Paulina Reichel, WIMiIP Inżynieria Obliczeniowa Grupa 2

Cel projektu:

Poznanie działania reguły Hebba na przykładzie rozpoznawania emotikon.

Realizacja projektu:

- ✓ Wygenerowanie danych uczących i testujących, zawierających 4 różne emotikony np. czarno-białe, wymiar 8x8 pikseli dla jednej emotikony. przykładowe wzory emotikon:
- ✓ Przygotowanie sieci oraz reguły Hebba w wersji z i bez współczynnika zapominania.
- ✓ Uczenie sieci dla różnych współczynników uczenia i zapominania.
- ✓ Testowanie sieci

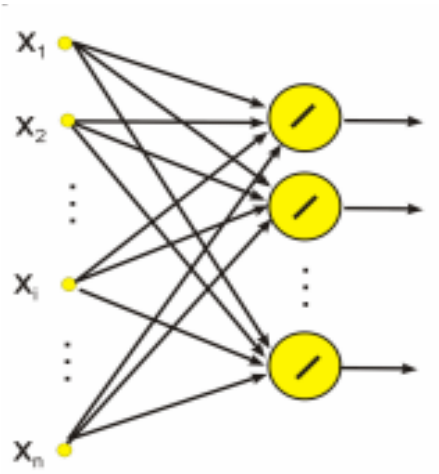
Wstęp teoretyczny:

Sieć neuronowa:

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

Sieć jednokierunkowa:

- ✓ neurony ułożone w jednej warstwie zasilanej z węzłów wejściowych,
- ✓ połączenie węzłów wejściowych i wyjściowych jest pełne (każdy węzeł jest połączony z każdym neuronem),
- ✓ przepływ sygnału występuje w jednym kierunku, od wejścia do wyjścia,
- ✓ węzły wejściowe nie tworzą warstwy neuronów, gdyż nie zachodzi w nich żaden proces obliczeniowy,
- ✓ sieć tego rodzaju nazywa się perceptronem jednowarstwowym



Uczenie według zasad Hebba:

Reguła ta występuje z nauczycielem jak i bez nauczyciela. Hebb zauważył podczas badań działania komórek nerwowych, że połączenie pomiędzy dwiema komórkami jest wzmacniane, jeśli w tym samym czasie obie komórki są aktywne. Zaproponował on algorytm, zgodnie z którym modyfikację wag przeprowadza się następująco:

$$w_i(t+1) = w_i(t) + nyx_i$$

Oznaczenia:

- ✓ i-numer wagi neuronu,
- ✓ t-numer iteracji w epoce,
- ✓ y-sygnał wyjściowy neuronu,
- ✓ x-wartość wejściowa neuronu,
- ✓ η - współczynnik uczenia (0,1).

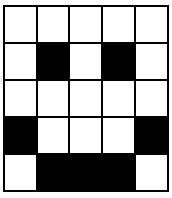
Ograniczenia:

przebieg uczenia zależy od wartości początkowych wag

- ✓ nie ma gwarancji, że jednej klasie wzorców będzie odpowiadał jeden neuron
- ✓ nie ma gwarancji, że wszystkie klasy wzorców będą miały oddzielne reprezentacje w postaci oddzielnych zbiorów aktywnych neuronów.

Przebieg ćwiczenia:

- ✓ Wygenerowanie danych uczących – 4 wybranych emotikon
- ✓ Przyjęcie matrycy 8x8, gdzie 1 odpowiada polom o kolorze czarnym, natomiast 0 o kolorze białym. Przykładowo:

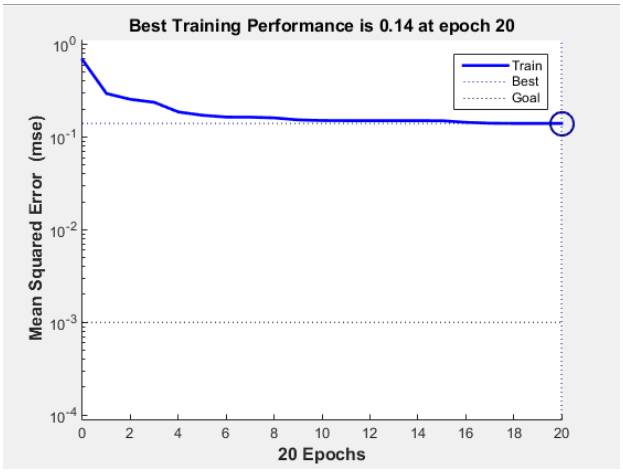


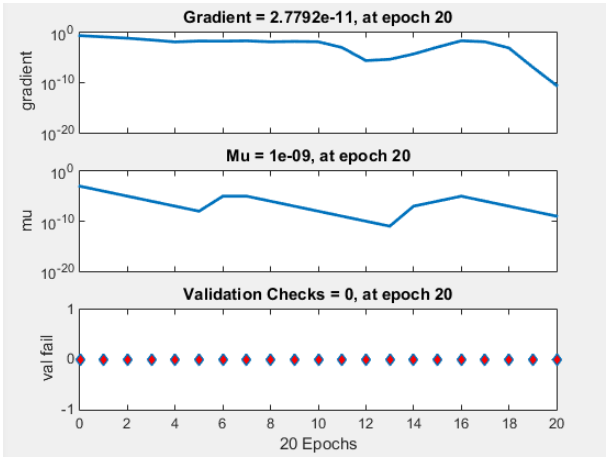
0	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**1**
0	**1**	**1**	**1**	**0**

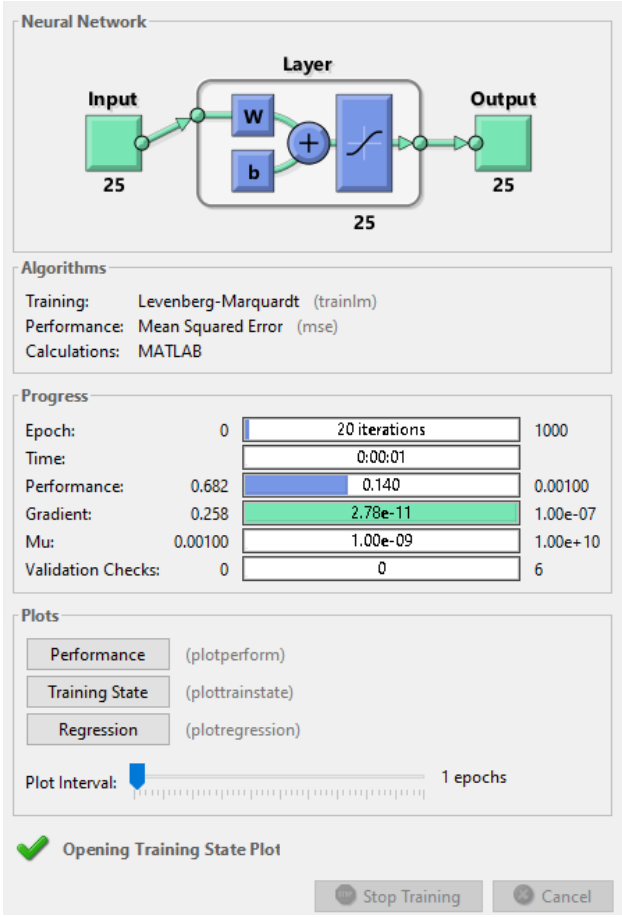
```
a_testowe= [0;0;0;0;0;
            0;1;0;1;0;
            0;0;0;0;0;
            1;0;0;0;1;
            0;1;1;1;0]
```

- ✓ Wygenerowanie danych wejściowych w postaci tabeli „WEJSCIE”, w której każdy wektor (emotikona) zajmował jedną kolumnę, stąd dane wyjściowe
- ✓ Wykorzystanie funkcją ‘newff’ ze zmiennymi parametrami uczenia (epokami, średnim błędem kwadratowym oraz współczynnikiem zapominania jak i współczynnikiem uczenia)
- ✓ Przeprowadzenie uczenia sieci według zasad Hebba (dokładny opis parametrów i użytych funkcji w listingu programu) przy zmiennych parametrach oraz w wersji z i bez współczynnika zapominania

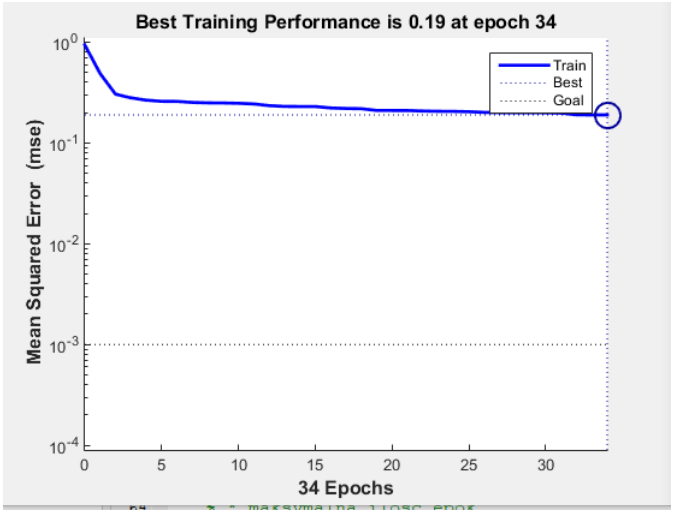
Przykładowe zrzuty ekranu działania programu:

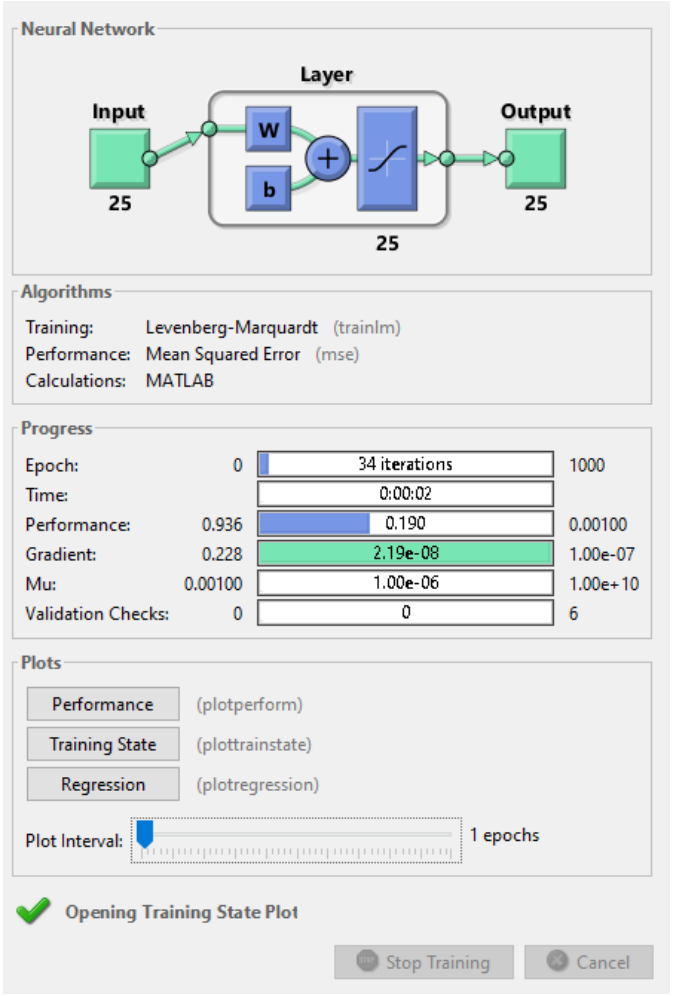


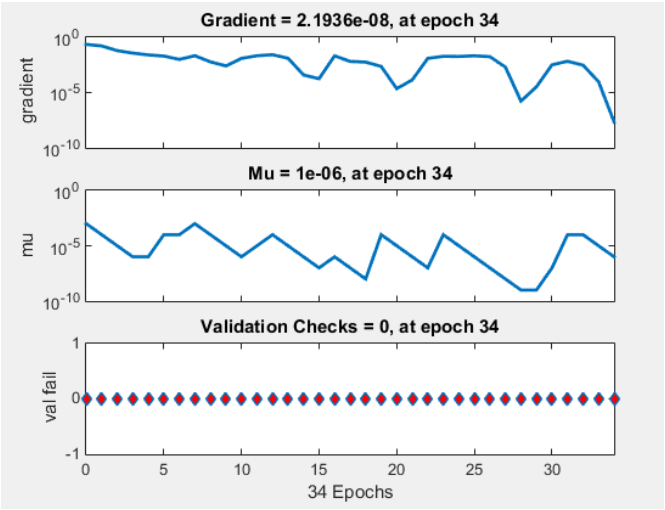




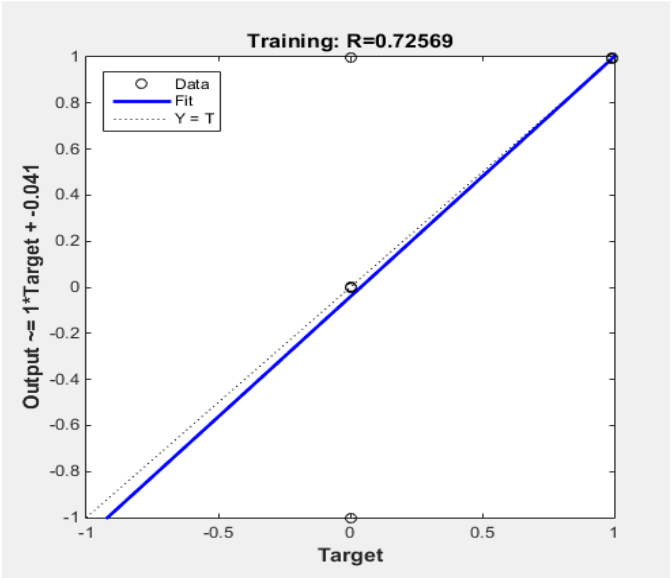
współczynnik zapominania = 0.5
współczynnik uczenia = 0.99
maksymalna ilość epok= 1000
cel wydajności sieci = 0.001
wskaźnik uczenia sieci=0.5







współczynnik zapominania = 0.0
współczynnik uczenia = 0.99
maksymalna ilość epok= 1000
cel wydajności sieci = 0.001
wskaźnik uczenia sieci=0.5



Przykładowe wyniki dla różnych kombinacji parametrów:

współczynnik zapominania	0.0			0.001			0.5		
współczynnik uczenia	0.01	0.1	0.99	0.01	0.1	0.99	0.01	0.1	0.99
ilość potrzebnych epok	12	**19**	**12**	**10**	**11**	**20**	**17**	**27**	**22**
Wyniki programu dla poszczególnych emotikon oraz ilość potrzebnych epok:									
☺	8.1026e-12	-1.8918e-13	0	-1	-1	1.1369e-13	4.4409e-16	-2.2204e-16	3.7748e-15
:O	-6.1317e-09	-1	2.2204e-16	2.7195e-10	2.2449e-13	-2.2204e-16	1	0	2.2204e-16
☹	-2.2204e-16	2.2204e-16	-1	2.2204e-16	-9.5457e-13	0	-1	2.2204e-16	2.2204e-16
😐	**0**	-1	-0.7764e-15	1	-2.2204e-16	-4.5714e-09	-7.9936e-15	2.2204e-16	-2.2204e-16

Analiza i wnioski:

- ✓ Reguła uczenia sieci według Hebba zawiera wiele ograniczeń
- ✓ Przebieg uczenia zależy od wartości początkowych wag
- ✓ Nie ma gwarancji, że jednej klasie wzorców będzie odpowiadał jeden neuron
- ✓ Nie ma gwarancji, że wszystkie klasy wzorców będą miały oddzielne reprezentacje w postaci oddzielnych zbiorów aktywnych neuronów
- ✓ Wraz ze zmniejszaniem współczynnika zapominania, dobierane wagi zmieniają swoje wartości
- ✓ Istnieje możliwość uzyskania wartości dobieranej wagi, poprzez wprowadzenie dwóch par takich samych danych wejściowych
- ✓ Wysoki współczynnik uczenia skutkował szybszym wzrostem wartości wag metody. Dobranie nieodpowiedniego współczynnika uczenia sieci może zwiększać ryzyko występowania błędów podczas treningu sieci (np. dane mogą wyjść poza swój zakres)
- ✓ Reguła Hebba pozawala na uczenie sieci bez nauczyciela
- ✓ Ze względu na użyty algorytm można zauważyć, że sieć w porównaniu do wcześniejszych metod użytych na zajęciach zdecydowanie potrzebuje więcej czasu na naukę. Ma na to wpływ przymus nauki bez nauczyciela. Na podstawie tej informacji można wnioskować, że uzyskane wyniki nie świadczą o błędzie - tylko o zbyt wczesnym ich odczytaniu
- ✓ Zmiany wag w dużym stopniu zależą od współczynnika zapominania. Gdy go nie mamy nie ma stabilizacji i wagi mogą rosnąć w nieskończoność. Lepiej radzi sobie sieć w której występuję współczynnik zapominania, który zdecydowanie poprawia stabilność procesu uczenia.
- ✓ Współczynnik zapominania powinien być niewielką wartością. Przyjęcie dużej jego wartości powoduje, że neuron zapomina większość tego, co zdołał nauczyć się w przeszłości. Jego wartości nie powinna przekraczać 0,1, dzięki temu neuron zachowuje większość informacji zgromadzonej w procesie uczenia, a jednocześnie możliwe jest ustabilizowanie wag na określonym poziomie.

- ✓ Skuteczność procesu uczenia zależy od współczynnika uczenia. Wraz z jego wzrostem proces uczenia jest poprawniejszy. Jest to wytłumaczalne z tego względu, że im ta wartość jest większa tym przyrost wag, które na samym początku są niewielkie jest szybszy, a więc i proces uczenia przebiega szybciej.

Listing programu:

```
%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 litere - 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 = 0.5;
% * 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));
```