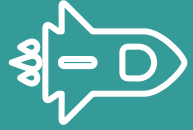




Cześć!

Krzysztof Bork-Ceszlak

Data Science Enthusiast, specialized in Deep Learning



Deep Learning

Rozkład jazdy:

- Czym jest deep learning?
 - Jakie są rodzaje głębokiego uczenia ?
 - Budowa sztucznej sieci neuronowej.
- Proces uczenia

Czym jest deep learning?
Gdzie jest ?...

SZTUCZNA INTELIGENCJA



Czym jest deep learning?
Gdzie jest ?...



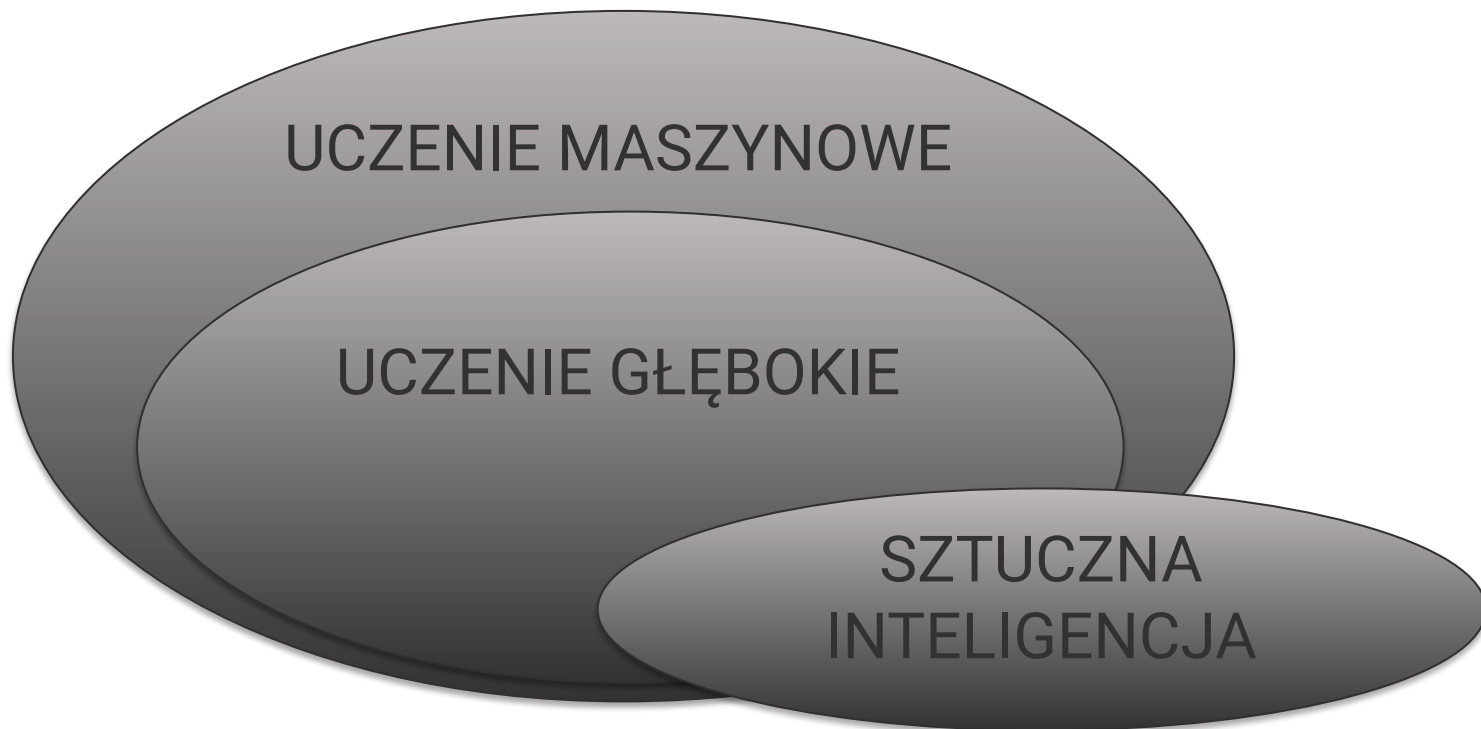
Czym jest deep learning?
Gdzie jest ?...



Czym jest deep learning? Gdzie jest ?...

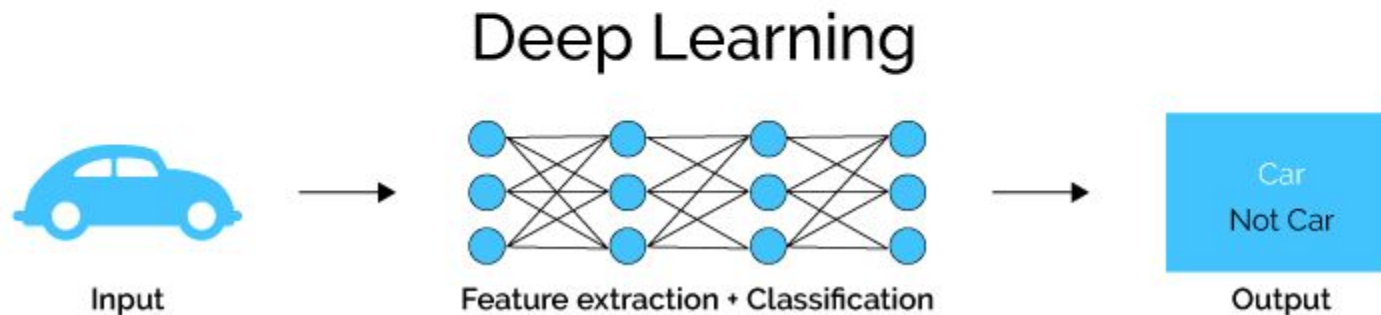
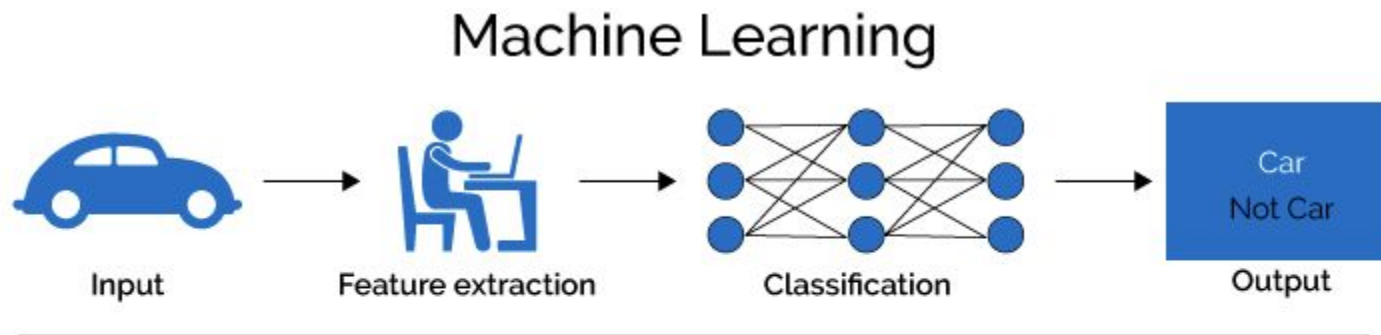


Czym jest deep learning? Gdzie jest ?...



Czym jest deep learning?

Czym się różni od ML?



Czym jest deep learning?

Czym się różni od ML?

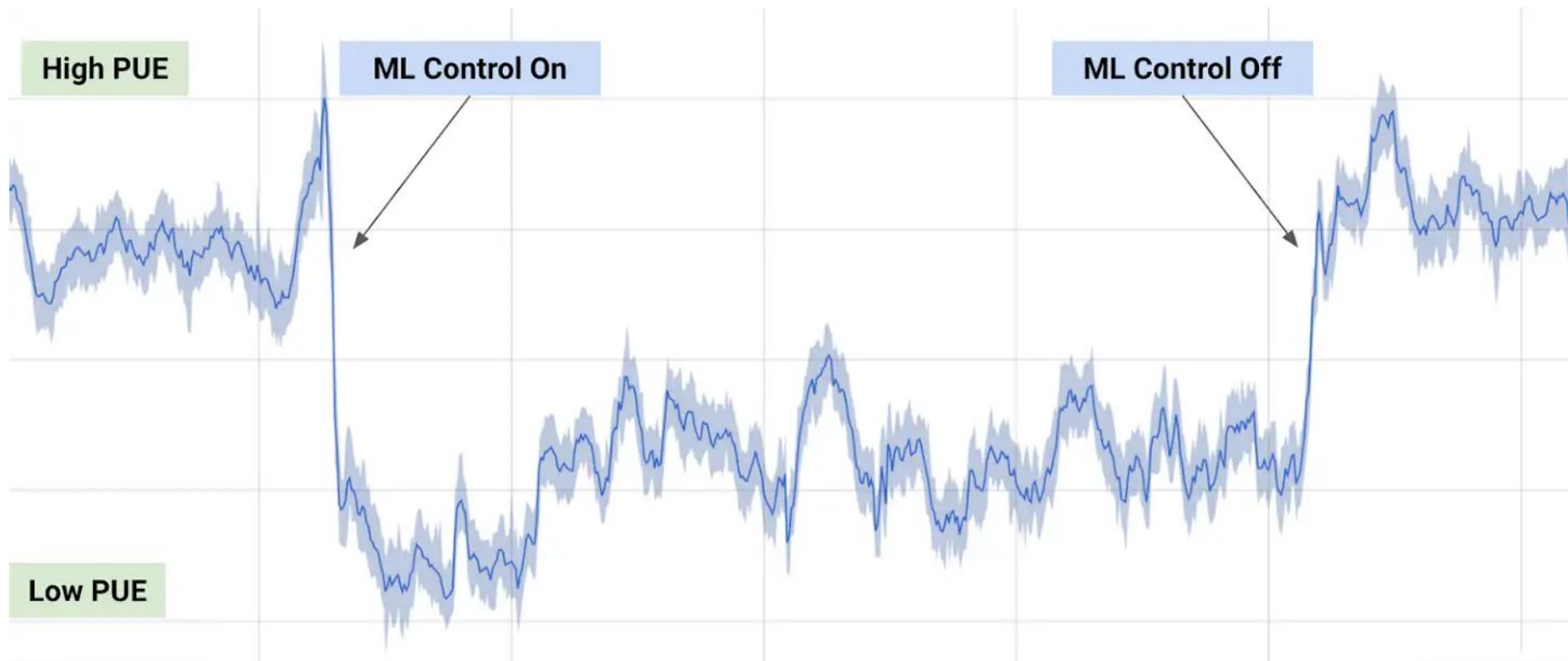
- Posiada ,uczący się feature extractor'
- Modele bywają ogromne
- Na ogół potrzebuje ogromną ilość danych
- Oraz mocny sprzęt do obliczeń ;)
- Do niemal każdego problemu można dostosować odpowiednią architekturę sieci neuronowej

Rodzaje głębokiego uczenia

- Predykcja wartości danych „tabelkowych”
- Rozpoznawanie obrazów
- Detekcja obiektów
- Klasyfikacja tekstu
- Translacja
- Wiele wiele innych...

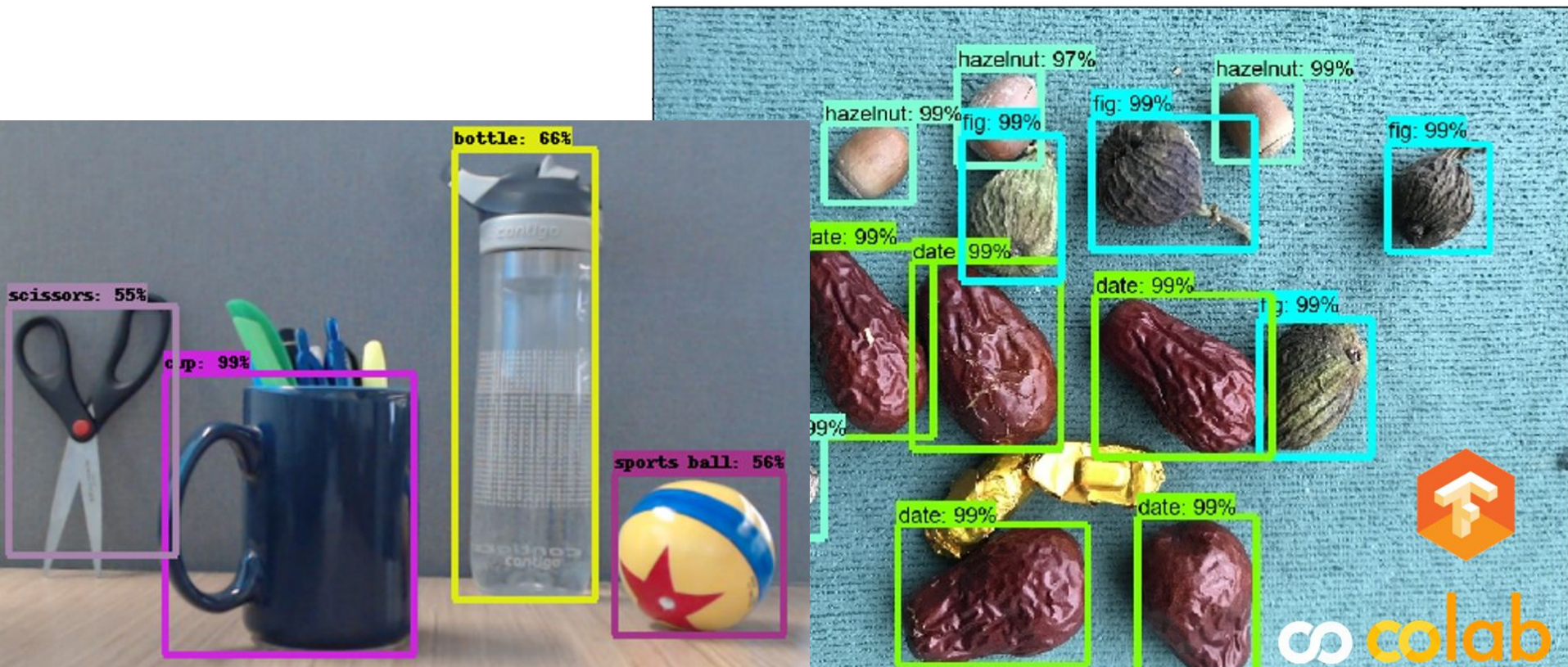
Rodzaje głębokiego uczenia

optymalizacja kosztów zużycia energii



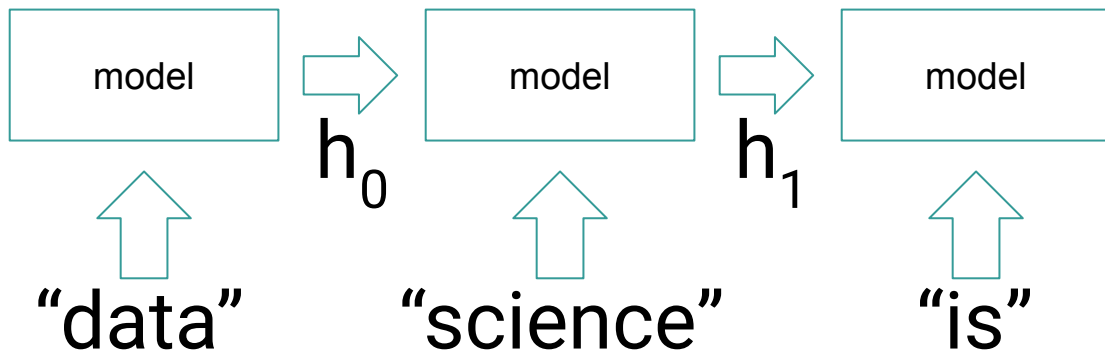
Rodzaje głębokiego uczenia

detekcja obiektów – konwolucyjne sieci neuronowe



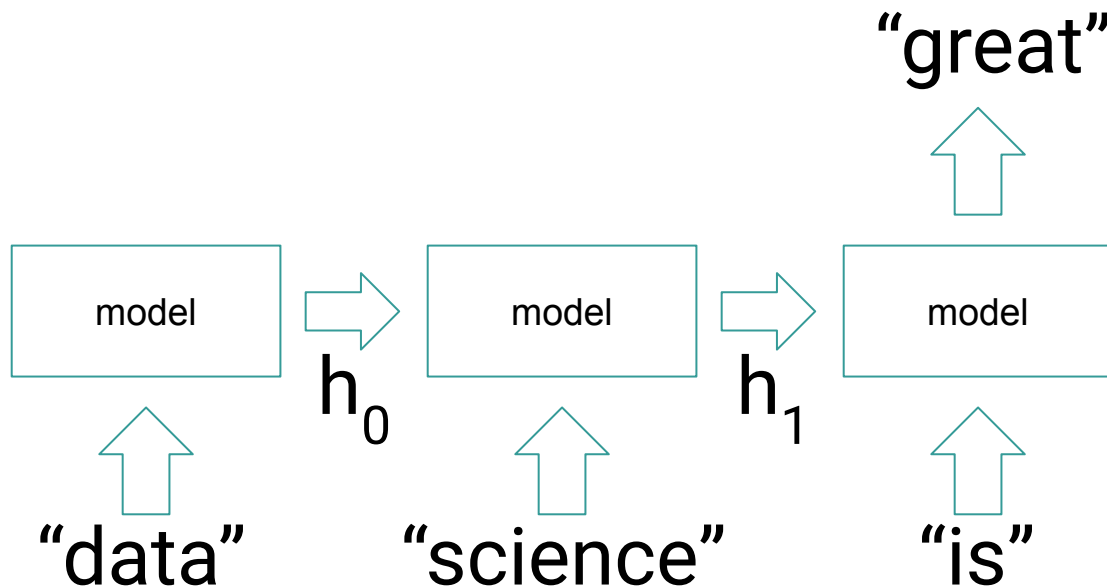
Rodzaje głębokiego uczenia

generowanie tekstu – rekurencyjne sieci neuronowe



Rodzaje głębokiego uczenia

generowanie tekstu – rekurencyjne sieci neuronowe



Rodzaje głębokiego uczenia

generowanie tekstu – gpt3.5 & ChatGPT



Chciałbym porozmawiać w języku polskim.



Oczywiście, z przyjemnością porozmawiam z Tobą w języku polskim. W czym mogę Ci pomóc?



Jakie są sposoby na prowadzenie warsztatów z nauki programowania?



Istnieje kilka sposobów na prowadzenie warsztatów z nauki programowania, w tym:

1. Metoda projektowa - polega na realizacji konkretnego projektu od początku do końca.
2. Nauczanie przez naśladowanie - uczestnicy naśladują prowadzącego w wykonywaniu krok po kroku konkretnych czynności.
3. Ćwiczenia praktyczne - uczestnicy wykonują samodzielnie ćwiczenia, które pomagają im zrozumieć konkretne zagadnienie.
4. Dyskusje i pytania - prowadzący umożliwia uczestnikom zadawanie pytań i prowadzenie dyskusji na tematy związane z programowaniem.
5. Kombinacja powyższych metod - wiele warsztatów łączy kilka powyższych metod, aby umożliwić uczestnikom jak najlepsze zrozumienie i przyswojenie wiedzy.

Ważne jest, aby dobierać odpowiednie metody do poziomu zaawansowania uczestników i ich indywidualnych potrzeb.

Rodzaje głębokiego uczenia

generowanie dźwięku – magenta tensorflow

MuseNet

by OpenAI



Rodzaje głębokiego uczenia

generowanie rzeczywistego obrazu



Rodzaje głębokiego uczenia

uczenie ze wzmocnieniem



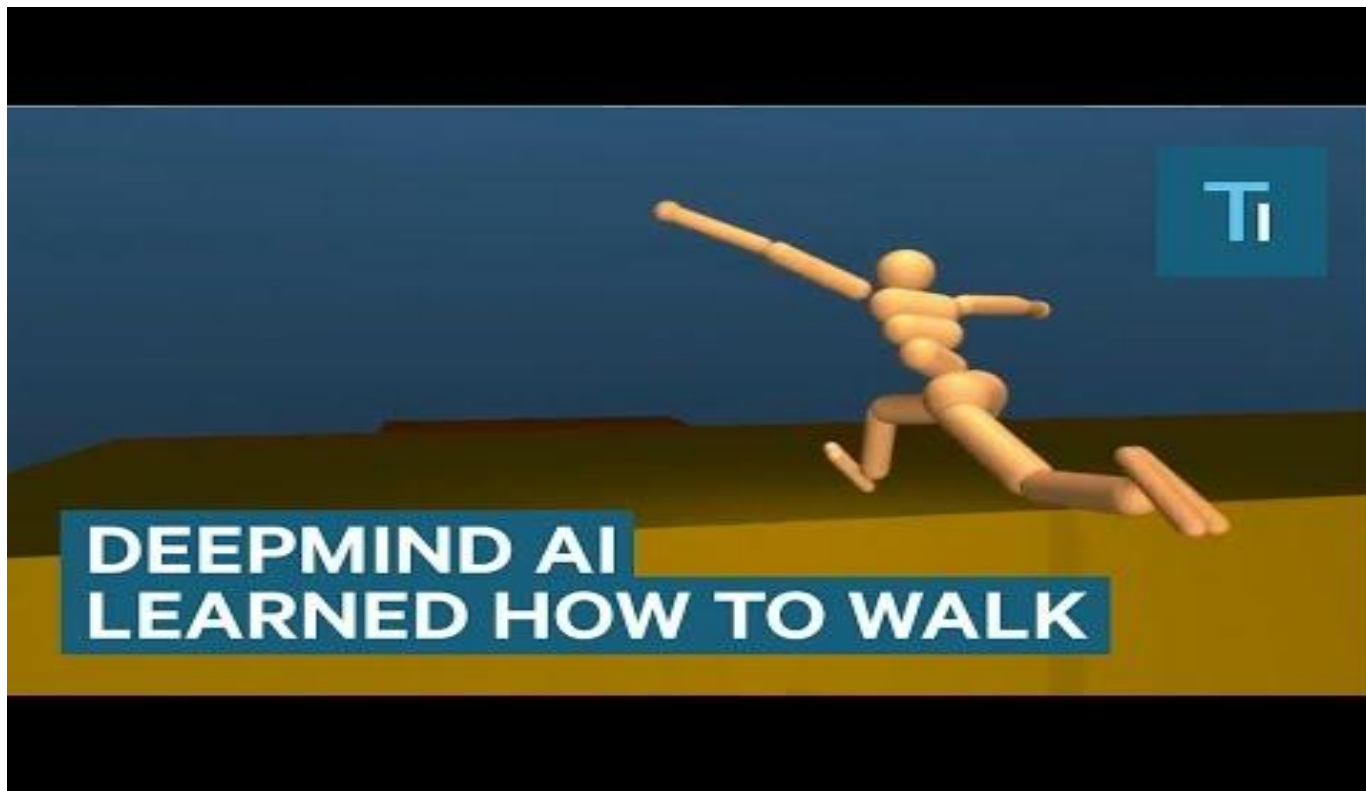
NAGRODA



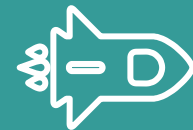
KARA

Rodzaje głębokiego uczenia

uczenie ze wzmocnieniem



More news
about AI



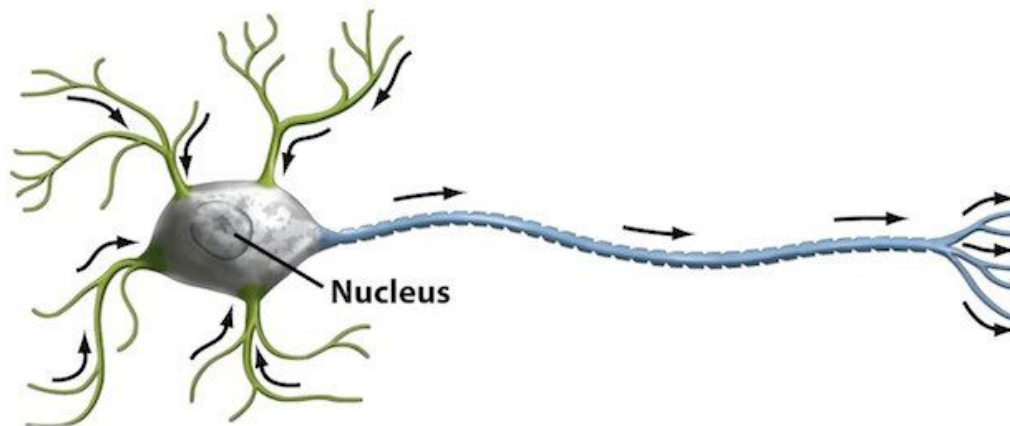
Sieć neuronowa

Koncepcja sieci neuronowej

Skąd taki pomysł?

Koncepcja sieci neuronowej

Skąd taki pomysł?



Koncepcja sieci neuronowej

Skąd taki pomysł?

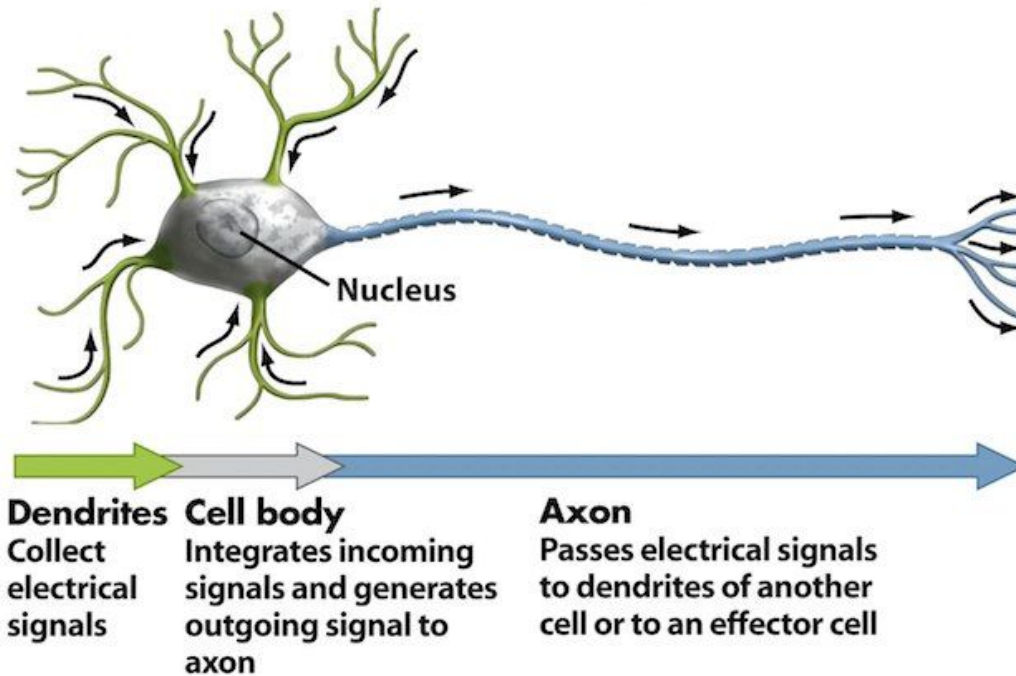
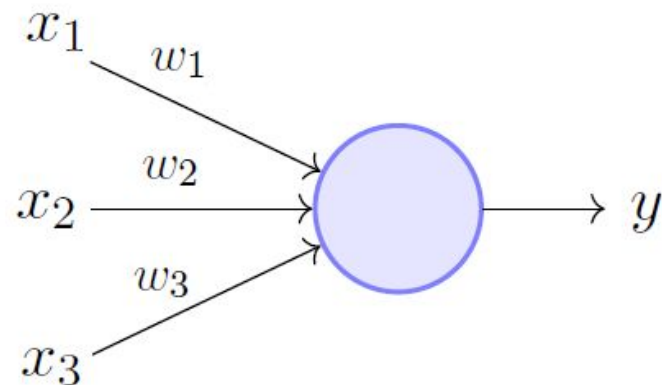
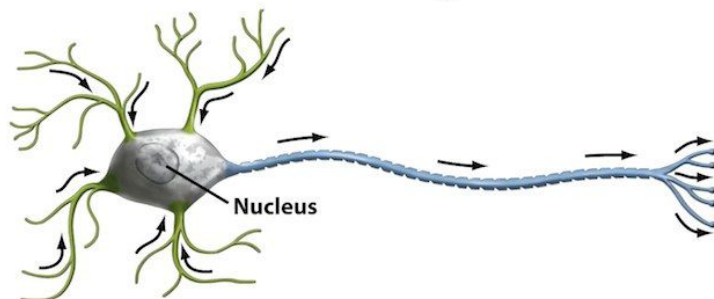


Figure 45-2b Biological Science, 2/e
© 2005 Pearson Prentice Hall, Inc.

Koncepcja sieci neuronowej

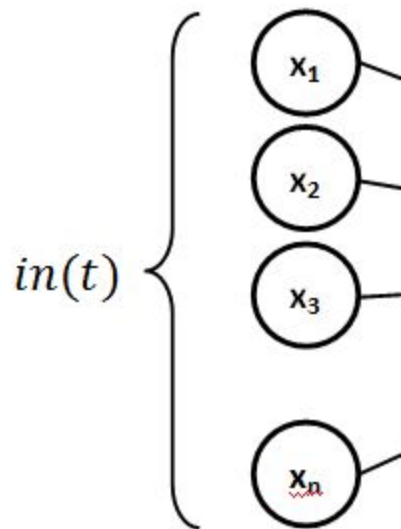
Skąd taki pomysł?



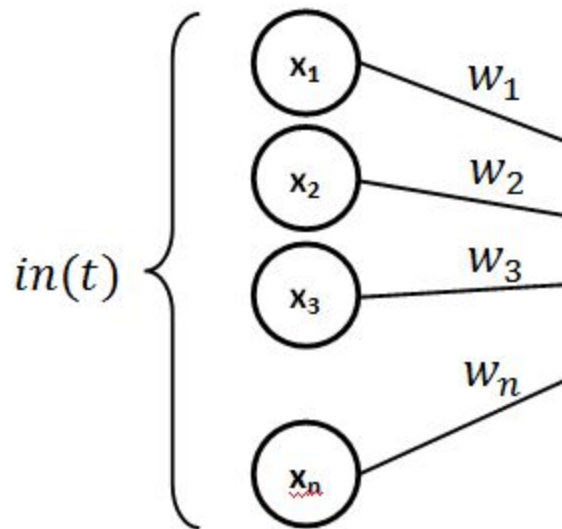
Perceptron Model (Minsky-Papert in 1969)

Koncepcja sieci neuronowej

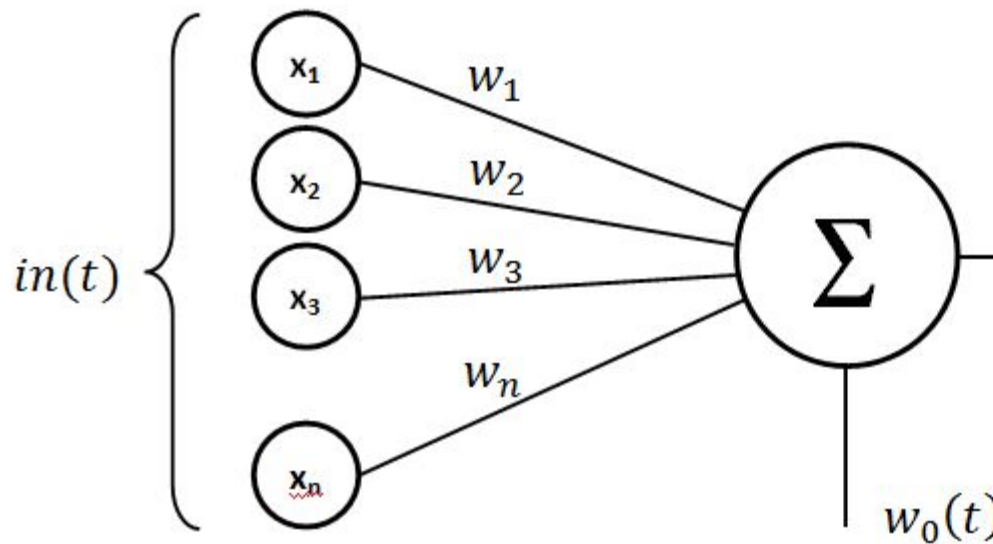
perceptron



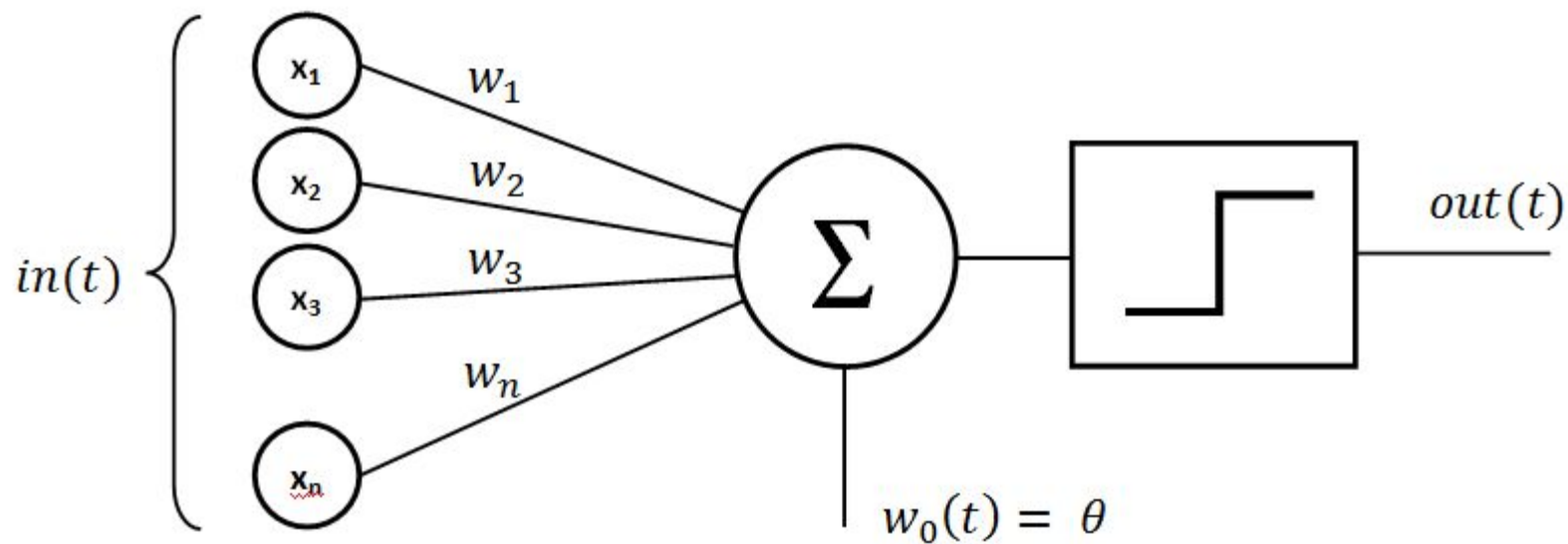
Koncepcja sieci neuronowej perceptron

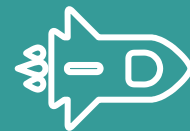


Koncepcja sieci neuronowej perceptron



Koncepcja sieci neuronowej perceptron





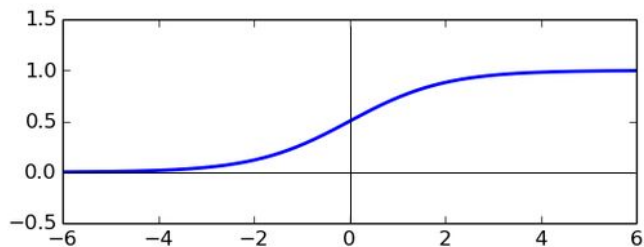
Funkcje aktywacji

Koncepcja sieci neuronowej

Funkcje aktywacji

Koncepcja sieci neuronowej

Funkcje aktywacji

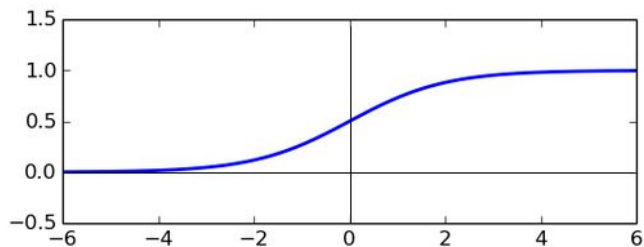


Sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

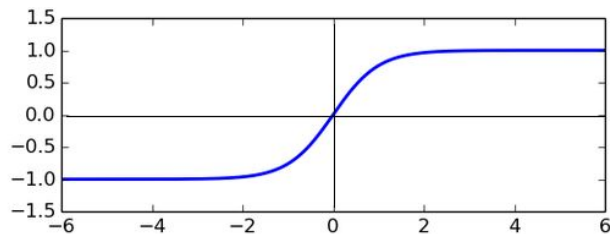
Koncepcja sieci neuronowej

Funkcje aktywacji



Sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

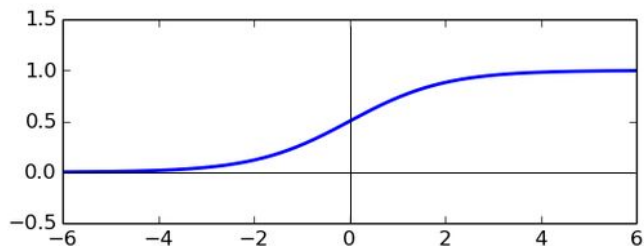


Hyperbolic Tangent

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

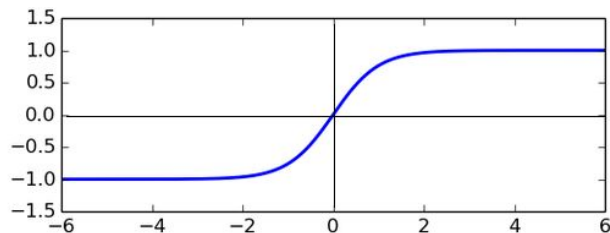
Koncepcja sieci neuronowej

Funkcje aktywacji



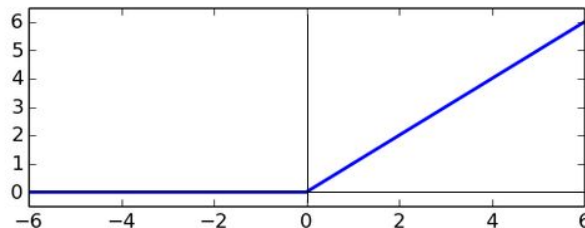
Sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



Hyperbolic Tangent

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$


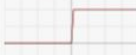









Rectified Linear

$$\phi(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$


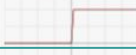







Koncepcja sieci neuronowej

Funkcje aktywacji

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Konceptcja sieci neuronowej

Funkcje aktywacji

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$



Czy jeden neuron wystarczy ?

Koncepcja sieci neuronowej

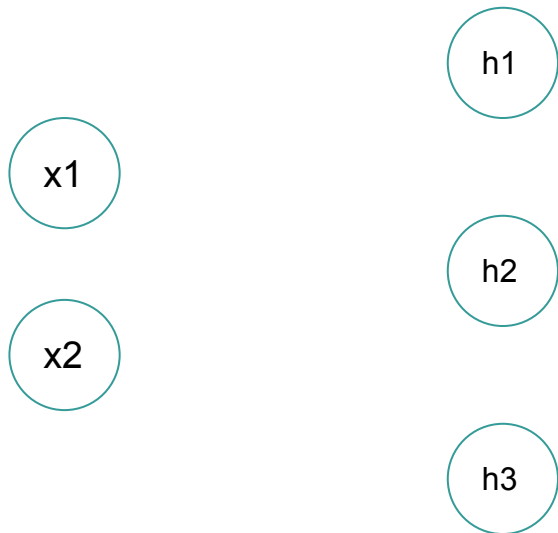
warstwy

x1

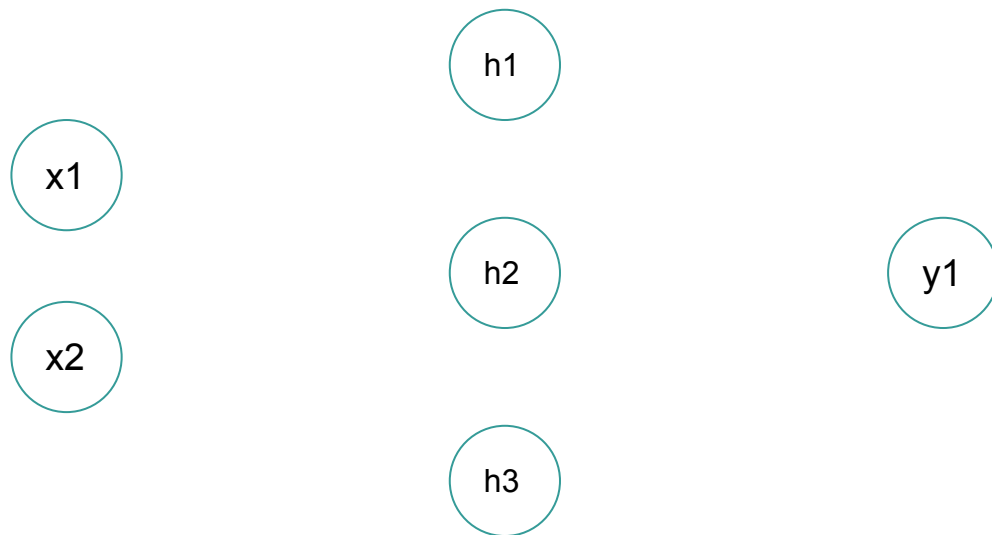
x2

Koncepcja sieci neuronowej

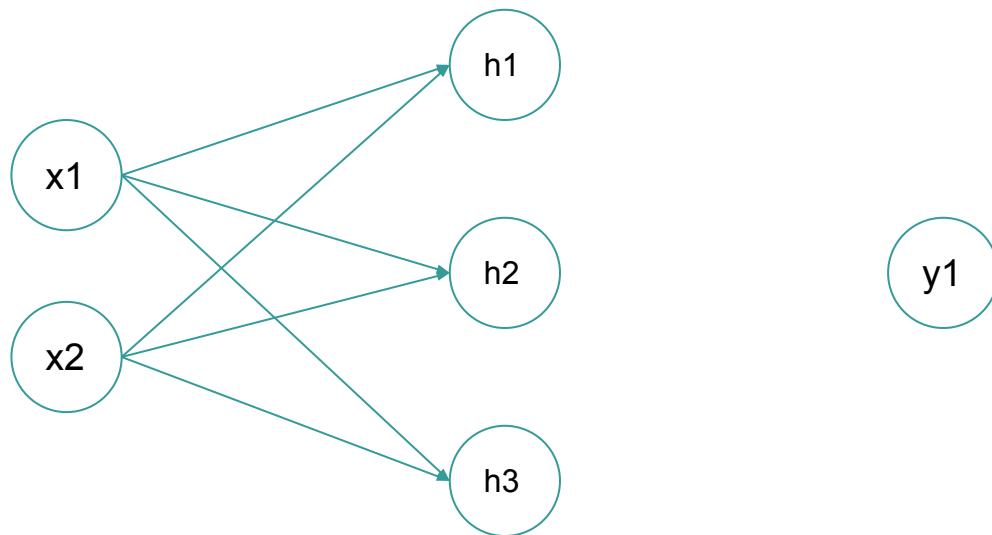
warstwy



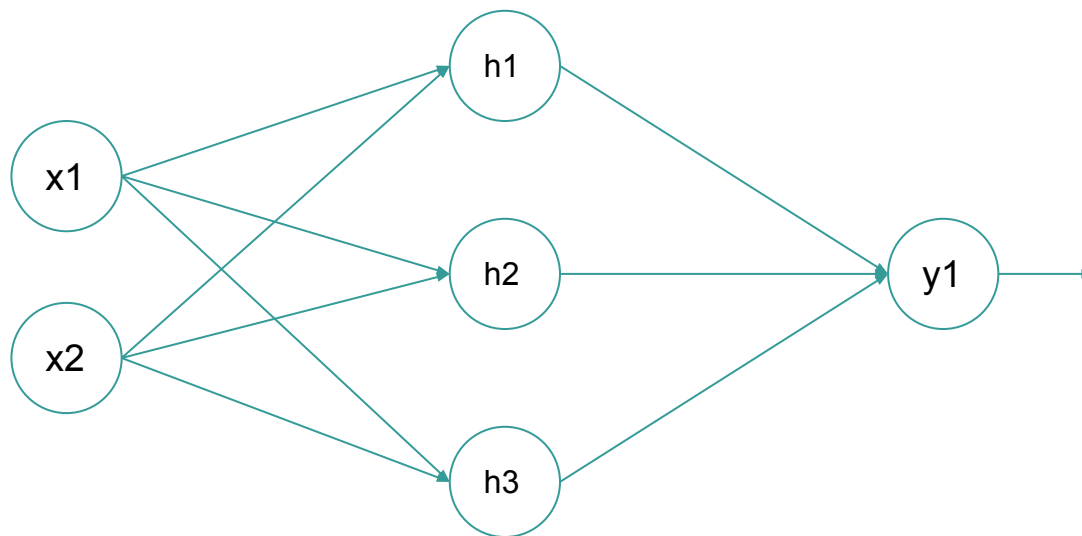
Koncepcja sieci neuronowej warstwy



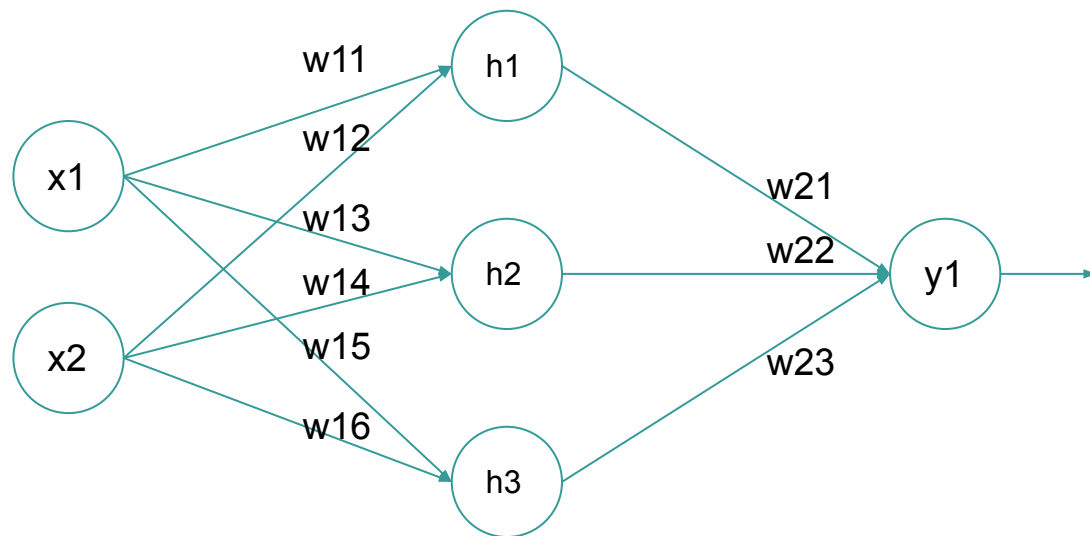
Koncepcja sieci neuronowej warstwy

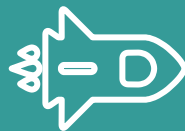


Koncepcja sieci neuronowej warstwy



Koncepcja sieci neuronowej warstwy

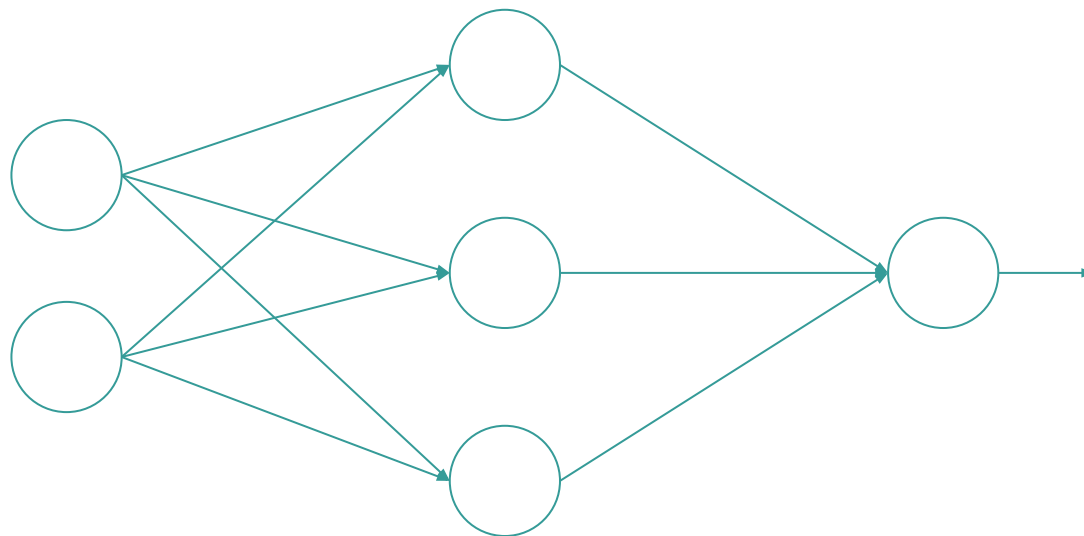




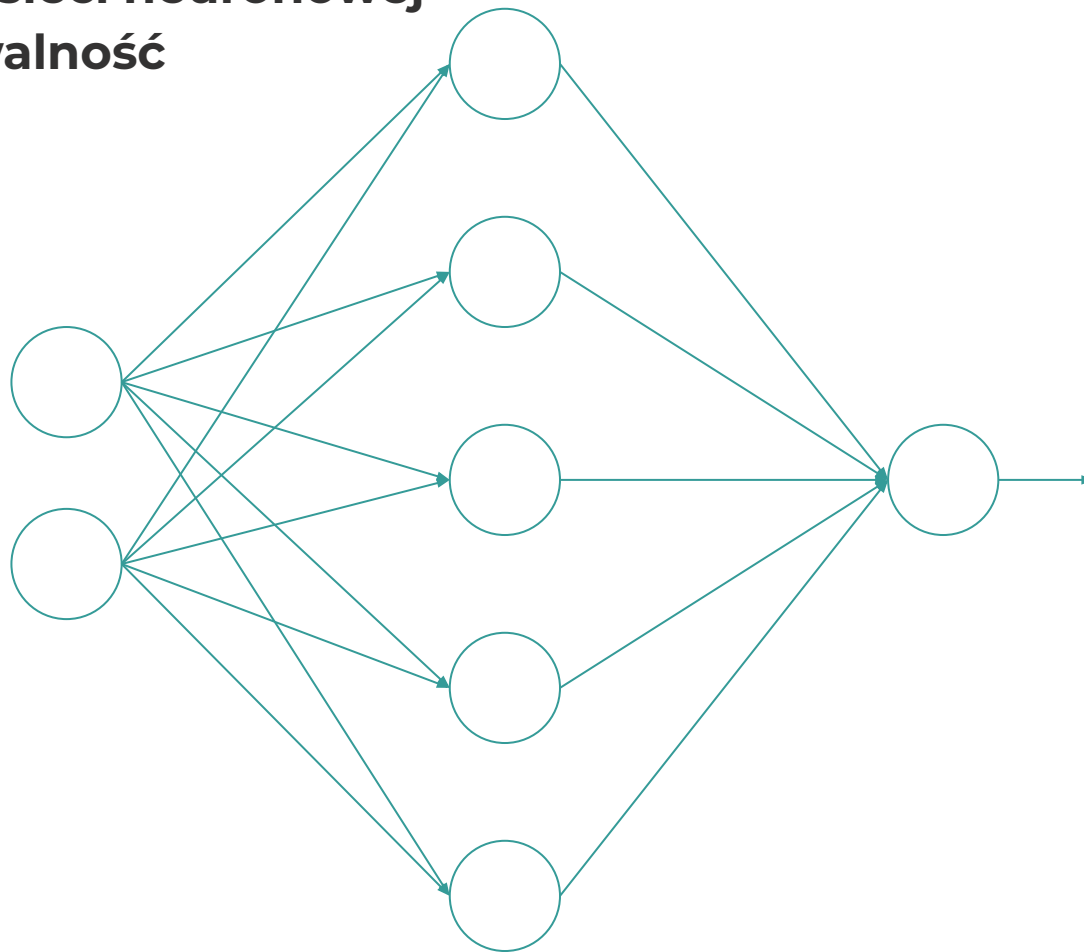
modyfikowalność

Koncepcja sieci neuronowej

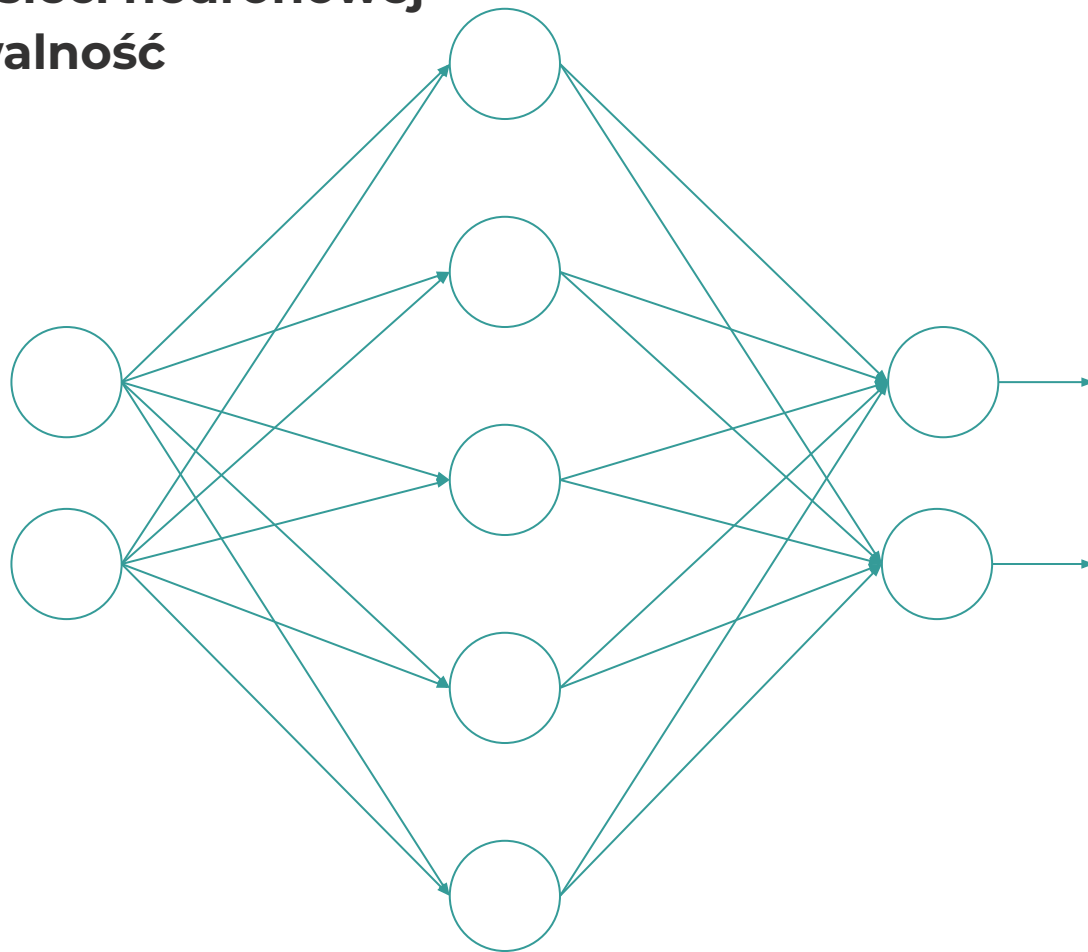
modyfikowalność



Koncepcja sieci neuronowej modyfikowalność

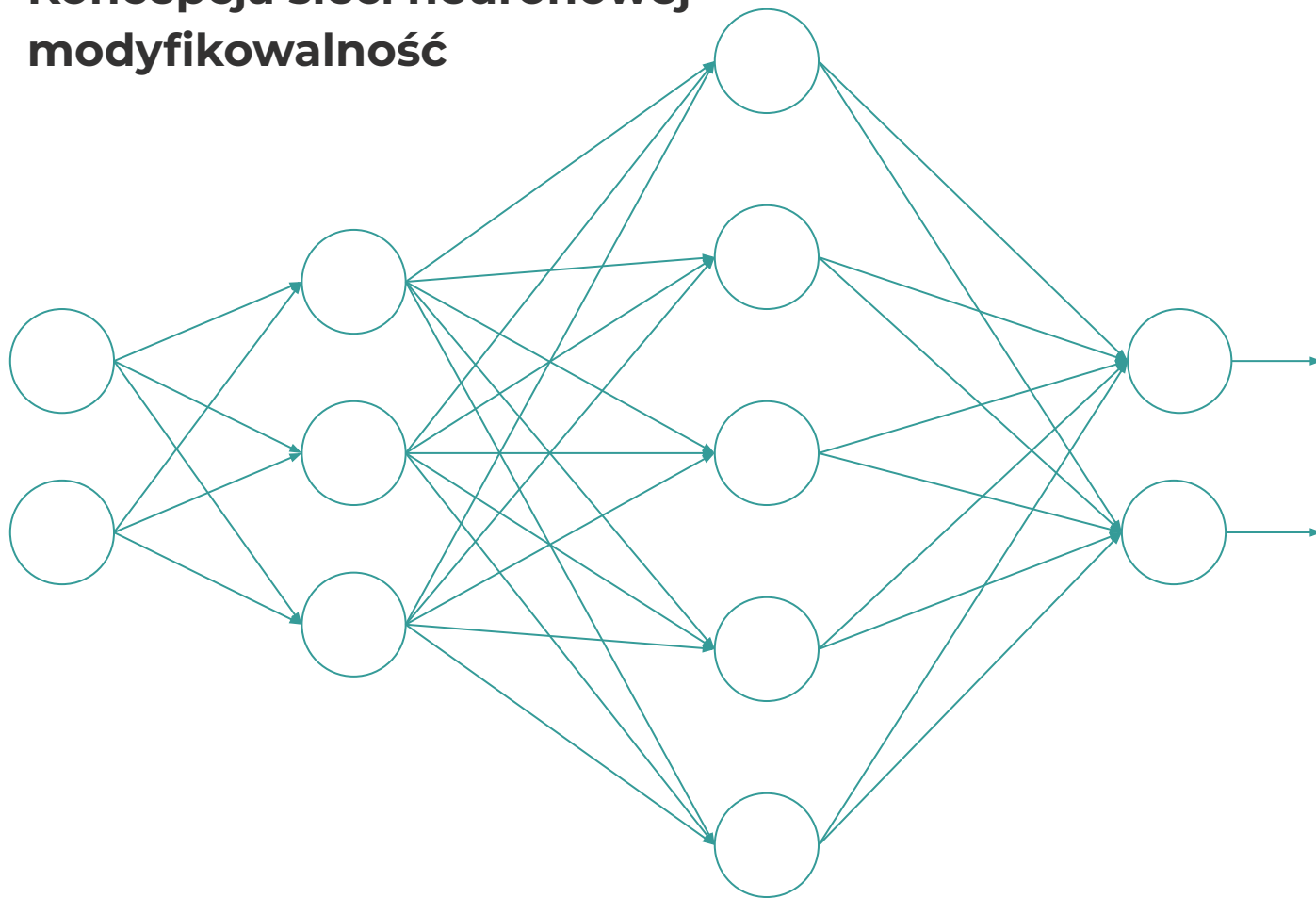


Koncepcja sieci neuronowej modyfikowalność



Koncepcja sieci neuronowej

modyfikowalność





Optymalizacja

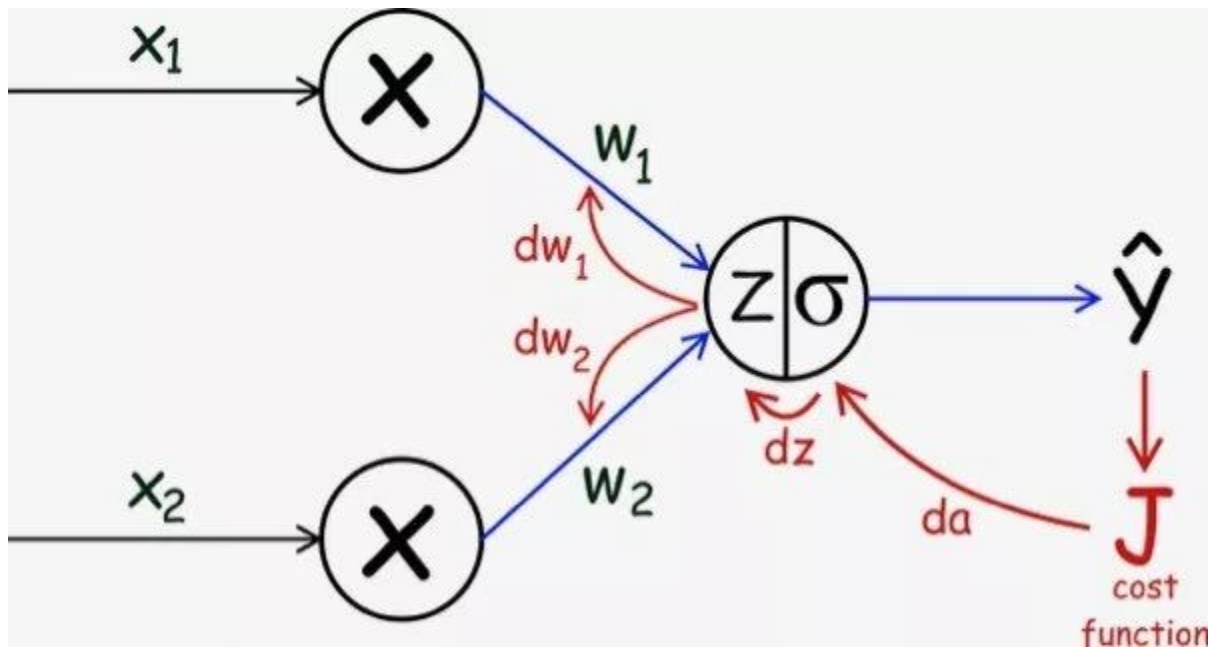
Optymalizacja

Funkcja kosztu

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

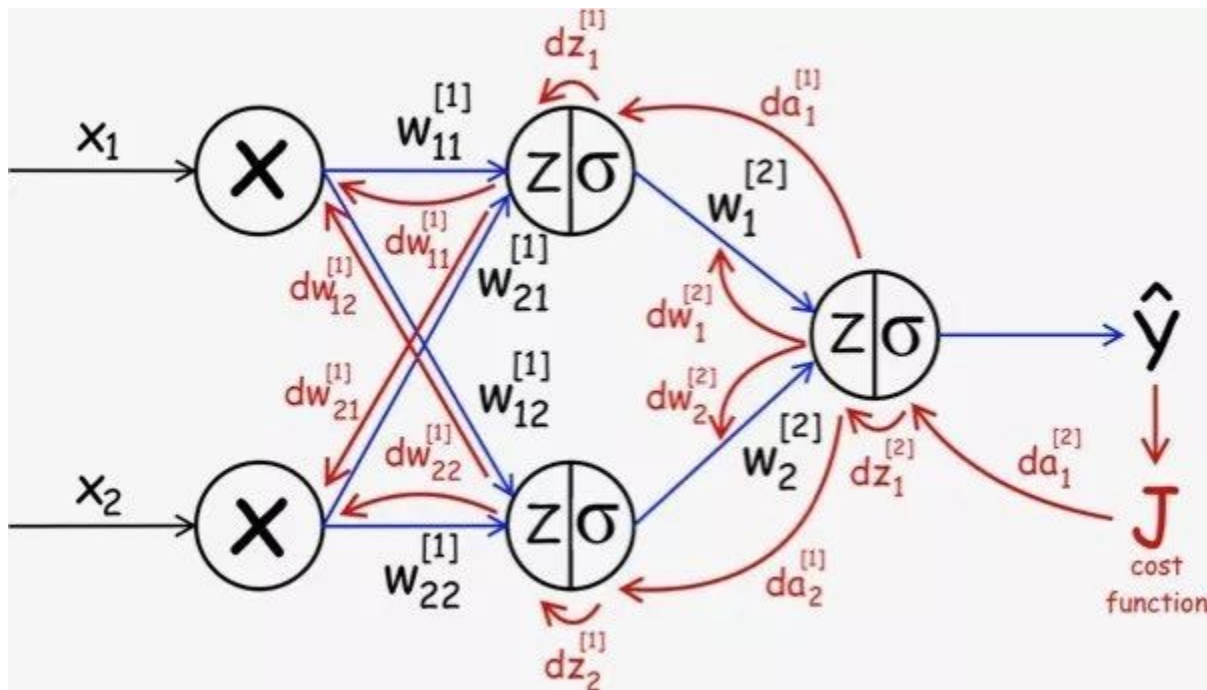
Optymalizacja

Wsteczna propagacja



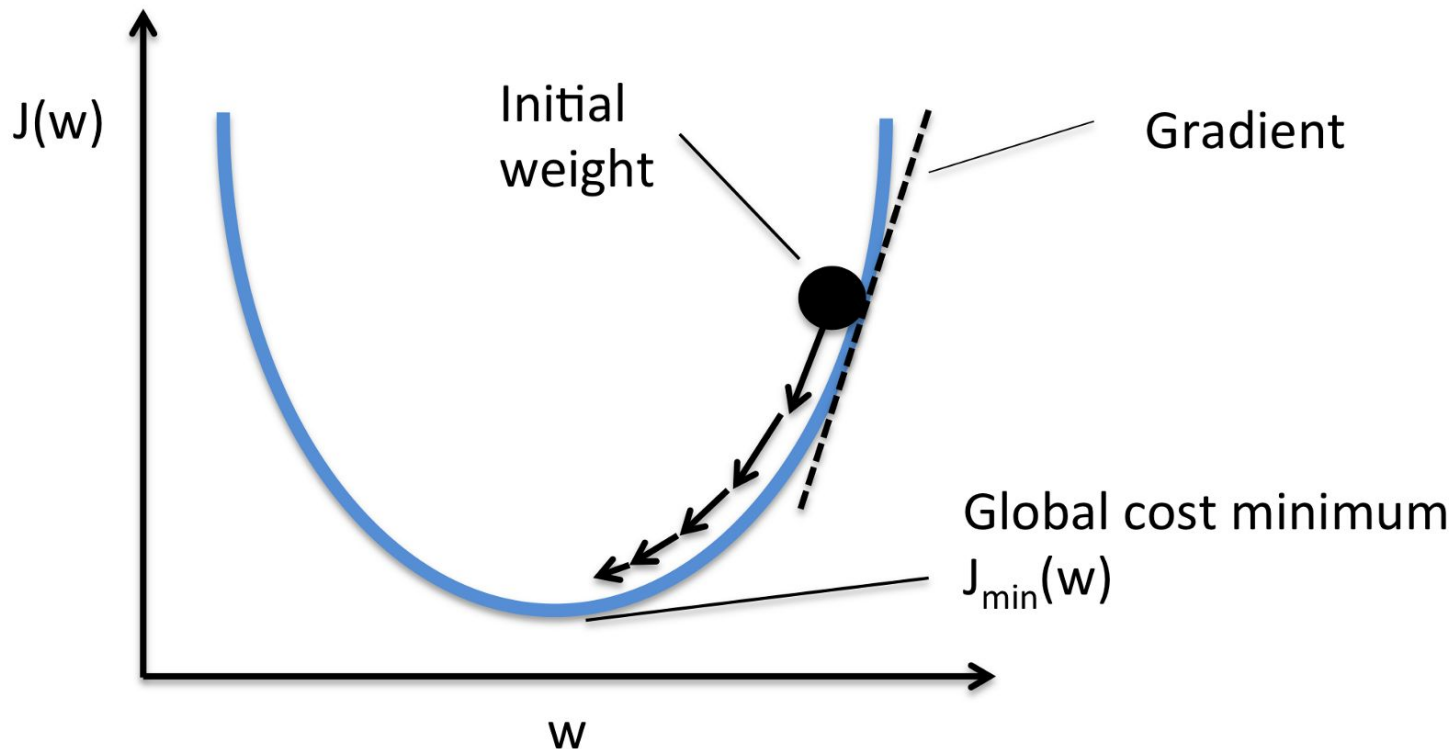
Optymalizacja

Wsteczna propagacja



Optymalizacja

Gradient descent



Optymalizacja

Korekta wag

$$W_{\text{new}} = W_{\text{curr}} - lr * (dErr/W_{\text{curr}})$$

$(dErr/W_{\text{curr}})$ – pochodna błędu po danej wadze, w tensorflow będzie to tzw. *gradient*

Optymalizacja

Wsteczna propagacja - chain rule

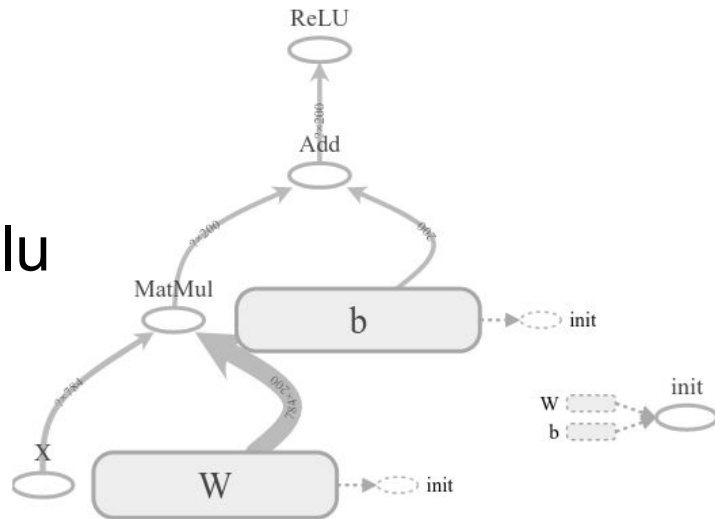
$$F(x) = f(g(x))$$

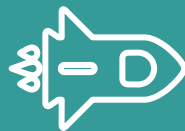
$$F'(x) = f'(g(x))g'(x)$$



tensorflow

- tworzenie modeli dl
- Bazuje na tensorach
- Tworzy z operacji graf modelu
- Zoptymalizowane pod gpu

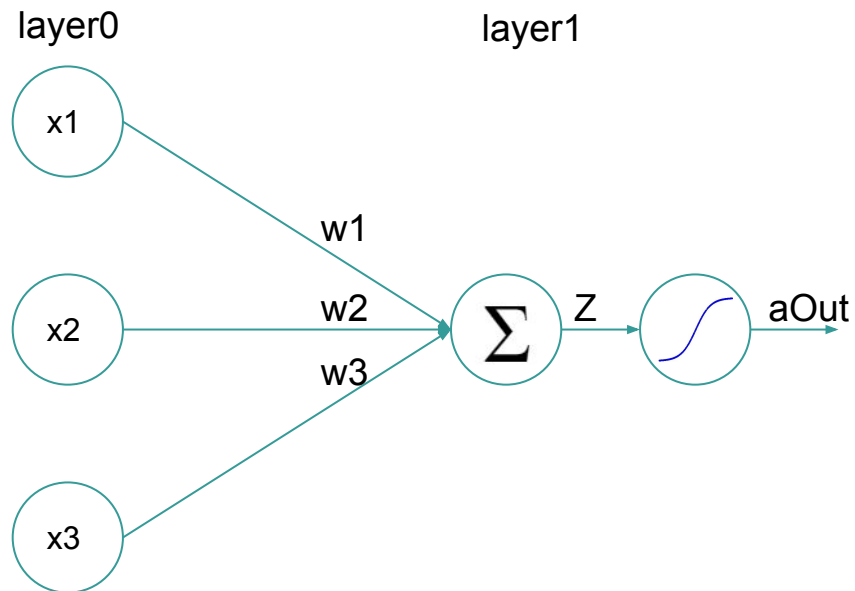




Implementacja sieci

Implementacja sieci neuronowej

Dwuwarstwowa



Implementacja sieci neuronowej Dwuwarstwowa

forward:

$$Z = input * W$$

$$y = f_{activation}(Z)$$

Implementacja sieci neuronowej Dwuwarstwowa

backward:

$$W_{new} = W_{current} - lr * \frac{dError}{dW}$$

$$\frac{dError}{dW} = \frac{dError}{dActivationOut} * \frac{dActivationOut}{dZ} * \frac{dZ}{dW}$$

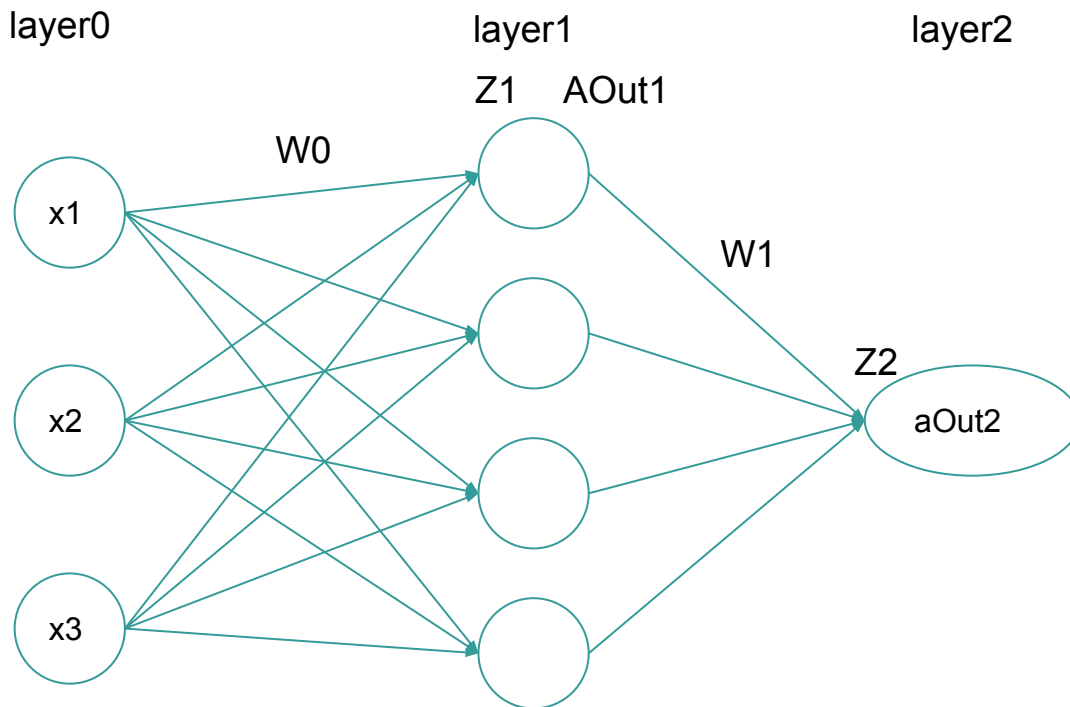
$$\frac{dError}{dActivationOut} = activationOut - y_{reference}$$

$$\frac{dActivationOut}{dZ} = ActivationFunctionDerivaive(Z)$$

$$\frac{dZ}{dW} = activationOut_{previousLayer}$$

Implementacja sieci neuronowej

Trójwarstwowa



Implementacja sieci neuronowej Trójwarstwowa

forward:

$$Z_1 = input * W_0$$

$$activationOut_1 = f_{activation}(Z_1)$$

$$Z_2 = activationOut_1 * W_1$$

$$y = f_{activation}(Z_2)$$

Implementacja sieci neuronowej Trójwarstwowa

backward:

third layer/output(2)

$$W_{1new} = W_{1current} - lr * \frac{dError}{dW_1}$$

$$\frac{dError}{dW_1} = \frac{dError}{dActivationOut_2} * \frac{dActivationOut}{dZ_2} * \frac{dZ_2}{dW_1}$$

$$\frac{dError}{dActivationOut_2} = activationOut_2 - y_{reference}$$

$$\frac{dActivationOut}{dZ_2} = ActivationFunctionDerivaive(Z_2)$$

$$\frac{dZ_2}{dW_1} = activationOut_1^T$$

Implementacja sieci neuronowej Trójwarstwowa

second layer(1)

$$W_{0new} = W_{0current} - lr * \frac{dError}{dW_0}$$

$$\frac{dError}{dW_0} = \frac{dError}{dActivationOut_1} * \frac{dActivationOut}{dZ_1} * \frac{dZ_1}{dW_0}$$

$$\frac{dError}{dActivationOut_1} = \frac{dError}{dActivationOut_2} * \frac{dActivationOut}{dZ_2} * W_1^T$$

$$\frac{dActivationOut}{dZ_1} = ActivationFunctionDerivaive(Z_1)$$

$$\frac{dZ_1}{dW_0} = input^T$$

forward:

$$Z_n = activationOut_{(n-1)} * W_{(n-1)}$$

$$activationOut_n = f_{activation}(Z_n)$$

Implementacja sieci neuronowej n-warstwowa

backward:

$$W_{(n-1) \text{ new}} = W_{(n-1) \text{ current}} - lr * \frac{dError}{dW_{(n-1)}}$$

$$\frac{dError}{dW_{(n-1)}} = \frac{dError}{dActivationOut_n} * \frac{dActivationOut}{dZ_n} * \frac{dZ_n}{dW_{(n-1)}}$$

$$\frac{dError}{dActivationOut_n} = \frac{dError}{dActivationOut_{(n+1)}} * \frac{dActivationOut}{dZ_{(n+1)}} * W_n^T$$

$$\frac{dActivationOut}{dZ_n} = ActivationFunctionDerivaive(Z_n)$$

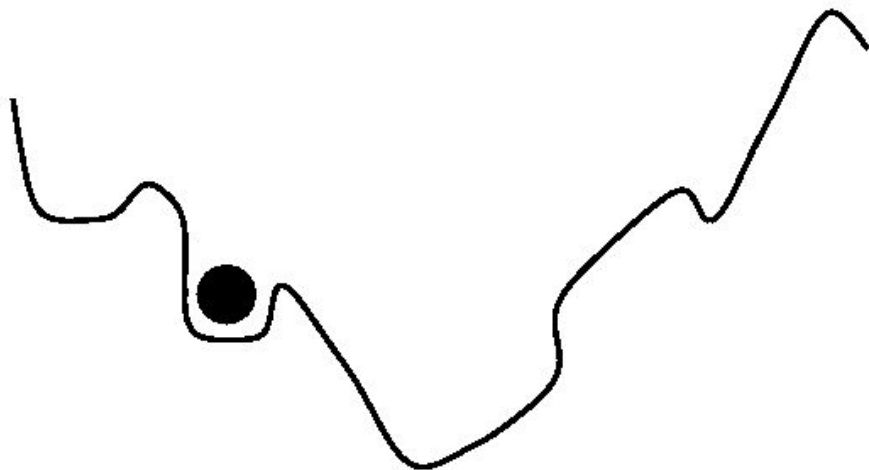
$$\frac{dZ_n}{dW_{(n-1)}} = activationOut_n^T$$



Co możemy do sieci dołączyć?

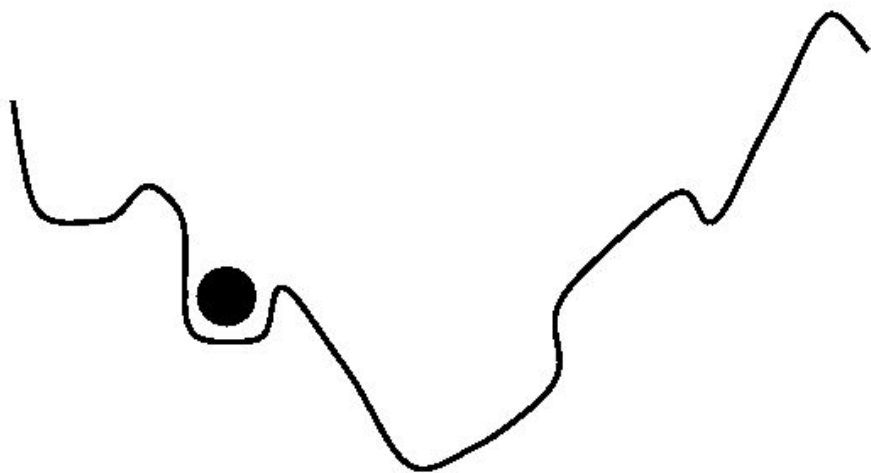
Co możemy do sieci dołączyć?

Człon momentum



Co możemy do sieci dołączyć?

Człon momentum



$$\Delta w_{ij} = \left(\eta * \frac{\partial E}{\partial w_{ij}} \right)$$

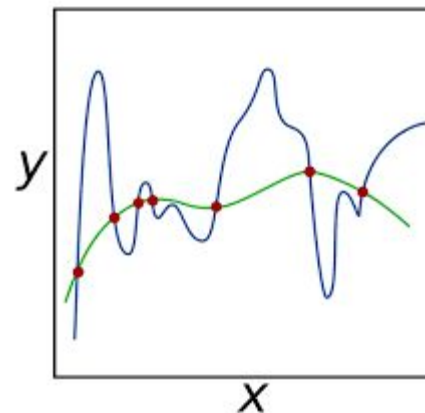
weight increment learning rate weight gradient

$$\Delta w_{ij} = \left(\eta * \frac{\partial E}{\partial w_{ij}} \right) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor weight increment, previous iteration

Co możemy do sieci dołączyć?

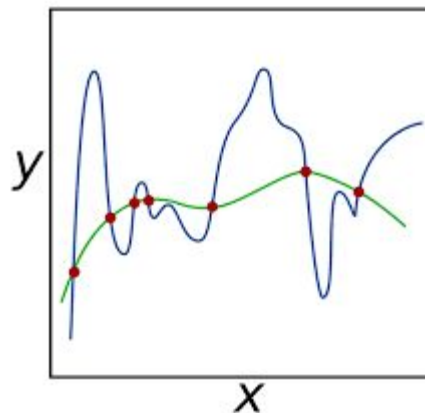
Regularyzacja



Co możemy do sieci dołączyć?

Regularyzacja

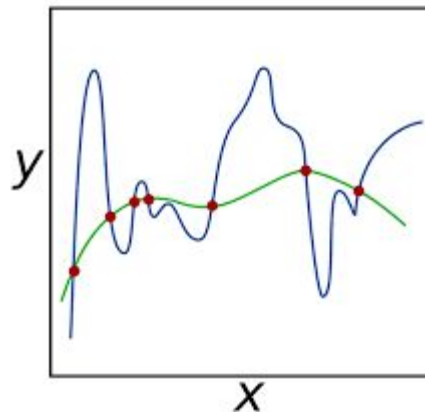
$$C = -\frac{1}{n} \sum_{x_j} \left[y_j \ln a_j^L + (1 - y_j) \ln(1 - a_j^L) \right] + \frac{\lambda}{2n} \sum_w w^2.$$



Co możemy do sieci dołączyć?

Regularyzacja

$$C = -\frac{1}{n} \sum_{x_j} \left[y_j \ln a_j^L + (1 - y_j) \ln(1 - a_j^L) \right] + \frac{\lambda}{2n} \sum_w w^2$$



Zalety

- Teoretycznie do każdego problemu można dostosować odpowiednią sieć neuronową
- Radzi sobie z dużą ilością danych
- Dowolne wejście/wyjście
- Skuteczne zarówno dla regresji jak i klasyfikacji

Wady

- Zanikający gradient
- Eksplodujący gradient
- Złożoność obliczeniowa
- wielkość

Źródła

- <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>
- <http://www.emergentmind.com/neural-network>
- <https://www.youtube.com/watch?v=llg3gGewQ5U>
- https://en.wikipedia.org/wiki/Chain_rule
- <https://cloud.google.com/blog/products/gcp/understanding-neural-networks-with-tensorflow-playground>
- <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>
- <http://www.adeveloperdiary.com/data-science/machine-learning/understand-and-implement-the-backpropagation-algorithm-from-scratch-in-python/>



Dzięki

You can find me at
[@Krzysztof Bork-Ceszlak](#) & ceszlak.krzysztof@gmail.com