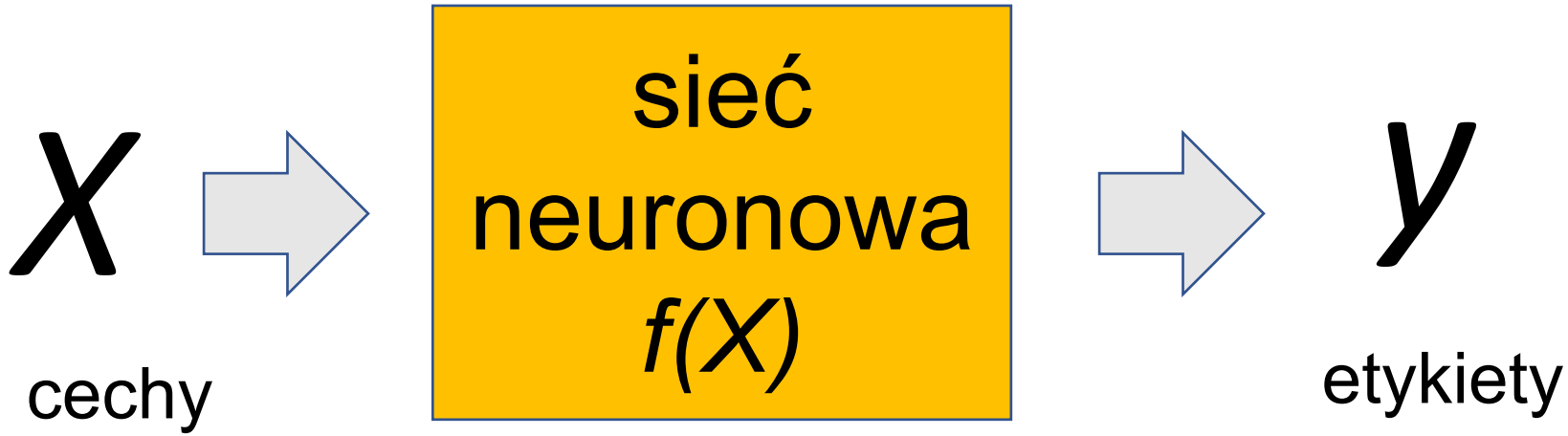
Wstęp do sieci neuronowych

Paulina Tomaszewska



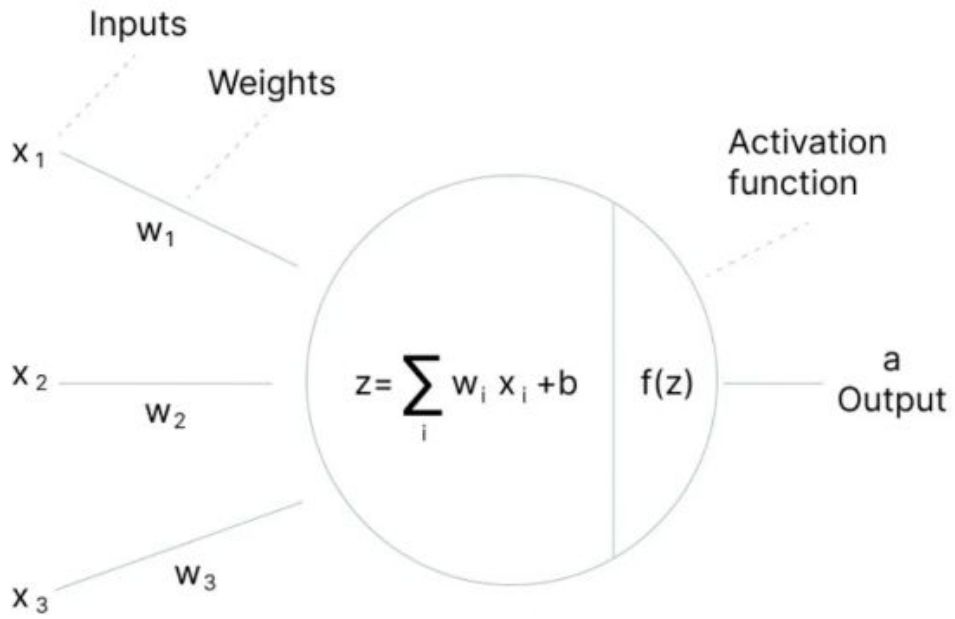
Sieci neuronowe to **skomplikowane funkcje** z dużą liczbą parametrów. Jest to jeden z rodzajów modeli stosowanych w AI.

(Sztuczne) sieci neuronowe są zbudowane ze (sztucznych) neuronów

W prezentacji są stosowane pewne uproszczenia i skróty myślowe, aby ułatwić intuicyjne zrozumienie treści.

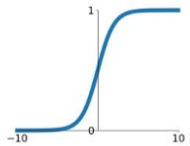
Rozbudowane opisy zostały przygotowane z myślą o osobach, które nie uczestniczyły w zajęciach, aby umożliwić im samodzielne zapoznanie się z treściami.

Sztuczny neuron (zaproponowany w 1943)

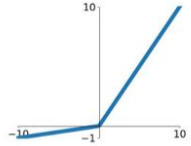


Activation Functions

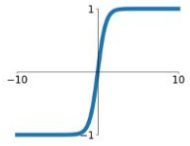
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



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

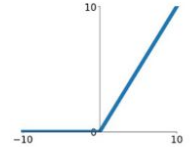


tanh
$\tanh(x)$

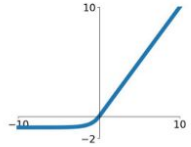


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



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



W neuronie wykonywana są następujące operacje:

1. mnożenie poszczególnych wag (w_i) i odpowiadających im wejść (x_i)
2. sumowanie składników z pkt. 1 wraz z wyrazem wolnym (b)
3. wyliczenie wartości funkcji aktywacji (f), której wejściem jest wynik z pkt. 2

https://aiml.com/what-is-a-multilayer-perceptron-mlp/

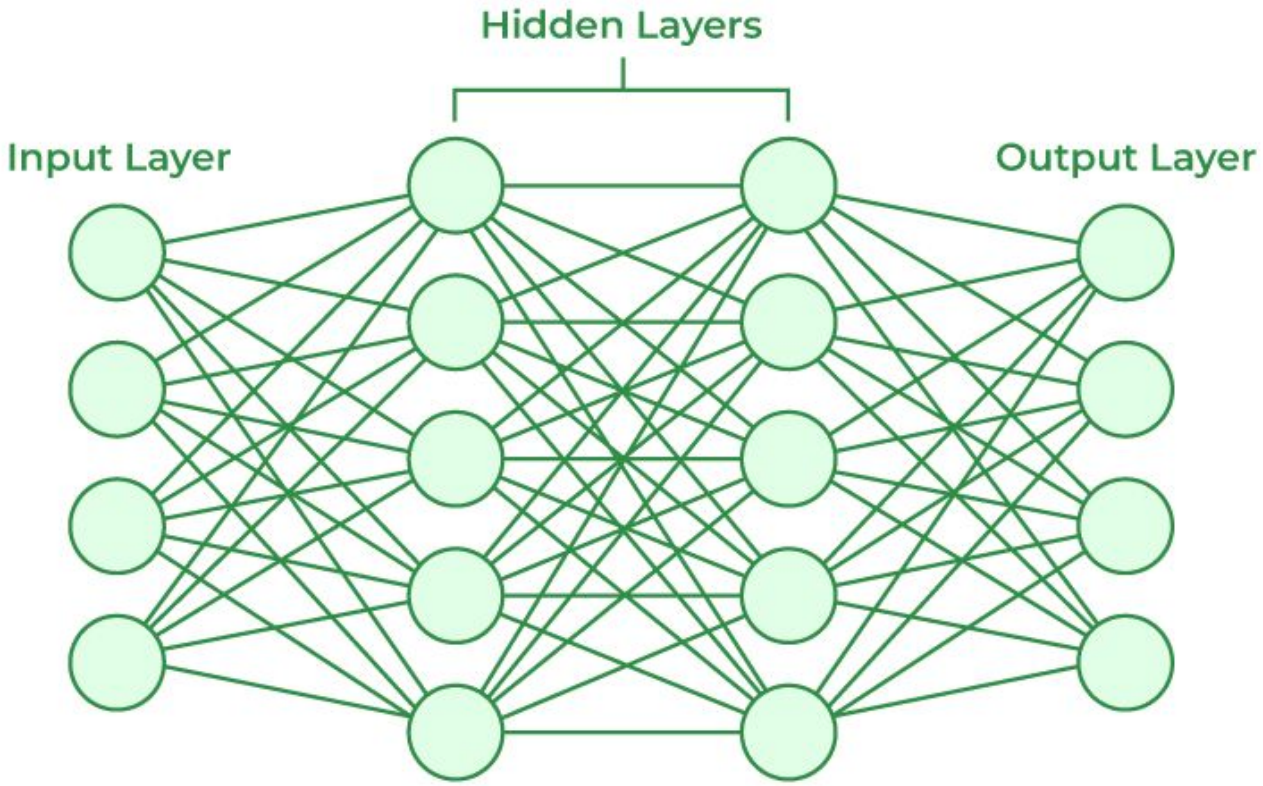
https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

Perceptron wielowarstwowy
(ang. *Multilayer perceptron, MLP*)

- Jest to najprostszy rodzaj sieci neuronowej
- Stosuje się nazwy: warstwy gęste albo w pełni połączone

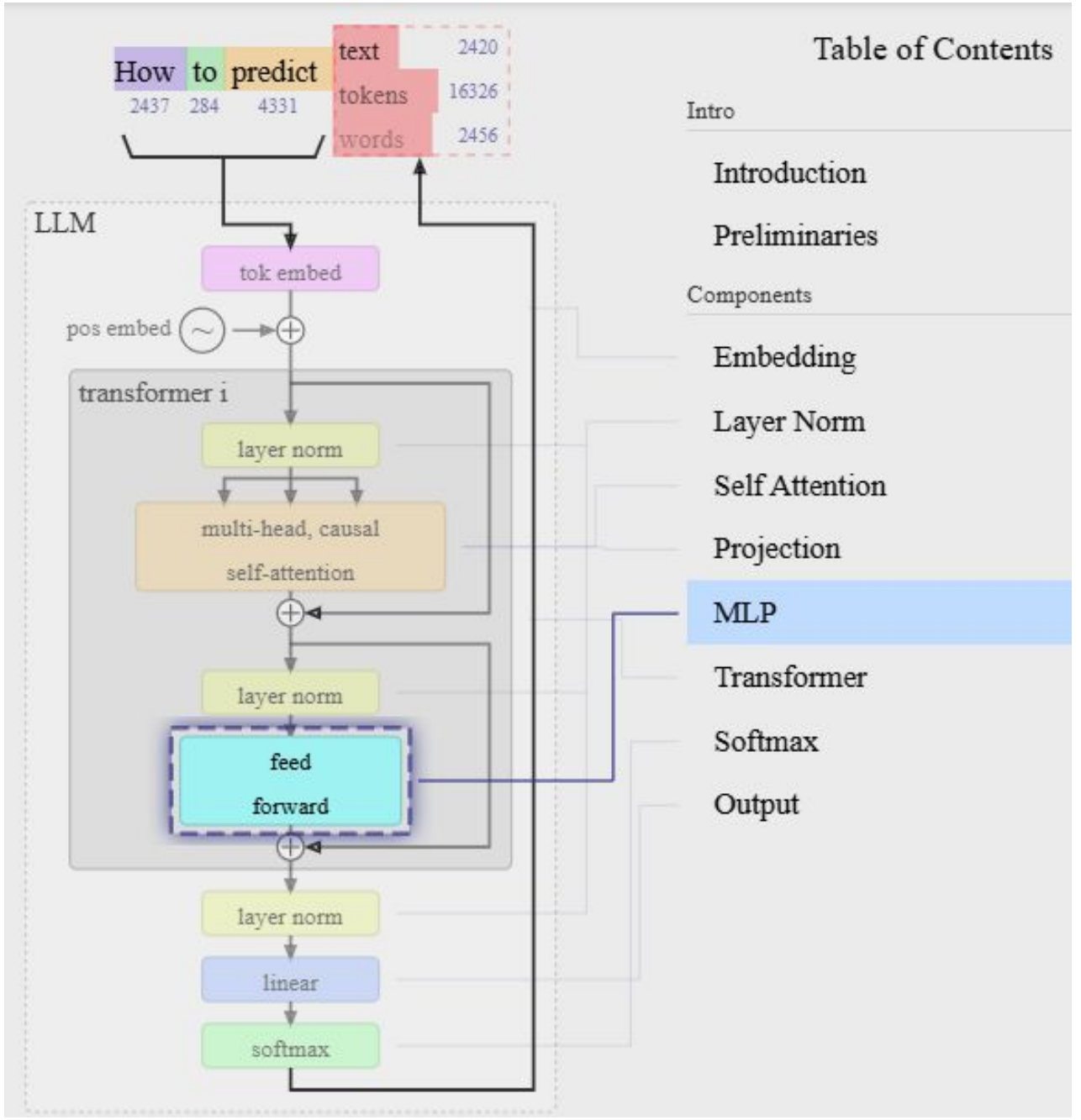
Pytania:

- ile warstw potrzebujemy?
- ile powinno być neuronów w każdej warstwie?



https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

MLP - jeden z komponentów Dużych Modeli Językowych (*ang. Lange Language Models, LLMs*) stosowanych m.in. w ChatGPT.



https://bbycroft.net/llm

Uproszczony przykład:
przewidywanie temperatury następnego dnia

Mamy dane historyczne:

X - temperatura w danym dniu, poziom wilgotności

y - temperatura w dniu kolejnym

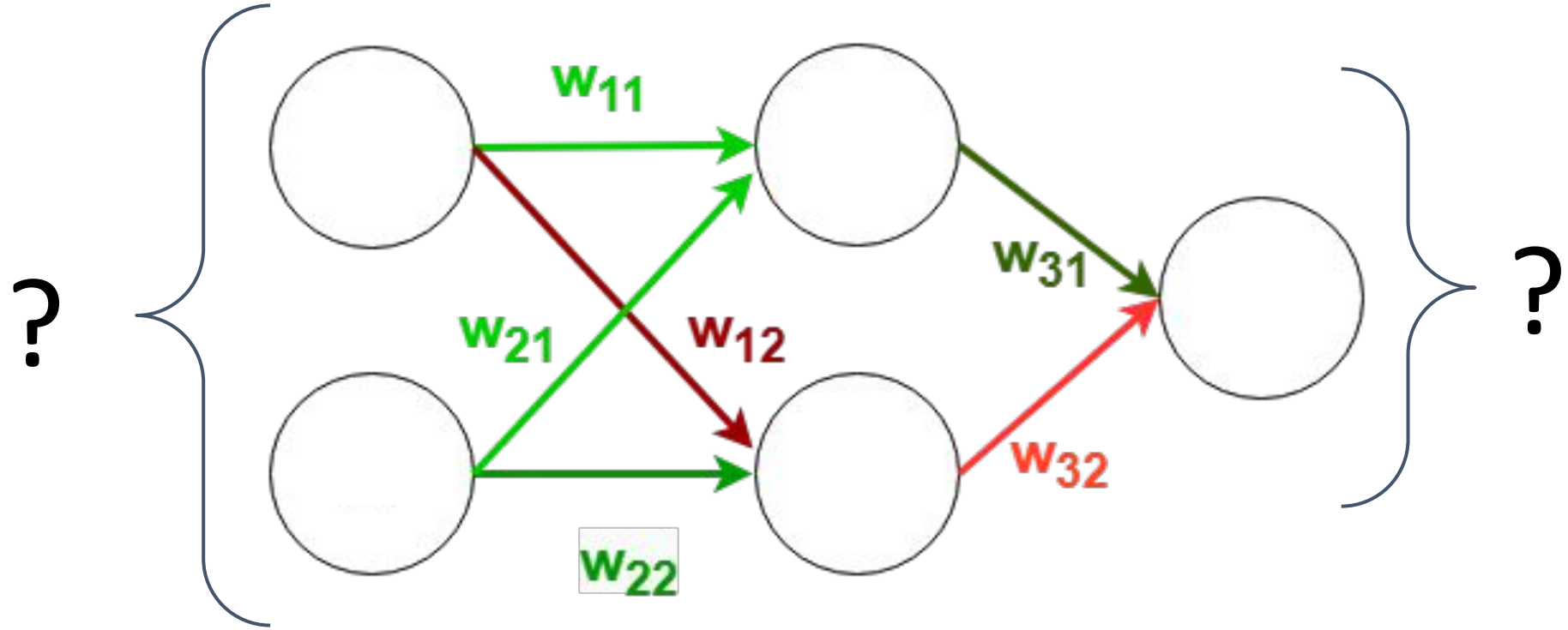
Użyjmy sieci neuronowych

Uwaga: W praktyce, rzadko użyjemy sieci neuronowych do analizy prostych danych tabelarycznych.

Jest to przykład dla celów edukacyjnych, aby wyjaśnić jak sieci neuronowe działają.

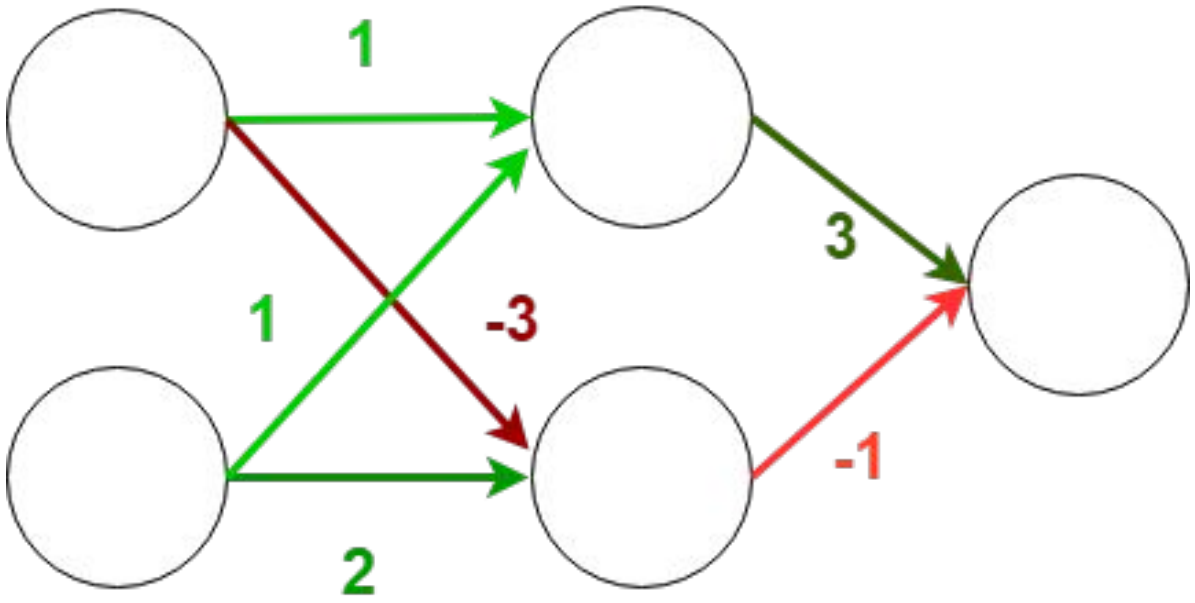
Krok 0: Zdefiniuj architekturę sieci neuronowej

Czy wiemy ile neuronów powinno być w pierwszej i ostatniej warstwie?



Tak, bo wiemy jaki jest rozmiar próbek w danych historycznych.

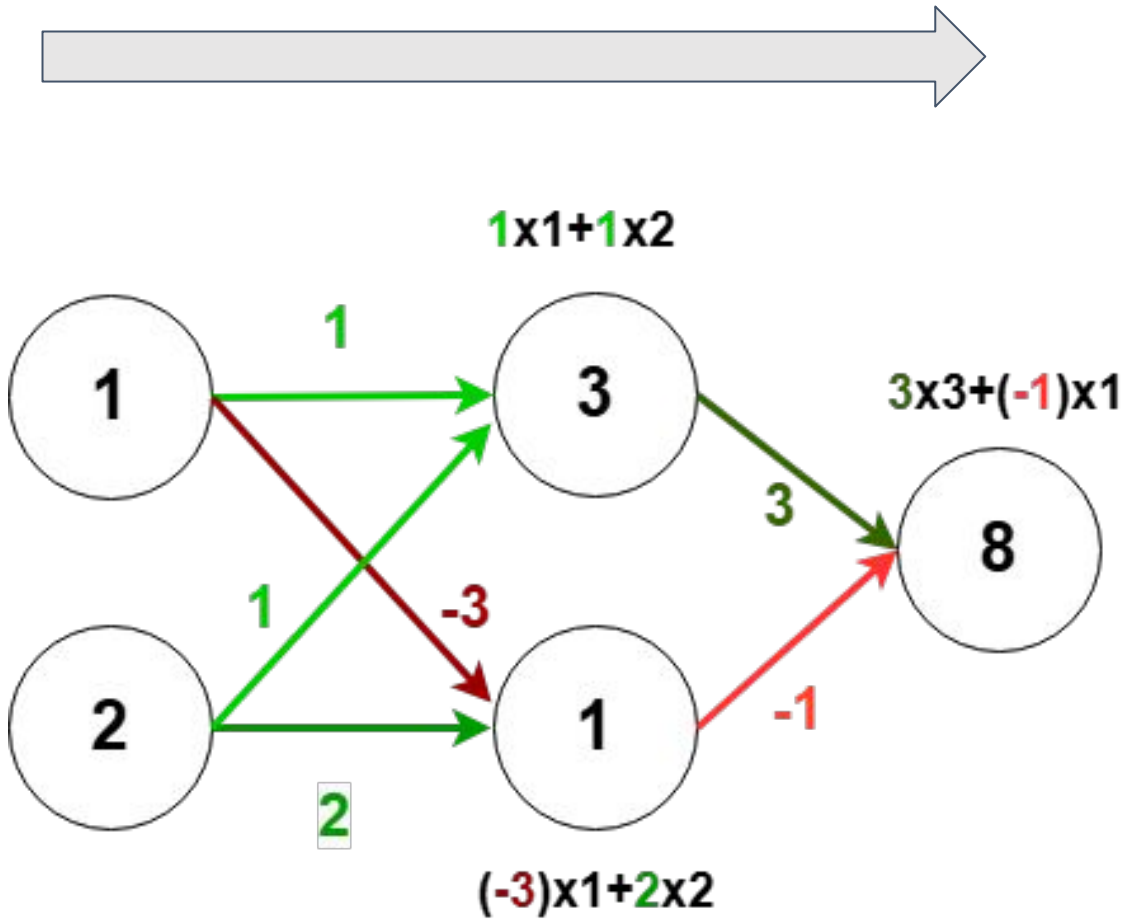
Krok 0: Zainicjalizuj wagi



Jak?

- losowo
- np. z rozkładu Gaussa, inicjalizacja Xaviera

Krok 1: Inferencja (*ang. forward pass*)



Używamy cech (X) jako wejście, aby wyliczyć wyjście modelu zgodnie ze wzorami opisujących działanie neuronów.

Uwaga: W praktyce w wyliczeniach stosuje się operacje na macierzach zamiast wyliczeń “element po elemencie” dla lepszej szybkości.

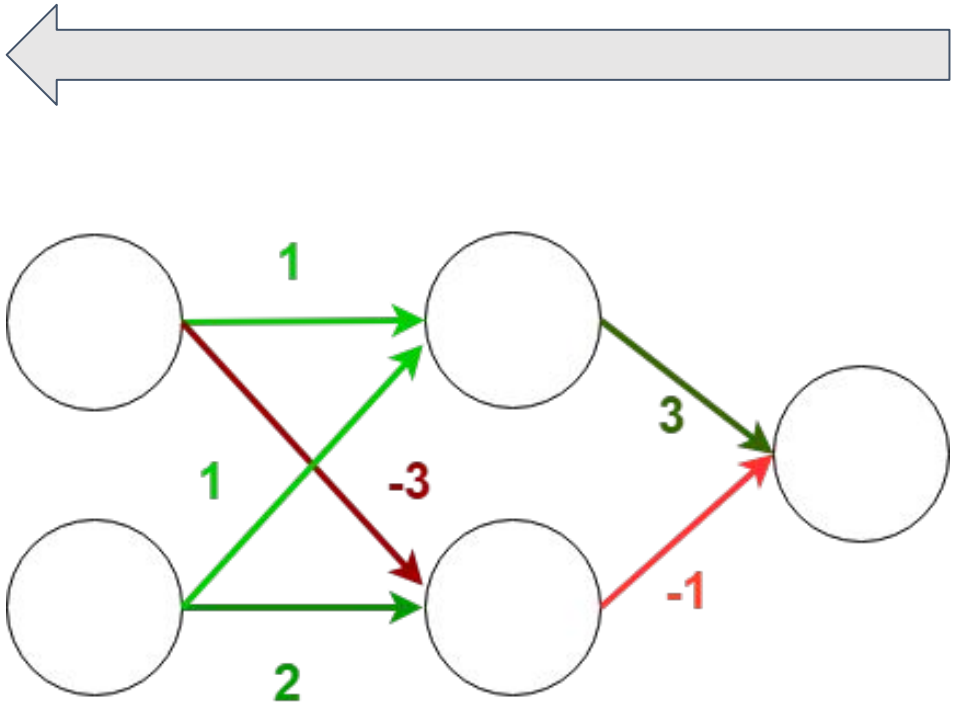
Krok 2: Sprawdź jak dobre jest wyjście modelu

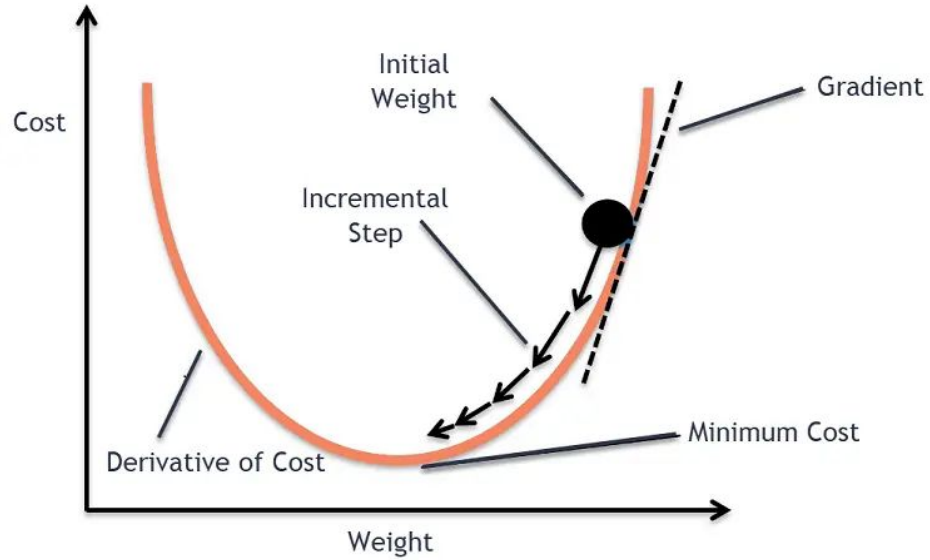
Model zwrócił wartość 8, ale prawdziwa historyczna wartość (etykieta) to 9.

Jaki jest błąd?
Policzymy to korzystając z **funkcji kosztu**/ straty (ang. *cost function*).
Intuicyjnie, błąd = wartość zwrócona - etykieta = 8 - 9 = -1

W praktyce w zadaniach regresji stosujemy jednak raczej błąd średniokwadratowy (ang. *mean squared error*, MSE) jako sposób na określenie jak bardzo model się pomylił.
błąd = ½ (wartość zwrócona- etykieta)^2 = ½ (8-9)^2 = ½

Krok 3: Propagacja wsteczna (ang. *backpropagation*)

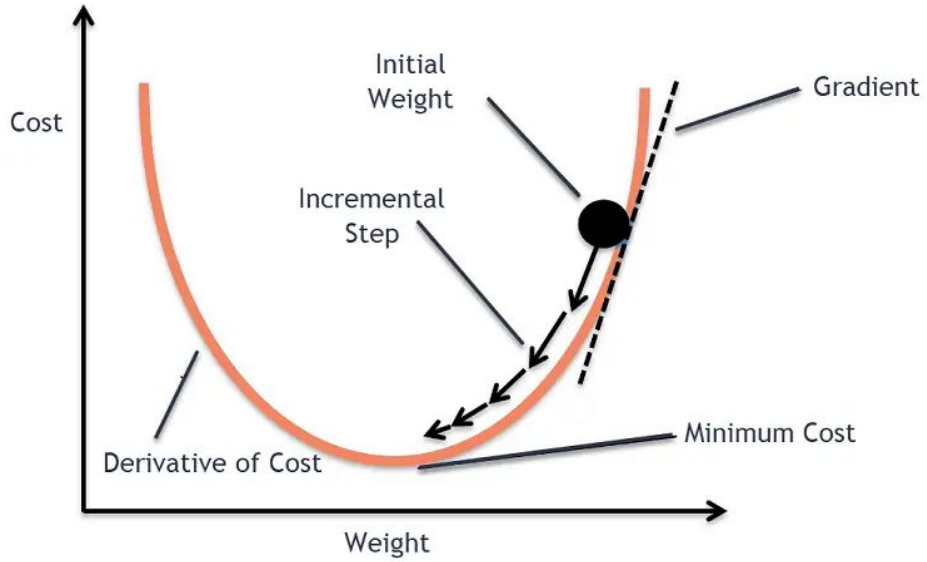


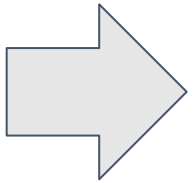


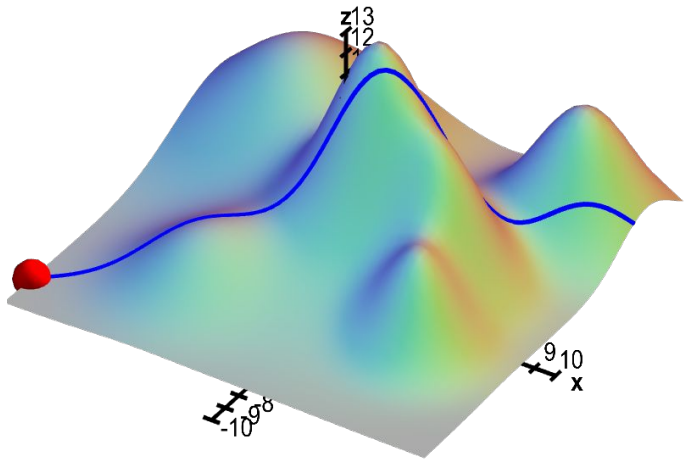
Obliczamy gradient funkcji kosztu względem wag modelu. Więcej informacji o gradientach w notebook na repozytorium.

https://www.malicksarr.com/gradient-descent-in-neural-network/

Zauważ, że funkcja kosztu w sieciach neuronowych ma skomplikowany “kształt” i istnieje ryzyko, że zamiast trafić do globalnego minimum trafimy do lokalnego minimum.







https://www.malicksarr.com/gradient-descent-in-neural-network/

https://mathinsight.org/chain_rule_multivariable_introduction

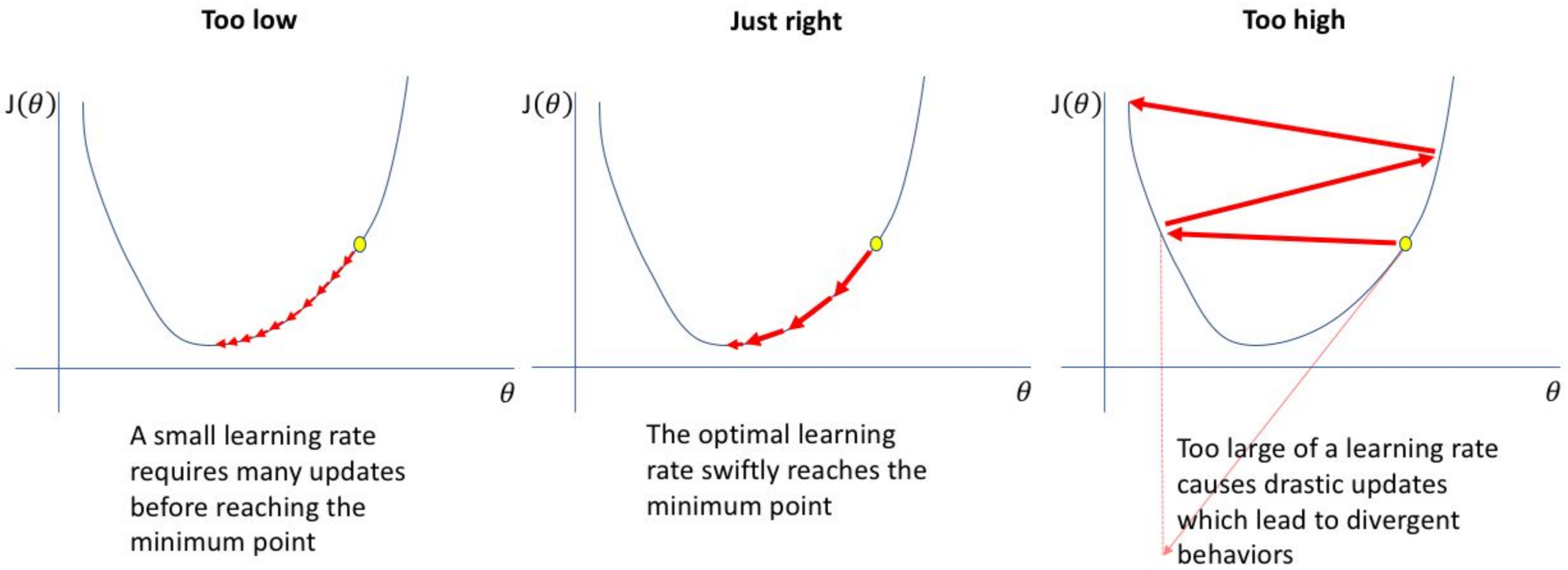
Krok 4: Aktualizacja wag modelu

Model nie zwraca poprawnych wartości, ponieważ ma niepoprawne wagi. Na początku ustawiliśmy je na losowe wartości. Poprawimy wagi korzystając z prostej reguły

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

lr - *learning rate* (odpowiedzialny na szybkość zmian wag)

Uwaga: Są bardziej skomplikowane wzory na zmianę wag modelu. Powyższa metoda to metoda spadku gradientu (ang. *stochastic gradient decent*, SGD) ale są inne.

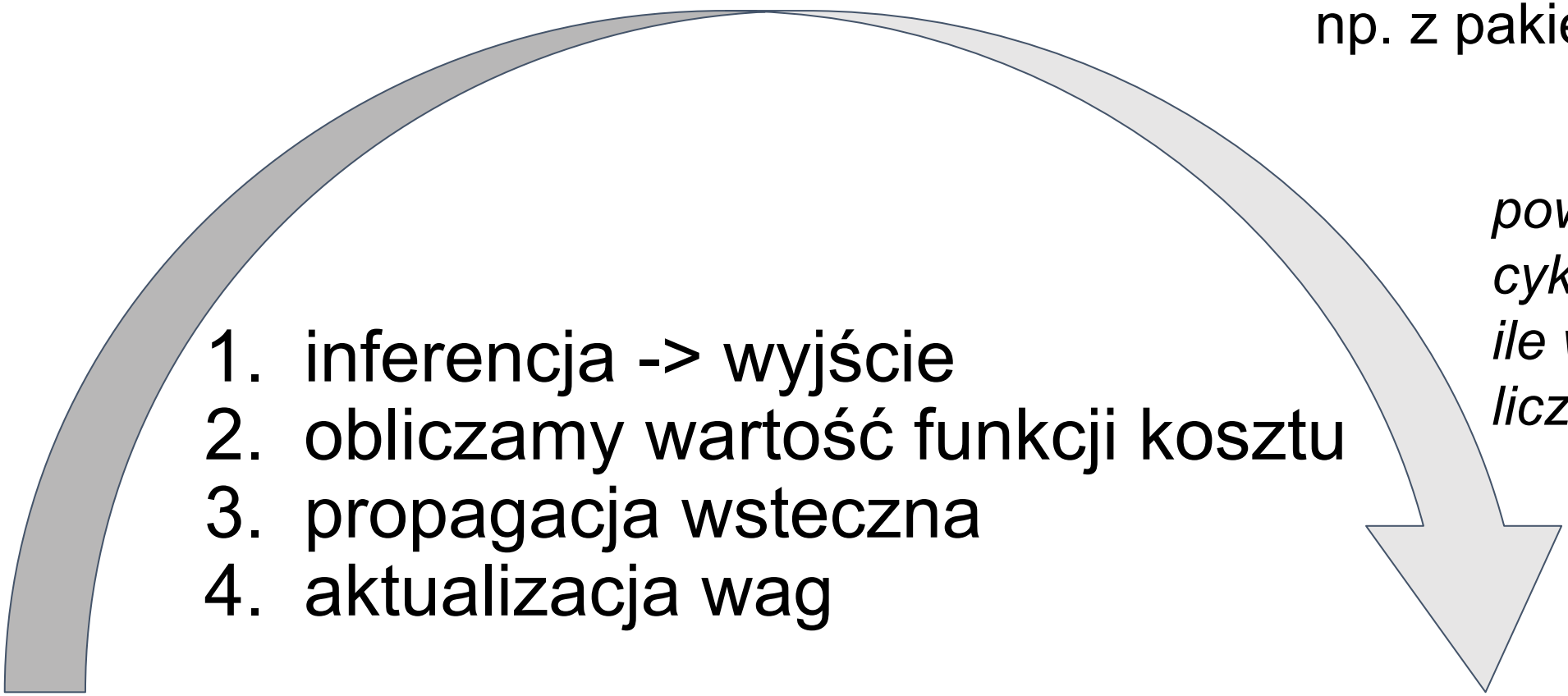


Dobór odpowiedniego hiperparametru *learning rate* jest kluczowy, aby trening nie trwał zbyt długo i aby nie „ominąć” minimum.

https://www.jeremyjordan.me/nn-learning-rate/

Co się dzieje dalej?

W praktyce nie trzeba wszystkiego implementować od zera - można skorzystać np. z pakietu *pytorch*.



1. inferencja -> wyjście
2. obliczamy wartość funkcji kosztu
3. propagacja wsteczna
4. aktualizacja wag

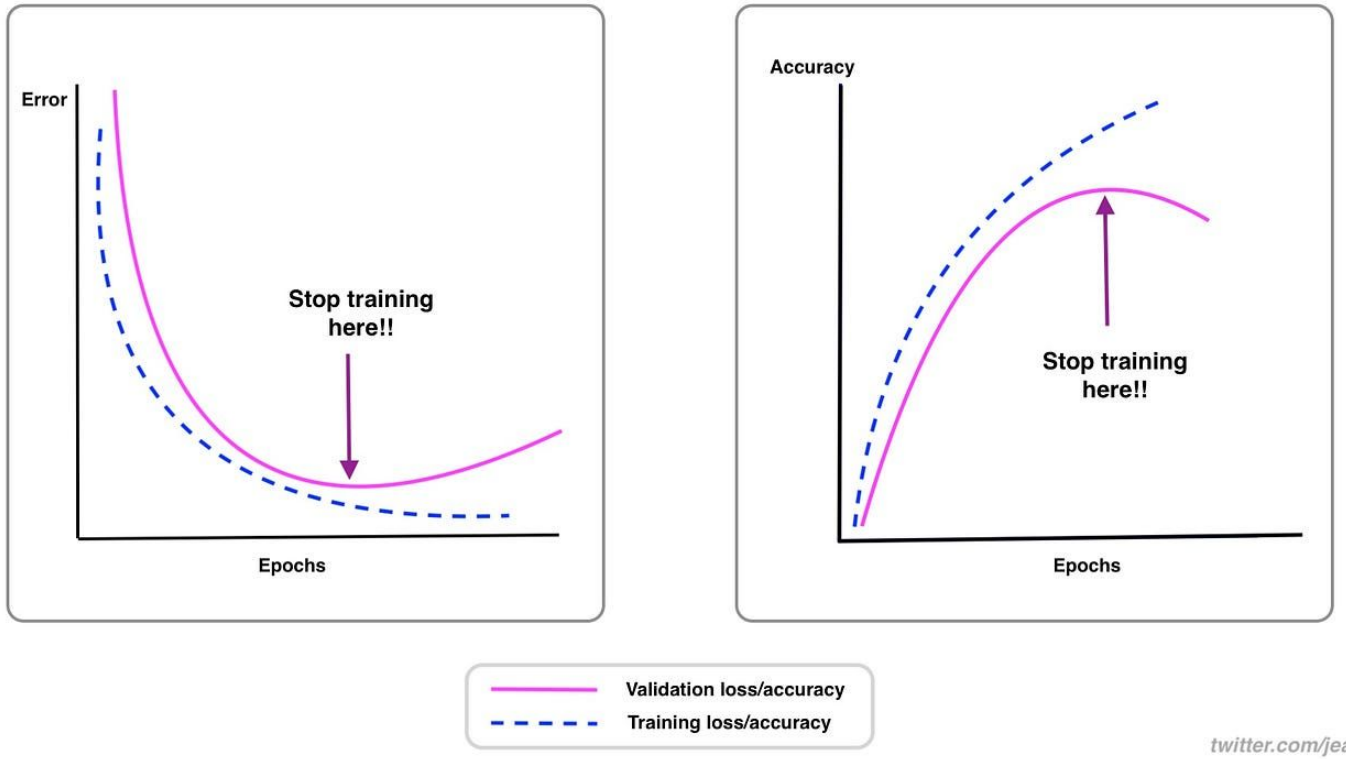
powtarzamy to cyklicznie tyle razy, ile wynosi zdefiniowana liczba epok

Epoka kończy się, gdy wszystkie dane treningowe zostały wykorzystane do aktualizacji wag modelu.

Jak sprawdzamy czy model będzie użyteczny w przyszłości?

Sprawdzamy czy model ma zdolność do generalizacji czyli poprawnych predykcji w przypadku danych, których nie wykorzystywaliśmy podczas treningu. Dane, na których ewaluujemy model podczas treningu nazywamy walidacyjnymi.

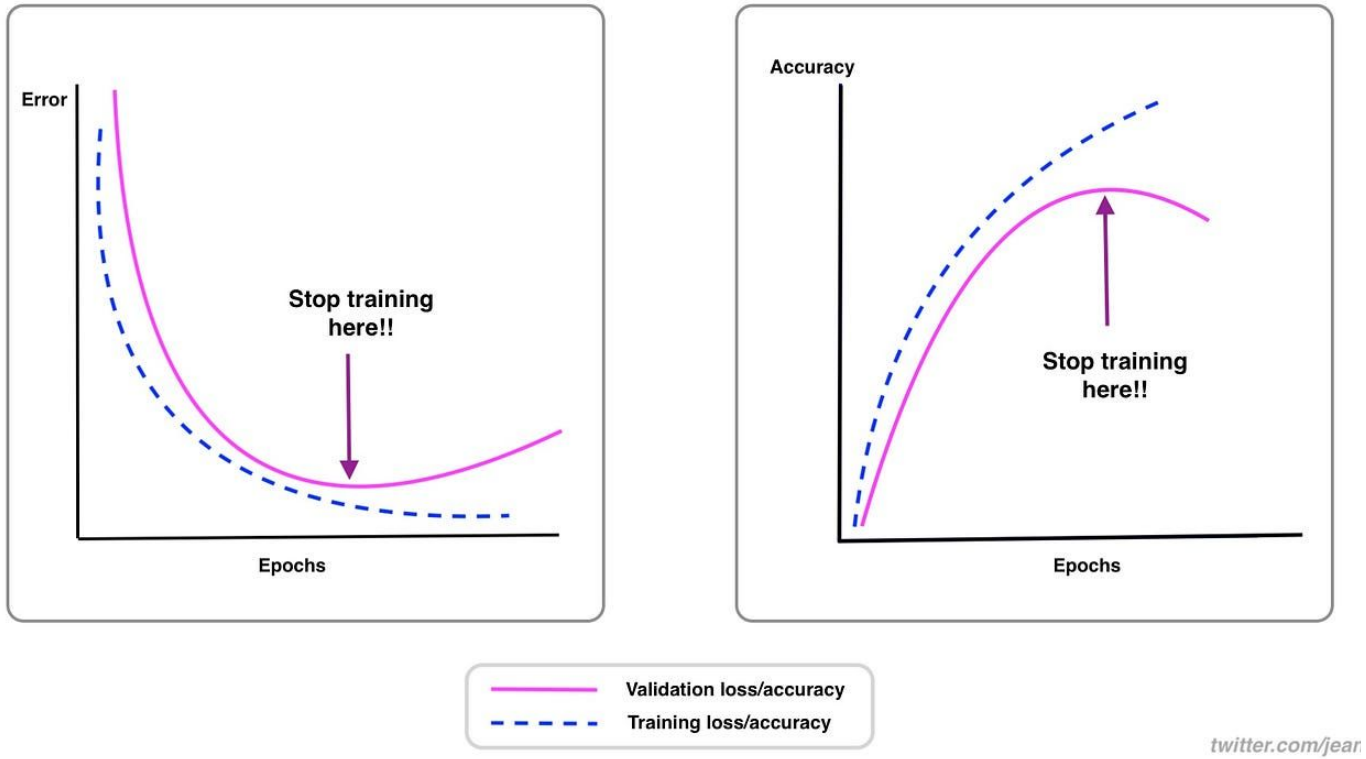
Jak uniknąć przeuczenia modelu?



- chcemy, aby *performance* modelu na zbiorze treningowym i walidacyjnym był zbliżony
- jeżeli w trakcie treningu początkowo spadający błąd na zbiorze walidacyjnym zacznie rosnąć, oznacza to, że model zaczyna się przeuczać (ang. *overfitting*). Może to oznaczać, że model zamiast uczyć się ogólnych wzorców w danych, zaczął “zapamiętywać” dane treningowe.

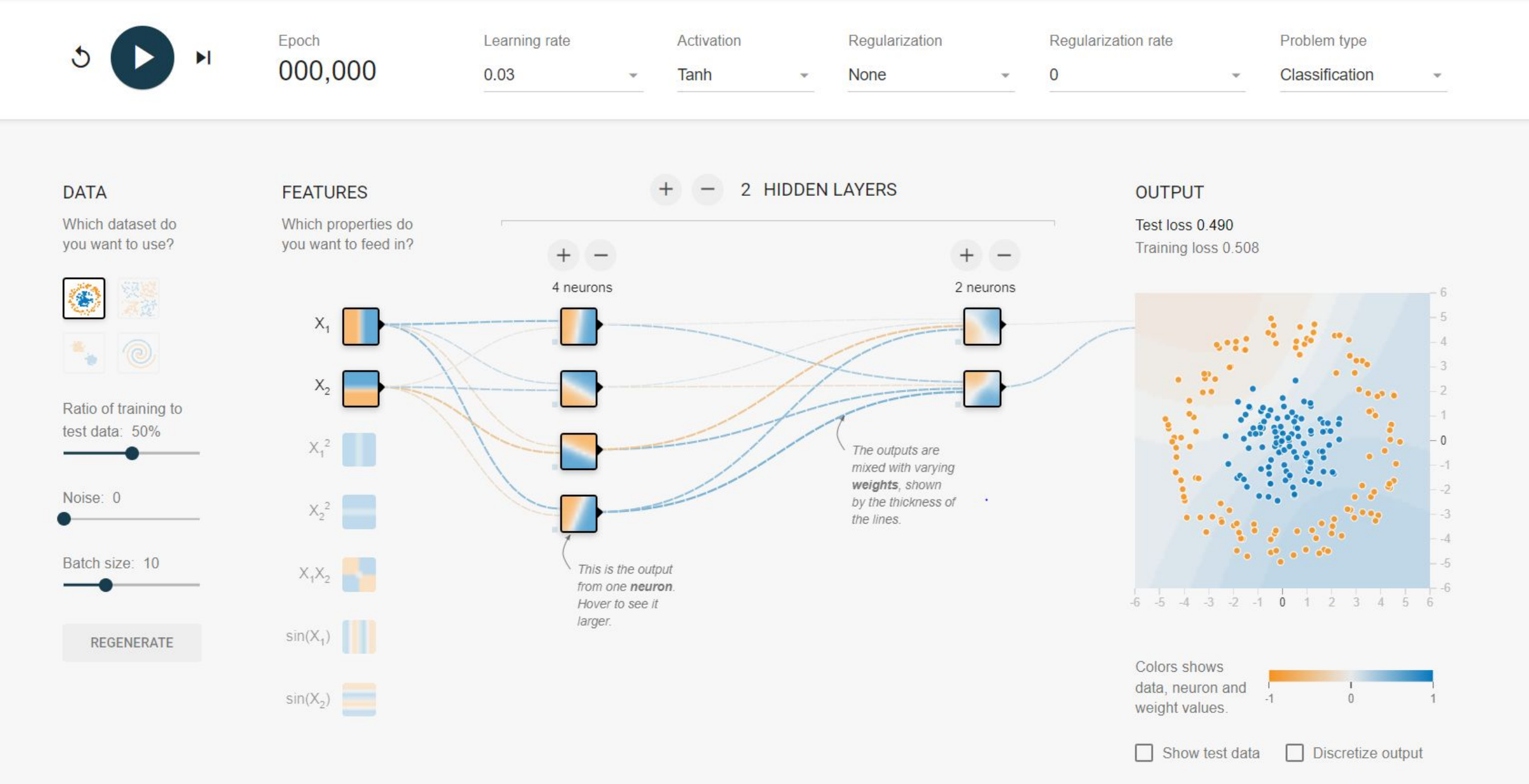
https://jeande.medium.com/early-stopping-explained-62eebce11

Jak uniknąć przeuczenia modelu?



- gdy zidentyfikujemy, że błąd modelu na zbiorze walidacyjnym zaczyna rosnąć, wdrażamy procedurę *early stopping* z parametrem *patience*
- *patience* określa “ile epok jesteśmy gotowi poczekać bez poprawy modelu, zanim uznamy, że dalszy trening nie ma sensu i go przerwiemy”

https://jeande.medium.com/early-stopping-explained-62eebce11



Na tej stronie internetowej można “wyklikać” sieć neuronową: https://playground.tensorflow.org/