PB, 2009–2010

Sztuczna Inteligencja

Tematy projektów – Sieci Neuronowe

Projekt 1

Stwórz projekt implementujący jednokierunkową sztuczną neuronową złożoną z neuronów typu sigmoidalnego z algorytmem uczenia opartym na wstecznej propagacji błędów z wykorzystaniem metody największego spadku (metoda gradientowa).

Sieć powinna umożliwiać:

- definiowanie współczynnika uczenia (X pkt.),
- konfigurację ilości warstw ukrytych (X pkt.),
- ustalenie ilości neuronów w poszczególnych warstwach (X pkt.),
- odczyt danych treningowych i testujących ze zbiorów *.tab oraz *.arff (X pkt.),
- wizualizację błędów w trakcie procesu uczenia (X pkt.),
- ustalenie warunku zatrzymania procesu uczenia (maksymalny błąd lub ilość iteracji) (X pkt.),
- wybór metody największego spadku z członem momentum (X pkt.).
- wykorzystanie innych funkcji aktywacji neuronów (unipolarna, bipolarna),

Projekt 2

Stwórz aplikację wizualizującą działanie oraz uczenie pojedynczego neuronu typu:

- perceptron (X pkt.),
- Adaline (X pkt.).

Aplikacja powinna umożliwiać wyznaczenie równania prostej rozdzielającej dwa zbiory punktów podanych na płaszczyźnie (pod warunkiem, że taka prosta istnieje) i dodatkowo:

- wizualizację tej prostej oraz punktów (X pkt.),
- wykres błędów procesu uczenia (X pkt.),
- ustalenie warunku zatrzymania procesu uczenia (maksymalny błąd lub ilość iteracji) (X pkt.).

Pomoc do projektów

Projekt 1

Działanie algorytmu dla pojedynczej epoki rozpoczyna się od podania pierwszego wzorca uczącego na wejście sieci. Najpierw jest on przetwarzany przez pierwszą, następnie przez kolejną *k*-tą warstwę neuronów, gdzie przetworzenie przez pojedynczy *i*-ty neuron w tej warstwie dane jest zależnością:

$$y_i^k = f(s_i^k(t)) = f\left(\sum_{j=0}^{N_{k-1}} w_{ij}^k(t)\, x_j^k(t)\right)$$

Otrzymane w ten sposób sygnały z warstwy poprzedniej (*k*-tej) stają się sygnałami wejściowymi dla warstwy kolejnej (*k*+1). Znając sygnały wyjściowe warstwy ostatniej (*L*) oraz sygnał wzorcowy d_i (pożądany sygnał warstwy wyjściowej dla danej próbki), można obliczyć błąd na wyjściu sieci zgodnie z zależnością:

$$Q_i^L(t) = d_i^L(t) - y_i^L(t)$$

Teraz można zmodyfikować wagi neuronów warstwy ostatniej korzystając z reguły delta. Modyfikacja wagi *j*-tej neuronu *i*-tego przebiega zgodnie z wzorem:

$w_{ij}^L(t+1) = w_{ij}^L(t) + 2\eta\,\delta_i^L(t)\,x_j^L(t)$, gdzie η – współczynnik uczenia (0,1)

$\delta_i^L = Q_i^L(t)\,f'(s_i^L(t))$, gdzie f' – pochodna przyjętej funkcji aktywacji

Po tym kroku następuje modyfikacja wag neuronów warstw poprzednich (wsteczna propagacja błędów) zgodnie z zależnościami:

$$w_{ij}^k(t+1) = w_{ij}^k(t) + 2\eta\,\delta_i^k(t)\,x_j^k(t)$$

$$\delta_i^k = Q_i^k(t)\,f'(s_i^k(t))$$

$$Q_i^k(t) = \sum_{m=1}^{N_{k+1}} \delta_m^{k+1}(t)\, w_{mi}^{k+1}(t)$$

Po modyfikacji wag neuronów w warstwie pierwszej, na wejście sieci podawana jest kolejna próbka.

Dodanie członu momentum polega na przebudowaniu wzoru do modyfikacji wag:

$w_{ij}^{k}(t+1)=w_{ij}^{k}(t)+2\eta\,\delta_i^k(t)\,x_j^k(t)+\alpha[w_{ij}^{k}(t)-w_{ij}^{k}(t-1)]$, gdzie α – momentum (0,1)

Jak łatwo zauważyć, człon momentum modyfikuje daną wagę w zależności od wielkości jej zmiany w poprzednim kroku.

Funkcje unipolarna i bipolarna neuronu sigmoidalnego dane są wzorami:

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

Pierwsze pochodne tych funkcji dane są wzorami:

- unipolarna: $f'(x)=\beta f(x)(1-f(x))$
- bipolarna: $f'(x)=\beta(1-f^2(x))$

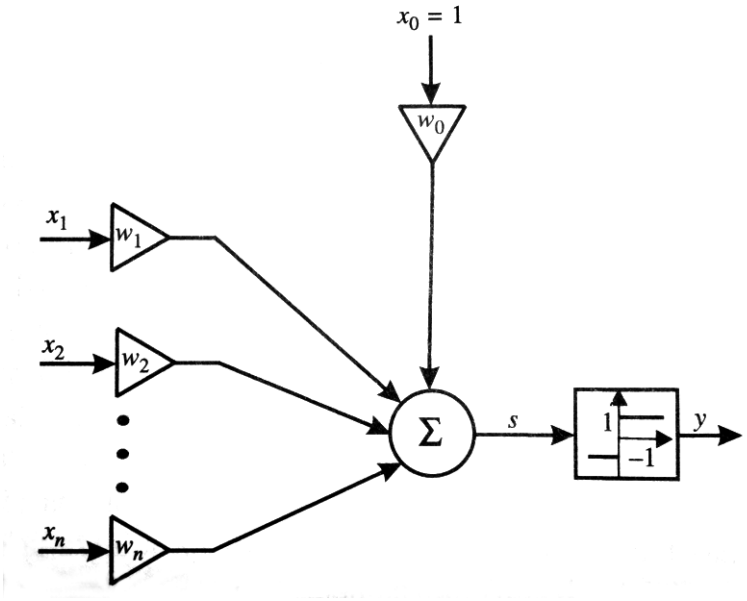
Podsumowanie wykorzystywanych oznaczeń:

- *i, m* – numer neuronu,
- *j* – numer wagi neuronu,
- *k* – numer warstwy,
- *L* – warstwa wyjściowa (numer ostatniej warstwy),
- *t* – iteracja w epoce, numer próbki uczącej,
- *x* – wartość wejściowa neuronu.

Projekt 2

Perceptron

Na rysunku nr 1 przedstawiony jest schemat neuronu typu perceptron.



Rys. 1: Schemat neuronu typu perceptron

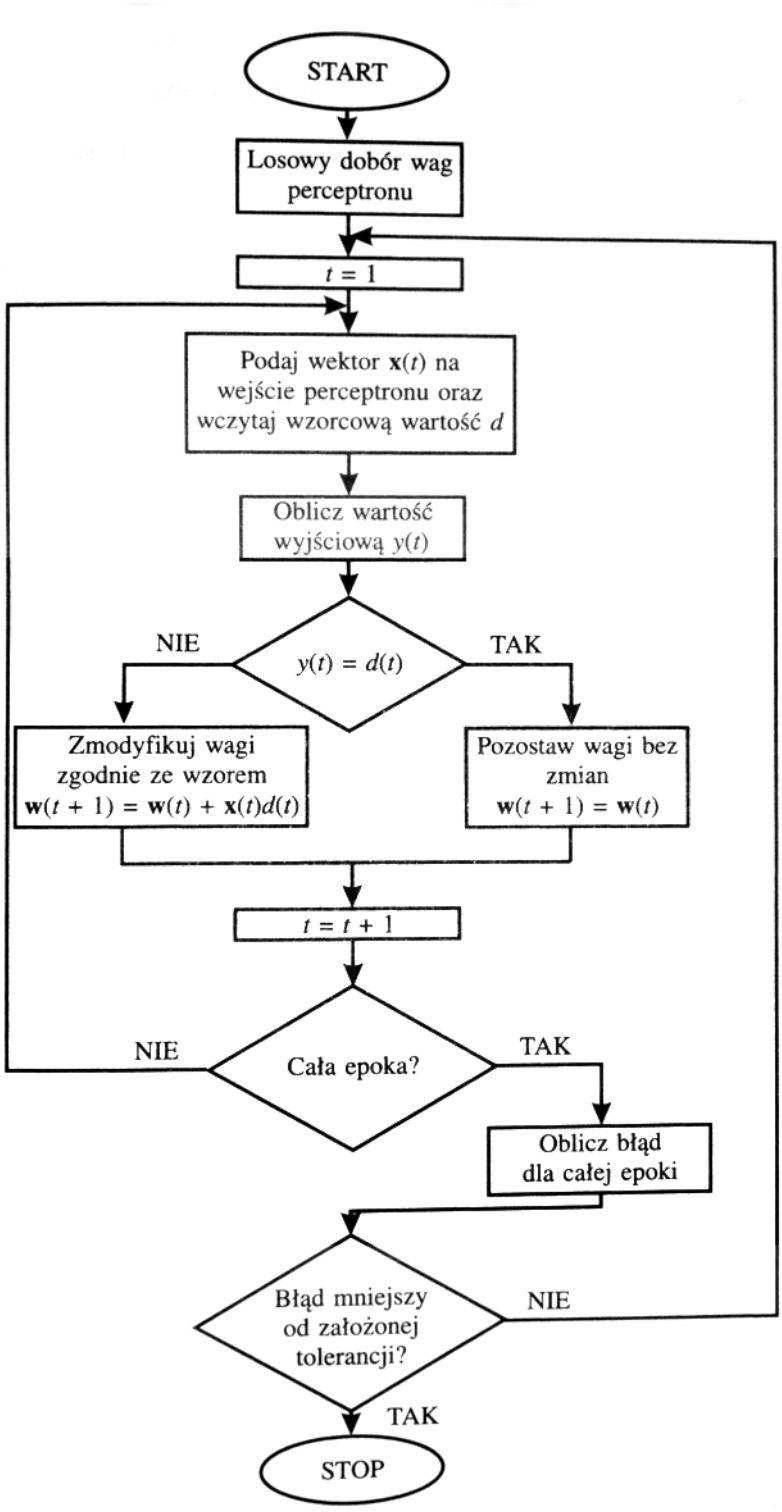
Funkcja aktywacji tego neuronu dana jest zależnością:

$$f(x)=\begin{cases} 1, & gdy\ x>0 \\ -1, & gdy\ x\leq 0 \end{cases}$$

Wartość wyjścia *y* perceptronu opisana jest równaniem:

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

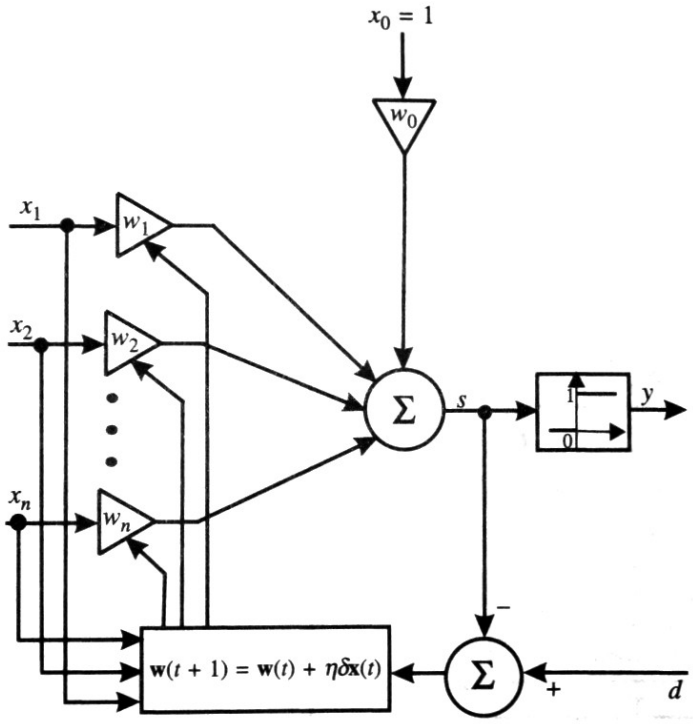
Algorytm uczenia neuronu typu perceptron przedstawiony jest w postaci schematu blokowego na rysunku nr 2.



Rys. 2: Algorytm uczenia neuronu typu perceptron

Adaline

Na rysunku nr 3 przedstawiony jest schemat neuronu typu Adaline (ang. *Adaptive Linear Neuron*). Widać wyraźnie, że różnica pomiędzy neuronem typu perceptron oraz neuronem typu Adaline polega na tym, że w procesie uczenia sygnał wzorcowy *d* jest porównywany z sygnałem *s* sumatora.



Rys. 3: Schemat neuronu typu Adaline

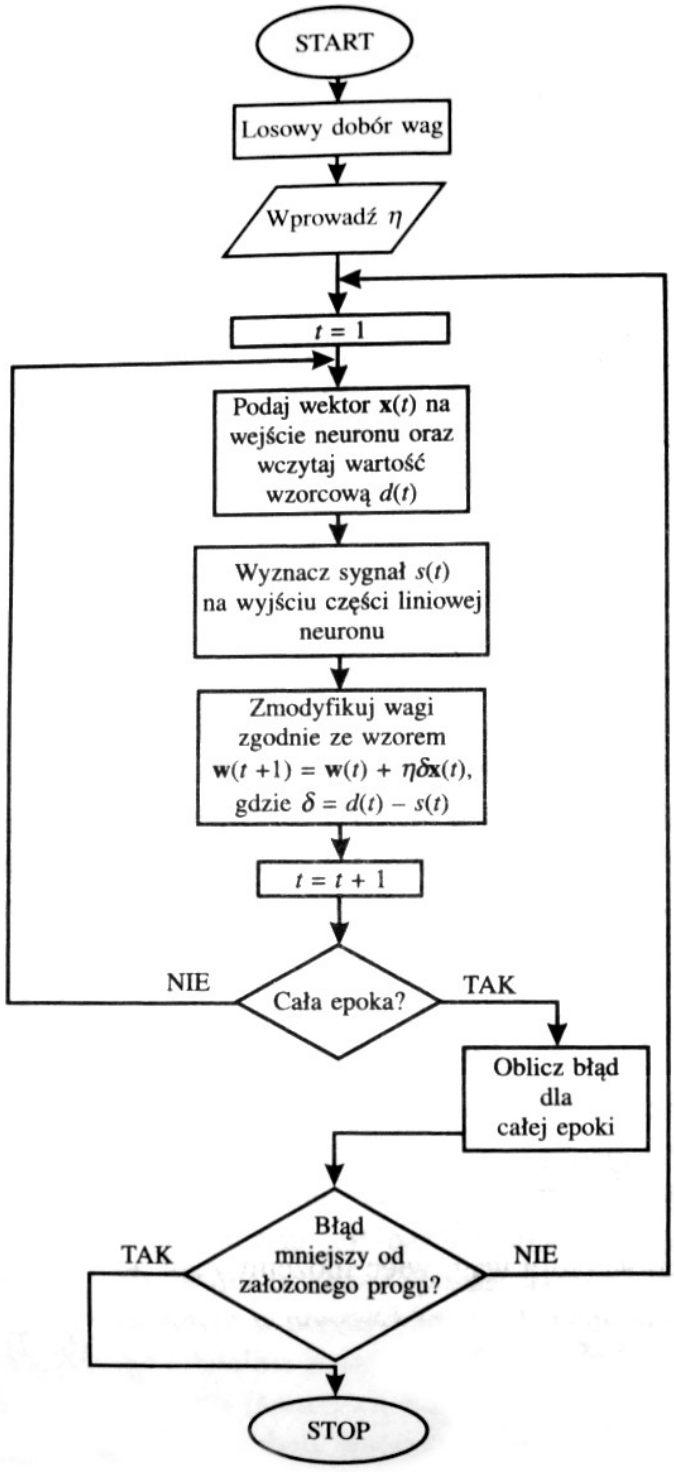
Funkcja aktywacji tego neuronu dana jest zależnością:

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

Wartość wyjścia *y* neuronu typu Adaline opisana jest równaniem:

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

Algorytm uczenia neuronu typu Adaline przedstawiony jest w postaci schematu blokowego na rysunku nr 4.



Rys. 4: Algorytm uczenia neuronu typu Adaline

Podsumowanie wykorzystywanych oznaczeń:

- i – numer wagi neuronu,
- t – numer iteracji w epoce, numer próbki uczącej,
- d – sygnał wzorcowy,
- y – sygnał wyjściowy neuronu,
- s – sygnał wyjściowy sumatora neuronu,
- x – wartość wejściowa neuronu,
- η – współczynnik uczenia (0,1).

Przykład

Neurony o dwóch wejściach można wykorzystać do podziału płaszczyzny dwuwymiarowej na dwie części za pomocą prostej. Podział ten wyznacza prosta o równaniu:

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

Po przekształceniu:

$$x_2 = -\frac{w_1}{w_2} \cdot x_1 - \frac{w_0}{w_2}$$

Czyli współczynniki takiej prostej zależą od wartości wag neuronu.

Teraz stosując algorytm uczenia neuronu można te współczynniki wyznaczyć.

Przykładowy ciąg uczący:

x_1	x_2	$d(x)$
2	1	1
2	2	1
0	6	1
-2	8	-1
-2	0	-1
0	0	-1
4	-20	-1