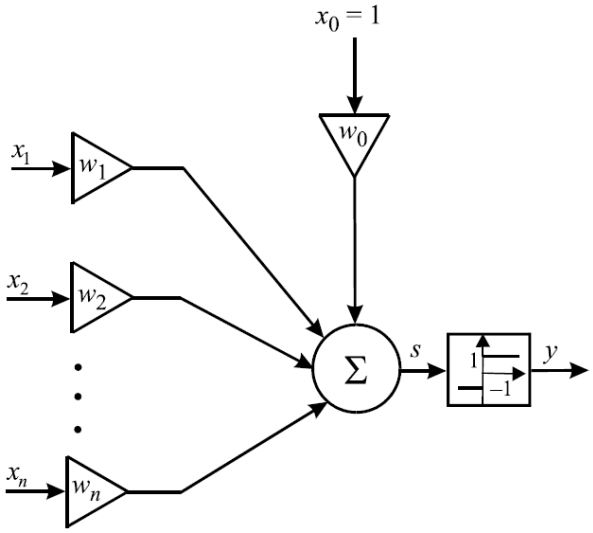
SYSTEMY AI – LAB 4.1 (2023)

Zadanie 1

Struktura **perceptronu**:



- *n* wejść: x_0=1 (bias), x_1, x_2,…, x_n
- 1 wyjście: y

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

gdzie θ, w_1,…, w_n (w_0 jest oznaczane przez θ) są wagami i

$$f(s) = \begin{cases} 1, & s > 0 \\ -1, & s \le 0 \end{cases}$$

jest *funkcją aktywacji*.

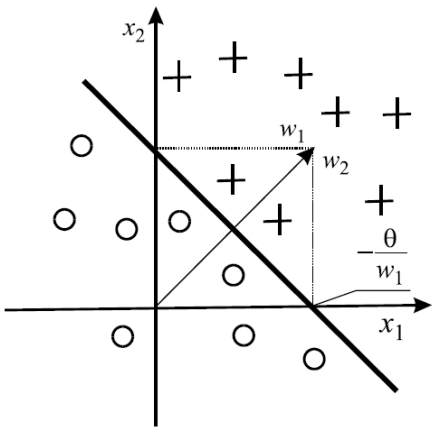
- Perceptron może klasyfikować “obiekty” które należą do dwóch liniowo separowalnych zbiorów.

  Dla *n*=2 zbiory te są separowalne przez linię prostą:

$$x_1 w_1 + x_2 w_2 + \theta = 0$$

  Wówczas:

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{\theta}{w_2}$$



W celu nauczenia percepronu klasyfikacji musimy użyć pewnego ciągu uczącego tzn. zbioru $D=\{(\boldsymbol{x}_1,d_1),\ldots,(\boldsymbol{x}_m,d_m)\}$ *m* przykładów gdzie:

- $\boldsymbol{x}_j$ – jest n-wymiarowym wektorem wejść.
- d_j – jest wartością oczekiwaną na wyjściu dla wektora $\boldsymbol{x}_j$.

Algorytm uczenia:

1. Nadaj wartości początkowe wagom θ, $w_1,\ldots, w_n$. Wagi mogą być początkowo równe 0 lub wybrane losowo.
2. Dla każdego elementu $(\boldsymbol{x}_j,d_j)$ z ciągu uczącego :
    a. Oblicz y.
    b. Jeżeli klasyfikacja jest poprawna ($y=d_j$), nic nie rób.
    c. Jeżeli klasyfikacja nie jest poprawna ($y\neq d_j$), wówczas zmodyfikuj wszystkie wagi: $w_i = w_i + d_j x_{ji}$
3. Powtarzaj tę procedurę aż wszystkie $\boldsymbol{x}_i$ będą zaklasyfikowane poprawnie.

POLECENIE:

Napisz program będący implementacją perceptronu dla poniższego ciągu uczącego:

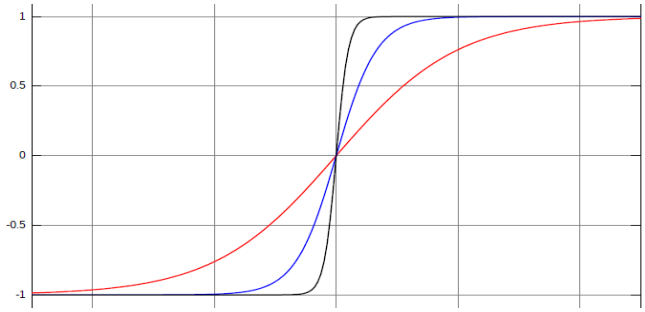
x_1	x_2	d
2	1	1
2	2	1
0	6	1
-2	10	-1
-2	0	-1
0	0	-1
4	-20	-1

Po nauczeniu znajdź równanie prostej decyzyjnej.

Zadanie 2

Funkcja aktywacji **neuronu Hebba** – funkcja bipolarna:

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



Uczenie polega na znalezieniu wag minimalizujących błąd:

$$E = \frac{1}{2}(d - f(\sum_{i=o}^{n} x_i w_i))^2$$

Modyfikacja wag:

$$w_i = w_i + \eta dx_i$$

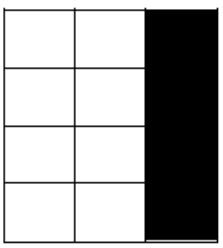
gdzie $\eta \in [0,1]$ jest *współczynnikiem uczenia.*

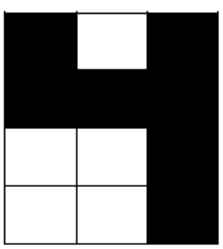
Algorytm uczenia:

1. Nadaj wartości początkowe wagom θ, $w_1,\ldots, w_n$. Wagi mogą być początkowo równe 0 lub wybrane losowo.
2. Dla każdego elementu $(\boldsymbol{x}_j, d_j)$ z ciągu uczącego :
    a. Oblicz y.
    b. Zmodyfikuj wszystkie wagi.
    c. Policz błąd E.
3. Jeżeli błąd sumaryczny dla całej epoki nie jest mniejszy od założonego E_{MAX} wówczas wróć do pkt 2.

POLECENIE:

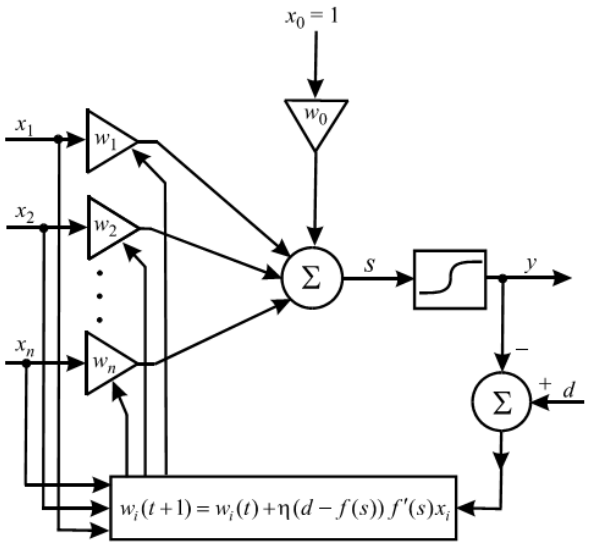
Napisz program dla neuronu Hebba rozpoznającego dwie cyfry:





Zadanie 3

Neuron sigmoidalny z unipolarną funkcją aktywacji:



Unipolarna funkcja aktywacji:

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

Wyjście neuronu:

$$y(t) = f\left(\sum_{i=0}^{n} w_i(t)\, x_i(t)\right).$$

Uczenie polega na znalezieniu wag minimalizujących błąd:

$$Q(\mathbf{w}) = \frac{1}{2}\left[d - f\left(\sum_{i=0}^{n} w_i x_i\right)\right]^2$$

Modyfikacja wag:

$$w_i(t+1) = w_i(t) - \eta\delta x_i = w_i(t) + \eta(d - f(s))f'(s)x_i.$$

gdzie $\eta \in [0,1]$ jest *współczynnikiem uczenia.*

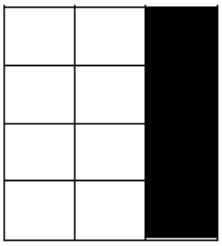
Pochodna:

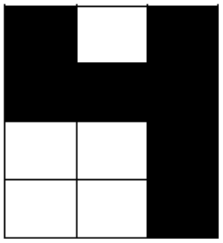
$$f'(x) = \beta f(x)(1 - f(x))$$

Algorytm uczenia taki jak dla neuronu Hebba.

POLECENIE:

Napisz program dla neuronu **sigmoidalny z unipolarną funkcją aktywacji** rozpoznającego dwie cyfry:





Po nauczeniu sprawdź, jak są klasyfikowane następujące obrazki: