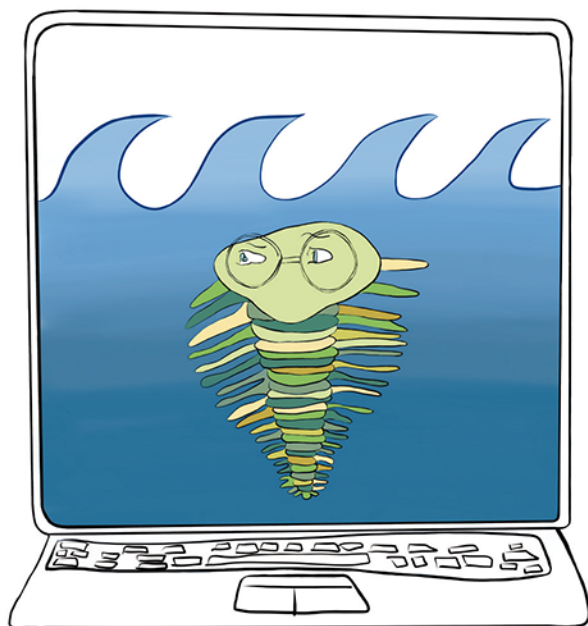




# UCZENIE GŁĘBOKIE I SZTUCZNA INTELIGENCJA

Interaktywny przewodnik ilustrowany



**JON KROHN**

GRANT BEYLEVELD, AGLAÉ BASSENS



Tytuł oryginału: Deep Learning Illustrated A Visual, Interactive Guide to Artificial Intelligence  
(Addison-Wesley Data & Analytics Series)

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-283-7914-5

Authorized translation from the English language edition, entitled DEEP LEARNING ILLUSTRATED: A VISUAL, INTERACTIVE GUIDE TO ARTIFICIAL INTELLIGENCE, by Jon Krohn published by Pearson Education, Inc, publishing as Addison Wesley Professional, Copyright © 2020 Pearson Education, Inc.

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 Pearson Education, Inc.

POLISH language edition published by HELION S.A., Copyright © 2021.

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.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/uczgsi>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

# Spis treści

<b>Spis ilustracji</b> .....	<b>xvii</b>
<b>Spis tabel</b> .....	<b>xxvii</b>
<b>Przykłady kodu</b> .....	<b>xxix</b>
<b>Przedmowa</b> .....	<b>xxxiii</b>
<b>Wstęp</b> .....	<b>xxxv</b>
<b>Podziękowania</b> .....	<b>xxxix</b>
<b>O autorach</b> .....	<b>xli</b>
<b>I Wprowadzenie do głębokiego uczenia</b> .....	<b>1</b>
<b>1. Widzenie biologiczne i maszynowe</b> .....	<b>3</b>
Widzenie biologiczne .....	3
Widzenie maszynowe .....	8
Neocognitron .....	8
LeNet-5 .....	9
Tradycyjne podejście w uczeniu maszynowym .....	12
Sieć ImageNet i konkurs ILSVRC .....	13
AlexNet .....	14
Plac zabaw TensorFlow .....	17
Quick, Draw! .....	19
Podsumowanie .....	19
<b>2. Języki ludzki i maszynowy</b> .....	<b>21</b>
Przetwarzanie języka naturalnego przy użyciu głębokiego uczenia .....	21
Sieci głębokiego uczenia automatycznie uczą się reprezentacji danych ....	22
Przetwarzanie języka naturalnego .....	23
Krótka historia wykorzystania głębokiego uczenia w przetwarzaniu języka naturalnego .....	24
Matematyczne reprezentacje języka .....	25
Kodowanie słów 1 z n .....	26
Wektory słów .....	27
Działania na wektorach słów .....	29
word2viz .....	30
Reprezentacje lokalne i rozproszone .....	32
Elementy naturalnego języka ludzi .....	33

Google Duplex .....	35
Podsumowanie .....	37
<b>3. Sztuka maszynowa .....</b>	<b>39</b>
Zakrapiana impreza .....	39
Arytmetyka nieprawdziwych twarzy .....	41
Transfer stylu: konwersja zdjęcia na obraz Moneta (i odwrotnie) .....	44
Tworzenie własnych, fotograficznie realistycznych rysunków .....	45
Tworzenie realistycznych obrazów na podstawie tekstu .....	45
Przetwarzanie obrazów przy użyciu głębokiego uczenia .....	46
Podsumowanie .....	47
<b>4. Maszyny i gry .....</b>	<b>49</b>
Głębokie uczenie, sztuczna inteligencja i inne bestie .....	49
Sztuczna inteligencja .....	49
Uczenie maszynowe .....	50
Uczenie się reprezentacji .....	51
Sztuczne sieci neuronowe .....	51
Głębokie uczenie .....	51
Widzenie maszynowe .....	52
Przetwarzanie języka naturalnego .....	53
Trzy kategorie problemów uczenia maszynowego .....	53
Uczenie nadzorowane .....	53
Uczenie nienadzorowane .....	54
Uczenie przez wzmacnianie .....	54
Głębokie uczenie przez wzmacnianie .....	56
Gry wideo .....	57
Gry planszowe .....	60
AlphaGo .....	60
AlphaGo Zero .....	62
AlphaZero .....	64
Manipulowanie obiektami .....	66
Popularne środowiska dla głębokiego uczenia przez wzmacnianie .....	68
OpenAI Gym .....	68
DeepMind Lab .....	68
Unity ML-Agents .....	71
Trzy kategorie sztucznej inteligencji .....	71
Sztuczna wąska inteligencja .....	71
Sztuczna ogólna inteligencja .....	71
Sztuczna superinteligencja .....	72
Podsumowanie .....	72

<b>II</b>	<b>Podstawy teoretyczne w ilustracjach</b>	<b>73</b>
	<b>5. Wóz (kodu) przed koniem (teorii)</b>	<b>75</b>
	Wymagania	75
	Instalacja	76
	Prosta sieć i biblioteka Keras	76
	Baza MNIST odręcznych cyfr	76
	Schemat sieci	78
	Załadowanie danych	79
	Przeformatowanie danych	81
	Zaprojektowanie architektury sieci neuronowej	82
	Trenowanie modelu sieci neuronowej	83
	Podsumowanie	84
	<b>6. Sztuczny neuron wykrywający hot dogi</b>	<b>85</b>
	Biologiczna neuroanatomia	85
	Perceptron	86
	Wykrywacz: „hot dog” — „nie hot dog”	86
	Najważniejsza formuła w tej książce	90
	Współczesne neurony i funkcje aktywacji	91
	Neuron sigmoidalny	92
	Neuron tangensowy	94
	Neuron ReLU	94
	Wybór neuronu	96
	Podsumowanie	96
	Kluczowe pojęcia	97
	<b>7. Sztuczne sieci neuronowe</b>	<b>99</b>
	Warstwa wejściowa	99
	Warstwa zagęszczona	99
	Zagęszczona sieć wykrywająca hot dogi	101
	Propagacja w przód w pierwszej warstwie ukrytej	102
	Propagacja w przód w kolejnych warstwach	103
	Warstwa softmax w sieci wykrywającej hot dogi	105
	Nowe spojrzenie na naszą płytką sieć	107
	Podsumowanie	109
	Kluczowe pojęcia	109
	<b>8. Trenowanie głębokich sieci</b>	<b>111</b>
	Funkcja kosztu	111
	Kwadratowa funkcja straty	112
	Wysycenie neuronu	112
	Entropia skrośna	113

Optymalizacja — uczenie się minimalizowania kosztu .....	115
Gradient prosty .....	115
Szybkość uczenia się .....	117
Wielkość paczki i stochastyczny gradient prosty .....	119
Ucieczka z lokalnego minimum .....	121
Propagacja wstecz .....	123
Dobór liczby warstw ukrytych i liczby neuronów .....	124
Średniogłęboka sieć w Keras .....	126
Podsumowanie .....	129
Kluczowe pojęcia .....	129
<b>9. Usprawnianie głębokich sieci .....</b>	<b>131</b>
Inicjalizacja wag .....	131
Rozkłady Xaviera Glorota .....	134
Niestabilne gradienty .....	137
Zanikające gradienty .....	137
Eksplodujące gradienty .....	137
Normalizacja paczki .....	138
Uogólnianie modelu (zapobieganie nadmiernemu dopasowaniu) .....	139
Regularyzacja L1 i L2 .....	141
Dropout .....	141
Powiększenie danych .....	144
Pomysłowe optymalizatory .....	144
Impet .....	144
Impet Nesterowa .....	145
AdaGrad .....	145
AdaDelta i RMSprop .....	146
Adam .....	146
Głęboka sieć neuronowa w Keras .....	147
Regresja .....	148
TensorBoard .....	151
Podsumowanie .....	153
Kluczowe pojęcia .....	154
<b>III Interaktywne zastosowania głębokiego uczenia .....</b>	<b>155</b>
<b>10. Widzenie maszynowe .....</b>	<b>157</b>
Konwolucyjne sieci neuronowe .....	157
Dwuwymiarowa struktura obrazów .....	157
Złożoność obliczeniowa .....	158
Warstwy konwolucyjne .....	158
Wielokrotne filtry .....	160
Konwolucyjny przykład .....	161
Hiperparametry filtru konwolucyjnego .....	165

Warstwy redukujące .....	166
Sieć LeNet-5 w Keras .....	168
Sieci AlexNet i VGGNet w Keras .....	173
Sieci rezydualne .....	176
Zanikający gradient — zmora głębokiej sieci konwolucyjnej .....	176
Połączenia rezydualne .....	177
Sieć ResNet .....	178
Zastosowania widzenia maszynowego .....	180
Wykrywanie obiektów .....	180
Segmentacja obrazów .....	183
Uczenie transferowe .....	185
Sieci kapsułowe .....	189
Podsumowanie .....	190
Kluczowe pojęcia .....	191
<b>11. Przetwarzanie języka naturalnego .....</b>	<b>193</b>
Wstępne przetwarzanie danych języka naturalnego .....	193
Tokenizacja .....	196
Zamiana wszystkich znaków na małe litery .....	198
Usunięcie stop-słów i interpunkcji .....	198
Stemming .....	199
Przetwarzanie n-gramów .....	200
Wstępne przetworzenie całego korpusu .....	202
Osadzanie słów przy użyciu techniki word2vec .....	205
Teoretyczne podstawy techniki word2vec .....	206
Ocenianie wektorów słów .....	208
Stosowanie techniki word2vec .....	208
Tworzenie wykresów wektorów słów .....	212
Pole pod krzywą ROC .....	215
Macierz pomyłek .....	217
Wyliczenie pola pod krzywą ROC .....	218
Klasyfikowanie języka naturalnego za pomocą znanych sieci .....	220
Załadowanie recenzji filmów z bazy IMDb .....	221
Badanie bazy IMDb .....	224
Ujednoczenie długości recenzji .....	227
Sieć zagęszczona .....	227
Sieci konwolucyjne .....	234
Sieci przystosowane do danych sekwencyjnych .....	238
Rekurencyjne sieci neuronowe .....	238
Jednostka LSTM .....	242
Dwukierunkowa jednostka LSTM .....	245
Spiętrzone modele rekurencyjne .....	246

Techniki sekwencja-sekwencja i atencja .....	248
Uczenie transferowe w przetwarzaniu języka naturalnego .....	249
Modele niesekwencyjne: funkcyjny interfejs API biblioteki Keras .....	249
Podsumowanie .....	254
Kluczowe pojęcia .....	255
<b>12. Sieci GAN .....</b>	<b>257</b>
Teoretyczne podstawy sieci GAN .....	257
Zbiór rysunków Quick, Draw! .....	260
Sieć dyskryminatora .....	263
Sieć generatora .....	266
Sieć antagonistyczna .....	269
Trening sieci GAN .....	271
Podsumowanie .....	278
Kluczowe pojęcia .....	279
<b>13. Głębokie uczenie przez wzmacnianie .....</b>	<b>281</b>
Podstawy teoretyczne uczenia przez wzmacnianie .....	281
Gra Cart-Pole .....	282
Proces decyzyjny Markowa .....	284
Optymalna polityka .....	286
Podstawy teoretyczne sieci głębokiego Q-uczenia .....	287
Funkcja wartości .....	288
Funkcja Q-wartości .....	288
Szacowanie optymalnej Q-wartości .....	289
Definiowanie agenta sieci DQN .....	290
Parametry inicjalizacyjne .....	293
Tworzenie sieci neuronowej agenta .....	294
Rejestrowanie przebiegu gry .....	295
Trening poprzez odtwarzanie zapisanych informacji .....	295
Wybór czynności do wykonania .....	297
Zapisywanie i ładowanie parametrów modelu .....	297
Interakcja ze środowiskiem OpenAI Gym .....	297
Optymalizacja hiperparametrów za pomocą narzędzia SLM Lab .....	301
Agenci nie tylko DQN .....	303
Algorytmy gradientu polityki i REINFORCE .....	304
Algorytm aktor-krytyk .....	304
Podsumowanie .....	305
Kluczowe pojęcia .....	306



<b>IV</b>	<b>Ty i sztuczna inteligencja .....</b>	<b>307</b>
	<b>14. Rozwijanie własnych projektów głębokiego uczenia ..</b>	<b>309</b>
	Pomysły na projekty głębokiego uczenia .....	309
	Widzenie maszynowe i sieci GAN .....	309
	Przetwarzanie języka naturalnego .....	311
	Głębokie uczenie przez wzmacnianie .....	312
	Przekształcanie istniejących projektów uczenia maszynowego .....	312
	Materiały do kolejnych projektów .....	313
	Projekty pożytku publicznego .....	314
	Proces modelowania i strojenie hiperparametrów .....	314
	Automatyzacja strojenia hiperparametrów .....	316
	Biblioteki głębokiego uczenia .....	317
	Keras i TensorFlow .....	317
	PyTorch .....	319
	MXNet, CNTK, Caffe i inne biblioteki .....	320
	Software 2.0 .....	320
	Nadejście ogólnej sztucznej inteligencji .....	322
	Podsumowanie .....	324
	<b>Dodatki .....</b>	<b>327</b>
	<b>A Formalna notacja sieci neuronowych .....</b>	<b>329</b>
	<b>B Propagacja wstecz .....</b>	<b>331</b>
	<b>C Biblioteka PyTorch .....</b>	<b>335</b>
	Funkcjonalności biblioteki PyTorch .....	335
	System autograd .....	335
	Zasada „definiowanie przez działanie” .....	335
	Biblioteka PyTorch kontra TensorFlow .....	336
	PyTorch w praktyce .....	337
	Instalacja .....	337
	Podstawowe jednostki .....	337
	Tworzenie głębokiej sieci neuronowej .....	339
	<b>Źródła ilustracji .....</b>	<b>341</b>



# 6.

## Sztuczny neuron wykrywający hot dogi

**P**o zachwytach nad świetlanymi wizjami zastosowań głębokiego uczenia i po zakodowaniu w pełni funkcjonalnej sieci neuronowej w rozdziale 5., przyszedł czas na zagłębienie się w teorię leżącą u podstaw tej dziedziny nauki. Zaczniemy od rozłożenia na czynniki pierwsze sztucznych neuronów, czyli podstawowych jednostek, które połączone ze sobą tworzą sieć neuronową.

### Biologiczna neuroanatomia

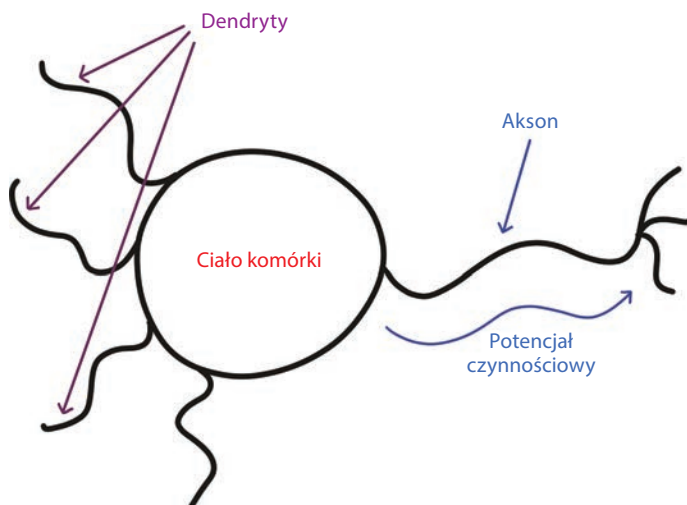
Jak napisaliśmy w początkowych akapitach tej książki, inspiracją do opracowania sztucznych neuronów były neurony biologiczne. Spójrzmy więc na rysunek 6.1 i przyswójmy sobie zawiłości pierwszego wykładu z neuroanatomii. Biologiczny neuron odbiera sygnały wejściowe za pomocą licznych (zazwyczaj tysięcy) **dendrytów**, przy czym każdy dendryt odbiera sygnały od innego neuronu w układzie nerwowym, czyli w biologicznej sieci neuronowej. Gdy sygnał poprzez dendryt dotrze do **ciała komórki**, wprowadza niewielką zmianę w panującym w niej napięciu elektrycznym<sup>1</sup>. Niektóre dendryty nieznacznie zwiększają, a inne zmniejszają to napięcie. Jeżeli sumaryczny efekt tych zmian spowoduje wzrost napięcia z poziomu spoczynkowego  $-70$  mV do krytycznego progu  $-50$  mV, neuron wyzwala tzw. **potencjał czynnościowy**, który wędruje przez akson do innych neuronów w sieci.

Podsumowując, biologiczny neuron wykonuje sekwencje następujących trzech czynności:

1. **Odbiera informacje** od innych neuronów.
2. **Przekształca otrzymane informacje** w zmiany napięcia w ciele komórki.
3. Jeżeli zmiana jest większa niż określona wartość progowa, neuron **wysyła sygnał do innych neuronów** w sieci.

---

<sup>1</sup> Ścisłej, zmienia różnicę potencjałów między wnętrzem komórki a jej otoczeniem.



Rysunek 6.1. Anatomia biologicznego neuronu



Kolory tekstu odpowiadają kolorom na rysunku 6.1 oznaczającym odpowiednio dendryt, ciało komórki i akson. Technikę tę będziemy stosować wielokrotnie w tej książce, również podczas opisywania kluczowych równań i użytych w nich zmiennych.

## Perceptron

Pod koniec lat 50. ubiegłego wieku amerykański neurobiolog Frank Rosenblatt (rysunek 6.2) opublikował artykuł na temat opracowanego przez siebie **perceptronu** — algorytmu inspirowanego funkcjonowaniem biologicznych neuronów. Była to pierwsza formuła sztucznego neuronu<sup>2</sup>. Perceptron (rysunek 6.3), podobnie jak jego biologiczny pierwowzór, wykonuje następujące czynności:

1. **Odbiera sygnały wejściowe** od wielu innych neuronów.
2. **Agreguje sygnały wejściowe** wykonując proste działanie zwane **sumowaniem ważonym**.
3. **Generuje sygnał wyjściowy**, jeżeli suma ważona przekroczy zadany poziom. Sygnał wysyła do wielu innych neuronów w sieci.

## Wykrywacz: „hot dog” — „nie hot dog”

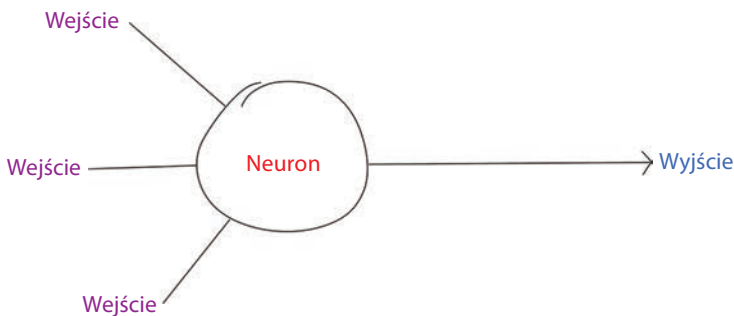
Aby zrozumieć działanie perceptronu przeanalizujemy przyjemny przykład. Będzie to perceptron specjalizujący się w rozróżnianiu, czy dany przedmiot jest hot dogiem, czy nim nie jest.

Najważniejszą cechą perceptronu jest możliwość odbierania na wejściu i generowania na wyjściu wyłącznie danych binarnych. Dlatego nasz perceptron wykrywający hot dogi

<sup>2</sup> F. Rosenblatt, *The perceptron: A probabilistic model for information storage and the organization in the brain*, „Psychological Review”, 1958, nr 65, s. 386 – 408.



**Rysunek 6.2.** Frank Rosenblatt, amerykański neurobiolog i badacz zachowań. Większość swoich prac wykonał w Cornell Aeronautical Laboratory, m.in. skonstruował tam swój perceptron Mark I. To urządzenie, najstarszy relikwiarz sztucznej inteligencji, można dziś oglądać w Smithsonian Institution w Waszyngtonie



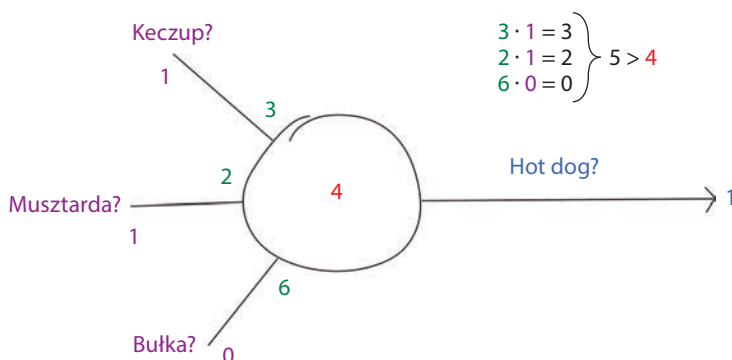
**Rysunek 6.3.** Schemat perceptronu, czyli pierwszego sztucznego neuronu. Zwróć uwagę na podobieństwo do biologicznego neuronu pokazanego na rysunku 6.1

musi na swoich trzech wejściach, wskazujących odpowiednio, czy przedmiot zawiera keczup, musztardę lub bułkę, odbierać wartości 0 lub 1. Spójrzmy na rysunek 6.4:

- Pierwsze wejście (fioletowa **jedyńka**) oznacza, że obiekt prezentowany perceptronowi zawiera keczup.
- Drugie wejście (również fioletowa **jedyńka**) oznacza, że obiekt zawiera musztardę.
- Trzecie wejście (fioletowe **zero**) oznacza, że obiekt nie zawiera bułki.

Aby określić, czy obiekt jest hot dogiem, czy nie, perceptron niezależnie *waży* każdą z trzech wartości wejściowych<sup>3</sup>. W tym całkowicie wymyślonym przykładzie wagi dobraliśmy dowolnie. Waga 6 przypisana bułce oznacza, że jest to najistotniejszy czynnik decydujący o tym, czy obiekt jest hot dogiem, czy nim nie jest. Wagę równą 3 ma keczup, a najmniejszą, równą 2, musztarda.

3 Jeśli potrafisz modelować regresję, ten paradygmat powinien brzmieć znajomo.



**Rysunek 6.4.** Pierwszy przykład perceptronu wykrywającego hot dogi. W tym przypadku odpowiedź perceptronu brzmi, że obiekt rzeczywiście jest hot dogiem

Wyliczmy ważoną sumę wartości wejściowych. Każdą wartość osobno mnożymy przez jej wagę, a uzyskane wyniki sumujemy. Najpierw wyliczmy ważone wejścia:

1. *keczap*:  $3 \cdot 1 = 3$
2. *musztarda*:  $2 \cdot 1 = 2$
3. *bułka*:  $6 \cdot 0 = 0$

Na podstawie powyższych iloczynów możemy wyliczyć ważoną sumę wartości wejściowych  $3 + 2 + 0 = 5$ . Uogólniając powyższy przykład, formuła ważonej sumy wartości wejściowych jest następująca:

$$\sum_{i=1}^n w_i x_i \quad (6.1)$$

gdzie:

- $w_i$  oznacza wagę  $i$ -tej wartości wejściowej (w naszym przykładzie  $w_1 = 3$ ,  $w_2 = 2$ ,  $w_3 = 0$ ),
- $x_i$  oznacza  $i$ -tą wartość wejściową (w naszym przykładzie  $x_1 = 1$ ,  $x_2 = 1$ ,  $x_3 = 0$ ),
- $w_i x_i$  oznacza iloczyn  $w_i$  i  $x_i$ , czyli ważoną  $i$ -tą wartość,
- $\sum_{i=1}^n$  oznacza sumę wszystkich ważonych wartości wejściowych  $w_i x_i$ , gdzie  $n$  jest całkowitą liczbą wartości wejściowych (w naszym przykładzie są trzy wartości, ale w sztucznym neuronie może ich być dowolnie dużo).

Ostatnią operacją wykonywaną przez algorytm perceptronu jest sprawdzenie, czy ważona suma wartości wejściowych jest większa od *wartości progowej*. Tak jak wcześniej w tym przykładzie, tutaj również odgórnie wybraliśmy wartość **4** (czerwona cyfra w środku neuronu na rysunku 6.4). Algorytm perceptronu wygląda następująco:

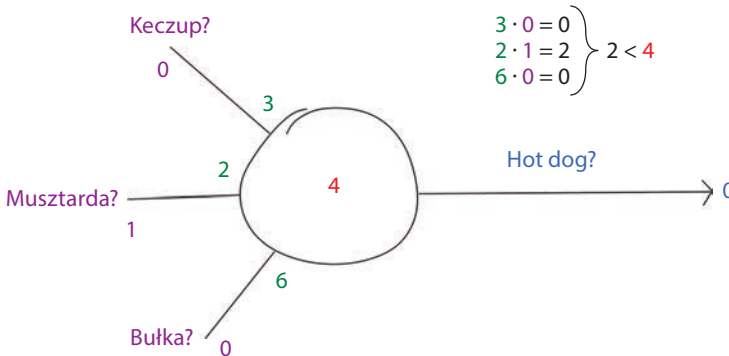
$$\sum_{i=1}^n w_i x_i \begin{cases} > \text{wartość progowa, wyjście } 1 \\ \leq \text{wartość progowa, wyjście } 0 \end{cases} \quad (6.2)$$

Przy czym:

- jeżeli ważona suma wartości wejściowych jest większa od **wartości progowej**, na wyjściu pojawia się liczba **1** oznaczająca, że według perceptronu dany obiekt jest hot dogiem,
- jeżeli ważona suma wartości wejściowych jest mniejsza lub równa **wartości progowej**, na wyjściu pojawia się liczba **0** oznaczająca, że według perceptronu dany obiekt nie jest hot dogiem.

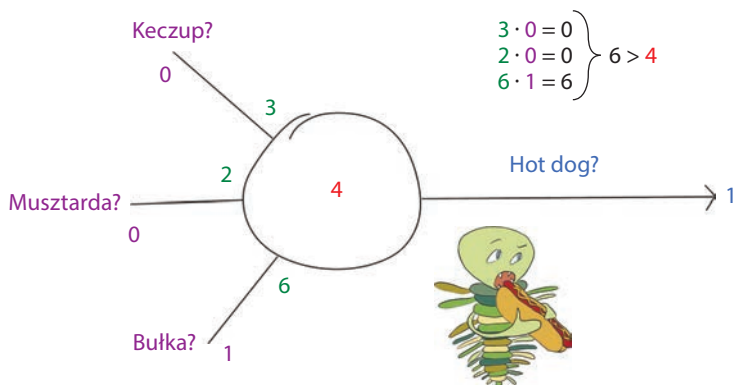
Podsumowując przykład z rysunku 6.4: ważona suma równa 5 jest większa od **wartości progowej** równej 4, więc na wyjściu naszego perceptronu wykrywającego hot dogi pojawia się liczba **1**.

Zmieńmy teraz nieco nasz przykład. Na rysunku 6.5 obiekt oceniany przez perceptron zawiera tylko musztardę; nie ma keczupu ani bułki. W tym przypadku ważona suma wartości wejściowych jest równa 2. Ponieważ jest ona mniejsza od **wartości progowej** perceptronu, na jego wyjściu pojawia się liczba **0** oznaczająca, że dany obiekt *nie* jest hot dogiem.



**Rysunek 6.5.** Drugi przykład perceptronu wykrywającego hot dogi. W tym przypadku perceptron prognozuje, że dany obiekt nie jest hot dogiem

W trzecim, ostatnim przykładzie, pokazanym na rysunku 6.6, neuron ocenia obiekt, który nie ma keczupu ani musztardy, ale *ma* bułkę. W tym przypadku ważona suma wartości wejściowych jest równa 6. Ponieważ jest większa od **wartości progowej**, algorytm prognozuje, że obiekt jest hot dogiem i zwraca liczbę **1**.



**Rysunek 6.6.** Trzeci przykład perceptronu wykrywającego hot dogi. W tym przypadku perceptron prognozuje, że dany obiekt jest hot dogiem

## Najważniejsza formuła w tej książce

Aby sformułować uproszczone, uniwersalne równanie perceptronu, musimy wprowadzić termin **odchylenie** (ang. *bias*), które oznaczymy jako  $b$ , równe ujemnej **wartości progowej** sztucznego neuronu:

$$b \equiv -\text{wartość progowa} \quad (6.3)$$

Odchylenie i wagi to wszystkie parametry neuronu decydujące o tym, co neuron generuje na wyjściu w odpowiedzi na dane wejściowe.

Po wyjaśnieniu pojęcia odchylenia, możemy sformułować najczęściej przytaczaną formułę perceptronu:

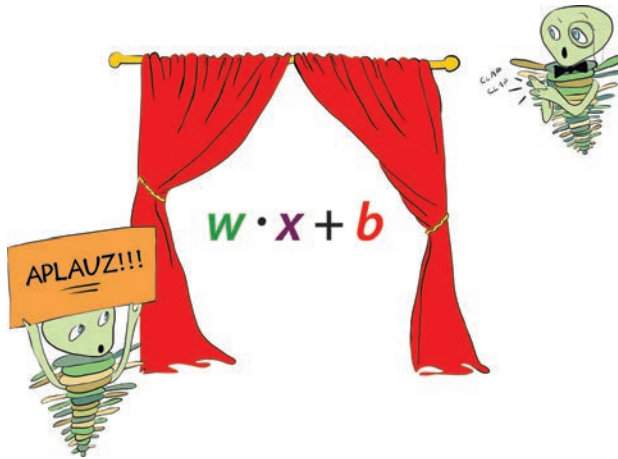
$$\text{wyjście} \begin{cases} 1, \text{ jeżeli } wx + b > 0 \\ 0 \text{ w przeciwnym wypadku} \end{cases} \quad (6.4)$$

Zwróć uwagę, że do pierwotnej formuły 6.2 perceptronu wprowadziliśmy pięć zmian:

1. W miejscu **wartości progowej** umieściliśmy odchylenie  $b$ .
2. Odchylenie  $b$  przenieśliśmy na tę samą stronę formuły, na której znajdują się pozostałe zmienne.
3. Użyliśmy macierzy  $w$  reprezentującej wszystkie wagi  $w_i$  od  $w_1$  do  $w_n$ .
4. Analogicznie, użyliśmy macierzy  $x$  reprezentującej wszystkie wagi  $x_i$  od  $x_1$  do  $x_n$ .
5. Użyliśmy **iloczynu skalarnego**  $wx$  upraszczającego zapis ważonej sumy wartości wejściowych (bardziej rozbudowaną postać przedstawia formuła 6.1:  $\sum_{i=1}^n w_i x_i$ ).

Sercem formuły 6.4 perceptronu jest zapis  $wx + b$ , który dla podkreślenia jego wagi wycięliśmy i umieściliśmy na rysunku 6.7. *Jeżeli z tego rozdziału zapamiętasz tylko jedną rzecz, niech będzie to formuła z trzema zmiennymi, opisująca działanie sztucznego neuronu w ogólności.* W tej książce będziemy się do niej odwoływać wielokrotnie.





**Rysunek 6.7.** Ogólna formuła opisująca działanie sztucznego neuronu, do której będziemy wracać od czasu do czasu. Jest to najważniejsza formuła w tej książce

Aby możliwie uprościć arytmetykę w perceptronie wykrywającym hot dogi, wszystkie parametry, które zdefiniowaliśmy — wagi i odchylenie — przyjmowały wartości dodatnie. W praktyce parametry mogą jednak mieć również ujemne wartości, a do tego rzadko są liczbami całkowitymi. Parametry są zgrabniejszymi wartościami zmiennoprzecinkowymi.

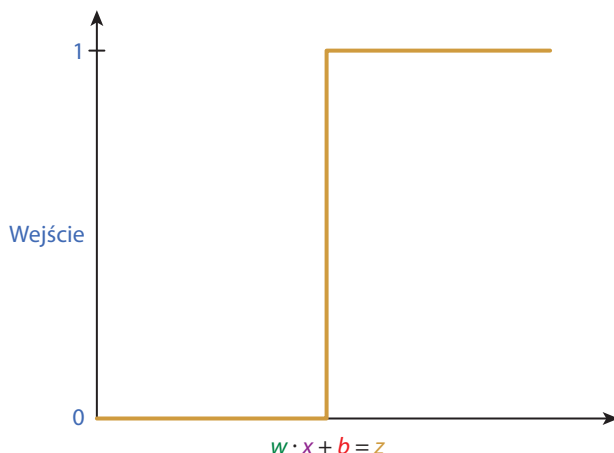
Ponadto, w opisanych przykładach wartości parametrów zostały przyjęte odgórnie. W rzeczywistości są one wyliczane podczas trenowania sieci na danych. W rozdziale 8. opiszemy, jak taki trening parametrów neuronów wygląda w praktyce.



## Współczesne neurony i funkcje aktywacji

Współczesne sztuczne neurony, takie jak tworzące ukrytą warstwę płytkiej sieci, którą zbudowaliśmy w rozdziale 5. (patrz rysunek 5.4 oraz notatnik *Płytką sieć w Keras*), nie są perceptronami. Perceptron stanowi wprawdzie stosunkowo proste wprowadzenie do sztucznych neuronów, ale obecnie nie jest powszechnie używany. Jego największy mankament polega na tym, że odbiera on wyłącznie binarne dane wejściowe i generuje tylko binarne dane wyjściowe. W praktyce często zdarza się, że trzeba prognozować wyniki na podstawie ciągłych danych wejściowych, a nie liczb binarnych. Sam ten fakt sprawia, że perceptrony nie są przydatne.

Mniej oczywistym, za to jeszcze istotniejszym skutkiem powyższego ograniczenia jest to, że trenowanie perceptronu jest trudne. Przeanalizujmy rysunek 6.8, na którym wprowadziliśmy nową zmienną  $z$ , stanowiącą skróconą formę zaczej formuły  $w \cdot x + b$  z rysunku 6.7. Jeżeli zmienna ta ma wartość mniejszą lub równą zero, perceptron generuje najmniejszą możliwą wartość wyjściową, czyli 0. Jeżeli wartość  $z$  jest choćby minimalnie dodatnia, perceptron zwraca największą możliwą wartość równą 1. Takie gwałtowne i ekstremalne przejście nie jest pożądane podczas treningu. W rzeczywistości zmienne  $w$  i  $b$  są nieznacznie modyfikowane, jeżeli wydaje się, że spowoduje to poprawę wartości



**Rysunek 6.8.** Przejście wartości wyjściowej perceptronu z zera do jedynki jest gwałtowne, przez co trudno jest precyzyjnie manipulować zmiennymi  $w$  i  $b$ , aby uzyskać pożądaną wartość wyjściową

wyjściowej<sup>4</sup>. W przypadku perceptronu większość niewielkich korekt  $w$  i  $b$  nie ma żadnego wpływu na jego skuteczność. Zmienna  $z$  przyjmuje zazwyczaj wartości znacznie mniejsze lub znacznie większe od zera. Już samo takie działanie jest niekorzystne, ale sytuacja jest jeszcze gorsza: od czasu do czasu niewielka korekta  $w$  i  $b$  powoduje zmianę wartości  $z$  z ujemnej na dodatnią lub odwrotnie, skutkując drastycznymi zmianami wartości wyjściowej z 0 do 1 (lub odwrotnie). Z definicji perceptron nie jest finezyjnym wynalazkiem — albo krzyczy, albo milczy.

## Neuron sigmoidalny

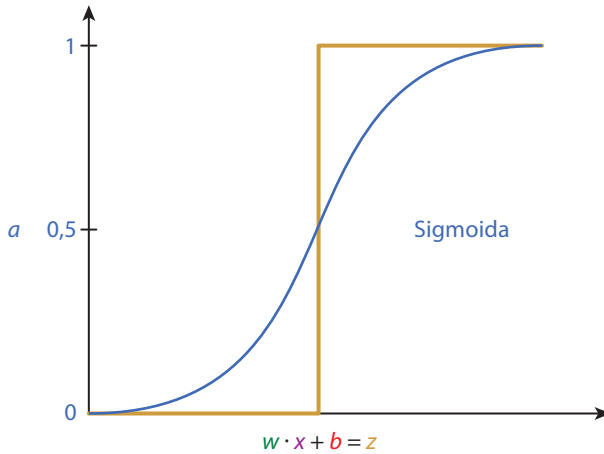
Rysunek 6.9 przedstawia alternatywę dla problematycznie działającego perceptronu: krzywą łagodnie wznoszącą się od 0 do 1. Jest to **sigmoida** o równaniu  $\sigma(z) = \frac{1}{1+e^{-z}}$ , gdzie:

- $z$  jest odpowiednikiem wyrażenia  $w \cdot x + b$ ,
- $e$  oznacza zaczynającą się od 2,718 stałą matematyczną, najbardziej znaną ze swojej głównej roli odgrywanej w naturalnej funkcji wykładniczej,
- $\sigma$  jest grecką literą *sigma*, od której pochodzi nazwa krzywej.

Sigmoida jest naszym pierwszym przykładem funkcji aktywacji sztucznych neuronów. Być może ją kojarzysz, ponieważ w rozdziale 5. była użyta w neuronach tworzących ukrytą warstwę w notatniku *Płytki sieć w Keras*. Jak się przekonasz w tym podrozdziale, sigmoida jest kanoniczną funkcją aktywacji. Jest tak podstawowa, że tradycyjnie litera  $\sigma$  jest używana na oznaczenie każdej funkcji aktywacji. Sygnał wyjściowy funkcji aktywacji dowolnego neuronu jest nazywany po prostu jego aktywacją. W tej książce oznaczamy go literą  $a$ , jak na pionowej osi na rysunku 6.9.

Naszym zdaniem nie trzeba znać równania sigmoidy ani żadnej innej funkcji aktywacji. Łatwiej jest zrozumieć działanie danej funkcji eksperymentując z nią. Dlatego prześledź

<sup>4</sup> Poprawa w tym przypadku oznacza przybliżenie wyjścia do faktycznej wartości  $y$  dla danej wartości wejściowej  $x$ . Zajmiemy się tym dokładniej w rozdziale 8.



Rysunek 6.9. Sigmoida jako funkcja aktywacji

z nami kod załączonego do książki notatnika *Sigmoida*. Jedyną zależnością w tym notatniku jest stała  $e$ , którą ładujemy za pomocą instrukcji `from math import e`. Dalej następuje ciekawa część, w której definiujemy sigmoidę:

```
def sigmoid(z):
    return 1/(1+e**(-z))
```

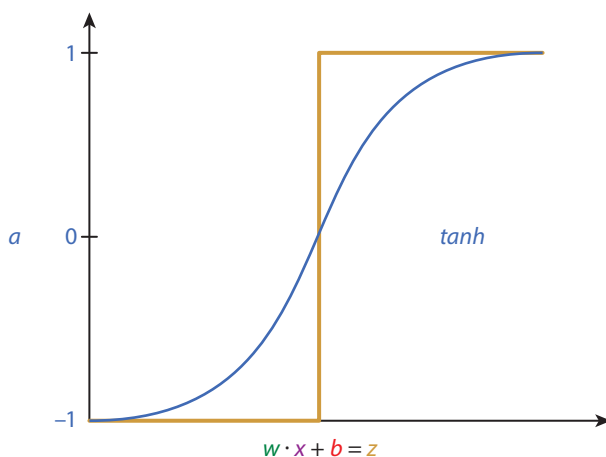
Jak pokazuje rysunek 6.9 i wynik wyrażenia `sigmoid(0.00001)`, wartość wejściowa bliska zeru powoduje, że funkcja zwraca wynik równy mniej więcej 0.5. Coraz większe dodatnie wartości wejściowe skutkują generowaniem wartości wyjściowych coraz bliższych jedności. W ekstremalnym przypadku wartość wejściowa równa 10000 powoduje uzyskanie wartości wyjściowej równej 1.0. Dla coraz mniejszych ujemnych wartości wejściowych funkcja zwraca wyniki łagodnie zbliżające się do zera. Na przykład wyrażenie `sigmoid(-1)` zwraca wynik 0.2689, a `sigmoid(-10)` wynik  $4.5398e-05^5$ .

Sztuczny neuron, którego funkcją aktywacji jest sigmoida, jest nazywany **neuronem sigmoidalnym**. Jego przewaga nad perceptronem jest oczywista: niewielkie, stopniowe zmiany parametru  $w$  lub  $b$  skutkują niewielkimi, stopniowymi zmianami wartości  $z$  i równie niewielkimi, stopniowymi zmianami aktywacji  $a$ . Wyjątkiem są duże ujemne lub dodatnie zmiany  $z$ . Dla ekstremalnych ujemnych wartości  $z$  neuron sigmoidalny, podobnie jak perceptron, zwraca wynik 0, a dla dodatnich — 1. Podobnie, niewielkie modyfikacje wag i odchylenia wprowadzane podczas treningu mają niewielki lub żaden wpływ na wartość wyjściową i w efekcie trening kończy się. To zjawisko nosi nazwę **wysycenia** neuronu i jest typowe dla większości funkcji aktywacji. Na szczęście, są metody zapobiegania wysyceniu, które poznasz w rozdziale 9.

5 Nie należy mylić litery  $e$  w zapisie  $4.5398e-05$  z podstawą logarytmu naturalnego. W wynikach generowanych przez kod litera  $t$  oznacza wykładnik dziesiętny, w tym przypadku jest to liczba  $4,5398 \cdot 10^{-5}$ .

## Neuron tangensowy

Znanym kuzynem neuronu sigmoidalnego jest neuron tangensowy. Jego funkcja aktywacji jest opisana równaniem  $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$  i pokazana na rysunku 6.10. Kształtem przypomina sigmoidę, ale różni się od niej tym, że wartości sigmoidy zawierają się w przedziale  $[0; 1]$ , a tangensa hiperbolicznego w zakresie  $[-1; 1]$ . Wbrew pozorom, nie jest to jedynie kosmetyczna różnica. Ujemne wartości wejściowe  $z$  odpowiadają ujemnym wartościom aktywacji  $a$ , wartość wejściowa  $z = 0$  odpowiada  $a = 0$ , natomiast dodatnie wartości wejściowe  $z$  odpowiadają dodatnim wartościom aktywacji  $a$ . Średnio, wartości wyjściowe neuronu tangensowego oscylują w granicach zera. Jak się dowiesz dokładniej w rozdziałach 7. – 9., wartości bliskie zera są zazwyczaj wartościami wejściowymi  $x$  dla innych sztucznych neuronów. Oznacza to z kolei, że prawdopodobieństwo wysycenia neuronu jest mniejsze, a cała sieć może się uczyć skuteczniej.



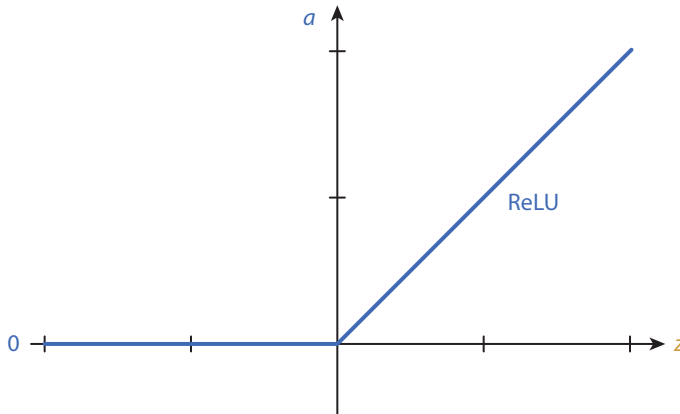
Rysunek 6.10. Tangens hiperboliczny jako funkcja aktywacji

## Neuron ReLU

Ostatnim typem neuronu, który szczegółowo opisujemy w tej książce, jest **ReLU** (ang. *rectified linear unit* — **rektyfikowana jednostka liniowa**), którego funkcję aktywacji przedstawia rysunek 6.11. Funkcja ta, daleko odbiegająca kształtem od sigmoidy i tangensa hiperbolicznego, została spopularyzowana przez Vinoda Naira i Geoffa Hinton<sup>6</sup> (rysunek 1.16), zainspirowanych właściwościami biologicznych neuronów<sup>7</sup>. Równanie funkcji ReLU ma postać  $a = \max(0, z)$ .

6 V. Nair, G. Hinton, *Rectified linear units improve restricted Boltzmann machines*, materiały z konferencji International Conference on Machine Learning, 2010.

7 Potencjał czynnościowy biologicznych neuronów jest wyzwalany tylko w trybie „dodatnim” — nie ma trybu „ujemnego”. Patrz R. Hahnloser i H. Seung, *Permitted and forbidden sets in symmetric threshold-linear networks*, „Advances in Neural Information Processing Systems”, 2000, nr 13.



Rysunek 6.11. ReLU jako funkcja aktywacji

Funkcja ReLU nie jest skomplikowana:

- Jeżeli wartość  $z$  jest dodatnia, funkcja zwraca nietkniętą wartość wyjściową  $a = z$ .
- Jeżeli  $z$  jest równe zero lub mniejsze od tej wartości, wtedy aktywacja  $a = 0$ .

ReLU jest jedną z najprostszych *nieliniowych* funkcji, jaką można sobie wyobrazić. Podobnie jak w przypadku sigmoidy i tangensa hiperbolicznego, jej wyjście  $a$  nie zmienia się liniowo wraz ze zmianami  $z$ . ReLU to w rzeczywistości dwie połączone ze sobą różne funkcje liniowe: jedna zwraca wartość 0 dla ujemnych wartości wejściowych, a druga wartość  $z$  dla wartości dodatnich. Nieliniowość jest krytyczną cechą wszystkich funkcji aktywacji stosowanych w modelach głębokiego uczenia. Jak to pokazał Michael Nielsen w serii ciekawych interaktywnych apletów w e-booku *Neural Networks and Deep Learning*<sup>8</sup>, dzięki nieliniowości można w modelu głębokiego uczenia przybliżyć każdą funkcję ciągłą. Uniwersalna możliwość przybliżania danych wyjściowych  $y$  w zależności od danych wejściowych  $x$  jest jedną z charakterystycznych właściwości głębokiego uczenia. Podejście to okazuje się bardzo skuteczne w szerokim zakresie zastosowań.

Stosunkowo prosta nieliniowość funkcji ReLU jest jej zaletą. Jak się dowiesz w rozdziale 8., opisanie procesu uczenia się wartości  $w$  i  $b$  wymaga użycia równań różniczkowych cząstkowych, których wyliczenie jest o wiele prostsze w przypadku liniowych części funkcji ReLU niż nieliniowych, np. sigmoidy lub tangensa hiperbolicznego<sup>9</sup>. Potwierdzeniem użyteczności neuronów ReLU była oparta na nich sieć AlexNet (rysunek 1.17), która w 2012 r. przyczyniła się do tego, że istniejące wzorce widzenia maszynowego legły w gruzach i rozpoczęła się era głębokiego uczenia. Obecnie ReLU są najczęściej stosowanym typem neuronu w ukrytych warstwach sieci głębokiego uczenia. Pojawia się również w większości dołączonych do tej książki notatników Jupyter.

<sup>8</sup> [neuralnetworksanddeeplearning.com/chap4.html](http://neuralnetworksanddeeplearning.com/chap4.html)

<sup>9</sup> Ponadto, jak coraz częściej potwierdzają badania, aktywacje ReLU sprzyjają rozrzedzaniu parametrów. Oznacza to, że mniej skomplikowane funkcje wykazują tendencję do lepszego uogólniania danych weryfikacyjnych. Więcej o uogólnianiu modeli dowiesz się w rozdziale 9.

## Wybór neuronu

W każdej ukrytej warstwie sztucznej sieci neuronowej można zastosować taką funkcję aktywacji, jaka się żywnie podoba. Zważywszy, że najlepiej jest wybrać funkcję nieliniową, aby przybliżyć funkcję ciągłą, pole wyboru jest dość szerokie. Aby Ci ułatwić podjęcie decyzji, uszeregowaliśmy opisane w tym rozdziale neurony od najmniej do najbardziej polecanych.

1. *Perceptron*, ze swoimi binarnymi wejściami i radykalnymi, binarnymi skokami na wyjściu, nie ma praktycznego zastosowania w modelach głębokiego uczenia.
2. Neuron *sigmoidalny* to opcja do przyjęcia, ale oparta na nim sieć zazwyczaj uczy się wolniej niż w przypadku użycia neuronu tangensowego lub ReLU. Dlatego użycie neuronów sigmoidalnych zalecamy tylko w sytuacjach, w których pożądane są dane wyjściowe z zakresu  $[0; 1]$ <sup>10</sup>.
3. Neuron tangensowy to racjonalny wybór. Jak pisaliśmy wcześniej, wartości wyjściowe oscylujące w okolicach zera sprawiają, że sieć uczy się szybko.
4. Preferowanym przez nas neuronem jest ReLU, ponieważ dzięki niemu wydajność obliczeń w algorytmach uczenia jest wysoka. Z naszego doświadczenia wynika, że oparta na tego typu neuronie sztuczna sieć dobrze się kalibruje w krótkim czasie.

Oprócz neuronów omówionych w tym rozdziale, istnieje szerokie spektrum funkcji aktywacyjnych, których lista stale się powiększa. W chwili pisania tej książki biblioteka Keras<sup>11</sup> oferowała kilka zaawansowanych funkcji, m.in. *nieszczelną ReLU*, *parametryczną ReLU* i *wykładniczą liniową*. Wszystkie trzy wywodzą się z funkcji ReLU. Zachęcamy Cię do zapoznania się w wolnej chwili z ich opisem w dokumentacji biblioteki. Ponadto, możesz wymieniać neurony stosowane w notatnikach Jupyter na dowolne inne i porównywać wyniki. Będzie nam miło, jeżeli się okaże, że Twoje sieci neuronowe będą dokładniejsze i bardziej wydajne od naszych.

## Podsumowanie

W tym rozdziale opisaliśmy podstawy matematyczne jednostek tworzących sztuczną sieć neuronową, w tym modele głębokiego uczenia. Podsumowaliśmy zalety i wady najpopularniejszych typów neuronów i przedstawiliśmy zalecenia, które z nich należy stosować w modelach głębokiego uczenia. W rozdziale 7. opiszemy, jak łączy się sztuczne neurony w sieć, która może się uczyć surowych danych, i jak przybliża się skomplikowane funkcje.

10 W rozdziałach 7. i 11. opiszemy kilka takich przypadków, przede wszystkim sieć klasyfikatora binarnego, w której warstwa wyjściowa składa się z pojedynczego neuronu sigmoidalnego.

11 Dokumentacja jest dostępna pod adresem [keras.io/layers/advanced-activations](https://keras.io/layers/advanced-activations).

## Kluczowe pojęcia

W kolejnych rozdziałach książki będziemy stopniowo rozszerzać listę kluczowych pojęć. Jeżeli je zapamiętasz, bez trudności zrozumiesz kolejne rozdziały, a po zakończeniu lektury tej książki będziesz posiadał solidną wiedzę o teorii i zastosowaniach głębokiego uczenia. Opisane dotychczas najbardziej krytyczne pojęcia są następujące:

- parametry:
  - waga  $w$
  - odchylenie  $b$
- aktywacja  $a$
- sztuczne neurony:
  - sigmoidalny
  - tangensowy
  - ReLU





# PROGRAM PARTNERSKI

— GRUPY HELION —

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: PRZEKONAJ SIĘ NA WŁASNE OCZY!

Uczenie maszynowe jest przyszłością naszej cywilizacji. Już dziś wywiera ogromny wpływ na nasze życie. Odmieniło kształt wielu sektorów: usług konsumpcyjnych, inżynierii, bankowości, medycyny czy produkcji. Trudno też przewidzieć zmiany, jakie potęga sieci neuronowych przyniesie nam w nadchodzących latach. Osoby zajmujące się zawodowo uczeniem głębokim i sieciami neuronowymi mogą liczyć na ekscytujące możliwości, jednak zaawansowana matematyka i teoria stanowiące podstawę uczenia maszynowego mogą zniechęcać do prób poważnego zajęcia się tą dziedziną.

Ta książka jest nowatorskim podręcznikiem, w którym w zrozumiały, intuicyjny sposób opisano techniki sztucznej inteligencji. Została wzbogacona kolorowymi ilustracjami i zrozumiałym kodem, dzięki czemu pozwala o wiele łatwiej zagłębić się w złożoność modeli głębokiego uczenia. Trudniejsze zagadnienia matematyczne zostały ograniczone do niezbędnego minimum, przedstawiono je jednak w sposób maksymalnie przystępny. Po lekturze zrozumiesz, czym jest głębokie uczenie, dlaczego stało się tak popularne i jak się ma do innych dziedzin uczenia maszynowego. W pragmatyczny sposób opisano takie aspekty zastosowań głębokiego uczenia jak widzenie maszynowe, przetwarzanie języka naturalnego, generowanie obrazów, a nawet gra w różne gry. Prezentowane treści uzupełnia praktyczny kod i szereg wskazówek dotyczących korzystania z bibliotek Keras i TensorFlow.

W książce między innymi:

- teoretyczne podstawy sztucznej inteligencji, w tym sieci neuronowe i ich trening oraz optymalizacja
- sieci konwolucyjne, rekurencyjne, GAN, głębokie uczenie przez wzmacnianie
- potencjał systemów głębokiego uczenia
- narzędzia do tworzenia, stosowania i usprawniania modeli głębokiego uczenia
- tworzenie interaktywnych aplikacji opartych na głębokim uczeniu

**JON KROHN** kieruje zespołem badaczy danych w firmie untapt. Jest wykładowcą akademickim i autorem uznanej serii podręczników.

**GRANT BEYLEVELD** zajmuje się zastosowaniem głębokiego uczenia w przetwarzaniu języka naturalnego.

**AGLAÉ BASSENS** jest belgijską artystką: ilustratorką, malarką i autorką murali.

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

Sprawdź nasze szkolenia!  
SZKOLENIA  
AKADEMIA IT & BUSINESS  
HELIONSZKOLENIA.PL

KOD KORZYŚCI  
Sięgnij po więcej!



ISBN 978-83-283-7914-5



9 788328 379145

 **Pearson**  
Addison-Wesley

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 99,00 zł