Podstawy sztucznej inteligencji

Kinga Synowiec, Inżynieria obliczeniowa, grupa 2

Sprawozdanie nr 4

- **Temat ćwiczenia:**
  Uczenie sieci regułą Hebba.

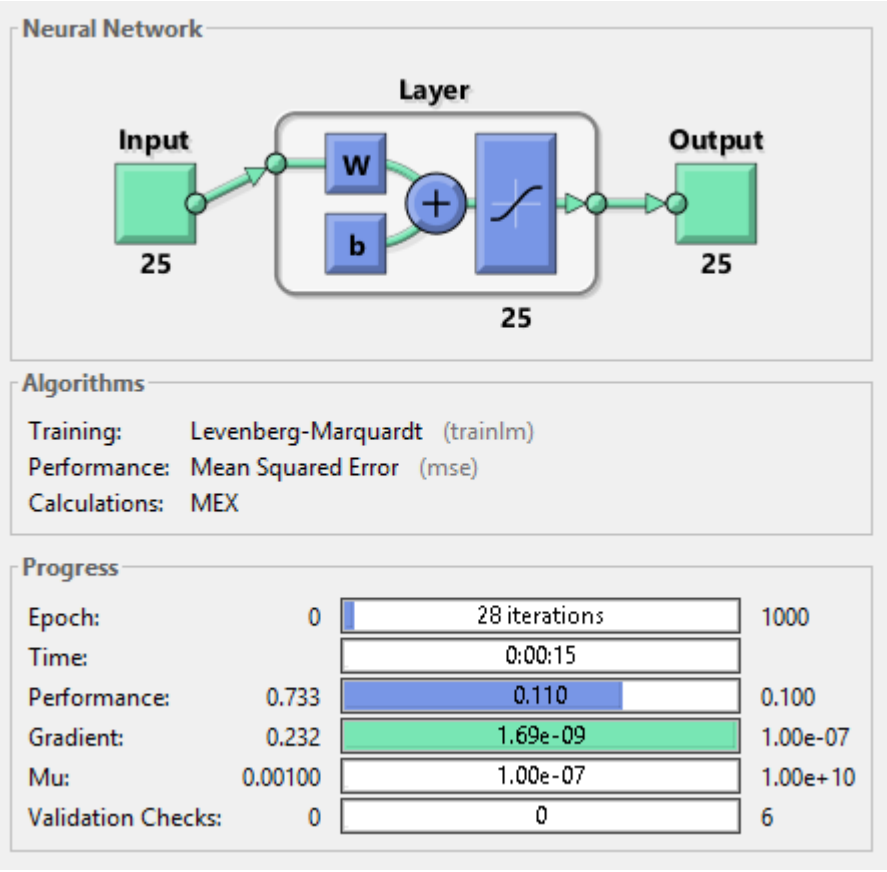
- **Cel ćwiczenia:**
  Poznanie działania reguły Hebba na przykładzie rozpoznawania emotikon (użyte emotikony : ☺ :O ☹ 😐)

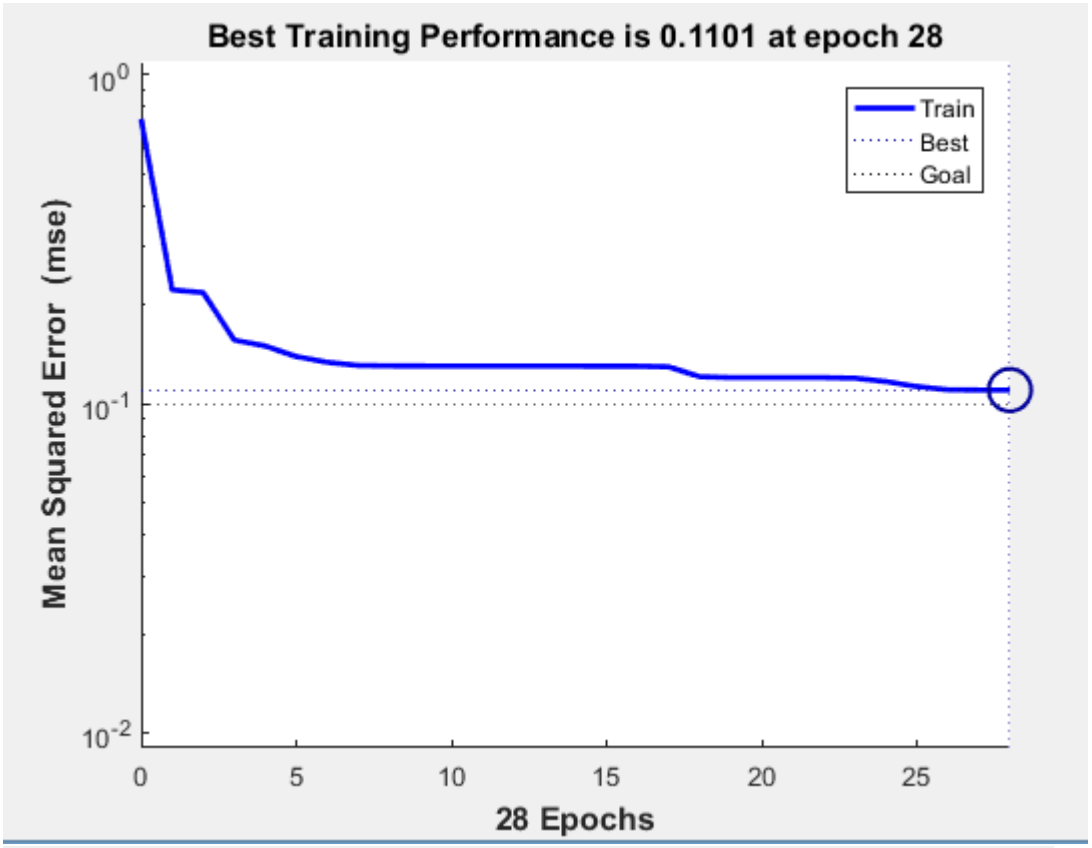
- **Wstęp teoretyczny:**
  - *Sieć neuronowa (sztuczna sieć neuronowa)* – ogólna nazwa struktur matematycznych i ich programowych lub sprzętowych modeli, realizujących obliczenia lub przetwarzanie sygnałów poprzez rzędy elementów, zwanych sztucznymi neuronami, wykonujących pewną podstawową operację na swoim wejściu. Oryginalną inspiracją takiej struktury była budowa naturalnych neuronów, łączących je synaps, oraz układów nerwowych, w szczególności mózgu.
  - *Reguła Hebba* - Jest to jedna 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, nikt nie określa 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. Interpretacja tych zachowań może być taka, że 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". Po ustaleniu się sygnałów wyjściowych wszystkich neuronów w całej sieci - wszystkie wagi wszystkich neuronów są zmieniane, przy czym wielkość odpowiedniej zmiany wyznaczana jest na podstawie iloczynu sygnału wejściowego, wchodzącego na dane wejście (to którego wagę zmieniamy) i sygnału wyjściowego produkowanego przez neuron, w którym modyfikujemy wagi. Łatwo zauważyć, że jest to właśnie realizacja postulatu Hebba - w efekcie opisanego wyżej algorytmu połączenia między źródłami silnych sygnałów i neuronami które na nie silnie reagują są wzmacniane.
  Dokładniejsza analiza procesu samouczenia metodą Hebba pozwala stwierdzić, że w wyniku konsekwentnego stosowania opisanego algorytmu początkowe, najczęściej przypadkowe „preferencje" neuronów ulegają systematycznemu wzmacnianiu i dokładnej polaryzacji. Jeśli

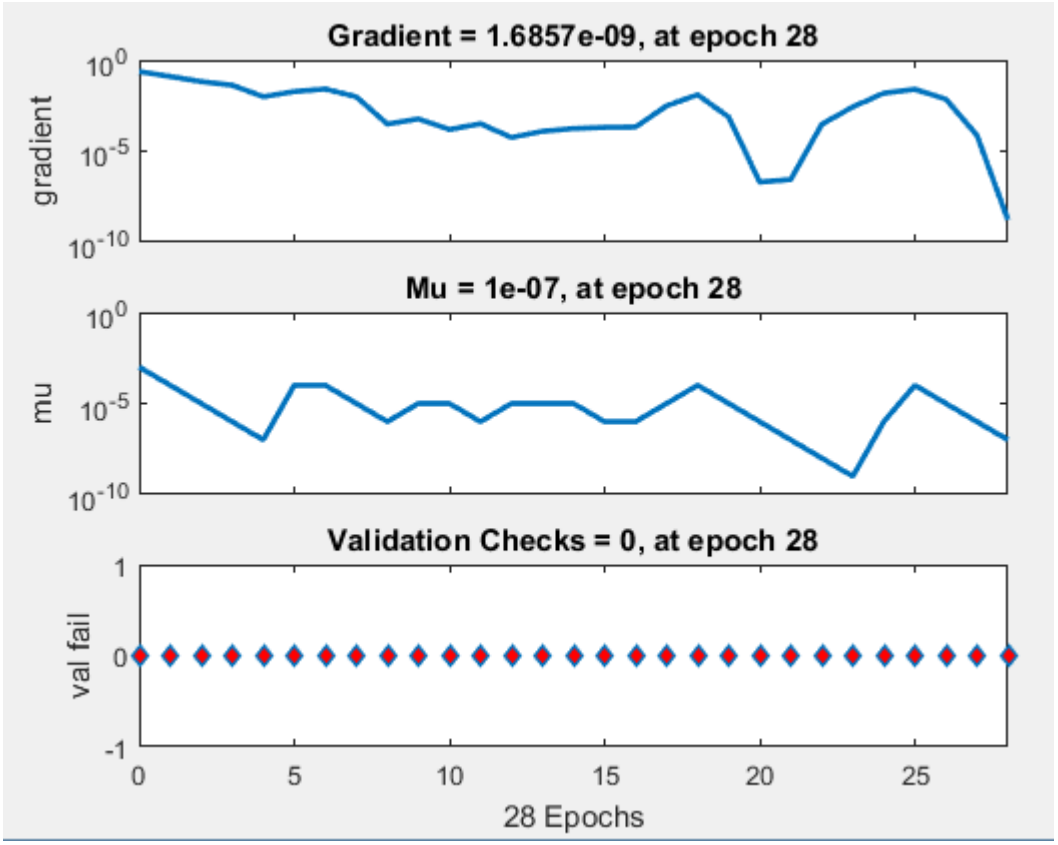
jakiś neuron miał „wrodzoną skłonność” do akceptowania sygnałów pewnego rodzaju - to w miarę kolejnych pokazów nauczy się te sygnały rozpoznawać coraz dokładniej i coraz bardziej precyzyjnie. Po dłuższym czasie takiego samouczenia w sieci powstaną zatem wzorce poszczególnych typów występujących na wejściu sieci sygnałów. W wyniku tego procesu sygnały podobne do siebie będą w miarę postępu uczenia coraz skuteczniej grupowane i rozpoznawane przez pewne neurony, zaś inne typy sygnałów staną się „obiektem zainteresowania” innych neuronów W wyniku tego procesu samouczenia sieć nauczy się, ile klas podobnych do siebie sygnałów pojawia się na jej wejściach oraz sama przyporządkuje tym klasom sygnałów neurony, które nauczą się je rozróżniać, rozpoznawać i sygnalizować.

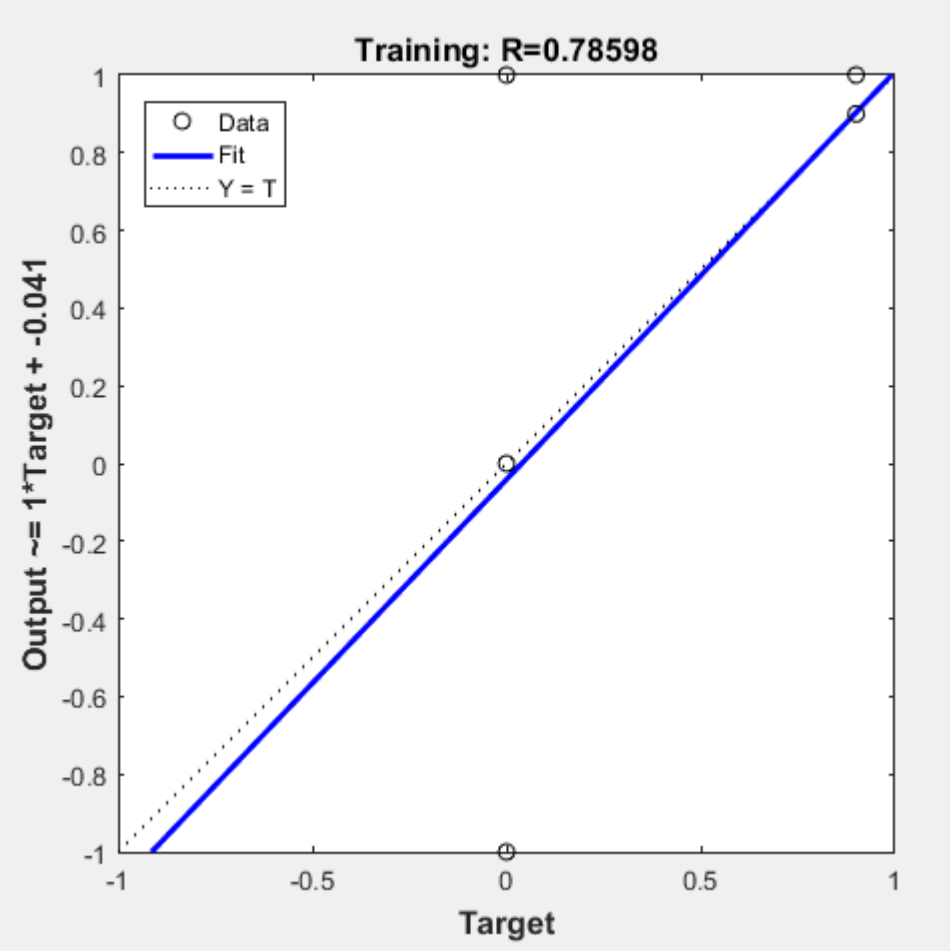
//źródło : http://www.poltynk.pl/marcin/reguly.html

- **Zrzuty ekranu przedstawiające przykładowe rezultaty po uruchomieniu programu:**
  - parametry algorytmu Hebba : współczynnik zapomnienia = 0.0, współczynnik uczenia = 0.9; parametry treningu sieci : maksymalna ilość epok = 1000, cel wydajności sieci = 0.1, wskaźnik uczenia sieci = 0.2




Best Training Performance is 0.1101 at epoch 28
Mean Squared Error (mse)
Train
Best
Goal
28 Epochs


Gradient = 1.6857e-09, at epoch 28
gradient
Mu = 1e-07, at epoch 28
mu
Validation Checks = 0, at epoch 28
val fail
28 Epochs

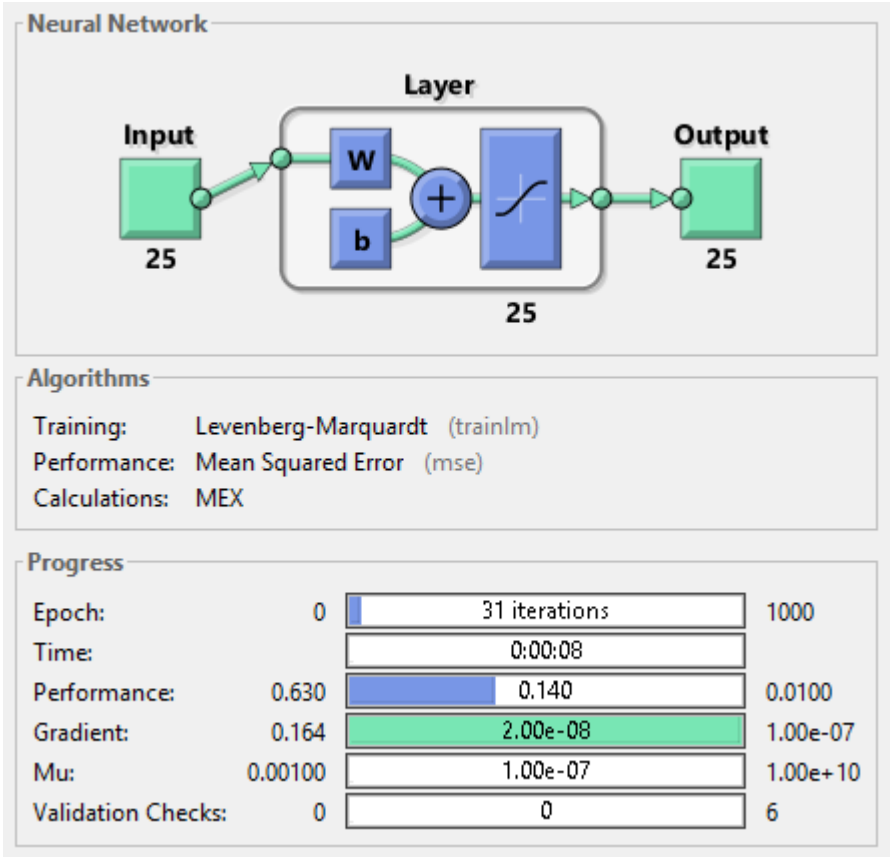


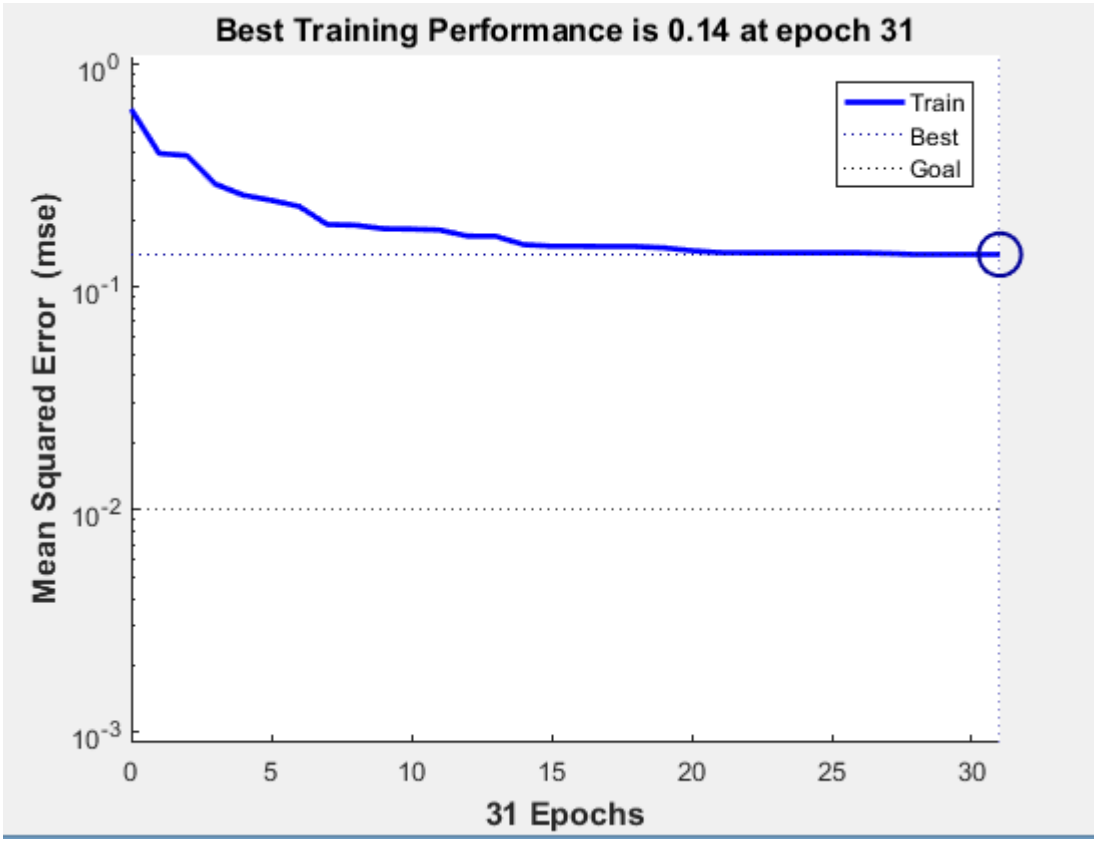
Wyniki funkcji efekt i efekt_1:

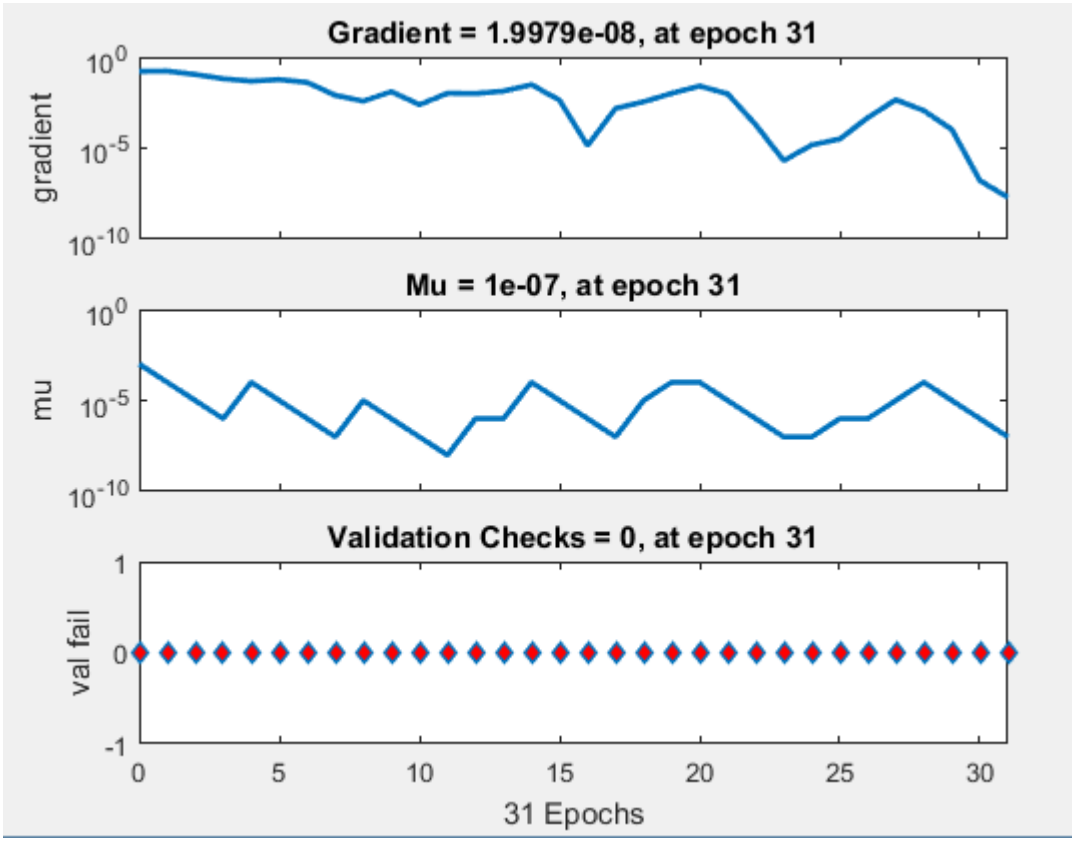
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0.9000	0	0.9000	0	0	0	0	0	0	0.9000
2	0	0	0	0	0	0	0.9000	0	0.9000	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0.9000	0	0.9000	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0.9000	0	0.9000	0	0	0	0	0	0	0

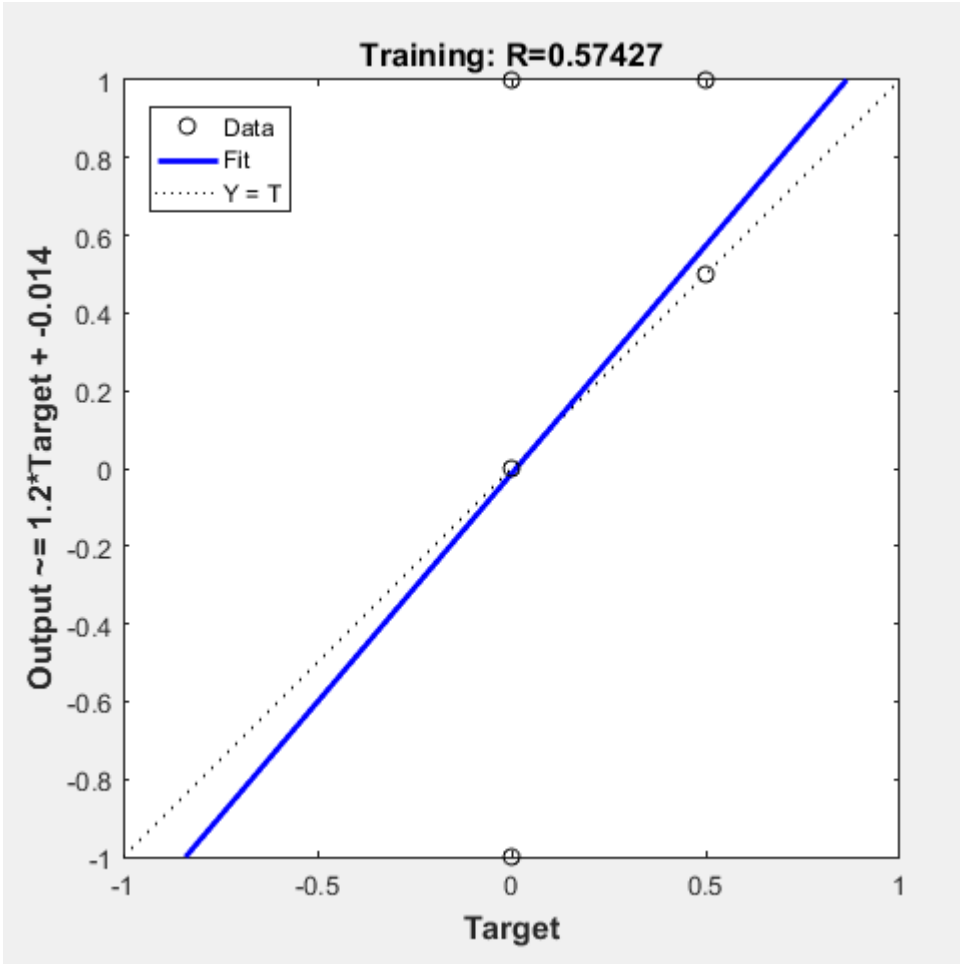
	1
1	-7.1054e-15
2	2.2204e-16
3	0
4	-5.3291e-15
5	-2.2204e-16
6	2.2204e-16
7	0.9000
8	-2.2204e-16
9	0.9000
10	-1.0000
11	2.2204e-16
12	0
13	0
14	1.0000
15	0
16	0.9000
17	-4.4409e-16
18	-4.4409e-16
19	1
20	0.9000

- parametry algorytmu Hebba : współczynnik zapomnienia = 0.5, współczynnik uczenia = 0.5; parametry treningu sieci : maksymalna ilość epok = 1000, cel wydajności sieci = 0.01, wskaźnik uczenia sieci = 0.5






Gradient = 1.9979e-08, at epoch 31
gradient
Mu = 1e-07, at epoch 31
mu
Validation Checks = 0, at epoch 31
val fail
31 Epochs


Training: R=0.57427
Data
Fit
Y = T
Output ~= 1.2*Target + -0.014
Target

Wyniki funkcji efekt i efekt_1:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0.5000	0	0.5000	0	0	0	0	0	0	0.5000
2	0	0	0	0	0	0	0.5000	0	0.5000	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0.5000	0	0.5000	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0.5000	0	0.5000	0	0	0	0	0	0	0

	1
1	4.4409e-16
2	0
3	-1
4	1
5	-1
6	-3.8192e-14
7	0.5000
8	2.2204e-16
9	0.5000
10	0
11	-2.2204e-16
12	-4.9249e-13
13	2.2204e-16
14	-1.0000
15	2.2204e-16
16	0.5000
17	-1.8490e-10
18	2.2204e-16
19	0
20	0.5000

- **Wnioski:**

  - Reguła Hebba jest metodą uczenia zawierającą wiele ograniczeń.
  - Wyższy współczynnik uczenia wpływał na wzrost wartości wag.
  - Podczas wykonywania programu istotne jest, aby użyć odpowiednich parametrów, gdyż w zależności od nich wyniki znacząco się różnią.
  - Efekt działania programu jest lepszy dzięki ograniczeniu metody Hebba poprzez zastosowanie współczynnika zapomnienia.
  - Współczynnik zapomnienia swoją rolę realizuje poprzez zmianę wag w taki sposób, aby nie osiągały zbyt dużych wartości. Poprawia zdecydowanie stabilność uczenia.
  - Współczynnik ten powinien być stosunkowo małą wartością, gdyż wówczas neuron zachowuje informacje, których się nauczył. W przypadku wartości większych neuron zapomina rzeczy, których poprzednio się nauczył.
  - Gdyby współczynnik uczenia został błędnie dobrany, wówczas dane mogłyby wyjść poza swój zakres lub mógłby wystąpić jakiś inny błąd.
  - Współczynnik uczenia wpływa jednak na skuteczność procesu uczenia. Im większy owy parametr, tym szybciej następuje przyrost wag, co z kolei wpływa na tempo zachodzenia całego procesu uczenia.

- **Listing całego kodu programu wykonanego w MATLABie:**

```
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));
```