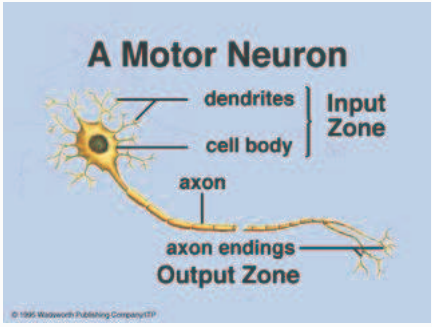
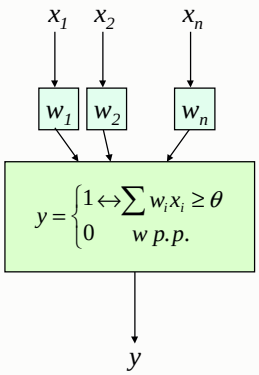
Sieci neuronowe - uczenie

www.qed.pl/ai/nai2003

Perceptron - przypomnienie





Uczenie perceptronu

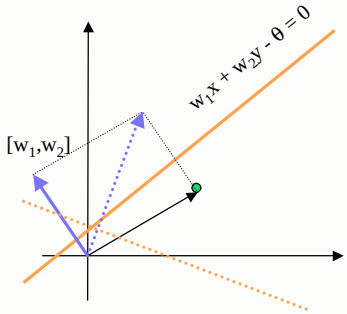
- **Wejście:**
  - Ciąg przykładów uczących ze znanymi odpowiedziami
- **Proces uczenia:**
  - Inicjujemy wagi losowo
  - Dla każdego przykładu, jeśli odpowiedź jest nieprawidłowa, to

    $w_1 += \alpha x_1$

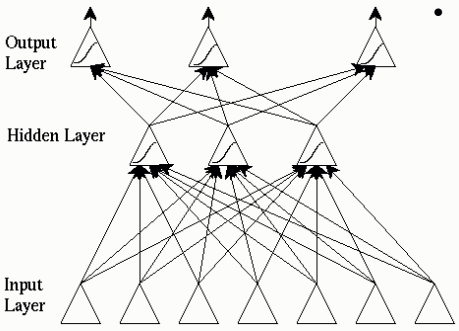
    $w_2 += \alpha x_2$

    $\theta -= \alpha$

    gdzie α jest równe różnicy między odpowiedzią prawidłową a otrzymaną.



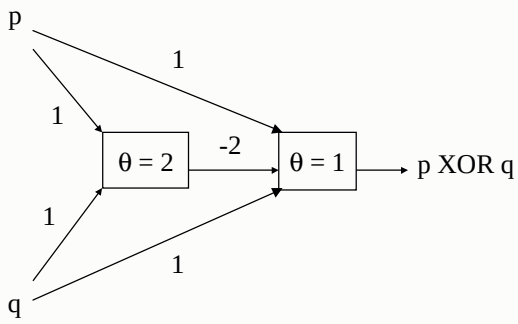
SIECI WIELOWARSTWOWE



- Wyjścia neuronów należących do warstwy niższej połączone są z wejściami neuronów należących do warstwy wyższej
  - np. metodą „każdy z każdym”
- Działanie sieci polega na liczeniu odpowiedzi neuronów w kolejnych warstwach
- Nie jest znana ogólna metoda projektowania optymalnej architektury sieci neuronowej

SIECI PERCEPTRONÓW

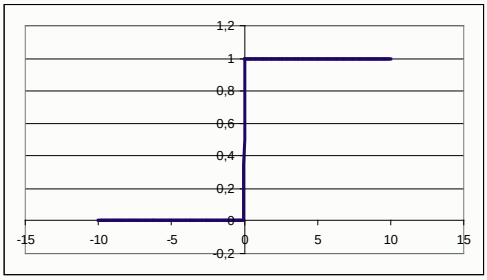
Potrafią reprezentować dowolną funkcję boolowską (opartą na rachunku zdań)

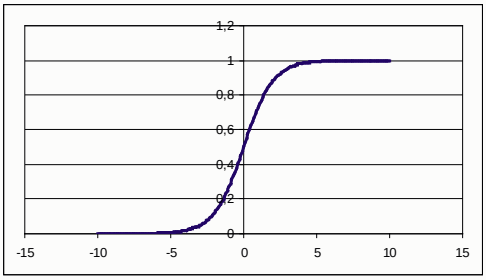


Funkcje aktywacji

- Progowe

$$f(z)=\begin{cases}1 \Leftrightarrow z \geq 0\\ 0 \Leftrightarrow z < 0\end{cases}$$





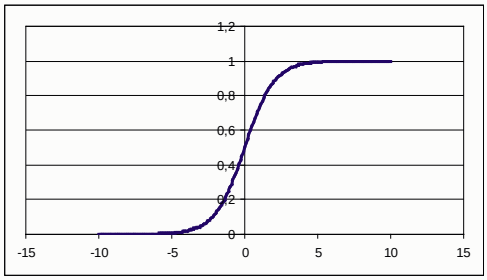
- Sigmoidalne

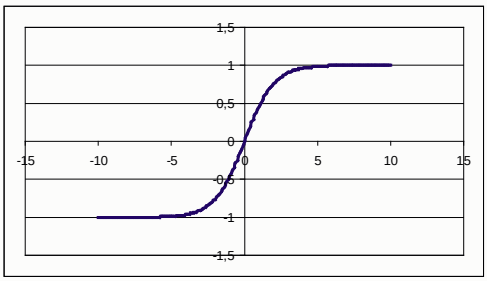
$$f(z)=\frac{1}{1+e^{-z}}$$

FUNKCJE AKTYWACJI (2)

- Unipolarne

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

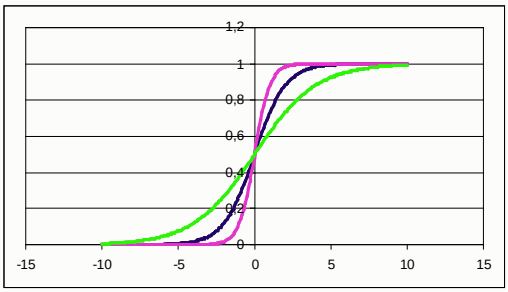




- Bipolarne

$$f(z)=\frac{2}{1+e^{-z}}-1$$

FUNKCJE AKTYWACJI (3)



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

α = 2.0

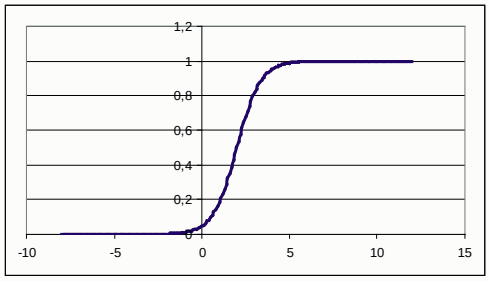
α = 1.0

α = 0.5

$$\lim_{\alpha\to 0} f_\alpha(z)=0.5 \qquad \lim_{\alpha\to+\infty} f_\alpha(z)=\begin{cases}1 & \Leftrightarrow \quad z>0\\ 0.5 & \Leftrightarrow \quad z=0\\ 0 & \Leftrightarrow \quad z<0\end{cases}$$

FUNKCJE AKTYWACJI (4)

$$f_{\theta,\alpha}(z) = \frac{1}{1+e^{-\alpha(z-\theta)}}$$



$\theta = 2$

$\alpha = 1.5$

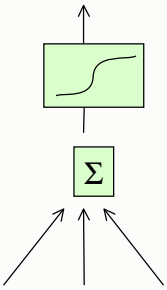
FUNKCJE AKTYWACJI (5)

- Zasady ogólne:
  - Ciągłość (zachowanie stabilności sieci jako modelu rzeczywistego)
  - Różniczkowalność (zastosowanie propagacji wstecznej)
  - Monotoniczność (intuicje związane z aktywacją komórek neuronowych)
  - Nieliniowość (możliwości ekspresji)

SIECI NEURONOWE

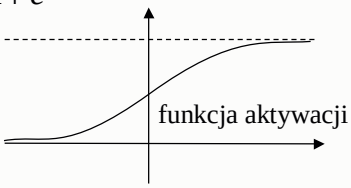
Potrafią modelować (dowolnie dokładnie przybliżać) funkcje rzeczywiste

(z tw. Kołmogorowa)

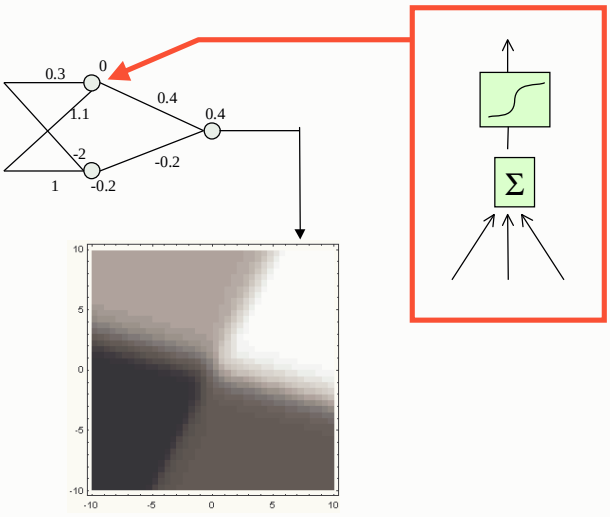


$$y = f\left(w_0 + \sum_{i=1}^{n} w_i x_i \right)$$

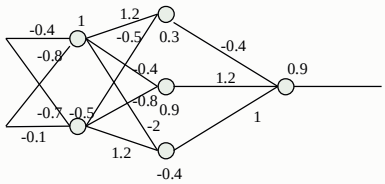
$$f(z) = \frac{1}{1+e^{-z}}$$

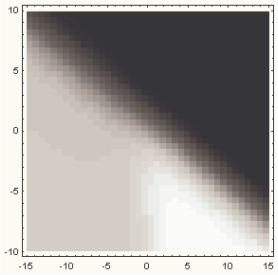


SIECI NEURONOWE



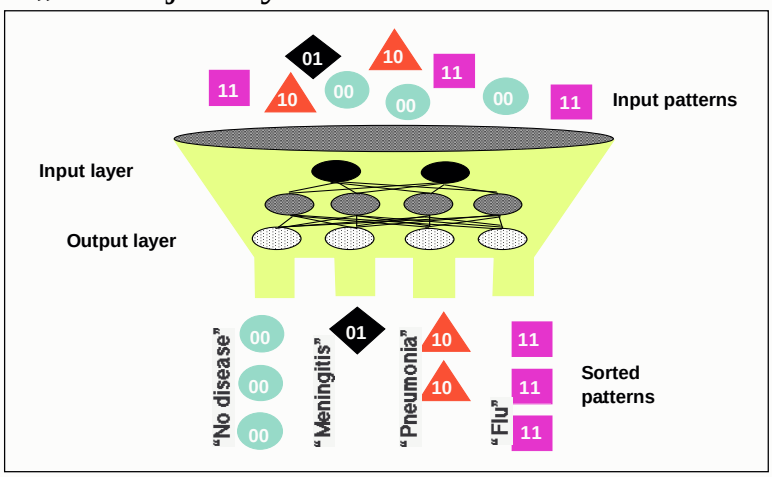
SIECI NEURONOWE



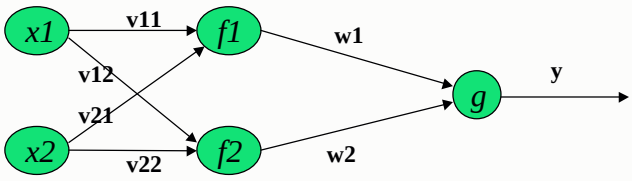


SIECI NEURONOWE

Potrafią klasyfikować obiekty na zasadzie „czarnej skrzynki”



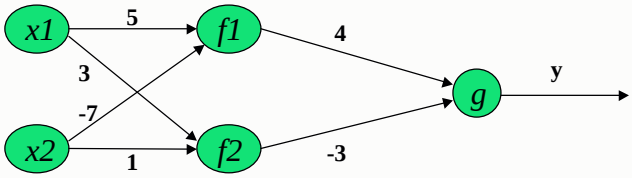
SIECI JAKO FUNKCJE ZŁOŻONE (1)



$$y = g\left(w_1 f_1\left(v_{11}x_1 + v_{21}x_2\right) + w_2 f_2\left(v_{12}x_1 + v_{22}x_2\right)\right)$$

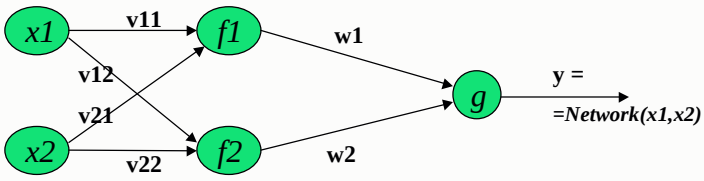
$$y = Network\left(x_1, x_2\right)$$

SIECI JAKO FUNKCJE ZŁOŻONE (2)



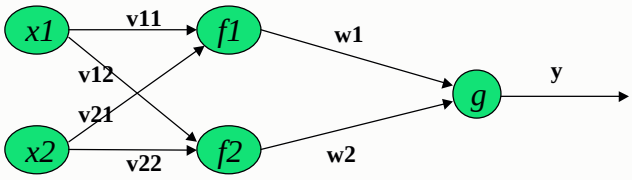
$$y = \begin{cases} 1 \Leftrightarrow \dfrac{4}{1+e^{-3(5x_1-7x_2)}} - 3\left(\dfrac{2}{1+e^{-2(3x_1+x_2)}} - 1\right) \geq 8 \\ 0 \Leftrightarrow \dfrac{4}{1+e^{-3(5x_1-7x_2)}} - 3\left(\dfrac{2}{1+e^{-2(3x_1+x_2)}} - 1\right) < 8 \end{cases}$$

SIECI JAKO FUNKCJE ZŁOŻONE (3)



- Jeśli wszystkie poszczególne funkcje aktywacji są liniowe, to funkcja *Network* jest również liniowa
- Architektura wielowarstwowa daje zatem nowe możliwości tylko w przypadku stosowania funkcji nieliniowych

SIECI JAKO FUNKCJE ZŁOŻONE – przypadek liniowy



- Niech

  $f_i(x1,x2) = a_i*(x1*v1_i + x2*v2_i) + b_i$

  $g(z1,z2) = a*(z1*w1 + z2*w2) + b$
- Wtedy

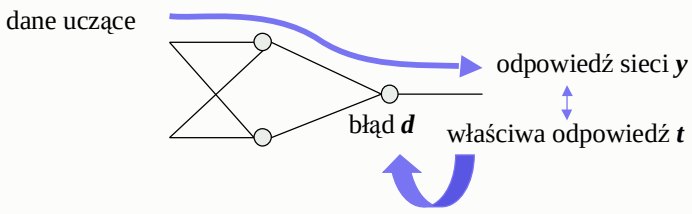
  Network(x1,x2) = A1*x1 + A2*x2 + B
- Np.:

  A1 = a*(a1*v1*w1 + a2*v2*w2)

PROPAGACJA WSTECZNA (1)

- Chcemy “wytrenować” wagi połączeń między kolejnymi warstwami neuronów
- Inicjujemy wagi losowo (na małe wartości)
- Dla danego wektora uczącego obliczamy odpowiedź sieci (warstwa po warstwie)
- Każdy neuron wyjściowy oblicza swój błąd, odnoszący się do różnicy pomiędzy obliczoną odpowiedzią **y** oraz poprawną odpowiedzią **t**

PROPAGACJA WSTECZNA (2)



Błąd sieci definiowany jest zazwyczaj jako

$$d = \frac{1}{2}(y - t)^2$$

PROPAGACJA WSTECZNA (3)

- Oznaczmy przez:
  - $g: R \rightarrow R$ – funkcję aktywacji w neuronie
  - $w_1, ..., w_K$ – wagi połączeń wchodzących
  - $z_1, ..., z_K$ – sygnały napływające do neuronu z poprzedniej warstwy
- Błąd neuronu traktujemy jako funkcję wag połączeń do niego prowadzących:

$$d(w_1,...,w_K) = \frac{1}{2}\left(g(w_1 z_1 + ... + w_K z_K) - t\right)^2$$

PRZYKŁAD (1)

- Rozpatrzmy model, w którym:
  - Funkcja aktywacji przyjmuje postać

$$g(s) = \frac{1}{1+e^{-3(s+2)}}$$

  - Wektor wag połączeń = [1;-3;2]
- Załóżmy, że dla danego przykładu:
  - Odpowiedź powinna wynosić $t = 0.5$
  - Z poprzedniej warstwy dochodzą sygnały [0;1;0.3]

PRZYKŁAD (2)

- Liczymy wejściową sumę ważoną:

$$s = w_1 x_1 + w_2 x_2 + w_3 x_3 = 1 \cdot 0 + (-3) \cdot 1 + 2 \cdot 0.3 = -2.4$$

- Liczymy odpowiedź neuronu:

$$y = g(s) = \frac{1}{1 + e^{-3(-2.4+2)}} = \frac{1}{1 + e^{1.2}} \approx 0.23$$

- Błąd wynosi:

$$d = \frac{1}{2}(0.23 - 0.5)^2 \approx 0.036$$

IDEA ROZKŁADU BŁĘDU

- Musimy „rozłożyć” otrzymany błąd na połączenia wprowadzające sygnały do danego neuronu
- Składową błędu dla każdego ***j***-tego połączenia określamy jako pochodną cząstkową błędu względem ***j***-tej wagi
- Składowych tych będziemy mogli użyć do zmodyfikowania ustawień poszczególnych wag połączeń

OBLICZANIE POCHODNEJ

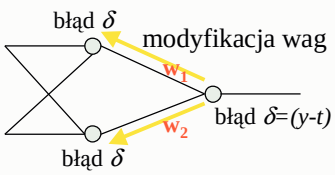
$$\frac{\partial d(w_1,\ldots,w_K)}{\partial w_j} = (y-t)\cdot g'(s)\cdot z_j$$

$$= \frac{\partial \frac{1}{2}\left(g(w_1 z_1 + \ldots + w_K z_K) - t\right)^2}{\partial w_j}$$

$$= \frac{\partial \frac{1}{2}(y-t)^2}{\partial y} \cdot \frac{\partial g(s)}{\partial s} \cdot \frac{\partial (w_1 z_1 + \ldots + w_K z_K)}{\partial w_j}$$

PROPAGACJA WSTECZNA (4)

- Idea:
  - Wektor wag połączeń powinniśmy przesunąć w kierunku przeciwnym do wektora gradientu błędu (z pewnym współczynnikiem uczenia η)
- Realizacja: $\Delta w_j = \eta \cdot (y-t)\cdot g'(s)\cdot z_j$



Błędy są następnie propagowane w kierunku poprzednich warstw, tzn. zamiast $(y-t)$ wstawiamy:

$$\delta = \sum_{i=1}^{n} w_i \delta_i$$

czyli neuron w warstwie ukrytej "zbiera" błąd z neuronów, z którymi jest połączony (uwzględniając wagi sprzed modyfikacji).