Michalina Nikiel
WIMiIP, Inżynieria Obliczeniowa
rok 3, grupa laboratoryjna nr 2

Podstawy sztucznej inteligencji – projekt nr 4 – sprawozdanie

Uczenie sieci regułą Hebba.

Cel ćwiczenia:

Celem ćwiczenia jest poznanie działania reguły Hebba na przykładzie rozpoznawania emotikon.

Zrealizowane kroki:

- Wygenerowanie danych uczących i testujących, zawierających 4 różne emotikony.
- Przygotowanie sieci oraz reguły Hebba w wersji z i bez współczynnika zapominania.
- Uczenie sieci dla różnych współczynników uczenia i zapominania.
- Testowanie sieci.

Sprawozdanie zawiera kolejno:

- Zagadnienia teoretyczne: opis budowy użytej sieci i reguły Hebba.
- Zestawienie otrzymanych wyników.
- Wnioski wraz z analiza i dyskusją błędów uczenia opracowanej sieci.
- Listing całego kodu programu.

Zagadnienia teoretyczne:

Sieć neuronowa jest systemem dokonującym określonych obliczeń na zasadzie równoczesnej pracy wielu połączonych ze sobą elementów zwanych neuronami.

Sieć wielowarstwowa jest to najpopularniejszy typ sztucznych sieci neuronowych. Sieć tego typu składa się zwykle z jednej warstwy wejściowej, kilku warstw ukrytych oraz jednej warstwy wyjściowej.

Reguła Hebba polega na interpretacji zachowań aktywnych neuronów. Jeśli aktywny neuron A jest cyklicznie pobudzany przez drugi neuron B, to staje się on jeszcze bardziej czuły na pobudzenie tego neuronu. Reguła Hebba składa się z następujących punktów:

- Jeżeli połączone synapsą neurony A i B są pobudzane jednocześnie (synchronicznie) to połączenie synaptyczne między nimi jest wzmacniane.
- Jeżeli neurony A i B połączone synapsą nie są pobudzane jednocześnie (asynchronicznie) to połączenie pomiędzy nimi jest osłabiane.

Reguła Hebba jest jedną z najpopularniejszych metod samouczenia sieci neuronowych. Polega ona na tym, że sieci pokazuje się kolejne przykłady sygnałów wejściowych nie podając żadnych informacji o tym, co z tymi sygnałami należy zrobić. Sieć obserwuje otoczenie i odbiera różne sygnały nie zostają określone jednak jakie znaczenie mają pokazujące się obiekty i jakie są pomiędzy nimi zależności. Sieć na podstawie obserwacji występujących sygnałów stopniowo sama odkrywa, jakie jest ich znaczenie i również sama ustala zachodzące między sygnałami zależności. Po podaniu do sieci neuronowej każdego kolejnego zestawu sygnałów wejściowych tworzy się w tej sieci pewien rozkład sygnałów wyjściowych - niektóre neurony sieci są pobudzone bardzo silnie inne słabiej, a jeszcze inne mają sygnały wyjściowe wręcz ujemne. Proces samouczenia ma niestety wady. W porównaniu z procesem uczenia z nauczycielem samouczenie jest zwykle znacznie powolniejsze. Co więcej bez nauczyciela nie można z góry określić, który neuron wyspecjalizuje się w rozpoznawania której klasy sygnałów. Stanowi to pewną trudność przy wykorzystywaniu i interpretacji wyników pracy sieci. Co więcej - nie można określić, czy sieć uczona w ten sposób nauczy się wszystkich prezentowanych jej wzorców. Dlatego sieć przeznaczona do samouczenia musi być większa niż sieć wykonująca to samo zadanie, ale trenowana w sposób klasyczny z udziałem nauczyciela.

Metoda Hebba posiada wiele wad, są nimi m. in.:

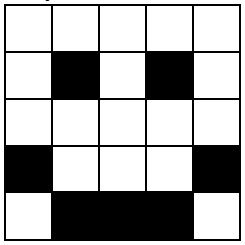
- niska efektywność uczenia,
- przemnożony wpływ początkowych wartości wag sieci,
- wagi w przypadku tej reguły mogą rosnąć bez ograniczeń,
- proces uczenia sieci nigdy nie zakończy się samodzielnie,
- nie ma pewności, że jednej klasie wzorców będzie odpowiadał jeden neuron,

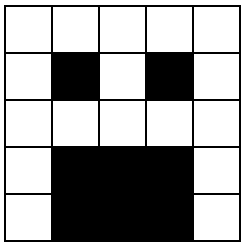
- nie ma również gwarancji, że wszystkie klasy wzorców będą miały oddzielne reprezentacje w postaci oddzielnych zbiorów aktywnych neuronów.

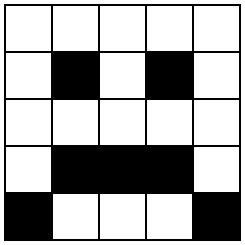
Bardzo istotną kwestią jest wybór początkowych wartości wag neuronów sieci przeznaczonej do samouczenia. Wartości te mają bardzo silny wpływ na ostateczne zachowanie sieci, ponieważ proces uczenia jedynie pogłębia i doskonali pewne tendencje istniejące w sieci od samego początku. Od jakości początkowych właściwości sieci silnie zależy do czego sieć dojdzie na końcu procesu uczenia.

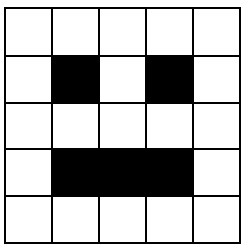
Zestawienie otrzymanych wyników w postaci ekranu i opisu:

- Przebieg wykonanego ćwiczenia:
  - ✓ Wygenerowanie danych uczących dla 4 wybranych emotikon.
  - ✓ Przyjęta zostaje macierz wielkości 5x5, gdzie pole czarne określa się jako 1, natomiast białe jako 0.
  - ✓ Wybrane zostały następujące emotikony:



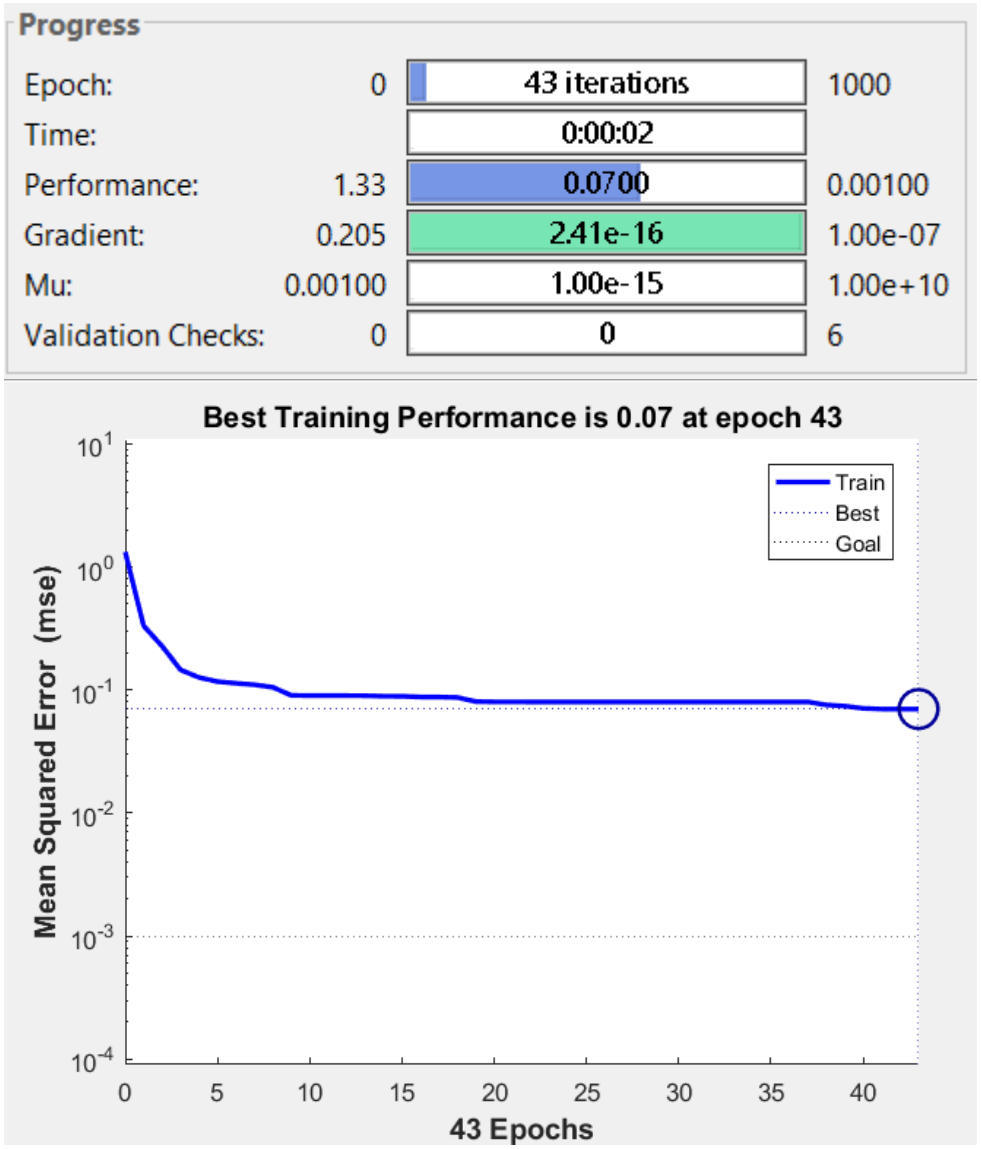


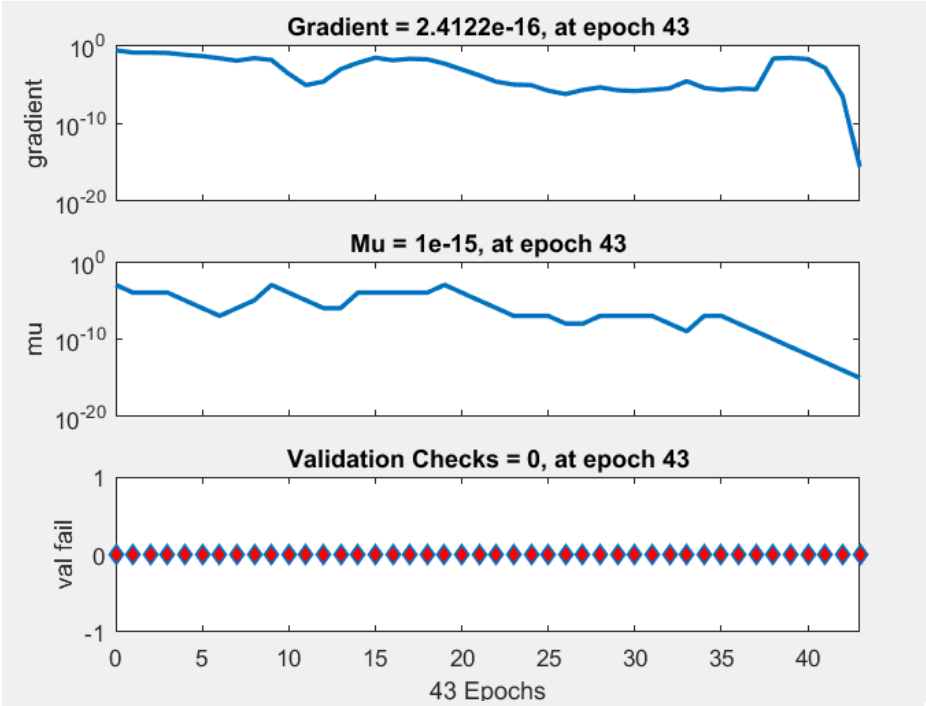


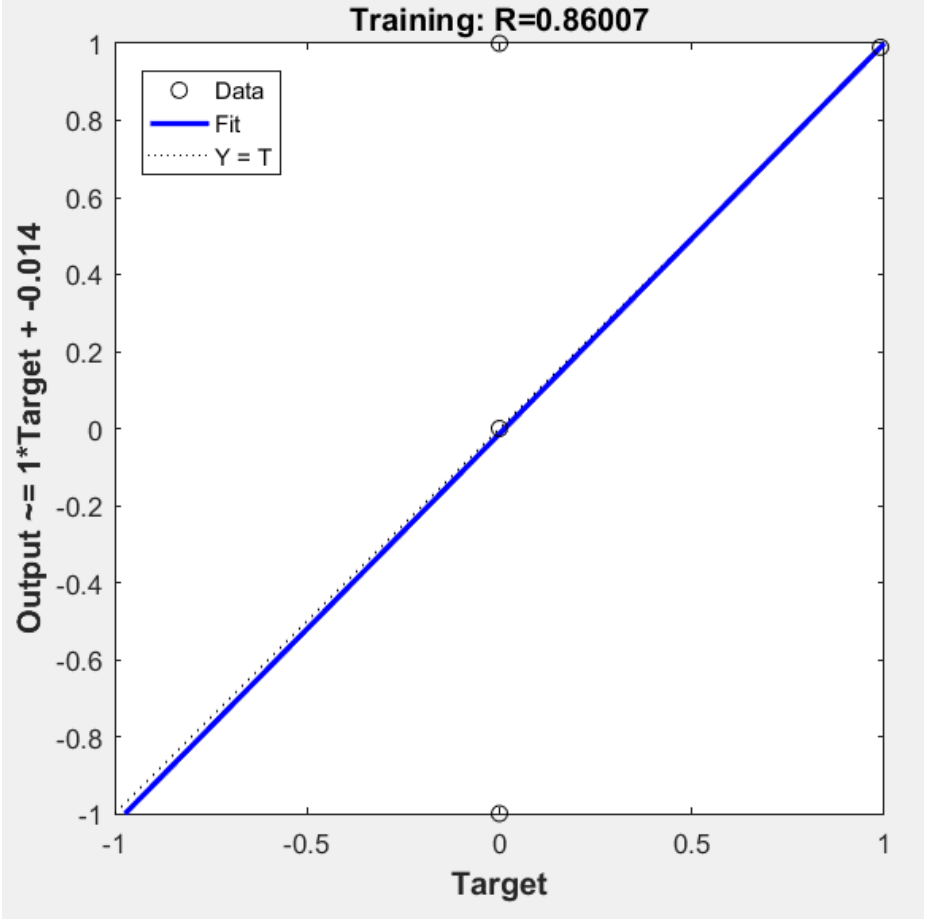


- ✓ Wygenerowanie danych wejściowych.
- ✓ Użycie funkcji „newff” z wykorzystaniem różnych parametrów uczenia oraz uczenie sieci zgodnie z zasadą Hebba.

- Zrzuty ekranu:
  - ✓ Parametry algorytmu Hebba: współczynnik zapomnienia: 0.0, współczynnik uczenia: 0.99; Parametry treningu sieci: maksymalna ilość epok: 1000, cel wydajności sieci: 0.001, wskaźnik uczenia sieci: 0.5.




Gradient = 2.4122e-16, at epoch 43
gradient
Mu = 1e-15, at epoch 43
mu
Validation Checks = 0, at epoch 43
val fail
43 Epochs


Training: R=0.86007
Data
Fit
Y = T
Output ~= 1*Target + -0.014
Target

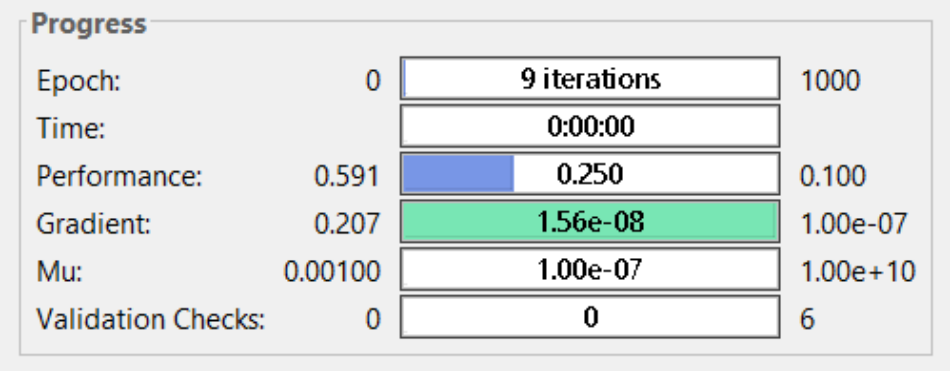
Wynik funkcji efekt oraz efekt_1:

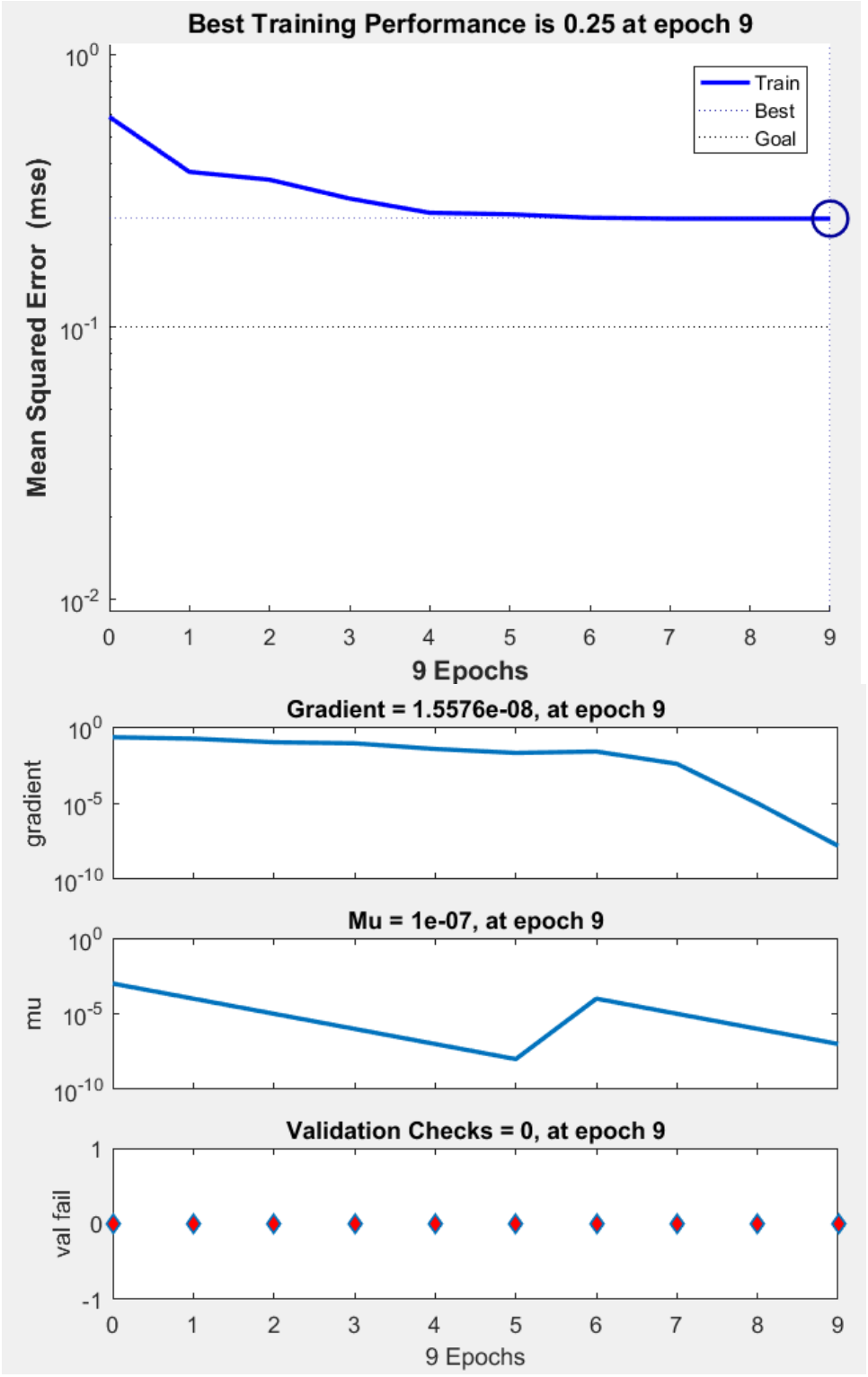
```
efekt = wagiHebba;
efekt_1 = sim(net, a_testowe);
```

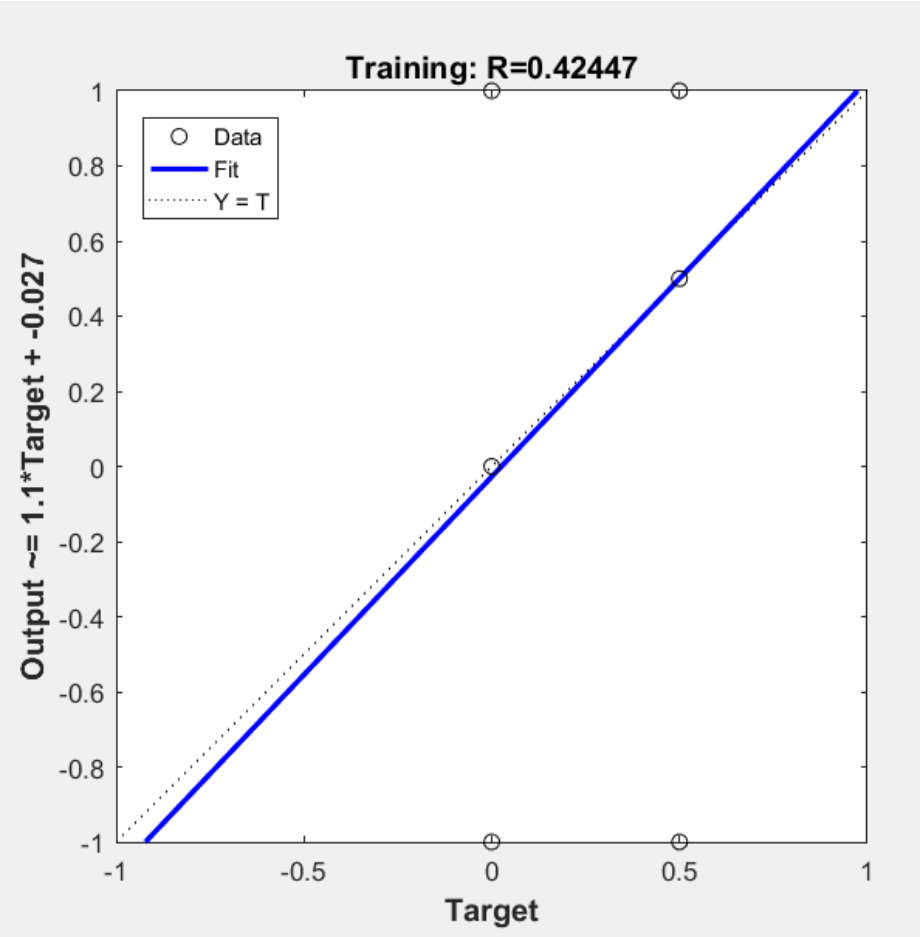
3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0.9900	0	0.9900	0	0	0	0	0	0	0.9900
0	0	0	0	0.9900	0	0.9900	0	0	0	0	0	0	0
0	0	0	0	0.9900	0	0.9900	0	0	0	0	0	0	0
0	0	0	0	0.9900	0	0.9900	0	0	0	0	0	0	0

1	-8.8818e-16
2	3.5527e-15
3	0
4	0
5	0
6	-2.2204e-16
7	0.9900
8	0
9	0.9900
10	2.2204e-16
11	0
12	0

✓ Parametry algorytmu Hebba: współczynnik zapomnienia: 0.5, współczynnik uczenia: 0.5; Parametry treningu sieci: maksymalna ilość epok: 1000, cel wydajności sieci: 0.1, wskaźnik uczenia sieci: 0.1.




Best Training Performance is 0.25 at epoch 9
Mean Squared Error (mse)
Train
Best
Goal
9 Epochs
Gradient = 1.5576e-08, at epoch 9
gradient
Mu = 1e-07, at epoch 9
mu
Validation Checks = 0, at epoch 9
val fail
9 Epochs



Wynik funkcji efekt oraz efekt_1:

```
efekt = wagiHebba;
efekt_1 = sim(net, a_testowe);
```

3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0.5000	0	0.5000	0	0	0	0	0
0	0	0	0	0.5000	0	0.5000	0	0	0	0	0
0	0	0	0	0.5000	0	0.5000	0	0	0	0	0
0	0	0	0	0.5000	0	0.5000	0	0	0	0	0

1	-3.4861e-14
2	2.4009e-07
3	-2.2204e-16
4	-1
5	2.1316e-14
6	2.2204e-16
7	0.5000
8	-2.2204e-16
9	0.5000
10	1.0000
11	3.5527e-15
12	4.4409e-16

Wnioski:

- Udało się zrealizować projekt.
- Algorytm poprawnie wykonuje swoje zadanie, trenując odpowiednie dane.
- Najwięcej trudności sprawiło utworzenie odpowiednich danych wejściowych tak aby utworzyły emotikony.
- Ustawienie odpowiednich parametrów kluczowo wpływa na wynik programu. Wyniki są bardzo różne w zależności od różnych parametrów.
- Lepsze rezultaty okazały się w drugim przypadku, program był szybszy i wykonał się wykorzystując mniejszą liczbę epok.
- Prawidłowy efekt uzyskano za pomocą stosowania reguły Hebba oraz ograniczenia dla tej metody, czyli współczynnika zapominania.
- Współczynnik zapominania jest bardzo istotny, gdyż pozwala odpowiednio zmieniać wagi, tak aby nie osiągały zbyt dużych wartości.
- Dobranie nieodpowiedniego współczynnika uczenia sieci może zwiększać ryzyko występowania błędów podczas treningu sieci (np. dane mogą wyjść poza swój zakres).

Listing kodu głównego i funkcji Rastrigin 3D:

```
close all; clear all; clc;

%wejscia do sieci oraz minimalne oraz maksymalne wartosci
wejsc
%(25 par 0&1 - osobno dla kazdej z danych uczacych)

start=[0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;
     0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;
     0 1; 0 1; 0 1; 0 1; 0 1;];

%ilosc wyjsc z sieci (jedna warstwa - 25 neutronow na
wyjsciu)
wyjscia_s = 25;

%uzycie funkcji newff
net = newff(start, wyjscia_s, {'tansig'}, 'trainlm',
'learnh');
```

```
 %kolumnowa reprezentacja binarna 4 emotikonow dla tablicy
8x4
            %:):O:(:|
WEJSCIE =  [ 0 0 0 0
             0 0 0 0
             0 0 0 0
             0 0 0 0
             0 0 0 0
             0 0 0 0
             1 1 1 1
             0 0 0 0
             1 1 1 1
             0 0 0 0
             0 0 0 0
             0 0 0 0
             0 0 0 0
             0 0 0 0
             0 0 0 0
             1 0 0 0
             0 1 1 1
             0 1 1 1
             0 1 1 1
             1 0 0 0
             0 0 1 0
             1 1 0 0
             1 1 0 0
             1 1 0 0
             0 0 1 0
             ];
%zmienna, ktora reprezentuje, czy uzytkownik "trafil" w
wybrana przez
%siebie emotikone - 1 oznacza trafienie, 0 - chybienie
WYJSCIE = [1 0 0 0 ;  %:)
           0 1 0 0 ;  %:O
           0 0 1 0 ;  %:(
           0 0 0 1 ]; %:|


%PARAMETRY ALGORYTMU HEBBA
% * wspolczynnik zapominania
lp.dr = 0.0;
% * wspolczynnik uczenia
lp.lr = 0.99;

%dostosowanie parametrów sieci do metody Hebba
wagiHebba = learnh([], WEJSCIE, [], [], WYJSCIE, [], [],
[], [], [], lp, []);
```

```
%PARAMETRY TRENINGU SIECI:
% * maksymalna ilosc epok
net.trainParam.epochs = 1000;
% * cel wydajnosci sieci
net.trainParam.goal = 0.001;
% * wskaznik uczenia sieci
net.trainParam.lr=0.5;
whebb=wagiHebba';
net = train(net, WEJSCIE, whebb);

%dane testowe
a_testowe= [0;0;0;0;0;
            0;1;0;1;0;
            0;0;0;0;0;
            1;0;0;0;1;
            0;1;1;1;0]; %:)


b_testowe=[0;0;0;0;0;
           0;1;0;1;0;
           0;0;0;0;0;
           0;1;1;1;0;
           0;1;1;1;0;]; %:O



c_testowe=[0;0;0;0;0;
           0;1;0;1;0;
           0;0;0;0;0;
           0;1;1;1;0;
           1;0;0;0;1;]; %:(

d_testowe=[0;0;0;0;0;
           0;1;0;1;0;
           0;0;0;0;0;
           0;1;1;1;0;
           0;0;0;0;0;]; %:|

efekt = wagiHebba;
%symulacja sieci net
efekt_1 = sim(net, a_testowe);

%wypisywanie wartosci reguly Hebba, wypisywanie kolejnych
wierszy
disp('Jednokrotne wykorzystanie reguly Hebba: ')
disp(':) = '), disp(sum(efekt(1, ':')));
```

```
disp(':O = '), disp(sum(efekt(2, ':')));
disp(':( = '), disp(sum(efekt(3, ':')));
disp(':| = '), disp(sum(efekt(4, ':')));


%wypisywanie wartosci
disp('Dzialanie algorytmu z wykorzystaniem r. Hebba dla
emotikon: ')
disp(':) = '), disp(efekt_1(1));
disp(':O = '), disp(efekt_1(2));
disp(':( = '), disp(efekt_1(3));
disp(':| = '), disp(efekt_1(4));
```