Marcin Damek
Inżynieria obliczeniowa, gr 1
Nr. albumu 285952

Sprawozdanie numer 4

Uczenie sieci regułą Hebba

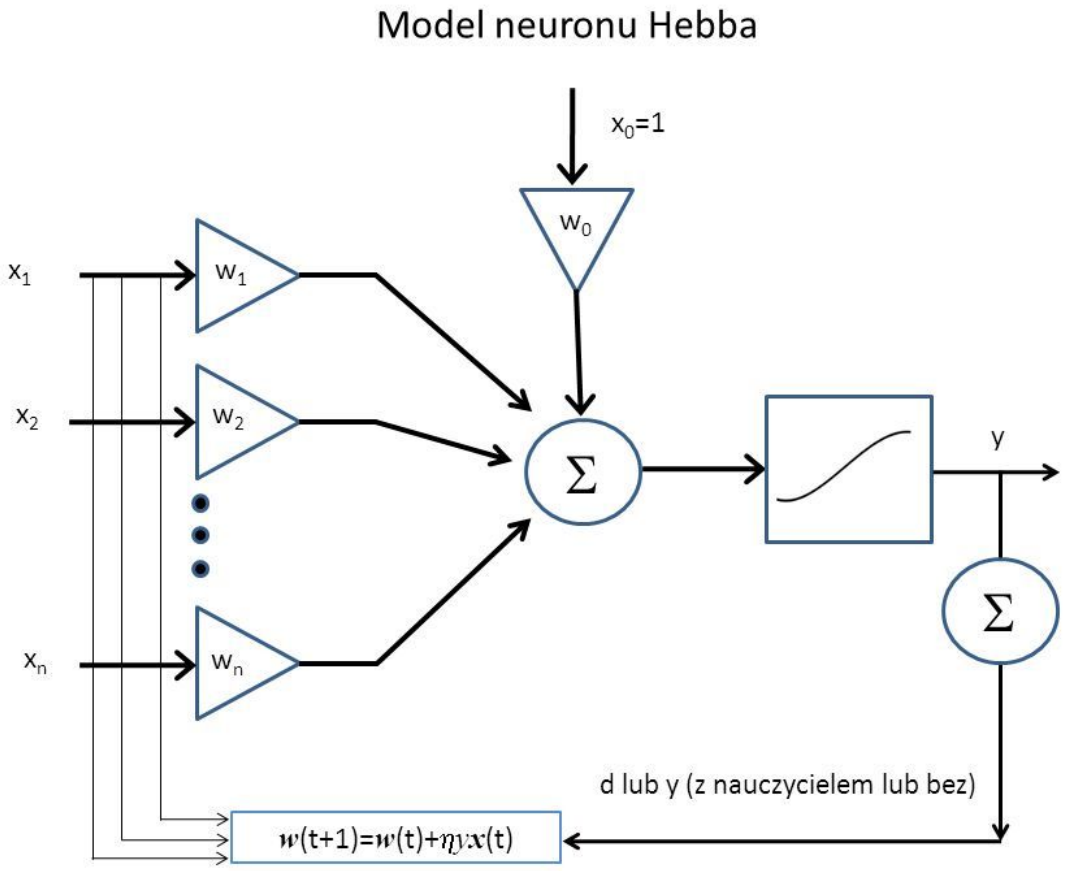
Celem ćwiczenia jest poznanie działania reguły Hebba na przykładzie rozpoznawania emotikon.

Część teoretyczna

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.

$$w_{ij}^{new} = w_{ij}^{old} + \alpha\, f_i(a_{iq}) g_j(p_{jq})$$

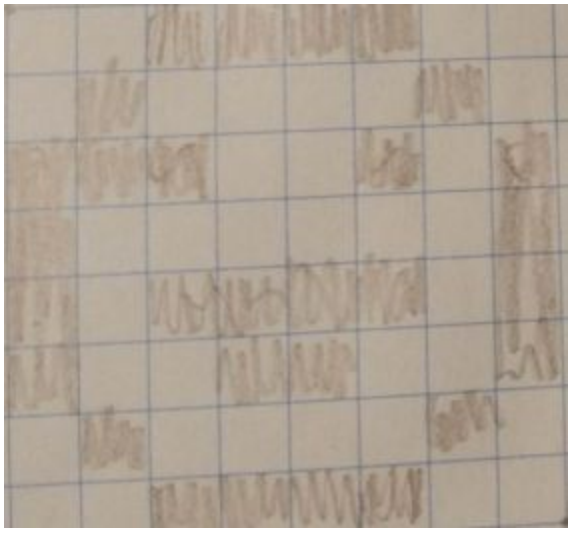
Presynaptic Signal

Postsynaptic Signal

Część praktyczna

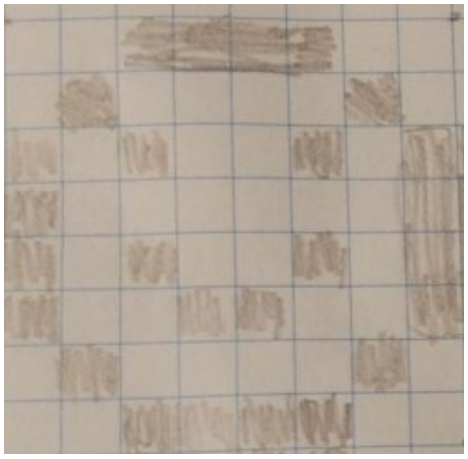
Do wykonania zadania użyłem pakietu MATLAB wraz z narzędziem Neural Network Training Tool. Dzięki niemu w stosunkowo łatwy sposób możemy tworzyć i uczyć sieć oraz w intuicyjny sposób analizować wyniki. Do nauki wygenerowałem 4 emotikony zbudowane z 0 oraz 1 na planie kwadratu 8 x 8. Białe pola zostały oznaczone jako 0 natomiast czarne jako 1. Po lewej graficzna reprezentacja emotikony a po prawej po przekształceniu w macierz. Emotikony wykorzystane do testów wyglądają następująco:

Duży uśmiech



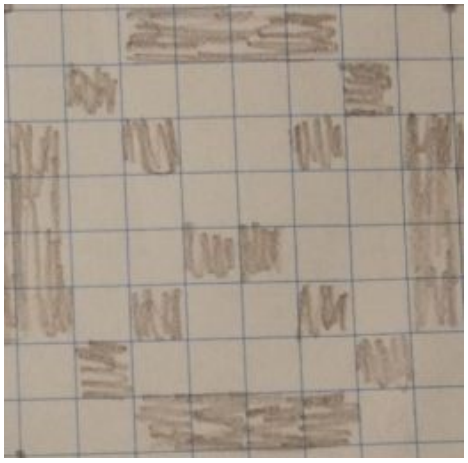
```
0 0 1 1 1 1 0 0,
0 1 0 0 0 0 1 0,
1 0 1 0 0 1 0 1,
1 0 0 0 0 0 0 1,
1 0 0 1 1 0 0 1,
1 0 1 0 0 1 0 1,
0 1 0 0 0 0 1 0,
0 0 1 1 1 1 0 0,
```

Uśmiech



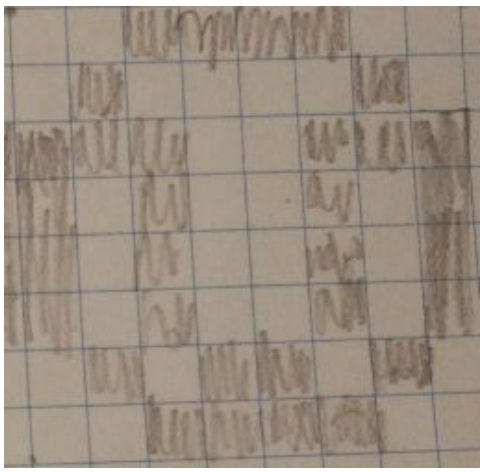
```
0 0 1 1 1 1 0 0,
0 1 0 0 0 0 1 0,
1 0 1 0 0 1 0 1,
1 0 0 0 0 0 0 1,
1 0 1 0 0 1 0 1,
1 0 0 1 1 0 0 1,
0 1 0 0 0 0 1 0,
0 0 1 1 1 1 0 0,
```

Smutek



```
0 0 1 1 1 1 0 0
0 1 0 0 0 0 1 0
1 0 1 0 0 1 0 1
1 0 0 0 0 0 0 1
1 0 1 1 1 1 0 1
1 0 0 1 1 0 0 1
0 1 0 0 0 0 1 0
0 0 1 1 1 1 0 0
```

Płacz



```
0 0 1 1 1 1 0 0,
0 1 0 0 0 0 1 0,
1 1 1 0 0 1 1 1,
1 0 1 0 0 1 0 1,
1 0 1 0 0 1 0 1,
1 0 1 0 0 1 0 1,
0 1 0 1 1 0 1 0,
0 0 1 1 1 1 0 0,
```

Wszystkie emotikony zapisałem w macierzy, tak aby każdy kolejny wiersz mający 8 elementów z emotikony był zapisany w pionie, wszystkie wiersze po kolei pionowo. Emotikony jako macierz wejściowa zapisałem do zmiennej która nazwałem wejscie. Macierz ma wymiar 64 x 4, ponieważ w pionie są umieszczone wszystkie punkty emotikony i używam 4 emotikon. W zmiennej wyjście zapisałem macierz 4 x 4 która w każdym wierszu przechowuje 1 dla wersji, że emotikona została wybrana oraz 0 dla wersji przeciwnej.

Do wykonania zadania użyłem gotowych funkcji z pakietu Matlab.
Dzięki funkcji `newff(P,T,S)` utworzyłem sieć neuronową gdzie

```
% pierwszy parametr - zakresy kolejnych wejść
% drugi parametr - ilość neuronów w każdej kolejnej warstwie
% trzeci parametr - funkcje przynależności, np. hardlim, tansig, logsig, purelin, satlin
```

learnh(W,P,Z,N,A,T,E,gW,gA,D,LP,LS) - dzięki tej funkcji mogłem wykorzystać regułę Hebba. W miejsce P wstawiłem macierz wejściową, w miejsce A macierz wyjsciową a w miejsce LP pozostałe parametry pozwalające sterować całą siecią.

Wyniki

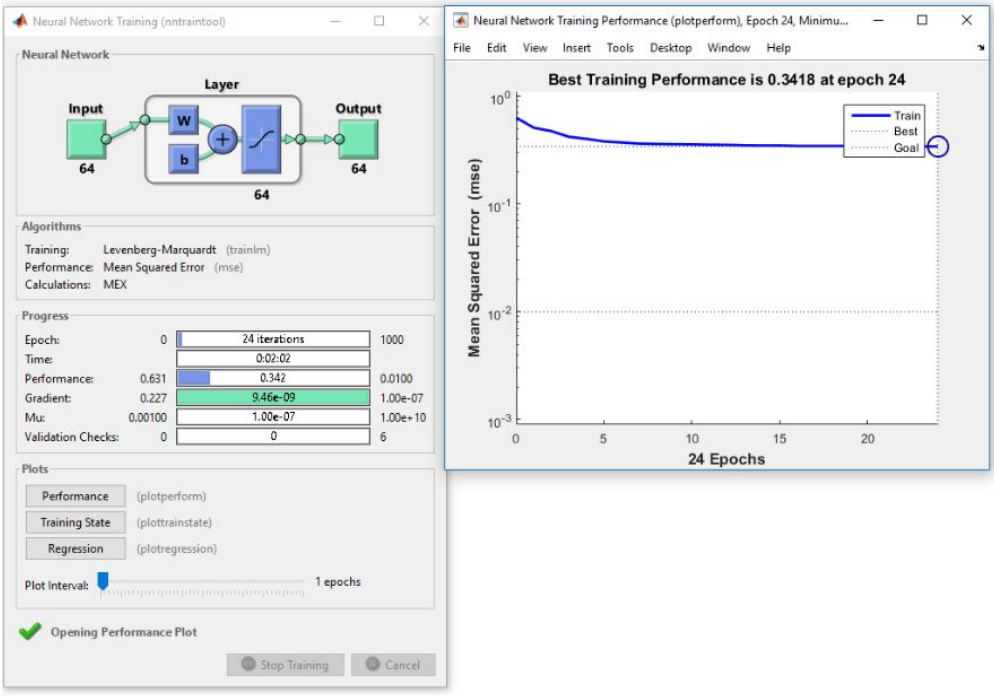
	uśmiech	
0/0.01	0/0.1	0/0.7
-1	-1	0.1464
0.1/0.01	0.1/0.1	0.1/0.7
-1	0.9205	0.005
0.7/0.01	0.7/0.1	0.7/0.7
-1	0.9969	-1

	płacz	
0/0.01	0/0.1	0/0.7
-1	-1	-0.2952
0.1/0.01	0.1/0.1	0.1/0.7
-1	0.6582	0.7924
0.7/0.01	0.7/0.1	0.7/0.7
-1	1	-1

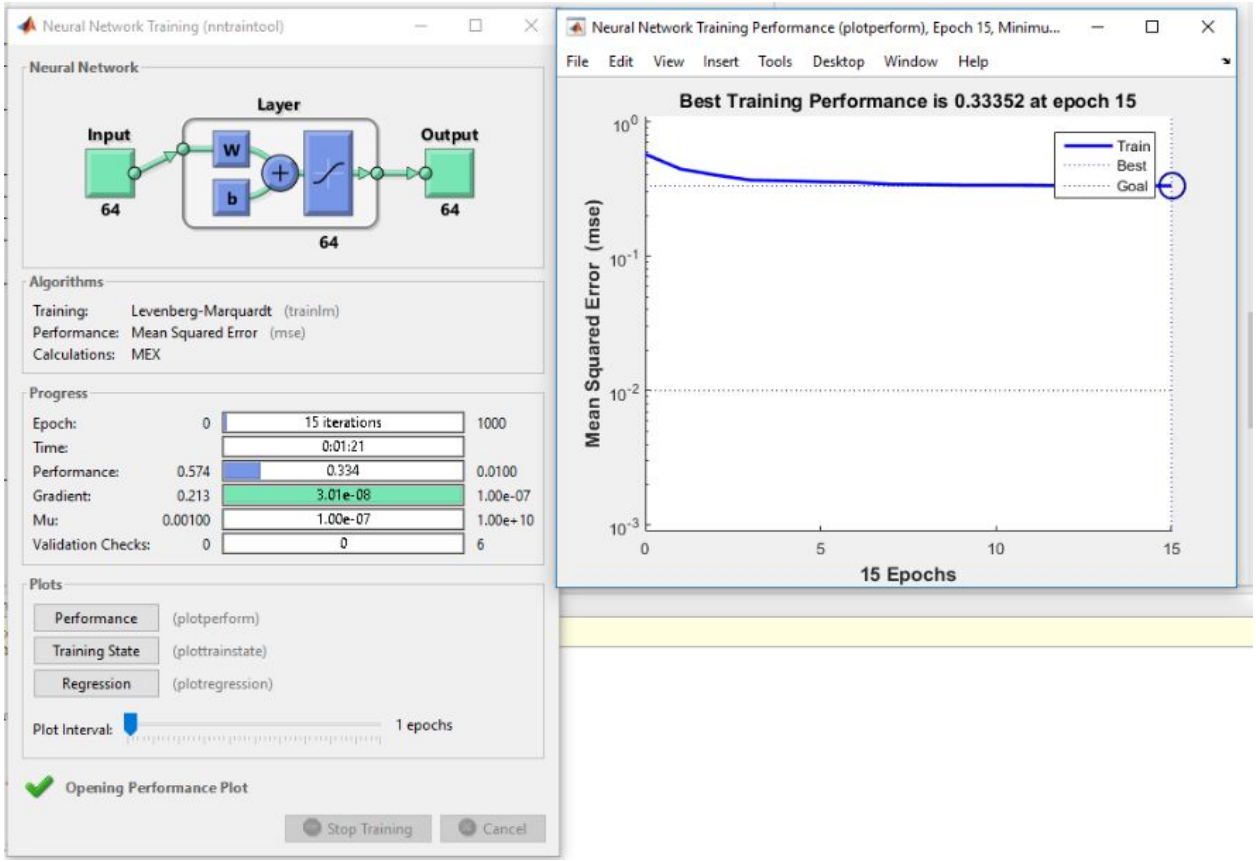
	smutek	
0/0.01	0/0.1	0/0.7
-1	-1	-0.4094
0.1/0.01	0.1/0.1	0.1/0.7
-1	0.2076	-0.0684
0.7/0.01	0.7/0.1	0.7/0.7
0.9985	1	-1

	duży uśmiech	
0/0.01	0/0.1	0/0.7
-1	1	-0.2798
0.1/0.01	0.1/0.1	0.1/0.7
-1	0.9471	-0.0635
0.7/0.01	0.7/0.1	0.7/0.7
-1	0.997	-1

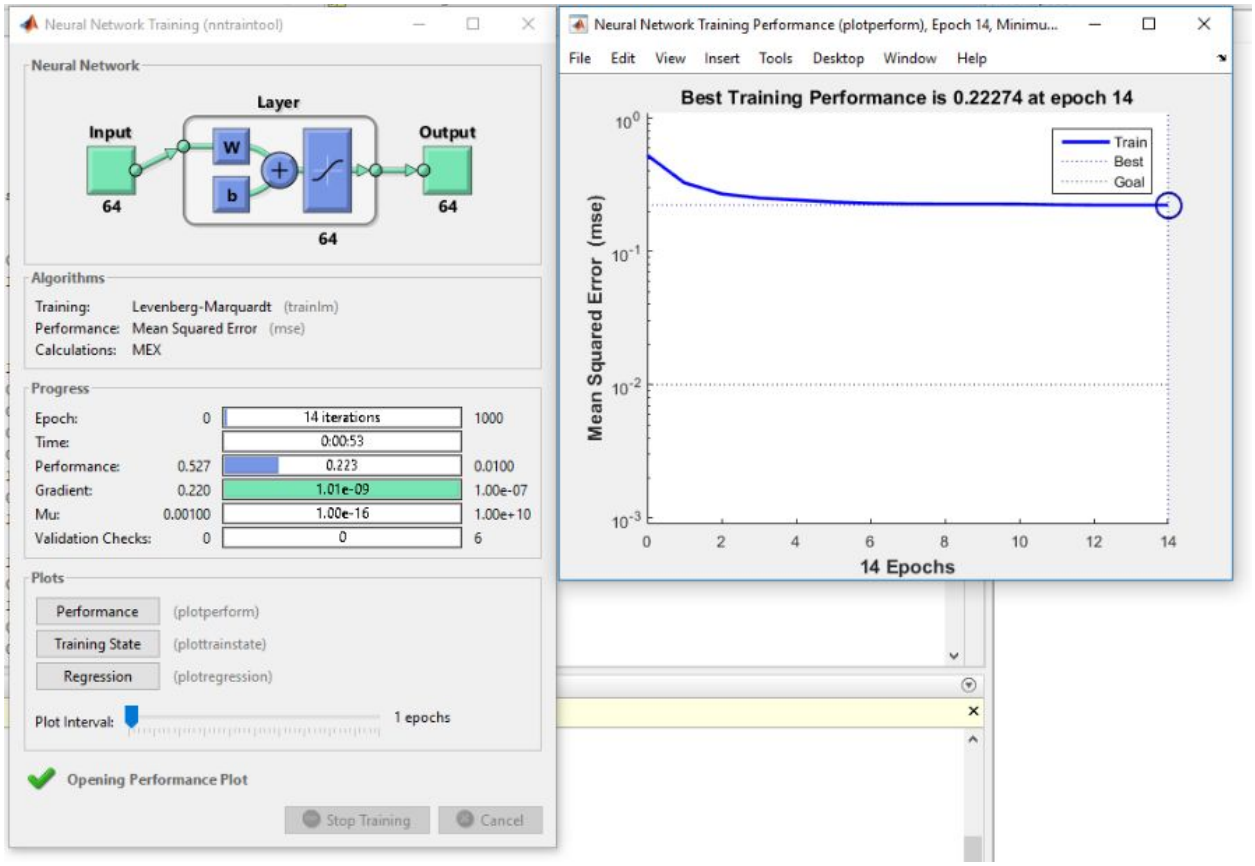
Najlepszy wynik dla 0.1/0.1



Najlepszy wynik dla 0.7/0.1



Najlepszy wynik dla 0/0.7



Analiza i wnioski

Uczenie sieci tą metodą trwa często o wiele dłużej. Reguła Hebba pozwala na uczenie sieci bez udziału nauczyciela. Na wykresach widzimy miejsce w którym jakość treningu zaczyna bardzo maleć - może to wskazywać na przeuczenie sieci. Interpretacja danych przez algorytm nie gwarantuje poprawności. Reguła Hebba opiera się na uczeniu bez nauczyciela, przez co sieć jest zmuszona samodzielnie decydować o skutkach w oparciu o dane wejściowe. Wysoki współczynnik uczenia powodował szybszy wzrost wartości wag metody. Dobranie nieodpowiedniego współczynnika uczenia sieci zwiększało ryzyko występowania błędów podczas treningu sieci. Reguła Hebba jest dość dobrą alternatywą dla poprzednich metod z uwagi na to, że pozwala na uczenie nienadzorowane.

Kod programu

```
clear all; close all; clc;

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; 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; 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 = 64;

net = newff(start, ilosc_wyjsc, {'tansig'}, 'trainlm', 'learnh');

 wejscie = [ 0 0 0 0;
            0 0 0 0;
            1 1 1 1;
            1 1 1 1;
            1 1 1 1;
            1 1 1 1;
            0 0 0 0;
            0 0 0 0;
            0 0 0 0;
            1 1 1 1;
            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 1 0;
            1 1 1 1;
            0 0 0 0;
            0 0 0 0;
            1 1 1 1;
            0 0 1 0;
            1 1 1 1;
            1 1 1 1;
            0 0 0 0;
            0 0 1 0;
            0 0 0 0;
            0 0 0 0;
            0 0 1 0;
            0 0 0 0;
```

```
        1 1 1 1;
        1 1 1 1;
        0 0 0 0;
        1 0 1 1;
        0 1 0 1;
        0 1 0 1;
        1 0 1 1;
        0 0 0 0;
        1 1 1 1;
        1 1 1 1;
        0 0 0 0;
        0 1 1 0;
        1 0 0 1;
        1 0 0 1;
        0 1 1 0;
        0 0 0 0;
        1 1 1 1;
        0 0 0 0;
        1 1 1 1;
        0 0 0 0;
        0 0 1 0;
        0 0 1 0;
        0 0 0 0;
        1 1 1 1;
        0 0 0 0;
        0 0 0 0;
        0 0 0 0;
        1 1 1 1;
        1 1 1 1;
        1 1 1 1;
        1 1 1 1;
        0 0 0 0;
        0 0 0 0;];


wyjscie = [ 1 0 0 0;
        0 1 0 0;
        0 0 1 0;
        0 0 0 1;];


lp.dr = 0.7;
lp.lr = 0.7;

wagiHebba = learnh(0, wejscie, 0, 0, wyjscie, 0, 0, 0, 0, 0, lp, 0);


net.trainParam.epochs = 1000;
net.trainParam.goal = 0.01;
abc = wagiHebba';
net = train(net, wejscie, abc);

 usmiechTestowy = [0 0 1 1 1 1 0 0;
                0 1 0 0 0 0 1 0;
                1 0 1 0 0 1 0 1;
                1 0 0 0 0 0 0 1;
                1 0 1 0 0 1 0 1;
```

```
                1 0 0 1 1 0 0 1;
                0 1 0 0 0 0 1 0;
                0 0 1 1 1 1 0 0;];

smutekTestowy =[0 0 1 1 1 1 0 0;
                0 1 0 0 0 0 1 0;
                1 0 1 0 0 1 0 1;
                1 0 0 0 0 0 0 1;
                1 0 0 1 1 0 0 1;
                1 0 1 0 0 1 0 1;
                0 1 0 0 0 0 1 0;
                0 0 1 1 1 1 0 0;];

placzTestowy = [0 0 1 1 1 1 0 0;
                0 1 0 0 0 0 1 0;
                1 1 1 0 0 1 1 1;
                1 0 1 0 0 1 0 1;
                1 0 1 0 0 1 0 1;
                1 0 1 0 0 1 0 1;
                0 1 0 1 1 0 1 0;
                0 0 1 1 1 1 0 0;];

duzyUsmiechTestowy=[0 0 1 1 1 1 0 0;
                    0 1 0 0 0 0 1 0;
                    1 0 1 0 0 1 0 1;
                    1 0 0 0 0 0 0 1;
                    1 0 1 1 1 1 0 1;
                    1 0 0 1 1 0 0 1;
                    0 1 0 0 0 0 1 0;
                    0 0 1 1 1 1 0 0;];

 efektHebba = wagiHebba;
 wynikUsmiech = sim(net, usmiechTestowy);
 wynikSmutek = sim(net, smutekTestowy);
 wynikPlacz = sim(net, placzTestowy);
 wynikDuzyUsmiech = sim(net, duzyUsmiechTestowy);

 disp('Usmiech: '), disp(wynikUsmiech(1));
 disp('Smutek: '), disp(wynikSmutek(1));
 disp('Placz: '), disp(wynikPlacz(1));
 disp('Duzy Usmiech: '), disp(wynikDuzyUsmiech(1));
```