



Wstęp do uczenia głębokiego

Paulina Tomaszewska

Artificial Intelligence

AI involves techniques that equip computers to emulate human behavior, enabling them to learn, make decisions, recognize patterns, and solve complex problems in a manner akin to human intelligence.

Machine Learning

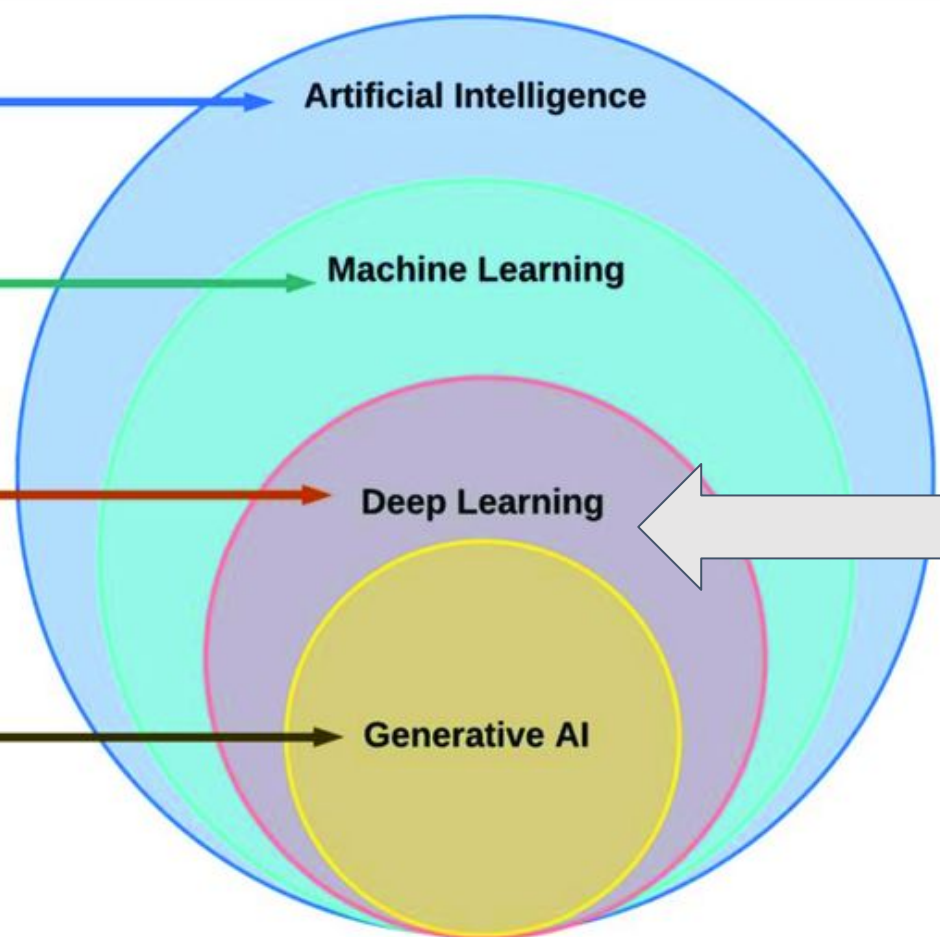
ML is a subset of AI, uses advanced algorithms to detect patterns in large data sets, allowing machines to learn and adapt. ML algorithms use supervised or unsupervised learning methods.

Deep Learning

DL is a subset of ML which uses neural networks for in-depth data processing and analytical tasks. DL leverages multiple layers of artificial neural networks to extract high-level features from raw input data, simulating the way human brains perceive and understand the world.

Generative AI

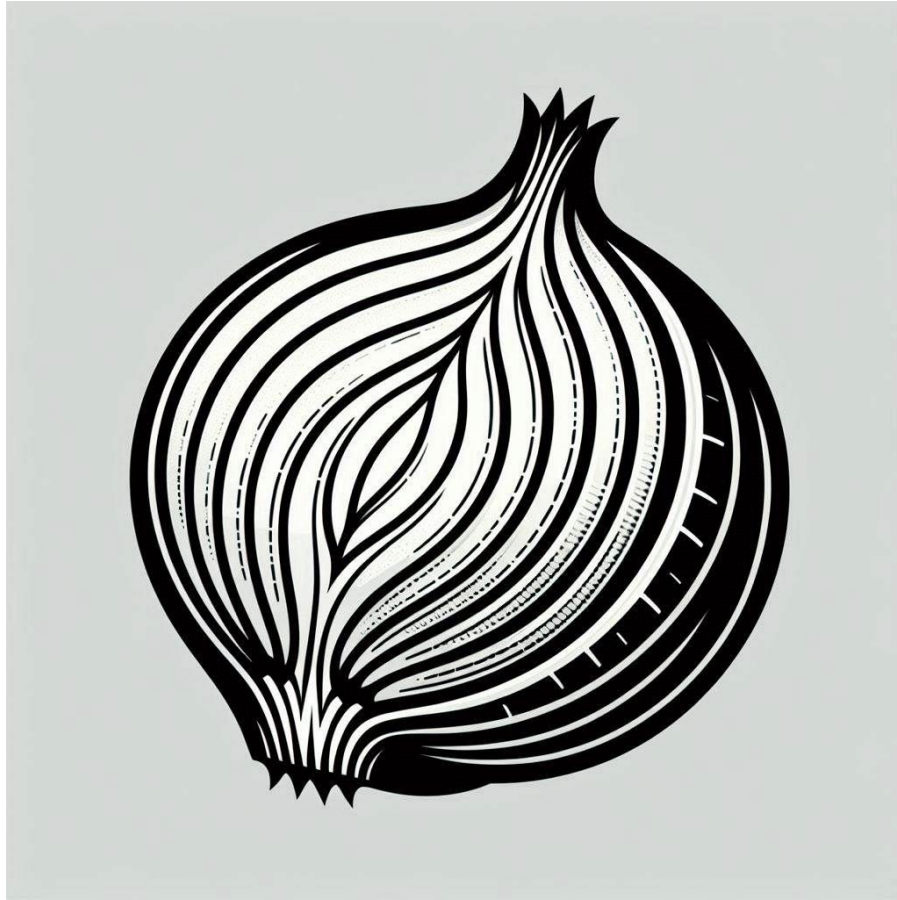
Generative AI is a subset of DL models that generates content like text, images, or code based on provided input. Trained on vast data sets, these models detect patterns and create outputs without explicit instruction, using a mix of supervised and unsupervised learning.



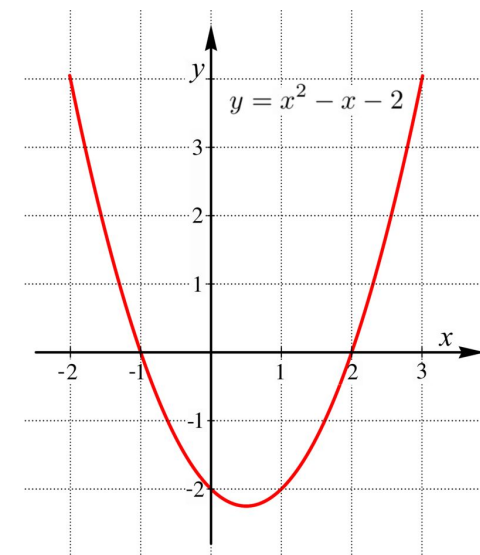
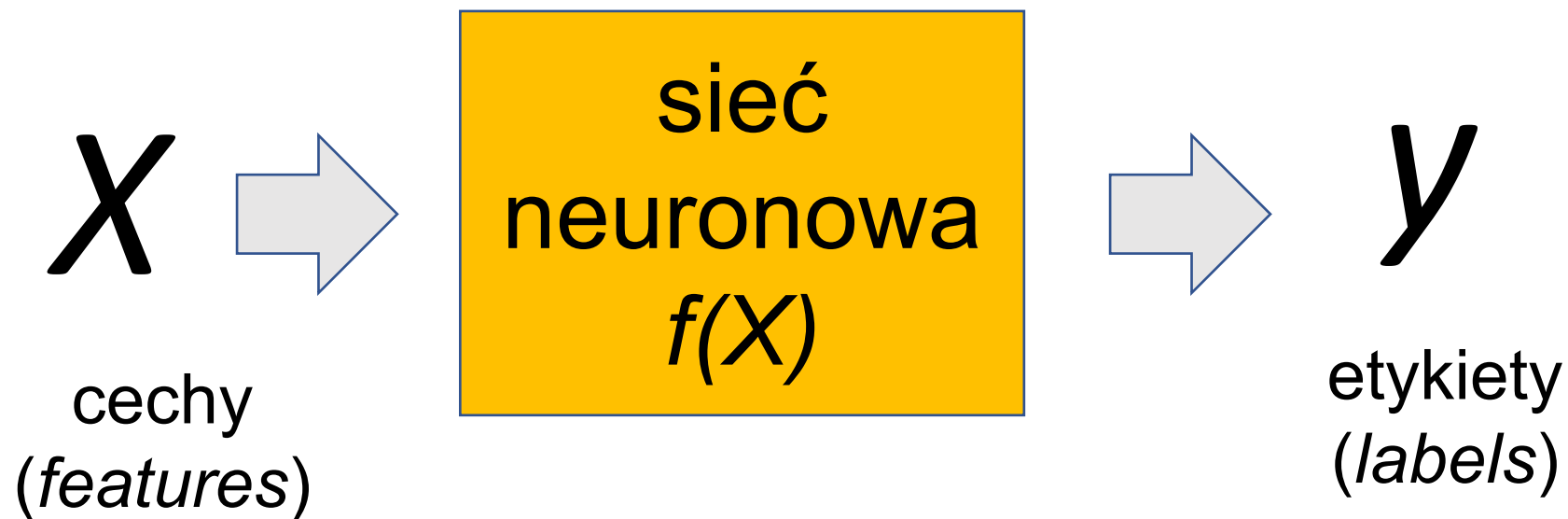
dzisiaj będzie
wstęp do
tego obszaru

Sztuczne sieci neuronowe, uczenie głębokie (*Deep Learning*)

Niektórzy mówią, że sieci neuronowe są podobne do cebuli. Dlaczego?
Zarówno sieci neuronowe jak i cebule mają warstwy.

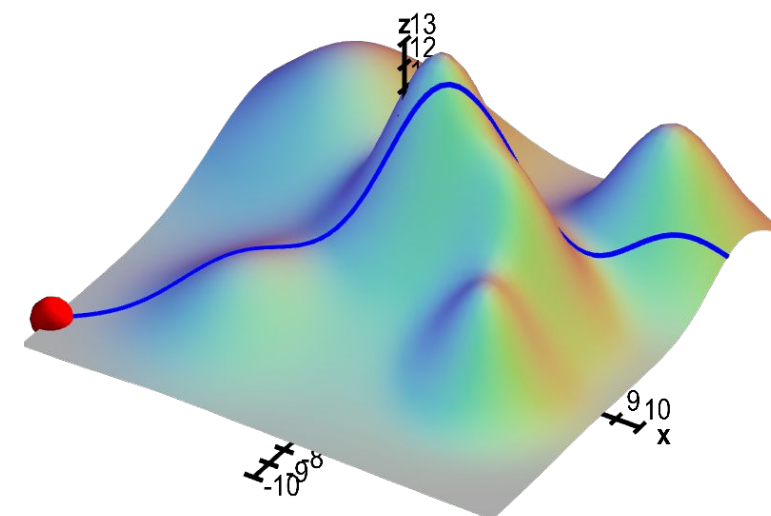


wygenerowane z użyciem DALL- E 3



https://pl.wikipedia.org/wiki/Funkcja_kwadratowa

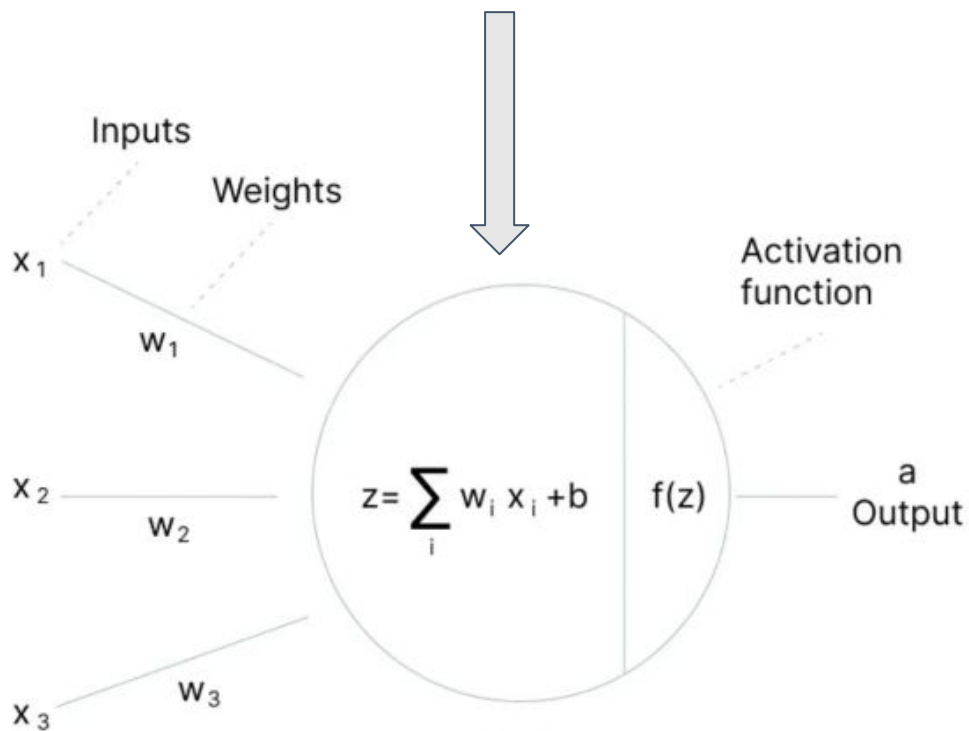
Sieć neuronowa (*neural network*, *NN*) jest zazwyczaj bardzo skomplikowaną funkcją z wieloma nieznanymi parametrami (taką funkcję nazywać też będziemy modelem). Celem jest wytrenować model tak, że jego wyjście, $f(X)$, będzie jak najbliższe wartości prawdziwej, y .



https://mathinsight.org/chain_rule_multivariable_introduction

**Sztuczne sieci neuronowe
składają się ze sztucznych
neuronów**

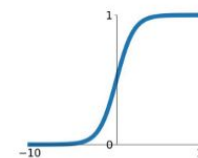
Sztuczny neuron (zaproponowany w 1943)



Funkcje aktywacji

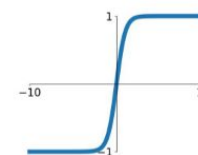
Sigmoid

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



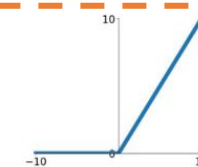
tanh

$$\tanh(x)$$



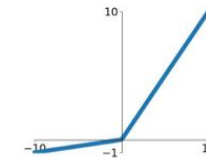
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

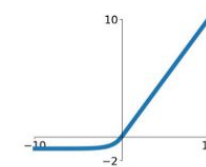


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



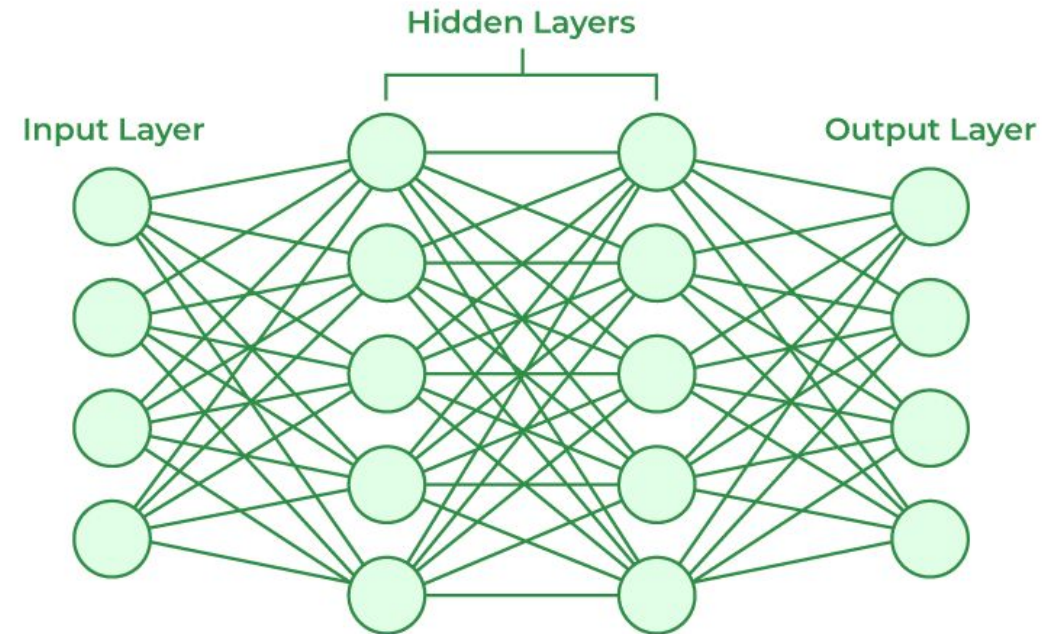
W neuronie liczymy sumę iloczynów wejściowych wartości (x_i) i wag (w_i), do czego dodajemy potem wyraz wolny (b). Wyliczona wartość jest wejściem do funkcji aktywacji.

Najprostsza sieć neuronowa - perceptron wielowarstwowy (*multilayer perceptron, MLP*)

- hierarchiczna struktura uporządkowanych neuronów

Pytania:

- ile powinno być warstw?
- ile powinno być neuronów w jednej warstwie?



nie ma prostej reguły

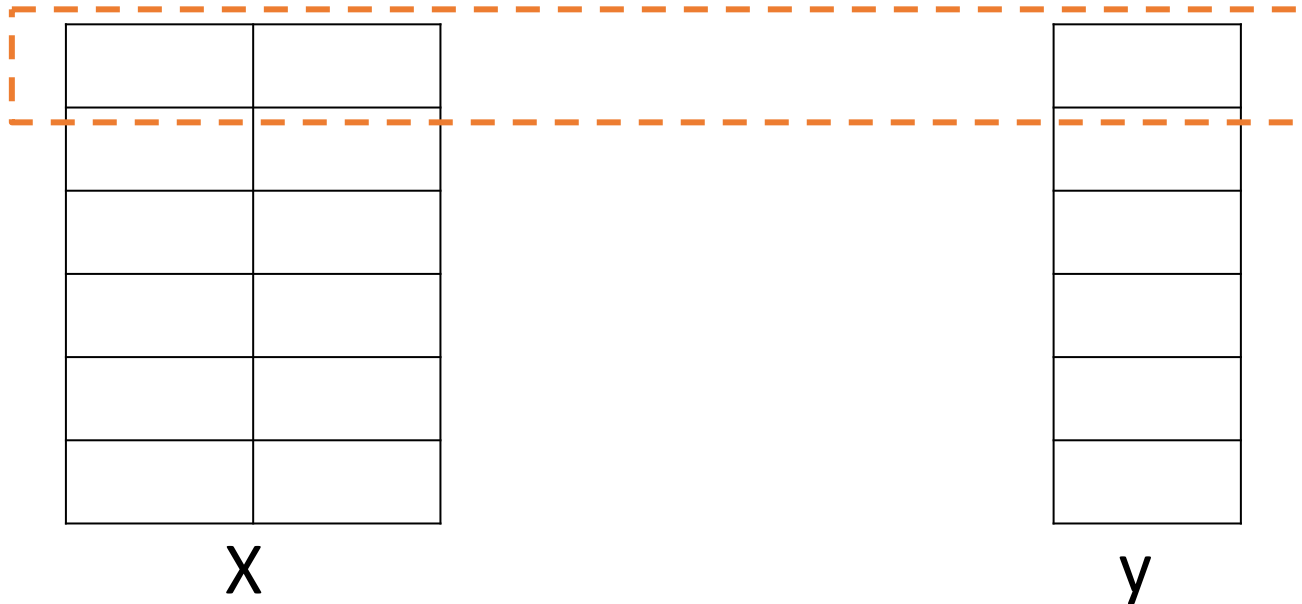
Uproszczony przykład:

**przewidzieć temperaturę
w kolejnym dniu**

Mamy dane z przeszłości:

X - temperatura w dniu dzisiejszym, poziom wilgotności

y - temperatura w dniu kolejnym



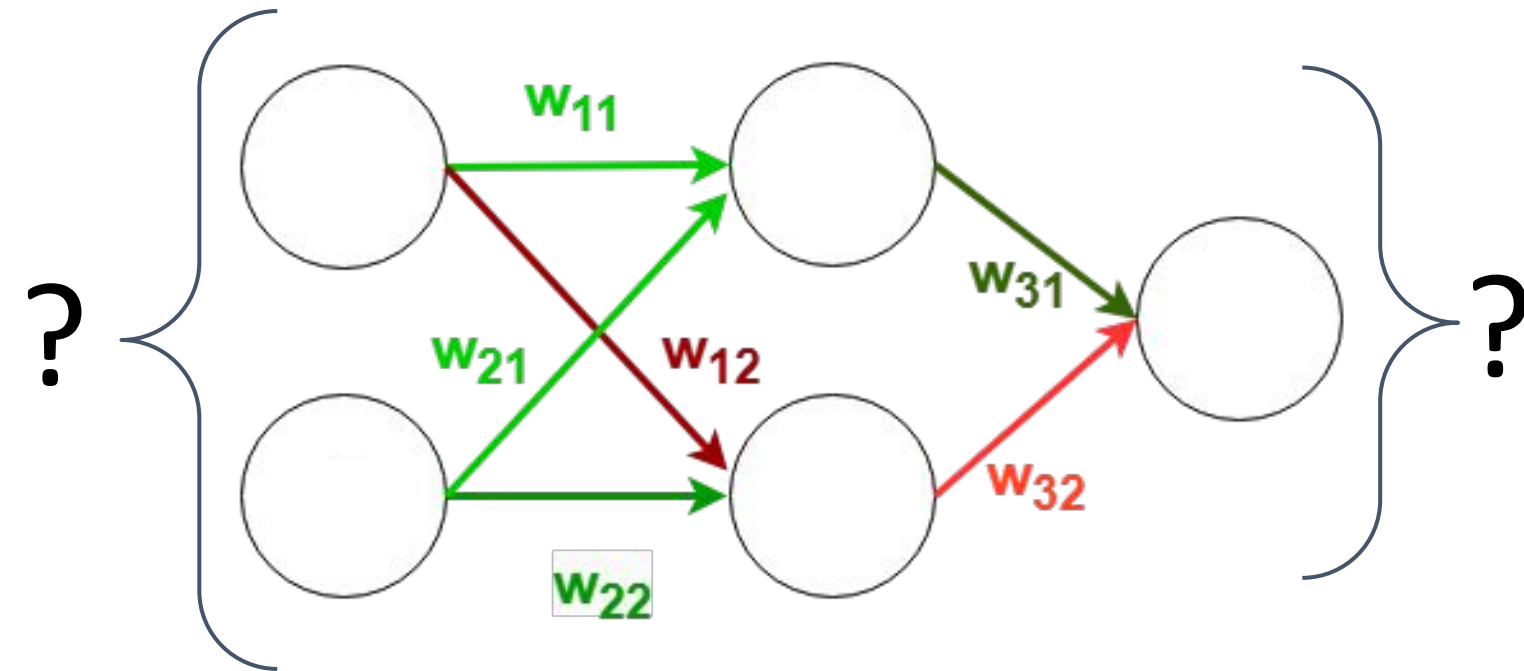
jeden rekord/obserwacja

Użyjmy sieci neuronowych

Uwaga: W praktyce, rzadko używa się sieci neuronowych do analizy tak małych danych tabelarycznych
(to jest przykład na potrzeby edukacyjne, aby wyjaśnić w łatwy sposób działanie sieci neuronowych)

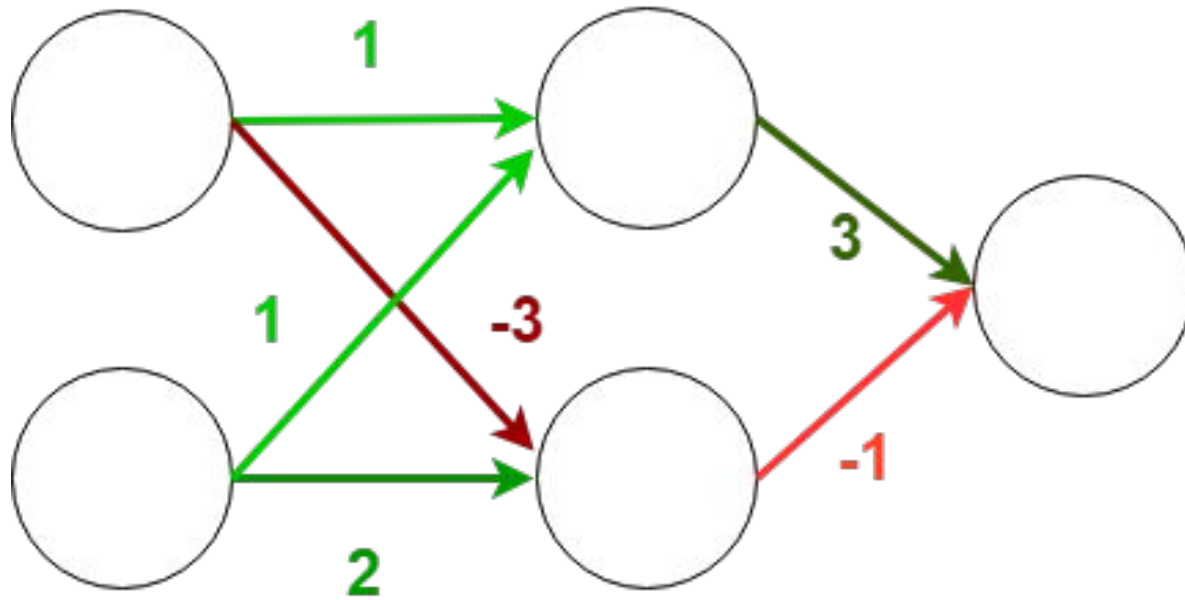
Krok 0: Zdefiniuj architekturę sieci neuronowej

Czy wiemy ile neuronów powinno być w warstwie wejściowej i wyjściowej?



- w pierwszej warstwie jest tyle neuronów ile wartości w pojedynczym rekordzie w danych wejściowych X
- w ostatniej warstwie jest tyle neuronów ile wartości chcemy przewidzieć - pojedynczy rekord w tabeli y

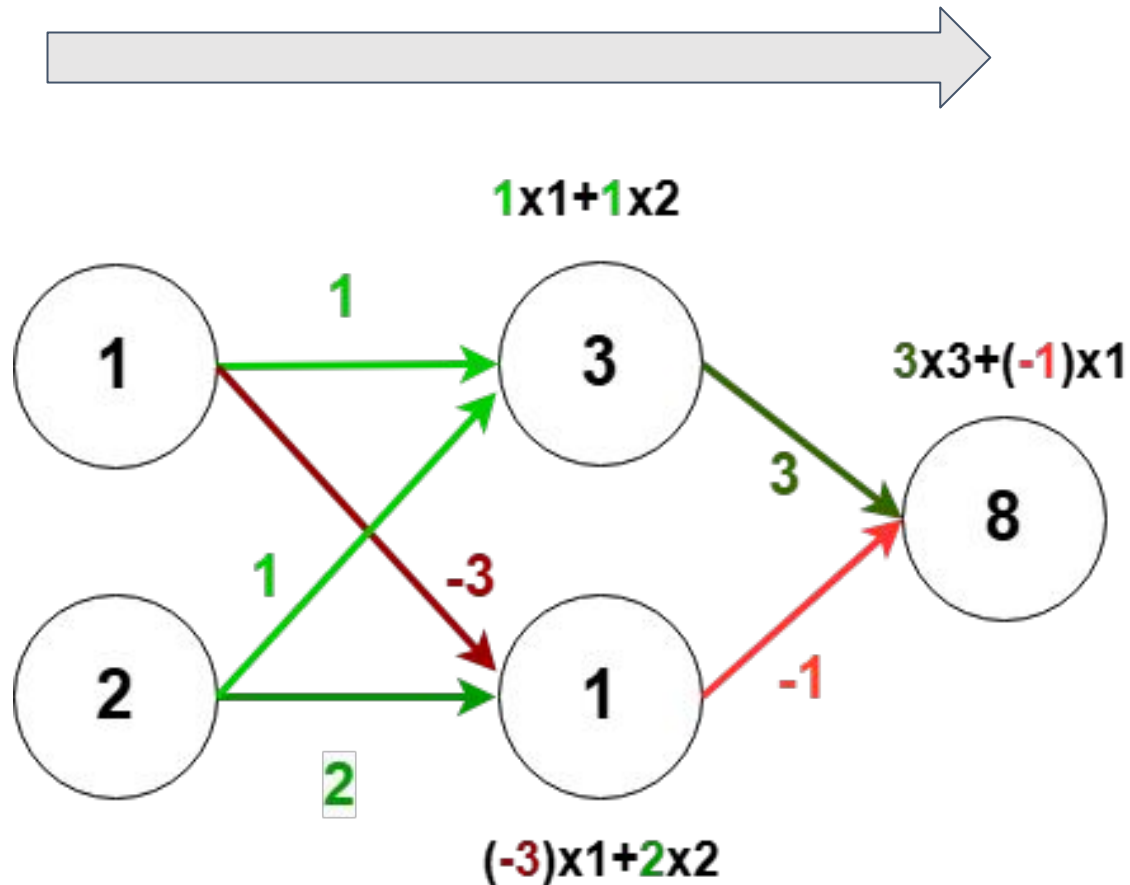
Krok 1:
Zainicjalizuj wartości wag
(ustal wartości początkowe)



jak?

- losowo
- często korzysta się z inicjalizacji metodą Xavier

Krok 2: Obliczenia w przód, inferencja (*forward pass, inference*)



Używamy cech (X) z jako wejście do modelu, aby otrzymać wyjście modelu (predykcję).

Założenia upraszczające:

- $b=0$
- funkcja aktywacji *ReLU*

Krok 3:

Sprawdź jak “dobre” jest wyjście modelu

Wyjście modelu to 8, ale prawidłowa wartość (etykieta) to 9.

Jaki jest błąd modelu? Policzymy go z użyciem **funkcji kosztu** (*cost function*).

Intuicyjnie:

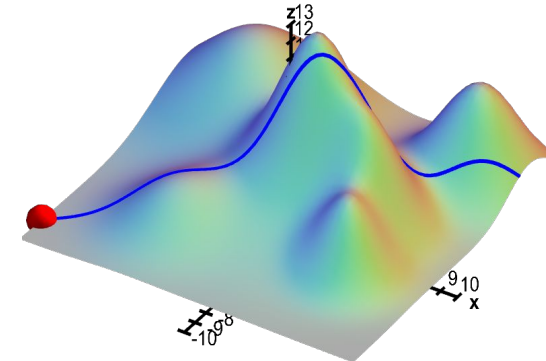
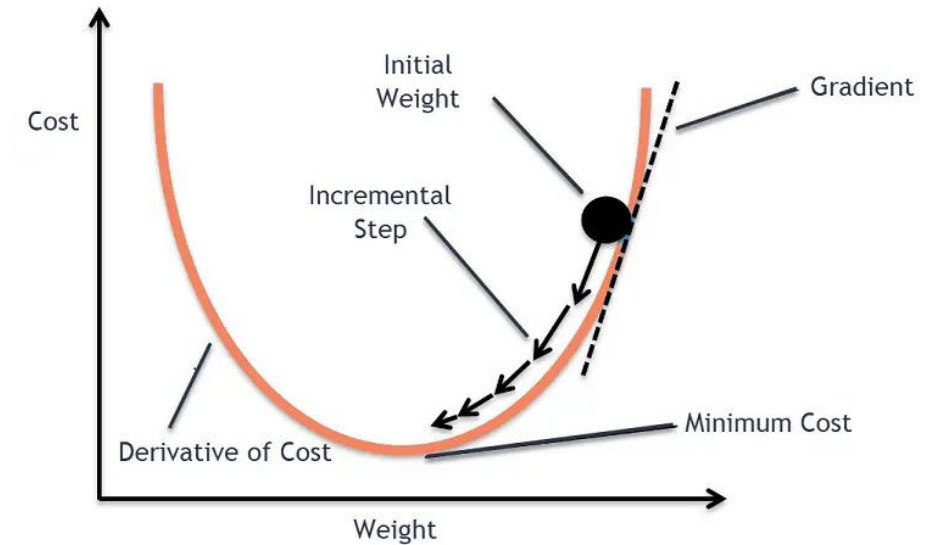
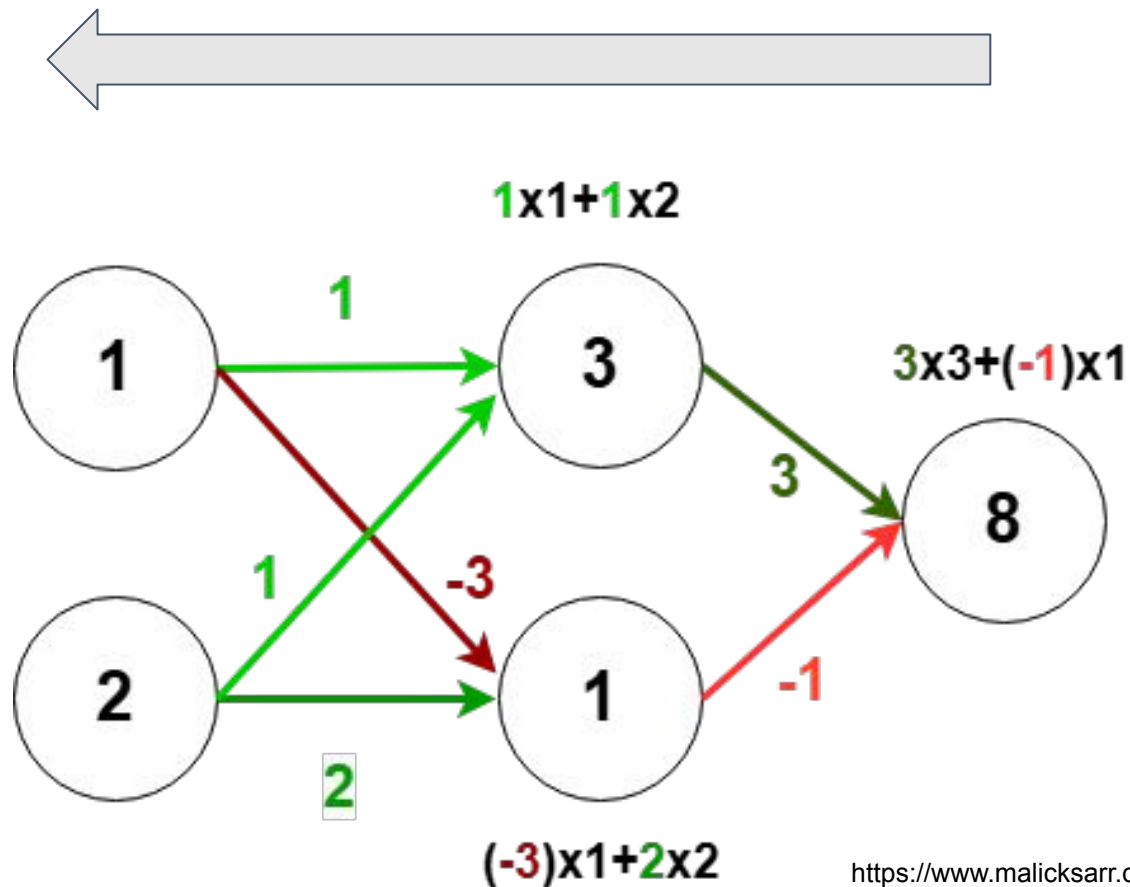
$$\text{błąd} = \text{wartość przewidziana} - \text{etykieta} = 8 - 9 = -1$$

W praktyce, w problemach regresji zazwyczaj używa się definicji błędu średniokwadratowego (Mean Square Error, MSE).

$$\text{błąd} = \frac{1}{2} (\text{wartość przewidziana} - \text{etykieta})^2 = \frac{1}{2} (8-9)^2 = \frac{1}{2}$$

Krok 4: Propagacja wsteczna (*backpropagation*)

Liczymy gradient funkcji kosztu
względem wag modelu.



Krok 5: Zaktualizuj wagi modelu

Wyjścia modelu nie są poprawne, bo ma niewłaściwe wagi, które było początkowo losowo zainicjalizowane. Chcemy zmienić wagi, tak aby model dawał wartości bliższe etykiетom.

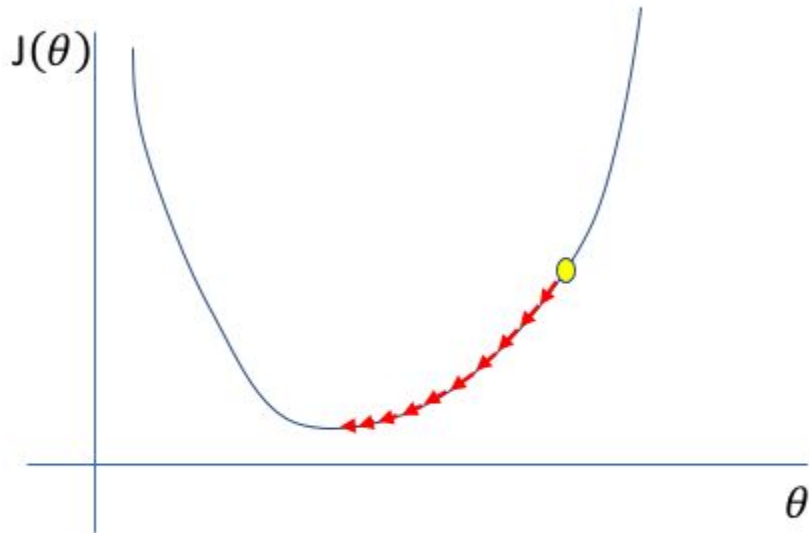
$$W_{t+1} = W_t - lr * gradient(błąd)$$

lr - learning rate

(parametr odpowiedzialny za to jak bardzo zmieniamy wartości, a więc na szybkość treningu)

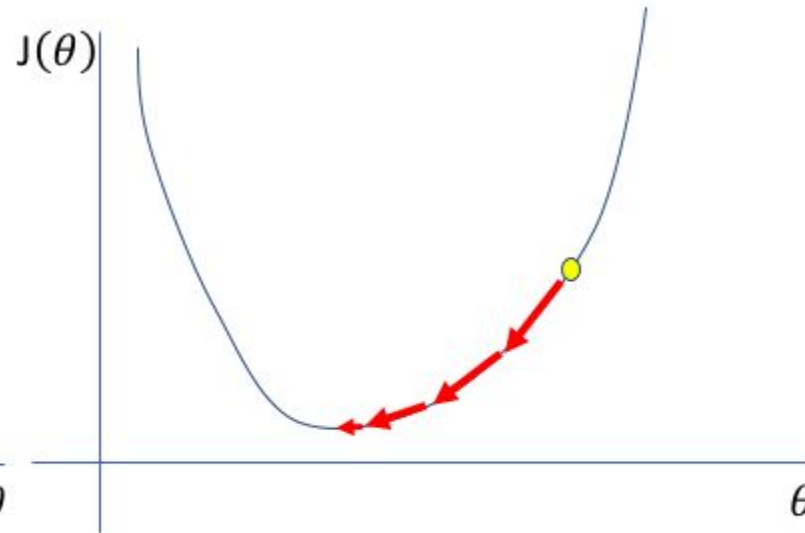
Jak wartość *learning rate* wpływa na proces treningu sieci neuronowej?

Too low



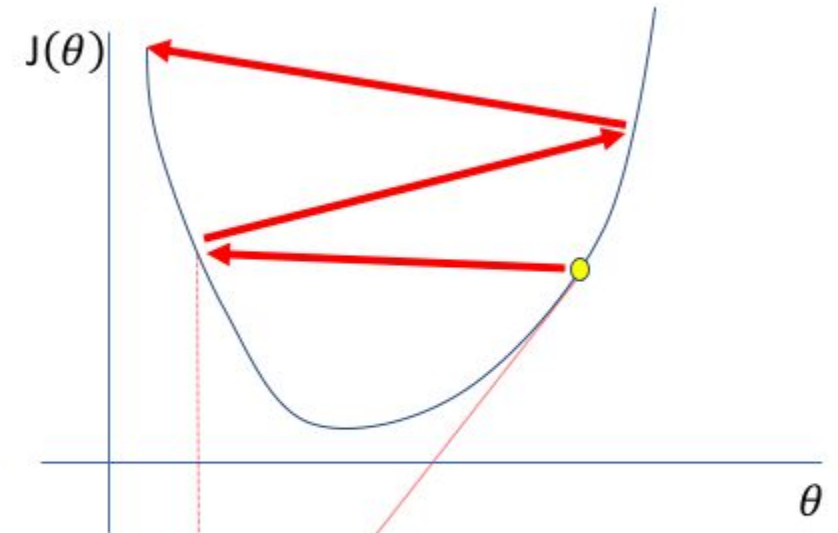
A small learning rate requires many updates before reaching the minimum point

Just right



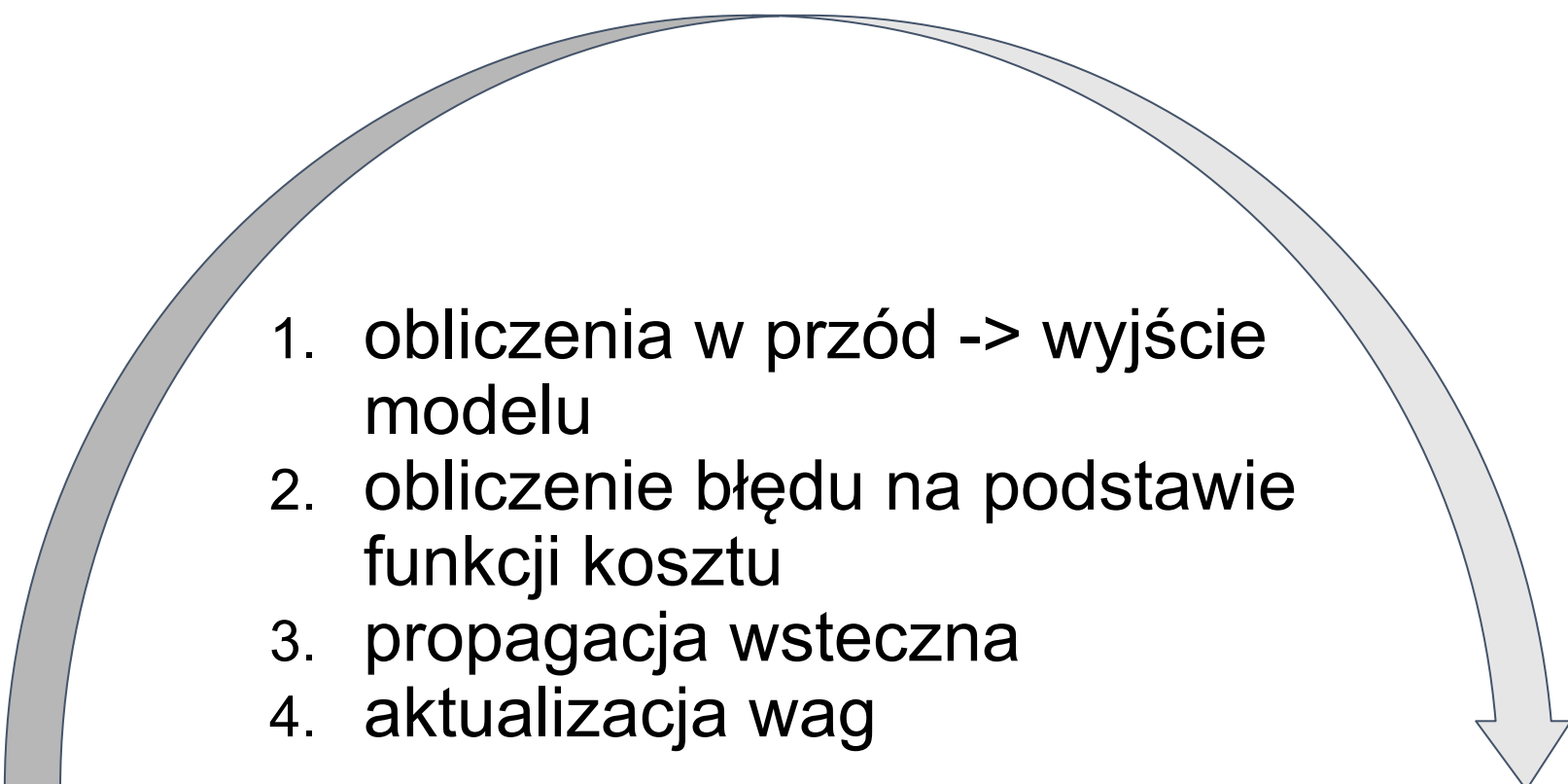
The optimal learning rate swiftly reaches the minimum point

Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

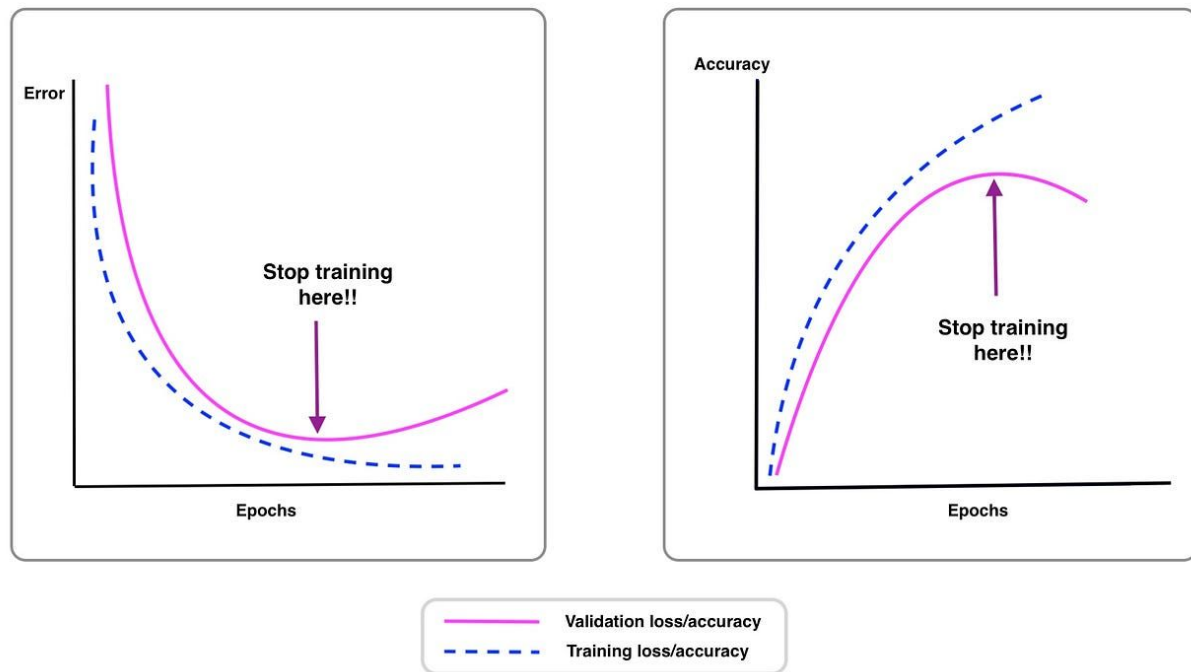
Jakie są kolejne kroki?

- 
1. obliczenia w przód -> wyjście modelu
 2. obliczenie błędu na podstawie funkcji kosztu
 3. propagacja wsteczna
 4. aktualizacja wag

Te kroki wykonujemy dla wszystkich “paczek” danych (*batches*). Gdy wszystkie dane treningowe zostaną użyte, mówimy, że epoka się kończy i zaczyna się kolejna (jest ich tyle ile ustawimy).

W praktyce, aby wykonać te kroki korzystamy z gotowych funkcji w pakietach takich jak np. *pytorch*.

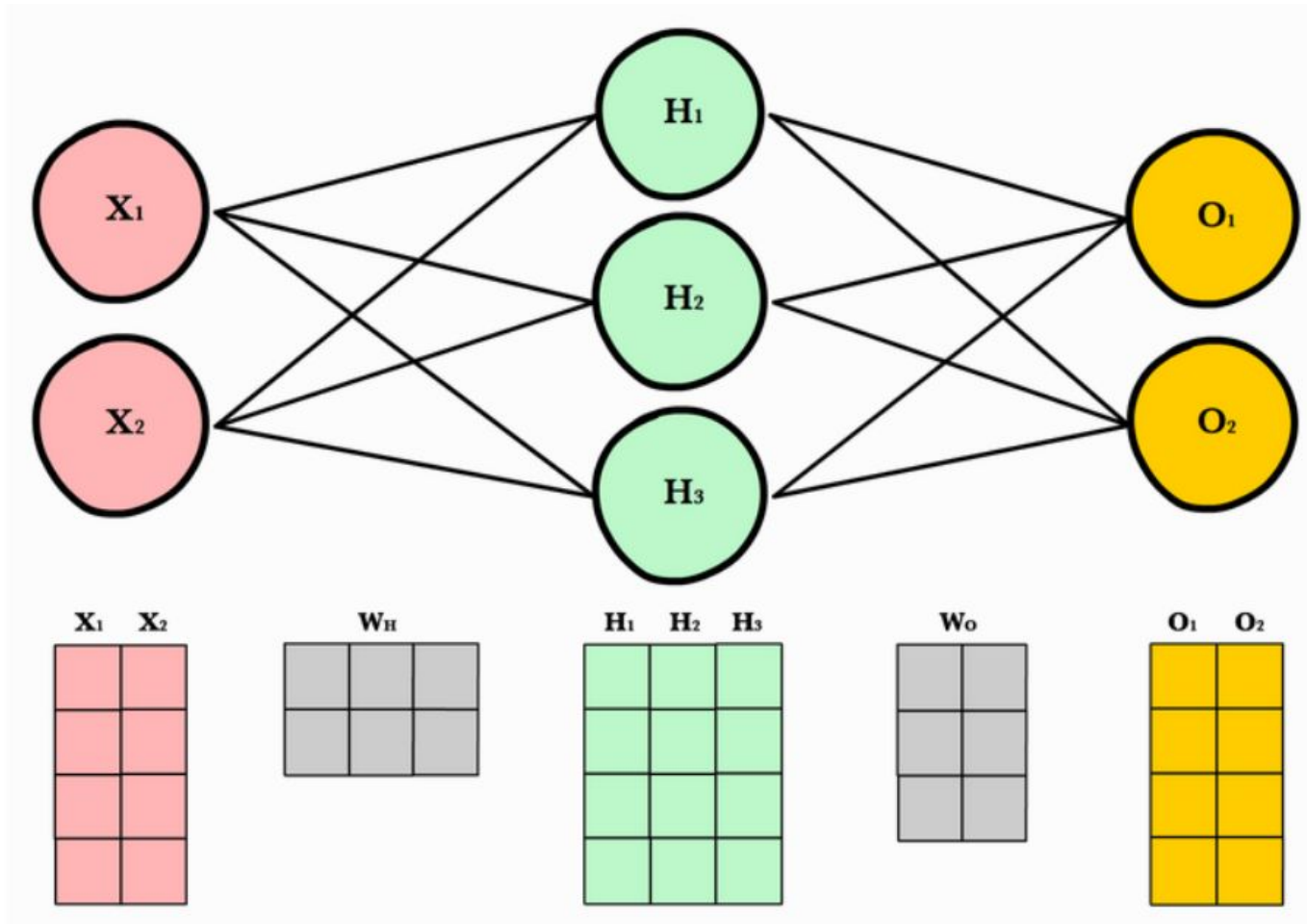
Jak sprawdzamy czy model będzie dobrze działał w przyszłości?



Celem treningu modelu jest to, aby dobrze działał w przyszłości czyli na danych, których “nie widział”. Badamy tą zdolność korzystając ze zbioru walidacyjnego. Proces wygląda tak, że w ramach jednej epoki, trenujemy model na danych treningowych a potem sprawdzamy poprawność jego predykcji na zbiorze walidacyjnym (na którym nie trenujemy modelu).

Jeżeli jakość modelu na danych walidacyjnych zacznie spadać wówczas stosujemy mechanizm *early stopping* i kończymy trening. Finalny model sprawdzamy jeszcze na zbiorze testowym.

Jak działanie sieci neuronowych wiąże się z macierzami?



Dane wejściowe są macierzami z wartościami (wierszy jest tyle ile wynosi *batch size*). Podobnie wagi są zorganizowane w macierzach, aby przyspieszyć obliczenia.



Epoch
000,000

Learning rate

0.03

Activation

Tanh

Regularization

None

Regularization rate

0

Problem type

Classification

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



REGENERATE

FEATURES

Which properties do you want to feed in?

- X_1
- X_2
- X_1^2
- X_2^2
- $X_1 X_2$
- $\sin(X_1)$
- $\sin(X_2)$

+ - 2 HIDDEN LAYERS



4 neurons



This is the output from one **neuron**. Hover to see it larger.



2 neurons

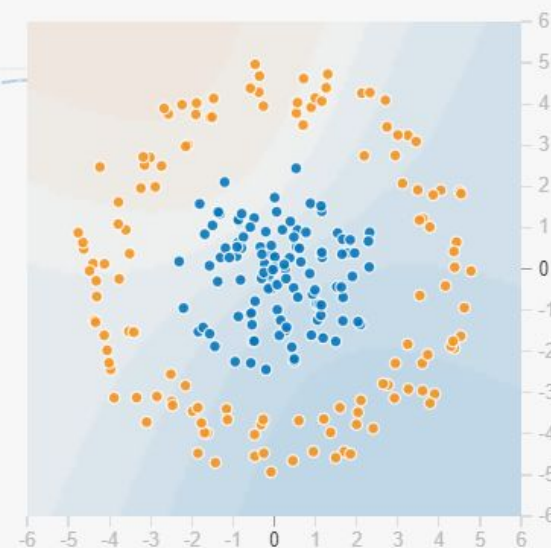


The outputs are mixed with varying **weights**, shown by the thickness of the lines.

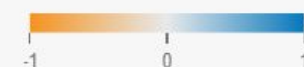
OUTPUT

Test loss 0.490

Training loss 0.508



Colors shows data, neuron and weight values.



☐ Show test data

☐ Discretize output

<https://playground.tensorflow.org/>