Kamil Majchrowicz IO

Podstawy Sztucznej Inteligencji

Sprawozdanie nr 4

Uczenie sieci metodą Hebba

1. Cel ćwiczenia

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

2. Zadania do wykonania

- Wygenerowanie danych uczących i testujących, zawierających 4 różne emotikony np. czarno-białe, wymiar 8x8 pikseli dla jednej emotikony.

- Przygotowanie (implementacja lub wykorzystanie gotowych narzędzi) sieci oraz reguły Hebba w wersji z i bez współczynnika zapominania.

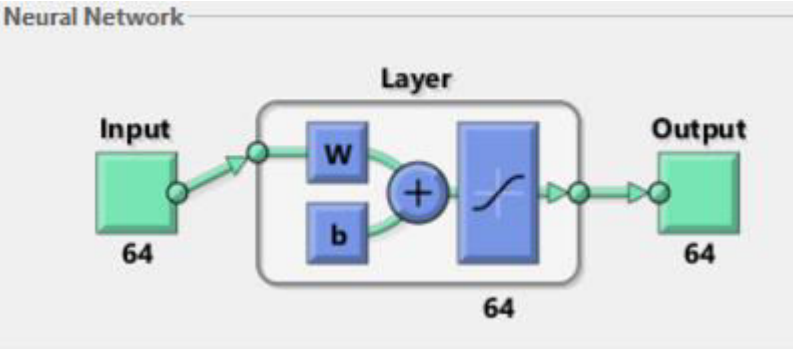
- Uczenie sieci dla różnych współczynników uczenia i zapominania.

- Testowanie sieci.

3. Opis działania reguły Hebba

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.



Hebb zaproponował algorytm, zgodnie z którym modyfikację wag przeprowadza się następująco:

$$w_i(t+1) = w_i(t) + \eta y x_i$$

Oznaczenia:

• i-numer wagi neuronu,

• t-numer iteracji w epoce,

• y-sygnał wyjściowy neuronu,

• x-wartość wejściowa neuronu,

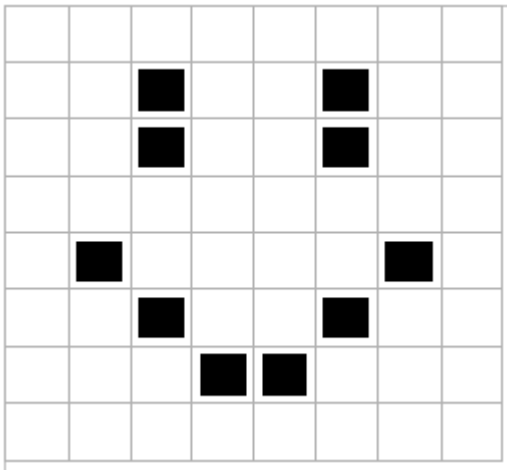
• η - współczynnik uczenia (0,1).

Do przeanalizowania działania sieci można wykorzystać 4 wygenerowane emotikony:

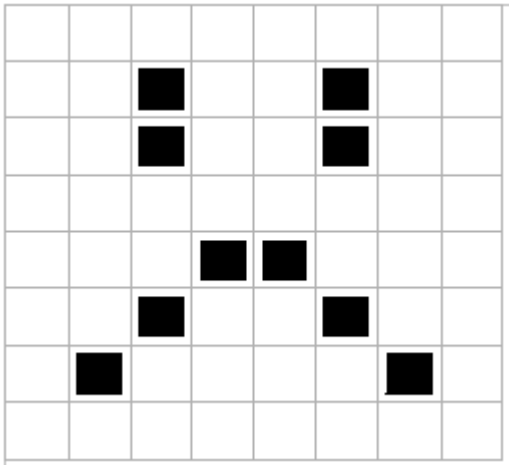
-SMILE, SAD, KISS, CONFUSE przedstawione w formie matryc 8x8 z reprezentacją binarną.

kwadraty wypełnione - 1 , puste - 0

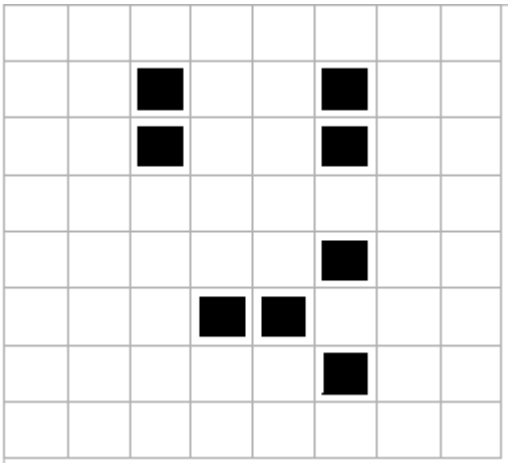
SMILE



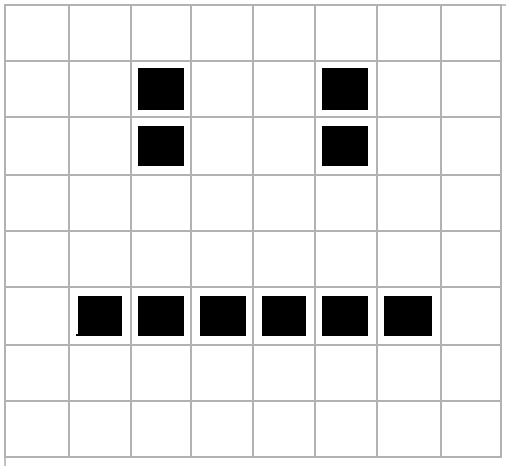
SAD



KISS



CONFUSE



Kod programu:

```
%%
close all; clear all; clc;

%wejścia do sieci z wartościami min i max
minmax=[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];

%ilość wyjść z sieci
wyjscia = 64;

%użycie funkcji newff(tworzy nową sieć)
net = newff(minmax, wyjscia,{'tansig'}, 'trainlm','learnh');

% Emotikony SMILE/SAD/KISS/CONFUSE w wersi binarnej:
input = [0 0 0 0;
 0 0 0 0;
 0 0 0 0;
 0 0 0 0;
 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;
 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;
 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;
 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 0 0 0;
 0 1 0 0;
 0 1 0 0;
 0 0 1 0;
 1 0 0 0;
 0 0 0 0;
 0 0 0 0;
 0 0 0 1;
 1 1 0 1;
 0 0 1 1;
 0 0 1 1;
 1 1 0 1;
 0 0 0 1;
 0 0 0 0;
 0 0 0 0;
 0 1 0 0;
 0 0 0 0;
 1 0 0 0;
 1 0 0 0;
 0 0 1 0;
 0 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 0 0;
 0 0 0 0;
 0 0 0 0;
 ];

%zmienna zawierająca 1 gdy trafimy w emotikon i 0 gdy chybimy
output = [ 1 0 0 0   %SMILE
           0 1 0 0   %SAD
           0 0 1 0   %KISS
           0 0 0 1]; %CONFUSE

%parametry reguły Hebba
lp.dr = 0.1; %współczynnik zapominania
lp.lr = 0.1; %współczynnik uczenia
```

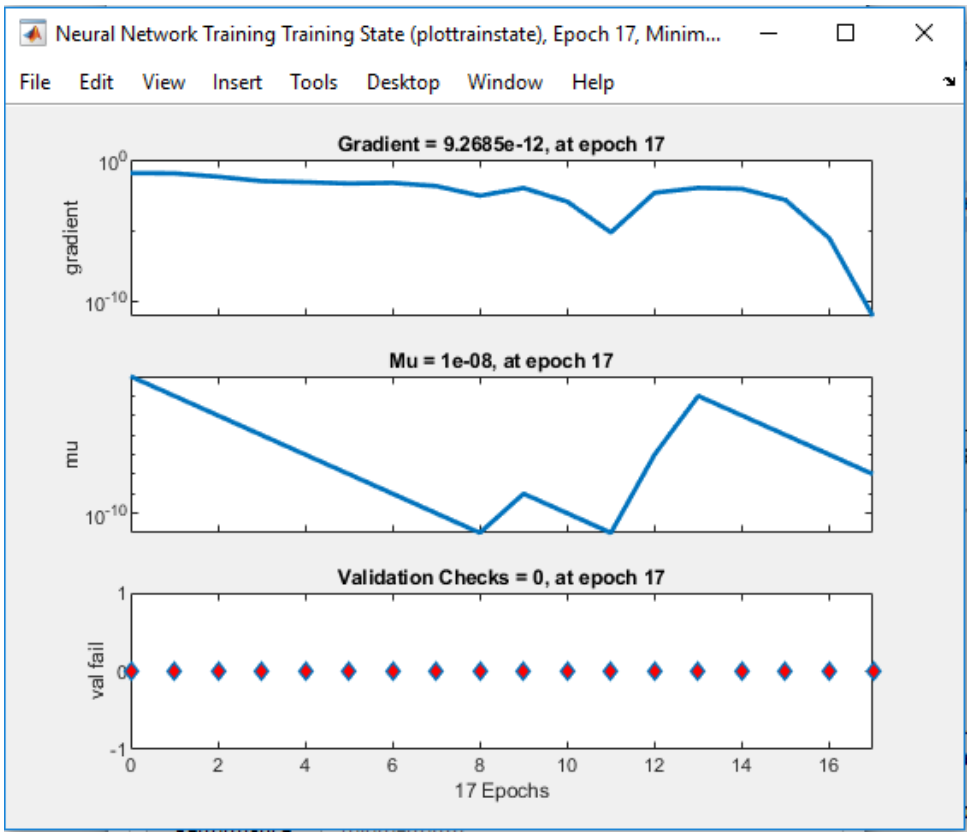
```
%Reguła Hebba:
hebb = learnh( [], input, [], [], output, [], [], [], [], [], lp, []);
heb=hebb';
net.trainParam.epochs = 1000;
net.trainParam.goal = 0.001;
net.trainParam.lr=0.5;

%trenowanie sieci z użyciem reguły Hebba
net = train(net, input, heb);

%DANE TESTUJACE
smile = [0 0 0 0 0 0 0 0; 0 0 1 0 0 1 0 0;0 0 1 0 0 1 0 0;
0 0 0 0 0 0 0 0; 0 1 0 0 0 0 1 0;0 0 1 0 0 1 0 0;
0 0 0 1 1 0 0 0; 0 0 0 0 0 0 0 0 ];
sad = [0 0 0 0 0 0 0 0; 0 0 1 0 0 1 0 0; 0 0 1 0 0 1 0 0;
0 0 0 0 0 0 0 0; 0 0 0 1 1 0 0 0; 0 0 1 0 0 1 0 0;
0 1 0 0 0 0 1 0; 0 0 0 0 0 0 0 0 ];
kiss=[0 0 0 0 0 0 0 0; 0 0 1 0 0 1 0 0; 0 0 1 0 0 1 0 0;
0 0 0 0 0 0 0 0; 0 0 0 0 0 0 1 0 0; 0 0 0 1 1 0 0 0;
0 0 0 0 0 1 0 0; 0 0 0 0 0 0 0 0 ];
confuse = [ 0 0 0 0 0 0 0 0; 0 0 1 0 0 1 0 0; 0 0 1 0 0 1 0 0;
0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0; 0 1 1 1 1 1 1 0;
 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 ];

%sprawdzenie czy sieć została poprawnie wytrenowana
test = sim(net, smile);
test1 = sim(net, sad);
test2 = sim(net, kiss);
test3 = sim(net, confuse);

%wypisanie wyników:
disp('SMILE ='), disp(test(1));
disp('SAD ='), disp(test1(2));
disp('KISS ='), disp(test2(3));
disp('CONFUSE ='), disp(test3(4));
```

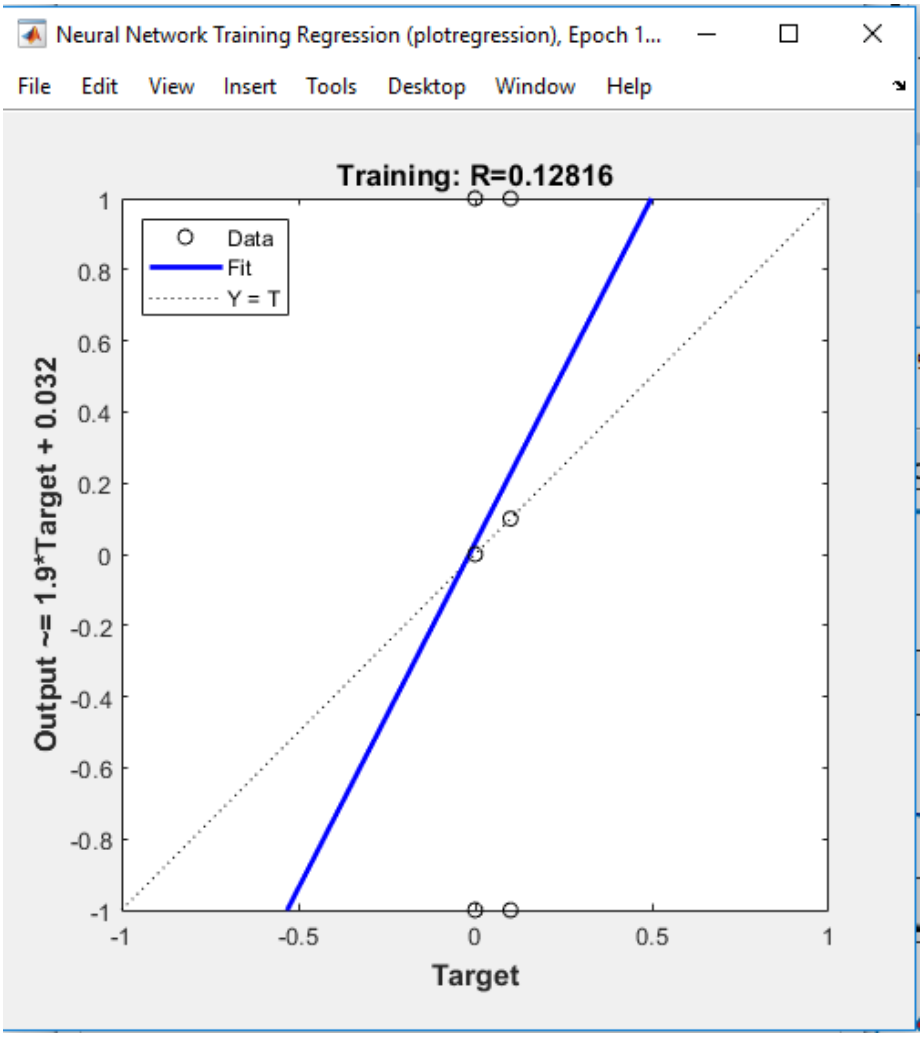
Przykładowe wyniki:

Współczynniki uczenia/zapominania	SMILE	SAD	KISS	CONFUSE	Liczba epok
0.001/0	1	1	0.2851	0.8201	24
0.01/0.01	0.7581	0.9252	0.1837	-0.0268	14
0.1/0.1	0.9012	-0.9842	1	-0.6344	16
0.5/0.1	-0.682	-1	0.9542	1	29
0.99/0.5	0.7795	-1	-1	0.8652	12

Testy dla wartości współczynników 0.1/0.1


Neural Network Training Performance (plotperform), Epoch 17, Minimu...
File Edit View Insert Tools Desktop Window Help
Best Training Performance is 0.28875 at epoch 17
Train
Best
Goal
Mean Squared Error (mse)
17 Epochs


Neural Network Training Training State (plottrainstate), Epoch 17, Minim...
File Edit View Insert Tools Desktop Window Help
Gradient = 9.2685e-12, at epoch 17
gradient
Mu = 1e-08, at epoch 17
mu
Validation Checks = 0, at epoch 17
val fail
17 Epochs



Wnioski:

- Metoda Hebba opiera się na uczeniu bez nauczyciela, a co za tym idzie musi sama wyciągać wnioski na podstawie posiadanych danych, co nie zawsze kończy sie poprawnym działaniem

-Sieć oparta na regule Hebba potrzebuje o wiele więcej czasu niż taka sama sieć uczona przez nauczyciela.

- Ustalanie wartości wag ma bardzo ważne znaczenie dla procesu nauki

- Jest bardzo podobna do nauki prawdziwego biologicznego neuronu