmy użytkownika 10). Przekazywane są też dwa dodatkowe parametry, `only_unknowns=True` i `is_row=False`. Pierwszy z nich informuje

<sup>8</sup> Oscar Celma, wersja 1.0 dokumentacji pakietu Python Recsys, <http://mng.bz/UBiX>.

<sup>9</sup> C. Lanczos, *An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators*, „Journal of Research of the National Bureau of Standards” 45, nr 4 (październik 1950), <http://mng.bz/cB2d>.

<sup>10</sup> Doug Rohde, SVDLIBC, <http://tedlab.mit.edu/~dr/SVDLIBC/>.

metodę, że należy ustalić oceny tylko dla tych filmów, które nie zostały ocenione przez użytkownika 10. Drugi związek jest z osobliwą cechą używanej biblioteki. Ustawienie `is_row=False` określa, że rekomendacje są określone na podstawie kolumn, czyli użytkowników, a nie produktów (reprezentowanych wierszach). Ponieważ używany tu model wykorzystuje rozkład macierzy na czynniki, można też podać identyfikator produktu (filmu) i ustawić opcję `is_row=True`, aby uzyskać listę użytkowników, którym warto zarekomendować dany film. W odróżnieniu od opisanych we wcześniejszej części rozdziału metod opartych na użytkownikach i produktach ta technika pozwala uzyskać podobne wyniki dla obu tych grup bez konieczności wprowadzania znacznych zmian w kodzie.

Wystarczy już tych szczegółów — co z wynikowymi rekomendacjami? Przyjrzymy się im. Listing 3.10 przedstawia rekomendowane filmy.

#### **Listing 3.10. Dane wyjściowe systemu rekomendacji opartego na modelu**

```
[items_full[str(x[0])].get_data() for x in films]

[{'Genres': 'Action|Adventure\n', 'Title': 'Sanjuro (1962)'},  
 {'Genres': 'Crime|Drama\n', 'Title': 'Pulp Fiction (1994)'},  
 {'Genres': 'Crime|Thriller\n', 'Title': 'Usual Suspects. The (1995)'},  
 {'Genres': 'Comedy|Crime\n', 'Title': 'Some Like It Hot (1959)'},  
 {'Genres': 'Drama\n', 'Title': 'World of Apu, The (Apur Sansar) (1959)'},  
 {'Genres': 'Documentary\n', 'Title': 'For All Mankind (1989)'},  
 {'Genres': 'Animation|Comedy\n', 'Title': 'Creature Comforts (1990)'},  
 {'Genres': 'Comedy|Drama|Romance\n', 'Title': 'City Lights (1931)'},  
 {'Genres': 'Drama\n', 'Title': 'Pather Panchali (1955)'},  
 {'Genres': 'Drama\n',  
 'Title': '400 Blows. The (Les Quatre cents coups) (1959)'}]
```

Widac tu, że użytkownikowi 10 zarekomendowane zostały filmy z gatunków filmy akcji/przygodowe, kryminalny/dramaty i kryminalny/thrillery. Wygląda na to, że ten użytkownik lubi też komedie i filmy animowane. Lista wygląda dobrze, ale na razie nie wiemy nic o tym użytkowniku. Kod z listingu 3.11 sprawdza oceny użytkownika 10, co pozwala zobaczyć, jakie filmy ta osoba obejrzała w przeszłości i jak się jej one spodobały.

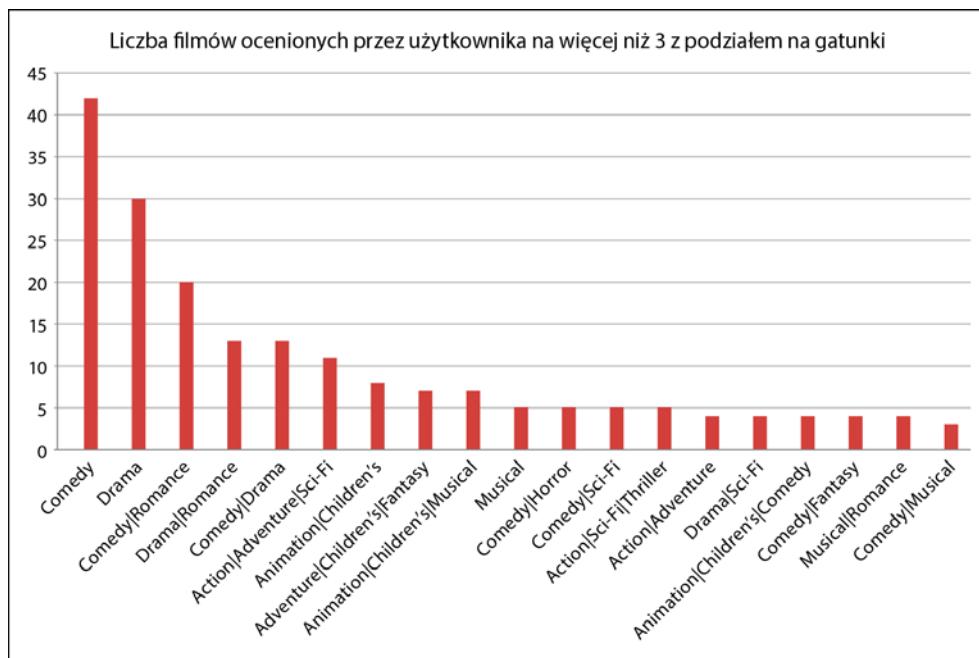
#### **Listing 3.11. Filmy ocenione przez użytkownika 10**

```
get_name_item_reviewed(10,user_full.items_full)

[(2622, "Midsummer Night's Dream, A (1999)", 'Comedy|Fantasy\n', 5.0),  
 (3358, 'Defending Your Life (1991)', 'Comedy|Romance\n', 5.0),  
 (1682, 'Truman Show, The (1998)', 'Drama\n', 5.0),  
 (2125, 'Ever After: A Cinderella Story (1998)', 'Drama|Romance\n', 5.0),  
 (1253, 'Day the Earth Stood Still, The (1951)', 'Drama|Sci-Fi\n', 5.0),  
 (720,  
 'Wallace and Gromit: The Best of Aardman Animation (1996)',  
 'Animation\n',  
 5.0),  
 (3500, 'Mr. Saturday Night (1992)', 'Comedy|Drama\n', 5.0),  
 (1257, 'Better Off Dead... (1985)', 'Comedy\n', 5.0),
```

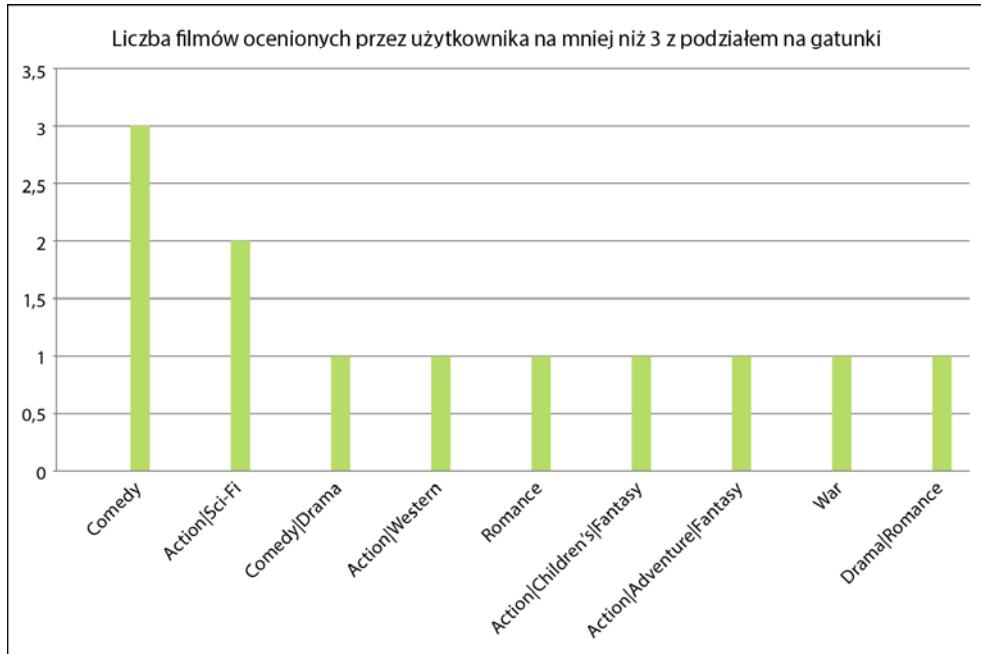
- (3501, "Murphy's Romance (1985)", 'Comedy|Romance\n', 5.0),  
 (1831, 'Lost in Space (1998)', 'Action|Sci-Fi|Thriller\n', 5.0),  
 (3363, 'American Graffiti (1973)', 'Comedy|Drama\n', 5.0),  
 (587, 'Ghost (1990)', 'Comedy|Romance|Thriller\n', 5.0),  
 (150, 'Apollo 13 (1995)', 'Drama\n', 5.0),  
 (1, 'Toy Story (1995)', "Animation|Children's|Comedy\n", 5.0),  
 (2, 'Jumanji (1995)', "Adventure|Children's|Fantasy\n", 5.0), ...

Jest to tylko bardzo niewielki wycinek z 401 filmów ocenionych przez tego użytkownika. Na liście znajdują się jednak filmy animowane i komedie (obrazy *Wallace and Gromit* oraz *Creature Comforts* zostały wyprodukowane przez tę samą wytwórnię, Aardman Animations). Ponadto i na liście ocenionych filmów, i wśród rekomendacji występują komedie i dramaty. Zauważ jednak, że użytkownik przyznał 154 oceny 5, dlatego ta krótka lista nie reprezentuje szczegółowo zainteresowań tej osoby. Przyjrzyjmy się więc dokładniej danym i zobaczymy rozkład gatunków filmów obejrzanych przez użytkownika 10. Na rysunku 3.3 przedstawione są gatunki obejmujące filmy, którym użytkownik przyznał ocenę wyższą niż 3. Rysunek 3.4 przedstawia gatunki zawierające filmy, które ta osoba oceniła na mniej niż 3. Filmy ocenione na 3 są pomijane.



Rysunek 3.3. Liczba filmów ocenionych przez użytkownika 10 na 4 lub 5. Filmy są podzielone na gatunki, co oznacza, że użytkownik ocenił ponad 40 filmów z gatunku Comedy na 4 lub 5. Wyświetlonych jest tylko 19 najczęściej występujących na liście gatunków

Teraz wyniki zaczynają nabierać sensu. Rysunek 3.3 pokazuje, że użytkownik najwyraźniej lubi gatunki Comedy i Drama, ponieważ to z nich pochodzi zdecydowana większość pozytywnie ocenionych przez tę osobę filmów. Gdy wrócisz do rekomendacji



**Rysunek 3.4.** Liczba podzielonych na gatunki filmów, którym użytkownik 10 przyznał oceny niższe niż 3. Są to trzy filmy z gatunku Comedy, dwa z gatunku Action/Sci-Fi, a także pojedyncze filmy z kilku innych kategorii

z listingu 3.10, zobaczyś, że są one z tym zgodne. Siedem z dziesięciu proponowanych filmów jest z gatunku Comedy lub Drama. Także gatunki Romance, Action/Adventure/Fantasy i Animation są często pozytywnie oceniane przez użytkownika i występują na liście rekomendacji. Zaskakujące może być uwzględnienie w propozycjach filmów z gatunku Thriller, ponieważ nie występują one wysoko (lub w ogóle ich nie ma) na liście rzeczywistych ocen. Można to jednak wyjaśnić działaniem filtrowania kolaboratywnego.

W używanym zbiorze danych występują filmy wysoko ocenione przez użytkownika i niżej ocenione obrazy z różnych gatunków. Algorytm wykrył w danych wzorce i zarekomendował nowe filmy także z nieznanych użytkownikowi gatunków. Nie jest to zaskoczeniem, ponieważ algorytm z systemu rekomendacji powinien próbować generować propozycje wykraczające poza oczywiste możliwości. Po zastanowieniu okazuje się, że proponowane filmy nie są zaskakujące. Miłośnikom dramatów może czasem spodobać się kryminał lub thriller.

Na rysunku 3.4 pokazany jest histogram gatunków filmów, które użytkownik 10 ocenił nisko. Widać, że filmy z gatunku Comedy i Action/Sci-fi otrzymały przynajmniej po dwie niskie oceny. Należy jednak zachować ostrożność w trakcie interpretowania tych wyników. Przyjrzyj się skalom z rysunków 3.3 i 3.4. Użytkownik 10 zdecydowanie ma skłonność do przyznawania filmom wysokich ocen i lubi komedie (ponad 40 z nich ocenił pozytywnie). Istnieją techniki, które pozwalają uwzględniać takie

skłonności w filtrowaniu kolaboratywnym opartym na modelu. Jednak zamiast oma-wać tu te metody, odsyłamy Czytelników do literatury<sup>11</sup>.

### **3.5.3. Rekomendacje z wykorzystaniem rozkładu SVD – określanie użytkowników, których może zainteresować dany film**

Pokazaliśmy już, jak za pomocą rozkładu SVD zarekomendować filmy użytkownikom. Dzięki elastyczności filtrowania kolaboratywnego opartego na modelu w równie prosty sposób można ustalić, jakim użytkownikom może spodobać się dany film (czyli którym osobom należy go zarekomendować). Łatwo się domyślić, że to podejście może być przydatne w świecie reklamy, gdzie należy ustalić, którym użytkownikom warto zareklamować dany produkt.

Załóżmy, że pracujesz w wytwórni LucasFilm i odpowiadasz za promocję nowego filmu z serii *Gwiezdne wojny — Star Wars, Episode 1: The Phantom Menace* (1999). Jak przy użyciu rozkładu SVD i istniejącego zbioru danych znaleźć użytkowników, którym warto zaproponować ten obraz? Odpowiedź znajdziesz na listingu 3.12.

#### **Listing 3.12. Używanie filtrowania kolaboratywnego według modelu do znalezienia użytkowników, którym warto zarekomendować dany film**

```
> items_full[str(2628)].get_data()
{'Genres': 'Action|Adventure|Fantasy|Sci-Fi\n',
 'Title': 'Star Wars: Episode I - The Phantom Menace (1999)'}
> users_for_star_wars = svd.recommend(2628,only_unknowns=True)
> users_for_star_wars
[(446, 4.7741731815492816),
 (3324, 4.7601341930085157),
 (2339, 4.7352608789782398),
 (1131, 4.6541316195384743),
 (5069, 4.6479235651508217),
 (4755, 4.6444117760840502),
 (4634, 4.6308837065012067),
 (4649, 4.619985795550809),
 (1856, 4.5846499038453166),
 (4273, 4.5803152198983419)]
```

Najpierw kod sprawdza, czy rekomendacje będą dotyczyć odpowiedniego filmu. Tu produkt o identyfikatorze 2628 oczywiście jest odpowiednim filmem z serii *Gwiezdne wojny*. Wywołanie metody `svd.recommend()` z tym identyfikatorem i bez opcji `is_row=False` powoduje wygenerowanie rekomendacji pod kątem wierszy (czyli użytkowników). Zwarcane są identyfikatory pierwszych 10 użytkowników wraz z prognozowanymi ocenami, jakie te osoby wystawią danemu filmowi.

Przyjrzyj się teraz listingowi 3.13, aby zobaczyć, czy te rekomendacje są intuicyjnie sensowne.

Kod najpierw pobiera filmy ocenione przez osoby, którym zarekomendowano dany obraz, po czym lista tych filmów jest spłaszczana. Należy spłaszczyć listę `list` (na której

<sup>11</sup> Arkadiusz Paterek, *Improving Regularized Singular Value Decomposition for Collaborative Filtering*, „Proceedings of the ACM SIGKDD” (2007), <http://mng.bz/TQnD>.

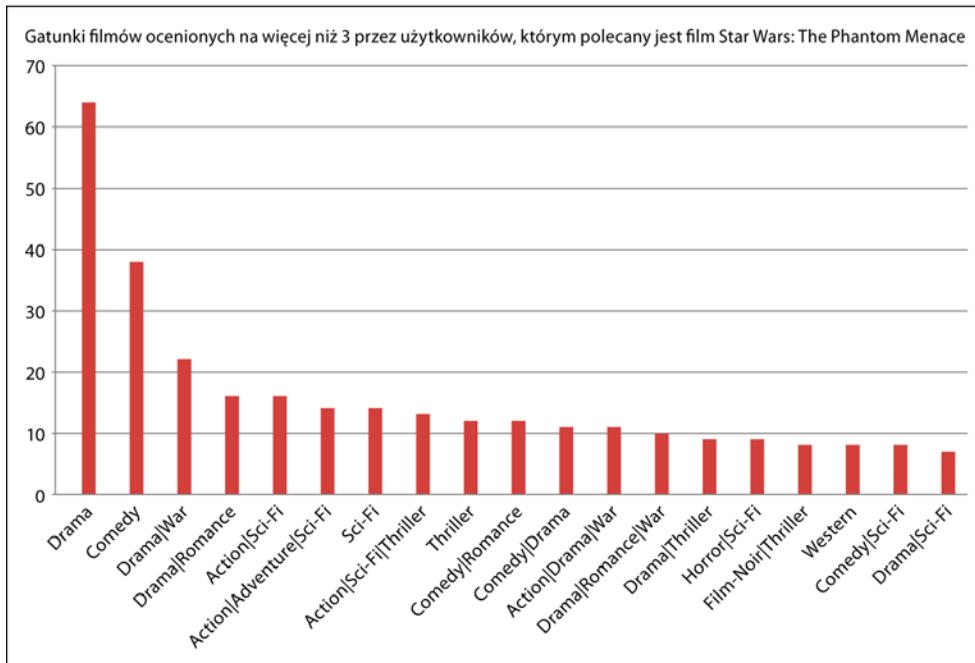
**Listing 3.13. Analizowanie ocen użytkowników, którym algorytm zarekomendował film „Star Wars: TPM”**

```
> movies_reviewed_by_sw_rec =[get_name_item_reviewed(x[0],  
user_full,items_full) for x in users_for_star_wars]  
> movies_flatten = [movie for movie_list in movies_reviewed_by_sw_rec for  
<movie in movie_list]  
> movie_aggregate = movies_by_category(movies_flatten, 3)  
> movies_sort = sorted(movie_aggregate,key=lambda x: x[1], reverse=True)  
> movies_sort  
  
[['Drama\n', 64],  
 ['Comedy\n', 38],  
 ['Drama|War\n', 22],  
 ['Drama|Romance\n', 16],  
 ['Action|Sci-Fi\n', 16],  
 ['Action|Adventure|Sci-Fi\n', 14],  
 ['Sci-Fi\n', 14],  
 ['Action|Sci-Fi|Thriller\n', 13],  
 ['Thriller\n', 12],  
 ['Comedy|Romance\n', 12],  
 ['Comedy|Drama\n', 11],  
 ['Action|Drama|War\n', 11],  
 ['Drama|Romance|War\n', 10],  
 ['Drama|Thriller\n', 9],  
 ['Horror|Sci-Fi\n', 9],  
 ['Film-Noir|Thriller\n', 8],  
 ['Western\n', 8],  
 ['Comedy|Sci-Fi\n', 8],  
 ['Drama|Sci-Fi\n', 7]...]
```

każdy element to lista filmów ocenionych przez daną osobę) do postaci obejmującej oceny wszystkich uwzględnianych użytkowników. Metoda `movies_by_category` tworzy listę zawierającą gatunki filmów i liczbę ocen, jaką filmom z tych kategorii wystawili analizowani użytkownicy. Parametr tej metody umożliwia filtrowanie filmów na podstawie wysokości ocen. Tu uwzględniane są tylko te filmy, które uzyskały ocenę 3 lub wyższą. W ostatnim kroku wyjściowa lista jest sortowana, tak aby gatunki z największą liczbą ocenionych filmów znalazły się na początku.

Zbudowaliśmy więc listę gatunków najlepiej ocenianych przez zbiór użytkowników, którym program chce polecić film *Star Wars: The Phantom Menace*. Rysunek 3.5 przedstawia tę listę w formie graficznej.

Warto zwrócić tu uwagę na kilka kwestii. Po pierwsze, użytkownicy z tego zagregowanego zbioru wyraźnie lubią filmy z gatunku Sci-fi. Aż 4 z 10 czołowych kategorii zawiera w nazwie gatunku człon *Sci-fi* oznaczający fantastykę naukową. Intuicyjnie jest to zgodne z oczekiwaniemi. Seria *Gwiezdne wojny* należy do tego gatunku, dlatego uzasadnione jest, że nowy film zostanie zarekomendowany miłośnikom fantastyki naukowej. Nieco zaskakujące może być to, że dwa najpopularniejsze gatunki wśród użytkowników z tej grupy to dramat i komedia. Na pozór jest to niezrozumiałe, pamiętaj jednak, że do tych gatunków należy bardzo dużo filmów. Pamiętaj też, że filmy z tych gatunków mogą być oceniane wysoko przez wszystkich użytkowników ze zbioru danych, co jest następnym przykładem błędu systematycznego.



Rysunek 3.5. Liczba ocen wyższych niż 3 zagregowanych według gatunków. Uwzględnieni są wszyscy użytkownicy ze zbioru osób, którym ma zostać polecony film „Star Wars: The Phantom Menace”

W poprzednich punktach wprowadziliśmy Cię w świat systemów rekomendacji i przedstawiliśmy szczegóły systemów opartych na użytkownikach i modelu. Do wyboru filmów dla użytkownika i użytkowników dla filmu używaliśmy zbioru danych MovieLens (jest to niekomercyjny zbiór danych obejmujący milion ocen wystawionych przez 6000 użytkowników 4000 filmów).

Choć filtrowanie kolaboratywne według użytkowników jest proste do zrozumienia i wyjaśnienia (łatwo jest wyjaśnić użytkownikowi, dlaczego polecono mu dany film — ponieważ dana osoba jest podobna do innych, którym ten film się spodobał), może prowadzić do problemów, gdy dane są rzadkie, czyli gdy liczba filmów i użytkowników jest duża, ale zestaw filmów ocenianych przez różne osoby pokrywa się w niewielkim stopniu. Natomiast filtrowanie kolaboratywne według modelu nie umożliwia łatwego wyjaśnienia rekomendacji, ale może sprawdzać się lepiej, gdy dane są rzadkie. Jest tak, ponieważ uwzględniana jest przestrzeń zmiennych ukrytych, łącząca użytkowników z filmami. Jak zawsze to osoba stosująca uczenie maszynowe musi wybrać rozwiązanie, które najlepiej pasuje do wymogów zadania.

### 3.6. Konkurs Netflix Prize

Trudno jest pisać o systemie rekomendacji i nie wspomnieć przy tym o Netfliksie. Przeprowadzony w 2009 roku konkurs Netflix Prize okazał się wielkim sukcesem. Uczestnicy ze 186 państw nadesłali 41 000 propozycji<sup>12</sup>. Zadanie polegało na uzyskaniu najlepszego pierwiastka błędu średniokwadratowego (ang. *root-mean square error* — **RMSE**) na podstawie rekomendacji filmów użytkownikom według ich wcześniejszych ocen. Zespoły zgłaszały nowe rozwiązania oceniane za pomocą publicznego znanego zbioru testowego (różnego od nieujawnianego zbioru testowego stosowanego przez firmę Netflix). Zgłoszenia napływały do ostatnich minut konkursu, który okazał się bardzo ekscytujący! Równie ciekawa jest historia prowadząca do opracowania algorytmów.

Ostatecznie nagrodę miliona dolarów przyznano zespołowi BellKor's Pragmatic Chaos. Składał się on z trzech czołowych uczestników konkursu: Team Pragmatic Theory, Team BellKor i Team Big Chaos. Opracowany przez nich algorytm był lepszy o 10,06% od napisanego w firmie Netflix algorytmu Cinematch, co zapewniło im nagrodę dla pierwszego zespołu, który przekroczy poziom 10% poprawy w stosunku do algorytmu Netfliksa. Zamiast opisywać szczegółowo zwycięskiego rozwiązania, wyjaśniamy je na ogólnym poziomie i omawiamy jego wpływ. Osobom zainteresowanym szczegółami gorąco polecamy teksty: „The BellKor Solution to the Netflix Grand Prize”<sup>13</sup>, „The BellKor 2008 Solution to the Netflix Prize”<sup>14</sup> i „The Pragmatic Theory Solution to the Netflix Grand Prize”<sup>15</sup>. Te teksty zebrane razem stanowią kompletną dokumentację zwycięskiego podejścia.

Zespół połączył setki pojedynczych predyktorów, aby uzyskać wyniki skuteczniejsze od każdego z nich osobno. Dzięki połączeniu wielu algorytmów każdy z nich ujmował specyficzne aspekty zależności między użytkownikami i produktami, a zespołowi udało się otrzymać wyjątkowe wyniki. Ciekawym aspektem tego konkursu jest to, że najlepsze wyniki zostały uzyskane po tym, jak trzy zespoły połączyły swoje predyktory.

Może zaskoczy Cię informacja, że wiele zagadnień potrzebnych do zrozumienia zwycięskiego algorytmu zostało już przedstawionych w tym rozdziale lub wcześniej. Omawiamy tu jeden z predyktorów (opracowany przez zespół BellKor) i pokróćce wyjaśniamy, jak został on połączony z innymi w celu uzyskania rekomendacji.

Zespół BellKor opracował predyktor podstawowy. W tym celu zamodelował występujące w zbiorze danych błędy systematyczne dotyczące filmów i użytkowników. Uwzględnił też aspekt czasu. Pomogło to algorytmowi wykrywać zmiany w ogólnej popularności filmu (na przykład jej wzrost po ponownym wydaniu filmu), a także skłonność użytkowników do zmian typowych zachowań (na przykład zwiększenie lub spadek krytyczności). Zespół zastosował też opisany wcześniej rozkład SVD. Dzięki temu udało się zamodelować globalne wzorce, takie jak opisane wcześniej, natomiast

<sup>12</sup> Netflix, <http://www.netflixprize.com/leaderboard> (2009).

<sup>13</sup> Yehuda Koren, *The BellKor Solution to the Netflix Grand Prize*, sierpień 2009, <http://mng.bz/TwOD>.

<sup>14</sup> Robert M. Bell, Yehuda Koren i Chris Volinsky, *The BellKor 2008 Solution to the Netflix Prize*, <http://mng.bz/mW25>.

<sup>15</sup> Martin Piotte i Martin Chabbert, *The Pragmatic Theory Solution to the Netflix Grand Prize*, sierpień 2009, <http://mng.bz/1et7>.

tego rodzaju podejście często nie wykrywa lokalnych wzorców w danych. Aby temu zapobiec, zespół dodał metodę uwzględniania „sąsiadów”, podobną do filtrowania kolaboratywnego według produktów lub użytkowników. To pomogło wykryć lokalne wzorce widoczne wśród użytkowników i produktów oraz w sąsiednich danych. Ostatecznie był to tylko jeden z 454 potencjalnych predyktorów uwzględnianych przy ustalaniu zbioru technik, których wyniki były łączone za pomocą drzew GDBT (ang. *gradient-boosted decision trees*)<sup>16,17</sup>.

Choć zwycięski algorytm uzyskał imponujące wyniki, Netflix nie zaimplementował tego rozwiązania. Zamiast niego zdecydował się na wprowadzenie prostszego, choć mniej precyzyjnego algorytmu (dającego poprawę tylko o 8,43% w porównaniu z poprawą 10,06%, jaką zapewniało zwycięskie rozwiązanie). To ten mniej precyzyjny algorytm wygrał nagrodę Progress Prize i został zaimplementowany jeszcze przed zakończeniem konkursu. Firma Netflix uzasadniła to tym, że wzrost skuteczności oferowany przez zwycięski algorytm nie uzasadniał dodatkowego wysiłku, jaki byłby niezbędny w celu zapewnienia skalowalności tego rozwiązania.

Twórcy inteligentnych algorytmów mogą nauczyć się stąd ważnej lekcji. Należy skoncentrować się nie tylko na skuteczności proponowanego rozwiązania, ale też na kosztach. Jest też inny powód, dla którego Netflix nie zaimplementował zwycięskiego algorytmu — do czasu zakończenia konkursu branża zaczęła się zmieniać. Rekomendacje były kluczowe dla Netfliksa w czasach wypożyczania filmów DVD, ponieważ umożliwiały firmie odpowiedni dobór obrazów dla ograniczonej liczby przesyłanych pocztą wypożyczanych płyt. Ponieważ teraz coraz większe znaczenie odgrywa strumieniowanie, rekomendacje stają się mniej istotne, ponieważ filmy są udostępniane niemal natychmiast. Użytkownik może obejrzeć dowolny film w dowolnym miejscu i dowolnym czasie. Z tego wynika druga ważna lekcja: szybko opracowany akceptowalny inteligentny algorytm często jest bardziej wartościowy niż niezwykle precyzyjne rozwiązanie zdubowane później.

### **3.7. Ocenianie systemu rekommendacji**

Komercyjne systemy rekommendacji działają w wymagających warunkach. Liczba użytkowników zwykle jest mierzona w milionach, a liczba produktów — w setkach tysięcy. Dodatkowym wymogiem jest prezentowanie rekommendacji w czasie rzeczywistym (zwykle czas reakcji powinien być krótszy niż sekunda) bez rezygnowania z ich jakości. Przeważnie efekt ten uzyskuje się za pomocą warstwy pamięci podręcznej. Przeprowadzanie rozkładu macierzy metodą SVD zwykle nie odbywa się w czasie rzeczywistym i zależy od wielu czynników, na przykład od ilości dostępnej pamięci i, co ważniejsze, implementacji metody SVD. Istotne może być też to, jak długo użytkownik jest gotów czekać na wynik.

---

<sup>16</sup> Jerome H. Friedman, *Stochastic Gradient Boosting*, 26 marca 1999, <https://statweb.stanford.edu/~jhff/ftp/stobst.pdf>.

<sup>17</sup> Jerry Ye, Jyh-Herng Chow i inni, *Stochastic Gradient Boosted Distributed Decision Trees*, „Proceedings of the 18th ACM CIKM” (listopad 2009), <http://mng.bz/WiMO>.

Wiesz już, że dzięki zbieraniu ocen od wszystkich użytkowników można z czasem zwiększyć precyzyję predykcji. Jednak w czasie rzeczywistym niezbędne jest prezentowanie dobrych rekomendacji nowym użytkownikom, dla których z definicji nie jest dostępna duża liczba ocen.

Innym rygorystycznym wymogiem dotyczącym wysokiej jakości systemów rekomendacji jest możliwość aktualizowania prognoz na podstawie napływających ocen. W dużych komercyjnych witrynach w ciągu kilku godzin użytkownicy mogą zakupić tysiące produktów i wystawić podobną liczbę ocen. W jeden dzień liczba zakupów i ocen może sięgać dziesiątek tysięcy lub być jeszcze wyższa. Możliwość aktualizowania systemu rekomendacji na podstawie takich dodatkowych informacji jest ważna, a proces ten musi odbywać się w trybie online bez przestojów.

Załóżmy, że piszesz system rekomendacji i jesteś zadowolony z jego szybkości i ilości danych, jakie potrafi obsłużyć. Czy takie rozwiązanie jest dobre? Szybki i skalowalny system rekomendacji nie jest przydatny, jeśli generuje rekomendacje niskiej jakości. Zastanówmy się więc nad oceną precyzji systemu rekomendacji. W literaturze przedmiotu znajdziesz dziesiątki miar ilościowych i zestaw metod jakościowych służących do oceny wyników systemów rekomendacji. Duża liczba miar i metod jest dowodem na to, jak trudno jest przeprowadzić sensowną, uczciwą i precyzyjną ocenę rekomendacji. Jeśli interesuje Cię ten temat, wiele informacji znajdziesz w artykule przeglądowym Herlockera, Konstana, Terveena i Riedla<sup>18</sup>.

Tu przedstawiamy miarę pierwiastka błędu średniokwadratowego (ang. *root-mean square error* — **RMSE**) dla ocen prognozowanych na podstawie zbioru danych MovieLens. RMSE to prosta, ale solidna technika oceny precyzji rekomendacji. Ta miara ma dwie główne cechy: zawsze rośnie (nie otrzymujesz bonusów za poprawne prognozy ocen), a dzięki podnoszeniu różnic do kwadratu duże rozbieżności ( $>1$ ) są jeszcze zwiększone. Nie ma przy tym znaczenia, czy prognozowana ocena jest wyższa, czy niższa od rzeczywistej.

Zdaniem niektórych badaczy miara RMSE jest zbyt uproszczona. Zastanów się nad dwoma sytuacjami. W pierwszej zarekomendowano użytkownikowi film, prognożąc, że ta osoba przypisze mu cztery gwiazdki. Jednak film zdecydowanie się nie spodobał (użytkownik przypisał mu dwie gwiazdki). W drugim scenariuszu zarekomendowano film, prognożując przyznanie trzech gwiazdek. Użytkownikowi film bardzo się spodobał (został oceniony na pięć gwiazdek). W obu sytuacjach wpływ rozbieżności na miarę RMSE jest taki sam, jednak niezadowolenie użytkownika będzie większe w pierwszym przypadku. Na listingu 3.14 znajdziesz kod obliczający miarę RMSE w rozwiązaniu wykorzystującym filtrowanie kolaboratywne według modelu. Wcześniej wspomnialiśmy, że dostępne są też inne miary. Zachęcamy do ich wypróbowania.

---

<sup>18</sup> Jonathan L. Herlocker i współpracownicy, *Evaluating Collaborative Filtering Recommender Systems*, „ACM Transactions on Information Systems” 22, nr 1 (styczeń 2004).

**Listing 3.14. Obliczanie miary RMSE dla systemu rekomendacji**

```
from recsys.evaluation.prediction import RMSE

err = RMSE()
for rating, item_id, user_id in data.get():
    try:
        prediction = svd.predict(item_id, user_id)
        err.add(rating, prediction)
    except KeyError, k:
        continue

print 'RMSE wynosi ' + str(err.compute())

err = RMSE()
print d
for rating, item_id, user_id in data.get():
    try:
        prediction = svd.predict(item_id, user_id)
        rmse.add(rating, pred_rating)
    except KeyError, k:
        continue

print 'RMSE wynosi ' + err.compute()
```

**3.8. Podsumowanie**

- Objasniliśmy tu zagadnienia odległości i podobieństwa między użytkownikami i między produktami. Pokazaliśmy, że nie istnieją uniwersalne rozwiązania — miary podobieństwa trzeba starannie dobierać. Wzory na podobieństwo muszą generować wyniki zgodne z kilkoma podstawowymi regułami. Oprócz tego możesz wybrać wzór, który daje wyniki najlepiej dostosowane do potrzeb.
- Istnieją dwie ogólne klasy technik generowania rekomendacji: filtrowanie kolaboratywne i podejście oparte na treści. Tu skoncentrowaliśmy się na pierwszej z tych klas i przedstawiliśmy trzy różne metody generowania rekomendacji: według użytkowników, według produktów i według modelu.
- Wygenerowaliśmy rekomendacje produktów i użytkowników. W obu przypadkach przyjrzaliśmy się rozkładowi ocen dla gatunków filmów. Gatunki nie są idealnym narzędziem do analizowania filtrowania kolaboratywnego na podstawie modelu, jednak pozwalają uchwycić pewne aspekty preferencji użytkowników i tematyki filmów.
- Na ostatnich stronach rozdziału przyjrzaliśmy się konkursowi Netflix Prize. Choć został on zakończony w 2009 roku, nadal jest niemal synonimem pojęcia „rekomendacje”. Przedstawiliśmy tu ogólny opis zwycięskiego rozwiązania, jednak zapoznanie się z wszystkimi zawiłościami tego algorytmu pozostawiamy Czytelnikom.



# *Klasyfikowanie — umieszczanie elementów tam, gdzie ich miejsce*

---

## **Zawartość rozdziału:**

- Techniki klasyfikowania
- Wykrywanie oszustw za pomocą regresji logistycznej
- Klasyfikowanie w bardzo dużych zbiorach danych

„A co to jest?” — to jedno z najczęściej zadawanych przez dzieci pytań. Popularność tego pytania wśród maluchów (których docieklewość jest równie wspaniała co uciążliwa) nie powinna zaskakiwać. Aby zrozumieć otaczający nas świat, porządkujemy spostrzeżenia w grupy i kategorie. W poprzednim rozdziale zaprezentowaliśmy algorytmy pomagające wykryć strukturę w danych. W tym rozdziale omawiamy algorytmy *klasyfikujące*, które pomagają przypisać każdy punkt danych do odpowiedniej kategorii lub *klasy* (stąd nazwa *klasyfikacja*). Proces klasyfikacji pozwala odpowiedzieć na pytanie dziecka za pomocą zdania w postaci: „To statek”, „To drzewo”, „To dom” itd.

Można uznać, że w ramach umysłowego przetwarzania najpierw określana jest struktura, a potem następuje klasyfikowanie. Gdy mówimy dziecku, że trzyma w ręku statek, pośrednio wiemy, że statek to środek transportu powiązany z wodą (a nie z powietrzem lub lądem) i unoszący się na niej. Ta utajona wiedza jest dostępna przed zaklasyfikowaniem przedmiotu jako statku i zawiera możliwy wybór. W podobny sposób

określanie klastrów danych lub przekształcanie punktów danych z pierwotnego zbioru może okazać się bardzo przydatnym krokiem, odkrywającym cenne informacje, które można wykorzystać do klasyfikowania.

W dalszych punktach prezentujemy liczne praktyczne przykłady, w których klasyfikacja jest istotna. Następnie przedstawiamy przegląd i taksonomię dostępnych klasyfikatorów. Oczywiście nie możemy opisać tu wszystkich klasyfikatorów, dlatego prezentowany przegląd pomoże Ci znaleźć odpowiednią literaturę. Poznasz też regresję logistyczną. Jest to technika należąca do ogólniejszej klasy algorytmów — *ogółmionych modeli liniowych*. Omawiamy jej zastosowanie do klasyfikowania oszustw i wykrywania nietypowych zachowań w internecie. Wspominamy też o innych dokonaniach z tej dziedziny (ten temat szczegółowo opisujemy w następnym rozdziale).

Jak stwierdzić, czy punkt danych przypisales do najbardziej odpowiedniej klasy? Jak określić, czy klasyfikator A jest lepszy od klasyfikatora B? Jeśli czytałeś kiedyś ulotki reklamujące narzędzia z obszaru analityki biznesowej, może natrafisz na stwierdzenia typu: „Skuteczność naszego klasyfikatora wynosi 75%”. Co z tego wynika? Czy taki klasyfikator jest przydatny? Odpowiedzi na te pytania znajdziesz w podrozdziale 4.4. W rozdziale omawiamy też klasyfikowanie dużych zbiorów punktów danych i przeprowadzanie wydajnej klasyfikacji na żywo.

Zacznijmy od omówienia zastosowań klasyfikacji i przedstawienia pojęć technicznych, które napotkasz w tekście. Do czego więc przydaje się klasyfikacja i jakie praktyczne problemy pozwala rozwiązywać?

## 4.1. Do czego potrzebna jest klasyfikacja?

Choć może nie zdajesz sobie z tego sprawy, codziennie stykasz się z klasyfikacją. W życiu codziennym dania w menu restauracji są poklasyfikowane według kategorii takich jak sałatki, przystawki, specjalności kuchni, makarony, owoce morza itd. W gazetach artykuły są poklasyfikowane według tematyki: polityka, sport, biznes, wiadomości ze świata, rozrywka itd. Tak więc klasyfikacje są powszechnie w naszym życiu.

Książkom w bibliotece przypisana jest *sygnatura* składająca się z dwóch numerów: *numeru KDD* (klasyfikacji dziesiętnej Deweya) i *numeru Cuttera*. Podstawowe kategorie w tym systemie to na przykład: dzieła treści ogólnej, religia, nauki przyrodnicze, matematyka itd. W amerykańskiej Bibliotece Kongresu obowiązuje niezależny system klasyfikacji, opracowany pod koniec XIX wieku i na początku XX wieku na potrzeby porządkowania dostępnych w niej zbiorów.

W XX wieku system Biblioteki Kongresu został przyjęty także w wielu innych bibliotekach — przede wszystkim w dużych bibliotekach uniwersyteckich w Stanach Zjednoczonych. Wspominamy tu o dwóch systemach klasyfikowania książek, ponieważ system Biblioteki Kongresu nie jest równie hierarchiczny jak system KDD, w którym hierarchiczne zależności między tematami są odzwierciedlane w numerach.

W medycynie liczne systemy klasyfikacji są stosowane do diagnozowania urazów i chorób. Na przykład radiolodzy i ortopedzi posługują się systemem Schatzkera do klasyfikowania złamań bliższej nasady kości piszczelowej (jest to skomplikowany uraz kolana). Istnieją też systemy klasyfikacji dotyczące uszkodzeń rdzenia kręgowego,

śpiączki, wstrząsu mózgu, urazowych uszkodzeń mózgu itd. Zapewne zetknęłeś się z określeniem *Homo sapiens*. *Homo* oznacza tu nasz rodzaj, a *sapiens* — gatunek. Tę klasyfikację można rozszerzyć o inne atrybuty: *rodzinę, rząd, gromadę, typ* itd.

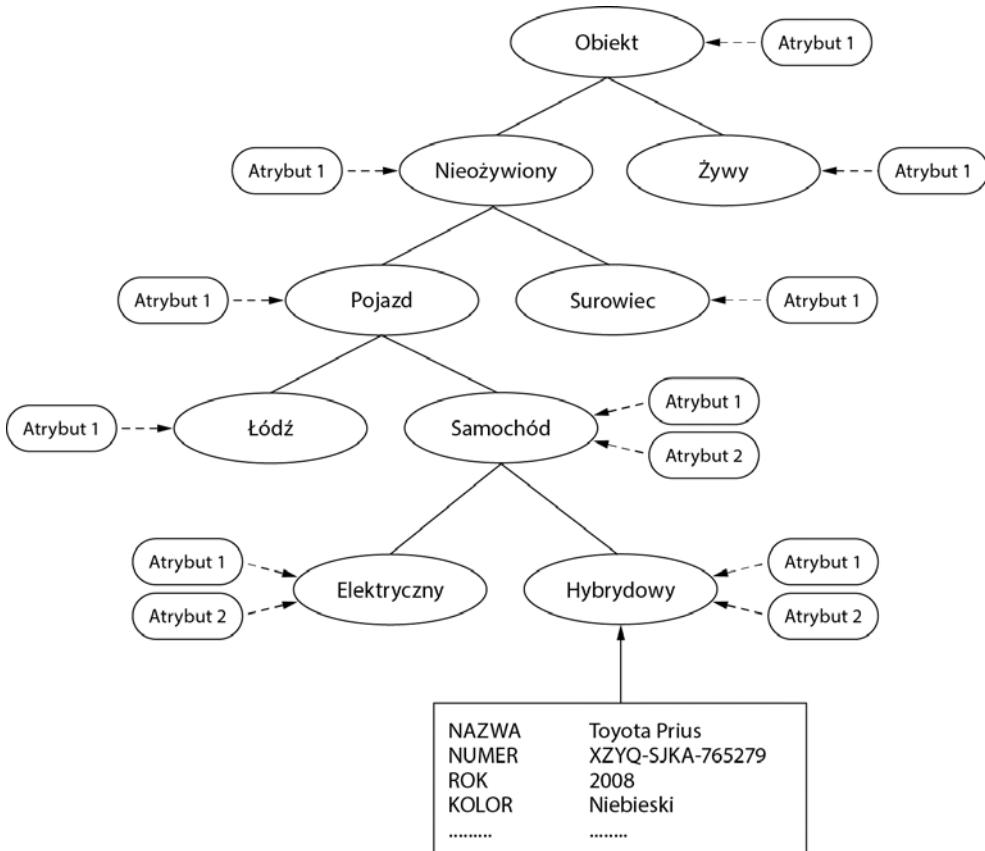
Ogólnie im więcej atrybutów uwzględniasz, tym precyzyjniejsza będzie klasyfikacja. „Duża” liczba atrybutów jest zwykle korzystna, przy czym trzeba uwzględnić pewne zastrzeżenia. Kłopotliwym symptomem związanym z dużą liczbą atrybutów jest opisane w rozdziale 2. *przekleństwo wymiarów*.

Może pamiętasz, że „przekleństwo wymiarów” polega na tym, że wraz ze wzrostem liczby atrybutów przestrzeń staje się coraz bardziej jednorodna. Oznacza to, że odległości między dwoma punktami będą podobne niezależnie od tego, które punkty wybierzesz i jakie miary zastosujesz do określania odległości. W takiej sytuacji łatwo jest wyznaczyć, która kategoria jest bliższa danemu punktowi danych, ponieważ niezależnie od miejsca zajmowanego w przestrzeni wszystko wydaje się oddalone w podobnym stopniu.

*Plaskie struktury odniesienia* zwykle nie są tak rozbudowane jak *hierarchiczne struktury odniesienia*, które z kolei są prostsze od struktur hierarchicznych oraz wzbogaconych semantycznie. Semantycznie wzbogacona struktura odniesienia związana jest z *ontologiami*, czyli taksonomiami z (przeważnie) ręcznie określonym znaczeniem, pozwalającymi przedstawiać wiedzę specyficzną dla dziedziny. Na potrzeby tej książki ontologia obejmuje trzy aspekty: *pojęcia, instancje i atrybuty*.

Na rysunku 4.1 zilustrowany jest mały wycinek bardzo prostej ogólnej ontologii służącej do klasyfikowania pojazdów. Pojęcia są reprezentowane za pomocą elips, instancje mają postać prostokątów, a atrybuty to prostokąty z zaokrąglonymi rogami. Zwróć uwagę na to, że atrybuty są dziedziczone. Jeśli atrybut 1 jest przypisany do korzenia drzewa pojęć, kaskadowo przechodzi do liści tego drzewa. Dlatego wartość atrybutu 1 można przypisać do instancji reprezentujących i lódź, i samochód. Jednak tylko instancje reprezentujące samochód przyjmują wartość atrybutu 2. Atrybutem 1 może być tu *Nazwa*, która z przyczyn praktycznych zawsze jest przydatna, natomiast atrybutem 2 może być *Liczba kół*. Atrybuty są definiowane na poziomie pojęć, jednak tylko instancje przyjmują konkretne, unikatowe wartości, ponieważ tylko instancje reprezentują rzeczywiste obiekty.

Możesz przyjąć, że pojęcia to odpowiedniki klas z języków obiektowych, instancje to odpowiedniki konkretnych obiektów takich klas, a atrybuty to zmienne w tych obiektach. Kod bazowy, w którym klasy są pogrupowane według funkcji lub komponentów w pakiety, w którym używane jest dziedziczenie do abstrakcyjnego ujęcia wspólnych struktur i operacji i w którym odpowiednio stosowana jest hermetyzacja, jest wyższej jakości niż kod nieposiadający tych cech. W programowaniu obiektowym zdefiniowanie klasy oznacza zdefiniowanie typu danych zmiennych, jednak bez określania ich wartości (chyba że tworzona jest stała). To samo dotyczy omawianych tu zagadnień. Pojęcia dziedziczą atrybuty po pojęciach z wyższych poziomów drzewa, a instancja powstaje po przypisaniu wartości do atrybutów danego pojęcia. Jest to dobra robocza definicja, która sprawdza się w 80 – 90% sytuacji. Jeśli będziesz miał wątpliwości, możesz wrócić do opisanej tu analogii, aby lepiej zrozumieć używaną strukturę.



Rysunek 4.1. Podstawowe elementy struktury odniesienia (bardzo prostej ontologii). Elipsy reprezentują pojęcia, a zaokrąglone prostokąty oznaczają atrybuty. Prostokąty przedstawiają instancje. Pojęcia dziedziczą atrybuty z wyższych poziomów

Oczywiście można przedstawić inne systemy klasyfikacji. Występują one wszędzie. Ważne jest, że klasyfikowanie danych jest związane z ich porządkowaniem. Systemy klasyfikacji ułatwiają komunikację, ponieważ zmniejszają liczbę błędów wynikających z wieloznaczności. Pomagają też porządkować myśli i planować działania. Struktura odniesienia, używana do porządkowania danych, może być bardzo prosta (na przykład zbiór etykietek), jak i zaawansowana (*ontologia semantyczna*). Słyszaleś kiedyś o *sieci semantycznej* (ang. *Semantic Web*)? Jej istotą są liczne technologie i formalne specyfikacje związane z tworzeniem, stosowaniem i utrzymywaniem ontologii semantycznych. Ontologie są też przydatne w architekturach opartych na modelu<sup>1</sup>, które powstają w ramach prowadzonej przez organizację OMG (ang. *Object Management Group*, <http://www.omg.org>) inicjatywy z obszaru projektowania oprogramowania.

<sup>1</sup> D. Gasevic, D. Djuric i V. Devedzic, *Model Driven Architecture and Ontology Development* (Springer, 2006).

Pomyśl o bazie danych, z której korzystasz. Możliwe, że zarządzasz sklepem internetowym, intranetowym systemem zarządzania dokumentami, internetowym serwisem typu mashup lub inną aplikacją sieciową. Gdy zastanowisz się nad danymi i sposobami ich uporządkowania, zdasz sobie sprawę z wartości systemu klasyfikacji w danej aplikacji. Od podrozdziału 4.3 przedstawiamy mechanizmy klasyfikacji z aplikacji sieciowych, aby pokazać, jak używać algorytmów klasyfikacji i jakie problemy mogą przy tym wystąpić. Najpierw jednak zapoznaj się z przeglądem klasyfikatorów. Jeśli jednak chcesz szybko zabrać się do pracy, możesz pominąć następny podrozdział.

## 4.2. Przegląd klasyfikatorów

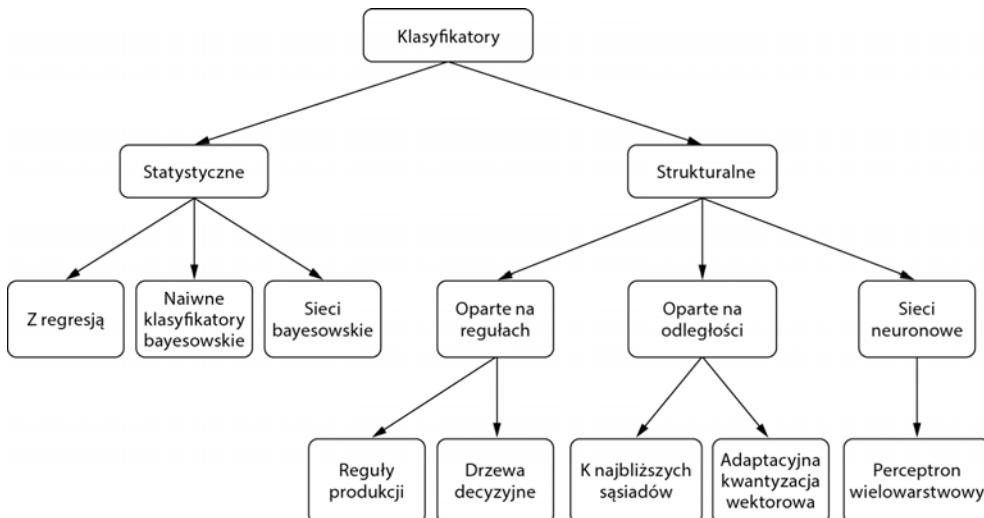
Jednym ze sposobów postrzegania zbioru wszystkich systemów klasyfikacji jest uwzględnienie używanej struktury odniesienia. Na ogólnym poziomie wszystkie systemy klasyfikacji można podzielić na dwie ogólne kategorie: *binarne* i *wieloklasowe*. Binarne systemy klasyfikacji, jak wskazuje na to nazwa, udzielają odpowiedzi typu tak/nie na pytanie: „Czy ten punkt danych należy do klasy X?”. Medyczne systemy diagnostyczne mogą informować, czy pacjent jest chory na raka. System klasyfikacji imigrantów może określać, czy dana osoba jest terrorystą. Natomiast wieloklasowe systemy klasyfikacji przypisują punkt danych do konkretnej klasy (jednej z wielu); mogą na przykład przypisywać wiadomość w serwisie informacyjnym do kategorii.

Wieloklasowe systemy klasyfikacji można dodatkowo podzielić na podstawie dwóch kryteriów: tego, czy podział na klasy jest ciągły, czy nieciągły, oraz tego, czy struktura klas jest płaska (jest na przykład listą nazw), czy hierarchiczna. Systemy KDD i ICD-10 to przykłady systemów klasyfikacji o skończonej liczbie nieciągłych klas. Wynikiem klasyfikacji może być też ciągła zmienność. Jest tak na przykład wtedy, gdy klasyfikacja służy do generowania *prognoz*. Jeśli jako dane wejściowe podasz ceny akcji w poniedziałek, wtorek, środę i czwartek, a chcesz ustalić cenę akcji w piątek, problem możesz przedstawić jako klasyfikację wieloklasową z klasami nieciągłymi lub ciągłymi. Wersja z klasami nieciągłymi może przewidywać, czy cena akcji w piątek wzrośnie, nie zmieni się lub spadnie. Wersja z klasami ciągłymi może prognozować konkretną cenę akcji.

Podział systemów klasyfikacji na podstawie używanych technik nie jest tak jednoznaczny ani powszechnie akceptowany. Można jednak przyjąć, że w branży stosunkowo często uwzględnia się dwie ogólne kategorie. Pierwsza obejmuje *algorytmy statystyczne*, a druga — *algorytmy strukturalne*. Ten podział przedstawiamy na rysunku 4.2.

Są trzy odmiany algorytmów statystycznych. Algorytmy wykorzystujące regresję wyjątkowo dobrze nadają się do prognozowania wartości zmiennych ciągłych. Te algorytmy są oparte na założeniu, że wystarczy dopasować dane do konkretnego modelu. Często ten model jest funkcją liniową badanej zmiennej, choć nie musi tak być (przekonasz się o tym w podrozdziale 4.3). Innego rodzaju statystyczne algorytmy klasyfikacji wykorzystują twierdzenie Bayesa. Stosunkowo skuteczna nowa technika statystyczna łączy twierdzenie Bayesa z probabilistycznymi strukturami sieciowymi, które opisują zależności między różnymi atrybutami z danego problemu z dziedziny klasyfikacji.

Są trzy główne typy algorytmów strukturalnych: *oparte na regułach* (obejmują one reguły jeśli-to i drzewa decyzyjne), *oparte na odległości* (zwykle dzieli się je na algorytmy



Rysunek 4.2. Przegląd algorytmów klasyfikacji na podstawie ich projektu

funkcyjne i z najbliższymi sąsiadami) i *sieci neuronowe*. Sieci neuronowe stanowią odrębną kategorię, choć warto wspomnieć, że wykryte i dokładnie przebadane zostały analogie między niektórymi sieciami neuronowymi a zaawansowanymi algorytmami statystycznymi (na przykład procesami gaussowskim). W następnych punktach dokładniej omawiamy strukturalne i statystyczne algorytmy klasyfikacji.

#### 4.2.1. Strukturalne algorytmy klasyfikacji

Na rysunku 4.2 widać, że gałąź algorytmów strukturalnych opartych na regułach obejmuje algorytmy z *regułami produkcji* (klauzulami jeśli-to) i algorytmy z drzewami decyzyjnymi. Reguły produkcji mogą być określone ręcznie przez ekspertów lub wynioskowane z drzew decyzyjnych. Algorytmy oparte na regułach są zwykle implementowane w postaci systemów produkcyjnych z *wnioskowaniem w przód*. Najlepszy algorytm z tej kategorii to *Rete*<sup>2</sup> (łacińskie słowo *rete* oznacza sieć). Stanowi on podstawę popularnych bibliotek, takich jak CLIPS, Jess i Soar.

W algorytmach opartych na drzewach decyzyjnych wykorzystywany jest prosty, ale wartościowy pomysł. Czytałeś kiedyś powieść *Opowieść wigilijna* Charlesa Dickensa? Dickens opisuje w niej grę „tak lub nie”. Polega ona na tym, że bratanek Scrooge'a myśli o czymś, a pozostałe osoby muszą ustalić, co to jest, zadając pytania, na które można odpowiadać tylko „tak” lub „nie”. Wersje tej gry występują w wielu kulturach. Jest ona popularna wśród dzieci w krajach hiszpańskojęzycznych, gdzie nosi nazwę *veo, veo*. Działanie większości algorytmów z drzewami decyzyjnymi jest podobne do przebiegu tych gier — odpowiedzi na pytania pozwalają wyeliminować możliwie dużą liczbę kandydatów na podstawie udzielonych informacji. Takie algorytmy mają szereg zalet. Między innymi są łatwe w użyciu i wydajne obliczeniowo. Wady tych algorytmów

<sup>2</sup> Charles Forgy, *Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem*, „Artificial Intelligence” 19 (1982): 17 – 37.

stają się widoczne zwłaszcza wtedy, gdy przetwarzane są zmienne ciągłe, ponieważ trzeba wtedy dokonać podziału. Kontynuum wartości trzeba podzielić na skończoną liczbę grup, aby utworzyć drzewo. Algorytmy z drzewami decyzyjnymi zwykle nie radzą sobie dobrze z generalizowaniem danych, dlatego nie działają dobrze dla niespotykanych wcześniej wartości. Często używane algorytmy z tej kategorii to C5.0 (w systemie Unix) i See5 (w systemie Microsoft Windows). Stosuje się je w wielu produktach komercyjnych, takich jak IBM SPSS Modeler (dawna nazwa to Clementine)<sup>3</sup> i RuleQuest<sup>4</sup>.

Druga gałąź algorytmów strukturalnych obejmuje algorytmy oparte na odległości. We wcześniejszych rozdziałach przedstawiliśmy i stosowaliśmy miarę podobieństwa i uogólnioną odległość. Algorytmy z tej grupy są stosunkowo intuicyjne, jednak łatwo o błędne ich zastosowanie, co prowadzi do błędnych wyników klasyfikacji z powodu braku bezpośrednich powiązań między wieloma atrybutami punktów danych. Jedna miara podobieństwa nie pozwala w pełni ująć różnic między atrybutami. Aby algorytm oparty na odległości był skuteczny, niezbędna jest staranna normalizacja i analiza przestrzeni atrybutów. Jednak gdy liczba wymiarów jest niska, w wielu sytuacjach algorytmy z tej kategorii działają poprawnie i są łatwe do zaimplementowania. Algorytmy wykorzystujące odległości można dodatkowo podzielić na funkcyjne i oparte na najbliższym sąsiedzie.

Klasyfikatory funkcyjne wyznaczają przybliżenie danych za pomocą funkcji (jak wskazuje na to ich nazwa). Działają one podobnie jak algorytmy regresji. Różnica związana jest z celem stosowania funkcji. W regresji funkcja jest używana jako model rozkładu prawdopodobieństwa. W klasyfikatorach funkcyjnych ważne jest tylko liczbowe przybliżenie danych. W praktyce trudne (i zwykle bezcelowe) jest odróżnianie regresji liniowej od liniowego przybliżenia za pomocą minimalizacji błędu kwadratowego.

Algorytmy wyznaczania najbliższego sąsiada próbują znaleźć dla każdego punktu danych najbliższą klasę. Dzięki zastosowaniu wzorów, które pokazaliśmy wcześniej w omówieniu generalizowania odległości, można obliczyć odległość każdego punktu danych od każdej uwzględnianej klasy. Obiekt jest przypisywany do najbliższej mu klasy. Prawdopodobnie najczęściej używanym algorytmem klasyfikacji z tej grupy jest algorytm k najbliższych sąsiadów (ang. *k-nearest neighbors* — kNN). Dobrze przebadany i często stosowany jest też inny algorytm — adaptacyjnej kwantyzacji wektorowej (ang. *learning vector quantization* — LVQ).

Algorytmy z obszaru sieci neuronowych to odrębna podkategoria algorytmów strukturalnych. Ich zrozumienie wymaga solidnej wiedzy matematycznej. W rozdziale 6. staramy się zaprezentować je z perspektywy obliczeniowej, unikając zbytniego wnikania w szczegóły matematyczne. Podstawowy pomysł, na którym oparta jest ta rodzina algorytmów klasyfikacji, to tworzenie sztucznej sieci węzłów obliczeniowych, analogicznej do biologicznej struktury ludzkiego mózgu, składającej się z *neuronów* i łączących je *synaps*.

Sieci neuronowe dobrze sprawdzają się w różnych zadaniach. Cechują się jednak dwoma podstawowymi wadami: nie istnieje metodologia projektowa dostosowana do

<sup>3</sup> Tom Khabaza, *The Story of Clementine* (Internal Report, Integral Solutions Limited, 1999).

<sup>4</sup> Ross Quinlan, *RuleQuest Research: Data Mining Tools*, <http://www.rulequest.com>.

dużej liczby problemów, a ponadto trudno jest interpretować wyniki klasifikacji z użyciem sieci neuronowych. Klasyfikator może mylić się rzadko, jednak nie rozumiemy, dlaczego tak się dzieje. To dlatego sieci neuronowe są uważane za technikę „czarnej skrzynki” — w odróżnieniu od drzew decyzyjnych i algorytmów opartych na regułach, gdzie zaklasyfikowanie określonego punktu danych łatwo jest zinterpretować.

#### **4.2.2. Statystyczne algorytmy klasifikacji**

Algorytmy regresji wyznaczają wzór najlepiej dopasowany do danych. Najczęściej używany wzór to funkcja liniowa dla wartości wejściowych<sup>5</sup>. Algorytmy regresji są stosowane zwykle wtedy, gdy punkty danych są z natury zmiennymi liczbowymi (na przykład wymiarami obiektu, wagą człowieka lub temperaturą powietrza). W odróżnieniu od algorytmów bayesowskich gorzej sprawdzają się dla danych kategorialnych (na przykład statusu pracownika lub opisu oceny kredytowej). Ponadto jeśli model danych jest liniowy, trudno jest uzasadnić przymiotnik *statystyczne*. Regresja liniowa nie różni się od zadania z poziomu liceum polegającego na dopasowaniu linii do zbioru punktów o określonych współrzędnych  $x$  i  $y$ .

Proces staje się ciekawszy i nabiera charakteru podejścia statystycznego, gdy używana jest *regresja logistyczna*. Wtedy model (funkcja logistyczna) generuje wartości z przedziału od 0 do 1, które można interpretować jako prawdopodobieństwo przynależności do danej klasy. Ta technika działa dobrze na potrzeby klasifikacji binarnej, gdy wyznaczony zostanie próg prawdopodobieństwa. Ten model jest często stosowany w przemyśle i omawiamy go w następnym podrozdziale. Pokażemy, jak zastosować go do wykrywania oszustw w internecie. Jest to wyjątkowo trudne zadanie, ponieważ oszustwa zachodzą bardzo rzadko w porównaniu ze standardowymi zdarzeniami.

W wielu technikach z kategorii algorytmów statystycznych używane jest twierdzenie z teorii prawdopodobieństwa — *twierdzenie Bayesa*<sup>6</sup>. Algorytmy z tej grupy cechują się tym, że atrybuty problemu są niezależne od siebie i mają jawnie ilościowy charakter. Fascynującym aspektem algorytmów bayesowskich jest to, że działają dobrze nawet wtedy, gdy założenie dotyczące niezależności atrybutów nie jest spełnione.

Sieci bayesowskie to stosunkowo nowe podejście do uczenia maszynowego, łączące twierdzenia Bayesa z zaletami podejść strukturalnych (takich jak drzewa decyzyjne). Naiwne klasyfikatory bayesowskie potrafią reprezentować proste rozkłady prawdopodobieństwa, ale nie radzą sobie z ujmowaniem probabilistycznej struktury danych, jeśli taka występuje. Dzięki zastosowaniu dającej duże możliwości reprezentacji w postaci *skierowanych grafów acyklicznych* relacje probabilistyczne między atrybutami można przedstawić w formie graficznej.

W tej książce nie omawiamy sieci bayesowskich. Jeśli chcesz dowiedzieć się więcej na ten temat, zapoznaj się z książką *Learning Bayesian Networks*<sup>7</sup>.

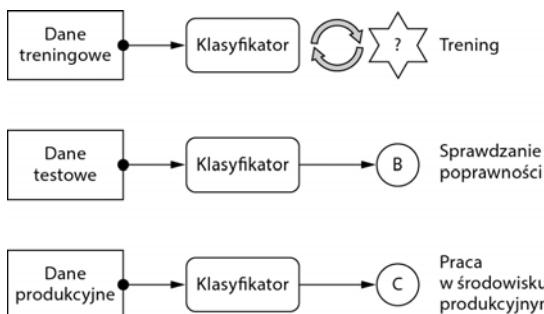
<sup>5</sup> Trevor Hastie, Robert Tibshirani i Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, wydanie drugie (Springer-Verlag, 2009).

<sup>6</sup> Kevin Murphy, *Machine Learning: A Probabilistic Perspective* (MIT Press, 2012).

<sup>7</sup> R.E. Neapolitan, *Learning Bayesian Networks* (Prentice-Hall, 2003).

### 4.2.3. Cykl życia klasyfikatora

Niezależnie od typu klasyfikatora, jaki wybierzesz na potrzeby aplikacji, jego cykl życia wpasowuje się w ogólny schemat przedstawiony na rysunku 4.3. Cykl życia klasyfikatora obejmuje trzy etapy: *trening*, *sprawdzanie poprawności (testowanie)* i *praca w środowisku produkcyjnym*.



Rysunek 4.3. Cykl życia klasyfikatora: trening, sprawdzanie poprawności (testowanie) i praca w środowisku produkcyjnym

Na etapie treningu klasyfikatorowi przekazywane są punkty danych przypisane już do odpowiedniej klasy. Każdy klasyfikator obejmuje szereg parametrów, które trzeba określić. Celem fazy treningowej jest ustalenie tych parametrów (graficznie przedstawiłyśmy to za pomocą znaku zapytania w gwiazdce). Na etapie sprawdzania poprawności należy zbadać klasyfikator, aby przed zastosowaniem go w środowisku produkcyjnym upewnić się, że generuje wystarczająco wiarygodne wyniki. Użyliśmy liter B w kółku, aby podkreślić, że głównym celem jest ustalenie poziomu błędu. Jednak jakość klasyfikatora można i należy mierzyć za pomocą różnych miar (w podrozdziale 4.4 omawiamy wiarygodność i koszty klasyfikacji). Dane używane na etapie sprawdzania poprawności (dane testowe) muszą być różne od danych z etapu treningu (od danych treningowych).

Przed przeniesieniem klasyfikatora do środowiska produkcyjnego etapy treningu i sprawdzania poprawności czasem powtarzane są wielokrotnie, ponieważ mogą wystąpić parametry konfiguracyjne, które nie zostały zidentyfikowane w procesie treningu, ale są podawane jako dane wejściowe projektu klasyfikatora. Oznacza to, że można napisać oprogramowanie, które obejmuje klasyfikator i parametry konfiguracyjne oraz przeprowadza automatyczne testy i sprawdzanie poprawności dużej liczby projektów. W testach można sprawdzać nawet klasyfikatory o różnym charakterze — na przykład naiwne klasyfikatory bayesowskie, sieci neuronowe i drzewa decyzyjne. Możesz albo wybrać najlepszy klasyfikator zgodnie z pomiarami jakości z etapów sprawdzania poprawności, albo połączyć wszystkie klasyfikatory w *metaklaszefikator*. To podejście zyskuje popularność w branży i konsekwentnie daje lepsze wyniki w wielu różnych zastosowaniach. Warto przypomnieć, że łączenie inteligentnych algorytmów omówiliśmy w kontekście systemów rekomendacji w rozdziale 3.

Na etapie pracy w środowisku produkcyjnym klasyfikator jest używany w aktywnym systemie do klasyfikowania na bieżąco danych. W tej fazie parametry zwykle się nie zmieniają. Można jednak wzbogacić wyniki klasyfikatora, rozbudowując go (już na etapie pracy w środowisku produkcyjnym) o krótki etap treningowy z informacjami

zwrotnymi od człowieka. Te trzy etapy są powtarzane wraz z napływaniem nowych danych z systemu produkcyjnego i generowaniem lepszych pomysłów na działanie klasyfikatora. W literaturze znajdziesz wiele różnych przeglądów algorytmów klasyfikacji<sup>8</sup>.

Aby zapewnić wystarczająco szczegółowy i praktyczny przykład, w pozostałej części rozdziału koncentrujemy się na algorytmach statystycznych, a zwłaszcza na klasyfikowaniu z użyciem regresji logistycznej. Wybraliśmy to podejście, ponieważ jest ono często używane w aplikacjach sieciowych w przemyśle. Omawiany przykład dotyczy klasyfikowania transakcji w internecie jako prawidłowe i oszustwa.

### **4.3. Wykrywanie oszustw za pomocą regresji logistycznej**

W tym podrozdziale koncentrujemy się na stosowaniu *regresji logistycznej*. Jest to algorytm klasyfikacji statystycznej, który jako dane wejściowe przyjmuje zestaw cech i docelową zmienną, które odwzorowuje na poziom prawdopodobieństwa z przedziału od 0,0 do 1,0. W kontekście wykrywania oszustw tymi cechami mogą być aspekty transakcji: kwota, godzina, dzień tygodnia itd. Docelowa zmienna może określać rzeczywiste dane — to, czy transakcja została ostatecznie uznana za oszustwo. Może zauważłeś, że algorytm zwraca ciągle dane wyjściowe, natomiast celem jest uzyskanie binarnych danych wyjściowych (na przykład 1 oznaczające oszustwo i 0 oznaczające poprawną transakcję). To docelowe odwzorowanie uzyskujemy, ustawiając wartość 1, jeśli prawdopodobieństwo oszustwa jest większe niż 0,5. W przeciwnym razie ustawiana jest wartość 0.

Zanim przejdziemy do szczegółów regresji logistycznej, warto przedstawić krótkie wprowadzenie do regresji liniowej (regresję logistyczną można traktować jak jej proste rozwinięcie). Większość zagadnień występujących w obu podejściach jest taka sama, dlatego będzie to dobra „rozgrzewka” przed zetknięciem się z bardziej zaawansowaną matematyką.

#### **4.3.1. Wprowadzenie do regresji liniowej**

Zanim zrozumiesz regresję logistyczną, powinieneś opanować regresję liniową. Jeśli cofniesz się myślą do początków nauki matematyki, może przypomnisz sobie równanie na prostą w przestrzeni euklidesowej:

$$y = mx + c$$

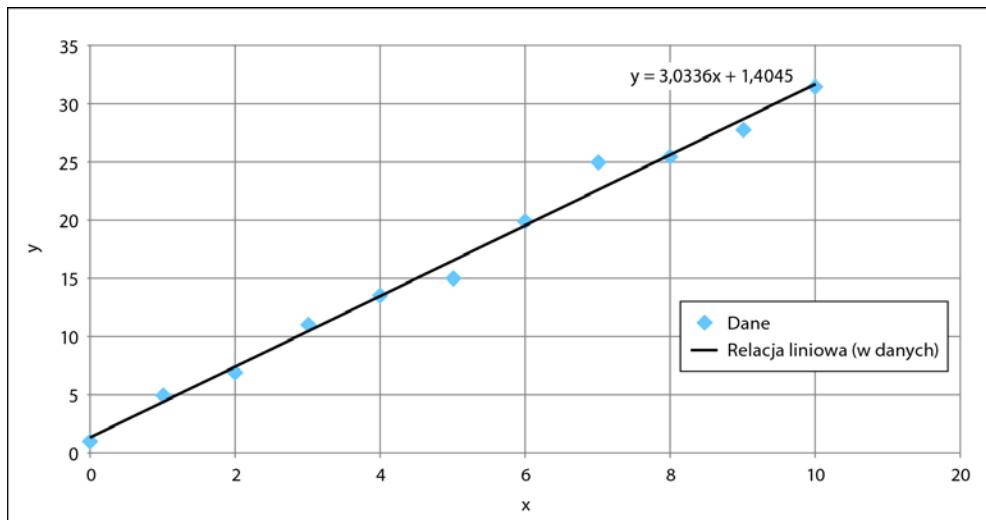
Wartości ( $x, y$ ) reprezentują tu współrzędne punktu danych w przestrzeni,  $m$  określa nachylenie względem osi, a  $c$  to przesunięcie początkowego punktu. W prosty sposób można to opisać tak: nachylenie ( $m$ ) określa, jak szybko zmienia się  $x$  względem zmiany  $y$ . Jeśli  $x$  zmieni się o  $\delta x$ ,  $y$  zmieni się o  $m \times \delta x$ . Przesunięcie (punkt początkowy) to wartość  $y$  dla  $x$  równego zero.

Ten wzór wyznacza *relację liniową*. Jeśli chcesz opracować model dla dwóch zmiennych za pomocą relacji liniowej, najpierw musisz się upewnić, że modelowaną

---

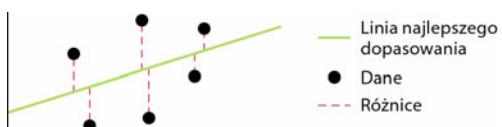
<sup>8</sup> L. Holmström, P. Koistinen, J. Laaksonen i E. Oja, *Neural and Statistical Classifiers—Taxonomy and Two Case Studies*, „IEEE Transactions on Neural Networks” 8, nr 1 (1997): 5 – 17.

zależność można wystarczająco dobrze przybliżyć w taki sposób. Jeśli tego nie zrobisz, możesz uzyskać model niskiej jakości, który nie stanowi uogólnienia analizowanego zjawiska. Na rysunku 4.4 pokazana jest graficzna reprezentacja modelu liniowego z przykładowymi danymi, w których można zauważać zależność liniową. Rozwiązaliśmy pokazany wcześniej wzór na relację liniową i narysowaliśmy linię prostą nałożoną na dane.



Rysunek 4.4. Graficzna ilustracja modelu liniowego. Punkty danych wskazują na wyraźną relację liniową między x a y. Pokazana jest tu linia najlepszego dopasowania zamodelowana za pomocą wzoru  $y = mx + c$

Zauważ, że dane nie pokrywają się idealnie z linią. W praktyce rejestrowane dane zwykle nie są w pełni zgodne z prognozowanym wzorcem, ale podlegają błędowi systematycznemu i szumowi, co opisaliśmy w rozdziale 2. Dlatego najlepsze, co możemy zrobić, to rozwiązać równanie i wyznaczyć linię najlepszego dopasowania. Taka linia jest wyznaczana w wyniku minimalizacji różnic (błędów) między modelem a samymi danymi. Ilustruje to rysunek 4.5.



Rysunek 4.5. Obliczanie różnic, czyli odległości między modelem a danymi. Na rysunku miarą błędu jest odległość w pionie na osi y, jednak dostępne są też inne metody

Teraz widać, że linia najlepszego dopasowania została uzyskana w wyniku modyfikowania parametrów modelu (tu są tylko dwa parametry: nachylenie,  $m$ , oraz przesunięcie,  $c$ ) do momentu zminimalizowania kwadratów różnic. Używane są kwadraty różnic, ponieważ nie ma wtedy znaczenia kierunek błędu, czyli to, czy punkt znajduje się pod, czy nad linią modelu. Ten etap to *trening modelu*. Zwykle model wykorzystujący linię prostą nie jest ograniczony do jednej zmiennej,  $x$ . Można go uogólnić na  $n+1$  wymiarów, posługując się następującym zapisem:

$$y = m_1x_1 + m_2x_2 + \dots + m_{n-1}x_{n-1} + m_nx_n + c$$

W tym podejściu modelowanie przebiega w bardzo podobny sposób. Zmieniają się tylko funkcja wyznaczająca różnice i szczegóły treningu, ponieważ trzeba uwzględnić większą liczbę wymiarów. Dlatego algorytm treningu musi aktualizować  $m_i$  dla wszystkich  $i = 1, \dots, n$ , a także  $c$ .

Opisaliśmy tu uniwersalny mechanizm uczenia maszynowego: trzeba przeprowadzić trening modelu, aby jak najlepiej uogólnić zebrane dane. To podejście jest takie samo dla modelów wszystkich typów (liniowych i innych), dlatego w praktyce często zetkniesz się z tym wzorcem.

Może zastanawiasz się, dlatego nie stosujemy regresji liniowej do wykrywania oszustw. Takie rozwiązanie jest możliwe, ale z wielu powodów nie należy go stosować. W następnym punkcie szczegółowo omawiamy regresję logistyczną i wyjaśniamy, dlaczego jest ona znacznie lepszym sposobem wykrywania oszustw.

#### **4.3.2. Od regresji liniowej do logistycznej**

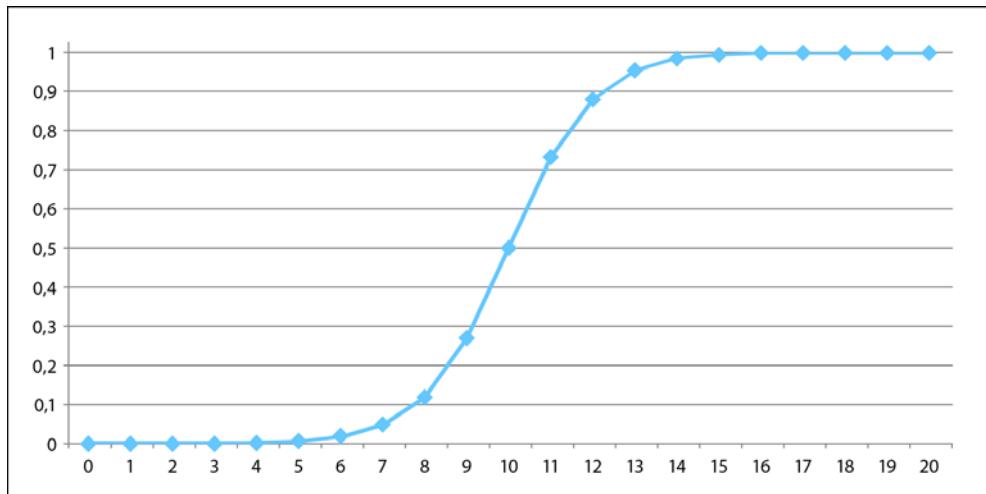
W poprzednim punkcie omówiliśmy podstawy regresji (przede wszystkim liniowej). Niestety, próba wykrywania oszustw za pomocą modelu liniowego skutkuje pewnymi problemami. Pierwszy z nich dotyczy danych wyjściowych modelu (*przeciwdziedziny*). W modelu liniowym  $y$  przyjmuje wartości od minus nieskończoności do plus nieskończoności. W algorytmie wykrywania oszustw nie jest to odpowiednie, ponieważ lepsze byłyby dane wyjściowe z określonego przedziału: od 0,0 dla bardzo niskiego prawdopodobieństwa oszustwa do 1,0 dla bardzo wysokiego prawdopodobieństwa. Można wtedy określić próg i klasyfikować wartości powyżej 0,5 jako wskazujące na oszustwo, a niższe jako oznaczające poprawną transakcję.

Drugi problem dotyczy liniowości modelu. Jeśli rozważysz prosty liniowy model wykrywania oszustw uwzględniający dwa wymiary (czyli jedną cechę;  $y$  określa wtedy prawdopodobieństwo oszustwa, a  $x$  to ciągła cecha opisująca transakcję), wtedy jednostkowa zmiana  $x$  skutkuje liniową zmianą  $y$ . Jednak nie zawsze jest to wskazane!

Przeprowadźmy krótki eksperyment myślowy, aby to zilustrować. Jeśli przyjąć, że ciągła zmienność  $x$  reprezentuje wartość transakcji, można uznać, że wysokie wartości wskazują na oszustwa. Pomyśl o zmianie  $x$  o 100 złotych. Jeśli model liniowy zwraca określone prawdopodobieństwo oszustwa dla kwoty 50 złotych, różnica między nim a prawdopodobieństwem oszustwa dla kwoty 150 złotych wynosi  $m \times 100$ . Jeśli przyrzysz się różnicę prawdopodobieństwa oszustwa dla kwot 250 000 i 250 100, będzie ona taka sama —  $m \times 100$ . W omawianej aplikacji nie jest to sensowne.

Idealne rozwiązanie powinno zwracać prawdopodobieństwo oszustwa rosnące szybko wraz ze wzrostem  $x$  w określonym zakresie. Dla bardzo wysokich wartości prawdopodobieństwo powinno rosnąć wolniej. Podobnie wraz ze spadkiem wartości  $x$  we wspomnianym zakresie ryzyko oszustwa powinno spadać szybko, a dla bardzo niskich wartości ten spadek powinien być niewielki. Wtedy różnice w danych wyjściowych klasyfikatora dla pary wartości 100 i 5000 złotych będą większe niż dla pary 5000 i 10 000 złotych. Intuicyjnie jest to uzasadnione, ponieważ przejście do poziomu tysięcy złotych ma większy wpływ na prawdopodobieństwo oszustwa niż dodanie kolejnych tysięcy do

już dużej kwoty. Może da się zastosować coś innego niż prostą linię, na przykład krzywą, która lepiej modeluje taką sytuację? Na szczęście jest to możliwe. Służy do tego krzywa logistyczna. Rysunek 4.6 ilustruje na takiej krzywej reakcje wartości  $y$  na zmiany wartości  $x$ .



**Rysunek 4.6.** Reakcje wartości  $y$  na zmiany wartości  $x$  na krzywej logistycznej. Ta konkretna krzywa ma środek w punkcie  $x = 10$  i została wyznaczona z użyciem 20 próbek równomiernie rozłożonych na osi  $x$ . Zwróć uwagę na większe różnice w wartościach  $y$  dla wartości  $x$  zbliżonych do 10 i mniejsze różnice wartości  $y$  w okolicach  $x = 0$  i  $x = 20$

Zauważ, że zmiany w wartości  $y$  są znacznie większe w pobliżu  $x = 10$  niż dla  $x = 0$  i  $x = 20$ . Dokładnie takie wyniki uznaliśmy za idealne dla tworzonego predyktora prawdopodobieństwa. Poniższe równanie przedstawia ogólną postać krzywej logistycznej.

$$y = \frac{L}{1 + e^{-k(x-x_0)}}$$

Zauważ, że na rysunku 4.6 wartości parametrów to:  $k = 1$ ,  $L=1$  i  $x_0 = 10$ .  $L$  określa tu wartość maksymalną krzywej, a  $x_0$  to przesunięcie krzywej na osi  $x$ . Zwykle (i dalej w tej książce) przyjmujemy, że  $L = 1$ , ponieważ wartość maksymalna krzywej powinna być równa 1.

Przedstawioną postać można łatwo rozwinać o obsługę wielu zmiennych. W tym celu należy podać dodatkowe zmienne w wykładniku  $e$ , co daje następującą postać:

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

Do tej pory wszystko przebiega sprawnie. Utworzyliśmy model o nieliniowej reakcji obsługujący wiele zmiennych wejściowych. Jakie jest jednak jego znaczenie? Przeprowadźmy proste operacje na równaniu, aby zobaczyć, do czego dojdziemy. Jeśli przyjmiemy, że  $t = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$ , oraz pomnożymy góre i dół równania przez  $e^t$ , uzyskamy następującą postać:

$$y = \frac{e^t}{1 + e^t}$$

Następny krok wymaga aktu wiary. Wiemy, że  $y$  określa prawdopodobieństwo wystąpienia zdarzenia dla cech podanych w wykładniku  $e$ . Dlatego przekształcamy równanie, aby wyrazić wartość  $y/(1-y)$ , czyli prawdopodobieństwo wystąpienia zdarzenia podzielone przez prawdopodobieństwo jego niewystąpienia. Daje to *szanse* (ang. *odds*) zdarzenia. Wyznaczmy najpierw  $1-y$ , a następnie podstawmy uzyskaną wartość w równaniu:

$$\begin{aligned}1 - y &= 1 - \frac{e^t}{1 + e^t} \\1 - y &= \frac{1 + e^t - e^t}{1 + e^t} \\1 - y &= \frac{1}{1 + e^t}\end{aligned}$$

Po otrzymaniu równania dla  $1-y$  (czyli modelu prawdopodobieństwa niewystąpienia zdarzenia) można przedstawić szanse za pomocą początkowej definicji  $y$ :

$$\begin{aligned}\frac{y}{1 - y} &= \frac{e^t}{1 + e^t} \quad \left/ \frac{1}{1 + e^t}\right. \\ \frac{y}{1 - y} &= e^t\end{aligned}$$

Po podstawieniu pod  $t$  pierwotnej wartości i wyciągnięciu po obu stronach logarytmu naturalnego uzyskamy ostateczną postać równania:

$$\ln\left(\frac{y}{1 - y}\right) = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

Porównaj ją z pierwotnym równaniem z punktu 4.3.1. Co zauważłeś? To prawda, oba modele są liniowe, jednak pierwotny model jako liniową kombinację zmiennych wejściowych wyznaczał prawdopodobieństwo, natomiast nowy model regresji logistycznej jako liniową kombinację zmiennych wejściowych określa *logarytm szans* (ang. *log odds*). Trening może przebiegać bardzo podobnie jak w modelu liniowym. Należy modyfikować wartości  $\beta$ , aby uzyskać najlepsze dopasowanie modelu do danych (zminimalizować różnice). W celu zdefiniowania *dopasowania* i zminimalizowania błędu można znaleźć taki zestaw parametrów, który zapewnia maksymalizację funkcji prawdopodobieństwa wystąpienia danych na podstawie modelu. W praktyce polega to na uruchomieniu algorytmu dla danych (raz lub wielokrotnie) i modyfikowaniu wartości  $\beta$ , aby wzmacniać znaczenie cech występujących w oszustwach i zmniejszać znaczenie cech obecnych w poprawnych transakcjach.

Ostatnia uwaga przed przejściem do kodu — końcowe równanie ujawnia ważny aspekt tego, jak należy interpretować parametry regresji logistycznej. Założymy, że

przeprowadziliśmy trening modelu i wszystkie cechy z wyjątkiem  $x_1$  są stałe. Dla każdej jednostki  $x_1$  logarytm szans w modelu rośnie o  $\beta_1$ . Podniesienie  $e$  do tej potęgi da wzrost szans dla tego samego wzrostu wartości cechy  $x_1$ . Dlatego jednostkowa zmiana w  $x_1$  zwiększa szanse o  $e^{\beta_1}$  (przy założeniu, że wszystkie pozostałe cechy są stałe).

#### 4.3.3. Implementowanie wykrywania oszustw

W poprzednich punktach przedstawiliśmy podstawy teorii związane z przydatną techniką regresji logistycznej. Jednak na razie wystarczy już teorię! Zapewne nie możesz się doczekać implementowania rozwiązania z użyciem tej techniki. W tym punkcie pokazujemy, jak to zrobić.

Zacznijmy od prostego zbioru danych<sup>9</sup> obejmującego szereg cech transakcji finansowej i określonych przez człowieka rzeczywistych wyników, wskazujących, czy transakcja została uznana za oszustwo, czy nie. Te dane są pokazane na rysunku 4.7.

IsFraud	Amount	Country	TimeOfTransaction	BusinessType	NumberOfTransactionsAtThisShop	DayOfWeek
0	10	DK	13	2	0	1
0	100	LT	18	1	3	4
0	49.99	AT	11	4	3	0
0	12	AUS	9	6	4	3
0	250	UK	12	1	5	6
0	149.99	UK	17	2	2	5
0	10	UK	16	8	1	4
0	49.99	DK	12	9	4	3
0	18	UK	14	2	3	6
0	27	DK	10	1	5	5
0	40	DK	11	1	6	2
0	2	UK	10	2	7	4
0	34.99	UK	9	4	8	3
0	2	UK	8	9	9	0
1	18000	LT	13	9	0	0
1	20000	LT	14	9	0	0
1	19000	LT	13	9	0	0
1	6000	LT	13	9	1	4
1	9000	LT	12	9	0	4
1	5000	LT	12	9	0	4
1	20000	LT	12	9	0	0
1	10000	LT	12	9	0	6
1	20000.01	UK	13	1	0	0
1	21000	LT	13	9	0	0
1	11000	LT	13	9	1	6
1	210000	UK	14	1	0	0
1	22000	UK	12	1	1	0
1	280000	LT	12	9	0	0
1	15000	LT	12	9	0	6

Rysunek 4.7. Dane używane do wykrywania oszustw

Na rysunku 4.7 kolumny określają, czy transakcja była oszustwem, kwotę transakcji, kraj jej przeprowadzenia, typ firmy, w jakiej wykonano transakcję, liczbę wcześniejszych

<sup>9</sup> Ten zbiór danych został opracowany ręcznie na potrzeby zademonstrowania regresji logistycznej. W dalszych rozdziałach stosujemy tę technikę do rzeczywistych danych.

transakcji przeprowadzonych w danym miejscu oraz dzień tygodnia. W tym punkcie celem jest zastosowanie implementacji regresji logistycznej z pakietu scikit-learn do skutecznego prognozowania i wskazywania oszustw.

Zanim przejdziemy dalej, zastanów się, czy coś w danych zwraca Twoją uwagę. Czy dostrzegasz jakieś zależności między cechami danych a wartością w kolumnie IsFraud? Wygląda na to, że transakcje o wysokiej wartości dokonywane po raz pierwszy w danym miejscu często są oszustwami. Ciekawie będzie zobaczyć, czy model uwzględnia tę informację. Wczytajmy teraz biblioteki i zainportujmy dane, co ilustruje listing 4.1.

#### Listing 4.1. Importowanie bibliotek i zbioru danych

```
import csv
import numpy as np
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn import linear_model, datasets, cross_validation
import matplotlib.pyplot as plt

dataset = []
f = open('./fraud_data_3.csv', 'rU')
try:
    reader = csv.reader(f, delimiter=',')
    next(reader, None)                                ← Pomijanie nagłówków z pierwszego wiersza
    for row in reader:
        dataset.append(row)
finally:
    f.close()
```

To było łatwe! Kod wczytał kilka standardowych bibliotek Pythona oraz kilka bibliotek z pakietu scikit-learn (omówimy je w dalszej części tego ćwiczenia). W procesie wczytywania danych pomijany jest wiersz pierwszy, zawierający nagłówki.

Następny krok związany jest z używaniem danych ze zbioru. Może pamiętasz, że są dwa rodzaje danych opisujących cechy: kategoryalne i ciągłe. Pierwszy z tych typów powoduje przypisanie wartości cechy do jednej z różnych kategorii. Tu BusinessType to dobry przykład cechy kategoryalnej. Choć typ jest tu zapisany za pomocą liczb całkowitych, kolejność wartości nie ma znaczenia; w ramach klasyfikacji ważna jest tylko przynależność do kategorii firmy. Dla cech ciągłych dane pochodzą z rozkładu ciągłego. Cecha może przyjmować dowolną poprawną wartość, a kolejność dwóch liczb jest istotna. Dobrym przykładem jest tu kolumna Amount, w której wartości można porównywać ze sobą. Wartość 50 jest większa niż 1. W kodzie tworzona jest maska definiująca, które wartości są kategoryalne, a które ciągłe. Ilustruje to listing 4.2, gdzie wykonywany jest też kolejny krok — pobieranie ciągłych i kategoryalnych cech, co pozwala traktować je w odmienny sposób.

#### Listing 4.2. Tworzenie maski dla cech kategoryalnych

```
target = np.array([x[0] for x in dataset])
data = np.array([x[1:] for x in dataset])
# Amount, Country, TimeOfTransaction, BusinessType,
# NumberOfTransactionsAtThisShop, DayOfWeek
```

```

categorical_mask = [False,True,True,True,False,True]
enc = LabelEncoder()

for i in range(0,data.shape[1]):
    if(categorical_mask[i]):
        label_encoder = enc.fit(data[:,i])
        print "Klasy kategorialne:", label_encoder.classes_
        integer_classes = label_encoder.transform(label_encoder.classes_)
        print "Klasy całkowitoliczbowe:", integer_classes
        t = label_encoder.transform(data[:, i])
        data[:, i] = t

```

Na listingu 4.2 dzieje się kilka ważnych rzeczy. Po pierwsze, tworzone są dwie nowe tablice: target i data. Pierwsza zawiera wynik klasyfikacji (czyli określa, czy transakcja była oszustwem), a druga obejmuje wszystkie cechy ze zbioru danych. Dalej tablica data jest modyfikowana, dzięki czemu wszystkie dane kategorialne otrzymują nowe etykiety. Jest to dobry zwyczaj, ponieważ pozwala zakodować dowolny typ danych za pomocą zbioru różnych liczb całkowitych. Na przykład gdyby typ firmy był zapisany za pomocą łańcuchów znaków, zostałby odwzorowany na zbiór liczb całkowitych zrozumiałych w trakcie treningu modelu.

W kolejnym kroku, pokazanym na listingu 4.3, dane kategorialne są oddzielane od ciągłych. Robimy to, ponieważ dane kategorialne zostaną zakodowane w specjalny sposób za pomocą *kodowania one hot*<sup>10</sup>.

#### Listing 4.3. Oddzielanie i kodowanie cech kategorialnych

```

mask = np.ones(data.shape, dtype=bool) ← Tworzenie obiektu maski
                                             pełnej wartości True
for i in range(0,data.shape[1]): ← Zapełnianie kolumn zerami
    if(categorical_mask[i]):           | dla danych kategorialnych
        mask[:,i]=False
data_non_categoricals = data[:, np.all(mask, axis=0)] ← Pobieranie danych ciągłych
data_categoricals = data[:,~np.all(mask, axis=0)] ← Pobieranie danych
                                                               kategorialnych
hotenc = OneHotEncoder()
hot_encoder = hotenc.fit(data_categoricals) ← Kodowanie wyłącznie danych kategorialnych
encoded_hot = hot_encoder.transform(data_categoricals)

```

Kod z listingu 4.3 rozbiija cechy na dwie tablice: data\_non\_categoricals, w której zapisywane są cechy ciągłe, i data\_categoricals, do której trafiają cechy kategorialne. Następnie cechy z drugiej tablicy są kodowane metodą one-hot, aby wartości przyjmowane przez każdą zmienną kategorialną były niezależne od siebie i aby kolejność wartości zakodowanych cech nie wpływała na końcowe wyniki.

Przyjrzyj się najpierw działaniu prostego modelu uwzględniającego same cechy ciągłe. Ilustruje to listing 4.4.

<sup>10</sup>Kevin Murphy, *Machine Learning: A Probabilistic Perspective* (MIT Press, 2012).

### Kodowanie one-hot

W tym przykładzie omawiamy zastosowanie kodowania one-hot do cech kategoryalnych. Czym jednak jest to kodowanie i dlaczego warto je stosować? Na potrzeby regresji logistycznej wszystkie cechy są ostatecznie kodowane za pomocą liczb. Dlatego jeśli nie potraktujesz zmiennych kategoryalnych w odpowiedni sposób, możesz natrafić na problemy.

Do zilustrowania tej techniki posłużymy się prostym przykładem. W zbiorze danych znajduje się pole określające kraj, w którym transakcja została przeprowadzona. Ten kraj trzeba zakodować za pomocą liczby. Używany jest do tego obiekt typu `LabelEncoder` z pakietu scikit-learn. Na potrzeby przykładu przyjmijmy, że UK = 1, DK = 2, a LT = 3.

Jeśli w regresji logistycznej przedstawisz te wartości za pomocą jednej zmiennej, ustalisz jeden współczynnik,  $\beta$ , określający, jak zmiana kraju wpływa na zmianę logarytmu szans oszustwa. Co jednak oznacza zmiana kraju UK (1) na DK (2) lub UK (1) na LT (3)? Co zróbić, jeśli dla państw LT i UK wyniki są takie same, więc warto połączyć je w jedną grupę?

Ponieważ dla takich cech kolejność nie jest istotna, nie ma sensu traktować ich w opisany sposób. Dlatego używane jest kodowanie one-hot. Zamiast tworzyć jedną cechę, używamy trzech, z których tylko jedna może być aktywna (gorąca; ang. *hot*). Na przykład UK można zakodować jako 100, LT jako 010, a DK jako 001. Wtedy uczenie dotyczy trzech parametrów, po jednym dla każdej cechy. Dzięki temu nie trzeba porządkować wartości, ponieważ trzy cechy są traktowane niezależnie, przy czym w danym momencie aktywna może być tylko jedna z nich.

#### Listing 4.4. Prosty model uwzględniający tylko cechy ciągłe

```
new_data=data_non_categoricals
new_data=new_data.astype(np.float)

X_train, X_test, y_train, y_test =
    cross_validation.train_test_split(new_data,
        target,
        test_size=0.4,
        random_state=0,dtype=float)                                     ← | Tworzenie zbioru treningowego
                                                               i testowego w wyniku podziału
                                                               danych na porcję (60% i 40%)

logreg = linear_model.LogisticRegression(tol=1e-10)           ← | Trening modelu opartego
logreg.fit(X_train,y_train)                                     ← | na regresji logistycznej
log_output = logreg.predict_log_proba(X_test)                  ← | Pobieranie danych wyjściowych
                                                               z modelu dla zbioru testowego

print("Szanse: " + str(np.exp(logreg.coef_)))
print("Punkt przecięcia osi dla szans: " + str(np.exp(logreg.intercept_)))
print("Punkt przecięcia osi dla prawdopodobieństwa: " +
      str(np.exp(logreg.intercept_)/(1+np.exp(logreg.intercept_)))

f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
plt.setp((ax1,ax2),xticks=[])

ax1.scatter(range(0,
    len(log_output[:,1]),1),
    log_output[:,1],
    s=100,
    label='Log. prawd.'.color='Blue',alpha=0.5)

ax1.scatter(range(0,len(y_test),1),
    y_test,
    label='Wyniki'.
```

```

s=250,
color='Green',alpha=0.5)

ax1.legend(bbox_to_anchor=(0., 1.02, 1., 0.102),
           ncol=2,
           loc=3,
           mode="expand",
           borderaxespad=0.)

ax1.set_xlabel('Przypadki testowe')
ax1.set_ylabel('Rzeczywiste wyniki / Log. prawd. wg modelu')

prob_output = [np.exp(x) for x in log_output[:,1]]
ax2.scatter(range(0,len(prob_output)),1,
            prob_output,
            s=100,
            label='Prawd.',
            color='Blue',
            alpha=0.5)

ax2.scatter(range(0,len(y_test)),1,
            y_test,
            label='Wyniki',
            s=250,
            color='Green',
            alpha=0.5)

ax2.legend(bbox_to_anchor=(0., 1.02, 1., 0.102),
           ncol=2,
           loc=3,
           mode="expand",
           borderaxespad=0.)

ax2.set_xlabel('Przypadki testowe')
ax2.set_ylabel('Rzeczywiste wyniki / Prawd. wg modelu')

plt.show()

```

Gdy uruchomisz ten listing, kod zwróci szereg informacji. Przede wszystkim poznasz wartości  $e^\beta$  dla wyuczonych parametrów, ponadto  $e^{\beta_0}$  i punkt przecięcia osi dla prawdopodobieństwa (wartość  $y$ , gdy wszystkie wartości  $x$  są równe 0):

```

Szanse: [[ 1.00137002  0.47029208]]
Punkt przecięcia osi dla szans: [ 0.50889666]
Punkt przecięcia osi dla prawdopodobieństwa: [ 0.33726409]

```

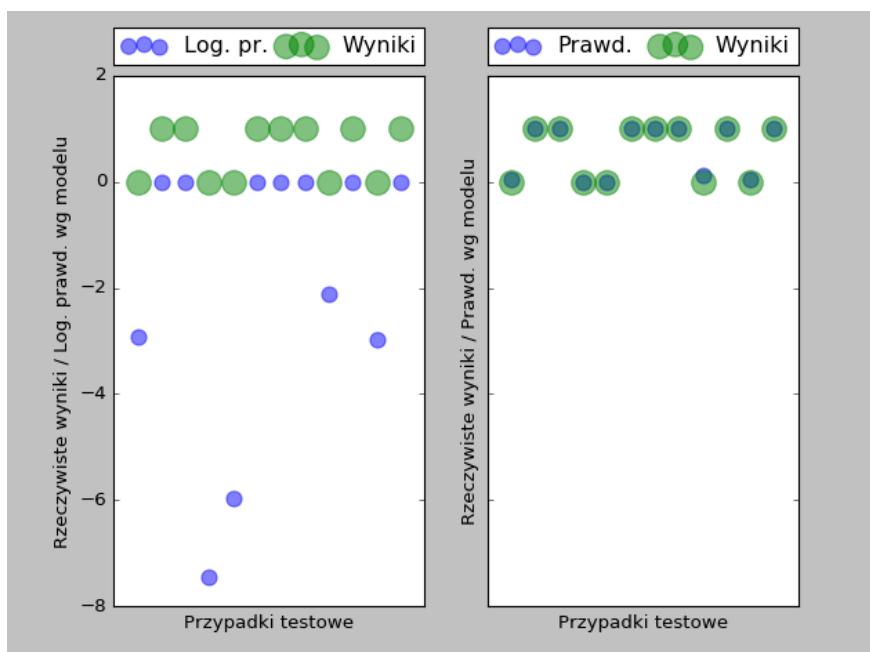
W ten sposób dowiadujemy się o kilku kwestiach. Pierwszy wiersz oznacza, że gdy wszystkie pozostałe cechy są stałe, jednostkowy wzrost kwoty transakcji zwiększa szanse wystąpienia oszustwa o 1,00137002. Gdy przyjrzesz się danym, stwierdzisz, że intuicyjnie ma to sens. Widać, że w badanym zbiorze danych wartość transakcji oznaczonych jako oszustwa jest wyższa.

Druga wartość to parametr  $\beta$  powiązany z drugą ciągłą zmienną, liczbą transakcji przeprowadzonych w danym miejscu. Widać tu, że jednostkowy wzrost wartości tej cechy zmniejsza ryzyko oszustwa o 0,47. Tak więc każda kolejna transakcja użytkownika

w danej lokalizacji zmniejsza prawdopodobieństwo oszustwa mniej więcej o połowę. Oszustwa są więc prawdopodobne w placówkach nowych dla użytkownika, a nie w regularnie przez niego odwiedzanych.

Punkt przecięcia osi dla szans to wartość  $e^\beta_0$ . Są to szanse w sytuacji, gdy wszystkie pozostałe wartości są równe 0. Na podstawie punktu przecięcia osi dla szans obliczane jest też prawdopodobieństwo oszustwa, gdy cena i liczba transakcji są równe zero. Zauważ, że ta liczba (oznaczająca około 50%) jest mniejsza niż prawdopodobieństwo oszustwa bez określania warunków dla zmiennych. Można to wyjaśnić tym, że oszustwa zwykle zdarzają się w przypadku transakcji o wysokiej wartości. Punkt przecięcia osi wyznacza prawdopodobieństwo, gdy wszystkie  $x_i$  są równe zero. Obejmuje to też czysto teoretyczne transakcje o wartości零. To zaniża prawdopodobieństwo, ponieważ nie wykryto żadnych oszustw z tak niską wartością transakcji.

Ostateczne dane wyjściowe z tego kodu to wykres z danymi wyjściowymi z modelu i rzeczywistymi wartościami (rysunek 4.8). Dane wyjściowe są tu zaprezentowane w dwóch postaciach. Po lewej stronie pokazane są rzeczywiste wyniki i logarytmy prawdopodobieństwa. Po prawej stronie przedstawione są rzeczywiste wyniki z nałożonym prawdopodobieństwem. W obu sytuacjach każda wartość na osi x jest powiązana z jednym przypadkiem testowym.



**Rysunek 4.8.** Dane wyjściowe z modelu nałożone na rzeczywiste wyniki z danych testowych. Rzeczywiste wyniki mogą przyjmować tylko wartości 1 (oszustwo) i 0 (poprawne transakcje). Logarytm prawdopodobieństwa może przyjmować wartości od 0,0 do  $-\infty$  (wykres po lewej). Po przeprowadzeniu potęgowania logarytmu uzyskujemy szacowane prawdopodobieństwo oszustwa z przedziału od 1,0 do 0,0. Zauważ, że większość przypadków została dobrze zaklasyfikowana. Określające prawdopodobieństwo oszustwa dane wyjściowe z modelu (prawy wykres) są bliskie rzeczywistym wynikom (wynik oznaczający oszustwo to 1)

Na rysunku 4.8 rzeczywiste wyniki przyjmują tylko wartości 1 i 0 (oznaczające oszustwo i poprawną transakcję). Na prawym wykresie dane wyjściowe modelu znajdują się w przedziale od 1,0 do 0,0 i reprezentują wyniki z modelu opartego na regresji logistycznej po treningu. Zwykle między danymi rzeczywistymi i wynikami modelu nie powinno być różnic. Danym niepowiązanym z oszustwami (wynik 0) model powinien przypisywać prawdopodobieństwo 0,0, natomiast danym powiązanym z oszustwem (wynik 1) — prawdopodobieństwo 1,0. Wykres pokazuje, że utworzony model sprawdza się całkiem dobrze. Jednak punktom danych 9 i 11 (patrząc od lewej) przypisane zostało prawdopodobieństwo oszustwa nieco wyższe niż zerowe.

Do tej pory uwzględnialiśmy tylko dwie z sześciu zmiennych dostępnych do modelowania transakcji. Dodajmy teraz cztery zmienne kategorialne z kodowaniem hot-one, aby zobaczyć, jak zmienia działanie modelu (zobacz listing 4.5).

#### **Listing 4.5. Używanie zmiennych kategorialnych i ciągłych w regresji logistycznej**

```
new_data = np.append(data_non_categoricals,encoded_hot.todense(),1)
new_data=new_data.astype(np.float)

X_train, X_test, y_train, y_test =
    cross_validation.train_test_split(new_data, target, test_size=0.4,
    random_state=0.dtype=float)

logreg = linear_model.LogisticRegression(tol=1e-10)
logreg.fit(X_train,y_train)
log_output = logreg.predict_log_proba(X_test)

print("Szanse: " + str(np.exp(logreg.coef_)))
print("Punkt przecięcia osi dla szans: " + str(np.exp(logreg.intercept_)))
print("Punkt przecięcia osi dla prawdopodobieństwa: " +
      str(np.exp(logreg.intercept_)/(1+np.exp(logreg.intercept_)))))

f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
plt.setp((ax1,ax2).xticks=[])

ax1.scatter(range(0,len(log_output[:,1]),1),
            log_output[:,1],
            s=100,
            label='Log. prawd.',color='Blue',alpha=0.5)

ax1.scatter(range(0,
                  len(y_test),1),
            y_test,
            label='Wyniki',
            s=250,
            color='Green',
            alpha=0.5)

ax1.legend(bbox_to_anchor=(0., 1.02, 1., 0.102),
           ncol=2,
           loc=3,
           mode="expand",
           borderaxespad=0.)
```

```

ax1.set_xlabel('Przypadki testowe')
ax1.set_ylabel('Rzeczywiste wyniki / Log. prawd. wg modelu')
prob_output = [np.exp(x) for x in log_output[:,1]]

ax2.scatter(range(0,len(prob_output)),1,
            prob_output,
            s=100,
            label='Prawd.',
            color='Blue',
            alpha=0.5)

ax2.scatter(range(0,len(y_test)),1,
            y_test,
            label='Wyniki',
            s=250,
            color='Green',
            alpha=0.5)

ax2.legend(bbox_to_anchor=(0., 1.02, 1., 0.102),
           ncol=2,
           loc=3,
           mode="expand", borderaxespad=0.)

ax2.set_xlabel('Przypadki testowe')
ax2.set_ylabel('Rzeczywiste wyniki / Prawd. wg modelu')
plt.show()

```

Kod na listingu 4.5 różni się od kodu z listingu 4.4 tylko tym, że teraz tablica new\_data jest budowana z użyciem danych ciągłych i danych kategorialnych (zakodowanych metodą one-hot). Cały pozostały kod jest taki sam. Zauważ jednak, że dane wyjściowe są wyraźnie inne. Najpierw pojawiają się dane pokazane poniżej, po czym kod wyświetla wykres zbliżony do zawartości rysunku 4.8.

```

Szanse: [[ 1.00142244  0.48843238  1.          0.97071389
           0.77855355  0.95091843  0.85321151  0.98791223
           0.99385787  0.95983606  0.81406157  1.
           0.869761     1.          0.94819493  1.
           0.96913702  0.91475145  0.81320387  0.99837556
           0.97071389  0.869761     0.9778946   1.
           0.81320387  0.99385787  0.94499953  0.82707014
           0.98791223  0.98257044]]

```

Punkt przecięcia osi dla szans: [ 0.61316832]

Punkt przecięcia osi dla prawdopodobieństwa: [ 0.38010188]

Patrząc na dwa wykresy zwrócone przez obie wersje eksperymentu, można uznać, że dodatkowe cechy mają bardzo niewielki wpływ na skuteczność klasyfikatora. Więcej na temat skuteczności dowiesz się z następnego punktu. Najpierw jednak powinieneś zrozumieć, dlaczego w tym eksperymencie parametry modelu przyjmują tak odmienną postać w porównaniu z poprzednim przykładem.

Zauważ, że w nowej wersji używanych jest w sumie 30 parametrów modelu. Z czego to wynika, skoro jest tylko sześć cech? Powodem jest sposób kodowania. Na podstawie każdej możliwej wartości zmiennej kategorialnej tworzona jest cecha ustawiana na wartość 1 tylko wtedy, gdy dana wartość została zaobserwowana. To oznacza, że w danym

momencie aktywna („gorąca”) może być tylko jedna z wartości. Analiza parametrów modelu utworzonego w wyniku uczenia pokazuje, że dla dwóch pierwszych cech wartości nie zmieniły się znacznie w porównaniu z poprzednim eksperymentem. Dla większości pozostałych cech uzyskane wartości to 1 i mniej. Wskazuje to, że niezerowe wartości zakodowanych cech zmniejszają prawdopodobieństwo oszustwa, przy czym nie w takim stopniu, aby zmieniać ostateczną klasyfikację danych w drugim eksperymencie.

Warto dodać, że te eksperymenty to tylko przykład. Nie zaleca się treningu modelu na podstawie tak niewielkiej liczby punktów danych. Ponadto wyniki uzyskane dla znacznie większej liczby cech prawie na pewno byłyby inne od otrzymanych tutaj.

Spróbujmy teraz podsumować, czego udało nam się dokonać w tym podrozdziale. Zaczęliśmy od najprostszego modelu liniowego, czyli narysowania linii prostej biegającej przez punkty danych. Następnie rozbudowaliśmy ten model do wersji liniowej określającej logarytm szans wystąpienia docelowej zmiennej. Ta dająca duże możliwości technika pozwala na to, by zmiana atrybutu skutkowała nielinową zmianą szans (ale liniową zmianą logarytmu szans). Pokazaliśmy, jak zastosować tę technikę do rozwiązania praktycznego problemu, wymagającego ustalenia, czy transakcja finansowa została wykonana przez oszusta. Przedstawiliśmy też służący do tego kod. W dalszych rozdziałach wracamy do regresji logistycznej, na razie jednak żegnamy się z wykrywaniem oszustw.

#### 4.4. Czy wyniki są wiarygodne?

Załóżmy, że zbudowałeś klasyfikator oparty na regresji logistycznej, sieciach neuronowych lub innej technice. Jak stwierdzić, czy dane rozwiązanie działa dobrze? Jak zbadać, czy utworzony inteligentny algorytm można zastosować w środowisku produkcyjnym i zdobyć podziw współpracowników oraz pochwały przełożonego? Ocena klasyfikatora jest równie ważna jak proces jego budowania. W branży (a zwłaszcza na spotkaniach z przedstawicielami handlowymi!) zetkniesz się z różnymi stwierdzeniami — od przesadnych po zupełnie bezsensowne. Ten podrozdział ma pomóc Ci oceniać własne klasyfikatory (jeśli jesteś programistą) i pomóc w ustaleniu skuteczności rozwiązań rozwijanych przez inne osoby (jeżeli jesteś programistą lub menedżerem produktu).

Zacznijmy od stwierdzenia, że żaden klasyfikator nie radzi sobie równie dobrze w przypadku każdego problemu i dowolnego zbioru danych. Możesz potraktować to jako informatyczną wersję powiedzeń „nikt nie jest wszechwiedzący” i „każdy popełnia błędy”. Techniki uczenia omawiane w związku z klasyfikowaniem należą do kategorii uczenia nadzorowanego. Uczenie jest „nadzorowane”, ponieważ klasyfikator przechodzi przez proces treningu na podstawie znanych klasyfikacji i pod kontrolą próbuje nauczyć się informacji zawartych w treningowym zbiorze danych. Łatwo się domyślić, że podobieństwo danych treningowych do danych ze środowiska produkcyjnego ma bardzo duży wpływ na powodzenie klasyfikacji.

Aby omówienie było zrozumiałe, przedstawiamy tu zestaw pojęć i łączymy je z zagadnieniami wprowadzonymi w podrozdziale 1.6. W celu uproszczenia opisu uwzględniamy standardowy problem klasyfikacji binarnej (na przykład identyfikowanie niechciających wiadomości lub oszustw). Założymy, że chcesz wykrywać, czy daną wiadomość

należy uznać za spam. Podstawowym narzędziem do oceny wiarygodności klasyfikatora i częstym punktem wyjścia do takich analiz jest *macierz błędów* (ang. *confusion matrix*). Jest to prosta macierz, gdzie wiersze określają kategorię, do której klasyfikator przypisuje dany obiekt, a kolumny oznaczają kategorię, do której dany obiekt rzeczywiście należy. Dla klasifikacji binarnej taka macierz obejmuje tylko cztery komórki. Ogólny przypadek (klasyfikacja z uwzględnieniem wielu klas) działa w podobny sposób, jednak wymaga dużo bardziej skomplikowanych analiz.

W tabeli 4.1 przedstawiona jest macierz błędów dla klasifikacji binarnej (dotyczącej na przykład odfiltrowywania spamu lub wykrywania oszustw). W tej tabeli pokazane są możliwe wyniki klasifikacji binarnej. Jeśli klasyfikator uzna daną wiadomość za spam, jest to wskazanie *pozytywne*. W przeciwnym razie wskazanie jest *negatywne*. Oczywiście samo wskazanie może być poprawne (trafne) lub niepoprawne (fałszywe). Dlatego macierz obejmuje cztery wyniki — możliwe kombinacje wskazań pozytywnego i negatywnego oraz trafnego i fałszywego. Pozwala to też zauważać dwa rodzaje błędów. Błędy pierwszego rodzaju to fałszywe wskazania pozytywne. Są to *błędy pierwszego rodzaju*. Inny typ pomyłek to fałszywe wskazania negatywne — *błędy drugiego rodzaju*. Można opisać to w prosty sposób: gdy popełniasz błąd pierwszego rodzaju, skazujesz niewinną osobę, a błąd drugiego rodzaju oznacza uwolnienie winnego. Ta analogia dobrze nadaje się do wyjaśnienia znaczenia kosztów klasifikacji. Wolter wolałby uwolnić 100 winnych osób niż skazać jedną niewinną. Ta wrażliwość nadal cechuje europejskie sądy. Morał z tej anegdoty jest taki, że decyzje prowadzą do konsekwencji, które nie zawsze są symetryczne. Jest to prawda zwłaszcza w kontekście klasifikacji z wieloma klasami. Przyjrzyj się pokazanym dalej definicjom (niektóre z nich zostały przedstawione w podrozdziale 1.6).

- *Współczynnik FP = FP/N, gdzie N = TN+FP*
- *Swoistość = 1-współczynnik FP = TN/N*
- *Czułość = TP/P, gdzie P = TP+FN*
- *Precyzja = TP/(TP+FP)*
- *Trafność = (TP+TN)/(P+N)*
- *F-score = precyzja × trafność*

**Tabela 4.1.** Typowa macierz błędów dla prostego problemu klasifikacji binarnej

	<b>Wskazanie pozytywne</b>	<b>Wskazanie negatywne</b>
<b>Trafne</b>	Trafne predykcje pozytywne (TP)	Trafne predykcje negatywne (TN)
<b>Fałszywe</b>	Fałszywe predykcje pozytywne (FP)	Fałszywe predykcje negatywne (FN)

Założmy, że dowiadujesz się o klasyfikatorze o trafności szacowanej na 75%. Jak bliskie są te szacunki rzeczywistej trafności klasyfikatora? Jak prawdopodobne jest uzyskanie trafności 75% po powtórzeniu klasifikacji na innych danych? Aby odpowiedzieć na to pytanie, posłużymy się procesem nazywanym w statystyce *procesem Bernoulliego*. Jest on opisywany jako sekwencja niezależnych zdarzeń, których skutki są uznawane za sukces lub niepowodzenie. Doskonale pasuje to do zadania wykrywania oszustw, ogólnie, klasifikacji binarnej. Przyjmijmy, że rzeczywista trafność to A\*, a zmierzona trafność wynosi A. Celem jest ustalenie, czy A jest dobrym oszacowaniem A\*.

Może pamiętasz z kursów statystyki pojęcie *przedziału ufności*. Jest to miara pewności, jaką można przypisać konkretnemu twierdzeniu. Jeśli trafność wynosi 75% dla zbioru 100 wiadomości, poziom zaufania do wyniku nie będzie wysoki. Jeżeli jednak podobną trafność uzyskasz dla zbioru 100 000 wiadomości, zaufanie będzie znacznie wyższe. Intuicyjnie zrozumiałe jest, że wraz ze wzrostem wielkości zbioru przedział ufności maleje i można uznać wyniki za bardziej przekonujące. Można wykazać, że dla zbioru Bernoulliego obejmującego 100 próbek rzeczywista trafność z pewnością 80% wynosi między 69,1% a 80,1%<sup>11</sup>. Jeśli zwiększyesz dziesięciokrotnie wielkość zbioru używanego do pomiaru trafności klasyfikatora, to z tym samym poziomem pewności (80%) rzeczywista trafność będzie znajdować się w nowym przedziale od 73,2% do 76,7%. Wzory na wyznaczanie takich przedziałów znajdziesz w każdym dobrym podręczniku do statystyki. Teoretycznie te wyniki są poprawne dla prób obejmujących więcej niż 30 elementów. W praktyce liczba elementów może być dowolna.

Niestety, w praktyce często masz mniej elementów, niżbyś chciał. Aby rozwiązać ten problem, specjalisci od uczenia maszynowego opracowali liczne techniki pomagające ocenić wiarygodność wyników klasyfikacji dla niewielkich zbiorów danych. Standardowa metoda oceny to *dziesięciokrotna walidacja krzyżowa* (ang. *10-fold cross-validation*). Jest to prosta procedura, którą najlepiej jest przedstawić na przykładzie. Założmy, że danych jest 1000 wiadomości, które zostały już ręcznie poklasyfikowane. Aby ocenić klasyfikator, część tych wiadomości należy przypisać do zbioru treningowego, a część — do zbioru testowego. Dziesięciokrotna walidacja krzyżowa wymaga podzielenia 1000 wiadomości na 10 grup po 100 elementów. Każda porcja 100 wiadomości powinna zawierać mniej więcej taki sam odsetek nieuchanych wiadomości co cały 1000-elementowy zbiór. Następnie należy przeprowadzić trening klasyfikatora na podstawie dziewięciu z tych grup. Po zakończeniu treningu trzeba przetestować klasyfikator z użyciem 100 wiadomości, które nie były uwzględniane w czasie treningu. Następnie można dokonać pomiarów (z których część została opisana wcześniej); zwykle mierzona jest trafność klasyfikatora. Cały proces powtarzany jest dziesięciokrotnie. Za każdym razem test przeprowadzany jest na innej grupie 100 wiadomości. Po zakończeniu prób dostępnych jest dziesięć wartości trafności, na podstawie których można obliczyć średnią trafność.

Może zastanawiasz się, czy trafność będzie inna, gdy pierwotny zbiór danych rozbijesz na 8 lub 12 części. Tak się stanie; uzyskanie identycznych wyników jest mało prawdopodobne. Jednak nowa uśredniona trafność będzie zbliżona do tej uzyskanej wcześniej. Wyniki z dużej liczby testów na różnych zbiorach danych i z użyciem wielu różnych klasyfikatorów wskazują na to, że dziesięciokrotna walidacja krzyżowa generuje reprezentatywną miarę klasyfikatora.

Skrajną odmianą dziesięciokrotnej walidacji krzyżowej jest umieszczanie w zbiorze treningowym wszystkich wiadomości oprócz jednej i używanie tej ostatniej w teście. Ta technika to *leave-one-out* (czyli „z pominięciem jednego”). Ma ona teoretyczne zalety, jednak dla zbiorów danych stosowanych w praktyce (obejmujących setki tysięcy, a nawet miliony elementów) koszty obliczeń są często zniechęcające. Można zdecydować

<sup>11</sup> Ian Witten i Eibe Frank, *Data Mining* (Morgan Kaufmann, 2005).

się pomijać pojedyncze elementy, ale stosować ten zabieg tylko do części elementów zbioru. Ta metoda to *bootstrap*. Jej podstawowe założenie jest takie, że można utworzyć zbiór treningowy, pobierając próbki z pierwotnego zbioru danych ze zwracaniem. Każdy element z pierwotnego zbioru może więc zostać pobrany więcej niż raz. Jeśli utworzysz zbiór treningowy z 1000 wiadomości, konkretna wiadomość może wystąpić w nim wielokrotnie. W testach uwzględnianych będzie wtedy około 368 wiadomości, które nie wystąpiły w zbiorze treningowym. Wielkość zbioru treningowego to wciąż 1000 wiadomości, ponieważ część z pozostałych 632 wiadomości powtarza się w zbiorze treningowym. Matematyczne wyjaśnienie tych wartości znajdziesz w książce *Data Mining* Wittena i Franka.

Badacze ustalili, że narysowanie wykresu wartości TPR względem FPR jest przydatną metodą analizowania wiarygodności klasyfikatora. Z rozdziału 1. dowiedziałeś się, że te wykresy to *krzywe ROC*, mające źródło w teorii wykrywania sygnału z lat 70. ubiegłego wieku. W ostatnich latach pojawiło się wiele prac z obszaru uczenia maszynowego, w których wykresy ROC są używane do analizowania skuteczności klasyfikatorów (jednego lub grupy). Podstawowe założenie jest takie, że krzywa ROC powinna być jak najbardziej oddalonej od przekątnej wykresu TPR/FPR.

W praktyce systemy klasyfikacji są często używane jako systemy wspomagania podejmowania decyzji. Błędy w klasyfikacji mogą prowadzić do niewłaściwych decyzji. W niektórych sytuacjach podjęcie błędnej decyzji jest wprawdzie niepożądane, ale stosunkowo niegroźne. Jednak w innych przypadkach od decyzji może zależeć czyjeś życie. Pomyśl o lekarzu, który w wyniku błędnej diagnozy nie wykrył nowotworu, lub o znajdującym się w krytycznej sytuacji w przestrzeni kosmicznej kosmonaucie polegającym na wynikach z Twojego klasyfikatora. W trakcie oceny systemu klasyfikacji należy zbadać zarówno poziom wiarygodności, jak i powiązane koszty klasyfikacji. Gdy stosowana jest klasyfikacja binarna, można określić *funkcję kosztów* (jest to funkcja współczynników FP i FN).

Oto podsumowanie — jednym z najważniejszych aspektów klasyfikatorów jest wiarygodność wyników. W tym podrozdziale opisaliśmy różne miary pomagające ją ocenić: precyję, trafność, czułość i swoistość. Połączenie tych miar może dawać nowe miary, takie jak wskaźnik F-score. Opisaliśmy też walidację krzyżową wyników, polegającą na podziale zbioru treningowego na różne sposoby i sprawdzaniu zmienności miar klasyfikatora wraz ze zmianami w zbiorach danych. W następnym podrozdziale omawiamy różne problemy związane z dużymi zbiorami danych.

## 4.5. Klasyfikowanie w bardzo dużych zbiorach danych

Zbiory danych używane przez naukowców są często małe w porównaniu z rozwiązaniami stosowanymi w praktyce. Transakcyjne zbiory danych w dużych korporacjach obejmują od 10 do 100 milionów rekordów lub nawet więcej. Podobnej wielkości są zbiory roszczeń ubezpieczeniowych, pozycji w dziennikach firm telekomunikacyjnych, odnotowanych cen akcji, kliknięć, pozycji w dziennikach kontrolnych itd. Dlatego w produkcyjnych aplikacjach przetwarzanie dużych zbiorów danych jest normą, a nie wyjątkiem. Nie ma przy tym znaczenia, czy dana aplikacja działa w sieci WWW.

Klasifikowanie w bardzo dużych zbiorach danych zasługuje na specjalną uwagę z przynajmniej trzech powodów. Oto one: poprawne reprezentowanie zbioru danych w zbiorze treningowym, złożoność obliczeniowa fazy treningowej i szybkość działania klasyfikatora.

Niezależnie od dziedziny, w jakiej działa aplikacja, i od mechanizmów udostępnianych przez klasyfikator trzeba zapewnić, że dane treningowe są reprezentatywną próbką danych ze środowiska produkcyjnego. Nie możesz zakładać, że klasyfikator będzie działał tak dobrze, jak wskazują na to pomiary z etapu sprawdzania poprawności, jeśli dane treningowe nie odzwierciedlają w wysokim stopniu danych produkcyjnych. Powtarzamy to, aby podkreślić znaczenie tej kwestii! W wielu sytuacjach początkowa eksytacja szybko zmienia się w rozczarowanie, ponieważ wspomniany warunek nie jest spełniony. Jak jednak zapewnić reprezentatywność danych treningowych?

Zadanie klasifikacji binarnej jest łatwiejsze do rozwiązania, ponieważ występują tylko dwie klasy. Wiadomość może być albo chciana, albo niechciana. Transakcja albo jest oszustwem, albo nim nie jest. W takiej sytuacji, przy założeniu, że dostępna jest wystarczająca liczba elementów treningowych z obu klas, należy skoncentrować się na pokryciu wartości atrybutów w elementach treningowych. Można posłużyć się tu czysto empiryczną oceną („Tak, to akceptowalne rozwiązanie. Mamy wystarczająco dużo wartości. Można udostępnić kod w środowisku produkcyjnym”), w pełni naukowym podejściem (próbkowanie danych w czasie i sprawdzanie, czy rozkład statystyczny próbek jest taki sam jak w danych treningowych) lub pośrednim rozwiązaniem. W praktyce zwykle stosuje się ostatnią z tych opcji. Można nazwać ją półempiryczną metodą uczenia nadzorowanego. Empiryczny aspekt dotyczy tego, że w ramach oceny kompletności zbioru treningowego przyjmowanych jest szereg sensownych założeń odzwierciedlających zrozumienie i znajomość danych używanych w aplikacji. Aspekt naukowy związany jest z podstawowym zbieraniem statystycznych informacji o danych, na przykład: wartości minimalnej i maksymalnej, średniej, mediany, wartości odstających, procentu brakujących wartości atrybutów itd. Te informacje można wykorzystać do wyboru nieuwzględnianych wcześniej danych z aplikacji i dołączenia ich do zbioru treningowego.

Klasifikacja wieloklasowa jest oparta na podobnych zasadach co klasifikacja binarna. Jednak oprócz wcześniej wspomnianych wskazówek trzeba uwzględnić dodatkową złożoność. Nowy problem polega na tym, że elementy należy dobrać w taki sposób, by wszystkie klasy były równie dobrze reprezentowane w zbiorze treningowym. Odróżnianie 1000 klas jest znacznie trudniejsze niż wybór jednej z dwóch możliwości. Wielowymiarowa (wieloatrybutowa) klasifikacja wieloklasowa związana jest z dodatkowymi trudnościami wynikającymi z „przekleństwa wymiarów” (zobacz rozdział 2.).

Jeśli baza danych zawiera 100 milionów rekordów, naturalnie zechcesz wykorzystać wszystkie dostępne dane. Na etapie projektowania klasyfikatora powinieneś zastanowić się nad skalowaniem kodu klasyfikatora w fazach treningowej i sprawdzania poprawności. Jeżeli zamierzasz podwoić wielkość danych treningowych, odpowiedz sobie na następujące pytania:

- O ile wzrośnie czas treningu klasyfikatora?
- Jaka będzie trafność klasyfikatora dla nowego (większego) zbioru danych?

Prawdopodobnie powinieneś uwzględnić więcej miar jakości niż samą trafność. Zapewne zechcesz też uwzględnić inne ilości danych (czterokrotność pierwotnych danych, ich ośmiokrotność itd.). Zasada powinna być jedna zrozumiała. Możliwe, że klasyfikator działa świetnie dla małego przykładowego zbioru danych (szybko się uczy i cechuje się wysoką trafnością), natomiast znacznie gorzej sprawdza się dla dużo większego zbioru danych. Jest to istotne, ponieważ czas wprowadzania produktu na rynek zawsze jest ważny, a „inteligentne” moduły aplikacji powinny podlegać tym samym regułom budowania kodu co pozostałe części oprogramowania.

Ważna jest też szybkość pracy klasyfikatora na trzecim etapie cyklu życia — w środowisku produkcyjnym. Możliwe, że klasyfikator szybko zakończył trening i zapewnia wysoką trafność, nie będzie to przydatne, jeśli nie skaluje się dobrze w środowisku produkcyjnym. Na etapie sprawdzania poprawności powinieneś zmierzyć szybkość działania i zależność od wielkości danych. Założmy, że używasz klasyfikatora, którego czas działania rośnie kwadratowo względem ilości danych. Jeśli wielkość danych zostanie podwojona, czas ich przetwarzania wydłuży się czterokrotnie. Ponadto przyjmijmy, że inteligentny moduł używa klasyfikatora w tle do wykrywania oszustw. Jeśli w ramach sprawdzania poprawności klasyfikacja 10 000 rekordów zajęła 10 minut, to przetwarzanie 10 milionów rekordów zajmie około 10 milionów minut! Zapewne nie masz do dyspozycji takiej ilości czasu, dlatego powinieneś albo wybrać inny klasyfikator, albo poprawić szybkość działania obecnego rozwiązania. W systemach produkcyjnych użytkownicy często rezygnują z trafności na rzecz szybkości. Jeśli klasyfikator jest niezwykle trafny, ale też skrajnie powolny, zwykle okazuje się bezużyteczny.

Zwróć uwagę na osobliwości używanego systemu klasyfikacji. Jeśli stosujesz system oparty na regułach, możesz natrafić na *problem użyteczności* (ang. *utility problem*). Proces uczenia się (poznawania reguł) może prowadzić do ogólnego spowolnienia systemu w środowisku produkcyjnym. Istnieją sposoby na uniknięcie lub przynajmniej ograniczenie tego problemu<sup>12</sup>. Musisz jednak je znać i upewnić się, że dane rozwiązanie jest zgodne z tymi technikami. Spadek wydajności nie jest oczywiście jedynym problemem. Musisz też zapewnić sposoby zarządzania regułami i porządkowania ich. Jest to problem z dziedziny inżynierii, którego rozwiązanie w dużym stopniu zależy od dziedziny, w jakiej działa aplikacja. Zwykle im bardziej skomplikowany jest kod klasyfikatora, tym więcej staranności potrzeba, aby zrozumieć charakterystykę działania klasyfikatora (jego szybkość i jakość).

## 4.6. Podsumowanie

- Przedstawiliśmy tu taksonomię ogólnej dziedziny klasyfikowania. Uwzględniliśmy algorytmy statystyczne i strukturalne. Algorytmy statystyczne próbują dopasować do danych określoną funkcję, często z użyciem algorytmicznych technik w celu zapewnienia maksymalnego prawdopodobieństwa uzyskania rzeczywistych danych na podstawie modelu. W algorytmach strukturalnych uwzględniane są

---

<sup>12</sup>R.B. Doorenbos, *Production Matching for Large Learning Systems*, praca doktorska, Carnegie Mellon (CMU, 1995).

aspekty takie jak odległość między punktami lub zestawy reguł dzielące przestrzeń cech w celu wykrycia zależności między cechami a docelowymi wynikami.

- Skoncentrowaliśmy się na konkretnym algorytmie statystycznym, regresji logistycznej, i przedstawiliśmy jego intuicyjne podstawy, pokazując jego powiązania z modelem prostoliniowym. Regresja logistyczna jest liniowa, jeśli chodzi o logarytm szans wystąpienia docelowej zmiennej, i stanowi przydatny sposób szacowania prawdopodobieństwa wystąpienia zdarzenia o wielu cechach. Pokazaliśmy to na podstawie malego zbioru danych zawierającego oszuńcze i poprawne transakcje finansowe.
- W końcowej części rozdziału opisaliśmy zestaw miar działania algorytmu i omówiliśmy problemy z implementacją klasyfikowania w dużych zbiorach danych.
- W świecie dużych zbiorów danych algorytmy klasyfikacji są często stosowane do bardzo dużych zbiorów danych, przy czym nierzadko muszą działać w czasie rzeczywistym. Rodzi to nowe wyzwania, o których tu tylko napomknęliśmy. Skalowanie procesu treningu pod kątem dużych zbiorów danych wymaga dobrej znajomości zarówno analizowanej dziedziny, jak i stosowanych technik uczenia maszynowego. Sama teoria nie wystarczy. Duża część prac i zasobów związana jest z podtrzymywaniem działania systemów podejmowania decyzji w środowisku produkcyjnym.



# *Studium przypadku — prognozowanie kliknięć w reklamie internetowej*

## **Zawartość rozdziału:**

- Stosowany w rzeczywistości duży inteligentny system
- Targetowanie użytkowników na podstawie danych o korzystaniu z internetu
- Stosowanie regresji logistycznej do oceny skłonności użytkowników do interakcji

Prognozowanie kliknięć w internecie to konkretny problem ze świata reklamy. Jest to doskonały przykład problemu z dużą ilością szybko generowanych danych, w którym decyzje trzeba podejmować z jak najmniejszym opóźnieniem. Rozwiązań z tego obszaru mają liczne zastosowania. Handel elektroniczny, optymalizacja witryn, media społecznościowe, internet rzeczy, układy czujników i gry internetowe prowadzą do szybkiego generowania danych. W tych obszarach przydatne jest szybkie przetwarzanie danych i podejmowanie decyzji na podstawie możliwie najświeższych informacji.

Za każdym razem, gdy uruchamiasz przeglądarkę i otwierasz stronę internetową, podejmowane są tysiące, a nawet miliony decyzji, aby ustalić, jakie reklamy mają się pojawić. Te decyzje są podejmowane w wyniku komunikacji między wieloma źródłami danych i ustalenia, czy konkretna reklama wywrze pozytywny wpływ na użytkownika.

Cały proces musi zachodzić szybko, ponieważ decyzje trzeba podjąć przed zakończeniem wczytywania strony.

Z perspektywy reklamodawcy można tylko oszacować pozytywny wpływ na podstawie interakcji z reklamą. Należy więc spróbować przewidzieć prawdopodobieństwo interakcji użytkownika z reklamą na stronie na podstawie wszystkich informacji związanych z użytkownikiem i kontekstem. Ponieważ w sieci WWW szybko generowane są duże ilości danych, a decyzje trzeba podejmować szybko, zadanie nie jest łatwe.

Rozwiązań prezentowanych w tym rozdziale można wykorzystać w różnych dziedzinach. Pomyśl na przykład o inteligentnym mieście, w którym wszystkie samochody są wyposażone w czujniki określające prędkość i wybraną drogę. Można wtedy szybko podejmować decyzje o otwieraniu i zamykaniu kontrapasów lub przekierowywaniu ruchu w celu eliminowania korków. Występuje tu podobny wzorzec co przy wyświetlaniu reklam. Czy możliwe jest pobieranie szybko napływających dużych ilości danych i budowanie na tej podstawie modeli pozwalających na szybkie podejmowanie decyzji? Na szczęście tak! Aby odpowiednio objąć rozwiązanie, trzeba zagłębić się w mechanizmy reklamy internetowej. Jednak w celu maksymalizacji ogólności tekstu staramy się ograniczyć aspekty specyficzne dla reklamy i skoncentrować na inteligentnych algorytmach, w miarę możliwości nawiązując do innych obszarów zastosowań.

## **5.1. Historia i informacje wstępne**

Internauci mają bardzo zróżnicowane opinie na temat reklamy internetowej. Mimo to wydaje się, że branża ta zostanie z nami na dłużej. Wydatki na reklamę internetową co roku rosną o około 5%, a w 2015 roku w Stanach Zjednoczonych miały wynieść 189,38 miliarda dolarów. Zgodnie z prognozami ten wzrost utrzyma się do roku 2018, kiedy to ta kwota ma wynieść 220,55 miliarda dolarów<sup>1</sup>. Ponieważ coraz większe kwoty przepływają z tradycyjnych mediów (takich jak prasa i telewizja) do świata internetu, a reklamodawcy zaczynają wykorzystywać nowe sposoby interakcji z mediami i urządzeniami, rolą technologii w tej branży będzie rosnąć.

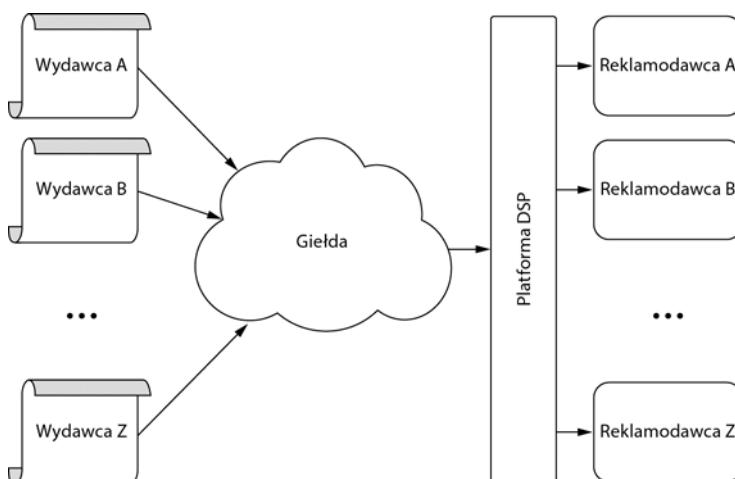
Początkowy wzrost popularności reklamy internetowej po części wynikał z większych możliwości pomiaru skuteczności<sup>2</sup>. Reklamodawcy mogli po raz pierwszy dokładnie zmierzyć, do ilu użytkowników dotarli i, co ważniejsze, jak później zachowywały się te osoby. Nastąpił rozwitk branży, ponieważ w tradycyjnych kanałach skuteczność reklamy można było mierzyć tylko na podstawie jej pośredniego wpływu na poziom sprzedaży. Można było ustalić, czy dany użytkownik kupił produkt i jakie zachowania do tego doprowadziły. Możliwe stało się także testowanie efektywności reklam „w miejscu”. Reklamodawcy mogą testować wiele różnych reklam, rysunków i tekstów w czasie rzeczywistym, a następnie eliminować te, które nie zapewniają oczekiwanych efektów, i zwiększać nakłady na bardziej skuteczne wersje. Tego rodzaju optymalizacja ma wiele zastosowań także poza światem reklamy.

<sup>1</sup> Total US Ad Spending to See Largest Increase Since 2004, „eMarketer”, 2 lipca 2014, <http://mng.bz/AnIG>.

<sup>2</sup> Bill Gurley, *The End of CPM, „Above the Crowd”*, 10 lipca 2000, <http://mng.bz/Vq6D>.

Nastąpił więc ruch w kierunku podejmowania decyzji na podstawie danych. Możliwe stało się rejestrowanie coraz większych ilości danych i wykorzystywanie ich do lepszego targetowania użytkowników. To z kolei doprowadziło do powstania zupełnie nowego ekosystemu, który ma służyć zarówno klientom, jak i sprzedawcom. W tym rozdziale omawiamy *gieldę* (ang. *exchange*) i *platformę DSP* (ang. *demand-side platform* — **DSP**). I robimy to dokładnie. Zobaczysz też, jak wykorzystać zebrane informacje o użytkownikach, aby zwiększyć prawdopodobieństwo interakcji tych osób z reklamą.

Na rysunku 5.1 przedstawiona jest graficzna ilustracja omawianego ekosystemu. Widać tu, że giełda umożliwia wydawcom (witrynom takim jak <http://theguardian.com>, <http://huffingtonpost.com> i innym) sprzedaż przestrzeni reklamowej (slotów i obszarów o określonej wielkości) konkretnym reklamodawcom (firmom takim jak Nike, Adidas, O2, Vodafone itd.). Reklamodawcy często kupują miejsca na reklamy za pomocą *platfromy DSP*.



**Rysunek 5.1.**  
Ekosystem reklamy internetowej. Wydawcy udostępniają swoją ofertę (miejsca na reklamy) na giełdzie, do której możliwy jest dostęp programowy (za pomocą inteligentnych algorytmów). Platforma DSP gromadzi informacje o popycie na miejsce na reklamy od reklamodawców i kupuje dla zainteresowanych te miejsca

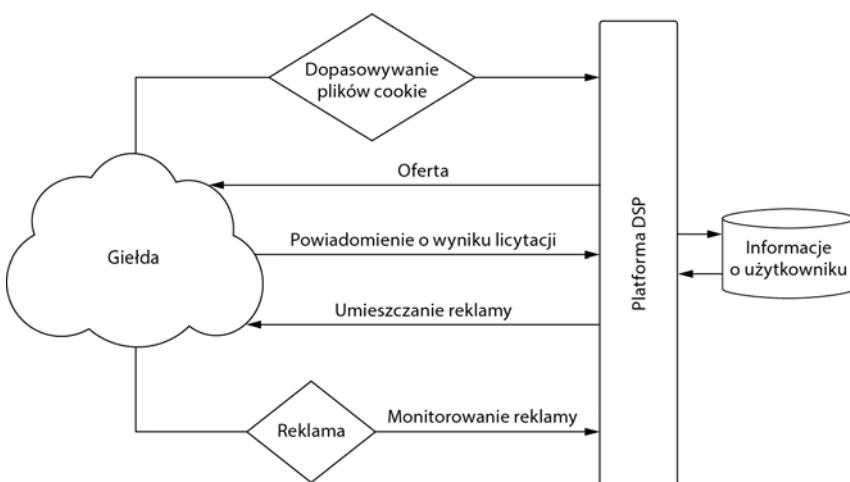
Platforma DSP obsługuje wielu reklamodawców i łączy ich siłę nabywczą, aby dotrzeć do najlepszych wydawców w sieci WWW. W sieci działa wiele platform DSP: Turn (<http://www.turn.com>), MediaMath (<http://www.mediamath.com>), DataXu (<http://www.dataxu.com>) i inne. Te platformy to kanały umożliwiające zakup miejsc na reklamy. Widoczna jest tu analogia do handlu akcjami. Nie każdy inwestor zostanie wpuszczony na parkiet giełdowy, dlatego domy maklerskie działają na rzecz wielu inwestorów indywidualnych chcących kupić akcje. Platforma DSP to odpowiednik domu maklerskiego, a giełda reklam to odpowiednik giełdy papierów wartościowych. Platforma DSP zbiera popyt (podobnie jak dom maklerski), a obie giełdy pełnią funkcję platformy, na której można handlować określonymi aktywami.

Na rynkach reklamy i papierów wartościowych obecnie częściej handlem zajmują się intelligentne algorytmy niż ludzie. W świecie finansów ta technika nosi nazwę **HFT** (ang. *high-frequency trading*). W świecie reklamy nazywamy ją *programmatic buying*, ponieważ przestrzeń reklamowa jest kupowana za pomocą programu komputerowego (inteligentnego algorytmu), a nie przy pomocy człowieka.

Inteligentny algorytm służy do określania wartości użytkownika i dostępnej przestrzeni reklamowej, co wpływa na cenę proponowaną (i ostatecznie płaconą) za wyświetlenie reklamy danej osobie. W dalszych podrozdziałach przedstawiamy algorytm używany do określania wspomnianej wartości. Najpierw jednak powinieneś dowiedzieć się więcej o procesie udostępniania sprzedawanej przestrzeni reklamowej na giełdzie. Dalej przedstawiamy rzeczywiste dane i omawiamy komplikacje, które pojawiają się w trakcie treningu dużych zbiorów danych. Bez dalszego przedłużania — zaczynajmy!

## 5.2. Giełda

Na rysunku 5.1 przedstawiliśmy na ogólnym poziomie, jak sprzedawana i kupowana jest przestrzeń reklamowa (miejscła, na których w witrynie wyświetlane są reklamy). Za pojęciem *giełda* kryje się wysoce skomplikowany mechanizm. W tym podrozdziale wyjaśniamy te komplikacje i przedstawiamy ustrukturyzowaną wersję procesu pozwalającą zilustrować go krok po kroku. Rysunek 5.2 przedstawia ten proces w graficznej postaci.



**Rysunek 5.2.** Graficzna ilustracja etapów zakupu przestrzeni reklamowej na giełdzie. Zdarzenia zachodzą od góry do dołu. Najpierw ma miejsce dopasowywanie plików cookie. Może się to odbywać albo na giełdzie, albo w platformie DSP. Gdy platforma DSP otrzyma unikatowy identyfikator, wykorzystuje go do sprawdzenia dodatkowych informacji o użytkowniku i ustalenia oferowanej kwoty (ten etap omawiamy od punktu 5.4.5). Jeśli oferta zostanie przyjęta, zwarcane jest powiadomienie rejestrowane przez platformę DSP. Następny etap polega na ustaleniu przez platformę DSP wyświetlanej reklamy. Fizycznie reklama może być przekazywana na stronę przez samą giełdę lub przez platformę DSP. Informacje o zdarzeniach związanych z reklamą są kierowane z powrotem do platformy DSP niezależnie od tego, gdzie reklama się znajduje

### 5.2.1. Dopasowywanie plików cookie

Jest to pierwszy krok wykonywany za każdym razem, gdy wczytywana jest strona z reklamami. *Plik cookie* to przechowywany po stronie przeglądarki unikatowy identyfikator użytkownika w każdej domenie (lub witrynie albo usłudze sieciowej). Pliki cookie są unikatowe, ale istnieją tylko przez ustalony czas lub do momentu opróżnienia przez

użytkownika pamięci podręcznej z takimi plikami w przeglądarce. Z powodu ścisłych zabezpieczeń obowiązujących w przeglądarce różne usługi sieciowe nie mogą współużytkować plików cookie. Jeśli giełda ma przekazać dane użytkownika platformie DSP, aby umożliwić ustalenie, czy warto wyświetlić reklamę danej osobie, musi zajść jeden z dwóch procesów.

### **DOPASOWYWANIE PLIKÓW COOKIE PO STRONIE GIEŁDY**

Pierwsze rozwiązywanie wymaga unikatowego odwzorowania plików cookie z platformy DSP na pliki cookie z giełdy przechowywane po jej stronie. W tym scenariuszu podczas wczytywania witryny następuje komunikacja z giełdą, gdzie wyszukiwany jest odpowiedni identyfikator pliku cookie na potrzeby danej platformy DSP. Następnie do platformy DSP zwarcane są szczegółowe informacje o przestrzeni reklamowej wraz z unikalnym identyfikatorem przeznaczonym dla tej platformy DSP. Potem platforma DSP musi ustalić dodatkowe informacje o użytkowniku na podstawie własnych źródeł i zdecydować, co zrobić z oferowaną przestrzenią. Zaletą tej metody jest to, że nie trzeba samodzielnie zarządzać usługą przetwarzającą pliki cookie, choć operator giełdy prawie na pewno obciąży klienta za to kosztami.

### **DOPASOWYWANIE PLIKÓW COOKIE W PLATFORMIE DSP**

Druga możliwość to zarządzanie usługą przetwarzania plików cookie po stronie platformy DSP. Gdy strona jest wczytywana, giełda kontaktuje się z platformą DSP, podając identyfikator pliku cookie (i dodatkowe informacje o przestrzeni reklamowej), a klient samodzielnie wyszukuje potrzebne dane. To oczywiście zwiększa złożoność oprogramowania podejmującego decyzje, ponieważ wyszukiwanie danych musi odbywać się błyskawicznie (wymiana komunikatów z giełdą musi odbywać się w ciągu milisekund). Początkowo może to wydawać się tańsze, jednak usługę trzeba zbudować i konserwować.

W obu modelach na poziomie platformy DSP efekt jest taki sam. Za każdym razem, gdy wczytywana jest strona, na której za pośrednictwem giełdy oferowana jest przestrzeń reklamowa, platforma DSP pobiera unikatowy identyfikator użytkownika pozwalający wyszukać dodatkowe informacje. Platforma DSP pobiera też informacje o dostępnej przestrzeni reklamowej. Wykorzystanie tych informacji przez platformę DSP to istota intelligentnego algorytmu prezentowanego w tym rozdziale. Aby dana platforma DSP była konkurencyjna względem innych platform, musi wykorzystywać wszystkie historyczne dane do prognozowania przeszłych zachowań danego użytkownika. Dzięki temu platforma może zaproponować odpowiednią cenę za wyświetlenie reklamy użytkownikowi, maksymalizując prawdopodobieństwo pokazania mu reklamy i minimalizując ryzyko zmarnowania pieniędzy.

#### **5.2.2. Oferty**

Gdy określony jest już unikatowy identyfikator użytkownika i ogólne informacje na temat oferowanej przestrzeni reklamowej, platforma DSP musi podjąć dwie decyzje: czy wyświetlić reklamę w danej przestrzeni i jaką kwotę warto za tę możliwość zaoferować.

Te na pozór proste decyzje są związane z całą serią komplikacji. Jeśli operator platformy DSP pobiera opłaty za kliknięcie, może szacować oferty na podstawie

prawdopodobieństwa kliknięcia (a tym samym i zysku finansowego). To podstawowe podejście jest tematem punktu 5.4.4. Możliwe jednak, że cele są inne. Budżety kampanii reklamowych o odmiennych zakresach są różne. Czasem dla reklamodawcy ważne jest, by zmieścić się w budżecie lub rozdzielić go równomiernie na cały czas trwania kampanii. Kampanie mogą też współzawodniczyć o tę samą przestrzeń reklamową u określonych wydawców. Możliwe też, że reklamodawcy nie interesują kliknięcia, ale konwersje (czyli sytuacje, gdy użytkownik kliknie reklamę, a następnie kupi produkt). Widać więc, że na pozór proste zadanie okazuje się trudnym problemem z zakresu optymalizacji wymagającym uwzględnienia wielu celów!

### **5.2.3. Powiadomienie o wygranej (lub przegranej) w licytacji**

Gdy platforma DSP ustali, czy i w jakiej wysokości składać ofertę, przesyła ją, a następnie otrzymuje powiadomienie od giełdy o wyniku licytacji. Jeśli oferta okazała się za niska, nie są podejmowane żadne dalsze działania. Ponieważ dana platforma DSP nie otrzymała przestrzeni reklamowej, rejestruje porażkę i jest to koniec komunikacji związanej z tą konkretną możliwością reklamową.

Jeśli dana platforma DSP otrzymała przestrzeń reklamową, giełda zwraca cenę tej przestrzeni. W większości giełd stosowane są aukcje w *systemie Vickreya*<sup>3</sup>, w których zwycięzca (oferujący najwyższą cenę) płaci tylko tyle, ile wynosiła druga najwyższa oferta. Dlatego ostateczną cenę trzeba zwrócić do platformy DSP, aby została zarejestrowana przed przejęciem do następnego etapu.

### **5.2.4. Umieszczanie reklamy**

Umieszczanie reklamy to najprostszy z wszystkich etapów. Po wygraniu przestrzeni reklamowej trzeba w niej wyświetlić reklamę. Na wszystkich giełdach używane są reklamy o zdefiniowanych wymiarach<sup>4</sup>, a wszyscy reklamodawcy i wszystkie kampanie udostępniają wcześniej przygotowany zestaw reklam (ang. *creatives*). Na niektórych giełdach obowiązuje proces akceptacji reklam, pozwalający sprawdzić reklamy pod kątem jakości i praw do marek jeszcze przed złożeniem oferty.

Podobnie jak dopasowywanie plików cookie, tak i umieszczanie reklamy może być wykonywane z poziomu platformy DSP lub giełdy. To drugie podejście zapewnia większe bezpieczeństwo giełdzie, ale związane jest z dodatkowymi kosztami dla operatora platformy DSP.

### **5.2.5. Monitorowanie reklam**

Ostatni etap w procesie związanym z aukcją umożliwia zbieranie danych na temat wyświetlonej reklamy. Te informacje z kilku przyczyn są bezcenne. Pierwszy powód jest taki, że reklamodawcy muszą wiedzieć, czy użytkownik wszedł w interakcję z reklamą — czy kliknął ją, obejrzał lub udostępnił dalej. Bez tego nie da się uzyskać miar, za pomocą których firmy mogą mierzyć sukces. Nie da się też generować faktur opar-

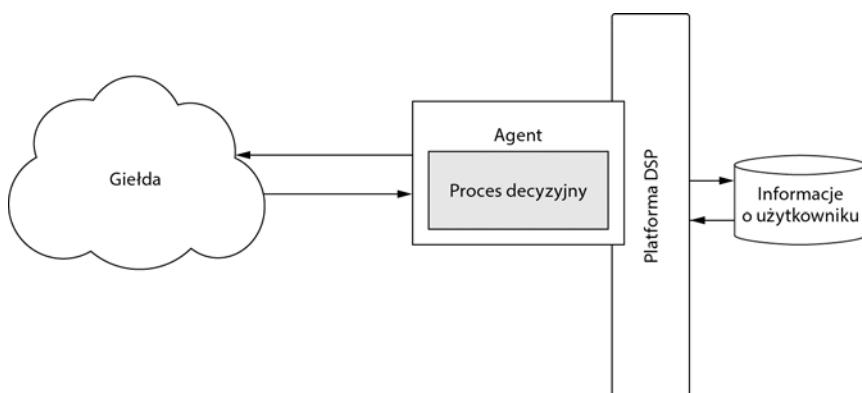
<sup>3</sup> Vijay Krishna, *Auction Theory*, wydanie drugie (Academic Press, 2009).

<sup>4</sup> Internet Advertising Bureau, *IAB Standard Ad Unit Portfolio*, <http://www.iab.com/guidelines/iab-standard-ad-unit-portfolio>.

tych na takich miarach! Drugi powód dotyczy wagi tych informacji dla operatora platformy DSP. Jeśli wiadomo, co interesuje użytkownika, można tworzyć i wyświetlać reklamy prowadzące do bardziej pozytywnych reakcji ze strony danego użytkownika i podobnych mu osób. Monitorowanie to zarówno wymóg dotyczący bieżącej działalności, jak i narzędzie do skutecznego prowadzenia biznesu.

### 5.3. Czym jest agent?

Zrozumienie procesu licytacji to pierwszy krok w kierunku zrozumienia kontekstu, w jakim musi działać inteligentny algorytm. Do tej pory przedstawiliśmy tylko ogólny obraz kupowania reklam na giełdzie. Wiesz tylko tyle, że otrzymujesz od giełdy pewne informacje i musisz zwrócić ofertę, aby zapewnić sobie możliwość wyświetlenia reklamy określonymu użytkownikowi w danej przestrzeni reklamowej. W praktyce używany jest do tego agent działający mniej więcej tak, jak ilustruje to rysunek 5.3.



**Rysunek 5.3.** Agent składa się z zewnętrznej powłoki (wykonującej podstawowe zadania związane ze składaniem ofert) i z systemu podejmowania decyzji (uwzględniającego wszystkie dane dotyczące oferty)

Agent składa się z zewnętrznej powłoki i rdzenia. Zewnętrzna powłoka odpowiada za podstawowe zadania związane ze składaniem ofert — na przykład za szybkie pobieranie informacji o użytkowniku i wysyłanie odpowiedzi do giełdy. W rdzeniu agenta działa system podejmowania decyzji, odpowiedzialny za uwzględnienie wszystkich danych o użytkowniku, przestrzeni reklamowej i kontekstu oraz zwracanie szacowanego prawdopodobieństwa kliknięcia. W dalszej części tego podrozdziału omawiamy wybrane wymagania dotyczące zewnętrznej powłoki agenta, a następnie, w podrozdziale 5.4, przechodzimy do szczegółów systemu podejmowania decyzji (powłoki wewnętrznej).

#### 5.3.1. Wymagania stawiane agentowi

Ogólnie agenta można oceniać na podstawie wyników w opisanych dalej kategoriach. Choć przedstawiona lista nie jest kompletna, omówione tu kwestie są niezwykle istotne w zastosowaniach wymagających szybkiej reakcji. Opisujemy tu konsekwencje różnych wyborów i możliwe kompromisy:

- **Szybkość.** Jest to najważniejsza z przedstawianych kategorii. Agent *musi* działać szybko. Wszystkie giełdy narzucają partnerom bardzo wygórowane wymagania w obszarze czasu reakcji. Od rozpoczęcia wczytywania strony do przeprowadzenia aukcji i przydzielenia przestrzeni reklamowej mija zwykle około 100 milisekund. Regularne przekraczanie tego czasu przez agenta może skutkować niewykorzystanymi możliwościami, a w najgorszym przypadku wykluczeniem z giełdy. Operator giełdy musi dbać o jej reputację wśród wydawców i nie może ryzykować pustych slotów reklamowych, które nie tylko wyglądają nieatrakcyjnie na stronie wydawcy, ale też oznaczają dla niego niewykorzystaną możliwość zarobku.
- **Prostota.** Jednym ze sposobów na osiągnięcie dużej szybkości jest prosty projekt. Agent powinien wykonywać tylko jedno zadanie oraz być łatwy w debugowaniu i konserwacji. Pamiętaj, że gdy agent jest aktywny i bierze udział w aukcjach, każdy jego przestój oznacza utratę dochodów dla Ciebie i reklamodawców.
- **Trafność.** Giełdy udostępniają wiele możliwości. Jedną z przyczyn trudności w korzystaniu z giełdy jest to, że łatwo jest kupić dużo mało wartościowych miejsc na reklamę i w dodatku zrobić to bardzo szybko! Dlatego niezwykle ważna jest trafność agenta. Z czasem powinieneś móc zauważać dodatnią korelację między oferowaną ceną a skutecznością reklam.
- **Rozszerzalność.** Działa wiele giełd, na których można kupować przestrzeń reklamową. Niektóre miejsca na reklamę są dostępne tylko na jednej giełdzie, inne — na wielu. Aby zmaksymalizować zasięg danej platformy DSP, warto zintegrować ją z jak największą liczbą giełd. Dlatego rozszerzanie agenta powinno być łatwe. Nie poddaj się pokusie ścisłego powiązania agenta z pierwszą giełdą, z którą będziesz go integrował. Zadbaj o to, by rozwiązanie było jak najbardziej uniwersalne.
- **Ocena.** Gdy agent już działa i osiąga zadowalające wyniki, warto rozpocząć usprawnienie procesu decyzyjnego, aby pozyskiwać jak najlepszych użytkowników jak najniższym kosztem. Każdą modyfikację agenta i procesu decyzyjnego trzeba monitorować, aby upewnić się, że zmiany przynoszą oczekiwane efekty. Często skutki zmian stają się widoczne dopiero po kilku dniach lub tygodniach, a nie w czasie minut i godzin.

#### 5.4. Czym jest system podejmowania decyzji?

W poprzednich punktach przedstawiliśmy podstawy procesu kupowania przestrzeni reklamowej w internecie za pośrednictwem giełdy. Opisaliśmy, jak zadanie to jest wykonywane w praktyce za pomocą agenta. W tym podrozdziale koncentrujemy się na konkretnym aspekcie agenta. Wyjaśniamy, jak wykorzystać nieprzetworzone dane udostępniane przez giełdę i przekształcić je na kwotę, jaką warto zapłacić za emisję reklamy. Przyjrzyj się teraz rodzajom informacji, jakimi można posłużyć się do generowania takich wartości.

#### **5.4.1. Informacje o użytkowniku**

W praktyce jedyną informacją udostępnianą na temat użytkownika jest identyfikator pliku cookie. To platforma DSP musi rejestrować i przechowywać takie informacje. Gdy użytkownik zostaje napotkany po raz pierwszy, żadne informacje o nim nie są dostępne. Jeśli ta osoba wejdzie w interakcję z wyświetlona reklamą, zostanie to zapisane razem z innymi interakcjami z reklamami z danej platformy DSP. Użytkownik może na przykład uruchomić reklamę filmową lub zamknąć reklamę pełnoekranową. Te proste interakcje są zapisywane, a później sprawdzane po napotkaniu użytkownika. W ten sposób platforma DSP może zbudować obraz działań użytkownika wskazujących na wysokie prawdopodobieństwo kliknięcia (a także takich, które zwykle nie prowadzą do kliknięć).

#### **5.4.2. Informacje o przestrzeni reklamowej**

Obok danych o użytkowniku także informacje o przestrzeni reklamowej mogą służyć do prognozowania kliknięć. Niezależnie od użytkownika reklamy o określonej wielkości i u określonych wydawców często dają lepsze wyniki. Na przykład reklamy w czołowych witrynach informacyjnych (takich jak <http://theguardian.co.uk>) mogą charakteryzować się wyższym współczynnikiem kliknięć (ang. *click-through rate — CTR*) niż reklamy w witrynach małych sklepów.

#### **5.4.3. Informacje o kontekście**

Są też źródła informacji, które nie należą do kategorii *użytkownik* lub *przestrzeń reklamowa*. Zwykle trudniej je wykorzystać, ale są one równie wartościowe. Wiadomo na przykład, że długie weekendy (na przykład z czarnym piątkiem<sup>5</sup> i cyberponiedziałkiem<sup>6</sup>) zwykle cechują się wyższym współczynnikiem kliknięć, natomiast w słoneczne dni liczba kliknięć jest mniejsza niż w deszczowe, ponieważ więcej osób wychodzi do tradycyjnych sklepów na zakupy, zamiast pozostać w domu i kupować w internecie.

#### **5.4.4. Przygotowywanie danych**

Pierwszy etap budowania systemu podejmowania decyzji to przygotowanie danych. Ponieważ miarą sukcesu jest kliknięcie reklamy przez użytkownika, możliwe są dwa scenariusze: pozytywny (dana osoba kliknęła reklamę) i negatywny (użytkownik nie kliknął reklamy). Potrzebne są więc dane treningowe reprezentujące obie te możliwości. Ponieważ ilość rejestrowanych danych jest bardzo duża, ich zbieranie może sprawiać trudności — związane nie tylko z przechowywaniem informacji, ale też z trenowaniem modelu. Potrzebny jest sposób na zmniejszenie ilości danych bez zakłócania występujących w nich wzorców. Ten efekt można uzyskać w wyniku redukcji danych, czyli losowego wyboru mniejszego podzbioru danych.

---

<sup>5</sup> BBC News, *Black Friday: Online Spending Surge in UK and US*, 27 listopada 2015, <http://www.bbc.co.uk/news/business-34931837>.

<sup>6</sup> BBC News, *Cyber Monday Adds to Online Sales Surge*, 30 listopada 2015, <http://www.bbc.co.uk/news/business-34961959>.

Generowanie nowego, mniejszego zbioru danych odbywa się w następujący sposób: przede wszystkim należy zachować wszystkie emisje reklamy prowadzące do kliknięcia. Tych cennych punktów danych nie należy redukować, jednak ponieważ reprezentują one tylko niewielki odsetek wszystkich emisji reklam (średni współczynnik kliknięć w branży wynosi znacznie poniżej 1%), redukcja nie jest tu potrzebna. Zamiast tego można znacznie zredukować emisje, które nie doprowadziły do kliknięcia. Dzięki temu wielkość zbioru danych będzie dużo bardziej akceptowalna. Skutkuje to także zrównoważeniem zbioru danych. Zamiast bardzo dużego zbioru z niewielką liczbą skutecznych emisji (z jakim tylko niewielkie techniki klasyfikacji i regresji potrafią sobie dobrze radzić) otrzymujesz dużo bardziej zrównoważony zbiór, w którym skuteczne emisje stanowią większy odsetek wszystkich danych. Więcej o równoważeniu zbioru danych dowieš się z punktu 5.5.2.

#### **5.4.5. Model dla systemu podejmowania decyzji**

W rozdziale 4. opisaliśmy regresję logistyczną i posłuzyliśmy się nią do wykrywania oszustw w bardzo małym zbiorze danych. To samo podejście było z powodzeniem używane w wielu platformach DSP do obliczania prawdopodobieństwa interakcji użytkownika z reklamą. W poprzednim rozdziale zastosowaliśmy sztywny próg dla danych wyjściowych z modelu regresji logistycznej, aby utworzyć klasyfikator. Jednak w tym rozdziale wykorzystujemy dane wyjściowe w nieprzetworzonej postaci, traktując je jak prawdopodobieństwo kliknięcia. Przypomnij sobie ogólną postać modelu regresji logistycznej:

$$\ln \left( \frac{y}{1-y} \right) = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

W tym wzorze  $y$  to prawdopodobieństwo analizowanego zdarzenia,  $1-y$  to prawdopodobieństwo odwrotnego zdarzenia,  $\beta_i$  dla  $i > 0$  reprezentuje współczynniki cech, a  $\beta_0$  określa wartość progową. Aby ustalić logarytm szansy kliknięcia, do treningu modelu wykorzystamy dane przygotowane w sposób opisany w punkcie 5.4.4. Na ich podstawie będziemy obliczać logarytm szansy (i później prawdopodobieństwo) kliknięcia w czasie składania ofert, podstawiając odpowiednie wartości w pokazanym wzorze.

#### **5.4.6. Odwzorowywanie prognozowanego współczynnika kliknięć na oferowaną kwotę**

Gdy dostępny jest wytrenowany w opisany sposób model, to w momencie zażądania przez giełdę oferty można stosunkowo łatwo odwzorować dane wyjściowe z modelu na oferowaną kwotę. Aby ułatwić zrozumienie tekstu, warto zdefiniować kilka pojęć:

- **CPI** — koszt wyświetlenia reklamy (ang. *cost per impression*),
- **CPM** — koszt za 1000 wyświetleń reklamy,
- **CTR** — współczynnik kliknięć (liczba kliknięć/liczba emisji),
- **CPC** — koszt kliknięcia (ang. *cost per click*).

Zwykle giełdy przyjmują oferty za CPM. Zamiast określić proponowaną kwotę dla każdej emisji, wystarczy podać ofertę za 1000 wyświetleń. Przy założeniu, że klient podaje docelowy koszt kliknięcia (czyli ile jest gotów zapłacić za kliknięcie), otrzymujemy dobrze zdefiniowane zadanie:

$$\text{Oferta CMP} = \text{CPC} \times E(\text{CTR}) \times 1000$$

$$\text{Oferta CMP} = \text{CPC} \times y \times 1000$$

Oznacza to, że podawana oferta powinna być proporcjonalna do prawdopodobieństwa kliknięcia (ustalonego na podstawie informacji o przestrzeni reklamowej i użytkowniku) pomnożonego przez koszt kliknięcia i przez 1000.

#### **5.4.7. Inżynieria cech**

W punktach od 5.4.1 do 5.4.4 opisaliśmy podstawowe cechy, które mogą być dostępne w czasie zgłoszania ofert. Jednak cechy te nie zawsze są używane w ich podstawowej postaci. Te cechy służą do generowania setek innych cech. Liczne z nich stają się binarnymi wskaźnikami informującymi o statusie użytkownika. Można na przykład uwzględnić odwidzone domeny i wielkość reklam, aby dla każdej kombinacji tych elementów uzyskać odrębną cechę. Dzięki temu nie trzeba zakładać, że domena i wielkość przestrzeni reklamowej są niezależnymi predyktorami kliknięć. Pozwala to modelowi wybierać dające najlepsze wyniki przestrzenie reklamowe w poszczególnych witrynach. Wielu operatorów platform DSP i projektantów algorytmów z tego obszaru nie ujawnia — co zrozumiałe — takich szczegółów, ponieważ stanowią one tajemnicę sukcesu. Grupy z dziedziny nauk o danych stale starają się za pomocą inżynierii cech poprawić działanie algorytmów.

#### **5.4.8. Trening modelu**

Ponieważ dostępne są duże ilości danych treningowych, ważne jest, aby możliwe było uczenie z wykorzystaniem pamięci zewnętrznej. Choć Python z pakietem scikit-learn doskonale nadaje się do eksploracji danych i przetwarzania umiarkowanych ich ilości, nie sprawdza się, gdy wielkość zbioru danych jest naprawdę duża. Dzieje się tak, ponieważ pakiet scikit-learn wymaga, aby wszystkie przetwarzane dane znajdowały się w pamięci. Dlatego zdecydowaliśmy się zastosować otwartą bibliotekę Vowpal Wabbit (VW) Johna Langforda ([https://github.com/JohnLangford/vowpal\\_wabbit](https://github.com/JohnLangford/vowpal_wabbit)). Biblioteka VW cieszy się coraz większym zainteresowaniem i jest aktywnie rozwijana przez wiele osób ze społeczności zainteresowanej uczeniem maszynowym. Przedstawiamy tu krótkie wprowadzenie do tej biblioteki, dzięki czemu będziesz mógł ocenić dane z giełdy. Zachęcamy jednak do zapoznania się z dokumentacją VW dostępną w serwisie GitHub<sup>7,8</sup>.

---

<sup>7</sup> John Langford, *Vowpal Wabbit command-line arguments*, <http://mng.bz/YTlo>.

<sup>8</sup> John Langford, samouczek projektu Vowpal Wabbit, <http://mng.bz/GI40>.

## 5.5. **Predykcja kliknięć za pomocą biblioteki Vowpal Wabbit**

Wcześniej wspomnieliśmy, że Vowpal Wabbit to szybka biblioteka do uczenia maszynowego, która potrafi trenować modele bez wczytywania wszystkich danych do pamięci. VW współdziała z algorytmami z kategorii *liniowego uczenia nadzorowanego* (przy czym obecnie udostępnia też obsługę algorytmów nieliniowych). Uczenie jest *nadzorowane*, ponieważ klasy są określane przez programistę, oraz *liniowe*, ponieważ (jak pokazaliśmy wcześniej) używany model jest liniowy.

W tym podrozdziale wykorzystamy możliwości biblioteki VW i rzeczywiste dane wygenerowane w czołowej platformie DSP Criteo. Firma Criteo udostępnia te dane, aby pobudzić rozwój badań w omawianej dziedzinie, jest w tym jednak pewien haczyk. Dane są pozbawione informacji semantycznych, dlatego nie wiadomo, z jakiego obszaru pochodzą i jakie jest znaczenie poszczególnych wartości. Pobierz kompletny zbiór danych (zajmuje on 4,5 GB)<sup>9</sup>, a dalej poznasz kroki potrzebne do opracowania i uruchomienia klasyfikatora. Dowiesz się też, jak uniknąć niektórych pułapek czyhających na przyszłych praktyków.

Najpierw trzeba zainstalować bibliotekę VW. Szczegółowe informacje znajdziesz w serwisie GitHub<sup>10</sup>, a także w pliku z archiwum z przykładami z książki. Ponieważ VW to aktywnie prowadzony projekt badawczy, dokumentacja jest czasem nieaktualna względem najnowszej wersji kodu. Dlatego najnowsze informacje znajdziesz na liście mailingowej, za pośrednictwem której możesz też skontaktować się z autorami VW.

Dostęp do narzędzia VW jest możliwy z poziomu wiersza poleceń. Sterowanie odbywa się głównie za pomocą opcji wiersza poleceń. VW daje bardzo duże możliwości, a tu omawiamy tylko niektóre funkcje. Szczegółowe informacje znajdziesz w serwisie GitHub i na liście mailingowej. Gorąco zachęcamy do korzystania z tych źródeł, jeśli zamierzasz rozwijać inne aplikacje przy użyciu narzędzia VW.

### 5.5.1. **Format danych używany w VW**

W VW używany jest specyficzny format danych, który daje dużo swobody, ale początkowo może być trudny do zrozumienia. Zanim zaczniesz, skonwertuj plik udostępniany przez Criteo na postać zrozumiałą dla VW. Przykładowe nieprzetworzone dane udostępniane przez Criteo przedstawia rysunek 5.4.

Criteo udostępnia 13 cech całkowitoliczbowych i 26 cech kategorialnych. Dane są prezentowane w takiej postaci, że w jednym wierszu najpierw występują wszystkie cechy całkowitoliczbowe, a następnie cechy kategorialne.

Zgodnie z dołączonym do danych plikiem *readme* cechy całkowitoliczbowe dotyczą głównie liczników (opisujących na przykład to, ile razy użytkownik coś zrobił). Wartości cech kategorialnych są przedstawione w postaci 32-bitowych skrótów. To oznacza, że wartości całkowitoliczbowe charakteryzują się semantyką całkowitoliczbową (większa wartość oznacza więcej wystąpień danego zdarzenia), nie można jednak przypisać semantyki wartościom kategorialnym. Założymy, że każda wartość kategorialna repre-

<sup>9</sup> Criteo, *Kaggle Display Advertising Challenge Dataset*, <http://mng.bz/75iS>.

<sup>10</sup> John Langford, wiki projektu Vowpal Wabbit, <http://mng.bz/xt7H>.

Class	Integer 1	Integer 2	Integer 12	Integer 13	Categorical 1	Categorical 2	Categorical 25	Categorical 26
0	1	7		5	0	68fd1e64	80e26c9b	7b4723c4
0	2	35	...	44	1	68fd1e64	f0cf0024	41274cd7
1	2	3		1	14	287e684f	0a519c5c	c18be181

Czy użytkownik wszedł w interakcję? Te kolumny przedstawiają 13 cech reprezentowanych za pomocą liczb całkowitych. Prawdopodobnie są to liczniaki zdarzeń związane z użytkownikiem Następne kolumny opisują 26 cech kategorialnych związanych ze zgłoszaniem ofert

**Rysunek 5.4.** Przegląd nieprzetworzonych danych udostępnianych przez Criteo. Pierwsze pole związane jest z prognozowanym zdarzeniem. Opisuje ono wykonanie przez użytkownika określonej operacji (na przykład kliknięcia reklamy). Następnych 13 kolumn to cechy całkowitoliczbowe reprezentujące liczniaki. Wyższa liczba oznacza, że dane zdarzenie zaszło większą liczbą razy. Kolejnych 26 cech to dane kategorialne, zapewne reprezentujące aspekty związane z żądaniami podania ofert (na przykład witrynę, z której wystosowano takie żądanie)

zentuje określoną cechę — na przykład witrynę, którą użytkownik przeglądał w momencie zgłoszenia żądania podania oferty. Zauważ, że zastosowanie skrótów zapewnia spójność tych danych. Ten sam skrót zawsze reprezentuje tę samą pierwotną wartość. Dlatego choć znaczenie poszczególnych skrótów jest nieznane, pozostaje ono takie samo dla każdego wystąpienia danego skrótu.

Aby dane były zgodne z formatem używanym w VW, najpierw trzeba zmienić etykiety klas. W VW wynik pozytywny jest kodowany za pomocą wartości 1, a negatywny — jako -1. Jeszcze ważniejsze jest to, by zmienić format każdego wiersza, dodając symbol potoku (|) rozdzielający poszczególne cechy. Należy też określić, które cechy są kategorialne, a które ciągłe. Jest to niezwykle istotne, ponieważ jeśli nie wprowadzisz takiego rozróżnienia, model nie będzie mógł uwzględnić opisanej wcześniej semantyki liczników. Wtedy każda unikatowa wartość liczniaka będzie traktowana jak etykieta, a model nie będzie mógł wykryć, że wszystkie wartości liczniaka są powiązane ze sobą zgodnie z semantyką liczb całkowitych (na przykład 2 jest większe niż 1, 3 jest większe niż 2 itd.). Jeśli przyjrzyz się katalogowi z materiałami do tego rozdziału, zobaczysz, że napisaliśmy już dla Ciebie potrzebny kod!

Aby kontynuować wykonywanie przykładu z tego rozdziału, ściagnij plik *dac.tar.gz* z witryny firmy Criteo<sup>11</sup>, wypakuj ten plik i pobierz plik z danymi *train.txt* z podkatalogu *dac*. Następnie uruchom dla tego pliku treningowego skrypt Pythona, wywołując następujące polecenie:

```
python ch5-criteo-process.py </ścieżka/do/train.txt>
```

To spowoduje utworzenie dwóch plików, *train\_vw\_file* i *test\_vw\_file*. Są one tworzone w katalogu, w którym znajduje się wspomniany skrypt. Zauważ, że jeśli chcesz, możesz zmienić lokalizację plików wyjściowych z tego skryptu, podając dwa dodatkowe parametry:

```
python ch5-criteo-process.py </ścieżka/do/train.txt> </ścieżka/do/train_vw_file>
→</ścieżka/do/test_vw_file>
```

<sup>11</sup>Criteo, <http://mng.bz/75iS>.

Każdy wynikowy plik zawiera dane w następującym formacie:

```
-1 |i1 c:1 |i2 c:1 |i3 c:5 |i4 c:0 |i5 c:1382 |i6 c:4 |i7 c:15 |i8 c:2 |i9
↳c:181 |i10 c:1 |i11 c:2 |i13 c:2 |c1 68fd1e64 |c2 80e26c9b |c3 fb936136
↳|c4 7b4723c4 |c5 25c83c98 |c6 7e0ccccc |c7 de7995b8 |c8 1f89b562 |c9
↳a73ee510 |c10 a8cd5504 |c11 b2cb9c98 |c12 37c9c164 |c13 2824a5f6 |c14
↳1adce6ef |c15 8ba8b39a |c16 891b62e7 |c17 e5ba7672 |c18 f54016b9 |c19
↳21ddcdc9 |c20 b1252a9d |c21 07b5194c |c23 3a171ecb |c24 c5c50484 |c25
↳e8b83407 |c26 9727dd16
```

Przyjrzymy się dokładnie tym danym. Atrybuty są w nich rozdzielone symbolem potoku (|). Zauważ, że pierwsza wartość to -1. Jest to docelowa klasa dla tych danych. Dane dotyczą więc reklamy, która nie została kliknięta przez użytkownika. Dodatnia wartość 1 oznaczałaby kliknięcie.

Zwróć uwagę na czytelne dla człowieka etykiety: i1, i2, i3 itd. Są to przestrzenie nazw. Wybraliśmy przestrzenie nazw rozpoczynające się od i do opisu cech całkowitoliczbowych i przestrzenie nazw rozpoczynające się od c do opisu cech kategorialnych. Przestrzenie nazw to zaawansowany mechanizm biblioteki VW umożliwiający eksperymenty z użyciem różnych przestrzeni. W ten sposób można na przykład generować współczynniki modelu dla cech pochodnych za pomocą prostej opcji podawanej w wierszu poleceń. Jeśli wiesz, że przestrzeń nazw c1 reprezentuje witrynę, z której nadeszło żądanie podania oferty, a przestrzeń nazw c2 określa wielkość reklamy, możesz łatwo dodać do modelu nową cechę „wielkość reklamy dla poszczególnych witryn”. Zauważ też, że dla niektórych przestrzeni nazw dostępne są dodatkowe informacje oznaczone symbolem c:. Są to nazwy cech z danej przestrzeni nazw. Jeśli nazwa nie jest podana, zostaje ustalona na podstawie wartości. Ten mechanizm pozwala podkreślić różnice między zmiennymi kategorialnymi a ciągłymi. Dalej przedstawiamy konkretny przykład, aby mieć pewność, że rozumiesz to zagadnienie.

Przestrzeń nazw c1 z poprzedniego przykładu obejmuje cechę 68fd1e64. Niejawnie używana jest wartość 1 określająca, że cecha 68fd1e64 jest obecna. Jest to więc cecha kategorialna. Można ją zapisać tak: 68fd1e64:1. Z kolei przestrzeń nazw i1 obejmuje cechę c. Jest ona powiązana ze zmienną ciągłą 1 określającą liczbę wystąpień zdarzenia (o nazwie c) powiązanego z i1. W praktyce takie rozróżnienia są ważne, ponieważ skutkują różnicą między wyznaczaniem w treningu współczynnika dla zmiennej ciągłej (jest to prawidłowe rozwiązanie dla semantyki całkowitoliczbowej) lub dla każdej napotkanej wartości tej zmiennej (to błędne podejście!). Nie omawiamy dalej tego zagadnienia, zachęcamy jednak do zapoznania się z dokumentacją przestrzeni nazw w wiki projektu<sup>12</sup>.

Wcześniej wspomnieliśmy, że trzeba utworzyć dwa pliki. Można je wygenerować bezpośrednio na podstawie danych treningowych od firmy Criteo (*train.txt*). Ponieważ dane z Criteo zostały pierwotnie przygotowane na potrzeby konkursu, plik z danymi testowymi (*test.txt*) dostępny w pakiecie nie obejmuje żadnych etykiet. Dlatego niemożliwe jest omówienie skuteczności klasyfikatora bez zgłoszenia go do konkursu. Aby utwo-

---

<sup>12</sup>John Langford, format wejściowy biblioteki Vowpal Wabbit, <http://mng.bz/2v6f>.

### W jaki sposób cechy są reprezentowane w VW?

W VW używany jest wektor bitowy o określonej długości (*tablica cech*), a każda cecha jest wiązana za pomocą skrótu z tym wektorem. Wagi są ustalane w wyniku uczenia nie dla cech, ale dla elementów wspomnianego wektora.

Ma to szereg zalet, w tym zredukowaną przestrzeń cech o stałej wielkości używaną na etapie uczenia, jednak może skutkować kolizjami, gdy dwie różne cechy zostaną powiązane z tym samym bitem. Zwykle (jeśli wielkość wektora jest odpowiednia) ma to jednak pomijalny wpływ na trafność wytrenowanego klasyfikatora. Inny skutek tego rozwiązania jest taki, że w celu ustalenia wpływu cechy (analizy wyuczonych wag) trzeba określić skrót odpowiadający elementowi wektora bitowego oraz obliczyć iloczyn skalarny wektora i wag. Omawiamy tę kwestię dalej.

rzyć własny zbiór testowy, dzielimy dane z pliku *train.txt* na dwie części w stosunku 70 do 30; 70% danych treningowych trafia do nowego pliku z treningowym zbiorem danych (*train\_vw\_file*), natomiast pozostałe 30% jest zapisywane w pliku testowym (*test\_vw\_file*), który posłuży do oceny rozwiązań. Skoro już rozumiesz używany format danych i wewnętrzną reprezentację danych w bibliotece VW, a także uzyskałeś zbiory treningowy i testowy, możemy przejść do przygotowywania danych.

#### 5.5.2. Przygotowywanie zbioru danych

Najpierw warto określić poziom zrównoważenia danych, czyli tego, czy liczba wyników pozytywnych jest mniej więcej zbliżona do liczby wyników negatywnych. Jest to ważny krok, ponieważ używany dalej algorytm treningowy jest wrażliwy na takie aspekty danych. Jeśli zbiór danych nie jest zrównoważony, współczynniki modelu logistycznego będą aktualizowane pod kątem negatywnych wyników częściej niż pod kątem wyników pozytywnych. Oznacza to, że wyniki negatywne będą miały większy wpływ niż pozytywne. Nie jest to idealne rozwiązanie. Z tym problemem najlepiej poradzić sobie, wymuszając zrównoważenie danych przed treningiem.

Równowagę można zapewnić za pomocą operacji z poziomu wiersza poleceń. Uwaga dla rozsądnych: używane pliki są duże, dlatego ich edycja za pomocą edytora tekstu może spowodować przepełnienie pamięci i awarię aplikacji. Na listingu 5.1 znajdziesz bezpieczną technikę zapewniania równowagi w danych.

##### Listing 5.1. Ustalanie poziomu zrównoważenia danych

```
wc -l train_vw_file ← | Używanie liczby słów (wc) do ustalenia liczby wierszy
> 32090601 train_vw_file | w pliku z danymi

grep -c '^1' train_vw_file ← | Zliczanie wierszy rozpoczęjących się od -1
> 23869264 | (przykład negatywny)

grep -c '^1' train_vw_file ← | Zliczanie wierszy rozpoczęjących się od -1
> 8221337 | (przykład pozytywny)
```

Posłużyliśmy się tu prostymi narzędziami z wiersza poleceń, aby ustalić liczbę pozytywnych i negatywnych przykładów w zbiorze treningowym. Każdy przykład treningowy zajmuje w pliku z danymi odrębny wiersz. Dlatego za pomocą pierwszej instrukcji można określić, ile elementów znajduje się w zbiorze treningowym. W wierszach

drugim i trzecim używamy narzędzia grep do pobrania i zliczenia wierszy rozpoczęjących się od wyniku negatywnego i pozytywnego. Widać tu, że liczba przykładów negatywnych jest mniej więcej trzy razy wyższa od liczby przykładów pozytywnych. Zauważ, że te dane nie są reprezentatywne dla standardowego współczynnika kliknięć. Kliknięcia zwykle stanowią znacznie mniejszy odsetek wszystkich przykładów. Uważamy, że ten zbiór danych został w nieujawnionym stopniu zredukowany przez firmę Criteo.

Następnie trzeba wymieszać dane. Dlaczego? VW stosuje *uczenie na żywo* (ang. *online learning*). Pamiętaj, że używane jest podejście z wykorzystaniem pamięci zewnętrznej, dlatego w pamięci nie musi znajdować się cały zbiór danych. Aby możliwe było skuteczne uczenie wiersz po wierszu, dane muszą być pobierane losowo zgodnie z ich rozkładem. Najprościej zapewnić to za pomocą wymieszania danych przed ich przetwarzaniem wiersz po wierszu. Więcej informacji o stochastycznym spadku wzduż gradientu (ang. *stochastic gradient descent* — SGD), który jest stosowaną tu do treningu modelu metodą uczenia na żywo, znajdziesz w teksthach *Online Learning and Stochastic Approximations*<sup>13</sup> i *Stochastic Gradient Tricks*<sup>14</sup> Bottou.

Do równoważenia i mieszania zbioru danych na potrzeby treningu ponownie użyjemy narzędzi z wiersza poleceń. Na listingu 5.2 pokazane są instrukcje pozwalające pobrać, wymieszać i utworzyć zbiór danych.

#### Listing 5.2. Równoważenie, mieszanie i tworzenie treningowego zbioru danych

```

Pobieranie negatywnych przykładów, mieszanie ich
i zapisywanie w pliku negative_examples.dat
grep '^1' train_vw_file | sort -R > negative_examples.dat

Pobieranie pozytywnych
przykładów, mieszanie ich
i zapisywanie w pliku
positive_examples.dat
grep '^1' train_vw_file | sort -R > positive_examples.dat
awk 'NR % 3 == 0' negative_examples.dat > \
    negative_examples_downsampled.dat

Użycie narzędzia awk
do pobrania co trzeciego
przykładu z pliku
negative_examples.dat
cat negative_examples_downsampled.dat > all_examples.dat
cat positive_examples.dat >> all_examples.dat

Zapisywanie zredukowanych
przykładów negatywnych
w pliku all_examples.dat
cat all_examples.dat | sort -R > all_examples_shuffled.dat
awk 'NR % 10 == 0' all_examples_shuffled.dat > \
    all_examples_shuffled_down.dat

Redukowanie
zbioru
treningowego
Dołączanie przykładów
pozytywnych do negatywnych
Mieszanie zrównoważonego
zbioru treningowego
```

**Pobieranie negatywnych przykładów, mieszanie ich  
i zapisywanie w pliku negative\_examples.dat**

**Pobieranie pozytywnych  
przykładów, mieszanie ich  
i zapisywanie w pliku  
positive\_examples.dat**

**Użycie narzędzia awk  
do pobrania co trzeciego  
przykładu z pliku  
negative\_examples.dat**

**Zapisywanie zredukowanych  
przykładów negatywnych  
w pliku all\_examples.dat**

**Dołączanie przykładów  
pozytywnych do negatywnych**

**Mieszanie zrównoważonego  
zbioru treningowego**

W tym listingu używane są narzędzia z wiersza poleceń do redukcji i mieszania danych. W efekcie uzyskiwany jest zrównoważony zbiór danych. Tak jak na listingu 5.1, również i tu używane jest narzędzie grep do wyboru z pliku treningowego potrzebnych

<sup>13</sup> Leon Bottou, *Online Learning and Stochastic Approximations*, w: „*Online Learning and Neural Networks*”, David Saad (Cambridge University Press, 1998).

<sup>14</sup> Leon Bottou, *Stochastic Gradient Tricks*, w: „*Neural Networks: Tricks of the Trade*”, red. Gregoire Montavon, (Springer-Verlag, 2012): 421 – 36.

wierszy (przykładów pozytywnych lub negatywnych), po czym dane są mieszane za pomocą instrukcji `sort` i zapisywane w pliku wyjściowym. Za pomocą narzędzia `awk` kod pobiera co trzeci przykład negatywny, po czym przenosi dane między plikami. Listing 5.2 jest opatrzony wieloma opisami dla osób zainteresowanych szczegółami. W wyniku uruchomienia tego fragmentu kodu powstaje zrównoważony i wymieszany zbiór treningowy w pliku `all_examples_shuffled.dat`. Ten zbiór treningowy jest później redukowany do jednej dziesiątej pierwotnej wielkości, co pozwala uzyskać łatwiejszy do przetwarzania zbiór danych w pliku `all_examples_shuffled_down.dat`. Oba te zbioru danych są zrównoważone, przy czym jeden z nich (mniejszy) jest podziobrem drugiego. W dalszej części rozdziału używamy zredukowanego zbioru danych, ponieważ pozwala szybciej uzyskać model po treningu. Warto jednak zauważyć, że ponieważ VW działa z wykorzystaniem pamięci zewnętrznej, trening z użyciem dowolnego z obu zbiorów nie powinien skutkować problemami. Jedyne różnice dotyczą czasu potrzebnego na ukończenie treningu i wielkości potrzebnej tablicy cech (co omawiamy dalej). Przeprowadzenie treningu i testów z użyciem większego zbioru danych pozostawiam Czytelnikom.

Gdy dane są już wymieszane i gotowe do zastosowania w ramach treningu, można wywołać bibliotekę VW. Listing 5.3 przedstawia kod odpowiedzialny za trening i przedstawienie modelu w formie czytelnej dla ludzi.

#### Listing 5.3. Trening modelu opartego na regresji logistycznej z użyciem biblioteki VW

```
vw all_examples_shuffled_down.dat \
  --loss_function=logistic \
  -c \
  -b 22 \
  -passes=3 \
  -f model.vw
```

**Trening modelu opartego na regresji logistycznej z użyciem pamięci podręcznej (-c), tablicy cech o wielkości 22 (-b 22)<sup>15</sup> i trzech przebiegów po zbiorze danych. Model jest zapisywany w pliku model.vw.**

```
vw all_examples_shuffled_down.dat \
  -t \
  -i model.vw \
  --invert_hash readable.model
```

**Uruchomienie narzędzia vw w trybie testowym (-t) dla tych samych danych z użyciem wytrewnowanego modelu. Celem nie jest przeprowadzenie testu na tych samych danych, ale jednokrotne przetworzenie danych i uzyskanie pierwotnych wartości na podstawie skrótów.**

```
cat readable.model | awk 'NR > 9 {print}' | \
  sort -r -g -k 3 -t : | head -1000 > readable_model_sorted_top
```

**Sortowanie czytelnego modelu według ustalonego w wyniku uczenia parametru i zachowywanie 1000 pierwszych elementów.**

<sup>15</sup>Ta wartość została wybrana, ponieważ z analiz używanego zbioru danych wynika, że liczba cech wynosi ponad 2,1 miliona. Aby móc przedstawić je w unikatowy sposób, potrzebny jest binarny wektor cech o wielkości przynajmniej  $\log_2(2,1 \times 10^6)$ , co po zaokrągleniu w góre do najbliższej liczby całkowitej daje 22. Zauważ, że dla pełnego (niezredukowanego) zbioru danych ta wartość zapewne będzie inna. Możesz łatwo ustalić liczbę cech dla pełnego zbioru danych, uruchamiając kod z listingu 5.3 dla takiego zbioru i zliczając wiersze z pliku `readable.model`.

Pierwszy wiersz uruchamia trening modelu za pomocą narzędzia VW. Kilka opcji z wiersza poleceń uruchamia określone funkcje. Więcej informacji na ich temat znajdziesz we wspomnianym już artykule Langforda *Vowpal Wabbit command-line arguments*. Aby zastosować regresję logistyczną, należy nakazać narzędziu VW użycie logistycznej funkcji straty w celu obliczenia różnicy między oczekiwany danymi wyjściowymi a predykcjami. Dzięki temu na podstawie metody SGD można zmodyfikować współczynniki i uzyskać konwergencję, gdy liniowa kombinacja współczynników i cech zbliży się do logarytmu szans uzyskania docelowej zmiennej. Opcja `-c` informuje narzędzie VW, że ma zastosować pamięć podręczną. Wtedy dane wejściowe są zapisywane w natywnym formacie VW, a kolejne przebiegi przez dane są znacznie szybsze. Opcja `-b` powoduje użycie 22 bitów na wektor cech, co jest odpowiednią wielkością dla używanego zbioru danych. Opcja `-f` powoduje tu zapisanie wynikowego modelu (a dokładniej jego współczynników) w pliku *model.vw*.

Podstawy treningu modelu są proste do zrozumienia, co pokazaliśmy w poprzednim rozdziale. Należy aktualizować współczynniki w celu zminimalizowania błędów w prognozach (wartości funkcji straty). Aby ten proces był wydajny, VW zapisuje cechy za pomocą skrótów. Dlatego każda cecha  $f$  jest reprezentowana za pomocą powiązanej maski bitowej określonej za pomocą skrótu  $h(f)$ . Dzięki temu można szybko znaleźć cechę w celu zmiany jej współczynników w każdym przebiegu procesu uczenia. Ma to dodatkową zaletę, ponieważ można zastosować sztuczkę z haszowaniem<sup>16</sup>, pozwalającą zapisać bardzo dużą przestrzeń cech za pomocą reprezentacji o stałej wielkości. Prrowadzi to do ograniczenia kosztów uczenia się bez nadmiernego wpływu na moc gnostyczną.

Plik *model.vw* zawiera wektor wag ustalonych w wyniku nauki. Niestety, dla człowieka te wartości nie są zrozumiałe, ponieważ nie wiadomo, w jaki sposób czytelne dla człowieka cechy są powiązane z wektorem. Dla praktyków z obszaru uczenia maszynowego czytelne cechy są ważne, ponieważ pozwalają zobaczyć, które cechy są istotne w modelu. Z tym związany jest drugi wiersz listingu 5.3. Narzędzie vw jest uruchamiane ponownie, tym razem w trybie testowym (`-t`) z użyciem wytrenowanego modelu. Zwykle nie należy przeprowadzać testów z użyciem danych z etapu treningu, jednak tu wyniki testu nie są istotne. Dzięki przetworzeniu danych treningowych w trybie testowym z ustawioną opcją `-invert_hash` tworzona jest tabela łącząca czytelne nazwy cech z ich znaczeniem dla klasyfikatora określonym na podstawie wag powiązanych ze skrótami cech.

Ponieważ liczba wartości ustalonych w procesie uczenia jest tak duża i są one zapisywane w pliku *readable.model* bez określonej kolejności, przed analizą współczynników plik ten należy posortować. Używane są do tego narzędzia z wiersza poleceń `awk`, `sort` i `head`. Kod pomija pierwszych dziewięć wierszy (obejmują one nagłówki i informacje wstępne), a następnie sortuje pozostałe wiersze według trzeciej kolumny (instrukcja `sort` jest informowana, że kolumny są oznaczone za pomocą dwukropka `:`). Ostateczne dane wyjściowe są kierowane do pliku *readable\_model\_sorted\_top*. Na lis-

<sup>16</sup> Olivier Chapelle, Eren Manavoglu i Romer Rosales, *Simple and Scalable Response Prediction for Display Advertising*, „ACM Transactions on Intelligent Systems and Technology” 5, nr 4 (2015): 61.

tingu 5.4 znajdziesz 20 pierwszych współczynników. Wymienione są tu cechy, których obecność (przy założeniu stałości pozostałych cech) powoduje największy wzrost logarytmu szans.

**Listing 5.4. Czytelne dla człowieka dane wyjściowe z narzędzia VW**

c24^e5c2573e:395946:3.420166	❶	<b>Najważniejsze cechy</b>
c3^6dd570f4:3211833:1.561701		
c12^d29f3d52:2216996:1.523759		
c12^977d7916:2216996:1.523759		
c13^61537f27:2291896:1.373842		
c12^d27ed0ed:2291896:1.373842		
c12^b4ae013a:2291896:1.373842		
c16^9c91bbc5:1155379:1.297896		
c3^f25ae70a:64455:1.258160		
c26^ee268cbb:64455:1.258160		
c21^cfa6d1ae:64455:1.258160		
c26^06016427:3795516:1.243104		
c15^2d65361c:506605:1.240656		
c12^13972841:506605:1.240656		
c3^e2f2a6c7:3316053:1.234188		
c18^eafb2187:3316053:1.234188		
c18^fe74f288:3396459:1.218989		
c3^8d935e77:2404744:1.214586		
c4^f9a7e394:966724:1.212140		
c3^811ddf6f:966724:1.212140		

Na listingu 5.4 pokazany jest fragment danych wyjściowych czytelnych dla ludzi. Dane po znaku ^ to wartości cech w nieprzetworzonych danych. Kolumny druga i trzecia zawierają skrót i współczynnik. Tu przedstawiamy tylko 20 pierwszych współczynników (w zredukowanym zbiorze ich liczba to ponad dwa miliony; prawdopodobnie wynika to z tego, że w zbiorze danych występują domeny i każda z nich jest kodowana jako odrębna cecha), jednak widoczne są w nich ciekawe trendy. Wygląda na to, że najważniejszą cechą jest c24, następnie c3 i c12 ❶. Obecność każdej z tych trzech cech powoduje duży wzrost logarytmu szans — przy założeniu, że pozostałe cechy są stałe. Zauważ, że nie występuje tu żadna cecha o semantyce całkowitoliczbowej.

Możliwe, że zauważysz powtarzające się wartości wag niektórych cech z tej listy. Dzieje się tak, ponieważ skróty tych cech są powiązane z tymi samymi bitami wektora cech (następuje kolizja skrótów). Wskazuje na to fakt, że liczba po pierwszym przecinku jest taka sama dla kilku pozycji z listingu 5.4. Wpływ tych cech jest ujednolicany do tej samej wartości, gdy wyznaczany jest iloczyn skalarny wektora cech i ustalonych wag.

Nie jest to idealne rozwiązanie, jednak nie musi dawać złych efektów. Wpływ cech jest tu uśredniany przez mechanizm uczenia się. Jeśli któraś z takich ujednolicionych cech jest obecna, łączny pozytywny lub negatywny wpływ jest dodawany do logarytmu prawdopodobieństwa wystąpienia prognozowanego zdarzenia<sup>17</sup>.

<sup>17</sup>Więcej informacji o generowaniu skrótów cech i unikaniu kolizji znajdziesz na stronie <http://mng.bz/WQi7>.

Niestety, ponieważ firma Criteo udostępnia dane po anonimizacji, nie da się ustalić, co oznaczają poszczególne cechy. Gdyby jednak było to możliwe, byłoby to bardzo przydatne w kontekście analiz i pozwoliłoby dobrze opisać model! Jednak tak naprawdę interesuje nas ustalenie, jak skuteczny jest model. Na szczęście można to zrobić bez określania znaczenia cech. We wcześniejszym rozdziale wspomnieliśmy o mierze powierzchni pod krzywą ROC. Jeśli jej wartość jest bliska 1, klasyfikator jest niemal doskonały. Zastosujemy teraz tę technikę do zbioru testowego z niewidzianymi wcześniej elementami. Najpierw jednak trzeba posłużyć się narzędziem VW i wcześniej wytrenowanym modelem do ustalenia predykcji na podstawie danych testowych.

### 5.5.3. Testowanie modelu

Na listingu 5.5 przedstawiony jest kod analizujący zbiór testowy za pomocą modelu po treningu. Kod generuje prawdopodobieństwa i ustala rzeczywiste wyniki, co pozwoli w kolejnym kroku wyznaczyć krzywą ROC.

#### Listing 5.5. Testowanie modelu po treningu za pomocą narzędzia VW

```
vw -d test_vw_file \
-t \
-i model.vw \
--loss_function=logistic \
-r predictions.out
~/dev/vowpal_wabbit/utl/logistic -0 predictions.out > \
probabilities.out | cut -d ' ' -f 1 test_vw_file | \
sed -e 's/^1/0/' > ground_truth.dat
```

W pierwszym poleceniu używane są opcje `-d` (do wskazania pliku z danymi) i `-t` (oznacza testy bez uczenia). Opcja `-i` pozwala wczytać wcześniej wytrenowany model do pamięci, a opcja `-r` służy do zwracania nieprzetworzonych predykcji. W kolejnym wierszu używana jest funkcja pomocnicza przekształcająca nieprzetworzone predykcje (o postaci  $\beta_0 + \sum_i \beta_i x_i$ ) na prawdopodobieństwo określone w punkcie 5.4.5 jako  $y$ .

Zauważ, że w swoim systemie prawdopodobnie będziesz musiał zmienić ścieżkę do funkcji pomocniczej. Skrypt *logistic* znajdziesz w podkatalogu *utl* w repozytorium narzędzia VW (<http://mng.bz/pu5U>). Komunikat ten jest pobierany zgodnie z instrukcjami z pliku *requirements.txt* z archiwum z kodem do książki.

Wynikowe prawdopodobieństwo jest zapisywane w pliku *probabilities.out*. Ostatni wiersz listingu 5.5 rozdziela kolumny pliku treningowego, używając jako separatora spacji, i pobiera pierwszą kolumnę, zawierającą rzeczywiste zdarzenia. Te dane są przekazywane do instrukcji *sed*, która zastępuje wystąpienia wartości `-1` (tak VW zapisuje negatywne zdarzenia w regresji logistycznej) wartościami `0`. Dane wyjściowe są zapisywane w pliku *ground\_truth.dat*.

Teraz możemy ocenić skuteczność modelu na podstawie danych testowych. Na listingu 5.6 pokazany jest kod wykorzystujący pakiet scikit-learn. Pakiet ten jest potrzebny do obliczenia powierzchni pod krzywą ROC i wykresu z tą krzywą (zobacz rysunek 5.5).

**Listing 5.6. Generowanie krzywej ROC za pomocą pakietu scikit-learn**

```

import numpy as np
import pylab as pl
from sklearn import svm, datasets
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc

ground_truth_file_name = './ground_truth.dat'
probability_file_name = './probabilities.out'

ground_truth_file = open(ground_truth_file_name, 'r')
probability_file = open(probability_file_name, 'r')

ground_truth = np.array(map(int, ground_truth_file))
probabilities = np.array(map(float, probability_file))

ground_truth_file.close()
probability_file.close()

# Źródło: http://scikitlearn.org/stable/  
auto\_examples/model\_selection/plot\_roc.html
fpr, tpr, thresholds = roc_curve(ground_truth, probabilities)
roc_auc = auc(fpr, tpr)
print "Obszar pod krzywą ROC: %f" % roc_auc

pl.clf()
pl.plot(fpr, tpr, label='Krzywa ROC (obszar = %0.2f)' % roc_auc)
pl.plot([0, 1], [0, 1], 'k--')
pl.xlim([0.0, 1.0])
pl.ylim([0.0, 1.0])
pl.xlabel('Współczynnik fałszywych wskazań pozytywnych')
pl.ylabel('Współczynnik trafnych wskazań pozytywnych')
pl.title('ROC')
pl.legend(loc="lower right")
pl.show()

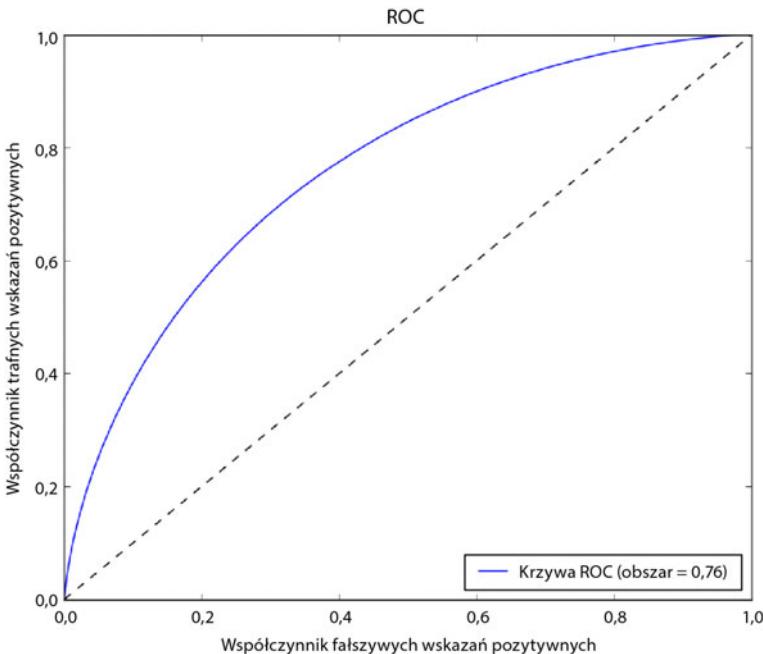
```

**Używanie pakietu scikit-learn do obliczenia współczynników fałszywych wskazań pozytywnych i trafnych wskazań pozytywnych (i powiązanych wartości progowych)**

Po instrukcjach importu następuje 11 wierszy z operacjami wstępymi — wczytywaniem rzeczywistych wyników i prawdopodobieństw potrzebnych do wygenerowania krzywej ROC. Dane związane z krzywą ROC są pobierane w jednym wierszu. Reszta kodu generuje wykres i pochodzi z dokumentacji pakietu scikit-learn<sup>18</sup>. Dane wyjściowe z tego kodu pokazane są na rysunku 5.5.

Wynikowy obszar pod krzywą to 0,76, uzyskany dla modelu po treningu i nieużywanych wcześniej danych testowych. Warto przekształcić to na łatwiejszą do intuicyjnego zrozumienia wartość. Jeśli ustalisz próg odpowiadający punktowi na krzywej ROC najbliższemu lewemu górnemu rogowi rysunku 5.5 i posłużysz się nim do odróżniania warunków prognozujących kliknięcie od warunków sugerujących brak kliknięcia, otrzymasz współczynnik trafnych wskazań pozytywnych na poziomie około 0,7 (czyli gdy model prognozuje kliknięcie, ma rację w 70% sytuacji) i współczynnik fałszywych wskazań pozytywnych równy około 0,3 (czyli gdy model prognozuje kliknięcie, myli się w 30% przypadków). Aby uzyskać te wartości, znajdź punkt najbliższego lewemu górnemu

<sup>18</sup> scikit-learn, *Receiver Operating Characteristic (ROC)*, <http://mng.bz/9jBY>.



Rysunek 5.5. Krzywa ROC dla modelu uzyskana na podstawie danych testowych.  
Obszar pod krzywą wynosi 0,76

narożnikowi na górnjej krzywej z rysunku i odczytaj wartości na osiach x i y. Jednak w praktyce modele tego rodzaju służą nie tyle do klasyfikowania emisji, które mogą zakończyć się kliknięciem, co do obliczania, ile warto zapłacić za wyświetlenie reklamy (zgodnie z procesem opisanym w punkcie 5.4.6).

#### **5.5.4. Kalibrowanie modelu**

W poprzednim punkcie wspomnieliśmy, że dane wyjściowe z wytrenowanego modelu nie są używane do utworzenia samego klasyfikatora. Dane wyjściowe z modelu opartego na regresji logistycznej służą do szacowania prawdopodobieństwa kliknięcia. Jednak aby umożliwić skuteczny trening modelu, zmieniliśmy częstotliwość pozytywnych zdarzeń w danych, tak aby liczba zdarzeń pozytywnych i negatywnych w danych treningowych była podobna. Dlatego aby uzyskać za pomocą modelu dane wyjściowe reprezentujące rzeczywiste prawdopodobieństwo kliknięcia, należy przeprowadzić *kalibrację*.

Używana tu metoda pochodzi z pracy Oliviera Chapelle'a i współpracowników<sup>19</sup>. W tej technice wykorzystano proste spostrzeżenie. Przypomnij sobie, że w utworzonym modelu logarytm szansy wystąpienia pozytywnego zdarzenia jest limiowo powiązany z danymi wejściowymi za pomocą wag ustalonych w procesie uczenia:

$$\frac{\Pr(y=1|\mathbf{x}, \boldsymbol{\beta})}{\Pr(y=-1|\mathbf{x}, \boldsymbol{\beta})} = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

<sup>19</sup>Chapelle i współpracownicy, *Simple and Scalable Response Prediction for Display Advertising*, „ACM Transactions on Intelligent Systems and Technology” 5, nr 4 (styczeń 2014): 1 – 34.

Prawdopodobieństwo występuje w liczniku i mianowniku tego równania. Uwzględnij teraz tylko rozkład danych. Jeśli zastosujesz regułę Bayesa do góry i dolu wzoru na szanse i przeprowadzisz uproszczenie, uzyskasz następującą relację:

$$\frac{\Pr(y = 1|x)}{\Pr(y = -1|x)} = \frac{\Pr(x|y = 1)\Pr(y = 1)}{\Pr(x|y = -1)\Pr(y = -1)}$$

Wykorzystując fakt, że zredukowane zostały tylko negatywne przypadki, oraz (tak jak w pracy Chapelle'a) używając  $Pr'$  do oznaczenia wynikowego rozkładu prawdopodobieństwa po redukcji danych, można zapisać tę zależność w następującej postaci:

$$\frac{\Pr(y = 1|x)}{\Pr(y = -1|x)} = \frac{Pr'(x|y = 1)\Pr'(y = 1)}{Pr'(x|y = -1)\Pr'(y = -1)/r}$$

Tu  $r$  oznacza stopień redukcji negatywnych przypadków<sup>20</sup>. To prowadzi do następującej relacji między szansami z pierwotnego zbioru danych i zredukowanego zbioru:

$$\frac{\Pr(y = 1|x)}{\Pr(y = -1|x)} = r \frac{\Pr'(y = 1|x)}{\Pr'(y = -1|x)}$$

Wróćmy teraz do pierwotnego równania modelu:

$$\ln\left(r \frac{Pr'(y = 1|x, \beta)}{Pr'(y = -1|x, \beta)}\right) = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

Oto równoznaczny zapis:

$$\ln\left(\frac{Pr'(y = 1|x, \beta)}{Pr'(y = -1|x, \beta)}\right) = (\beta_0 - \ln(r)) + \beta_1 x_1 + \dots + \beta_n x_n$$

Możesz przeprowadzić trening z użyciem zredukowanego zbioru danych, usuwając tylko negatywne przypadki. Jeśli jednak chcesz uzyskać na podstawie modelu dane wyjściowe precyzyjnie odzwierciedlające prawdopodobieństwo zdarzenia pozytywnego, musisz wykonać dodatkowy krok.

Widać tu, że dla modelu po treningu z użyciem zredukowanego zbioru punkt przecięcia osi jest  $\ln(r)$  niższy niż dla kompletnego zbioru. Jeśli więc chcesz uzyskać poprawny wynik, musisz dodać  $\ln(r)$  do otrzymanej wartości. Nie zapominaj jednak, że choć sami też zredukowaliśmy dane, wcześniej firma Criteo zredukowała je o nieznaną wartość! Dlatego nie da się skalibrować tego modelu bez dodatkowych informacji.

W tym podrozdziale omówiliśmy dużo materiału i opisaliśmy podstawy prognozowania kliknięć w reklamy w internecie. Zauważ, że to zadanie nie ogranicza się do opracowania dobrego modelu. Zwykle trzeba uwzględnić sprzeczne oczekiwania wielu reklamodawców, stosowane przez nich miary sukcesu i oczekiwane marże. Zbudowanie skutecznego modelu to dopiero początek. W wielu firmach z branży technologii reklamy całe zespoły badaczy z dziedziny nauk o danych pracują nad tym problemem. W następnym podrozdziale poznasz kilka komplikacji, na jakie możesz natrafić, gdy stosujesz modele takie jak pokazany.

---

<sup>20</sup>Na przykład jeśli w pierwotnym zbiorze danych występuje 1000 pozytywnych przypadków i 9000 negatywnych, a w celu zrównoważenia zbioru uwzględnianych jest 1000 negatywnych przypadków, stosowane  $r = 1/9$ .

## **5.6. Komplikacje związane z budowaniem systemu podejmowania decyzji**

Stare przysłowie mówi, że diabeł tkwi w szczegółach. Aby zbudować skuteczny system prognozowania kliknięć, trzeba uwzględnić różne problemy i odpowiedzieć na szereg pytań. W tym podrozdziale stawiamy kilka otwartych pytań, nad którymi powinieneś się zastanowić. W czasie gdy powstaje ta książka, społeczność zainteresowana uczeniem maszynowym w kontekście technologii reklamy wciąż nie ustaliła jeszcze odpowiedzi na liczne z takich pytań. W dalszych akapitach często omawiamy zagadnienia z perspektywy branży reklamowej, jednak przedstawiane problemy są istotne także w innych dziedzinach.

Najpierw zajmijmy się kwestią częstotliwości treningów i przerw między nimi. Jak często należy przeprowadzać trening modelu? Jak długo model pozostaje wiarygodny? Możliwe, że chcesz zbudować model w dniu  $n$ , a następnie zastosować go w dniu  $n+1$ . Co jednak zrobić, jeśli dzisiejszy model nie jest reprezentatywny dla dnia jutrajszego? Możesz wytrenować model na potrzeby całego tygodnia lub zbudować modele dla poszczególnych dni tygodnia, które mogą zapewniać lepsze wyniki, choć nie jest to pewne. Potrzebny jest sposób na śledzenie skuteczności modelu w czasie.

To prowadzi do pytania o *dryfowanie*. Jeśli zbudujesz model w dniu  $n$  i zastosujesz go do kolejnych dni,  $n+1$ ,  $n+2$  itd., zapewne zauważysz, że model dryfuje. Oznacza to, że z czasem skuteczność modelu spadnie. Dzieje się tak, ponieważ zachowania użytkowników się zmieniają. W świecie reklamy wydawcy zmieniają zawartość serwisów, niektóre witryny znikają z rynku, a ważne są nawet zmiany pogody! Dlatego jeśli nie stosujesz uczenia w czasie rzeczywistym i nie aktualizujesz modelu na podstawie każdego napotkanego punktu danych, model staje się nieaktualny już w momencie zakończenia treningu. A im dłużej będzie używany, tym bardziej stanie się nieaktualny.

Dochodzimy w ten sposób do pytania o potok kroków związanych z treningiem. Pamiętaj, że do systemu nieustannie napływają dane dotyczące milionów interakcji w sieci WWW. Wymaga to ciągłego rejestrowania, przekierowywania, przechowywania, wstępного przetwarzania i podstawowego przetwarzania tych informacji i generowania treningowych zbiorów danych w każdym momencie, w każdej godzinie, w każdym dniu i tygodniu. Jest to złożone zadanie wymagające solidnej pracy inżynierowej. Po zapoznaniu się z dodatkiem z tej książki powinieneś dostrzec, że zbieranie i przygotowywanie danych to pierwszy krok w *potoku* kroków wykonywanych w ramach większej intelligentnej aplikacji. Ten potok może obejmować etapy pobierania, oczyszczania i redukowania danych, treningu oraz udostępniania modelu. Każdy z tych etapów jest uruchamiany po zakończeniu poprzedniego i działa dla danych z określonego okresu (na przykład z godziny).

## **5.7. Przyszłość prognozowania zdarzeń w czasie rzeczywistym**

W tym rozdziale przedstawiliśmy praktyczny problem, który wymaga podejmowania decyzji w czasie rzeczywistym na podstawie bardzo dużej ilości danych. Prognozowanie kliknięć to problem specyficzny dla świata reklamy, ale wiele rozwiązań z tego

obszaru można łatwo zastosować także w innych dziedzinach. Wierzę, że w przyszłości proces prognozowania w czasie rzeczywistym zostanie usprawniony w dwóch ważnych aspektach: dzięki pobieraniu i przetwarzaniu danych w czasie rzeczywistym oraz dzięki modelem adaptacyjnym i reaktywnym.

Jeśli chodzi o pierwszy z tych aspektów, to już omówiliśmy znaczenie potoków danych, służących do pobierania, kształtowania i przetwarzania danych na potrzeby treningu i stosowania modelu. Do dziś w wielu systemach porcje danych są przesyłane z jednego etapu do następnego. Wynika to z tego, że większość systemów przetwarzania danych była budowana w ten sposób. Dopiero niedawno zaczęły się pojawiać systemy przetwarzania strumieniowego takie jak Storm, Storm Trident i Spark Streaming, a także ekskcytujące rozwiązania MillWheel<sup>21</sup> i FlumeJava<sup>22</sup> z firmy Google. Takie systemy mogą zmienić świat prognoz, skracając czas od zarejestrowania zdarzenia do treningu modelu z uwzględnieniem tego zdarzenia. Niskie opóźnienie jest możliwe, ponieważ każdy punkt danych może przepływać przez system bez konieczności oczekiwania na utworzenie zawierającej go kompletnej porcji danych.

To prowadzi do drugiego ważnego aspektu — modeli adaptacyjnych i reaktywnych. Dzięki zmniejszeniu opóźnienia w przepływie danych przez system dane można znacznie szybciej przekazywać do etapu treningu modelu. Jeśli uda się skrócić opóźnienie między zarejestrowaniem tego punktu danych a uzyskaniem ujmującego go modelu, możliwe będzie szybsze uwzględnianie zdarzeń. Wtedy możliwe stanie się przejście od treningu na podstawie porcji danych do prawdziwego uczenia się na żywo. Modele nie będą już tylko aktualizowane sekwencyjnie z użyciem pamięci zewnętrznej; możliwe stanie się aktualizowanie sekwencyjne modelu *wraz z napływem danych*.

## 5.8. Podsumowanie

- Zbudowaliśmy inteligentny algorytm do prognozowania kliknięć w internecie.
- Przedstawiliśmy完备的 wprowadzenie do bardzo ważnego zastosowania intelligentnej sieci WWW. Bardzo nieliczne aplikacje zyskały (i utrzymują) takie zainteresowanie w społeczności skupionej na uczeniu maszynowym.
- Pokazaliśmy, jak stosować narzędzie Vowpal Wabbit — bibliotekę do uczenia maszynowego z obsługą pamięci zewnętrznej. Umożliwia ona trening bez wczytywania wszystkich danych do pamięci.
- Omówiliśmy przyszłość prognozowania kliknięć w czasie rzeczywistym w sieci WWW.

---

<sup>21</sup>Tyler Akidau i współpracownicy, *MillWheel: Fault-Tolerant Stream Processing at Internet Scale*, „The 39th International Conference on Very Large Data Bases” (VLDB, 2013): 734 – 46.

<sup>22</sup>Craig Chambers i współpracownicy, *FlumeJava: Easy, Efficient Data-Parallel Pipelines*, „ACM SIGPLAN Conference on Programming Language Design and Implementation” (ACM, 2010): 363 – 75.



# *Uczenie głębokie i sieci neuronowe*

---

## **Zawartość rozdziału:**

- Podstawy sieci neuronowych
- Wprowadzenie do uczenia głębokiego
- Rozpoznawanie cyfr z wykorzystaniem ograniczonych maszyn Boltzmanna

Obecnie dużo się mówi o uczeniu głębokim. Wiele osób uważa, że będzie to następny duży krok naprzód w dziedzinach uczenia maszynowego i sztucznej inteligencji. W tym rozdziale staramy się ograniczyć retoryczne zabiegi i przedstawić konkretne fakty. Do czasu zakończenia lektury tego rozdziału poznasz *perceptrony* (podstawowe cegiелki sieci wykorzystujących uczenie głębokie) i zrozumiesz, jak łączyć je w sieci głębokie. Sieci neuronowe były zapowiedzią pojawienia się perceptronów, dlatego omawiamy je przed przejściem do głębszych sieci dających większe możliwości. Z tymi głębszymi sieciami związane są istotne wyzwania dotyczące treningu i reprezentacji danych, dlatego przed przejściem do tych zagadnień powinieneś zapewnić sobie solidne podstawy.

Jednak na samym początku omawiamy naturę uczenia głębokiego i rodzaje problemów, do jakich stosowane było to rozwiązanie. Wyjaśniamy też, co jest przyczyną jego sukcesów. Dzięki temu powinieneś zrozumieć podstawowe motywacje do stosowania uczenia głębokiego i zyskać ramy, w których będziesz mógł osadzić bardziej skomplikowane zagadnienia teoretyczne omawiane w dalszej części rozdziału. Pamiętaj, że jest to dziedzina, w której wciąż prowadzonych jest wiele badań. Dlatego zachęcamy,

abyś śledził najnowsze dokonania w literaturze przedmiotu. Startup.ML<sup>1</sup> i KD Nuggets<sup>2</sup> to tylko wybrane źródła, w których znajdziesz przegląd najnowszych osiągnięć z omawianej dziedziny. Nakłaniamy jednak do samodzielnego eksperymentów i wyciągania własnych wniosków!

## 6.1. Intuicyjne omówienie uczenia głębokiego

Aby zrozumieć uczenie głębokie, zastanów się nad aplikacją do rozpoznawania obrazów. Jak zbudować klasyfikator, który na podstawie rysunku lub filmu będzie potrafił rozpoznawać obiekty? Taka aplikacja może mieć wiele zastosowań. Po pojawienniu się nurtu *quantified self*<sup>3,4</sup> i urządzenia Google Glass można wyobrazić sobie aplikacje, które rozpoznają obiekty widoczne dla użytkownika i prezentują w okularach wzbogaconą wizję rzeczywistości.

Pomyśl na przykład o rozpoznawaniu samochodu. W uczeniu głębokim budowane są warstwy interpretacyjne, z których każda wykorzystuje dane z poprzedniej warstwy. Na rysunku 6.1 pokazane są możliwe warstwy interpretacyjne sieci głębokiej wytrenowanej pod kątem rozpoznawania samochodów. Zarówno ten przykład, jak i niektóre z dalszych rysunków pochodzą z poświęconego omawianym zagadnieniom wykładu Andrew Nga<sup>5</sup>.

W dolnej części rysunku 6.1 widocznych jest kilka zdjęć samochodów. Stanowią one zbiór danych treningowych. Jak wykorzystać uczenie głębokie, aby aplikacja bez ręcznych opisów zawartości tych zdjęć wykryła podobieństwa między nimi (czyli fakt, że na każdym z nich znajduje się samochód)? Algorytm nie otrzymuje informacji, że na zdjęciu widoczny jest samochód.

Widać tu, że w uczeniu głębokim wykorzystywane są stopniowo abstrakcje wyższego poziomu oparte na abstrakcjach niższego poziomu. W problemie rozpoznawania obrazów punktem wyjścia jest najmniejsza porcja informacji na rysunku: piksel. Do zbudowania podstawowych cech wykorzystywany jest cały zbiór obrazów (przypomnij sobie rozdział 2., w którym omawialiśmy wydobycie struktury z danych). Grupy tych cech są łączone w celu wykrywania elementów z wyższego poziomu abstrakcji, na przykład linii i krzywych. Na kolejnym poziomie linie i krzywe są łączone w części samochodu występujące w zbiorze treningowym. W następnym kroku części są grupowane w celu uzyskania detektorów obiektów w postaci całych samochodów.

Warto zwrócić tu uwagę na dwie ważne kwestie. Po pierwsze, nie jest stosowana jawną inżynierią cech. W poprzednim rozdziale wspomnieliśmy o znaczeniu przygotowania dobrej reprezentacji danych. To zagadnienie omawialiśmy w kontekście prognozowania kliknięć reklam. Ekspertki z tej dziedziny zwykle samodzielnie przygotowują

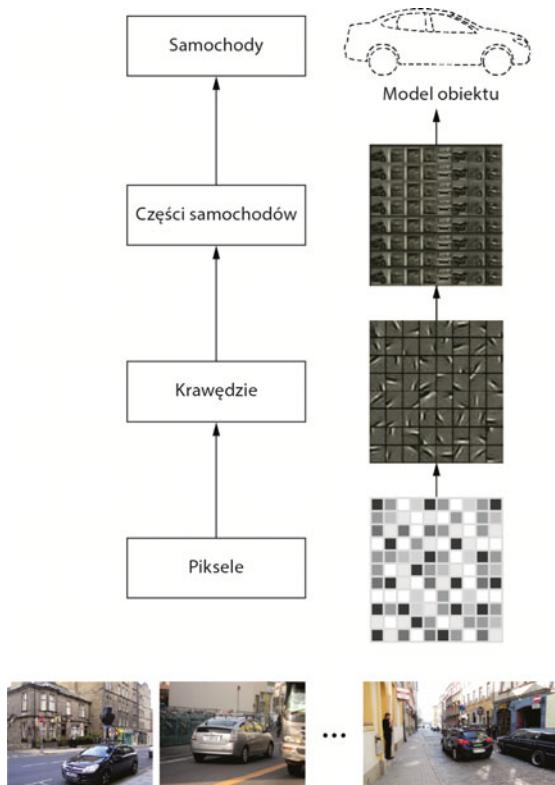
<sup>1</sup> Startup.ML, „Deep Learning News”, 30 czerwca 2015, <http://news.startup.ml>.

<sup>2</sup> KD Nuggets, „Deep Learning”, <http://www.kdnuggets.com/tag/deep-learning>.

<sup>3</sup> Gina Neff i Dawn Nafus, *The Quantified Self* (MIT Press, 2016).

<sup>4</sup> Deborah Lupton, *The Quantified Self* (Polity Press, 2016).

<sup>5</sup> Andrew Ng, *Bay Area Vision Meeting: Unsupervised Feature Learning and Deep Learning*, YouTube, 7 marca 2011, <http://mng.bz/2cR9>.



**Rysunek 6.1.** Wizualizacja sieci głębokiej do rozpoznawania samochodów. Niektóre materiały graficzne pochodzą ze wspomnianego wcześniej wykładu Andrew Nga na omawiane tematy. Podstawowy treningowy zestaw zdjęć jest używany do opanowania wykrywania krawędzi. Krawędzie można łączyć w celu wykrywania części samochodu, które z kolei są łączone w celu wykrywania typu obiektu. Tu tym obiektem jest samochód

dane. Jednak teraz wykonywane jest nienadzorowane uczenie cech. Oznacza to, że system uczy się reprezentacji danych bez jawnej interakcji z użytkownikiem. Można uznać to za analogię do rozpoznawania obiektów przez ludzi. Ludzie naprawdę dobrze radzą sobie z wykrywaniem wzorców!

Drugą wartą uwagi kwestią jest to, że wykrywane obiekty nie zostały jawnie określone jako samochody. Jeśli wejściowy zbiór obrazów jest wystarczająco zróżnicowany, detektory z najwyższego poziomu powinny sprawdzać się dla dowolnych zaprezentowanych samochodów. Nie uprzedzajmy jednak faktów; najpierw należy wyjaśnić podstawy związane z sieciami neuronowymi.

## 6.2. Sieci neuronowe

Sieci neuronowe nie są nową technologią. Pojawiły się już w latach 40. ubiegłego wieku. Ta technologia jest inspirowana biologią. Neuron wyjściowy jest aktywowany na podstawie danych wejściowych z kilku powiązanych neuronów wejściowych. Sieci neuronowe nazywane są czasem *sztucznymi* sieciami neuronowymi, ponieważ w sztuczny sposób uzyskują efekt podobny jak w ludzkim mózgu. Jeubin Huang<sup>6</sup> przedstawia

<sup>6</sup> Jeubin Huang, *Overview of Cerebral Function*, „Merck Manual”, 1 września 2015, <http://mng.bz/128W>.

wprowadzenie do biologii ludzkiego mózgu. Choć wiele aspektów pracy naszych mózgów wciąż jest nieznanych, potrafimy zrozumieć jego podstawowe cegiełki. To jednak, w jaki sposób powstaje świadomość, nadal jest dla nas tajemnicą.

Neurony w mózgu za pomocą dendrytów rejestrują pozytywne (*wzbudzające*) i negatywne (*hamujące*) informacje wyjściowe z innych neuronów i kodują je za pomocą elektrycznych impulsów przesyłanych przez akson. Akson rozdziela się i dociera do setek lub tysięcy dendrytów innych neuronów. Między aksonem a wejściowymi dendrytami kolejnych neuronów znajduje się niewielka luka nazywana *synapsą*. Elektryczne impulsy są przekształcane na chemiczne sygnały wpływające na dendryt następnego neuronu. W tym scenariuszu wynik uczenia jest kodowany przez same neurony. Neurony przesyłają wiadomości aksonami tylko wtedy, gdy poziom pobudzenia jest wystarczająco wysoki.

Na rysunku 6.2 pokazane są opracowane przez McCullocha i Pittsa schematy ludzkiego neuronu i sztucznego neuronu (jest to *model MCP*<sup>7</sup>). Sztuczny neuron jest zbudowany z użyciem sumowania i wartości progowej. Logiczne dane wejściowe (tylko pozytywne) są otrzymywane za pośrednictwem odpowiedników dendrytów, po czym obliczana jest suma z uwzględnieniem wag. Jeśli dane wyjściowe przekroczą określony próg (ang. *threshold*), a wejście hamujące nie jest aktywne, neuron generuje wartość pozytywną. Jeżeli wejście hamujące jest aktywne, dane wyjściowe są hamowane. Dane wyjściowe mogą być przekazywane jako dane wejściowe dla kolejnych neuronów za pośrednictwem odpowiedników dendrytów. Gdy się nad zastanowisz, że (jeśli pominąć wejście hamujące) jest to model liniowy w przestrzeni  $n$ -wymiarowej ( $n$  to liczba wejść neuronu) z powiązanymi współczynnikami. Tu zakładamy, że wszystkie wejścia prowadzące do dendrytów pochodzą z różnych źródeł. Jednak teoretycznie to samo źródło można powiązać z neuronem wielokrotnie, jeśli ma być ono bardziej istotne od innych. Jest to odpowiednik zwiększenia wagi danego wejścia.

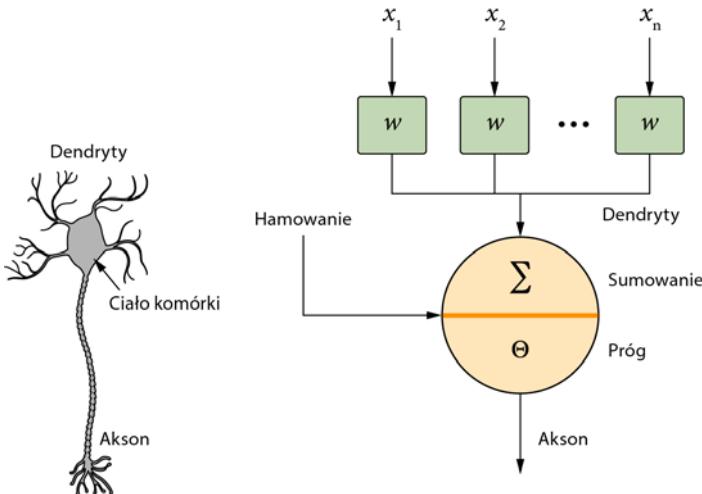
Na rysunku 6.3 pokazane jest działanie tego modelu dla  $n = 1$ . Na tym rysunku używany jest prosty ręcznie utworzony neuron z wagą jednostkową ( $w = 1$ ). Wejście neuronu przyjmuje wartości od  $-10$  do  $10$ , a sumowanie wartości wejściowych z uwzględnieniem wag odbywa się na osi  $y$ . Próg jest równy  $0$ , a neuron generuje sygnał tylko wtedy, gdy wartość wejściowa jest większa niż wartość progowa.

### 6.3. Perceptron

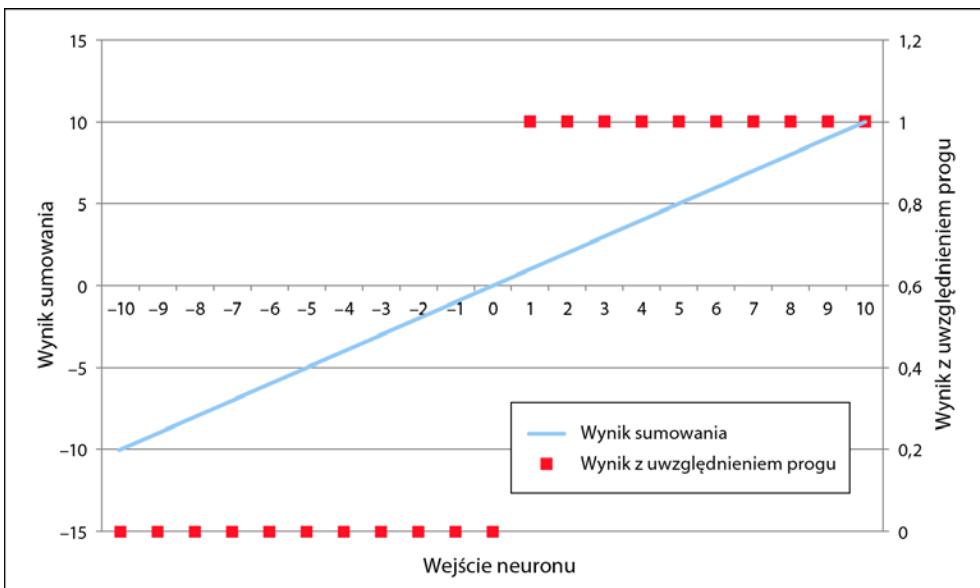
W poprzednim podrozdziale przedstawiliśmy neuron w modelu MCP. Ten podstawowy model umożliwia uczenie i generalizowanie danych treningowych, ale w bardzo ograniczonym zakresie. Można jednak uzyskać lepsze efekty. Służy do tego perceptron. Jest on zbudowany na podstawie modelu MCP i ma trzy ważne cechy<sup>8, 9</sup>:

<sup>7</sup> Warren S. McCulloch i Walter H. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, „Bulletin of Mathematical Biophysics” 5 (1943): 115 – 33.

<sup>8</sup> Frank Rosenblatt, *The Perceptron — A Perceiving and Recognizing Automaton* (Cornell Aeronautical Laboratory, 1957).



Rysunek 6.2. Po lewej stronie pokazany jest schemat biologicznego ludzkiego neuronu. Po prawej przedstawiona jest inspirowana ludzkim mózgiem sieć neuronowa zaimplementowana z użyciem sumowania z wagami, wejścia hamującego i progu



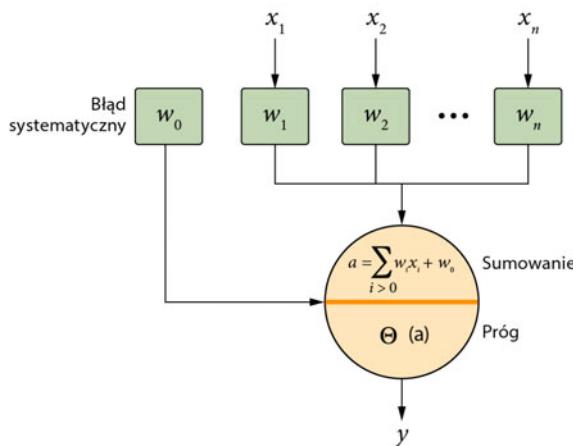
Rysunek 6.3. Model MCP jako dwuliniowy model bez wejścia hamującego. Wagi modelu odpowiadają współczynnikom modelu liniowego. Tu neuron ma tylko jedno wejście o wagie równej 1. Gdy próg jest równy 0, aktywacja neuronu następuje dla wartości wejściowych większych niż 0. Wartości wejściowe większe niż 0 skutkują aktywacją neuronu

- Do sumy dodawany jest *błąd systematyczny* (ang. *bias*) uwzględniany przy sprawdzaniu progu. Ma to kilka celów. Po pierwsze, pozwala uwzględnić błąd statystyczny występujący w neuronach wejściowych. Po drugie, umożliwia standardyzację progów z użyciem wartości (na przykład zera) bez utraty ogólności.

<sup>9</sup> Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, „Psychological Review” 65, nr 6 (listopad 1958): 386 – 408.

- W perceptronie wagi wartości wejściowych mogą być niezależne od siebie i ujemne. Oznacza to dwie ważne rzeczy. Po pierwsze, neuronu nie trzeba wielokrotnie wiązać z wejściem, aby zwiększyć znaczenie danego wejścia. Po drugie, wejście z dendrytu może mieć wpływ hamujący (choć nie blokujący), jeśli przypisana mu jest ujemna waga.
- Opracowanie perceptronu dało początek algorytmom uczącym się optymalnych wag na podstawie zbioru danych wejściowych i wyjściowych.

Na rysunku 6.4 pokazana jest graficzna ilustracja nowego rozszerzonego modelu. Tak jak wcześniej tworzona jest wartość pośrednia w wyniku sumowania wartości wejściowych z uwzględnieniem wag. Zwróć jednak uwagę na zastosowanie wartości błędu systematycznego,  $w_0$ . Jest on wyznaczany razem z wagami wejść na etapie treningu. Więcej na ten temat dowiesz się w dalszych punktach. Wartość pośrednia ( $a$ ) jest następnie przekazywana do funkcji sprawdzającej próg, aby ustalić końcowy wynik,  $y$ .



**Rysunek 6.4.** Perceptron. Wartości wejściowe od  $x_1$  do  $x_n$  są mnożone przez przypisane im wagie, a następnie do sumy dodawany jest błąd systematyczny perceptronu,  $w_0$ . Wartość wyjściowa,  $a$ , jest następnie przekazywana do funkcji sprawdzającej próg w celu ustalenia danych wyjściowych

### 6.3.1. Trening

Wiesz już, że sieć neuronowa składa się z licznych podstawowych elementów, perceptronów. Teraz pora przyjrzeć się treningowi pojedynczych perceptronów. Na czym polega *trening* perceptronów? Posłużmy się konkretnym przykładem z logiczną funkcją I. Zastanówmy się nad perceptronem z dwoma wejściami binarnymi i binarną funkcją aktywującą z progiem 0. W jaki sposób nauczyć perceptron wag, aby dane wyjściowe perceptronu wynosiły 1 wtedy i tylko wtedy, gdy oba wejścia są równe 1? Ujmijmy to inaczej — czy można dobrać wagi (na ciągłej skali) w taki sposób, aby suma ważona wejść była większa od 0, gdy oba wejścia są równe 0, i mniejsza od 0 w pozostałych sytuacjach? Przedstawmy ten problem formalnie. Poniżej  $\mathbf{x}$  to wektor binarnych wartości wejściowych, a  $\mathbf{w}$  to wektor wag dla wejść:

$$\mathbf{x} = (x_1, x_2), \mathbf{w} = (w_1, w_2)$$

Należy więc ustalić wagi w taki sposób, aby dla kombinacji binarnych wejść  $x_1, x_2$  zachowane były następujące ograniczenia:

$$x \cdot w = \begin{cases} > 0, \text{ gdy } x_1 = x_2 = 1 \\ \leq 0 \text{ w przeciwnym razie} \end{cases}$$

Niestety, dla tak postawionego problemu nie istnieje rozwiązanie. Możliwe są teraz dwa wyjścia. Można albo zmienić próg i zdefiniować go jako wartość inną niż 0, albo dodać przesunięcie. Są to analogiczne rozwiązania. Na potrzeby dalszego omówienia stosujemy drugą z tych możliwości, co daje następujące nowe wektory:

$$\mathbf{x} = (1, x_1, x_2), \mathbf{w} = (w_0, w_1, w_2)$$

Równania pozostają bez zmian. Teraz widać, że staranny dobór wag umożliwia uzyskanie funkcji I. Pomyśl o sytuacji, gdy  $w_1 = 1$ ,  $w_2 = 1$  i  $w_0 = -1,5$ . W tabeli 6.1 pokazane są dane wyjściowe perceptronu i dane wyjściowe funkcji I.

**Tabela 6.1.** Porównanie danych wyjściowych perceptronu z danymi wyjściowymi logicznej funkcji I, która zwraca 1, jeśli oba wejścia są równe 1. Wyniki są podane dla ustawień  $w_1 = 1$ ,  $w_2 = 1$  i  $w_0 = -1,5$

<b>x<sub>1</sub></b>	<b>x<sub>2</sub></b>	<b>w<sub>0</sub></b>	<b>Suma ważona</b>	<b>Znak sumy ważonej</b>	<b>x<sub>1</sub> I x<sub>2</sub></b>
1	0	-1,5	-0,5	Ujemny	0
0	1	-1,5	-0,5	Ujemny	0
0	0	-1,5	-1,5	Ujemny	0
1	1	-1,5	0,5	Dodatni	1

Po zrozumieniu, że możliwe jest przedstawienie logicznej funkcji I za pomocą perceptronu, trzeba opracować systematyczny sposób uczenia wag w nadzorowany sposób. Można ująć to tak: jak nauczyć perceptron wag, gdy dostępny jest zbiór danych obejmujący wejścia i wyjścia (tu są one powiązane liniowo)? Umożliwia to algorytm dla perceptronów opracowany przez Rosenblatta<sup>10,11</sup>. Na listingu 6.1 pokazany jest pseudokod tego algorytmu.

#### Listing 6.1. Algorytm uczenia perceptronów

Inicjowanie **w** losowymi małymi liczbami

Dla każdego elementu ze zbioru treningowego:

Oblicz wartość wyjściową perceptronu dla danego elementu.

Dla każdej wagi przeprowadź aktualizację w zależności od poprawności danych wyjściowych.

Na razie wszystko idzie dobrze. Rozwiążanie wydaje się proste, prawda? Zaczynamy od ustawienia dla wag niewielkich losowych wartości, a następnie iteracyjnie przetwarzamy punkty danych i aktualizujemy wagi w zależności od tego, czy perceptron zwrócił poprawny wynik. Aktualizowanie wag ma miejsce tylko wtedy, gdy dane wyjściowe

<sup>10</sup>Rosenblatt, *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*.

<sup>11</sup>Rosenblatt, *Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanisms* (Spartan Books, 1962).

są błędne. W przeciwnym razie wagi się nie zmieniają. Ponadto wagi są aktualizowane w taki sposób, że ich wielkość jest przybliżana do wartości wektorów wejściowych, a znak jest ustawiany na znak wartości wyjściowej. Na listingu 6.2 przedstawione jest to w formalnej postaci.

#### Listing 6.2. Algorytm uczenia perceptronów (2)

Inicjowanie w losowymi małymi liczbami

Dla każdego przykładu  $j$ :

$$y_j(t) = \sum_k w_k(t) \cdot x_{j,k}$$

Oblicza wartość wyjściową perceptronu na podstawie aktualnych wag (czas  $t$ ).

Dla każdej wagi  $k$ :

$$w_k(t+1) = w_k(t) + (d_j - y_j(t)) \cdot x_{j,k}$$

Aktualizowanie cech. Zauważ, że zamiast pętli często stosowana jest tu operacja na wektorze.

Cechy są aktualizowane tylko wtedy, gdy oczekiwana wartość wyjściowa  $d_j$  jest różna od rzeczywistej. Wtedy waga przyjmuje znak poprawnej odpowiedzi oraz wartość wyznaczaną przez dane wejście i skalę  $\eta$ . Wartość  $\eta$  określa szybkość aktualizowania wag w algorytmie.

Jeśli dane wejściowe są liniowo separowalne, taki algorytm gwarantuje osiągnięcie konwergencji<sup>12</sup>.

#### 6.3.2. Trening perceptronu z użyciem pakietu scikit-learn

Wcześniej przedstawiliśmy najprostszą postać sieci neuronowych — perceptron. Wyjaśniliśmy też, jak przebiega ich trening. Teraz pora zastosować pakiet scikit-learn i zobaczyć, jak przeprowadzić trening perceptronu z użyciem danych. Na listingu 6.3 znajduje się kod importujący potrzebne narzędzia i generujący zawierającą punkty danych tablicę z pakietu NumPy.

#### Listing 6.3. Generowanie danych na potrzeby uczenia perceptronu

```
import numpy as np
import matplotlib.pyplot as plt
import random
from sklearn.linear_model import perceptron
```

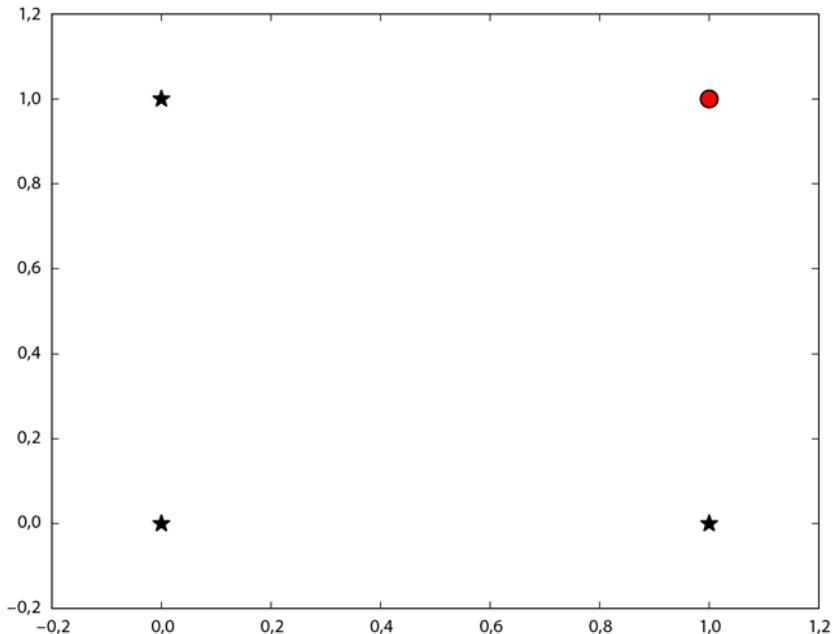
```
data = np.array([[0.1], [0.0], [1.0], [1.1]])
target = np.array([0, 0, 1])
```

Tworzenie czterech punktów danych w tablicy z pakietu NumPy.

Określanie klas tych punktów danych. Tu tylko dla punktu danych  $x = 1, y = 1$  docelowa klasa to 1. Pozostałe punkty danych są przypisane do klasy 0.

Na listingu 6.3 importowane są narzędzia potrzebne do przykładu z jednym perceptronem. Kod tworzy też bardzo krótki zbiór danych z czterema punktami danych. Ten zbiór danych jest zapisywany w tablicy z pakietu NumPy (jej nazwa to `data`). Każdy punkt danych jest przypisywany do klasy 0 lub 1, a klasy są zapisywane w tablicy `target`. Na rysunku 6.5 te dane są przedstawione w formie graficznej.

<sup>12</sup>Brian Ripley, *Pattern Recognition and Neural Networks* (Cambridge University Press, 1996).



**Rysunek 6.5.** Graficzny obraz danych dla jednego perceptronu. Dane z klasy 1 są reprezentowane za pomocą kropki, natomiast dane z klasy 0 — za pomocą gwiazdki. Perceptron ma nauczyć się rozróżniać punkty danych z tych klas

Na rysunku jedyny punkt danych z pozytywnej klasy (o etykiecie 1) ma współrzędne (1,1) i jest reprezentowany przez kropkę. Pozostałe punkty danych są przypisane do negatywnej klasy. Na listingu 6.4 znajdziesz kod służący do treningu perceptronu i zwracający współczynniki ( $w_1$  i  $w_2$  powiązane z wartościami  $x_1$  i  $x_2$ ) i błąd systematyczny  $w_0$ .

#### Listing 6.4. Trening pojedynczego perceptronu

```
p = perceptron.Perceptron(n_iter=100) ← Tworzy jeden perceptron.  
p_out = p.fit(data,target) ← Parametr n_iter określa liczbę iteracji z użyciem danych w trakcie treningu.  
print p_out ← Przeprowadza trening perceptronu z użyciem danych i powiązanych z nimi docelowych klas.  
msg = ("Współczynniki: %s, Punkt przecięcia: %s") ← Wyświetla współczynniki i błąd systematyczny perceptronu.  
print msg % (str(p.coef_),str(p.intercept_))
```

Dane wyjściowe wyglądają mniej więcej tak:

```
Perceptron(alpha=0.0001, class_weight=None, eta0=1.0, fit_intercept=True,  
n_iter=100, n_jobs=1, penalty=None, random_state=0, shuffle=False,  
verbose=0, warm_start=False)  
Współczynniki: [[ 3.  2.]] .Punkt przecięcia: [-4.]
```

Pierwszy wiersz określa parametry używane w trakcie treningu perceptronu. W drugim wierszu znajdują się uzyskane wagi i błąd systematyczny. Nie przejmuj się, jeśli uzyskałeś

nieco inne współczynniki i punkt przecięcia osi. Istnieje wiele rozwiązań omawianego problemu, a przedstawiony algorytm uczenia może zwrócić dowolne z nich. Aby lepiej zrozumieć parametry, warto zapoznać się z ich opisem w dokumentacji pakietu scikit-learn<sup>13</sup>.

### 6.3.3. Geometryczna interpretacja działania perceptronu dla dwóch wejść

W przedstawionym przykładzie z powodzeniem ukończyliśmy trening jednego perceptronu i uzyskaliśmy wagi oraz błąd systematyczny. To świetnie! Jak jednak zinterpretować te wagi w intuicyjny sposób? Na szczęście można to zrobić w przestrzeni dwuwymiarowej, a także rozszerzyć tę ilustrację na przestrzeń o większej liczbie wymiarów.

Rozważ perceptron o tylko dwóch wartościach wejściowych. Z rysunku 6.4 wynika, że:

$$y = w_0 + w_1 x_1 + w_2 x_2$$

Ten wzór powinien wyglądać znajomo. Wygląda on jak równanie płaszczyzny (w trzech wymiarach) dla  $x_1$ ,  $x_2$  i  $y$ . Jeśli wymiary nie pokrywają się ze sobą, ta płaszczyzna przecina się z płaszczyzną rzutowania widoczną na rysunku 6.5 i otrzymujemy linię. Gdy spojrzeć na nią z perspektywy uwzględnionej na rysunku 6.5, punkty po jednej stronie linii odpowiadają wartościom  $x \cdot w > 0$ , natomiast punkty po drugiej stronie — wartościom  $x \cdot w < 0$ . Punkty na linii odpowiadają wartościom  $x \cdot w = 0$ . Posłużmy się konkretnym przykładem i zwizualizujmy te rozważania. Wyuczone wcześniej współczynniki dają następujące równanie płaszczyzny:

$$y = -4 + 3x_1 + 2x_2$$

Wartość  $y$  jest określana pod kątem 90 stopni do płaszczyzny ( $x_1$ ,  $x_2$ ), dlatego można ją traktować jak linię biegnącą od oczu patrzącego i przechodzącą przez płaszczyznę rzutowania. Wartość  $y$  na płaszczyźnie rzutowania wynosi 0, dlatego można ustalić punkt przecięcia, podstawiając 0 pod  $y$  w poprzednim równaniu:

$$0 = -4 + 3x_1 + 2x_2$$

$$4 - 3x_1 = 2x_2$$

$$x_2 = \frac{4}{2} - \frac{3}{2}x_1$$

Ostatni wiersz jest zgodny ze standardową postacią zapisu linii prostej. Teraz wystarczy ją narysować, aby zobaczyć, w jaki sposób perceptron rozdzielił dane treningowe. Potrzebny kod jest pokazany na listingu 6.5, a dane wyjściowe zobaczysz na rysunku 6.6.

#### Listing 6.5. Wyświetlanie danych wyjściowych z perceptronu

```
colors = np.array(['k', 'r'])           ←
markers = np.array(['*', 'o'])          ← Konfigurowanie tablicy kolorów i rysowanie punktów
for data,target in zip(data,target):    ← danych. Czarna gwiazdka reprezentuje klasę 0, a czerwona
                                         ← kropka – klasę 1.
```

<sup>13</sup> scikit-learn, Perceptron, <http://mng.bz/U20J>.

```

plt.scatter(data[0],data[1],s=100,
            c=colors[target].marker=markers[target])

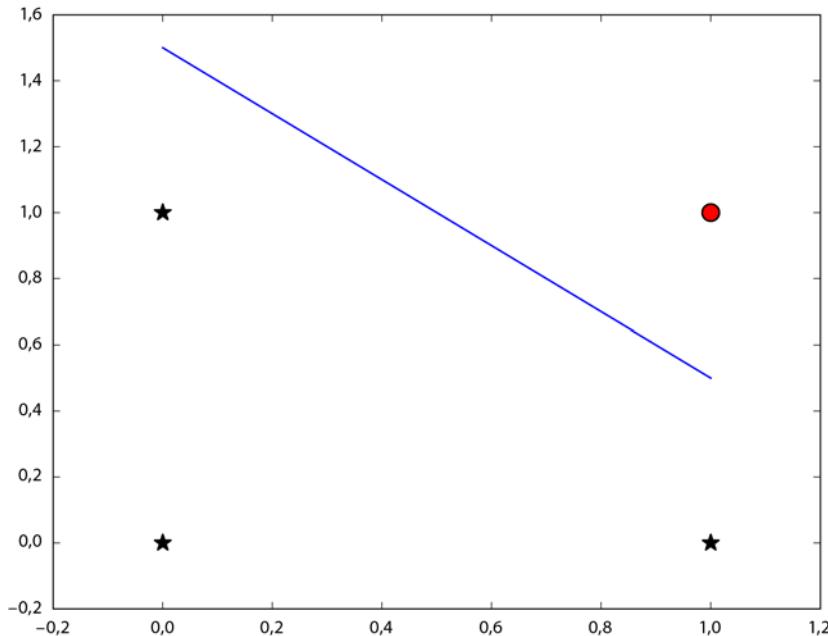
grad = -p.coef_[0][0]/p.coef_[0][1]
intercept = -p.intercept_/p.coef_[0][1]

x_vals = np.linspace(0,1)           ← Oblicza parametry linii
y_vals = grad*x_vals + intercept  | (wyznaczonej przez przecięcie
plt.plot(x_vals,y_vals)           | dwóch płaszczyzn).

plt.show()

```

**Tworzy punkty danych i rysuje linię przecięcia dwóch płaszczyzn.**



Rysunek 6.6. Rzut płaszczyzny podziału na płaszczyznę rzutowania ( $y = 0$ ).

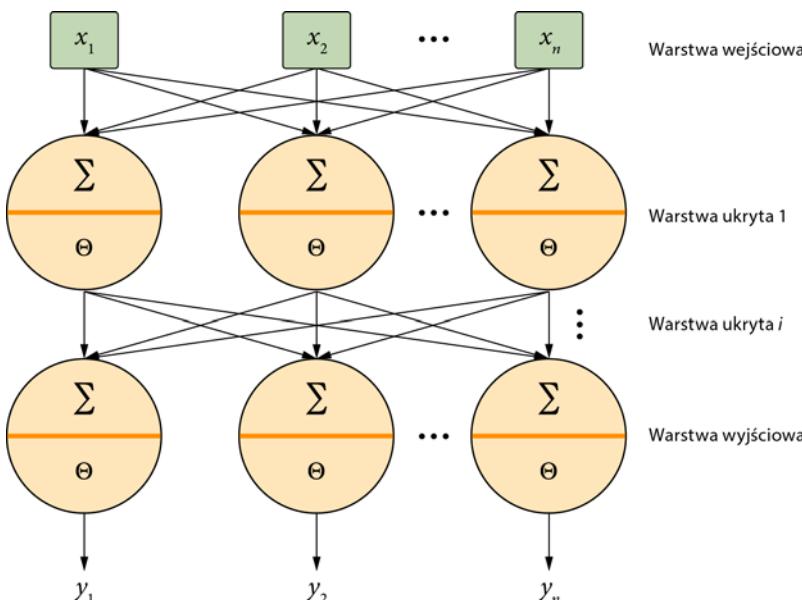
Wszystkie punkty w prawej górnej części wykresu są zgodne z ograniczeniem  $w \cdot x > 0$ , natomiast punkty znajdujące się na lewo i w dół od linii są zgodne z ograniczeniem  $w \cdot x < 0$

Rysowane są tu cztery punkty danych i rzut wyznaczonej przez perceptron w wyniku uczenia płaszczyzny, która dzieli te punkty. Pozwala ona uzyskać podział pokazany na rysunku 6.6. Dla większej liczby zmiennych wejściowych można przyjąć, że dane znajdują się w przestrzeni  $n$ -wymiarowej, a perceptron dzieli je za pomocą hiperplaszczyzny w  $n+1$  wymiarach. Powinieneś teraz widzieć, że podstawowa liniowa postać perceptronu (z aktywacją na podstawie progu) jest odpowiednikiem podziału punktów z użyciem hiperplaszczyzny. Dlatego modele tego rodzaju można stosować wyłącznie wtedy, gdy dane są liniowo separowalne. W innych sytuacjach użycie tej techniki do podziału danych na klasy pozytywne i negatywne nie jest możliwe.

## 6.4. Perceptrony wielowarstwowe

We wcześniejszych punktach przyjrzaliśmy się uczeniu głębszemu na bardzo ogólnym poziomie oraz omówiliśmy podstawy sieci neuronowych (a dokładniej — jednostkę sieci neuronowych nazywaną perceptronem). Pokazaliśmy też, że podstawowa postać perceptronu działa zgodnie z modelem liniowym.

Aby przeprowadzić nieliniowy podział danych, można zachować prostą funkcję aktywacji na podstawie progu i zwiększyć złożoność architektury sieci w celu uzyskania *wielowarstwowej sieci jednokierunkowej* (ang. *multilayer feed-forward networks*). W sieciach tego typu perceptrony są uporządkowane w warstwy. Dane wejściowe każdej warstwy pochodzą z warstwy poprzedniej, a dane wyjściowe poszczególnych warstw są danymi wejściowymi dla następnych. Określenie *jednokierunkowa* związanego jest z tym, że dane płyną tylko w jedną stronę sieci (nie występują w niej cykle). Rysunek 6.7 przedstawia to zagadnienie w formie graficznej i stanowi rozwinięcie rysunku 6.3.



**Rysunek 6.7.** Perceptron wielowarstwowy. Przyjrzyj się mu od góry do dołu: obejmuje warstwę wejściową (wektor wartości  $x$ ), szereg warstw ukrytych i warstwę wyjściową zwracającą wektor wartości  $y$

Aby zilustrować nieliniowość, rozważmy bardzo krótki przykład, z którym perceptron sobie nie radzi — funkcję XOR. Ten przykład pochodzi z opublikowanej w 1969 roku książki Minsky’ego i Paperta *Perceptrons: An Introduction to Computational Geometry*<sup>14</sup>.

<sup>14</sup> Marvin Minsky i Seymour Papert, *Perceptrons: An Introduction to Computational Geometry* (MIT Press, 1969). Jest to książka o ciekawej i kontrowersyjnej historii. Często uważa się, że na wiele lat spowolniła ona postępy w pracach nad perceptronami wielowarstwowymi. Wynikało to z tego, że Minsky i Papert sądzili, iż perceptrony wielowarstwowe nie dają dodatkowych możliwości obliczeniowych w porównaniu z prostszymi perceptronami (a ponieważ prostsze perceptrony nie potrafią

Dalej pokazujemy, jak przybliżyć działanie tej funkcji z użyciem perceptronu dwuwarstwowego, oraz omawiamy algorytm z propagacją wsteczną używany do treningu takich sieci. Działanie funkcji XOR pokazane jest w tabeli 6.2.

<b>x<sub>1</sub></b>	<b>x<sub>2</sub></b>	<b>Dane wyjściowe</b>
0	0	0
0	1	1
1	0	1
1	1	0

**Tabela 6.2.** Wartości wejściowe i wyjściowe funkcji XOR.  
Funkcja ta zwraca 1, jeśli albo  $x_1$ , albo  $x_2$  mają taką wartość.  
Jeżeli wartość obu zmiennych jest taka sama (1 lub 0), funkcja zwraca 0

Jeśli przedstawić funkcję XOR w formie graficznej, posługując się konwencjami z rysunku 6.5, powstanie rysunek 6.8. Widać tu, że dane wyjściowe funkcji XOR nie są liniowo separowalne w przestrzeni dwuwymiarowej. Nie istnieje jedna hiperplaszczyzna, która pozwala rozdzielić elementy klas pozytywnej i negatywnej. Spróbuj narysować na tym wykresie linię w taki sposób, by po jednej stronie znalazły się elementy klasy pozytywnej, a po drugiej — elementy klasy negatywnej. Jest to niemożliwe. Można jednak byłoby rozdzielić punkty danych, gdyby istniało więcej hiperplaszczyn i sposób na ich połączenie. Zróbcmy to! Takie rozwiązanie to odpowiednik sieci z jedną warstwą ukrytą i ostateczną warstwą łączącą dane wyjściowe. Na rysunku 6.9 taka sieć jest przedstawiona w formie graficznej.

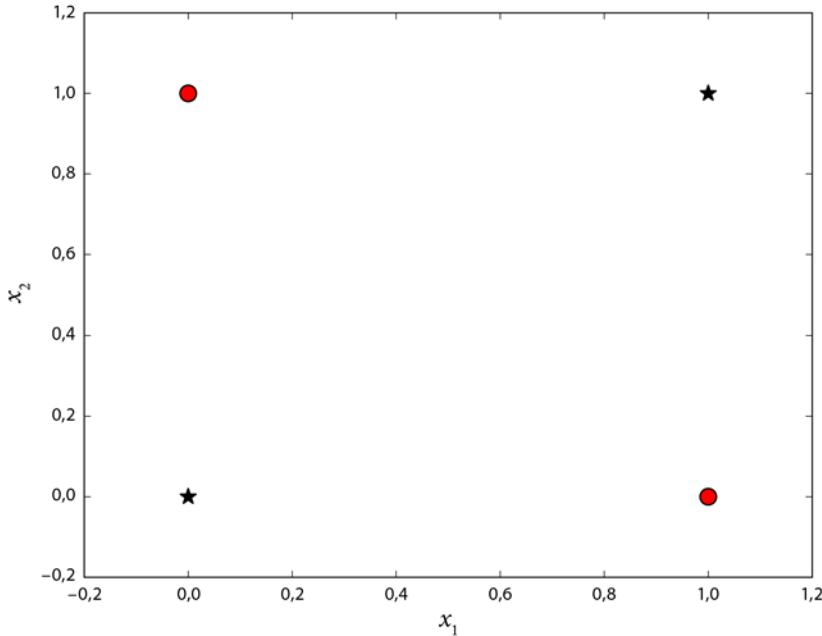
Na rysunku widoczna jest dwuwarstwowa sieć z warstwą ukrytą. Sieć przyjmuje dwie wartości wejściowe i generuje jedną wartość wyjściową. Z każdym węzłem ukrytym i z węzłem wyjściowym powiązany jest błąd systematyczny. Pamiętaj, że jego wartość zawsze jest równa 1, choć zmieniają się jego wagи. Dzięki temu w trakcie treningu sieć może nauczyć się zarówno profilu aktywacji, jak i przesunięć wartości. Poświeć chwilę na przekonanie się, że pokazane rozwiązanie rzeczywiście działa.

Każdy ukryty węzeł tworzy jedną hiperplaszczynę (podobnie jak robi to pojedynczy perceptron z rysunku 6.6). Te hiperplaszczyny są łączone przez końcowy perceptron. Ten końcowy perceptron działa jak bramka I z dwoma warunkami. Pierwszy z nich jest taki, że dane wejściowe z przestrzeni  $x_1, x_2$  znajdują się powyżej dolnej linii z rysunku 6.10. Drugi dotyczy tego, że dane wejściowe muszą znajdować się poniżej górnej linii z rysunku 6.10. W ten sposób dwuwarstwowy perceptron wyznacza w przestrzeni pas obejmujący oba pozytywne elementy z danych treningowych, ale nie obejmuje elementów negatywnych. W ten sposób udało się podzielić nieliniowo separowalny zbiór danych.

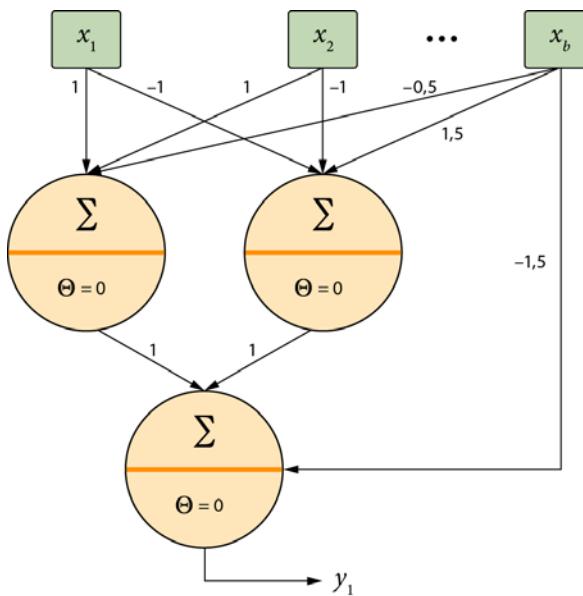
W tym punkcie omówiliśmy zastosowanie sieci neuronowych do nieliniowo separowalnych zbiorów danych. Na podstawie funkcji XOR pokazaliśmy, że można ręcznie zbudować sieć neuronową, która dzieli nieliniowo separowalny zbiór danych, i przedstać intuicyjną geometryczną interpretację działania takiej sieci. Pomineliśmy jednak

---

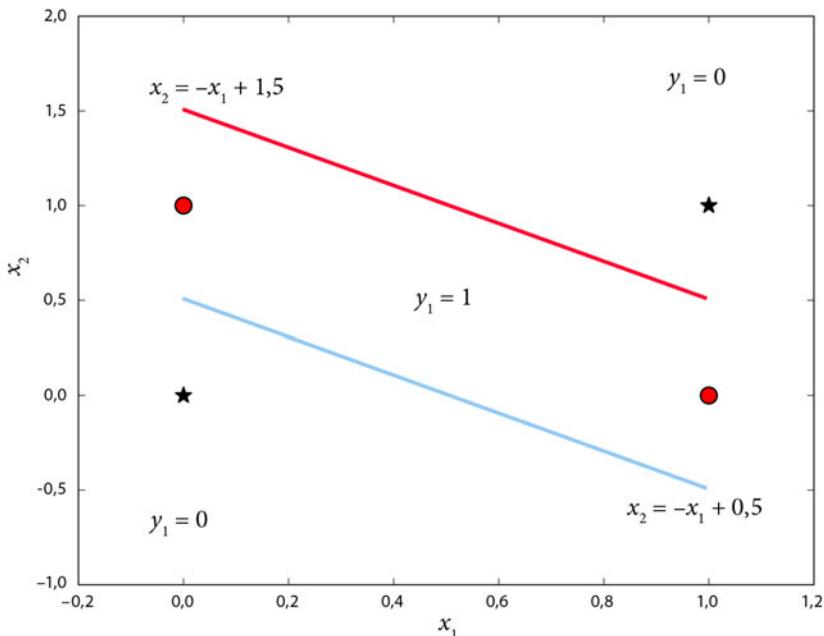
reprezentować funkcji niektórych typów, na przykład funkcji XOR, nie potrafią tego także perceptrony wielowarstwowe). Mogło to spowodować, że naukowcy na wiele lat zrezygnowali z badania perceptronów. Wspomniane kontrowersyjne twierdzenia zostały odwołane w późniejszych wydaniach książki.



**Rysunek 6.8.** Graficzna reprezentacja funkcji XOR. Elementom klasy pozytywnej odpowiadają kółka, a elementom klasy negatywnej — gwiazdki. W jednej hiperpłaszczyźnie nie da się rozdzielić tych punktów danych na dwa zbiory, co oznacza, że ten zbiór danych nie jest separowalny liniowo



**Rysunek 6.9.** Perceptron dwuwarstwowy może działać tak jak funkcja XOR, która przeprowadza podział nieliniowy. Wartości na połączeniach określają wagę tych połączeń. Dodanie błędu systematycznego zapewnia poprawne działanie sieci dla progu aktywacji równego 0. Dwa ukryte neurony odpowiadają dwóm hiperpłaszczyznom. Końcowy, łączący dane perceptron to odpowiednik logicznej funkcji I dla danych wyjściowych z dwóch ukrytych neuronów. Można przyjąć, że końcowy perceptron wybiera obszary płaszczyzny ( $x_1, x_2$ ), w których oba ukryte neurony są jednocześnie aktywne



**Rysunek 6.10.** Podział nieliniowo separowalnego zbioru danych za pomocą sieci neuronowej przedstawionej na rysunku 6.9. Zauważ, że neuron pokazany po lewej stronie na rysunku 6.9 daje na płaszczyźnie rzutowania dolną linię, natomiast neuron widoczny po prawej stronie tworzy górną linię. Lewy neuron generuje wartość wyjściową 1 tylko wtedy, gdy wartość wejściowa znajduje się powyżej dolnej linii, natomiast prawy neuron generuje wartość wyjściową 1 tylko wtedy, gdy wartość wejściowa znajduje się poniżej górnej linii. Końcowy neuron łączący zwraca  $y_1 = 1$  tylko wtedy, gdy oba podane warunki są spełnione. Dlatego sieć generuje wartość 1 tylko dla punktów danych z wąskiego pasa między linią górną a dolną.

pewien istotny krok. Ważne jest, by móc automatycznie ustalić wagę dla sieci na podstawie treningowego zbioru danych. Te wagie można następnie wykorzystać do klasyfikowania i prognozowania danych innych niż pierwotne dane wejściowe. Tym zagadniением poświęcony jest następny punkt.

#### 6.4.1. Trening z wykorzystaniem propagacji wstecznej

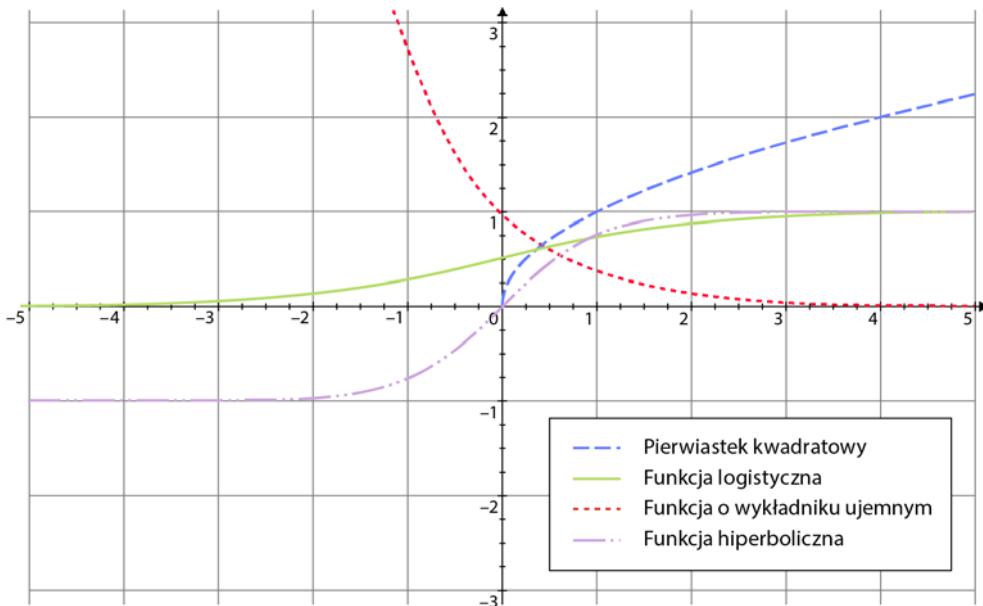
W poprzednim przykładzie jako funkcję aktywacji neuronu wykorzystaliśmy funkcję skokową. Oznacza to, że używaliśmy jednej wartości progowej, której przekroczenie powoduje aktywację neuronu. Niestety, wymyślenie metody automatycznego treningu takiej sieci jest trudne. Dzieje się tak, ponieważ funkcja skokowa nie umożliwia zakodowania niepewności. Określony w niej próg jest sztywny.

Odpowiedniesze byłoby użycie funkcji, która przybliża działanie funkcji skokowej, ale pracuje w stopniowy sposób. W tym podejściu niewielka zmiana jednej z wag prowadzi do niewielkiej zmiany w działaniu całej sieci. Właśnie takie rozwiązanie zastosujemy. Zamiast posługiwać się funkcją skokową, zastąpimy ją ogólniejszą funkcją aktywacji. W następnym punkcie pokrótko przedstawiamy często używane funkcje aktywacji, a następnie wyjaśniamy, jak wybór jednej z nich pomaga w opracowaniu algorytmu treningowego.

### 6.4.2. Funkcje aktywacji

Warto poświęcić chwilę na przyjrzenie się funkcjom aktywacji, które można zastosować dla perceptronu. Z najprostszym przypadkiem już się zapoznałeś — to sztywny próg w okolicach wartości 0. Przy przesunięciu równym zero daje to profil aktywacji pokazany na rysunku 6.3. Jakie są inne możliwości? Rysunek 6.11 przedstawia profile aktywacji kilku innych funkcji. Oto ich definicje:

- *Pierwiastek kwadratowy* —  $y = \sqrt{x}$ , dziedzina  $[0, \infty]$ , przeciwdziedzina  $[0, \infty]$ .
- *Funkcja logistyczna* —  $1/(1+e^{-x})$ , dziedzina  $[-\infty, \infty]$ , przeciwdziedzina  $[0, 1]$ .
- *Funkcja o wykładniku ujemnym* —  $e^{-x}$ , dziedzina  $[-\infty, \infty]$ , przeciwdziedzina  $[0, \infty]$ .
- *Funkcja hiperboliczna* —  $(e^x - e^{-x})/(e^x + e^{-x})$ , dziedzina  $[-\infty, \infty]$ , przeciwdziedzina  $[-1, 1]$ . Zauważ, że jest to odpowiednik funkcji logistycznej z danymi wyjściowymi przekształconymi na inną przeciwdziedzinę.



**Rysunek 6.11.** Profile danych wyjściowych dla kilku funkcji aktywacji, przedstawione w dziedzinie z rysunku 6.3. Przedstawione tu profile aktywacji dotyczą pierwiastka kwadratowego, funkcji logistycznej, funkcji o wykładniku ujemnym i funkcji hiperbolicznej

Zastosowanie takich funkcji aktywacji umożliwia tworzenie i trening wielowarstwowych sieci neuronowych przybliżających znacznie większą klasę funkcji. Najważniejszą cechą funkcji aktywacji jest to, że są różniczkowalne. Z następnego punktu dowiesz się, dlaczego tak jest. W dalszej części rozdziału posługujemy się funkcją logistyczną (pierwszy raz zetknąłeś się z nią w rozdziale 4.). Ma ona odpowiednią dziedzinę i przeciwdziedzinę oraz jest często stosowana w omawianym celu przez innych autorów.

### Regresja logistyczna, perceptron i uogólniony model liniowy

Przypomnij sobie rozdział 4., w którym przedstawiliśmy regresję logistyczną. Ustaliliśmy tam, że model z odpowiedzią liniową byłby nieodpowiedni dla szacowania prawdopodobieństwa. Dlatego zastosowaliśmy funkcję logistyczną zwracającą odpowiedź w postaci krzywej, aby uzyskać przydatniejsze wyniki. Następnie ustaliliśmy, że logarytm szans jest liniowy ze względu na kombinację wag i zmiennych wejściowych oraz zastosowaliśmy opisany model do rozwiązania problemu klasyfikowania.

W tym rozdziale zaczęliśmy od prostego zjawiska biologicznego i zbudowaliśmy formalny model obliczeniowy, który je ujmie. Na razie w ogóle nie uwzględnialiśmy prawdopodobieństwa; zamiast tego zaczęliśmy od aktywacji neuronu. Intuicyjnie można przedstawić to w bardziej ogólny sposób i uzyskać następujące równanie:

$$y = \frac{1}{1 + e^{-(w_0 + w_1x_1 + \dots + w_nx_n)}}$$

Natrafiliśmy tu na bardziej ogólną klasę problemów — *uogólnione modele liniowe* (ang. *generalized linear models* — **GLM**)<sup>a</sup>. W modelach tej klasy model liniowy ( $w_0 + w_1x_1 + \dots + w_nx_n$ ) jest powiązany ze zmienną wyjściową ( $y$ ) za pomocą funkcji łączającej, którą tu jest funkcja logistyczna  $1/(1+e^{-x})$ .

Równoznaczność algorytmów i zjawisk często występuje w uczeniu maszynowym. Zetknąłeś się już z tym w tej książce. Przypomnij sobie podrozdział 2.5, w którym omawialiśmy równoznaczność maksymalizacji wartości oczekiwanej z użyciem gaussowskiego modelu mieszanego z określona kowariancją oraz z użyciem zwykłego algorytmu k-srednich. Powodem takich analogii jest zwykle to, że wielu naukowców rozpoczęyna badania od różnych punktów wyjścia i odkrywa równoznaczność rozwiązań w wyniku uogólniania podstawowych cegiełek.

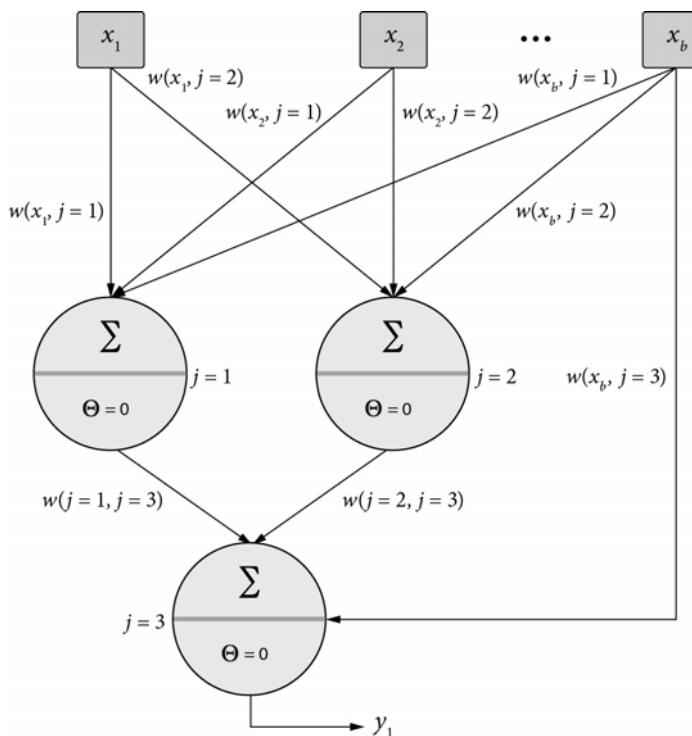
<sup>a</sup> Peter McCullagh i John A. Nelder, *Generalized Linear Models* (Chapman and Hall/CRC, 1989).

### 6.4.3. Intuicyjne wyjaśnienie propagacji wstecznej

Aby intuicyjnie wyjaśnić propagację wstecną, posłużymy się ponownie funkcją XOR. Jednak tym razem spróbujemy ustalić wagę w wyniku uczenia, a nie określać je ręcznie. Ponadto od teraz funkcją aktywacji będzie funkcja logistyczna (sigmoidalna). Na rysunku 6.12 pokazana jest graficzna ilustracja nowej sieci.

Zadanie polega na ustaleniu za pomocą uczenia  $w(a,b) \forall a,b$  na podstawie treningowego zbioru danych. Czy uda się wymyślić algorytm, który minimalizuje błąd (kwadrat różnic między oczekiwanyymi a rzeczywistymi wartościami  $y_1$ ) dla tego zbioru danych, gdy dane wejściowe z tego zbioru zostaną przekazane do sieci?

Jednym z rozwiązań jest algorytm *propagacji wstecznej*. Działa on w następujący sposób: najpierw inicjuje wszystkie wagę losowymi wartościami, a następnie przepuszcza jeden element z danych treningowych przez sieć. Oblicza błąd na podstawie danych wyjściowych i stosuje *propagację wsteczną* tego błędu przez sieć (stąd nazwa tej techniki). Każda waga w sieci jest zmieniana w kierunku, który pozwala zminimalizować błąd sieci. Proces ten jest powtarzany do momentu spełnienia warunku końcowego (na przykład wykonania określonej liczby iteracji lub uzyskania konwergencji).



**Rysunek 6.12.** Ilustracja propagacji wstecznej. Dany jest zbiór danych wejściowych  $x_1, x_2$  i docelowa zmienna  $y_1$ , wyznaczana na podstawie funkcji XOR i wartości  $x_1, x_2$ . Czy możliwe jest nauczenie sieci wartości  $w$ , tak by zminimalizować kwadraty różnic między wynikami ze zbioru treningowego a danymi wyjściowymi sieci? W tym przykładzie jako funkcji aktywacji używamy funkcji logistycznej:

$$\Theta = \frac{1}{1 + e^{-(w_0 + \sum_j w_j x_j)}}$$

#### 6.4.4. Teoria propagacji wstecznej

Aby ułatwić zrozumienie tego zagadnienia, regułę aktualizacji wag w propagacji wstecznej można rozbić na dwie części: aktualizowanie wag powiązanych z neuronami wyjściowymi i aktualizowanie wag powiązanych z neuronami ukrytymi. Logicznie obie te operacje przebiegają identycznie, jednak obliczenia matematyczne dla drugiego z tych zadań są bardziej skomplikowane. Dlatego tu omawiamy tylko pierwszą z tych operacji, aby dać Ci przedsmak propagacji wstecznej. Jeśli chcesz zrozumieć ją w całości, zapoznaj się z przełomowym tekstem z magazynu *Nature*<sup>15</sup>.

Oto pierwsza kwestia istotna w kontekście treningu — interesuje nas to, w jakim stopniu błąd w danych wyjściowych zmienia się względem zmiany wartości jednej wagi. Tylko na tej podstawie można zmieniać wagę, by minimalizować błąd w danych wyjściowych. Zaczniemy od ustalenia pochodnej częstkowej błędu z uwzględnieniem wagi konkretnego wejścia z niższej warstwy. Zakładamy na razie, że wszystkie pozostałe wagie są stałe. Używamy tu wzoru na pochodną funkcji złożonej:

<sup>15</sup> David E. Rumelhart, Geoffrey E. Hinton i Ronald J. Williams, *Learning Representations by BackPropagating Errors*, „Nature” 323 (październik 1986): 533 – 36.

$\frac{\delta E}{\delta w(i, j)}$	<b>Stopień zmiany błędu na podstawie zmiany wagi</b>
$\frac{\delta E}{\delta o(j)} \times$	<b>① Błąd na podstawie danych wyjściowych funkcji aktywacji</b>
$\frac{\delta o(j)}{\delta n(j)} \times$	<b>② Dane wyjściowe funkcji aktywacji na podstawie sumy ważonej wartości wejściowych</b>
$\frac{\delta n(j)}{\delta w(i, j)}$	<b>③ Suma ważona wartości wejściowych z uwzględnieniem poszczególnych wag</b>

Ujmijmy to prosto: stopień zmiany błędu w danych wyjściowych jest powiązany z wagą za pośrednictwem stopnia zmiany w wyrażeniach ①, ② i ③.

Może pamiętasz z wcześniejszych punktów, że używamy funkcji logistycznej, ponieważ ułatwia trening. Głównym tego powodem jest to, że ta funkcja jest różniczkowalna. Powinieneś już rozumieć, dlaczego jest to konieczne. Wyraz ② równania jest odpowiednikiem pochodnej funkcji aktywacji. Można to zapisać w następujący sposób:

$$\frac{\delta o(j)}{\delta n(j)} = \frac{\delta}{\delta n(j)} \Theta(n(j)) = \Theta(n(j))(1 - \Theta(n_j))$$

Oznacza to, że stopień zmiany danych wyjściowych funkcji aktywacji można zapisać w kategoriach tejże funkcji! Jeśli zdolamy obliczyć wyrażenia ① i ③, ustalimy kierunek, w którym należy zmienić wagę, aby zminimalizować błąd w danych wyjściowych. Okazuje się, że obliczenie tych wyrażeń jest możliwe. Wyrażenie ③ można bezpośrednio zróżniczkować:

$$\frac{\delta n(j)}{\delta w(i, j)} = x_i$$

Oznacza to, że stopień zmiany danych wejściowych funkcji aktywacji z uwzględnieniem konkretnej wagi łączącej  $i$  z  $j$  jest zależny tylko od wartości  $x_i$ . Ponieważ analizowaliśmy tylko warstwę wyjściową, określenie różniczki błędu na podstawie danych wyjściowych ① jest łatwe, jeśli bezpośrednio wykorzystamy wzór na błąd:

$$\frac{\delta E}{\delta o(j)} = \frac{\delta}{\delta o(j)} (o_{oczekiwana} - o(j))^2 = 2(o(j) - o_{oczekiwana})$$

Teraz można przedstawić kompletną regułę aktualizacji wagi dla węzła wyjściowego:

$$-\alpha x_i 2(o(j) - o_{oczekiwana}) \Theta(n(j))(1 - \Theta(n_j))$$

Waga jest więc aktualizowana na podstawie powiązanej z nią wartości, różnicy między danymi wyjściowymi a oczekiwana wartością i danych wyjściowych pochodnej funkcji aktywacji dla wszystkich wejść i wag. Zauważ, że do wzoru dodaliśmy znak minus i wyraz alfa. Znak minus sprawia, że zmiany będą zmniejszały błąd, a alfa określa stopień tych zmian.

Dzięki temu powinieneś zrozumieć, w jaki sposób aktualizowane są wagi powiązane z warstwą wyjściową. Aktualizowanie warstwy wewnętrznej odbywa się według

podobnych reguł. Trzeba jednak zastosować wzór na pochodną funkcji złożonej, aby ustalić wpływ wartości wyjściowej z danego węzła wewnętrznego na ogólny błąd sieci. Należy więc ustalić stopień zmiany wejść i wyjść na ścieżce prowadzącej z danego węzła do węzła wyjściowego. Tylko na tej podstawie można ustalić, jaki będzie stopień zmiany błędu w danych wyjściowych na podstawie określonej zmiany wag dla węzła wewnętrznego. W ten sposób uzyskujemy kompletną postać propagacji wstecznej<sup>16</sup>.

#### **6.4.5. Wielowarstwowe sieci neuronowe w pakiecie scikit-learn**

Jeśli już rozumiesz perceptron wielowarstwowy i teorię treningu z użyciem propagacji wstecznej, pora wrócić do Pythona i napisać kod. Ponieważ pakiet scikit-learn nie obejmuje implementacji wielowarstwowych perceptronów, zastosujemy pakiet PyBrain<sup>17</sup>. Jest on przeznaczony właśnie do budowania i treningu sieci neuronowych. Na listingu 6.6 pokazany jest pierwszy fragment kodu potrzebnego do zbudowania sieci neuronowej analogicznej do tej z rysunku 6.12. Operacje importu potrzebne do uruchomienia tego kodu znajdziesz w kompletnym pliku dostępnym w witrynie książki.

##### **Listing 6.6. Budowanie wielowarstwowego perceptronu za pomocą pakietu PyBrain**

#Tworzenie modułów sieci

```
net = FeedForwardNetwork()
in1 = LinearLayer(2)
hid1 = SigmoidLayer(2)
out1 = LinearLayer(1)
b = BiasUnit()
```

Najpierw tworzony jest obiekt typu FeedForwardNetwork. Budowane są też warstwy neuronów: wejściowa (in1), wyjściowa (out1) i ukryta (hid1). Zauważ, że warstwy wejściowa i wyjściowa używają zwykłej funkcji aktywacji (z progiem równym 0). Dla warstwy ukrytej na potrzeby treningu stosowana jest sigmoidalna funkcja aktywacji. W kodzie tworzony jest też obiekt błędu systematycznego. Na razie elementy te nie tworzą sieci neuronowej, ponieważ warstwy nie zostały połączone ze sobą. To zadanie wykonuje kod z listingu 6.7.

##### **Listing 6.7. Budowanie wielowarstwowego perceptronu za pomocą pakietu PyBrain (2)**

#Tworzenie połączeń

```
in_to_h = FullConnection(in1, hid1)
h_to_out = FullConnection(hid1, out1)
bias_to_h = FullConnection(b,hid1)
bias_to_out = FullConnection(b,out1)
```

#Dodawanie modułów do sieci

```
net.addInputModule(in1)
net.addModule(hid1);
net.addModule(b)
net.addOutputModule(out1)
```

<sup>16</sup>Rumelhart i współpracownicy, *Learning Representations by Back-Propagating Errors*, 533 – 36.

<sup>17</sup>Tom Schaul i współpracownicy, *PyBrain*, „Journal of Machine Learning Research” 11 (2010): 743 – 46.

```
#Dodawanie połączeń do sieci i sortowanie ich
net.addConnection(in_to_h)
net.addConnection(h_to_out)
net.addConnection(bias_to_h)
net.addConnection(bias_to_out)
net.sortModules()
```

Ten kod tworzy obiekty reprezentujące połączenia i dodaje zbudowane wcześniej neurony (moduły) oraz połączenia między nimi do obiektu typu `FeedForwardNetwork`. Wywołanie `sortModules()` kończy tworzenie sieci.

Przed przejściem dalej poświęć chwilę na zapoznanie się z typem `FullConnection`. Tu tworzone są cztery obiekty tego typu przekazywane do obiektu reprezentującego sieć. Konstruktor tego typu przyjmuje dwie warstwy, a wewnętrznie obiekt tworzy połączenie między każdym neuronem z warstwy podanej jako pierwszy parametr a każdym neuronem z warstwy podanej w drugim parametrze. Metoda wywołana na końcu sortuje moduły w obiekcie typu `FeedForwardNetwork` i przeprowadza wewnętrzne imijowanie danych.

Po utworzeniu sieci neuronowej analogicznej do tej z rysunku 6.12 należy ustalić jej wagę! Potrzebne będą do tego dane. Na listingu 6.8 pokazany jest potrzebny kod. Duża jego część pochodzi z dokumentacji pakietu PyBrain<sup>18</sup>.

#### Listing 6.8. Trening sieci neuronowej

```
d = [(0.0), ← Tworzenie zbioru danych i powiązanych z nimi docelowych wyników zgodnych z funkcją XOR.
      (0.1),
      (1.0),
      (1.1)] ← Tworzenie pustego obiektu typu SupervisedDataSet (jest to typ z pakietu PyBrain).

#Docelowe klasy
c = [0.1, 1.0] ← Dwa wejścia, jedno wyjście ← Powtarzany tysiąc razy losowy dobór czterech punktów danych i dodawanie ich do zbioru treningowego.

data_set = SupervisedDataSet(2, 1) # Dwa wejścia, jedno wyjście ← Powtarzany tysiąc razy losowy dobór czterech punktów danych i dodawanie ich do zbioru treningowego.

random.seed()
for i in xrange(1000):
    r = random.randint(0,3)
    data_set.addSample(d[r],c[r]) ← Tworzenie nowego obiektu do treningu z użyciem propagacji wstecznej. Obiekt łączony jest z siecią, zbiorem danych i współczynnikiem szybkości uczenia.

backprop_trainer = \
BackpropTrainer(net, data_set, ← Przeprowadzanie propagacji wstecznej 50 razy dla tego samego 1000 punktów danych. Po każdej iteracji wyświetlany jest błąd.
    learningrate=0.1) ← Przeprowadzanie propagacji wstecznej 50 razy dla tego samego 1000 punktów danych. Po każdej iteracji wyświetlany jest błąd.

for i in xrange(50):
    err = backprop_trainer.train()
    print "Iteracja %d, błąd: %.5f" % (i, err)
```

Z punktu 6.4.4 wiesz, że propagacja wsteczna bada przestrzeń wag, aby zminimalizować błąd między danymi wyjściowymi a oczekiwany wynikami. Każde wywołanie metody `train()` prowadzi do aktualizacji wag w taki sposób, by sieć neuronowa lepiej reprezentowała funkcję generującą dane. To oznacza, że w każdym wywołaniu tej

<sup>18</sup> Krótkie wprowadzenie do pakietu PyBrain, <http://pybrain.org/docs/index.html#quickstart>.

metody potrzebna będzie stosunkowo duża ilość danych (w przykładzie z funkcją XOR cztery punkty danych nie wystarczą). Aby rozwiązać ten problem, generujemy wiele punktów danych na podstawie rozkładu dla funkcji XOR i wykorzystujemy te punkty danych do treningu sieci z użyciem propagacji wstecznej. Liczba iteracji potrzebna do wyznaczenia globalnego minimum zależy od wielu czynników. Jednym z nich jest współczynnik szybkości uczenia. Określa on, jak szybko wagę są aktualizowane w każdej iteracji treningu. Mniejszy współczynnik wydłuża czas do konwergencji (czyli ustalenia globalnego minimum) i może doprowadzić do utknięcia na poziomie lokalnego optimum, jeśli płaszczyzna optymalizacji jest niewypukła. Większy współczynnik przyspiesza zmiany, ale grozi „przeskoczeniem” przez globalne minimum. Przyjrzymy się danym wygenerowanym przez kod z listingu 6.8 i posłużmy się nimi do zilustrowania tych dociekań:

```
Iteracja 0. error: 0.1824
Iteracja 1. błąd: 0.1399
Iteracja 2. błąd: 0.1384
Iteracja 3. błąd: 0.1406
Iteracja 4. błąd: 0.1264
Iteracja 5. błąd: 0.1333
Iteracja 6. błąd: 0.1398
Iteracja 7. błąd: 0.1374
Iteracja 8. błąd: 0.1317
Iteracja 9. błąd: 0.1332
...

```

Widać tu, że początkowo kolejne wywołania prowadzą do zmniejszenia błędu. Wiadomo, że problem ma przynajmniej jedno rozwiązanie, jednak propagacja wsteczna nie gwarantuje jego znalezienia. W niektórych sytuacjach błąd zmniejsza się do pewnego poziomu i nie da się uzyskać dalszej poprawy. Może się tak zdarzyć, gdy współczynnik uczenia jest zbyt niski i płaszczyzna optymalizacji jest niewypukła (występują na niej minima lokalne). Jeśli natomiast współczynnik uczenia jest zbyt wysoki, sieć może „przeskakiwać” wokół globalnego rozwiązania, a nawet oddalić się od tego obszaru i wpaść w region minimum lokalnego albo „przeskakiwać” między nieoptymalnymi (lokalnymi) rozwiązaniami. W obu sytuacjach efekt jest taki sam — sieć nie znajduje minimum globalnego.

Ponieważ wyniki zależą od początkowych wartości wag, nie potrafimy stwierdzić, czy w Twoim przypadku kod szybko uzyska konwergencję. Spróbuj uruchomić rozwiązanie kilkakrotnie. Spróbuj też eksperymentować ze współczynnikiem szybkości uczenia z listingu 6.8. Jak wysoki może być ten współczynnik, by algorytm nie utykał w większości sytuacji w rozwiązaniach lokalnych? W praktyce dobrą współpracą zawsze wymaga kompromisu między ryzykiem znalezienia nieoptymalnych rozwiązań a szybkością. Dlatego staraj się wybrać najwyższy współczynnik, który daje poprawny wynik. Eksperymentuj z tą wartością do czasu uzyskania sieci, która uzyska konwergencję z błędem równym zero.

#### **6.4.6. Perceptron wielowarstwowy po zakończeniu nauki**

W poprzednim przykładzie zbudowaliśmy perceptron wielowarstwowy za pomocą pakietu PyBrain i przeprowadziliśmy jego trening, tak by działał jak funkcja XOR. Jeśli udało Ci się uzyskać błąd równy zero, powinieneś móc wykonywać zadania z tego

punktu za pomocą własnego modelu! Zacznijmy od sprawdzenia modelu i pobrania wag sieci odpowiadającej rozwiązaniu z rysunku 6.12. Listing 6.9 pokazuje, jak to zrobić.

#### **Listing 6.9. Pobieranie wag sieci neuronowej po treningu**

```
#print net.params
print "[w(x_1,j=1),w(x_2,j=1),w(x_1,j=2),w(x_2,j=2)]: " + str(in_to_h.params)
print "[w(j=1,j=3),w(j=2,j=3)]: "+str(h_to_out.params)
print "[w(x_b,j=1),w(x_b,j=2)]: "+str(bias_to_h.params)
print "[w(x_b,j=3)]: "+str(bias_to_out.params)

> [w(x_1,j=1),w(x_2,j=1),w(x_1,j=2),w(x_2,j=2)]: [-2.32590226 2.25416963
-2.74926055 2.64570441]
> [w(j=1,j=3),w(j=2,j=3)]: [-2.57370943 2.66864851]
> [w(x_b,j=1),w(x_b,j=2)]: [ 1.29021983 -1.82249033]
> [w(x_b,j=3)]: [ 1.6469595]
```

Dane wyjściowe uzyskane na podstawie kodu z listingu 6.9 to wyniki przeprowadzonego przez nas treningu neuronu. Ty możesz uzyskać inne dane. Ważne jest, by sieć działała w poprawny sposób. Możesz dowiedzieć się, czy tak działa, aktywując sieć dla danych wejściowych i sprawdzając, czy dane wyjściowe są zgodne z oczekiwaniami. Przyjrzyj się listingowi 6.10.

#### **Listing 6.10. Aktywowanie sieci neuronowej**

```
print "Aktywowanie dla wartości 0.0. Wynik: " + str(net.activate([0.0]))
print "Aktywowanie dla wartości 0.1. Wynik: " + str(net.activate([0.1]))
print "Aktywowanie dla wartości 1.0. Wynik: " + str(net.activate([1.0]))
print "Aktywowanie dla wartości 1.1. Wynik: " + str(net.activate([1.1]))

> Aktywowanie dla wartości 0.0. Wynik: [ -1.33226763e-15]
> Aktywowanie dla wartości 0.1. Wynik: [ 1.]
> Aktywowanie dla wartości 1.0. Wynik: [ 1.]
> Aktywowanie dla wartości 1.1. Wynik: [ 1.55431223e-15]
```

Widać tu, że dane wyjściowe z sieci po treningu są bardzo bliskie 1 dla tych wzorców, które powinny dawać wynik pozytywny. Ponadto dane wyjściowe są bardzo bliskie 0 dla tych wzorców, które powinny dawać wynik negatywny. Ogólnie dla pozytywnych próbek testowych sieć powinna zwracać wyniki większe niż 0,5, a dla próbek negatywnych — wyniki mniejsze niż 0,5. Aby się upewnić, że w pełni rozumiesz tę sieć, spróbuj zmodyfikować dane wejściowe i prześledzić ich przetwarzanie przez sieć za pomocą arkusza kalkulacyjnego dostępnego w materiałach do książki.

## **6.5. Zwiększenie głębokości — od wielowarstwowych sieci neuronowych do uczenia głębokiego**

W wielu obszarach badań postępy są uzyskiwane zrywami. Przez pewien czas w danej dziedzinie nic się nie dzieje, po czym nagle, zwykle w wyniku pewnych odkryć lub osiągnięć, następuje okres gorączkowej aktywności. W obszarze sieci neuronowych jest podobnie. Mamy szczęście pracować w okresie ekscytujących dokonań, z których

większość związana jest z uczeniem głębokim. Przedstawiamy tu kilka z tych osiągnięć, a następnie prezentujemy najprostszy możliwy przykład sieci głębokiej. Dlaczego sieci neuronowe znów stały się modne? Wynika to ze szczęśliwego zbiegu wielu okoliczności.

Po pierwsze, dostępnych jest więcej danych niż kiedykolwiek wcześniej. Duże firmy internetowe mają dostęp do ogromnych repozytoriów danych graficznych, które można wykorzystać w ciekawy sposób. Może słyszałeś o opublikowanej przez firmę Google w 2012 roku pracy, w której opisano trening dziewięciowarstwowej sieci na podstawie 10 milionów obrazów pobranych z internetu<sup>19</sup>. Celem było rozpoznawanie pośrednich reprezentacji bez etykiet. Okazało się, że najczęściej publikowane były zdjęcia pyszczków kotów. W pewnym stopniu stanowi to argument na rzecz hipotezy, że większa liczba danych pozwala uzyskać lepsze wyniki niż bardziej zaawansowane algorytmy<sup>20</sup>. Jeszcze kilka lat wcześniej uzyskanie podobnego rozwiązania byłoby niemożliwe.

Po drugie, dokonał się duży postęp w zakresie wiedzy teoretycznej. Dopiero niedawne dokonania Geoffreya Hintona i jego współpracowników pozwoliły społeczności zrozumieć, że sieci głębokie można skutecznie trenować, traktując każdą warstwę jak ograniczoną maszynę Boltzmanna (ang. *Restricted Boltzmann Machine* — **RBM**)<sup>21, 22</sup>. Wiele sieci wykorzystujących uczenie głębokie jest obecnie budowanych z warstw maszyn RBM (więcej na ten temat dowiesz się dalej). Yann Le Cun, Yoshua Bengio i inni dokonali wielu dalszych teoretycznych postępów w tej dziedzinie. Zachęcam do zapoznania się z przeglądem ich prac, co pozwoli lepiej zrozumieć ten obszar<sup>23</sup>.

### **6.5.1. Ograniczone maszyny Boltzmanna**

W tym punkcie omawiamy maszyny RBM, a konkretnie jedną z ich odmian opartą na modelu Bernoulliego (ang. *Bernoulli Restricted Boltzmann Machines* — **BRBM**). Dalej wyjaśniamy, dlaczego jest to wyjątkowa wersja maszyn RBM. Maszyny RBM są wspominane w kontekście uczenia głębokiego, ponieważ dobrze uczą się cech. Dlatego można je wykorzystać w głębszych sieciach do nauki reprezentacji cech. Dane wyjściowe z takich maszyn mogą posłużyć jako dane wejściowe dla kolejnych maszyn RBM lub dla perceptronu wielowarstwowego. Wróć do podrozdziału 6.1 i przypomnij sobie przykład z wykrywaniem samochodów. Poświęciliśmy już trochę czasu na ogólne omówienie perceptronów wielowarstwowych. Teraz pora przyjrzeć się uczeniu głębokiemu z automatycznym wykrywaniem cech.

<sup>19</sup>Quoc V. Le i współpracownicy, *Building High-Level Features Using Large Scale Unsupervised Learning*, „ICML 2012: 29th International Conference on Machine Learning” (ICML, 2012): 1.

<sup>20</sup>Pedro Domingos, *A Few Useful Things to Know about Machine Learning*, „Communications of the ACM” (1 października 2012): 78 – 87.

<sup>21</sup>Miguel A. Carreira-Perpiñán i Geoffrey Hinton, *On Contrastive Divergence Learning*, „Society for Artificial Intelligence and Statistics” (2005): 33 – 40.

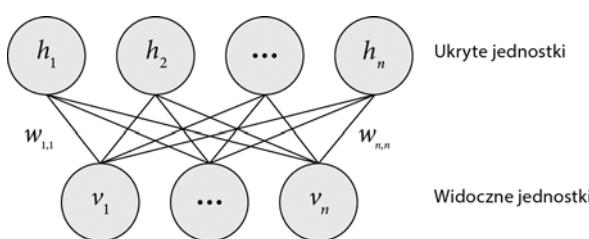
<sup>22</sup>G.E. Hinton i R.R. Salakhutdinov, *Reducing the Dimensionality of Data with Neural Networks*, „Science” (28 lipca 2006): 504 – 507.

<sup>23</sup>Yann LeCun, Yoshua Bengio i Geoffrey Hinton, *Deep Learning*, „Nature” 521 (maj 2015): 436 – 44.

W tym celu posłużymy się przykładem z dokumentacji pakietu scikit-learn<sup>24</sup>. W tym przykładzie maszyna BRBM jest używana do wyodrębnienia cech z dostępnego we wspomnianym pakiecie zbioru danych z cyframi. Następnie za pomocą regresji logistycznej dane są klasyfikowane z użyciem wyuczonych cech. Po przedstawieniu tego przykładu krótko opisujemy, jak budować głębsze sieci i wkroczyć w rozwijający świat uczenia głębokiego. Zanim jednak zaczniemy, powinniśmy się upewnić, że rozumiesz podstawy. Czym są maszyny BRBM?

### 6.5.2. Maszyny RBM

Na ogólnym poziomie maszyna RBM to graf dwudzielny, w którym węzły z każdego zbioru są połączone z wszystkimi węzłami z drugiego zbioru. „Ograniczoność” tych maszyn związana jest z tym, że widoczne węzły mogą być powiązane tylko z ukrytymi i na odwrót. W maszynach BRBM dodatkowym ograniczeniem jest to, że każdy węzeł musi być binarny. Na rysunku 6.13 maszyna RBM jest pokazana w formie graficznej.



**Rysunek 6.13.** Graficzna ilustracja maszyny RBM. Taka maszyna to graf dwudzielny obejmujący jednostki ukryte i widoczne. Każdy element z obu tych zbiorów jest powiązany z wszystkimi elementami z drugiego zbioru. Litera  $h$  oznacza wektor obejmujący jednostki ukryte, a litera  $v$  – wektor zawierający jednostki widoczne. Zauważ, że z każdym węzłem powiązana może być waga błędu systematycznego. Dla prostoty pomijamy tu te wagi

Dobrze, ale do czego może nam się to przydać? Wskazówką mogą być tu nazwy. Węzły widoczne reprezentują coś, co można zaobserwować. Może być to coś, czego dotyczy trening. Węzły ukryte reprezentują coś niewidocznego, o nieznanym lub niejawnym znaczeniu. Przypomnij sobie zmienne ukryte z rozdziału 3., w którym generowaliśmy rekommendacje. Pod wieloma względami są one podobne do jednostek ukrytych.

Dalej zapoznasz się z konkretnym przykładem maszyny RBM służącej do rozpoznawania obrazów, jednak w trakcie omawiania teorii warto myśleć o jakiejś docelowej aplikacji. Dlatego ukryte węzły możesz traktować jak gatunki, a widoczne — jak filmy lubiane przez daną osobę (jest to nawiązanie do rozdziału 3.). Widoczne węzły mogą też odpowiadać piosenkom, malarzom itd. Ważne jest to, że ukryte węzły ujmują niejawne pogrupowanie danych. Tu źródłem grup są preferencje użytkowników. Liczba widocznych jednostek jest zależna od problemu. W klasyfikacji binarnej używane są dwa węzły. Na potrzeby rekommendacji filmów może to być liczba filmów w zbiorze danych. Wzrost liczby ukrytych wartości zwiększa możliwość zamodelowania złożonych relacji za pomocą maszyny RBM, ale dzieje się to kosztem nadmiernego dopasowania

<sup>24</sup> scikit-learn, *Restricted Boltzmann Machine Features for Digit Classification*, <http://mng.bz/3N42>.

modelu do danych. Hinton<sup>25</sup> zaproponował sposób na wybór liczby jednostek ukrytych w zależności od złożoności danych i liczby próbek treningowych. Podobnie jak w perceptronie wielowarstwowym z logistyczną funkcją aktywacji prawdopodobieństwo aktywacji danej jednostki widocznej na podstawie wartości zmiennych ukrytych wynosi:

$$P(v_i = 1 | \mathbf{h}, \mathbf{W}) = \sigma\left(\sum_j w_{j,i} h_j + b_i\right)$$

W tym wzorze  $\sigma$  to funkcja logistyczna:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Wartości  $b_i$  to błąd systematyczny powiązany z danym widocznym węzłem. Można ująć to prosto: prawdopodobieństwo aktywacji węzła jest równa sumie iloczynów wag i wartości węzłów ukrytych (plus błąd systematyczny) przekazanej do funkcji logistycznej. Prawdopodobieństwa w warstwie ukrytej są wyrażone w podobny sposób:

$$P(h_j = 1 | \mathbf{v}, \mathbf{W}) = \sigma\left(\sum_i w_{j,i} v_i + c_j\right)$$

Tu  $c_j$  to błąd systematyczny powiązany z węzłem ukrytym  $h_j$ . Zastanów się nad błędem systematycznym. Wpływa on na prawdopodobieństwo aktywacji węzłów ukrytych i widocznych niezależnie od informacji z danych połączeń. Pod wieloma względami można go uznać za punkt wyjścia (ang. *prior*). W maszynie RBM używanej do rekommendowania filmów lub gatunków błąd systematyczny w węźle ukrytym może określać wyjściowe prawdopodobieństwo, że dany film należy do określonego gatunku. W węźle widocznym błąd systematyczny może być powiązany z wyjściowym prawdopodobieństwem, że dany film spodoba się określonej osobie niezależnie od preferowanych przez nią gatunków.

W praktyce maszyny RBM wymagają nauki — muszą nauczyć się macierzy wag  $\mathbf{W}$  na podstawie przypadków treningowych. Te przypadki treningowe mają postać wektorów stanu węzłów widocznych.

Przeprowadźmy krótki eksperyment myślowy, aby spróbować zrozumieć, jak przebiega proces uczenia. Załóżmy, że węzły ukryte reprezentują gatunki, a węzły widoczne odpowiadają filmom. Wtedy waga łącząca węzeł ukryty z widocznym powinna być wyższa, gdy określony film należy do danego gatunku. Natomiast gdy film nie należy do danego gatunku, waga powinna być niższa (możliwe, że ujemna). Aby zrozumieć, dlaczego się to sprawdza, zastanów się nad maszyną RBM z jednym węzłem ukrytym. Załóżmy, że węzeł ukryty reprezentuje gatunek „filmy akcji”, a filmem jest *Top Gun*. Gdy aktywowany jest węzeł ukryty, jedynym sposobem na zwiększenie prawdopodobieństwa aktywacji węzła widocznego jest wysoka waga łącząca oba węzły. Podobnie

---

<sup>25</sup> Geoffrey Hinton, *A Practical Guide to Training Restricted Boltzmann Machines* w: „Neural Networks: Tricks of the Trade”, red. Grégoire Montavon, G.B. Orr i K.R. Müller (University of Toronto, 2012): 599 – 619.

jeśli aktywowany jest widoczny węzeł z filmem *Top Gun*, jedyny sposób na zwiększenie prawdopodobieństwa aktywowania odpowiedniego gatunku to wysoka waga między filmem a gatunkiem.

Sieci RBM można trenować za pomocą uczenia opartego na energii (ang. *energy-based learning*)<sup>26</sup>. W trakcie treningu maksymalizowana jest zgodność między powiązanymi ze sobą węzłami ukrytymi i widocznymi. Używana jest tu miara o nazwie *energia*. Ta wartość spada, gdy zgodność między węzłami ukrytymi i widocznymi jest większa. Tak więc zmniejszenie energii,  $E$ , skutkuje lepszą konfiguracją sieci na podstawie danych treningowych:

$$E(\mathbf{v}, \mathbf{h}, \mathbf{W}) = -\left( \sum_i \sum_j w_{ij} v_i h_j + \sum_i b_i v_i + \sum_j c_j h_j \right)$$

Z tego wynika, że celem treningu jest minimalizacja wartości  $E(\mathbf{v}, \mathbf{h}, \mathbf{W})$  na podstawie przykładów treningowych. Po zakończeniu treningu można sprawdzić przestrzeń ukrytą pod kątem najbardziej prawdopodobnego wektora, gdy ustwiony jest określony wektor widoczny, a także najbardziej prawdopodobnego wektora widocznego dla określonej konfiguracji węzłów w przestrzeni ukrytej. W aplikacji przetwarzającej filmy i gatunki te operacje odpowiadają określaniu zainteresowania gatunkiem na podstawie listy filmów oraz rekomendacjom filmów na podstawie przestrzeni gatunków.

Wynikowa energia jest przekształcana na prawdopodobieństwo, po czym odpowiedź jest szukana w przestrzeni prawdopodobieństwa. LeCun wykazał, że prawdopodobieństwo można uzyskać za pomocą miary Gibbsa<sup>27</sup>. Na przykład prawdopodobieństwo dla widocznego wektora na podstawie wyuczonych wag i wektora ukrytego wynosi:

$$P(\mathbf{v}|\mathbf{h}, \mathbf{W}) = \frac{e^{-E(\mathbf{v}, \mathbf{h}, \mathbf{W})}}{\sum_{\mathbf{v} \in V} e^{-E(\mathbf{v}, \mathbf{h}, \mathbf{W})}}$$

Natomiast prawdopodobieństwo dla ukrytego wektora na podstawie wyuczonych wag i widocznego wektora to:

$$P(\mathbf{v}|\mathbf{h}, \mathbf{W}) = \frac{e^{-E(\mathbf{v}, \mathbf{h}, \mathbf{W})}}{\sum_{\mathbf{h} \in H} e^{-E(\mathbf{v}, \mathbf{h}, \mathbf{W})}}$$

Przyjrzyj się uważnie obu tym wzorom. Liczba w liczniku jest większa, gdy konfiguracje widocznych i ukrytych węzłów są ze sobą zgodne. Mianownik służy do normalizacji wartości z uwzględnieniem wszystkich możliwych stanów, co pozwala uzyskać wynik z przedziału od 0 do 1 (czyli prawdopodobieństwo).

Do tej pory wszystko wygląda dobrze. Jednak nie wyjaśniliśmy jeszcze, jak przebiega uczenie macierzy wag. Wydaje się, że zadanie jest trudne, ponieważ nie wiadomo

<sup>26</sup>Yann LeCun i współpracownicy, *Energy-Based Models in Document Recognition and Computer Vision, International Conference on Document Analysis and Recognition* (2007).

<sup>27</sup>Yann LeCun i współpracownicy, *A Tutorial on Energy-Based Learning*, w: „Predicting Structured Data”, red. G. Bakir i współpracownicy (MIT Press, 2006).

nic o węzłach ukrytych. Znane są tylko węzły widoczne. Pamiętaj, że zbiór treningowy obejmuje wyłącznie listę węzłów widocznych. W wielu sytuacjach trening maszyn RBM przebiega z użyciem metody dywergencji kontrastowej (ang. *contrastive divergence*), której autorstwo przypisywane jest Hintonowi<sup>28</sup>. Nie omawiamy tu tego algorytmu w całości, ponieważ znajdziesz go we wspomnianym źródle, prezentujemy jednak zarys rozwiązania.

Metoda dywergencji kontrastowej to technika przybliżania prawdopodobieństwa, w której wykonywana jest seria etapów próbkowania. Algorytm iteracyjnie przetwarza przykłady treningowe i wykonuje próbkowanie wstępne (stanów ukrytych na podstawie stanów widocznych) oraz do przodu (stanów widocznych na podstawie ukrytych). Wagi są inicjowane losowymi wartościami. Dla każdego wektora treningowego (węzłów widocznych) węzły ukryte są aktywowane z wcześniej określonym prawdopodobieństwem, po czym mierzony jest poziom zgodności. Następnie węzły ukryte są używane do aktywowania węzłów widocznych i ponownie dokonywany jest pomiar zgodności. Miary zgodności są potem łączone, a wagi zostają zmodyfikowane w kierunku, który prowadzi do spadku energii w całej sieci. Więcej szczegółów znajdziesz w pracy Hintona.

### 6.5.3. Maszyny RBM w praktyce

W tym punkcie pracujemy nad zmodyfikowaną wersją problemu klasyfikacji z użyciem regresji logistycznej zaprezentowanego w dokumentacji pakietu scikit-learn<sup>29</sup>. Na uznanie za ten obrazowy przykład zasługują Dauphin, Niculea i Synnaeve. Nie odbiegamy tu zanadto od zaprezentowanego przez nich materiału. Tak jak w poprzednich przykładach, tak i na listingu 6.11 pomijamy blok z instrukcjami importu i koncentrujemy się na kodzie. Kompletny kod znajdziesz w materiałach dostępnych w internecie.

#### Listing 6.11. Tworzenie zbioru danych

```
digits = datasets.load_digits()
X = np.asarray(digits.data, 'float32')
X, Y = nudge_dataset(X, digits.target)
X = (X - np.min(X, 0)) / (np.max(X, 0) + 0.0001) # Skalowanie do przedziału 0 – 1
X_train, X_test, Y_train, Y_test = train_test_split(X,
    Y, test_size=0.2, random_state=0)
```

Pierwszą operacją jest wczytywanie zbioru danych, przy czym tu robimy coś jeszcze. Na podstawie pierwotnego zbioru danych generujemy sztuczne próbki, wprowadzając w zbiorze danych liniowe przesunięcia o jeden piksel, znormalizowane w taki sposób, aby wartość każdego piksela znajdowała się w przedziale od 0 do 1. Tak więc na podstawie każdego opatrzonego etykietą rysunku generowane są cztery dodatkowe (z przesunięciem w góre, w dół, w prawo i w lewo) o tej samej etykiecie — cyfry reprezentowanej przez dany rysunek. Umożliwia to lepszą naukę reprezentacji danych (bez nadmiernej zależności od centralnego zlokalizowania cyfry) za pomocą tak małego zbioru.

<sup>28</sup>Geoffrey Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*, wersja 1, raport wewnętrzny, UTML TR 2010-003 (2 sierpnia 2010).

<sup>29</sup>scikit-learn, *Restricted Boltzmann Machine Features for Digit Classification*, <http://mng.bz/3N42>.

Za generowanie nowych rysunków odpowiada funkcja nudge\_dataset zdefiniowana na listingu 6.12.

#### Listing 6.12. Generowanie sztucznych danych

```
def nudge_dataset(X, Y):
    """
    Ten kod generuje zbiór danych pięciokrotnie większy od pierwotnego,
    przesuwając rysunki o wymiarach 8x8 z X o 1 piksel w lewo, w prawo, do góry i w dół.
    """
    direction_vectors = [[[0, 1, 0],[0, 0, 0],[0, 0, 0]],
                          [[0, 0, 0],[1, 0, 0],[0, 0, 0]],
                          [[0, 0, 0],[0, 0, 1],[0, 0, 0]],
                          [[0, 0, 0],[0, 0, 0],[0, 1, 0]]]
    shift = \
        lambda x, w: convolve(x.reshape((8, 8)), mode='constant', \
                               weights=w).ravel()
    X = np.concatenate([X] + \
        [np.apply_along_axis(shift, 1, X, vector) for vector in \
         direction_vectors])
    Y = np.concatenate([Y for _ in range(5)], axis=0)
    return X, Y
```

Na podstawie tych danych można łatwo utworzyć potok podejmowania decyzji obejmujący maszynę RBM i regresję logistyczną. Na listingu 6.13 przedstawiony jest kod, który konfiguruje ten model i przeprowadza jego trening.

#### Listing 6.13. Konfigurowanie i trening potoku obejmującego maszynę RBM i regresję logistyczną

```
# Używane modele
logistic = linear_model.LogisticRegression()
rbm = BernoulliRBM(random_state=0, verbose=True)

classifier = Pipeline(steps=[('rbm', rbm), ('logistic', logistic)])
#####
# Trening

# Hiperparametry. Zostały ustawione w wyniku walidacji krzyżowej z użyciem
# metody GridSearchCV. Tu pomijamy walidację krzyżową, aby przyspieszyć
# pracę.
rbm.learning_rate = 0.06
rbm.n_iter = 20
# Większa liczba komponentów zwykle pozwala zwiększyć skuteczność prognoz, ale
# kosztem wydłużenia czasu dopasowywania modelu do danych
rbm.n_components = 100
logistic.C = 6000.0

# Trening potoku obejmującego maszynę RBM i regresję logistyczną
classifier.fit(X_train, Y_train)

# Trening z użyciem regresji logistycznej
logistic_classifier = linear_model.LogisticRegression(C=100.0)
logistic_classifier.fit(X_train, Y_train)
```

Ten kod pochodzi bezpośrednio z dokumentacji pakietu scikit-learn<sup>30</sup>. Warto zwrócić tu uwagę na kilka istotnych aspektów. Hiperparametry (czyli parametry maszyny RBM) zostały dobrane specjalnie do używanego zbioru danych, abytrzymać obrazowy przykład. Więcej szczegółów znajdziesz we wspomnianej dokumentacji.

Ten kod robi niewiele oprócz ustawiania parametrów. Przygotowywany jest potok obejmujący maszynę RBM i klasyfikator oparty na regresji logistycznej. Dla porównania używany jest też niezależny klasyfikator oparty na regresji logistycznej. Niedługo zapoznasz się ze skutecznością obu rozwiązań. Na listingu 6.14 znajduje się kod, który to umożliwia.

#### **Listing 6.14. Ocena potoku obejmującego maszynę RBM i regresję logistyczną**

```
print("Regresja logistyczna z cechami z maszyny RBM:\n%s\n" % (
    metrics.classification_report(
        Y_test,
        classifier.predict(X_test))))
```

```
print("Regresja logistyczna dla cech opartych na samych pikselach:\n%s\n" % (
    metrics.classification_report(
        Y_test,
        logistic_classifier.predict(X_test))))
```

Dane wyjściowe z tego kodu zawierają szczegółowe podsumowanie obu podejść. Powinieneś zobaczyć, że potok obejmujący maszynę RBM i regresję logistyczną ma znaczną przewagę nad samą regresją logistyczną w zakresie precyzyji, czułości i wskaźnika F1-score. Dlaczego tak się dzieje? Jeśli narysujesz ukryte komponenty maszyny RBM, powinieneś lepiej to zrozumieć. Na listingu 6.15 przedstawiony jest potrzebny do tego kod, a rysunek 6.14 zawiera graficzną ilustrację ukrytych komponentów z maszyny RBM.

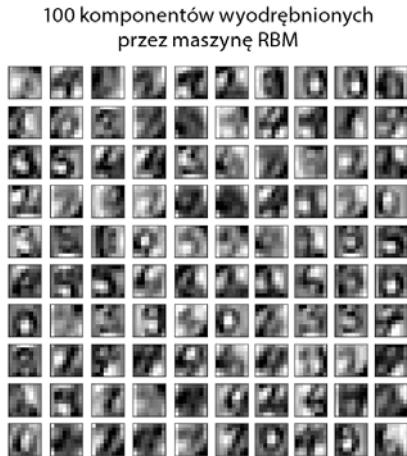
#### **Listing 6.15. Generowanie graficznej reprezentacji ukrytych jednostek**

```
plt.figure(figsize=(4.2, 4))
for i, comp in enumerate(rbm.components_):
    #print(i)
    #print(comp)
    plt.subplot(10, 10, i + 1)
    plt.imshow(comp.reshape((8, 8)),
               cmap=plt.cm.gray_r, interpolation='nearest')
    plt.xticks(())
    plt.yticks(())

plt.suptitle('100 komponentów wyodrębnionych przez maszynę RBM', fontsize=16)
plt.subplots_adjust(0.08, 0.02, 0.92, 0.85, 0.08, 0.23)
plt.show()
```

W modelu RBM występuje 100 węzłów ukrytych i 64 węzły widoczne (wynika to z wielkości używanych rysunków). Każdy kwadrat na rysunku 6.14 to przedstawiona za pomocą skali szarości interpretacja wag łączących poszczególne komponenty ukryte

<sup>30</sup> scikit-learn, *Restricted Boltzmann Machine Features for Digit Classification*, <http://mng.bz/3N42>.



**Rysunek 6.14.** Graficzna reprezentacja wag łączących jednostki ukryte i widoczne w maszynie RBM. Każdy kwadrat reprezentuje jednostkę ukrytą, a 64 wartości na skali szarości w każdej jednostce reprezentują wagi łączące daną jednostkę ukrytą z wszystkimi jednostkami widocznymi. W pewnym sensie ilustruje to, jak dobrze dana zmienna ukryta pomaga rozpoznawać obrazy podobne do pokazanych

z każdą widoczną jednostką. Każdy komponent ukryty można traktować jak jednostkę rozpoznającą widoczny obraz. W potoku model oparty na regresji logistycznej wykorzystuje jako wejście 100 prawdopodobieństw aktywacji ( $P(h_j=1 | v = \text{obraz})$ ) dla każdego  $j$ ). Dlatego zamiast przeprowadzać regresję logistyczną dla 64 nieprzetworzonych pikseli, wykonujemy ją dla 100 wejść, z których każde przyjmuje wysoką wartość, gdy dane wejściowe są podobne do tych pokazanych na rysunku 6.14. Gdy przypomnisz sobie pierwszy punkt rozdziału, zobaczysz, że utworzyliśmy sieć, która automatycznie nauczyła się pośrednich reprezentacji liczb z wykorzystaniem maszyny RBM. Utworzyliśmy więc jedną warstwę sieci głębokiej. Wyobraź sobie, co można osiągnąć za pomocą głębszych sieci i wielu warstw maszyn RBM generujących więcej pośrednich reprezentacji!

## 6.6. Podsumowanie

- Przedstawiliśmy krótki przegląd sieci neuronowych i ich związków z uczeniem głębokim. Zaczeliśmy od najprostszego modelu sieci neuronowych (MCP), po czym przeszliśmy do perceptronu i omówiliśmy jego powiązanie z regresją logistyczną.
- Odkryliśmy, że nie da się przedstawić funkcji nieliniowych za pomocą jednego perceptronu, natomiast można wykorzystać do tego perceptrony wielowarstwowe.
- Omówiliśmy trening perceptronów wielowarstwowych z użyciem propagacji wstecznej i różniczkowalnych funkcji aktywacji. Przedstawiliśmy też, jak posłużyć się pakietem PyBrain do nauczenia sieci działania funkcji nieliniowej za pomocą propagacji wstecznej.
- Opisaliśmy też nowe dokonania z dziedziny uczenia głębokiego: budowanie sieci wielowarstwowych, które potrafią uczyć się pośrednich reprezentacji danych.
- Skoncentrowaliśmy się na jednej sieci z tego obszaru, na maszynach RBM, i pokazaliśmy, jak zbudować prostą sieć głęboką dla zbioru danych reprezentujących cyfry. Użyliśmy jednej maszyny RBM i klasyfikatora opartego na regresji logistycznej.



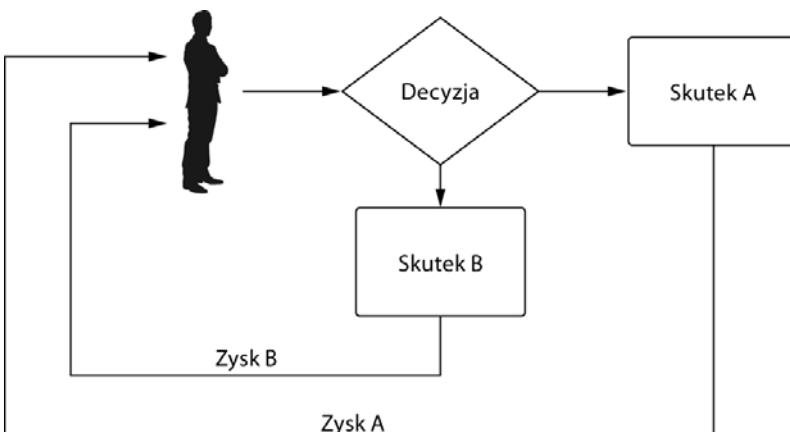
# *Dokonywanie właściwego wyboru*

## **Zawartość rozdziału:**

- Testy A/B
- Komplikacje związane z dokonywaniem właściwego wyboru
- Wieloręki bandyta

Gdy dostępnych jest wiele możliwości, to jak wybrać tę, która pozwala zmaksymalizować zyski (lub zminimalizować koszty)? Założmy, że możesz wybrać jedną z dwóch dróg do pracy. Jak wybrać tę, która pozwoli zminimalizować czas dojazdu? W tym przykładzie zysk związany jest z czasem dojazdu do pracy. Można jednak uwzględnić także koszt paliwa lub czas spędzony w korku.

Za pomocą technik opisanych w tym rozdziale można zoptymalizować dowolny proces, w którym opcja jest testowana wielokrotnie i za każdym razem, gdy zostanie wybrana, zwracany jest zysk. We wspomnianym przykładzie każdego dnia wybierana jest droga do pracy, a w dzienniku rejestrowany jest czas dojazdu. Z czasem pracownik może wykryć wzorce w danych (droga A zajmuje mniej czasu niż droga B) i zdecydować się na regularny wybór korzystniejszej opcji. Jaka jest właściwa strategia? Regularne wybieranie drogi A lub drogi B? Kiedy ilość danych jest wystarczająca do wyboru najlepszej trasy? Jak wygląda optymalna strategia testowania dróg? W tym rozdziale koncentrujemy się na takich pytaniach. Na rysunku 7.1 w formie graficznej przedstawiona jest definicja omawianego problemu.



**Rysunek 7.1.** Formalne ujęcie definicji problemu. Tu użytkownik musi podjąć decyzję prowadzącą do określonych skutków. Ze skutkami powiązane są zyski, o których użytkownik jest informowany. Użytkownik musi wielokrotnie podejmować tę decyzję. Jak opracować strategię pozwalającą zmaksymalizować zysk (lub zminimalizować koszt)?

Zakładamy, że użytkownik z rysunku wielokrotnie znajduje się w sytuacji, w której musi podjąć decyzję. Za każdym razem osiągany jest pewien skutek powiązany z zyskiem. Jeśli pracownik dojeżdża do pracy każdego dnia, codziennie musi wybierać trasę. Skutek to kombinacja wszystkich czynników związanych z drogą, a na ich podstawie określany jest zysk.

Choć nasz fikcyjny przykład doskonale pasuje do omawianego problemu, w inteligentnej sieci występuje wiele podobnych sytuacji. Dostępne są liczne aplikacje, w których możliwy jest szereg wyborów i ważne jest podjęcie właściwej decyzji:

- *Optymalizacja strony wejściowej.* Pojawienie się inteligentnej sieci sprawiło, że więcej sklepów prowadzi działalność tylko w internecie. W takich sklepach niezwykle istotna jest maksymalizacja współczynnika konwersji (odsetka osób, które po otwarciu witryny kupują produkty lub odwiedzają dalsze strony). Podejmowana decyzja dotyczy formatu i zawartości strony wejściowej (można wybierać spośród trzech lub czterech wersji). Celem jest wybór najlepszej strony wejściowej, która przyciąga użytkowników i maksymalizuje prawdopodobieństwo interakcji lub zakupu.
- *Optymalizacja materiałów reklamowych.* Z wcześniejszych rozdziałów wiesz, że reklama internetowa związana jest z licznymi wyzwaniami, z którymi można poradzić sobie za pomocą technik uczenia maszynowego. Jednym z ciekawych wyzwań jest wybór formatu lub określonych aspektów reklamy. Gdy już stwierdzisz, że chcesz pokazać reklamę, i ustalisz cenę jej emisji, musisz określić, co wyświetlisz w zakupionej przestrzeni. Możesz przeprowadzić testy z uwzględnieniem różnych decyzji. Wybór odpowiedniej kombinacji aspektów może prowadzić do powstania reklamy dającej lepsze wyniki niż inne kombinacje.

## 7.1. Testy A/B

Jak oceniać decyzje i jaką strategię należy przyjąć przy testowaniu skutków? Jedną z używanych technik są testy A/B. W branży testy A/B zyskały popularność w ostatnich latach, jednak osoby znające statystykę rozpoznają je jako prosty eksperyment z dwoma grupami. Przyjrzyjmy się bliżej tej technice, aby zrozumieć jej specyfikę.

W testach A/B badane są dwie grupy: A i B. Pierwsza z nich to grupa kontrolna, natomiast w drugiej zmieniany jest pewien czynnik. Na przykład w trakcie optymalizowania strony wejściowej grupie A pokazywana może być obecna strona wejściowa, a grupie B — nowa strona wejściowa ze zmienioną zawartością lub innym układem. Celem testów A/B jest ustalenie, czy nowy układ prowadzi do istotnej statystycznie zmiany współczynnika konwersji.

Warto zauważyć, że przypisywanie użytkowników do grup wymaga zastanowienia. W testach A/B dobór użytkowników do grup odbywa się losowo. Dlatego dla dużych zbiorów osób przekrój populacji w obu grupach powinien być taki sam (grupy nie powinny różnić się błędem systematycznym). Uważaj jednak, ponieważ gdy zbiór użytkowników jest niewielki, nie można przyjąć takiego założenia, dlatego należy rozważyć zastosowanie innych modeli eksperymentów<sup>1</sup>.

### 7.1.1. Teoria

Po utworzeniu dwóch dużych grup użytkowników — różniących się tylko stroną wejściową — należy sprawdzić, czy występuje istotna statystycznie różnica między współczynnikami konwersji w obu grupach. Dzięki dużej próbce można posłużyć się jednoznakowym testem z dla dwóch grup, jednak dla mniejszych próbek potrzebny może być test t<sup>2</sup>.

W teście z przyjmowane jest założenie, że dane mają rozkład normalny, a próbki są pobierane losowo. Celem jest sprawdzenie, czy między zbiorami testowym (grupa B) a kontrolnym (grupy A) występuje istotna statystycznie różnica. Jak przeprowadzany jest ten test?

Załóżmy, że grupy A i B zawierają po 5000 próbek. Należy sformułować matematycznie hipotezę zerową, zgodnie z którą między grupami nie występuje istotna statystycznie różnica współczynników konwersji dla populacji, i hipotezę alternatywną, zgodnie z którą taka różnica występuje.

Współczynnik konwersji dla próbki można uznać za zmienną losową o rozkładzie normalnym. Oznacza to, że współczynnik konwersji dla próbki to obserwacja pochodząca z rozkładu normalnego współczynników konwersji. Aby to zrozumieć, uwzględnij, że wiele eksperymentów przeprowadzonych na tej samej grupie może dawać nieco odmienne współczynniki konwersji. Za każdym razem, gdy pobierasz próbkę z grupy, uzyskujesz szacunkową wartość współczynnika konwersji dla populacji. Dotyczy to obu grup z testu A/B. Można więc utworzyć nową zmienną losową, także o rozkładzie

<sup>1</sup> Stuart Wilson i Rory MacLean, *Research Methods and Data Analysis for Psychology* (McGraw-Hill Education — Europe, 2011).

<sup>2</sup> Student, *The Probable Error of a Mean*, „Biometrika” 6, nr 1, red: E.S. Pearson i John Wishart (marzec 1908): 1 – 25.

normalnym, która stanowi połączenie zmiennych losowych z grup A i B. W ten sposób uzyskujemy rozkład różnic. Tę nową zmienną losową możemy nazwać  $X$  i zdefiniować w następujący sposób:

$$X = X_e - X_n$$

$X_e$  to zmienna losowa określająca współczynniki konwersji dla grupy eksperymentalnej, a  $X_n$  to zmienna losowa określająca współczynniki konwersji w grupie kontrolnej. Na podstawie nowej zmiennej losowej można zapisać hipotezy zerową i alternatywną. Hipotezę zerową można zdefiniować tak:

$$H_0: X = 0$$

Oznacza ona, że grupa eksperymentalna nie różni się od grupy kontrolnej. Obie zmienne losowe,  $X_e$  i  $X_n$ , mają rozkład skupiony wokół tej samej średniej dla populacji, dlatego nowa zmienna losowa,  $X$ , powinna mieć rozkład skupiony wokół 0. Hipotezę alternatywną możemy sformułować tak:

$$H_a: X > 0$$

Tu oczekiwane jest, że zmienna losowa w grupie eksperymentalnej przyjmuje wartości większe niż w grupie kontrolnej. Średnia dla populacji w grupie eksperymentalnej jest wyższa.

Można przeprowadzić jednoczynnikowy test z dla rozkładu  $X$  przy założeniu, że prawdziwa jest hipoteza zerowa, aby ustalić, czy istnieją dowody na prawdziwość hipotez alternatywnych. W tym celu należy wybrać próbki z rozkładu  $X$ , obliczyć wskaźnik z (ang. *standard score*) i porównać wynik ze znanimi poziomami istotności.

Pobieranie próbki z  $X$  to odpowiednik przeprowadzenia dwóch eksperymentów, ustalenia dla nich współczynników konwersji i odjęcia współczynnika konwersji dla grupy kontrolnej od współczynnika konwersji dla grupy eksperymentalnej. Na podstawie definicji wskaźnika z można zapisać następujący wzór:

$$z = (p_{\text{eksperymentalna}} - p_{\text{kontrolna}}) / SE$$

W tym wzorze  $p_{\text{eksperymentalna}}$  to współczynnik konwersji dla grupy eksperymentalnej,  $p_{\text{kontrolna}}$  to współczynnik konwersji dla grupy kontrolnej, a  $SE$  to błąd standardowy dla różnicy współczynników konwersji.

Aby ustalić błąd standardowy, zauważ, że konwersje mają rozkład dwumianowy. Dlatego odwiedziny w witrynie można traktować jak jedną próbę Bernoulliego o nieznanym prawdopodobieństwie pozytywnego skutku (konwersji). Jeśli liczba prób jest wystarczająco duża, można oszacować rozkład jako normalny za pomocą powszechnie stosowanej metody Walda<sup>3, 4</sup>. Aby uwzględnić niepewność współczynnika konwersji, można ustalić błąd standardowy ( $SE$ ) dla grupy eksperymentalnej i kontrolnej w pokazany poniżej sposób. W tym wzorze  $p$  to prawdopodobieństwo konwersji, a  $n$  to liczba prób:

<sup>3</sup> Lawrence D. Brown, T. Tony Cai i Anirban DasGupta, *Confidence Intervals for a Binomial Proportion and Asymptotic Expansions*, „The Annals of Statistics” 30, nr 1 (2002): 160 – 201.

<sup>4</sup> Sean Wallis, *Binomial Confidence Intervals and Contingency Tests: Mathematical Fundamentals and the Evaluation of Alternative Methods*, „Journal of Quantitative Linguistics” 20, nr 3 (2013): 178 – 208.

$$SE^2 = \frac{p(1-p)}{n}$$

Licznik wyprowadziliśmy z wariancji rozkładu dwumianowego ( $np(1-p)$ ), a w mianowniku uwzględnione jest to, że błąd współczynnika konwersji maleje wraz z liczbą prób. Ponieważ prawdopodobieństwo pozytywnego skutku odpowiada współczynnikowi konwersji, a błąd standardowy dwóch zmiennych można połączyć za pomocą dodawania, otrzymujemy:

$$\begin{aligned} SE^2 &= SE_{eksp.}^2 + SE_{kontr.}^2 \\ SE_{eksp.}^2 &= \frac{p_{eksp.}(1-p_{eksp.})}{n_{eksp.}} \\ SE_{kontr.}^2 &= \frac{p_{kontr.}(1-p_{kontr.})}{n_{kontr.}} \end{aligned}$$

Po podstawieniu wzór na test z można zapisać w pokazany poniżej sposób. Jest to wzór na przedział Walda w rozkładzie dwumianowym:

$$z = (p_{eksp.} - p_{kontr.}) / \sqrt{SE_{eksp.}^2 + SE_{kontr.}^2}$$

Im większa jest wartość  $z$ , tym mocniejszy jest dowód na nieprawdziwość hipotezy zerowej. Aby w teście jednoczynnikowym uzyskać przedział ufności dla poziomu ufności 90%, wartość  $z$  musi być większa niż 1,28. Oznacza to, że przy założeniu prawdziwości hipotezy zerowej (zgodnie z którą średnie populacji dla grup A i B są takie same) prawdopodobieństwo przypadkowego wystąpienia takiej (lub większej) różnicy współczynników konwersji jest mniejsze niż 10%. Można ująć to jeszcze inaczej: jeśli przy założeniu, że współczynniki konwersji dla grup kontrolnej i eksperimentalnej pochodzą z rozkładów o tej samej średniej, przeprowadzisz ten sam eksperyment 100 razy, to tylko w 10 przypadkach otrzymasz tak wysoką (lub jeszcze wyższą) wartość. Można zastosować jeszcze większe ograniczenia i uzyskać mocniejszy dowód przeciw hipotezie zerowej, wybierając przedział ufności dla poziomu ufności 95%. Wtedy wymagana wartość  $z$  to 1,65.

Przydatne może być zastanowienie się nad czynnikami wpływającymi na wartość  $z$ . Jeśli w danym momencie wybierzesz dwa współczynniki konwersji ze zbiorów eksperymentalnego i kontrolnego, większa różnica da większy wynik  $z$ , co stanowi mocniejszy dowód na to, że współczynniki pochodzą z różnych populacji o odmiennych średnich. Istotna jest też liczba próbek. Wiesz już, że większa liczba próbek prowadzi do ogólnie mniejszego błędu standardowego. Wynika to z tego, że szacunkowy współczynnik konwersji jest tym bardziej precyzyjny, im dłużej prowadzimy eksperyment. W kolejnym punkcie prezentujemy ten fakt oraz pokazujemy, jak za pomocą opisanej techniki przeprowadzić testy A/B w Pythonie.

### 7.1.2. Kod

Wyobraź sobie, że kierujesz witryną dużego sklepu, a zespół projektowy właśnie zmodyfikował stronę wejściową. Witrynę odwiedza około 20 000 użytkowników tygodniowo. Możliwe jest obliczenie współczynnika konwersji tych osób (czyli procentu użytkowników, którzy ostatecznie dokonali zakupu). Zespół projektowy zapewnia, że nowa witryna przyciągnie więcej klientów, jednak nie jesteś o tym przekonany i chcesz przeprowadzić testy A/B, aby sprawdzić, czy skuteczność strony wejściowej wzrośnie.

Użytkownicy w trakcie pierwszej wizyty w witrynie są losowo przydzielani do grup A i B, po czym pozostają przypisani do danej grupy przez cały czas trwania eksperymentu. Na koniec eksperymentu oceniany jest średni współczynnik konwersji dla obu grup. Dla nowej strony wejściowej wynosi on 0,002, a dla poprzedniej — 0,001. Chcesz ustalić, czy ten wzrost jest wystarczająco istotny, aby uzasadniał trwałą zmianę projektu strony wejściowej. Spójrz na kod, który pomoże Ci odpowiedzieć na to pytanie (patrz listing 7.1.).

#### Listing 7.1. Obliczanie wartości $z$ w omawianym eksperymencie

```
import math
import random
import numpy as np
import matplotlib.pyplot as plt

n_experiment = 10000
n_control = 10000

p_experiment= 0.002
p_control = 0.001

se_experiment_sq = p_experiment*(1-p_experiment) / n_experiment
se_control_sq = p_control*(1-p_control) / n_control

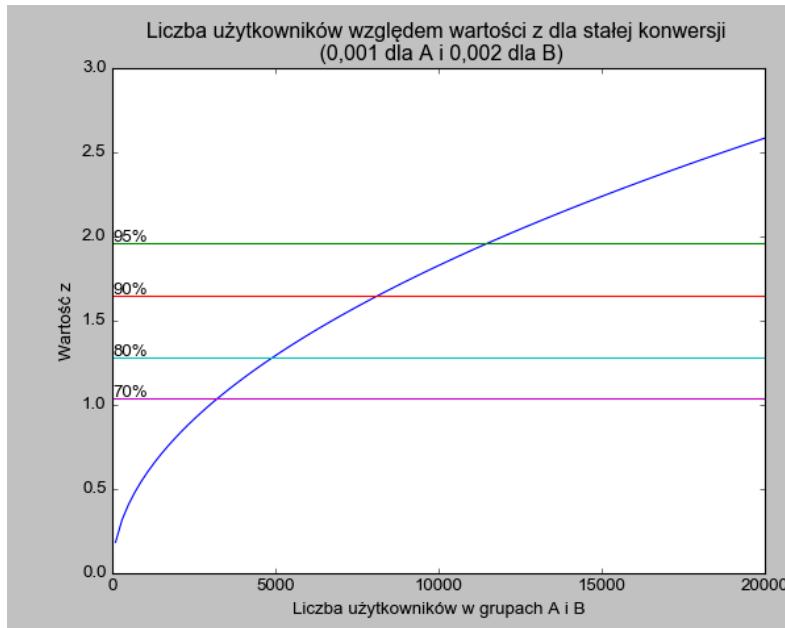
Z = (p_experiment-p_control)/math.sqrt(se_experiment_sq+se_control_sq)

print Z
```

Ten kod oblicza wartość  $z$  w omawianym eksperymencie. Dla użytych danych  $z$  wynosi 1,827. Przekracza to wartość dla przedziału ufności dla poziomu ufności 92%, ale wynosi mniej niż dla przedziału ufności dla poziomu ufności 95%. Można powiedzieć, że prawdopodobieństwo pochodzenia danych eksperymentalnych z rozkładu kontrolnego wynosi mniej niż 0,08. Tak więc dla przedziału 92% wynik jest istotny statystycznie. Należy więc odrzucić hipotezę zerową, zgodnie z którą między grupami nie występuje różnica, i przyjąć hipotezę alternatywną, wedle której współczynnik konwersji w drugiej grupie jest wyższy. Jeśli wszystkie pozostałe aspekty grup są kontrolowane, wynika z tego, że nowy projekt witryny przyniósł pozytywne skutki.

Z kodu powinieneś wywnioskować, że błąd standardowy ( $SE$ ) z rozkładów, z których pochodzą współczynniki konwersji, ma bezpośredni wpływ na uzyskaną wartość  $z$ . Im wyższy jest  $SE$  obu grup, tym niższa jest wartość  $z$ , a tym samym mniej znaczące wyniki (przy stałych wartościach  $p_{eksperymentalna}$  i  $p_{kontrolna}$ ). Zauważ ponadto, że zgod-

nie z definicją SE wartość  $z$  jest bezpośrednio powiązana z liczbą próbek (przy stałym prawdopodobieństwie konwersji). Na rysunku 7.2 przedstawiamy to w formie graficznej.



**Rysunek 7.2.** Współczynniki konwersji w grupach A i B, stosunek liczby użytkowników w obu grupach i wartość  $z$  są stałe. Przy założeniu, że współczynnik konwersji nie zmieni się po zebraniu dodatkowych danych, potrzeba około 3000 użytkowników w każdej grupie, aby uzyskać przedział ufności dla poziomu ufności 70%. Dla 80% liczba użytkowników rośnie do około 5000 osób na grupę, dla 90% jest to 7500 użytkowników na grupę, a dla 95% — 12 000 użytkowników na grupę

Widać tu, że przy stałych wartościach współczynników konwersji w obu grupach większa liczba użytkowników w testowanych grupach zwiększa pewność co do prawdziwości hipotezy alternatywnej. Intuicyjnie jest to zrozumiałe. Większy zbiór zebranych danych powinien zwiększać pewność co do wyniku. Zauważ, że podobny wykres można narysować dla stałej liczby użytkowników i zmieniających się różnic między grupami. Należy jednak zwrócić uwagę na to, że w omawianym kontekście nie należy oczekwać znaczących zmian skuteczności witryny. Pokazuje to, że konieczne jest zebranie dużej ilości danych, aby mieć pewność, że zmiany w witrynie dały istotną poprawę skuteczności. Przykładowy kod z tego rozdziału generuje wykres pokazany na rysunku 7.2. Spróbuj wygenerować podobny wykres z różnicami we współczynniku konwersji na osi x.

### 7.1.3. Adekwatność testów A/B

Wcześniej przedstawiliśmy podstawowe wprowadzenie do statystycznej techniki testowej — testu z. Omówiliśmy ją w kontekście testu A/B, w którym sprawdzaliśmy wpływ modyfikacji witryny na działanie sklepu w celu ustalenia, czy tę zmianę warto wprowadzić na stałe. Odkryliśmy, że gdy zmiana skuteczności jest bardzo mała (co jest typowe

w wielu inteligentnych aplikacjach sieciowych), grupy kontrolna i testowa muszą być duże. W omawianym fikcyjnym sklepie dochodzenie do rozstrzygających wniosków może potrwać sporo czasu (zależy to od liczby odwiedzających go codziennie użytkowników). W środowisku biznesowym związany jest z tym pewien problem: w trakcie testów efekty działalności mogą być nieoptimalne, ponieważ połowa użytkowników przypisana jest do grupy uzyskującej gorsze wyniki (może to być albo grupa testowa, albo grupa kontrolna), a firma musi czekać na ukończenie testów.

Jest to klasyczny *problem eksploracji i wykorzystywania wyników* (ang. *explore-exploit conundrum*). Trzeba wypróbować nieoptimalne podejścia w celu eksploracji możliwości i znalezienia dających lepsze wyniki rozwiązań, a jednocześnie wyniki należy wykorzystać jak najszybciej, aby uzyskać przewagę konkurencyjną. A co, gdyby można było wprowadzać nowe rozwiązania szybciej, bez oczekiwania na zakończenie testów? No cóż, jest to wykonalne. Poznaj wielorękiego bandytę (ang. *multi-armed bandit*).

## 7.2. Wieloręki bandyta

Nazwa *wieloręki bandyty* wzięła się od popularnej w kasynach gry — jednorękiego bandyty. Oto opis dla osób, które nie odwiedzają kasyn: jednoręki bandyta to maszyna, w której gracz pociąga za drążek („rękę”) i w zależności od wyświetlanej wartości albo otrzymuje wygraną, albo (co bardziej prawdopodobne) traci pieniądze. Można się domyślić, że szanse wygranej są tak skalkulowane, aby kasyno na tym zarabiało. Dlatego prawdopodobieństwo wygranej jest zwykle bardzo niewielkie.

(Teoretyczny) wieloręki bandyta to rozwinięcie tej maszyny. Wyobraź sobie, że stoisz przed rzędem jednorękich bandytów, z których każdy wypłaca nagrodę z niezależnym od innych prawdopodobieństwem. Gracz nie zna tych prawdopodobieństw. Jedyny sposób na ich ustalenie to gra na tychautomatach. Zadanie polega na takiej grze, by zmaksymalizować wygraną. Jaką strategią powinieneś się posłużyć? Ilustruje to rysunek 7.3.



**Rysunek 7.3.** Problem wielorękiego bandyty. Użytkownik stoi przed rzędem jednorękich bandytów o różnym prawdopodobieństwie wygranej. Użytkownik nie zna tych prawdopodobieństw i może je odkryć tylko w wyniku gry. Jaką strategię powinieneś przyjąć, aby zmaksymalizować wygraną? Użytkownik musi zbadać przestrzeń, aby ustalić wspomniane prawdopodobieństwa, a jednocześnie musi wykorzystać maszyny o najwyższym prawdopodobieństwie wygranej, aby zmaksymalizować zysk

### 7.2.1. Strategie stosowane w problemie wielorękiego bandyty

Przed przejściem do pisania kodu należy ściślej zdefiniować problem. Istnieje  $k$  maszyn z obserwowalnym prawdopodobieństwem wygranej równym  $p_k$ . Zakładamy, że w danym momencie można pociągnąć za tylko jeden drążek, a maszyna albo wypłaca wygraną,

albo tego nie robi (zgodnie z powiązanym prawdopodobieństwem). Jest to gra skończona z dozwoloną określona liczbą pociągnięć. W każdym momencie gry zdefiniowany jest horyzont  $H$  odpowiadający liczbie pociągnięć pozostałych do wykonania.

Użytkownik stara się zmaksymalizować wygraną z uwzględnieniem wszystkich maszyn. W dowolnym momencie może oszacować, jak dobrze sobie radzi, stosując miarę *straty* (ang. *regret*). Określa ona różnicę między wygraną, jaką gracz uzyskałby, gdyby miał „szklaną kulę” i w każdym kroku mógł wybrać optymalną maszynę, a rzeczywistymi wypłatami. Formalnie strata jest zdefiniowana tak:

$$\sigma = T\mu_{opt} - \sum_{t=1}^T r_t$$

W tym wzorze  $T$  to liczba wykonanych do tej pory kroków,  $r_t$  to wygrana w kroku  $t$ , a  $\mu_{opt}$  to średnia wygrana dla optymalnej maszyny. Im niższa strata, tym bardziej optymalna jest strategia. Jednak ponieważ ta miara jest zależna od przypadku (możliwe jest uzyskanie wyższej wygranej niż oczekiwana wygrana powiązana z najlepszym bandytem), zamiast nią można posłużyć się oczekiwana stratą. Formalnie jest ona zdefiniowana tak:

$$T\mu_{opt} - \sum_{t=1}^T \mu_t$$

W tym wzorze  $\mu_t$  to nieobserwowały średnia wypłata dla danej maszyny w czasie  $t$ . Ponieważ uwzględniana jest oczekiwana wygrana w wybranej strategii, będzie ona mniejsza lub równa względem oczekiwanej wygranej w strategii optymalnej (czyli wyboru za każdym razem maszyny z oczekiwana wygraną  $\mu_{opt}$ ).

W następnych podpunktach przedstawiamy nową zmienną, *epsilon*. W dalszych strategiach zobaczysz, że kontroluje ona kompromis między eksplorowaniem przestrzeni rozwiązań a wykorzystywaniem najlepszego znanego rozwiązania. Jest ona wyrażana za pomocą prawdopodobieństwa.

## NAJPIERW EPSILON

*Najpierw epsilon* (ang. *epsilon first*) to najprostsza strategia dla wielorękiego bandyty. Można ją uznać za odpowiednik przedstawionych wcześniej testów A/B. Gdy dany jest  $\epsilon$ , należy przeprowadzać eksplorację  $(1-\epsilon) \times N$  razy, gdzie  $N$  to łączna liczba prób dostępnych w grze. Pozostałe próby są nastawione na wykorzystywanie wiedzy.

Metoda `update_best_bandit` zapisuje sumę bieżącą wygranych dla poszczególnych maszyn i liczbę rozegranych na nich gier. Po każdej grze zmienna `best_bandit` jest aktualizowana, tak aby zawierała indeks maszyny, która do danego momentu dawała największe szanse wygranej. To rozwiązanie ilustruje pseudokod z listingu 7.2.

### Listing 7.2. Pseudokod strategii najpierw epsilon

```
epsilon=0.1
best_bandit # Indeks najlepszej maszyny w tablicy
bandit_array # Tablica obiektów reprezentujących maszyny
total_reward=0
```

```

number_trials
current_trial=0

number_explore_trials = (1-epsilon)*number_trials

while((number_trials-current_trial)>0):
    if(current_trial<number_explore_trials): ← Eksplorowanie przestrzeni rozwiązań
        random_bandit = rand(0,len(bandit_array))
        total_reward += play(bandit_array[random_bandit])
        update_best_bandit()#update the best bandit
    else: ← Wykorzystanie wiedzy
        total_reward +=play(bandit_array[best_bandit])

    current_trial+=1

```

### STRATEGIA ZACHŁANNA

W strategii zachłannej (ang. *epsilon greedy*)  $\epsilon$  określa prawdopodobieństwo eksploracji przestrzeni rozwiązań (zamiast gry na najlepszej do danego momentu maszynie). W bardziej formalnej postaci ilustruje to pseudokod z listingu 7.3.

#### Listing 7.3. Pseudokod strategii zachłannej

```

epsilon=0.1
best_bandit ← Indeks najlepszej maszyny w tablicy
bandit_array ← Tablica z obiektami reprezentującymi maszyny
total_reward=0
number_trials
current_trial=0

while((number_trials-current_trial)>0):
    random_float = rand(0,1)
    if(random_float<epsilon): ← Eksploracja przestrzeni rozwiązań
        random_bandit = rand(0,len(bandit_array))
        total_reward += play(bandit_array[random_bandit])
        update_best_bandit() ← Aktualizacja najlepszej maszyny
    else: ← Wykorzystanie wiedzy
        total_reward +=play(bandit_array[best_bandit])

    current_trial+=1

```

Zaletą tego podejścia jest to, że można zacząć korzystać z wiedzy na temat skuteczności maszyn bez oczekiwania na ukończenie etapu eksploracji. Bądź jednak ostrożny. Ten algorytm nie uwzględnia istotności statystycznej danych. Możliwe, że wysokie wyniki z konkretnej maszyny spowodują, iż gracz błędnie zacznie grać tylko na niej.Więcej na ten temat dowiesz się już wkrótce.

W tym podejściu  $\epsilon$  kontroluje prawdopodobieństwo, że gracz będzie eksplorował rozwiązania, zamiast korzystać z dotychczasowej wiedzy. Niskie wartości zmiennej epsilon zmniejszają prawdopodobieństwo eksploracji, natomiast wysokie je zwiększą. Widać, że trzeba tu zdecydować się na kompromis. Dobór wartości  $\epsilon$  zależy od wielu czynników. Zarówno liczba maszyn, jak i prawdopodobieństwo wygranej wpływają na przedstawioną wcześniej miarę straty. Oczywistym problemem jest też początkowo-

wanie pracy algorytmu. Na początku eksperymentu nic nie wiadomo o skuteczności którejkolwiek z maszyn (inaczej niż w strategii najpierw epsilon). Może istnieje lepsze rozwiązanie, pozwalające eksplorować przestrzeń, gdy horyzont jest odległy, i przechodzić do wykorzystywania wiedzy, gdy horyzont się zbliża?

### ZMNIEJSZANIE EPSILONU

Strategia *zmnieszania epsilonu* działa właśnie w opisany sposób. Na początku eksperymentu wartość  $\epsilon$  jest wysoka, dlatego prawdopodobieństwo eksploracji też jest duże. Wraz ze zbliżaniem się horyzontu ta wartość stopniowo maleje, co zwiększa prawdopodobieństwo wykorzystywania wiedzy. Przedstawia to listing 7.4.

**Listing 7.4. Pseudokod strategii zmnieszania epsilonu**

```

epsilon=1 ← Początkowo strategia tylko eksploruje
best_bandit
bandit_array
total_reward=0
number_trials
current_trial=0

while((number_trials-current_trial)>0):
    random_float = rand(0,1)
    if(random_float<epsilon): ← Eksplorowanie przestrzeni rozwiązań
        random_bandit = rand(0,len(bandit_array))
        total_reward += play(bandit_array[random_bandit])
        update_best_bandit() ← Aktualizowanie najlepszej maszyny
    else: ← Wykorzystywanie wiedzy
        total_reward += play(bandit_array[best_bandit])

    current_trial+=1
    epsilon = update_epsilon(epsilon)

```

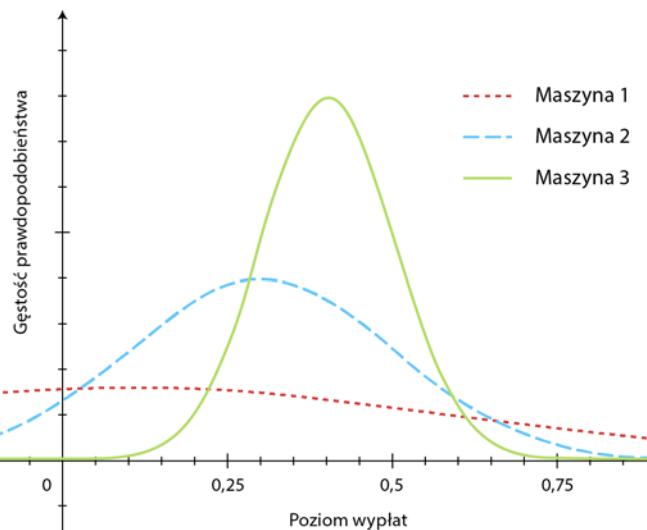
Zauważ, że istnieje kilka metod wyboru optymalnej szybkości modyfikowania wartości  $\epsilon$ . Zależy to od liczby maszyn, wartości  $N$  i wag określających wyplaty z maszyn.

### STRATEGIA BAYESOWSKA

Wcześniej wspomnialiśmy, że jednym z ograniczeń pokazanego algorytmu jest nieuwzględnianie istotności danych o skuteczności maszyn na etapie eksploracji. Dlatego choć możliwe jest wcześniejsze wykorzystanie wiedzy, może to skutkować błędnymi decyzjami. Nie chodzi o to, że opisane techniki nie są przydatne! Należy jednak zwracać baczną uwagę na parametry, aby mieć pewność, że wykorzystywanie wiedzy nie rozpocznie się za wcześnie lub że faza eksploracji nie będzie trwała zbyt długo.

Poznaj strategię bayesowską. Podobnie jak w testach A/B przyjmujemy w niej, że poziom wyplat w każdej maszynie można zamodelować za pomocą rozkładu. Początkowo dla każdej maszyny określona jest bardzo ogólna wartość a priori (ang. *prior*), ponieważ dla maszyn każdy poziom wyplat jest równie prawdopodobny. Im dłużej grasz na danej maszynie, tym więcej masz informacji na temat jej wyników. Dlatego można zaktualizować rozkład poziomu wyplat. W momencie wyboru maszyny należy pobrać

próbkę z każdego rozkładu poziomu wypłat i zdecydować się na maszynę z najwyższą wartością próbki. Rysunek 7.4 przedstawia graficznie zmieniające się w czasie informacje o trzech przykładowych maszynach.



**Rysunek 7.4.** Modelowanie wiedzy na temat poziomu wypłat trzech maszyn za pomocą strategii bayesowskiej. Średni poziom wypłat dla maszyn 1, 2 i 3 wynosi 0,1, 0,3 i 0,4. Maszyna 1 ma niższą średnią, ale znacznie większą wariancję. Maszyna 2 cechuje się wyższą średnią i niższą wariancją, a maszyna 3 – jeszcze większą średnią i jeszcze mniejszą wariancją. W trakcie wyboru maszyny z każdego rozkładu pobierana jest próbka i używana jest maszyna o największej wartości próbki. Po rozegraniu gry następuje aktualizacja odpowiedniego rozkładu. Dlatego nawet maszyny o niskim poziomie wypłat mogą „się poprawić”, jeśli ich średnie wypłaty są niepewne (wariancja jest wysoka)

Widać tu, że wiedza o rozkładach dla maszyn jest zapisana jako trzy rozkłady. Kolejne rozkłady mają wyższą średnią i niższą wariancję. Dlatego jesteśmy mniej pewni rzeczywistego poziomu wypłat dla maszyny o średniej 0,1 i bardziej pewni w przypadku maszyny o średniej 0,4. Ponieważ wybór maszyny odbywa się na podstawie próbki z każdego rozkładu, może się okazać, że wybrana zostanie maszyna o rozkładzie wokół wartości 0,1. Stanie się tak, gdy próbki z maszyn 2 i 3 będą wyjątkowo niskie, a próbka z maszyny 1 – wyjątkowo wysoka. Na listingu 7.5 pokazany jest pseudokod tego algorytmu.

#### Listing 7.5. Pseudokod strategii bayesowskiej

```

bandit_distribution_array ← Inicjowanie za pomocą odpowiednich wstępnych założeń
total_reward=0
number_trials
current_trial=0

while((number_trials-current_trial)>0):
    sample_array = sample(bandit_distribution_array)
    best_bandit = index_of(max(sample_array))

```

```

reward = play(bandit_array[best_bandit])
total_reward+=reward
current_trial+=1
update_distribution_array(best_bandit,reward)

```

Widoczna jest tu elegancja tego rozwiązania. Choć jego implementacja jest prosta, to podejście zarówno modeluje niepewność szacunków, jak i zapewnia doskonały sposób obliczania straty (w porównaniu z poprzednimi technikami). Przekonasz się o tym w następnym podrozdziale.

### 7.3. Strategia bayesowska w praktyce

W tym podrozdziale przygotowujemy eksperyment z trzema maszynami i przedstawiamy wyniki pokazujące, że im dłużej trwa gra, tym poziom straty jest niższy. Omawiamy też adekwatność tego podejścia do problemu i czynniki wpływające na to, jak długo należy stosować strategię bayesowską.

Może przypominasz sobie z poprzedniego podrozdziału, że rozkład prawdopodobieństwa powinien określać przekonanie dotyczące poziomu wyplat każdej maszyny w eksperymencie. Ponieważ w omawianym modelu możliwe są dwa wyjścia, można traktować gry jak próby Bernoulliego. Sprzężony rozkład a priori dla rozkładu Bernoulliego to rozkład beta<sup>5</sup>. Opisuje on rozkłady z dwoma parametrami: liczbą sukcesów i liczbą porażek. Te parametry to  $\alpha$  i  $\beta$ . Dla  $\alpha = \beta = 1$  rozkład beta jest odpowiednikiem rozkładu jednostajnego. Dla wysokich wartości  $\alpha$  i  $\beta$  staje się on odpowiednikiem rozkładu dwumianowego. Tu zastosujemy rozkład beta jako jednostajny rozkład a priori opisujący początkowe przekonanie o prawdopodobieństwie wyplaty dla wszystkich maszyn ( $\alpha = \beta = 1$ ) i będziemy aktualizować go na podstawie rozegranych gier i zaobserwowanych wyników. Najpierw przyjrzyj się klasie reprezentującej jednorękiego bandytę (zobacz listing 7.6).

**Listing 7.6. Klasa reprezentująca jednorękiego bandytę**

```

class Bandit:
    def __init__(self,probability):
        self.probability=probability

    def pull_handle(self):
        if random.random()<self.probability:
            return 1
        else:
            return 0

    def get_prob(self):
        return self.probability

```

W tej prostej klasie znajduje się konstruktor, który jako parametr przyjmuje prawdopodobieństwo wyplaty. Metoda `pull_handle` na podstawie tego prawdopodobieństwa

---

<sup>5</sup> Eric W. Weisstein, *Beta Distribution*, <http://mathworld.wolfram.com/BetaDistribution.html>.

określa, czy pociągnięcie drążka prowadzi do wygranej. Metoda pomocnicza `get_prob` umożliwia dostęp do prawdopodobieństwa wypłaty w danym obiekcie. Jest to podstawowa klasa w omawianym przykładzie. Często używamy jej w dalszych listingach.

Na listingu 7.7 zdefiniowana jest metoda pomocnicza służąca do pobierania próbki z każdego rozkładu reprezentującego jednorożkowego bandytę. Ta metoda także jest często używana (w ramach ustalania, który drążek należy pociągnąć zgodnie z obecnym statusem modelu).

#### Listing 7.7. Pobieranie próbki z rozkładu beta w strategii bayesowskiej

```
def sample_distributions_and_choose(bandit_params):
    sample_array = \
        [beta.rvs(param[0], param[1], size=1)[0] for param in bandit_params]
    return np.argmax(sample_array)
```

W tej metodzie używana jest metoda `beta.rvs` do pobierania losowej zmiennej z każdego rozkładu beta (o podanych na liście parametrach). Metoda zwraca indeks listy uzyskany na podstawie próbki o największej wartości. Ten indeks wyznacza drążek, który zostanie pociągnięty jako następny w eksperymencie z wielorękim bandytą.

Gdy te dwie metody są już gotowe, można przeprowadzić pierwszy eksperyment z użyciem strategii bayesowskiej. Większość kodu znajduje się na listingu 7.8, a listing 7.9 wyświetla początkowy rozkład przekonań a priori dotyczących poziomu wypłat.

#### Listing 7.8. Stosowanie strategii bayesowskiej

```
def run_single_regret(bandit_list, bandit_params, plays):
    sum_probs_chosen=0
    opt=np.zeros(plays)
    chosen=np.zeros(plays)
    bandit_probs = [x.get_prob() for x in bandit_list]
    opt_solution = max(bandit_probs)
    for i in range(0,plays):
        index = sample_distributions_and_choose(bandit_params)
        sum_probs_chosen+=bandit_probs[index]
        if(bandit_list[index].pull_handle()):
            bandit_params[index]=\
                (bandit_params[index][0]+1,bandit_params[index][1])
        else:
            bandit_params[index]=\
                (bandit_params[index][0],bandit_params[index][1]+1)
        opt[i] = (i+1)*opt_solution
        chosen[i] = sum_probs_chosen
    regret_total = map(sub.opt.chosen)
    return regret_total
```

Metoda `run_single_regret` przyjmuje jako parametry listę maszyn wraz z listą parametrów określających początkowy zakładany rozkład poziomu wypłat. Przekazywana jest też liczba prób do wykonania. Metoda zwraca oczekiwany strategii. Na początku metoda ustawia startową wartość `sum_probs_chosen` i inicjuje dwie tablice z pakietu NumPy przechowujące stan eksperymentu po każdej próbie. Dla każdej próby pobierane są próbki z rozkładów i wybierana jest maszyna, dla której uzyskano

najwyższą próbę. Na podstawie wyniku próby metoda aktualizuje parametry. Przed uruchomieniem kodu przyjrzyj się rozkładom początkowym (a priori), z których pobierane będą próbki. Te dane znajdziesz na listingu 7.9.

**Listing 7.9. Wyświetlanie rozkładów a priori w eksperymencie ze strategią bayesowską**

```
bandit_list = [Bandit(0.1), Bandit(0.3), Bandit(0.8)]
bandit_params = [(1,1),(1,1),(1,1)]

x = np.linspace(0.1, 100)
plt.plot(x,
          beta.pdf(x, bandit_params[0][0], bandit_params[0][1]),
          '-r*',
          alpha=0.6,
          label='Maszyna 1')

plt.plot(x,
          beta.pdf(x, bandit_params[1][0], bandit_params[1][1]),
          '-b+',
          alpha=0.6,
          label='Maszyna 2')

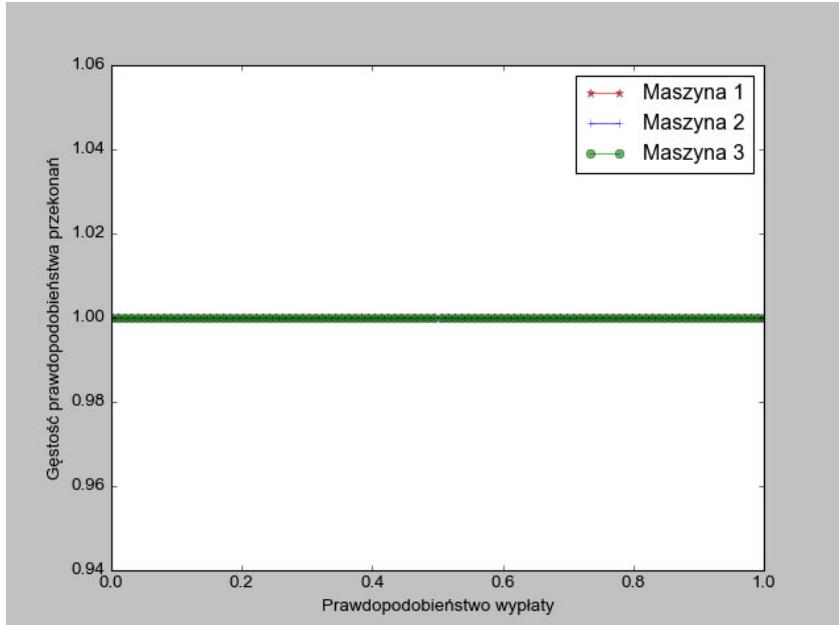
plt.plot(x,
          beta.pdf(x, bandit_params[2][0], bandit_params[2][1]),
          '-go',
          alpha=0.6,
          label='Maszyna 3')
plt.legend()
plt.xlabel("Prawdopodobieństwo wypłaty")
plt.ylabel("Gęstość prawdopodobieństwa przekonań")
plt.show()
```

Ten kod wyświetla początkowy rozkład przekonań dotyczących poziomu wypłaty powiązanych z każdą „ręką” maszyny. Przymijmy, że  $\alpha$  i  $\beta$  są równe 1. Dane wyjściowe są pokazane na rysunku 7.5. Ponadto kod inicjuje listę obiektów typu Bandit. Inicjowane są one z innymi prawdopodobieństwami wypłaty. W pierwszej maszynie wygrana jest uzyskiwana raz na dziesięć prób, w drugiej — trzy razy na dziesięć prób, a w trzeciej — osiem razy na dziesięć prób. Najwyraźniej są to bardzo uczciwi bandyci. Nigdy nie mieliśmy okazji grać w tak hojnym kasynie.

Zauważ, że funkcje gęstości prawdopodobieństwa są takie same. Dzieje się tak, ponieważ na temat każdej maszyny mamy dokładnie takie same (zerowe) informacje. Dalej zobaczysz, że wraz ze stosowaniem strategii rozkłady będą się zmieniać. Zastosujmy teraz jedną ze strategii i narysujmy oczekiwany poziom straty po każdej próbie. Potrzebny kod znajdziesz na listingu 7.10.

Używana jest tu metoda `run_single_regret`, do której przekazujemy te same początkowe wartości  $\beta$  co wcześniej, a także te same prawdopodobieństwa wypłat dla poszczególnych maszyn. Dane wyjściowe z listingu 7.10 przedstawia rysunek 7.6.

Na rysunku 7.6 pokazany jest skumulowany oczekiwany poziom straty dla jednego przebiegu strategii z 1000 prób. Oczekiwany poziom straty w danym kroku to



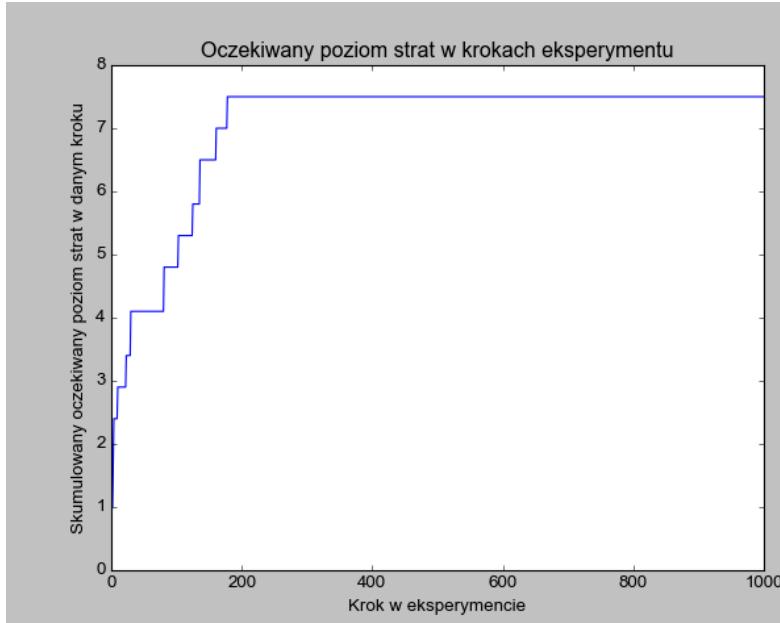
**Rysunek 7.5.** Funkcja gęstości prawdopodobieństwa dotycząca przekonań o prawdopodobieństwie wypłaty dla każdej maszyny. Zauważ, że trzy wykresy są identyczne i że obszar pod krzywą jest równy 1. Jest to zgodne z początkowym przekonaniem, że dla każdej maszyny prawdopodobieństwo wypłaty jest takie samo. Zmieni się to po zaobserwowaniu działania maszyn

**Listing 7.10. Generowanie wykresu dla jednego przebiegu strategii bayesowskiej**

```
plays=1000
bandit_list = [Bandit(0.1),Bandit(0.3),Bandit(0.8)]
bandit_params = [(1,1),(1,1),(1,1)]

regret_total = run_single_regret(bandit_list,bandit_params,plays)
plt.plot(regret_total)
plt.title("Oczekiwany poziom straty w krokach eksperymentu")
plt.xlabel("Krok w eksperymencji")
plt.ylabel("Skumulowany oczekiwany poziom straty w danym kroku")
plt.show()
```

oczekiwana różnica między optymalną a wybraną wypłatą po podjęciu określonej decyzji. Skumulowany oczekiwany poziom straty to suma tych wartości z wszystkich wcześniejszych kroków. W dobrym rozwiążaniu należy oczekiwać, że wykres tej sumy będzie się stopniowo wypłaszczać. Wskazuje to, że strategia nie podejmuje nieoptimalnych decyzji i w pełni wykorzystuje dostępną wiedzę. Przyjrzyj się teraz ponownie przekonaniom pod kątem parametrów rozkładu beta. Listing 7.11 przedstawia kod generujący graficzną ilustrację funkcji gęstości prawdopodobieństwa a posteriori (czyli po wykorzystaniu do nauki wszystkich dostępnych danych). Efekt uruchomienia tego kodu przedstawia rysunek 7.7.

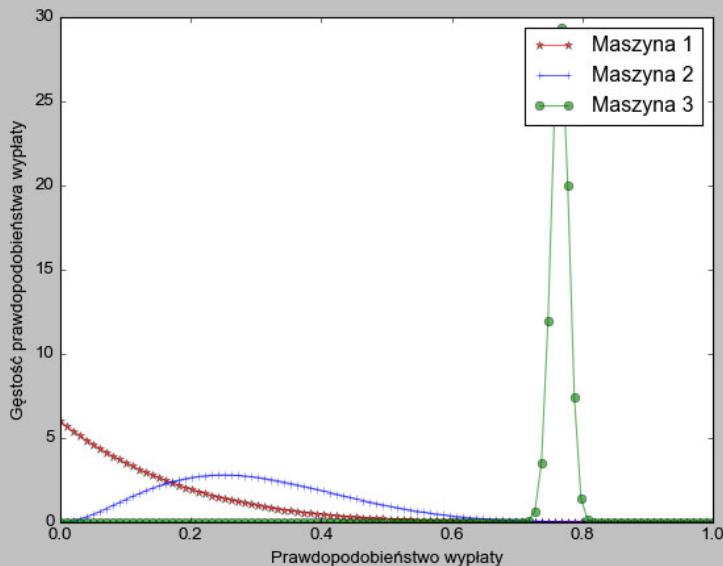


Rysunek 7.6. Łączny oczekiwany poziom straty dla jednego przebiegu z 1000 prób. Wiedza jest uzyściwana tylko w wyniku prób (eksploracji). Dlatego możliwe jest, że w trakcie uczenia się rzeczywistych prawdopodobieństw wypłaty podejmowane będą nieoptimalne decyzje

#### Listing 7.11. Rysowanie funkcji gęstości prawdopodobieństwa a posteriori

```
x = np.linspace(0.1, 100)
plt.plot(x,
          beta.pdf(x, bandit_params[0][0], bandit_params[0][1]),
          '-r*',
          alpha=0.6,
          label='Maszyna 1')
plt.plot(x,
          beta.pdf(x, bandit_params[1][0], bandit_params[1][1]),
          '-b+',
          alpha=0.6,
          label='Maszyna 2')
plt.plot(x,
          beta.pdf(x, bandit_params[2][0], bandit_params[2][1]),
          '-go',
          alpha=0.6,
          label='Maszyna 3')
plt.legend()
plt.xlabel("Prawdopodobieństwo wypłaty")
plt.ylabel("Gęstość prawdopodobieństwa przekonań")
plt.show()
```

Widac tu zupełnie inny zbiór rozkładów niż na rysunku 7.5. Pierwszy (maszyna 1) osiąga maksimum dla prawdopodobieństwa wypłaty równego 0, drugi (maszyna 2) dla prawdopodobieństwa wypłaty równego 0,4, natomiast trzeci (maszyna 3) dla prawdopodobieństwa



**Rysunek 7.7.** Rozkłady prawdopodobieństwa a posteriori dla jednego przebiegu o 1000 prób. Widoczne są tu przekonania dotyczące rzeczywistych rozkładów wypłat maszyn, dla których nieobserwowane rzeczywiste prawdopodobieństwa wynoszą 0,1, 0,3 i 0,8

wypłaty równejgo 0,8. Zwróć też uwagę na szerokość rozkładów. Dla maszyn 1 i 2 rozkłady są znacznie szersze niż dla maszyny 3, dla której uzyskaliśmy wąski rozkład wokół wartości 0,8.

Gdy przyjrzyisz się rysunkom 7.6 i 7.7, lepiej zrozumiesz działanie strategii. W trakcie pierwszych prób rozkłady przekonań dotyczące wszystkich maszyn są prawie identyczne, dlatego algorytm z równym prawdopodobieństwem wybiera dowolną maszynę. To prowadzi do szybko rosnącego skumulowanego poziomu straty. Wraz z kolejnymi próbami i przekazywaniem do modelu wyników każdej próby dominować zaczyna przekonanie co do skuteczności trzeciej maszyny i powiązane z nią najwyższe prawdopodobieństwo wypłaty. Im dłużej wybieramy tę maszynę, tym więcej uzyskujemy wygranych i tym mocniejsze staje się przekonanie, że rzeczywiste prawdopodobieństwo wypłaty wynosi około 0,8. Po 1000 wyborów generowanie próbek z funkcji gęstości prawdopodobieństwa prawie zawsze skutkuje uzyskaniem najwyższej wartości dla maszyny 3. Dlatego prawie zawsze wybierana jest maszyna 3 (dająca najlepsze wyniki), a poziom straty przestaje rosnąć (system nie podejmuje już błędnych decyzji).

Ciekawe jest to, że choć zakładana średnia wypłat maszyny 3 jest prawie identyczna z rzeczywistym prawdopodobieństwem, zakładane średnie dla maszyn 1 i 2 są mniej dokładne. Dzieje się tak, ponieważ dwie pierwsze maszyny są wybierane rzadziej. Tak naprawdę rzeczywiste wartości dla tych maszyn nie są ważne. Istotne jest prawdopodobieństwo, że wybrana maszyna daje najlepsze wyniki.

Pokazaliśmy tu ilustrację skuteczności jednej strategii. Z powodu probabilistycznej natury wyboru maszyny można zakładać, że wykres z rysunku 7.6 będzie nieco inny

(choć podobny) w każdym przebiegu tej samej strategii. Narysujmy 100 takich przebiegów, aby się przekonać, że niezależne przebiegi dają podobne wyniki. Potrzebny kod znajduje się na listingu 7.12.

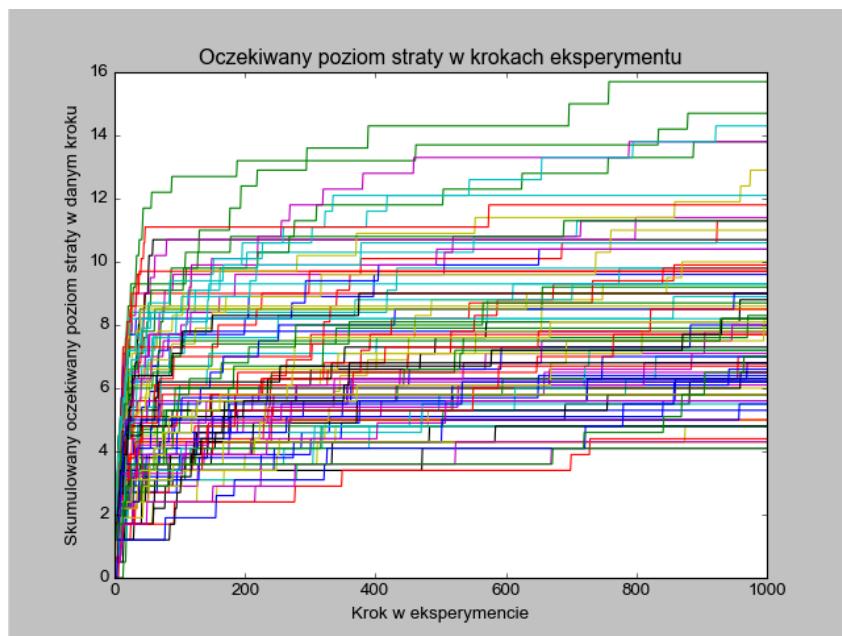
**Listing 7.12. Poziom straty w wielu przebiegach strategii bayesowskiej**

```
plays=1000
runs=100

for i in range(0,runs):
    bandit_list = [Bandit(0.1),Bandit(0.3),Bandit(0.8)]
    bandit_params = [(1,1),(1,1),(1,1)]
    regret_total = run_single_regret(bandit_list,bandit_params,plays)
    plt.plot(regret_total,label='%'%i)

plt.title("Oczekiwany poziom straty dla kroków eksperymentu")
plt.xlabel("Krok w eksperymencie")
plt.ylabel("Skumulowany oczekiwany poziom straty w danym kroku")
plt.show()
```

W każdym ze 100 przebiegów należy zresetować wyuczone parametry. Rysunek 7.8 przedstawia wykres wygenerowany przez pokazany kod.



**Rysunek 7.8.** Poziom straty dla 100 przebiegów eksperymentu z 1000 prób. Ponieważ strategia bayesowska jest probabilistyczna, nie można zagwarantować, że w każdym przebiegu wynik będzie identyczny

Widać tu, że w niezależnych przebiegach wyniki mogą być zupełnie inne, choć strategia jest inicjowana tymi samymi danymi. Skumulowany poziom straty wahaj się od około 4 (oczekiwana różnica między rzeczywistym wynikiem a optymalnym rezultatem

wynosi pięć wypłat) do 18. W większości przebiegów przed końcem eksperymentu skumulowany oczekiwany poziom straty osiąga stały poziom, co wskazuje na to, że zawsze podejmowane są optymalne decyzje. Jednak na podstawie tego wykresu trudno jest to stwierdzić z całą pewnością.

Lepiej byłoby wygenerować jeden ogólny wykres podsumowujący te informacje dla wszystkich przebiegów eksperymentu. Dokładnie tak działa kod z listingu 7.13. Dzięki uzyskaniu średniego oczekiwanej poziomu straty można określić, ile czasu zajmie strategii bayesowskiej nauczenie się najlepszego rozwiązania problemu dla określonej liczby maszyn i powiązanych z nimi parametrów.

#### **Listing 7.13. Obliczanie średniego skumulowanego oczekiwanej poziomu straty**

```
regret_sum=np.zeros(plays)

plays=1000
runs=100

for i in range(0,runs):
    bandit_list = [Bandit(0.1),Bandit(0.3),Bandit(0.8)]
    bandit_params = [(1,1),(1,1),(1,1)]
    regret_total = run_single_regret(bandit_list,bandit_params,plays)
    regret_sum=map(add,regret_sum, np.asarray(regret_total))

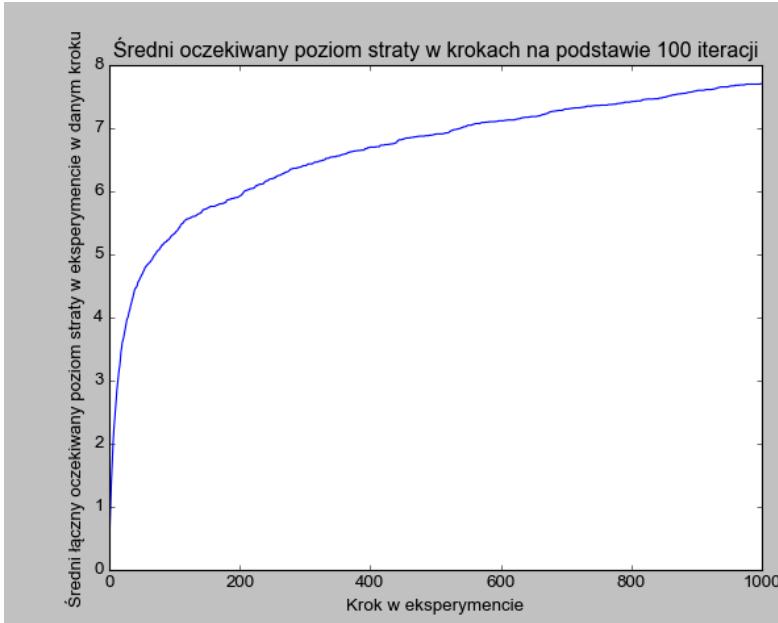
plt.plot(regret_sum/(runs*np.ones(plays)))
plt.title("Średni oczekiwany poziom straty w krokach na podstawie %s iteracji"%runs )
plt.xlabel("Krok eksperymentu")
plt.ylabel("Średni łączny oczekiwany poziom straty w eksperymencie w danym kroku")
plt.show()
```

Tak jak wcześniej uruchamiamy zbiór przebiegów eksperymentu, za każdym razem resetując parametry. W każdym kroku obliczany jest łączny skumulowany oczekiwany poziom straty, po czym wyliczana jest średnia w wyniku podzielenia tej tablicy przez tablicę z liczbą przebiegów. Uzyskane wyniki przedstawia rysunek 7.9.

Widać tu, że wraz ze wzrostem liczby kroków w eksperymencie spada szybkość narastania poziomu straty. Dzieje się tak, ponieważ na późniejszych etapach przebiegów strategia bayesowska zwykle podejmuje lepsze decyzje. Choć krzywa z rysunku 7.9 obrazuje, że w 1000 kroków nie udało się uzyskać konwergencji (nachylenie krzywej nie spada do zera), wygląda na to, że strategia bayesowska traci około dziesięciu wypłat w porównaniu z w pełni optymalnymi decyzjami. Ponieważ optymalna strategia zapewnia wypłatę w około 800 przypadkach ( $1000 \times 0,8 = 800$ ), nie jest to zły wynik.

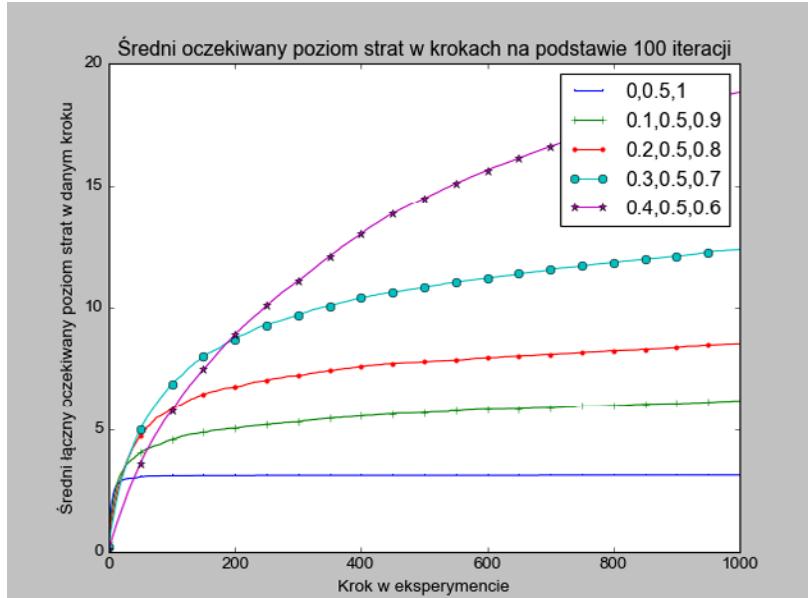
#### **CZYNNIKI WPŁYWAJĄCE NA STRATEGIĘ BAYESOWSKĄ**

W poprzednim podrozdziale przedstawiliśmy strategię bayesowską i pokazaliśmy, jak sprawdza się w problemie trzyrkiego bandyty z prawdopodobieństwami równymi 0,1, 0,3 i 0,8. Przy 1000 prób średnia oczekiwana strata wypłat wynosi tylko około 10/800, czyli 1,25%. Jest to jednak wyjątkowo dobry wynik. Przyjrzyjmy się innym czynnikom, które wpływają na skuteczność strategii bayesowskiej:

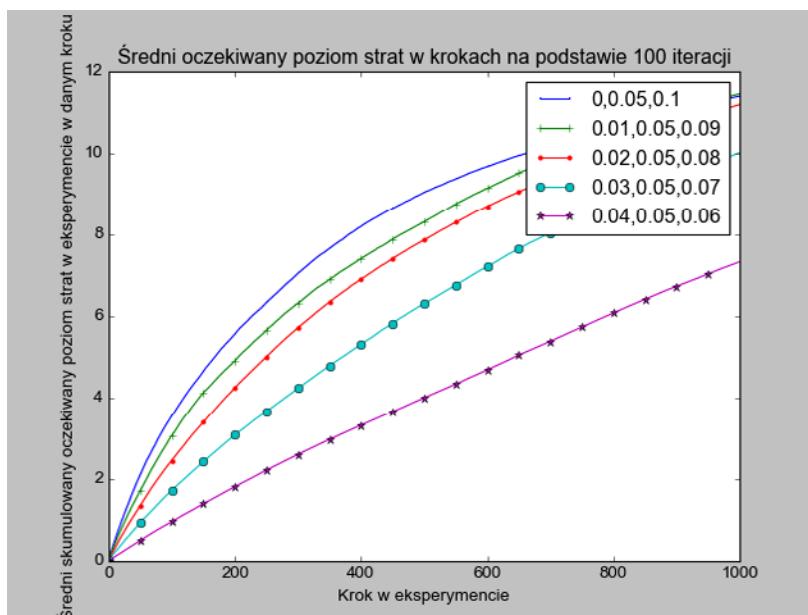


**Rysunek 7.9.** Średni skumulowany oczekiwany poziom straty dla 100 iteracji. Zauważ, że wraz ze wzrostem kroków nachylenie krzywej maleje. Dzieje się tak, ponieważ strategia bayesowska wraz z trwaniem eksperymentu podejmuje coraz lepsze decyzje. Dlatego ryzyko zwiększania się poziomu straty spada

- *Podobieństwo prawdopodobieństw.* Gdy prawdopodobieństwa w poszczególnych maszynach są podobne, rozkłady ustalone w toku uczenia w dużym stopniu się pokrywają. To oznacza, że strategia częściej „przypadkowo” eksploruje przestrzeń rozwiązań (ponieważ największa wartość próbki nie pochodzi z rozkładu o najwyższej średniej). Może to skutkować wyższym skumulowanym poziomem straty i dłuższym czasem do wypłaszczenia się wykresu (zobacz rysunek 7.10).
- *Skala prawdopodobieństwa.* Jeśli powtórzasz opisany eksperyment, ale zastąpisz pierwotne prawdopodobieństwa dziesięciokrotnie niższymi (zamiast 1 użyjesz 0,1, zamiast 0,9 — 0,09 itd.), dostrzeżesz ciekawe zjawisko. Uporządkowanie skumulowanych oczekiwanych poziomów straty zostanie odwrócone (zobacz rysunek 7.11). Dlaczego tak się dzieje? Zmniejszenie prawdopodobieństw sprawia, że pozytywne zdarzenia zachodzą znacznie rzadziej. Ma to kilka skutków. Po pierwsze, liczba kroków potrzebnych do osiągnięcia konwergencji staje się większa, ponieważ trzeba pociągać za drążki więcej razy, by uzyskać informacje o maszynach. Po drugie, ogólny poziom straty będzie wyższy niż dla maszyn o wyższych prawdopodobieństwach. Odwrócenie uporządkowania wynika z tego, że widzimy tu tylko początkową część krzywej poziomu straty. Porównaj rysunek 7.11 z częścią ilustrującą pierwszych 30 iteracji z rysunku 7.10. W drugim eksperymencie potrzeba 1000 iteracji, aby uzyskać tyle informacji co w 30 iteracjach pierwszego eksperymentu.

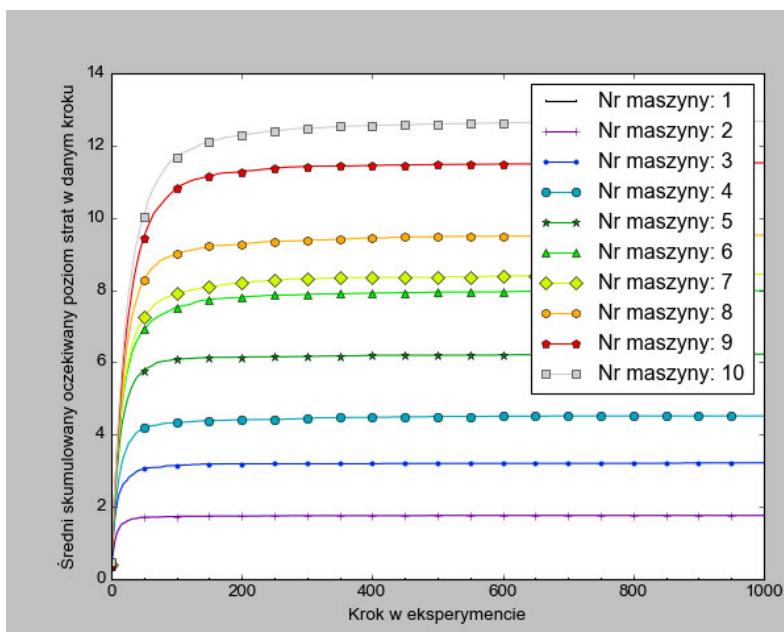


Rysunek 7.10. Skumulowany oczekiwany poziom straty dla pięciu eksperymentów (w każdym używane są trzy maszyny). Prawdopodobieństwa wypłat są podane w legendzie. Widać tu tendencję do wyższego poziomu straty i dłuższej eksploracji, gdy rzeczywiste prawdopodobieństwa dla maszyn są zbliżone do siebie



Rysunek 7.11. Skumulowany oczekiwany poziom straty dla pięciu eksperymentów z obniżonymi prawdopodobieństwami. Porównaj ten wykres z rysunkiem 7.10. Zwróć uwagę na odwrotne uporządkowanie krzywych i wolniejsze wypłaszczenie krzywych spowodowane niższymi prawdopodobieństwami

- *Liczba maszyn.* Powtórzmy eksperyment, zwiększając liczbę maszyn od jednego do dziesięciu. Za każdym razem maszyny dzielą przestrzeń prawdopodobieństwa na równe części. Jedna maszyna gwarantuje wyplatę, dla dwóch maszyn prawdopodobieństwa wyplaty wynoszą 1 i 0,5, dla trzech maszyn te prawdopodobieństwa to 1, 0,66 i 0,33 itd. Na rysunku 7.12 pokazane są dane wyjściowe z tego eksperymentu. Widoczna jest tendencja do wzrostu poziomu straty i wolniejszego uzyskiwania konwergencji, jednak różnica w szybkości dochodzenia do konwergencji nie jest tak duża jak między dwoma poprzednimi eksperymentami. Szybkość dochodzenia do konwergencji wydaje się zależeć głównie od ogólnych prawdopodobieństw wyplat oraz bliskości rozkładów prawdopodobieństw, a w mniejszym stopniu od liczby innych maszyn w eksperymencie.



Rysunek 7.12. Skumulowany oczekiwany poziom straty z dziesięciu eksperymentów z rosnącą liczbą maszyn. Prawdopodobieństwa wypłat maszyn dzielą przestrzeń prawdopodobieństwa na równe części (przy czym pomijana jest maszyna o zerowym prawdopodobieństwie wypłaty)

## 7.4. Testy A/B a strategia bayesowska

Do tej pory w rozdziale omówiliśmy kilka sposobów dokonywania właściwego wyboru. W społeczności dużo się mówi na temat wielorękich bandytów i technika ta stała się modna. Wiele osób uznaje ją za oczywisty zastępnik testów A/B. W końcu skoro można testować opcje i jednocześnie wprowadzać optymalizację, to czy rozwiązanie to nie jest lepsze od oczekiwania na uzyskanie statystycznej istotności, co jest konieczne w testach A/B?

I tak, i nie. Tak jak we wszystkich technikach z obszaru uczenia maszynowego trzeba uwzględnić zestaw wad i zalet. Należy stosować odpowiednie podejścia dostosowane do sytuacji. W tabeli 7.1 wymienione są wybrane kwestie, o których warto pamiętać w trakcie dokonywania wyboru między strategią bayesowską w problemie wielorękiego bandyty a testami A/B.

**Tabela 7.1.** Uwagi dotyczące strategii bayesowskiej i testów A/B

Strategia bayesowska	Testy A/B
Ciągła optymalizacja na żywo	Jednorazowa optymalizacja po zakończeniu testów
Wiele zmiennych	Jedna zmienna testowa
Brak jawnego pomiaru poziomu ufności	Wynik istotny statystycznie
Konwergencja jest zależna od liczby możliwości, prawdopodobieństw wypłaty i różnic między tymi prawdopodobieństwami	Konwergencja zależy od prawdopodobieństwa wypłaty i różnic między grupami kontrolną i testową
Poziom straty kumuluje się do czasu uzyskania konwergencji (zwykle jest on mniejszy niż maksymalny poziom straty)	Maksymalny poziom straty do czasu uzyskania wyniku; potem poziom straty spada do zera
Zwykle uzyskiwanie odpowiedzi zajmuje więcej czasu	Zwykle uzyskiwanie odpowiedzi trwa krócej

Z tabeli 7.1 powinieneś zapamiętać to, że testy A/B i strategia bayesowska w problemie wielorękiego bandyty mają odmienne zalety. Dlatego w niektórych sytuacjach jedna z tych technik może okazać się zupełnie niewłaściwa. Przede wszystkim chcemy zwrócić uwagę na szybkość dochodzenia do konwergencji (uzyskania istotności statystycznej w teście A/B lub zaprzestania kumulowania się poziomu straty w strategii bayesowskiej).

Przyjrzyjmy się teraz początkowemu przykładowi, optymalizacji witryny, i opiszmy adekwatność obu rozwiązań ze względu na szybkość konwergencji. Przede wszystkim zauważ, że zmiana zachowań użytkowników prawdopodobnie będzie niewielka ( $<0,01$ ), a wiadomo, że strategia bayesowska w takich warunkach dochodzi do konwergencji dłużej niż przy większych zmianach. Dodanie kolejnych możliwości (czyli testy większej liczby wersji strony wejściowej w jednym eksperymencie) także wpływie na czas dochodzenia do konwergencji. Co się stanie, jeśli rozkład konwersji użytkowników zmienia się szybciej, niż model może uzyskać konwergencję? Trendy sezonowe, wyprzedaże i inne zewnętrzne czynniki mogą wpływać na rozkład, który uważamy za statyczny. W takich sytuacjach przydatne może być staranne przemyślenie i zaplanowanie istotnego statystycznie testu A/B. Nie chcemy przez to powiedzieć, że w tej sytuacji strategia bayesowska jest bezużyteczna (wiemy, że może być przydatna), konieczne jest jednak zrozumienie charakterystyki jej działania. Nie jest ona uniwersalnym rozwiązaniem. Nie można przekazać do niej dowolnej liczby testów i oczekiwać, że będzie działać jak „szklana kula”. Zachowaj ostrożność!

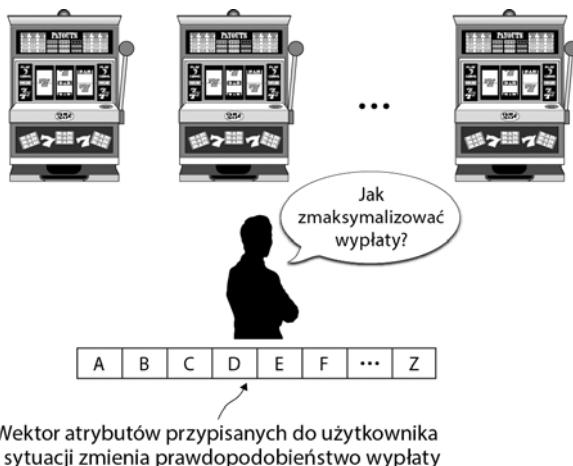
## 7.5. Rozwinięcia eksperymentu z wielorękim bandytą

Podobnie jak jest w przypadku uczenia głębokiego, tak i tu ekscytującym aspektem omawianej dziedziny są licznie pojawiające się aktualne prace badawcze. W czasie, gdy powstaje ta książka, prowadzonych jest wiele badań w tym obszarze. Zachęcamy do

zapoznania się z literaturą przedmiotu<sup>6, 7, 8</sup>. W tym podrozdziale omawiamy kilka ciekawych odkryć związanych z wielorękimi bandytami. W każdym punkcie znajdziesz dodatkowe informacje i odniesienia do badań.

### 7.5.1. Bandyci kontekstowi

*Bandyci kontekstowi*<sup>9</sup> to ważne rozwinięcie standardowego modelu, umożliwiające zakodowanie w danych wejściowych dodatkowych informacji wpływających na strategię gracza. To podejście związane jest z sytuacjami, w których prawdopodobieństwo wypłat nie jest stałe, tylko zależy od kontekstu. Graficzną ilustrację tego paradygmatu przedstawia rysunek 7.13.



**Rysunek 7.13. Bandyta kontekstowy.**  
Prawdopodobieństwo wypłaty nie jest stałe — zależy od kontekstu lub sytuacji. Możesz wyobrazić sobie, że do graczy przypisane są wektory atrybutów obowiązujące w danym momencie. Prawdopodobieństwo wypłaty w maszynach zmienia się w zależności od tych atrybutów. Najlepsza strategia ma minimalizować skumulowany poziom straty z uwzględnieniem tego faktu

Rozwiązań tego problemu mają bezpośrednie zastosowanie w świecie reklamy. Wcześniej w rozdziale wyjaśniliśmy, jak posłużyć się problemem bandytów do wyboru najbardziej optymalnych cech reklamy (na przykład odpowiedniego koloru lub kształtu). Rozwiązań tego problemu umożliwiają optymalizację globalnych cech reklamy, jednak nie uwzględniają specyfiki użytkownika, któremu reklama jest wyświetlana. Postęp w dziedzinie bandytów kontekstowych może pomóc to zmienić.

<sup>6</sup> John Myles White, *Bandit Algorithms for Website Optimization* (O'Reilly, 2012).

<sup>7</sup> Richard Weber, *On the Gittins Index for Multiarmed Bandits*, „Annals of Applied Probability” (1992): 1024 – 1033.

<sup>8</sup> J.C. Gittins, *Multi-armed Bandit Allocation Indices* (John Wiley and Sons, 1989).

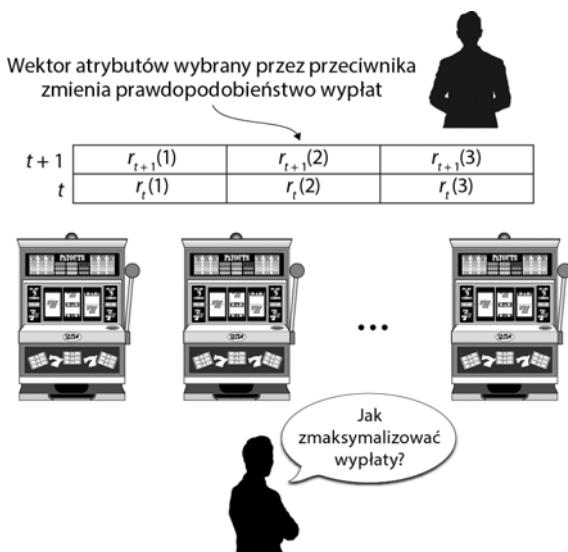
<sup>9</sup> John Langford, *Contextual Bandits*, „Machine Learning (Theory)”, 24 października 2007, <http://hunch.net/?p=298>.

### 7.5.2. Problem bandytów z przeciwnikiem

Wcześniej wspomnialiśmy, że do tej pory niejawnie zakładaliśmy, iż rozkład wyplat jest stały i nie zmienia się w trakcie gry. W rozwiązaniach *problemu bandytów z przeciwnikiem* (ang. *adversarial bandits*)<sup>10</sup> to założenie nie obowiązuje. W tym podejściu gra przebiega w następujący sposób:

1. Przeciwnik określa wektor o długości równej liczbie maszyn. W wektorze zapisane są nagrody powiązane z maszynami w danym kroku.
2. Gracz, bez świadomości wyboru dokonanego przez przeciwnika, wybiera maszynę na podstawie swojej strategii.
3. W grze z pełnymi informacjami gracz może poznać cały wektor nagród. W wersji z niepełnymi informacjami gracz poznaje tylko nagrodę powiązaną z wybraną maszyną.

Gra toczy się w ten sposób określona liczbę powtórzeń. Gracz musi zmaksymalizować wygrane i zminimalizować poziom straty. Graficzną ilustrację tego problemu przedstawia rysunek 7.14.



Rysunek 7.14. Problem bandytów z przeciwnikiem. W tej wersji problemu wielorękiego bandyty w rozwiązaniach nie są przyjmowane założenia co do rozkładów nagród. Zamiast tego problem jest modelowany jako gra między graczem a przeciwnikiem. W każdym kroku przeciwnik wybiera wektor nagród, po czym gracz wybiera maszynę. W jednej wersji (z pełnymi informacjami) gracz ma dostęp do kompletnego wektora nagród. W innej (z niepełnymi informacjami) gracz poznaje tylko nagrodę dla wybranej maszyny

## 7.6. Podsumowanie

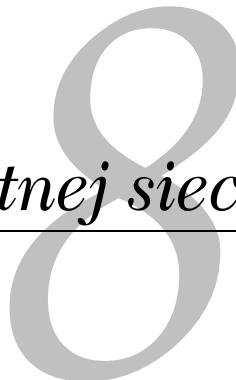
- Omówiliśmy tu kilka metod dokonywania właściwego wyboru. Przyjrzaliśmy się strategiom dokonywania wyboru jednej z wielu możliwości.
- Opisaliśmy zastosowanie testu z w testach A/B i wyjaśniliśmy znaczenie istotności statystycznej.

<sup>10</sup>P. Auer, N. Cesa-Bianchi, Y. Freund i Robert E. Shapire, *Gambling in a Rigged Casino: The Adversarial Multi-armed Bandit Problem*, „Foundations of Computer Science” (1995), Proceedings of the 36th Annual Symposium on Foundations of Computer Science (IEEE, 1995): 322 – 31.

- Pokazaliśmy, że im więcej danych zbierzesz, tym większą pewność uzyskasz co do potencjalnej zmiany, jaką zapewnia dane rozwiązanie.
- Objasniliśmy też inne czynniki wpływające na istotność statystyczną: wariancję w grupach i różnice między grupami.
- Przedstawiliśmy problem wielorękiego bandyty. Umożliwia on wykorzystanie początkowych informacji (jeszcze przed uzyskaniem istotności statystycznej) do probabilistycznego modyfikowania rozkładu wyborów.
- Zobaczyłeś, w jaki sposób rozwiązanie problemu wielorękiego bandyty pozwala obniżyć ogólny poziom straty.
- Wyjaśniliśmy, że strategia dla problemu wielorękiego bandyty nie jest uniwersalnym rozwiązaniem. Na szybkość dochodzenia do konwergencji wpływa wiele czynników. Omówiliśmy niektóre z nich i wskazaliśmy często występujące pułapki, których należy się wystrzegać.



# Przyszłość inteligentnej sieci



## Zawartość rozdziału:

- Streszczenie i przegląd
- Potencjalne przyszłe zastosowania inteligentnej sieci
- Wpływ inteligentnej sieci na społeczeństwo

W tej książce zaprezentowaliśmy obecny stan inteligentnej sieci i podstawy tego obszaru. Wyjaśniliśmy, czym jest intelligentny algorytm i jak ocenić jego wartość. Na tym etapie powinieneś umieć dostrzec liczne intelligentne algorytmy, z którymi każdego dnia wchodzisz w interakcje. Omówiliśmy też kwestie projektowe i najważniejsze pułapki, których należy wystrzegać się w codziennej pracy.

Obecnie obowiązująca filozofia w świecie uczenia maszynowego to: „Więcej danych daje lepsze wyniki niż bardziej zaawansowane algorytmy”<sup>1</sup>. W kontekście uczenia głębskiego wyjaśniliśmy, że dzięki zmianom w charakterze obliczeń i internetu możliwe jest osiągnięcie dużych postępów. W sieci WWW znajduje się tak dużo informacji, że niezwykle ważne stają się odpowiednie przesyłanie ich i dostęp do nich. Na tych zagadnieniach koncentrujemy się w dodatku. Choć może wydawać się to niepowiązane z czymś, co wiele osób uważa za intelligentne algorytmy, dla Czytelników zainteresowanych omawianymi tematami przynajmniej podstawowa wiedza na temat dostępu do danych z internetu jest niezwykle istotna. Przyszli praktycy w dziedzinie intelligentnych algorytmów będą musieli umieć określać wielkość, szybkość generowania i dostępność danych napływających w czasie rzeczywistym.

<sup>1</sup> Pedro Domingos, *A Few Useful Things to Know about Machine Learning*, „Communications of the ACM” 55, nr 10 (październik 2012): 78 – 87.

Omówiliśmy tu zagadnienia, które uważamy za najważniejsze w dziedzinie inteligentnych algorytmów: wykrywanie struktury, generowanie rekomendacji, klasyfikowanie, prognozowanie kliknięć, uczenie głębokie, dokonywanie wyborów i testy. Tematy te nie są w pełni niezależne od siebie. Tam, gdzie było to możliwe, staraliśmy się przedstawiać powiązania między poszczególnymi zagadnieniami. Zaprezentowany materiał możesz traktować jak wzorce projektowe — schematy konkretnych rozwiązań. Jeśli w przyszłości zetkniesz się z zadaniem, które wymaga wygenerowania rekomendacji, zatrzymaj się na rozdziale 3. Jeżeli będziesz musiał napisać system wybierający jedną z kilku możliwości, odśwież swoją pamięć, zaglądając do rozdziału 7.

Pamiętaj jednak, że nie jest to kompletny podręcznik. Inteligentne algorytmy to bardzo rozległy temat, obejmujący szereg powiązanych ze sobą dziedzin. Dlatego próba zawarcia całej tak obszernej wiedzy na kartach tej książki byłaby pozbawiona sensu. Mamy jednak nadzieję, że dzięki tej książce zobaczyłeś, jak podchodzić do problemów z nastawieniem na zastosowanie inteligentnych algorytmów, a także poznaleś niektóre z istniejących rozwiązań. Dobra wiadomość jest taka, że omawiana dziedzina wciąż się rozwija. Do rozwiązania pozostało wiele problemów specyficznych dla konkretnych aplikacji, do czego można posłużyć się licznymi dobrze przebadanymi technikami. Praktyk z tego obszaru musi połączyć wszystkie elementy, pamiętać o ograniczeniach i znaleźć drogę naprzód. Powodzenia!

## 8.1. Przyszłe zastosowania inteligentnej sieci

Książkę zaczęliśmy od pokazania, jak może działać rzeczywista inteligentna aplikacja sieciowa — produkt Google Now. Dobrym pomysłem wydaje się naszkicowanie możliwej drogi od chwili obecnej do przyszłości i zwrócenie uwagi na obszary, w których mogą pojawiać się nowe algorytmy. Niektóre z proponowanych rozwiązań mogą wejść do powszechnego użytku, inne mogą wydać się nierealistyczne. Pozostawiamy Czytelnikom stwierdzenie, które aplikacje mogą stać się rzeczywistością, a które pozostaną w sferze fantastyki naukowej.

### 8.1.1. Internet rzeczy

*Internet rzeczy* to ogólne pojęcie, pod którym kryje się nowy nurt informatyki. W internecie rzeczy wszystkie urządzenia są stale podłączone do internetu. Jest to urzeczywistnienie przedstawionej przez Marcę Weisera wizji przetwarzania bez granic (ang. *ubiquitous computing*)<sup>2</sup>, zgodnie z którą nośniki mocy obliczeniowej skurczyły się do tego stopnia, że wtopiły się w tło codzienności. Na razie internet rzeczy wciąż istnieje głównie w teorii, ponieważ dokonanie postępów w tym obszarze wymaga rozwiązania problemów związanych z bezpieczeństwem i standaryzacją komunikacji. Możesz stwierdzić, że postępy są widoczne w postaci rozwiązań z obszaru inteligentnych domów<sup>3</sup>, co jednak powiesz na wizję podłączonego do internetu mieszkania, które zrobi za Ciebie zakupy, umówi Cię na spotkanie, skomunikuje się ze smartfonem, rozpoczęcie gotowanie

<sup>2</sup> Marc Weiser, *The Computer for the 21st Century*, „Scientific American”, 1 września 1991: 66 – 75.

<sup>3</sup> Nest Thermostat, 01/01/2015, <https://nest.com/thermostat/meet-nest-thermostat/>.

obiadu i włączy pralkę? Na razie wydaje się to jednak mało realne. Aby zrealizować tę wizję, trzeba zaprojektować wiele inteligentnych algorytmów, a to oznacza wiele potencjalnych ścieżek prac.

### **8.1.2. Opieka zdrowotna w domu**

Rozwijając pomysł inteligentnego domu, można zrobić kolejny krok i zbudować mieszkanie badające stan zdrowia mieszkańców. Może to być przydatne zwłaszcza dla starszych i zniedołężniających osób oraz dla ludzi przebywających w domu na przepustce ze szpitala. Taki dom mógłby monitorować ogólne zachowania i trendy w aktywności<sup>4,5</sup> lub okresowo badać parametry życiowe mieszkańców. To pozwoliłoby przejść od podejścia, w którym lekarze znają tylko wycinkowy obraz zdrowia pacjenta, do modelu zapewniającego holistyczne informacje o stanie zdrowia (dostępne na przykład tylko w prywatnej sieci). Ostatecznie wielu osobom takie rozwiązania mogą zapewnić większe bezpieczeństwo i niezależność poza szpitalem. Wymaga to opracowania algorytmów, które potrafią zrozumieć i wykryć ostre stany chorobowe oraz odbiegające od normy wartości parametrów życiowych. W takich algorytmach trzeba zapewnić niski współczynnik nietrafnych predykcji negatywnych oraz zadbać o prywatność monitorowanych osób.

### **8.1.3. Autonomiczne samochody**

Jak większość użytkowników wie, już od jakiegoś czasu rozwijane są autonomiczne samochody firmy Google<sup>6</sup>. Można sobie wyobrazić, że w przyszłości powstaną sieci taksówkowe, w których algorytmicznie maksymalizowana będzie liczba przewożonych pasażerów i minimalizowany czas przejazdów. Można zrobić kolejny krok i wyobrazić sobie, że wszystkie pojazdy będą sterowane autonomicznie i podłączone do jednej sieci. Jeśli podjęte zostaną odpowiednie środki bezpieczeństwa, pomoże to zmaksymalizować przepływ ruchu, ograniczyć liczbę wypadków i lepiej chronić pasażerów, a wszystko to dzięki zastosowaniu inteligentnych algorytmów. Możesz stwierdzić, że już zmierzamy w tym kierunku, ponieważ niektórzy ubezpieczyciele oferują niższe (lub lepiej dostosowane) stawki dzięki zastosowaniu urządzeń monitorujących w ubezpieczonych pojazdach<sup>7,8</sup>.

---

<sup>4</sup> Douglas McIlwraith, *Wearable and Ambient Sensor Fusion for the Characterisation of Human Motion, „International Conference on Intelligent Robots and Systems” (IROS)* (IEEE, 2010): 505 – 510.

<sup>5</sup> Julien Pansiot, Danail Stoyanov, Douglas McIlwraith, Benny Lo i Guang-Zhong Yang, *Ambient and Wearable Sensor Fusion for Activity Recognition in Healthcare Monitoring Systems, „IFMBE proc. of the 4th International Workshop on Wearable and Implantable Body Sensor Networks”* (IFMBE, 2007): 208 – 212.

<sup>6</sup> Projekt autonomicznych samochodów firmy Google, <https://www.google.com/selfdrivingcar/>.

<sup>7</sup> Adam Tanner, *Data Monitoring Saves Some People Money on Car Insurance, But Some Will Pay More, „Forbes”*, 14 sierpnia 2013, <http://mng.bz/PWgf>.

<sup>8</sup> Leo Mirani, *Car Insurance Companies Want to Track Your Every Move — And You’re Going to Let Them, „Quartz”*, 9 lipca 2014, <http://mng.bz/1elQ>.

### **8.1.4. Spersonalizowane fizyczne reklamy**

W kilku miejscach tej książki pisaliśmy o personalizacji reklamy w internecie. A co powiesz na przeniesienie takich reklam z ekranu do rzeczywistego świata? Może zetknąłeś się już z ilustracją tej możliwości, przedstawioną w filmie *Raport mniejszości*<sup>9</sup>, gdzie spersonalizowane reklamy były fizycznie prezentowane odbiorcom komunikującym się ze środowiskiem. Uważasz, że to przesada? No cóż, w 2013 roku w Londynie intelligentne kosze na śmieci z ekranami reklamowymi używały otwartej sieci Wi-Fi do określania natężenia ruchu, zliczając unikatowe adresy MAC łączące się z daną siecią. Dzięki temu reklamodawcy mogli ustalić, czy ta sama osoba przechodziła obok kosza wielokrotnie, i potencjalnie dostosować do niej reklamy<sup>10</sup>. To rozwiązanie zostało niemal natychmiast wycofane<sup>11</sup>, jest jednak ciekawym dowodem na możliwość zastosowania omawianego pomysłu i materiałem do przemyśleń z obszaru etyki.

Rozwiązania z tej dziedziny mogą wyjść poza określone lokalizacje do sieci WWW i połączyć świat cyfrowy z fizycznym. Kliknięcia będzie można wtedy prognozować na podstawie innych czynników, na przykład przemieszczania się klientów w fizycznym świecie lub tego, czy odbiorcy widzieli reklamę w trakcie porannej drogi do pracy.

### **8.1.5. Sieć semantyczna**

Sieć WWW zawiera bogate informacje, jednak sposób interakcji z nimi zwykle wciąż pozostaje na stosunkowo prymitywnym poziomie. W dużym stopniu polegamy na tym, że wyszukiwarki zrozumieją nasze żądania (słowa kluczowe) i znajdą strony lub dokumenty powiązane z szukanym zagadnieniem. Ten model działa, ale znacznie różni się od naturalnego sposobu interakcji w gronie rodziny, znajomych i współpracowników<sup>12</sup>. Nie można zakodować nawet prostych zależności między obiektami (na przykład tego, że kot jest gatunkiem zwierzęcia), dlatego w ramach wyszukiwania na podstawie słów kluczowych nie da się wykorzystać takich dostępnych dla ludzi niejawnych informacji.

Komfort pracy byłby znacznie wyższy, gdyby można było zadawać w sieci WWW pytania i na podstawie zestawu zależności otrzymywać odpowiedzi wynikające z dedukcji lub tabele zwróconych informacji (reakcją na pytanie mogłoby także być wykonanie określonych operacji). Tak wygląda wizja *sieci semantycznej*. Pojęcie to zostało zaproponowane przez Tima Bernersa-Lee i współpracowników w 2001 roku<sup>13</sup>. Ich wizja sieci semantycznej oparta jest na *zakodowaniu* wiedzy w materiałach w internecie za pomocą

---

<sup>9</sup> Andrew Orlowski, *Facebook Brings Creepy ‘Minority Report’ — Style Ads One Step Closer*, „The Register”, 9 listopada 2015, <http://mng.bz/ML01>.

<sup>10</sup> Siraj Datoo, *This Recycling Bin Is Following You*, „Quartz”, 8 sierpnia 2013, <http://mng.bz/UUbt>.

<sup>11</sup> Joe Miller, *City of London Calls Halt to Smartphone Tracking Bins*, „BBC News”, 12 sierpnia 2013, <http://mng.bz/k56c>.

<sup>12</sup> Choć — jak pokazaliśmy — sytuacja w tym obszarze zmienia się szybko. Dowodem na to są produkty Google Now i Siri firmy Apple (<http://www.apple.com/uk/ios/siri/>).

<sup>13</sup> Tim Berners-Lee, James Hendler i Ora Lassila, *The Semantic Web*, „Scientific American”, 1 maja 2001: 34 – 43.

języka znaczników i istnieniu zestawu *ontologii* łączących fakty ze sobą i ułatwiających manipulowanie danymi za pomocą logiki. W sieci semantycznej mogą działać *agenty*, które znajdują informacje i wykonują operacje na podstawie dedukcji lub dowodów. Jedną z najważniejszych zalet tego podejścia jest to, że agent może wyjaśnić wnioski z dedukcji za pomocą języka podobnego do naturalnego. Jeśli użytkownik nie zgadza się z informacjami otrzymanymi od agenta, agent może zwrócić reguły dedukcji i ontologie, co pozwala sprawdzić logiczne kroki wykonane w ramach dedukcji.

Jeszcze trochę brakuje nam do płynnej i naturalnej komunikacji z siecią WWW w sposób opisany w początkowym przykładzie wspomnianej pracy, jednak wiele badań z tego obszaru znajduje zastosowanie w przemyśle. Jest jednak pewne, że każda próba wbudowania bogatej semantyki w duże zbiory informacji dostępnych w sieci WWW będzie korzystna dla projektantów inteligentnych algorytmów i umożliwi dostęp do wiedzy, a nie tylko do samych danych.

## **8.2. Społeczne implikacje rozwoju inteligentnej sieci**

W poprzednich podrozdziałach przedstawiliśmy potencjalne obszary rozwoju inteligentnych aplikacji. To, czy nasze wizje się sprawdzą, jest rzeczą sporną. Jedno jest jednak pewne — istnieją (lub są tworzone) technologie, które umożliwiają zrealizowanie tych wizji. Ważną kwestią jest wdrażanie tych technologii i aspekty prawne. Samo to, że coś można zrealizować, nie oznacza jeszcze, że należy to zrobić. Należy rozważyć wiele implikacji społecznych przed bezrefleksyjną realizacją opisanej wizji przyszłości.

Zastrzeżenia dotyczące większości inteligentnych algorytmów dotyczą prywatności i bezpieczeństwa. Użytkownicy mają prawo oczekwać określonego stopnia prywatności w związku ze swymi poczynaniami w internecie i poza nim. Mogą też oczekwać, że ich dane będą przechowywane w bezpiecznym miejscu i nie będą wykorzystywane w szkodliwy sposób. Dlatego projektanci inteligentnych aplikacji powinni uwzględnić te wymagania. Ignorując je, robisz to na własne ryzyko. Ostatecznie kwestia sprowadza się do użyteczności rozwiązań. Użytkownicy są gotowi zrezygnować z części kontroli nad informacjami, jeśli uzyskają mierzalne korzyści przeważające nad ryzykiem, a także mają pewność, że ich dane są w bezpiecznych rękach.

Dowodem na to jest rozpowszechnienie telefonów komórkowych. Te urządzenia umożliwiają precyzyjne śledzenie pozycji użytkownika w czasie rzeczywistym na całym świecie. Dane na ten temat są przechowywane przez firmę, która — przynajmniej teoretycznie — nie jest zainteresowana szczegółową lokalizacją użytkowników. W zamian za te dane możesz za wcisnięciem klawisza komunikować się z podobnie podłączonymi do sieci osobami. Widoczne są tu skrajne wady i zalety tego rozwiązania — udostępnianie szczegółowych informacji o lokalizacji w zamian za dostęp do bardzo wartościowego narzędzia. Uważamy, że rozpowszechnienie się telefonów komórkowych było tak szybkie z powodu korzyści związanych z natychmiastową komunikacją w czasie rzeczywistym. Przewidujemy, że w przyszłości zaobserwujemy więcej skrajnych przypadków udostępniania szczegółowych danych osobowych, jeśli potrzebujące ich inteligentne aplikacje będą wystarczająco pomocne w życiu użytkowników.

Sądzimy, że rozwój sieci inteligentnej może potoczyć się w różnych kierunkach. Użytkownicy są obecnie bardziej niż kiedykolwiek wcześniej świadomi swoich praw i chcą wiedzieć, jak ich dane są wykorzystywane. Nasze zadanie polega na uzyskaniu znaczących postępów w ulepszaniu życia użytkowników przy zachowaniu zgodności z ich oczekiwaniami co do prywatności i bezpieczeństwa. Twórcy inteligentnych algorytmów będą mieli istotny wpływ na przyszłość i powinni wykorzystać to w odpowiedzialny sposób.

# *Dodatek*

## *Pobieranie danych*

### *z sieci WWW*

---

Z tej książki dowiedziałeś się, że *inteligentne aplikacje* cechują się tym, że mogą zmieniać działanie na podstawie otrzymywanych informacji. Wynika z tego, że potrzebny jest mechanizm pobierania danych i dostępu do nich. Ponieważ piszemy tu o przetwarzaniu w skali sieci WWW, uzasadnione jest założenie, że potrzebny jest system zaprojektowany z myślą o następujących aspektach:

- *Ilość danych.* System powinien radzić sobie z przetwarzaniem danych w skali sieci WWW.
- *Skalowalność.* System powinien być konfigurowalny pod kątem zmieniającego się obciążenia.
- *Trwałość.* Przestoje lub nagłe zmiany w dostępności sieci nie powinny wpływać na spójność stanu danych.
- *Opóźnienie.* Czas między wygenerowaniem a przetwarzaniem danych powinien być krótki.
- *Elastyczność.* Dostęp do danych powinien być elastyczny. Wiele usług powinno mieć możliwość odczytu i zapisu informacji na platformie (każda na innym etapie przetwarzania).

W branży internetowej występują problemy związane z *rejestrowaniem zdarzeń*. Gdy zachodzi zdarzenie, tradycyjnie jest ono rejestrowane w pliku dziennika. W dalszych punktach szczegółowo omawiamy skutki stosowania plików dziennika, a następnie przedstawiamy inne rozwiązanie, które można ocenić na podstawie przedstawionej wcześniej listy. W ramach wprowadzenia i w celu przedstawienia przykładu, do którego będziemy odwoływać się w dalszej części dodatku, przedstawiamy przypadek użycia ze świata reklamy internetowej.

## Przykład – wyświetlanie reklam w internecie

Choć wiele osób narzeka na wszechobecność reklam w sieci WWW, nikt nie zaprzeczy, że nie da się od nich uciec! Przychody tej branży są mierzone w miliardach dolarów<sup>1</sup> i wydaje się, że wartość ta będzie stale rosła wraz ze wzrostem liczby osób korzystających z mediów za pośrednictwem urządzeń elektronicznych.

Podstawy funkcjonowania rynku reklamy internetowej są proste, jednak za tą prostotą kryje się znaczna złożoność. W najprostszej formie reklamodawcy płacą wydawcy albo za wyświetlenie reklamy, albo za interakcje użytkowników z nią. Istnieją też bardziej skomplikowane modele, jednak nie chcemy tu komplikować omówienia. W pierwszym przypadku płatności są dokonywane za tysiąc odsłon (ang. *cost per mille* — CPM). Oznacza to, że za każdy tysiąc wyświetleń reklamy reklamodawca płaci określoną kwotę wydawcy. W drugim modelu opłaty są ponoszone za kliknięcia (ang. *cost per click* — CPC). Reklamodawca płaci więc wydawcy za każdym razem, gdy klient zobaczy i kliknie reklamę.

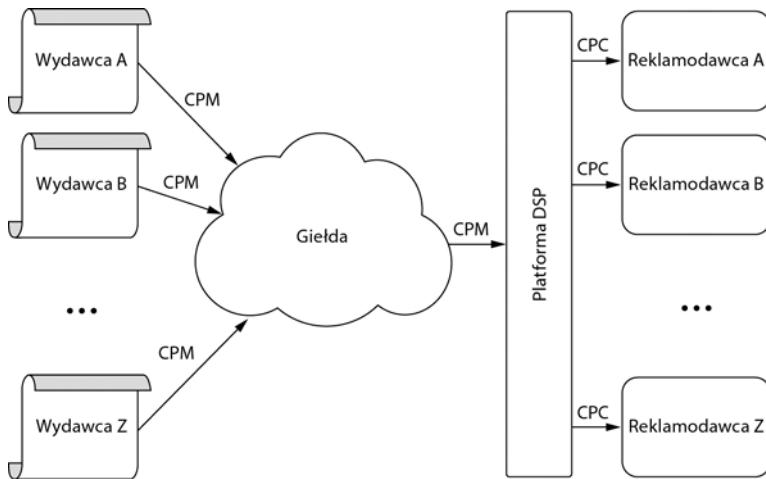
Złożoność pojawia się w procesie obsługi tych zdarzeń. Reklamodawcy nie współpracują bezpośrednio z wydawcami. Interakcje między stronami odbywają się za pośrednictwem giełdy. Możliwość emisji reklamy jest traktowana jak towar, o który współzawodniczą konkurenci. To prowadzi do wyświetlania bardziej adekwatnych reklam i (choć to sporna kwestia) lepszych doświadczeń z korzystania z sieci przez użytkowników.

Może zauważyłeś, że w modelu CPC istnieje możliwość arbitrażu. Firma działająca na rzecz popytu (reklamodawcy) może wykorzystywać dane zebrane na temat użytkowników i ich nawyków do odpowiedniego targetowania reklam w celu zwiększenia współczynnika kliknięć. Takie firmy mogą tanio kupować możliwości emisji reklam w modelu CPM, a następnie wykorzystywać dane na temat interakcji użytkowników z siecią WWW i z reklamami, aby uzyskać ponadprzeciętne wyniki. Przestrzeń reklamowa mogą później odsprzedawać dla zysku w modelu CPC. Proces ten jest pokazany na rysunku A.1. Opisaliśmy tu właśnie uproszczoną wersję platformy DSP korzystającej z inteligentnego algorytmu, co omówiliśmy w rozdziale 5. Ten przykład dobrze nadaje się także do zilustrowania świata zbierania danych.

## Dane dostępne w kontekście reklamy internetowej

Wiesz już, że inteligentne algorytmy działają na podstawie danych. Przyjrzyjmy się więc bliżej danym, które można zbierać na potrzeby targetowania reklam. Za każdym razem, gdy nasza fikcyjna platforma DSP komunikuje się z giełdą, następuje synchronizacja plików cookie. Albo giełda przekazuje do platformy DSP identyfikator użytkownika znany platformie, albo giełda przekazuje używany w niej identyfikator, a platforma DSP musi na tej podstawie wyszukać własny. Związane jest to z zabezpieczeniami strony i zasięgiem identyfikatorów. Dokładne objaśnienie tego procesu wykracza poza zakres przykładu. Wystarczy powiedzieć, że platforma DSP potrafi uzyskać identyfikator w odpowiednim dla siebie formacie, aby wyszukać informacje dotyczące zachowań użytkownika.

<sup>1</sup> Tim Berners-Lee, James Hendler i Ora Lassila, *The Semantic Web*, „Scientific American”, 1 maja 2001: 34 – 43.



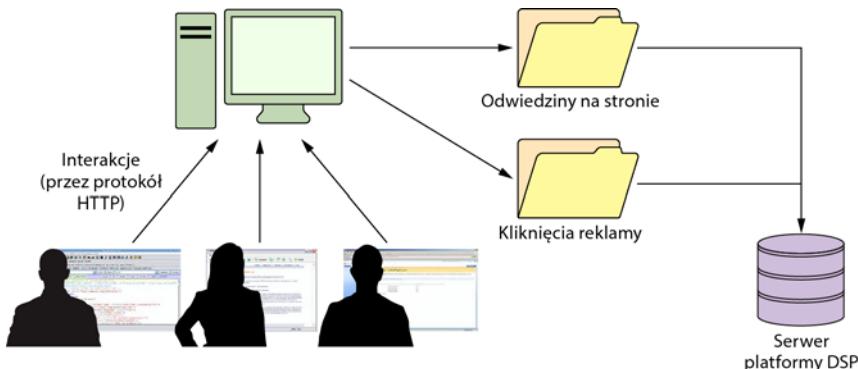
**Rysunek A.1.** Ilustracja platformy DSP. Taka platforma wykorzystuje wiedzę na temat użytkownika w celu zapewnienia lepszych interakcji. Umożliwia to arbitraż i odsprzedawanie w modelu CPC przestrzeni zakupionej w modelu CPM

Jakie informacje przechowuje platforma DSP? Są to wszystkie interakcje dotyczące obsługiwanych użytkowników, witryn i reklam. Aby to zilustrować, warto posłużyć się przykładem. Założymy, że użytkownik odwiedził witrynę firmy Adidas, a firma ta współpracuje z fikcyjną platformą DSP. Skutkuje to skomplikowanym procesem powiązania użytkownika z platformą DSP. Wszystkie interakcje użytkownika z daną witryną i witrynami innych partnerów stają się dostępne dla tej platformy. Jeśli więc użytkownik przegląda kolekcję butów do biegania, do platformy DSP przesyłany jest komunikat z informacją na ten temat. Jeżeli użytkownik przegląda kolekcję obuwia codziennego, do platformy też przekazywane są informacje o tym. Ciekawe jest to, że te wiadomości nie ograniczają się do witryny firmy Adidas. Jeśli użytkownik odwiedzi witrynę innego partnera danej platformy DSP, na przykład firmy WalMart, także ta informacja trafi do platformy DSP i zostanie powiązana z identyfikatorem danej osoby. Więcej o wykorzystywaniu tych informacji do określania grup użytkowników znajdziesz w rozdziale 5. Na razie wystarczy zapamiętać, że każda interakcja w sieci WWW skutkuje zapisaniem wielu informacji na temat użytkownika. Dla platformy DSP oznacza to istotne trudności z zakresu rejestrowania i przechowywania danych.

### **Rejestrowanie danych — naiwne rozwiązanie**

Jak zawsze istnieje naprawdę prosty sposób rejestrowania i przetwarzania danych. Jednak — jak może się domyślasz — jest on nieoptymalny. Aby zrozumieć, co dzieje się na zapleczu, przyjrzyj się hipotetycznej naiwnej architekturze przetwarzania danych w prostej platformie DSP. Graficzną ilustrację tej architektury przedstawia rysunek A.2.

Interakcje są rejestrowane w przeglądarce i przesyłane na potrzeby przetwarzania na serwery platformy DSP. Na rysunku widać, że serwer przyjmujący dane może obsługiwać



**Rysunek A.2.** Przegląd architektury przetwarzania dzienników zdarzeń w typowej platformie DSP. Przeglądarka rejestruje interakcje ze stroną i przesyła informacje na serwer (lub serwery) platformy DSP za pomocą protokołu HTTP. W zależności od typu interakcji dane są zapisywane w różnych plikach i lokalizacjach. Z czasem pliki te są scalane i importowane do bazy danych na potrzeby dalszego przetwarzania

wiele równoległych połączeń, dlatego trzeba go skonfigurować tak, aby radził sobie z dużym obciążeniem. Po otrzymaniu informacji musi wydarzyć się wiele rzeczy. Przede wszystkim zdarzenie jest rejestrowane w odpowiednim katalogu (zależy on od typu interakcji). Związane jest to z priorytetami. Zarejestrowane kliknięcia oznaczają pieniądze, dlatego minimalizowanie opóźnień w zapisie tych danych jest uzasadnione biznesowo.

Na potrzeby *przesyłania dzienników* (czyli przekazywania plików dziennika do bazy danych) warto okresowo finalizować pliki i tworzyć nowe, co pozwala jak najszybciej przesłać dane dalej. To oznacza, że pliki trzeba stale tworzyć, finalizować i odzyskiwać, gdy dane trafiają do systemu i przepływają przez niego.

Na tym etapie sfinalizowane dane są przesyłane do bazy danych w celu ich zaimportowania. Wymaga to płynnej obsługi różnych możliwych przyczyn błędów. Przede wszystkim uszkodzone pliki trzeba zabezpieczyć na potrzeby ręcznej interwencji. Przetworzone pliki należy oddzielić i zarchiwizować, a błędy w transmisji wymagają wznowienia operacji. Dane muszą koniecznie trafić do bazy w spójnym stanie. Bez tego nie można skutecznie generować rachunków ani raportów.

### Zarządzanie zbieraniem danych w dużej skali

Opisaliśmy tu bardzo prosty i hipotetyczny przykład potoku przetwarzania dzienników w platformie DSP. W praktyce w kilku obszarach zadanie jest dużo bardziej skomplikowane. Wróćmy do przedstawionej na początku dodatku listy wymagań i zbadajmy, na ile odpowiednie jest opisane rozwiązanie.

Aby umożliwić odbiór dużych ilości danych w takim systemie, trzeba zmaksymalizować liczbę połączeń obsługiwanych przez maszynę zbierającą dane. Oprócz tego potrzebny jest sposób na równoległe przetwarzanie dzienników z rejestrowanymi danymi. Potrzebny efekt można uzyskać za pomocą różnych wzorców. Za przetwarzanie mogą

odpowiadać pobliskie maszyny (bliskie witrynie lub użytkownikowi); inne rozwiązanie to losowe równoważenie obciążenia. Choć umożliwia to skalowanie horyzontalne, w pewnym momencie pliki trzeba ponownie połączyć. Jest to możliwe albo na etapie pośrednim, albo w bazie danych już po przesłaniu do niej plików. Problem z tym podejściem polega na pojawienniu się kilku punktów, w których może wystąpić błąd. Ponadto potrzebna jest warstwa koordynująca pracę węzłów i bazy danych. Możliwe, że baza będzie musiała czekać na wznowienie pracy przez uszkodzony węzeł i dopiero potem możliwe będzie połączenie oraz zimportowanie plików.

Choć taki system jest skalowalny, sposób zapewniania skalowalności jest ściśle zależny od używanej architektury. Na przykład znacznie łatwiej jest dodać węzeł obliczeniowy do systemu z losowym równoważeniem obciążenia niż do systemu starannie zbudowanego pod kątem obsługi konkretnego profilu obciążenia (na przykład z uwzględnieniem bliskości węzłów). W omawianym podejściu skalowalność nie jest gwarantowana ani automatyczna.

Trwałość danych trzeba zapewnić w projekcie systemu. Wspomnialiśmy już o możliwości niepowodzenia transmisji i uszkodzenia danych. Obsługa takich problemów wymaga dużej ilości kodu i koordynacji między różnymi etapami potoku przetwarzania. Choć opracowanie poprawnego rozwiązania jest możliwe, problem jest na tyle skomplikowany, że implementacja wymaga znacznej staranności.

Opóźnienie w przedstawionym systemie jest ściśle związane z kilkoma aspektami projektu. Aby zdarzenie zostało przesłane do bazy danych, musi najpierw trafić do mechanizmu zapisu pliku i zostać zarejestrowane w pliku, a sam plik musi później zostać zamknięty. Czas, kiedy to się stanie, zależy od obciążenia i sposobu obsługi kolejek na serwerze zbierającym dane, a także od maksymalnej wielkości plików dziennika. Po przesłaniu plik musi ponownie oczekiwany na zimportowanie do aplikacji bazodanowej.

Jeśli chodzi o elastyczność, odbiorca danych jest tylko jeden: baza. Większa liczba odbiorców doprowadzi do wzrostu ilości przesyłanych danych, ponieważ pliki dzienników trzeba przekazać do wszystkich jednostek. Ma to wpływ na logikę odpowiedzialną za spójność danych. Jeśli odbiorcy mogą znajdować się na różnym poziomie przetwarzania, umożliwiającą to logikę trzeba wbudować w kod odpowiedzialny za przesyłanie dzienników.

Mamy nadzieję, że ten prosty przykład jest wystarczająco obrazowy, aby przekonać się o tym, iż rejestrowanie danych w skali sieci WWW nie jest proste. Oczywiście wszystko zależy od aplikacji i stawianych jej wymagań, jednak próba zapewnienia cech wymienionych na początku dodatku pozwoli przygotować podstawy dla reszty aplikacji. Każda cecha ma istotny wpływ na inteligentne algorytmy i ich cykl życia. Na przykład jeśli dane treningowe trafiają do algorytmu z bardzo niskim opóźnieniem, można szybko trenować i wdrażać lepsze modele. Jeżeli możliwy jest elastyczny dostęp do danych, wiele algorytmów może równolegle korzystać z danych na różne sposoby. Czy nie byłoby dobrze, gdyby istniał system zaprojektowany i zbudowany z myślą o opisanych cechach?

## Poznaj system Kafka

Okazuje się, że taki system istnieje! Kafka organizacji Apache to rozproszona platforma przetwarzania dzienników przeznaczona do szybkiego przetwarzania dużych ilości danych. Istotą systemu jest rozproszony klaszter brokerów. Producenci danych publikują komunikaty w danym temacie (jest to strumień komunikatów określonego typu), a brokery odpowiadają za zapisywanie tych danych. Konsumenti mogą następnie zasubskrybować tematy i pobierać dane bezpośrednio od brokerów. Zanim przyjrzymy się szczegółowo tej dającej duże możliwości platformy, zabierzmy się do praktycznej pracy i napiszmy fragment kodu.

Najpierw należy skonfigurować i uruchomić we własnym systemie najnowszą wersję Kafki. W czasie, gdy powstaje ta książka, aktualnym stabilnym wydaniem jest wersja 0.10.0.0. Można ją pobrać bezpośrednio ze strony głównej projektu (<http://kafka.apache.org/>). Koniecznie pobierz wersję binarną dostosowaną do używanej przez Ciebie wersji Scali. W tym dodatku używamy Scali 2.10.4. Po pobraniu pliku wypakuj go w odpowiednim miejscu i przejdź do katalogu głównego archiwum. Następnie uruchom Kafkę z domyślną konfiguracją, wywołując dwie następujące instrukcje:

```
./bin/zookeeper-server-start.sh ./config/zookeeper.properties &
./bin/kafka-server-start.sh ./config/server.properties
```

To spowoduje uruchomienie Kafki. Domyślnie narzędzie używa portu 9092, co jest określone w pliku *server.properties*. Zauważ, że uruchamiane jest też narzędzie o nazwie ZooKeeper wiązane z portem 2181. Jest to decentralizowana usługa umożliwiająca współużytkowanie konfiguracji Kafki (wyjaśniamy to dalej). Do komunikowania się z Kafką będziesz potrzebował wiązań w wybranym języku programowania. Ponieważ w tej książce używamy Pythona, dołączenia się z naszą wersją Kafki (0.9.5) korzystamy z narzędzia *kafka-python*<sup>2</sup> Davida Arthura. Aby uruchomić przykłady z tego dodatku, dodaj to narzędzie do środowiska Pythona. Więcej szczegółów na temat instalacji znajdziesz w pliku *requirements.txt* z archiwum z kodem dostępnym na stronie poświęconej tej książce (<https://www.manning.com/books/algorithms-of-the-intelligent-web-second-edition>; spolszczena wersja jest dostępna w witrynie wydawnictwa Helion, <http://helion.pl>). Na listingu A.1 przedstawiony jest kod w Pythonie umożliwiający przesłanie pierwszego zestawu komunikatów do Kafki.

**Listing A.1. Proste publikowanie komunikatów w systemie Kafka**

```
Importowanie odpowiednich definicji z modułu kafka-python.
from kafka import KafkaClient, SimpleProducer
kafka = KafkaClient("localhost:9092")
producer = SimpleProducer(kafka)
producer.send_messages("test", "Witaj, Świecie! ")
producer.send_messages("test", "To drugi komunikat")
producer.send_messages("test", "A to trzeci!")

Tworzenie obiektu typu KafkaClient
prowadzącego do lokalnego
systemu Kafka.

Tworzenie obiektu typu SimpleProducer
używanego do publikowania komunikatów.

Wysyłanie do systemu Kafka
komunikatów w temacie „test”.
```

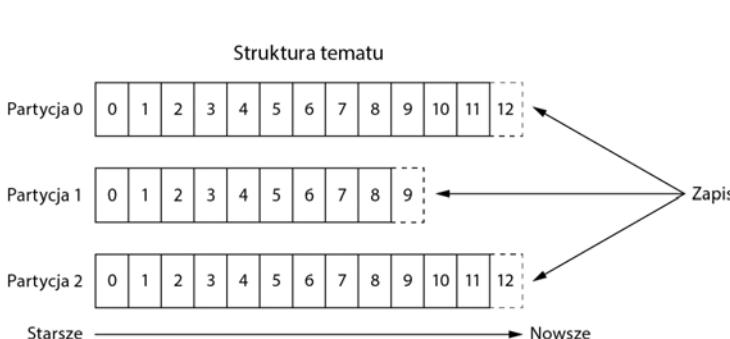
<sup>2</sup> David Arthur, *kafka-python*, <https://github.com/dpkp/kafka-python.git>.

Uruchom wszystkie wiersze z listingu A.1 w wierszu poleceń Pythona. Jeśli wszystko przebiegło poprawnie, zobaczysz dane wyjściowe podobne do poniższych.

```
[ProduceResponse(topic='test', partition=0, error=0, offset=1)]
[ProduceResponse(topic='test', partition=0, error=0, offset=2)]
[ProduceResponse(topic='test', partition=0, error=0, offset=3)]
```

Co się wydarzyło? Prześledźmy kod wiersz po wierszu. Po początkowym zimportowaniu danych z modułu kafka-python kod utworzył obiekt typu `KafkaClient`. Następnie przy jego użyciu utworzył obiekt typu `SimpleProducer`, przesłał trzy komunikaty w temacie `test` i zakończył pracę.

Przyjrzyjmy się dokładniej instrukcjom. Metoda `send_messages` zwraca listę obiektów typu `ProduceResponse` — po jednym dla każdego komunikatu przekazywanego przez tę metodę. Każdy z tych obiektów zawiera atrybuty `topic`, `partition`, `error` i `offset`. O atrybucie `topic` już wspomnieliśmy. A co z innymi? Na rysunku A.3 pokazane są zależności między tematami (atribut `topic`), partycjami (atribut `partition`) i przesunięciami (atribut `offset`).



Rysunek A.3.  
Struktura tematu na podstawie dokumentacji systemu Kafka<sup>3</sup>. Tematy składają się z podzielonych na partycje dzienników, do których można wyłącznie dodawać dane. Partycje są niemodyfikowalne — po ich zatwierdzeniu nie można wprowadzać w nich zmian

Tematy zawierają liczne partycje. Każda pojedyncza partycja musi być przechowywana u jednego brokera, przy czym partycje z jednego tematu mogą być przypisane do różnych brokerów. Służy to kilku celom. Po pierwsze, umożliwia skalowanie tematów z wykorzystaniem więcej niż jednego serwera. Choć każda partycja musi fizycznie mieścić się na maszynie, na której jest przechowywana, wielkość jednostki logicznej (tematu) może znacznie wykraczać poza jedną maszynę. Stosowane rozwiązań pozwalają też zapewnić redundancję i równoległość. Możliwe jest przechowywanie duplikatów partycji, a różne partycje mogą być wczytywane i przetwarzane przez różne maszyny. Liczby w partycjach na rysunku A.3 określają przesunięcie, które w unikatowy sposób określa element partycji. Z tego wynika niemodyfikowalność partycji.

Klaster przechowuje wszystkie opublikowane komunikaty przez czas skonfigurowany w temacie. Po tym czasie dzienniki są usuwane, aby zwolnić miejsce. To oznacza, że konsument może się cofnąć do danych, które już zostały przetworzone. Jest to wykonalne, ponieważ konsument może przechowywać stan związany z przetwarzaniem dziennika. Ten stan odpowiada przesunięciu w każdej partycji.

<sup>3</sup> Dokumentacja projektu Apache Kafka, Apache Projects, <http://kafka.apache.org/documentation.html>.

Wróćmy do odpowiedzi zwróconych przez metodę `send_messages`. Teraz można zrozumieć użyte atrybuty. Atrybuty `topic` i `partition` określają tematy i partycje, w których zapisywane są komunikaty. Atrybut `offset` wyznacza pozycję danych w określonej partycji. Błędy, jakie wystąpiły w trakcie zapisu, są dostępne w atrybucie `error`. Teraz, gdy lepiej rozumiesz wewnętrzne mechanizmy systemu Kafka, pora wrócić do przykładu i zobaczyć, czy uda nam się odnaleźć zapisane komunikaty. Listing A.2 zawiera potrzebny do tego kod.

#### **Listing A.2. Prosta subskrypcja w systemie Kafka**

```
from kafka import KafkaClient, SimpleConsumer
kafka = KafkaClient("localhost:9092")
consumer = SimpleConsumer(kafka, "mygroup", "test")
for message in consumer:
    print(message)
```

Ten kod tworzy obiekt typu `SimpleConsumer`, który jako argumenty przyjmuje obiekt typu `KafkaClient`, a także dwa argumenty tekstowe. Pierwszy z argumentów tekstowych określa *grupę konsumentów*, a drugi związany jest z tematem, z którego konsument będzie pobierał dane. Grupa konsumentów umożliwia koordynowanie zbioru konsumentów, tak by jeden komunikat opublikowany w temacie trafił do tylko jednego członka danej grupy. W tym przykładzie grupa (`mygroup`) obejmuje tylko jednego konsumenta, do którego trafiają wszystkie komunikaty przeznaczone dla tej grupy.

O typie `SimpleConsumer` warto wiedzieć coś jeszcze, co początkowo może nie być oczywiste. Zauważ, że nie podaliśmy żadnych partycji ani przesunięć, od których należy zacząć odczyt. Dzieje się tak, ponieważ tymi kwestiami zarządza właśnie obiekt typu `SimpleConsumer`. Dla każdej partycji z danego tematu i dla każdego konsumenta (oraz każdej grupy konsumentów) rejestrowane jest przesunięcie, do którego dany konsument dotarł. Te dane są zapisywane w narzędziu `ZooKeeper`, które (jak może sobie przypominasz) zostało uruchomione razem z systemem Kafka. Dlatego gdy ponownie uruchomisz przedstawiony kod, początkowe przesunięcie nie będzie równe zero — użytu zostanie pozycja zarejestrowana po ostatnim udanym odczycie. Spróbuj i sam się o tym przekonaj!

#### **Replikacja w systemie Kafka**

Jedną z cech systemu Kafka jest możliwość przechowywania replik partycji, co umożliwia ich równoległe przetwarzanie i chroni przed utratą danych. Zapiszmy teraz dziennik z poziomem replikacji równym 3. Oznacza to, że dane z każdej partycji są zapisywane przez trzy odrębne brokery.

Wymaga to zmiany konfiguracji uruchomieniowej systemu Kafka. Aby uzyskać poziom replikacji równy 3, trzeba uruchomić przynajmniej trzy brokery. W poprzednim przykładzie użyliśmy domyślnej wersji pliku `server.properties`. Teraz zmodyfikujemy go, a także utworzymy dwa dodatkowe pliki konfiguracyjne. Na listingu A.3 pokazane są fragmenty pliku `server.properties`, które wymagają zmiany.

**Listing A.3. Zmiany konfiguracyjne w pliku server.properties dotyczące drugiego brokera**

```
broker.id=1 ←———— Dla każdego brokera trzeba ustawić unikatową liczbę całkowitą  
port=9093  
log.dirs=/tmp/kafka-logs-1
```

Należy zmienić trzy różne parametry, aby za pomocą pliku właściwości uruchomić kilka instancji systemu Kafka. Po pierwsze, trzeba zmodyfikować parametr broker.id. Określa on unikatowy identyfikator brokera w klastrze systemu Kafka. Po drugie, ponieważ wszystkie trzy brokery mają działać w tym samym hoście, trzeba zmienić port. Po trzecie, każda instancja potrzebuje odrębnej lokalizacji na dane. Tę lokalizację określa parametr log.dirs. Na listingu A.3 pokazane są zmiany potrzebnych parametrów dla drugiego brokera. Utworzenie trzeciego pliku *server.properties* pozostawiamy jako ćwiczenie dla Czytelników. Po przygotowaniu tego pliku możesz dodać do klastra dwa nowe brokery za pomocą pokazanych poniżej poleceń (przy założeniu, że dostosowałeś nazwy i lokalizacje do opisanych tu nowych plików konfiguracyjnych):

```
./bin/kafka-server-start.sh ./config/server-1.properties  
./bin/kafka-server-start.sh ./config/server-2.properties
```

Zauważ, że ponieważ w pliku *server.properties* nie zmodyfikowaliśmy szczegółów dotyczących narzędzia ZooKeeper, wszystkie brokery będą komunikować się z tą samą instancją tego narzędzia. Nie trzeba zmieniać konfiguracji, aby zapewnić działanie brokerów w ramach tego samego klastra. Trzeba jednak wykonać dodatkowe operacje, aby utworzyć temat z nowym poziomem replikacji. Przejdź do katalogu systemu Kafka i wywołaj następującą instrukcję:

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 \  
--replication-factor 3 --partitions 1 --topic test-replicated-topic
```

To spowoduje utworzenie nowego tematu *test-replicated-topic*. Ten temat obejmuje jedną partycję i działa z poziomem replikacji równym 3. Aby się o tym przekonać, wywołaj następujące polecenie:

```
./bin/kafka-topics.sh --describe --zookeeper localhost:2181
```

Jeśli wszystko przebiegło poprawnie, zobaczysz dane wyjściowe podobne do tych z listingu A.4.

**Listing A.4. Podsumowanie informacji o temacie z systemu Kafka**

```
Topic:test  
PartitionCount:1  
ReplicationFactor:1  
Configs:  
    Topic: test  
    Partition: 0 Leader: 0 Replicas: 0 Isr: 0  
  
Topic:test-replicated-topic  
PartitionCount:1  
ReplicationFactor:3
```

Configs:

```
Topic: test-replicated-topic
Partition: 0 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
```

Warto poświęcić chwilę na zapoznanie się z tymi informacjami. Pomoże to w zrozumieniu dalszej części dodatku. Te dane wyjściowe zawierają podsumowanie tego, co zrobiliśmy z danymi. Na listingu A.1 automatycznie utworzyliśmy temat test o parametrach domyślnych określonych w pliku *server.properties*. W czasie, gdy powstaje ta książka, domyślnie tworzona jest jedna partycja na temat, a współczynnik replikacji wynosi 1 (co oznacza brak replikacji). Ostatnia kolumna, Isr, to akronim od określenia *In-Sync-Replicas* (czyli synchronizowane repliki). To zagadnienie i liderów omawiamy w następnym punkcie. W danych widoczny jest też dodany ostatnio temat *test-replicated-topic*. Zawiera on tylko jedną partycję, ale o współczynniku replikacji równym 3. Aby publikować komunikaty w nowym temacie, należy zmodyfikować kod z listingu A.1 (zobacz listing A.5).

#### **Listing A.5. Publikowanie w systemie Kafka z wykorzystaniem klastra**

```
from kafka import KafkaClient, SimpleProducer
kafka = KafkaClient("localhost:9092")
producer = SimpleProducer(kafka,
    async=False,
    req_acks=SimpleProducer.ACK_AFTER_CLUSTER_COMMIT,
    ack_timeout=2000)

producer.send_messages("test-replicated-topic", "Witamy w klastrze w systemie Kafka!")
producer.send_messages("test-replicated-topic", "Replikowany komunikat.")
producer.send_messages("test-replicated-topic", "Podobnie jak ten!")
```

**Obiekt typu SimpleProducer z opcjonalnymi parametrami. Tu używane są wywołania synchroniczne, a kod oczekuje na potwierdzenie przesłania komunikatu przez cały klaster. Limit czasu oczekiwania wynosi dwie sekundy.**

Tu pokazany jest inny sposób tworzenia obiektów typu *SimpleProducer*. Kod konfiguruje komunikację synchroniczną, co blokuje dalsze operacje do czasu potwierdzenia przesłania nowego komunikatu przez cały klaster i zwrócenia informacji o powodzeniu przez wspomniany obiekt. Zauważ, że można też ustawić opcję *async=True*. W takiej sytuacji obiekt typu *SimpleProducer* nie musi oczekiwania na odpowiedź.

Po uruchomieniu kodu z listingu A.5 zamknij brokerą powiązanego z plikiem *server-1.properties* i ponownie uruchom kod z listingu A.2. Tym razem zamiast tematu "test" użyj tematu "test-replicated-topic". Co zauważyleś? W tym scenariuszu nadal pojawiają się trzy nowe komunikaty, począwszy od "Witamy w klastrze w systemie Kafka!". Dlaczego tak się dzieje? Przyjrzyjmy się temu dokładniej. Gdy sprawdzisz zawartość tematów (za pomocą przedstawionej wcześniej instrukcji *./bin/kafka-topics.sh*), powinieneś zobaczyć zaktualizowane podsumowanie danych pokazane na listingu A.6.

#### **Listing A.6. Podsumowanie informacji o tematach z systemu Kafka (z nieaktywną repliką)**

```
Topic:test
PartitionCount:1
ReplicationFactor:1
Configs:
```

```
Topic: test Partition: 0 Leader: 0 Replicas: 0 Isr: 0
```

```
Topic:test-replicated-topic
PartitionCount:1
ReplicationFactor:3
Configs:
    Topic: test-replicated-topic Partition: 0 Leader: 0 Replicas:
    ↪0,1,2 Isr: 0,2
```

Widać tu, że wartość Isr zmieniła się z 0,1,2 na 0,2. Może zauważysz też (choć tu nie jest to widoczne), że zmienił się lider. Obie te zmiany związane są z nieaktywnością brokera 1, który został zamknięty. Dokładniej omawiamy to w dalszych punktach.

## POTWIERDZANIE PRzesŁANIA KOMUNIKATÓW

Zanim omówimy niskopoziomowe szczegóły replikacji, warto poświęcić czas na zrozumienie mechanizmu przesyłania komunikatów i różne poziomy potwierdzeń dostępne w klastrze. Należy zacząć od tego, że występują dwa rodzaje komunikacji: asynchroniczna i synchroniczna. W podejściu synchronicznym komunikaty są rozsyłane w tym samym wątku, w którym wywołano producenta. Producent może zablokować dalsze operacje i oczekiwać na odpowiedź przed przesaniem kolejnych komunikatów. W komunikacji asynchronicznej generowane są nowe wątki, które równolegle przesyłają komunikaty, bez blokowania dalszego kodu. Choć model synchroniczny jest najmniej bezpiecznym sposobem przesyłania komunikatów, umożliwia uzyskanie bardzo wysokiej przepustowości (komunikaty są łączone w porcje, aby zmniejszyć koszty związane z komunikacją). Dlatego gdy akceptowalne są okazjonalne niepowodzenia, komunikacja asynchroniczna może okazać się najlepszym rozwiązaniem.

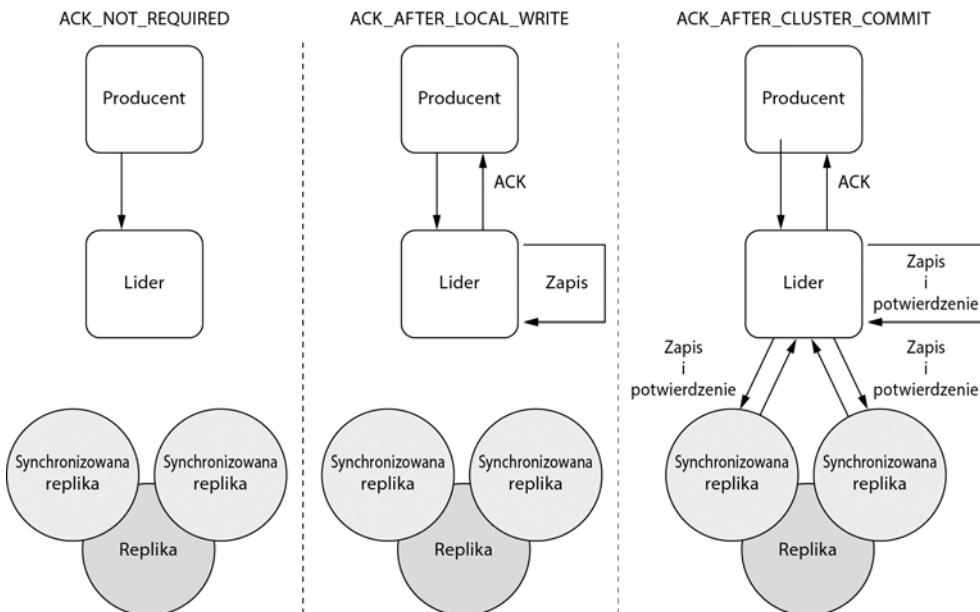
W tym przykładzie zakładamy, że kod działa synchronicznie. W tym modelu w trakcie tworzenia obiektu typu `SimpleProducer` dostępne są trzy możliwości<sup>4</sup>:

- `ACK_NOT_REQUIRED`. W tym trybie obiekt typu `SimpleProducer` nie generuje wielu wątków na potrzeby przesyłania danych; nie oczekuje też na potwierdzenie od lidera obsługującego docelowy temat. Prawie można powiedzieć, że to rozwiązanie łączy najgorsze cechy wszystkich technik, ponieważ brak potwierdzeń oznacza, że nie można zagwarantować otrzymania danych, a brak asynchroniczności powoduje duże obciążenie jednego wątku.
- `ACK_AFTER_LOCAL_WRITE`. Ta opcja gwarantuje, że lider przed odesaniem odpowiedzi do producenta zapisał dane w lokalnym dzienniku (choć jeszcze ich nie zatwierdził). Zauważ, że ta opcja nie gwarantuje spójności wszystkich replik, ponieważ jeśli lider bezpośrednio po wysłaniu potwierdzenia ulegnie awarii, repliki nigdy nie otrzymają nowych danych. W tym trybie uzyskiwane są gwarancje pośrednie, co jest kompromisem między szybkością zapisu i utrwalaniem danych.
- `ACK_AFTER_CLUSTER_COMMIT`. Ta opcja gwarantuje, że lider zwróci potwierdzenie dopiero po zatwierdzeniu lokalnego komunikatu. Oznacza to, że cały zestaw

<sup>4</sup> <http://kafka-python.readthedocs.org/en/latest/usage.html>.

synchronizowanych replik (zauważ, że nie musi to być kompletny zestaw wszystkich replik) potwierdził otrzymanie danych i zostały one lokalnie zatwierdzone. Ten model zapewnia największe gwarancje dostępności danych, ale — zgodnie z oczekiwaniemi — powoduje większe opóźnienie między przesłaniem komunikatu przez producenta a potwierdzeniem otrzymania danych. Jeśli choć jedna z synchronizowanych replik pozostaje dostępna, ta opcja gwarantuje dostępność danych.

Na rysunku A.4 te tryby potwierdzania pokazane są w formie graficznej.



**Rysunek A.4.** Różne poziomy potwierdzień w systemie Kafka. Od lewej do prawej zapewniają one coraz wyższy poziom dostępności danych kosztem opóźnienia związanego z potwierdzieniami

#### NA ZAPLECZU: REPLIKACJA, LIDERZY I SYNCHRONIZOWANE REPLIKI

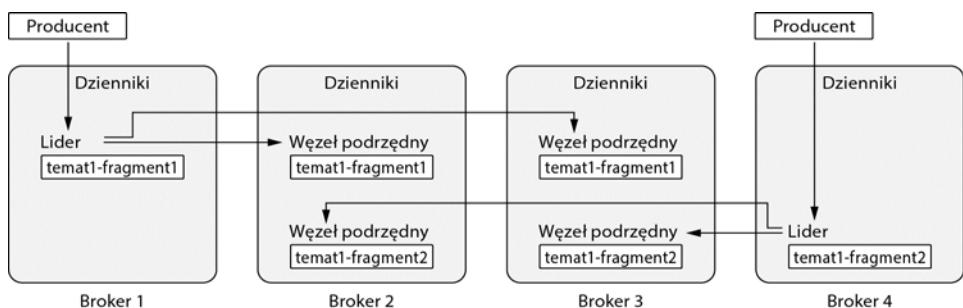
Wspomnieliśmy już o modelu replikacji w systemie Kafka. Pora przyjrzeć się jego szczegółom. Wiesz już, że tematy składają się z partycji, a każda z nich musi być przechowywana przez jednego brokera. Wiesz też, że tematy można tworzyć z określonym współczynnikiem replikacji, określającym poziom nadmiarowości danych.

Choć współczynnik replikacji można ustawić na poziomie tematu, kontroluje się go na poziomie partycji. Jeśli wrócisz do listingu A.6, zobaczysz ustawienie dla każdej partycji. Tu akurat tak się składa, że replikowany temat zawiera tylko jedną partycję używaną do celów demonstracyjnych. Dla partycji 0 tworzone są trzy repliki, ale tylko dwie z nich są zsynchronizowane. Liderem jest replika 0. Przyjrzyjmy się tej konfiguracji z perspektywy brokera.

Dla każdej partycji określany jest lider. Jest on w pewnym sensie właścicielem danej partycji. Wszystkie operacje odczytu i zapisu w danej partycji odbywają się za pośrednictwem lidera. Dlatego gdy producent publikuje komunikat w partycji, najpierw

przesyła go do lidera, który przed wysłaniem odpowiedzi może (choć nie musi) oczekwać na zatwierdzenie danych we własnym dzienniku i w dziennikach pozostałych replik. Ten protokół i dostępne opcje opisaliśmy wcześniej.

Zauważ, że potwierdzanie otrzymania komunikatów i wewnętrzne mechanizmy synchronizacji to różne, choć powiązane ze sobą zagadnienia. Są one oddzielone od siebie w tym sensie, że kod może ustawić dowolny tryb potwierdzania otrzymania komunikatów, ale nie ma to wpływu na próby zapewniania spójności przez klaster. Producent może jednak blokować pracę kodu do czasu wystąpienia określonych zdarzeń z protokołu replikacji i modyfikować dalsze działanie aplikacji w zależności od określonych skutków. To, w jaki sposób zapewniana jest synchronizacja replik (nawet w sytuacji tymczasowej lub trwałe awarii sieci), najlepiej pokazać na przykładzie synchronizowanych replik zaczerpniętym z pracy Juno Rao<sup>5</sup> (zobacz rysunek A.5).



**Rysunek A.5.** Replikacja w systemie Kafka (przykład zaczerpnięty z pracy Rao). W tym przykładzie występują cztery brokery i jeden temat o dwóch fragmentach. Broker 1 jest liderem dla partycji 1, a broker 4 jest liderem dla partycji 2

Na rysunku A.5 pokazany jest klaster w systemie Kafka obejmujący cztery brokery. Ten klaster zarządza jednym tematem o dwóch partycjach. Współczynnik replikacji dla tego tematu wynosi 3. Wyobraź sobie, że chcesz zapisać dane w partycji 1. Dane są przekazywane do brokeru 1, po czym za pomocą zbioru synchronizowanych replik zapewniana jest spójność danych w brokerach 1, 2 i 3.

Synchronizowane repliki to te, w których dzienniki komunikatów są aktualne względem dzienników lidera. Dlatego lider zawsze musi należeć do zbioru synchronizowanych replik. Gdy komunikat jest publikowany w maszynie lidera, wysyła ona komunikat do węzłów podrzędnych i oczekuje na zatwierdzenie komunikatu przez wszystkie repliki. Jeśli któryś z replik nie uda się zatwierdzić komunikatu, jest ona usuwana z zestawu synchronizowanych replik, a proces zatwierdzania kończy się z niepełną replikacją.

Zastanów się nad konkretnym przykładem opartym na rysunku A.5. Jeśli producent chce zapisać dane w temacie temat1-fragment1, najpierw komunikuje się z brokerem 1. Sposób blokowania używany przez producenta zależy od ustawionego poziomu potwierdzania komunikatów. Niezależnie od tego ustawienia system stara się zsynchronizować zbiór replik. W standardowym scenariuszu zbiór synchronizowanych replik powinien być równy całemu zbiorowi replik partycji. Dlatego broker 2 i broker 3 komunikują się

<sup>5</sup> Jun Rao, *Intra-cluster Replication inApache Kafka*, 2 lutego 2013, <http://mng.bz/9Z99>.

z brokerem 1 w celu przesyłania, zapisu i zatwierdzania dodawanych danych. Następnie broker 1 zatwierdza komunikat lokalnie i operacja się kończy. Jeśli jednak nie można skontaktować się z brokerem 3 (na przykład przekroczył został limit czasu oczekiwania), broker 1 usuwa brokera 3 z zestawu synchronizowanych replik, zatwierdza komunikat lokalnie i w brokerze 2, a potem kończy operację. W tej sytuacji dzienniki komunikatów w brokerach 1 i 2 są zsynchronizowane, natomiast broker 3 nie jest zsynchronizowany.

Gdy broker 3 wznowia pracę, musi zaktualizować dane względem zbioru replik. W tym celu komunikuje się z liderem (broker 1), by ustalić brakujące informacje. Na potrzeby tego procesu lider przechowuje tzw. *high watermark*, czyli wskaźnik do ostatniego zatwierdzonego komunikatu w zbiorze synchronizowanych replik. Broker 3 przycina dziennik komunikatów na tym poziomie, aktualizuje dane względem lidera i jest ponownie dodawany do zbioru synchronizowanych replik. Istnieje też inna możliwość — awaria samego lidera. Ponieważ narzędzie ZooKeeper służy do wykrywania awarii i zarządzania konfiguracją klastra, odpowiada też za wybór nowych liderów po awarii pierwotnych. Wtedy ZooKeeper musi poinformować wszystkie repliki o nowym liderze, tak by wiedziały, którego brokera powinny obserwować.

Warto zauważyć, że w tym modelu może nastąpić utrata danych. Na przykład jeśli lider stanie się niedostępny, dane zapisane, ale niezatwierdzone, zostaną utracone po wyborze nowego lidera. System Kafka został zaprojektowany pod kątem wysokiej przepustowości i zapewniania nadmiarowości w centrach danych. Stosowany w nim sposób synchronizacji sprawia, że system gorzej sprawdza się, gdy opóźnienie między replikami jest wysokie. Dlatego prawdopodobieństwo zmian elementów zbiorów synchronizowanych replik i wyboru nowego lidera powinno być niskie. W większości aplikacji rejestrujących zdarzenia niewielka utrata danych jest akceptowalna, ponieważ w zamian zapewniana jest wysoka przepustowość. W artykule, z którego pochodzi przedstawiony przykład, znajdziesz dużo bardziej kompletnie omówienie replikacji. Dlatego odsyłamy Cię do pracy Rao po więcej informacji.

### **Grupy konsumentów, równoważenie i kolejność**

Na listingu A.2 przedstawiliśmy zagadnienie grup konsumentów w systemie Kafka. Ten mechanizm umożliwia stosowanie Kafki zarówno jako kolejki, gdzie wielu konsumentów przetwarza elementy z pierwszej pozycji, a każdy komunikat trafia do jednego konsumenta, jak i jako systemu typu publikuj-subskrybuj, gdzie każdy komunikat jest rozsyłany do wszystkich konsumentów.

Pamiętaj, że po publikacji w temacie jednego komunikatu gwarantowane jest tylko dostarczenie go do jednego konsumenta z grupy. Dlatego gdy istnieje jeden temat i jedna grupa, system działa jak kolejka. Gdy każdy konsument należy do odrębnej grupy, Kafka działa jak system typu publikuj-subskrybuj, a każda grupa otrzymuje ten sam zestaw komunikatów.

Każda partycja jest przypisywana do jednego konsumenta z grupy, tak by tylko ten jeden konsument z tej grupy otrzymywał komunikaty z danej partycji. Dalej zobaczysz, że gdy partie są odpowiednio rozdzielone, to rozwiązanie pozwala zrównoważyć

obciążenie na poziomie obsługi komunikatów, choć powoduje przy tym, że maksymalna liczba konsumentów, którzy mogą przetwarzarć dane z tematu, jest ograniczona do liczby partycji.

Warto zauważyć, że Kafka zapewnia gwarancje kolejności tylko w ramach partycji. Intuicyjnie można przyjąć, że jeśli  $a$  zostało zapisane w partycji przed  $b$ , przetwarzanie tych danych powinno odbywać się w tej samej kolejności. Kafka nie daje jednak gwarancji kolejności dla zestawu partycji. Gdy konsumenti wczytują dane z wielu partycji, może się okazać, że najpierw odczytany zostanie komunikat zapisany później (jeśli oba zostały pobrane z innych partycji). Tak więc Kafka pomija niektóre gwarancje typowe dla kolejek, by umożliwić wyższą przepustowość dzięki równolegemu przetwarzaniu. W praktyce w większości aplikacji rejestrujących zdarzenia takie gwarancje są wystarczające — pod warunkiem że dane zostaną odpowiednio podzielone na partycje. Możesz na przykład rozdzielać dane o kliknięciach na podstawie wyniku dzielenia modulo identyfikatora pliku cookie lub użytkownika przez jakąś wartość. To gwarantuje równoważenie obsługi partycji, a także to, że wszystkie dane dotyczące danego użytkownika będą znajdowały się w tej samej partycji. Następnie możesz wykorzystać kolejność z tej partycji, aby spróbować określić, w jaki sposób użytkownicy wchodzą w interakcje z reklamami. Przyjrzymy się temu bliżej w następnym punkcie.

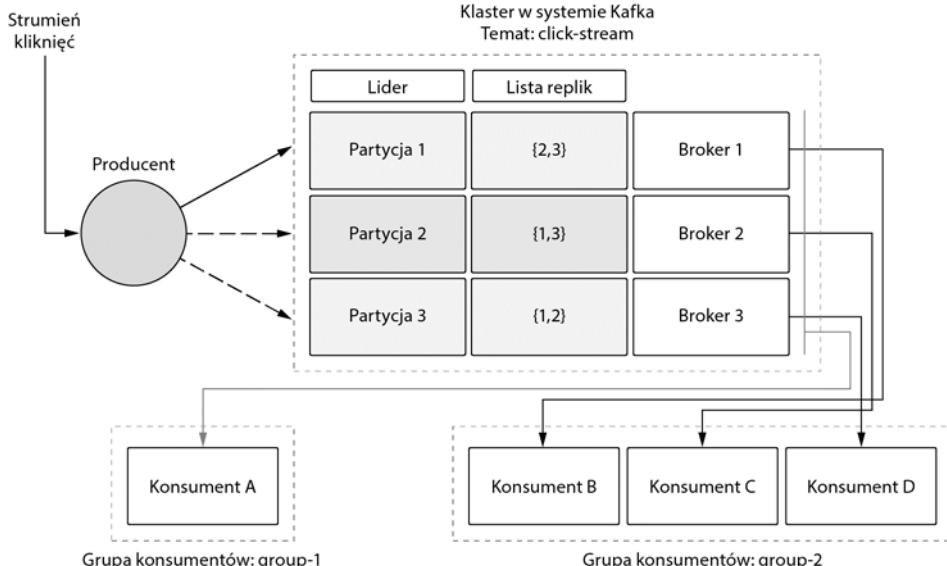
### Łączenie wszystkich elementów

W tym punkcie połączymy wszystkie zagadnienia i przedstawimy przykład ilustrujący elementy opisane do tego miejsca. Ponownie posłużymy się jednym tematem. Tu będzie to temat `click-streams` podzielony na trzy partycje z poziomem replikacji równym 3. Grupy konsumentów będą dwie: jedna z jednym konsumentem (`group-1`) i druga z trzema konsumentami (`group-2`). Zadbamy też o to, by wszystkie kliknięcia danego użytkownika trafiały do jednej partycji. Ilustrację końcowej konfiguracji systemu Kafka przedstawia rysunek A.6.

Wcześniej zaprezentowaliśmy kod potrzebny do utworzenia nowego klastra składającego się z dowolnej liczby brokerów oraz dodania nowego tematu i określenia liczby partycji oraz poziomu replikacji. Na listingu A.2 pokazaliśmy, jak utworzyć konsumenta w grupie. Nie wyjaśniliśmy jednak, jak przeprowadzić niestandardowy podział na partycje (wcześniej za obsługę partycji odpowiadał obiekt typu `SimpleProducer`). Aby zademonstrować tę operację, posłużymy się schematem danych pokazanym na listingu A.7. Jeśli chcesz wykonać omawiany przykład, te dane (w postaci pliku `A1.data`) znajdziesz w materiałach na stronie internetowej poświęconej książce.

Jest to fikcyjny i uproszczony zbiór danych oparty na interakcji ze stroną. Zauważ, że w tym schemacie występują trzy różne kolumny. Pierwsza z nich zawiera unikatowy identyfikator GUID. Druga obejmuje czas interakcji, a trzecia określa jej rodzaj. Są tylko trzy możliwe typy interakcji: `adview` (użytkownik zobaczył reklamę), `click` (użytkownik kliknął reklamę) i `convert` (użytkownik kupił produkt).

Załóżmy, że platforma DSP otrzymuje określoną kwotę za każde kliknięcie, przy czym kliknięcia tego samego użytkownika w ustalonym czasie (na przykład w okresie pięciu minut) są liczone jako jedno. Na listingu A.7 widać, że użytkownik wyróżniony



**Rysunek A.6.** Końcowa wersja klastra w systemie Kafka. Tu używamy niskopoziomowego producenta do samodzielnego podziału strumienia danych na partycje. Klaster obejmuje trzy brokery i trzy partycje oraz działa z poziomem replikacji równym 3. W przykładzie występują dwie grupy konsumentów — jedna (group-1) z jednym konsumentem i druga (group-2) z trzema

#### Listing A.7. Dane o kliknięciach użytkownika

381094c0-9e45-424f-90b6-ad64b06cc184	2014-01-02 17:33:07	click
6615087e-ea19-492c-869c-28fc1fa77588	2014-01-02 17:33:20	advview
d4889090-942b-4790-9831-362051b0847b	2014-01-02 17:34:01	advview
e6688db5-d31f-4ede-985a-115fb51836fb	2014-01-02 17:35:06	advview
e6688db5-d31f-4ede-985a-115fb51836fb	2014-01-02 17:35:30	click
e6688db5-d31f-4ede-985a-115fb51836fb	2014-01-02 17:37:01	click
e6688db5-d31f-4ede-985a-115fb51836fb	2014-01-02 17:39:00	convert
ae6ae5c9-acb2-479b-adcb-0c29623d921b	2014-01-02 17:40:00	advview
1ac384c1-1b2d-4ed0-b467-7c90b7ac42d8	2014-01-02 17:40:01	advview
280bfa16-07ac-49ed-a1a5-9ab50a754027	2014-01-02 17:40:03	click
dda0e95d-9c30-4f60-bb6a-febf05526b83	2014-01-02 17:40:05	advview
8a1204f1-5076-4d4c-8b23-84c77ad541d8	2014-01-02 17:40:10	advview
3bdb8f17-11cc-49cb-94cf-75676be909b7	2014-01-02 17:40:11	advview
69b61156-6c31-4317-aec5-bd48908b4973	2014-01-02 17:40:13	advview
69722471-0532-4f29-b2b4-2f9007604e4f	2014-01-02 17:40:14	advview
00e5edf6-a483-48fa-82ed-fbfa8a6b1e6	2014-01-02 17:40:15	advview
9398d369-6382-4be0-97bc-182b3713745f	2014-01-02 17:40:17	convert
f40c1588-e4e1-4f7d-8ef5-5f76046886fb	2014-01-02 17:40:18	advview
54823527-fe62-4a81-8551-6282309b0a3f	2014-01-02 17:40:20	click
46d6f178-7c11-48c1-a1d7-f7152e7b2f1c	2014-01-02 17:40:26	advview
4c4e545b-d194-4531-962f-66e9d3b6116d	2014-01-02 17:41:00	convert
42b311f5-ba84-4666-a901-03063f7504a9	2014-01-02 17:41:01	advview
bfa28923-c358-4741-bcbf-ff99b640ee14	2014-01-02 17:42:06	advview
54c29b39-5640-49b8-b610-6f2e6dc6bd1b	2014-01-02 17:42:10	convert
edf6c5d2-1373-4dbb-8528-925d525b4a42	2014-01-02 17:43:03	click
f7f6752f-03bf-43f1-927c-8acdafdf235e2	2014-01-02 17:43:11	advview
f4b7c0a6-b209-4cc4-b4e7-395489e0e724	2014-01-02 17:43:19	click

**Emisje reklam  
(advview), kliknięcia  
(click) i konwersje  
(convert)  
dla jednego  
użytkownika  
z krótkiego  
przedziału czasu.**

na listingu zobaczył reklamę i dwukrotnie kliknął ją przed konwersją (prawdopodobnie kupił reklamowany produkt!). Widać też, że odstęp między kliknięciami wynosi kilka minut. Dlatego potrzebne są dodatkowe operacje, jeśli platforma ma pominąć drugie kliknięcie i pozostawić w strumieniu interakcji tylko pierwsze.

Na listingu A.1 w początkowej części dodatku komunikaty publikowaliśmy w temacie test obejmującym jedną partycję. Teraz chcemy utworzyć więcej partycji i upewnić się, że wszystkie informacje dotyczące jednego użytkownika trafią do tej samej partycji. Ponadto idealnym rozwiązaniem byłoby zrównoważenie obsługi danych dzięki rozdzieleniu ich między dostępne partycje. Kod znajdziesz na listingu A.8.

#### Listing A.8. Niestandardowy podział na partycje za pomocą producenta

```
from kafka import KafkaClient
from kafka.common import ProduceRequest
from kafka.protocol import KafkaProtocol,create_message
kafka = KafkaClient("localhost:9092")
f = open('A1.data', 'r')                                     ← Otwieranie pliku z przykładowym strumieniem kliknięć.
for line in f:                                              ← Iteracyjne przetwarzanie danych.
    s = line.split("\t")[0]                                    ← Tworzenie skrótu identyfikatora GUID i dzielenie go modulo 3, aby uzyskać liczbę od 0 do 2.
    part = abs(hash(s)) % 3
    req = ProduceRequest(topic="click-streams",
                          partition=part,messages=[create_message(s)]) ← Tworzenie obiektu typu ProduceRequest określającego temat (click-streams) i partycję.
    resp = kafka.send_produce_request(payloads=[req],
                                       fail_on_error=True) ← Przesyłanie danych z odpowiedziami zapisanymi w obiekcie resp.
```

Ten prosty producent dzieli dane na podstawie identyfikatora GUID użytkownika. W produkcyjnej wersji takiego systemu strumień kliknięć pochodziłby bezpośrednio z przeglądarki lub powiązanego kodu. Jednak tu na potrzeby przykłady pobieramy dane bezpośrednio z pliku.

Jeśli broker z listingu A.6 wciąż jest nieaktywny, uruchom go ponownie. Jeśli chcesz wykonać opisany przykład, utwórz temat `click-streams` z poziomem replikacji równym 3. Zrób to przed uruchomieniem kodu z listingu A.8. W przeciwnym razie automatycznie utworzony zostanie temat o za małej liczbie partycji.

```
bin/kafka-topics.sh --create \
--zookeeper localhost:2181 \
--replication-factor 3 \
--partitions 3 \
--topic click-streams
```

Pamiętaj, że celem jest sprawić, by wszystkie dane dotyczące jednego użytkownika trafiały do tej samej partycji. Na listingu A.8 widać, że kod dzieli skrót każdego identyfikatora GUID modulo 3 ( $\% 3$ ). Dlatego dla wszystkich danych użytkownika uzyskiwany jest ten sam wynik, co pozwala zapisać je w tej samej partycji. Ta partycja nie jest przeznaczona wyłącznie dla jednego użytkownika — mogą do niej trafić także dane innych osób. Jednak w tym systemie jest to akceptowalne, ponieważ liczba partycji jest znacznie mniejsza niż liczba użytkowników. Ta technika niestandardowego podziału na partycje

ma też inną przydatną cechę. Jeśli identyfikatory GUID mają rozkład losowy, to samo dotyczy też wyników ich dzielenia modulo 3. To oznacza, że dane użytkowników będą w miarę równomiernie rozdzielone między partycje. Jest to ważne, aby na etapach publikacji i subskrypcji równomiernie zrównoważyć obciążenie między dostępne zasoby.

Biorąc za punkt wyjścia grupy konsumentów z rysunku A.6, przyjrzyjmy się przydziałowi konsumentów do partycji, aby upewnić się, że jeden konsument rzeczywiście otrzymuje wszystkie dane użytkownika. Grupa group-1 obejmuje tylko konsumenta A, dlatego pobiera on wszystkie dane z tematu click-streams. Odbywa się to w wyniku okresowej komunikacji z liderami wszystkich partycji z tematu. Ponieważ ten konsument otrzymuje dane z każdej partycji, to wszystkie dane dowolnego użytkownika trafiają do jednego konsumenta.

Ciekawsza jest grupa group-2, obejmująca konsumentów B, C i D. Pamiętaj, że grupa konsumentów sprawia, iż dany komunikat trafia do tylko jednego konsumenta z grupy. Odbywa się to w wyniku przypisania partycji do konsumenta. Lider każdej partycji komunikuje się z jednym konsumentem. Broker 1 (lider partycji 1) jest przypisany do konsumenta B, broker 2 do konsumenta C, a broker 3 do konsumenta D. Zauważ, że jest to maksymalna liczba konsumentów w tym przykładzie, ponieważ liczba konsumentów w grupie nie może przekraczać liczby partycji w powiązanym temacie.

Z powodu zastosowanej niestandardowej strategii podziału na partycje wiadomo, że komunikaty dotyczące jednego użytkownika otrzymuje tylko jeden konsument. Dzięki temu można na przykład przeprowadzić eliminację powtarzających się kliknięć (w celu usunięcia błędnych kliknięć i ustalenia, jaki rachunek platforma DSP powinna wyświetlić klientowi), ponieważ wszystkie potrzebne dane są przetwarzane przez jednego konsumenta, a zapewniane przez system Kafka gwarancje dotyczące kolejności oznaczają, że zdarzenia są pobierane zgodnie z kolejnością ich publikowania w partycji.

Możliwa jest też pośrednia sytuacja, której tu nie opisaliśmy. Gdy dwóch konsumentów przypada na trzy partycje, jeden z konsumentów odpowiada za dwie partycje. Wtedy przypisanie konsumenta do partycji odbywa się losowo. Wpływa to na równoważenie obciążenia konsumentów, ale zachowywana jest gwarancja, że jeden konsument otrzyma wszystkie dane jednego użytkownika.

### **Ocena systemu Kafka – rejestrowanie danych w dużej skali**

Do tej pory omawialiśmy system Kafka na podstawie klienta kafka-client, który pozwala komunikować się z klastrem za pomocą Pythona. To rozwiązanie przedstawiliśmy jako zastępnik samodzielnie opracowanej usługi przesyłania dzienników, która choć prosta do zrozumienia, okazuje się nieoptymalna w opisywanym scenariuszu. Dla kompletności przykładu i w celu zilustrowania projektu Kafki porównujemy rozwiązanie oparte na tym systemie z naiwną usługą. Pozwala to pokazać, że Kafka pomaga uzyskać lepsze wyniki w każdym z aspektów wymienionych na początku dodatku.

Jedną z najważniejszych cech aplikacji sieciowych jest możliwość obsługi dużych ilości danych. W naiwnym rozwiązaniu umożliwienie równoległego przetwarzania danych wymaga dodania do systemu kodu odpowiedzialnego za koordynowanie operacji, w którym łatwo o błąd. Zwróciliśmy też uwagę na to, że przetwarzane równolegle

dane trzeba na pewnym etapie ponownie połączyć. Kafka wykonuje te zadania automatycznie. Dzięki podziałowi danych na partie natychmiast uzyskujemy poziom równoległości odpowiadający liczbie partycji. Narzędzie ZooKeeper odpowiada za wykrywanie węzłów i zarządzanie nimi, a wyniki można łączyć po stronie konsumenta. Aby zmaksymalizować równoległość, w systemie Kafka należy skonfigurować odrębne, niepowiązane ze sobą tematy, o niezależnych wzorach korzystania z danych. Dzięki temu grupy konsumentów mogą pracować równolegle i potencjalnie przetwarzać dane w ramach ich pobierania.

Ważnym aspektem oceny naiwnego rozwiązania była skalowalność, ponieważ zależała ściśle od wybranej architektury. Więcej kliknięć w danym obszarze może oznaczać konieczność zwiększenia mocy obliczeniowych lokalnego serwera, co z kolei może wpływać na etap agregowania dzienników. System Kafka jest pomocny w takich sytuacjach, ponieważ pozwala dodać brokery do klastra i zrównoważyć obsługę partycji. Można też zwiększyć liczbę partycji, by optymalnie wykorzystać pulę brokerów. Dzięki temu skalowanie jest skcentralizowane na poziomie klastra, a nie rozproszone na różnych etapach ścieżki przesyłania dzienników. Ponadto system Kafka można skonfigurować na potrzeby obsługi wielu klastrów, wykorzystując kopie lustrzane. Wtedy opóźnienie między klastrami może być większe. Technika ta nadaje się do synchronizowania informacji między centrami danych.

Dużą zaletą systemu Kafka jest zapewnianie trwałości danych. Wyjaśniliśmy już, w jaki sposób Kafka replikuje dane. Gdy poziom replikacji jest wyższy niż 1, to podejście zapewnia odporność na błędy. Wtedy nawet całkowita awaria brokera nie wpływa na przepływ danych w systemie. Natomiast w naiwnym rozwiązaniu trzeba ręcznie tworzyć kopie danych i zarządzać nimi. Choć jest to wykonalne, powoduje znaczny wzrost złożoności usługi zarządzania plikami.

W naiwnym rozwiązaniu opóźnienie nie jest parametrem, który można modyfikować. Po wystąpieniu zdarzenia trzeba odczekać do czasu jego zarejestrowania, zamknięcia pliku i przesłania danych do docelowej lokalizacji. Dopiero potem można uznać zdarzenie za przetworzone. Natomiast w systemie Kafka opóźnienie odczytu i zapisu można zmieniać niezależnie. W etapie zapisu można ustalić poziom potwierdzeń wymaganych przed kontynuowaniem zapisu (zobacz punkt „Replikacja w systemie Kafka” we wcześniejszej części rozdziału). Można nawet zdecydować się na zapis asynchroniczny bez oczekiwania na potwierdzenie. Pozwala to zmniejszyć opóźnienie zapisu kosztem rezygnacji z gwarancji spójności. Zapis odbywa się wtedy szybko, ale trzeba uwzględnić możliwość tego, że awaria lidera doprowadzi do utraty niewielkiej ilości danych po wyborze nowego lidera. Jeśli chodzi o opóźnienie odczytu, to ponieważ rozpoczyna się on od ostatniej zarejestrowanej pozycji, lider może czekać dowolnie długo lub krótko (limitem jest okno przechowywania danych w klastrze) przed wznowieniem pracy i odczytem dalszych danych.

W przedstawionym wcześniej naiwnym rozwiązaniu z projektu wynikało, że konsument jest tylko jeden. Takie rozwiązanie działa jak potok, ponieważ wygenerowane dane ostatecznie trafiają do bazy. Jest to system oparty na przesyłaniu danych (ang. *push-based*). Nie podjęto żadnych działań, aby zapisywać dane w lokalizacji innej od ostatecznej, dlatego ponowne przesyłanie danych lub ich konsumowanie przez kilka

procesów stanowi odstępstwo od normy. Natomiast system Kafka jest zaprojektowany pod kątem wymienionych możliwości. Konsumowanie danych z użyciem wielu procesów jest łatwe — wystarczy utworzyć dodatkową grupę konsumentów. Także ponowne przesyłanie danych jest proste (pod warunkiem że odbywa się w oknie przechowywania danych w klastrze). Te cechy są bardzo przydatne w trakcie tworzenia i instalowania prototypowych wersji inteligentnych algorytmów, ponieważ potrzebne dane są stale dostępne. Dzięki temu można szybko trenować, instalować i testować algorytmy, co skraca czas między generowaniem danych a podejmowaniem decyzji na ich podstawie.

## ***Wzorce projektowe w systemie Kafka***

W ramach podsumowania przedstawionego materiału na zakończenie prezentujemy standardowe wzorce projektowe związane z używaniem systemu Kafka. Do tej pory ustaliliśmy, że Kafka świetnie sprawdza się jako platforma udostępniania danych. Jednak to za mało do budowania przydatnych aplikacji. Po opanowaniu przenoszenia danych z punktu A do B ważne jest, aby móc wykorzystać te dane w użyteczny sposób. W tym kontekście przedstawiamy dwa potencjalne przypadki użycia, które mogą okazać się interesujące w związku ze stosowaniem algorytmów z ostatnich rozdziałów tej książki.

Pierwszy przypadek użycia dotyczy przekształcania danych „w locie”. W drugim pokazujemy, jak przygotowywać dane do pobierania ich za pomocą kwerend w platformie do wsadowego przetwarzania danych. Zauważ, że zadania te się nie wykluczają! Przetwarzanie strumieniowe może mieć miejsce przed przesaniem danych do systemu przetwarzania wsadowego. Oba te procesy mogą też zachodzić równolegle.

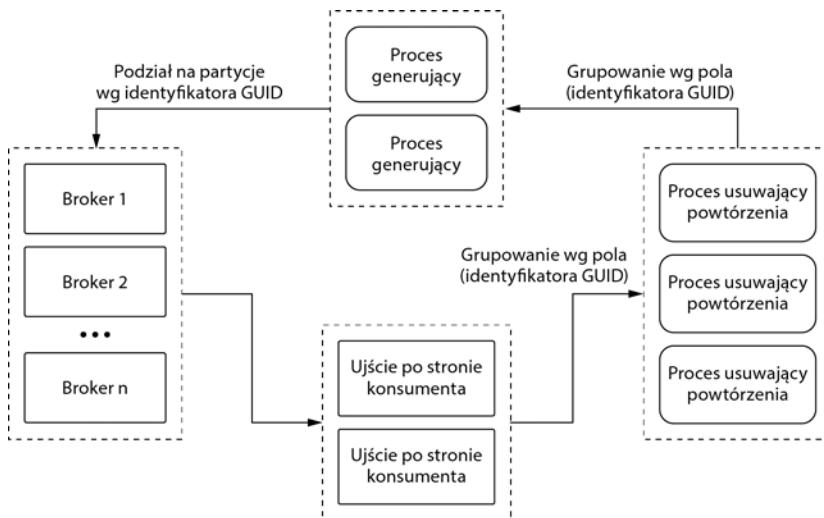
## ***Łączenie systemów Kafka i Storm***

Sam system Kafka nie potrafi przekształcać otrzymywanych danych. Jeśli chcesz przekształcać strumienie danych, musisz połączyć Kafkę z inną technologią — na przykład z systemem Storm<sup>6</sup>. Storm to działający w czasie rzeczywistym system rozproszony do przetwarzania strumieni danych. Umożliwia on przetwarzanie danych w dowolnie złożony sposób za pomocą kilku prostych elementów. Dwa z nich to ujścia (nazywane w tym kontekście *spout*) i procesy (nazywane *bolt*). Ujście pełni funkcję źródła danych, a proces to mechanizm ich przetwarzania, który potrafi przyjmować zestaw strumieni, wykonywać na nich operacje i ewentualnie generować nowy strumień. Te elementy w połączeniu z kilkoma sposobami grupowania i przekazywania danych między etapami (przesyłanie ich w losowej kolejności do następnego etapu, dzielenie ich na partie według wybranego pola, przesyłanie danych do wszystkich odbiorców itd.) stanowią dający duże możliwości sposób przetwarzania danych „w locie”. Takie połączenia mechanizmów nazywamy *topologiami*.

Za pomocą systemów Kafka i Storm można na przykład wyeliminować powtarzające się kliknięcia z tematu click-streams, o czym wspomnieliśmy wcześniej. W tym celu

<sup>6</sup> Sean T. Allen, Matthew Jankowski i Peter Pathirana, *Storm Applied: Strategies for Real-Time Event Processing* (Manning, 2015).

należy zastosować proces dla systemu Kafka, na przykład storm-kafka, i pobrać przy jego użyciu dane z tego systemu oraz wykorzystać je jako źródło danych w topologii. Przykład pokazany jest na rysunku A.7.



**Rysunek A.7.** W tym przykładzie grupa konsumentów obejmująca ujścia pobiera dane z klastra z systemu Kafka podzielonego na partie według identyfikatora GUID. To gwarantuje, że w każdym ujściu znajdują się wszystkie dane dotyczące określonego użytkownika. Za pomocą grupowania według pól dostępnego w systemie Storm można też dzielić dane na podstawie identyfikatorów GUID na etapie przesyłania ich do jednego z kilku procesów usuwających powtórzenia, operujących na rejestrówkach kliknięć. Ponownie grupując dane według identyfikatorów GUID, można przesłać je do kilku producentów, zapisujących pozbawione powtórzeń dane w nowym temacie (też dzielonym według identyfikatorów GUID)

W tej konfiguracji systemy Kafka i Storm są używane do eliminowania ze strumienia powtarzających się kliknięć, które zostały odnotowane w krótkim odstępie czasu. Używana grupa konsumentów zawiera ujścia systemu Storm, które pobierają dane z systemu Kafka tak samo jak dowolny inny konsument, ale przekazują je dalej w sposób skonfigurowany w systemie Storm. Tu używane jest *grupowanie według pól* (według identyfikatora GUID), po czym dane trafiają do trzech procesów odpowiedzialnych za usuwanie powtarzających się kliknięć. To grupowanie (podobnie jak w systemie Kafka) sprawia, że wszystkie dane użytkownika trafiają do jednego procesu. Jest to konieczne, aby można było usunąć powtarzające się kliknięcia. Dane wyjściowe z każdego procesu to pierwotny strumień minus powtarzające się dane. Informacje przesyłane do producentów są ponownie grupowane według pól. Pozwala to zachować na potrzeby producentów kolejność zdarzeń dotyczących poszczególnych użytkowników. Producenti przekazują następnie komunikaty z powrotem do klastra, rozdzielaając je na podstawie identyfikatora GUID między partie nowego tematu, w którym znajdują się dane z wyeliminowanymi powtórzeniami. Ponieważ dane są przetwarzane zgodnie z kolejnością ich rejestrowania, pozbawione powtórzeń dane dotyczące określonego użytkownika powinny trafić do partycji w mniej więcej takiej samej kolejności jak dane z powtórzeniami.

Warto zwrócić uwagę na określenie „mniej więcej”. Nie jest to precyzyjne! System Storm udostępnia kilka poziomów gwarancji dotyczących kolejności danych. W najprostszym trybie komunikaty mogą być pomijane, a nawet przesyłane ponownie (po przekroczeniu limitu czasu oczekiwania). Wtedy w danych wyjściowych powstaje kilka powtórzeń. Aby uzyskać *semantykę dokładnie jednego powtórzenia*, należy zastosować narzędzie Storm Trident. Zapewnia ono dodatkowe gwarancje w trakcie korzystania z topologii. Za pomocą tego narzędzia można zagwarantować, że nowy, pozbawiony powtórzeń temat będzie zawierał dane w kolejności takiej samej jak początkowa (z pominięciem duplikatów kliknąć).

Zwróć uwagę na skalalność tego rozwiązania. Teoretycznie na każdym z czterech etapów można skalować system za pomocą operacji równoległych bez naruszania sposobu jego działania. Grupowanie na podstawie pól w systemach Kafka i Storm tworzy potoki równoległe przetwarzania dla podzbiorów z przestrzeni identyfikatorów GUID. Pozwala to skutecznie skalować rozwiązanie wraz ze wzrostem liczby użytkowników systemu.

Krótką uwagę na temat procesów usuwających powtórzenia: jeśli dane napływają w odpowiedniej kolejności, usuwanie powtarzających się kliknięć to stosunkowo prosty proces. Każde kliknięcie danego użytkownika można zapisać w lokalnej pamięci i wykrywać kolejne kliknięcia tej osoby w określonym przedziale czasu. Te dodatkowe kliknięcia są usuwane, a po wykryciu kolejnego kliknięcia po upływie ustalonego przedziału czasu system usuwa dany element z lokalnej pamięci. To rozwiązanie działa tylko wtedy, gdy w systemie gwarantowane jest zachowanie kolejności zdarzeń dla użytkownika (pomyśl, co się stanie, jeśli komunikat zostanie ponownie przesłany w nieodpowiedniej kolejności do jednego z procesów usuwających powtórzenia i to w dodatku po upływie ustalonego przedziału czasu). Jeśli chcesz zastosować ten wzorzec, musisz starannie przemyśleć liczbę potrzebnych procesów i ilość pamięci dla każdego z nich.

### **Łączenie systemów Kafka i Hadoop**

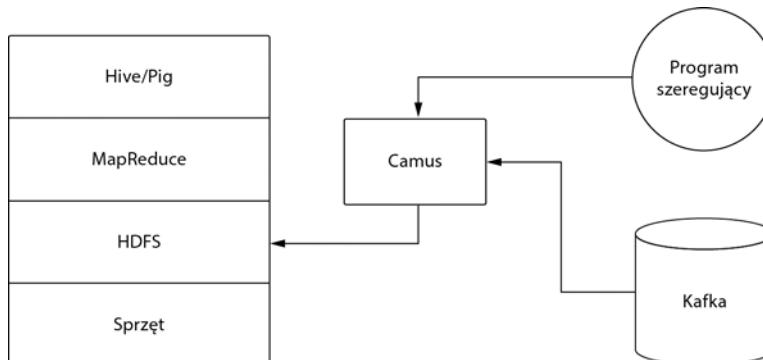
Przetwarzanie danych „w locie” to mechanizm dający duże możliwości. Pozwala on przekształcać dane napływające do systemu. Wadami tego podejścia jest to, że trzeba już na początku ustalić sposób przekształcania danych. W poprzednich przykładach wiedzieliśmy, jak eliminować powtórzenia, dlatego mogliśmy łatwo dodać tę operację do architektury przetwarzania strumienia.

Jednak w wielu sytuacjach początkowo nie wiadomo, co trzeba będzie obliczyć. Ustalenie tego stanowi dużą część pracy tradycyjnych analityków. Aby to ułatwić, należy przekazać dane do systemu Kafka w sposób umożliwiający obsługę kwerend. Do zilustrowania tego procesu posłużymy się Hadoopem (i pomocniczymi technologiami).

Apache Hadoop (<https://hadoop.apache.org>) to platforma umożliwiająca równoległe przetwarzanie dużych zbiorów danych na wielu komputerach. Używany jest w niej prosty paradymat przetwarzania, MapReduce, służący do wykonywania obliczeń w środowisku rozproszonym w znacznie szybszy sposób, niż jest to możliwe na jednej maszynie. Hadoop to platforma o bardzo dużych możliwościach. Jeśli chcesz lepiej zrozumieć, jak z niej korzystać, zapoznaj się z poświęconymi jej pracami.

Z Hadoopem powiązany jest system **HDFS** (ang. *Hadoop Distributed File System*), który tworzy abstrakcję systemu plików działającą w rozproszonym klastrze maszyn. Dostępymi w nich danymi można operować za pomocą modelu MapReduce, aby w stosunkowo krótkim czasie generować wyniki na podstawie bardzo dużych zbiorów danych. Dlatego przesyłanie danych z systemu Kafka do systemu HDFS na potrzeby ich przetwarzania może być przydatnym rozwiązaniem.

Na rysunku A.8 pokazana jest prosta architektura, która umożliwia uzyskanie tego efektu. Używany jest tu projekt Camus (<https://github.com/linkedin/camus>). Działa on jako zadanie w modelu MapReduce i wczytuje dane z tematów z systemu Kafka, po czym zapisuje je w rozproszonym systemie HDFS<sup>7</sup>. Pozwala to w równoległy sposób wczytywać dane z jednego systemu do innego. Jest to niezwykle przydatne, gdy pracujesz z bardzo dużymi zbiorami danych. Dla systemu Kafka Camus to następny konsument. Po uruchomieniu Camus wykrywa temat, wczytuje pozycje partycji z systemu HDFS i przydziela partycje do stałej liczby zadań (przebiegów w modelu MapReduce), które konsumują dane i zapisują je w systemie HDFS. Camus, tak jak inni omawiani do tej pory konsumenti, działa w trybie pobierania (ang. *pull*), dlatego trzeba go uruchomić za pomocą programu szeregującego.



**Rysunek A.8.** Integracja systemu Kafka z Hadoopem. Camus służy do generowania zadań w modelu MapReduce, które równolegle pobierają dane z systemu Kafka i umieszczają je w systemie HDFS. Zadania z projektu Camus trzeba uruchamiać okresowo, dlatego używany jest program szeregujący. Do wywoływania kwerend dotyczących zaimportowanych danych można używać wysokopoziomowych języków, takich jak Hive i Pig

Gdy dane są dostępne w systemie HDFS, można uruchomić zadania w modelu MapReduce lub zarejestrować dane za pomocą wysokopoziomowych języków kwerend, takich jak Hive (<https://hive.apache.org>) lub Pig (<https://pig.apache.org>) rozwijane przez organizację Apache. Takie języki zapewniają znajomy paradymat programowania analitykum uruchamiającym jednorazowe kwerendy dotyczące zaimportowanych danych.

<sup>7</sup> W trakcie powstawania tej książki projekt Camus został zastąpiony przez projekt nowej generacji o nazwie Gobblin (<https://github.com/linkedin/gobblin>). Stanowi on rozwinięcie projektu Camus i łączy go z kilkoma innymi projektami pobierającymi dane z serwisu LinkedIn.



# *Skorowidz*

---

## A

- adekwatność testów A/B, 191
- agent, 133
- AI, artificial intelligence, 26
- aktywowanie sieci neuronowej, 175
- algorytm
  - adaptacyjnej kwantyzacji wektorowej, 103
  - GMM, 40, 59
  - k najbliższych sąsiadów, 103
  - k-srednich, 45, 49, 59
  - Lloyda, 45, 49
  - uczenia perceptronów, 160
- algorytmy klasyfikacji, 102, 104
- analityka predykcyjna, PA, 26, 29
- analiza głównych składowych, PCA, 41, 61, 63
- aplikacja Google Now, 21

## B

- bandyta kontekstowy, 209
- biblioteka
  - SVLIBC, 86
  - Vowpal Wabbit, 138
  - VW, 143
- błąd, 171
  - systematyczny, 41, 65
  - średniookwadratowy, 93, 95
- BRBM, Bernoulli Restricted Boltzmann Machines, 176

## C

- cechy, 41, 137
- ciągłe, 114
- kategorialne, 112
- macierzy kwadratowych, 61
- centroid, 49
- CF, collaborative filtering, 76

- CPC, cost per click, 136, 220
- CPI, cost per impression, 136
- CPM, 136, 220
- CTR, 136
- cykl życia
  - inteligentnych algorytmów, 23
  - klasyfikatora, 105

## D

- dane, 41
  - o kliknięciach, 234
  - wyjściowe, 145
- decyzie, 134, 150, 186
- dekompozycja zbioru danych, 64
- dokonywanie wyboru, 185
- dopasowanie, 110
  - plików cookie, 130, 131
- drzewo GDBT, 94
- DSP, demand-side platform, 129
- działanie perceptronu, 162

## E

- elementy struktury odniesienia, 100
- EM, expectation maximization, 40

## F

- filtrowanie kolaboratywne, CF, 68, 76
  - według użytkowników, 77
- FNR, false negative rate, 32
- FPR, false positive rate, 32
- funkcja gęstości prawdopodobieństwa, 200–202
- funkcje
  - aktywacji, 168
  - odległości euklidesowej, 74

**G**

- gaussowski model mieszany, GGM, 40, 52, 55  
 generowanie  
   krzywej ROC, 147  
   rekomendacji, 85  
   sztucznych danych, 181  
 GGM, Gaussian mixture model, 40  
 giełda, 130  
 GLM, generalized linear models, 169  
 GMM, 52

**H**

- HDFS, Hadoop Distributed File System, 241  
 HFT, high-frequency trading, 129  
 histogram rozkładu Gaussa, 53

**I**

- implementowanie wykrywania oszustw, 111  
 importowanie  
   bibliotek, 112  
   zbioru danych, 112  
 informacje  
   o kontekście, 135  
   o przestrzeni reklamowej, 135  
   o użytkowniku, 135  
 inteligencja, 30  
 inteligentna sieć, 19  
 inteligentny algorytm, 21–24, 33  
   cykl życia, 23  
   generalizacja, 36  
   Google Now, 21  
   klasy, 26  
   ludzka intuicja, 36  
   miary skuteczności, 32  
   ocena działania, 30  
   skalowanie, 35  
   wiarygodność danych, 34  
   wielkość, 34  
   wnioskowanie, 34  
 intuicja, 36  
 inżynieria cech, 137

**J**

- jednostki ukryte, 183

**K**

- kalibrowanie modelu, 148  
 KISS, 26  
 klasa  
   RatingCountMatrix, 81  
   SimilarityMatrix, 80  
 klastrowanie, 43  
   zbioru danych, 51, 57  
 klastry, 46  
 klasy inteligentnych algorytmów, 26  
 klasyfikator, 105  
 klasyfikowanie, 97, 122  
 kodowanie  
   cech kategorialnych, 113  
   one-hot, 114  
 konkurs Netflix Prize, 93  
 korelacja, 37  
 koszt  
   kliknięcia, CPC, 136  
   wyświetlenia reklamy, CPI, 136  
 krzywa ROC, 32, 147  
 k-średnie, 45, 49

**L**

- liczba cech, 51  
 lokalizacja, 22

**Ł**

- łączenie systemów, 238, 240

**M**

- macierz  
   błędów, 120  
   kowariancji, 57, 62  
   kwadratowa, 61  
 maksymalizacja wartości oczekiwanej, EM, 40, 49, 55, 65  
 maszyny  
   BRBM, 177  
   RBM, 180  
 metoda  
   Lanczosa, 86  
   make\_ellipses, 60  
   recommend, 78  
 metody dywergencji kontrastowej, 180

- miara  
 pierwiastka błędu średniokwadratowego, 95  
 RMSE, 95  
 skuteczności, 32  
 mieszanie, 142  
 ML, machine learning, 26  
 model  
     Bernoulliego, 176  
     liniowy, 169  
     MCP, 157  
 modelowanie wiedzy, 196  
 monitorowanie reklam, 132
- N**
- narzędzie VW, 138, 145  
 nierówność trójkąta, 72
- O**
- obiekt typu SimpleProducer, 229  
 obliczanie  
     miary RMSE, 96  
     podobieństwa, 70  
     różnic, 107  
 ocenianie  
     działania inteligentnych algorytmów, 30, 32  
     predykci, 31  
     systemu rekomendacji, 94  
 oczekiwany poziom straty, 203, 207  
 oddzielanie cech kategorialnych, 113  
 odległość, 69  
     euklidesowa, 73  
 oferty, 131  
 ograniczona maszyna Boltzmanna, 176  
 osie danych, 60
- P**
- PA, predictive analytics, 26  
 pakiet  
     PyBrain, 172, 183  
     scikit-learn, 41, 42, 147  
 PCA, principal component analysis, 41  
 perceptron, 156, 162, 169  
     dwuwarstwowy, 166  
      wielowarstwowy, 164, 174  
 platforma DSP, 129, 131  
 plik server.properties, 227  
 pliki cookie, 130  
 płaskie struktury odniesienia, 99  
 pobieranie danych, 219
- odejmowanie decyzji, 134, 136, 150, 186  
 podobieństwo, 69, 70, 73, 75  
     prawdopodobieństw, 205  
 potwierdzanie przesłania komunikatów, 229  
 powiadomienie o wygranej, 132  
 prawdopodobieństwo wystąpienia zdarzenia, 110  
 predykcja, 31  
     kliknięć, 138  
 predyktor prawdopodobieństwa, 109  
 problem  
     bandytów, 210  
     klasyfikacji binarnej, 120  
     wielorękiego bandyty, 192  
 prognozowanie  
     kliknięć, 127  
     ocen, 79  
     zdarzeń, 150  
 prognozowany współczynnik kliknięć, 136  
 propagacja wsteczna, 167–170  
 przegląd klasyfikatorów, 101  
 przekleństwo wymiarów, 44  
 przestrzeń cech, 49, 65  
 przesyłanie dzienników, 222  
 przygotowywanie danych, 135, 141  
 publikowanie komunikatów, 224
- R**
- regresja  
     liniowa, 106  
     logistyczna, 106, 117, 143, 169, 181  
 reguła KISS, 26  
 rejestrowanie danych, 221  
 reklama, 132  
     internetowa, 220  
 rekomendacje, 67, 76, 84  
     filtrowanie kolaboratywne, 77  
     generowanie, 85  
     oparte na modelu, 82, 87  
     rozkład SVD, 82, 84, 90  
 replikacja, 226, 230  
 reprezentacja ukrytych jednostek, 182  
 RMSE, root-mean square error, 95  
 ROC, receiver operating characteristic, 32  
 rozkład  
     a priori, 199  
     beta, 198  
     Gaussa, 52, 55  
     macierzy na czynniki, 85  
     prawdopodobieństwa, 56, 202  
     według wartości osobliwych, SVD, 68, 82–85, 90  
 równoważenie, 142

**S**

- SGD, stochastic gradient descent, 142  
 sieci neuronowe, 155  
   perceptron, 156  
   RBM, 179  
   trening, 158  
   wielowarstwowe, 172  
 skala prawdopodobieństwa, 205  
 skalowanie inteligentnej aplikacji, 35  
 skumulowany oczekiwany poziom straty, 206  
 spadek wzduż gradientu, SGD, 142  
 statystyczne algorytmy klasyfikacji, 104  
 stosowanie strategii bayesowskiej, 198  
 strategia  
   bayesowska, 195, 197, 204, 207  
   najpierw epsilon, 193  
   zachłanna, 194  
   zmniejszania epsilonu, 195  
 struktura, 41  
 strukturalne algorytmy klasyfikacji, 102  
 subskrypcja, 226  
 SVD, singular value decomposition, 68, 82–85, 90  
 SVM, support vector machines, 28  
 symetria, 72  
 synchronizowane repliki, 230  
 system  
   Hadoop, 240  
   HDFS, 241  
   Kafka, 224  
     klaster, 234  
     poziomy potwierdzeń, 230  
     publikowanie, 228  
     publikowanie komunikatów, 224  
     rejestrowanie danych, 236  
     replikacja, 226, 231  
     subskrypcja, 226  
     wzorce projektowe, 238  
   Storm, 238  
 systemy  
   podejmowania decyzji, 134, 150  
   rekомendacji, 69, 76  
 sztuczna inteligencja, AI, 26  
 szum, 41

**T**

- teoria propagacji wstecznej, 170  
 testowanie modelu, 146  
 testy A/B, 187, 207

- TNR, negative rate, 32  
 TPR, positive rate, 31  
 transformacje osi danych, 60  
 trening  
   modelu, 137, 143  
   perceptronu, 160  
   sieci neuronowej, 173  
   z wykorzystaniem propagacji wstecznej, 167  
 treningowy zbiór danych, 142  
 tworzenie  
   wielowarstwowego perceptronu, 172  
   zbioru danych, 180

**U**

- uczenie  
   głębokie, 153, 175  
   maszynowe, ML, 26, 28  
   oparte na energii, 179  
 umieszczanie reklamy, 132  
 uogólnione modele liniowe, GLM, 169

**V**

- VW, Vowpal Wabbit, 138  
   format danych, 138

**W**

- wagi sieci neuronowej, 175  
 wartość  
   oczekiwana, 55  
   własna, 61  
 wektor własny, 61  
 wieloręki bandyta, 192, 208  
 wielowarstwowa sieć jednokierunkowa, 164  
   neuronowa, 172  
 wizualizacja sieci głębokiej, 155  
 współczynnik  
   kliknąć, CTR, 136  
   konwersji, 191  
   predykcji, 32  
 wykrywanie oszustw, 106, 111  
 wymagania stawiane agentowi, 133  
 wyniki, 119  
 wzór  
   na błąd, 171  
   na podobieństwo, 75

**Z**

- zarządzanie zbieraniem danych, 222
- zastosowania inteligentnej sieci
  - autonomiczne samochody, 215
  - internet rzeczy, 214
  - opieka zdrowotna, 215
  - sieć semantyczna, 216
  - spersonalizowane fizyczne reklamy, 216
- zbiór danych, 141, 142, 180
- Iris, 41, 42
- MovieLens, 69
- zmienne kategorialne, 117



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄZKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW  
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA  
**Helion SA**