

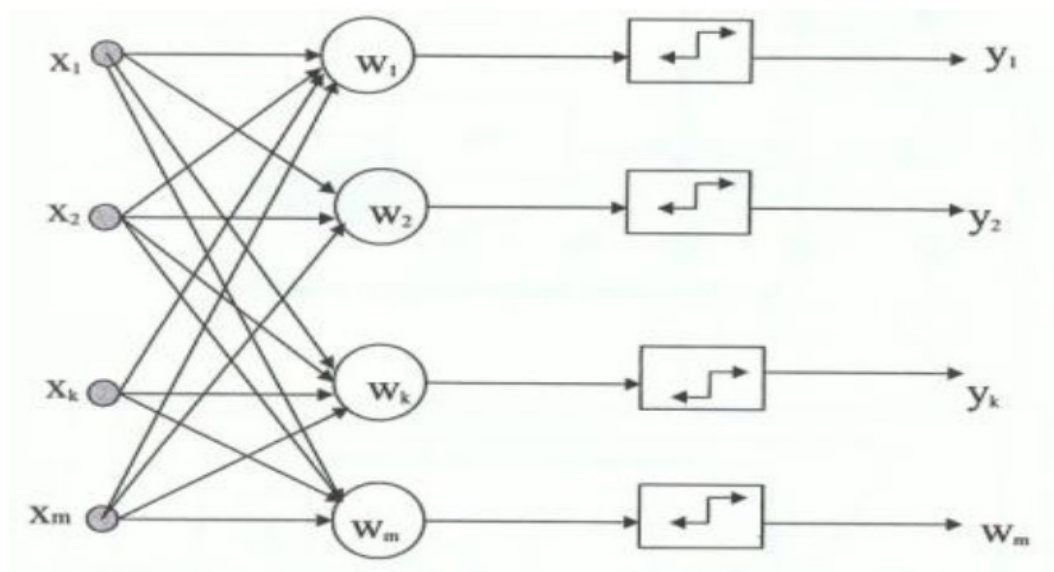
Grupa lab. <b>1</b>	Przedmiot <b>Podstawy Sztucznej Inteligencji</b>	Data wykonania. <b>16.11.2018</b>
Nr ćwicz. <b>2</b>	Temat ćwiczenia. <b>Budowa i działanie sieci jednowarstwowej</b>	
Imię i nazwisko. <b>Katarzyna Giądła</b>		Ocena i uwagi

### Część teoretyczna

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

**Sieć neuronowa** jest rodzajem architektury systemu komputerowego. Taka sieć przetwarza dane przez neurony pogrupowane w warstwie. Odpowiednie wyniki są uzyskiwane w procesie uczenia, który polega na modyfikowaniu wag tych neuronów, które są odpowiedzialne za błąd.

W tym projekcie skupimy się nad jednym z rodzajów sieci neuronowej – **sieci jednowarstwowej**. Jest ona zespołem kilku neuronów, przetwarzających sygnały z tych samych wejść (które nie tworzą warstwy neuronowej – nie zachodzi w nich proces obliczeniowy). W zależności od typu funkcji aktywacji sygnał jest przekazywany do wyjścia sieci. Wartości wektora wyjściowego porównywane są z zadaniem wektorem uczącym.



*Sieć jednokierunkowa jednowarstwowa*

### Część praktyczna

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*. Funkcja *newlin(P, T, ID, LR)* przyjmuje następujące parametry: *P* – macierz wejściowa, *T* – macierz wynikowa, *ID* – wektor ewentualnego opóźnienia na wejściu, *LR* – wskaźnik nauki.

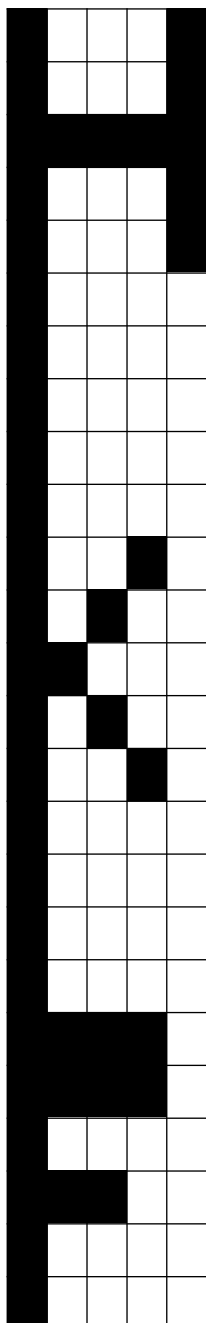
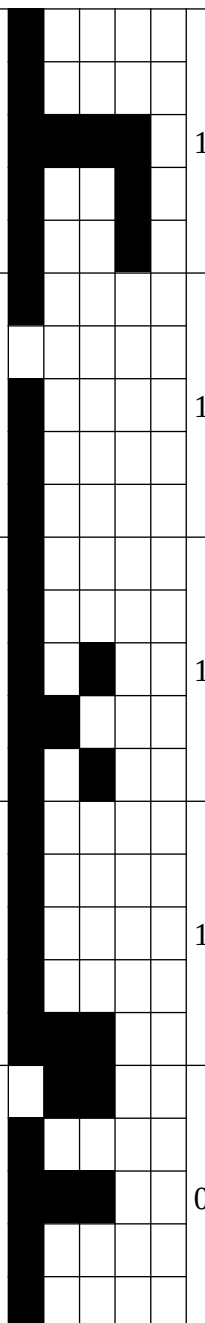
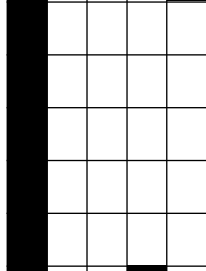
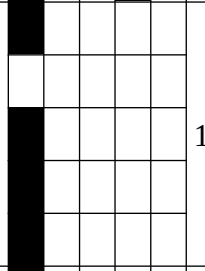
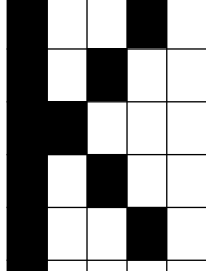
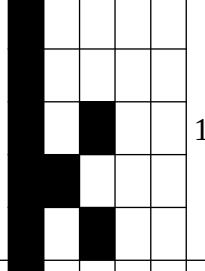
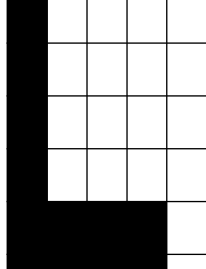
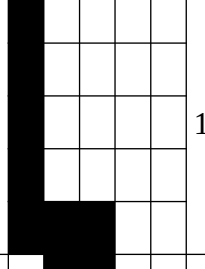
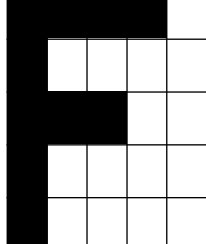
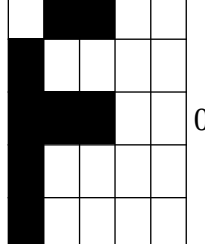
Pierwszym krokiem było wygenerowanie tablicy wejściowej zawierającej 10 małych i 10 wielkich liter (Literey A, B, C, D, E, H, I, K, L, F). Litery te zostały przedstawione w formie tablicy 5x5. Następnie taki obraz „zbinaryzowano” (polom białym przypisano wartość 0, a polom zamalowanym wartość 1). Następnie z przypisanych wartości utworzono ciąg binarny.

Przykład:

Litera					Krok 1 ( <i>binaryzacja</i> ):					Krok 2 ( <i>utworzenie ciągu</i> ):				
					0	1	1	1	0	01110 10001 10001 11111 10001				
					1	0	0	0	1					
					1	0	0	0	1					
					1	1	1	1	1					
					1	0	0	0	1					

Pozostałe litery:

Litera wielka	Kod	Litera mała	Kod
	01110 10001 10001 11111 10001		01100 00010 01110 10010 01111
	11100 10010 11100 10010 11100		10000 10000 11100 10010 11100
	11110 10000 10000 10000 11110		00000 00000 01100 10000 01100
	11100 10010 10010 10010 11100		00010 00010 01110 10010 01110
	11110 10000 11100 10000 11110		01100 10010 11100 10000 01100

	10001 10001 11111 10001 10001		10000 10000 11110 10010 10010
	10000 10000 10000 10000 10000		10000 00000 10000 10000 10000
	10010 10100 11000 10100 10010		10000 10000 10100 11000 10100
	10000 10000 10000 10000 11110		10000 10000 10000 10000 11100
	11110 10000 11100 10000 10000		01100 10000 11100 10000 10000

Wszystkie te ciągi binarne zapisano w tablicy *Learn\_in* – każda litera została zapisana w innej kolumnie. Wygenerowano również tablicę *Learn\_out*, w której przechowywano jest informację, czy litera jest wielka (wartość 1) lub mała (wartość 0).

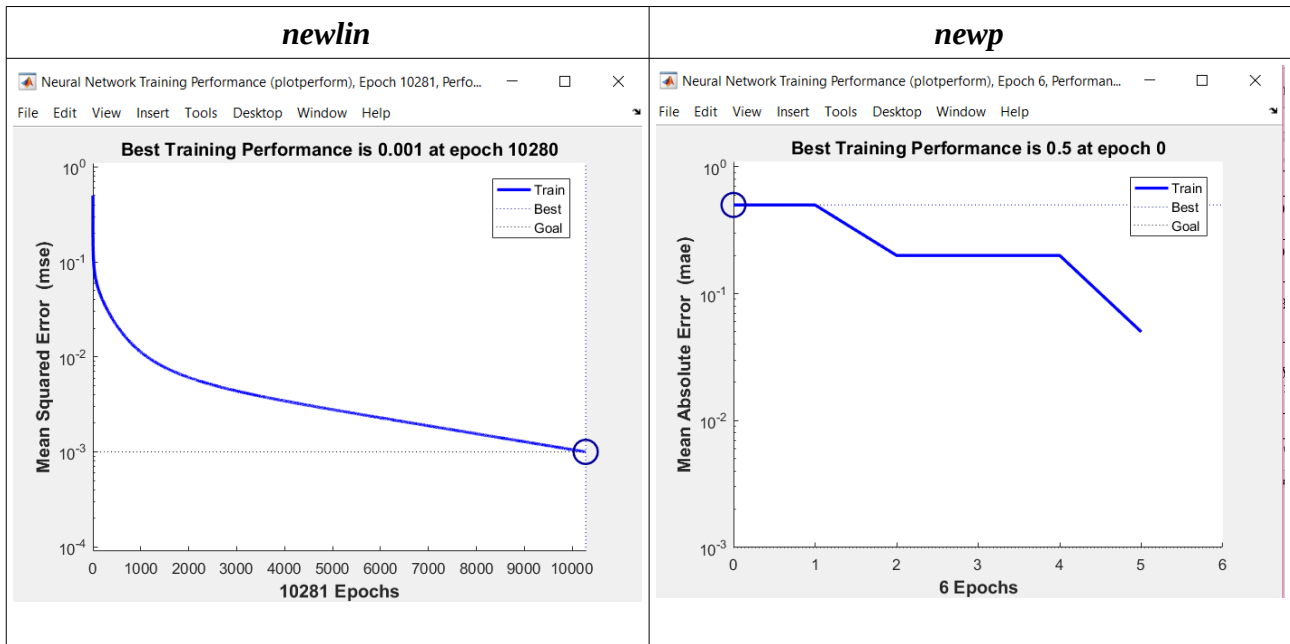
W poniższej tabeli zgromadzono dane i liczby obrazujące efekty uczenia sieci obiema metodami (zmienna *efekt*), ilość epok potrzebnych do nauczania sieci w zależności od zmiany współczynnika uczenia, jak również błędu średnio kwadratowego.

Błąd ś.k.	0.001			0.01			0.1		
Wsp. uczenia	0.001	0.01	0.1	0.001	0.01	0.1	0.001	0.01	0.1

Lite ra\ Fun kcja	new p	newl in	new p	newl in	new p	newl in	new p	newl in	new p	newl in	new p	newl in	new p	newl in	new p	newl in	new p	newl in
A	1	0.02 71	1	0.02 71	1	0.02 710	1	- 0.06 39	1	- 0.06 41	1	- 0.06 41	1	1.01 22	1	1.02 16	1	1.02 16
a	0	0.03 6	0	0.00 36	0	0.00 36	0	0.01 05	0	0.01 05	0	0.01 05	0	0.14 60	0	0.14 20	0	0.14 20
B	1	1.02 26	1	1.02 26	1	1.02 26	1	1.05 00	1	1.05 01	1	1.05 01	0	0.57 67	0	0.57 46	0	0.57 46
b	0	- 0.03 71	0	- 0.03 71	0	- 0.03 71	0	0.04 81	0	- 0.05 16	0	- 0.05 16	0	0.30 74	0	0.30 30	0	0.30 30
C	1	0. 9452	1	0.94 52	1	0.94 52	1	1.00 73	1	1.00 73	1	1.00 73	1	0.99 44	1	1.00 00	1	1
c	0	- 0.20 43	0	- 0.20 43	0	- 0.20 43	0	- 0.83 87	0	- 0.83 88	0	- 0.83 88	0	- 0.19 55	0	- 0.20 20	0	- 0.20 20
D	1	- 0.37 26	1	- 0.37 27	1	- 0.37 27	1	0.58 54	1	0.58 54	1	0.58 54	1	0.74 60	1	0.74 73	1	0.74 73
d	0	0.02 23	0	0.02 23	0	0.02 23	0	0.04 81	0	0.04 81	0	0.04 81	0	0.04 19	0	0.03 66	0	0.03 66
E	1	1.03 69	1	1.03 69	1	1.03 69	1	1.00 09	1	1.00 09	1	1.00 09	1	0.81 92	1	0.82 08	1	0.82 08
e	0	- 0.03 36	0	- 0.03 36	0	- 0.03 36	0	- 0.06 13	0	- 0.06 13	0	- 0.06 13	0	0.30 04	0	0.29 81	0	0.29 81
F	1	1.00 14	1	1.00 14	1	1.00 14	1	0.94 16	1	0.94 16	1	0.94 16	1	0.79 14	1	0.79 42	1	0.79 42
f	0	0.03 55	0	0.03 55	0	0.03 55	0	0.13 15	0	0.13 16	0	0.13 16	0	0.42 67	0	0.42 70	0	0.42 70
H	1	0.99 55	1	0.99 55	1	0.99 55	1	1.00 64	1	1.00 64	1	1.00 64	1	0.67 91	1	0.67 90	1	0.67 90
h	0	0.44 46	0	0.44 45	0	0.44 45	0	0.54 13	0	0.54 14	0	0.54 14	0	0.40 76	0	0.40 53	0	0.40 53
I	1	0.96 49	1	0.96 49	1	0.96 49	1	0.79 89	1	0.79 90	1	0.79 90	1	0.56 10	1	0.56 31	1	0.56 31
i	0	0.00 45	0	0.00 45	0	0.00 45	0	0.05 88	0	0.05 88	0	0.05 88	0	0.41 66	0	0.41 80	0	0.41 80
K	1	0.21 44	1	0.21 44	1	0.21 44	1	0.61 84	1	0.61 84	1	0.61 84	1	0.75 36	1	0.75 74	1	0.75 74
k	0	0.77	0	0.77	0	0.77	0	0.43	0	0.43	0	0.43	0	0.36	0	0.36	0	0.36

		70		7		70		24		24		24		23		09		09
<b>L</b>	1	1.00 04	1	1.00 04	1	1.00 04	1	0.85 82	1	0.85 83	1	0.85 83	1	0.58 88	1	0.58 97	1	0.58 97
<b>I</b>	0	0.09	0	0.09	0	0.09 00	0	0.29 66	0	0.29 66	0	0.29 66	0	0.52 18	0	0.52 20	0	0.52 20
<b>Licz ba epok</b>	6	1028 06	6	1028 1	6	1028 1	6	1133 2	6	1133	6	1133	5	161	5	16	5	16

Wygenerowałam również wykresy obrazujące, jak długo (ile epok) i z jakim błędem średniokwadratowym zmienia się funkcja ucząca, dla wsp. uczenia równemu 0.001.



## Wnioski

- Obie funkcje generują błędy, ponieważ sieci są tylko imitacją myślenia człowieka, nie zaś jego realnym odzwierciedleniem.
- Warto zauważyć, że dokładności zmiennej *efekt* są różne dla *newlin* i *newp* – dla funkcji *newp* otrzymujemy dokładność do jedności, natomiast dla funkcji *newlin* przyjmujemy wartości z dokładnością  $10^{-4}$ .
- Funkcja *newlin* generuje większą ilość błędów niż funkcja *newp*. Może to wynikać z faktu, że dane wejściowe przechodzące przez perceptron obliczane są przez tyle współczynników wagowych, ile jest danych wejściowych. Natomiast w funkcji *newlin* dane wejściowe przechodzą przez ilość wag równej kwadratowi ilości danych wejściowych. Każda z tych wag zawiera również swoje błędy średniokwadratowe. Takie błędy potrafią mieć ogromne znaczenie dla wyniku zmiennych.
- Najmniejsze błędy generują funkcje, gdzie parametry uczenia oraz błąd średniokwadratowy są sobie równe.
- Najszybciej sieć neuronowa uczy się, gdy błąd średniokwadratowy jest najniższy. Warto jednak zauważyć, że nie zawsze dłuższy czas nauki gwarantuje większą skuteczność treningu.

- Wbrew oczekiwaniom narzędzie *newlin* wykazuje gorszą skuteczność dla sieci jednowarstwowych, jak również jest narzędziem bardzo niestabilnym w wypadku elementarnych problemów.

## Listing kodu źródłowego

close all; clear all; clc;

```
wart_in = [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]; %wart_ in - jakie wartości może przyjmować  
węzeł wejściowy  
% -> ponieważ nasze obrazy przyjmują rozmiar 5x5,  
%zatem węzeł wejściowy może przyjąć 25 wartości 0 lub 1  
%ustawione pionowo
```

```
wart_out = 1; %ilość wyjść z sieci
```

```
%metoda 1 - perceptron
%net = newp(wart_in, wart_out);
%metoda 2 - tworzenie sieci jednowarstwowej
net = newlin(wart_in, wart_out, 0, 0.01);
```

### %kolumnowa reprezentacja binarna każdej litery

% A a B b C c D d E e F f H h I i K k L l

```
Learn_in = [0 0 1 1 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:

```
000000000000000100000000;
```

1 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1;

```
000000000000000000000000;
```

```
000000000000000000000000;
00000000000000000000000100;
```

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

10000000000000011000000;

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 0 0 1 0 0 0;

0 1 1 1 0 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;

```
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 1 0 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;

```
1000000000000000000000000000000;
```

`10000000000000000000000000000000;`

```
1 1 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0:
```

```
11110011000001000000,
10000000000010000000:
```

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 0 1 1;

0 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:

$$1100000000000100000000];$$

```
Learn_out = [1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0]; % 1 - duża litera, 0 - mała litera
```

```

net.name = 'Wielkosc liter';

net.trainParam.epochs = 30000;
net.trainParam.goal = 0.001; %błąd średniokwadratowy
net.trainParam.mu = 0.01; %nie jest potrzebna w metodzie 2, gdzie
%określiliśmy współczynnik uczenia
%net.trainParam.showWindow = false;

net = train(net, Learn_in, Learn_out); %uczenie sieci

%littery 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_1 = [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];
```

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

```
[efekt1 efekt2]
```

```
if round(efekt1) <= 0 %zaokrąglenie  
    disp('Mała litera');  
else  
    disp('Wielka litera');  
end
```

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