

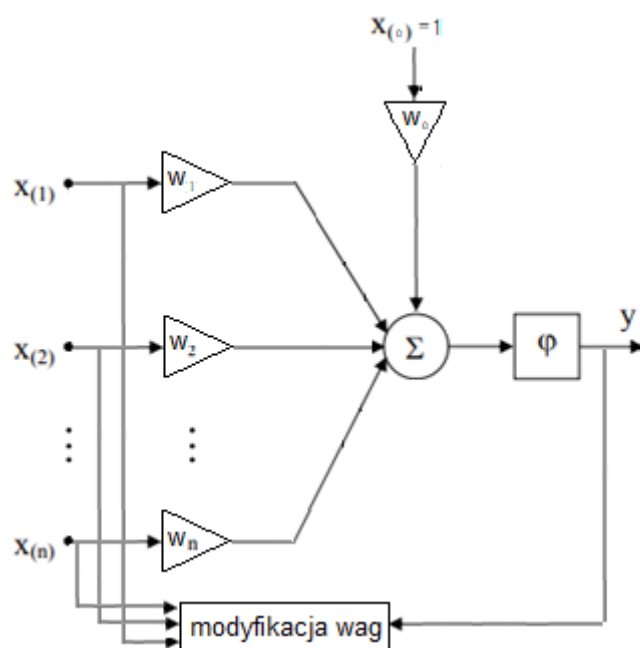
Podstawy sztucznej inteligencji Sprawozdanie ćwiczenie 4 Uczenie sieci regułą Hebba

Cel:

Celem ćwiczenia było poznanie działania reguły Hebba dla sieci jednowarstwowej na przykładzie grupowania liter alfabetu.

Wiedza:

W zadaniu wykorzystałam jednowarstwową sieć, składającą się z 20 neuronów i uczyłam ją zgodnie z regułą Hebba.



Model neuronu Hebba.

Reguła Hebba

Oparta na tworzeniu odruchów warunkowych zasada. *Jeśli aktywny neuron A jest cyklicznie pobudzany przez neuron B, to staje się on jeszcze bardziej czuły na pobudzenie tego neuronu.* Reguła zapisana w postaci równania:

$$w_{AB}(k+1) = w_{AB}(k) + \alpha x_A(k)x_B(k)$$

x_A, x_B – stany aktywacji neuronów A i B

w_{AB} – waga ich połączenia synaptycznego

α - stała dodatnia, sterująca procesem uczenia

Dodatkowo:

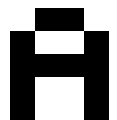
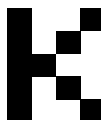
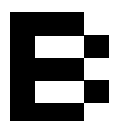

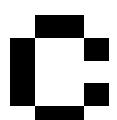
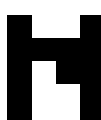
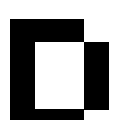

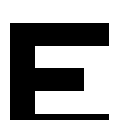

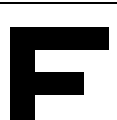

- Jeżeli neurony A i B połączone synapsą **są** pobudzone jednocześnie, (czyli są pobudzone **synchronicznie**), to połączenie synaptyczne między nimi jest **wzmacniane**.
- Jeżeli neurony A i B połączone synapsą **nie są** pobudzone jednocześnie, (czyli pobudzone są **asynchronicznie**), to połączenie pomiędzy nimi jest **osłabiane**.

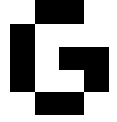
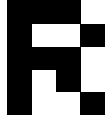
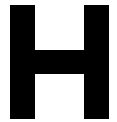
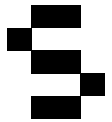
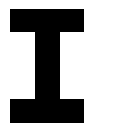

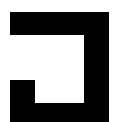
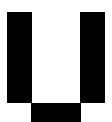
Reguła Hebba polega na pokazywaniu sieci neuronowej kolejnych przykładów sygnałów wejściowych nie podając informacji o tym, co należy zrobić z tymi sygnałami, nie zostaje określone znaczenie podanych obiektów ani zależności pomiędzy nimi (uczenie bez nauczyciela). Sieć obserwując odbierane sygnały stopniowo odkrywa znaczenie oraz zależności. Każdy zestaw sygnałów wejściowych jest przydzielany do rozkładu sygnałów wejściowych, w których niektóre neurony sieci są bardzo silnie pobudzone, a niektóre nawet ujemnie.

Uczenie bez nauczyciela jest znacznie wolniejsze. Dodatkowo nie daje pewności, czy sieć nauczy się poprawnie i wszystkich prezentowanych wzorców. Dlatego trenowanie konieczne jest z udziałem nauczyciela. Proces takiego uczenia nigdy nie zakończy się samodzielnie.

Zadania, które wykonałam w ramach ćwiczenia:

- Utworzyłam dane uczące i testujące, zawierające 20 dużych kolejnych liter alfabetu łacińskiego w postaci tablicy 4x5 pikseli dla jednej litery

	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	0	1	1	0	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	1	0	0	1
0	1	1	0																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	1	0																																								
1	1	0	0																																								
1	0	1	0																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	1	0	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	0		<table><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	1	1	1																																								
	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	1	0	1	1	0		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	1	1	1	1	0	1	1	1	0	0	1	1	0	0	1
0	1	1	0																																								
1	0	0	1																																								
1	0	0	0																																								
1	0	0	1																																								
0	1	1	0																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1	0		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1	1	1	0	0	1
1	1	1	0																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	1																																								
1	0	0	1																																								
1	1	0	1																																								
1	0	1	1																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	0	0	0	1	1	1	0	1	0	0	0	1	1	1	1		<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0	1	1	0
1	1	1	1																																								
1	0	0	0																																								
1	1	1	0																																								
1	0	0	0																																								
1	1	1	1																																								
0	1	1	0																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
0	1	1	0																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr></table>	1	1	1	1	1	0	0	0	1	1	1	0	1	0	0	0	1	0	0	0		<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr></table>	1	1	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	0
1	1	1	1																																								
1	0	0	0																																								
1	1	1	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	0																																								
1	0	0	0																																								

	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0		<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	1	1	0	1	0	0	1	1	1	1	0	1	0	1	0	1	0	0	1
0	1	1	0																																								
1	0	0	0																																								
1	0	1	1																																								
1	0	0	1																																								
0	1	1	0																																								
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	1	0																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1		<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0
1	0	0	1																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
0	1	1	0																																								
1	0	0	0																																								
0	1	1	0																																								
0	0	0	1																																								
0	1	1	0																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	1	0		<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	1	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
1	1	1	0																																								
0	1	0	0																																								
0	1	0	0																																								
0	1	0	0																																								
1	1	1	0																																								
1	1	1	0																																								
0	1	0	0																																								
0	1	0	0																																								
0	1	0	0																																								
0	1	0	0																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	1		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	1	0
1	1	1	1																																								
0	0	0	1																																								
0	0	0	1																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
0	1	1	0																																								

Tablice z graficznym i binarnym przedstawieniem danych uczących.

Tablice składające się z 20 pól, czarnych - 1 lub białych - 0.

Czarne pole oznacza, że w danym miejscu występuje element litery.

- Implementacja

- Zainicjowałam zmienną *start*, która reprezentuje maksymalną – 1 oraz minimalną – 0 wartości elementów wejściowych dla funkcji tworzących sieć neuronową.
- Ustawiłam liczbę elementów wektora wyjściowego sieci, czyli ilość neuronów w zmiennej *wyjścia_s* = 20.
- Wprowadziłam dane uczące (20 dużych liter) w postaci binarnej zapisanej kolumnowo do zmiennej *WEJSCIE*.
- Wygenerowałam sieć neuronową przy użyciu narzędzi z biblioteki Matlaba *Neural Network Toolbox*.
- Zainicjowałam zmienną *WYJSCIE*, której przypisywane były wyniki trafienia żądanej litery.
- Użyte metody i zmienne w implementacji:
 - funkcja *newff(start, wyjścia_s, {'tansig'}, 'trainlm', 'learnh')*, która tworzy sieć neuronową
 - zmienna *start*, która reprezentuje maksymalną oraz minimalną ilość wejść do sieci.
 - zmienna *wyjścia_s* z liczbą elementów wektora wyjściowego
 - paramet *tansig*, określający funkcję aktywacji – tangens hiperboliczny, użyty trzykrotnie, do stworzenia trójwarstwowej sieci
 - funkcja *trainlm*, która analizuje wartości wag oraz odchylenia
 - *learnh*, jako waga uczenia dla reguły Hebb'a
 - zmienna *net*, do której przypisywana jest nowo tworzona sieć
 - *net.trainParam.x* – ustawianie wartości parametrów treningu
 - *lp.dr* – wartość współczynnika zapominania dla reguły Hebb'a
 - *lp.lr* – wartość współczynnika uczenia dla reguły Hebb'a
 - dane uczące *WEJSCIE*, dane wyjściowe odpowiadające danym uczącym (1 – trafienie, 0 – błąd) *WYJSCIE*
 - *wagiHebba* – zmienna przechowująca wartości wag dla reguły Hebb'a za pomocą funkcji *learnh*

- funkcja *train(net, WEJSCIE, wagiHebba)*, do uczenia sieci z wykorzystaniem danych wejściowych i określonych wcześniej wag dla reguły Hebba
 - zmienna *efekt_hebba* – służąca do wypisania przykładowego działania reguły Hebba na zdefiniowanych wcześniej wartościach wag
 - zmienna *efekt* – zmienna służąca do wypisania wartości symulacji.
- Przeprowadziłam proces uczenia i przetestowałam działanie programu. Wyniki zamieściłam w tabelach poniżej.

wartość współczynnika uczenia	0,001	0,01	0,1
A	0,012	0,12	1,2
B	0,013	0,13	1,3
C	0,009	0,09	0,9
D	0,012	0,12	1,2
E	0,013	0,13	1,3
F	0,010	0,10	1,0
G	0,010	0,10	1,0
H	0,012	0,12	1,2
I	0,009	0,09	0,9
J	0,011	0,11	1,1
K	0,010	0,10	1,0
L	0,008	0,08	0,8
M	0,013	0,13	1,3
N	0,012	0,12	1,2
O	0,010	0,10	1,0
P	0,010	0,10	1,0
R	0,012	0,12	1,2
S	0,008	0,08	0,8
T	0,007	0,07	0,7
U	0,010	0,10	1,0
liczba epok	10	12	23

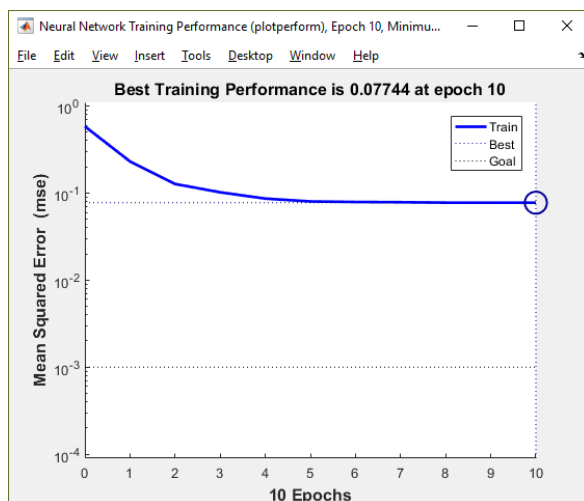
Tabela wyników dla pojedynczego wywołania reguły Hebba

wartość współczynnika uczenia	0,001	0,01	0,1
A	0,0000	0,0004	0,0000
B	0,0011	0,0106	1,0000
C	0,0000	0,0038	0,0000
D	0,0100	0,0100	0,1040
E	-1,0000	0,0050	0,0470
F	0,0010	0,0074	0,0744
G	0,0001	0,0010	0,0101
H	0,0008	-1,0000	0,0787
I	0,0007	0,0066	0,0663
J	0,0008	0,0079	0,0787
K	0,0007	0,0070	0,0705
L	0,0004	0,0050	0,0449
M	0,0008	0,0077	0,0766
N	0,0010	0,0098	0,0980
O	0,0004	0,0036	0,0356
P	0,0010	0,0098	0,0980
R	0,0010	0,0090	0,1000
S	0,0001	0,0012	0,0121
T	0,0008	0,0070	0,0703
U	0,0010	0,0110	0,1102
liczba epok	10	12	23

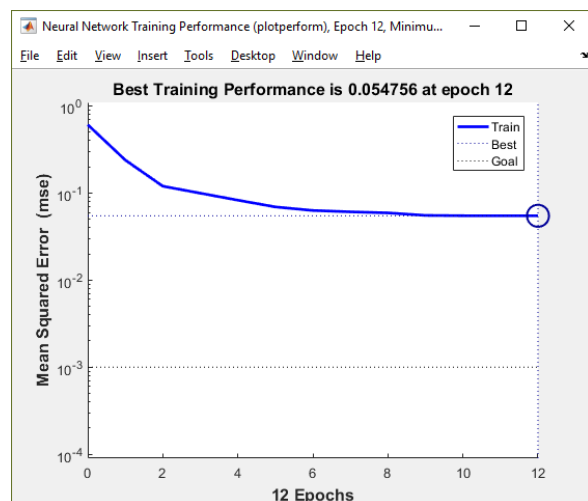
Tabela wyników dla wykonania programu regułą Hebba, dla różnych wag początkowych

Zaznaczone przeze mnie wartości są przykładem pogrupowania danych przez regułę Hebba. Pogrupowane litery w każdym przypadku w danej grupie otrzymywały taką samą lub bliską wartość.

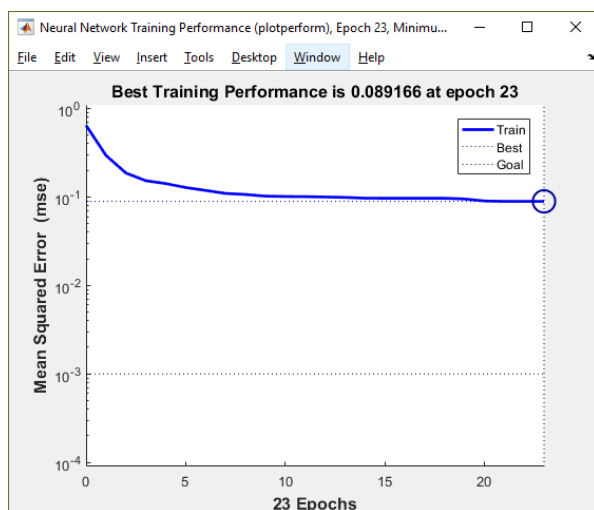
Wykresy najlepszej wydajności:



dla współczynnika uczenia 0,001



dla współczynnika uczenia 0,01



dla współczynnika uczenia 0,1

Wnioski:

Wysoki współczynnik uczenia skutkował szybszym wzrostem wartości wag, natomiast z tabel powyżej można wywnioskować, że wartości wag wpływają na otrzymywane wyniki. Również im współczynnik uczenia miał wyższą wartość, tym algorytm otrzymywał dokładniejsze wyniki, ale liczba epok potrzebnych do uczenia sieci także wzrastała.

Na wykresie wydajności uczenia można zauważyć, że w początkowej fazie treningu mogą najczęściej występować błędy. A także można zauważyć graniczną wartość, w której nie następuje już większy progres uczenia - w okolicach 5 epoki.

Użyta metoda nie daje gwarancji poprawności danych. W porównaniu z wcześniejszymi algorytmami reguła Hebb'a potrzebuje również więcej czasu na naukę. Jest to spowodowane przede wszystkim tym, że jest to model uczenia bez nauczyciela. Przez to algorytm sam musi decydować o poprawności efektów w oparciu o dane wejściowe oraz uczenie trwa dłużej. Jest to algorytm najbardziej zbliżony do naturalnej metody uczenia.

Listing kodu programu wraz z komentarzami:

```
close all; clear all; clc;

%wejścia do sieci oraz minimalne oraz maksymalne wartości wejść
%(20 par 0&1 - osobno dla każdej z danych uczących)
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;];

%ilość wyjść z sieci (jedna warstwa - 20 neuronów na wyjściu)
wyjścia_s = 20;

%użycie funkcji newff
net = newff(start, wyjścia_s, {'tansig'}, 'trainlm', 'learnh');

%kolumnowa reprezentacja binarna pierwszych 20 dużych liter dla tablicy 4x5
%A B C D E F G H I J K L M N O P R S T U
WEJSCIE = [0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1;
           1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0;
           1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0;
           0 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1;
           1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1;
           0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0;
           0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0;
           1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 0;
           1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0;
           1 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0;
           1 1 0 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 0;
           1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 0 0 1;
           1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1;
           0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0;
           0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0;
           1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 1;
           1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 0 0;
           0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 1;
           0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1;
           1 0 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0];

%zmienna, która reprezentuje, czy użytkownik "trafił" w wybraną przez
%siebie literę - 1 oznacza trafienie, 0 - chybienie
WYJSCIE = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %A
           0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %B
           0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %C
           0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %D
           0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %E
           0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %F
           0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0; %G
           0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0; %H
           0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0; %I
           0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0; %J
           0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0; %K
           0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0; %L
           0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0; %M
           0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0; %N
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0; %O
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0; %P
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0; %R
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0; %S
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0; %T
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]; %U
%A B C D E F G H I J K L M N O P R S T U
```

```

%PARAMETRY REGULY HEBBA
lp.dr = 0.5;           %wspolczynnik zapominania
lp.lr = 0.1;           %wspolczynnik uczenia

%dostosowanie parametrów sieci do metody Hebba
wagiHebba = learnh([0],WEJSCIE,[0],[0],WYJSCIE,[0],[0],[0],[0],[0],lp,[0]);

%PARAMETRY TRENINGU SIECI:
net.trainParam.epochs = 1000;   %maksymalna ilosc epok
net.trainParam.goal = 0.001;    %cel wydajnosci sieci
net.trainParam.lr=0.5;          %wskaznik uczenia sieci

net = train(net, WEJSCIE, wagiHebba);

%dane testowe
A_testowe = [0; 1; 1; 0;
             1; 0; 0; 1;
             1; 1; 1; 1;
             1; 0; 0; 1;
             1; 0; 0; 1];
B_testowe = [1; 1; 1; 0;
             1; 0; 0; 1;
             1; 1; 1; 0;
             1; 0; 0; 1;
             1; 1; 1; 0];
C_testowe = [0; 1; 1; 0;
             1; 0; 0; 1;
             1; 0; 0; 0;
             1; 0; 0; 1;
             0; 1; 1; 0];
D_testowe = [1; 1; 1; 0;
             1; 0; 0; 1;
             1; 0; 0; 1;
             1; 0; 0; 1;
             1; 1; 1; 0];
E_testowe = [1; 1; 1; 1;
             1; 0; 0; 0;
             1; 1; 1; 0;
             1; 0; 0; 0;
             1; 1; 1; 1];
F_testowe = [1; 1; 1; 1;
             1; 0; 0; 0;
             1; 1; 1; 0;
             1; 0; 0; 0;
             1; 0; 0; 0];
G_testowe = [0; 1; 1; 0;
             1; 0; 0; 0;
             1; 0; 1; 1;
             1; 0; 0; 1;
             0; 1; 1; 0];
H_testowe = [1; 0; 0; 1;
             1; 0; 0; 1;
             1; 1; 1; 1;
             1; 0; 0; 1;
             1; 0; 0; 1];
I_testowe = [1; 1; 1; 0;
             0; 1; 0; 0;
             0; 1; 0; 0;
             0; 1; 0; 0;
             1; 1; 1; 0];
J_testowe = [1; 1; 1; 1;
             0; 0; 0; 1;
             0; 0; 0; 1;
             1; 0; 0; 1;
             0; 1; 1; 1];
K_testowe = [1; 0; 0; 1;

```

```

        1; 0; 1; 0;
        1; 1; 0; 0;
        1; 0; 1; 0;
        1; 0; 0; 1];
L_testowe = [1; 0; 0; 0;
             1; 0; 0; 0;
             1; 0; 0; 0;
             1; 0; 0; 0;
             1; 1; 1; 1];
M_testowe = [1; 0; 0; 1;
             1; 1; 1; 1;
             1; 0; 1; 1;
             1; 0; 0; 1;
             1; 0; 0; 1];
N_testowe = [1; 0; 0; 1;
             1; 0; 0; 1;
             1; 1; 0; 1;
             1; 0; 1; 1;
             1; 0; 0; 1];
O_testowe = [0; 1; 1; 0;
             1; 0; 0; 1;
             1; 0; 0; 1;
             1; 0; 0; 1;
             0; 1; 1; 0];
P_testowe = [1; 1; 1; 0;
             1; 0; 0; 1;
             1; 1; 1; 0;
             1; 0; 0; 0;
             1; 0; 0; 0];
R_testowe = [1; 1; 1; 0;
             1; 0; 0; 1;
             1; 1; 1; 0;
             1; 0; 1; 0;
             1; 0; 0; 1];
S_testowe = [0; 1; 1; 0;
             1; 0; 0; 0;
             0; 1; 1; 0;
             0; 0; 0; 1;
             0; 1; 1; 0];
T_testowe = [1; 1; 1; 0;
             0; 1; 0; 0;
             0; 1; 0; 0;
             0; 1; 0; 0;
             0; 1; 0; 0];
U_testowe = [1; 0; 0; 1;
             1; 0; 0; 1;
             1; 0; 0; 1;
             1; 0; 0; 1;
             0; 1; 1; 0];

```

```

efekt_Hebba = wagiHebba;

```

```

%symulacja sieci net
efekt= sim(net, A_testowe);

```

```

%wypisywanie wartosci reguly Hebba, wypisywanie kolejnych wierszy
disp('Jednokrotne wykorzystanie reguly Hebba: ')
disp('A = '), disp(sum(efekt_Hebba (1, ':')));
disp('B = '), disp(sum(efekt_Hebba (2, ':')));
disp('C = '), disp(sum(efekt_Hebba (3, ':')));
disp('D = '), disp(sum(efekt_Hebba (4, ':')));
disp('E = '), disp(sum(efekt_Hebba (5, ':')));
disp('F = '), disp(sum(efekt_Hebba (6, ':')));
disp('G = '), disp(sum(efekt_Hebba (7, ':')));
disp('H = '), disp(sum(efekt_Hebba (8, ':')));
disp('I = '), disp(sum(efekt_Hebba (9, ':')));
disp('J = '), disp(sum(efekt_Hebba (10, ':')));
disp('K = '), disp(sum(efekt_Hebba (11, ':')));

```



```
disp('L = '), disp(sum(efekt_Hebba (12, ':')));  
disp('M = '), disp(sum(efekt_Hebba (13, ':')));  
disp('N = '), disp(sum(efekt_Hebba (14, ':')));  
disp('O = '), disp(sum(efekt_Hebba (15, ':')));  
disp('P = '), disp(sum(efekt_Hebba (16, ':')));  
disp('R = '), disp(sum(efekt_Hebba (17, ':')));  
disp('S = '), disp(sum(efekt_Hebba (18, ':')));  
disp('T = '), disp(sum(efekt_Hebba (19, ':')));  
disp('U = '), disp(sum(efekt_Hebba (20, ':')));
```

%wypisywanie wartosci dla poszczegolnych liter

```
disp('Dzialanie algorytmu z wykorzystaniem reguly Hebba dla wszystkich liter: ')  
disp('A = '), disp(efekt(1));  
disp('B = '), disp(efekt(2));  
disp('C = '), disp(efekt(3));  
disp('D = '), disp(efekt(4));  
disp('E = '), disp(efekt(5));  
disp('F = '), disp(efekt(6));  
disp('G = '), disp(efekt(7));  
disp('H = '), disp(efekt(8));  
disp('I = '), disp(efekt(9));  
disp('J = '), disp(efekt(10));  
disp('K = '), disp(efekt(11));  
disp('L = '), disp(efekt(12));  
disp('M = '), disp(efekt(13));  
disp('N = '), disp(efekt(14));  
disp('O = '), disp(efekt(15));  
disp('P = '), disp(efekt(16));  
disp('R = '), disp(efekt(17));  
disp('S = '), disp(efekt(18));  
disp('T = '), disp(efekt(19));  
disp('U = '), disp(efekt(20));
```