



GŁĘBOKIE

# GŁĘBOKIE SIECI

# GŁĘBOKIE SIECI NEURONOWE

Koło Naukowe Ucznia Maszynowego MIM UW  
Julia Bazińska  
Na bazie materiałów Mateusza Maciasa

# GŁĘBOKIE SIECI NEURONOWE



# ML a konwencjonalne programowanie

Tworzymy algorytm, który na podstawie danych stworzy algorytm, który rozwiąże zadanie.

## Programowanie:

- Trzeba zdefiniować procedurę prowadzącą do celu.
- Umiemy zdefiniować, czy nam się udało czy nie.
- Zwykle szukamy rozwiązania poprawnego lub najlepszego.

## Uczenie maszynowe:

- Wystarczy określić cel i podać przykłady rozwiązań.
- Musimy sami stwierdzić czy nam się udało.
- Szukamy przybliżenia rozwiązania.

# Przykłady

- Samojeżdżące samochody.
- Systemy rekomendacyjne (Netflix, Youtube etc.)
- Przewidywanie cen nieruchomości.
- Diagnozowanie raka.
- Przewidywanie pogody.
- Projektowanie leków.
- Granie w gry (atari, go).
- Rozpoznawanie mowy.
- Przewidywanie zachowań klientów.
- Deblurring.
- High Frequency Trading
- Generowanie zdjęć śmiesznych kotów.
- I inne.

sztuczna inteligencja

machine learning

deep learning





sztuczna inteligencja

machine learning

deep learning

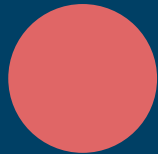
też skuteczne



sztuczna inteligencja

machine learning

deep learning



# Supervised learning



Mamy:

Zbiór próbek  
 $(X, Y)$

Chcemy:

Program który na bazie X  
przewidzi Y

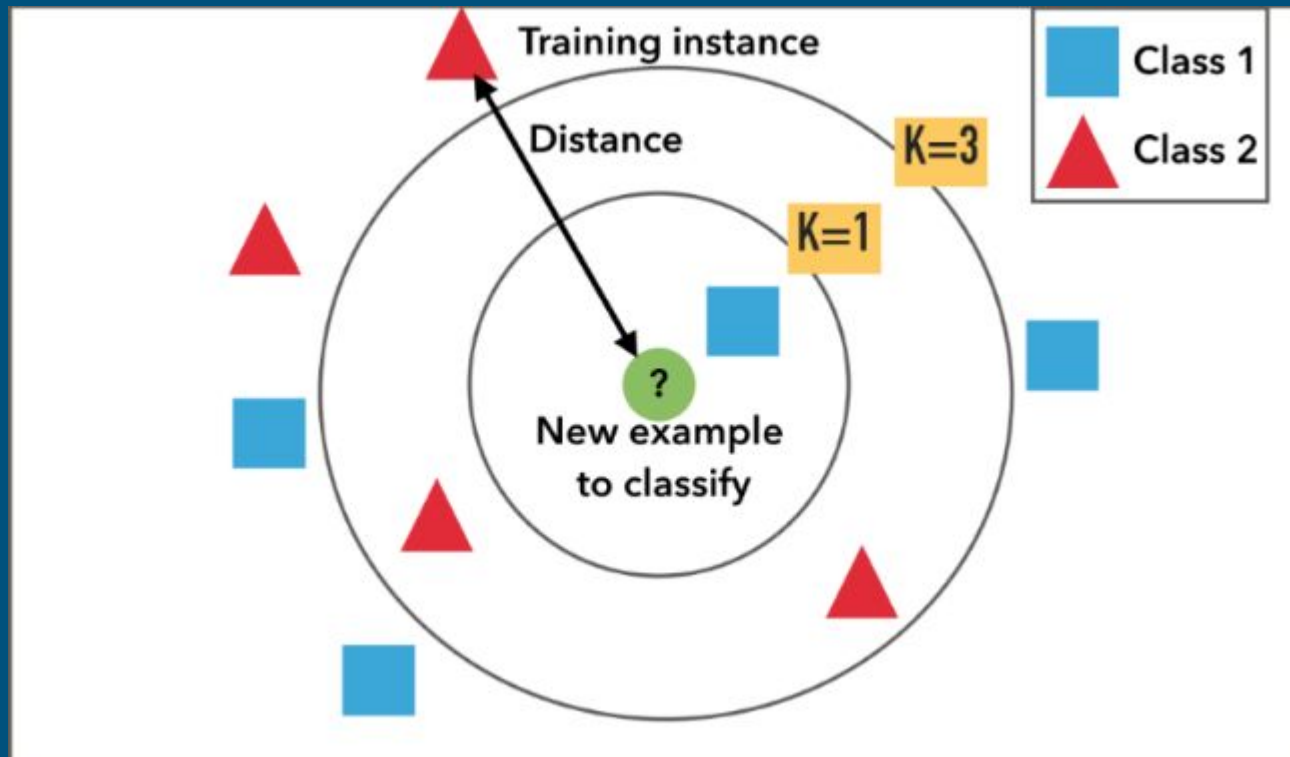
# Przykłady

X	Y
Informacje o mieszkaniu: rozmiar, dzielnica, rynek wtórny/pierwotny, rok budowy itp.	Za ile sprzedane?
Zdjęcia (kotów i psów)	Na zdjęciu jest kot czy pies?
Zdjęcie mammograficzne.	Czy jest rak piersi?
Historia kursu akcji do teraz.	Kurs akcji za minutę?
Obraz z kamerki na samochodzie.	O ile stopni przechylić kierownicę, aby utrzymać pas?
Dane o kliencie (wiek, zawód, zarobki, historia transakcji itp.)	Czy kupi, gdy do niego zadzwonimy?
Nagranie mowy.	Transkrypcja na tekst.

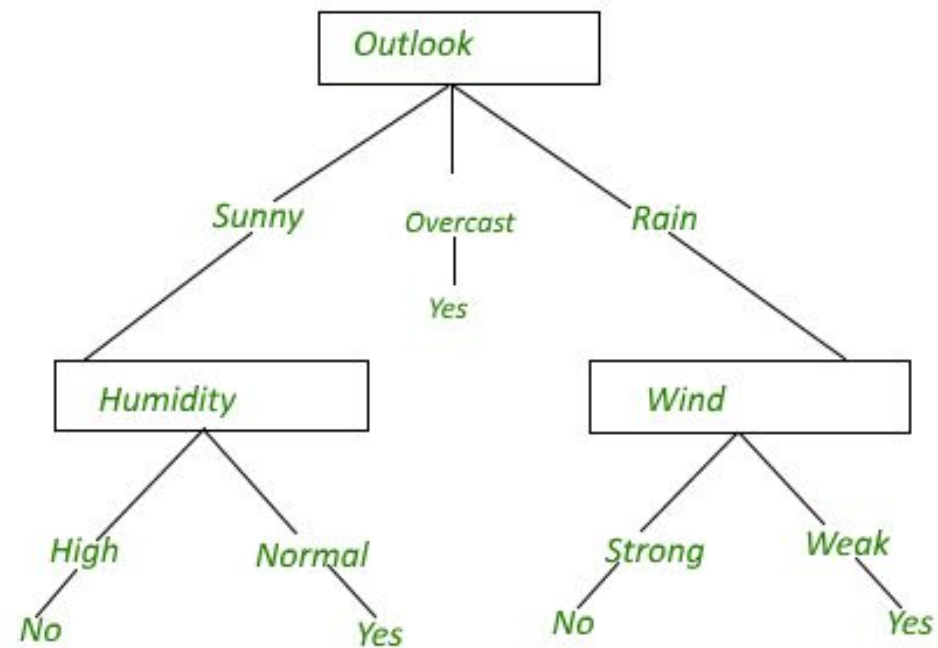
# Konfundujące słowo: model

- Model – to co bierze  $X$  i wypluwa  $y$  (funkcja mapująca  $x$  na  $y$ ).
- Model – pewien zbiór funkcji mapujących  $x$  na  $y$  (zbiór modeli w rozumieniu definicji 1).
- Model – pewien parametryzowalny zbiór funkcji mapujących  $x$  na  $y$  (zbiór modeli w rozumieniu definicji 1).
- Model – pewien zbiór funkcji mapujących  $x$  na  $y$  (zbiór modeli w rozumieniu definicji 1) wraz z algorytmem wyboru najlepszego z nich (z punktu wyboru naszych danych).
- Model – pewien parametryzowalny zbiór funkcji mapujących  $x$  na  $y$  (zbiór modeli w rozumieniu definicji 1) wraz z parametryzowalnym algorytmem wyboru najlepszego z nich (z punktu wyboru naszych danych).

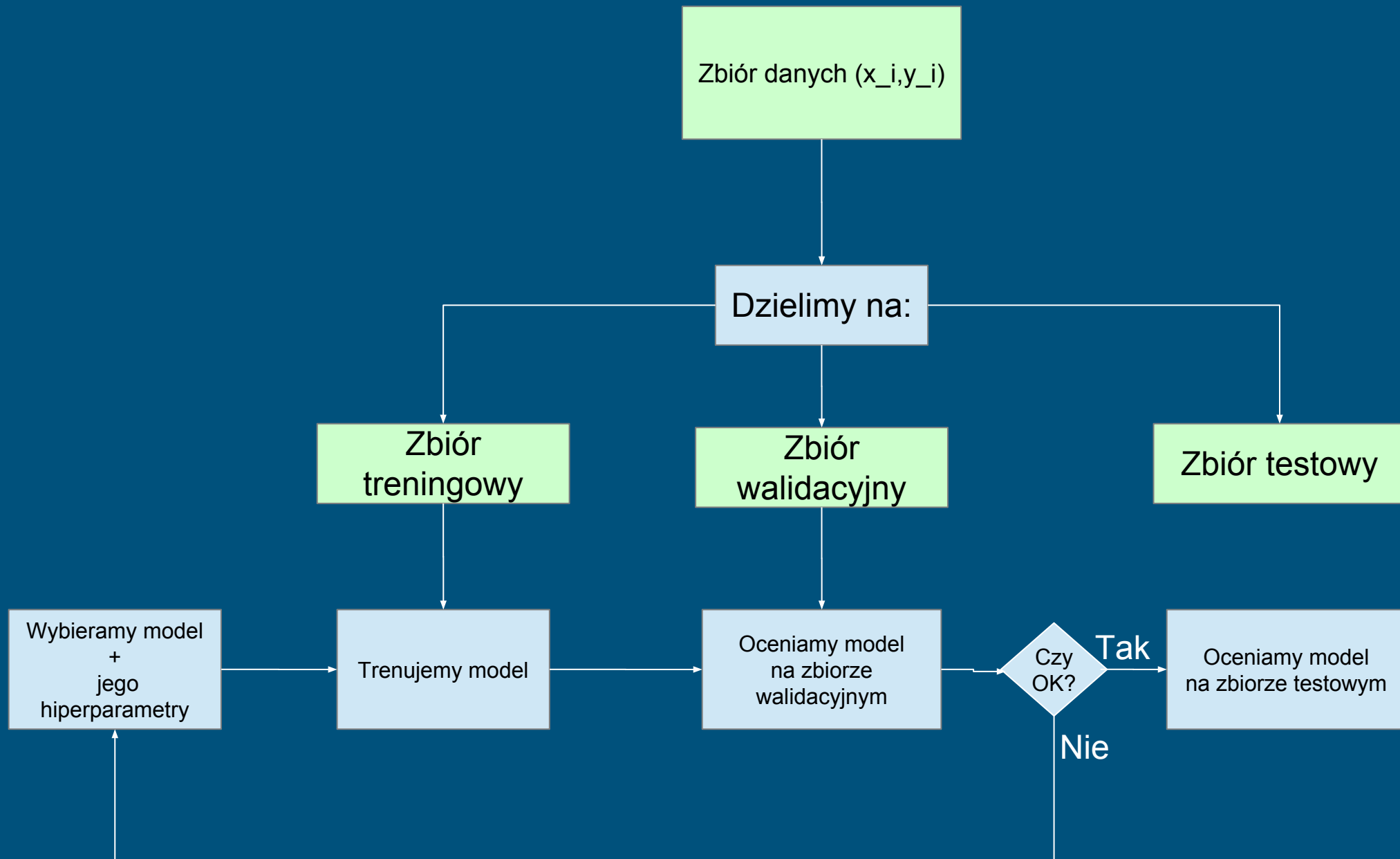
# Proste Modele: KNN



## Decision Tree for *PlayTennis*

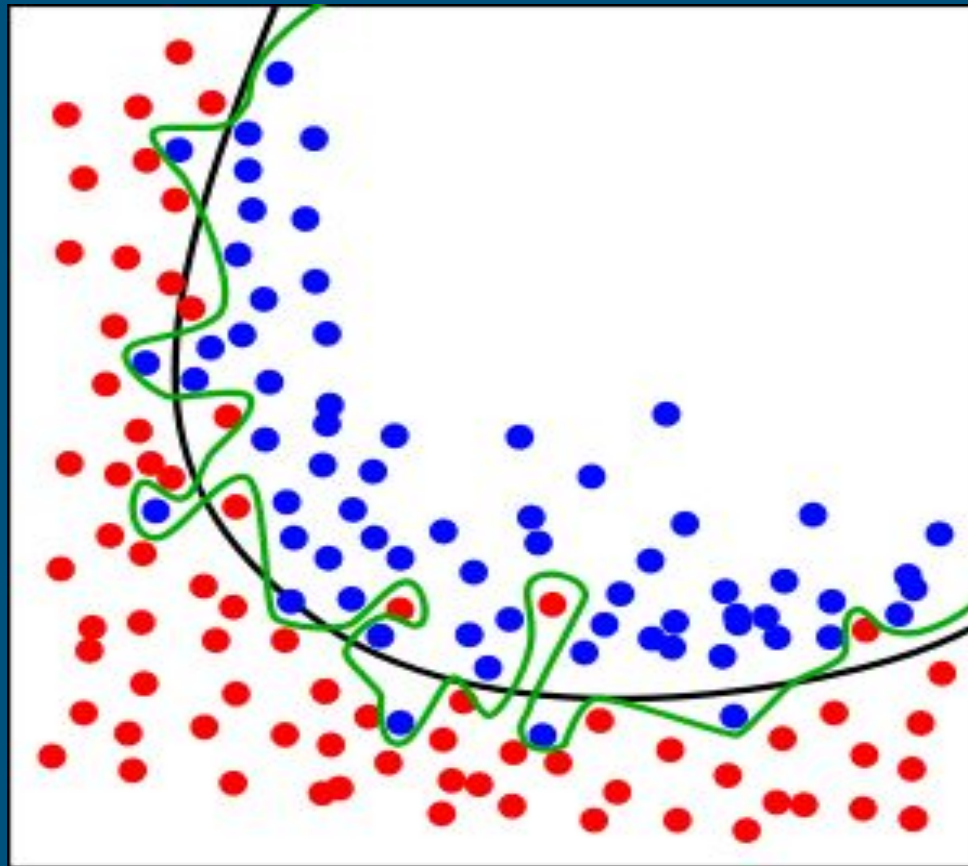


# Jak to robimy – czyli o metodologii i modelach





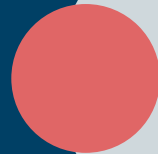
# Overfitting



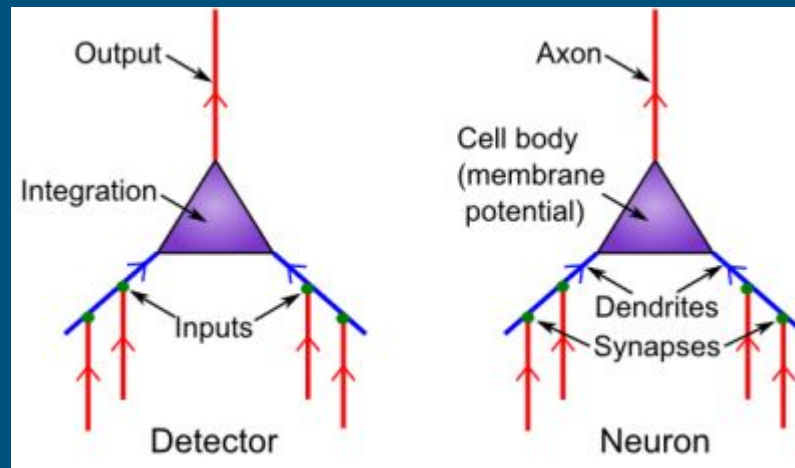
sztuczna inteligencja

machine learning

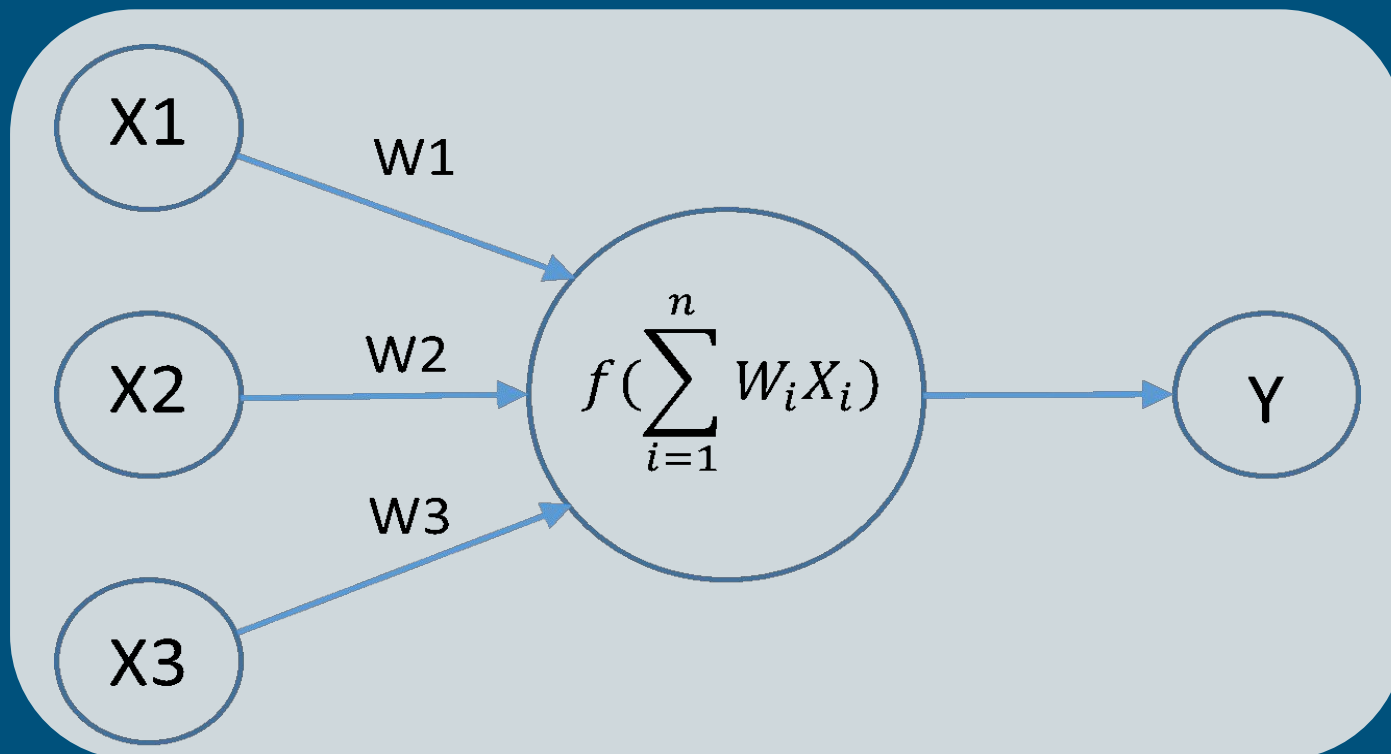
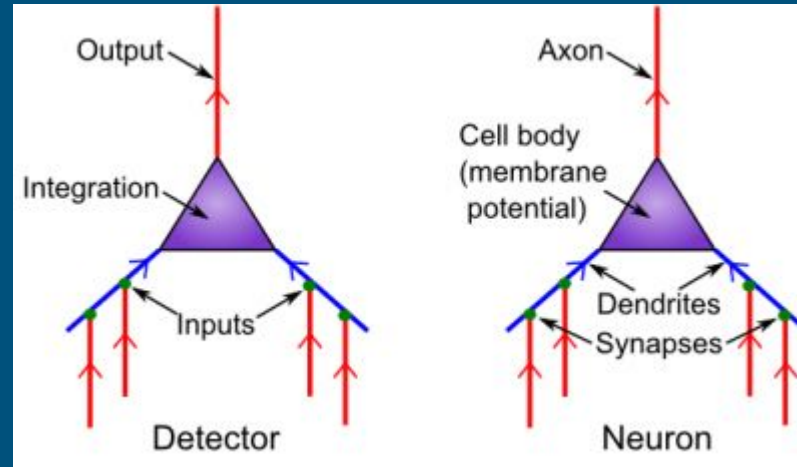
deep learning



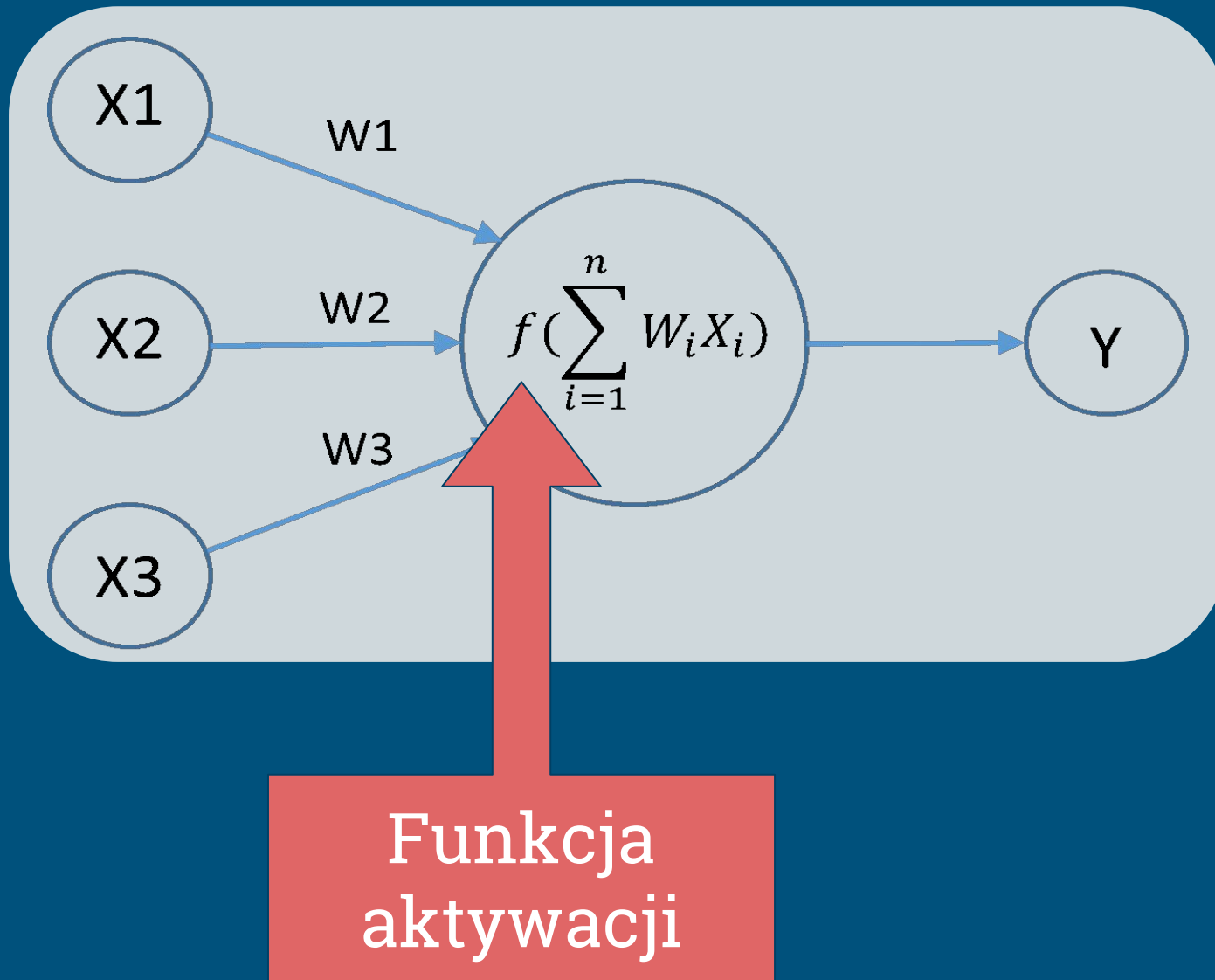
# Neuron



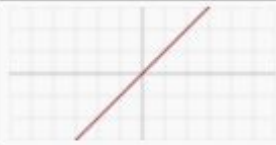
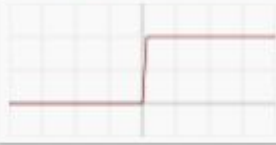

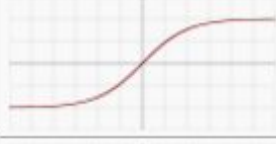



# Neuron



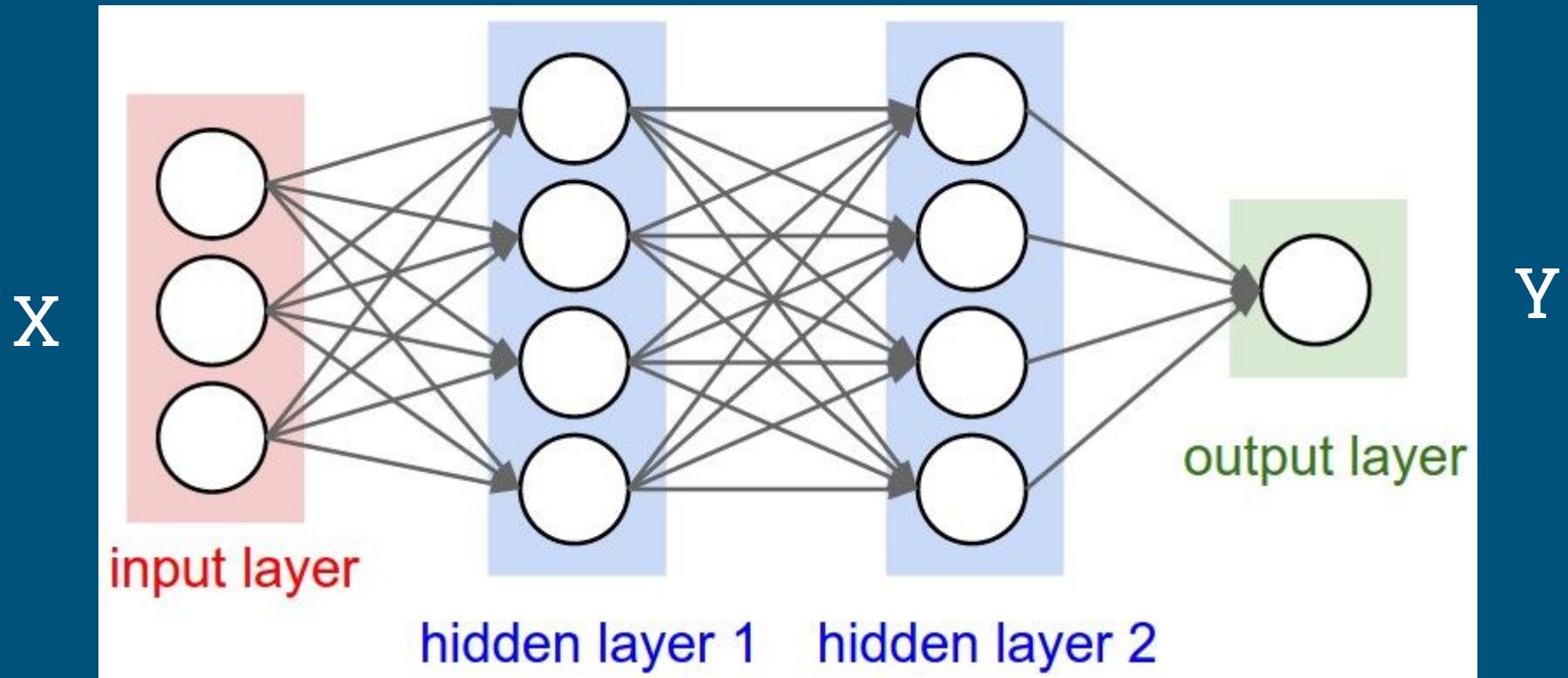
# Neuron



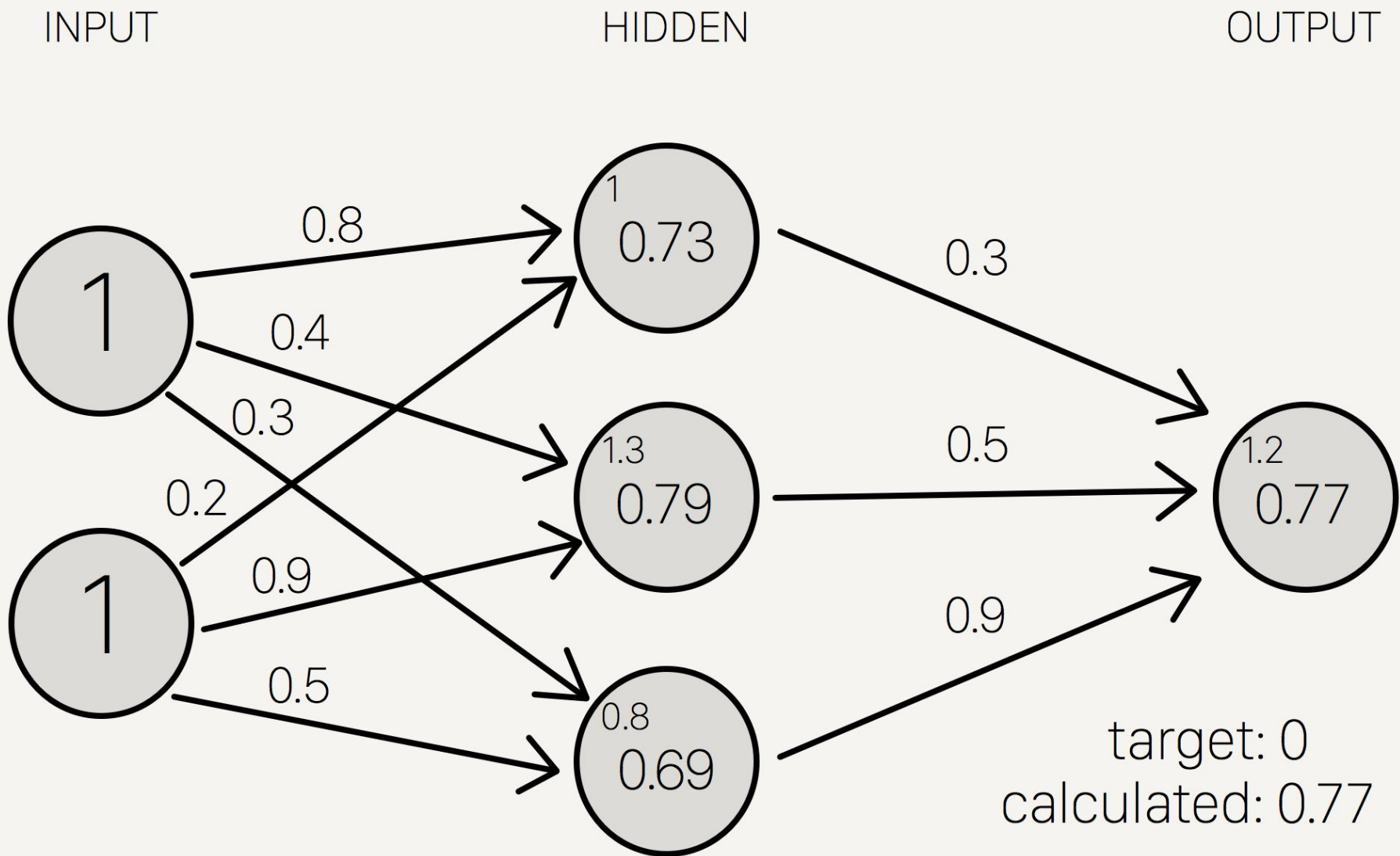
# Funkcja aktywacji

Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) <sup>[2]</sup>		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

# Sieć neuronowa



$$\hat{y}_i = f(W_1 \cdot f(W_2 \cdot f(W_3 \cdot x_i)))$$



Sigmoid:

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



# Proces uczenia

CEL:

Znaleźć takie wagi, aby sieć dobrze przewidywała  $Y$  na podstawie  $X$ .

# Proces uczenia

1. Dzielimy dane na batche,
2. Każdy batch przepuszczamy przez sieć (*forward pass*),
3. Obliczamy wartość funkcji straty  $J$ ,
4. Obliczamy gradienty  $J$  po wagach (*propagacja wsteczna, backpropagation*),
5. Zmieniamy wartości wag.

# Proces uczenia

Potrzebujemy:

- **funkcji straty** - miara, jak dobrze radzi sobie nasza sieć
- **optimizera** - metoda zmiany wag w sieci

# Funkcja straty

## Loss function

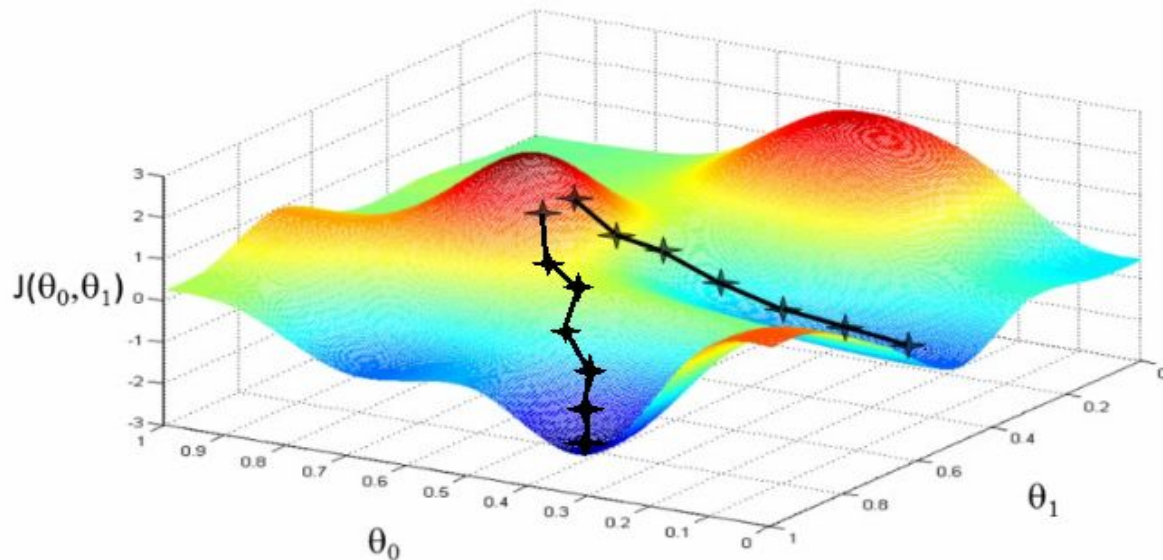
Musimy mierzyć jak dobrze radzi sobie sieć. Np. tak:

$$J(W) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)$$

$$\hat{y}_i = f(W_1 \cdot f(W_2 \cdot f(W_3 \cdot x_i)))$$

# Stochastic gradient descent

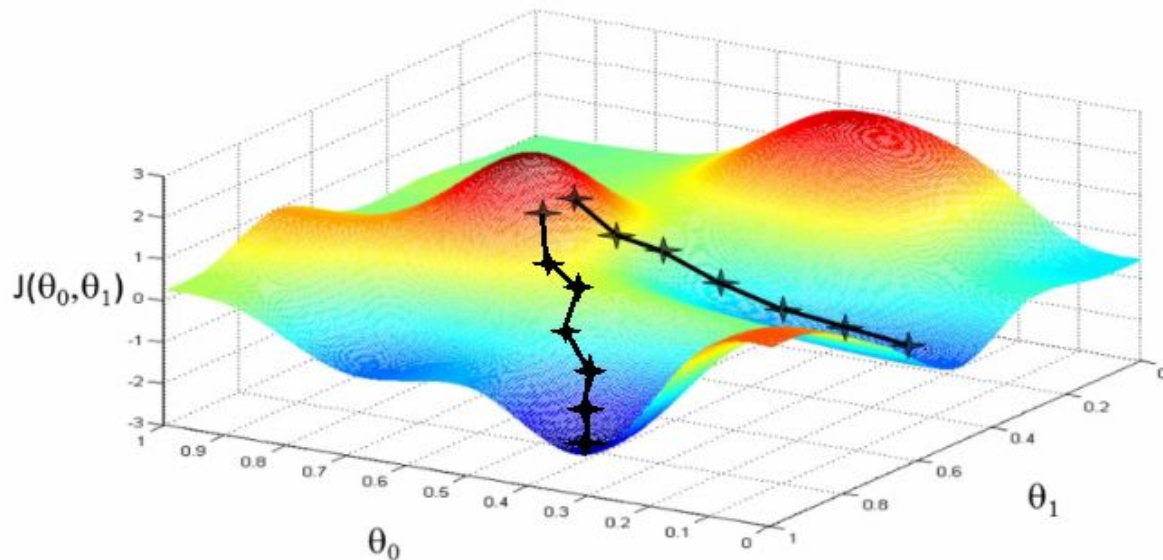
$$W_{ij} = W_{ij} - \alpha \frac{\delta J}{\delta W_{ij}}$$



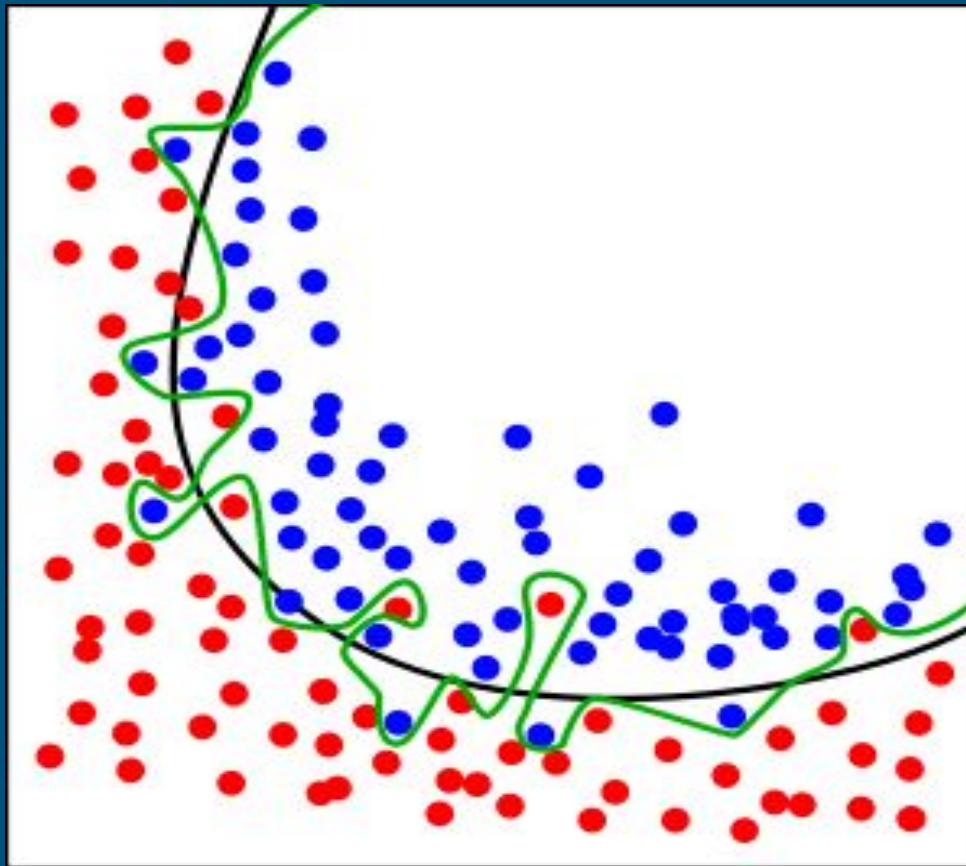
# Stochastic gradient descent

Learning rate

$$W_{ij} = W_{ij} - \alpha \frac{\delta J}{\delta W_{ij}}$$

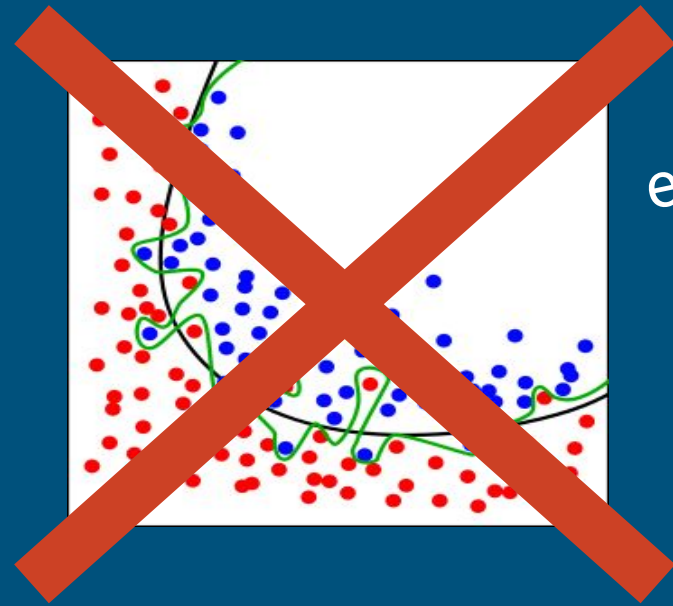


# Overfitting



pamiętacie jeszcze, prawda?

# Overfitting co robić, jak zwalczać?



regularyzacja  
dropout  
early stopping  
więcej danych  
i inne



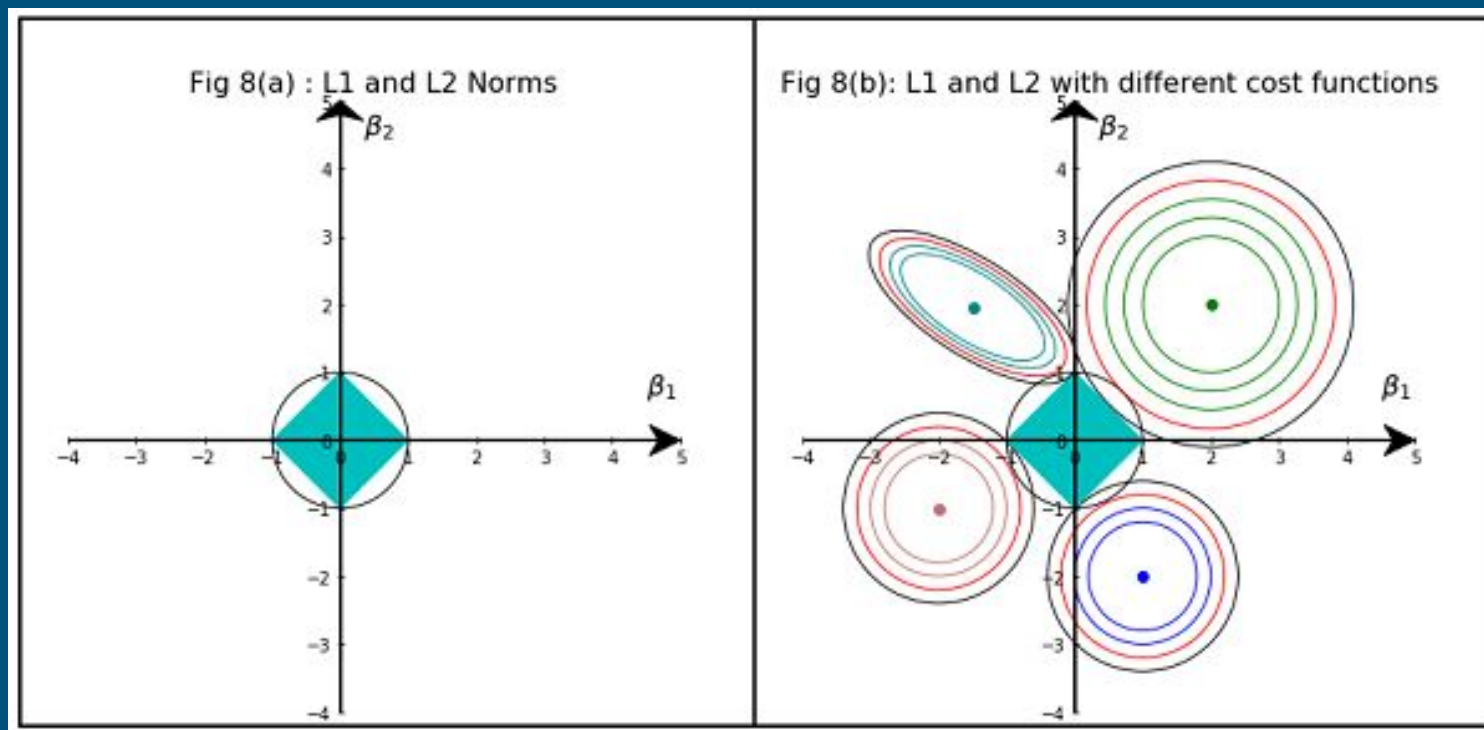


# Regularyzacja

- karajmy sieć za zbyt duże wagi
- dodając coś do funkcji straty

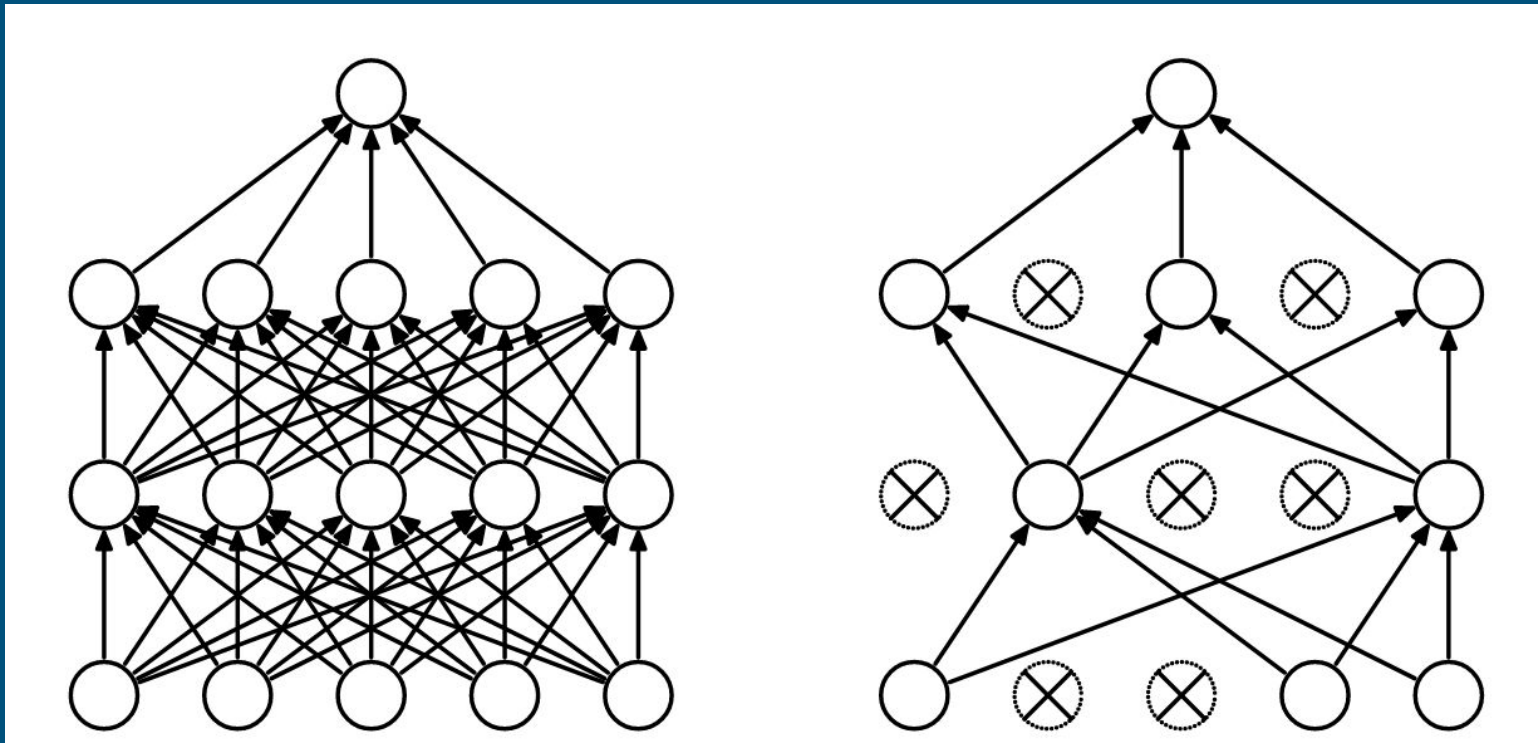
- L1: 
$$J(W) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^k |W_i|$$
- L2: 
$$J(W) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^k W_i^2$$

# Regularyzacja



# Dropout

- podczas treningu zerujemy losowe wagi z danym prawdopodobieństwem



Pytania?

# Źródła

<https://www.geeksforgeeks.org/decision-tree/>

<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

[https://grey.colorado.edu/mediawiki/sites/CompCogNeuro/images/thumb/5/53/fig\\_neuron\\_as\\_detect.png/400px-fig\\_neuron\\_as\\_detect.png](https://grey.colorado.edu/mediawiki/sites/CompCogNeuro/images/thumb/5/53/fig_neuron_as_detect.png/400px-fig_neuron_as_detect.png)

<https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/03/06100746/grad.png>

[http://www.conowego.pl/uploads/pics/Miniaturka\\_01.jpg](http://www.conowego.pl/uploads/pics/Miniaturka_01.jpg)

<http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf>

<https://i.stack.imgur.com/gzrsx.png>

<https://towardsdatascience.com/regularization-in-machine-learning-connecting-the-dots-c6e030bfadd>

<https://i.stack.imgur.com/19Cmk.gif>

<https://www.codeproject.com/KB/recipes/879043/GradientDescent.jpg>

<https://upload.wikimedia.org/wikipedia/commons/thumb/1/19/Overfitting.svg/300px-Overfitting.svg.png>

[https://upload.wikimedia.org/wikipedia/commons/thumb/6/60/ArtificialNeuronModel\\_english.png/600px-ArtificialNeuronModel\\_english.png](https://upload.wikimedia.org/wikipedia/commons/thumb/6/60/ArtificialNeuronModel_english.png/600px-ArtificialNeuronModel_english.png)

<https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7>