|  |  |  |
| --- | --- | --- |
| Grupa ćwicz. **2** | Data wykonania 31.10.2017 | Nr. Scenariusza  **2** |
| **Temat ćwiczenia:** Budowa i działanie sieci neuronowej | | |
| Imię i nazwisko  **Andrzej Pawlikowski** | | Ocena i Uwagi |

**Sieci jednokierunkowe**

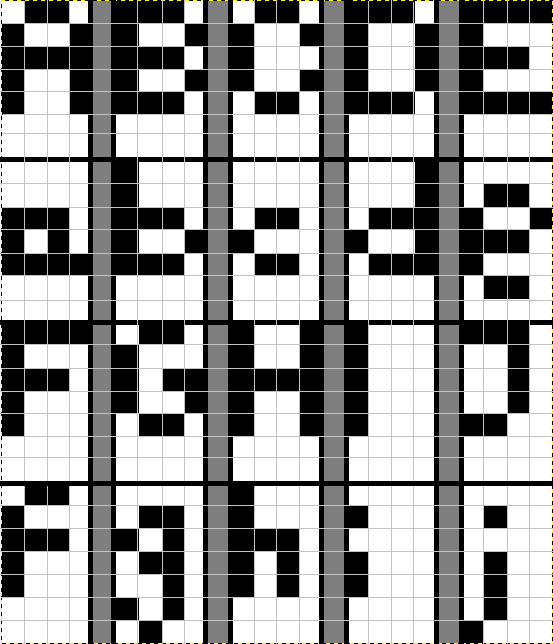
Sieci jednokierunkowe to sieci neuronowe, w których nie występuje [sprzężenie zwrotne](https://pl.wikipedia.org/wiki/Sprz%C4%99%C5%BCenie_zwrotne), czyli pojedynczy wzorzec lub sygnał przechodzi przez każdy neuron dokładnie raz w swoim cyklu.

**Cel ćwiczenia:**

Celem ćwiczenia jest poznanie budowy i działania jednowarstwowych sieci neuronowych oraz uczenie rozpoznawania wielkości liter.

**Wykonanie zadania:**

W celu realizacji projektu przygotowałem dane uczące składające się z dziesięciu dużych i dziesięciu małych liter alfabetu łacińskiego wygenerowanych w tablicy dwuwymiarowej 4x7.



Macierz reprezentującą dane uczące umieściłem w osobnym pliku. Plik ten składa się z 20 kolumn reprezentujących każdą literę oraz 28 wierszy wypełnionych odpowiednio zerami albo jedynkami.

W ramach ćwiczenia użyłem dwóch funkcji newlin oraz newp

**NEWLIN** - 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.

**NEWP –** tworzy jednowarstwową sieć neuronową złożoną z zadanej liczby perceptronów(sztuczny neuron z bipolarną lub unipolarną funkcją przejścia).

Parametry potrzebne do utworzenia sieci za pomocą tych funkcji:

newlin(pr, s, id, lr)

**newp(pr,s,tf,lf)**

pr – macierz(o rozmiarze Rx2) zawiera min i max dla R wejść sieci (w tym ćwiczeniu składa się z par 0 1)

s – liczba neuronów w sieci

id-wejściowy wektor opóźnienia, domyślnie = [0]

lr- współczynnik uczenia sieci, domyślnie=0.001

tf – funkcja aktywacji neuronów, domyślnie =’hardlim’

lf –nazwa funkcji używanej do modyfikacji wag; domyślnie=’learnp’

Przechodząc do uczenia naszej sieci pierwsze ustawiłem parametry treningu a następnie użyłem funkcji train której przekazujemy dane uczące (tablica z liter przekazana z osobnego pliku) oraz dane wyjściowe. Ustawiamy w nich odpowiednio 0 i 1 dla małych i dużych liter.

WY=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];

Manipulując współczynnikiem uczenia oraz błędem średniokwadratowym uzyskałem poniższe wyniki.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **newlin** | | | | |
| **współczynnik uczenia** | 0.01 | 0.001 | 0.0001 | 0.00001 |
| **epoki** | 6 | 54 | 589 | 5891 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Błąd śr.** | 0.1 | 0.01 | 0.001 |
| **Epoki** | 6 | 1581 | 35tys. |
| **A** | 0.8202 | 0.8853 | 0.9780 |
| **a** | 0.2271 | -0.0438 | -0.0131 |
| **B** | 0.8811 | 0.8888 | 1.0211 |
| **b** | 0.5121 | 0.0831 | -0.0007 |
| **C** | 0.7353 | 1.0428 | 0.9833 |
| **c** | 0.0930 | -0.1217 | -0.0574 |
| **D** | 0.9801 | 1.0410 | 0.9851 |
| **d** | 0.3631 | 0.1518 | 0.0727 |
| **E** | 0.7550 | 1.0213 | 1.0027 |
| **e** | 0.2048 | 0.0219 | 0.0004 |
| **F** | 0.7698 | 0.8686 | 0.9656 |
| **f** | 0.5995 | 0.2911 | 0.0603 |
| **G** | 0.7447 | 0.9223 | 1.0008 |
| **g** | 0.0603 | -0.0156 | -0.0019 |
| **H** | 0.6861 | 0.9667 | 0.9595 |
| **h** | 0.3846 | 0.0280 | 0.0099 |
| **I** | 0.4440 | 1.0538 | 1.0531 |
| **I** | 0.2489 | -0.0934 | -0.0204 |
| **J** | 0.5995 | 1.0350 | 1.0030 |
| **j** | 0.0561 | -0.0173 | 0.0002 |

**Funkcja uczenia learnp zastosowana dla metody newp:**

Algorytm zaimplementowany w tej funkcji oblicza zmianę wagi dW dla danego neuronu z wejścia P neuronu i błędu E zgodnie z zasadą uczenia perceptronu:

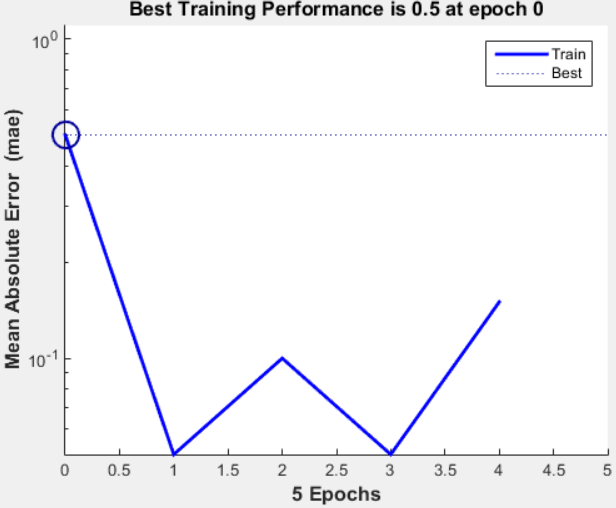
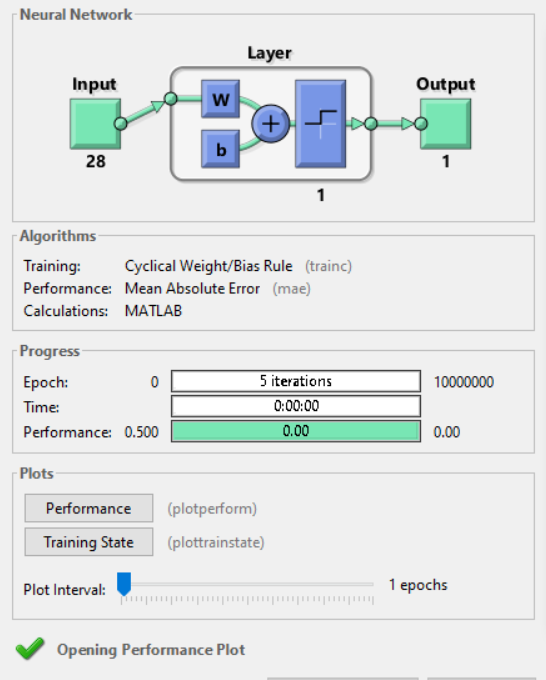
dw = 0, if e = 0

= p', if e = 1

= -p', if e = -1

Można to podsumować jako

dw = e\*p'



**Funkcja uczenia learnpn zastosowana dla metody newp:**

oblicza zmianę wagi dW dla danego neuronu z wejścia P neuronu i błędu E:

pn = p / sqrt(1 + p(1)^2 + p(2)^2) + ... + p(R)^2)

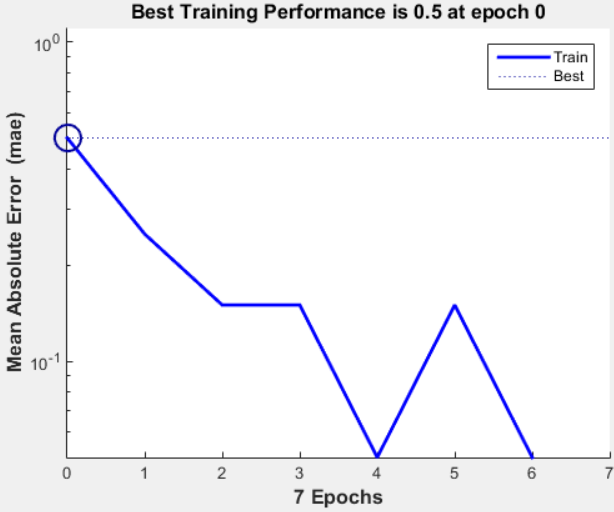
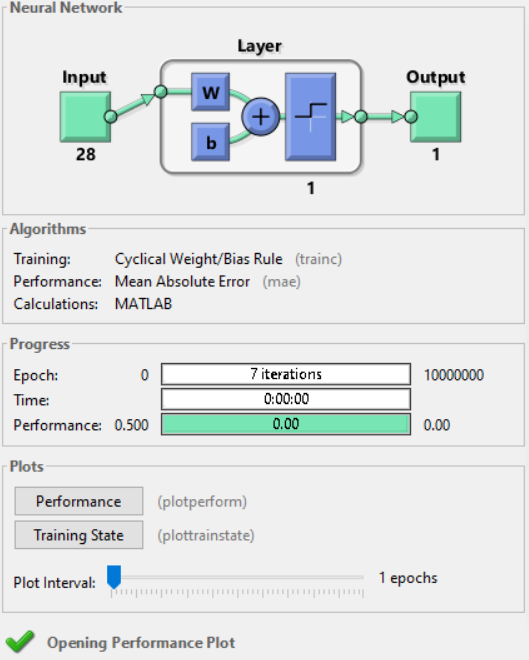
dw = 0, if e = 0

= pn', if e = 1

= -pn', if e = -1

Można to podsumować jako

dw = e\*pn'



**Wnioski:**

Dla funkcji newlin zmniejszając współczynnik uczenia zwiększała się ilość epok wyniki uczenia były takie same. Manipulując błędem średniokwadratowym uzyskiwaliśmy coraz dokładniejsze wyniki zmniejszając ten parametr. Ilość epok jednak wzrastała z dużo większą szybkością. Dla błędu średniokwadratowego 0.001 ilość epok wyniosła ponad 35tysięcy.

Porównując użyte funkcje newp daje lepsze efekty. Liczba epok i długość trwania procesu są znacznie mniejsze. Funkcja newp ogranicza ilość błędu efekt uczenia ponieważ daje dokładnie wartości 1 lub 0. Funkcje uczenia learnpn składa się z bardziej złożonego algorytmu. Nauczenie zajęło 7 epok dla algorytmu newlin wyniósł 5 epok.

W przypadku, gdy chce się szybko nauczyć sieć powinno się dać w miarę wysoki współczynnik uczenia ale należy liczyć się z tym, że algorytm będzie w mniej dokładny sposób rozróżniał duże litery od małych co w niektórych przypadkach może się przełożyć na niepoprawne wyniki.

Oba algorytmy nie popełniały błędów jeśli otrzymały pełny zestaw uczący, poprawnie rozróżniając duże litery od małych.

Problemy pojawiały się gdy wprowadzono do programu literę nieuwzględnioną w danych uczących – algorytmy porównywały ją do wcześniej otrzymanych danych co czasami skutkowało rozpoznaniem małej litery jako dużej lub odwrotnie.

W przypadku, gdy chce się szybko nauczyć sieć powinno się dać w miarę wysoki współczynnik uczenia ale należy liczyć się z tym, że algorytm będzie w mniej dokładny sposób rozróżniał duże litery od małych co w niektórych przypadkach może się przełożyć na niepoprawne wyniki.

W celu poprawienia skuteczności działania programu można wykorzystać macierze o większych rozmiarach co skutkowałoby spowolnieniem programu jednak zmniejszyło by prawdopodobieństwo pomyłki spowodowanych przez większą ilość porównywanych punktów.

Kod programu:

close all; clear all; clc;

PR=[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;];

S=1;

%net = newlin(PR,S,[0],0.01); %metoda 1

net = newp(PR,S,'hardlim','learnpn'); %metoda 2

%Wczytywanie danych z pliku

filename = 'alfabet.txt';

delimiterIn = ' ';

WE = importdata(filename,delimiterIn);

WY=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];

net.trainParam.epochs = 10000000;

%net.trainParam.goal = 0.01;

%net.trainParam.lr =0.01

net = train(net, WE, WY);

for i=1:20

%zmienna test jest wektorem przedstawiacym litere:

%A a B b C c D d E e F f G g H h I i J j

test=WE(:,i);

efekt=sim(net, test);%testowanie sieci

disp(efekt)

end

Dane uczące:

0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0

1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0 0 1 0

1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0 0 1 0

0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0

1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0

1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0

1 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0

1 1 1 1 0 1 0 1 1 0 1 1 0 0 1 1 0 0 0 0

1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0

1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0 0 0

1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1

0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0

1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0

1 1 1 1 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 0

0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1

0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 0

1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 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 0 1 0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 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 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 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 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