Maciej Krzywda,
Inżynieria Obliczeniowa, IMiIP
Podstawy Sztucznej Inteligencji
nr albumu: 293102

Sprawozdanie 2

Tytuł projektu:

Budowa i działanie sieci jednowarstwowej

- **Cel projektu:**

Celem ćwiczenia jest poznanie budowy i działania jednowarstwowych sieci neuronowych oraz uczenie rozpoznawania wielkości liter.

Zadanie jakie mieliśmy wykonać było oparte o implementację sieci jednowarstwowej w środowisku MATLAB.

- **Przebieg ćwiczenia**

1. **Wygenerowanie danych uczących i testujących, zawierających 10 dużych i 10 małych liter dowolnie wybranego alfabetu w postaci dwuwymiarowej tablicy .**

2. **Przygotowanie dwóch jednowarstwowych sieci - każda według. innego algorytmu**

3. **Uczenie sieci dla przy różnych współczynnikach uczenia.**

4. **Testowanie sieci**

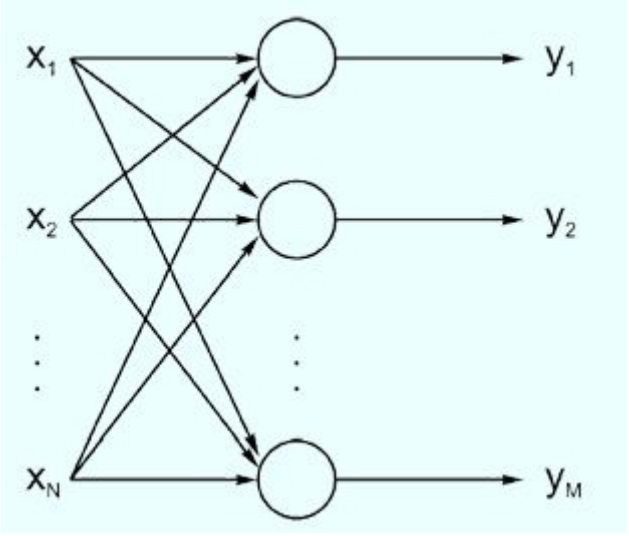
● Część Teoretyczna

Sieć neuronowa – model matematyczny inspirowany budową naturalnych neuronów znajdujących się w mózgu człowieka. Modele takie składają się z neuronów które to realizują podstawowe obliczenia na swoim wejściu i nazywamy je sztucznymi neuronami.

Systemy oparte o sieci neuronowe wykazują zdolności uczenia się, na podstawie przykładów uczymy taką sieć aby następnie ona już samodzielnie analizowała nowe dane przedstawione w innym lecz analogicznym kontekście.

Sieć neuronowa spisuje się świetnie w zastosowaniach przy których klasyczny komputer nie mógłby sobie poradzić tzn. Uogólnienie posiadanej wiedzy na inne przypadki włącznie z wymyśleniem nowego sposobu rozwiązania nieznanego jej dotychczas problemu, odporność na uszkodzenia – niepełne, błędne dane, sprawne działanie nawet gdy część jej elementów nie działa poprawnie lub nie działa w ogóle. Sieci neuronowe nadają się idealnie do rozwiązywania problemów przy których klasyczny język programowania nie daje sobie rady tzn. brakuje odpowiedniego algorytmu.

Sieci jednokierunkowe jednowarstwowe – w sieciach tego typu neurony ułożone są w jednej warstwie, która jest zasilana z węzłów wejściowych. Przepływ sygnału w tego typu sieciach przebiega zawsze w ściśle określonym kierunku: od warstwy wejściowej do warstwy wyjściowej.
Na węzłach wchodzących nie znajdują się warstwy neuronów, gdyż nie zachodzi w nich żaden proces obliczeniowy.
Dobór wag następuje tu w procesie uczenia sieci, czyli dopasowania sygnałów wyjściowych yi do wartości, której oczekujemy di .



● Część Praktyczna

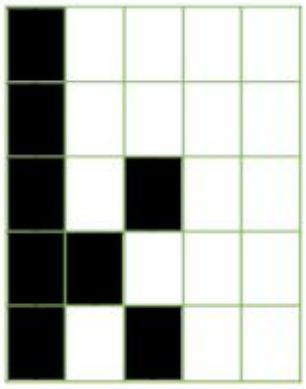
NET = NEWLIN(PR,S,ID,LF)

PR - macierz o wymiarach Rx2, gdzie R jest liczbą wejść sieci (tj. liczbą współrzędnych wektorów wejściowych); pierwsza kolumna zawiera minimalne wartości kolejnych współrzędnych wektorów wejściowych, druga kolumna – maksymalne wartości tych współrzędnych
S - liczba neuronów sieci, tj. wyjść sieci (neurony tworzą automatycznie warstwę wyjściową)
ID - wektor opóźnień poszczególnych elementów wektora wejść sieci; domyślnie ID = [0]
LR - stała szybkości uczenia (treningu) sieci liniowej; domyślnie LR = 0.01
NET - struktura (obiekt) zawierająca opis architektury, metod treningu, wartości liczbowe wag oraz inne parametry sieci perceptronowej.

NET = NEWP(PR,S,TF,LF)

PR - macierz o wymiarach Rx2, gdzie R jest liczbą wejść sieci (liczbą współrzędnych wektorów wejściowych); pierwsza kolumna macierzy R zawiera minimalne wartości kolejnych współrzędnych wektorów wejściowych, druga kolumna – maksymalne wartości tych współrzędnych
S - liczba neuronów sieci
TF - nazwa funkcji aktywacji neuronów (zmienna tekstowa); nazwa domyślna = 'hardlm'; dopuszczalne wartości parametru TF to: 'hardlim' i 'hardlims'
LF - nazwa funkcji trenowania sieci perceptronowej (zmienna tekstowa); nazwa domyślna = 'learnp'; dopuszczalne wartości parametru LF to: 'learnp' i 'learnpn'
NET - struktura (obiekt) zawierająca opis architektury, metod treningu, wartości liczbowe wag i progów oraz inne parametry sieci perceptronowej.

Przykładowy zapis litery k (małe k) na matrycy 5x5:



1	0	0	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	0	1	0	0

Przebieg ćwiczenia

Projekt został zrealizowany za pomocą pakietu MATLAB. Zadaniem było zaimplementowanie jednowarstwowej sieci neuronowej.

Zgodnie z definicją jednowarstwowej sieci neuronowej możemy skorzystać z dwóch modeli – pierwszym z nich jest tworzony w poprzednim projekcie (za pomocą omówionej funkcji newp) .

Drugim sposobem jest utworzenie jednowarstwowej sieci za pomocą funkcji newlin z biblioteki Neural Network Toolbox.

Pierwszym krokiem było wygenerowanie tablicy wejściowej zawierającej 10 małych i 10 wielkich liter (Litery A, B, C, D, E, H, I, K, L, F).

Litery te zostały przedstawione w formie tablicy 5x5.

Następnie taki obraz „zbinaryzowany" (polom białym przypisano wartość 0, a polom zamalowanym wartość 1).

Następnie z przypisanych wartości utworzono ciąg binarny.

- **Listing programu z opisem poszczególnych kroków.**

```
close all; clear all; clc;
% Wartości które przyjmuje węzeł na wejściu
input_value = [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
output_value = 1;

%metoda 1 - tworzenie perceptronu
%net = newp(input_value, output_value);

%metoda 2 - tworzenie sieci jednowarstwowej
net = newlin(input_value, output_value);

 % Reprezentacja kolumnowa liter
input_learn = [0 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1;
 1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0;
 1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0;
 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0;
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 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 1 0 0;
 1 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0;
 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0;
 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1;
 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0;
 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0;
 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0;
 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
 1 1 1 1 1 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 0 0 0 0 1 0 0;
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
 1 1 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0;
 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1;
 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1;
 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1;
 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 0 1 0;
```

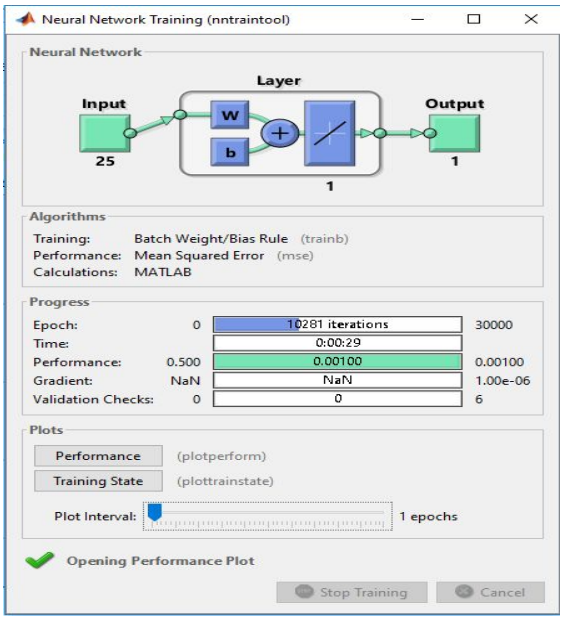
```
 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0];

% 1 - duża litera, 0 - mała litera
output_learn = [1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];
net.name = 'Wielkosc liter';
net.trainParam.epochs = 30000;

%błąd średniokwadratowy
net.trainParam.goal = 0.001;
net.trainParam.mu = 0.01;

%uczenie sieci
net = train(net, input_learn, output_learn);

%litery do testu
test_A = [0;1;1;1;0;
 1;0;0;0;1;
 1;0;0;0;1;
 1;1;1;1;1;
 1;0;0;0;1];
test_a = [0;1;1;0;0;
 0;0;0;1;0;
 0;1;1;1;0;
 1;0;0;1;0;
 0;1;1;1;1];
test_B = [1;1;1;0;0;
 1;0;0;1;0;
 1;1;1;0;0;
 1;0;0;1;0;
 1;1;1;0;0];
test_b = [1;0;0;0;0;
 1;0;0;0;0;
 1;1;1;0;0;
 1;0;0;1;0;
 1;1;1;0;0];
test_C = [1;1;1;1;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;1;1;1;0];
test_c = [0;0;0;0;0;
 0;0;0;0;0;
 0;1;1;0;0;
 1;0;0;0;0;
 0;1;1;0;0];
test_D = [1;1;1;0;0;
 1;0;0;1;0;
 1;0;0;1;0;
 1;0;0;1;0;
 1;1;1;0;0];
```

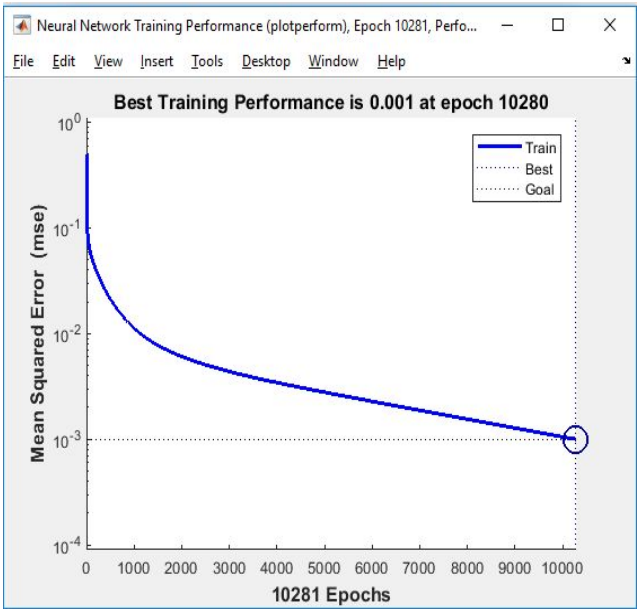
```
test_d = [0;0;0;1;0;
 0;0;0;1;0;
 0;1;1;1;0;
 1;0;0;1;0;
 0;1;1;1;0];
test_E = [1;1;1;1;0;
 1;0;0;0;0;
 1;1;1;0;0;
 1;0;0;0;0;
 1;1;1;1;0];
test_e = [0;1;1;0;0;
 1;0;0;1;0;
 1;1;1;0;0;
 1;0;0;0;0;
 0;1;1;0;0];
test_F = [1;1;1;1;0;
 1;0;0;0;0;
 1;1;1;0;0;
 1;0;0;0;0;
 1;0;0;0;0];
test_f = [0;1;1;0;0;
 1;0;0;0;0;
 1;1;1;0;0;
 1;0;0;0;0;
 1;0;0;0;0];
test_H = [1;0;0;0;1;
 1;0;0;0;1;
 1;1;1;1;1;
 1;0;0;0;1;
 1;0;0;0;1];
test_h = [1;0;0;0;0;
 1;0;0;0;0;
 1;1;1;1;0;
 1;0;0;1;0;
 1;0;0;1;0];
test_I = [1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0];
test_i = [1;0;0;0;0;
 0;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0];
test_K = [1;0;0;1;0;
 1;0;1;0;0;
 1;1;0;0;0;
 1;0;1;0;0;
 1;0;0;1;0];
```

```
test_k = [1;0;0;0;0;
 1;0;0;0;0;
 1;0;1;0;0;
 1;1;0;0;0;
 1;0;1;0;0];
test_L = [1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;1;1;1;0];
test_l = [1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;0;0;0;0;
 1;1;1;0;0];

%test sieci
efekt1 = sim(net, test_A);
efekt2 = sim(net, test_a);

[efekt1 efekt2]
if round(efekt1) <= 0
 disp('Mała litera');
else
 disp('Wielka litera');
end
if round(efekt2) <= 0
 disp('Mała litera');
else
 disp('Wielka litera');
end
```

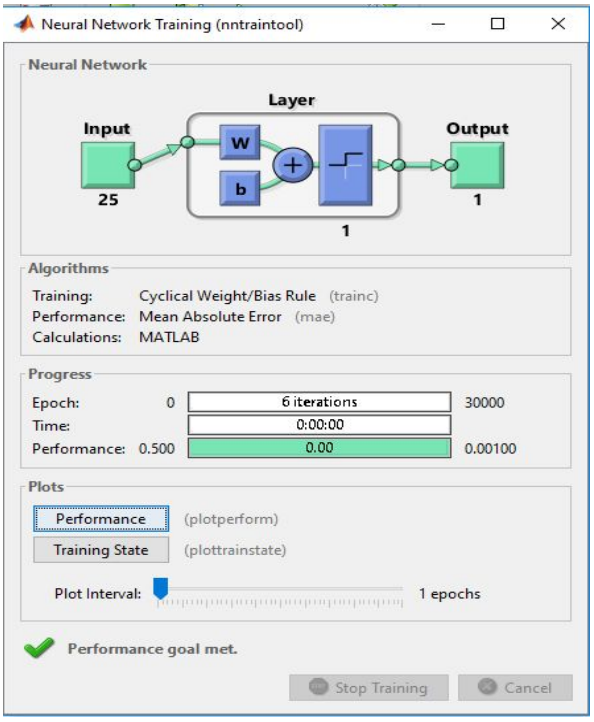
Poniżej prezentacja wyników dla powyższego kodu:

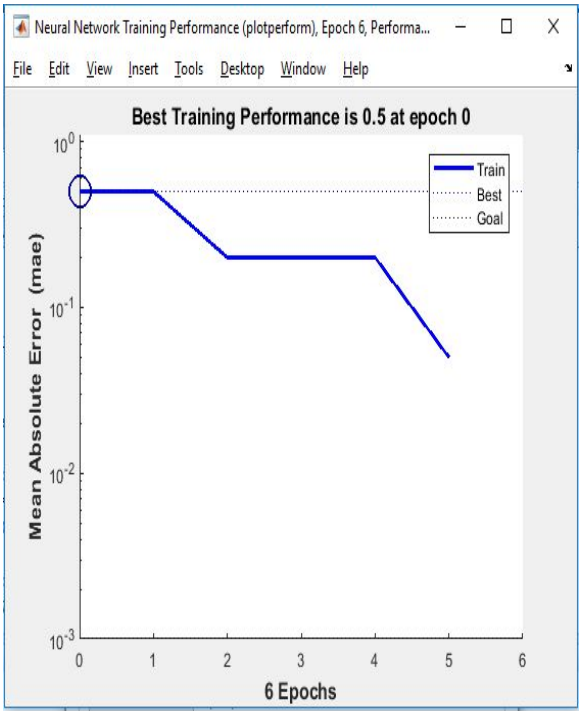
`Output dla newlin()`





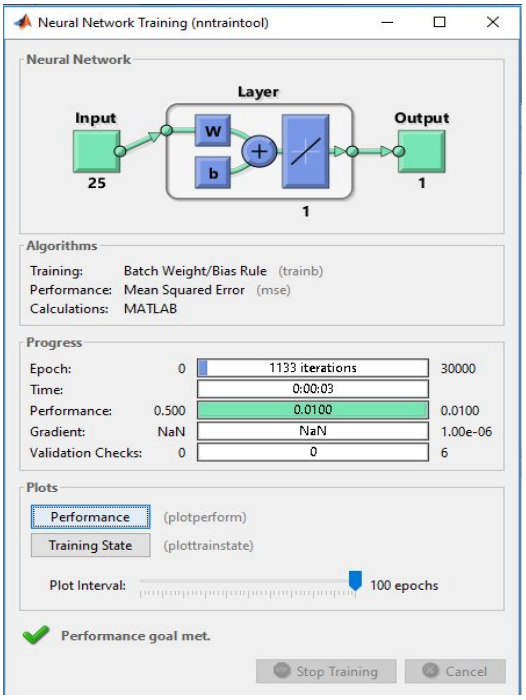
`Dla porównania z siecią jednowarstwowa prezentuje wynik uzyskany dla perceptronu:`

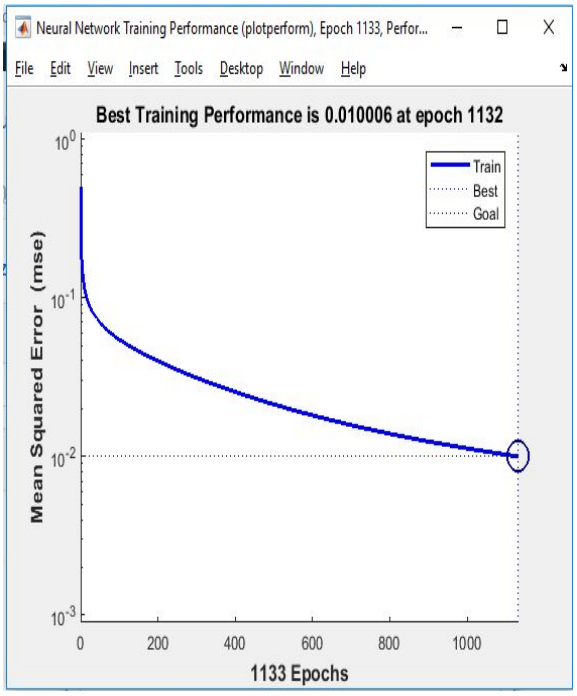




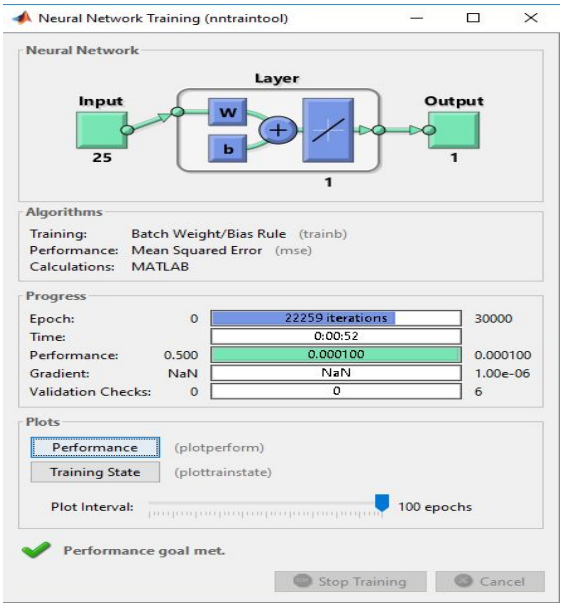
- **Uczenie sieci dla różnych wartości błędu średniokwadratowego.**

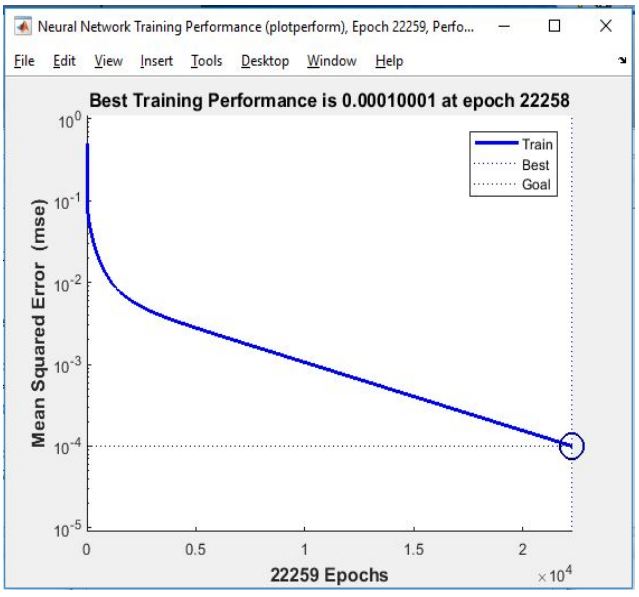
```
%błąd średniokwadratowy
net.trainParam.goal = 0.01;
```



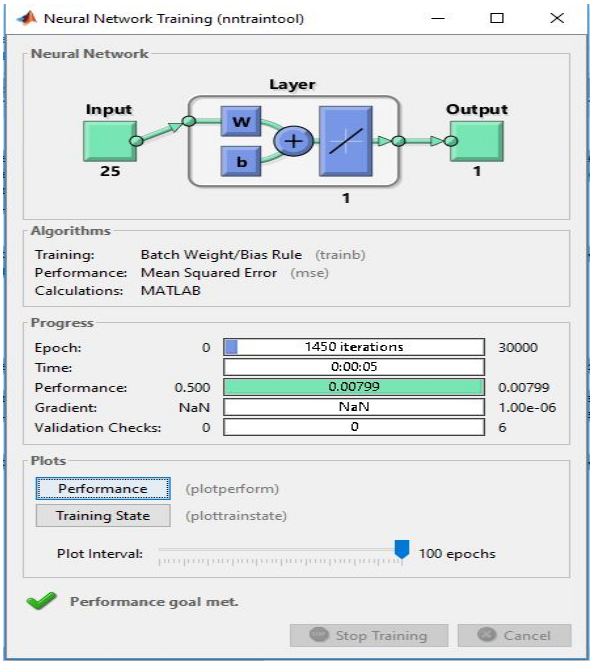


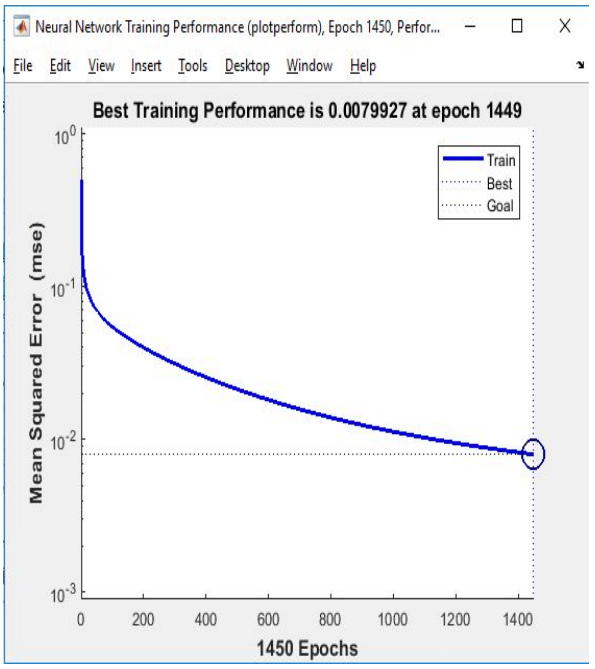

```
%błąd średniokwadratowy
net.trainParam.goal = 0.0001;
```





```
%błąd średniokwadratowy
net.trainParam.goal = 0.00799;
```





Dla różnych współczynników uczenia, ale dla tej samej wartości błędu średniokwadratowego przeprowadziłem pomiary które umieściłem w tabeli. Jeżeli wartość >0.5 -> Duża litera natomiast gdy <0.5 -> Mała litera

Lp.	Litera	Wsp. Uczenia=0.1		Wsp. Uczenia=0.01		Wsp. Uczenia=0.001	
		newlin	newp	newlin	newp	newlin	newp
1	A	0.028	1	0.028	1	0.028	1
2	a	0.004	0	0.004	0	0.004	0
3	K	0.2144	1	0.2144	1	0.2144	1
4	k	0.77	0	0.77	0	0.77	0
5	F	1.0014	1	1.0014	1	1.0014	1
6	f	0.00354	0	0.00354	0	0.00354	0
7	B	1.022	1	1.022	1	1.022	1
8	b	0.039	0	0.039	0	0.039	0
9	C	0.950	1	0.950	1	0.950	1
10	c	0.020	0	0.020	0	0.020	0

Wnioski

W sprawozdaniu zdefiniowałem, że 1 oznacza literę dużą a 0 literę małą, „newp” prawie w każdym przypadku wskazuje liczbę całkowitą, więc nauka przebiega sprawnie i z pozytywnym skutkiem.

Ilość iteracji (epochs) potrzebnych do nauki wynosi 0.5. Program zwraca dokładnie wartość jeden lub zero, w zależności od wyniku (0 - litera mała, 1 - litera duża).
Program z wykorzystaniem metody newp() jest szybki, co łatwo zauważyć po ilości iteracji których potrzebuje (dla uczenia metodą inną niż nasza ilość iteracji potrafi drastycznie wzrosnąć - np. podczas uczenia funkcją newlin ilość iteracji potrafi nawet sięgać paru tysięcy). Z tego wynika, że funkcja której użyliśmy (newp) jest szybsza (nie potrzebuje dużej ilości iteracji).

Działanie programu przeprowadziłam dla różnych wartości wag, oceniałem jakość rozpoznawania liter przez sieć, a następnie zwiększałem, lub zmniejszałem liczbę wag.

Załączone wykresy zarówno dla newlin() jak i newp() obrazują jak długo trwają (czas trwania określa ilość epok) oraz z jakim błędem średniokwadratowym zmienia się funkcja ucząca.

Widzimy, że na funkcję „newlin” nie wpływa zmiana współczynnika wag, liczba epok i liczba „dokładności” powtarza się, natomiast jest zależna od współczynnika błędu średniokwadratowego, zwiększa liczbę epok o co najmniej 200, gdy zmienimy z wartości 0.01 na 0.001 a już napewno na 0.0001.

Błąd średniokwadratowy pokazuje że im jest on mniejszy tym sieć neuronowa uczy się szybciej, lecz nie zawsze czas nauki gwarantuje większą skuteczność treningu.