Michalina Nikiel
WIMiIP, Inżynieria Obliczeniowa
rok 3, grupa laboratoryjna nr 2

Podstawy sztucznej inteligencji – projekt nr 2 – sprawozdanie

Budowa i działanie sieci jednowarstwowej

Cel ćwiczenia:

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

Zrealizowane kroki:

- Wygenerowanie danych uczących i testujących, zawierających 10 dużych i 10 małych liter alfabetu w postaci dwuwymiarowej tablicy.
- Przygotowanie dwóch jednowarstwowych sieci.
- Uczenie sieci
- Testowanie sieci.

Sprawozdanie zawiera kolejno:

- Zagadnienia teoretyczne: opis budowy sieci i algorytmów uczenia.
- Zestawienie otrzymanych wyników w postaci tabeli i zrzutów ekranu wyników programu.
- Analiza błędów oraz wnioski i podsumowanie
- Listing programu

Zagadnienia teoretyczne:

Sieć neuronowa – nazwa struktur matematycznych i ich programowych lub sprzętowych modeli. Wykonuje obliczenia za pomocą powiązanych ze sobą elementów (neuronów). Sieć neuronowa została zaprojektowana w taki sposób by przypominać działanie ludzkiego mózgu złożonego z naturalnych neuronów i łączących je synaps.

Sieć jednokierunkowa – jest to sieć neuronowa, w której nie występuje zjawisko sprzężenia zwrotnego oznacza to, iż każdy sygnał przechodzi przez każdy neuron dokładnie raz w swoim cyklu.

Opis użytych metod i funkcji:

- **newlin(start, wyjscia_s)** - tworzy jednowarstwową sieć neuronową, przyjmuje odpowiednio argumenty:
    - **start** – wartości minimalne i maksymalne dla elementów wejściowych sieci dla funkcji tworzących sieć neuronową, składa się z 25 par, które odpowiednio oznaczają wartość maksymalną - 1 oraz minimalną - 0,
    - **wyjscia_s** – liczba elementów wektora wyjściowego sieci, jego wartość jest równa 1 (przechowuje informacje czy dana litera jest duża, czy też nie).
- **newp(start, wyjscia_s, TF, LF)** - tworzy prostą sieć neuronową, funkcja ta pobiera dane wejściowe, a następnie zwraca perceptron. Jej argumentami podobnie jak w funkcji newlin są start i wyjscia_s, ale dodatkowo może być złożona z:
    - **TF** – funkcji transferu, gdy nie zostanie podana domyślnie przyjmuje 'hardlim',
    - **LF** – funkcji uczenia, gdy nie zostanie podana domyślnie przyjmuje 'learnp'.
- **net** – zmienna, do której będzie przypisywana nowa tworzona sieć neutronowa,
- **WEJSCIE** – dane uczące sieci neuronowej (litery alfabetu zapisane kolumnowo),
- **WYJSCIE** – zmienna przechowujące dane wyjściowe odpowiadające danym uczącym (gdzie 1 odpowiada dużej literze, a 0 małej),
- **net.trainParam.x** – określenie parametrów treningu sieci (maksymalnej liczby epok, błąd średniokwadratowy, wartość współczynnika uczenia się sieci),
- **train(net, WEJSCIE, WYJSCIE)** – uczenie (trening) sieci net z wykorzystaniem danych wejściowych i wyjściowych,
- ***_testowe** – zmienna dla danych testujących (w miejscu gwiazdki odpowiednio dla danej litery, np. A_testowe, a_testowe),
- **efekt = sim(net, *_test)** – symulacja sieci net, parametrami funkcji sim są odpowiednio sieć oraz litera, która ma być rozpoznana przez sieć, na końcu wynik testu zostaje wpisany do zmiennej efekt,
- **if round(efekt) == 0 { disp('Mala litera'); } else { disp('Duza litera'); } end** – jest to zaokrąglona wartość efekt, na której podstawie zostaje wypisany komunikat czy podana litera jest duża - 1 czy mała - 0 (zaokrąglenie: 1 dla wartości >= 0.5 i 0 dla pozostałych).

Zestawienie otrzymanych wyników w postaci tabeli:

Współczynnik uczenia → / Litera ↓ Funkcja →	0.001		0.01		0.1	
	newlin	newp	newlin	newp	newlin	newp
A	0.9510	1.0000	0.9510	1.0000	0.9510	1.0000
a	0.0628	0.0000	0.0628	0.0000	0.0628	0.0000
B	0.9369	1.0000	0.9369	1.0000	0.9369	1.0000
b	0.0135	0.0000	0.0135	0.0000	0.0135	0.0000
L	0.8603	1.0000	0.8603	1.0000	0.8603	1.0000
l	0.3100	0.0000	0.3100	0.0000	0.3100	0.0000
K	1.0042	1.0000	1.0042	1.0000	1.0042	1.0000
k	0.0011	0.0000	0.0011	0.0000	0.0011	0.0000
H	1.0243	1.0000	1.0243	1.0000	1.0243	1.0000
h	0.0089	0.0000	0.0089	0.0000	0.0089	0.0000
Liczba epok	1792	8	1792	8	1792	4

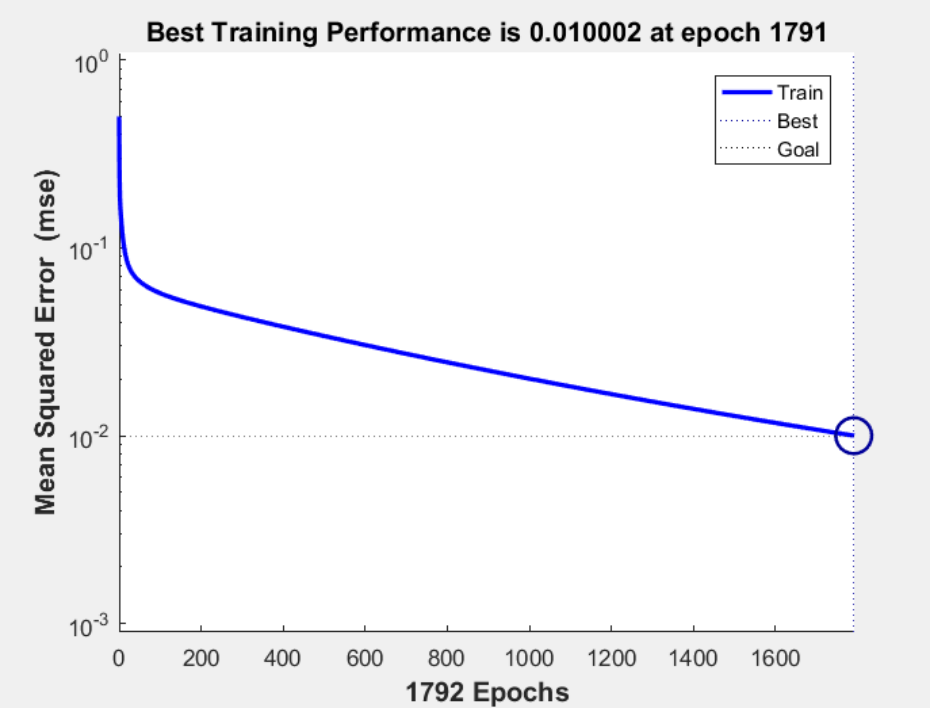
Tabela przedstawia wartości zmiennej **efekt:**

```
%testowanie dzialania sieci
efekt = sim(net, A_testowe)
```

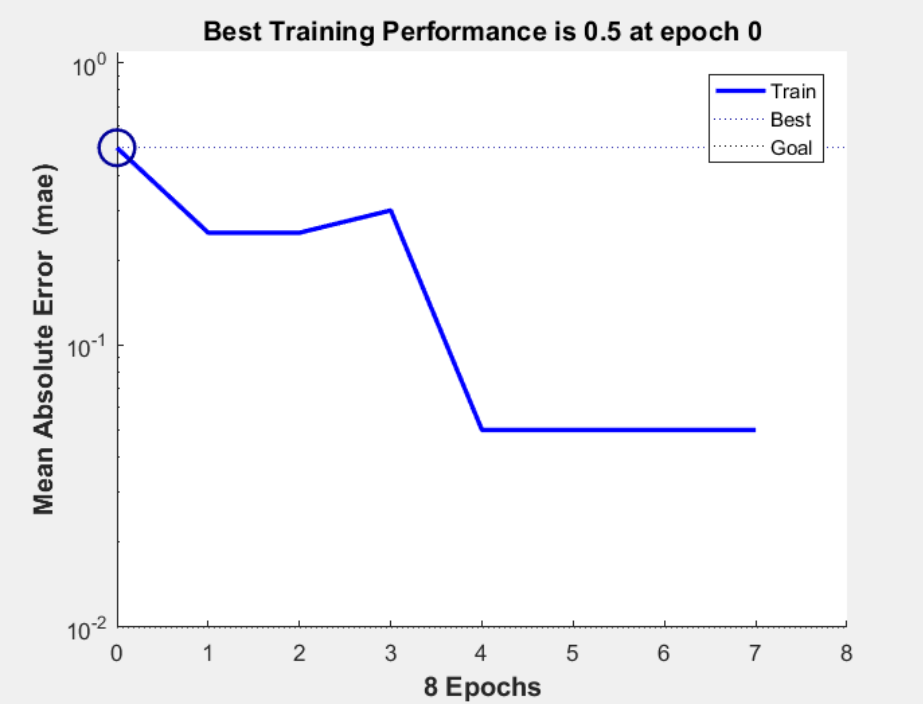
po kompilacji programu dla funkcji newlin oraz newp. Wszystkie cyfry zostały zaokrąglone do czwartego miejsca po przecinku, im wygenerowana cyfra jest bliższa wartości równej 1 oznacza to, że testowana litera jest literą dużą. Dodatkowo w tabeli przedstawiona została liczba epok.

Zrzuty ekranu działania programu:

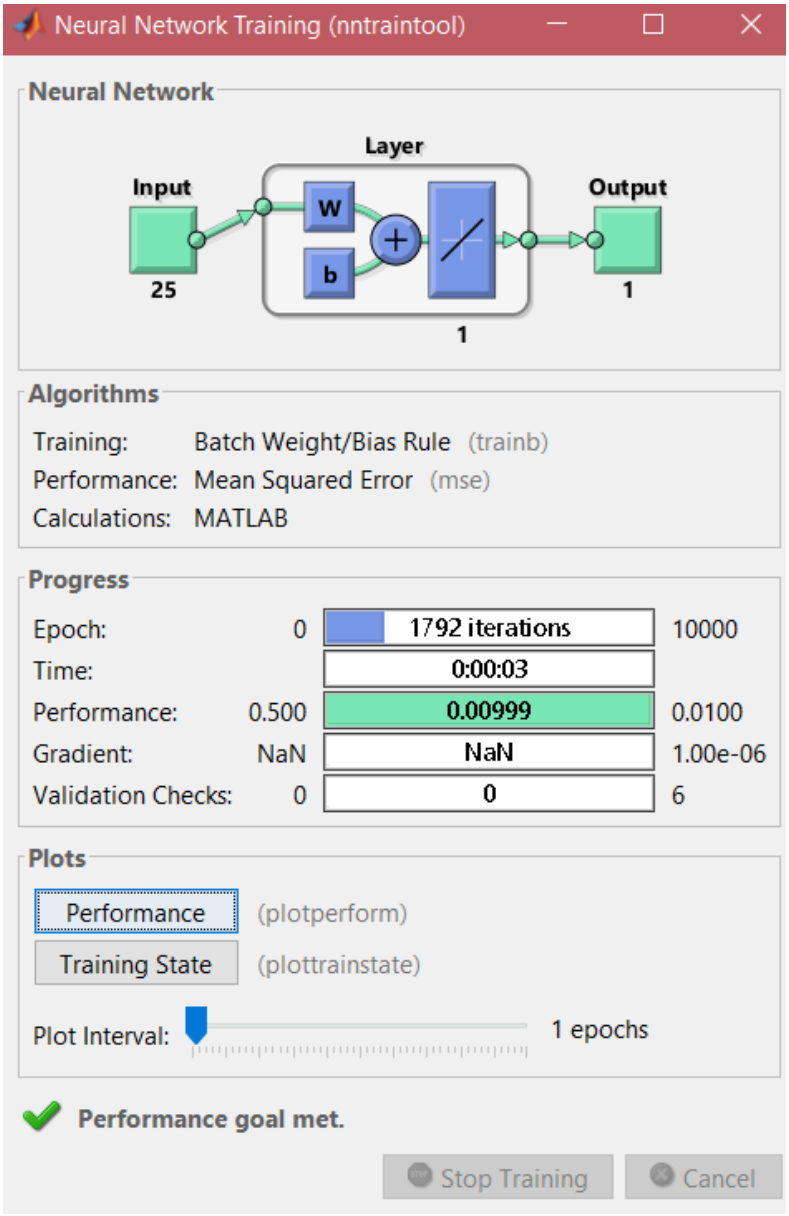
- Zrzut ekranu dla funkcji newlin dla litery „A” przy ustawieniu współczynnika uczenia na 0.001 oraz błędu średniokwadratowego na 0.01



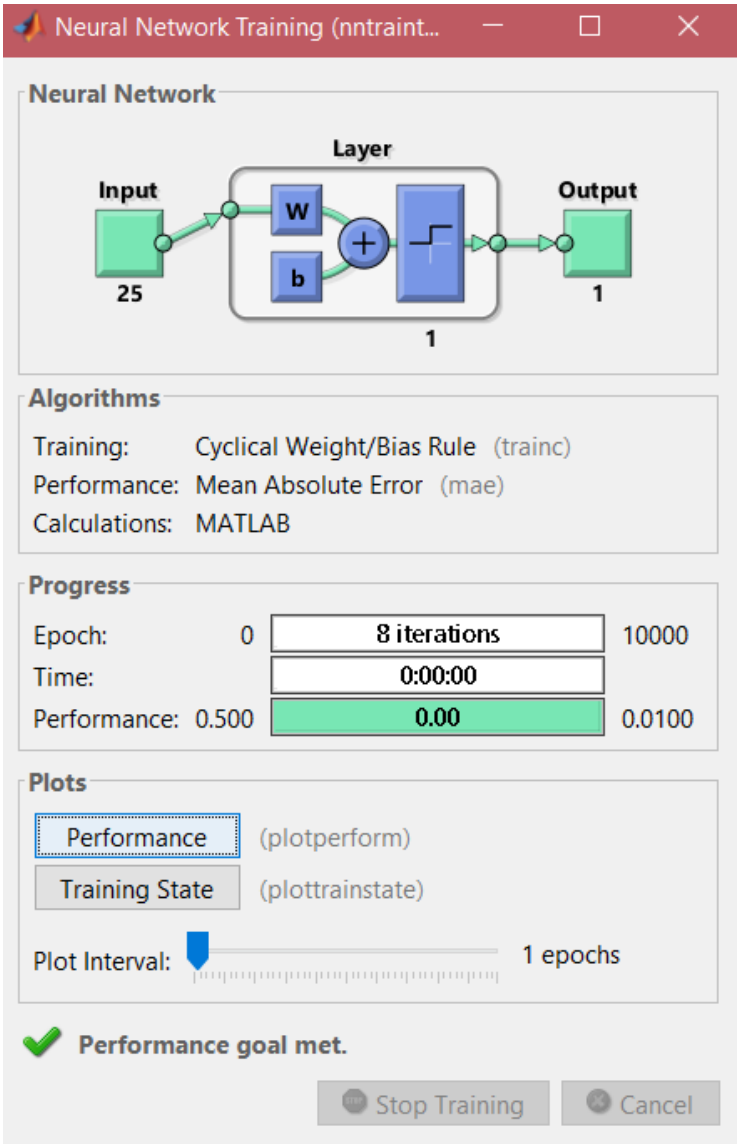
- Zrzut dla funkcji newp dla litery „A” przy ustawieniu współczynnika uczenia na 0.001 oraz błędu średniokwadratowego na 0.01



- Zrzut ekranu wyniku działania programu dla funkcji newlin dla litery „A” przy ustawieniu współczynnika uczenia na 0.001 oraz błędu średniokwadratowego na 0.01.



- Zrzut ekranu wyniku działania programu dla funkcji newp dla litery „A” przy ustawieniu współczynnika uczenia na 0.001 oraz błędu średniokwadratowego na 0.01.



Analiza, podsumowanie i wnioski:

- Zmiana wartości parametrów współczynnika uczenia, nie ma wpływu na otrzymane wyniki, czy też skrócenie czasu uczenia się z zastosowaniem poszczególnych algorytmów.
- Stosując funkcję newp znacznie zmniejszamy liczbę iteracji(epok) potrzebnych do uzyskania dobrych wyników.
- Funkcja newp dawała lepsze efekty w porównaniu do newlin - do uczenia sieci potrzebowała zdecydowanie mniejszą liczbę epok (co miało wpływ na długość działania programu, im mniejsza liczba epok, tym program działał szybciej). Ponadto analizując tabelę można zauważyć, że newp posiadała wartości równe 1 lub 0, natomiast funkcja newlin dawała zdecydowanie mniej klarowne wyniki (takie wyniki mogły zwiększyć ilość występowania błędów).
- Funkcja newlin była zależna od błędu średniokwadratowego, im mniejsza była jego wartość, tym liczba epok potrzebnych do uczenia sieci się zwiększała.
- Program zwraca dokładnie wartość jeden lub zero, w zależności od wyniku (0 - mała, 1 - duża).
- Program jest bardzo szybki, można to dostrzec po ilości iteracji które potrzebuje - zastosowanie innej metody może sprawić że czas uczenia bardzo drastycznie wzrośnie.

Listing programu:

```
%%
close all; clear all; clc;

%wejscia do sieci oraz minimalne oraz maksymalne wartosci
wejsc
start = [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;];
%wyjscia sieci (maksymalna ilosc wyjsc)
wyjscia_s = 1;
net = newlin(start, wyjscia_s); %metoda 1
%net = newp(start, wyjscia_s); %metoda 2
%kolumnowa reprezentacja binarna pierwszych 10 liter
alfabetu dla tablicy 5x5
%A a B b C c D d E e F f H h I i K k L l
WEJSCIE = [0 0 1 1 0 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 1 0 0 0;
0 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 0 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;
1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0;
1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 0;
1 1 0 0 0 0 1 1 0 0 0 0 1 0 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;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 1 1 1 0 0 1 1 0 0 0 0 0 0 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 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1;
0 1 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 1 0 0 0 1 0 0 0 1 1 1;
0 1 0 0 1 0 0 1 1 0 0 0 0 0 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;];
7
%A a B b C c D d E e F f H h I i K k L l
WYJSCIE = [1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];
%parametry treningu sieci
net.trainParam.epochs = 10000; %maksymalna ilosc epok
net.trainParam.goal = 0.01; %blad sredniokwadratowy
net.trainParam.mu = 0.001; %wspolczynnik uczenia sieci
%uczenie sieci (train)
net = train(net, WEJSCIE, WYJSCIE);
%dane testowe
A_testowe = [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];
a_testowe = [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];
B_testowe = [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];
b_testowe = [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];
C_testowe = [0; 1; 1; 1; 0;
1; 0; 0; 0; 0;
1; 0; 0; 0; 0;
1; 0; 0; 0; 0;
0; 1; 1; 1; 0];
c_testowe = [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];
D_testowe = [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];
d_testowe = [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];
E_testowe = [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];
8
e_testowe = [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];
F_testowe = [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];
f_testowe = [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];
H_testowe = [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];
h_testowe = [1; 0; 0; 0; 0;
1; 0; 0; 0; 0;
1; 1; 1; 0; 0;
1; 0; 1; 0; 0;
1; 0; 1; 0; 0];
I_testowe = [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];
i_testowe = [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];
K_testowe = [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];
k_testowe = [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];
L_testowe = [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];
l_testowe = [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];
9
%testowanie dzialania sieci
efekt = sim(net, A_testowe)
%wypisanie wyniku uczenia przez siec
if round(efekt) == 0
disp('Mala litera');
else
disp('Duza litera');
end
```