NADZĘDZIE SZTUCZNEJ INTELIGENCJI

Uczenie sieci neuronowych

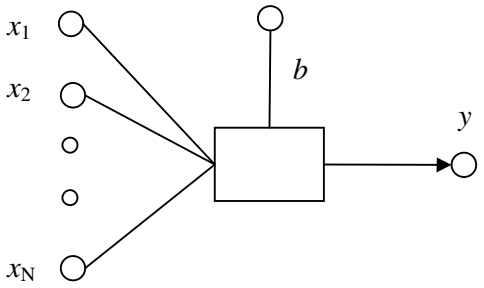
CEL: Znaleźć **układ wag,** aby *zminimalizować* **funkcję błędu**

I. Uczenie perceptronu

1. Reguła perceptronowa (dla dyskretnych neuronów)

$$W_{new} = W_{old} + \eta\,(d-y)X$$
$$b_{new} = b_{old} + \eta\,(d-y)$$

gdzie: W_{new} : nowy wektor wag
W_{old} : wektor wag w poprzednim kroku uczenia
b_{new} : nowa wartość odchylenia
b_{old} : wartość odchylenia w poprzednim kroku uczenia
η : współczynnik uczenia (wsp. korekcji)
d : żądana odpowiedź
y : otrzymana odpowiedź
X : wektor wejściowy



2. Reguła DELTA (dla ciągłych neuronów)

$$W_{new} = W_{old} + \eta\,(d-y)\,f'(net)X$$
$$b_{new} = b_{old} + \eta\,(d-y)\,f'(net)$$

Pochodne niektórych funkcji aktywacji

a) Funkcja *liniowa*

$f(x) = x \rightarrow f'(x) = 1$

b) Funkcja *sigmoidalna unipolarna*

$$f(x) = \frac{1}{1+e^{-\lambda x}} \rightarrow f'(x) = \lambda f(x)\,[1 - f(x)]$$

c) Funkcja *sigmoidalna bipolarna*

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1 \rightarrow f'(x) = \frac{\lambda}{2}\left[1 - f^2(x)\right]$$

d) Funkcja *tangens hiperboliczny*

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \rightarrow f'(x) = 1 - f^2(x)$$

3. Algorytm uczenia

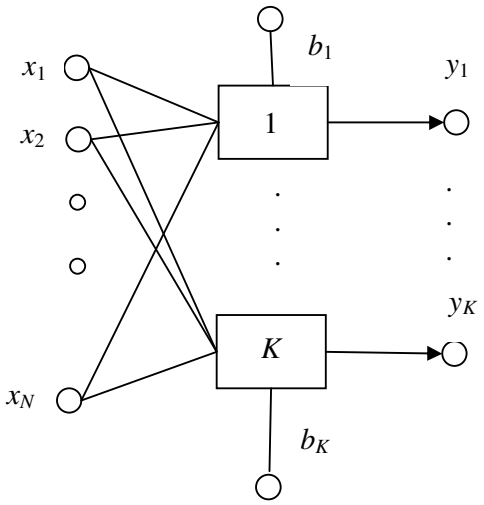
Wejście: Ciąg treningowy $\{(X_1, d_1), (X_2, d_2), \ldots, (X_p, d_p)\}$
Wyjście: Wektor wag W, wartość odchylenia b

Krok 1: Wylosuj początkowy wektor wag W i poczatkowe odchylenie b
Krok 2: Dla każdego wektora uczącego X
2.1 Wyznacz y: $y = f(WX + b)$
2.2 Uaktualizuj wagi

$W_{new} = W_{old} + \eta.(d - y).X$ (gdy neuron jest dyskretny)
$b_{new} = b_{old} + \eta\,(d - y)$
$W_{new} = W_{old} + \eta.(d - y).f'(net).X$ (gdy neuron jest ciągły)
$b_{new} = b_{old} + \eta\,(d - y).f'(net)$

Krok 3: Jeśli wagi pozostały bez zmian lub $E<E_{min}$ to **stop**, wpp. powrót do **Krok 2**

II. Uczenie jednowarstwowej sieci neuronowej



Wejście: Ciąg treningowy $\{(X_1, D_1), (X_2, D_2),.., (X_p, D_p)\}$
Wyjście: Macierz wag $W_{K\times N}$, wektor odchyleń $B_{K\times 1}$
Oznaczenie: W^i : Wektor wag i-tego neuronu
d^i : i–ta składowa żądanego wektora wyjściowego
y^i : i-ta składowa wektora wyjściowego sieci

Algorytm uczenia

Krok 1: Wylosuj początkową macierz wag W i początkowy wektor odchyleń B
Krok 2: Dla każdego wektora uczącego X
2.1 Wyznacz Y: $Y = f(W.X + B)$
2.2 Uaktualizuj wagi (dla i =1,...,K)

$$W^i_{new} = W^i_{old} + \eta.(d^i - y^i).X$$

$$b^i_{new} = b^i_{old} + \eta.(d^i - y^i)$$

(gdy neuron jest dyskretny)

$$W^i_{new} = W^i_{old} + \eta.(d^i - y^i).f'(net_i)X$$

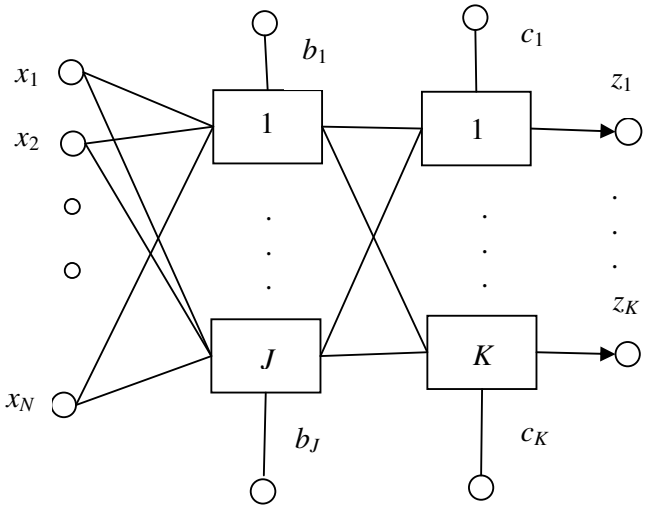
$$b^i_{new} = b^i_{old} + \eta.(d^i - y^i).f'(net_i)$$

(gdy neuron jest ciągły)
Krok 3: Jeśli wagi pozostały bez zmian lub $E<E_{min}$ to **stop**, wpp. powrót do **Krok 2**

III. Uczenie sieci wielowarstwowej:

Algorytm wstecznej propagacji błędu

Wymaganie: Sieć składa się wyłącznie z **neuronów ciągłych**



Wejście: Ciąg treningowy $\{(X_1, D_1),.., (X_p, D_p)\}$
Wyjście: Macierze wag: $W_{J\times N}$, $V_{K\times J}$

Wektory odchyleń: $B_{J\times 1}$, $C_{K\times 1}$

1. Wyznaczanie błędu neuronu w sieci

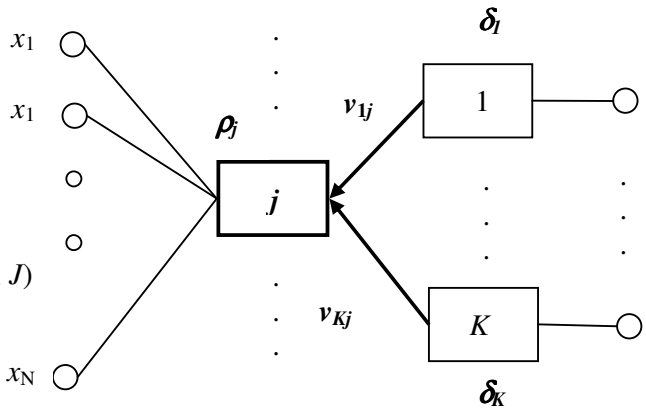
a) Warstwa wyjściowa

$\delta_k = f'(net_k)\,(d_k - y_k)$

(dla $k = 1, 2, ..., K$)

b) Warstwa ukryta

$$\rho_j = f'(net_k)\sum_{k=1}^{K} v_{kj}\delta_k \quad (\text{dla } j = 1, 2, ..., J)$$



2. <u>Algorytm uczenia</u>

Krok 1: Wylosuj początkowe macierze wag W, V i początkowe wektory odchyleń B, C

Krok 2: Dla każdego wektora uczącego X

2.1 Wyznacz wektor wyjściowej z I warstwy (ukrytej)

$Y = f(W.X + B)$

2.2 Wyznacz wektor wyjściowej z II warstwy (wyjściowej)

$Z = f(V.Y + C)$

2.3 Wyznacz błędy neuronów

a) Warstwa wyjściowa:

$\delta_k = f'(net_k)\,(d_k - y_k)$ (dla $k = 1, 2, ..., K$)

b) Warstwa ukryta:

$$\rho_j = f'(net_k)\sum_{k=1}^{K} v_{kj}\delta_k \quad (\text{dla } j = 1, 2, ..., J)$$

2.2 Uaktualizuj wagi (dla $i = 1,...,K$)

a) Warstwa wyjściowa:

$V_{new}^k = V_{old}^k + \eta.\delta_k Y$ (dla $k = 1, 2, ..., K$)

$c_{new}^k = c_{old}^k + \eta.\delta_k$

b) Warstwa ukryta:

$W_{new}^j = W_{old}^j + \eta.\rho_j X$ (dla $j = 1, 2, ..., J$)

$b_{new}^j = b_{old}^j + \eta.\rho_j$

Krok 3: Jeśli wagi pozostały bez zmian lub $E<E_{min}$ to **stop**, wpp. powrót do **Krok 2**

IV. Funkcja błędu

1. Sieć z jednym wyjściem:

$$E = \frac{1}{2}\sum_{i=1}^{P}(d_i - y_i)^2$$

P - liczba wektorów uczących

2. Sieć z wieloma wyjściami

$$E = \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{K}(d_i^{(j)} - y_i^{(j)})^2$$

P - liczba wektorów uczących, K – liczba wyjść sieci