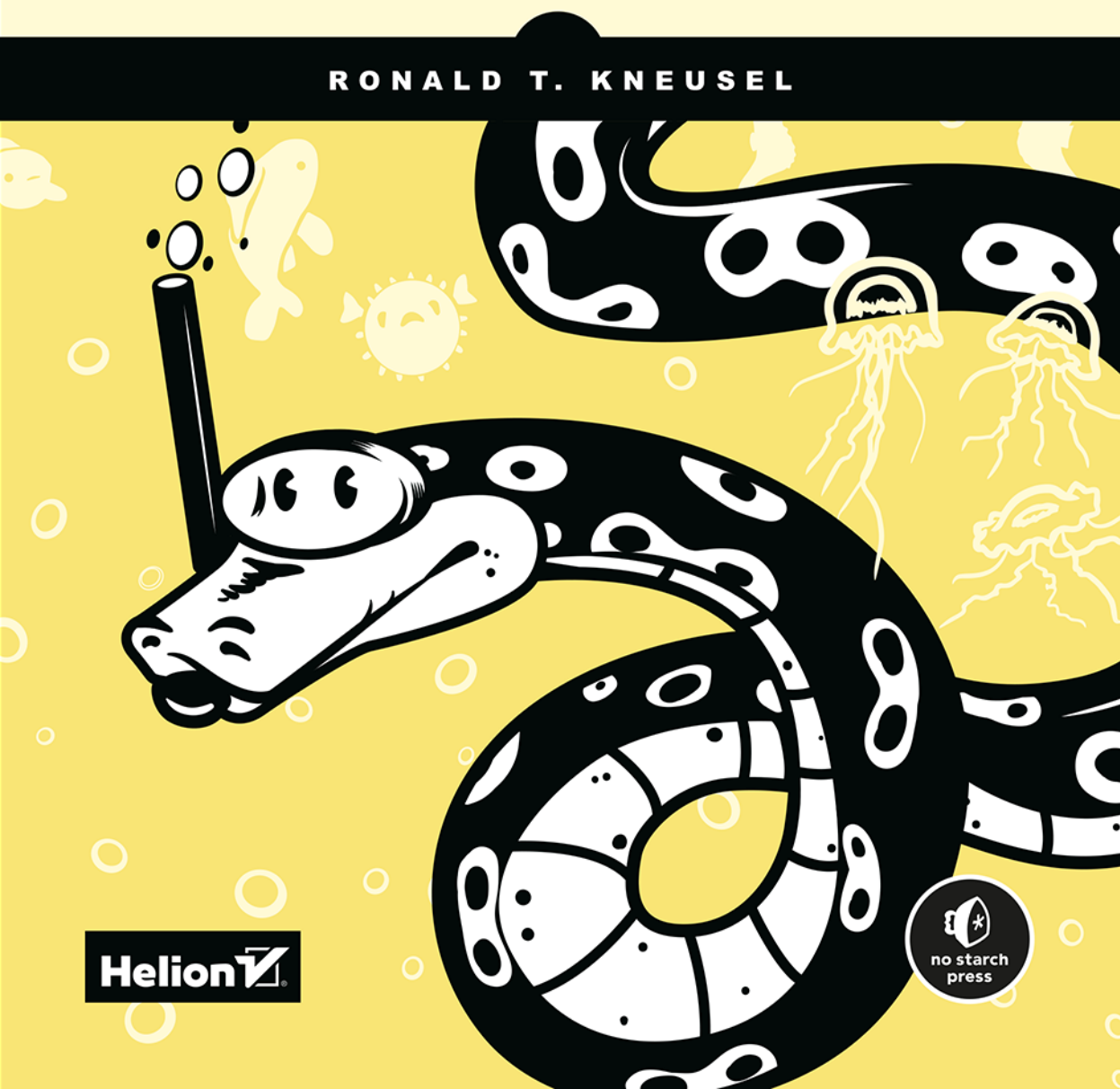


DEEP LEARNING

PRAKTYCZNE WPROWADZENIE
Z ZASTOSOWANIEM ŚRODOWISKA PYTHONA

RONALD T. KNEUSEL



Helion



Tytuł oryginału: Practical Deep Learning A Python-Based Introduction

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-283-8859-8

Copyright © 2021 by Ronald T. Kneusel. Title of English-language original: Practical Deep Learning: A Python-Based Introduction, ISBN 9781718500747, published by No Starch Press Inc. 245 8th Street, San Francisco, California United States 94103. The Polish-language edition Copyright © 2022 by Helion S.A. under license by No Starch Press Inc. All rights reserved.

The following images are reproduced with permission. Figure 4-2: the middle image is licensed under the Creative Commons Attribution 2.0 Generic license (https://commons.wikimedia.org/wiki/File:Border_Collie_liver_portrait.jpg) and the right image is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license (https://commons.wikimedia.org/wiki/File:Lynn-red_merle_Aussie_12_Months.jpg).

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 jakiegokolwiek 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 właścicieli.

Autor oraz wydawca 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 wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/delepw>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<https://ftp.helion.pl/przyklady/delepw.zip>

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

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

Spis treści

PRZEDMOWA	13
PODZIĘKOWANIA	16
WSTĘP	17
1	
PIERWSZE KROKI	24
Środowisko operacyjne	24
NumPy	25
scikit-learn	25
Keras i TensorFlow	25
Instalacja narzędzi	26
Podstawy algebry liniowej	27
Wektory	27
Macierze	28
Mnożenie wektorów i macierzy	28
Statystyka i prawdopodobieństwo	30
Statystyka opisowa	30
Rozkłady prawdopodobieństwa	31
Testy statystyczne	32
Procesory graficzne (GPU)	33
Podsumowanie	33
2	
KORZYSTANIE Z PYTHONA	34
Interpreter Pythona	34
Instrukcje i białe znaki	35
Zmienne i podstawowe struktury danych	36
Przedstawianie liczb	37
Zmienne	37
łańcuchy znaków	38
Listy	38
Słowniki	42

Struktury sterowania	43
Instrukcje if-elif-else	43
Pętle for	44
Pętle while	46
Instrukcje break i continue	47
Instrukcja with	48
Obsługa błędów za pomocą bloków try-except	48
Funkcje	49
Moduły	51
Podsumowanie	53

3

BIBLIOTEKA NUMPY	54
Dlaczego NumPy?	54
Tablice a listy	55
Testowanie szybkości tablic i list	56
Podstawowe tablice	58
Definiowanie tablicy za pomocą funkcji, np. array	59
Definiowanie tablic wypełnionych zerami i jedynekami	62
Dostęp do elementów tablicy	64
Indeksowanie tablicowe	64
Uzyskiwanie wycinków tablicy	65
Wielokropek	68
Operatory i rozgłaszanie	69
Dane wejściowe i wyjściowe tablic	72
Liczby losowe	75
Biblioteka NumPy i obrazy	76
Podsumowanie	78

4

PRACA Z DANYMI	79
Klasy i etykiety	79
Cechy i wektory cech	80
Rodzaje cech	81
Dobór cech i kłątwa wymiarowości	83
Własności dobrego zestawu danych	86
Interpolacja i ekstrapolacja	86
Główny rozkład prawdopodobieństwa	89
Rozkład a priori	89
Przykłady mylące	90
Rozmiar zestawu danych	91
Przygotowanie danych	92
Skalowanie cech	92
Brakujące cechy	97

Dane uczące, walidacyjne i testowe	98
Trzy podzbiory	98
Dzielenie zestawu danych	99
k-krotny sprawdzian krzyżowy	105
Analiza danych	107
Wyszukiwanie problemów z danymi	107
Opowieści ku przestrodze	112
Podsumowanie	113
 5	
BUDOWANIE ZESTAWÓW DANYCH	114
Kosańce (zestaw danych Iris)	114
Nowotwory piersi (zestaw danych Breast Cancer)	117
Cyfry zapisane pismem odręcznym (zestaw danych MNIST)	119
Różne obrazy (zestaw danych CIFAR-10)	122
Rozszerzanie danych	124
Dlaczego należy rozszerzać dane uczące?	125
Sposoby rozszerzania danych	126
Rozszerzanie zestawu danych Iris	127
Rozszerzanie zestawu danych CIFAR-10	133
Podsumowanie	138
 6	
KLASYCZNE UCZENIE MASZYNOWE	139
Algorytm najbliższego centroidu	140
Algorytm k najbliższych sąsiadów	144
Naiwny klasyfikator Bayesa	146
Drzewa decyzyjne i lasy losowe	150
Rekurencja	153
Budowanie drzew decyzyjnych	154
Lasy losowe	155
Maszyny wektorów nośnych	157
Marginesy	157
Wektory nośne	159
Optymalizacja	160
Jądra	160
Podsumowanie	161
 7	
EKSPERYMENTOWANIE Z KLASYCZNYMI MODELAMI	162
Eksperymenty z użyciem zestawu danych Iris	162
Testowanie klasycznych modeli	163
Implementacja klasyfikatora najbliższego centroidu	167

Eksperymenty z użyciem zestawu danych Breast Cancer	168
Dwa pierwsze przebiegi testowe	169
Skutek losowych podziałów	172
Dodawanie k-krotnego sprawdzianu krzyżowego	174
Poszukiwanie hiperparametrów	179
Eksperymenty z użyciem zestawu danych MNIST	185
Testowanie klasycznych modeli	185
Analiza czasu działania	192
Eksperymenty z głównymi składowymi analizy PCA	195
Tasowanie zestawu danych	197
Podsumowanie klasycznych modeli	199
Algorytm najbliższego centroidu	199
Algorytm k najbliższych sąsiadów	200
Naiwny klasyfikator Bayesa	200
Drzewa decyzyjne	201
Lasy losowe	201
Maszyny wektorów nośnych	202
Kiedy używać klasycznych modeli?	202
Korzystanie z małych zestawów danych	202
Ograniczone zasoby obliczeniowe	203
Dostęp do wyjaśnialnych modeli	203
Praca z danymi wektorowymi	203
Podsumowanie	204

8

WPROWADZENIE DO SIECI NEURONOWYCH	205
Anatomia sieci neuronowej	206
Neuron	207
Funkcje aktywacji	208
Architektura sieci	212
Warstwy wyjściowe	214
Wagi i obciążenia	216
Implementacja prostej sieci neuronowej	217
Przygotowanie zestawu danych	217
Implementacja sieci neuronowej	219
Uczenie i testowanie sieci neuronowej	221
Podsumowanie	223

9

UCZENIE SIECI NEURONOWEJ	224
Ogólny opis	224
Algorytm gradientu prostego	225
Wyszukiwanie minimów	227
Aktualizowanie wag	229

Stochastyczny spadek wzdłuż gradientu	230
Grupy i minigrupy	230
Funkcje wypukłe i niewypukłe	232
Kończenie uczenia	233
Aktualizowanie współczynnika uczenia	234
Momentum	235
Propagacja wsteczna	235
Propagacja wsteczna — ujęcie pierwsze	236
Propagacja wsteczna — ujęcie drugie	240
Funkcje straty	243
Błąd bezwzględny i błąd średniokwadratowy	244
Entropia krzyżowa	245
Inicjalizowanie wag	246
Przetrenowanie i regularyzacja	248
Przetrenowanie	248
Regularyzacja	251
Regularyzacja L2	251
Porzucanie	253
Podsumowanie	255

10

EKSPERYMENTOWANIE Z SIECIAMI NEURONOWYMI257

Nasz zestaw danych	257
Klasa MLPClassifier	258
Struktura sieci i funkcje aktywacji	259
Kod	259
Wyniki	263
Rozmiar grupy	267
Podstawowy współczynnik uczenia	271
Rozmiar zbioru uczącego	274
Regularyzacja L2	275
Momentum	278
Inicjalizacja wag	279
Kolejność cech	284
Podsumowanie	286

11

OCENIANIE MODELI287

Definicje i założenia	287
Dlaczego dokładność jest niewystarczająca?	288
Macierz pomyłek 2×2	290
Wskaźniki wyprowadzane z macierzy pomyłek	293
Wyprowadzanie wskaźników na podstawie macierzy pomyłek	294
Interpretowanie modeli za pomocą wskaźników	297

Zaawansowane wskaźniki	299
Informatywność i nacechowanie	300
Wskaźnik F1	300
Współczynnik kappa Cohena	301
Współczynnik korelacji Matthews'a	302
Implementacja zaawansowanych wskaźników	302
Krzywa charakterystyki operacyjnej odbiornika	304
Dobór modeli	304
Rysowanie wykresu wskaźników	306
Analiza krzywej ROC	307
Porównywanie modeli za pomocą analizy ROC	310
Generowanie krzywej ROC	312
Krzywa precyzji-czułości	314
Przypadki wieloklasowe	315
Rozszerzanie macierzy pomyłek	315
Obliczanie dokładności ważonej	318
Wieloklasowy współczynnik korelacji Matthews'a	321
Podsumowanie	322
 12	
WPROWADZENIE DO SPLOTOWYCH SIECI NEURONOWYCH	323
Dlaczego spłotowe sieci neuronowe?	324
Spłot	325
Skanowanie za pomocą jądra	325
Spłot w przetwarzaniu obrazów	328
Anatomia spłotowej sieci neuronowej	329
Różne typy warstw	330
Przepuszczanie danych przez sieć spłotową	332
Warstwy spłotowe	333
Mechanizm działania warstwy spłotowej	333
Korzystanie z warstwy spłotowej	336
Wielokrotne warstwy spłotowe	339
Inicjalizacja warstwy spłotowej	340
Warstwy łączące	340
Warstwy w pełni połączone	343
Pełne warstwy spłotowe	344
Krok po kroku	346
Podsumowanie	349
 13	
EKSPERYMENTOWANIE Z BIBLIOTEKĄ KERAS I ZESTAWEM DANYCH MNIST	351
Budowanie sieci spłotowych w bibliotece Keras	352
Wczytywanie danych MNIST	352
Budowanie modelu	354

Uczenie i ocena modelu	356
Tworzenie wykresu funkcji błędu	358
Podstawowe eksperymenty	360
Eksperymenty na architekturze	361
Rozmiar zbioru uczącego, minigrup oraz liczba epok	365
Optymalizatory	369
Pełne sieci splotowe	371
Budowa i trenowanie modelu	371
Przygotowanie obrazów testowych	374
Testowanie modelu	375
Potasowane cyfry MNIST	383
Podsumowanie	384
 14	
EKSPERYMENTOWANIE Z ZESTAWEM DANYCH CIFAR-10	386
Zestaw CIFAR-10 — przypomnienie	386
Praca na pełnym zestawie CIFAR-10	387
Budowanie modeli	388
Analizowanie modeli	392
Zwierzę czy pojazd?	395
Model binarny czy wieloklasowy?	400
Uczenie transferowe	405
Strojenie modelu	411
Przygotowanie zestawów danych	412
Dostosowanie modelu do strojenia	415
Testowanie modelu	418
Podsumowanie	420
 15	
STUDIUM PRZYPADKU: KLASYFIKOWANIE PRÓBEK DŹWIĘKOWYCH	421
Budowanie zestawu danych	422
Rozszerzanie zestawu danych	423
Wstępne przetwarzanie danych	427
Klasyfikowanie cech dźwiękowych	430
Modele klasyczne	430
Tradycyjna sieć neuronowa	432
Splotowa sieć neuronowa	433
Spektrogramy	439
Klasyfikowanie spektrogramów	443
Inicjalizacja, regularyzacja i normalizacja wsadowa	445
Analiza macierzy pomyłek	448
Zespoły	449
Podsumowanie	454

DALSZE KROKI	456
Co dalej z sieciami splotowymi?	456
Uczenie przez wzmacnianie i uczenie nienadzorowane	457
Generatywne sieci przeciwstawne	458
Rekurencyjne sieci neuronowe	459
Zasoby internetowe	459
Konferencje	460
Książka	462
Cześć i dzięki za ryby	462
SKOROWIDZ	463

1

Pierwsze kroki



ROZDZIAŁ TEN STANOWI WPROWADZENIE DO NASZEGO ŚRODOWISKA OPERACYJNEGO ORAZ ZAWIERA SZCZEGÓŁY JEGO KONFIGURACJI. ZNAJDZIESZ TU TAKŻE POWTÓRZENIE PEWNYCH ELEMENTÓW MATEMATYKI, z których będziemy korzystać w dalszej części książki. Ostatni podrozdział został poświęcony krótkiemu omówieniu procesorów graficznych (ang. graphic processing unit — GPU), które mają duże znaczenie w uczeniu głębokim. Na szczęście w naszym przypadku nie odgrywają istotnej roli, dlatego nie musisz się martwić; korzystanie z tej książki nie będzie wiązało się z niespodziewanymi olbrzymimi kosztami.

Środowisko operacyjne

W tym podrozdziale zajmiemy się opisem środowiska, z którego będziemy korzystać aż do końca książki. Zakładamy tu, że używasz 64-bitowego Linuksa. Rodzaj dystrybucji nie jest zbyt istotny, ale żeby dodatkowo ułatwić wybór, przyjmijmy także, że korzystamy z Linuksa Ubuntu 20.04. Mając na uwadze znakomite wsparcie dystrybucji Ubuntu, uważamy, że nowsza wersja nie powinna powodować problemów. Python jest naszym *lingua franca*, powszechnym językiem uczenia maszynowego. Dokładniej mówiąc, będziemy używać Pythona w wersji 3.8.2; jest to wersja dostępna w Ubuntu 20.04.

Przyjrzyjmy się pokrótce używanym przez nas narzędziom Pythona.

NumPy

NumPy jest biblioteką Pythona dodającą do środowiska przetwarzanie tablicowe. Listy Pythona można wykorzystywać jako jednowymiarowe tablice, jednak w praktyce są zbyt powolne i ograniczone. Biblioteka NumPy dodaje funkcje tablicowe, których brakuje Pythonowi; są one niezbędne w wielu zastosowaniach naukowych. NumPy stanowi bibliotekę bazową wykorzystywaną przez wszystkie pozostałe biblioteki, z których będziemy korzystać.

scikit-learn

Wszystkie omówione w tej książce tradycyjne modele uczenia maszynowego dostępne są w znakomitej bibliotece **scikit-learn** (czyli `sklearn`, jak jest nazywana po wczytaniu do Pythona). Zwróć uwagę, że zapis nazwy pozbawiony jest dużych liter, gdyż takiej właśnie notacji używają konsekwentnie sami twórcy w dokumentacji biblioteki. Biblioteka ta wykorzystuje tablice NumPy. Implementuje ona interfejs ustandaryzowany dla wielu różnych modeli uczenia maszynowego, a także całe mnóstwo funkcji, o których nie zdążę nawet wspomnieć. Bardzo polecam zapoznanie się z oficjalną dokumentacją `sklearn` (<https://scikit-learn.org/stable/index.html>) w miarę ustawicznego zapoznawania się z uczeniem maszynowym i narzędziami służącymi do projektowania modeli.

Keras i TensorFlow

Samo w sobie zrozumienie uczenia głębokiego nie jest łatwe, nie wspominając nawet o jego wydajnej oraz poprawnej implementacji, dlatego zamiast próbować rozpisywać od podstaw splotową sieć neuronową, skorzystamy z popularnego, aktywnie rozwijanego narzędzia. Od samego początku społeczność uczenia głębokiego wspiera rozwój narzędzi ułatwiających korzystanie z sieci głębokich i udostępnia otwarte narzędzia objęte bardzo przyjaznymi licencjami. W czasie pisania tej książki dostępnych jest wiele popularnych narzędzi przygotowanych z myślą o Pythonie. Są to między innymi:

- Keras,
- PyTorch,
- Caffe,
- Caffe2,
- Apache MXnet.

Niektóre z tych narzędzi są prężnie rozwijane, o innych stopniowo zapominamy. Obecnie jednak jednym z mających najbardziej oddaną społeczność jest biblioteka Keras, wykorzystująca z kolei bibliotekę TensorFlow, dlatego będziemy z niej korzystać w tej książce.

Keras (<https://keras.io/>) jest biblioteką uczenia głębokiego wykorzystującą do działania bibliotekę **TensorFlow** (<https://www.tensorflow.org/>). Z kolei TensorFlow to otwarty produkt firmy Google, implementujący kluczowe funkcje głębokich sieci neuronowych na wielu różnych platformach. Wybraliśmy bibliotekę

Keras nie tylko z powodu popularności i błyskawicznego rozwoju, lecz także dlatego, że jest prosta w obsłudze. Naszym celem jest zrozumienie uczenia głębokiego w stopniu umożliwiającym implementowanie modeli, a następnie korzystanie z nich przy minimum złożoności programistycznej.

Instalacja narzędzi

Nie jesteśmy w stanie opisać procesu instalacji narzędzi na wszystkich systemach operacyjnych i na wszystkich konfiguracjach sprzętowych. Zamiast tego zaprezentujemy instrukcje opisujące krok po kroku proces instalacji dla określonego systemu operacyjnego będącego naszym systemem referencyjnym. Etapy te, wraz z minimalnymi wymaganymi wersjami bibliotek, powinny wystarczyć większości czytelników do przygotowania działającego środowiska roboczego.

Pamiętaj, że zakładamy pracę w środowisku linuksowym, dokładniej w dystrybucji Ubuntu 20.04. Ubuntu jest szeroko rozpowszechnioną dystrybucją Linuksa i działa na niemal każdym nowym komputerze. Inne dystrybucje Linuksa również sprawdzają się jako środowisko pracy, podobnie jak system macOS, ale podane tu instrukcje są przeznaczone dla dystrybucji Ubuntu. Społeczność uczenia maszynowego raczej nie korzysta z systemu Windows. Istnieją jednak narzędzia przystosowane do tego systemu; zatem odważny czytelnik może spróbować także tam swych sił¹.

Świeżo zainstalowana dystrybucja Ubuntu 20.04 zawiera domyślnie środowisko Python 3.8.2. W celu zainstalowania pozostałych pakietów musimy wejść do powłoki i wpisać poniższe polecenia w podanej kolejności:

```
$ sudo apt-get update
$ sudo apt-get install python3-pip
$ sudo apt-get install build-essential python3-dev
$ sudo apt-get install python3-setuptools python3-numpy
$ sudo apt-get install python3-scipy libatlas-base-dev
$ sudo apt-get install python3-matplotlib
$ pip3 install scikit-learn
$ pip3 install tensorflow
$ pip3 install pillow
$ pip3 install h5py
$ pip3 install keras
```

Po zakończeniu instalacji będziemy dysponować następującymi wersjami narzędzi i bibliotek:

¹ Nie jest to do końca prawda. System Windows jest dobrze wspierany pod względem środowiska Python i bibliotek uczenia maszynowego. Osoby zainteresowane mogą chcieć zainstalować środowisko *Anaconda*, a następnie dołączyć wymagane biblioteki; opisane w książce przykłady działają zarówno w Linuksie, jak i w systemie Windows — *przyp. tłum.*

```
NumPy 1.17.4
sklearn 0.23.2
keras 2.4.3
tensorflow 2.2.0
pillow 7.0.0
h5py 2.10.0
matplotlib 3.1.2
```

Biblioteka pillow służy do przetwarzania obrazów, h5py pozwala na pracę z plikami w formacie HDF5, natomiast matplotlib umożliwia tworzenie wykresów. **HDF5** to ogólny, hierarchiczny format plików służący do przechowywania danych naukowych. Biblioteka Keras przechowuje w tym formacie parametry modelu.

W dwóch następnych podrozdziałach znajdziesz łagodne wprowadzenie do pewnych pojęć matematycznych spotykanych w dalszej części książki.

Podstawy algebry liniowej

Przyjrzyjmy się teraz wektorom i macierzom. Opisujący te twory dział matematyki nazywany jest **algebrą liniową** lub teorią macierzy. Jak można się domyślić, algebra liniowa jest całkiem skomplikowaną dziedziną. Nam wystarczy wiedzieć, czym jest wektor oraz macierz, a także jak mnożymy dwa wektory, dwie macierze lub wektor przez macierz. Przekonamy się później, że dzięki temu uzyskujemy bardzo przydatny mechanizm implementowania określonych modeli, zwłaszcza sieci neuronowych.

Zacznijmy od wektorów.

Wektory

Wektor (ang. *vector*) jest jednowymiarową listą wartości liczbowych. W ujęciu matematycznym może przybierać on następującą postać:

$$a = [0, 1, 2, 3, 4]$$

Trzeci element jest zapisywany jako $a_2 = 2$. Zwróć uwagę, że korzystamy z konwencji programistycznej, zgodnie z którą indeksy wyznaczamy od wartości 0, zatem a_2 daje nam trzeci element wektora.

Powyższy wektor zapisaliśmy poziomo, dlatego nazywamy go **wektorem wierszowym** (ang. *row vector*). Jednak wektory stosowane w wyrażeniach matematycznych zazwyczaj są zapisywane w pionie:

$$a = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Wektor zapisywany w pionie nazywany jest **wektorem kolumnowym** (ang. *column vector*). W naszym przykładzie zawiera on pięć elementów, dlatego mówimy, że jest pięcioelementowym wektorem kolumnowym. W tej książce wektor symbolizuje pojedynczą **próbę** (ang. *sample*), czyli jeden zestaw cech wprowadzanych na wejściu modelu.

W ujęciu matematycznym wektory służą do reprezentowania punktów w przestrzeni. Jeżeli weźmiemy dwuwymiarową płaszczyznę kartezjańską, to wyznaczamy położenie danego punktu za pomocą dwuelementowego wektora (x, y) , gdzie x stanowi odległość wzdłuż osi x , natomiast y jest odległością wzdłuż osi y . Wektor taki określa punkt w dwóch wymiarach, mimo że sam jest jednowymiarowy. Gdybyśmy mieli punkt w trzech wymiarach, musielibyśmy użyć wektora trójelementowego (x, y, z) .

W uczeniu maszynowym wektory służą często do symbolizowania sygnałów wejściowych do modelu, dlatego pracujemy z dziesiątkami, a nawet setkami wymiarów. Oczywiście nie jesteśmy w stanie zaznaczać ich na wykresie jako punktów w przestrzeni, ale tym właśnie są w ujęciu matematycznym. Jak się przekonamy, niektóre modele, na przykład algorytm k najbliższych sąsiadów, wykorzystują wektory cech właśnie jako punkty w wielowymiarowej przestrzeni.

Macierze

Macierz (ang. *matrix*) jest dwuwymiarową tablicą zawierającą wartości liczbowe, w której dany element określamy za pomocą numeru wiersza i kolumny. Poniżej widzimy przykładową macierz:

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Jeżeli odnosimy się do liczby 6, to zapisujemy $a_{1,2} = 6$. Ponownie indeksujemy tu od wartości 0. Macierz a zawiera po trzy wiersze i kolumny, dlatego nazywamy ją macierzą 3×3 .

Mnożenie wektorów i macierzy

Mnożenie dwóch wektorów jest niezmiernie proste; wystarczy pomnożyć elementy znajdujące się na tych samych pozycjach. Na przykład:

$$[1,2,3] \cdot [4,5,6] = [4,10,18]$$

Jest to najczęściej spotykany sposób mnożenia tablicy podczas używania biblioteki NumPy i będziemy z niego bardzo często korzystać w kolejnych rozdziałach. Jednak w typowej matematyce nie jest on zbyt często używany.

W przypadku klasycznie matematycznego iloczynu wektorów musimy brać pod uwagę fakt, czy są one wierszowe, czy kolumnowe. Będziemy pracować na dwóch wektorach: $A = (a, b, c)$ i $B = (d, e, f)$, które, zgodnie z przyjętą matematyczną konwencją, są domyślnie kolumnowe. Dodanie indeksu górnego T

przekształca wektor kolumnowy w wierszowy. Matematycznie dopuszczalnymi sposobami mnożenia A i B są:

$$AB^T = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} d & e & f \end{bmatrix} = \begin{bmatrix} ad & ae & af \\ bd & be & bf \\ cd & ce & cf \end{bmatrix}$$

Operacja ta nazywana jest **iloczynem zewnętrznym** (ang. *outer product*). Z kolei operacja

$$A^T B = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} d \\ e \\ f \end{bmatrix} = ad + be + cf$$

jest **iloczynem wewnętrznym** (ang. *inner product*), znanym także jako **iloczyn skalarny** (ang. *dot product*). Zwróć uwagę, że w wyniku iloczynu zewnętrznego otrzymujemy macierz, a rezultatem iloczynu wewnętrznego jest pojedyncza liczba (**skalar**).

Podczas mnożenia macierzy i wektora ten drugi zazwyczaj znajduje się po prawej stronie macierzy. Operację można przeprowadzić tylko wtedy, gdy liczba kolumn macierzy jest równa liczbie elementów wektora (który również w tym przypadku powinien być kolumnowy). W konsekwencji otrzymujemy wektor o liczbie elementów równej liczbie wierszy macierzy ($ax + by + cz$ jest jednym elementem).

Na przykład:

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \end{bmatrix}$$

Pomnożyliśmy tu macierz 2×3 przez wektor kolumnowy 3×1 , dzięki czemu otrzymaliśmy wektor 2×1 . Zwróć uwagę, że liczba kolumn macierzy i wierszy wektora musi być taka sama, w przeciwnym razie nie można wykonać operacji mnożenia. Ponadto wartości w wektorze wynikowym są sumami iloczynów macierzy i wektora. Ta sama reguła dotyczy iloczynu dwóch macierzy:

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} A & B \\ C & D \\ E & F \end{bmatrix} = \begin{bmatrix} aA + bC + cE & aB + bD + cF \\ aA + eC + fE & dB + eD + fF \end{bmatrix}$$

Iloczyn macierzy 2×3 przez macierz 3×2 dał nam macierz 2×2 .

Podczas omawiania sieci spłotowych będziemy pracować z tablicami trój-, a nawet czterowymiarowymi. Obiekty takie są nazywane **tensorami** (ang. *tensor*). Jeśli wyobrazisz sobie stos macierzy mających takie same rozmiary, otrzymasz tensor trójwymiarowy; w takim przypadku pierwszy indeks służy do określenia interesującej nas macierzy, a za pomocą dwóch pozostałych wyznaczamy interesujący nas element tejże macierzy. Z kolei jeżeli ułożysz stos trójwymiarowych tensorów,

to otrzymasz tensor czterowymiarowy, w którym pierwszy indeks wyznacza interesujący nas tensor trójwymiarowy.

Główne wnioski z tego podrozdziału są takie, że wektory mają jeden wymiar, macierze mają dwa wymiary, istnieją pewne reguły mnożenia tych obiektów, a także że nasze biblioteki będą korzystać w pewnym momencie z czterowymiarowych tensorów. Jeszcze wrócimy do tych wniosków w dalszej części książki.

Statystyka i prawdopodobieństwo

Działy statystyki i probabilistyki są tak rozległe, że często można jedynie wspomnieć o nich zaledwie kilka słów albo od razu napisać poświęconą im całą książkę czy dwie. Z tego powodu wspomnę tu jedynie o niektórych kluczowych aspektach i pozostawię Ci dokończanie się w tych tematach zależnie od Twoich potrzeb. Zakładam, że na podstawie rzutów monetą i kostką pozyskałeś pewną elementarną wiedzę na temat prawdopodobieństwa.

Statystyka opisowa

Podczas przeprowadzania eksperymentów musimy zgłaszać wyniki w jakiś znaczący sposób. W naszym przypadku oznacza to zazwyczaj określanie wartości średniej plus minus wartość znana jako **błąd standardowy średniej** (ang. *standard error of the mean* — SE). Zdefiniujmy ten błąd na przykładzie.

W przypadku dużej liczby pomiarów x , na przykład długości określonej części kwiatu, możemy obliczyć średnią ($\bar{x}$) przez dodanie wszystkich wartości i podzielenie tej sumy przez liczbę wszystkich dodanych wartości. Po otrzymaniu średniej możemy obliczyć średnią rozpiętość poszczególnych wartości wokół średniej poprzez odjęcie każdej wartości od średniej, podniesienie wyniku do kwadratu, a następnie zsumowanie tak otrzymanych wartości i podzielenie otrzymanej sumy przez liczbę elementów minus jeden. W ten sposób uzyskujemy **wariancję** (ang. *variance*). Pierwiastek kwadratowy z wariancji nazywany jest **odchyleniem standardowym** (σ , ang. *standard deviation*), do którego powrócimy za chwilę. Za pomocą odchylenia standardowego możemy policzyć błąd standardowy jako $SE = \sigma/\sqrt{n}$, gdzie n jest liczbą wartości użytych do obliczenia średniej. Im mniejsza jest wartość SE, tym wartości są ściślej upakowane wokół średniej. Możemy interpretować tę wartość jako niepewność dotyczącą średniej. Zgodnie z nią przewidujemy, że rzeczywista wartość średnia (której nie znamy) mieści się w przedziale między $\bar{x} - SE$ a $\bar{x} + SE$.

Czasami zastępujemy średnią medianą. **Mediana** (ang. *median*) jest wartością środkową, poniżej której znajduje się dokładnie połowa prób. Aby określić medianę danego zestawu wartości, najpierw sortujemy je numerycznie, a następnie wyznaczamy wartość środkową. W przypadku nieparzystej liczby elementów próby wartość tę wyznacza środkowy element, natomiast jeżeli mamy parzystą liczbę elementów, medianą jest wartość średnia między dwoma środkowymi elementami. Mediana czasami okazuje się lepsza od średniej, jeżeli próby nie mają

dobrego, równomiernego rozrzutu wokół średniej. Klasycznym przykładem są dochody. Kilka najbogatszych osób tak przesuwą średnią do góry, że przestaje ona mieć większe znaczenie. Rozsądniejszym rozwiązaniem okazuje się mediana, która ukazuje jedną połowę osób zarabiających mniej i drugą zawierającą ludzi o wyższych dochodach.

W późniejszych rozdziałach będziemy zajmować się **statystyką opisową** (ang. *descriptive statistics*). Są to wartości obliczone dla zestawu danych, pomagające zrozumieć jego własności. Przed chwilą wymienilem trzy takie wartości: średnią, medianę i odchylenie standardowe. Nauczmy się je interpretować i zaznaczać na wykresach w celu lepszego zrozumienia zestawów danych.

Rozkłady prawdopodobieństwa

W tej książce będziemy korzystać z pojęcia **rozkładu prawdopodobieństwa** (ang. *probability distribution*). Można traktować go jako swoistą wyrocznię, która, zapytana, przepowie jakąś liczbę lub zbiór liczb. Na przykład podczas uczenia modelu korzystamy z liczb lub zbiorów liczb, które mierzymy; liczby te biorą się z danego rozkładu prawdopodobieństwa. Jest to tak zwany **rozkład główny** (ang. *parent distribution*). Możemy traktować go jako generator danych dostarczanych do modelu; w bardziej filozoficznym ujęciu jest to idealny zestaw danych, którego przybliżenie stanowią nasze dane.

Istnieje wiele różnych rozkładów prawdopodobieństwa; niektóre z nich mają nawet własne nazwy. Dwoma najpowszechniejszymi są rozkład jednostajny i normalny. Niewątpliwie miałeś już do czynienia z rozkładem jednostajnym: ma on miejsce podczas rzucania kostką do gry. Skoro kostka ma sześć ścian, to wiemy, że prawdopodobieństwo wyrzucenia dowolnej wartości od 1 do 6 jest jednakowe. Jeżeli rzucimy kostką sto razy i spisemy otrzymywane wyniki, to zauważymy, że każdy numer został wyrzucony w przybliżeniu jednakową liczbę razy oraz że na dłuższą metę łatwo uwierzyć, że wartości te się wyrównają.

Rozkład jednostajny (ang. *uniform distribution*) jest wyrocznią, która z jednakowym prawdopodobieństwem może udzielić nam każdej z dopuszczalnych odpowiedzi. W ujęciu matematycznym zapisujemy go jako $U(a, b)$, gdzie litera U oznacza rozkład jednostajny (od ang. *uniform*), natomiast a i b wyznaczają zakres wartości losowanych z tego rozkładu. Dopuszczalne są liczby rzeczywiste, chyba że jawnie wymusimy losowanie wyłącznie liczb całkowitych. Symbolicznie zapisujemy $x \sim U(0, 1)$, co oznacza, że x jest wartością zwracaną przez wyrocznię losującą z jednakowym prawdopodobieństwem dane z zakresu liczb rzeczywistych w przedziale $(0, 1)$. Nawiasy otwarte („(” i „)”) przy oznaczaniu przedziałów oznaczają przedział otwarty, czyli taki, w którym wartości graniczne są wykluczone z przedziału, natomiast nawiasy zamknięte („[” i „]”) wyznaczają przedział domknięty, uwzględniający w przedziale wartość graniczną oznaczoną takim nawiasem. Zatem $U[0, 1)$ zwraca wartości w przedziale od 0 do 1, gdzie wartość 0 jest uwzględniona w przedziale, natomiast wartość 1 zostaje pominięta.

Wykres **rozkładu normalnego** (ang. *normal distribution*), zwanego także **rozkładem Gaussa** (ang. *Gaussian distribution*) przybiera kształt dzwonowaty; w takim

przypadku jedna wartość ma największe prawdopodobieństwo wystąpienia, natomiast prawdopodobieństwo wystąpienia innych wartości maleje wraz z oddalaniem się od tego punktu. Najbardziej prawdopodobną wartością jest średnia $\bar{x}$, natomiast parametrem określającym szybkość redukcji prawdopodobieństwa do zera (które jednak zawsze pozostaje niezerowe) jest odchylenie standardowe σ (sigma). W naszym przypadku jeżeli chcemy losować próbę z rozkładu normalnego, zapiszemy $x \sim N(\bar{x}, \sigma)$, co oznacza, że x jest losowana z rozkładu normalnego o średniej $\bar{x}$ i odchyleniu standardowym σ .

Testy statystyczne

Kolejnym zagadnieniem, które będzie pojawiać się od czasu do czasu, jest koncepcja **testu statystycznego** (ang. *statistical test*), czyli pomiaru służącego do określania prawdziwości danej hipotezy. Zazwyczaj hipoteza odnosi się do zbiorów pomiarów; mianowicie hipoteza zakłada, że dwa zbiory pomiarów wywodzą się z tego samego rozkładu głównego. Jeżeli statystyka obliczona na podstawie testu wykracza poza pewien określony zakres, to odrzucamy hipotezę i stwierdzamy, że mamy dowód na pochodzenie obydwu zestawów danych z dwóch *różnych* rozkładów.

Tutaj będziemy zazwyczaj używać **testu t**, powszechnego testu statystycznego, w którym przyjmujemy rozkład normalny danych. Skoro zakładamy, że rozkład danych jest normalny, co może, ale nie musi być prawdą, to test ten nazywamy **testem parametrycznym** (ang. *parametric test*).

Czasami będziemy korzystać z **testu U Manna-Whitneya**, który przypomina test t, gdyż pomaga określić, czy dane dwie próby pochodzą z tego samego rozkładu głównego, nie zakładamy tu jednak odgórnie określonego rozkładu danych. Jest to przykład **testu nieparametrycznego** (ang. *nonparametric test*).

Bez względu na to, czy test jest parametryczny, czy nieparametryczny, zawsze otrzymujemy ostatecznie tak zwaną **wartość p** (ang. *p-value*). Stanowi ona prawdopodobieństwo wystąpienia obliczonej wartości testu w przypadku, gdyby hipoteza o pochodzeniu prób z tego samego rozkładu głównego była prawdziwa. Mała wartość p stanowi dowód na nieprawdziwość hipotezy.

Zazwyczaj granicą wartości p jest 0,05, co oznacza, że w 1 na 20 przypadków możemy zmierzyć daną wartość testu statystycznego (testu t lub U Manna-Whitneya), nawet gdyby próby pochodziły z tego samego rozkładu głównego. Ostatnie lata udowodniły jednak, że taki próg jest zbyt łagodny. Gdy wartość p jest bliska granicy 0,05, ale jej nie przekracza, to zaczynamy uznawać, że może istnieć jakiś dowód zaprzeczający hipotezie. Jeżeli wartość p wynosi, powiedzmy, 0,001 lub nawet mniej, to uzyskujemy mocny dowód na pochodzenie prób z dwóch różnych rozkładów. W takim przypadku mówimy, że różnica jest **statystycznie istotna/znacząca** (ang. *statistically significant*).

Procesory graficzne (GPU)

Jedną z technologii umożliwiających rozwój współczesnego uczenia głębokiego są potężne **procesory graficzne** (ang. *graphics processing unit* — *GPU*). Są to miniaturowe komputery stanowiące część karty graficznej. Pierwotnie zostały zaprojektowane z myślą o grach komputerowych, ale potężne możliwości równoległego przetwarzania operacji zostały dostosowane do skrajnych wymogów obliczeniowych modeli głębokich sieci neuronowych. Mnóstwo postępów z ostatnich lat nie byłoby możliwych bez niesamowitych możliwości obliczeniowych układów GPU, które są dostępne nawet na prostych komputerach stacjonarnych. Liderem w tworzeniu układów GPU przeznaczonych do uczenia głębokiego jest firma NVIDIA, a dzięki technologii **CUDA** (ang. *Compute Unified Device Architecture*) odegrała kluczową rolę w sukcesie uczenia głębokiego. Nie byłoby przesadą stwierdzenie, że bez układów GPU uczenie głębokie w ogóle nie miałyby miejsca, a z pewnością nie byłoby tak rozpowszechnione.

Skoro o tym mowa, nie oczekujemy, że w omawianych modelach muszą być wykorzystywane układy GPU. Będziemy używać na tyle małych zestawów danych i modeli, że do ich uczenia w rozsądnym krótkim czasie wystarczy zwyczajny procesor komputera. Wymusiliśmy zresztą ten wybór na etapie instalowania pakietów, ponieważ wybraliśmy wersję biblioteki TensorFlow działającą jedynie na zwykłym procesorze.

Jeżeli dysponujesz kartą graficzną wyposażoną w architekturę CUDA i chcesz skorzystać z niej podczas pracy nad przykładami opisującymi uczenie głębokie, to możesz to zrobić, jeśli jednak nie masz takiego układu graficznego, nie planuj jego zakupu wyłącznie w tym celu. Jeśli korzystasz z układu GPU, przed instalacją pakietów zainstaluj najpierw prawidłowo architekturę CUDA, a następnie bibliotekę TensorFlow obsługującą karty graficzne. Biblioteka scikit-learn korzysta wyłącznie ze zwykłego procesora.

Podsumowanie

W niniejszym rozdziale zapoznaliśmy się z naszym środowiskiem operacyjnym. Następnie omówiliśmy kluczowe biblioteki Pythona, z których będziemy korzystać w dalszej części książki, a także przygotowaliśmy instrukcję instalacji tych narzędzi przy założeniu, że będziesz używać dystrybucji Ubuntu 20.04. Jak już wspomnieliśmy, narzędzia te działają równie dobrze w innych dystrybucjach Linuksa oraz systemach operacyjnych. Przeszliśmy dalej do krótkiego omówienia aparatu matematycznego, który będzie nam później potrzebny, a rozdział zakończyliśmy opisem znaczenia układów GPU oraz wyjaśnieniem, dlaczego nie są wymagane podczas pracy na omawianych tu modelach.

Następny rozdział został poświęcony podstawom Pythona.

Skorowidz

A

aktywacja/pobudzenie, 346
 warstwy gęstej, 349
algebra liniowa, 27
 macierz, 28
 mnożenie
 macierzy, 29
 macierzy przez wektor, 29
 wektorów, 28
tensor, 29
wektor, 27
 iloczyn wewnętrzny, 29
 iloczyn zewnętrzny, 29
 kolumnowy, 28
 wierszowy, 27
algorytm
 Adadelta, 356, 369
 Adagrad, 356
 budowania drzew
 decyzyjnych, 154
 Glorota, 279
 gradientu prostego, 225, 229
 aktualizowanie wag, 229
 znajdowanie minimów,
 227
 inicjalizacji wag, 445
 k najbliższych sąsiadów,
 k-NN, 144
 hiperparametr, 179
 wady, 200
 zalety, 199

 najbliższego centroidu, 140
 wady, 199
 zalety, 199
 propagacji wstecznej, 235
 rozwiązujący, solver, 260
 SGD, 230–235, 391
 aktualizowanie
 współczynnika uczenia,
 234
 funkcje niewypukłe, 232
 funkcje wypukłe, 232
 grupy, 230
 liczba etapów, 269
 minigrupy, 230
 momentum, 235
 z propagacją wsteczną,
 236
 sortowania bąbelkowego,
 191
 tworzenia drzewa
 decyzyjnego, 156
 wizualizacji t-SNE, 349
algorytmny
 rekurencyjne, 153
 zachłanne, 155
analiza
 czasu działania, 192
 danych, 106
 elementy odstające, 107
 wyszukiwanie
 problemów, 107
 złe oznakowanie, 107

 głównych składowych, PCA,
 127, 128, 190, 194
 zestaw danych MNIST,
 195
AUC, area under curve, 311, 398

B

biblioteka, *Patrz także* moduł
 h5py, 27
 Keras, 25, 119, 351, 406
 librosa, 424
 math, 52
 matplotlib, 27
 NumPy, 25, 54–78
 pillow, PIL, 27, 76
 scikit-learn, 25
 sklearn, 25, 76, 115, 163, 409
 TensorFlow, 25
blok try-except, 48
błąd, error, 241, 242
 bezwzględny, 244
 bieżącej warstwy, 243
 dla danych
 niepotasowanych, 384
 potasowanych, 384
 walidacyjnych, 359
 dla zbioru uczącego, 359
 dla zestawu MNIST, 365
 kwadratowy, 238
 model głęboki, 394
 model płytki, 394
 parametr α , 277
 parametr μ , 279

błąd, error
 różne schematy inicjalizacji, 283
 standardowy, 58, 108
 standardowy średniej, SE, 30
 średniokwadratowy, 244
 tasowanie pikseli, 286
 Breast Cancer, 117, 168,
Patrz zestaw danych
 budowanie
 drzew decyzyjnych, 154
 etykiet, 401
 map cieplnych, 379
 modelu CIFAR-10
 głębokiego, 389
 płytkiego, 389
 modelu MNIST, 354
 sieci spłotowych, 352
 spektrogramów, 441
 zestawu danych, 114
 CIFAR-10, 123
 ESC-10, 422
 MNIST, 120
 prostego, 217
 z obrazami, 413

C

cechy, feature, 20, 80
 brakujące, 97
 dobór, feature selection, 83
 kolejność, 284
 centroidy klas, 143
 CIFAR-10, 122, 386, *Patrz*
 zestaw danych
 CNN, convolutional neural
 network, 323
 CSV, comma-separated
 values, 73
 CUDA, Compute Unified
 Device Architecture, 33
 czas uniksowy, epoch time, 51
 czułość, sensitivity, recall, hit
 rate, *Patrz* wskaźnik TPR,
 294, 397
 czystość, purity, 155

D

dane
 testowe, test data, 98, 172
 uczące, training data, 98, 172
 walidacyjne, validation data,
 98
 wejściowe, input, 86
 wektorowe, 203
 definiowanie
 funkcji, 49
 tablicy, 59
 typu danych, 59
 dobór cech, feature selection, 83
 dokładność, accuracy, 288, 367
 ważona, 318
 dopasowanie krzywej, curve
 fitting, 248
 drzewo
 decyzyjne, decision tree,
 150
 algorytm, 154
 wady, 201
 zalety, 201
 zestaw danych Iris, 152
 kd, k-d tree, 194
 metryczne, ball tree, 194
 uczenia maszynowego, 21
 dynamiczny język
 programowania, 37

E

efektywne pole recepcyjne,
 effective receptive field, 339
 ekstrapolacja, extrapolation, 86
 entropia krzyżowa, 245
 epoka, epoch, 230, 268
 liczba, 365
 ESC-10, 422, *Patrz* zestaw
 danych
 etykieta, label, 79, 80

F

filtr, 334
 Lanczosa, 337
 format pliku, *Patrz* pliki
 funkcja, 49
 arrange, 65, 67
 argsort, 102
 array, 59, 60
 confusion_matrix, 315
 ctime, 52
 dot, 71, 72
 enumerate, 45
 epoch, 276
 fit, 276
 load, 74
 loadtxt, 73
 make_classification, 100
 ones, 62
 pitch_shift, 427
 random.normal, 76
 random.random, 76
 random.seed, 76
 range, 44
 save, 74
 savetxt, 74
 savez, 74
 savez_compressed, 75
 score, 276
 shape, 61
 sqrt, 52
 time, 57
 zeros, 62
 funkcje
 aktywacji, activation
 function, 208, 241, 259,
 261, 263
 h, 216
 liniowa, linear function, 208
 logistyczna, *Patrz*
 sigmoidalna
 ReLU, 210, 212, 242, 264
 sigmoidalna, sigmoid,
 210, 211, 264
 tangensa hiperbolicznego,
 hyperbolic tangent,
 210, 211, 264

przestępne,
 transcendental
 function, 209
 trygonometryczne,
 trigonometric function,
 209
 niewypukłe, 232
 straty, loss function, 225,
 238, 243, 245, 392
 błąd bezwzględny, 244
 błąd średniokwadratowy,
 244
 dla zbioru uczącego, 439,
 446
 dla zbioru walidacyjnego,
 439, 446
 entropia krzyżowa, 245
 znajdowanie minimum,
 227
 wielomianowe, 249
 wypukłe, 232

G

GAN, generative adversarial
 network, 458
 generowanie
 liczb pseudolosowych, 46, 75
 sztucznego zestawu danych,
 102
 wartości macierzy pomyłek,
 292
 Glorot Xavier, 247, 279
 główne składowe, principal
 component, 128
 gradient prosty, gradient
 descent, 225
 graf, graph, 206
 grupa, 230

H

h5py, 27
 He Kaiming, 247

hiperparametr,
 hyperparameter, 160, 179
 hiperpłaszczyzna, hyperplane,
 157

I

iloczyn
 Hadamarda, Hadamard
 product, 242
 skalarny, dot product, 29, 71
 wewnętrzny, inner product,
 29, 160
 zewnętrzny, outer product,
 29
 indeksowanie tablicowe, array
 indexing, 64
 informatywność, informedness,
 300
 inicjalizowanie wag, 246, 281
 Glorota, Glorot
 initialization, 247
 He, 247
 instrukcja
 break, 47
 continue, 47
 if-elif-else, 43
 with, 48
 interpolacja, interpolation, 86
 interpreter Pythona, 34
 Iris, 114, 162, *Patrz* zestaw
 danych

J

jądro, kernel, 161, 325
 Gaussa, Gaussian kernel,
 161, 182
 liniowe, 161
 RBF, *Patrz* Gaussa
 rezultat użycia, 338
 skanowanie, 325
 jednostajny główny rozkład
 prawdopodobieństwa, 89

K

kanal alfa, 76
 Keras, 25, 119, 351, 406
 moduł mnist, 353
 obliczanie wskaźników, 396
 optymalizatory, 369
 sieci splotowe, 352
 strojenie modelu, 416
 usuwanie warstwy, 417
 zestaw danych
 CIFAR-10, 122
 MNIST, 119
 k-krotny sprawdzian krzyżowy,
 105, 106
 implementacja, 174
 zestaw Breast Cancer, 176
 klasa, class, 79
 DecisionTreeClassifier, 152
 LinearSVC, 186, 191
 MLPClassifier, 234, 247,
 258, 261
 konstruktor, 258, 259
 obiekt, 268
 RandomForestClassifier,
 180
 SVC, 166, 186, 191, 192
 SVM, 160
 klasyfikacja, classification, 79
 klasyfikator
 binarny, 291
 k-NN, 144, 145
 strojenie, 179
 najbliższego centroidu, 143
 implementacja, 167
 zestaw danych Iris, 167
 najbliższego sąsiada, 144
 typu jeden przeciw reszcie,
 one versus rest, 400
 wieloklasowy, 404
 klasyfikowanie
 cech dźwiękowych
 modele klasyczne, 430
 sieć CNN, 434
 sieć neuronowa, 432
 spektrogramów, 443

kłątwa wymiarowości, curse of dimensionality, 83, 84
kodowanie gorącojedynkowe, one-hot encoding, 83
kopiowanie listy, 40
korzeń, root, 153
krawędź, edge, 206
krotka, tuple, 42
krzywa
charakterystyki operacyjnej odbiornika, ROC, 304, 398
analiza, 307
generowanie, 312
porównywanie modeli, 310
wskaźnik AUC, 311
precyzji-czułości, PR, 314

L

las losowy, random forest, 155, 156
strojenie, 180
wady, 201
zalety, 201
librosa, 424
liczby
całkowite, integer, 37, 81
losowe, random number, 75
zmiennoprzecinkowe, floating-point number, 37, 81
lista, list, 38, 56
dodawanie elementów, 39
kopiowanie, 40
zwracanie elementów, 40
liść, leaf node, 152
losowanie prób, 102
LSTM, long short-term memory, 459

Ł

łańcuch znaków, string, 38

M

macierz, matrix, 28, 56, 71
mnożenie, 28, 71, 242
mnożenie przez wektor, 29
transpozycja, 242
macierze
dokładności, accuracy matrix, 317
pomylek
 10×10 , 315
 2×2 , 290
analiza, 448
generowanie wartości, 292
modelu lasu losowego, 410
przypadki wieloklasowe, 315
rozszerzanie, 315
wskaźniki, 293, 294
wykrywanie
patologicznych modeli, 317
wyniki, 293, 452, 453
wyświetlanie, 403
mapa
cech, feature map, 331
ciepłna, heatmap, 379, 382
margines, margin, 158
maksymalny, 159
współczynnik tolerancji, 160
maszyna wektorów nośnych, SVM, 157, 159
hiperparametr, 160
jądra, 160, 183
marginesy, 157
ocena uczenia, 184
optymalizacja, 160
strojenie, 181
wady, 202
zalety, 202
zestaw danych Breast Cancer, 185
math, 52
matplotlib, 27

MCC, Matthews correlation coefficient, 302
mediana, median, 30
metoda
append, 39
keys, 42
show, 77
minigrupa, minibatch, 230, 231
rozmiar, 365
minimum
funkcji straty, 227
lokalne, local minimum, 228
MNIST, 119, 185, 189, 192, 351, *Patrz* zestaw danych
mnożenie
macierzy, 28, 71, 242
wektorów, 28
model, 20
binarny, 400
CIFAR-10
analizowanie modeli, 392
budowanie etykiet, 401
dokładność, 391
etykiety klas, 399
głęboki, deep model, 387, 389
obliczanie wskaźników, 396
płytki, shallow model, 387, 389
rozdzielanie kategorii, 395
testowanie, 390
uczenie, 390
klasyczny, 139
algorytm k najbliższych sąsiadów, k-NN, 144, 199
algorytm najbliższego centroidu, 140, 199
drzewo decyzyjne, 150, 200
las losowy, 155, 201
maszyna wektorów nośnych, SVM, 157, 202
naiwny klasyfikator Bayesa, 146, 200

- klasyfikujący obrazy, 417
- MNIST, 354
 - modyfikowanie architektury, 362
 - ocena, 356
 - uczenie, 356
 - wynik uczenia, 357
- wieloklasowy, 400
- modele, 20
 - analiza czasu działania, 192
 - cechy, 80
 - brakujące, 97
 - dobór, 83
 - czas uczenia, 368
 - dokładność, 195
 - maksymalna, 197
 - dostosowanie do strojenia, 415
 - etykiety, 79
 - interpolacja, 86
 - k-krotny sprawdzian krzyżowy, 174
 - klasy, 79
 - klasyczne, 139
 - dane wektorowe, 203
 - ograniczona moc obliczeniowa, 203
 - testowanie, 163
 - używanie, 202
 - wektory osadzeń, 408
 - zestaw danych Breast Cancer, 170–172, 177, 180, 181
 - zestaw danych MNIST, 186, 187, 356
 - kłątwa wymiarowości, 83
 - mylące przykłady danych, 91
 - ocena uczenia, 170–172, 177, 180, 181
 - parametry, 86
 - pojemność, 91
 - przygotowanie zestawu danych, 79
 - rozmiar zestawu danych, 91
 - strojenie, 411
 - testowanie, 163, 375, 418

- trenowanie, 81, 86, 105, 412, 417
- uczenia maszynowego, 139–161
- uczenie
 - nadzorowane, 20
 - nienadzorowane, 20
- wektor cech, 80, 92
- zestaw danych, 86
 - Breast Cancer, 169, 172, 174
 - Iris, 163
 - MNIST, 189, 198
- modulo, 46
- moduł, module, 51
 - copy, 41
 - decomposition, 129
 - metrics, 314, 315
 - mnist, 353
 - pdb, 49
 - pickle, 220
 - random, 75
 - time, 51
 - wavfile, 424
- momentum, 235, 278

N

- nacechowanie, markedness, 300
- nadmierne dopasowania, overfitting, 156
- naiwny klasyfikator Bayesa, 146
 - wady, 200
 - zalety, 200
- najbliższy
 - centroid, nearest centroid, 143
 - sąsiad, nearest neighbor, 144
- neuron, 207
- niezmienniczość przestrzenna, spatial invariance, 324
- normalizacja, normalizing, 95
 - wsadowa, 445, 447
 - zestawu danych, 169

- notacja dużego O, 191
- NPV, negative predictive value, 295
- NumPy, 25, 54–78
 - liczby losowe, 75
 - obrazy, 76
 - operatory, 69
 - rozgłaszanie, 69
 - szybkość biblioteki, 56
 - tablice, 59
 - typy danych, 60
 - wykrawanie wycinków, 68
 - zestaw danych MNIST, 120

O

- obciążenie, bias, 208, 216
- obiekt PIL, 77, 137
- obliczanie pochodnej cząstkowej, 238
- obraz, 76
 - odbicie lustrzane, 135
 - odwrócenie, 134
 - przycięcie symetryczne, 135
 - rotacja, 134
- obrazy testowe, 374
- obsługa błędów, 48
- odchylenie standardowe, standard deviation, 30, 58, 169, 246
 - jednostkowe, 94
- odległość, distance, 142
 - euklidesowa, Euclidean distance, 142, 252
- odsetek
 - falszywie negatywnych, FNR, 397
 - falszywie pozytywnych, FPR, 397
- operacja
 - łączenia, 341
 - maksymalizująca, max pooling, 341
 - softmax, 214
 - splotu, 325
 - symbol *, 335

- operacje
 - macierzowe, 71
 - na wektorach, 71
- operator, 69
 - dzielenia całkowitego, 70
 - T, 242
- optymalizator, 369
 - czas uczenia, 370
 - dokładność, 370
- osadzenie, embedding, 405

P

- PCA, principal component analysis, 127
- perceptron wielowarstwowy, multi-layer perceptron, MLP, 208, *Patrz sieć neuronowa*
- pętla
 - for, 44
 - while, 46
 - z oddolnym sprawdzaniem, 46
 - z odgórnym sprawdzaniem, 46
- pillow, PIL, 27, 76
- plik
 - bc_experiments.py, 169, 292
 - bc_kfold.py, 174, 176
 - bc_rbf_svm_search.py, 183
 - box_plot.py, 109
 - cifar10_cnn_SGD.py, 388
 - cifar10_augment.py, 135
 - cifar10_cnn.py, 388
 - cifar10_cnn_cat_dog_fine_
 - ↳ tune_3.py, 416
 - cifar10_cnn_model.h5, 406
 - cifar10_cnn_vehicles.py, 414
 - cifar10_cnn_vehicles_
 - ↳ model.h5, 415
 - feature_stats.py, 109
 - iris_centroids.py, 167
 - iris_data_augmentation.py, 130
 - iris_experiments.py, 163
 - make_shifted_mnist_
 - ↳ dataset.py, 381
 - mnist_2x2_tables.py, 297
 - mnist_even_odd.py, 306
 - mnist_experiments.py, 186, 187, 297
 - mnist_experiments_base_
 - ↳ lr.py, 272
 - mnist_nn_experiments.py, 259, 261
 - mnist_nn_experiments_
 - ↳ init.py, 280
 - mnist_nn_experiments_
 - ↳ L2.py, 276
 - mnist_nn_experiments_
 - ↳ momentum.py, 278
 - mnist_nn_experiments_
 - ↳ samples.py, 274
 - mnist_pca.py, 195
 - multilayer_perceptron.py, 267
 - nn_iris_dataset.py, 217
 - nn_iris_evaluate.py, 219
 - nn_iris_mlpclassifier.py, 221
 - numpy_speed_test.py, 56
 - transfer_learning.py, 406
- pliki
 - .csv, 73, 74
 - .npy, 74
 - dźwiękowe, 423, 427
 - .wav, 421, 425
 - graficzne, 77
 - .jpeg, 76
 - .png, 76
 - HDF5, 27, 357
 - plaszczyna, plane, 157
 - maksymalnego marginesu, 160
 - pochodna, derivative, 227
 - cząstkowa, partial derivative, 237
 - obliczanie, 238
 - funkcji ReLU, 242
 - podział danych, 172
 - pojemność, capacity, 91
 - pole recepcyjne, receptive field, 339
 - porzucanie, dropout, 253
 - PPV, positive predictive value, 295
 - PR, precision-recall curve, 314
 - prawdopodobieństwo, *Patrz także* rozkład prawdopodobieństwa rotacji obrazu, 137
 - wystąpienia cechy, 149
 - precyzja, precision, 295, 397
 - procesory graficzne, graphics processing unit, GPU, 33
 - programowanie kwadratowe, quadratic programming, 160
 - propagacja wsteczna, backpropagation, 225, 235, 236, 240
 - próba, sample, 28
 - samowsporna, bootstrap sample, 156
 - przekształcenia
 - obrazu, 134
 - obrazu PIL w tablicę, 137
 - tablicy NumPy w obiekt, 137
 - przestrzeń
 - cech, 141, 142, 143
 - nazw, namespace, 51
 - przesunięcie czasowe, time shifting, 424
 - przetrenowanie, 156, 248
 - przykłady mylące, confuser, 91
 - punkt siodłowy, saddle point, 233
 - Python, 34, *Patrz także* NumPy
 - białe znaki, 35
 - funkcje, 49
 - idiomy, 137
 - instrukcje, 35
 - struktury
 - danych, 36
 - sterowania, 43
 - zmienne, 36

R

radialna funkcja bazowa, RBF, 161, 182
 regresja liniowa, linear regression, 87
 regularyzacja, regularization, 126, 225, 248, 251, 445
 L2, L2 regularization, 252, 275
 reguła łańcuchowa, 237, 238
 dla pochodnych, 236
 rekurencja, recursion, 153
 RGB, red, green, blue, 76
 RNN, recurrent neural networks, 459
 ROC, receiver operating characteristics curve, 304
 rozgłaszanie, broadcasting, 69
 rozkład prawdopodobieństwa, probability distribution, 31
 a posteriori, 146
 a priori, 89, 146
 Gausa, Gaussian distribution, 31
 główny, parent distribution, 31, 89
 jednostajny, uniform distribution, 31, 89, 246
 normalny, normal distribution, 31
 wspólny, joint probability, 148
 rozmiar
 grupy, 267
 minigrupy, 365
 zbioru uczącego, 365
 rozstęp ćwiartkowy, 111
 rozszerzanie/dogenerowanie danych, data augmentation, 124
 analiza
 głównych składowych, 130
 zestaw CIFAR-10, 133
 zestaw Iris, 127
 równanie liniowe, 226

S

scikit-learn, 25, *Patrz* sklearn
 segmentacja semantyczna, semantic segmentation, 346
 SGD, stochastic gradient descent, 230
 sieć neuronowa
 aktualizowanie współczynnika uczenia, 234
 algorytm gradientu prostego, 225, 229
 algorytm rozwiązujący, 260
 architektura, 212, 259
 funkcje aktywacji, 207, 216, 259
 implementacja, 217
 neuron, 207
 obciążenie, bias, 208, 216
 ocena uczenia w zależności od funkcji aktywacji, 264
 liczby parametrów, 266
 rozmiaru minigrupy, 270
 rozmiaru zbioru uczącego, 275
 współczynnika uczenia, 273
 pełna, 213
 przetrenowanie, 248
 regularyzacja, 251
 softmax, 214
 testowanie, 221
 uczenie, 221, 224–56
 aktualizowanie wag, 229
 algorytm gradientu prostego, 225
 algorytm SGD, 230
 dobór wartości początkowych, 229
 epoka, 230
 funkcje aktywacji, 259, 261
 funkcje straty, 243

inicjalizowanie wag, 246, 279
 kończenie uczenia, 233
 klasa MLPClassifier, 258
 kolejność cech, 284
 porzucanie, 253, 254
 propagacja wsteczna, 235, 236
 przetrenowanie, 248
 regularyzacja, 248
 regularyzacja L2, 275
 rozmiar grupy, 267
 rozmiar zbioru uczącego, 274
 struktura, 259
 wsadowe, 230, 231
 współczynnik uczenia, 271
 wyniki, 263
 znajdowanie minimów, 227
 wagi, 208, 216
 warstwa wyjściowa, 214
 warstwy ukryte, 216
 węzeł, 207, 208
 zestaw danych, 217
 Iris, 219, 221
 MNIST, 257
 sieć neuronowa spłotowa, CNN, 323
 aktywacja/pobudzenie, 346
 architektura, 329
 budowanie modelu MNIST, 354
 model
 binarny, 400
 głęboki, 389
 płytki, 389
 wieloklasowy, 400
 niezmienniczość przestrzenna, 324
 operacja spłotu, 325
 jądro, 325
 krok, 325
 przetwarzanie obrazów, 328
 tryb zerowy, 327
 uzupełnianie zerami, 327

sieć neuronowa spłotowa,
 CNN
 segmentacja semantyczna,
 346
 strojenie, 411
 tasowanie danych, 383
 uczenie transferowe, 405
 w pełni spłotowa, 345, 346
 mapa cieplna, 378, 382
 testowanie, 374, 375
 trenowanie, 371
 zestaw MNIST, 371
 warstwy
 gęste, 331
 łączące, 331, 340
 ReLU, 330
 spłotowe, 331, 333, 336,
 340
 spłotowe pełne, 344
 spłotowe wielokrotne,
 339
 spłaszczające, 331
 tablice
 czterowymiarowe, 331
 w pełni połączone, 343
 wczytywanie danych, 352
 wyznaczanie klas pojazdów
 i zwierząt, 395
 zestaw danych CIFAR-10,
 386
 sieci
 generatywne sieci
 przeciwstawne, GAN, 458
 rekurencyjne sieci
 neuronowe, RNN, 459
 skalar, 29
 sklearn, 25, 76, 115, 163, 409
 drzewo decyzyjne, 152
 ostrzeżenia, 263
 wizualizacja, 349
 współczynnik tolerancji, 160
 zestaw danych
 Breast Cancer, 117
 Iris, 114
 słownik, dictionary, 42
 softmax, 214

spektrogram dźwięku, 440
 spłot, convolution, 325
 jądra z obrazem, 325
 standaryzacja, standardization,
 95
 statystyka opisowa, descriptive
 statistics, 30, 31
 statystyki, 109
 stochastyczny spadek wzdłuż
 gradientu, *Patrz* algorytm
 SGD, 230
 strategia
 jeden na jednego, 192
 jeden na resztę, 192
 strojenie, fine-tuning, 412
 klasyfikatora k-NN, 179
 lasu losowego, 180
 maszyny SVM, 181
 modelu, 416
 struktura
 danych
 krotka, 42
 lista, 38
 słownik, 42
 tablica, 55
 sieci neutronowej, 212, 259
 sterowania, control
 structures, 43
 studium przypadku, 421
 klasyfikowanie cech
 dźwiękowych, 430
 modele klasyczne, 430
 sieć CNN, 434
 sieć neuronowa, 432
 klasyfikowanie
 spektrogramów, 443
 spektrogramy, 439
 styczna, tangent, 227
 SVM, support vector machine,
 157
 swoistość, specificity, *Patrz*
 wskaźnik TNR, 294, 397
 sygnały, 208
 symbol
 sumy, 148
 iloczynu, 148
 sztuczna inteligencja, SI, 20

Ś

średnia, 30, 94
 harmoniczna, harmonic
 mean, 301
 środkowanie wokół średniej,
 mean centering, 94

T

tablica, array, 55, 58
 dane wejściowe, 72
 dane wyjściowe, 72
 definiowanie, 59
 indeksowanie, 64
 trójwymiarowa, 61
 użycie wielokropka, 68
 wykrawanie wycinków, 65,
 68
 technologia CUDA, 33
 tensor, tensor, 29
 TensorFlow, 25
 test
 parametryczny, parametric
 test, 32
 nieparametryczny,
 nonparametric test, 32
 statystyczny, statistical test,
 32
 U Manna-Whitneya, 32
 testowanie
 modeli klasycznych, 163
 modelu, 418
 TNR, true negative rate, 294,
 397
 TPR, true positive rate, 294,
 397
 transformata Fouriera, Fourier
 transform, 439
 transpozycja, 160, 242
 t-SNE, 349
 twierdzenie Bayesa, 146
 typy danych NumPy, 60

U

Ubuntu, 26

uczenie

bezprzykładowe, zero-shot learning, 202

głębokie, deep learning, 14, 20

kilkuprzykładowe, few-shot learning, 202

kompleksowe, end-to-end training, 349

maszynowe, 20, 139–61

algorytm k najbliższych sąsiadów, k-NN, 144

algorytm najbliższego centroidu, 140

drzewa decyzyjne, 150

lasy losowe, 155

maszyny wektorów

nośnych, SVM, 157

naiwny klasyfikator

Bayesa, 146

nadzorowane, supervised learning, 20, 86

nienadzorowane, unsupervised learning, 20, 458

przez wzmacnianie, reinforcement learning, 20, 458

transferowe, transfer learning, 202, 405

wsadowe, batch training, 230, 231

W

waga, weight, 208, 216

aktualizowanie, 229

inicjalizacja, 279

Glorota, 247

He, 247

schematy, 281

zanik, 252

wariancja, variance, 30

warstwa

gęsta, 331, 349

łącząca, 331, 340

ReLU, 330

splotowa, 331, 333, 336, 340

inicjalizacja, 340

poła recepcyjne, 339

pełna, 344

wielokrotna, 339

splaszczająca, 331

w pełni połączona, 343

wyjściowa, 214

warstwy

ukryte, 216

usuwanie, 417

wartości

dyskretne, 81

kategoryzujące, 82

logiczne, 40

porządkowe, 82

przedziałowe, 81

wartość

None, 40

NULL, 40

p, p-value, 32

wektor, 27, 56, 70

cech, feature vector, 81, 84, 92

cech dźwięku, 429

wierszowy, row vector, 27, 161

kolumnowy, column vector, 28

nośny, 159

osadzeń, 408

wektory

dodawanie, 216

iloczyn

skalarny, 71

wewnętrzny, 29

zewewnętrzny, 29

mnożenie, 28

węzeł, node, 206, 207

sygnał wejściowy, 208

sygnał wyjściowy, 208

wagi, weight, 208

wiarygodność, likelihood, 146
widmo mocy, power spectrum, 439

wielokropek, 69

wielomian, 249

wskaźnik, metric, 287

AUC, area under curve, 311

F1, F1 score, 300

Giniego, Gini index, 155

informatywność,

informedness, 300

nacechowanie, markedness, 300

odsetek prawdziwie

negatywnych, TNR, 294, 397

odsetek prawdziwie

pozytywnych, TPR, 294, 397

wartość predykcyjna

dodatnia, PPV, 295

wartość predykcyjna

ujemna, NPV, 295

współczynnik kappa

Cohena, 301

współczynnik korelacji

Matthewsa, MCC, 302

wieloklasowy, 320

współczynnik uczenia, 234, 271

aktualizowanie, 234

wskaźniki

interpretowanie modeli, 297

obliczanie, 396

wykresy, 306, 310

wyniki, 298

zaawansowane, 299

implementacja, 302

wyniki, 303

wycinek tablicy, array slice, 63

wygładzanie Laplace'a, Laplace smoothing, 150

wykrawanie wycinków, slicing, 65, 68

wykres
 analiza PCA, 128
 dwuwymiarowa przestrzeń
 cech, 141–143
 funkcja błędu, 358
 pudełkowy, box plot, 109,
 110, 116
 punktowy cech, 127
 t-SNE, 408
 wyrażenie listowe, list
 comprehension, 46

Z

zanik wagi, weight decay, 252
 zbiór uczący, training set, 20
 rozmiar, 274, 365
 zdarzenia niezależne, 148
 zespół, ensemble, 156, 449
 zestaw cech, feature, 20, 28
 zestaw danych, dataset, 86
 Breast Cancer, 117, 168
 k-krotny sprawdzian
 krzyżowy, 174
 macierz pomyłek 2×2 ,
 293, 296
 oceny modeli, 170–172,
 177
 randomizacja, 172

cechy, 80
 CIFAR-10, 122, 386
 przygotowanie, 388
 rozszerzanie zestawu,
 133, 135, 412
 ESC-10, 422
 budowanie, 422
 rozszerzanie, 423–426
 wstępne przetwarzanie,
 427
 zmniejszanie liczby
 próbek, 428
 etykiety, 79
 Iris, 114, 162
 analiza PCA, 128
 drzewo decyzyjne, 152
 klasyfikator najbliższego
 centroidu, 167
 modele klasyczne, 163
 rozszerzanie zestawu,
 127, 130
 sieć neuronowa, 221
 klasy, 79
 MNIST, 119, 185, 189, 192,
 351
 analiza czasu działania,
 192
 analiza PCA, 195
 dokładność modelu, 197
 krzywe ROC, 311
 oceny modeli, 189, 198

pełne sieci splotowe, 371
 przetwarzanie wstępne,
 189
 sieć neuronowa
 tasowanie, 197, 198, 383
 tworzenie obrazów
 testowych, 374
 uczenie klasycznych
 modeli, 185–187
 wskaźniki
 zaawansowane, 303
 znormalizowany, 186
 podział, 99
 rozmiar, 91
 skalowanie cech, 92
 wektory cech, 80
 własności, 86
 znormalizowany, 168, 186
 zmiana
 stosunku podziału danych,
 172
 tonacji, pitch shifting, 424
 zmienne, variable, 37
 znak
 *, 335
 procentu, 46
 tabulatora, \t, 73
 wielokropka, 68

PROGRAM PARTNERSKI

— GRUPY HELION —

- 
- A top-down view of four hands holding four interlocking puzzle pieces. The top-left piece is red, the top-right is dark blue, the bottom-left is light blue, and the bottom-right is a slightly darker shade of light blue. The hands are positioned around the pieces, with fingers visible holding them. The background is a solid grey.
1. ZAREJESTRUJ SIĘ
 2. PREZENTUJ KSIĄŻKI
 3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

UCZENIE
GŁĘBOKIE:
PRZYSZEDŁ CZAS
NA TWÓJ PIERWSZY
MODEL!



Uczenie głębokie fascynuje wielu inżynierów i praktyków. Mimo że systemy oparte na uczeniu maszynowym stosuje się w rozlicznych branżach, wciąż są uważane za niepokojącą technologię. Istotnie, w wypadku na przykład sieci neuronowych nie wiemy, czego dokładnie uczy się model. Możemy tylko ocenić, czy dobrze realizuje swoje zadanie. Wydaje się, że w sposobie pracy algorytmów uczenia głębokiego tkwi magia. Właśnie dlatego dobrze jest zająć się faktami i dowiedzieć się, na czym w rzeczywistości polega uczenie maszynowe, a zwłaszcza uczenie głębokie.

Ta książka jest przystępnym przewodnikiem po uczeniu maszynowym. Aby zrozumieć zawartą w niej treść, wystarczy podstawowa umiejętność programowania i znajomość matematyki na poziomie szkoły średniej. Znalazło się tu omówienie podstawowych pojęć i wyjaśnienie mechanizmów rządzących uczeniem głębokim. Dzięki lekturze dowiesz się, czym się charakteryzuje dobry zbiór danych uczących, jak ocenić skuteczność modelu i jak korzystać z takich modeli jak najbliższych sąsiadów, lasy losowe czy maszyna wektorów nośnych. Sporo miejsca poświęcono również sieciom neuronowym, mechanizmom ich działania i technikom treningu. I chociaż nie znajdziesz tutaj gotowych receptur, to zdobędziesz wiedzę potrzebną, by od podstaw zaprojektować działający model uczenia głębokiego.

W książce między innymi:

- budowanie dobrego zestawu danych uczących
- praca z bibliotekami scikit-learn i Keras
- klasyczne modele uczenia maszynowego
- mechanizm działania i uczenia sieci neuronowych
- modele wykorzystujące splotowe sieci neuronowe
- przygotowanie od podstaw działającego modelu

Dr Ronald T. Kneusel uczeniem maszynowym zajmuje się zawodowo od 2003 roku, specjalizuje się w dziedzinie przetwarzania obrazów. Jego fascynacja komputerami jednak trwa znacznie dłużej, bo od 1981 roku. Obecnie pracuje w spółce L3Harris Technologies, gdzie buduje systemy uczenia głębokiego. Jest autorem popularnych książek na temat pisania kodu i uczenia głębokiego.

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 978-83-283-8859-8



9 788328 388598

Cena: 99,00 zł

