



Kraków, 31.10.2017r.

**Akademia Górniczo-Hutnicza  
im. Stanisława Staszica w Krakowie**

# **Podstawy Sztucznej Inteligencji**

## **Sprawozdanie numer 2**

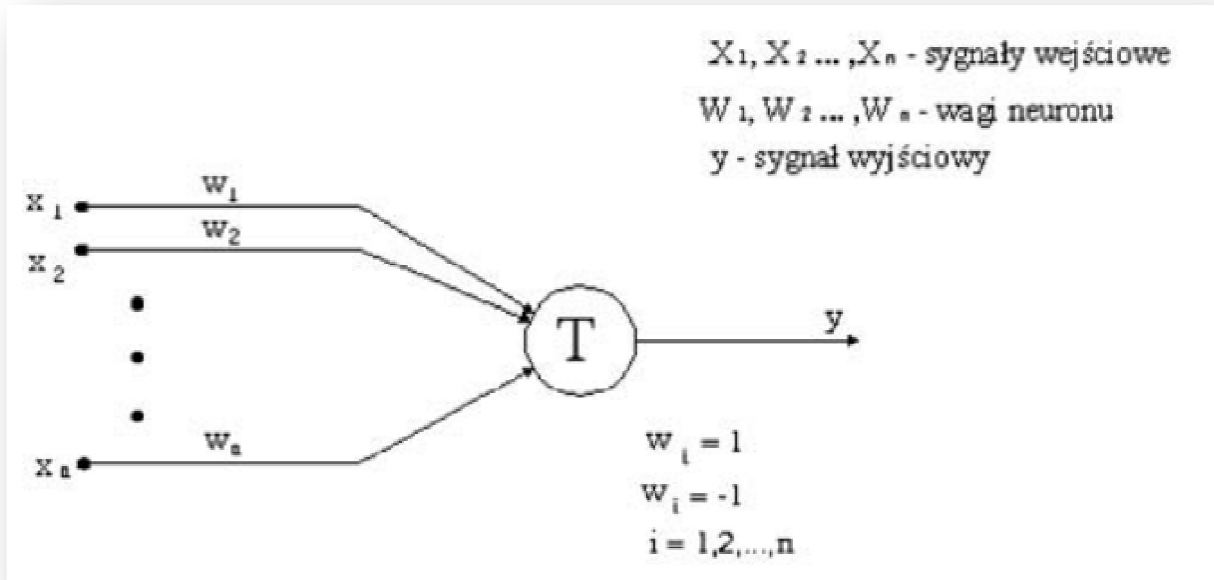
Budowa i działanie sieci jednowarstwowej

**Inżynieria Obliczeniowa, WiMiP  
Urszula Ślusarz  
nr. indeksu: 286132**

## 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 punkcie  $x_1, x_2 \dots x_n$  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:

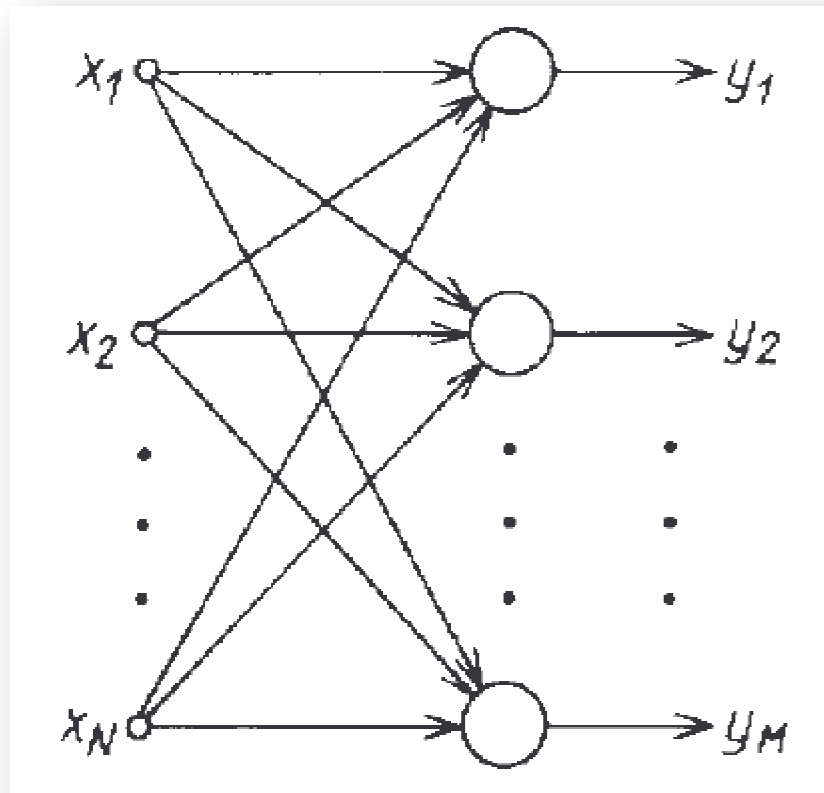
$$y^{k+1} = \begin{cases} 1, & \text{gdy } \sum_{i=1}^n w_i x_i^k \geq T, \\ 0, & \text{gdy } \sum_{i=1}^n w_i x_i^k < T, \end{cases}$$

Gdzie  $k = 0, 1, 2 \dots$  są kolejnymi momentami czasu. W okresie  $k$  oraz  $k+1$  upływa jednostkowy czas opóźnienia,  $w_1$  jest multiplikatywną wagą przypisaną połączeniu wejścia  $i$  z błoną neuron. Dla synaps pobudzających  $w_1 = +1$ , dla synaps hamujących  $w_1 = -1$ . Poniżej wartości progowej  $T$  neuron 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  $y_i$  do wartości, której oczekujemy  $d_i$ .



## 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:

- Wygenerowanie danych uczących i testujących, zawierających 10 dużych i 10 małych liter dowolnie wybranego alfabetu w postaci dwuwymiarowej tablicy.
- Przygotowanie dwóch jednowarstwowych sieci - każda według innego algorytmu
- Uczenie sieci dla przy różnych współczynnikach uczenia.
- 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:

```
6
7     letterA= [0 0 1 0 0 ...
8               0 1 0 1 0 ...
9               0 1 0 1 0 ...
10              1 0 0 0 1 ...
11              1 1 1 1 1 ...
12              1 0 0 0 1 ...
13              1 0 0 0 1 ]';
14
15     lettera= [0 0 0 0 0 ...
16               0 0 0 0 0 ...
17               1 1 1 0 0 ...
18               0 0 0 1 0 ...
19               1 1 1 1 0 ...
20               1 0 0 1 0 ...
21               1 1 1 1 1 ]';
22
23     letterD = [1 1 1 1 0 ...
24               1 0 0 0 1 ...
25               1 0 0 0 1 ...
26               1 0 0 0 1 ...
27               1 0 0 0 1 ...
28               1 0 0 0 1 ...
29               1 1 1 1 0 ]';
30
31     letterd= [0 0 0 0 0 ...
32               0 0 0 1 0 ...
33               0 0 0 1 0 ...
34               0 0 0 1 0 ...
35               1 1 1 1 0 ...
36               1 0 0 1 0 ...
37               1 1 1 1 0 ]';
38
39     letterE = [1 1 1 1 1 ...
40               1 0 0 0 0 ...
41               1 0 0 0 0 ...
42               1 1 1 1 0 ...
43               1 0 0 0 0 ...
44               1 0 0 0 0 ...
45               1 1 1 1 1 ]';
46
47     lettere= [0 0 0 0 0 ...
48               0 0 0 0 0 ...
49               0 1 1 0 0 ...
50               1 0 0 1 0 ...
51               1 1 1 1 0 ...
52               1 0 0 0 0 ...
53               0 1 1 1 0 ]';
```

```

56     letterF = [1 1 1 1 1 ...
57               1 0 0 0 0 ...
58               1 0 0 0 0 ...
59               1 1 1 1 0 ...
60               1 0 0 0 0 ...
61               1 0 0 0 0 ...
62               1 0 0 0 0 ]';
63
64     letterf=  [0 0 1 1 0 ...
65               0 1 0 0 0 ...
66               1 1 1 0 0 ...
67               0 1 0 0 0 ...
68               0 1 0 0 0 ...
69               0 1 0 0 0 ...
70               0 1 0 0 0 ]';
71
72     letterN=  [1 0 0 0 1 ...
73               1 0 0 0 1 ...
74               1 1 0 0 1 ...
75               1 0 1 0 1 ...
76               1 0 0 1 1 ...
77               1 0 0 0 1 ...
78               1 0 0 0 1]';
79
80     lettern=  [0 0 0 0 0 ...
81               0 0 0 0 0 ...
82               1 1 1 1 0 ...
83               1 0 0 1 0 ...
84               1 0 0 1 0 ...
85               1 0 0 1 0 ...
86               1 0 0 1 1]';
87
88     letterP=  [1 1 1 1 0 ...
89               1 0 0 0 1 ...
90               1 0 0 0 1 ...
91               1 1 1 1 0 ...
92               1 0 0 0 0 ...
93               1 0 0 0 0 ...
94               1 0 0 0 0]';
95
96     letterp=  [ 0 0 0 0 0 ...
97               0 0 0 0 0 ...
98               1 1 1 1 0 ...
99               1 0 0 1 0 ...
100              1 1 1 1 0 ...
101              1 0 0 0 0 ...
102              1 0 0 0 0]';
103
104     letterR=  [1 1 1 1 0 ...
105               1 0 0 0 1 ...
106               1 0 0 0 1 ...
107               1 1 1 1 0 ...
108               1 0 1 0 0 ...
109               1 0 0 1 0 ...
110               1 0 0 0 1]';
111
112     letterr=  [0 0 0 0 0 ...
113               0 0 0 0 0 ...
114               0 1 1 1 1 ...
115               0 0 1 0 1 ...
116               0 0 1 0 0 ...
117               0 0 1 0 0 ...
118               0 0 1 0 0]';

```

```

120 letterT= [1 1 1 1 1 ...
121           0 0 1 0 0 ...
122           0 0 1 0 0 ...
123           0 0 1 0 0 ...
124           0 0 1 0 0 ...
125           0 0 1 0 0 ...
126           0 0 1 0 0]';
127
128 lettert= [0 0 1 0 0 ...
129           0 0 1 0 0 ...
130           0 1 1 1 0 ...
131           0 0 1 0 0 ...
132           0 0 1 0 0 ...
133           0 0 1 0 1 ...
134           0 0 1 1 0]';
135
136 letterU= [1 0 0 0 1 ...
137           1 0 0 0 1 ...
138           1 0 0 0 1 ...
139           1 0 0 0 1 ...
140           1 0 0 0 1 ...
141           1 0 0 0 1 ...
142           0 1 1 1 0]';
143
144 letteru= [0 0 0 0 0 ...
145           0 0 0 0 0 ...
146           0 0 0 0 0 ...
147           1 0 0 1 0 ...
148           1 0 0 1 1 ...
149           1 0 0 1 0 ...
150           0 1 1 0 1]';
151
152 letterY= [1 0 0 0 1 ...
153           1 0 0 0 1 ...
154           0 1 0 1 0 ...
155           0 0 1 0 0 ...
156           0 0 1 0 0 ...
157           0 0 1 0 0 ...
158           0 0 1 0 0]';
159
160 lettery= [0 0 0 0 0 ...
161           1 0 0 0 1 ...
162           1 0 0 1 1 ...
163           0 1 1 1 1 ...
164           1 0 0 1 0 ...
165           0 1 0 1 0 ...
166           0 0 1 0 0]';

```

Liniowym rozwinięciem tych macryc 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 0 1 0 1 0 1
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 0 1 1 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 0 1 1 1 0 0 1
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 0 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;0;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 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;0;1;1;1;0;0;0;0;0;0;1;0;1;1;1;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 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;0;];
230          %0 0 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
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;0;1;0;0;0;0;0;1;0;0;0;1;0;1;1;1;0;1;0;0;1;0;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 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;0;1;0;0;0;0;1;1;1;1;0;1;0;0;0;0;0;1;0;0;0;0;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 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;0;0;0;0;1;1;0;0;1;0;0;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 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

```

```

241 - testujacyF=[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;];
242 - %0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
243 - %A a D d E e F f N n P p R r T t U u Y y
244 - 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;0;1;0;0;0;];
245 - %0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
246 - %A a D d E e F f N n P p R r T t U u Y y
247 - testujacyN=[1;0;0;0;1;1;0;0;0;0;1;1;1;0;0;0;1;1;0;0;1;1;0;0;0;1;1;0;0;0;1;];
248 - %0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
249 - %A a D d E e F f N n P p R r T t U u Y y
250 - testujacyN=[0;0;0;0;0;0;0;0;0;0;1;1;1;1;0;0;1;0;0;0;1;0;0;0;1;0;0;0;1;0;0;1;1;];
251 - %0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
252 - %A a D d E e F f N n P p R r T t U u Y y
253 - testujacyP=[1;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;1;1;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;];
254 - %0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
255 - %A a D d E e F f N n P p R r T t U u Y y
256 - testujacyP=[0;0;0;0;0;0;0;0;0;0;0;1;1;1;0;0;1;0;0;0;1;0;0;0;0;0;1;0;0;0;0;];
257 - %0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
258 - %A a D d E e F f N n P p R r T t U u Y y
259 - testujacyR=[1;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;1;1;0;1;0;1;0;0;1;0;0;1;0;0;0;1;];
260 - %0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
261 - %A a D d E e F f N n P p R r T t U u Y y
262 - testujacyr=[0;0;0;0;0;0;0;0;0;0;0;1;1;1;0;0;1;0;0;1;0;0;0;0;0;1;0;0;0;0;1;0;0;];
263 - %0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
264 - %A a D d E e F f N n P p R r T t U u Y y
265 - 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;];
266 - %0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
267 - %A a D d E e F f N n P p R r T t U u Y y
268 - testujacyt=[0;0;1;0;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;0;1;1;];
269 - %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
270 - %A a D d E e F f N n P p R r T t U u Y y
271 - 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;1;1;];
272 - %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
273 - %A a D d E e F f N n P p R r T t U u Y y
274 - testujacyu=[0;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;];
275 - %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
276 - %A a D d E e F f N n P p R r T t U u Y y;
277 - testujacyY=[1;0;0;0;1;1;0;0;0;1;0;0;1;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;];
278 - %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
279 - %A a D d E e F f N n P p R r T t U u Y y
280 - testujacyy=[0;0;0;0;0;1;0;0;0;1;1;0;0;1;1;0;1;1;1;1;0;0;1;0;0;1;0;0;0;1;0;0;];
281 - %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
282 - %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);

```

5 - min_max=[0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;
6 - 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;
7 - 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;
8 - 0 1; 0 1; 0 1;0 1;0 1];

```

```

9 - ilosc_wyjsc=1;
10 - %net=newlin(min_max,ilosc_wyjsc);
11 - 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  $R \times 2$ , 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  $R \times 2$ , 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)
```

Znaczenie :

**literka**- jest to wybrana litera do testu

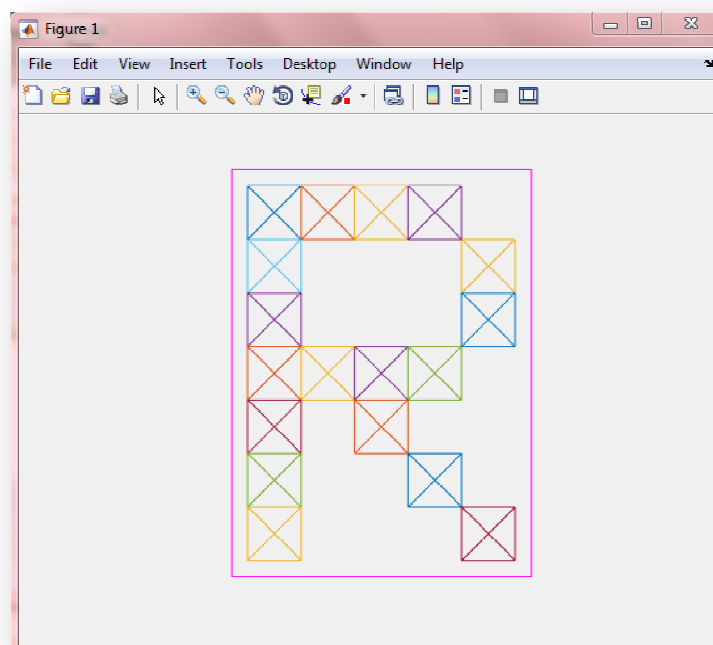
**plotchar**- wyświetlenie przesłanego elementu

**sim()**- funkcja wyznacza wartości wyjść sieci neuronowej.

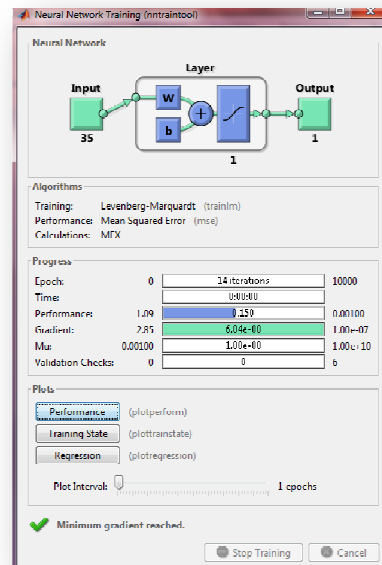
**round()**- zaokrągla liczbę rzeczywistą

**Przedstawiam działanie programu:**

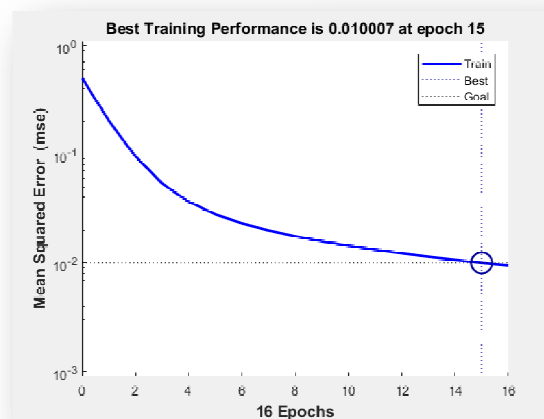
1. Wywołuję literkę „R”.
2. Otrzymuję obraz tej literki:



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

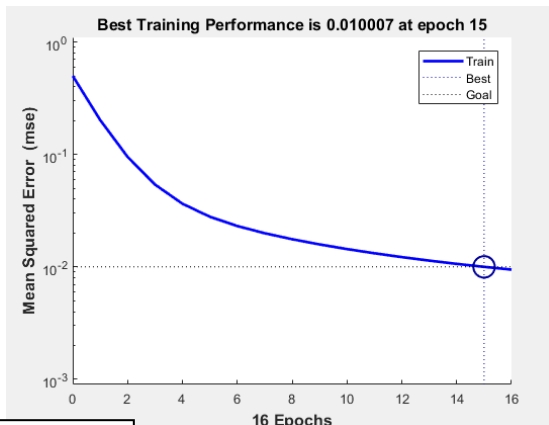
Działanie programu przeprowadziłam dla różnych wartości wag, oceniałam jakość rozpoznawania liter przez sieć, a następnie zwiększałam, lub zmniejszałam liczbę wag.

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.

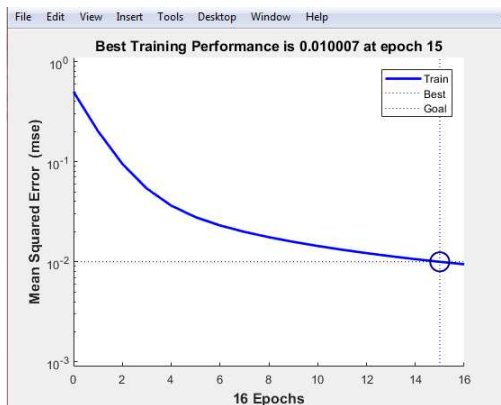
Dla funkcji newlin, wyniki „dokładności” są bardziej zróżnicowane, więc przedstawię dla kilku przykładowych liter oraz dla różnych wartości wagowych.

Dla błędu średniokwadratowego=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 średniokwadratowego=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.)