Michał Radziejowski – Scenariusz 4 Sprawozdanie

Cel:

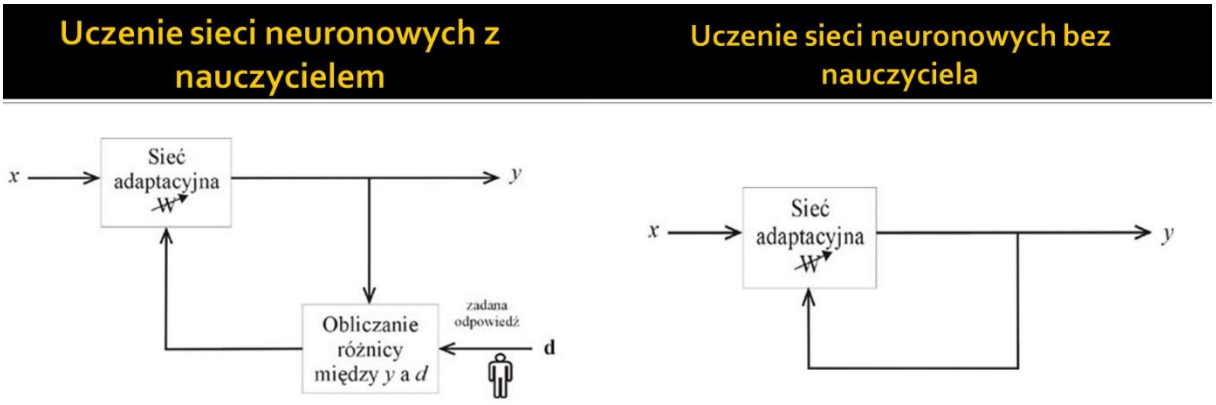
Celem ćwiczenia jest poznanie działania reguły Hebba, dla sieci jednowarstwowej na przykładzie grupowania liter alfabetu.

Wykonanie zadania:

1) Wygenerowano dane uczące i testujące, zawierające 20 wielkich liter alfabetu łacińskiego w postaci macierzy, każda wielkości 5x5.
2) Sieć została zaimplementowana w programie MatLab, dane uczące i testowe jako zbiór cyfr 1 oraz 0 (odpowiednio reprezentacja litery, oraz puste pole).
3) Uczono sieć, zmieniając zarówno współczynnik uczenia jak i zapominania
4) Przetestowano poprawne działanie sieci, bazując na zastosowaniu reguły Hebba do grupowania liter alfabetu

Opis zadania i algorytmu:

Do wykonania zadania, należało zastosować algorytm nauki bez nauczyciela. Skutkowało to znaczącą różnicą w algorytmie, w którym to po raz pierwszy nauczyciel nie podpowiadał prawidłowej wartości lecz sieć uczyła się sama bazując jedynie na własnej analizie reakcji na pobudzenia. Poniżej przykładowy schemat działania sieci z nauczycielem oraz bez.



Reguła Hebba, jest jedną z najpopularniejszych metod samouczenia sieci neuronowych. Jest to wcześniej wspomniana metoda nauki bez nauczyciela, polegająca na tym iż sieci pokazuje się kolejne przykłady sygnałów wejściowych, nie podając żadnych informacji o tym co z tymi sygnałami należy zrobić. Poprzez obserwacje otoczenia i odbiór różnych sygnałów sieć

stopniowo sama odkrywa jakie są zależności pomiędzy danymi wejścia i jakie jest ich znaczenie.

Po podaniu do sieci neuronowej kolejnych zestawów sygnałów wejściowych, tworzy się swego rodzaju rozkład – niektóre neurony są pobudzone bardzo silnie, inne słabiej, a jeszcze inne mogą mieć sygnały wyjściowe nawet ujemne. Interpretacja tych zachowań może być taka, iż niektóre neurony „rozpoznają" podawane sygnały jako „własne" (czyli takie, które są skłonne akceptować), inne traktują je „obojętnie", zaś jeszcze u innych neuronów wzbudzają one wręcz „awersję". Poprzez dokładniejszą analizę tego procesu, jesteśmy w stanie stwierdzić iż w momencie konsekwentnego stosowania algorytmu Hebba, początkowe (najczęściej przypadkowe) „preferencje" neuronów ulegają systematycznemu wzmacnianiu. Jeżeli jakiś neuron miał przypadkowo nadaną swego rodzaju skłonność do akceptowania sygnałów, to w miarę następnych epok uczenia będzie on w stanie nauczyć się te sygnały rozpoznawać co raz dokładniej i precyzyjniej. W wyniku tego procesu, sygnały podobne do siebie będą stopniowo w miarę postępu uczenia coraz skuteczniej grupowane i rozpoznawane przez kolejne grupy neuronów. Końcowym rezultatem procesu samouczenia jest sieć, która nauczy się ile klas podobnych do siebie sygnałów pojawia się na wejściach i sama przyporządkuje tym klasom sygnałów neurony, które nauczą się je rozróżniać i rozpoznawać.

Przykłady działania programu:

Program po jego wywołaniu tworzy sieć neuronową oraz obrazy danych wejściowych w postaci tablic:

```
TestLiteraA = [0 1 1 1 0;
               1 0 0 0 1;
               1 1 1 1 1;
               1 0 0 0 1;
               1 0 0 0 1];

TestLiteraB = [1 1 1 1 0;
               1 0 0 0 1;
               1 1 1 1 0;
               1 0 0 0 1;
               1 1 1 1 0];
```

Po inicjacji wag z zakresu od 1 do -1, sieć zostaje przetestowana na podstawie danych początkowych:

```
TEST:
A -0.879029
B -0.525745
C 0.559906
D -0.944799
E 0.385148
F -0.280960
G -0.630897
H -0.459086
I 0.708061
J -0.553450
K 0.911262
L -0.952940
M -0.900191
N -0.772268
O -0.689742
P -0.606099
R -0.050813
S -0.371371
T 0.785626
U -0.977442
```

Następnie, program wypisuje nam wagi wejściowe jak i wyjściowe:

```
WAGI WEJSCIOWE: WAGI WYJSCIOWE:
 0.062419        -0.320466
 -0.782364       -0.339093
 0.263533        -0.339093
 -0.747000       -0.331445
 -0.731393       -0.499363
 -0.802812       -0.360354
 -0.715946       -0.046148
 -0.663497       -0.204523
 -0.607502       0.015330
 -0.365040       -0.024316
 -0.367142       -0.187189
 -0.564873       -0.168598
 -0.497916       -0.457029
 0.785845        -0.198710
 0.406446        -0.099778
 0.111476        -0.187189
 -0.631133       0.018774
 -0.575938       -0.204523
 -0.845306       0.047060
 0.827601        -0.254024
 0.413430        -0.303668
 0.115578        -0.332958
 -0.373142       -0.462733
 -0.667593       -0.306537
 0.244995        -0.053380
```

Na samym końcu, program drukuje output nauczonej sieci, wraz z opcjonalnym testem litery która nie była określona w danych wejściowych. W tym wypadku jest to litera W.

```
TEST:
A 0.161563
B 0.990357
C 0.254896
D 0.649746
E 0.990845
F 0.523203
G 0.936987
H -0.027523
I -0.162929
J -0.435511
K -0.014356
L -0.586813
M -0.356031
N -0.385498
O 0.083020
P 0.075487
R 0.081787
S 0.936325
T 0.016675
U -0.106483

ans =

   -0.4445
```

Wyniki:

Wyniki zostały zebrane w arkuszu Excel oraz pogrupowane według wartości dla każdej litery:

1) Dane optymalne

Współczynnik nauki:	0,2
Współczynnik zapominania:	0,1
Liczba epok:	100000

A	-0,16156	0,161563	0,161563
B	-0,99036	0,990357	0,990357
C	-0,2549	0,254896	0,254896
D	-0,64975	0,649746	0,649746
E	-0,99085	0,990845	0,990845
F	-0,5232	0,523203	0,523203
G	-0,93699	0,936987	0,936987
H	0,027523	-0,02752	-0,02752
I	0,162929	-0,16293	-0,16293
J	0,435511	-0,43551	-0,43551

K	0,014356	-0,01436	-0,01436
L	0,586813	-0,58681	-0,58681
M	0,356031	-0,35603	-0,35603
N	0,385498	-0,3855	-0,3855
O	-0,08302	0,08302	0,08302
P	-0,07549	0,075487	0,075487
R	-0,08179	0,081787	0,081787
S	-0,93633	0,936325	0,936325
T	-0,01668	0,016675	0,016675
U	0,106483	-0,10648	-0,10648

2) Nauka bez współczynnika zapominania:

Współczynnik nauki:	0,2
Współczynnik zapominania:	-
Liczba epok:	100000

A	0,990018	-0,77737	-0,94646
B	0,971297	-0,98298	-0,7119
C	0,727806	-0,41621	-0,60532
D	0,582296	-0,95862	-0,42471
E	0,992349	-0,806	-0,9262
F	0,94775	-0,30019	-0,95261
G	0,492234	-0,75225	-0,52904
H	0,98512	-0,72557	-0,9851
I	-0,36668	-0,53844	0,372212
J	-0,22906	-0,74787	0,50026
K	-0,77342	-0,71758	-0,7308
L	-0,86995	-0,85173	-0,66469
M	-0,91289	-0,84357	0,647851
N	-0,99518	-0,85975	-0,92718
O	0,456977	-0,80661	-0,28919
P	0,984569	-0,46458	0,91799
R	0,935941	-0,53804	0,999715
S	0,519786	-0,91213	0,500363
T	0,240144	-0,03554	0,239294
U	0,428905	-0,75771	0,434308

3) Zmiana współczynnika nauki

Współczynnik nauki:	Zmienny
Współczynnik zapominania:	0,1
Liczba epok:	10000

	0,01	0,1	0,13	0,18	0,23	0,25	0,3
A	0	-0,02471	0,156633	-0,23472	0,050021	0,037606	-0,2833
B	0	-0,7562	0,905606	-0,98777	0,968261	-0,98847	-0,88992
C	0	0,107569	0,108115	-0,27744	0,138755	-0,15709	-0,73383
D	0	-0,48026	0,654579	-0,72625	0,546672	-0,67287	-0,99994
E	0	-0,81372	0,952757	-0,9985	0,998494	-0,98394	-0,64689
F	0	0,025154	0,190732	-0,42525	0,582461	-0,66061	-0,79719
G	0	-0,50398	0,725351	-0,89806	0,949592	-0,95548	-0,91262
H	0	0,179167	-0,06392	0,009554	-0,02058	-0,12666	-0,91354
I	0	0,754609	-0,551	0,243734	-0,10915	-0,01551	0,091371
J	0	0,51965	-0,35361	0,253744	-0,54018	0,565896	0,332394
K	0	0,287673	-0,1427	0,079326	0,137911	-0,32831	-0,81114
L	0	0,725146	-0,63298	0,569598	-0,7098	0,553109	-0,47802
M	0	0,373325	-0,28698	0,291737	-0,34756	0,114978	-0,8409
N	0	0,379684	-0,29909	0,31322	-0,39195	0,138904	-0,81574
O	0	0,028603	0,145153	-0,19611	-0,05499	0,052259	-0,644
P	0	0,18365	-0,02405	-0,09738	-0,00881	-0,0667	-0,28054
R	0	0,028192	0,110396	-0,18341	-0,11842	0,056814	-0,26112
S	0	-0,1747	0,479951	-0,84586	0,9624	-0,95241	-0,95681
T	0	0,614917	-0,36317	0,049132	0,185779	-0,47114	-0,39122
U	0	0,231341	-0,07551	0,04908	-0,1253	-0,1121	-0,99998

4) Zmiana współczynnika zapominania

Współczynnik nauki:	0,2
Współczynnik zapominania:	Zmienny
Liczba epok:	10000

	0,05	0,1	0,2	0,28	0,35	0,5
A	-0,01607	-0,16156	-0,07465	0,016041	-0,22735	-0,8571
B	0,958231	-0,99036	-0,93631	-0,8872	0,770087	-0,09675
C	0,22042	-0,2549	-0,09368	-0,00407	-0,14457	-0,73714
D	0,507169	-0,64975	-0,6815	-0,68206	0,59181	-0,1493
E	0,977206	-0,99085	-0,94793	-0,85661	0,700802	-0,21424
F	0,611322	-0,5232	-0,14524	0,022431	-0,24412	-0,86587
G	0,966156	-0,93699	-0,82406	-0,75312	0,633227	-0,17709
H	0,054351	0,027523	0,168649	0,177915	-0,25608	-0,75465
I	-0,16937	0,162929	0,429149	0,614279	-0,76219	-0,99731
J	-0,72031	0,435511	0,259775	0,361085	-0,47342	-0,88794
K	-0,02405	0,014356	0,298066	0,382028	-0,50571	-0,90008
L	-0,49397	0,586813	0,723176	0,723164	-0,75423	-0,95648
M	-0,26654	0,356031	0,416852	0,400478	-0,44444	-0,81714
N	-0,2328	0,385498	0,433018	0,400739	-0,44338	-0,81953
O	-0,17142	-0,08302	-0,22146	-0,25801	0,167738	-0,48613
P	-0,04888	-0,07549	0,122332	0,208052	-0,40414	-0,92217
R	-0,09089	-0,08179	0,079294	0,160409	-0,37794	-0,92343

S	0,959557	-0,93633	-0,70211	-0,46284	0,227534	-0,6416
T	0,118305	-0,01668	0,282057	0,487462	-0,64973	-0,97486
U	-0,10165	0,106483	0,020865	-0,09798	0,138474	-0,32703

Analiza wyników i wnioski:

Zacznijmy od pierwszego zestawu danych, obrazującego trzy próby przy współczynniku nauki równym 0.2 oraz współczynniku zapominania równym 0.1. Wyniki przy takich danych są stabilne. Kolor tabel obrazuje 3 najbardziej wyróżniające się grupy liter:

- grupa wokół wartości ~0,1

- grupa wokół wartości ~0,5

- grupa wokół wartości ~0,9

Ponadto widzimy, iż wyniki poszczególnych prób różnią się zazwyczaj znakiem (wartość dodatnia lub ujemna). Jest to spowodowane początkowymi wartościami wag, których zakres oscylował pomiędzy -1 a 1.

Drugi test został przeprowadzony na zestawie danych, nie posiadających współczynnika zapominania. Wynikiem braku tej wartości, była zmiana wartości wag końcowych, powodująca znaczne różnice pomiędzy wynikami końcowymi. Za wyjątkiem liter K L N oraz E, nie jesteśmy w stanie wyróżnić grup oraz wartości, wokół których oscylują rezultaty dla każdej danej testowej. Próba druga, była momentem w którym pojawiło się najwięcej wyników sprzecznych, przez które nie jesteśmy w stanie przypisać wartości do litery.

Test na trzecim zestawie danych polegał na działaniu sieci bazując na zmiennym współczynniku nauki. Pierwszym rzucającym się w oczy, jest output zawierający same zera. Stwarza to sytuację niemożliwą do interpretacji, oraz wyklucza tak niską wartość współczynnika nauki z dalszych testów. Pozostałe wartości oscylowały wokół swoich średnich, z wyjątkiem w tabeli z wartością 0.3 w która w większości posiadała rezultaty niestabilne, o mocno różniącej się wartości w porównaniu do tabel poprzednich. Po usunięciu tej tabeli jesteśmy w stanie stworzyć grupy.

Ostatnim zestawem danych był test bazujący na zmiennym współczynniku zapominania. Również na pierwszy rzut oka widzimy, iż przy dużym współczynniku zapominania wartości wszystkich liter są do siebie bardzo podobne i nie tworzą grup z wartościami poprzednich wyników. Po odjęciu tego rezultatu z wyniku ogólnego, możemy zauważyć co najmniej trzy główne grupy neuronów, których rezultaty oscylują wokół porównywalnych wyników.

Jak widać po przeanalizowaniu powyższych wyników, proces samouczenia w porównaniu z uczeniem z nauczycielem ma pewne wady. Bez obecności implementacji nauczyciela, nie

jesteśmy w stanie określić z góry jaki neuron nauczy się rozpoznawać daną grupę sygnałów, a samo uczenie ze skutkiem pozytywnym trwa znacznie dłużej. Ponadto, w sieciach tego typu bardzo dużą rolę kładzie się na wybór początkowych wartości wag neuronów. Jak już zostało wspomniane w pierwszych częściach sprawozdania, proces uczenia takiej sieci polega na pogłębianiu czy też doskonaleniu pewnych tendencji, już istniejących w sieci w momencie jej implementacji. Rezultatem takiego rozumowania jest sieć, w której koniec procesu uczenia silnie zależy od jakości jej początkowych właściwości. Pozostawiając dobór wartości wag mechanizmowi losowemu możemy spowodować, iż nasza sieć neuronowa nie będzie w stanie wystarczająco zróżnicować swoje działanie tak, aby znaleźć w swojej strukturze reprezentację dla wszystkich występujących grup. Mimo tych mankamentów, na które należy brać poprawkę, nasza sieć działała poprawnie dla odpowiednich współczynników (pierwszy zestaw danych) i bez żadnych problemów udało jej się odnaleźć podobne do siebie litery. Współczynnik nauki miał duży wpływ na zdolność uczenia sieci, gdyż przy jego małej wartości sieć zwracała jedynie zera, nie będąc w stanie wygenerować żadnych sensownych wyników. Podobnie patrząc na współczynnik zapominania – musi on być obecny w tego typu sieciach, gdyż bez niego wyniki są chaotyczne, a wagi końcowe wysokie. Biorąc również pod uwagę fakt, iż niektórych zadań nie jesteśmy w stanie wykonać przy pomocy sieci posiadającej nauczyciela - gdyż po prostu nie znamy prawidłowych rozwiązań – stosowanie sieci opartych na regule Hebba znajduje szerokie zastosowanie.

Listing kodu:

```
clear all; close all; clc;

%DEKLARACJE DANYCH WEJSCIOWYCH - TABLIC z LITERAMI
 TestLiteraA = [0 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 1;
                1 0 0 0 1;
                1 0 0 0 1];

 TestLiteraB = [1 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 0];

 TestLiteraC = [0 1 1 1 1;
                1 0 0 0 0;
                1 0 0 0 0;
                1 0 0 0 0;
                0 1 1 1 1];

 TestLiteraD = [1 1 1 1 0;
                1 0 0 0 1;
                1 0 0 0 1;
                1 0 0 0 1;
```

```
              1 1 1 1 0];

TestLiteraE = [1 1 1 1 1;
              1 0 0 0 0;
              1 1 1 1 0;
              1 0 0 0 0;
              1 1 1 1 1];

TestLiteraF = [1 1 1 1 1;
              1 0 0 0 0;
              1 1 1 1 0;
              1 0 0 0 0;
              1 0 0 0 0];

TestLiteraG = [0 1 1 1 1;
              1 0 0 0 0;
              1 0 1 1 1;
              1 0 0 0 1;
              0 1 1 1 0];

TestLiteraH = [1 0 0 0 1;
              1 0 0 0 1;
              1 1 1 1 1;
              1 0 0 0 1;
              1 0 0 0 1];

TestLiteraI = [0 1 1 1 0;
              0 0 1 0 0;
              0 0 1 0 0;
              0 0 1 0 0;
              0 1 1 1 0];

TestLiteraJ = [0 1 1 1 1;
              0 0 0 0 1;
              0 0 0 0 1;
              0 1 0 0 1;
              0 0 1 1 1];

TestLiteraK = [1 0 0 1 1;
              1 0 1 0 0;
              1 1 0 0 0;
              1 0 1 0 0;
              1 0 0 1 1];

TestLiteraL = [1 0 0 0 0;
              1 0 0 0 0;
              1 0 0 0 0;
              1 0 0 0 0;
              1 1 1 1 1];

TestLiteraM = [1 0 0 0 1;
              1 1 0 1 1;
              1 0 1 0 1;
              1 0 0 0 1;
              1 0 0 0 1];
```

```
 TestLiteraN = [1 0 0 0 1;
                1 1 0 0 1;
                1 0 1 0 1;
                1 0 0 1 1;
                1 0 0 0 1];

 TestLiteraO = [0 1 1 1 0;
                1 0 0 0 1;
                1 0 0 0 1;
                1 0 0 0 1;
                0 1 1 1 0];

 TestLiteraP = [1 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 0;
                1 0 0 0 0;
                1 0 0 0 0];

 TestLiteraR = [1 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 0;
                1 0 0 1 0;
                1 0 0 0 1];

 TestLiteraS = [0 1 1 1 1;
                1 0 0 0 0;
                0 1 1 1 0;
                0 0 0 0 1;
                1 1 1 1 0];

 TestLiteraT = [1 1 1 1 1;
                0 0 1 0 0;
                0 0 1 0 0;
                0 0 1 0 0;
                0 0 1 0 0];

 TestLiteraU = [1 0 0 0 1;
                1 0 0 0 1;
                1 0 0 0 1;
                1 0 0 0 1;
                0 1 1 1 0];

%DEKLARACJA TABLICY LITER
Litery = ["A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N"
"O" "P" "R" "S" "T" "U"];

iterator=1; %TWORZENIE SIECI
dane = zeros(5*5, 20); %ZEROWANIE MACIERZY POMOCNICZNEJ DO WYNIKOW
KONCOWYCH

for i = 1 : 5
    for j = 1 : 5
    dane(iterator, :) = [TestLiteraA(i, j) TestLiteraB(i, j)
TestLiteraC(i, j) TestLiteraD(i, j) TestLiteraE(i, j) TestLiteraF(i,
j) TestLiteraG(i, j) TestLiteraH(i, j) TestLiteraI(i, j)
TestLiteraJ(i, j) TestLiteraK(i, j) TestLiteraL(i, j) TestLiteraM(i,
```

```
j) TestLiteraN(i, j) TestLiteraO(i, j) TestLiteraP(i, j)
TestLiteraR(i, j) TestLiteraS(i, j) TestLiteraT(i, j) TestLiteraU(i,
j)];
    iterator=iterator+1;
    end
end

dane = dane'; %TRANSPONOWANIE MACIERZY
test =[1 0 0 0 1, 1 0 0 0 1, 1 0 1 0 1, 1 1 0 1 1, 1 0 0 0 1];
%MACIERZ DO TESTOWANIA KONRETNEGO ZNAKU (LITERA W)
Wagi = zainicjuj_wagi(size(dane,2)); %INICJACJA WAG LOSOWYMI
WARTOŚCIAMI Z ZAKRESU <-1, 1>
Hebb_Test(Wagi, dane, Litery); %TESTOWANIE SIECI HEBBA DLA DANYCH
POCZĄTKOWYCH
Wagi = Hebb_Trening(Wagi, dane, 0.2, 10000, 2); %TRENOWANIE
Hebb_Test(Wagi, dane, Litery); %TESTOWANIE HEBBA DLA
ZAKTUALIZOWANYCH DANYCH
wyznacz_wartosc(test, Wagi) %WYZNACZENIE WARTOSCI NA WYJSCIU DLA
TESTOWANEJ LITERY

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%FUNKCJA INICJUJACA WAGI
function [wagi] = zainicjuj_wagi(rozmiar)
    wagi = (rand(rozmiar,1)*2) - 1;
end

%FUNKCJA WYZNACZAJACA WARTOSC DLA PODANEJ LITERY TESTOWEJ
function [wyjscie] = wyznacz_wartosc(dane,wagi)
    wektor_wag = wagi'*dane';
    wyjscie = sin(wektor_wag);
end

%FUNKCJA AKTUALIZUJACA WAGI BEZ WSPOLCZYNNIKA ZAPOMINANIA
function [aktualizowane_wagi] =
aktualizuj_wagi_bez_zapominania(wagi,wartosc,dane,wspolczynnik_uczen
ia)
    aktualizowane_wagi = wagi +
(wspolczynnik_uczenia*wartosc)*dane';
end

%FUNKCJA AKTUALIZUJACA WAGI ZE WSPOLCZYNNIKIEM ZAPOMINANIA
function [aktualizowane_wagi] =
aktualizuj_wagi_z_zapominaniem(wagi,wartosc,dane,wspolczynnik_uczeni
a,wspolczynnik_zapominania)
    aktualizowane_wagi = wagi*(1-wspolczynnik_zapominania) +
(wspolczynnik_uczenia*wartosc)*dane';
end

%FUNKCJA WYPISUJACA WEKTOR WAG
function [] = wypisz_wektor_wag(wektor_wag)
    for i = 1:numel(wektor_wag)
        fprintf(' %f \n',wektor_wag(i));
    end
    fprintf('\n');
```

```
end

%FUNKCJA REALIZUJACA UCZENIE METODA HEBBA
function [wagi] = Hebb_Trening(wagi,
dane_wejsciowe,wspolczynnik_uczenia,kroki_uczenia,zapominanie)
fprintf('WAGI WEJSCIOWE:\n');
wypisz_wektor_wag(wagi);
for k = 1:kroki_uczenia
    for i = 1:size(dane_wejsciowe,1)
        wartosc = wyznacz_wartosc(dane_wejsciowe(i,:),wagi);
        if (zapominanie == 1)
        wagi =
aktualizuj_wagi_bez_zapominania(wagi,wartosc,dane_wejsciowe(i,:),wsp
olczynnik_uczenia);
        end
        if(zapominanie == 2)
        wagi =
aktualizuj_wagi_z_zapominaniem(wagi,wartosc,dane_wejsciowe(i,:),wspo
lczynnik_uczenia, 0.5);
        end
    end
end
fprintf('WAGI WYJSCIOWE:\n');
wypisz_wektor_wag(wagi);
end

%FUNKCJA REALIZUJACA TESTOWANIE SIECI UCZONEJ METODA HEBBA
function [] = Hebb_Test(wagi,dane,Litery)
    fprintf('TEST: \n');
    rozmiar = size(dane,1);
    for i = 1:rozmiar
        wartosc = (wyznacz_wartosc(dane(i,:),wagi));
        fprintf('%f \n', wartosc);
    end
end
```