Kamil Udziela
Inżynieria Obliczeniowa
Nr. indeksu: 286133

AGH

SPRAWOZDANIE:

Scenariusz 2 - Budowa i działanie sieci jednowarstwowej

Spis treści:

1. Cel projektu
2. Definicje
3. Użyte algorytmu
4. Elementy programu
5. Dane I/O programu
6. Wnioski
7. Listing kodu

1. Cel projektu:

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

2. Definicje, niezbędne do zrealizowania projektu:

- Sieć neuronowa:

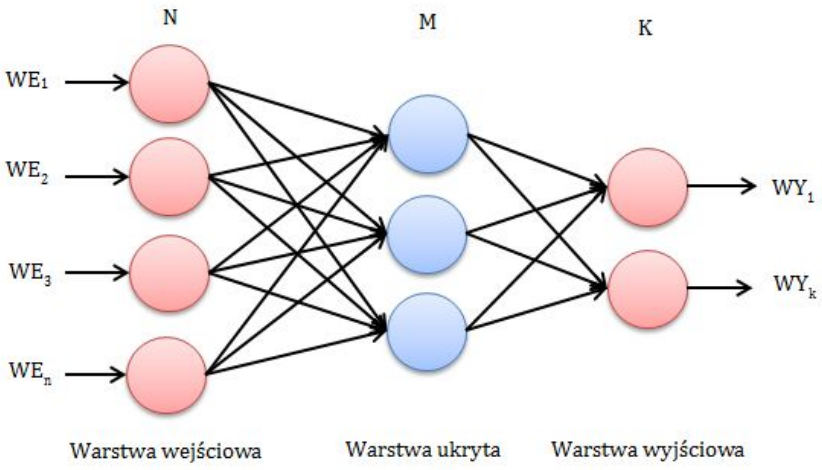
  Zbiór neuronów, realizujących różne cele. W przypadku sztucznych sieci neuronowych jest to sztuczna struktura, zaprojektowana i zbudowana w taki sposób, aby modelowała działanie naturalnego układu nerwowego, w szczególności mózgu.

- Sieć jednokierunkowa:

  Sieć neuronowa, składająca się z neuronów ułożonych w taki sposób, aby kierunek przepływu sygnałów był jeden. Połączenie między-warstwowe w sieci jednokierunkowej występuje tylko między kolejnymi warstwami tej sieci. Sieć jednokierunkowa posiada warstwy:

  - wejściową
  - wyjściową
  - warstwy ukryte

  Układ sieci jednokierunkowej możemy traktować, jako układ aproksymacji funkcji nieliniowej wielu zmiennych (y = f[u]).



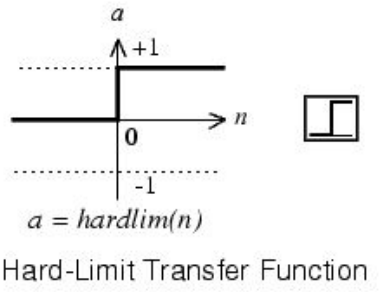
- Sieć jednowarstwowa:
  Mają tylko jedną warstwę sieci neuronowej (neuronów). Sieci jednowarstwowe mogą rozwiązać jedynie wąską klasę problemów.

3. Algorytmy użyte podczas wykonywania projektu (w Matlab):

➔ **NewP** - funkcja Newp tworzy jednowarstwową sieć neuronową, złożoną z określonej liczby neuronów o funkcjach aktywacji „twardego” perceptronu.

Parametry tej funkcji:

- macierz określająca liczbę wejść do naszej sieci neuronowej
- liczba neuronów w sieci
- funkcja aktywacji neuronów (funkcja ‘hardlim’).
  Hardlim - funkcja skoku jednostkowego:



Hard-Limit Transfer Function

- funkcja trenowania sieci neuronów (‘learnp’)

Funkcje uczące:

- learnbp - uczenie ze wsteczną propagacją błędu
- learnbpm - uczenie ze wsteczną propagacją błędu z zastosowaniem metody momentum.

➔**Init** - funkcja init służy do inicjowania stworzonej przez nas sieci neuronowej. Wartości wag i progów, gdy inicjujemy funkcją init są losowe. Na wejście funkcji init dostarczamy nazwę inicjowanej przez nas sieci (net2).

➔**Sim** - symulacja wykreowanej przez nasz program sieci neuronowej. Argumentami funkcji sim są: nasza sieć neuronowa oraz tablicę wszystkich elementów, które dana sieć ma się nauczyć. W przypadku mojej sieci jest to wektor duze_litery lub wektor male_litery.

➔**Train** - funkcja uczenia sieci neuronowej. (Argumenty j.w.)

4. Działanie i przebieg naszego programu:

I. Tworzenie tablicy wszystkich dużych liter i oddzielnie małych liter i inicjowanie ich.

```
%Zbiór naszych małych liter
a=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 ];
b=[1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 ];
c=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 ];
d=[0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1 0 1 1 1 1 ];
e=[0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 ];
f=[0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 ];
g=[0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1 0 ];
h=[1 0 0 0 0 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 1 0 0 1 0 ];
i=[0 0 0 0 0 0 0 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 1 0 0 ];
j=[0 0 0 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 1 0 0 0 1 0 0 ];

male_litery=[a;b;c;d;e;f;g;h;i;j];
male_litery=male_litery';
```

```
wielkie_litery=
    [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 1 0 0 0 1 1 0 0 0 1;
    1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0;
    0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0;
    1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0;
    1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1;
    1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0;
    0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 1 1 1 0;
    1 0 0 0 1 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 1 0 0 0 1;
    0 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0;
    1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0
    a;b;c;d;e;f;g;h;i;j];
wielkie_litery=wielkie_litery';
```

II. Tworzenie wektora danych wejściowych, który potrzebny jest nam do określenia warunku, że dla 0 jest litera mała, a dla 1 duża.

```
%Wektor danych wejściowych
Wektor_in=[1 ; 1 ; 1; 1; 1; 1; 1; 1; 1; 1;0;0;0;0;0;0;0;0;0;0];
Wektor_in=Wektor_in';
```

III. Stworzenie i zainicjowanie naszej sieci neuronowej.

```
%Tworzenie sieci neuronowej
net2 = newp([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], 1);

%Inicjowanie sieci neuronowej
net2 = init(net2);
```

IV. Symulacja naszej sieci.

```
%Symulacja sieci neuronowej
Siec_litery = sim(net2,wielkie_litery);
```

V. Trening sieci neuronowej stworzonej przez nas.

```
%Trening ( uczenie ) sieci neuronowej
net2 = train(net2,wielkie_litery,Wektor_in);
```

VI. Ponowna symulacja

```
%Ponowna symulacja sieci neuronowej
Siec_litery = sim(net2,wielkie_litery);
```

VII. Stworzenie wektora litery, którą będzie sprawdzać nasza sieć i inicjowanie jej.

```
%Litera która sprawdza nasza sieć neuronowa
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 1 0 0 0 1 1 0 0 0 1];
 A=A';
```

lub

```
a_mal=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 ];
a_mal = a_mal';
```

VIII. Symulacja sieci z użyciem stworzonej przez nas litery.

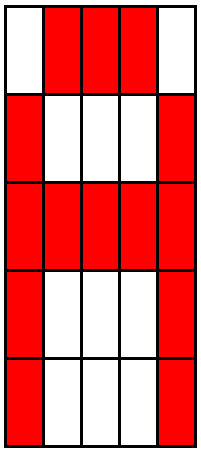
```
%Symulacja sieci z podaniem naszej sprawdzanej litery
Siec_litery = sim(net2,A);
Siec_litery = Siec_litery';
```

IX. Stworzenie warunku, który dla odpowiednio małej lub dużej litery napisze jaka ona jest.

```
%Warunek sieci neuronowej:
if Siec_litery==1 disp('Litera jest duza');
else disp('Litera jest mala');
end
```

5. Dane wejściowe i wyjściowe programu:

Dane wejściowe, w naszym programie to zbiór liter do nauki przez naszą sieć neuronową. Litery są reprezentowane przez zbiór zer i jedynek, zgodnie ze schematem (przykład dla litery A):



Pola czerwone to jedynki a pola białe to zero, a więc tabela reprezentująca literę A będzie miała następującą postać:

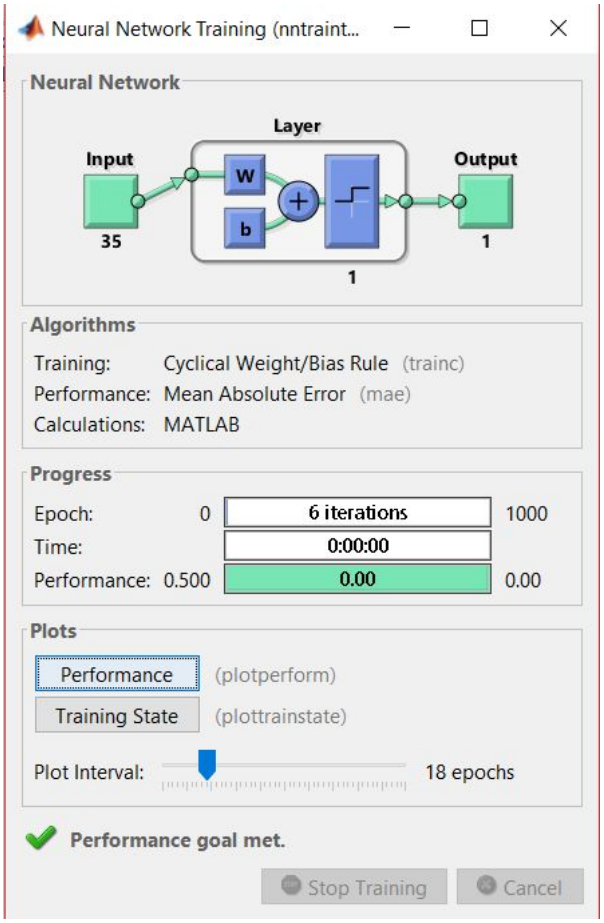
0	1	1	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

Dane wyjściowe naszej sieci to informacja o tym czy litera jest duża czy mała, zwraca 0 lub 1. Dla zera litera jest mała, a dla 1 litera jest duża.

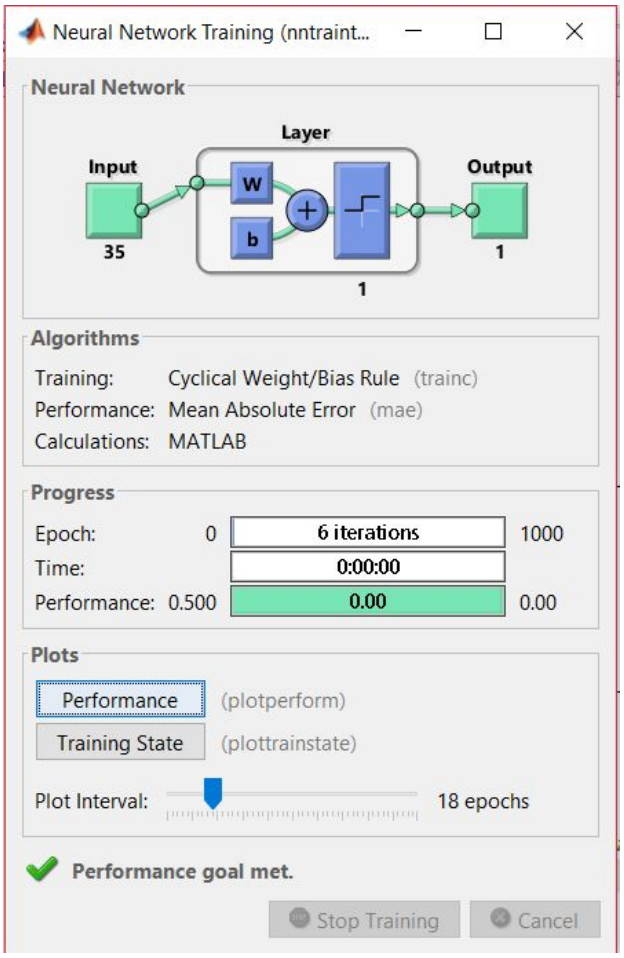
Komunikat Matlab: *‘Litera jest duza’* lub *‘Litera jest mala’* .

6. Wnioski i wyniki:

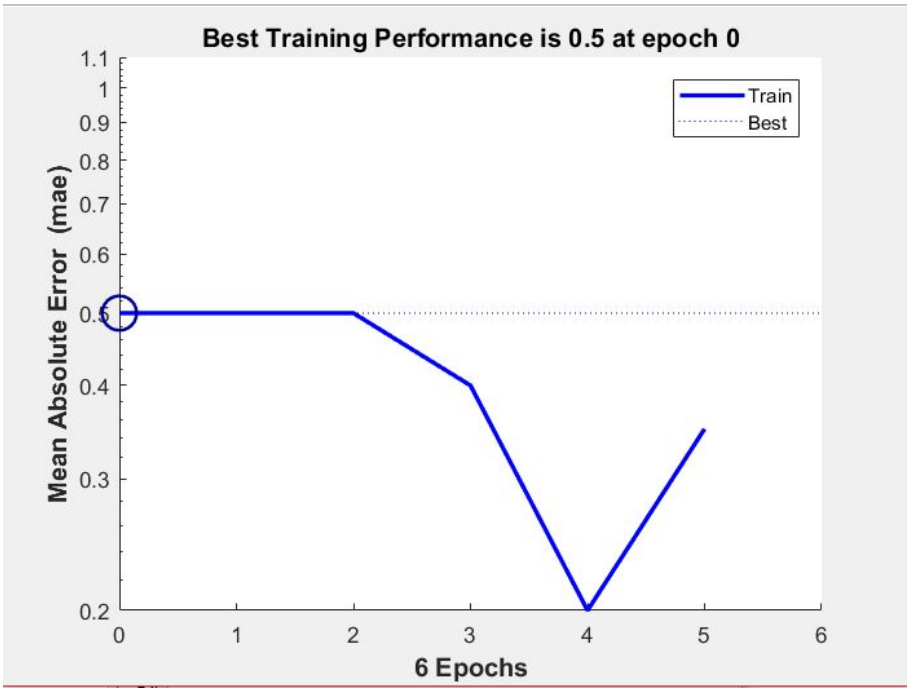
a). Wynik programu dla litery A (dużej):



b). Wynik programu dla litery a (małej):



c) Wykres pokazujący najlepsze powtórzenie treningu:



Wnioski:

Ilość iteracji (epochs) potrzebnych do nauki wynosi 0.5. Program zwraca dokładnie wartość jeden lub zero, w zależności od wyniku (0 - litera mała, 1 - litrera duża). Program 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).

7. Listing kodu :

```
close all;clear all;clc
%Litera która sprawdza nasza sieć neuronowa
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 1 0 0 0 1 1 0 0 0 1];
A=A';
B = [1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0];
B = B';
a_mal=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 ];
a_mal = a_mal';
%Zbiór naszych małych liter
a=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 ];
b=[1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 ];
c=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 ];
d=[0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1 0 1 1 1 1 ];
e=[0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 ];
f=[0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 ];
g=[0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1 0 ];
h=[1 0 0 0 0 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 1 0 0 1 0 ];
i=[0 0 0 0 0 0 0 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 1 0 0 ];
j=[0 0 0 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 1 0 0 0 1 0 0 ];
male_litery=[a;b;c;d;e;f;g;h;i;j];
male_litery=male_litery';
```

```
%Zbiór naszych wielkich liter
wielkie_litery=[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 1 0 0 0 1 1 0 0 0
1;
1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0;
0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0;
1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0;
1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1;
1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0;
0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 1 1 1 0;
1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1;
0 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0;
1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0
a;b;c;d;e;f;g;h;i;j];
wielkie_litery=wielkie_litery';
%Wektor danych wejściowych
Wektor_in=[1 ; 1 ; 1; 1; 1; 1; 1; 1; 1; 1;0;0;0;0;0;0;0;0;0;0;0];
Wektor_in=Wektor_in';
%Tworzenie sieci neuronowej
net2 = newp([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], 1);
%Inicjowanie sieci neuronowej
net2 = init(net2);
%Symulacja sieci neuronowej
Siec_litery = sim(net2,wielkie_litery);
%Trening ( uczenie ) sieci neuronowej
net2 = train(net2,wielkie_litery,Wektor_in);
%Ponowna symulacja sieci neuronowej
Siec_litery = sim(net2,wielkie_litery);
%Symulacja sieci z podaniem naszej sprawdzanej litery
Siec_litery = sim(net2,A);
Siec_litery = Siec_litery';
%Warunek sieci neuronowej:
if Siec_litery==1 disp('Litera jest duza');
else disp('Litera jest mala');
end
```