Anna Szylak
IO rok 3

15.11.2018r.

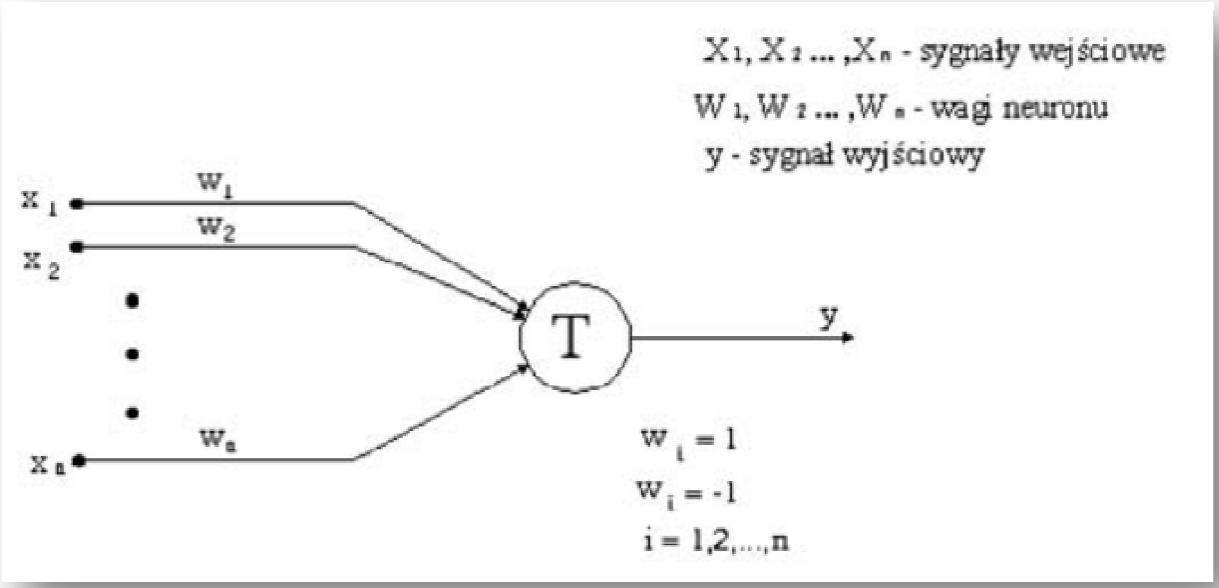
Podstawy sztucznej inteligencji

Sprawozdanie – budowa i działanie sieci jednowarstwowej

1. Wstęp teoretyczny

Aby móc mówić o sieci jednowarstwowej, przypomnijmy sobie teoretycznie czym jest:

- Neuron- budowa i zasada działania tego modelu została oparta na swoim biologicznym odpowiedniku. Założeniem było tu zastąpienie neuronu jednostką binarną. Model, który został przez nich zaproponowany wyglądał następująco:



Sygnał wejściowy w punkcje x1 , x2 ... xn ma wartość binarną 0 lub 1. Jeśli w chwili k pojawi się impuls, punkt przyjmuje wartość 1, jeśli nie, przyjmuje wartość 0. Za sygnał wyjściowy przyjmujemy wartość y. Reguła aktywacji neuronu przyjmuje postać:

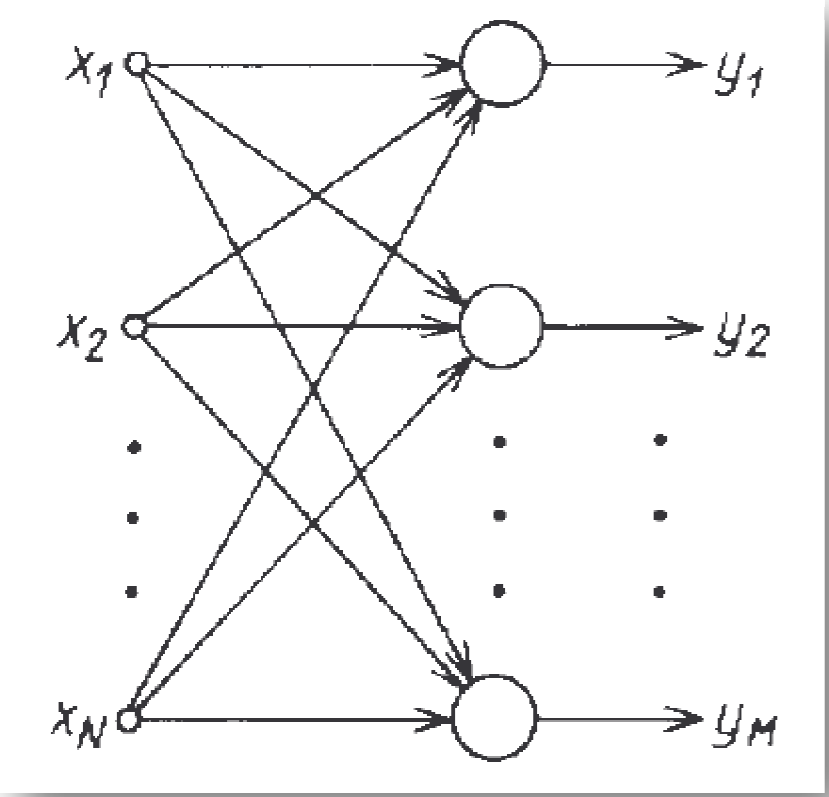
$$y^{k+1} = \begin{cases} 1, & gdy \sum_{i=1}^{n} w_i x_i^k \geq T, \\ 0, & gdy \sum_{i=1}^{n} w_i x_i^k < T, \end{cases}$$

Gdzie k = 0,1,2 ... są kolejnymi momentami czasu. W okresie k oraz k+1 upływa jednostkowy czas opóźnienia, w1 jest multiplikatywną wagą przypisaną połączeniu wejścia i z błoną neuron. Dla synaps pobudzających w1 = +1, dla synaps hamujących w1 = -1. Poniżej wartości progowej T neutron nie działa. Możliwości i właściwości sieci są wynikiem wspólnego działania bardzo wielu połączonych ze sobą elementów w tym pojedynczych neuronów. Wzajemna współ- praca oraz sposób połączenia neuronów między sobą spowodował powstanie róż- nych typów sieci. Każdy typ sieci jest z kolei powiązany z odpowiednią metodą doboru wag (uczenia).

Rozróżniamy kilka rodzajów neuronów:

-warstwy wejściowej
-warstw ukrytych
-warstwy wyjściowej

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 .



2. Cel ćwiczenia

Głównym celem ćwiczenia było poznanie budowy i działania jednowarstwowych sieci neuronowych oraz uczenie rozpoznawania wielkości liter.

Ćwiczenie wykonałam w programie MatLab.

Kroki, które należy wykonać:
a) Wygenerowanie danych uczących i testujących, zawierających 10 dużych i 10 małych liter dowolnie wybranego alfabetu w postaci dwuwymiarowej tablicy .
b) Przygotowanie dwóch jednowarstwowych sieci - każda według. innego algorytmu
c) Uczenie sieci dla przy różnych współczynnikach uczenia.
d) Testowanie sieci.

3. Zadanie do wykonania , opis działania +listing kodu

W pierwszej kolejności musiałam wygenerować dane uczące i testujące, które będą zawierać po 10 dużych i 10 małych liter alfabetu.

Litery wybrane przeze mnie: **A a D d E e F f N n P p R r T t U u Y y**

Dane uczące:

```
letterA=   [0 0 1 0 0 ...
            0 1 0 1 0 ...
            0 1 0 1 0 ...
            1 0 0 0 1 ...
            1 1 1 1 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ]';

lettera=   [0 0 0 0 0 ...
            0 0 0 0 0 ...
            1 1 1 0 0 ...
            0 0 0 1 0 ...
            1 1 1 1 0 ...
            1 0 0 1 0 ...
            1 1 1 1 1 ]';

letterD  = [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 ]';

letterd=   [0 0 0 0 0 ...
            0 0 0 1 0 ...
            0 0 0 1 0 ...
            0 0 0 1 0 ...
            1 1 1 1 0 ...
            1 0 0 1 0 ...
            1 1 1 1 0 ]';

letterE  = [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 ]';

lettere=   [0 0 0 0 0 ...
            0 0 0 0 0 ...
            0 1 1 0 0 ...
            1 0 0 1 0 ...
            1 1 1 1 0 ...
            1 0 0 0 0 ...
            0 1 1 1 0 ]';
```

```
letterF =  [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 ]';

letterf=   [0 0 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 ...
            0 1 0 0 0 ]';

letterN=   [1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 0 0 1 ...
            1 0 1 0 1 ...
            1 0 0 1 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1]';

lettern=   [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 0 0 1 0 ...
            1 0 0 1 1]';

letterP=   [1 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 1 1 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0]';

letterp=   [ 0 0 0 0 0 ...
             0 0 0 0 0 ...
             1 1 1 1 0 ...
             1 0 0 1 0 ...
             1 1 1 1 0 ...
             1 0 0 0 0 ...
             1 0 0 0 0]';

letterR=   [1 1 1 1 0 ...
            1 0 0 0 1  ...
            1 0 0 0 1 ...
            1 1 1 1 0 ...
            1 0 1 0 0 ...
            1 0 0 1 0 ...
            1 0 0 0 1]';

letterr=   [0 0 0 0 0 ...
            0 0 0 0 0 ...
            0 1 1 1 1 ...
            0 0 1 0 1 ...
            0 0 1 0 0 ...
            0 0 1 0 0 ...
            0 0 1 0 0]';
```

```
    letterT=   [1 1 1 1 1 ...
                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 0 1 0 0]';

    lettert=   [0 0 1 0 0 ...
                0 0 1 0 0 ...
                0 1 1 1 0 ...
                0 0 1 0 0 ...
                0 0 1 0 0 ...
                0 0 1 0 1 ...
                0 0 1 1 0]';

    letterU=  [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 0 0 0 1 ...
               0 1 1 1 0]';
```

```
    letteru=  [0 0 0 0 0 ...
               0 0 0 0 0 ...
               0 0 0 0 0 ...
               1 0 0 1 0 ...
               1 0 0 1 1 ...
               1 0 0 1 0 ...
               0 1 1 0 1]';

    letterY=  [1 0 0 0 1 ...
               1 0 0 0 1 ...
               0 1 0 1 0 ...
               0 0 1 0 0 ...
               0 0 1 0 0 ...
               0 0 1 0 0 ...
               0 0 1 0 0]';

    lettery=  [0 0 0 0 0 ...
               1 0 0 0 1 ...
               1 0 0 1 1 ...
               0 1 1 1 1 ...
               1 0 0 1 0 ...
               0 1 0 1 0 ...
               0 0 1 0 0]';
```

Liniowym rozwinięciem tych matryc będą wektory wejściowe ciągu uczącego:

```
171     in=[0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
172         1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0
173         1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0
174         1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0
175         0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0
176         1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 1 1
177         0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
178         0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
179         0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
180         1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 1
181         1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 0 1
182         0 1 0 0 0 1 0 1 1 1 0 1 0 1 0 1 0 0 1 0
183         0 1 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0
184         0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 1
185         1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1
186         1 0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0
187         1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0 1
188         1 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 0 0 1 1
189         1 1 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1
190         1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1
191         1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1
192         0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0
193         0 1 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 0
194         0 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 1
195         1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0
196         1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0
197         0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
198         0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0
199         0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1
200         1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0
201         1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0
202         0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0
203         0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1
204         0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0
205         1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0];
206        %A a D d E e F f N n P p R r T t U u Y y

216 -   out=[ 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 ];
```

Wyjściową (out) macierzą będą liczby 1 symbolizujący literę dużą lub 0 literę małą.

Dane testujące:

```
223 -   testujacyA=[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;];
224             %1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
225             %A a D d E e F f N n P p R r T t U u Y y
226 -   testujacya=[0;0;0;0;0;0;0;0;0;0;1;1;1;0;0;0;0;0;1;0;1;1;1;1;1;0;1;0;0;1;0;1;1;1;1;1;];
227             %0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
228             %A a D d E e F f N n P p R r T t U u Y y
229 -   testujacyD=[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;1;0;];
230             %0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
231             %A a D d E e F f N n P p R r T t U u Y y
232 -   testujacyd=[0;0;0;0;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;1;1;1;1;1;0;1;0;0;1;0;1;1;1;1;1;0;];
233             %0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
234             %A a D d E e F f N n P p R r T t U u Y y
235 -   testujacyE=[1;1;1;1;1;1;0;0;0;0;1;0;0;0;0;1;1;1;1;1;0;1;0;0;0;0;1;0;0;0;0;1;1;1;1;1;];
236             %0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
237             %A a D d E e F f N n P p R r T t U u Y y
238 -   testujacye=[0;0;0;0;0;0;0;0;0;0;0;1;1;0;0;1;0;0;1;0;1;1;1;1;1;0;1;0;0;0;0;0;1;1;1;0;];
239             %0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
240             %A a D d E e F f N n P p R r T t U u Y y
```

```
testujacyF=[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 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyf=[0;0;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;1;0;0;0;];
            %0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyN=[1;0;0;0;1;1;0;0;0;1;1;1;0;0;1;1;0;1;0;1;1;0;0;1;1;1;0;0;0;1;1;0;0;0;1;];
            %0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyn=[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;0;0;1;0;1;0;0;1;1;];
            %0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyP=[1;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;1;1;1;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;];
            %0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyp=[0;0;0;0;0;0;0;0;0;0;1;1;1;1;0;1;0;0;1;0;1;1;1;1;0;1;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
            %A a D d E e F f N n P p R r T t U u Y y
testujacyR=[1;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;1;1;1;0;1;0;1;0;0;1;0;0;1;0;1;0;0;0;1;];
            %0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyr=[0;0;0;0;0;0;0;0;0;0;1;1;1;1;0;0;1;0;1;0;0;1;0;0;0;0;1;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
            %A a D d E e F f N n P p R r T t U u Y y
testujacyT=[1;1;1;1;1;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;0;1;0;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyt=[0;0;1;0;0;0;0;1;0;0;0;1;1;1;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;1;0;0;1;1;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyU=[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;0;0;0;1;0;1;1;1;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
            %A a D d E e F f N n P p R r T t U u Y y
testujacyu=[0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;1;0;1;0;1;1;1;0;0;1;0;0;1;1;0;1;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
            %A a D d E e F f N n P p R r T t U u Y y;
testujacyY=[1;0;0;0;1;1;0;0;0;1;0;1;0;1;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
            %A a D d E e F f N n P p R r T t U u Y y
testujacyy=[0;0;0;0;0;1;0;0;0;1;1;0;0;1;1;0;1;1;1;1;1;1;0;0;1;0;1;0;0;1;0;1;0;0;0;1;0;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
            %A a D d E e F f N n P p R r T t U u Y y
```

Litery są utworzone na matrycy 5 x 7 czyli na **35 polach**. I tyle wejść ma sieć, mogą zostać wypełnione tylko liczbą 0(min) albo 1(max);

```
min_max=[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];
```

```
ilosc_wyjsc=1;
%net=newlin(min_max,ilosc_wyjsc);
net = newp(min_max,ilosc_wyjsc,'hardlim','learnpn');
```

pierwszy parametr - zakresy kolejnych wejść,
drugi parametr - ilość neuronów w każdej kolejnej warstwie,
trzeci parametr - funkcje przynależności.

Funkcje aktywacji:
learnpn- zmiany wag i progów przy pomocy znormalizowanej metody uczenia perceptronu

Wyżej pokazałam stworzenie dwóch jednowarstwowych sieci , każda według innego algorytmu, co było moim kolejnym krokiem.

Krótki opis i budowa funkcji:

Poniżej przedstawię krótki schemat budowy funkcji **newp** (tworzenie jednowarstwowej sieci złożonej z „twardych" perceptronów):
Funkcja tworzy jednowarstwową sieć neuronową, złożoną z zadanej liczby neuronów o funkcjach aktywacji „twardego" perceptronu (ang. *Hardlimit perceptron*).

Wywołanie funkcji: **NET = NEWP(PR, S, TF, LF)**

WEJŚCIE:
PR - macierz o wymiarach *R*x*2*, 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 = 'hardlim'; 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'
WYJŚCIE:
NET - struktura (obiekt) zawierająca opis architektury, metod treningu, wartości liczbowe wag i progów oraz inne parametry sieci perceptronowej.

Oraz budowy funkcji newlin (tworzenie jednowarstwowej sieci złożonej z neuronów liniowych):
Funkcja tworzy jednowarstwową sieć neuronową, złożoną z zadanej liczby neuronów o liniowych funkcjach aktywacji. Tego typu sieć jest zwykle wykorzystywana jako filtr adaptacyjny do przetwarzania sygnałów lub predykcji szeregów czasowych.

Wywołanie funkcji: **NET = NEWLIN(PR, S, ID, LF)**

WEJŚCIE:
PR - macierz o wymiarach *R*x*2*, 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
WYJŚCIE:
NET - struktura (obiekt) zawierająca opis architektury, metod treningu, wartości liczbowe wag oraz inne parametry sieci perceptronowej.

Kolejnym krokiem i zarazem ostatnim, było uczenie sieci dla różnych współczynników.

```
281 -    net.trainParam.epochs = 2500;
282 -    net.trainParam.goal = 0.01;
283 -    net.trainParam.mu = 0.001;
284 -    net= train(net,in,out);
```

Znaczenia:
net.trainParam.epochs- maksymalna liczba epok uczenia
net.trainParam.goal- oczekiwana końcowa wartość funkcji celu
net.trainParam.mu- współczynnik uczenia

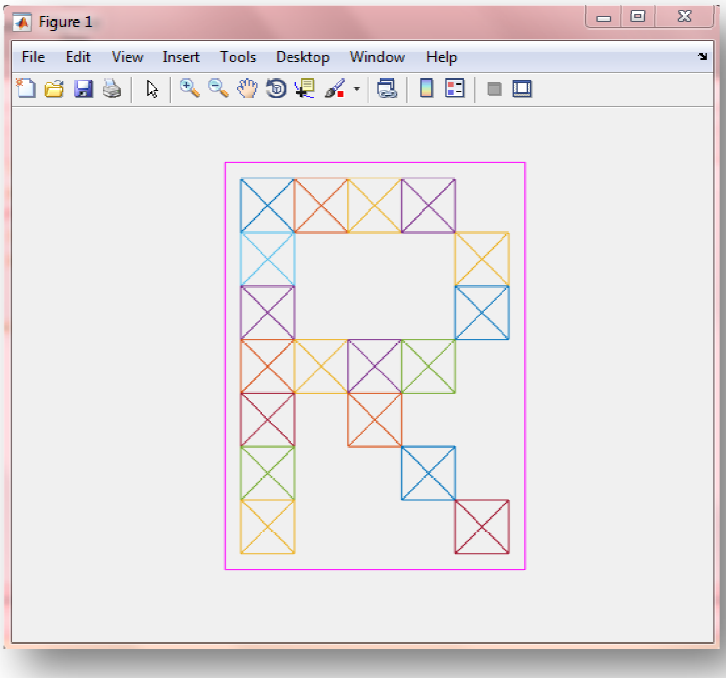
Poniżej przedstawiam wywoływanie literki i wyświetlanie jej oraz pętle, której zadaniem jest sprawdzenie, czy literka jest mała, czy duża.

```
285      %Wywołujemy przykładową literę z alfabetu, może to być R:
286 -    literka=testujacyR;
287      %Wyświetlamy ją:
288 -    plotchar(literka);
289
290 -    a=sim(net,literka);
291 -    if round(a) == 0
292 -        disp('To jest mała litera');
293 -    else
294 -        disp('To jest duża litera');
295 -    end
296 -      disp (a)
```

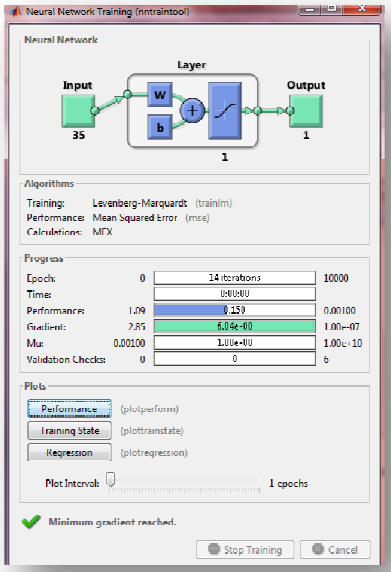
Znacznie :
literka- jest to wybrana litera do testu
plotchar- wyświetlenie przesłanego elementu
sim()- funkcja wyznacza wartości wyjść sieci neuronowej.
round()- zaokrągla liczbę rzeczywistą

Przedstawiam działanie programu:

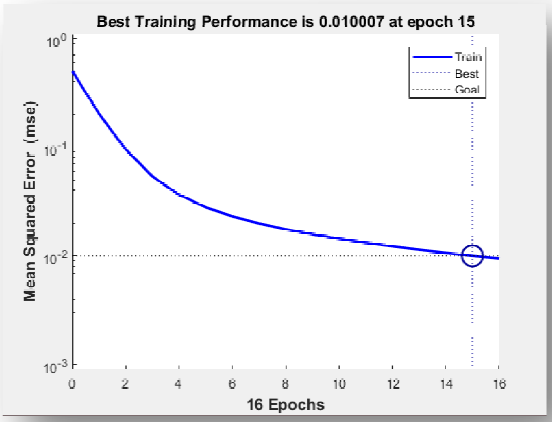
1. Wywołuję literkę „R”.
2. Otrzymuję obraz tej literki:



3. Otrzymuję „Neural Network Training”:



4. Plots-> Performance



5. Dostaję wynik o wielkości literki oraz wynik dokładności:

```
To jest duża litera
    1.0000
```

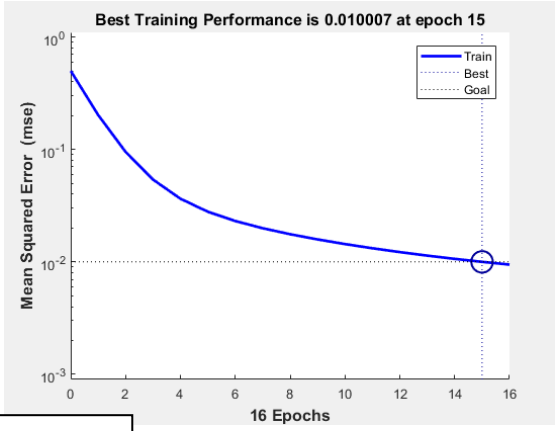
4.Wnioski

Działanie programu przeprowadziłam dla różnych wartości wag, oceniałam jakość rozpoznawania liter przez sieć, a następnie zwiększałam, lub zmniejszałam liczbę wag.
Funkcja „newp” rozpoznawała wielkości liter w bardzo krótkim czasie, wystarczyły jej 3 epoki i jej „dokładność” w liczeniu za pomocą działania: „a=sim(net,literka)”(,gdzie „literka” oznacza daną literę testującą), wynosi prawie 100%. Wyżej w sprawozdaniu definiowałam, 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.

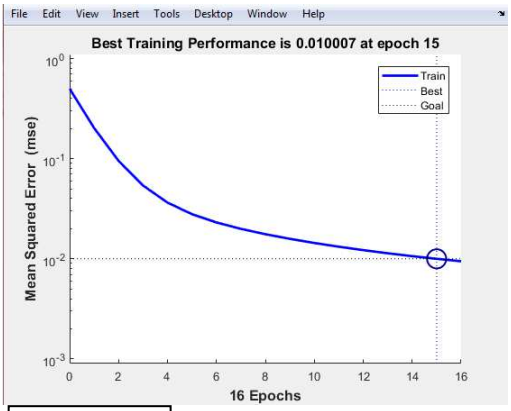
Dla funkcji newlin, wyniki „dokładności” są bardziej zróżnicowane, więc przedstawię dla kilku przykładowych liter oraz dla różnych wartości wagowych.

Dla **błędu średniokwaratowego=0.01**

newlin	Dokładność, waga=0.1	Epok	Dokładność, waga=0.01	Epok	Dokładność, waga=0.001	Epok
A	0.9113	16	0.9113	16	0.9113	16
R	1.0042	16	1.0042	16	1.0042	16
r	0.1240	16	0.1240	16	0.1240	16
p	0,0035	16	0,0035	16	0,0035	16



Litera A



Litera R

Dla **błędu średniokwaratowego=0.001**

newlin	Dokładność, waga=0.1	Epok	Dokładność, waga=0.01	Epok	Dokładność, waga=0.001	Epok
A	0.9321	203	0.9321	203	0.9321	203
R	1.0346	203	1.0346	203	1.0346	203
r	0.0195	203	0.0195	203	0.0195	203
p	0.0289	203	0.0289	203	0.0289	203

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.

Reasumując funkcja „newp” jest szybka i dokładna, przeprowadza ona uczenie za pomocą funkcji „trainc”, która realizuje cykliczne szkolenie przyrostowe, natomiast funkcja „newlin” za pomocą „tarinb”, który jest procesem zmian wag i progów(uczenia) w trybie „wsadowym”.

(W sprawozdaniu umieszczony został cały kod programu, jednak nie jest on w całości wklejony, tylko w częściach w celu dołączenia swoich wniosków, opisu danej funkcji i zasady ich działania, jednak w repozytorium można znaleźć sprawozdanie jak i osobny „ciągły” kod z MatLab’a.)