

Grupa lab. <b>2</b>	Przedmiot <b>Podstawy Sztucznej Inteligencji</b>	Data wykonania. <b>15.12.2018</b>
Nr ćwicz. <b>4</b>	Temat ćwiczenia. <b>Uczenie sieci regułą Hebba</b>	
Imię i nazwisko. <b>Katarzyna Giądła</b>		Ocena i uwagi

### Część teoretyczna

Celem tego ćwiczenia było poznanie działania reguły Hebba na przykładzie rozpoznawania emotikon.

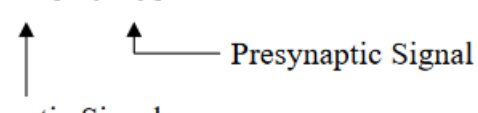
Metoda samouczenia sieci neuronowych Hebba jest jedną z najpopularniejszych. Opiera się ono na postulacie Hebba mówiącym, że połączenia między źródłami sygnałów i neuronami, które na nie silnie reagują są wzmacniane.

W sieciach neuronowych możliwe jest nauczanie neuronów reakcji na bodźce, tzn. reagowania na dane podawane na wejściu sieci. Sieci pokazuje się kolejne przykłady sygnałów wejściowych, nie podając żadnych informacji o tym, co z tymi sygnałami należy zrobić. Przez to jest czasem nazywana uczeniem korelacyjnym.

Dopiero sieć przez obserwację otoczenia i odbieranie różnych sygnałów sama odkrywa zależności występujące między sygnałami. Podczas procesu uczenia, w miarę napływających wzorców (wartościami wag) sieć adaptuje sekwencyjnie swoje wagi, aby prawidłowo rozpoznawać dane.

Dobór początkowych wartości wag neuronów sieci przeznaczonej do uczenia jest bardzo istotną kwestią. Proces uczenia bowiem tylko pogłębia i doskonali pewne tendencje istniejące w sieci od samego początku.

Po dłuższym czasie takiego samouczenia w sieci powstają 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 obiektami zainteresowania innych neuronów. Dzięki temu 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

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


Wagi mogą przybierać wartości dowolnie duże, gdyż w każdym cyklu uczącym następuje proces sumowania.

Jedną z metod stabilizacji procesu uczenia jest wprowadzenie tzw. współczynnika zapomnienia  $0 < \gamma < 1$ , który jest wykorzystywany przy ustalaniu nowych wag następująco:

$$w_{ij}(t+1) = w_{ij}(t)(1 - \gamma) + \Delta w_{ij}$$

Algorytmy samouczące posiadają pewne dosyć istotne wady:

- stosunkowo niska efektywność uczenia
- możliwość pomijania niektórych klas w nauczanej sieci
- Powstawanie redundantnych nadprezentacji klas
- uczenie bez nauczyciela jest zawsze powolniejsze
- potrzeba o wiele więcej elementów warstwy wyjściowej niż wynosi oczekiwana liczba różnych wzorów, które sieć ma rozpoznawać

## Część praktyczna

Do wykonania projektu użyłam pakietu *MATLAB*, a konkretniej narzędzia *Neural Networking Training Tool*. Do procesu uczenia sieci wykorzystałam sieć jednowarstwową o tangensowej funkcji aktywacji oraz wykorzystującym algorytm wstecznej propagacji (by wagi mogły być modyfikowane).

Stworzyłam również emotikony o rozmiarze 5x5, za pomocą których będziemy uczyć naszą sieć. Pola białe mają wartość 0, a pola czarne wartość 1:

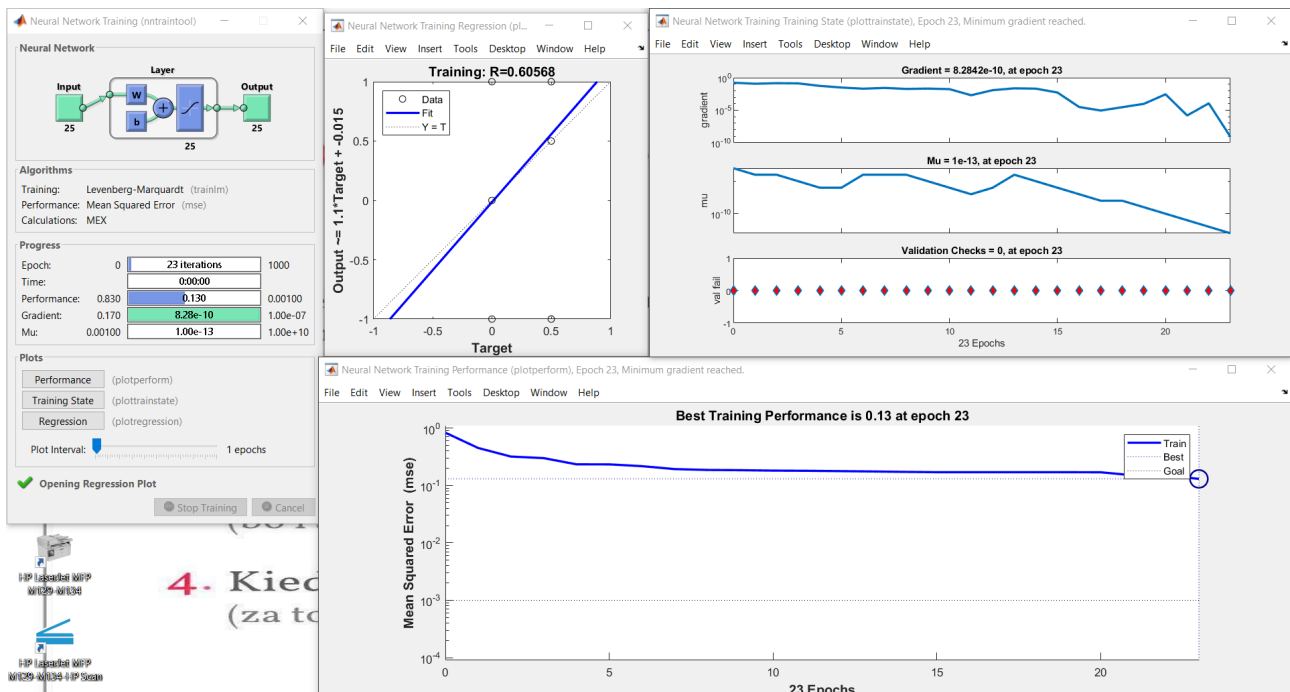
Minka	Kod	Minka	Kod
	00000 01001 00000 10001 011110		00000 01010 00000 01110 10001
	01010 00000 11111 00101 00010		01010 00000 11111 10001 01110

Następnie wygenerowałam współczynniki Hebba w wersji ze współczynnikiem zapominania i w wersji bez zapominania.

Następnie przetestowałam działanie sieci ze współczynnikiem uczenia całej sieci, współczynnikiem uczenia algorytmu Hebba oraz współczynnikiem zapominania (oraz bez tego współczynnika).

<i>Współczynniki Hebba w zależności od współczynnika uczenia i użycia (lub nie) współczynnika uczenia</i>												
Współczynnik uczenia	0.1				0.5				0.9			
Współczynnik zapominania	0.1	0.5	0.9	brak	0.1	0.5	0.9	brak	0.1	0.5	0.9	brak
Emotka												
:)	0,7	0,7	0,7	0,7	3,5	3,5	3,5	3,5	6,3	6,3	6,3	6,3
:(	0,7	0,7	0,7	0,7	3,5	3,5	3,5	3,5	6,3	6,3	6,3	6,3
:P	0,9	0,9	0,9	0,9	4,5	4,5	4,5	4,5	8,1	8,1	8,1	8,1
:D	1,2	1,2	1,2	1,2	6	6	6	6	10,8	10,8	10,8	10,8

<i>Efekty uczenia całego algorytmu w zależności od współczynnika uczenia i użycia (lub nie) współczynnika uczenia (jako wartość współczynnika uczenia Hebba przyjmuję 0.5)</i>													
Współczynnik uczenia	0.1				0.5				0.9				
Współczynnik zapominania	0.1	0.5	0.9	brak	0.1	0.5	0.9	brak	0.1	0.5	0.9	brak	
Emotka													
:)	4,00E+00	7,50E+00	-5,00E-01	5,00E+00	3,50E+00	4,00E+00	4,50E+00	4,50E+00	2,50E+00	1,50E+00	4,50E+00	5,00E-01	
:(	-1,50E+00	4,50E+00	2,50E+00	6,00E+00	2,50E+00	7,91E-07	7,00E+00	4,50E+00	4,50E+00	1,00E+00	2,50E+00	4,00E+00	
:P	2,50E+00	3,00E+00	4,00E+00	7,50E+00	5,50E+00	6,50E+00	2,50E+00	6,50E+00	4,50E+00	2,00E+00	3,50E+00	5,00E+00	
:D	5,00E+00	3,00E+00	4,50E+00	6,00E+00	6,50E+00	7,00E+00	3,50E+00	7,50E+00	4,00E+00	8,00E+00	6,00E+00	3,00E+00	
Ilość epok	19	15	20	20	23	26	29	28	16	32	21	10	



Wykresy dla algorytmu o wsp. uczenia Hebb'a, wsp. uczenia całego algorytmu i wsp. zapominania równemu 0.5

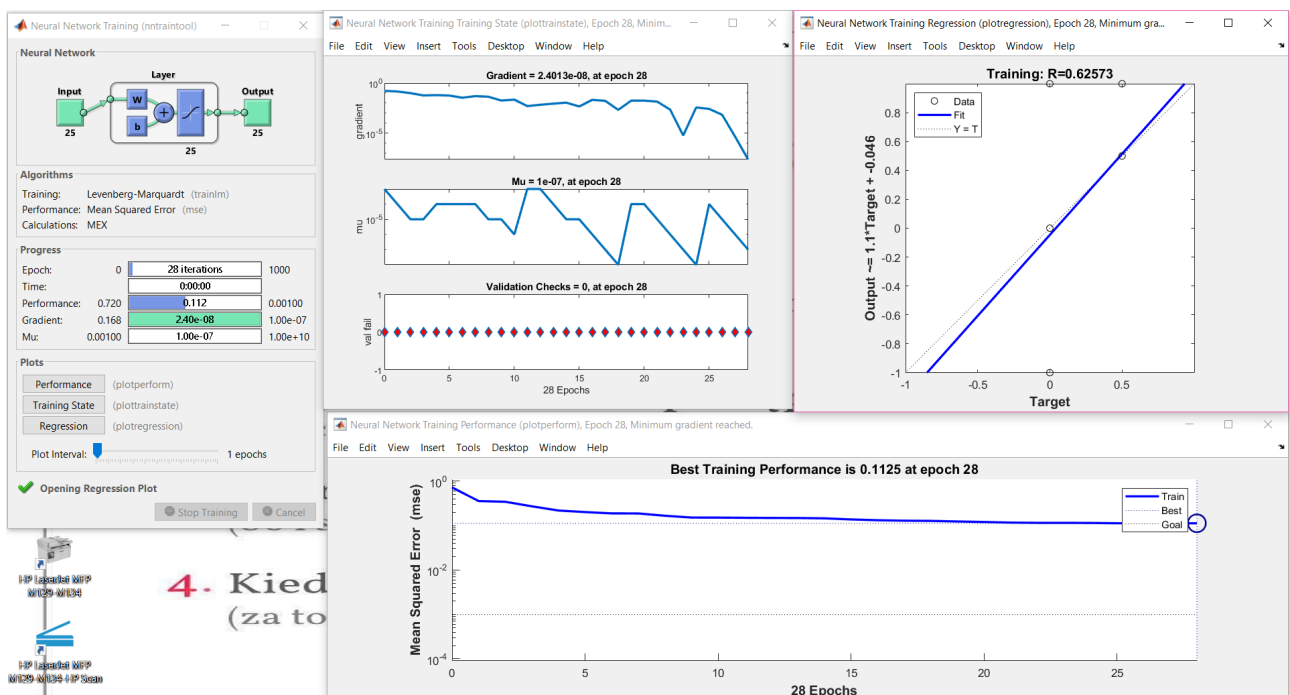
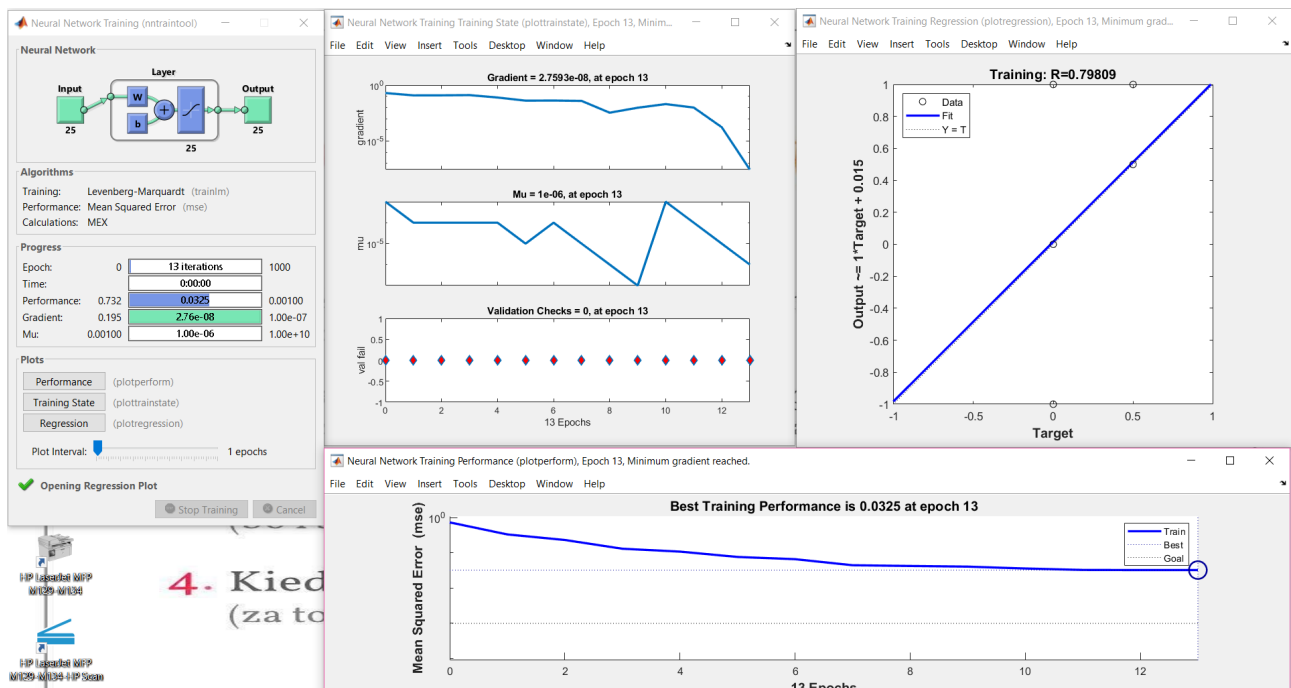


Figure 1: Wykresy dla algorytmu o wsp. uczenia Hebb'a równemu 0.5, wsp. uczenia całego algorytmu 0.9 i braku współczynnika zapominania



Wykresy dla uczenia sieci o wsp. uczenia Hebb'a równemu 0.5, wsp. zapominania na poziomie 0.5 i wsp. uczenia algorytmu równemu 0.1

## Wnioski

- Środowisko *MATLAB* dostarcza nam narzędzia umożliwiające analizę danych pod kątem samouczenia sieci neuronowych.
- Można w naszym przypadku dostrzec, że praktycznie niewidoczny jest związek między zmianą wsp. uczenia Hebb'a a wsp. zapominania. Może to wynikać z faktu, że wartości są na tyle niewiele, że ciężko jest o dostrzeżenie znaczących różnic między wynikami.
- Łatwo zauważyć, że w zależności od doborów współczynników mamy do czynienia ze zmieniającą się charakterystyką błędu średniokwadratowego. Jeśli przyjąć kryterium, że uznajemy charakterystykę liniową za najlepszą, to najbardziej zbliżona temu jest sytuacja, gdy wszystkie współczynniki są na równym poziomie, a najmniej stabilne, gdy są duże różnice między poszczególnymi współczynnikami.
- Uczenie bez współczynnika zapominania jest krótsze, jednak przez to wyniki mogą bardziej odbiegać od tych, które założyliśmy.
- Algorytmy samouczące możemy wykorzystać do analizy regresji i wyszukiwania zależności między danymi – dane te jednak nie mogą stanowić podstaw dalszych rozważań, a mogą być jedynie potwierdzeniem stawianych przez nas hipotez.

## Listingi kodów źródłowych

```
close all; clear 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]; %format danych wejściowych

out_s = 25; %ilość wyjść z sieci

net = newff(start, out_s, {'tansig'}, 'trainlm', 'learnh');
%stworzenie prostej sieci
%posługującej się tangensem hiperbolicznym jako funkcja
aktywacji, algorytm wstecznej propagacji

in_value = [0 0 0 0;
            0 0 1 1;
            0 0 0 0;
            0 0 1 1;
            0 0 0 0;

            0 0 0 0;
            1 1 0 0;
            0 0 0 0;
            1 1 0 0;
            0 0 0 0;

            0 0 1 1;
            0 0 1 1;
            0 0 1 1;
            0 0 1 1;
            0 0 1 1;

            1 0 0 1;
            0 1 0 0;
            0 1 1 0;
            0 1 0 0;
            1 0 0 1;

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

out_value = [1 0 0 0; % :)
            0 1 0 0; % :(
```

```

0 0 1 0; % :P
0 0 0 1]; % :D

lp.dr = 0.1; %wsp. zapominania
lp.lr = 0.5; %wsp. uczenia dla wsp. Hebba

wagiHebba = learnh([], in_value, [], [], out_value, [], [],
[], [], [], lp, []); %ustalenie wag Hebba

net.trainParam.epochs = 1000; %maks. liczba epok
net.trainParam.goal = 0.001; %błąd średniokwadratowy
net.trainParam.lr = 0.5; %wsp. uczenia algorytmu

net = train(net, in_value, wagiHebba'); %uczenie danych

test_smile = [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];

test_sad = [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];

test_tongue = [0;1;0;1;0;
               0;0;0;0;0;
               1;1;1;1;1;
               0;0;1;0;1;
               0;0;0;1;0];

test_bigSmile = [0;1;0;1;0;
                 0;0;0;0;0;
                 1;1;1;1;1;
                 1;0;0;0;1;
                 0;1;1;1;0];

efektHebba = wagiHebba; %zapisanie pierwotnych współczynników
Hebba

efekt = sim(net, in_value); %test sieci

disp('Współczynniki Hebba: ')
disp(':) = '), disp(sum(efektHebba(1, ':')));
disp(':( = '), disp(sum(efektHebba(2, ':')));
disp(':P = '), disp(sum(efektHebba(3, ':')));
disp(':D = '), disp(sum(efektHebba(4, ':')));

```

```
disp('Działanie algorytmu z wykorzystaniem reguły Hebba: ')  
disp(':) = '), disp(sum(efekt(:, 1)));  
disp(':( = '), disp(sum(efekt(:, 2)));  
disp(':P = '), disp(sum(efekt(:, 3)));  
disp(':D = '), disp(sum(efekt(:, 4)));
```