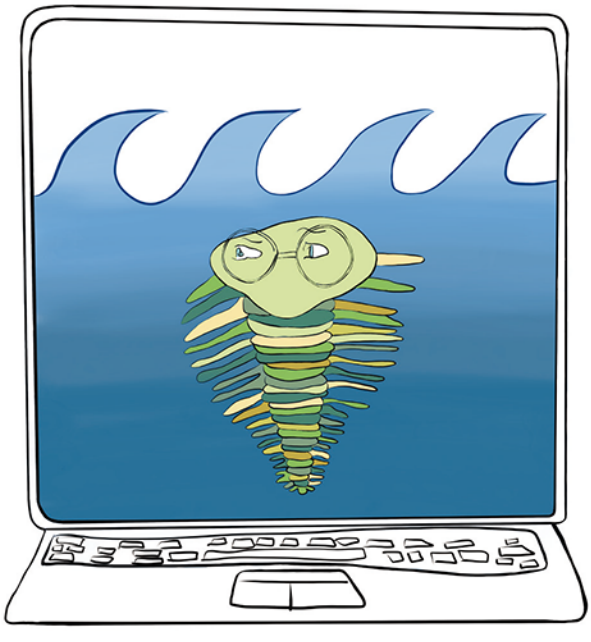
UCZENIE GŁĘBOKIE I SZTUCZNA INTELIGENCJA

Interaktywny przewodnik ilustrowany



JON KROHN

GRANT BEYLEVELD, AGLAÉ BASSENS

Helion

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

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

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

Printed in Poland.

Spis treści

6.

Sztuczny neuron wykrywający hot dogi

Po 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.

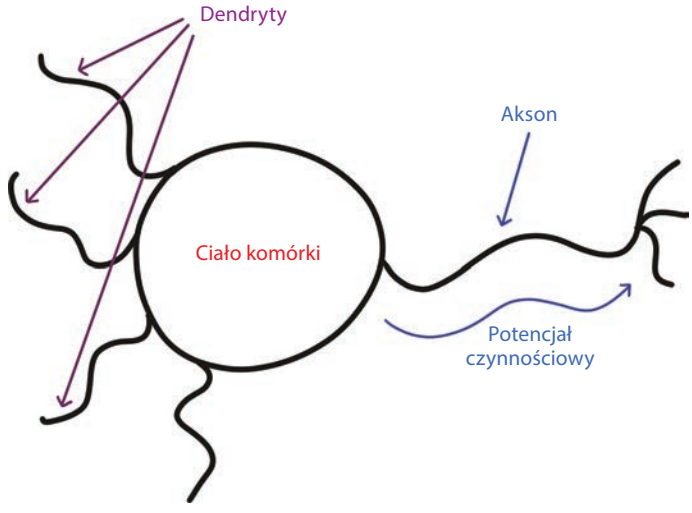
Biologiczna neuroanatomia

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[1]. 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.

1 Ściślej, 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[2]. 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.

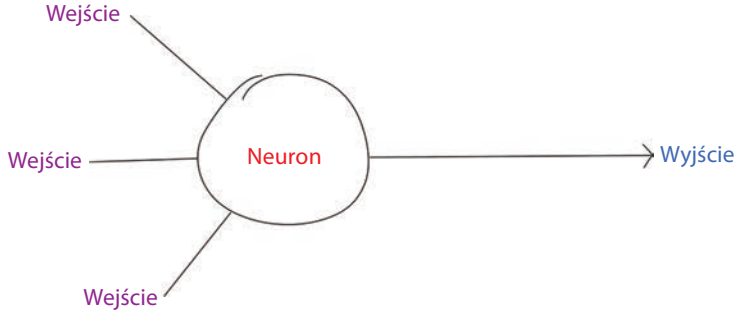
Wykrywacz: „hot dog" — „nie hot dog"

Aby zrozumieć działanie perceptronu przeanalizujmy 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

2 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 relikt sztucznej inteligencji, można dziś oglądać w Smithsonian Institution w Waszyngtonie



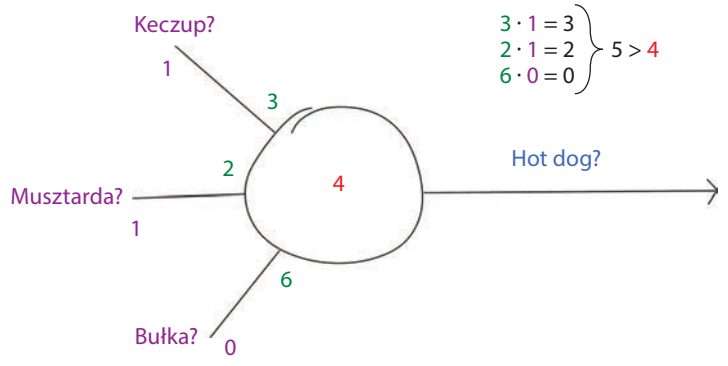
Rysunek 6.3. Schemat perceptronu, czyli pierwszego sztucznego neuronu. Zwróć uwagę na podobieństwo do biologicznego neuronu pokazanego na rysunku 6.1

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 jedynka) oznacza, że obiekt prezentowany perceptronowi zawiera keczup.
- Drugie wejście (również fioletowa jedynka) 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[3]. 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. *keczup*: $3 \cdot 1 = 3$
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:

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

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,
- $\Sigma_{i=1}^{n}$ 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:

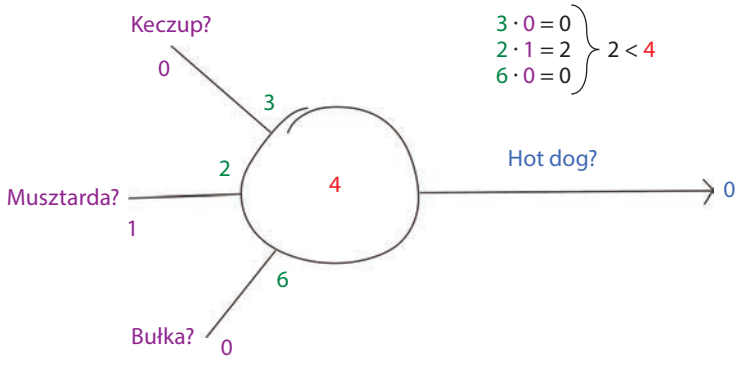
$$\sum_{i=1}^{n} w_i x_i \begin{cases} > \textit{wartość progowa}, \text{wyjście } 1 \\ \leq \textit{wartość progowa}, \text{wyjście } 0 \end{cases} \qquad (6.2)$$

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.

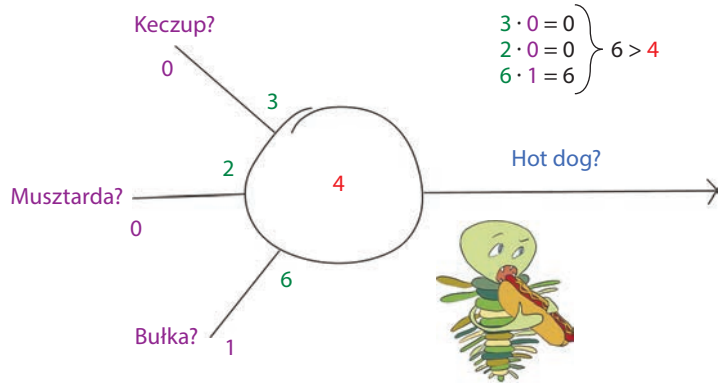
Podsumowując przykład z rysunku 6.4: ważona suma równa 5 jest większa od wartości progowej równej 4, więc na wyjściu naszego perceptronu wykrywającego hot dogi pojawia się liczba 1.

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:

$$b \equiv -\textit{wartość progowa} \qquad (6.3)$$

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

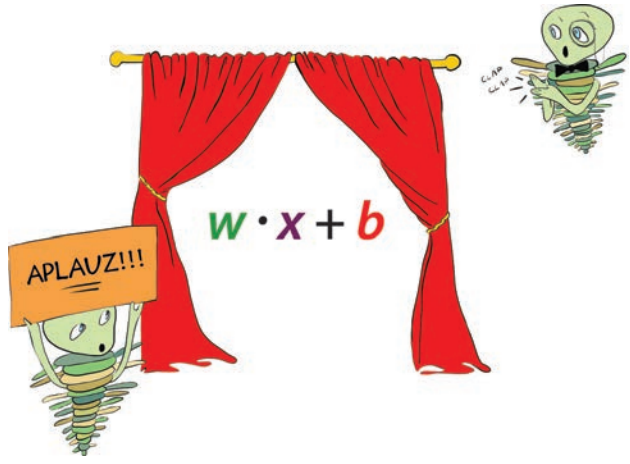
Po wyjaśnieniu pojęcia odchylenia, możemy sformułować najczęściej przytaczaną formułę perceptronu:

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

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

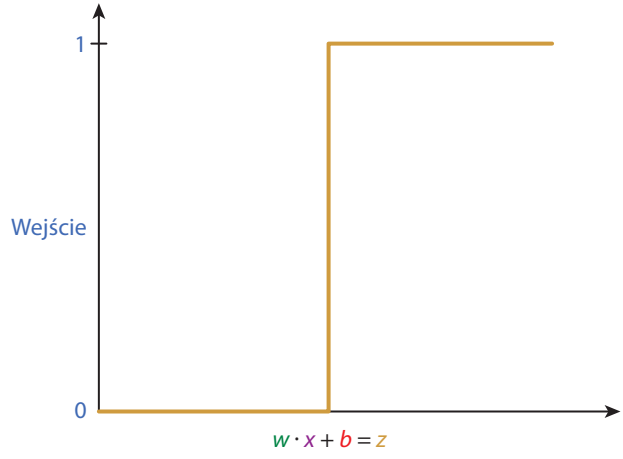
Aby możliwie uprościć arytmetykę w perceptronie wykrywającym hot dogi, wszystkie parametry, które zdefiniowaliśmy — wagi i odchylenie — przyjmowały wartości dodatnie. W praktyce parametry mogą jednak mieć również ujemne wartości, a do tego rzadko są liczbami całkowitymi. Parametry są zgrabniejszymi wartościami 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 *Płytka 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 zacnej formuły $w \cdot x + b$ z rysunku 6.7. Jeżeli zmienna ta ma wartość mniejszą lub równą zeru, 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[4]. 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

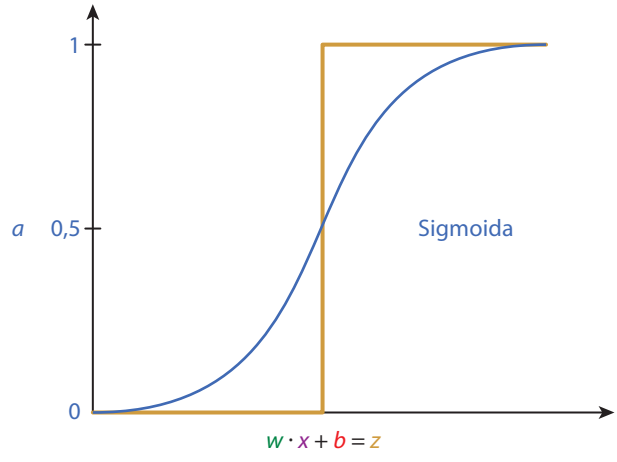
Rysunek 6.9 przedstawia alternatywę dla problematycznie działającego perceptronu: krzywą łagodnie wznoszącą się od 0 do 1. Jest to **sigmoida** o równaniu $\sigma(z) = \frac{1}{1+e^{-z}}$, gdzie:

- 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,
- σ 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 *Płytka sieć w Keras*. Jak się przekonasz w tym podrozdziale, sigmoida jest kanoniczną funkcją aktywacji. Jest tak podstawowa, że tradycyjnie litera σ 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

4 *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)
```

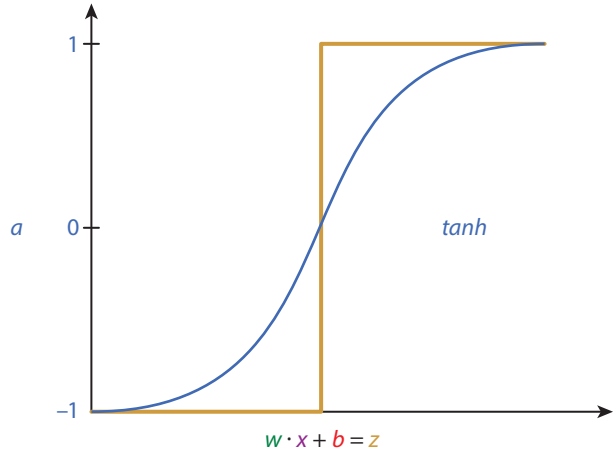
Jak pokazuje rysunek 6.9 i wynik wyrażenia `sigmoid(0.00001)`, wartość wejściowa bliska zeru powoduje, że funkcja zwraca wynik równy mniej więcej `0.5`. Coraz większe dodatnie wartości wejściowe skutkują generowaniem wartości wyjściowych coraz bliższych jedności. W ekstremalnym przypadku wartość wejściowa 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 ta 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.



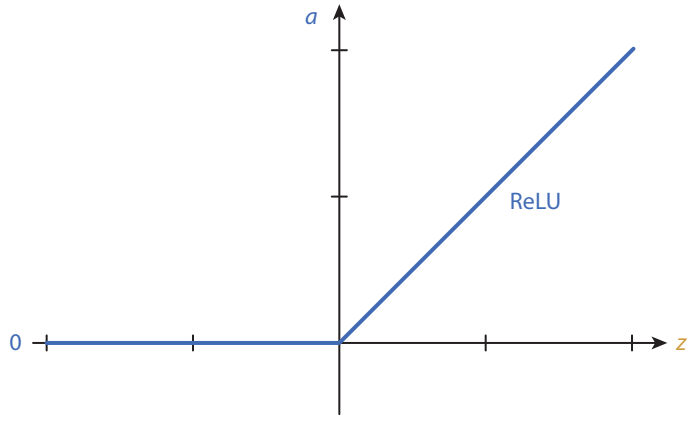
Rysunek 6.10. Tangens hiperboliczny jako funkcja aktywacji

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 Hintona[6] (rysunek 1.16), zainspirowanych właściwościami biologicznych neuronów[7]. 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*[8], 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[9]. Potwierdzeniem użyteczności neuronów ReLU była oparta na nich sieć AlexNet (rysunek 1.17), która w 2012 r. przyczyniła się to 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.

8 *neuralnetworksanddeeplearning.com/chap4.html*

9 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][10].
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[11] 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*.

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

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 konsumenckich, 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

KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-7914-5
9 788328 379145
Cena: 99,00 zł
INFORMATYKA W NAJLEPSZYM WYDANIU
Pearson
Addison-Wesley