Gabriela Ciołek

Nr indeksu: 293083

Inżynieria Obliczeniowa

Wydział Inżynierii Metali i Informatyki Przemysłowej

Akademia Górniczo-Hutnicza w Krakowie

**Sprawozdanie 4**

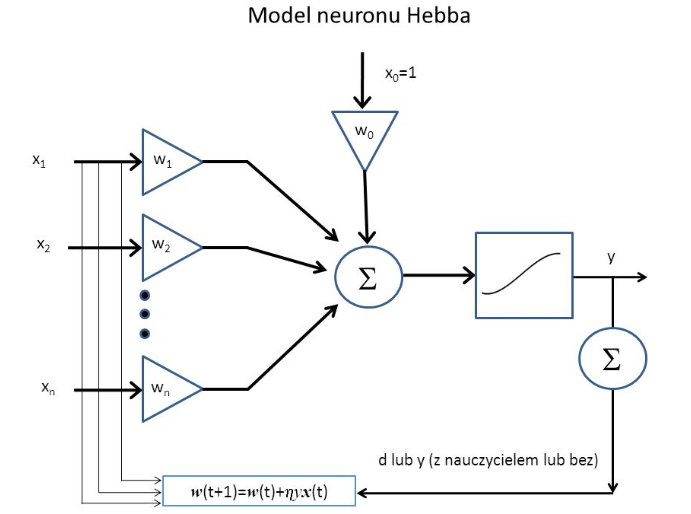
Przedmiot: Podstawy sztucznej inteligencji

Temat: Uczenie sieci regułą Hebba

**CEL:**

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

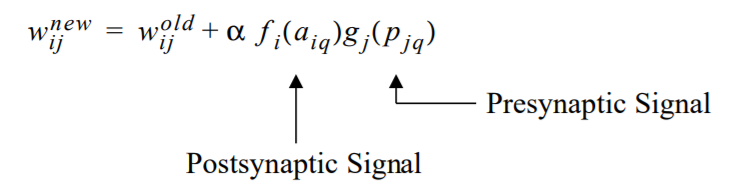
**TEORIA:**

****

**REGUŁA HEBBA**

Jest to jedna z najpopularniejszych metod samo uczenia 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.



**LISTING KODU WRAZ Z KOMENTARZAMI:**

close all; clear all; clc;

%wejścia do sieci z min i max wartościami

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

ilosc\_wyj = 64;

%użycie funkcji tworzącej sieć

net = newff(minmax, ilosc\_wyj,{'tansig'}, 'trainlm', 'learnh');

%kolumnowe wprowadzenie emotikon w formie 0-1

% smile/shock/confuse/sad

input = [ 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 0 0;

1 1 1 1;

0 0 0 0;

0 0 0 0;

1 1 1 1;

0 0 0 0;

1 1 1 1;

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 1 1 1;

1 1 1 1;

0 0 0 0;

1 0 1 0;

0 1 1 1;

0 1 1 1;

1 0 1 0;

0 0 0 0;

1 1 1 1;

1 1 1 1;

0 0 0 0;

0 0 0 1;

1 1 0 0;

1 1 0 0;

0 0 0 1;

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;

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;

];

%zmienna zawierająca 1 gdy trafimy w emotikon i 0 gdy chybimy

output = [ 1 0 0 0 %smile

0 1 0 0 %shock

0 0 1 0 %confuse

0 0 0 1]; %sad

%parametry reguły Hebba

lp.dr = 0.5; %wsp. zapominania

lp.lr = 0.9; %wsp. uczenia

%użycie reguły Hebba

hebb = learnh( [], input, [], [], output, [], [], [], [], [], lp, []);

heb=hebb';

net.trainParam.epochs = 1000;

net.trainParam.goal = 0.01;

%trenowanie sieci z użyciem reguły Hebba

net = train(net, input, heb);

%DANE TESTUJACE

smile = [ 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];

shock = [ 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 0 1 1 0 0 1; 0 1 0 0 0 0 1 0; 0 0 1 1 1 1 0 0];

confuse = [ 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];

sad = [ 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];

%sprawdzenie poprawności wytrenowanej sieci

test = sim(net, smile);

test1 = sim(net, shock);

test2 = sim(net, confuse);

test3 = sim(net, sad);

%wypisanie wartości

disp('SMILE ='), disp(test(1));

disp('SHOCK ='), disp(test1(1));

disp('CONFUSE ='), disp(test2(1));

disp('SAD ='), disp(test3(1));

**OPIS WYKONANEGO ZADANIA:**

Do moich zadań należało:

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

Stworzyłam 4 emotikony, które powstały na macierzy wielkości 8x8. Białemu polu przyporządkowałam wartość 0, natomiast czarnemu wartość 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

Obraz zawierający krzyżówka, tekst, owoce, wewnątrz

Opis wygenerowany automatycznie

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

Obraz zawierający krzyżówka, tekst, owoce, sąsiadująco

Opis wygenerowany automatycznie

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

Obraz zawierający krzyżówka, tekst, owoce, wewnątrz

Opis wygenerowany automatycznie

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

Obraz zawierający krzyżówka, tekst, owoce, czarny

Opis wygenerowany automatycznie

Następnie powyższe macierze zapisałam w formie kolumnowej jako zmienną *input*. Dane testujące powstały z powyższych macierzy i zostały zapisane w formie wierszowej, gdzie każde przejście do nowego wiersza jest zaznaczone znakiem ; .

Stworzyłam także tablicę 64 wartości (0,1), które odpowiadają wielkości macierzy, ponieważ na każdym polu może znaleźć się tylko wartość 0-puste lub 1-pełne.

Zmienna *output* zawiera wyjście gdzie 1 oznacza, że dany emotikon powstał, a 0 jest przeciwieństwem.

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

Skorzystałam z gotowych narzędzi pakietu Matlab, które zawierają już przygotowane funkcje tworzące sieć i wykorzystujące regułę Hebba.

Funkcja *newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)*

*PR – macierz wejściowa z*

*[S1 S2...SNl] – liczba neuronów w kolejnych warstwach*

*{TF1 TF2...TFNl} – funkcje aktywacji neuronów w kolejnych warstwach ( u mnie tansig czyli tangens hiperboliczny*

*BTF,BLF,PF – funkcje wykorzystywane do treningu sieci*

Funkcja *learnh(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)*

Gdzie w odpowiednie miejsca wstawiłam macierz wejściową, wyjściową i parametry służące regule Hebba.

1. Uczenie sieci dla różnych współczynników uczenia i zapominania.

|  |  |  |  |
| --- | --- | --- | --- |
| EMOTIKON SMILE | | | |
| Wartość współczynnika uczenia / wartość współczynnika zapominania | | | |
| Próba | 0.01 / 0.01 | 0.1 / 0.1 | 0.5 / 0.5 |
| 1 | -0.7096 | 1 | -1 |
| 2 | 1 | 1 | 0.5801 |
| 3 | 0.4374 | 0.5436 | -0.8171 |
| 4 | -1 | -0.8520 | 0.8112 |
| 5 | -0.8830 | 1 | 0.6000 |

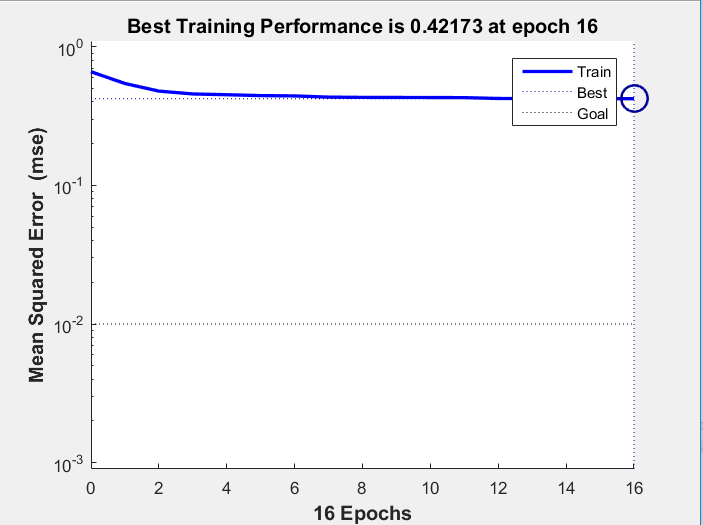
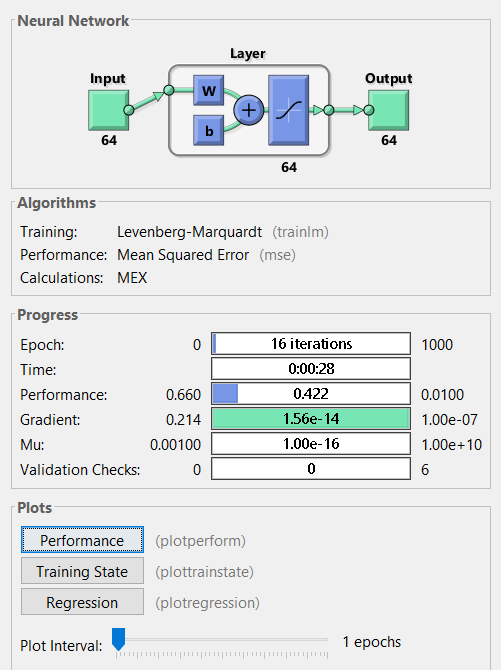
|  |  |  |  |
| --- | --- | --- | --- |
| EMOTIKON SHOCK | | | |
| Wartość współczynnika uczenia / wartość współczynnika zapominania | | | |
| Próba | 0.01 / 0.01 | 0.1 / 0.1 | 0.5 / 0.5 |
| 1 | -0.1986 | 1 | -1 |
| 2 | 1 | -0.4893 | -0.4557 |
| 3 | 0.1491 | 1 | 0.2064 |
| 4 | -1 | 0.9852 | 0.9192 |
| 5 | -0.2091 | 1 | -0.1354 |

|  |  |  |  |
| --- | --- | --- | --- |
| EMOTIKON CONFUSE | | | |
| Wartość współczynnika uczenia / wartość współczynnika zapominania | | | |
| Próba | 0.01 / 0.01 | 0.1 / 0.1 | 0.5 / 0.5 |
| 1 | -0.5272 | 1 | -0.9834 |
| 2 | 1 | 1 | 0.4274 |
| 3 | 0.6064 | 1 | -0.3183 |
| 4 | -1 | 0.7723 | 0.9770 |
| 5 | -0.6268 | 1 | -0.0335 |

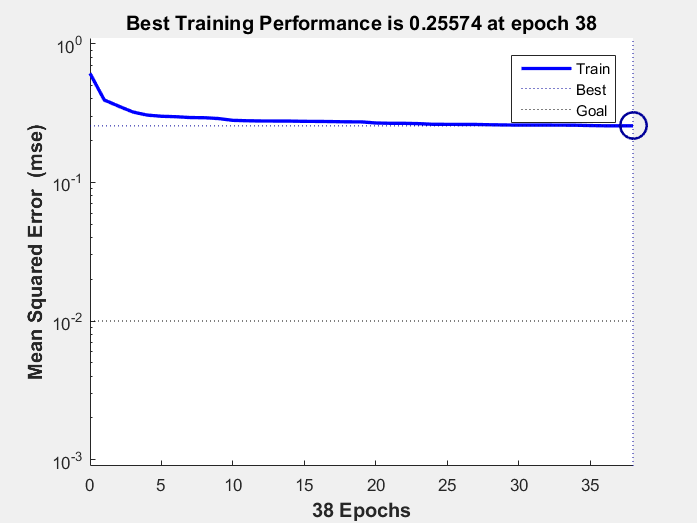
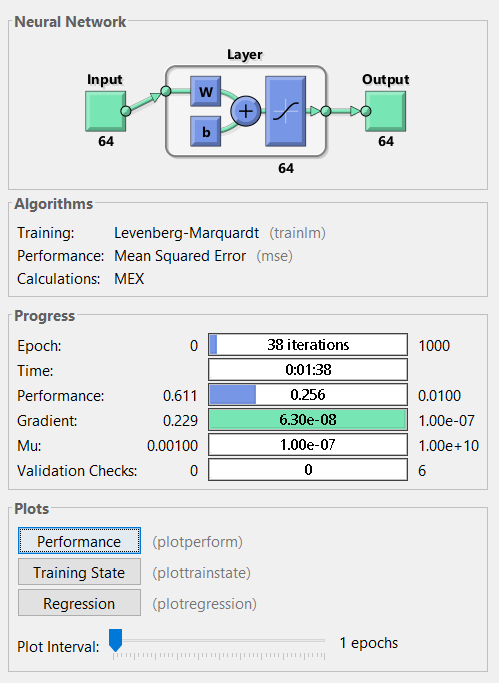
|  |  |  |  |
| --- | --- | --- | --- |
| EMOTIKON SAD | | | |
| Wartość współczynnika uczenia / wartość współczynnika zapominania | | | |
| Próba | 0.01 / 0.01 | 0.1 / 0.1 | 0.5 / 0.5 |
| 1 | 0.5638 | 1 | -1 |
| 2 | 1 | 0.3350 | -0.5065 |
| 3 | 0.5542 | 1 | 0.6986 |
| 4 | -1 | -0.9076 | 0.6985 |
| 5 | 0.5705 | 1 | -0.5098 |

d) Testowanie sieci.

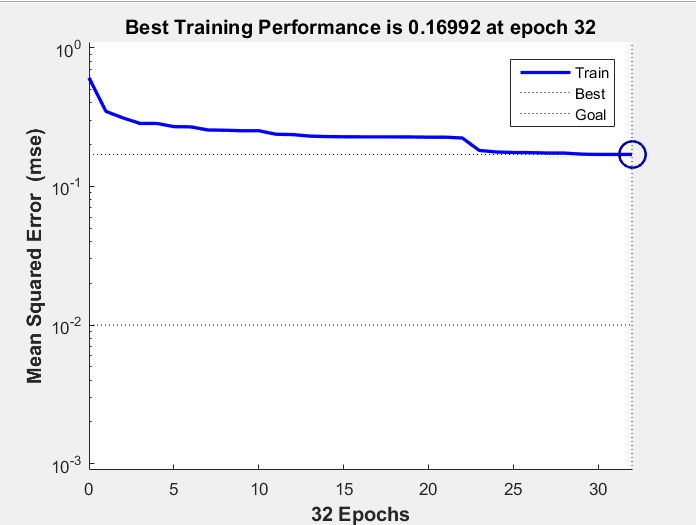
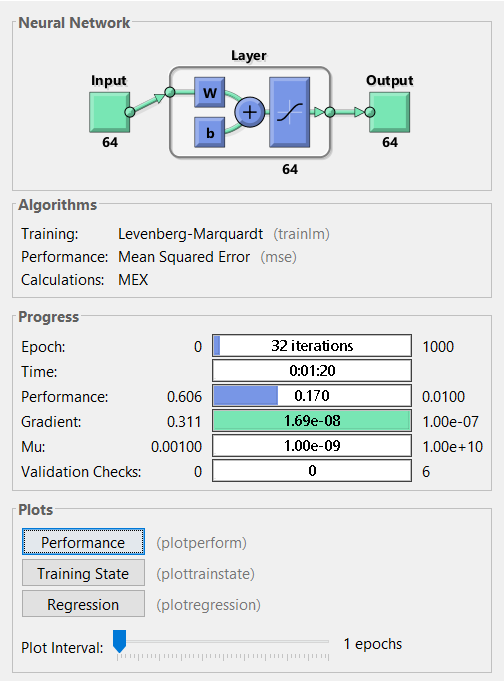
**DLA WARTOŚCI 0.01 / 0.01**

****

**DLA WARTOŚCI 0.1 / 0.1**

****

**DLA WARTOŚCI 0.5 / 0.5**

****

**WNIOSKI:**

- Proces samo uczenia ma wady. W porównaniu z procesem uczenia z nauczycielem samo uczenie jest zwykle znacznie powolniejsze.

- Nie można określić, czy sieć uczona w ten sposób nauczy się wszystkich prezentowanych jej wzorców. Dlatego sieć przeznaczona do samouczenia musi być większa niż sieć wykonująca to samo zadanie, ale trenowana w sposób klasyczny, z udziałem nauczyciela.

- Bardzo istotną kwestią jest wybór początkowych wartości wag neuronów sieci przeznaczonej do samouczenia. Wartości te mają bardzo silny wpływ na ostateczne zachowanie sieci.

- Najlepiej dobranymi parametrami uczenia były wartości zbliżone 0.1. Najwięcej trafień padało gdy trenowaliśmy sieć dla takich wartości.

- Wiązało się też to z najdłuższym procesem uczenia, ponieważ zajmowało to 38 epok, ale wyniki były najdokładniejsze.

- Przy innych wprowadzonych wartościach sieć uczyła się szybciej, ale osiągała słabsze wyniki.

- Ujemne wartości są prawdopodobnie spowodowane faktem, iż neurony nie traktują wprowadzonych do nich wartości jako własne i wyliczają je na minusie.