Paulina Reichel, IO gr.2

Temat projektu:

Budowa i działanie sieci jednowarstwowej.

Cel:

- ✓ 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 jednej litery. przykładowe
- ✓ Przygotowanie dwóch jednowarstwowych sieci - każda wg. innego algorytmu
- ✓ Uczenie sieci dla przy różnych współczynnikach uczenia
- ✓ Testowanie sieci

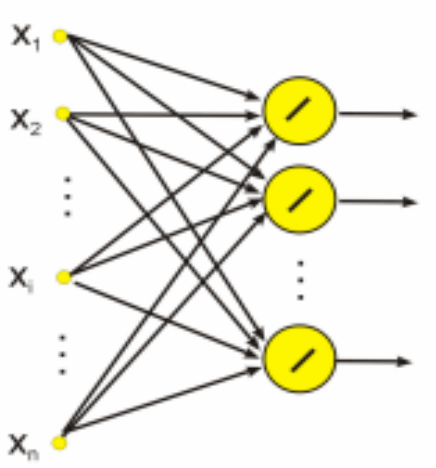
Teoria:

Sieć neuronowa:

nazwa struktur matematycznych i ich programowych lub sprzętowych modeli, realizujących obliczenia lub przetwarzanie sygnałów poprzez rzędy elementów, zwanych sztucznymi neuronami, wykonujących pewną podstawową operację na swoim wejściu. Oryginalną inspiracją takiej struktury była budowa naturalnych neuronów, łączących je synaps, oraz układów nerwowych, w szczególności mózgu

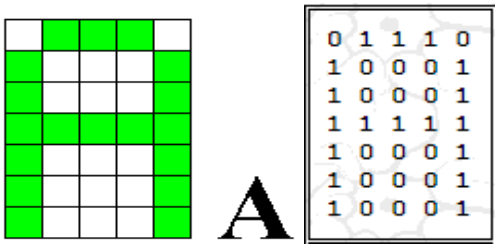
Sieć jednokierunkowa:

- ✓ neurony ułożone w jednej warstwie zasilanej z węzłów wejściowych,
- ✓ połączenie węzłów wejściowych i wyjściowych jest pełne (każdy węzeł jest połączony z każdym neuronem),
- ✓ przepływ sygnału występuje w jednym kierunku, od wejścia do wyjścia,
- ✓ węzły wejściowe nie tworzą warstwy neuronów, gdyż nie zachodzi w nich żaden proces obliczeniowy,
- ✓ sieć tego rodzaju nazywa się perceptronem jednowarstwowym



Przygotowanie danych uczących:

- ✓ Na proces uczenia się sieci jednowarstwowej skład się kilka podstawowych etapów takich jak: wybór zbioru uczącego, architektury sieci, dobór parametrów uczenia, prezentacja zbioru uczącego, modyfikacja wag i wreszcie powtarzanie procesu uczenia aż do osiągnięcia warunków zakończenia.
- ✓ W zrealizowanym projekcie dane uczące to zgodnie z instrukcją 20 liter zawierających 10 dużych i 10 małych liter. . Litery te są utworzone na matrycy 5 x 5 czyli na 25 polach, co jest równoznaczne z ilością wejść.
- ✓ Przykład przygotowania danych dla litery „A":



Widok tablicowy: A = [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]

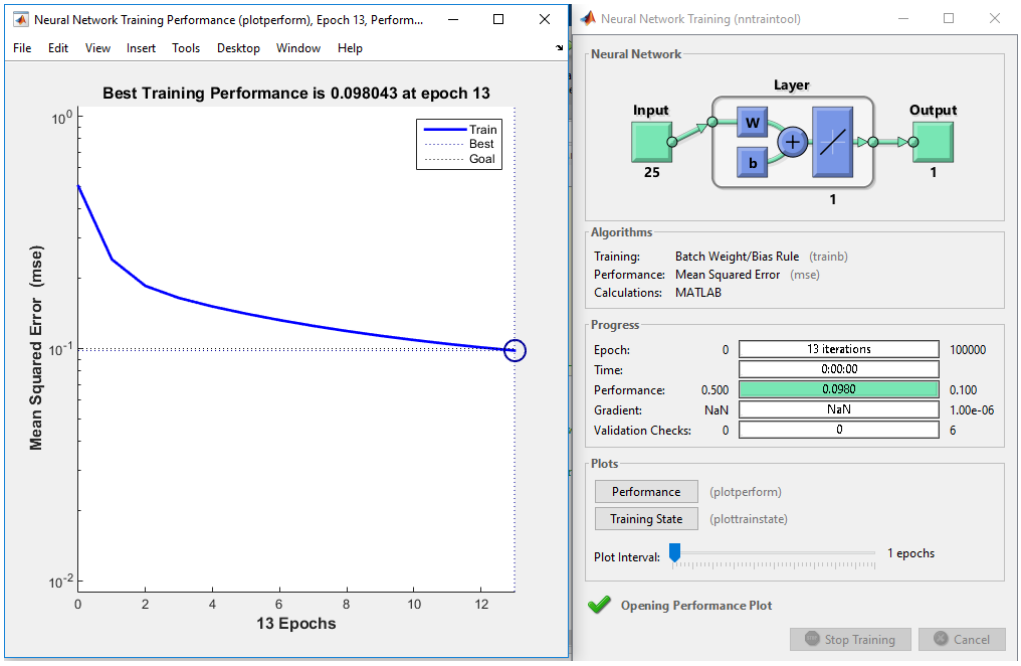
- ✓ Wszystkie dane wejściowe zostały przedstawione w tabeli „iData", w której każdy wektor zajmuje jedną kolumnę, stąd dane wyjściowe to tablica.
- ✓ Funkcje 'newp' oraz 'newlin' ze zmiennymi parametrami uczenia (epokami, średnim błędem kwadratowym oraz współczynnikiem uczenia) zostały wykorzystane do przeprowadzenia ćwiczenia (opis parametrów jak i szczegółowy opis funkcji zostały zawarte w listingu programu w komentarzach)

Funkcja	newlin				newp			
Błąd średniokw.	0.001		0.01		0.001		0.01	
Współczynnik uczenia	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01
Potrzebne epoki	5406	5406	1792	1792	8	8	8	8
Przybliżenia dla poszczególnych liter								
A	0.9835	0.9835	0.9510	0.9510	1	1	1	1
a	0.0152	0.0152	0.0628	0.0628	0	0	0	0
B	1.0168	1.0168	0.9369	0.9369	1	1	1	1
b	-0.0204	-0.0204	0.0135	0.0135	0	0	0	0
C	1.0079	1.0079	0.9878	0.9878	1	1	1	1
c	-0.0425	-0.0425	-0.1745	-0.1745	1	1	1	1
D	0.9637	0.9637	0.9162	0.9162	0	0	0	0
d	0.0232	0.0232	0.0588	0.0588	1	1	1	1

Przykładowe zrzuty ekranu działania programu:

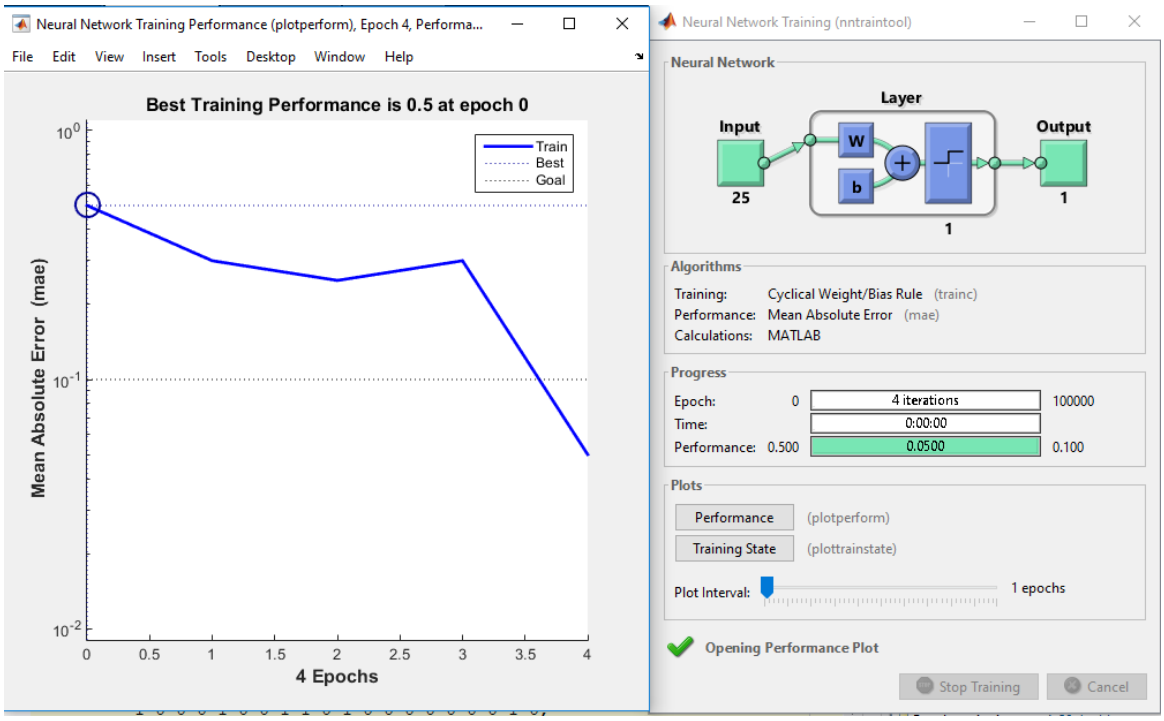
Dla funkcji newlin:

- ✓ Błąd średniokwadratowy 0.1,
- ✓ Współczynnik uczenia 0.1
- ✓ Litera F



Dla funkcji newp:

- ✓ Błąd średniokwadratowy 0.1,
- ✓ Współczynnik uczenia 0.1
- ✓ Litera F



Analiza i wnioski:

- ✓ Udało się zrealizować wszystkie kroki
- ✓ zmiana wartości parametrów współczynnika uczenia, nie ma wpływu na otrzymane wyniki, czy też skrócenie czasu uczenia się z zastosowaniem poszczególnych algorytmów
- ✓ Zmniejszenie błędu średnio kwadratowego ma znaczny wpływ na szybkość nauki jak i dokładność uzyskiwanych w testach wyników
- ✓ Wraz ze zmniejszeniem błędu średniokwadratowego, np. z 0,1 na 0,001, zarówno dokładność wyników jak i czas potrzebny na naukę zwiększa się. Przykładowo w programie z wykorzystaniem newlin dla litery B, gdzie zwiększenie błędu pozwoliło zmniejszyć liczbę epok z 5406 na 13, tracona jest dokładność (przybliżenie dla błędu 0,001 wynosiło 1,0168 natomiast dla błędu 0,1 już 0,5261)
- ✓ Funkcja newp pozwala na zmniejszenie liczby iteracji(epok) potrzebnych do uzyskania dobrych wyników
- ✓ Przy zastosowaniu funkcji newp, nieoptymalne jest wprowadzanie większego błędu z wykorzystaniem takiego zestawu danych, gdyż może to prowadzić do błędu (np. w przypadku litery B).
- ✓ Podstawową różnicą między dwoma zastosowanymi metodami jest fakt, że przy użyciu *newlin* wszystkie parametry mają duży wpływ na proces uczenia i przede wszystkim na jego rezultaty, co widać w załączonej tabeli, natomiast *newp* w każdych warunkach radziła sobie równie dobrze. Znaczą rolę odgrywa tutaj funkcja aktywacji. Pierwsza z wspomnianych metod wykorzystuje liniowe funkcje aktywacji, które mają ograniczone możliwości i wykorzystuje się je najczęściej do prostych klasyfikacji w przeciwieństwie do *hardlimit*.
- ✓ Przeprowadzone badanie udowadnia, że nie zawsze jedna metoda będzie idealna dla każdego problemu i warto dostosować wykorzystywane narzędzia do poziomu trudności zagadnienia.

Listing całego programu z komentarzami opisującymi funkcje i parametry:

```
%% close all; clear all;
clc;
%-------------------------------------------------------------------------
%PR - min / max wartość x ilość wejść (każda matryca 5x5 = 25 wejść)
PR=[01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01;01
;01;];
PR2= [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; ];
%-------------------------------------------------------------------------
%S - liczba neuronów sieci
S = 1;
%-------------------------------------------------------------------------
%net - struktura, zawierająca opis architektury, metod treningu, wartości
%liczbowe wag i progów oraz inne parametry sieci perceptronowej
%-------------------------------------------------------------------------
%wersje algorytmu ------------- zakomentować/odkomentować odpowiedni net
%net=newlin(PR,S);
net=newp(PR2,S);
%-------------------------------------------------------------------------
%newlin - Tworzenie jednowarstwowej sieci złożonej z neuronów liniowych
%newp - Tworzenie jednowarstwowej sieci złożonej z twardych perceptronów
```

```
%TL - Nazwa funkcji aktywacji neuronów domyślnie = 'hardlim'
%trainlm - Nazwa funkcji trenowania sieci perceptronowej domyślnie =
%'learnp'
%-------------------------------------------------------------------------
%Dane wejściowe (każdy wektor wejściowy zajmuje jedną kolumnę)
% G oraz J zostały pominięte dla ułatwienia (możliwość realizacji przy
% matrycach o większych wymiarach

%         A a B b C c D d E e F f H h I i L l K k
iData = [ 0 0 1 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 1 0 0 0 0 0 0 0 0;
          1 1 1 0 1 0 1 0 1 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 0 0 1 0;
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
1 0 1 1 1 0 1 0 1 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 0 0 0 0 1 0;
0 1 1 0 0 0 1 1 0 1 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 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 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 1 0;
          1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 1;
          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 0 0 1;
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0;
0 1 1 1 0 0 1 1 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 1 0 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 1 1 0 0;
          0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 0 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 0;];
%-------------------------------------------------------------------------

%Dane testowe.:

A = [ 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 = [ 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 = [ 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 = [ 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 = [ 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 = [ 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 = [ 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 = [ 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 = [ 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 ];
```

```
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 ];
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 ];
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 ];
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 ];
h = [ 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 = [ 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 = [ 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 ];
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 ];
l = [ 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 ];
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 ];
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 ];
%-------------------------------------------------------------------------

%Dane Wyjściowe (oznaczone 1 - duża litera, 0 - mała litera, D-duża, m-mała
%         A a B b C c D d E e F f H h I i L l K k %
D m D m D m D m D m D m D m D m D m D m D m oData = [ 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 ];
%-------------------------------------------------------------------------
% Zmienne parametrów uczenia się

% Maksymalna liczba epok trwania
trningu net.trainParam.epochs = 100000;
% Błąd średniokwadratowy
net.trainParam.goal = 0.001; %
Współczynnik uczenia net.trainParam.mu
= 0.001;
%-------------------------------------------------------------------------
%Przed treningiem
Przed_treningiem_dane=sim(net,iData);
%-------------------------------------------------------------------------
%Trening
net=train(net,iData,oData);
%-------------------------------------------------------------------------
%%
%Po treningu
%-------------------------należy wpisać literę jaką ma sprawdzić!
Po_treningu_dane=sim(net,B);
%-------------------------------------------------------------------------
%Przybliżenie i przypisanie wartości po treningu dla ustalenia wielkości
%litery
if round(Po_treningu_dane)==0, disp('mala
litera'); else disp('Duza litera'); end;
disp(Po_treningu_dane)
```