Ćwiczenie 10

SIECI NEURONOWE I – SIECI JEDNOKIERUNKOWE

Celem ćwiczenia jest zaznajomienie studentów z podstawowymi pojęciami z zakresu sieci neuronowych, takimi jak: neuron, sieć neuronowa, funkcja aktywacji, struktura sieci neuronowej, uczenie sieci neuronowej. Podczas ćwiczenia badana jest wielowarstwowa jednokierunkowa sieć neuronowa (wielowarstwowy perceptron) wraz z jej algorytmem uczenia - algorytmem propagacji wstecz. W szczególności badana jest możliwość aproksymacji dowolnej funkcji za pomocą tego typu sieci.

I. WSTĘP

Początki badań nad modelami biologicznych (naturalnych) sieci neuronowych sięgają lat czterdziestych dwudziestego wieku, kiedy to sformułowano matematyczny model pojedynczego neuronu. Od tego czasu do chwili obecnej dziedzina wiedzy zajmująca się modelami sieci neuronowych (sztucznymi sieciami neuronowymi) rozwinęła się w sposób, który z pewnością zadziwił wielu luminarzy współczesnej nauki. Rozwój ten dotyczy zarówno samej teorii sztucznych sieci neuronowych, a więc ich struktur i algorytmów uczenia, jak również dziedzin ich zastosowania. Prawdopodobnie łatwiej obecnie wymienić obszary nauki, w których sieci neuronowych nie usiłowano jeszcze zastosować niż te, w których z mniejszym lub większym powodzeniem sztuczne sieci neuronowe są wykorzystywane.

Obecnie w wyrażeniu "sztuczne sieci neuronowe", określającym software'ową lub (rzadziej) hardware'ową implementację matematycznego modelu sieci neuronowej, najczęściej pomija się przymiotnik "sztuczne",

mówiąc po prostu "sieci neuronowe". My również w niniejszej instrukcji przyjmiemy taką konwencję.

II. PODSTAWOWE POJĘCIA

1. Sieć neuronowa i jej elementy składowe

Sieć neuronowa

Siecią neuronową nazywamy układ wzajemnie połączonych podstawowych elementów nazywanych neuronami.

Model neuronu

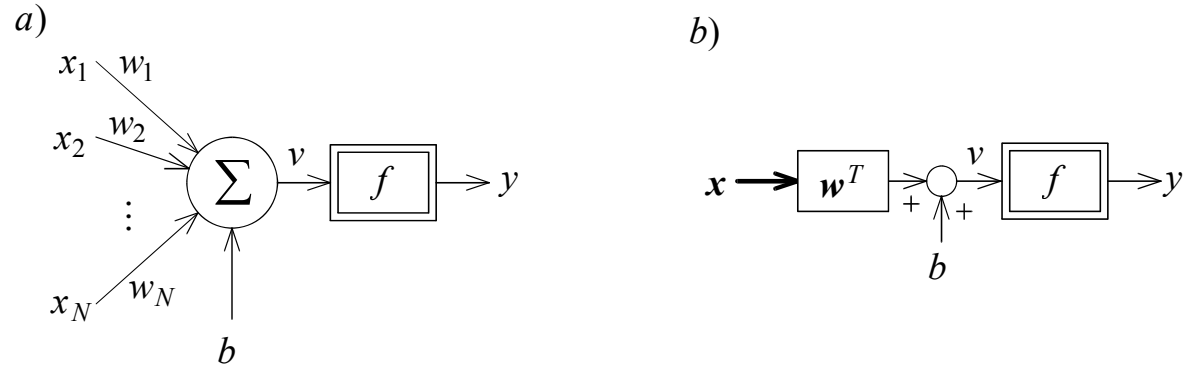
Neuron jest elementem statycznym posiadającym N wejść i jedno wyjście. Został przedstawiony na rysunku 1a. Sygnał wyjściowy neuronu zależy od sygnałów wejściowych i wyraża się następującym wzorem [1]:

$$y = f\left(\sum_{j=1}^{N} w_j x_j + b\right) \tag{1}$$

gdzie:

y	- sygnał wyjściowy neuronu,
$x_1, x_2, \ldots, x_N$	- sygnały wejściowe neuronu,
$f(\cdot)$	- funkcja aktywacji neuronu,
$w_1, w_2, \ldots, w_N$	- wagi neuronu,
b	- składnik stały (przesunięcie).

Wielkość występująca we wzorze (1) w nawiasie, będąca ważoną sumą sygnałów wejściowych i składnika stałego, nazywana jest pobudzeniem neuronu i oznaczana tutaj będzie symbolem v.



Rys.1. Model neuronu

[1] W literaturze spotykane są również inne, bardziej złożone, modele neuronu.

Dla uproszczenia wzoru (1) często zakłada się istnienie dodatkowego sygnału x_0 stale równego jedności, który wchodzi do ważonej sumy z wagą $w_0 = b$. Mamy wtedy

$$y = f\left(\sum_{j=0}^{N} w_j x_j \right)$$

Przesunięcie *b* można więc traktować jako zerową wagę neuronu.

Po wprowadzeniu zapisu wektorowego sygnałów wejściowych i wag $\boldsymbol{x}^T = [x_1, x_2, \ldots x_N]$, $\boldsymbol{w}^T = [w_1, w_2, \ldots, w_N]$ uzyskamy następującą postać wzoru (1):

$$y = f(\boldsymbol{w}^T \boldsymbol{x} + b) \tag{2}$$

Graficznie zależność (2) została przedstawiona na rysunku 1b. Sygnał wektorowy *x* został na nim zaznaczony pogrubioną linią.

Funkcja aktywacji

Funkcja aktywacji neuronu może być dowolną (w ogólności nieliniową) funkcją przekształcającą pobudzenie neuronu *v* w sygnał wyjściowy *y*

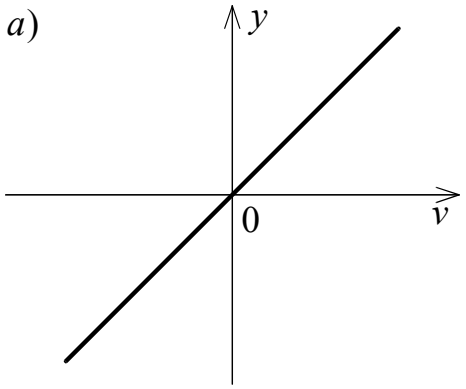
$$y = f(v). \tag{3}$$

Przykłady takich funkcji zostały przedstawione na rys. 2

W sieciach neuronowych uczonych przy użyciu metod gradientowych wykorzystuje się neurony, których funkcje aktywacji są różniczkowalne ze względu na *v*. Spośród prezentowanych na rys.2. funkcji aktywacji cechę taką posiadają funkcje *a*), *e*) i *f*), a więc funkcja liniowa, tangens hiperboliczny i funkcja sigmoidalna. Funkcje te ponadto charakteryzują się cechą wygodną ze względów obliczeniowych. Mianowicie wartości ich pochodnych w prosty sposób zależą od wartości tych funkcji. Dla funkcji *a*), *e*) i *f*) mamy odpowiednio:

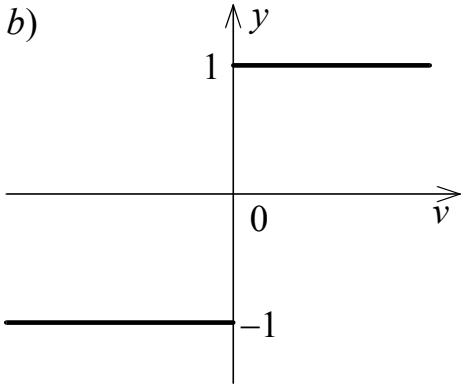
$$\begin{aligned}
y = f(v) = v \quad &\Rightarrow \quad f'(v) = 1 \\
y = f(v) = \mathrm{th}(v) \quad &\Rightarrow \quad f'(v) = \frac{1}{2}(1+y)(1-y) \\
y = f(v) = \frac{1}{1+e^{-v}} \quad &\Rightarrow \quad f'(v) = y(1-y)
\end{aligned} \tag{4}$$

a)



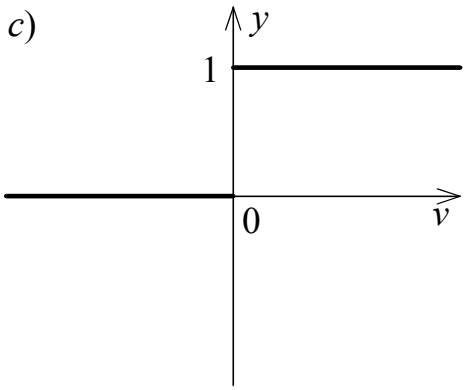
$$y = v$$

b)



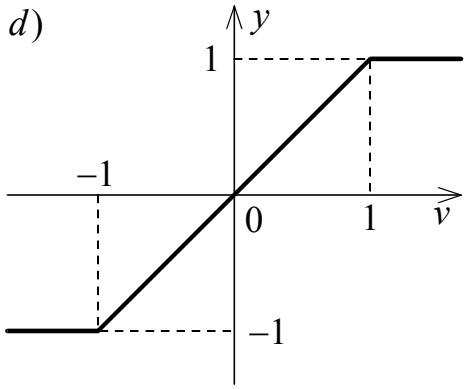
$$y = \mathrm{sgn}(v) = \begin{cases} -1 & gdy \quad v < 0 \\ 1 & gdy \quad v \geq 0 \end{cases}$$

c)



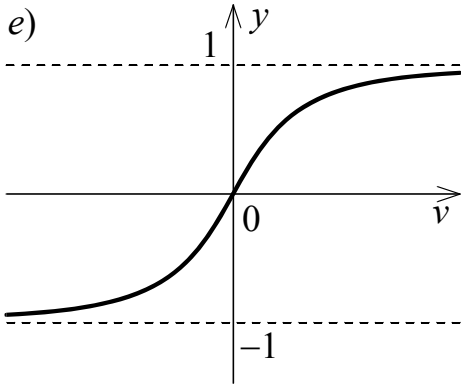
$$y = \mathbf{1}(v) = \begin{cases} 0 & gdy \quad v < 0 \\ 1 & gdy \quad v \geq 0 \end{cases}$$

d)



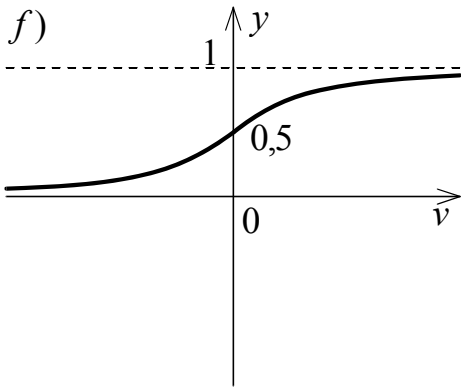
$$y = \begin{cases} -1 & gdy \quad v < -1 \\ v & gdy \quad |v| \leq 1 \\ 1 & gdy \quad v > 1 \end{cases}$$

e)



$$y = \mathrm{th}(v) = \frac{1 - e^{-v}}{1 + e^{-v}}$$

f)



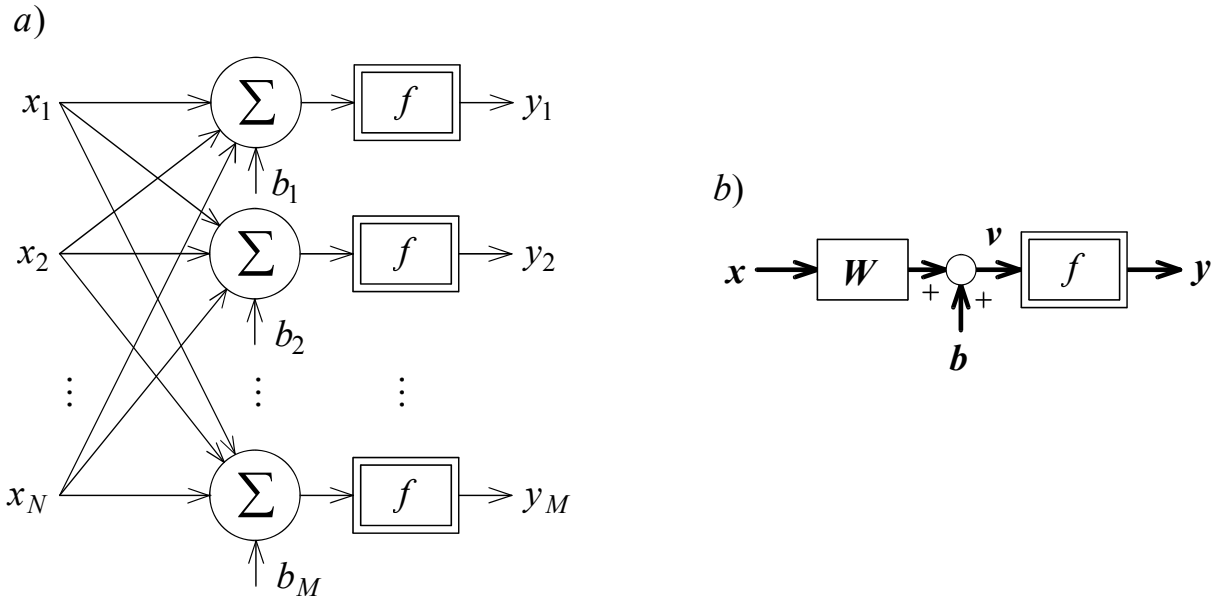
$$y = \frac{1}{1 + e^{-v}}$$

Rys. 2. Przykłady funkcji aktywacji neuronu

Warstwa neuronów

Często wygodnie jest wydzielić spośród wszystkich neuronów sieci pewne grupy nazywane warstwami. Cechą wspólną neuronów należących do jednej warstwy jest to, że posiadają te same sygnały wejściowe.

Rozpatrzmy warstwę złożoną z M neuronów - rysunek 3a.



Rys.3. Warstwa neuronów

Sygnał wyjściowy i-tego neuronu dany jest wzorem

$$y_i = f\left(\sum_{j=0}^{N} w_{ij} x_j + b_i\right), \qquad i = 1,2,\ldots,M. \tag{5}$$

Podobnie jak wcześniej możemy zastosować zapis macierzowy. Jeśli wprowadzimy oznaczenia:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}, \quad \boldsymbol{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MN} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix} \tag{6}$$

uzyskamy macierzową postać równania (5)

$$\boldsymbol{y} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}). \tag{7}$$

Ilustruje to rysunek 3b.

2. Projektowanie sieci neuronowych

Zanim przystąpi się do wykorzystania sieci neuronowej do konkretnego celu należy ją zaprojektować. Projektowanie sieci neuronowej można podzielić na dwa etapy: określenie struktury sieci neuronowej oraz uczenie sieci neuronowej.

Określenie struktury sieci neuronowej

Polega ono na:

a) określeniu liczby neuronów sieci,

b) określeniu dla każdego neuronu postaci funkcji aktywacji,

c) określeniu dla każdego neuronu sygnałów wejściowych,

Jeżeli sygnały wejściowe grupy neuronów są jednakowe, można je traktować jako warstwę neuronów. Z reguły dla neuronów jednej warstwy przyjmuje się taką samą funkcję aktywacji.

Uczenie sieci neuronowej

Uczenie sieci neuronowej polega na takim doborze wag tej sieci, aby spełniała ona wymagania stawiane przed nią przez jej projektanta.

W praktyce zdarza się, że wyodrębnione powyżej fazy pracy sieci neuronowej nie są rozdzielne. W szczególności podczas wykorzystywania (stosowania) sieci neuronowej do konkretnego celu może następować jej "douczanie". Ponadto w niektórych algorytmach podczas uczenia sieci neuronowej możliwa jest modyfikacja jej struktury.

II. WIELOWARSTWOWA JEDNOKIERUNKOWA SIEĆ NEURONOWA (WIELOWARSTWOWY PERCEPTRON)

1. Struktura sieci

Wielowarstwowa jednokierunkowa sieć neuronowa została przedstawiona na rysunku 4a.

Struktura takiej sieci posiada następujące cechy:

a) neurony podzielone są na P warstw,

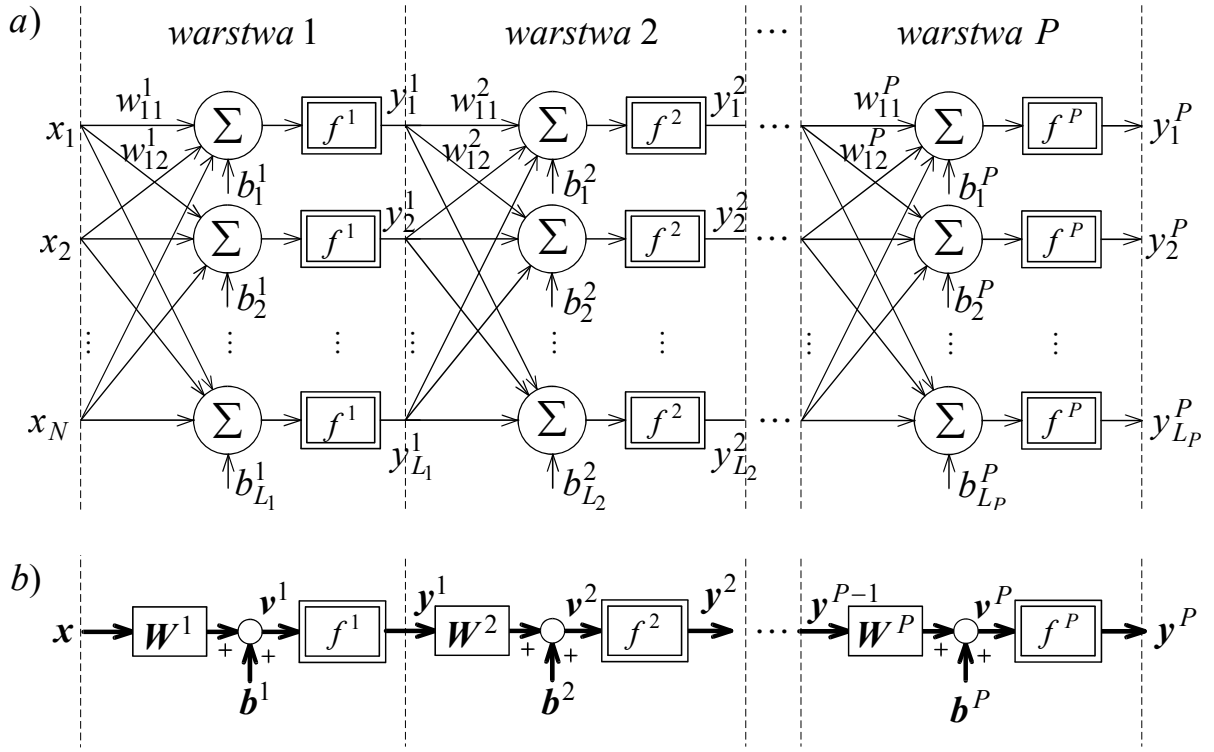
b) sygnałami wejściowymi k-tej (następnej) warstwy są sygnały wyjściowe warstwy $k-1$-szej (poprzedniej),

c) zewnętrzne sygnały wejściowe podawane są na wejścia neuronów warstwy pierwszej,

d) sygnały wyjściowe warstwy P-tej (ostatniej) są sygnałami wyjściowymi całej sieci.

Przyjęło się nazywać wszystkie warstwy sieci, za wyjątkiem ostatniej, warstwami ukrytymi.

Wielowarstwowa jednokierunkowa sieć neuronowa nazywana jest również wielowarstwowym perceptronem [2].



Rys.4. Wielowarstwowy perceptron

Przyjmujemy konwencję, zgodnie z którą indeks górny oznacza numer warstwy. Sygnał wyjściowy *i*-tego neuronu *k*-tej warstwy dany jest wzorem:

$$y_i^k = f^k\left(\sum_{j=1}^{L_{k-1}} w_{ij}^k y_j^{k-1} + b_i^k\right) \tag{8}$$

gdzie:

w_{ij}^k - waga połączenia *i*-tego neuronu *k*-tej warstwy z *j*-tym neuronem warstwy wcześniejszej,

b_i^k - przesunięcie *i*-tego neuronu *k*-tej warstwy,

f^k - funkcja aktywacji neuronów *k*-tej warstwy,

L_k - liczba neuronów w warstwie *k*-tej.

[2]Niekiedy nazwa ta jest rezerwowana jedynie dla wielowarstwowych jednokierunkowych sieci neuronowych, których neurony mają funkcje aktywacji w postaci skoku jednostkowego.

Wejściami pierwszej warstwy są sygnały $x_1, x_2, \ldots, x_N$. Stąd aby wzór (8) obowiązywał również dla warstwy pierwszej przyjmuje się $y_i^0 = x_i$ dla $i = 1,2,\ldots,N$.

Wykorzystując jak poprzednio zapis macierzowy otrzymamy postać macierzową wzoru (8):

$$\boldsymbol{y}^k = f(\boldsymbol{W}^k \boldsymbol{y}^{k-1} + \boldsymbol{b}^k) \tag{9}$$

Odpowiada temu graficzna reprezentacja przedstawiona na rysunku 4b.

2. Algorytm propagacji wstecz

Zadaniem algorytmu uczenia wielowarstwowego perceptronu jest taki dobór wag neuronów sieci aby dla danego sygnału wejściowego sieci *x* sygnał na jej wyjściu y^P był równy sygnałowi zadanemu *d*. Aby to osiągnąć definiuje się wskaźnik jakości wyrażający miarę niedopasowania wektorów y^P i *d*

$$J = \frac{1}{2}\sum_{i=1}^{L_P}(y_i^P - d_i)^2 . \tag{10}$$

Par {*x*,*d*} jest oczywiście więcej. Wszystkie one tworzą tzw. zbiór uczący. W kolejnych iteracjach algorytmu uwzględnia się jedną parę ze zbioru uczącego. Doboru wag sieci dokonuje się przy użyciu iteracyjnej metody gradientowej nazywanej algorytmem propagacji wstecz (ang. backpropagation):

$$w_{ij}^k(t+1) = w_{ij}^k(t) - \eta\frac{\partial J}{\partial w_{ij}^k(t)}, \tag{11}$$

$$b_i^k(t+1) = b_i^k(t) - \eta\frac{\partial J}{\partial b_i^k(t)} \tag{12}$$

gdzie *t* oznacza numer iteracji algorytmu propagacji wstecz. "Nowa" wartość wagi sieci neuronowej zależy od jej "starej" wartości oraz od pochodnej wskaźnika jakości względem tej wagi. Zmiana wagi następuje w kierunku "malenia" wskaźnika jakości *J*. Współczynnik η jest małą dodatnią liczbą i nazywany jest współczynnikiem nauki.

Idea algorytmu polega na sposobie wyznaczenia pochodnych występujących we wzorach (11), (12). Wyprowadzimy obecnie wzory służące do ich wyznaczania. Dla uproszczenia dalszych rozważań będziemy pomijać

numer iteracji t. Ponadto założymy (co nie wpłynie na ogólność rozważań), że funkcje aktywacji wszystkich neuronów sieci są jednakowe.

Aby wyznaczyć pochodną $\frac{\partial J}{\partial w_{ij}^k}$ spójrzmy jeszcze raz na rysunek 4. Waga w_{ij}^k wpływa na wartość wskaźnika jakości jedynie poprzez pobudzenie v_i^k. Poszukiwana pochodna równa jest zatem

$$\frac{\partial J}{\partial w_{ij}^k} = \frac{\partial J}{\partial v_i^k}\frac{\partial v_i^k}{\partial w_{ij}^k}. \tag{13}$$

Zauważmy, że drugi czynnik w otrzymanym wzorze $\frac{\partial v_i^k}{\partial w_{ij}^k} = y_j^{k-1}$. Wprowadźmy ponadto oznaczenie $\delta_i^k = \frac{\partial J}{\partial v_i^k}$. Otrzymamy wzór

$$\frac{\partial J}{\partial w_{ij}^k} = \delta_i^k y_j^{k-1}, \tag{14}$$

w którym nieznana jest wielkość δ_i^k. Dla ostatniej (wyjściowej) warstwy sieci otrzymamy

$$\delta_i^P = \frac{\partial J}{\partial v_i^P} = \frac{\partial J}{\partial y_i^P}\frac{\partial y_i^P}{\partial v_i^P}. \tag{15}$$

Korzystając z (10) możemy wyznaczyć pierwszy czynnik prawej strony równania (15):

$$\frac{\partial J}{\partial y_i^P} = y_i^P - d_i. \tag{16}$$

Drugi czynnik jest natomiast pochodną funkcji aktywacji $f'(v_i^P)$. Dla ostatniej warstwy sieci otrzymujemy więc ostatecznie

$$\delta_i^P = f'(v_i^P)(y_i^P - d_i). \tag{17}$$

Dla neuronów warstw wcześniejszych (ukrytych) dostajemy natomiast

$$\delta_i^k = \frac{\partial J}{\partial v_i^k} = \frac{\partial J}{\partial y_i^k}\frac{\partial y_i^k}{\partial v_i^k} = \frac{\partial J}{\partial y_i^k} f'(v_i^k). \tag{18}$$

Ponieważ y_i^k wpływa na wskaźnik jakości poprzez stany pobudzeń wszystkich neuronów warstwy następnej v_m^{k+1}, pierwszy czynnik prawej strony równania (18) możemy rozpisać następująco

$$\frac{\partial J}{\partial y_i^k} = \sum_{m=1}^{L_{k+1}} \frac{\partial J}{\partial v_m^{k+1}} \frac{\partial v_m^{k+1}}{\partial y_i^k} = \sum_{m=1}^{L_{k+1}} \delta_m^{k+1} w_{mi}^{k+1}. \tag{19}$$

Podstawiając (19) do (18) otrzymamy ostatecznie wzór na δ_i^k dla warstw wcześniejszych

$$\delta_i^k = f'(v_i^k) \sum_{m=1}^{L_{k+1}} \delta_m^{k+1} w_{mi}^{k+1}; \qquad k = 1, \ldots, P-1. \tag{20}$$

Pozostaje jeszcze wyznaczenie pochodnych $\frac{\partial J}{\partial b_i^k}$ występujących we wzorze (12). Łatwo wykazać, że

$$\frac{\partial J}{\partial b_i^k} = \delta_i^k. \tag{21}$$

Sprawdzenie tego pozostawiamy czytelnikowi.

Jak widać ze wzoru (20) odpowiednie pochodne wyznaczane są w kierunku przeciwnym do kierunku przepływu sygnałów w sieci - stąd nazwa algorytmu.

Podobnie jak poprzednio możemy wprowadzić oznaczenia macierzowe:

$$\frac{\partial J}{\partial \boldsymbol{W}^k} = \begin{bmatrix} \frac{\partial J}{\partial w_{11}^k} & \frac{\partial J}{\partial w_{12}^k} & \cdots & \frac{\partial J}{\partial w_{1L_{k-1}}^k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial w_{L_k 1}^k} & \frac{\partial J}{\partial w_{L_k 2}^k} & \cdots & \frac{\partial J}{\partial w_{L_k L_{k-1}}^k} \end{bmatrix}, \quad \frac{\partial J}{\partial \boldsymbol{y}^k} = \begin{bmatrix} \frac{\partial J}{\partial y_1^k} \\ \vdots \\ \frac{\partial J}{\partial y_{L_k}^k} \end{bmatrix}, \quad \boldsymbol{\delta}^k = \begin{bmatrix} \delta_1^k \\ \vdots \\ \delta_{L_k}^k \end{bmatrix}$$

$$f'(\boldsymbol{v}^k) = \begin{bmatrix} f'(v_1^k) & 0 & \cdots & 0 \\ 0 & f'(v_2^k) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f'(v_{L_k}^k) \end{bmatrix}. \tag{22}$$

Uzyskamy wtedy zwięzłe macierzowe postaci wzorów (11), (12), (14), (16) i (21).

$$\boldsymbol{W}^k(t+1) = \boldsymbol{W}^k(t) - \eta \frac{\partial J}{\boldsymbol{W}^k(t)} \tag{23}$$

$$\boldsymbol{b}^k(t+1) = \boldsymbol{b}^k(t) - \eta \frac{\partial J}{\boldsymbol{b}^k(t)} \tag{24}$$

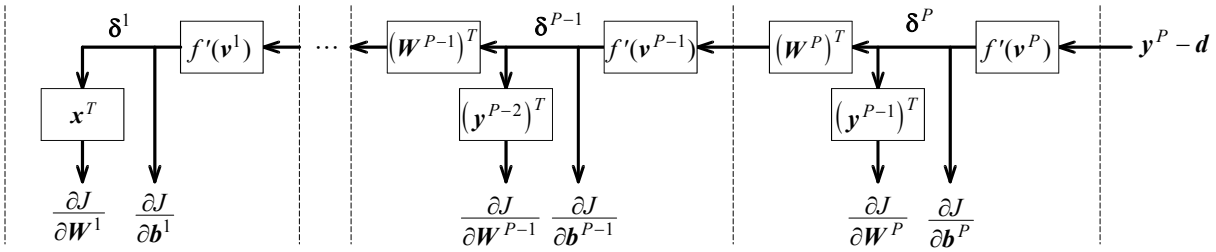
$$\frac{\partial J}{\partial \boldsymbol{W}^k} = \delta^k (\boldsymbol{y}^{k-1})^T \tag{25}$$

$$\frac{\partial J}{\partial \boldsymbol{b}^k} = \delta^k \tag{26}$$

$$\delta^P = f'(\boldsymbol{v}^P)(\boldsymbol{y}^P - \boldsymbol{d}) \tag{27}$$

$$\delta^k = f'(\boldsymbol{v}^k)(\boldsymbol{W}^{k+1})^T \delta^{k+1}; \quad k = 1, \ldots, P-1 \tag{28}$$

Propagację wstecz opisaną wzorami (23) - (28) można przedstawić w postaci schematu blokowego przedstawionego na rysunku 5.



Rys.5. Graficzna reprezentacja algorytmu propagacji wstecz

Podsumowując wszystkie wcześniejsze rozważania możemy teraz przedstawić algorytm uczenia wielowarstwowego perceptronu.

Algorytm propagacji wstecz

1. Ustaw wszystkie wagi sieci jako małe losowe liczby.
2. Podaj na wejście sieci kolejny sygnał uczący *x*.
3. Dokonaj propagacji wprzód zgodnie z (9) wyznaczając wszystkie sygnały sieci.
4. Porównaj sygnał wyjściowy sieci $\boldsymbol{y}^P$ z zadanym sygnałem wejściowym $\boldsymbol{d}$.
5. Dokonaj propagacji błędu wstecz zgodnie z (27) i (28).
6. Wyznacz nowe wartości wag korzystając z (23) - (26).
7. Wróć do punktu 2.

Początkowa randomizacja wag jest konieczna gdyż dla zerowych wartości wag odpowiednie pochodne byłyby również zerowe, co uniemożliwiłoby skuteczną naukę sieci.

Pochodną funkcji aktywacji $f'(\cdot)$ występującą we wzorach (27), (28) wygodnie jest obliczyć korzystając z zależności (4).

3. Własności wielowarstwowego perceptronu i algorytmu propagacji wstecz

Podstawową własnością wielowarstwowego perceptronu, która zadecydowała o tym, że jest to obecnie najchętniej i najczęściej wykorzystywana sieć neuronowa, jest jej zdolność do aproksymacji dowolnych nieliniowych funkcji.

Zdolności te są poparte odpowiednimi twierdzeniami. W szczególności dowodzi się, że za pomocą dwuwarstwowej jednokierunkowej sieci neuronowej można z dowolną dokładnością aproksymować dowolną ciągłą nieliniową funkcję nad dowolnym domkniętym i ograniczonym obszarem, przy czym dokładność tej aproksymacji zależy od liczby neuronów w warstwie pierwszej (ukrytej).

Niestety fakt, że dana sieć neuronowa o określonej strukturze posiada możliwość aproksymacji zadanej funkcji nie oznacza, że stosując algorytm propagacji wstecz zawsze uda się to osiągnąć. Przeszkodą może być zbyt duża wartość współczynnika nauki η co może spowodować destabilizację algorytm. Innym powodem może okazać się osiągnięcie przez algorytm tzw. lokalnego minimum. Algorytm propagacji wstecz jest bowiem algorytmem gradientowym minimalizującym wskaźnik jakości w przestrzeni wag. Wymiar tej przestrzeni jest równy liczbie wszystkich wag sieci neuronowej. Oprócz globalnego minimum wskaźnika jakości w przestrzeni tej mogą wystąpić minima lokalne, po osiągnięciu których algorytm zatrzymuje się. Rozwiązaniem tego problemu może być "wytrącenie" algorytmu z tego punktu równowagi poprzez niewielką zmianę wag lub ponowne zainicjowanie algorytmu dla innych początkowych wartości wag. Inną wadą standardowego algorytmu propagacji wstecz jest jego stosunkowo wolna zbieżność.

Reasumując, w przypadku gdy uczenie sieci nie zakończyło się sukcesem należy wykonać jedną z poniższych czynności:

- zmienić wartość współczynnika nauki η,
- zainicjować algorytm startując od innych początkowych wartości wag,
- w przypadku gdy powyższe działania nie przynoszą pozytywnego skutku zmienić strukturę sieci.

Nadmienić w tym miejscu wypada, że dużą rolę odgrywa doświadczenie projektanta sieci neuronowej.

4. Warianty algorytmu propagacji wstecz

Aby wyeliminować lub zmniejszyć wpływ wymienionych wyżej wad algorytmu propagacji wstecz stosuje się różne jego modyfikacje mające na celu usprawnienie procesu uczenia. Zainteresowanych odsyłamy do podanej literatury, w szczególności do [5]. Wymienimy tutaj jedynie dwa warianty omawianego algorytmu – uczenie ze współczynnikiem momentum oraz kumulacyjną propagację wsteczną.

Uczenie ze współczynnikiem momentum

Polega ono na uzupełnieniu wzorów (23) i (24) o składniki wnoszące informację o zmianie wag w poprzedniej iteracji algorytmu

$$\boldsymbol{W}^k(t+1) = \boldsymbol{W}^k(t) - \eta\frac{\partial J}{\boldsymbol{W}^k(t)} + \alpha\left(\boldsymbol{W}^k(t) - \boldsymbol{W}^k(t-1)\right). \tag{29}$$

$$\boldsymbol{b}^k(t+1) = \boldsymbol{b}^k(t) - \eta\frac{\partial J}{\boldsymbol{b}^k(t)} + \alpha\left(\boldsymbol{b}^k(t) - \boldsymbol{b}^k(t-1)\right) \tag{30}$$

Współczynnik α dobierany jest z przedziału $[0\,,1)$. Modyfikacja ta sprawia, że trajektoria w przestrzeni wag sieci neuronowej "wygładza się", a dla obszarów przestrzeni wag w których moduł gradientu wskaźnika jakości jest mały i uczenie przebiegałoby powoli, następuje przyśpieszenie procesu uczenia.

Kumulacyjny algorytm propagacji wstecz

Polega on na tym, że modyfikacji wag nie dokonuje się w każdym kroku algorytmu ale pochodne występujące we wzorach (23) i (24) sumuje się dla całego zbioru uczącego. Dopiero otrzymana suma jest wykorzystywana do modyfikacji wag.

III. PRZEBIEG ĆWICZENIA

1. Napisać program realizujący proces uczenia metodą propagacji wstecz wielowarstwowego perceptronu zadanej przez prowadzącego funkcji.
2. Zbadać wpływ:
   a) liczby warstw sieci,
   b) postaci funkcji aktywacji neuronów poszczególnych warstw sieci,
   c) liczby neuronów w poszczególnych warstwach,
   d) wartości współczynnika nauki i momentum,

   na:
   a) przebieg procesu uczenia sieci,
   b) jakość aproksymacji zadanej funkcji.

IV. SPRAWOZDANIE

Sprawozdanie powinno zawierać pełną dokumentację z przeprowadzenia ćwiczenia zgodnie z zadanym jego przebiegiem oraz omówienie uzyskanych rezultatów.

LITERATURA

1. J. Hertz, A. Krogh, R.G. Palmer, *Wstęp do teorii obliczeń neuronowych*, WNT, Warszawa 1993,
2. R. Tadeusiewicz, *Sieci neuronowe*, Akademicka Oficyna wydawnicza, Warszawa 1993,
3. J. Korbicz, A. Obuchowicz, D. Uciński, *Sztuczne sieci neuronowe - podstawy i zastosowania*, Akademicka Oficyna Wydawnicza, Warszawa 1994,
4. S. Osowski, *Sieci neuronowe*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1996,
5. S. Osowski, *Sieci neuronowe w ujęciu algorytmicznym*, WNT, Warszawa 1996.