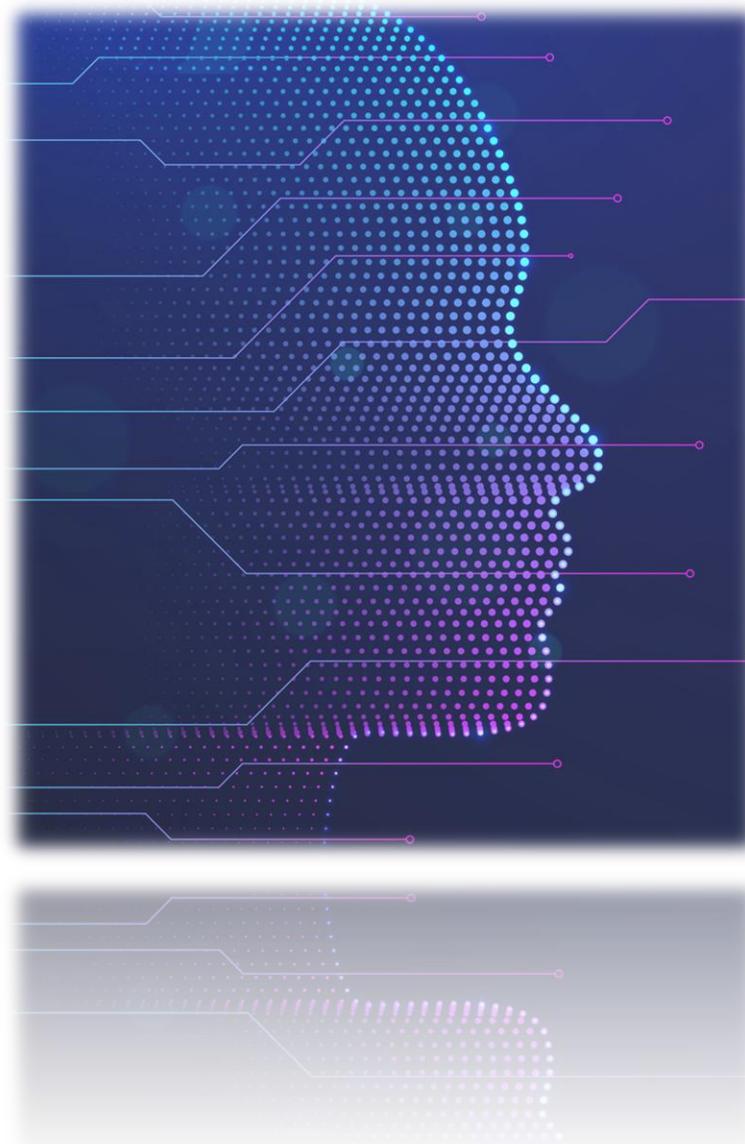


# Inteligencja obliczeniowa w analizie danych



**Wstęp do  
sztucznych  
sieci neuronowych**

Prof. dr hab. inż. Norbert Skoczylas

# Gdzie jesteśmy

## Algorytmy heurystyczne

Algorytmy probabilistyczne

Algorytmy genetyczne

Strategie ewolucyjne

Metody roju cząstek,  
mrówkowe, symulowane  
wyżarzanie...

## Logika rozmyta

Rozmyte systemy wnioskujące

Systemy eksperckie

Sterowanie rozmyte

## Sztuczne sieci neuronowe

Sieci neuronowe – teoria

Sieci neuronowe – typowe  
aplikacje

Głębokie sieci neuronowe

Sieci neuronowe – aplikacje  
inżynierskie

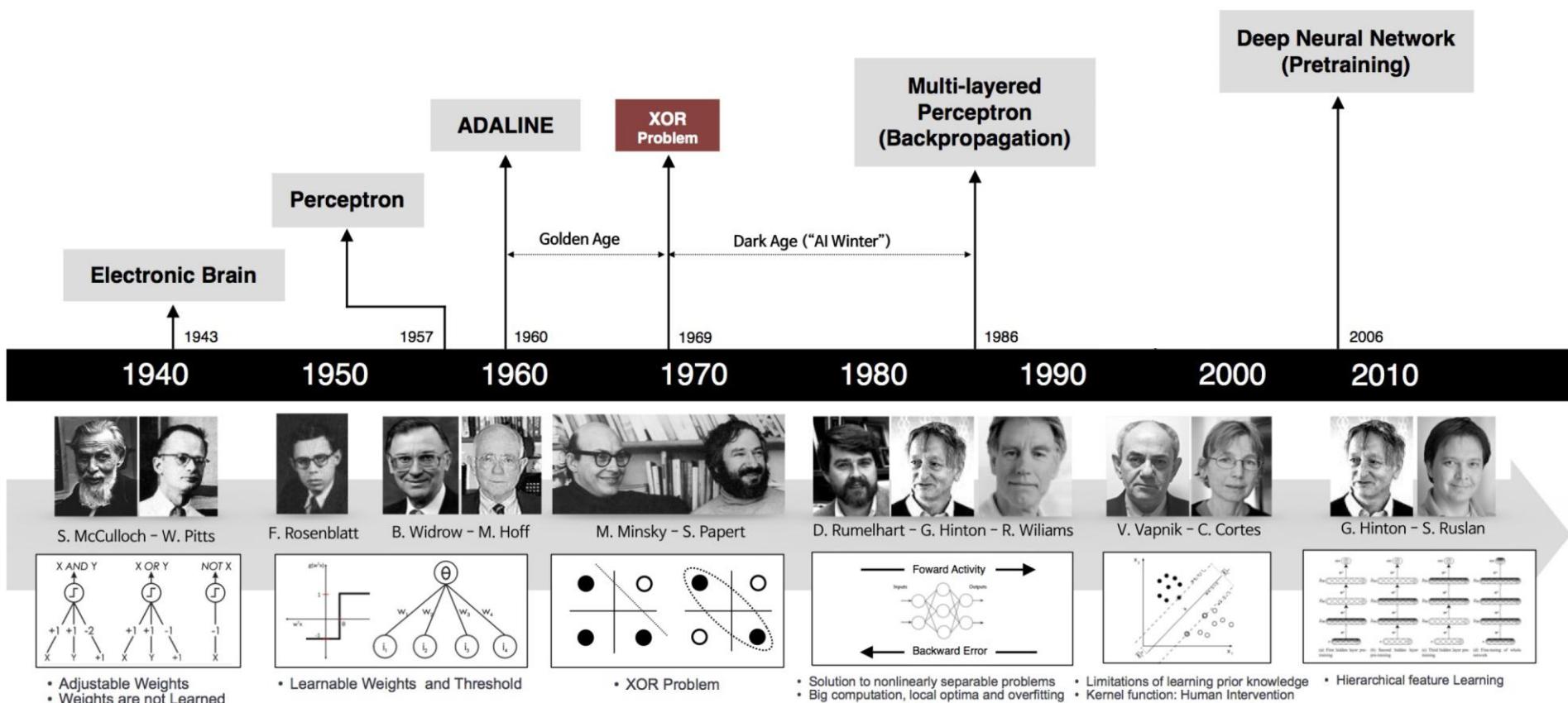
## Filary inteligencji Obliczeniowej

Algorytmy  
heurystyczne

Logika  
rozmyta

Sztuczne  
sieci neuronowe

# Historia..

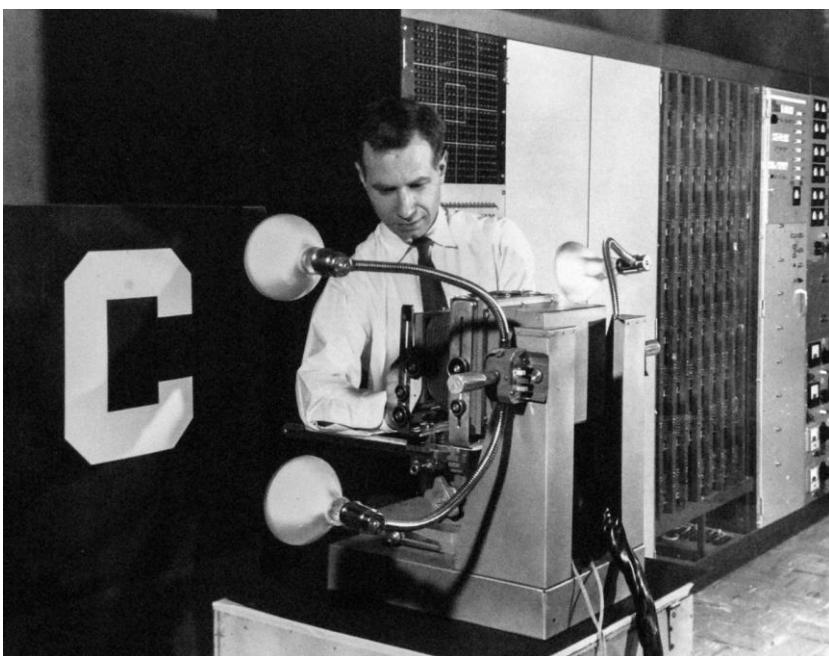
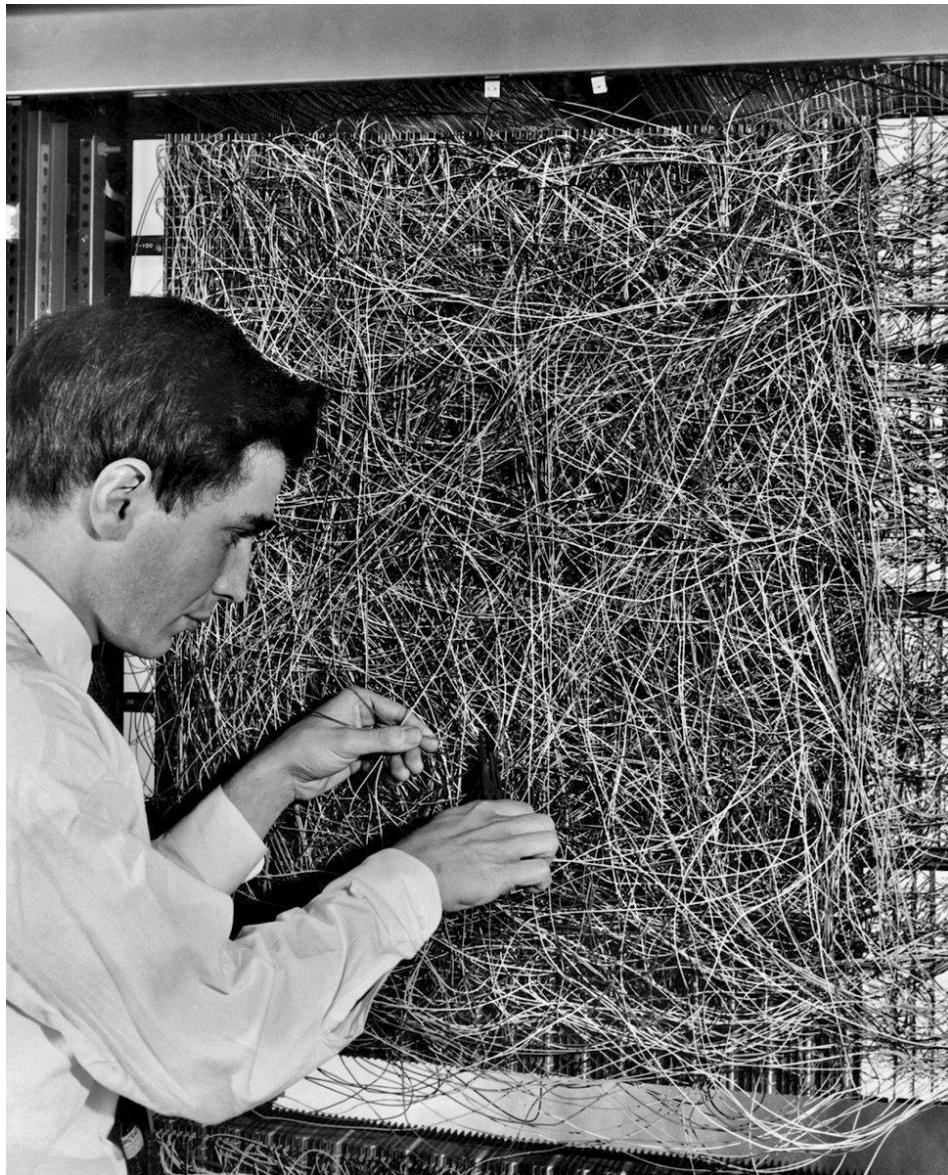


*Jak to się zaczęło ..*



## Rosenblatta

Elektroniczny „perceptron” – 256 elektronicznych neuronów – wagi były potencjomertami



## **Nazwijmy najważniejsze fakty ..**

Sieć neuronowa to urządzenie techniczne lub algorytm, którego działanie wzorowane jest w pewnym stopniu na działaniu sieci biologicznych komórek nerwowych.

Zazwyczaj składa się z siatki połączonych ze sobą elementów, z których każdy posiada pewną liczbę wejść i jedno wyjście.

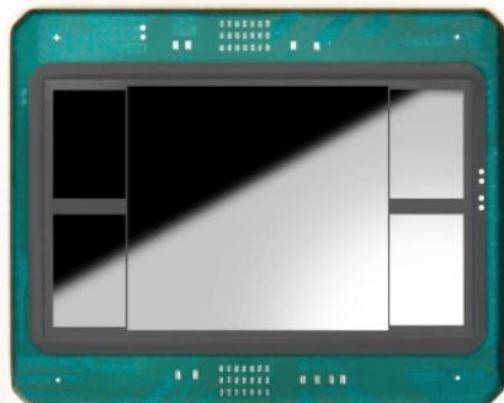
Wyjścia z poszczególnych elementów są połączone z wejściami innych tworząc sieć.

Zależność pomiędzy wejściami i wyjściem jest modyfikowana dla każdego elementu z osobna w procesie uczenia.

Zależność pomiędzy sygnałem wejściowym a wyjściowym całej sieci jest interpretowana jako rozwiązanie problemu.

# **Nazwijmy najważniejsze fakty ..**

intel® Nervana™ NNP-T



Implementacje hardwarowe są znacznie szybsze, jednak mniej elastyczne ..

Często problem zaczyna się rozwiązywać za pomocą implementacji algorytmicznych, natomiast aplikacje wymagające dużych mocy obliczeniowych powierza się procesorom neuronowym ..

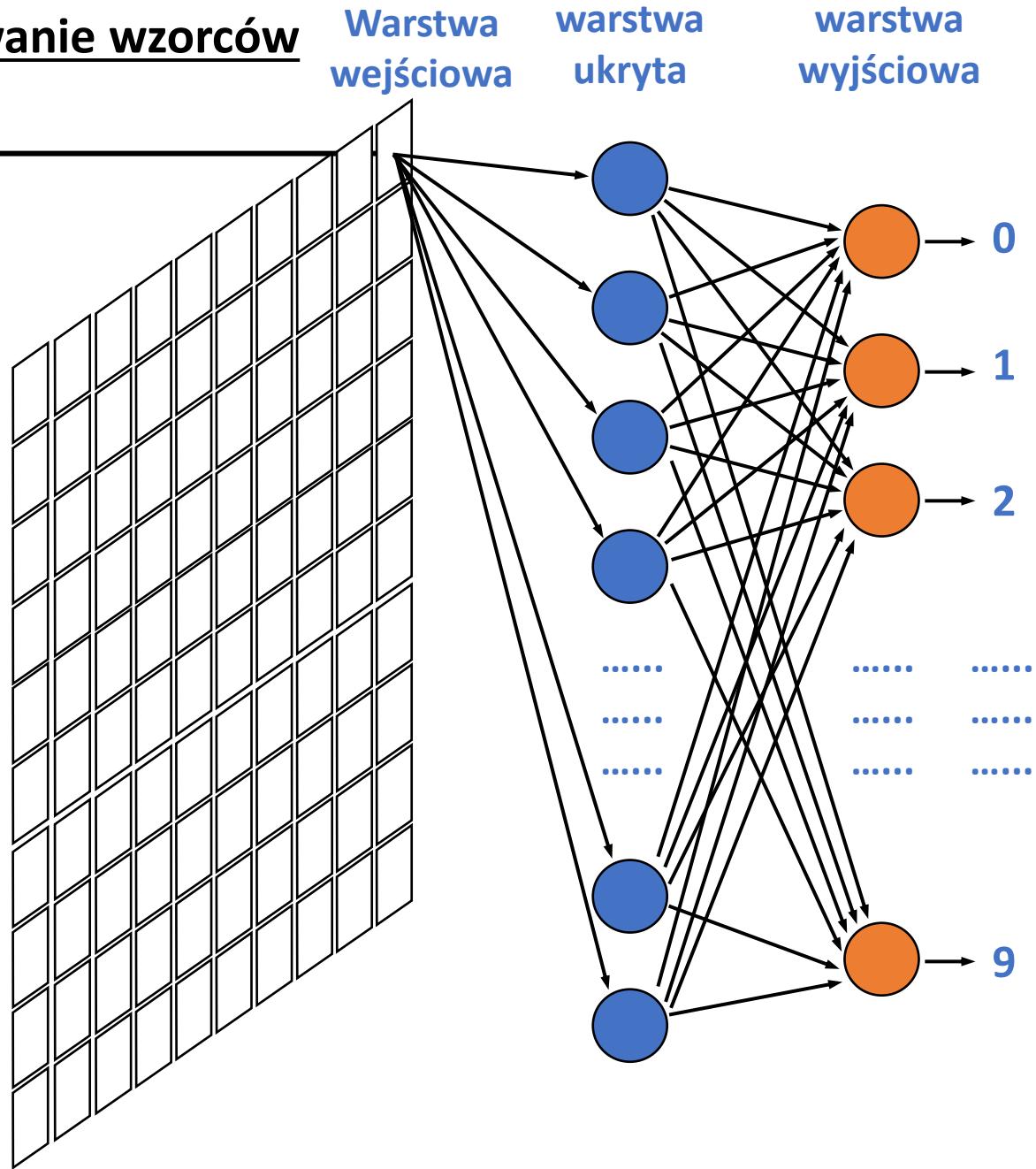
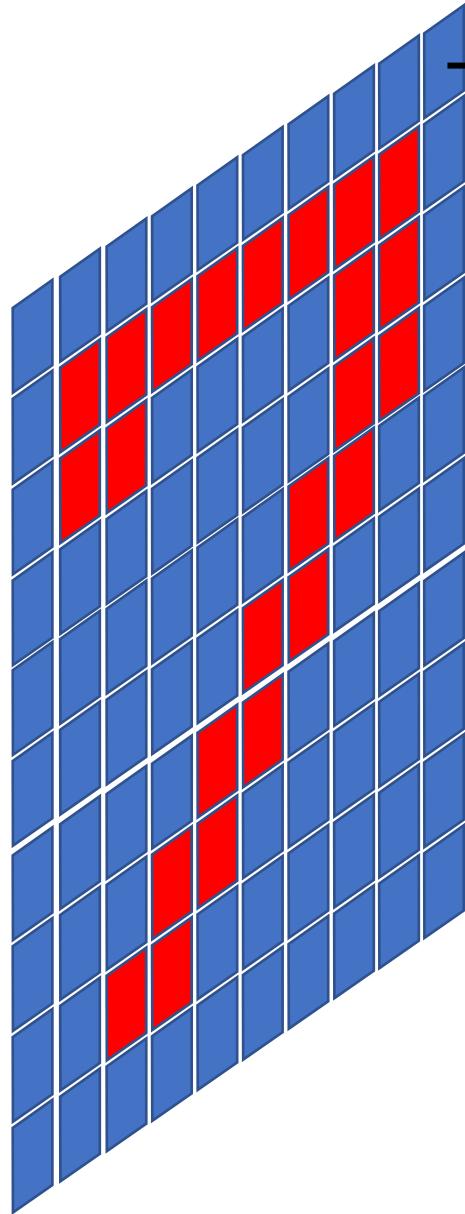
Procesory neuronowe spotkamy w urządzeniach do przetwarzania obrazu oraz elementach automatyki pracujących w czasie rzeczywistym ..

np... rozdzielcość 4K, nagrywanie filmów portretowych



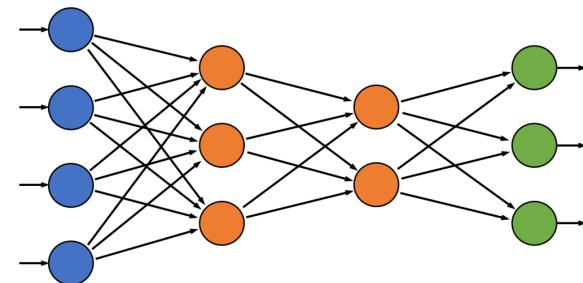
# Najczęstsze zastosowania ..

## Klasyfikacja i rozpoznanie wzorców



# Najczęstsze zastosowania ..

## Modelowanie



Model wykonany za pomocą sieci neuronowej – prosty w realizacji – staramy się odtworzyć pracę obiektu, ucząc sieć na podstawie rejestrowanych parametrów pracy systemu w różnych sytuacjach

Model matematyczny (fizyczny) zjawiska. Dzięki niemu możemy zrozumieć jak działa obiekt ... Ale najczęściej jest trudny do wykonania

$$(\omega) = \frac{1}{\pi} \int f(t) \cdot \cos(\omega t) dt$$
$$(\omega) = \frac{1}{\pi} \int f(t) \cdot \sin(\omega t) dt$$
$$(t) = \int_0^\infty a(\omega) \cdot \cos(\omega t) + b(\omega) \cdot \sin(\omega t)$$
$$a(\omega) = \frac{1}{\pi} \int f(t) \cdot \cos(\omega t) dt$$
$$b(\omega) = \frac{1}{\pi} \int f(t) \cdot \sin(\omega t) dt$$
$$C_n = \frac{1}{\pi} \int f(t) e^{-jn\pi t} dt$$
$$f(t) = \sum_{n=-\infty}^{\infty} C_n \cdot e^{jn\pi t}$$
$$C(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$
$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} C(\omega) e^{j\omega t} d\omega$$
$$u(t) = \begin{cases} 1, & t > 0 \\ 0, & t \leq 0 \end{cases}$$
$$\hat{F}[a \cdot f(t) + b \cdot g(t)] = a \hat{f}(\omega) + b \hat{g}(\omega), \quad a, b \in \mathbb{R}$$
$$f(t) = \sqrt{a(\omega)} \cdot \cos(\omega t) + b(\omega) \cdot \sin(\omega t)$$

# Najczęstsze zastosowania ..

## Modelowanie



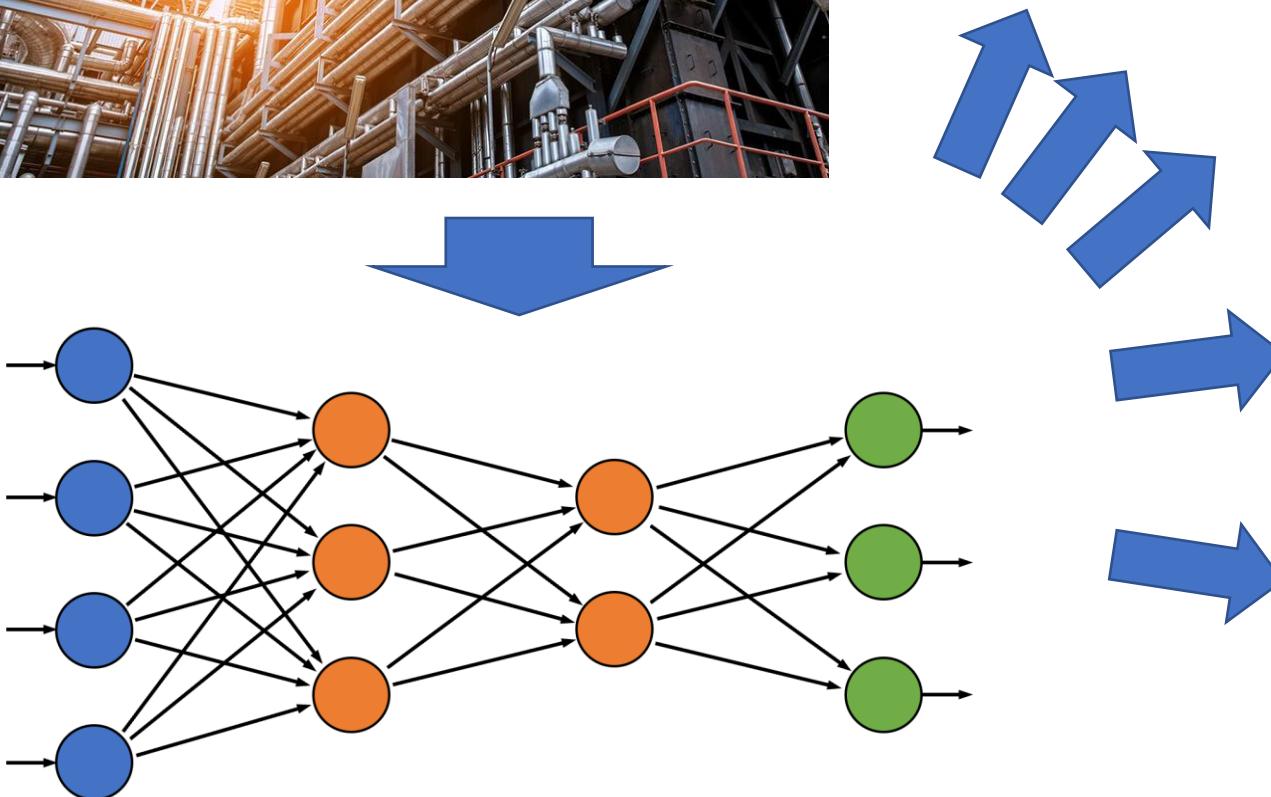
Prognozowanie



Projektowanie



Sterowanie



Kontrola

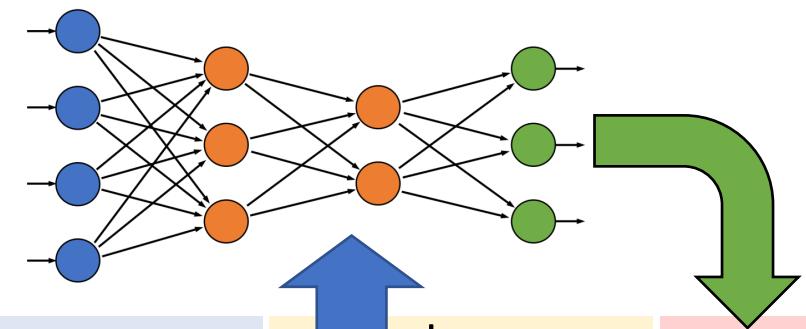
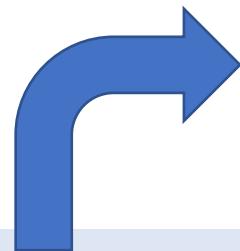


Diagnostyka

# Najczęstsze zastosowania ..

## Prognozowanie

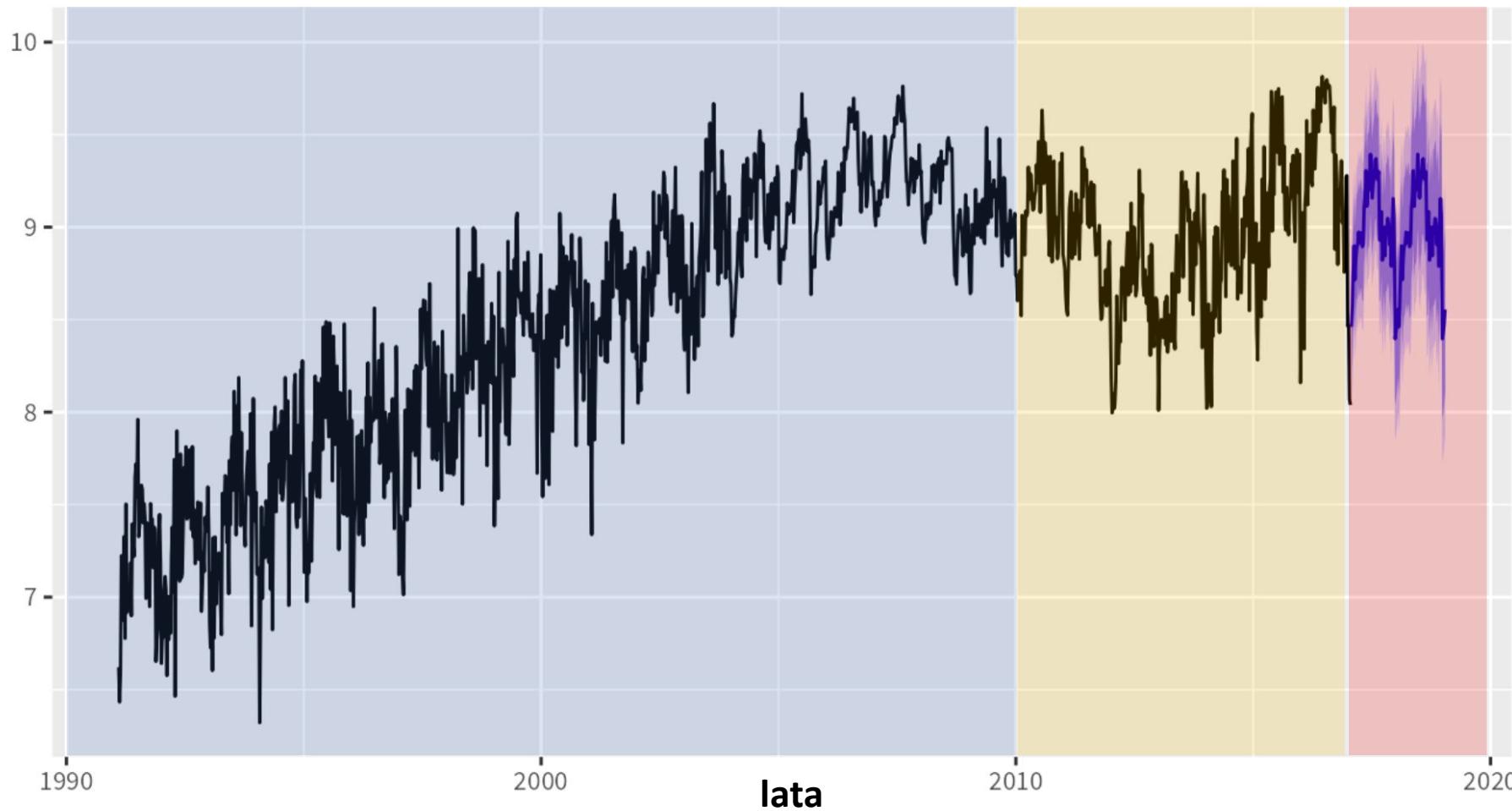
Bazując na wartościach szeregu czasowego z przeszłości prognozujemy przyszłość



dane, którymi uczymy sieć

dane wejściowe do prognozy

prognoza



# Najczęstsze zastosowania ..

## Przetwarzanie obrazów



before

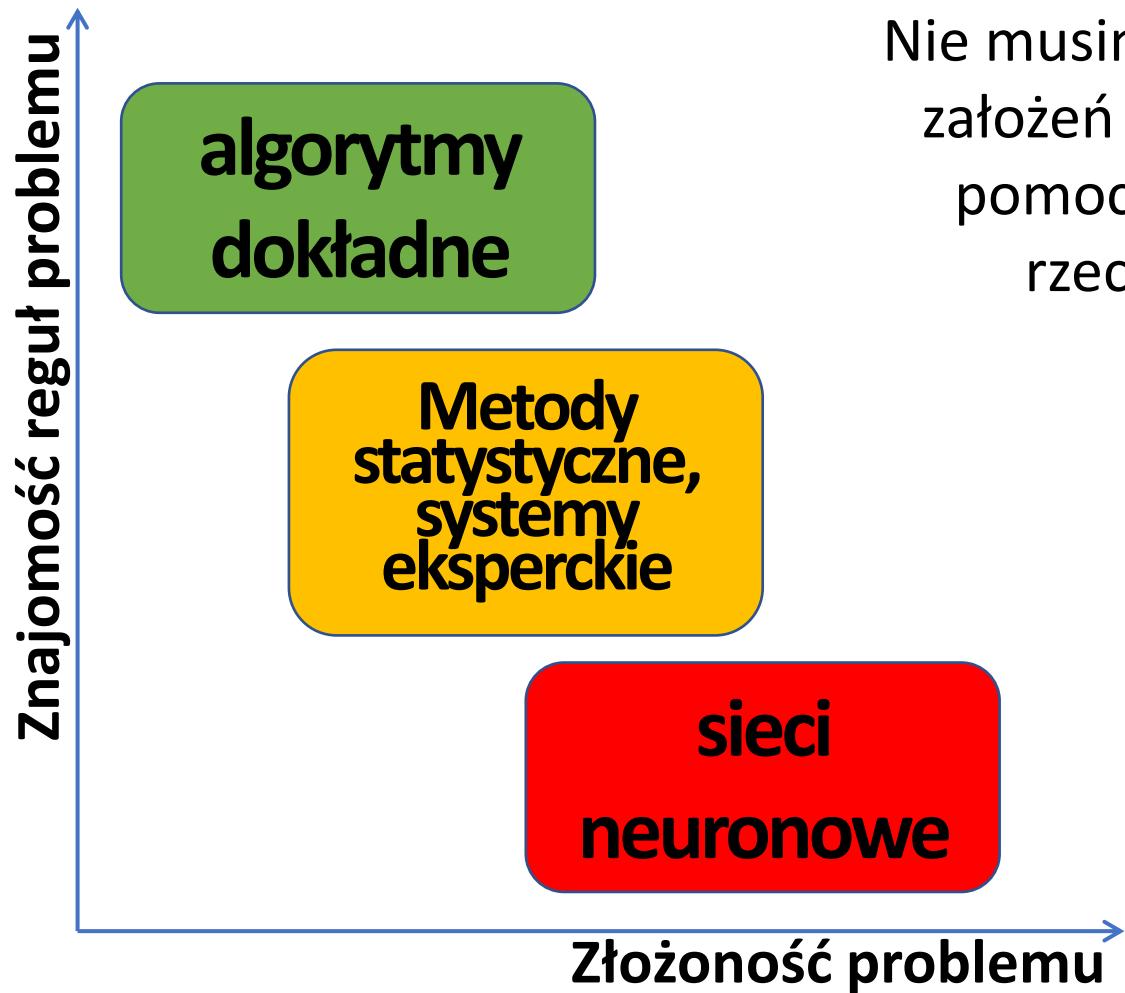
after



EDITED BY AI



*Najogólniej – kiedy sięgniemy po sieci .. jaką osiągniemy potencjalnie korzyść ..*



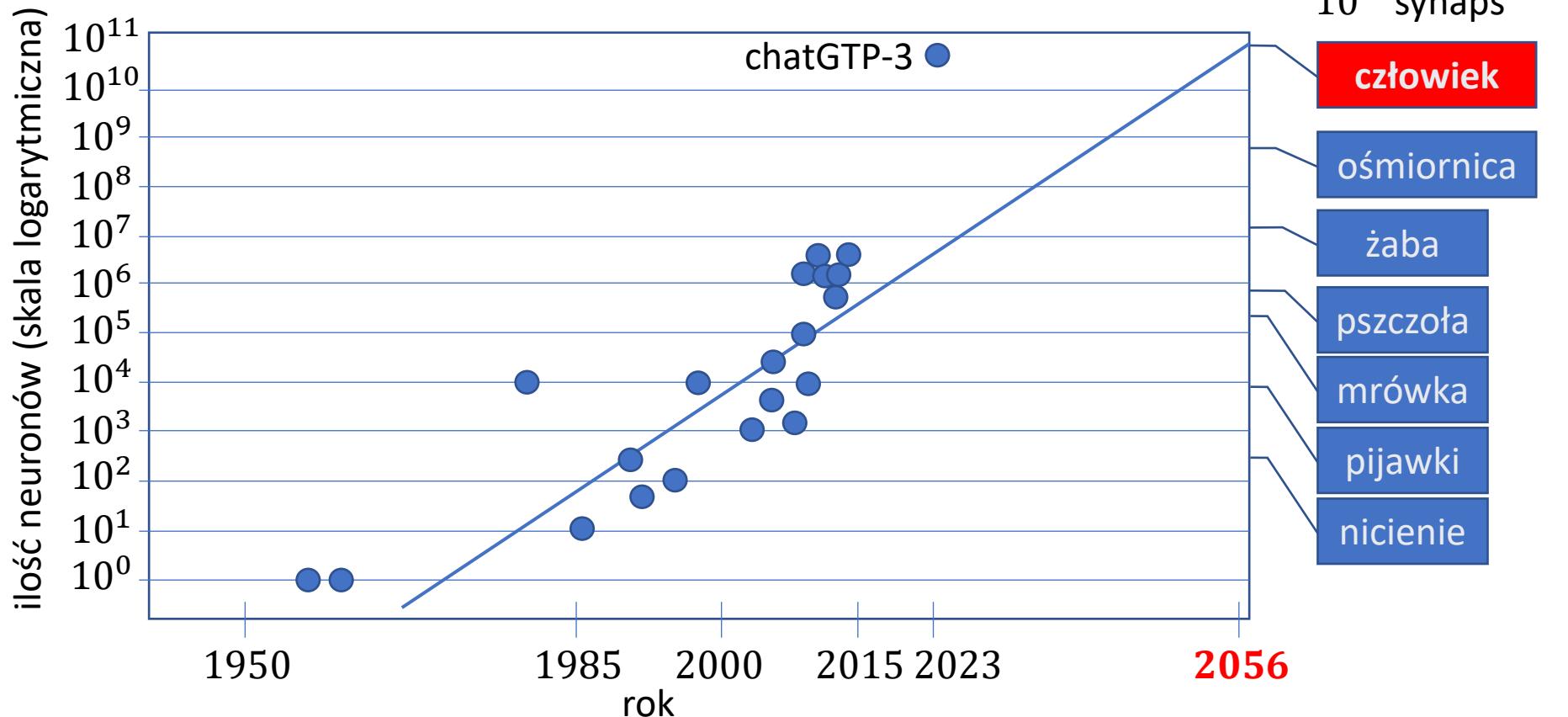
Nie musimy stawiać jakichkolwiek założeń dla modeli tworzonych z pomocą sieci – odzwierciedlają rzeczywistość niezależnie od złożoności ..

Nie musimy ograniczać się do problemów posiadających definicję formalną ..

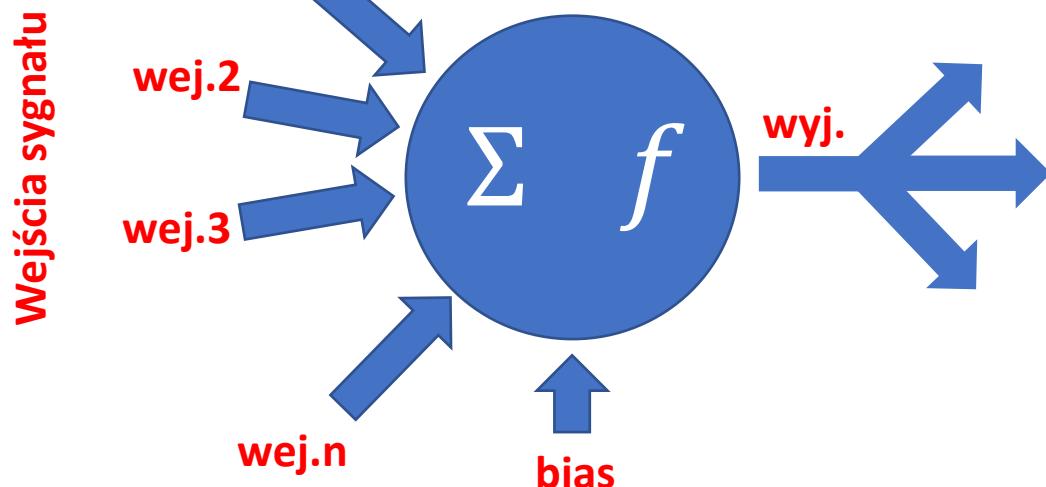
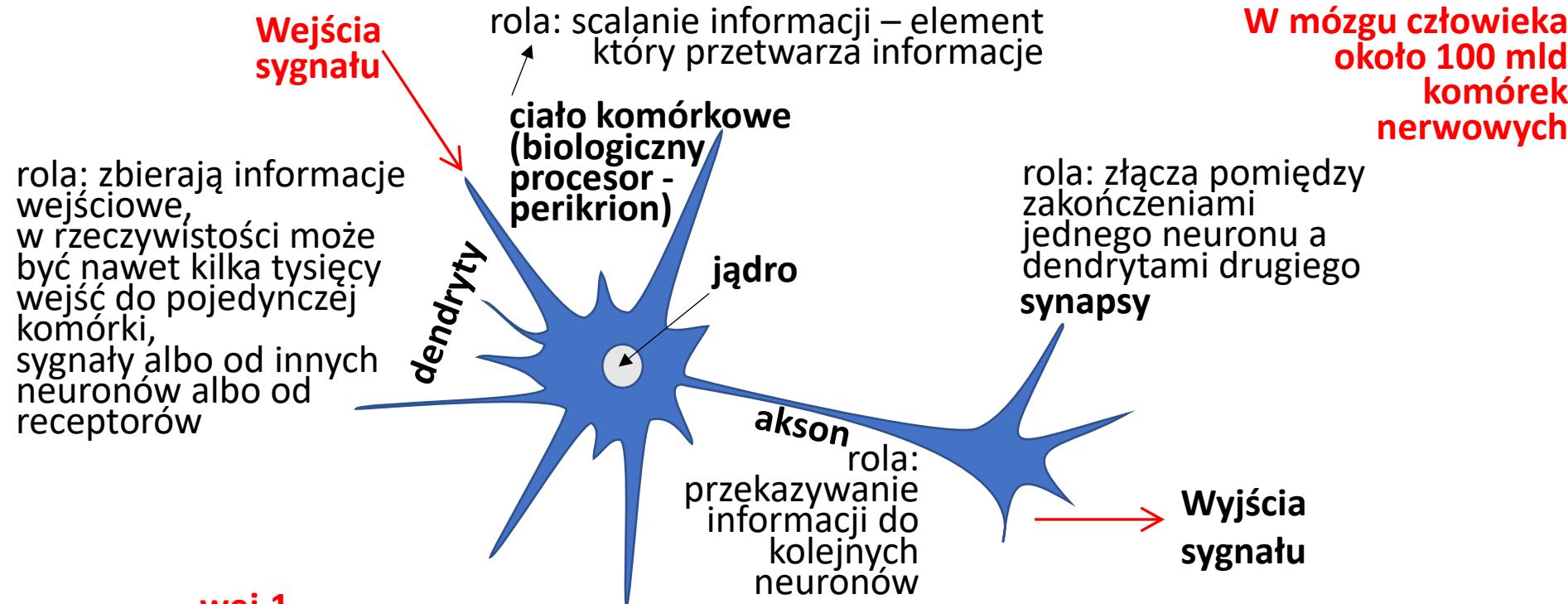
Zastępujemy proces programowania oraz „typowego – ręcznego” tworzenia modelu, procesem uczenia sieci..

# *Na jakim etapie rozwoju jesteśmy ..*

Liczba neuronów podwaja się co około 2.5 roku ...



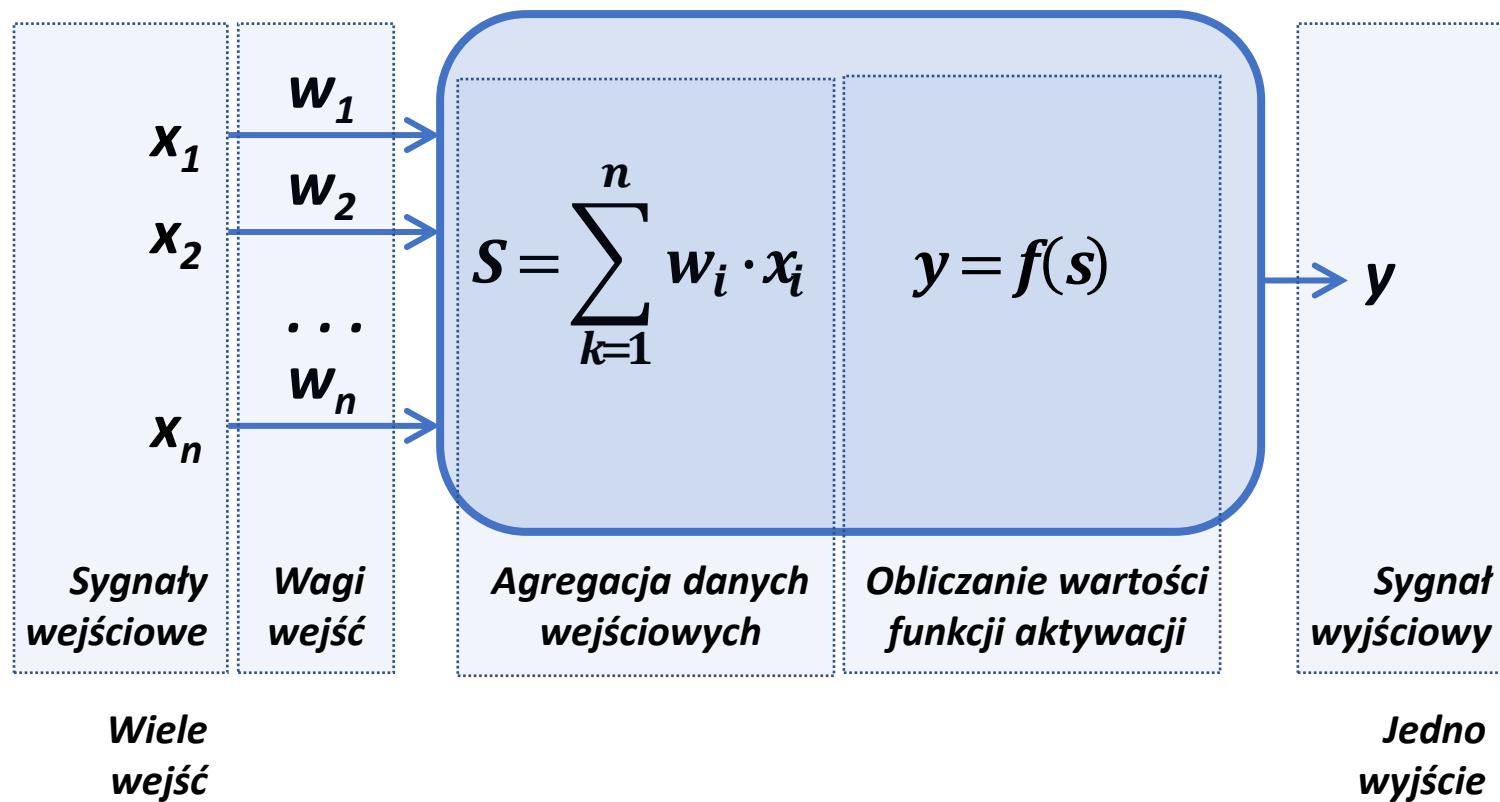
## **Podstawowa jednostka budulcowa – neuron ..**



**Wiele wejść i jedno wyjście**

W sztucznych sieciach neuronowych traktuje się sztuczny neuron jako jednostkę obliczeniową, która na podstawie określonej funkcji aktywacji oblicza na wyjściu pewną wartość na podstawie sumy ważonych danych wejściowych.

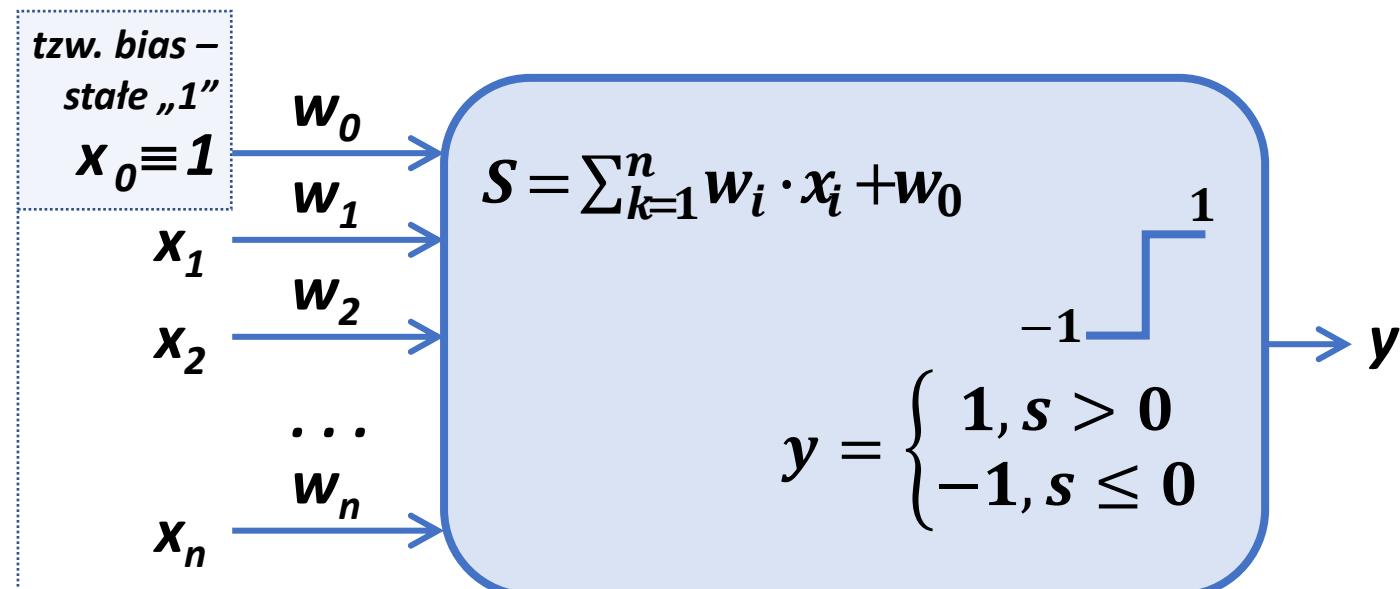
# Najbardziej ogólna koncepcja neuronu (perceptronu)..



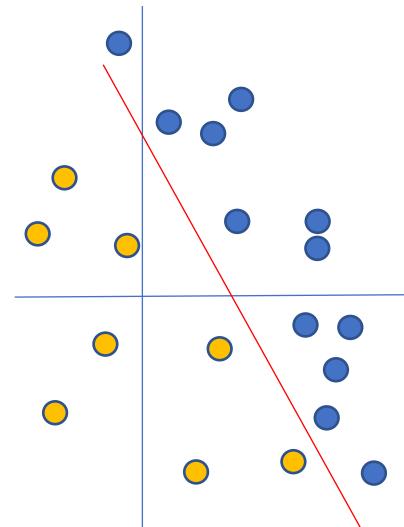
*Wagi mają zasadniczy wpływ na zachowanie neuronów  
– zmieniając wagę zmieniamy zachowanie neuronów  
- to jest mechanizm uczenia perceptronu*

# Perceptron prosty – neuron Rosenblatta (klasyfikator binarny)

$w_i$  - mogą być dodatnie (synapsy pobudzające) lub ujemne (synapsy hamujące) mogą być także zerowe – wtedy sygnał  $x_i$  nie ma wpływu na neuron

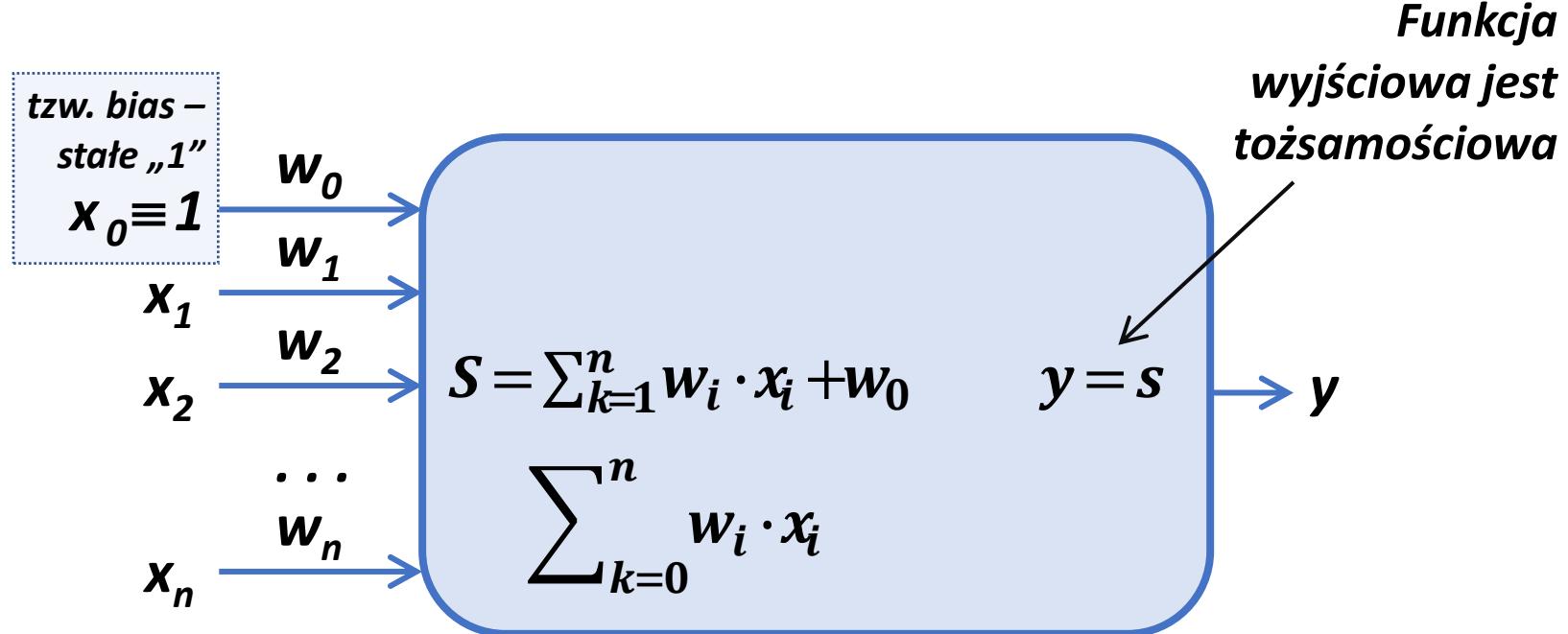


Zawsze wartość +1, podawany na dodatkowe wejście neuronu. Waga związana z BIASem podlega procesowi uczenia podobnie jak wszystkie inne wagi.



Interpretacja geometryczna: dwie klasy liniowo separowane są oddzielone przez prostą decyzyjną (w przypadku dwóch wejść) – bez biasu przechodzi ona przez początek układu współrzędnych ..  
W przypadku n-wymiarowym mamy do czynienia z hiperpłaszczyzną decyzyjną ..

# Neuron liniowy – najprostszy, ale często przydatny

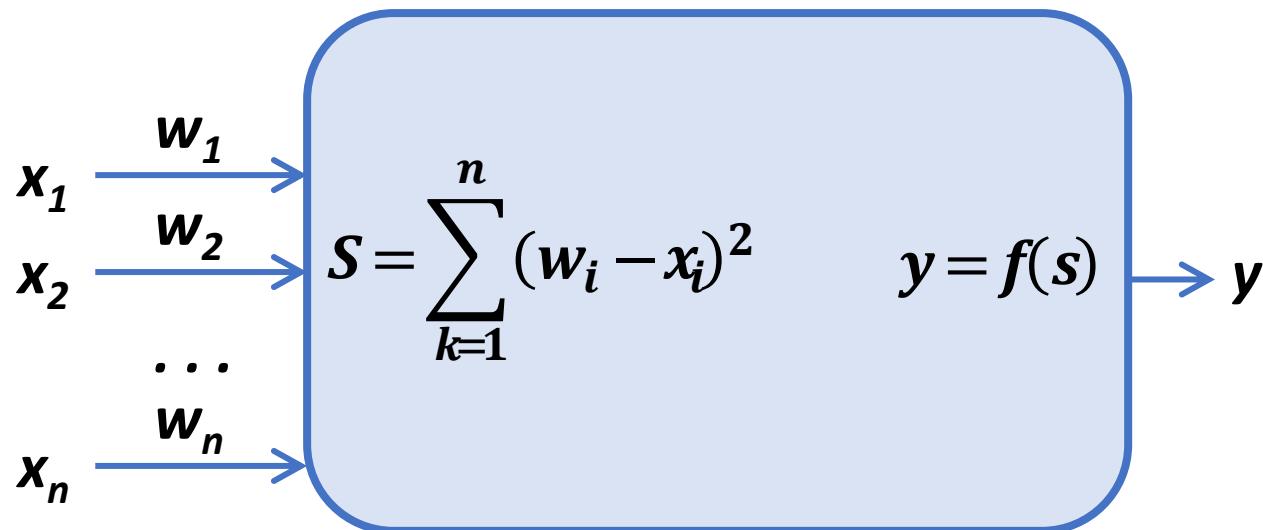


*Neuron liniowy zawsze będzie generował sygnał wyjściowy przechodzący przez początek układu współrzędnych – aby to ograniczenie wyeliminować stosuje się sygnał *bias'u* – stała wartość 1 na jednym z wejść ..*

Z neuronów liniowych można budować tylko sieci liniowe  
- użyteczne, ale mające ograniczone zastosowanie.

Sieć liniowa, budowana z neuronów liniowych, z zasady nie posiada warstw ukrytych, bo nawet jeśli się je wprowadzi, to nie wzbogacą one zachowania sieci

## **Neuron nieliniowy – radialny**

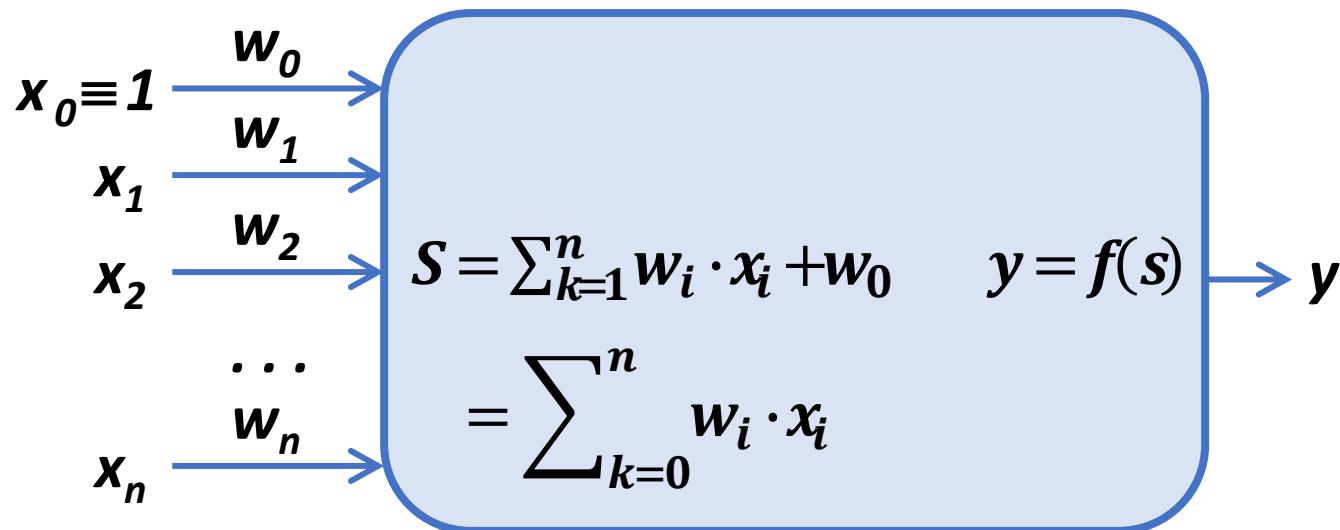


Neurony radialne stosowane są w sieciach radialnych **RBF** (Radial Basis Functions)

**Struktura neuronu** radialnego zakłada użycie radialnej agregacji danych wejściowych oraz **funkcji Gaussa jako funkcji aktywacji**.

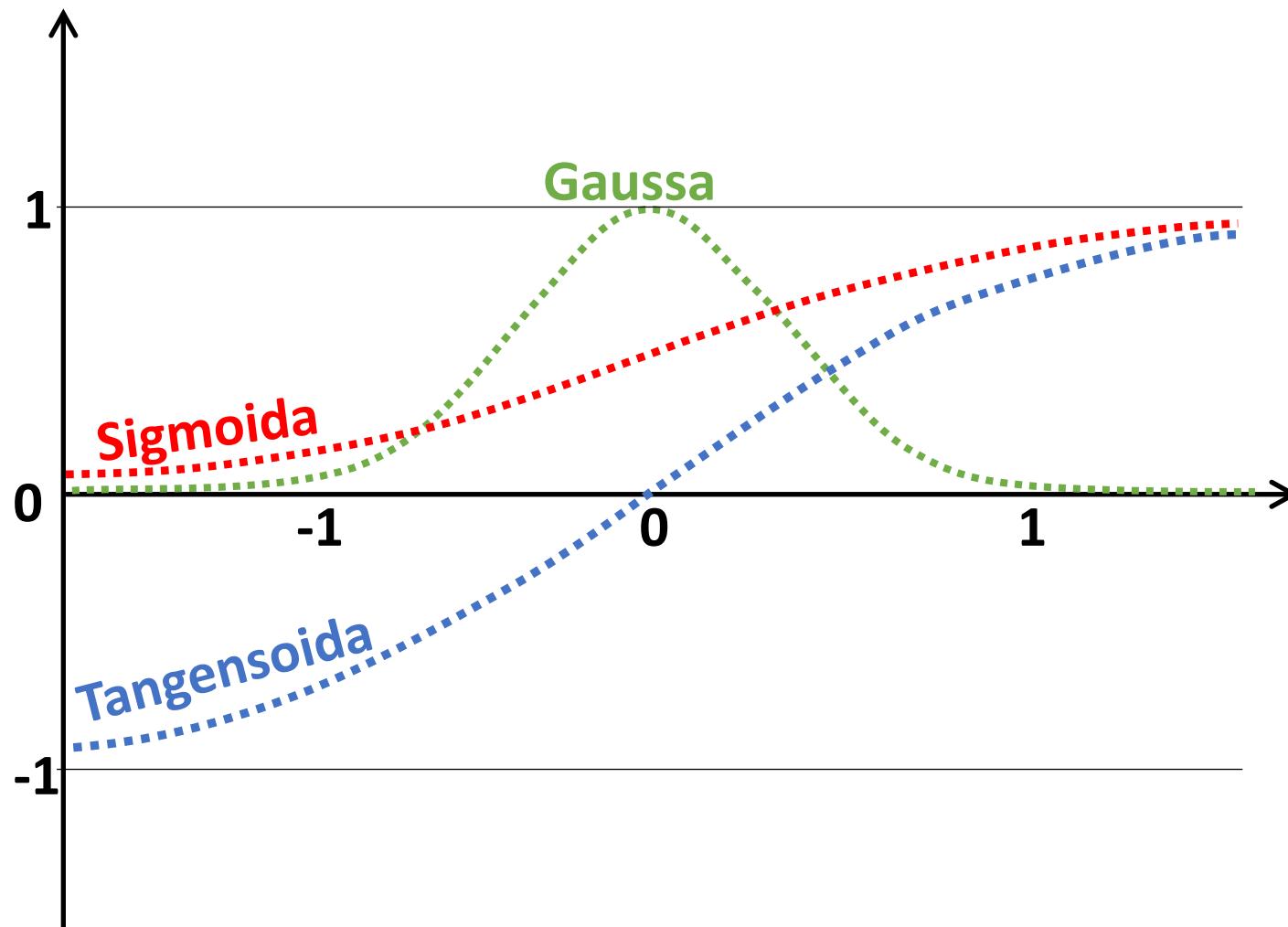
→ Rodzaj jednokierunkowej sieci neuronowej z neuronami radialnymi - służą do rozpoznawania powtarzalnych i charakterystycznych cech grup (skupisk) danych wejściowych.

## *Neuron nieliniowy (z nieliniową funkcją aktywacji)*



Funkcja aktywacji wiąże pobudzenie neuronu,  
na które składają się sygnały wejściowe  
oraz wagi z sygnałem wyjściowym

# Różne postacie funkcji aktywacji (funkcji przejścia)



Nieliniowość neuronu zostaje wymuszona nieliniową funkcją przejścia ..

Pochodna funkcji przejścia także powinna być „przyjazna” .. bo występuje w formułach uczących sieć ..

Testowano mnóstwo różnych nieliniowych funkcji, ale do praktyki przeszły:

## Sigmoida

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

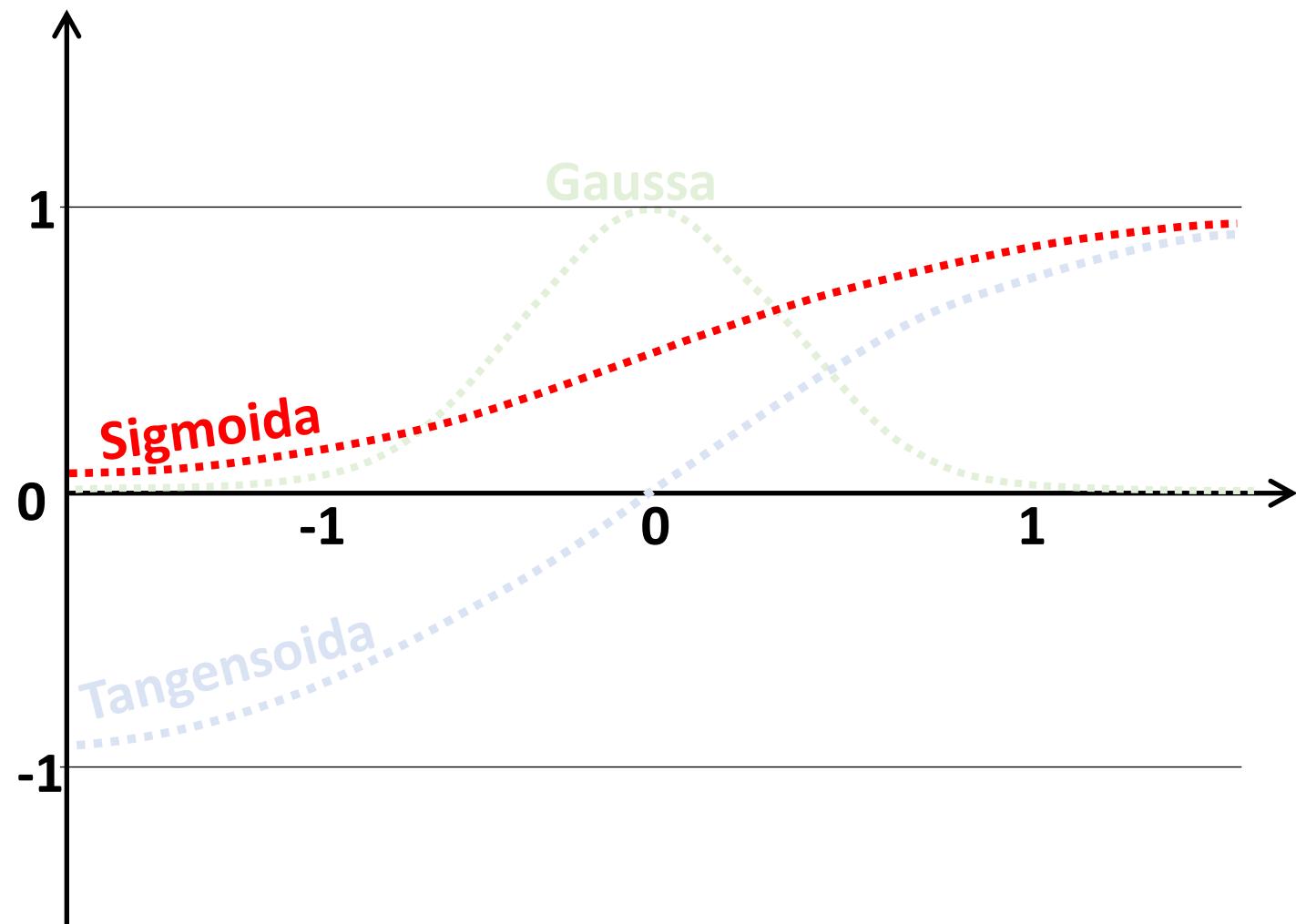
## Tangensoida

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$f'(x) = f(x) - f(x)^2$$

## Gaussa

$$f(x) = e^{-x^2}$$
$$f'(x) = -2x \cdot f(x)$$

# Różne postacie funkcji aktywacji (funkcji przejścia)



Sigmoid jest najczęściej wykorzystywana.

Posiada 2 naturalne asymptoty.

Dla bardzo niskich wartości pobudzenia neuronu dąży do zera.

Dla bardzo wysokich wartości pobudzenia wchodzi w „nasycenie” i dalej już nie rośnie ..

Podobnie działa biologia ..

Sieci bazujące na sigmoidalnych funkachach przejścia bardzo dobrze działają.

## Sigmoida

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

$$f'(x) = f(x)(1 - f(x))$$

## Tangensoida

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

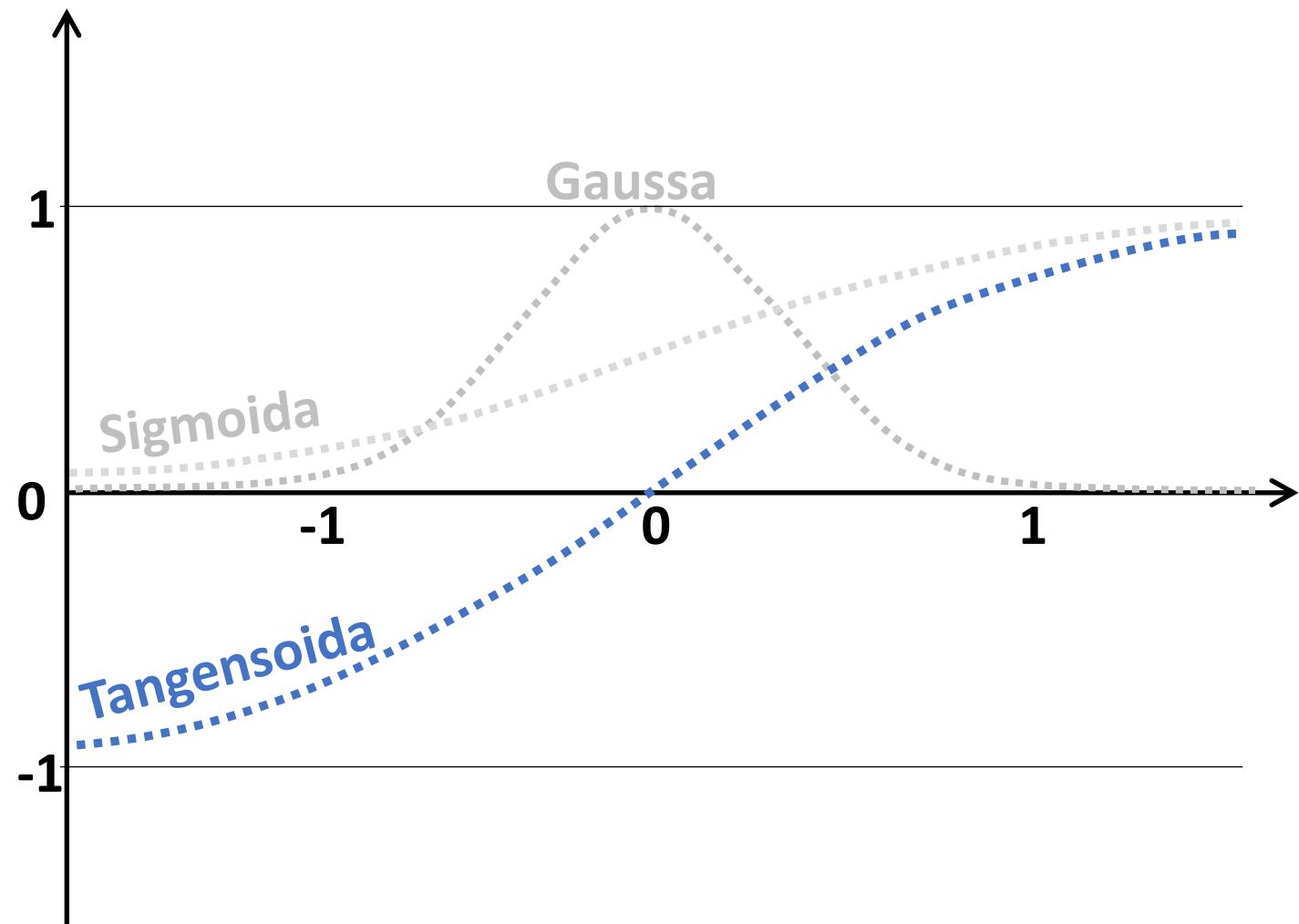
$$f'(x) = f(x) - f(x)^2$$

## Gaussa

$$f(x) = e^{-x^2}$$

$$f'(x) = -2x \cdot f(x)$$

# Różne postacie funkcji aktywacji (funkcji przejścia)



Sigmoida

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

Tangensoida

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$f'(x) = f(x) - f(x)^2$$

Gaussa

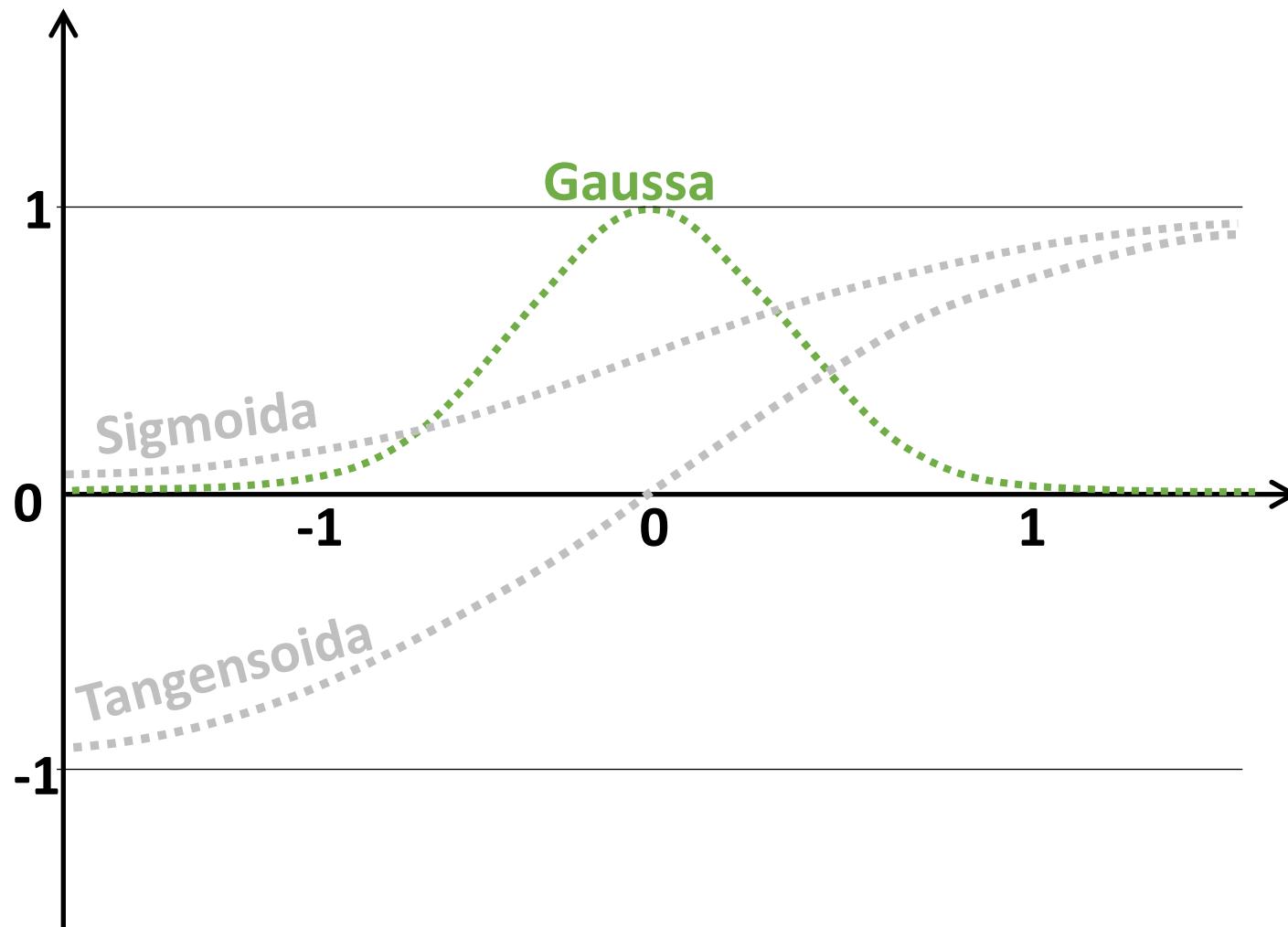
$$f(x) = e^{-x^2}$$
$$f'(x) = -2x \cdot f(x)$$

Wartości tej funkcji mogą być także ujemne ..  
Wszelkie badania wskazują, że w mózgu nie występują sygnały o ujemnym potencjale..

Przypuszczało, że dzięki temu będzie można zyskać „nową jakość”..

Taka funkcja czasem daje dobre rezultaty, ale w większości przypadków nie jest lepsza od Sigmoidy..

# Różne postacie funkcji aktywacji (funkcji przejścia)



Jest niemonotoniczna ..

W naturze takie sygnały także nie były obserwowane w biologicznych neuronach

Bywają przydatne czasami w specjalistycznych zastosowaniach

Sigmoida

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

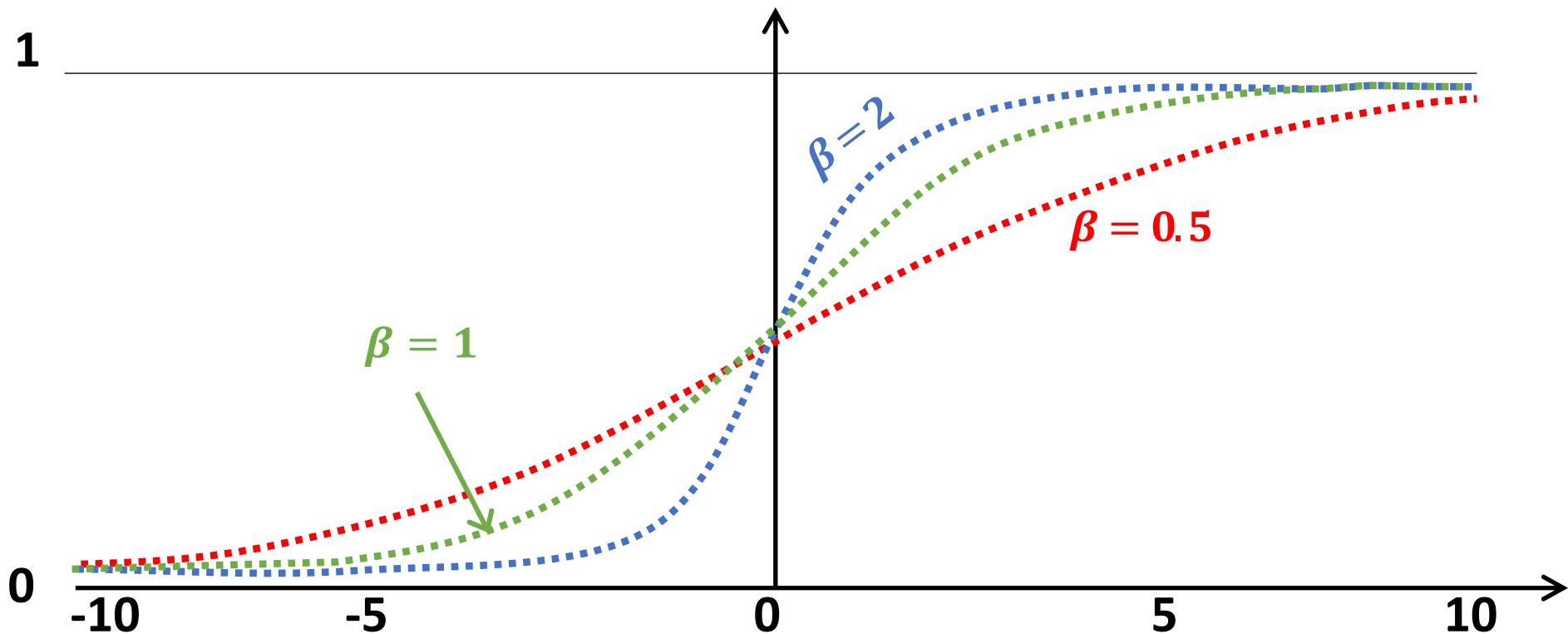
Tangensoida

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$f'(x) = f(x) - f(x)^2$$

Gaussa

$$f(x) = e^{-x^2}$$
$$f'(x) = -2x \cdot f(x)$$

## Różne postacie funkcji aktywacji (funkcji przejścia)



Sigmoida

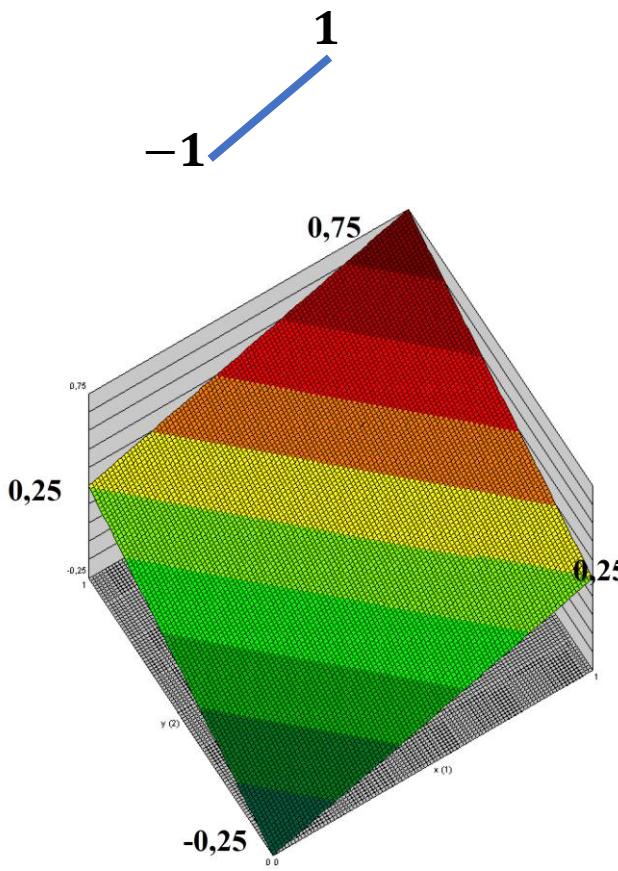
$$f(x) = \frac{1}{1 + e^{-\beta x}}$$

W zależności od parametru  $\beta$   
„współczynnik skalowania”)  
narastanie funkcji może być  
mniej lub bardziej dynamiczne..

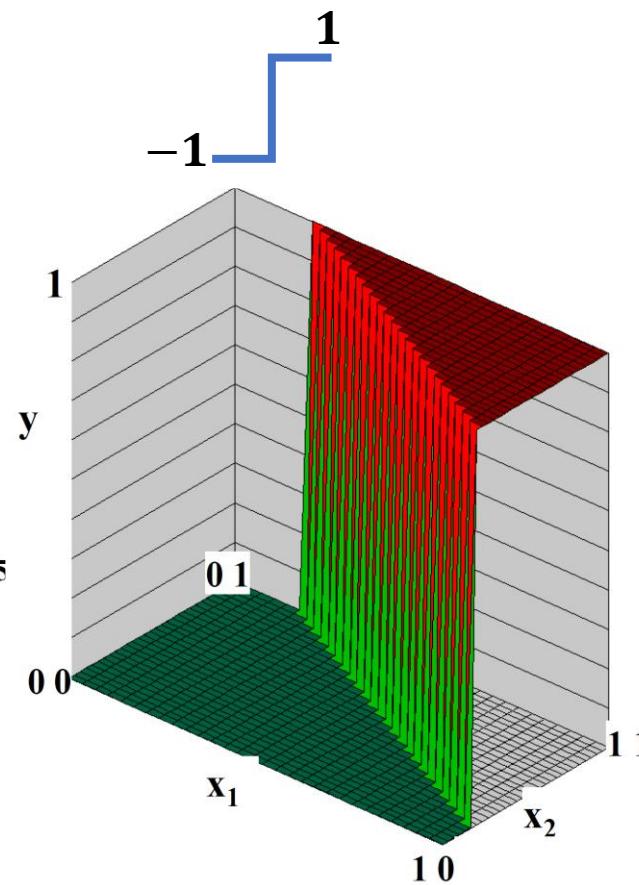
# Pojedyncze neurony – schematyczna ilustracja możliwości ..

Rozwiążanie „problemu AND” – interpretacja graficzna możliwości

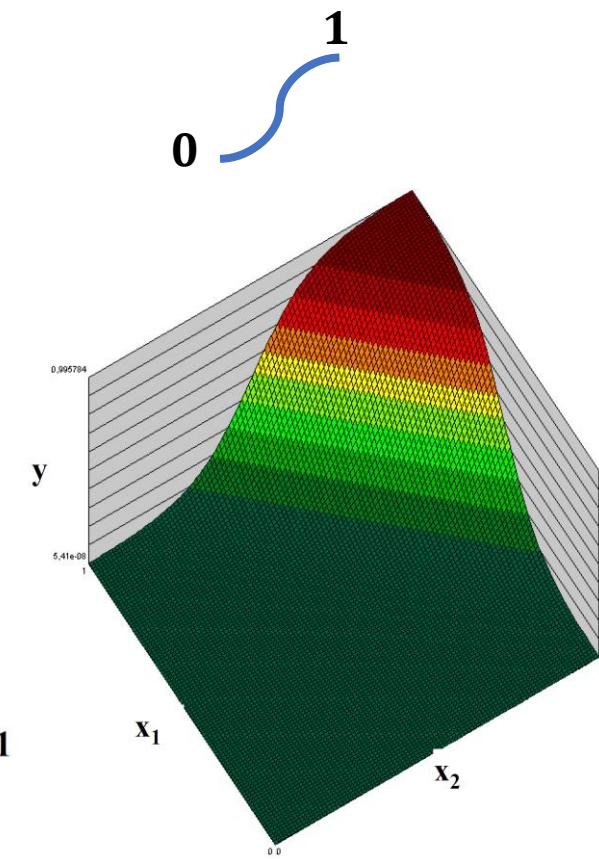
neuron liniowy,  
liniowa funkcja aktywacji



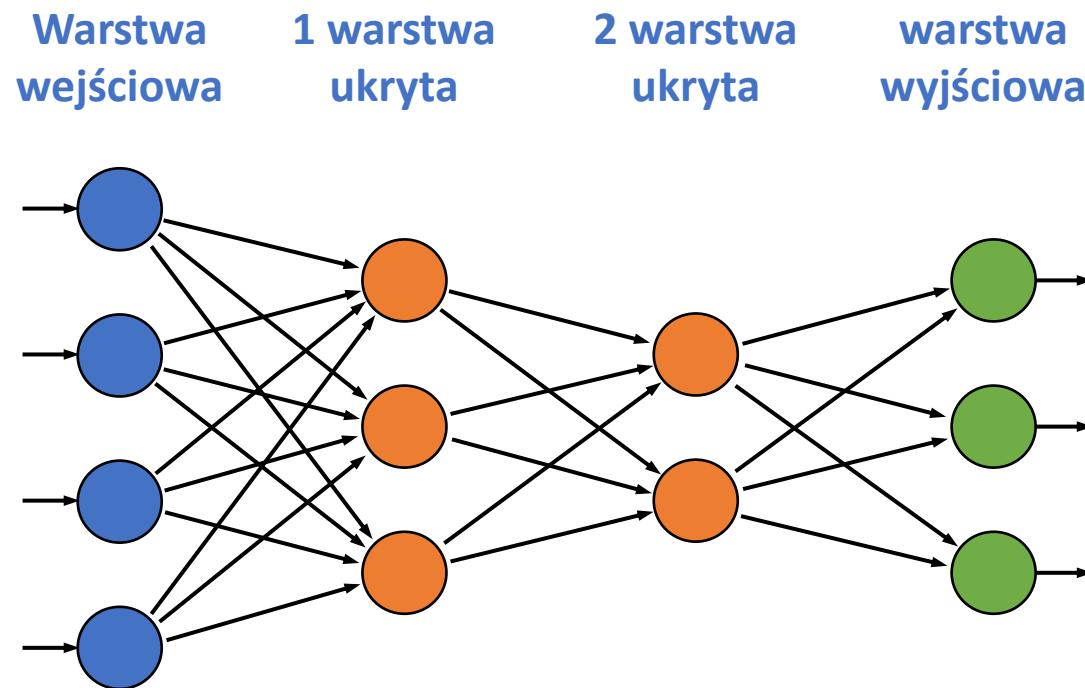
neuron Rosenblatta,  
skokowa funkcja aktywacji



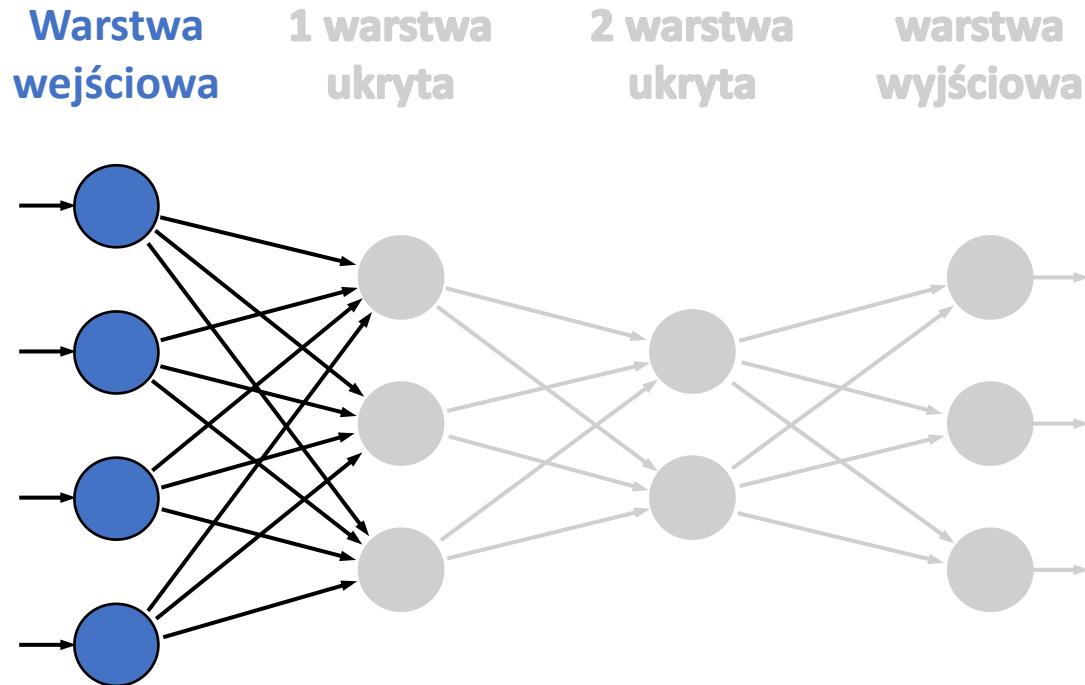
Neuron nieliniowy,  
sigmoidalna funkcja aktywacji



# *Warstwy w sieciach neuronowych*



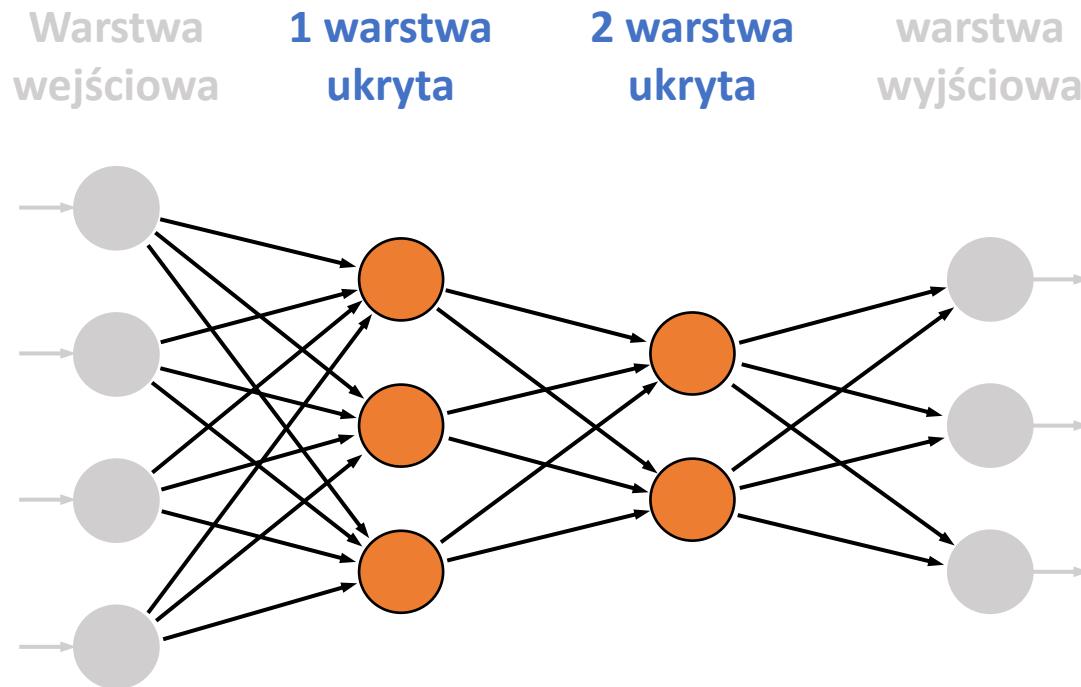
# Warstwy w sieciach neuronowych



Warstwa wejściowa nie uczestniczy bezpośrednio w wypracowywaniu odpowiedzi sieci.

Rozprowadzają one dane wejściowe w postaci sygnałów do neuronów pierwszej warstwy ukrytej.

# Warstwy w sieciach neuronowych



Warstwa nazwana ukrytą, gdyż do neuronów tej warstwy nie ma bezpośredniego dostępu ani od strony wejścia sieci, ani od strony jej wyjścia..

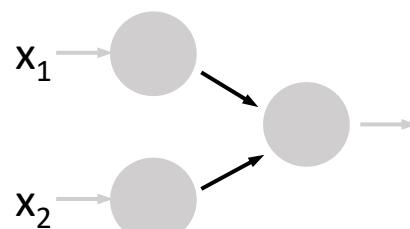
Celem tej warstwy jest przetwarzanie danych wejściowych w taki sposób, żeby uzyskane dane wyjściowe były przydatne dla potrzeb wypracowania rozwiązania całego zadania w warstwie wyjściowej sieci.

Liczba neuronów w warstwie ukrytej decyduje o jej „inteligencji”, jednak przyczynia się do większej trudności uczenia.

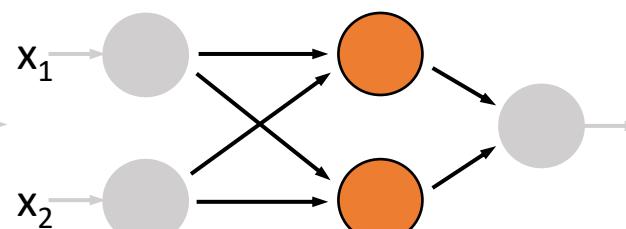
# Warstwy w sieciach neuronowych

## Liczebność warstw ukrytych

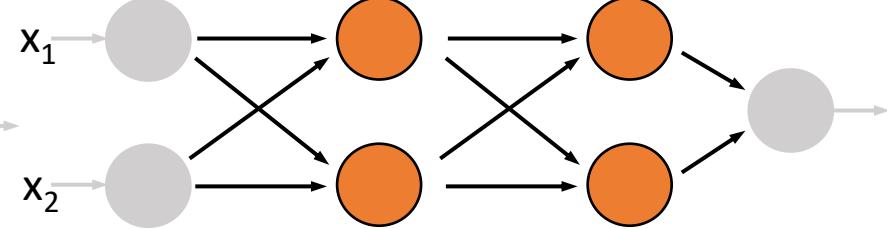
brak warstw ukrytych



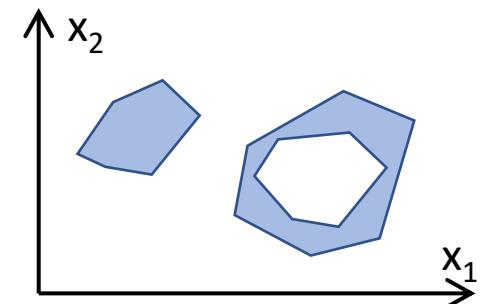
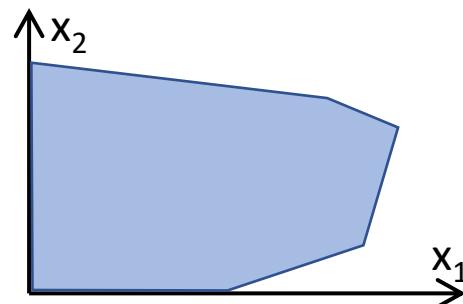
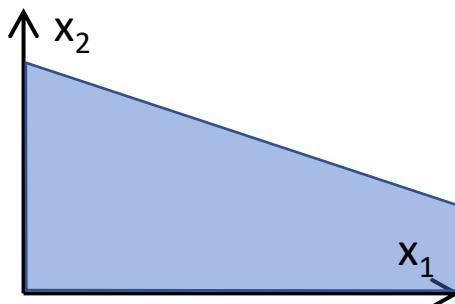
1 warstwa ukryta



2 warstwy ukryte



kształty obszarów decyzyjnych, jakie mogą tworzyć sieci o różnej liczbie warstw ukrytych



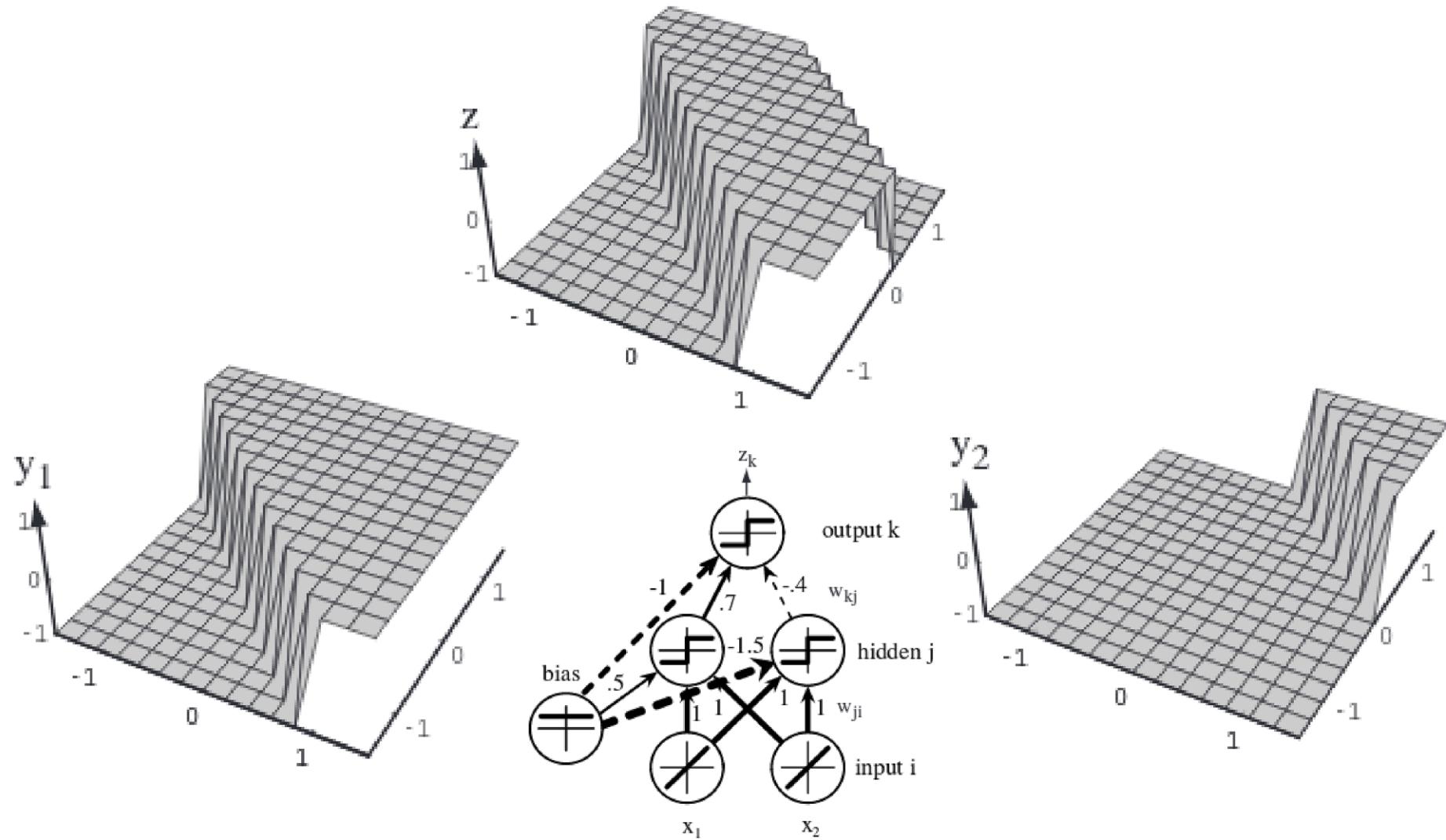
Podział na dwie części  
(bądź w przypadku  
wielowymiarowym  
hiperplaszczyzny)

dowolny jednospójny obszar  
o wypukłym obrysie

obszary decyzyjne otoczone  
niewypukłą powierzchnią  
graniczną, a także  
niejednospójne

Zastosowanie jeszcze większej liczby warstw ukrytych  
nie wzbogaca obszarów decyzyjnych, więc jest niecelowe.

*Pojedynczy neuron nie rozwiąże „problemu XOR”, ale kilka w sieci już to potrafi ..*

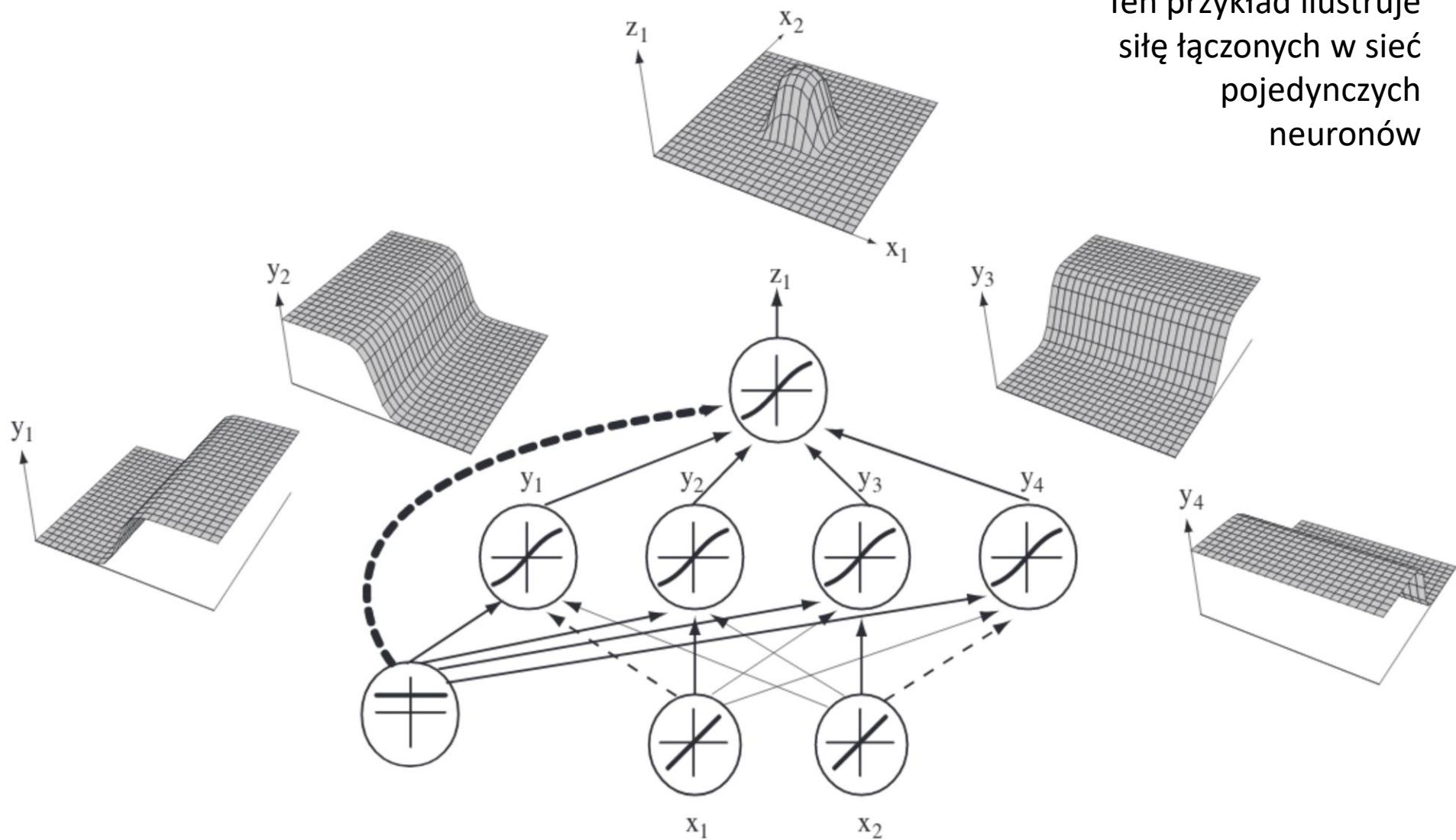


Rys: Duda, Hart, Pattern Classification

# Niewielka sieć złożona z perceptronów

(2 w warstwie wejściowej, 4 w warstwie ukrytej, 1 w warstwie wyjściowej)

Ten przykład ilustruje siłę łączonych w sieć pojedynczych neuronów



# Podstawowa sieć – MLP Multilayer Perceptron – jednokierunkowe sieci wielowarstwowe

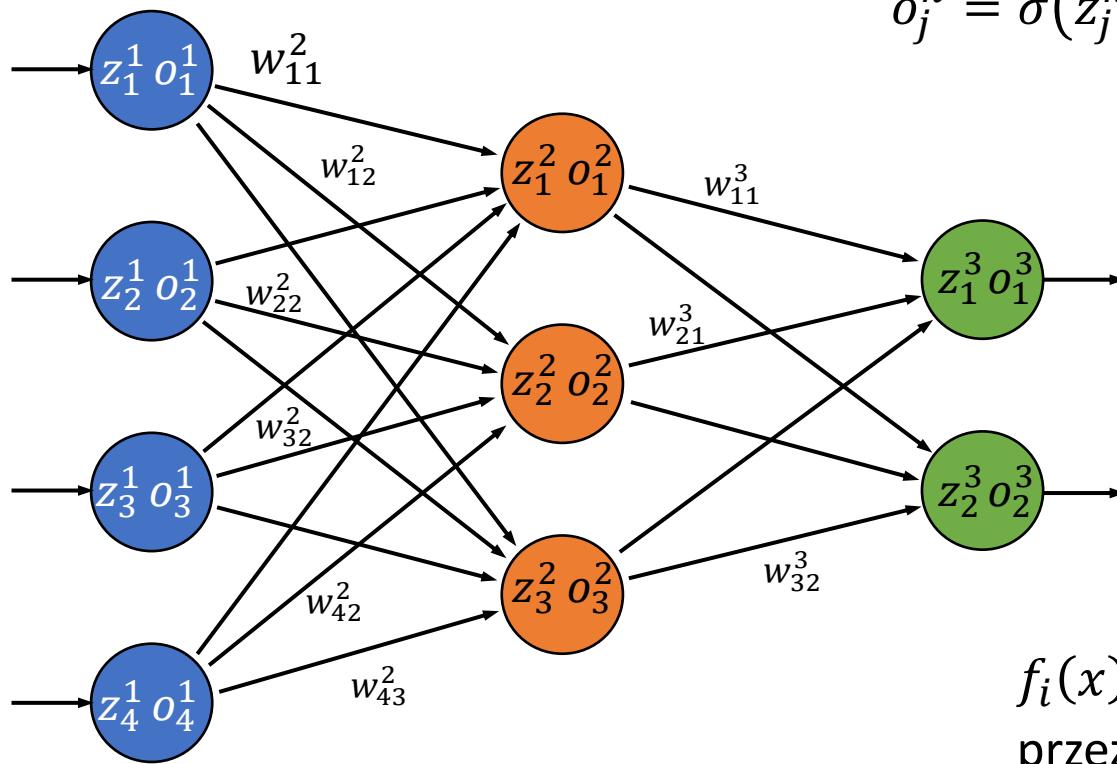
$M$  - liczba warstw ( $M > 2$ ) – przynajmniej jedna warstwa ukryta

$w_{ij}^k$  - waga łącząca element  $i$  należący do warstwy  $k-1$  oraz element  $j$  z warstwy  $k$

$z_j^k$  - aktywacja neuronu  $j$  w warstwie  $k$  :  $z_j^k = \sum w_{ij}^k \cdot o_i^{k-1}$

$o_j^k$  - sygnał wychodzący z elementu  $j$  w warstwie  $k$  :

$$o_j^k = \sigma(z_j^k) = \sigma(\sum w_{ij}^k \cdot o_i^{k-1})$$



$f_i(x) = o_j^M$  - funkcja realizowana przez wyjście  $i$  sieci MLP

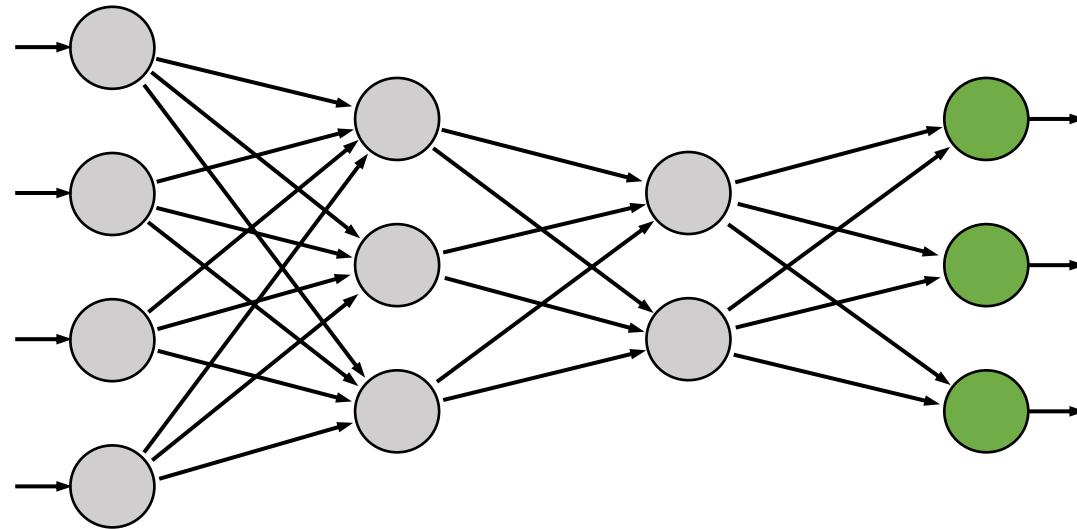
# ***Warstwy w sieciach neuronowych***

# Warstwa wejściowa

# 1 warstwa ukryta

## 2 warstwa ukryta

# warstwa wyjściowa



Zbiór neuronów,  
których sygnały  
wyjściowe są  
traktowane jako  
wyjście całej sieci

Odpowiedź sieci wymaga często dodatkowej interpretacji – na przykład w sieciach klasyfikacyjnych rozpoznanie ustala się na podstawie tego, który neuron warstwy wyjściowej prezentuje największą wartość sygnału.

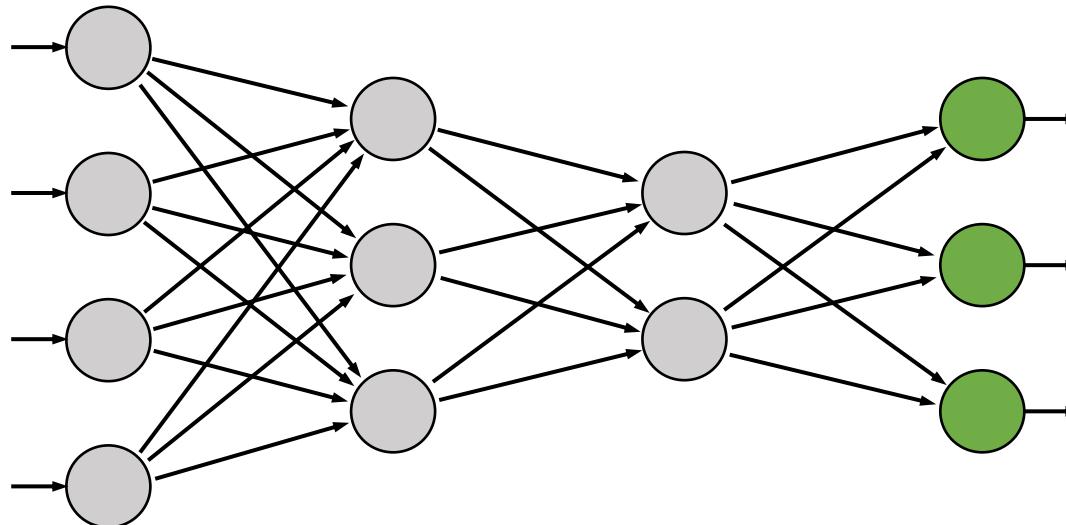
# Warstwy w sieciach neuronowych

Warstwa  
wejściowa

1 warstwa  
ukryta

2 warstwa  
ukryta

warstwa  
wyjściowa



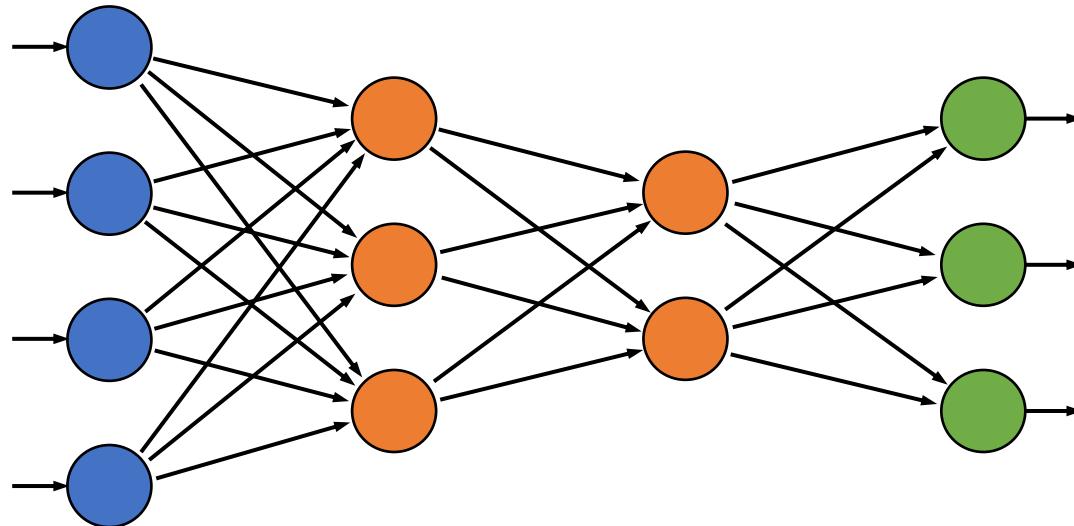
Dane wyjściowe w postaci liczbowej, które są efektem pracy sieci, najczęściej – poza ewentualnym przeskalowaniem, nie wymagają żadnych dodatkowej obróbki.

Sieć z wyjściami o charakterze jakościowym ma za zadanie zasugerować decyzję lub wybrać jedną z możliwości. Wtedy optymalne jest kodowanie jeden-z-N – sieć ma N neuronów wyjściowych (poszczególne możliwości reprezentowane są jednym neuronem w warstwie wyjściowej).

Neurony wyjściowe najczęściej pracują na sygnałach zgodnych z sigmoidalną funkcją aktywacji, (wartości od 0 do 1). W procesie decyzyjnym najczęściej określa się warunek: wyjście analizowanego neuronu wyższe od poziomu akceptacji, przy poziomach pozostałych wyjść niższych od poziomu odrzucenia.

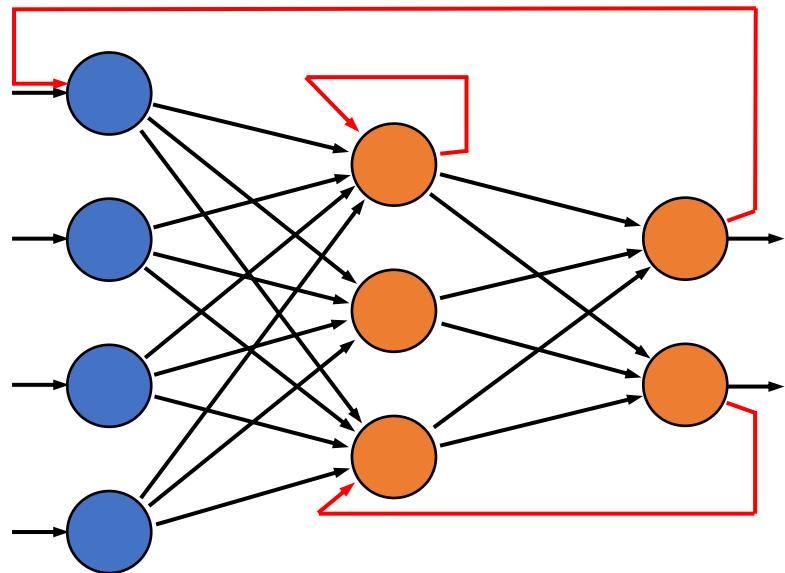
# *Typy połączeń pomiędzy neuronami*

Połączenie od warstw wcześniejszych do kolejnych  
(od wejścia, poprzez warstwy ukryte, do wyjścia) –  
**sieci jednokierunkowe - feedforward**



# *Typy połączeń pomiędzy neuronami*

Od warstw wcześniejszych do kolejnych oraz w drugą stronę – **sieć rekurencyjna**, sygnał otrzymany na wyjściu sieci trafia powtórnie na jej wejście (sprzężenie zwrotne)



Jednorazowe pobudzenie struktury ze sprzężeniem zwrotnym może generować całą sekwencję nowych zjawisk i sygnałów, ponieważ sygnały z wyjścia sieci trafiają ponownie na jej wejścia, generując nowe sygnały aż do ustabilizowania się sygnałów wyjściowych.

# **Typy połączeń pomiędzy neuronami - sieć rekurencyjna**

Neurony z połączonymi rekurencyjnymi  
zachowują się podobnie do pamięci

potencjalnie są w stanie modelować relacje  
występujące z dowolnie długim odstępem  
czasowym (nieskończony kontekst)

informacja z przyszłości może zawierać istotną dane,  
np. dźwięki kolejnych fonemów, znaczenie słów w  
zdaniu ..

wyjścia sieci w dowolnym momencie  
zależą od całej sekwencji wejściowej

zastosowanie: ↓↓↓

# **Typy połączeń pomiędzy neuronami - sieć rekurencyjna - zastosowanie**

- Dane sekwencyjne lub zależne od czasu: sygnał audio, tekst, ruch obiektów, EEG, ...
- Modelowanie sekwencji: generowanie tekstu (chatboty), generowanie muzyki, etykietowanie obrazów, sterowanie ruchem robota, ...
- Przetwarzanie sekwencji: maszynowe tłumaczenie, rozpoznawanie mowy, rozpoznawanie pisma, przetwarzanie strumieni wideo, ...
- Klasyfikacja sygnałów czasowych, wykrywanie wzorców w sekwencjach, predykcja (np. przewidywanie notowań giełdowych) ..

## MACHINE LEARNING AND DEEP LEARNING



Classification  
Learner



Deep Network  
Designer



Deep Network  
Quantizer



Experiment  
Manager



Neural Net  
Clustering



Neural Net  
Fitting



Neural Net  
Pattern Recog...



Neural Net  
Time  
Series



Regression  
Learner

## *Wróćmy do podstawowej sieci MLP – na co nam pozwala:*

Uniwersalny aproksymator: sieć MLP z jedną warstwą ukrytą jest w stanie aproksymować dowolną funkcję ciągła z dowolną dokładnością.

Dwie warstwy ukryte rozszerzają możliwości na funkcje nieciągłe.

Klasyfikator potrafi zrealizować dowolne granice decyzji  
(obszary nie muszą być wypukłe ani połączone)

Neurony ukryte:  
transformacja nieliniowa do przestrzeni odwzorowań, tworząca nowe cechy za pomocą nieliniowych kombinacji

Wiele zastosowań: klasyfikacja,  
wielowymiarowa regresja

## ***Na co będziemy musieli zwrócić uwagę***

Dobór architektury sieci: Ile neuronów w warstwie ? Jakie funkcje aktywacji?

Przeuczenie i generalizacja - zbyt duża liczba optymalizowanych parametrów powoduje przeuczenie, zbyt mała - może generować zbyt proste rozwiązania

Przygotowanie danych uczących: normalizacja, standaryzacja, kodowanie wyjść, ...

Regularyzacja - metody ograniczenia zjawiska przeuczenia, np. modyfikacje funkcji błędu, ograniczenie liczby parametrów

Inicjalizacja parametrów - szybkość treningu i wynik zależy od punktu startowego

Ocena modelu: zbiory walidacyjne, testowe, kroswalidacja

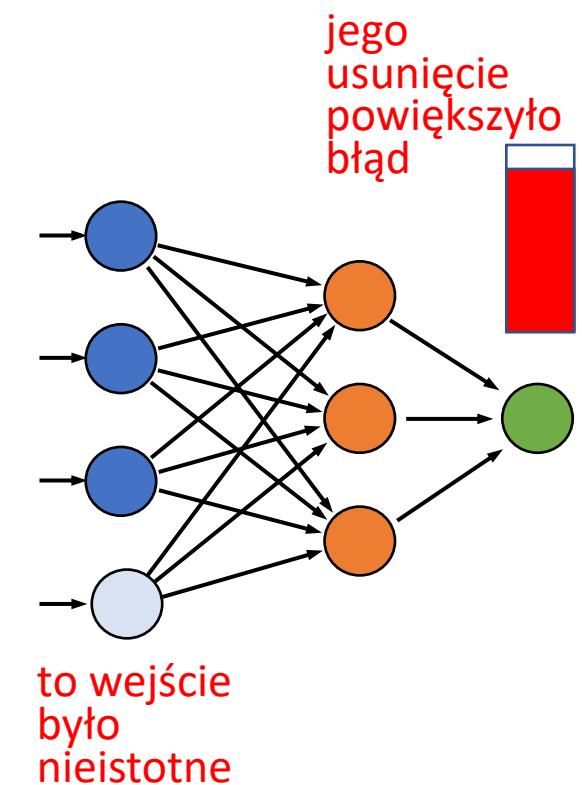
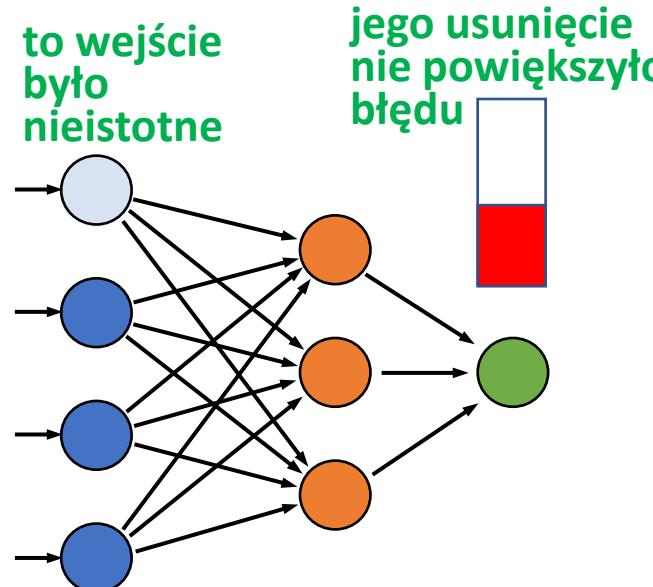
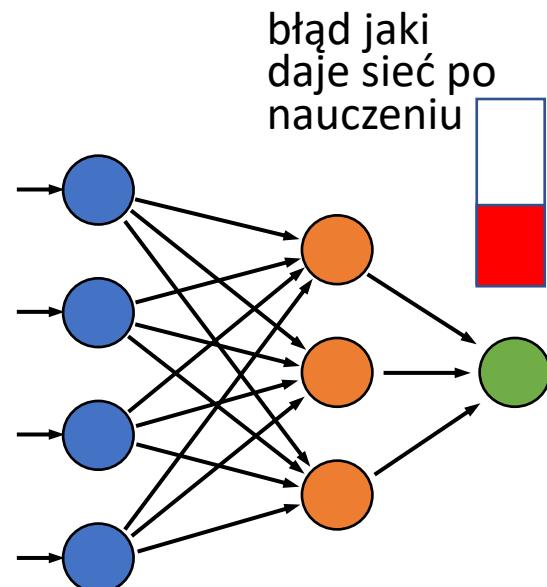
## Kroki podczas budowy sieci: przygotowanie danych wejściowych:

odpowiednie przygotowanie ilościowych danych wejściowych dla sieci neuronowej – normalizacja, minimalizacja danych silnie skorelowanych, usuwanie danych o niskiej wariancji ... przeskalowanie jest korzystne, gdyż funkcje aktywacji pracują najczęściej w zakresie 0-1 ..

Ile wejść? Czym więcej „wymiarów” sieci (ilości wejść) tym zapotrzebowanie na ilość danych uczących rośnie!

Często dane wejściowe mające charakter jakościowy kodujemy jak 1 z n ..

Analiza wrażliwości sieci pozwala nam eliminować mało istotne wejścia sieci ..



## Kroki podczas budowy sieci: Utworzenie sieci:

Sieć liniowa/nieliniowa (być może problem który rozwiązuje ma charakter liniowy?)?

Uczenie sieci liniowej jest szybkie i proste .. Ale sieć liniowa nie ma warstw ukrytych – rozwiązuje problemy dość proste ..

Sieć nieliniowa – wybór struktury sieci ..

Ile neuronów warstwy ukrytej ? pierwsze przybliżenie:  $k = \sqrt{n \cdot m}$

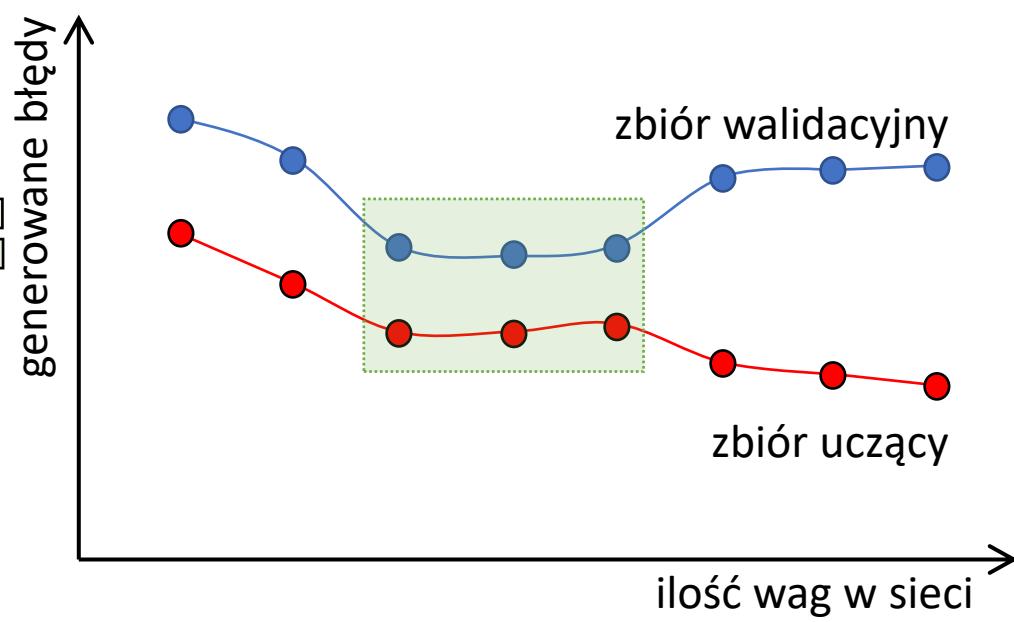
średnia geometryczna  
ilości wejść  $n$  i wyjść  
 $m$  sieci

Ilość wag w okresła ilość stopni swobody –

nie powinna przekraczać liczby przypadków uczących  $i$  – zalecane ( $w < i/10$ )

Czy więcej neuronów w warstwie ukrytej to zawsze lepiej ?

Najczęściej tak, ale nie wtedy, gdy mamy ograniczoną liczebność zbioru uczącego !



## **Koncepcja uczenia sieci neuronowych**

Celem uczenia jest dostosowywania sieci neuronowej, poprzez ukierunkowaną modyfikację wag połączeń pomiędzy neuronami, do rozwiązywania określonego typu zadań, poprzez przykłady rozwiązań zawarte w zbiorze uczącym.

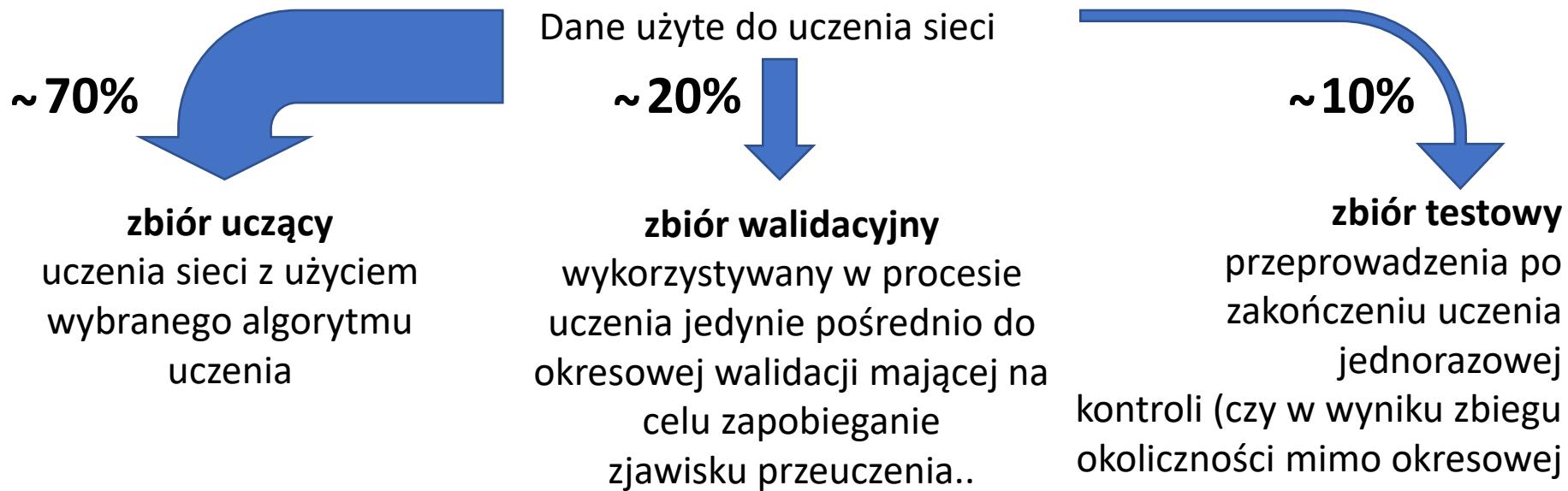
**Działanie algorytmu uczenia polega na pokazywaniu kolejnych przypadków uczących wraz z informacją podawaną przez nauczyciela, dotyczącą wymaganej poprawnej odpowiedzi sieci (tak zwana odpowiedź wzorcowa).**

**Ogólna idea procesu uczenia polega na minimalizacji funkcji błędu. Podczas działania algorytmu uczenia dochodzi do iteracyjnego modyfikowania wag w sieci neuronowej.**

Kryterium zatrzymania algorytmu związane jest z wykorzystaniem zbioru walidacyjnego sygnalizującego moment, kiedy sieć zaczyna tracić zdolność generalizacji wyników uczenia.

## **Podział zbioru wejściowego na kategorie**

Zbiór przypadków uczących, czyli zadań zawierających dane wejściowe oraz skojarzone z nimi odpowiedzi wzorcowe losowo dzielony jest na zbiory:



Główny problemem związanym z wydzieleniem podzbiorów polega na konieczności posiadania w każdym zbiorze przypadków uczących reprezentatywnych dla całego zbioru. Przypadki unikatowe bądź wyłącznie standardowe, nie mogą pojawić się wyłącznie w jednym ze zbiorów.

Wielkość błędu dla zbioru testowego powinna być zbliżona do wielkości błędu dla zbioru walidacyjnego. Jeśli tak nie jest, to uczenie sieci trzeba powtórzyć, ponownie dzieląc w sposób losowy zbiór uczący i wydzielając nowy zbiór walidacyjny oraz zbiór testowy.

## ***Uczenie***

Uczenie jest to proces oparty na prezentacji przypadków uczących oraz stopniowym dopasowywaniu sieci do tego, by nabyła ona umiejętność rozwiązywania przedstawianych zadań.

Dopasowywanie to opiera się na porównywaniu odpowiedzi udzielanych przez sieć z odpowiedziami wzorcowymi.

Wprowadzana korekta błędu powoduje, że sieć po każdej prezentacji zwiększa szansę udzielenia odpowiedzi bardziej zbliżonej do odpowiedzi wzorcowej.

***Jednocześnie zmierza się do tego, żeby sieć uogólniła wyuczoną umiejętność na inne, podobne zadania, nieprezentowane w trakcie uczenia – czyli generalizacja.***

**Formalnie uczenie to iteracyjny proces estymacji optymalnych wartości parametrów sieci (najczęściej wag) na podstawie zbioru uczącego.**

**Metodę zmiany wag w trakcie procesu uczenia wyznacza używany algorytm uczenia.**

## ***Uczenie***

Sieci wielowarstwowe uczymy zwykle metodami nadzorowanymi (tzn. z nauczycielem):

1. Podajemy na wejście sygnał wejściowy (zwykle w postaci wektora lub macierzy danych)
2. Obliczamy wartości wyjściowe neuronów w 1. warstwie i poprzez ich połączenia z neuronami w 2. warstwie podajemy te wartości jako sygnały wejściowe dla neuronów w 2. warstwie.
3. Obliczamy wartości wyjściowe w kolejnych warstwach tym samym sposobem.
4. Wartości wyznaczone w ostatniej warstwie stanowią zarazem odpowiedź sieci na podany sygnał wejściowy.
5. Sygnał ten porównujemy z wzorcowym (określonym przez nauczyciela) i wyznaczamy błąd.
6. Korzystając metod gradientowych propagujemy błąd wstecz przez sieć i dokonujemy korekty wartości wag połączeń synaptycznych

## Funkcja błędu

Błąd popełniany przez sieć neuronową zależy od współczynników wag występujących w sieci i poprawianych przez algorytmy uczenia.

Aby móc ocenić ten błąd najczęściej korzysta się z sumy kwadratów odchyłek ...

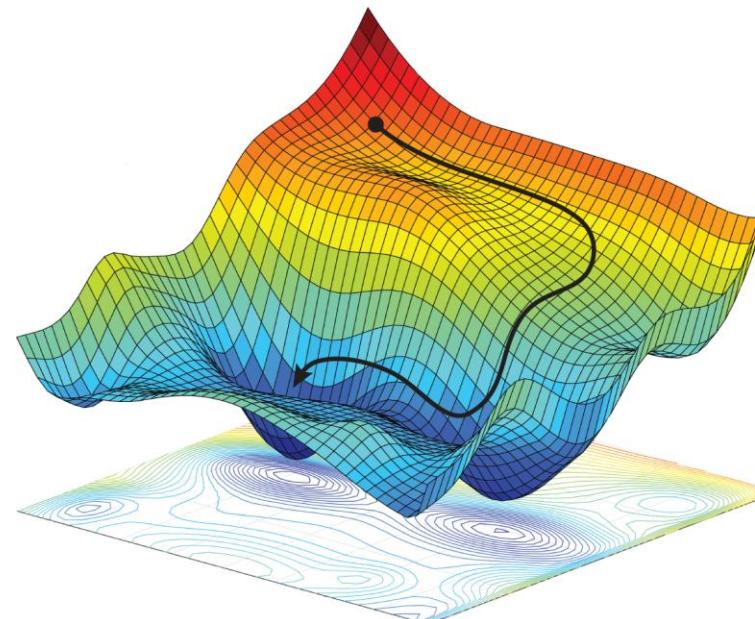
Błąd średniokwadratowy Mean Squared Error MSE

$$MSE = \frac{1}{N} \sum (z - y)^2$$

$z$  – oczekiwane wyjścia  
 $y$  – wyjścia sieci

Czym wobec tego formalnie jest uczenie sieci ? Jest problemem optymalizacyjnym, poszukiwania takiej kombinacji wag połączeń w sieci, aby błąd średniokwadratowy pomiędzy tym co oblicza sieć, a tym co oczekujemy aby obliczała, był jak najmniejszy ..

Algorytm wstecznej propagacji błędów polega na szukaniu kierunku spadku wielowymiarowej przestrzeni błędu ... i na takich modyfikacjach wartości wag  $w_{ij}$ , żeby zmniejszać wartość funkcji błędu w kierunku jej najszybszego spadku ..

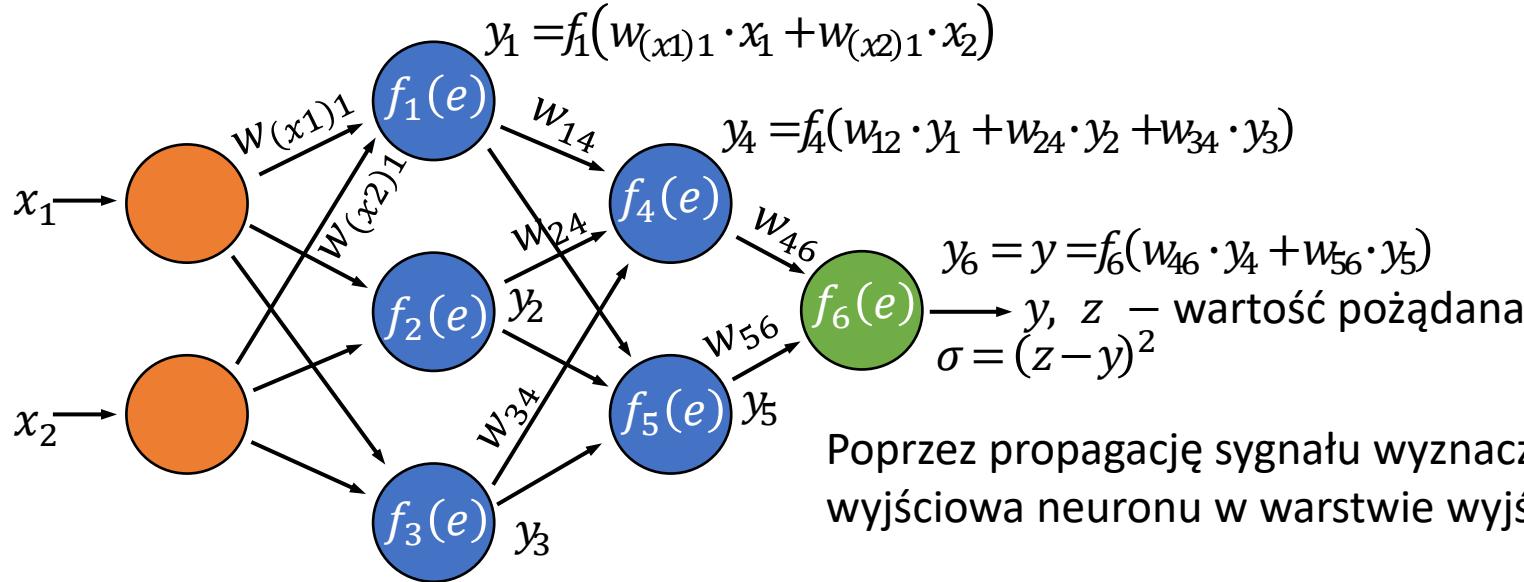


Często wykorzystywano metody ewolucyjne do optymalizacji wag sieci i minimalizacji funkcji błędu ..

# Metoda wstecznej propagacji błędów

umożliwiająca uczenie sieci wielowarstwowych poprzez propagację różnicy pomiędzy pożądanym a otrzymanym sygnałem na wyjściu sieci

Faza propagacji sygnału od wejścia (wektora  $x$ ) do wyjścia.



Poprzez propagację sygnału wyznaczana jest wartość wyjściowa neuronu w warstwie wyjściowej  $y_6$ .

Obliczona wartość wyjściowa jest porównywana z wartością pożądaną ( $z$ ) wyznaczoną w zbiorze uczącym przez nauczyciela.

Różnica pomiędzy wartością pożądaną i uzyskaną wynosi  $\sigma = (z - y)^2$ .

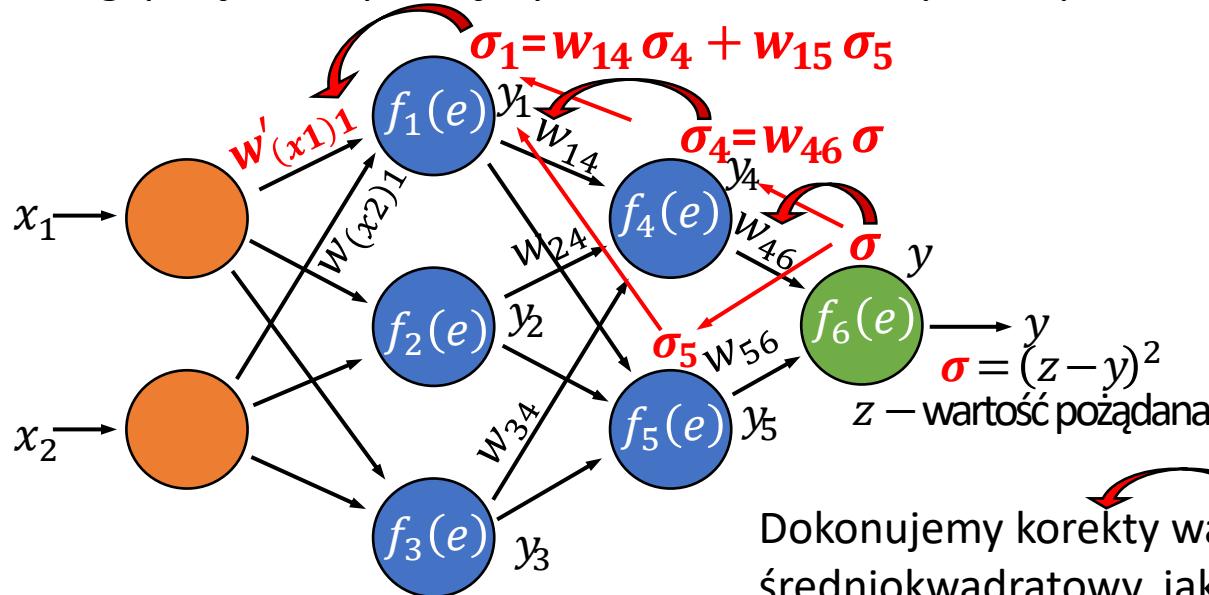
# Metoda wstecznej propagacji błędów

Wartość błędu jest propagowana wstecz, ważona zgodnie z wagą połączenia między neuronami i sumowana w tych neuronach celem wyznaczenia ich błędu.

Faza wstecznej propagacji błędu od wyjścia  $y$  w kierunku wejść sieci



Przechodząc z błędem wstecz z 3 warstwy do 2, błąd jest ważony zgodnie z aktualną wartością wagi połączenia pomiędzy neuronem warstwy 3 i odpowiednim neuronem warstwy 2.



Po powrocie do warstwy 1 mamy rozpropagowane błędy po całej sieci...

Następnie dokonywana jest korekta wartości wag sieci.

Dokonujemy korekty wag sieci tak, żeby zmniejszyły błąd średniokwadratowy, jaki był obliczony na wyjściu sieci.

W tym celu korzystamy z uogólnionej reguły delta, korzystając z pochodnej cząstkowej funkcji aktywacji  $\frac{df_i(e)}{de}$ .

Współczynnik  $a$  służy stopniowej adaptacji sieci oraz określa szybkość uczenia się (początkowo duży, na końcu uczenia mały – umożliwia przejścia od minimów lokalnych do minimum globalnego).

$$w'_{(x1)1} = w_{(x1)1} + a \cdot \sigma_1 \frac{df_1(e)}{de} x_1$$

$$w'_{14} = w_{14} + a \cdot \sigma_4 \frac{df_4(e)}{de} y_1$$

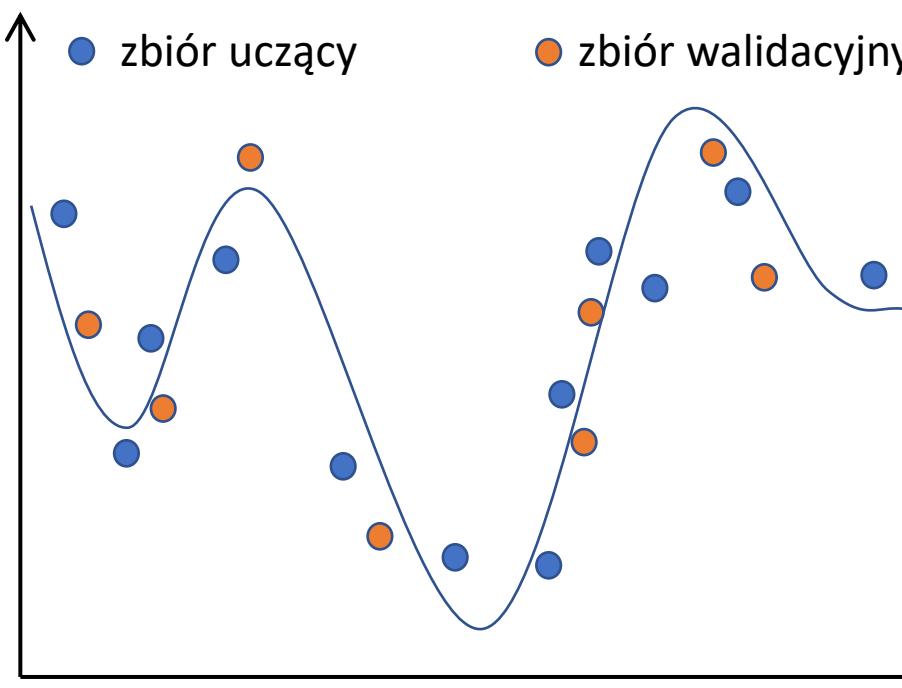
$$w'_{46} = w_{46} + a \cdot \sigma \frac{df_6(e)}{de} y_4$$

## Generalizacja

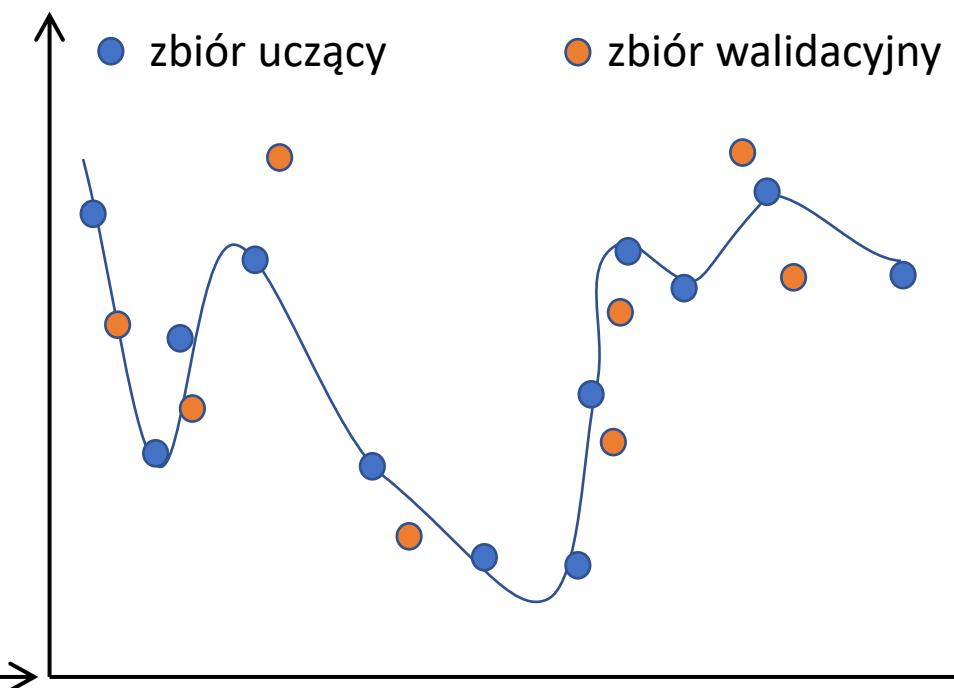
Sens użycia sieci neuronowej polega na tym, że musi ona (po nauczeniu) rozwiązywać zadania podobne do tych, na których była uczona – ale nie identyczne z nimi. Takie przeniesienie nabytej wiedzy na nowe przypadki nazywane jest generalizacją. Zagrożeniem dla generalizacji jest przeuczenie.

Gdy sieć jest przeuczona – następuje nadmierne dopasowanie jej zachowania do drugorzędnych (nieistotnych) szczegółów konkretnych przypadków uczących – nie mających istotnego znaczenia z punktu widzenia istotnych cech rozwiązywanego zadania.

Sieć poprawnie generalizująca problem



Sieć która utraciła zdolność generalizacji



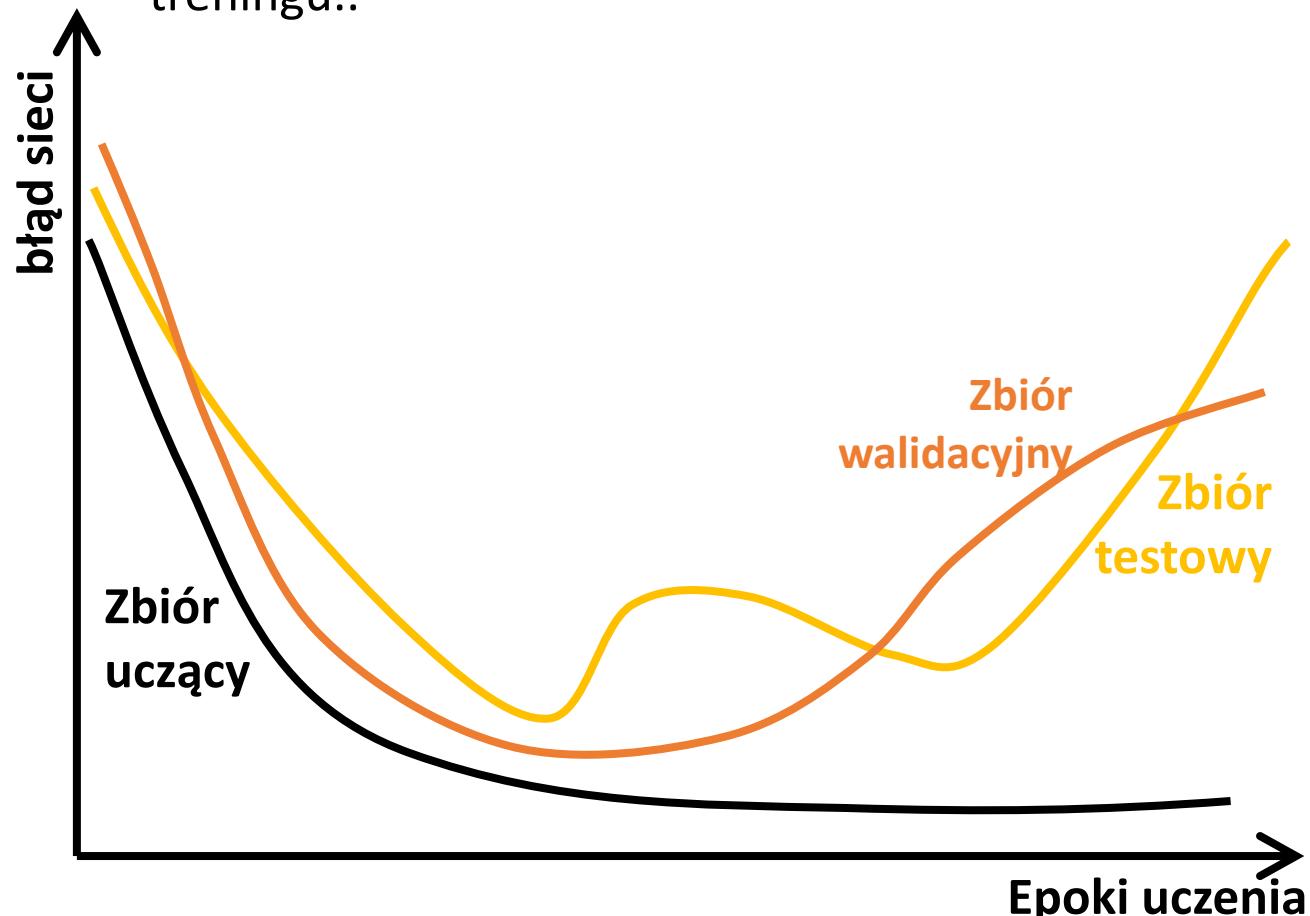
## *Generalizacja i przeuczenie (ilość epok nauczania)*

Na zbiorze treningowym błąd jest w stanie zawsze osiągnąć 0..

Dążymy do minimalizacji błędu na danych, które nie zostały użyte w treningu..

Generalizacja jest celem uczenia - uogólnienie reguły..

Zbyt długie uczenie sieci neuronowej powoduje, że sieć nadmiernie uzależnia swoje działanie od cech użytych do uczenia przypadków uczących – w tym także od cech drugorzędnych, nie dających podstaw do generalizacji.

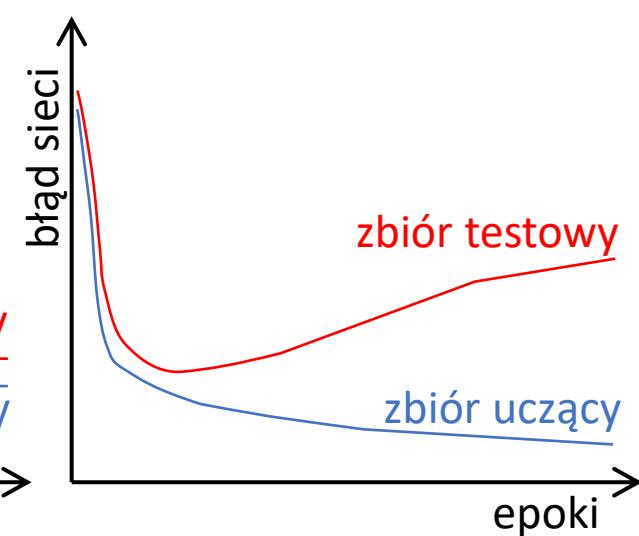
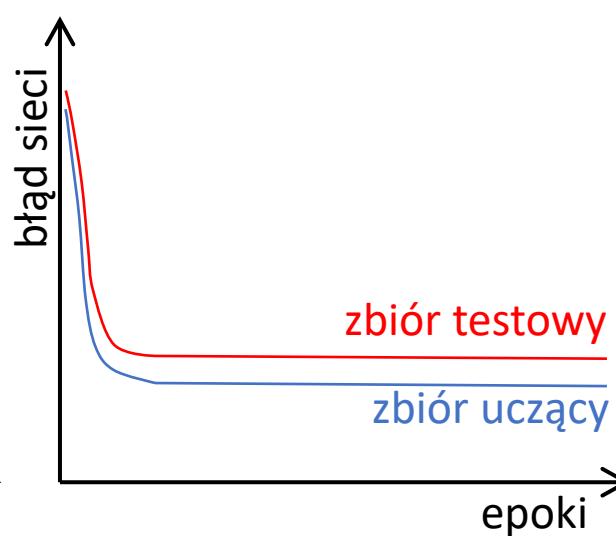
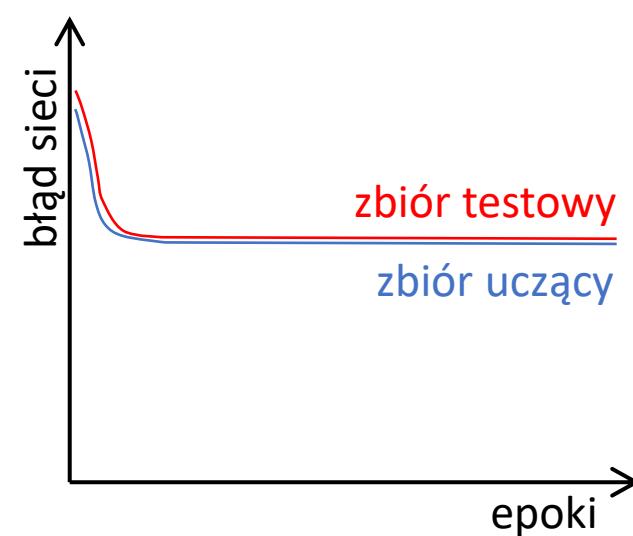


## Przeuczenie (ilość epok nauczania) – interpretacja wykresów uczenia

niedouczenie:  
błąd testowy oraz  
błąd treningowy  
pozostają wysokie

prawidłowy przebieg  
uczenia

przeuczenie:  
błąd testowy rośnie ↑,  
a błąd treningowy maleje ↓



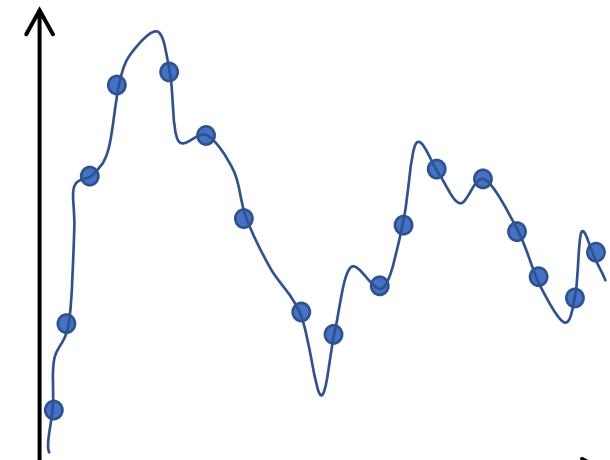
# *Generalizacja i przeuczenie (rozmiar warstwy ukrytej)*

*za dużo*

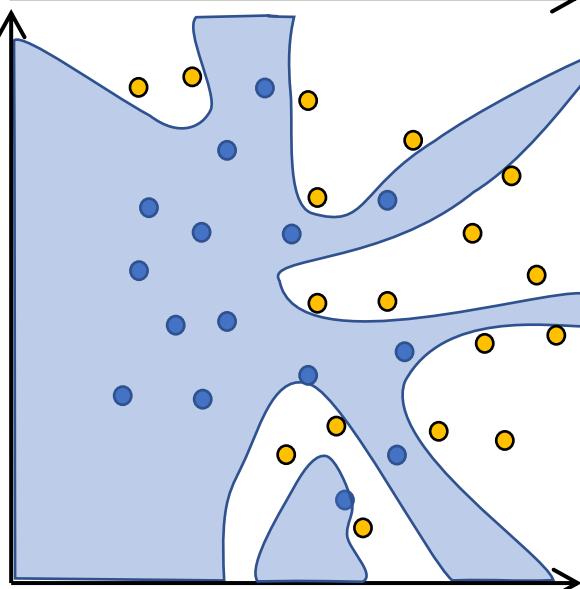
*optymalnie*

*za mało*

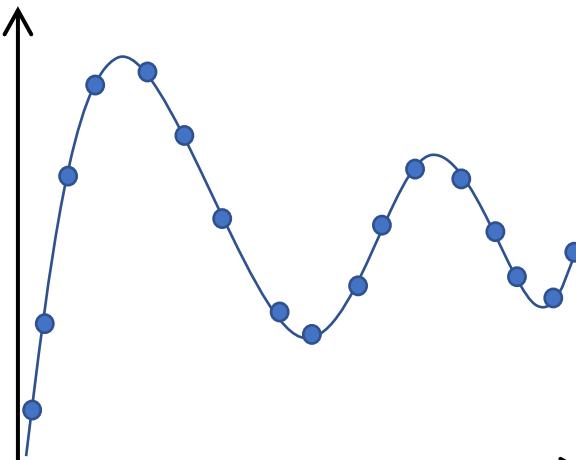
problem regresji



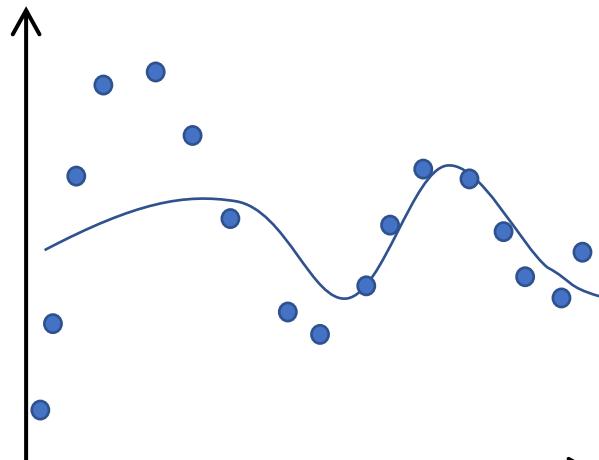
problem klasyfikacji



20 neuronów  
w warstwie ukrytej



6 neuronów  
w warstwie ukrytej

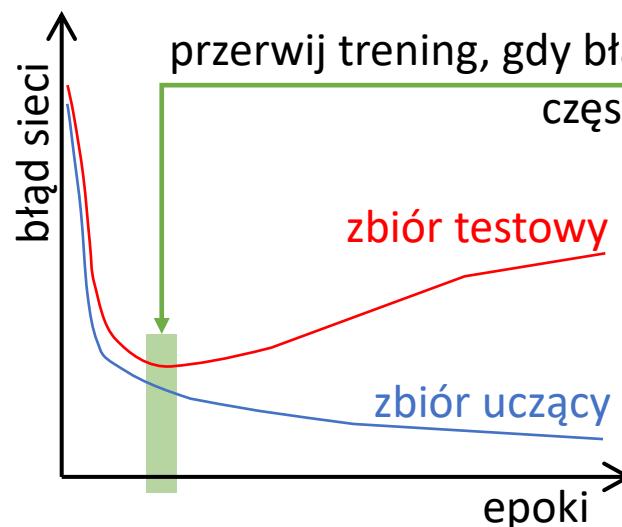


3 neurony  
w warstwie ukrytej

## **Co można zrobić aby poprawić generalizowalność problemu przez sieć ?**

- ✓ dobór wielkości modelu (ilości neuronów i wag), preferowane są najprostsze rozwiązania
- ✓ z pośród wielu modeli wybierz ten o najmniejszym błędzie walidacyjnym i najmniejszej liczbie parametrów (neuronów)
- ✓ usuwanie nadmiarowych połączeń lub neuronów z wytrenowanej sieci (wagi zerowe oznaczają brak połączenia)
- ✓ zwiększenie liczby danych
- ✓ selekcja cech - wybór tylko istotnych cech do treningu
- ✓ dodanie szumu do wag lub danych treningowych
- ✓ wcześnie zatrzymanie treningu, zanim błąd walidacyjny wzrośnie

zapamiętaj  
model o  
najmniejszym  
błędzie  
walidacyjnym



przerwij trening, gdy błąd walidacyjny zacznie rosnąć,  
często wystarczy tylko kilka iteracji

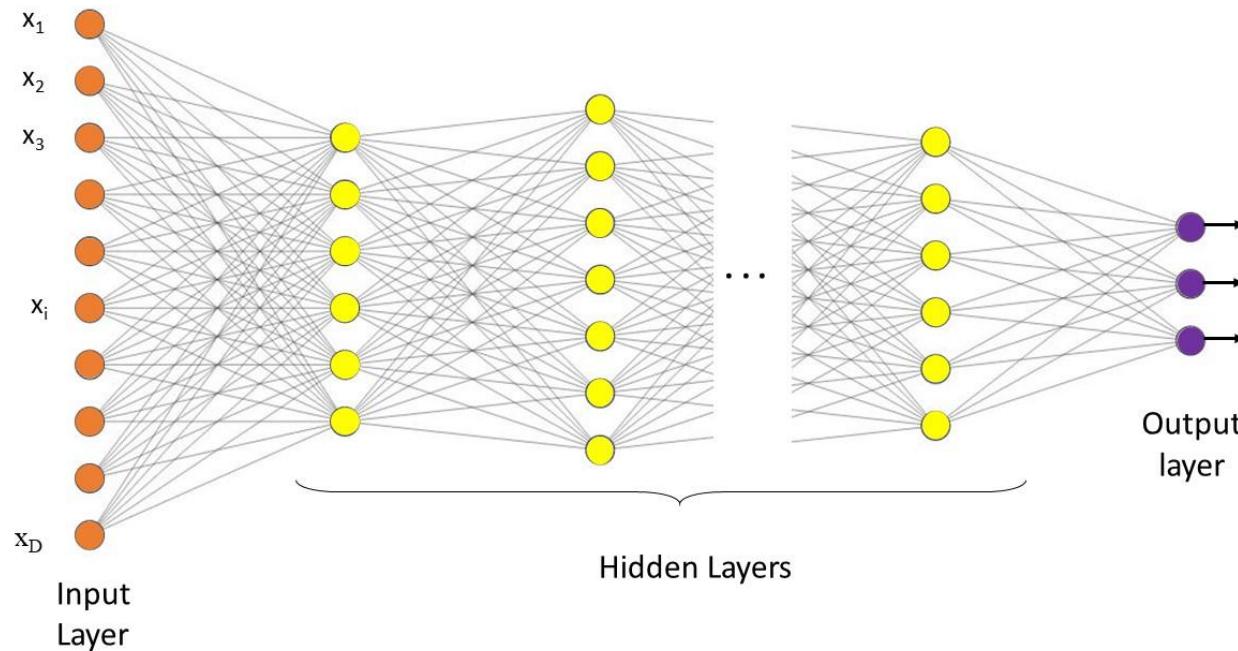
**zbiór testowy**

**zbiór uczący**

# **Gdzie kończy się tradycyjne podejście i zaczyna się uczenie głębokie ?**

Głębokie uczenie - metody uczenia maszynowego, które:

- zbudowane są z wielu warstw realizujących nieliniowe transformacje
- tworzą model hierarchiczny warstwy odwzorowują kolejne poziomy abstrakcji, najniższe warstwy tworzą reprezentację najprostszych cech z (surowego) sygnału wejściowego, wyższe warstwy - generują bardziej ogólne koncepty bazując na relacjach z poprzednich warstw.



# **Gdzie kończy się tradycyjne podejście i zaczyna się uczenie głębokie ?**

## **głębokie uczenie $\Leftrightarrow$ płytkie uczenie**

głębokie uczenie, gdy model ma więcej niż 2 warstwy (nieliniowych transformacji) od wejścia do wyjścia

do 2 warstw (nieliniowych transformacji), np. MLP z jedną warstwą ukrytą

możliwość pracy na surowych danych, modele end-to-end

wymagają odpowiednio przygotowanych cech

model hierarchiczny, wiele transformacji od wejścia do wyjścia

każda „warstwa” tworzy reprezentację danych na innym poziomie abstrakcji

**Gdzie kończy się tradycyjne podejście i zaczyna się uczenie głębokie ?**

**co można zyskać ?**

„Płytkie” modele wymagają o wiele więcej (wykładniczo) neuronów  
(rozrastając się „wszerz”)

„Płytkie” ale szerokie modele łatwiej się przetrenowują,  
nie są w stanie odszukać lepszego rozwiązania

Złożone problemy wymagają złożonych modeli

**jednak już**

siec MLP z jedną warstwą ukrytą jest w stanie aproksymować  
dowolną funkcję z dowolną dokładnością

Deep learning nie taki prosty, problemy bywają nie tylko z treningiem  
ale z interpretacją działania modeli

Dzisiejszy wykład :  
wstęp do neuro

obligatoryjne laborki –  
projekt fuzzy – 8 i 9 maja..

Kolejny wykład neuro 21 maja.. –  
analiza przykładów zastosowania

14 maja.. –

mam obronę – wykład odwołany ..

kolejne obligatoryjne laborki –  
projekt neuro – 22 i 23 maja..

18 czerwca – termin zero egzaminu

27 czerwca – pierwszy termin egzaminu

# Co i kiedy ??

Maj

2025 Czerwiec

2025

Pn	Wt	Śr	Cz	Pt	So	N
1	2	3	4			
5	6	7	8	9	10	11
12	13	<del>14</del>	juwenalia		17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Pn	Wt	Śr	Cz	Pt	So	N
1	2	3	4	5	6	7
9	10	11	12	13	14	15
16	17	<del>18</del>	19	20	21	22
23	24	25		27	sesja	

1 1 maja/  
genetyczny

2 5 maja/  
fuzzy

1 5 czerwca/  
neuro

TERMINY  
ODDANIA  
PROJEKTÓW

TERMINY  
ODDANIA  
PROJEKTÓW

TERMINY  
ODDANIA  
PROJEKTÓW

Wszelkie problemy  
z genetycznym – piszcie albo przyjdźcie na  
nieobligatoryjne, albo do mnie (p.317)

Nazwisko Imię	Monte	Genetyc	Fuzzy	Neuro	Ile oddanych	Ocena z projektów
A Lucja Weronika	3/21/2025	4/19/2025			2	3
B Jakub Marcel	3/24/2025				1	2
B Maja	3/21/2025				1	2
Ch Jakub Hubert	4/3/2025	3/30/2025			2	3
Ch Wiktor Jan	3/30/2025				1	2
D Konrad Adam	4/6/2024				1	2
D Kamil Stanisław	3/19/2025	4/19/2025			2	3
D Adam	4/4/2025				1	2
D Tomasz Piotr	3/31/2025	4/22/2025			2	3
D Gabriela	3/30/2025				1	2
D Jan Bartosz	3/21/2025	3/30/2025			2	3
E Maurycy	4/1/2025				1	2
F Julia	4/6/2025				1	2
F Aleksandra Maria	3/27/2025	4/27/2025			2	3
G Zuzanna Ewa	4/6/2025				1	2
G Patrycja	3/19/2025				1	2
G Mikołaj	4/6/2025				1	2
G Konrad Wojciech	4/3/2025	4/26/2025			2	3
G Jakub Przemysław	3/19/2025				1	2
H Karol	4/3/2025				1	2
H Aleksander Jakub	3/21/2025				1	2
I Bartosz	4/6/2025				1	2
J Filip Andrzej	4/6/2025	4/29/2025			2	3
J Aleksandra Monika	4/6/2025				1	2
J Julia Weronika	3/29/2025	4/30/2025			2	3
J Roksana Kamila	3/24/2025	4/28/2025			2	3
J Weronika	4/6/2025				1	2
Ki Maria	3/31/2025	4/18/2025			2	3
Kn Maria	3/17/2025	4/8/2025			2	3
K Karolina Agnieszka	3/27/2025				1	2
K Oliwier Piotr	3/15/2025	4/11/2025			2	3
K Julia Anita	3/27/2025				1	2
K Bartłomiej Mariusz	3/23/2025	4/30/2025			2	3
K Dawid Tomasz	3/20/2025				1	2
K Natalia Katarzyna	3/21/2025	4/23/2025			2	3
K Kacper Bartłomiej	4/5/2025				1	2
L Patryk	4/3/2025				1	2
M Patrycja Anna	3/21/2025				1	2
M Gerard	3/20/2025	5/4/2025			2	3
M Jakub Franciszek	3/16/2025	4/23/2025			2	3
M Eliza Klaudia	3/28/2025	4/23/2025			2	3
M Karolina	4/6/2025	4/24/2025			2	3
O Joanna Julia	3/26/2025				1	2
P Mirosław	3/25/2025				1	2
P Szymon Mateusz	4/3/2025				1	2
Pie Bartosz	4/6/2025				1	2
Piw Bartłomiej Jakub	3/24/2025				1	2
P Magdalena Maria	4/4/2025				1	2
P Dominika	3/31/2025				1	2
R Gabriel Mieczysław	4/4/2025	5/5/2025			2	3
S Dominik	3/30/2025	4/23/2025			2	3
S Marcin Jan	3/17/2025	4/15/2025			2	3
S Julia Krystyna	3/25/2025				1	2
S Julita	3/28/2025				1	2
S Szymon Marcin	3/16/2025	4/22/2025			2	3
S Weronika					0	2
S Martyna Joanna	3/26/2025				1	2
S Wiktorja Danuta	4/4/2024				1	2
S Aleksandra	3/27/2025	5/5/2025			2	3
S Anna Sara	4/3/2025				1	2
Szcz Magdalena Anna	4/6/2025				1	2
Szt Magdalena	3/19/2025	4/25/2025			2	3
Sz Mikołaj	4/3/2025				1	2
T Katarzyna	4/1/2025	4/28/2025			2	3
T Witold	3/17/2025				1	2
W Julia Barbara	3/26/2025				1	2
W Katarzyna	4/6/2025				1	2
W Oliwia Klaudia	3/25/2025				1	2
Z Michał	3/24/2025				1	2
Z Karolina	4/2/2025				1	2

Montecarlo – 100%

Genetyczny – 38%

ile % / średnia 100,0 37,7 0,0 0,0 1,357 2,37

# Inteligencja obliczeniowa w analizie danych



dziękuję za uwagę !