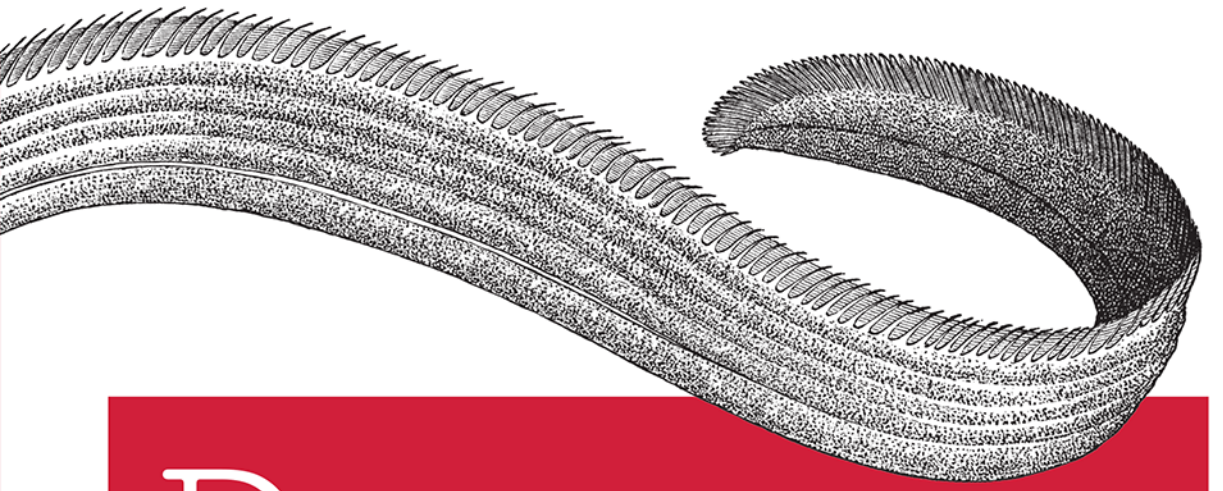
O'REILLY®



Deep Learning

PRAKTYCZNE WPROWADZENIE

Helion

Josh Patterson, Adam Gibson

Tytuł oryginału: Deep Learning: A Practitioner's Approach

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-283-4227-9

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

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

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Podstawy uczenia maszynowego

Stworzyć niezbity fakt z pajęczyny domysłów.
— Neal Stephenson, *Zamieć*

Uczące się maszyny

W ostatniej dekadzie zainteresowanie dziedziną uczenia maszynowego wręcz eksplodowało. Mówi się o nim w programach naukowych, na konferencjach gospodarczych i niemal codziennie w „Wall Street Journal". W całym tym zgiełku często pojawia się pytanie, co można i co chciałoby się osiągnąć za pomocą uczenia maszynowego. Uogólniając, **uczenie maszynowe** opiera się na algorytmach wyodrębniania informacji z surowych danych i reprezentowania ich w formie określonego modelu. Model ten jest następnie wykorzystywany do przetwarzania kolejnych niezamodelowanych danych.

Jednym z modeli stosowanych w uczeniu maszynowym są sieci neuronowe. Sieci te są znane już od prawie 50 lat. Ich podstawowym elementem jest **węzeł**, którego dalekim odpowiednikiem jest neuron w mózgu ssaka. Połączenia pomiędzy węzłami modeluje się podobnie jak połączenia neuronów, które są budowane w miarę upływu czasu (poprzez „uczenie"). W następnych dwóch rozdziałach omówimy dokładniej funkcjonowanie tych modeli.

W połowie lat 80. i na początku 90. poczyniono znaczne postępy w rozwoju architektury sieci neuronowych. Jednak wymagania dotyczące czasu i ilości danych potrzebnych do uzyskania dobrych wyników opóźniały zastosowanie sieci w praktyce, przez co zainteresowanie nimi osłabło. Ale od początku XXI wieku moce obliczeniowe komputerów zaczęły wykładniczo rosnąć. Była to informatyczna „eksplozja kambryjska", w wyniku której pojawiły się niespotykane wcześniej możliwości obliczeniowe. W ciągu tej trwającej dekadę ewolucji narodziła się idea głębokiego uczenia, która okazała się poważnym rywalem dla dotychczasowych idei uczenia maszynowego, wygrywającym wiele konkurencji na tym polu. Do 2017 roku zainteresowanie głębokim uczeniem nie osłabło i dzisiaj w dziedzinie uczenia maszynowego spotyka się je na każdym kroku.

W kolejnych częściach rozdziału przedstawimy naszą definicję głębokiego uczenia. Książka ma taki układ, abyś Ty — praktyk — po wzięciu jej z półki mógł:

- zapoznać się z podstawowymi zagadnieniami z dziedziny algebry liniowej i uczenia maszynowego,
- poznać podstawy sieci neuronowych,
- poznać cztery podstawowe typy sieci głębokich,
- wykorzystać opisane przykłady we własnych, praktycznych odmianach sieci głębokich.

Mamy nadzieję, że zaprezentowany tu materiał będzie dla Ciebie zrozumiały i będziesz go mógł wykorzystać w praktyce. Zacznijmy tę książkę od krótkiego wykładu, czym jest uczenie maszynowe, i od przedstawienia najważniejszych pojęć, które będą Ci potrzebne do lepszego zrozumienia pozostałej części książki.

Jak uczą się maszyny?

Aby opisać, jak uczą się maszyny, musimy najpierw zdefiniować zwrot „uczenie się". W potocznym języku oznacza on „nabywanie wiedzy poprzez studiowanie, doświadczanie lub przejmowanie jej od innych osób". Zawężając nieco punkt widzenia, można powiedzieć, że uczenie maszynowe polega na stosowaniu algorytmów tworzenia strukturalnych opisów danych na podstawie ich przykładów. Komputer sam dowiaduje się czegoś o strukturze surowych danych. *Strukturalny opis* to inny termin określający model danych zawierający informacje o surowych danych. Struktury te, czyli modele, wykorzystuje się do prognozowania nieznanych danych i mogą one mieć różne formy, na przykład:

- drzewa decyzyjnego,
- regresji liniowej,
- wag sieci neuronowej.

W każdej z powyższych form stosuje się inne reguły prognozowania nieznanych danych na podstawie istniejących danych. W przypadku drzewa decyzyjnego tworzy się zestaw reguł w formie drzewiastej struktury, a w regresji liniowej zestaw parametrów reprezentujących dane wejściowe.

Uczenie maszynowe versus eksploracja danych

Pojęcie *eksploracja danych* jest znane od kilku dekad i, jak większość innych pojęć z dziedziny uczenia maszynowego, bywa niepoprawnie interpretowane i stosowane. W kontekście tej książki praktyczna „eksploracja danych" oznacza „wyodrębnianie informacji z danych". Uczenie maszynowe natomiast oznacza algorytmy wykorzystywane w eksploracji danych do tworzenia strukturalnych opisów surowych danych. Poniżej przedstawiamy prostą analogię do eksploracji danych:

- Aby poznać znaczenie jakiegoś pojęcia
  — potrzebne są nam przykładowe surowe dane.
- Przykładowe dane mają postać wierszy lub instancji danych
  — układających się w określoną strukturę.
- Maszyna uczy się danego pojęcia na podstawie struktury danych
  — stosując algorytmy uczenia maszynowego.

Cały powyższy proces można traktować jako „eksplorację danych".

Sieć neuronowa ma tzw. **wektor parametrów**, reprezentujący wagi połączeń pomiędzy węzłami. Ten model szczegółowo opiszemy w dalszej części rozdziału.

Arthur Samuel, pionier sztucznej inteligencji, pracujący w IBM i na uniwersytecie w Stanford (*http://infolab.stanford.edu/pub/voy/museum/samuel.html*), zdefiniował uczenie maszynowe w następujący sposób:

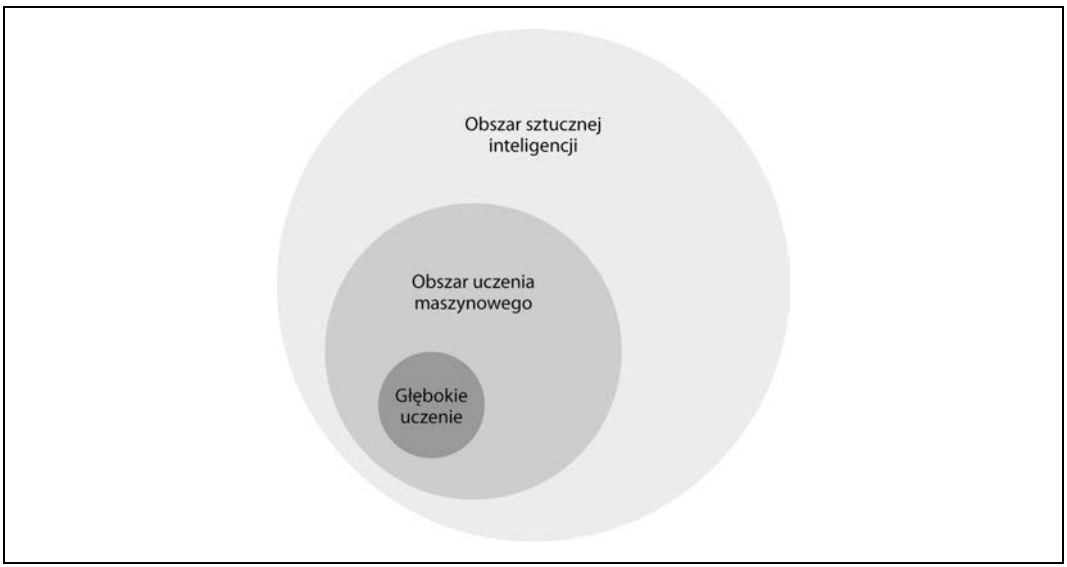
> *Badania mające na celu stworzenie uczących się komputerów bez konieczności ich jawnego programowania.*

Samuel napisał program do gry w warcaby, zmieniający strategię gry na podstawie szacowanego prawdopodobieństwa wygranej lub przegranej w zależności od wykonywanych ruchów. Ten podstawowy schemat szukania wzorców, które prowadzą do wygrania lub przegrania partii, a następnie utrwalania zwycięskich wzorców, do dzisiaj stanowi istotę uczenia maszynowego i sztucznej inteligencji.

Pomysł, aby zbudować maszynę, która potrafi się uczyć, jak osiągać zamierzony cel, zaprzątał umysły uczonych od dziesięcioleci. Prawdopodobnie najlepiej ujęli to prekursorzy nowoczesnej sztucznej inteligencji, Stuart Russell (*https://www2.eecs.berkeley.edu/Faculty/Homepages/russell.html*) i Peter Norvig (*http://norvig.com*) w książce *Artificial Intelligence: A Modern Approach* (*http://aima.cs.berkeley.edu*):

> *W jaki sposób powolny, słaby umysł, czy to biologiczny, czy elektroniczny, może postrzegać, rozumieć, przewidywać i oddziaływać na świat, który jest znacznie większy od niego?*

Ten cytat nawiązuje do idei uczenia się, której inspiracją były procesy i algorytmy spotykane w naturze. Rysunek 1.1 ilustruje relacje pomiędzy sztuczną inteligencją, uczeniem maszynowym i głębokim uczeniem.



Rysunek 1.1. Relacje pomiędzy sztuczną inteligencją, uczeniem maszynowym i głębokim uczeniem

Obszar badań nad sztuczną inteligencją jest szeroki i znany od długiego czasu. Głębokie uczenie stanowi część obszaru uczenia maszynowego, które z kolei jest częścią obszaru sztucznej inteligencji. Przyjrzyjmy się teraz innym korzeniom głębokiego uczenia i sprawdźmy, jak biologia zainspirowała twórców sieci neuronowych.

Biologiczna inspiracja

Biologiczne sieci neuronowe (mózgi) składają się z ok. 86 miliardów połączonych ze sobą neuronów.

Liczba połączeń neuronów w ludzkim mózgu

Uczeni ostrożnie szacują, że w ludzkim mózgu istnieje ponad 500 trylionów połączeń między neuronami. Dzisiejsze największe sztuczne sieci nie mogą się nawet zbliżyć do tej liczby.

Pod względem przetwarzania informacji biologiczny neuron jest fascynującą jednostką, która może przetwarzać i przesyłać informacje za pomocą sygnałów elektrycznych i związków chemicznych. Neuron jest uważany za podstawowy komponent centralnego układu nerwowego złożonego z mózgu i rdzenia kręgowego oraz zwojów obwodowego układu nerwowego. Jak się dowiesz w dalszej części rozdziału, struktura sztucznych sieci neuronowych jest znacznie prostsza.

Porównywanie sieci biologicznych i sztucznych

Biologiczne sieci neuronowe są zdecydowanie (o kilka rzędów wielkości) bardziej skomplikowane niż ich sztuczne odpowiedniki!

Sztuczne sieci neuronowe mają dwie cechy, które upodobniają je pod względem działania do mózgu. Pierwszą jest najbardziej podstawowa jednostka, czyli **sztuczny neuron** (zwany **węzłem**). Sztuczne neurony odpowiadają biologicznym neuronom w mózgu i, podobnie jak one, są stymulowane przez impulsy wejściowe. Sztuczne neurony przesyłają do innych neuronów niektóre (nie wszystkie) informacje, często przekształcone. W miarę czytania tego rozdziału dowiesz się dokładniej, na czym polega przekształcanie informacji w kontekście sieci neuronowych.

Drugą cechą jest możliwość uczenia się przez sztuczne neurony przekazywania tylko przydatnych sygnałów, podobnie jak biologiczne neurony uczą się przekazywać tylko te sygnały, które są potrzebne do osiągnięcia większego celu wyznaczonego przez mózg. W tym rozdziale rozwiniemy tę ideę i dowiesz się, dlaczego sztuczne sieci neuronowe są w stanie odzwierciedlać działanie ich biologicznych odpowiedników za pomocą bitów informacji i funkcji.

Biologiczna inspiracja w informatyce

Biologia stanowi dla informatyków inspirację nie tylko do budowy sztucznych sieci neuronowych. W ciągu ostatnich 50 lat badacze znaleźli w przyrodzie inne obliczeniowe inspiracje, między innymi:

- zwyczaje mrówek,
- zwyczaje termitów[I],
- zwyczaje pszczół,
- algorytmy genetyczne.

Na przykład mrowiska stanowią dla naukowców potężny, rozporoszony komputer, w którym żadna mrówka nie jest pojedynczym punktem awarii (*https://mitpress.mit.edu/books/ant-colony-optimization*). Mrówki nieustannie zmieniają wykonywane przez siebie czynności w celu zminimalizowania obciążenia. Stosują przy tym metaheurystyczne metody, na przykład stygmergię ilościową. Mrówki w skupiskach są w stanie wykonywać żmudne prace, bronić się, budować mrowiska, zdobywać pożywienie, delegując przy tym optymalną liczbę swoich przedstawicieli do każdego zadania, przy czym nie ma jednego osobnika, który by koordynował te prace.

Czym jest głębokie uczenie?

Dla wielu osób zdefiniowanie pojęcia głębokiego uczenia jest nie lada wyzwaniem, ponieważ w ciągu ostatniej dekady zmieniało ono stopniowo swoje znaczenie. Jedna ze stosowanych definicji stanowi, że głębokie uczenie jest cechą „sieci neuronowej złożonej z więcej niż dwóch warstw". Problem z tą definicją polega na tym, że sugeruje ona, że pojęcie głębokiego uczenia istnieje od lat 80. ubiegłego wieku. W rzeczywistości sieci neuronowe przewyższają architektonicznie wcześniejsze formy sieci (i dodatkowo charakteryzują się znacznie większymi mocami obliczeniowymi) i w ciągu ostatnich lat dostarczyły spektakularnych wyników. Poniżej wymienionych jest kilka aspektów ewolucji sieci neuronowych:

- większa liczba neuronów w porównaniu z wcześniejszymi sieciami,
- bardziej skomplikowane połączenia pomiędzy warstwami i neuronami,
- eksplozja mocy obliczeniowych wykorzystywanych do uczenia się,
- automatyczna ekstrakcja cech.

W tej książce przyjmiemy definicję głębokiego uczenia jako cechę sieci neuronowej o dużej liczbie parametrów i warstw. Sieć ta ma strukturę jednego z czterech niżej wymienionych podstawowych typów:

- nienadzorowanej, nienauczonej sieci,
- sieci konwolucyjnej,
- sieci powracającej,
- sieci rekurencyjnej.

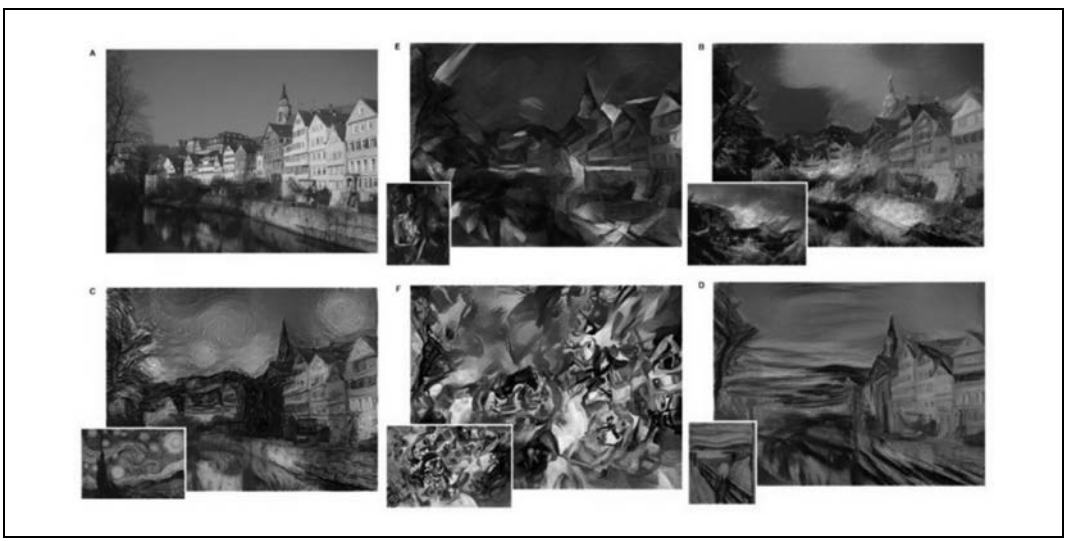
[I] Patterson, *TinyTermite: A Secure Routing Algorithm*, 2008 (*http://scholar.utc.edu/theses/184/*); oraz Sartipi i Patterson, *TinyTermite: A Secure Routing Algorithm on Intel Mote 2 Sensor Network Platform*, 2009 (*http://www.aaai.org/ocs/index.php/IAAI/IAAI09/paper/viewFile/235/1030*).

Ponadto istnieją odmiany wyżej wymienionych typów sieci, na przykład hybrydowa sieć konwolucyjno-powracająca. W tej książce jednak skupimy się na czterech podstawowych typach sieci.

Automatyczna ekstrakcja cech daje ogromną przewagę sieciom głębokiego uczenia nad tradycyjnymi algorytmami uczenia maszynowego. Ekstrakcja cech oznacza dobór charakterystyk zbioru danych, które można uznać dla tych danych za typowe. Kiedyś praktycy uczenia maszynowego spędzali miesiące, lata, a nawet dziesiątki lat na ręcznym tworzeniu rozbudowanych zestawów cech klasyfikujących dane. Gdy w 2006 roku nastąpił wielki wybuch sieci głębokiego uczenia, najnowocześniejsze w owym czasie algorytmy uczenia maszynowego wykorzystywały owoce dziesięcioleci pracy naukowców klasyfikujących dane. Głębokie uczenie przewyższa pod względem precyzji działania niemal wszystkie konwencjonalne algorytmy i wymaga minimalnego zaangażowania człowieka. Dzięki głębokim sieciom zespoły naukowców mogą oszczędzić sobie wiele znoju, potu i łez i poświęcić się innym, bardziej konstruktywnym zajęciom.

Wchodząc głębiej w las

W ciągu ostatnich lat znacznie wzrosła wśród informatyków świadomość przewagi głębokiego uczenia nad innymi technikami uczenia maszynowego. Stało się tak po części dlatego, że idea ta oferuje nie tylko najwyższą precyzję spośród znanych technik, ale także wykazuje twórcze cechy, które fascynują również naukowców niebędących informatykami. Przykładem może być tworzenie dzieł sztuki. Sieci głębokiego uczenia mogą się uczyć dzieł wybitnych malarzy i tworzyć własne obrazy w nauczonym stylu, co demonstruje rysunek 1.2.



Rysunek 1.2. Stylizowane obrazy (Gatys i in.[II], 2015 r.)

[II] Gatys i in., *A Neural Algorithm of Artistic Style*, 2015 (*http://arxiv.org/pdf/1508.06576v1.pdf*).

Dało to początek wielu filozoficznym dyskusjom typu: „Czy maszyny mogą być twórcze?” albo „Czym jest twórczość?”. Zostawiamy Ci te pytania do rozmyślań na później. Uczenie maszynowe rozwijało się przez lata, podobnie jak zmieniają się pory roku: powoli, lecz systematycznie, aż któregoś dnia obudziliśmy się i zobaczyliśmy, że komputery wygrywają turnieje Jeopardy lub Go Grand Master.

Czy maszyny mogą być inteligentne i dorównać pod tym względem ludziom? Czym jest i jak wielka może być sztuczna inteligencja? Na te pytania nie znamy jeszcze odpowiedzi i nie odpowiemy na nie wyczerpująco w tej książce. Chcemy za to pokazać wybrane obszary sztucznej inteligencji, które można dzisiaj wykorzystywać w naszym otoczeniu, praktycznie stosując głębokie uczenie.

Szersza dyskusja o sztucznej inteligencji

Jeżeli chcesz się dowiedzieć więcej o sztucznej inteligencji, zajrzyj do dodatku A.

Postawienie pytań

Aby jak najlepiej poznać uczenie maszynowe, trzeba zacząć od postawienia odpowiednich pytań. Oto co musimy zdefiniować:

- jakie są dane wejściowe, z których chcemy wyekstrahować informacje (model);
- jaki model jest najbardziej odpowiedni dla tych danych;
- jakie informacje o nowych danych chcemy uzyskać, stosując ten model.

Jeżeli odpowiemy sobie na trzy powyższe pytania, będziemy mogli stworzyć proces uczenia maszynowego, który umożliwi nam zbudowanie modelu i uzyskanie żądanych informacji. Aby lepiej się przygotować, zapoznajmy się z kilkoma najważniejszymi pojęciami, które musi znać każdy praktyk uczenia maszynowego. Wrócimy do nich później, aby sprawdzić, jak mogą one nam pomóc w lepszym zrozumieniu istoty sieci neuronowych i głębokiego uczenia.

Matematyczne podstawy uczenia maszynowego — algebra liniowa

Algebra liniowa to opoka, na której opierają się uczenie maszynowe i głębokie uczenie. Teoria ta daje nam narzędzia matematyczne potrzebne do rozwiązywania równań wykorzystywanych przy tworzeniu modeli danych.

Dobrym podręcznikiem do algebry liniowej jest *Algebra liniowa* Jacka Kłopotowskiego (Oficyna Wydawnicza SGH, Warszawa 2012).

Zanim zajmiemy się tym tematem, przedstawmy kilka podstawowych pojęć. Pierwszym z nich jest *skalar*.

Skalar

W matematyce skalar jest jednym z elementów, z których składa się wektor. Skalar jest liczbą rzeczywistą i elementem definiującym przestrzeń wektorową.

W informatyce skalar jest synonimem **zmiennej** i oznacza miejsce w pamięci z przypisaną mu nazwą. W tym miejscu przechowywana jest pewna informacja zwana **wartością**.

Wektor

Na nasz użytek zdefiniujemy wektor w następujący sposób:

> *Dla danej dodatniej liczby naturalnej* n *wektor jest uporządkowaną* n*-ką, czyli zbiorem (tablicą)* n *liczb zwanych elementami lub skalarami.*

Strukturę zwaną wektorem tworzy się w procesie określanym jako *wektoryzacja*. Liczba elementów wektora jest nazywana nieformalnie jego „rzędem” (lub „długością”). Wektory mogą również reprezentować punkty w przestrzeni *n*-wymiarowej. W przestrzeni euklidesowej długość wektora to odległość od początku układu współrzędnych do punktu reprezentowanego przez koniec wektora.

W tekstach matematycznych wektor często zapisuje się w następujący sposób:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{bmatrix}$$

Lub w taki:

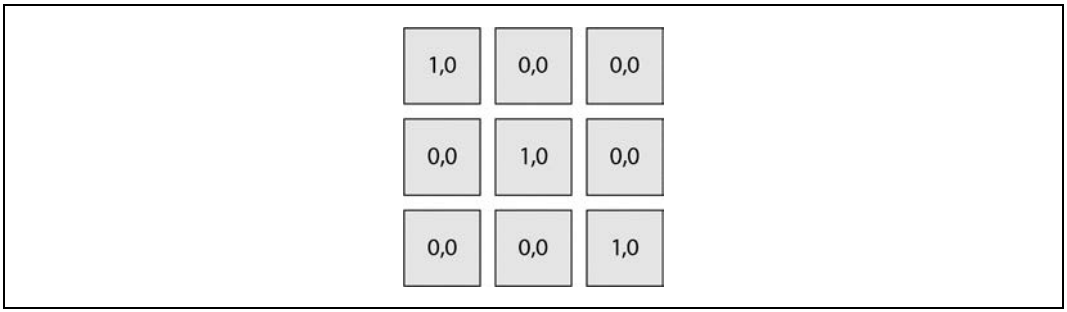
$$x = [x_1, x_2, x_3, \dots, x_n]$$

Istnieje wiele różnych metod wektoryzowania danych, które można wstępnie przetwarzać w wielu krokach, co skutkuje uzyskaniem modeli o różnych stopniach skuteczności. Szerzej temat przekształcania surowych danych w wektory opisujemy w dalszej części rozdziału, a dokładnie omówimy go w rozdziale 5.

Macierz

Macierz to grupa wektorów o równych liczbach elementów. Macierz jest zatem dwuwymiarową tablicą złożoną z wierszy i kolumn.

Na macierz złożoną z *n* wierszy i *m* kolumn mówi się, że ma wymiary *n*×*m*. Rysunek 1.3 przedstawia macierz o wymiarach 3×3. Macierze są podstawowymi strukturami wykorzystywanymi w algebrze liniowej i w uczeniu maszynowym, o czym przekonasz się, czytając ten rozdział.



Rysunek 1.3. Macierz o wymiarach 3×3

Tensor

Tensor to najbardziej podstawowe określenie wielowymiarowej tablicy. Jest to bardziej struktura matematyczna niż wektor. Wektor można traktować jako przypadek tensora.

W tensorze elementy wierszy ułożone są wzdłuż osi *y*, a elementy kolumn wzdłuż osi *x*. Każda z osi jest wymiarem, a tensor ma dodatkowe wymiary, jak również rząd. Dla porównania: skalar ma rząd równy 0, wektor rząd równy 1, a macierz rząd równy 2. Obiekt o rzędzie równym 3 lub wyższym określa się mianem tensora.

Hiperpłaszczyzna

Innym stosowanym w algebrze liniowej pojęciem, które musisz znać, jest **hiperpłaszczyzna**. W geometrii jest to podprzestrzeń o wymiarze o jeden mniejszym niż otaczająca ją przestrzeń. W przestrzeni trójwymiarowej hiperpłaszczyzna ma dwa wymiary. W przestrzeni dwuwymiarowej hiperpłaszczyzna ma jeden wymiar, czyli jest prostą.

Hiperpłaszczyzna jest pojęciem matematycznym dzielącym *n*-wymiarową przestrzeń na osobne „części", dzięki czemu wykorzystuje się ją do klasyfikowania danych. Optymalizacja parametrów hiperpłaszczyzny jest jednym z podstawowych zagadnień w modelowaniu liniowym, o czym dowiesz się z dalszej części rozdziału.

Stosowane działania matematyczne

W tej części opisujemy podstawowe działania matematyczne z zakresu algebry liniowej, które powinieneś znać.

Iloczyn skalarny

Najważniejszą operacją w algebrze liniowej, często wykorzystywaną w uczeniu maszynowym, jest **iloczyn skalarny**, zwany również niekiedy **iloczynem wewnętrznym**. Wykonuje się go na dwóch wektorach o równych liczbach elementów, a jego wynikiem jest pojedyncza liczba. Operacja polega na przemnożeniu przez siebie odpowiednich elementów wektorów i sumowaniu iloczynów. Nie wcho-

dząc zbyt głęboko w zawiłości matematyczne, warto na razie pamiętać, że uzyskany pojedynczy wynik zawiera mnóstwo informacji.

Przede wszystkim iloczyn skalarny mówi nam, jak duże są poszczególne elementy obu wektorów. Iloczyn dwóch wektorów o dużych elementach daje duży wynik, a wektorów o małych elementach — mały wynik. Jeżeli odpowiednie elementy wektorów podda się operacji matematycznej zwanej **normalizacją**, wówczas iloczyn skalarny jest miarą podobieństwa tych wektorów. Iloczyn skalarny dwóch znormalizowanych wektorów nosi nazwę **podobieństwa cosinusowego**.

Iloczyn Hadamarda

Inną operacją z dziedziny stosowanej algebry liniowej jest **iloczyn Hadamarda** (lub **iloczyn po współrzędnych**). Operacja ta obejmuje dwa wektory o równych liczbach elementów, a jej wynikiem jest wektor o tej samej liczbie elementów, przy czym poszczególne elementy są iloczynami odpowiednich elementów wektorów wejściowych.

Iloczyn tensorowy

Jest to operacja wykonywana na dwóch wektorach (zwana również **iloczynem diadycznym**). Każdy element pierwszego wektora jest mnożony przez każdy element odpowiedniego wiersza drugiego wektora. Wynikiem jest macierz złożona z wektorów.

Przekształcanie danych w wektory

W uczeniu maszynowym i w nauce o danych analizowane są dane wszelkich typów (tekst, czasowe szeregi liczb, nagrania dźwiękowe, obrazy, filmy). Kluczową kwestią jest uzyskanie wektorowej reprezentacji tychże danych.

Dlaczego stosując algorytmy uczenia, nie można przetwarzać surowych danych? Rzecz w tym, że uczenie maszynowe polega na stosowaniu zasad algebry liniowej i rozwiazywania układów równań. Danymi wejściowymi dla tych równań są liczby zmiennoprzecinkowe, dlatego surowe dane wejściowe trzeba w jakiś sposób przekształcić w te liczby. Tym zagadnieniem zajmiemy się w następnej części rozdziału poświęconej rozwiązywaniu układów równań. Przykładem surowych danych może być zbiór liczb opisujący różne odmiany irysów (*http://archive.ics.uci.edu/ml/datasets/Iris*):

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

Innym przykładem może być zwykły dokument tekstowy:

```
Szukaj, Cywil, szukaj!
Szukajcie, a znajdziecie.
```

```
Szukaj wiatru w polu.
```

Powyższe surowe dane są różnych typów i trzeba je zwektoryzować, aby przybrały formę nadającą się do wykorzystania w uczeniu maszynowym. Ostatecznie dane wejściowe muszą przybrać postać macierzy, ale po drodze można je przekształcić na pośrednie formaty (np. svmlight, jak w poniższym przykładzie). Celem jest przygotowanie dla algorytmu uczącego danych wejściowych w formacie wektorowym svmlight, jak niżej:

```
1.0 1:0.7500000000000001 2:0.41666666666666663 3:0.702127659574468 4:0.5652173913043479
2.0 1:0.6666666666666666 2:0.5 3:0.9148936170212765 4:0.6956521739130436
2.0 1:0.45833333333333326 2:0.3333333333333336 3:0.8085106382978723 4:0.7391304347826088
0.0 1:0.1666666666666665 2:1.0 3:0.021276595744680823
2.0 1:1.0 2:0.5833333333333334 3:0.9787234042553192 4:0.8260869565217392
1.0 1:0.3333333333333333 3:0.574468085106383 4:0.47826086956521746
1.0 1:0.7083333333333336 2:0.7500000000000002 3:0.6808510638297872 4:0.5652173913043479
1.0 1:0.916666666666667 2:0.6666666666666667 3:0.7659574468085107 4:0.5652173913043479
0.0 1:0.08333333333333343 2:0.5833333333333334 3:0.021276595744680823
2.0 1:0.6666666666666666 2:0.8333333333333333 3:1.0 4:1.0
1.0 1:0.9583333333333335 2:0.7500000000000002 3:0.723404255319149 4:0.5217391304347826
0.0 2:0.7500000000000002
```

Dane w tym formacie można łatwo umieścić w macierzy i w rzadkim wektorze kolumnowym z etykietami (pierwsza liczba w każdym wierszu). Oznaczone indeksami liczby w poszczególnych wierszach, czyli „cechy", umieszcza się w odpowiednich komórkach macierzy i w ten sposób przygotowuje do wykonywania na nich różnych operacji z dziedziny algebry liniowej w procesie uczenia maszynowego. Proces wektoryzacji danych szczegółowo opisujemy w rozdziale 8.

Tu pojawia się typowe pytanie: „Dlaczego dane przeznaczone do uczenia maszynowego muszą (powinny) być reprezentowane w postaci (rzadkiej) macierzy?". Aby na nie odpowiedzieć, musimy poczynić krótką dygresję o podstawach rozwiązywania układów równań.

Rozwiązywanie układów równań

W świecie algebry liniowej trzeba rozwiązywać układy równań liniowych w następującej postaci:

$$Ax = b$$

A oznacza macierz złożoną z wektorów danych wejściowych, natomiast b jest wektorem kolumnowym z etykietami dla wektorów macierzy A. Jeżeli weźmiemy trzy pierwsze wiersze z poprzedniego przykładu, zawierające serializowane, rzadkie dane, i zapiszemy je w formie wymaganej w algebrze liniowej, wtedy uzyskamy następujący wynik:

Kolumna 1	Kolumna 2	Kolumna 3	Kolumna 4
0.7500000000000001	0.41666666666666663	0.702127659574468	0.5652173913043479
0.6666666666666666	0.5	0.9148936170212765	0.6956521739130436
0.45833333333333326	0.3333333333333336	0.8085106382978723	0.7391304347826088

Powyższa macierz jest zmienną A w naszym równaniu. Poszczególne wartości w każdym wierszu są cechami danych wejściowych.

Czym jest cecha?

W uczeniu maszynowym cechą jest każda niezależna wartość w każdej kolumnie macierzy *A*. Cechy można odczytywać bezpośrednio z surowych danych wejściowych, jednak w większości przypadków wymagane jest przekształcanie surowych danych wejściowych na postać bardziej nadającą się do modelowania.

Załóżmy, że mamy kolumnę danych wejściowych zawierającą cztery różne etykiety tekstowe. Musimy przeanalizować wszystkie dane wejściowe i przypisać indeksy wykorzystywanym etykietom. Następnie uzyskane wartości (0, 1, 2, 3) musimy znormalizować do liczb z zakresu od 0,0 do 1,0 na podstawie indeksu etykiety w każdym wierszu i kolumnie macierzy. Tego typu przekształcenie znacznie ułatwia znalezienie w procesie uczenia maszynowego lepszego rozwiązania modelowanego problemu. Więcej informacji na temat wektoryzacji danych znajdziesz w rozdziale 5.

Dla każdej wartości w każdym wierszu trzeba znaleźć współczynnik dla funkcji prognostycznej, która da w wyniku wektor *b*, tj. etykietę dla każdego wiersza. Etykiety w serializowanych, rzadkich wektorach z poprzedniego przykładu wyglądają następująco:

```
Etykiety
1.0
2.0
2.0
```

Wspomniane wyżej współczynniki tworzą wektor kolumnowy *x* (zwany **wektorem parametrów**), pokazany na rysunku 1.4.

	Rekordy treningowe (A)					Wektor parametrów (x)		Wynik (b)
Wektor wejściowy 1	0,7500	0,4166	0,7021	0,5652		?		1,0
Wektor wejściowy 2	0,6666	0,5	0,9148	0,6956	•	?	=	2,0
Wektor wejściowy 3	0,4583	0,3333	0,8085	0,7391		?		3,0

Rysunek 1.4. Wizualizacja równania Ax = b

Układ równań jest określany jako spójny, jeżeli istnieje taki wektor parametrów *x*, dla którego rozwiązanie naszego równania można zapisać w następującej postaci:

$$x = A^{-1}b$$

Ważny jest związek pomiędzy wyrażeniem $x = A^{-1}b$ a metodą rozwiązywania równania. Powyższe wyrażenie reprezentuje jedynie rozwiązanie równania. Zmienna A^{-1} jest macierzą odwróconą, którą uzyskuje się w procesie **odwracania macierzy**. Zważywszy, że nie każdą macierz da się odwrócić, potrzebna jest metoda umożliwiająca rozwiązanie równania bez odwracania macierzy. Jedną z nich jest **rozkład macierzy**. Przykładową metodą rozkładania macierzy *A* w celu rozwiązania układu równań liniowych jest **metoda LU** (ang. *lower-upper* — macierze górna i dolna). Przejrzyjmy jeszcze inne podstawowe metody rozwiązywania układów równań liniowych.

Metody rozwiązywania układów równań liniowych

Są dwie podstawowe grupy metod rozwiązywania układów równań liniowych. Pierwsza to tzw. **metody bezpośrednie**, wymagające wykonania znanej liczby obliczeń. Drugą grupę stanowią **metody iteracyjne**, wymagające wykonania serii obliczeń przybliżanych do wyniku i określenia warunków, po spełnieniu których zwracany jest wektor *x*. Metody bezpośrednie są szczególnie skuteczne, jeżeli wszystkie dane treningowe (*A* i *b*) można zapisać w pamięci pojedynczego komputera. Przykładami popularnych metod bezpośrednich są metoda eliminacji Gaussa oraz metoda równań normalnych.

Metody iteracyjne

Metody iteracyjne są szczególnie skuteczne, gdy danych nie można pomieścić w pamięci pojedynczego komputera. Odczytując w pętli kolejne rekordy zapisane na dysku, można przetwarzać znacznie większe ilości danych. Typowym, najbardziej znanym przykładem metody iteracyjnej, stosowanej w nowoczesnym uczeniu maszynowym, jest metoda **stochastycznego gradientu prostego**, opisana w dalszej części rozdziału. Inne popularne metody to metoda **gradientu sprzężonego** i metoda **naprzemiennych najmniejszych kwadratów** (opisana dokładniej w rozdziale 3.). Metody iteracyjne okazały się również skalowalne, ponieważ pętle przetwarzania danych mogą obejmować nie tylko rekordy zapisane lokalnie, ale również rozproszone w klastrze komputerów. Wektory parametrów wyliczane przez poszczególne agenty są regularnie uśredniane i aktualizowane (szczegółowo tę metodę opisujemy w rozdziale 9.).

Metody iteracyjne w algebrze liniowej

Patrząc z matematycznego punktu widzenia, dane wejściowe trzeba przetworzyć za pomocą określonych algorytmów. Trzeba je przekształcić z surowej postaci na macierz *A*. Po krótkim zapoznaniu z podstawami algebry liniowej znamy już odpowiedź na pytanie, *dlaczego* należy zadawać sobie trud wektoryzowania danych wejściowych. W tej książce prezentujemy przykładowe kody, które pokazują, *jak* to należy robić. Wybór metody wektoryzacji danych ma wpływ na uzyskiwane wyniki uczenia maszynowego. W dalszej części książki dowiesz się, jak wstępnie przetwarzać dane przed wektoryzacją, aby uzyskiwać dokładniejsze modele.

Matematyczne podstawy uczenia maszynowego — statystyka

Abyśmy mogli iść dalej, zajmiemy się teraz statystyką w niezbędnym zakresie. Wyjaśnimy podstawowe pojęcia, takie jak:

- prawdopodobieństwo,
- rozkłady prawdopodobieństwa,
- szansa zdarzenia.

Oprócz tego istnieją podstawowe pojęcia z dziedziny statystyki opisowej i statystyki indukcyjnej, które chcemy wyjaśnić. Statystyka opisowa obejmuje:

- tworzenie histogramów,
- tworzenie wykresów pudełkowych,
- tworzenie wykresów punktowych,
- wyliczanie wartości średnich,
- wyliczanie odchyleń standardowych,
- wyliczanie współczynników korelacji.

Natomiast statystyka indukcyjna obejmuje techniki uogólniania na populację danych wyników uzyskanych na podstawie próbek. Poniżej wymienione są niektóre operacje z zakresu statystyki indukcyjnej:

- wyliczanie *p*-wartości,
- wyliczanie przedziałów ufności.

Pomiędzy statystyką opisową a indukcyjną istnieją następujące zależności:

- Statystyka opisowa obejmuje opisywanie prób na podstawie populacji (rozumowanie dedukcyjne).
- Statystyka indukcyjna obejmuje opisywanie populacji na podstawie prób.

Zanim dowiesz się, co próba danych może powiedzieć o całej populacji, musimy opisać ryzyko związane z pobieraniem próby z danej populacji.

Nie chcemy wdawać się tu w omawianie szerokiego tematu z dziedziny statystyki ogólnej, szczegółowo opisanego w wielu książkach. Ta część nie stanowi wyczerpującego wykładu o statystyce. Jej celem jest raczej wskazanie pewnych istotnych zagadnień, które możesz dokładniej przestudiować, korzystając z innych źródeł. Mając to na uwadze, zacznijmy od zdefiniowania pojęcia prawdopodobieństwa.

Prawdopodobieństwo

Prawdopodobieństwo zdarzenia E określa się liczbą z zakresu od 0 do 1. Wartość 0 oznacza, że zdarzenie E na pewno nie wystąpi, natomiast wartość 1, że na pewno wystąpi. Często prawdopodobieństwo wyraża się liczbą zmiennoprzecinkową, ale może to być również wartość procentowa od 0 do 100%. Wartość prawdopodobieństwa nie może być mniejsza niż 0% ani większa niż 100%. Przykładowo prawdopodobieństwo 0,35 można wyrazić jako 35%.

Klasycznym przykładem pomiaru prawdopodobieństwa jest obserwowanie, ile razy podczas rzucania monetą wypadnie orzeł lub reszka (dla obu stron prawdopodobieństwo jest równe 0,5). Prawdopodobieństwo dla całej przestrzeni zdarzeń elementarnych jest równe 1, ponieważ przestrzeń reprezentuje wszystkie możliwe wyniki danej próby. Na przykładzie dwóch wyników rzutu monetą (wypadnięcia orła i komplementarnego wypadnięcia reszki) widać, że prawdopodobieństwo 0,5 + 0,5 jest równe 1, ponieważ prawdopodobieństwo dla całej przestrzeni zdarzeń musi być równe tej liczbie. Prawdopodobieństwo zdarzenia opisuje się w poniższy sposób:

$$P(E) = 0,5$$

Powyższą formułę czyta się następująco:

Prawdopodobieństwo zdarzenia E jest równe 0,5.

Prawdopodobieństwo i szansa

Często początkujący adepci statystyki mylą pojęcia prawdopodobieństwa i szansy. Zanim pójdziemy dalej, wyjaśnijmy te dwa terminy.

Prawdopodobieństwo zdarzenia *E* definiuje się następująco:

P(E) = (liczba przypadków wystąpienia zdarzenia E) / (całkowita liczba przypadków)

Na podstawie tego wzoru można wyliczyć prawdopodobieństwo wyciągnięcia asa (4 karty) z talii (52 karty):

4/52 = 0,077

Natomiast **szansę** definiuje się jako:

(liczba przypadków wystąpienia zdarzenia E) / (liczba przypadków wystąpienia zdarzenia innego niż E)

Szansa wyciągnięcia asa jest więc równa:

4 / (52 – 4) = 1/12 = 0,0833333…

Podstawowa różnica pomiędzy obydwoma wzorami leży w mianowniku ułamka (całkowita liczba przypadków kontra liczba przypadków wystąpienia innego zdarzenia). Dlatego prawdopodobieństwo i szansa to dwa różne pojęcia statystyczne.

Prawdopodobieństwo jest podstawą funkcjonowania sieci neuronowych i głębokiego uczenia, ponieważ odgrywa kluczową rolę w ekstrahowaniu i klasyfikowaniu cech, czyli w dwóch najważniejszych operacjach. Szerszy wykład ze statystyki można znaleźć w książce *Statystyka dla bystrzaków. Wydanie II* Deborah J. Rumsey (Septem, Gliwice 2016).

Dodatkowa definicja: prawdopodobieństwo subiektywne i obiektywne

W statystyce stosowane są dwa podejścia: subiektywne i obiektywne. Podstawowa różnica pomiędzy nimi polega na definicji prawdopodobieństwa.

W podejściu obiektywnym prawdopodobieństwo ma sens jedynie w przypadku powtarzalnych pomiarów. Podczas dokonywania pomiarów obserwuje się lekkie odchylenia uzyskiwanych wyników wynikające z właściwości przyrządów pomiarowych. Jeżeli jakąś wielkość będziemy mierzyć bardzo wiele razy, częstość uzyskania określonej wartości jest miarą jej prawdopodobieństwa.

W podejściu subiektywnym prawdopodobieństwo oznacza pewność określonego stwierdzenia. Prawdopodobieństwo odzwierciedla naszą wiedzę o uzyskiwanych wynikach pomiarów. Nasza subiektywna wiedza o zdarzeniu jest ściśle związana z prawdopodobieństwem.

Prawdopodobieństwo obiektywne opiera się na wynikach wielu, a nawet bardzo wielu losowych eksperymentów, po których można stwierdzić szacowany wynik. Natomiast prawdopodobieństwo subiektywne opiera się na „przekonaniu" o wartości zmiennej (w terminologii matematycznej na jej „rozkładzie") i modyfikowaniu przekonania na podstawie nowych informacji o tej zmiennej.

Prawdopodobieństwo warunkowe

Gdy musimy określić prawdopodobieństwo danego zdarzenia uzależnionego od innego zdarzenia, mamy do czynienia z **prawdopodobieństwem warunkowym**, które w fachowej literaturze opisuje się wzorem:

$P(E \mid F)$,

gdzie:

E oznacza zdarzenie, którego prawdopodobieństwo chcemy znać;

F jest zdarzeniem, które już wystąpiło.

Na przykład prawdopodobieństwo śmierci hospitalizowanej osoby ze zdrowym sercem jest mniejsze niż osoby chorej. Można to wyrazić w następujący sposób:

$P(\text{śmierć} \mid \text{chore serce}) > P(\text{śmierć} \mid \text{zdrowe serce})$

Niekiedy na zdarzenie *F* mówi się „warunek". Prawdopodobieństwo warunkowe w uczeniu maszynowym i głębokim uczeniu jest interesujące, ponieważ często chcemy wiedzieć, kiedy ma miejsce kilka zdarzeń i jaki wpływ na siebie one wywierają. Chcemy znać prawdopodobieństwo warunkowe:

$P(E \mid F)$,

gdzie *E* oznacza etykietę, a *F* liczbę atrybutów jednostki, dla której prognozujemy etykietę *E*. Przykładem może być prognozowanie śmiertelności (zdarzenie *E*) na podstawie wyników obserwacji pacjentów w szpitalu (*F*).

Twierdzenie Bayesa

Jednym z bardziej znanych przykładów zastosowania teorii prawdopodobieństwa warunkowego jest twierdzenie (lub formuła) Bayesa. W medycynie jest ono stosowane do określania prawdopodobieństwa, że pacjent z pozytywnym wynikiem testu chorobowego faktycznie jest chory.

Formuła Bayesa dla dwóch dowolnych zdarzeń *A* i *B* ma następującą postać:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Prawdopodobieństwo a posteriori

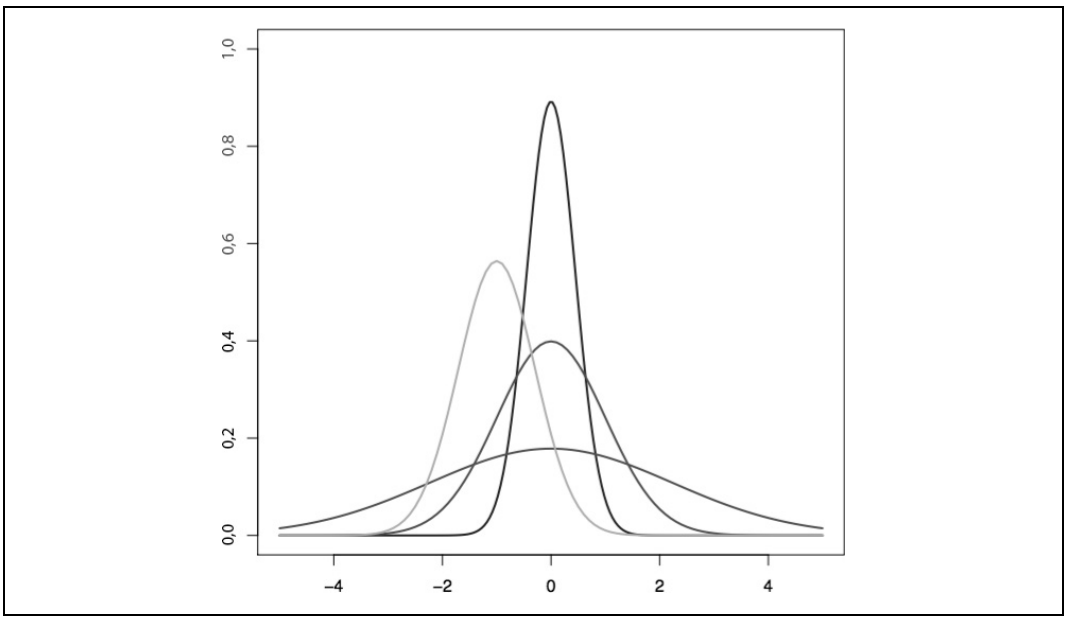
W statystyce subiektywnej prawdopodobieństwo *a posteriori* losowego zdarzenia jest warunkowym prawdopodobieństwem określonym po uzyskaniu dowodu. Rozkład prawdopodobieństwa *a posteriori* definiuje się jako rozkład prawdopodobieństwa nieznanej wielkości, określonego na podstawie wyników doświadczenia uznanego jako losowo zmienne. Tym zagadnieniem zajmiemy się jeszcze przy okazji omawiania funkcji aktywacji (w rozdziale 2.), w których surowe dane wejściowe są przekształcane na wartości prawdopodobieństwa *a posteriori*.

Rozkłady prawdopodobieństwa

Rozkład prawdopodobieństwa opisuje stochastyczną strukturę losowych wartości. W statystyce przyjmujemy założenia dotyczące rozkładu danych po to, aby móc wyciągać na ich temat wnioski. Potrzebna jest do tego celu formuła wyliczająca, jak często w wynikach pojawiają się poszczególne wartości. Powszechnie stosowany jest **rozkład normalny** (zwany również **rozkładem Gaussa** lub **krzywą dzwonową**). Jeżeli dane pasują do określonego teoretycznego rozkładu prawdopodobieństwa, można wtedy przyjmować założenia potrzebne do ich dalszego przetwarzania.

Rozkłady prawdopodobieństwa dzielą się na **ciągłe** i **dyskretne**. Rozkład dyskretny dotyczy skończonego zbioru wartości, natomiast rozkład ciągły — dowolnych wartości z określonego przedziału. Przykładem rozkładu ciągłego jest rozkład normalny, a dyskretnego — rozkład dwumianowy.

Rozkład normalny (patrz rysunek 1.5) jest zwany również rozkładem Gaussa, od nazwiska Karla Gaussa, matematyka i fizyka żyjącego w XVIII wieku. Rozkład ten definiuje się za pomocą wartości średniej i odchylenia standardowego, a wykresy jego odmian mają ogólnie taki sam kształt.



Rysunek 1.5. Przykłady rozkładu normalnego

Inne rozkłady stosowane w uczeniu maszynowym to:

- rozkład dwumianowy,
- odwrócony rozkład normalny,
- rozkład logarytmicznie normalny.

Rozkładanie danych treningowych jest w uczeniu maszynowym bardzo ważne, ponieważ na tej podstawie można decydować, jak należy wektoryzować modelowane dane.

Centralne twierdzenie graniczne

Jeżeli próba jest odpowiednio liczna, wtedy jej uśredniony rozkład jest zbliżony do rozkładu normalnego, niezależnie od rozkładu populacji, z której próba została pobrana.

Zgodnie z tym twierdzeniem można wyciągać wnioski statystyczne, wykonując testy oparte na przybliżonej normalności średniej, niezależnie od tego, czy populacja, z której próba została pobrana, ma rozkład normalny.

W informatyce twierdzenie to jest wykorzystywane w algorytmach pobierających próby danych z populacji o rozkładzie innym niż normalny. Efekt można stwierdzić, wykreślając histogram średniej próby z rozkładu normalnego.

Rozkłady z tzw. długimi ogonami (np. prawo Zipfa, prawo mocy, rozkład Pareto) są przykładami sytuacji, w których oprócz często występujących wartości znajdują się wartości o częstościach asymptotycznie malejących. Rozkłady te odkrył Benoît Mandelbrot w latach 50. ubiegłego wieku, a spopularyzował Chris Anderson w książce *Długi ogon. Ekonomia przyszłości — każdy konsument ma głos* (Media Rodzina, 2008).

Przykładem takiego rozkładu jest ranking sprzedawanych rzeczy, wśród których jest kilka wyjątkowo popularnych oraz bardzo dużo unikatowych, sprzedawanych w małych ilościach. Taki rozkład ranking – częstość wyników (głównie popularności rzeczy, czyli liczby sprzedanych egzemplarzy) często stanowi prawo mocy. Z tej perspektywy można traktować go jako rozkład z długim ogonem.

Rozkłady z długimi ogonami spotyka się w następujących sytuacjach:

Zniszczenia spowodowane trzęsieniem ziemi

Największe zniszczenia są w epicentrum; im większa odległość od epicentrum, tym zniszczenia są mniejsze.

Wydajność zbiorów

Wydajność zbiorów w modelu danych oscyluje w granicach średniej, ale czasami zdarzają się zbiory odstające od historycznych danych.

Prognozowanie śmierci pacjenta na OIOM-ie

Zdarzają się przypadki znacznie odstające od sytuacji, jakie mają miejsce podczas pobytu pacjenta na OIOM-ie.

Powyższe przykłady dotyczą treści tej książki w kontekście klasyfikowania problemów, ponieważ większość modeli statystycznych opiera się na wnioskach wyciągniętych na podstawie analizy mnóstwa danych. Jeżeli najbardziej interesujące zdarzenia mają miejsce w dalekiej części ogona, a model danych zbudowany na podstawie próby danych nie odzwierciedla takich przypadków, to znaczy, że model ten może funkcjonować w nieprzewidywalny sposób. Ten efekt może ujawnić się jeszcze wyraźniej w modelach nieliniowych, na przykład stosowanych w sieciach neuronowych. Taką sytuację będziemy traktować jako przypadek szczególny problemu „wewnątrz próby/na zewnątrz próby". Nawet doświadczony praktyk uczenia maszynowego może się zdziwić, że model doskonale sprawdzający się w przypadku próby danych treningowych zawodzi w przypadku użycia go w większej populacji danych.

Rozkłady z długimi ogonami dotyczą prawdopodobieństwa wystąpienia zdarzeń z obszaru pięciokrotnie większego niż odchylenie standardowe. Należy bardzo starannie definiować reprezentację zdarzeń w danych treningowych, aby zapobiegać ich nadmiernemu dopasowaniu. Dokładniej tym zagadnieniem zajmiemy się w dalszej części rozdziału opisującej nadmierne dopasowanie oraz w rozdziale 4. poświęconym regulowaniu modelu.

Próba i populacja

Populację danych definiuje się jako sumę wszystkich jednostek, które chcemy przeanalizować lub zamodelować podczas eksperymentu. Przykładem populacji są „wszyscy programiści Javy w województwie śląskim".

Próba to część populacji danych, która z założenia ma dokładnie reprezentować rozkład danych w populacji, bez obciążenia (tzn. bez przekłamań spowodowanych sposobem próbkowania populacji).

Metody wielokrotnego losowania

Bootstrap i **kroswalidacja** to dwie popularne metody wielokrotnego losowania danych, stosowane przez praktyków uczenia maszynowego. Metoda bootstrap polega na losowaniu próby z innej próby danych w celu utworzenia nowej próby o zrównoważonej liczbie wartości w poszczególnych klasach. Jest to metoda stosowana w przypadku, gdy trzeba modelować zbiór danych o silnie niezrównoważonych klasach.

Kroswalidacja (zwana również **sprawdzaniem krzyżowym**) to metoda oceniania wierności dopasowania modelu do zbioru treningowego. Polega ona na podziale zbioru na *N* części, z których pewna ich liczba jest wykorzystywana do uczenia, a pozostała do testowania modelu. Części danych przenosi się pomiędzy grupami dotąd, aż wyczerpią się wszystkie możliwe kombinacje. Nie istnieje stała liczba grup, jednak badania wykazały, że w praktyce dobrze sprawdza się 10 grup. Często również wyodrębnia się osobną grupę danych wykorzystywaną do weryfikowania poprawności modelu podczas treningu.

Błąd doboru próby

Błąd doboru próby pojawia się wtedy, gdy metoda próbkowania nie zapewnia wymaganej losowości danych, przez co próba jest nietrafna i nie reprezentuje wiernie modelowanej populacji. Należy o tym pamiętać podczas próbkowania danych, aby błąd doboru próby dla dużej populacji nie doprowadził do pogorszenia wierności uzyskanego modelu.

Szansa

Często na określenie prawdopodobieństwa wystąpienia jakiegoś zdarzenia, którego nie można wyrazić liczbą, używa się nieformalnego terminu **szansa**. Zazwyczaj oznacza on, że istnieje racjonalnie uzasadnione prawdopodobieństwo wystąpienia tego zdarzenia, które jednak nie ma miejsca z powodu nieznanych tymczasowo czynników. Nieformalnie określenie *szansa* jest synonimem **prawdopodobieństwa**.

Jak uczą się maszyny?

W poprzedniej części rozdziału poświęconej rozwiązywaniu układów równań liniowych opisaliśmy podstawy rozwiązywania równania typu $Ax = b$. Uczenie maszynowe opiera się na algorytmach minimalizujących błąd rozwiązania powyższego równania poprzez **optymalizację**.

Optymalizacja polega na wielokrotnym zmienianiu liczb w wektorze kolumnowym x (wektorze parametrów) dotąd, aż uzyska się taki zbiór liczb, który pozwoli uzyskać wyniki najbardziej zbliżone do rzeczywistych wartości. Każda waga w macierzy wag jest modyfikowana po wyliczeniu błędu za pomocą funkcji straty (na podstawie rzeczywistego wyniku w postaci wektora b). Macierz błędów zawierająca pewną część strat każdej wagi jest mnożona przez właściwe wagi.

W dalszej części rozdziału opisujemy stochastyczny gradient prosty, będący jedną z metod optymalizacji stosowaną w uczeniu maszynowym. W innych częściach książki zestawimy to pojęcie z innymi algorytmami optymalizacyjnymi. Opiszemy również podstawowe hiperparametry, m.in. regularyzację i współczynnik uczenia.

Regresja

Regresja oznacza funkcję wykorzystywaną do przewidywania rzeczywistych wyników. Tego typu funkcja szacuje wartość zmiennej zależnej na podstawie znanej zmiennej niezależnej. Najczęściej stosowaną funkcją jest **regresja liniowa**, wykorzystująca opisane wcześniej układy równań liniowych. Regresja liniowa to funkcja opisująca zależność zmiennych x i y, na podstawie której dla znanej wartości x można stosunkowo dokładnie przewidzieć wartość y.

Tworzenie modelu

W modelu regresji liniowej stosuje się liniową kombinację współczynników (z wektora parametrów x) i zmiennych wejściowych (wektora wejściowego). Model ten można opisać za pomocą następującego równania:

$$y = a + Bx,$$

gdzie a oznacza punkt przecięcia z osią Y, B cechy wejściowe, a x wektor parametrów.

Powyższe równanie można rozwinąć do następującej postaci:

$$y = a + b_0 * x_0 + b_1 * x_1 + \ldots + b_n * x_n$$

Przykładem prostego problemu, który można rozwiązać, wykorzystując regresję liniową, jest prognozowanie miesięcznego zużycia paliwa na podstawie długości drogi do pracy. W tym przypadku opłata za tankowanie jest funkcją pokonywanej odległości. Koszt paliwa jest zmienną zależną, a długość drogi zmienną niezależną. Należy zanotować obie wartości, a następnie zdefiniować następującą funkcję:

$$koszt = f(odległość)$$

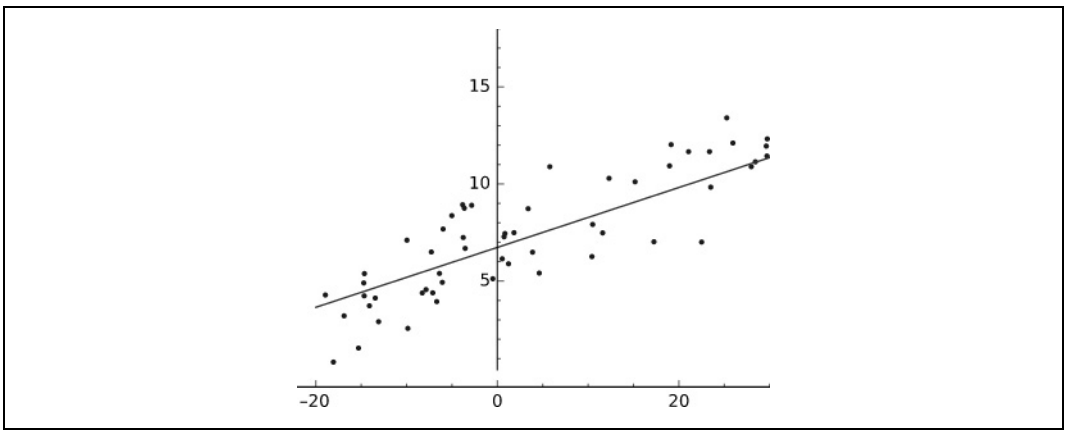
Na tej podstawie można przewidywać wiarygodne wydatki na paliwo w zależności od przejechanych kilometrów. W tym przykładzie modelem jest funkcja f, zmienną niezależną jest *odległość*, a zmienną zależną *koszt*.

Poniżej wymienionych jest kilka innych przykładów modeli regresji liniowej:

- prognozowanie wagi człowieka jako funkcji wzrostu,
- prognozowanie ceny nieruchomości na podstawie jej powierzchni.

Wizualizacja regresji liniowej

Wizualizacja regresji liniowej polega na znalezieniu prostej jak najlepiej dopasowanej do punktów reprezentujących dane, jak na rysunku 1.6.



Rysunek 1.6. Wizualizacja regresji liniowej

Dopasowanie polega na zdefiniowaniu funkcji *f(x)* dającej wartości *y* zbliżone do zmierzonych, rzeczywistych wartości. Prosta wyznaczona równaniem $y = f(x)$ przebiega w pobliżu punktów, których współrzędnymi są pary zmiennych zależnej i niezależnej.

Odniesienie do modelu regresji liniowej

Powyższą funkcję można odnieść do opisanego wcześniej równania $Ax = b$, gdzie A oznacza cechę (np. wagę lub powierzchnię) wszystkich danych wejściowych, które chcemy modelować. Każdy rekord danych jest wierszem w macierzy A. Wektor kolumnowy b zawiera wyniki dla wszystkich rekordów wejściowych w macierzy A. Wykorzystując funkcję błędu i metodę optymalizacji (np. stochastyczny gradient liniowy), można znaleźć taki zbiór parametrów x, dla którego różnica pomiędzy wartościami prognozowanymi a rzeczywistymi będzie najmniejsza.

Aby znaleźć wektor parametrów x, wykorzystując wspomniany wcześniej stochastyczny gradient liniowy, musimy mieć trzy komponenty:

Hipotezę dotyczącą danych

Iloczyn skalarny wektora parametrów x i cech wejściowych (jak w powyższym równaniu)

Funkcję kosztu

Błąd kwadratowy prognozy (wartości prognozowane — wartości rzeczywiste)

Funkcję aktualizującą

Pochodną funkcji błędu kwadratowego (funkcji kosztu)

W regresji liniowej wykorzystywane są linie proste, natomiast w innych rodzajach regresji wykorzystuje się krzywe, w równaniach których wykładnik parametru *x* jest większy od 1 (stąd czasami uczenie maszynowe określa się mianem „dopasowywania krzywych"). Idealnie dopasowana krzywa przechodzi przez każdy punkt danych. Jak na ironię, idealne dopasowanie jest bardzo złym modelem, ponieważ nie można za jego pomocą przewidzieć żadnych innych danych oprócz wykorzystanych do trenowania (model ten nie uogólnia poprawnie danych).

Klasyfikacja

Klasyfikacja to modelowanie danych polegające na tworzeniu klas wyników na podstawie zbioru cech wejściowych. Regresja daje odpowiedź typu „wartość", natomiast klasyfikacja wskazuje „rodzaj" danych. Zmienna zależna *y* jest kategorią, a nie liczbą.

Najbardziej podstawową formą klasyfikacji jest klasyfikacja binarna, w której stosowane są tylko dwie etykiety (dwie klasy: 0 i 1). Wynikiem klasyfikacji może być również liczba zmiennoprzecinkowa z zakresu od 0,0 do 1,0 odzwierciedlająca brak absolutnej pewności co do klasy danych. W takim przypadku przyjmuje się wartość progową (zazwyczaj 0,5) rozdzielającą dwie klasy. W literaturze często używane są pojęcia pozytywnej i negatywnej klasyfikacji (odpowiednio dla wartości 0,0 i 1,0). Więcej na ten temat piszemy w podrozdziale „Ocenianie modeli".

Przykłady klasyfikacji binarnej:

- ocena, czy pacjent jest chory, czy zdrowy,
- ocena, czy wiadomość jest spamem, czy wartościową informacją,
- ocena, czy transakcja jest legalna, czy nielegalna.

Model klasyfikacyjny może obejmować nie tylko dwie, ale *N* etykiet. Ten temat opiszemy szerzej w rozdziale poświęconym porównaniu sieci neuronowych o wielu wartościach wyjściowych z sieciami o jednej wartości wejściowej (klasyfikacja binarna). Klasyfikację opiszemy dokładniej w tym rozdziale w części dotyczącej regresji logistycznej i analizie pełnej architektury sieci neuronowych.

Rekomendacja

Rekomendacja to proces proponowania użytkownikowi systemu pewnych rzeczy na podstawie wyników obserwacji rzeczy oglądanych przez innych użytkowników. Jednym ze znanych wariantów algorytmu rekomendacyjnego jest tzw. **wspólne filtrowanie** (ang. *collaborative filtering*) spopularyzowane przez serwis Amazon.com.

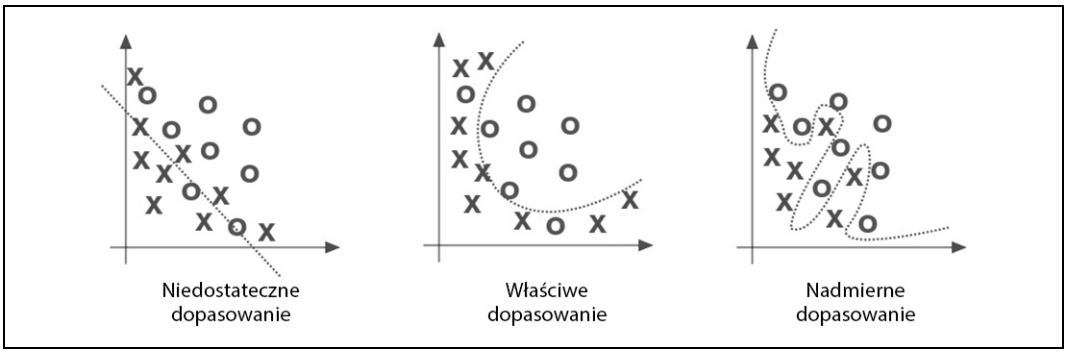
Klastrowanie

Klastrowanie to technika nienadzorowanego uczenia polegająca na iteracyjnym mierzeniu odległości pomiędzy elementami i przesuwaniu podobnych elementów tak, aby znajdowały się coraz bliżej siebie. Na zakończenie tego procesu elementy najciaśniej zgromadzone wokół *n* centroidów

są klasyfikowane jako grupy. Jedną z najbardziej znanych odmian tego procesu w uczeniu maszynowym jest klastrowanie metodą *k*-średnich.

Niedostateczne i nadmierne dopasowanie

Jak wspomnieliśmy wcześniej, algorytmy optymalizacyjne służą przede wszystkim do rozwiązywania problemu niedostatecznego dopasowania modelu, tj. do wytyczenia linii, która lepiej uśrednia dane. Dobrym przykładem niedostatecznego dopasowania jest linia prosta przechodząca przez obszar rozsianych punktów, jak na rysunku 1.7.



Rysunek 1.7. Niedostateczne i nadmierne dopasowanie w uczeniu maszynowym

Jeżeli linia jest zbyt dobrze dopasowana do punktów, pojawia się inny problem zwany nadmiernym dopasowaniem. Rozwiązanie problemu niedostatecznego dopasowania jest zawsze priorytetem w algorytmach uczenia maszynowego, ale dużo wysiłku wymaga znalezienie linii, która nie jest zbyt dobrze dopasowana do punktów. Jeżeli model jest nadmiernie dopasowany, wtedy jego błąd dla danych treningowych jest niewielki, ale taki model nie uogólnia całej populacji danych, która nas interesuje.

Innym sposobem opisania problemu nadmiernego dopasowania jest analiza prawdopodobnego rozproszenia danych. Treningowy zbiór punktów, przez które chcemy przeprowadzić linię, jest tylko próbką większego, nieznanego zbioru. Jeżeli na podstawie wytyczanej linii chcemy prognozować dane, linia ta musi być równie dobrze dopasowana do dużego zbioru. Dlatego musimy zakładać, że próba nieprecyzyjnie reprezentuje większy zbiór danych.

Optymalizacja

Opisany wyżej proces dopasowywania wag i uzyskiwania coraz dokładniejszych prognoz jest nazywany **optymalizacją parametrów**. Proces ten można traktować jako metodę naukową. Formułuje się hipotezę, a następnie wielokrotnie ją weryfikuje, porównując z rzeczywistością, udoskonala lub zmienia, tak aby jak najlepiej opisywała to, co się dzieje na świecie.

Każdy zbiór wag reprezentuje określoną hipotezę dotyczącą znaczenia danych wejściowych, tj. ich relacji z informacjami zawartymi w etykietach. Wagi reprezentują przypuszczenia, które chcemy potwierdzić, dotyczące korelacji pomiędzy danymi wejściowymi a docelowymi etykietami. Wszystkie możliwe wagi i ich kombinacje określa się mianem **przestrzeni hipotezy problemu**. Dążenie do

sformułowania jak najlepszej hipotezy polega na przeszukiwaniu przestrzeni problemu. Do tego celu wykorzystuje się algorytmy optymalizacyjne. Największą sztuką w uczeniu maszynowym jest podejmowanie decyzji, które parametry należy odrzucać, a które akceptować.

Granica decyzyjna i hiperpłaszczyzna

Termin **granica decyzyjna** oznacza *n*-wymiarową hiperpłaszczyznę utworzoną za pomocą wektora parametrów w modelowaniu liniowym.

Dopasowywanie linii do danych poprzez regulowanie kosztów (czyli odległości pomiędzy linią a punktami danych) jest istotą uczenia maszynowego. Linia może być lepiej lub gorzej dopasowana do punktów. Osiąga się to poprzez minimalizację ogólnej odległości pomiędzy linią a punktami. Celem jest zminimalizowanie sumy różnic pomiędzy punktem x na linii a odpowiadającym mu rzeczywistym punktem y. W przestrzeni trójwymiarowej można wyobrazić sobie obszar z dolinami i wzgórzami, przemierzany przez niewidomego wędrowca, który wyczuwa nachylenie terenu. Wędrowiec wykorzystuje algorytm optymalizacyjny, na przykład metodę gradientu prostego, do określenia kierunku prowadzącego w dół doliny. Celem jest znalezienie wag minimalizujących różnice pomiędzy prognozami sieci neuronowej (wektora $\hat{b}$, czyli iloczynu A i x) a prawdziwymi wartościami wynikającymi z testu (b), jak na rysunku 1.4. Wektor parametrów (x) jest miejscem, w którym powinny się znaleźć parametry. Dokładność sieci jest funkcją danych wejściowych i parametrów, a szybkość osiągania żądanej dokładności jest funkcją hiperparametrów sieci.

Hiperparametry

W uczeniu maszynowym stosuje się parametry modelu, jak również parametry regulacyjne, wpływające na jakość i szybkość uczenia się sieci. Te ostatnie zwane są **hiperparametrami** i są wykorzystywane w funkcji sterującej optymalizacją i doborem modelu podczas treningu algorytmu uczenia.

Konwergencja

Pojęcie **konwergencji** jest związane z algorytmem optymalizacyjnym wykorzystywanym do znajdowania wektora parametrów skutkujących uzyskaniem modelu o najmniejszym możliwym błędzie dla wszystkich prób treningowych. Algorytm optymalizacyjny „konwerguje" poprzez iteracyjne weryfikowanie różnych kombinacji parametrów.

Poniżej opisane są trzy najważniejsze atrybuty optymalizacji uczenia maszynowego:

Parametry

Służą do przekształcania danych wejściowych i ułatwiają klasyfikowanie danych generowanych przez sieć.

Funkcja straty

Służy do określania jakości klasyfikacji (wielkości błędu) w każdym kroku.

Funkcja optymalizacyjna

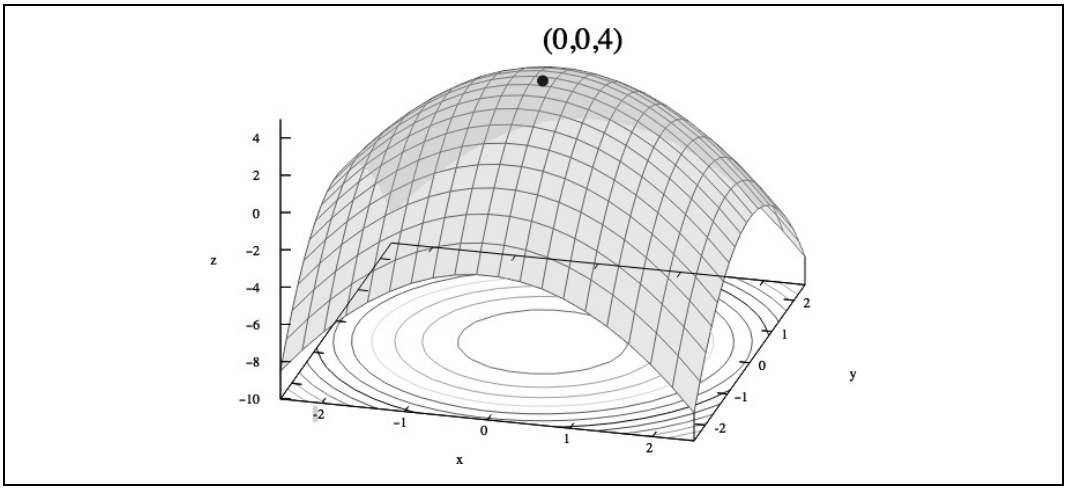
Służy do uzyskiwania modelu o najmniejszym błędzie.

Przyjrzyjmy się teraz bliżej węższej dziedzinie optymalizacji zwanej optymalizacją wypukłą.

Optymalizacja wypukła

W przypadku **optymalizacji wypukłej** algorytmy uczenia maszynowego wykorzystują wypukłe funkcje kosztów. Jeżeli oś *x* reprezentuje pojedynczą wagę, a oś *y* związany z tą wagą koszt, wówczas w pewnym idealnym punkcie osi *x* koszt będzie równy 0 i rósł wykładniczo w miarę oddalania się od tego punktu w każdym z obu kierunków.

Rysunek 1.8 przedstawia odwróconą funkcję kosztu.



Rysunek 1.8. Wizualizacja funkcji wypukłej

Innym sposobem wiązania parametrów z danymi jest szacowanie maksymalnej wiarygodności (ang. *Maximum Likelihood Estimation*, **MLE**). Funkcja MLE jest parabolą o gałęziach skierowanych w dół. Oś pionowa oznacza prawdopodobieństwo, a pozioma wartość parametru. Każdy punkt paraboli opisuje wiarygodność danych dla określonego zbioru parametrów. Algorytm MLE polega na iteracyjnym dobieraniu możliwych parametrów dotąd, aż zostanie znaleziona taka ich kombinacja, dla której dane będą najbardziej wiarygodne.

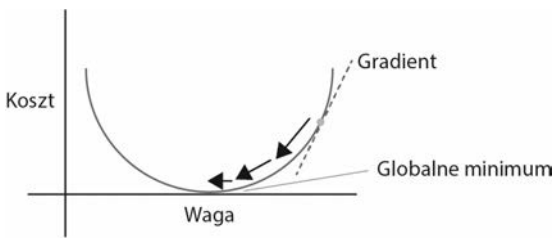
Analogią do minimalnej i maksymalnej wiarygodności danych są dwie strony monety. Jeżeli wyliczymy wartości funkcji kosztu dla dwóch wag i jednego błędu (w przestrzeni trójwymiarowej), uzyskamy powierzchnię przypominającą wydętą ku górze chustę lub odwróconą misę. Na podstawie nachylenia zbocza algorytm optymalizacyjny może określić kierunek, w którym należy modyfikować parametr. Tak działa opisany niżej algorytm gradientu prostego.

Gradient prosty

W metodzie gradientu prostego jakość prognozowanych przez sieć wyników (będących funkcją wag/parametrów) można porównać do obszaru, na którym wzgórza reprezentują miejsca (wartości parametrów, czyli wagi) o dużym błędzie prognozy, natomiast doliny są miejscami o mniejszych błędach. Na tym obszarze wybieramy jeden punkt z początkowymi wagami. Punkt ten wybieramy na podstawie własnej wiedzy (jeżeli na przykład sieć uczy się klasyfikować kwiaty według gatunków, to

wiemy, że wielkość płatków jest ważna, ale ich kolor już nie jest istotny). Jeżeli natomiast chcemy, aby sieć sama wykonała całą pracę, możemy wagi wybierać losowo.

Celem jest jak najszybsze poruszanie się po obszarze w dół, w kierunku miejsc o najmniejszych błędach. W algorytmie optymalizacyjnym, takim jak metoda gradientu prostego, można określić nachylenie zbocza dla każdej wagi i w ten sposób wyznaczyć kierunek w dół. Mierzone jest nachylenie (zmiana błędu w zależności od zmiany wagi) i na tej podstawie wykonywany jest kolejny krok w kierunku dna doliny. Wartość gradientu oblicza się z pochodnej funkcji błędu. Na tej podstawie algorytm optymalizacyjny wyznacza kierunek wykonania kolejnego kroku (patrz rysunek 1.9).



Rysunek 1.9. Zmiana wagi w kierunku minimalnego błędu w metodzie gradientu prostego

Pochodna reprezentuje „szybkość zmian" wartości funkcji. W optymalizacji wypukłej szukamy punktu, w którym pochodna funkcji ma wartość 0. Ten punkt jest również nazywany **punktem stacjonarnym** lub **punktem minimum** funkcji. Optymalizacja polega na minimalizowaniu wartości funkcji (oprócz odwrócenia funkcji kosztu).

Czym jest gradient?

Gradient to wielowymiarowe uogólnienie pochodnej funkcji jednowymiarowej *f*. Gradient jest reprezentowany przez wektor *n* pochodnych cząstkowych funkcji *f*. Jest on wykorzystywany w algorytmach optymalizacyjnych, ponieważ pozwala określić kierunek największych zmian wartości funkcji, tj. największego nachylenia wykresu.

W metodzie gradientu prostego nachylenie funkcji straty jest określane na podstawie jej pochodnej (popularne pojęcie w rachunku różniczkowym). W przypadku dwuwymiarowej funkcji straty wartość jej pochodnej w dowolnym punkcie jest po prostu równa tangensowi kąta nachylenia stycznej, tj. stosunkowi zmiany wartości *y* i zmiany wartości *x* albo stosunkowi wzniesienia do odległości.

Jak wiadomo z trygonometrii, tangens kąta jest stosunkiem długości pionowej do poziomej przyprostokątnej w trójkącie prostokątnym.

Jedną z definicji krzywej jest prosta stale zmieniająca swoje nachylenie. Nachylenie krzywej w każdym punkcie jest określone przez tangens kąta stycznej w tym punkcie. Ponieważ nachylenie wylicza się na podstawie dwóch punktów, jak właściwie określa się nachylenie w jednym punkcie? Pochodną oblicza się na podstawie współrzędnych dwóch punktów położonych blisko siebie. Punkty te następnie przybliża się do siebie coraz bardziej, aż odległość pomiędzy nimi będzie równa zeru. W matematyce uzyskana w ten sposób wartość nosi nazwę **granicy**.

Proces mierzenia strat i zmian wag podczas wykonywania kroku w kierunku punktu o mniejszym błędzie jest powtarzany dotąd, aż nie da się zejść niżej. W tym punkcie dokładność modelu jest największa. Wypukła funkcja straty (typowa w modelowaniu liniowym) ma tylko jedno globalne minimum.

Modelowanie liniowe, którego celem jest znalezienie wektora parametrów *x*, obejmuje trzy komponenty:

- hipotezę dotyczącą danych, na przykład „równanie wykorzystywane do prognozowania wyników w modelu";
- funkcję kosztu, zwaną również funkcją straty, na przykład „suma kwadratów błędów";
- funkcję aktualizującą, będącą pochodną funkcji straty.

Hipoteza jest kombinacją nauczonych parametrów *x* i wartości wejściowych (cech), dających w wyniku wartości klasyfikacyjne lub liczbowe. Funkcja kosztu dostarcza informacji o tym, jak daleko funkcja straty znajduje się od globalnego minimum. Jako funkcja aktualizująca zmieniająca wektor parametrów *x* wykorzystywana jest pochodna funkcji straty.

Obliczając pochodną funkcji straty, można modyfikować wartość każdego parametru w wektorze *x* tak, aby przybliżyć się do minimum. Wykorzystywanym w tym celu równaniom przyjrzymy się bliżej w dalszej części rozdziału poświęconej regresji liniowej i logistycznej (klasyfikacji).

Jednak w przypadku problemów nieliniowych uzyskanie czystej funkcji straty nie zawsze jest możliwe. Kłopot z nieliniowymi obszarami polega na tym, że mogą w nich istnieć liczne doliny, a stosując metodę gradientu liniowego, nie zawsze wiadomo, czy dojdziemy do najniższego punktu całego obszaru, czy jednej z jego wyżej położonych dolin. Najniższy punkt w najgłębszej dolinie jest nazywany **globalnym minimum**, natomiast najniższe punkty pozostałych dolin — **lokalnymi minimami**. Jeżeli stosując metodę gradientu liniowego, dojdziemy do lokalnego minimum, wpadniemy w pułapkę. Jest to jedna z wad tego algorytmu. W rozdziale 6., poświęconym hiperparametrom i prędkości uczenia, opiszemy, jak rozwiązać ten problem.

Inny problem ujawniający się w metodzie gradientu prostego jest związany z nieznormalizowanymi cechami. Termin „nieznormalizowane cechy" oznacza cechy wyrażane wartościami w bardzo różnych skalach. Jeżeli jedna cecha jest wyrażana w milionach jednostek, a inna w dziesiątkach, wtedy określenie najbardziej stromego zbocza minimalizującego błąd jest bardzo trudne.

Normalizacja

W rozdziale 8. szerzej opiszemy metody normalizacji danych pod kątem wektoryzacji. Pokażemy też kilka sposobów, jak sobie radzić z tym zadaniem.

Stochastyczny gradient prosty

W metodzie gradientu prostego wyliczana jest ogólna strata dla wszystkich prób treningowych, następnie obliczany jest gradient i aktualizowany wektor parametrów. W metodzie stochastycznego gradientu prostego wartość gradientu i wektor parametru są aktualizowane dla każdej próby treningowej.

Jak się okazało, dzięki temu uczenie odbywa się szybciej, a proces można łatwo zrównoleglić, o czym piszemy w dalszej części książki. Metoda stochastycznego gradientu prostego jest w pewnym sensie „wsadową" metodą gradientu prostego.

Minipaczki i stochastyczny gradient prosty

W jednej z odmian metody stochastycznego gradientu prostego do wyliczania gradientu jest wykorzystywanych kilka prób treningowych mniejszych niż pełny zestaw treningowy. Ta odmiana jest nazywana uczeniem na podstawie minipaczek. Jak się okazało, jest ona znacznie bardziej wydajna od metody wykorzystującej tylko jeden zbiór treningowy. Dzięki zastosowaniu minipaczek w metodzie stochastycznego gradientu prostego osiąga się również szybszą konwergencję, ponieważ gradient jest wyliczany w każdym kroku i wykorzystywanych jest w tym celu więcej prób treningowych.

W miarę powiększania wielkości minipaczki wyliczony gradient jest bardziej zbliżony do „prawdziwego" gradientu całego zbioru treningowego. Dzięki temu osiąga się większą wydajność obliczeń. Jeżeli minipaczka jest zbyt mała (np. zawiera tylko jeden rekord), wtedy komputer, a szczególnie procesor graficzny, nie jest wykorzystywany tak efektywnie, jak mógłby być. Jednakże nadmierne powiększanie minipaczki również jest nieefektywne, ponieważ taki sam gradient można wyliczyć mniejszym nakładem pracy (w niektórych przypadkach), stosując zwykłą metodę gradientu prostego.

Quasi-newtonowskie metody optymalizacyjne

Quasi-newtonowskie metody optymalizacyjne to algorytmy iteracyjne polegające na wykonywaniu serii „wyszukiwań liniowych". Od innych metod optymalizacyjnych różnią się sposobami wyboru kierunku wyszukiwania. Metody te opisane są w dalszej części książki.

Macierze Jacobiego i Hessego

Macierz Jacobiego to macierz o wymiarach $m \times n$ zbudowana z pochodnych cząstkowych pierwszego rzędu.

Macierz Hessego to macierz kwadratowa zbudowana z pochodnych cząstkowych drugiego rzędu. Opisuje ona krzywiznę funkcji wielu zmiennych. Macierz ta jest wykorzystywana w metodach newtonowskich do rozwiązywania problemów optymalizacyjnych dużej skali, ponieważ zawiera ona współczynniki równania kwadratowego w lokalnym rozwinięciu szeregu Taylora. W praktyce okazuje się, że obliczanie macierzy Hessego jest trudne. Zamiast tego stosuje się algorytmy quasi-newtonowskie. Przykładem takiego algorytmu jest L-BFGS, który bardziej szczegółowo opiszemy w rozdziale 2.

W tej książce rzadko odwołujemy się do macierzy Jacobiego i Hessego. Chcemy jednak, aby czytelnik o nich wiedział i znał szerszą rolę, jaką odgrywają w dziedzinie uczenia maszynowego.

Modele generatywne i dyskryminatywne

Model w zależności od swojego typu może generować wyniki różnego rodzaju. Istnieją dwa główne typy modeli: **generatywne** i **dyskryminatywne**. Modele generatywne podczas generowania wyników uwzględniają sposób utworzenia danych. Modele dyskryminatywne nie zajmują się sposobem utworzenia danych i po prostu klasyfikują dany sygnał wejściowy. Modele dyskryminatywne skupiają się na ścisłym modelowaniu granic pomiędzy klasami danych i wyznaczają ją dokładniej niż modele generatywne. Modele dyskryminatywne są zazwyczaj stosowane w uczeniu maszynowym do klasyfikowania danych.

Model generatywny uczy się wspólnego rozkładu prawdopodobieństwa $p(x, y)$, natomiast model dyskryminatywny uczy się **warunkowego** rozkładu $p(x \mid y)$. Rozkład $p(x \mid y)$ jest naturalnym rozkładem wartości x dającej wynik (lub klasę) y — stąd nazwa „model dyskryminatywny". Modele generatywne są wykorzystywane do generowania prawdopodobnych wyników na podstawie określonych danych wejściowych. Są tworzone jako probabilistyczne modele graficzne uwzględniające subtelne zależności między danymi.

Regresja logistyczna

Regresja logistyczna to popularny rodzaj klasyfikacji w uczeniu maszynowym. Stosuje się ją zarówno w klasyfikatorach binarnych, jak i wieloetykietowych (regresja logistyczna wielomianowa). Z technicznego punktu widzenia regresja logistyczna jest modelem regresyjnym z kategoryczną zmienną zależną. Binarny model logistyczny jest wykorzystywany do szacowania prawdopodobieństwa binarnego wyniku na podstawie jednej lub kilku zmiennych wejściowych (zmiennych niezależnych lub cech). Wynik jest statystycznym prawdopodobieństwem kategorii dla określonych predykatorów.

Podobnie jak w przypadku regresji liniowej, w modelu regresji logistycznej problem można wyrazić w postaci równania $Ax = b$, gdzie A oznacza macierz cech (np. wagę lub powierzchnię) wszystkich prób wejściowych, które chcemy modelować. Każdy rekord wejściowy jest wierszem w macierzy A, a wektor kolumnowy b zawiera wyniki dla wszystkich rekordów wejściowych w macierzy A. Wykorzystując funkcję kosztów i metodę optymalizacyjną, można znaleźć taki zbiór parametrów x, dla którego błąd wszystkich prognoz w porównaniu z rzeczywistymi wartościami będzie najmniejszy.

Jak poprzednio, do rozwiązania tego problemu stosuje się metodę stochastycznego gradientu prostego. Do znalezienia wektora x wykorzystywane są trzy komponenty:

Hipoteza dotycząca danych

$$f(x) = \frac{1}{1 + e^{-\theta x}}$$

Funkcja kosztu

„Oszacowanie maksymalnej wiarygodności"

Funkcja aktualizująca

Pochodna funkcji kosztu

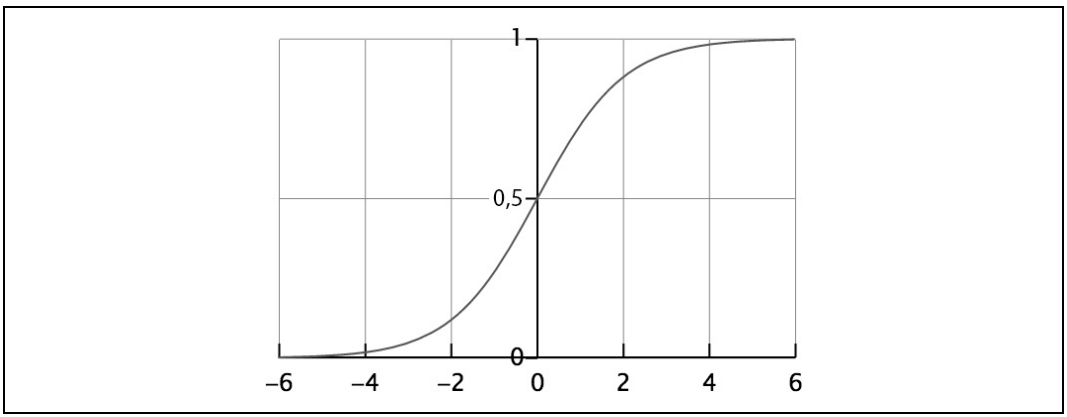
W tym przypadku danymi wejściowymi są zmienne niezależne (np. kolumny danych lub cechy), natomiast wynikami są zmienne zależne (np. etykiety). Prostą analogią jest funkcja regresji logistycznej wiążąca wartości wejściowe z wagami w celu określenia prawdopodobieństwa uzyskania wyniku. Przyjrzyjmy się bliżej funkcji logistycznej.

Funkcja logistyczna

W regresji logistycznej funkcję logistyczną („hipotezę") definiuje się w następujący sposób:

$$f(x) = \frac{1}{1 + e^{-\theta x}}$$

Jest to przydatna funkcja, ponieważ jej argumentem jest dowolna liczba z przedziału od minus do plus nieskończoności, a wynikiem liczba z przedziału od 0,0 do 1,0, którą można interpretować jako wartość prawdopodobieństwa. Rysunek 1.10 przedstawia wykres funkcji logistycznej.



Rysunek 1.10. Wykres funkcji logistycznej

Funkcja ta nosi również nazwę funkcji S-kształtnej, dającej wyniki z zakresu od 0,0 do 1,0. Z funkcją tą spotkamy się jeszcze w rozdziale 2. w części „Funkcje aktywacji".

Wyniki funkcji logistycznej

Funkcja logistyczna jest często oznaczana grecką literą sigma (σ), ponieważ dwuwymiarowy wykres zależności pomiędzy wartościami x i y jest podobny do wydłużonej litery „s". Asymptotyczne minimum i maksimum funkcji są odpowiednio równe 0 i 1.

Jeżeli y jest funkcją x i funkcja ta jest S-kształtna lub logistyczna, wówczas w miarę wzrostu wartości x wynik funkcji przybliża się do jedności, ponieważ stała e podniesiona do nieskończonej ujemnej potęgi jest równa zero. Natomiast im wartość x jest mniejsza od zera, tym wyrażenie $1 + e^{-\theta x}$ jest większe. Ponieważ wyrażenie to znajduje się w mianowniku ułamka, więc im większa jest wartość wyrażenia, tym wynik funkcji jest bliższy zeru.

W regresji logistycznej funkcja *f(x)* reprezentuje prawdopodobieństwo uzyskania wartości *y* równej 1 (lub *prawda*) dla danej wartości *x*. Jeżeli szacujemy prawdopodobieństwo, że dana wiadomość jest spamem, to wynik funkcji *f(x)* równy 0,6 oznacza, że istnieje 60-procentowe prawdopodobieństwo, że *y* jest równe 1 albo że wiadomość jest spamem. Jeżeli uczenie maszynowe zdefiniujemy jako metodę wnioskowania nieznanych wyników na podstawie znanych danych wejściowych, to wektor parametrów *x* w modelu regresji logistycznej określa siłę i pewność naszych dedukcji.

Transformacja logitowa

Funkcja logitowa jest odwróconą funkcją logistyczną („transformacja logistyczna").

Ocenianie modeli

Ocenianie modeli to proces, w którym określa się poprawność klasyfikacji i mierzy prognozowane wartości w określonym kontekście. Niekiedy interesuje nas tylko, jak często model dostarcza poprawnych prognoz, a w innym przypadku ważne jest, czy dany model daje poprawne prognozy określonego typu częściej niż inny model. W tej części rozdziału opiszemy takie zagadnienia jak szkodliwe pozytywne wyniki, nieszkodliwe negatywne wyniki i niezrównoważone klasy. Przyjrzyjmy się teraz podstawowemu narzędziu do oceniania modeli: **tablicy pomyłek**.

Tablica pomyłek

Tablica pomyłek (patrz rysunek 1.11), zwana również **macierzą błędów**, składa się z wierszy i kolumn zawierających prognozowane wartości i rzeczywiste wyniki (etykiety) klasyfikatora. Tablicy tej używa się do lepszego określenia, jak dobrze dany model lub klasyfikator nadaje się do generowania poprawnych wyników we właściwym czasie.

	Wynik pozytywny (prognozowany)	Wynik negatywny (prognozowany)
Wynik pozytywny (rzeczywisty)	Prawdziwie pozytywny	Fałszywie negatywny
Wynik negatywny (rzeczywisty)	Fałszywie pozytywny	Prawdziwie negatywny

Rysunek 1.11. Tablica pomyłek

Pomiar jakości wyników polega na zliczaniu następujących przypadków:

- Wynik prawdziwie pozytywny:

    - pozytywna prognoza,
    - pozytywna etykieta.
- Wynik fałszywie pozytywny:
    - pozytywna prognoza,
    - negatywna etykieta.
- Wynik prawdziwie negatywny:
    - negatywna prognoza,
    - negatywna etykieta.
- Wynik fałszywie negatywny:
    - negatywna prognoza,
    - pozytywna etykieta.

W tradycyjnej statystyce wynik fałszywie pozytywny jest zwany „błędem pierwszego rodzaju", a wynik fałszywie negatywny „błędem drugiego rodzaju". Poprzez zliczanie takich przypadków można uzyskać znacznie lepszą ocenę wydajności modelu niż zwykły procentowy odsetek poprawnych prognoz. Różne oceny modelu oblicza się na podstawie kombinacji liczb wystąpień wymienionych wyżej czterech przypadków. Wyniki umieszcza się w tablicy pomyłek, jak niżej:

```
Dokładność: 0,94
Precyzja: 0,8662
Pamięć: 0,8955
F1: 0,8806
```

Powyższy przykład zawiera cztery różne miary wykorzystywane do oceny modelu uczenia maszynowego. Krótko opiszemy każdą z nich. Wcześniej jednak przedstawmy podstawy oceniania czułości i swoistości modelu.

Czułość i swoistość modelu

Czułość i **swoistość** modelu to dwie różne miary stosowane do oceniania binarnego modelu klasyfikującego. Liczba prawdziwie pozytywnych wyników określa, jak często rekord wejściowy jest klasyfikowany jako pozytywny wynik, a więc klasyfikacja jest poprawna. Miara ta jest również zwana czułością modelu. Rozważmy przykład modelu klasyfikacji stanu pacjenta, który jest naprawdę chory. Czułość daje nam wyobrażenie, jak dobrze model zapobiega uzyskiwaniu wyników fałszywie negatywnych.

Czułość = PP / (PP + FN)

Gdyby nasz model sklasyfikował pacjenta z poprzedniego przykładu jako zdrowego i faktycznie pacjent ten byłby zdrowy, to uzyskany wynik byłby prawdziwie negatywny (swoisty). Swoistość określa, jak dobrze model zapobiega uzyskiwaniu wyników fałszywie pozytywnych.

Swoistość = PN / (PN + FP)

Często trzeba dokonywać kompromisu pomiędzy czułością a swoistością modelu. Przykładem jest model częściej stwierdzający poważną chorobę, ponieważ koszty błędnych diagnoz w przypadku

rzeczywiście chorych pacjentów są bardzo wysokie. Tego typu model ma niską swoistość. Poważna choroba może stanowić zagrożenie dla życia pacjenta i osób z jego otoczenia, dlatego model w takiej sytuacji musi mieć wysoką czułość. W idealnym świecie czułość modelu wynosi 100% (tj. model zawsze poprawnie stwierdza chorobę) i swoistość również równą 100% (tj. żaden zdrowy pacjent nie jest klasyfikowany jako chory).

Dokładność

Dokładność modelu to stopień zgodności pomiaru określonej wielkości z jej rzeczywistą wartością.

Dokładność = (PP + PN) / (PP + FP + FN + PN)

Dokładność może skutkować błędną oceną jakości modelu w przypadku dużej nierównowagi klas. Jeżeli po prostu wszystkie wyniki będą kwalifikowane do dużej klasy, wtedy model będzie automatycznie generował dużą liczbę poprawnych prognoz. Dokładność modelu będzie wprawdzie wysoka, ale model ten nie będzie miał praktycznego zastosowania (tj. nigdy nie będzie w stanie przewidzieć wyników należących do mniejszej klasy lub zdarzeń rzadziej występujących).

Precyzja

Stopień uzyskiwania tych samych wyników dla powtarzalnych pomiarów w takich samych warunkach w kontekście naukowym i statystycznym nosi nazwę **precyzji modelu**. Precyzja jest również nazywana **wartością pozytywnej prognozy**. Choć termin ten w mowie potocznej jest często stosowany zamiennie z dokładnością, w naukowym sensie oznacza jednak co innego.

Precyzja = PP / (PP + FP)

Pomiar może być dokładny i nieprecyzyjny, niedokładny i precyzyjny, niedokładny i nieprecyzyjny lub dokładny i precyzyjny. Pomiar jest uznawany za poprawny, jeżeli jest dokładny i precyzyjny.

Pamięć

Termin ten oznacza to samo co czułość modelu. Jest również nazywany **częstością wyników prawdziwie pozytywnych** lub **częstością trafień**.

F1

W klasyfikacji binarnej wartość F1 (F-miara) jest wyznacznikiem dokładności modelu. Jest to średnia harmoniczna opisanych wcześniej precyzji i czułości, zdefiniowana w następujący sposób:

F1 = 2PP / (2PP + FP + FN)

F1 przyjmuje wartości z przedziału od najgorszej 0,0 do najlepszej, pożądanej wartości 1,0. Zazwyczaj wskaźnik ten jest wykorzystywany do oceny jakości modeli wyszukujących informacje. W uczeniu maszynowym F1 jest traktowany jako ogólny wskaźnik jakości modelu.

Kontekst i interpretacja oceny

W procesie oceny modelu ważną rolę może odgrywać kontekst. W zależności od kontekstu stosuje się różnego rodzaju opisane wyżej oceny. Nierównowaga klas może odgrywać istotną rolę w wybo-

rze typu oceny, a w praktyce często okazuje się, że klasy i liczby etykiet nie są zrównoważone. Poniżej wymienione są typowe tego rodzaju sytuacje:

- prognozowanie kliknięć na stronach WWW,
- prognozowanie śmiertelności pacjentów na oddziale OIOM,
- wykrywanie nadużyć.

W powyższych kontekstach ogólna, praktyczna „procentowa poprawność" oceny modelu może być myląca. Przykładem jest zbiór danych z konkursu PhysioNet Challenge z 2012 roku (*https://physionet.org/challenge/2012*). Zadanie konkursowe polegało na zbudowaniu modelu wykorzystującego klasyfikator binarny do jak najdokładniejszego prognozowania śmiertelności pacjentów. Trudność tego zadania tkwiła w tym, że prognozowanie, iż pacjent będzie żył, było dość proste, ponieważ większość danych w zbiorze dotyczyła takiego przypadku. Celem było dokładne prognozowanie zejścia pacjenta i ta cecha stanowiła największą wartość modelu w praktycznym, medycznym kontekście. W tym konkursie model oceniany był w następujący sposób:

Ocena = MIN(precyzja, czułość)

Formuła w powyższym kształcie skutkowała tym, że uczestnicy konkursu skupiali się nie tylko na zbudowaniu modelu prognozującego, że w większości przypadków pacjent będzie żył, za co model uzyskałby wysoką ocenę F1, ale również na prognozowaniu zejścia (dzięki czemu model miałby zastosowanie praktyczne). Jest to doskonały przykład, jak kontekst może zmienić sposób oceniania modelu.

Sposoby na nierównowagę klas

W rozdziale 6. opiszemy praktyczne sposoby rozwiązywania problemu nierównowagi klas. Przyjrzymy się dokładniej różnym przypadkom nierównowagi i rozkładów błędów w kontekście klasyfikacji oraz regresji danych.

Poszerzanie wiedzy o uczeniu maszynowym

W tym rozdziale opisaliśmy najważniejsze pojęcia, niezbędne do wykorzystania uczenia maszynowego w praktyce. Przedstawiliśmy matematyczne podstawy modelowania danych oparte na równaniu:

$$Ax = b$$

Omówiliśmy również najważniejsze metody uzyskiwania cech dla macierzy A, zmieniania wektora parametrów x i wyliczania wyników w wektorze b. Zilustrowaliśmy kilka podstawowych sposobów modyfikowania wektora x w celu zminimalizowania oceny („straty") funkcji celu.

W kolejnych rozdziałach będziemy rozwijać opisane tu najważniejsze pojęcia. Dowiesz się, jak opisane podstawy są wykorzystywane w sieciach neuronowych i w głębokim uczeniu. Poznasz bardziej zaawansowane metody tworzenia macierzy A, zmieniania wektora x za pomocą metod optymalizacyjnych oraz mierzenia strat podczas uczenia się modelu. Teraz przejdźmy do rozdziału 2., w którym rozwiniemy opisane tu pojęcia, wykorzystując podstawy wiedzy o sieciach neuronowych.

Skorowidz

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

Q

R

S

Ś

T

U

V

W

Y

Z, Ż

O'REILLY®

Uczenie głębokie i sieci neuronowe: przyszłość, która dzieje się dziś!

Technologie wykorzystujące różne formy uczenia maszynowego zaczynają pojawiać się w różnych branżach. Możliwości w tym zakresie stale rosną, podobnie jak zainteresowanie i oczekiwania. Przed podjęciem decyzji o wdrożeniu w firmie tego rodzaju rozwiązań trzeba jednak zadać sobie pytanie, co można i co chciałoby się osiągnąć za pomocą sieci neuronowej. Generalnie uczenie maszynowe opiera się na algorytmach wyodrębniania informacji z surowych danych i reprezentowania ich jako modelu. Następnie model ten służy do przetwarzania kolejnych surowych danych. Co to jednak oznacza w praktyce i jak się implementuje takie algorytmy?

Niniejsza książka jest przydatnym przewodnikiem po uczeniu maszynowym i sieciach neuronowych. Zawiera praktyczne informacje, które doceni każdy programista stawiający pierwsze kroki w tej dziedzinie. Przedstawiono tu podstawy deep learningu i wyjaśniono takie pojęcia jak strojenie sieci, wielowątkowość, wektoryzowanie danych. Opisano, w jaki sposób można wykorzystać otwartą bibliotekę Deeplearning4j (DL4J) do kodowania profesjonalnych procesów uczenia głębokiego. Zaprezentowano metody i strategie trenowania sieci głębokich i uruchamiania procesów uczenia głębokiego w środowiskach Spark i Hadoop. Zagadnienia te zostały zilustrowane gotowymi do zastosowania, praktycznymi przykładami.

W tej książce między innymi:

- ogólne koncepcje uczenia maszynowego, uczenia głębokiego i sieci neuronowych
- ewolucja sieci neuronowych do sieci głębokich i ich rodzaje
- dobieranie rodzaju sieci do analizowanego zagadnienia
- strojenie sieci neuronowych i sieci głębokich
- stosowanie biblioteki DL4J w środowiskach Spark i Hadoop

Josh Patterson – jest uznanym autorytetem w dziedzinie przetwarzania wielkich ilości danych, uczenia maszynowego i uczenia głębokiego. Aktywnie działa na rzecz tworzenia otwartego oprogramowania, uczestniczy w takich projektach jak DL4J, Apache Mahout, Metronome, IterativeReduce, openPDC i JMotif.

Adam Gibson – specjalizuje się w uczeniu głębokim. Ma duże doświadczenie w budowaniu systemów do przetwarzania dużych ilości danych w czasie rzeczywistym. Z jego rozwiązań korzystają m.in. firmy z listy Fortune 500, towarzystwa ubezpieczeniowe, agencje public relations i startupy.

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

KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-4227-9
9 788328 342279
INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 77,00 zł