|  |  |  |
| --- | --- | --- |
| Grupa ćwicz. **2** | Data wykonania 17.11.2017 | Nr. Scenariusza  **3** |
| **Temat ćwiczenia:** Budowa i działanie sieci wielowarstwowej. | | |
| Imię i nazwisko  **Kamil Szczurkowski** | | Ocena i Uwagi |

**Cel ćwiczenia:**

Celem ćwiczenia jest poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie z użyciem algorytmu wstecznej propagacji błędu rozpoznawania konkretnych liter alfabetu.

**Wykonane zadania:**

1. Wygenerowano dane uczące i testujące, zawierające 20 dużych liter alfabetu łacińskiego w postaci dwuwymiarowej tablicy 7x5, reprezentowanej w kodzie jako jednowymiarowa tablica 35 elementowa
2. Wykorzystano narzedzie Matlab do stworzenia wielowarstwowej sieci (newff) oraz algorytmu wstecznej propagacji błędu( traingda) .
3. Uczono sieć dla różnych współczynników uczenia i bezwładności.
4. Testowano poprawność działania sieci

**Specyfikacja wykorzystanych funkcji:**

**sim** - Symulacja sieci neuronowej dla zadanych danych wejściowych. SIM Funkcja służy do wyznaczenia wyjść sieci neuronowej dla zadanej macierzy danych wejściowych. Argumenty *Pi*, *Ai*, *Pf*, *Af* są opcjonalne i nie będą używane przez sieci nieliniowe, wykorzystywane w ćwiczeniach

**newff** - Tworzenie wielowarstwowej jednokierunkowej sieci neuronowej, złożonej z neuronów o nieliniowych funkcjach aktywacji. ( wykorzystamy do stworzenia sieci jednowarstwowej)

NEWFF Funkcja tworzy wielowarstwową sieć neuronową; każda warstwa składa się z

zadanej liczby neuronów o nieliniowych funkcjach aktywacji (jakkolwiek funkcje aktywacji w poszczególnych warstwach mogą mieć również postać liniową).

**WEJŚCIE:**

**PR**- macierz o wymiarach *R*x*2*, gdzie *R* jest liczbą wejść sieci (współrzędnych wektorów wejściowych); pierwsza kolumna zawiera minimalne wartości kolejnych współrzędnych wektorów wejściowych, druga kolumna – maksymalne wartości tych współrzędnych

***Si***- liczba neuronów w *i*-tej warstwie sieci; liczba warstw wynosi *N1*

**TFi-** nazwa funkcji aktywacji neuronów w *i*-tej warstwie sieci (zmienna tekstowa);

domyślna = 'tansig' (tangens hiperboliczny); dopuszczalne wartości parametru *TF* to: ‘tansig’ i ‘logsig’ i ‘purelin’

**BTF**- nazwa funkcji, wykorzystywanej do treningu sieci (zmienna tekstowa); domyślnie *BTF* = ‘trainlm’ (metoda Levenberga-Marquardta)

**BLF**- nazwa funkcji, wykorzystywanej do wyznaczania korekcji wag sieci podczas treningu (zmienna tekstowa); domyślnie *BLF* = ‘learngd’; dopuszczalne wartości parametru *BLF* to: ‘learngd’ (gradient prosty) i ‘learngdm’ (gradient prosty z momentum)

**PF**- funkcja wyznaczająca wartość wskaźnika jakości treningu sieci jednokierunkowej (zmienna tekstowa); domyślnie *PF* = ‘mse’ (błąd średniokwadratowy); parametr ten moŜe oznaczać dowolną róŜniczkowalną funkcję błędu, np. ‘msereg’ (suma błędu średniokwadratowego i kwadratów wag sieci – metoda regularyzacji wag) lub ‘sse’ (suma kwadratów błędów)

**WYJŚCIE:**

**NET**- struktura (obiekt) zawierająca opis architektury, metod treningu, wartości liczbowe wag oraz inne parametry wielowarstwowej sieci jednokierunkowej.

**Trainrp** - metoda propagacji wstecznej błędu RPROP (metoda Riedmillera, Brauna)

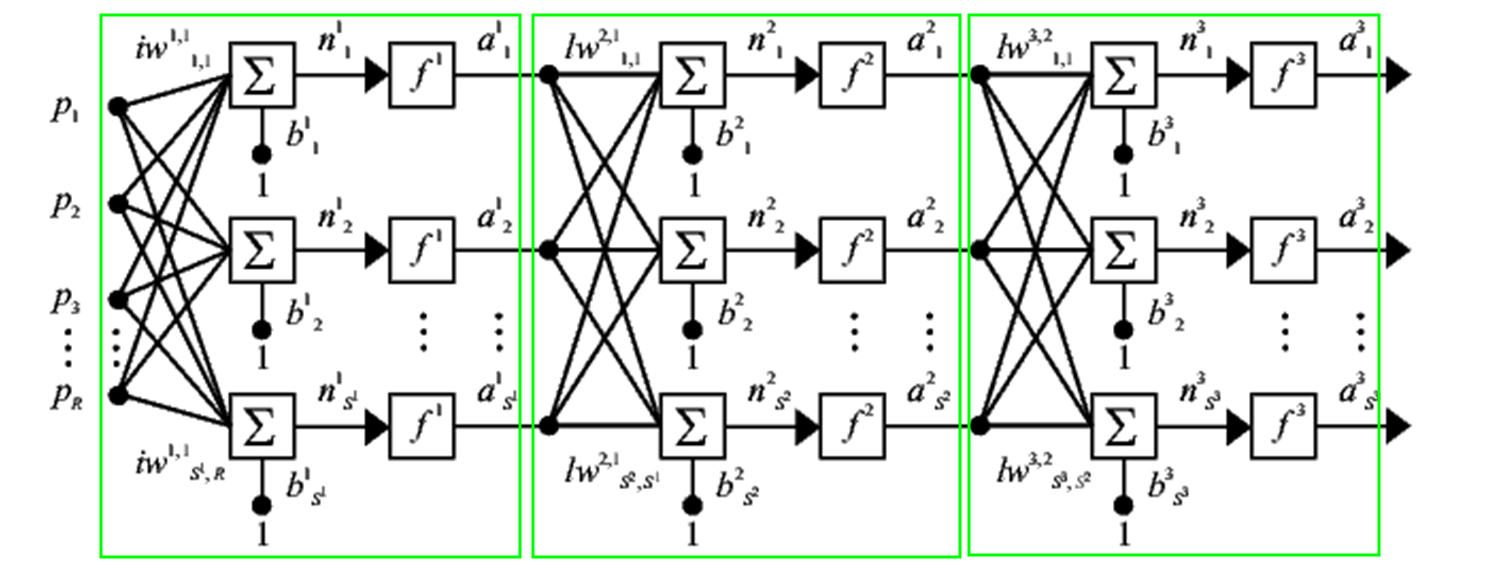
**Traingda**- metoda propagacji wstecznej błędu z adaptacyjną zmianą stałej szybkości

Uczenia

traingda może trenować dowolną sieć, o ile jej waga, funkcje wejściowe i transferowe mają funkcje pochodne. Wsteczna propagacja jest używana do obliczania pochodnych wydajności dperf w odniesieniu do zmiennych masy i odchylenia X. Każda zmienna jest dostosowywana w zależności od nachylenia gradientu:

dX = lr \* dperf / dX

W każdej epoce, jeśli wydajność spada w kierunku celu, wtedy współczynnik uczenia się zwiększa się o współczynnik lr\_inc. Jeśli wydajność wzrasta o więcej niż współczynnik max\_perf\_inc, szybkość uczenia się jest korygowana o współczynnik lr\_dec, a zmiana, która zwiększała wydajność, nie jest wykonywana.



Rysunek 1- Sieć wielowarstwowa

W zielonych ramkach zaznaczone są poszczególne warstwy: wejściowa, ukryta oraz wyjściowa. W tym przypadku jest jedna warstwa ukryta – ale w ogólności może ich być więcej.

**Wykonanie zadania:**

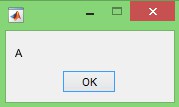
Do wykonania zadania użyłem zastawu danych z 20 literami. W sieci do rozpoznawania liter użyłem następująca ilość neuronów : 35 – 20, 35 – 19 – 20

**Wyniki:**

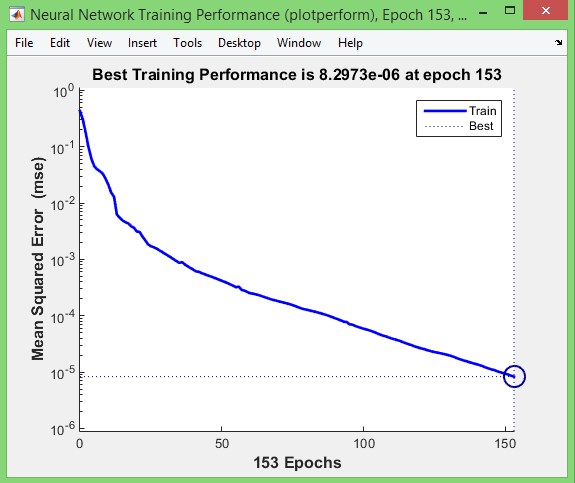
Kilka przykładowych pomiarów przeprowadzonych dla różnych liter oraz rożnej budowy sieci : Dla sieci 35 – 20 i trainrp:



Rysunek 2 - Testowana litera



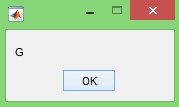
Rysunek 3 - Otrzymany wynik



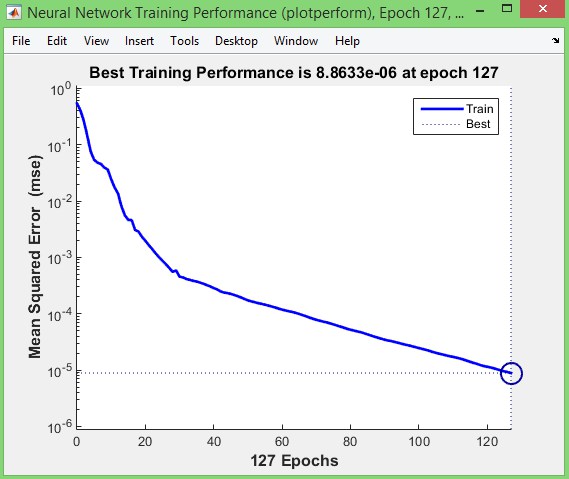
Rysunek 4 - Wykres uczenia się



Rysunek 5 - Testowana litera



Rysunek 6 - Otrzymany wynik



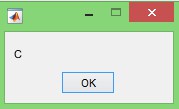
Rysunek 7 - Wykres uczenia się

Tabelka pomiarów znajduje się w wynikach.

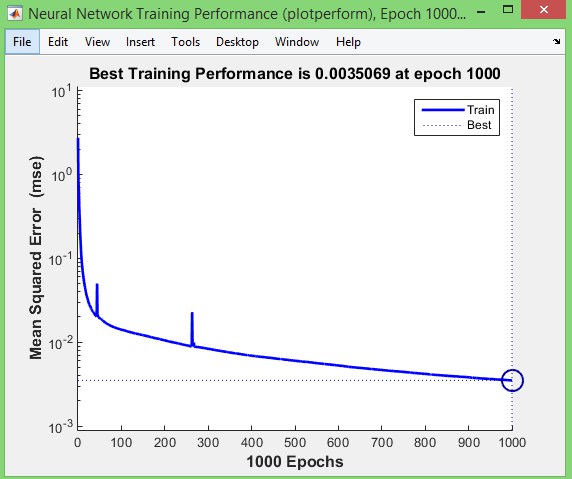
Dla sieci 35 – 19 – 20 i trainrp:



Rysunek 8- Testowana litera



Rysunek 9 - Otrzymany wynik

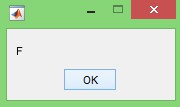


Rysunek 10- Wykres uczenia się

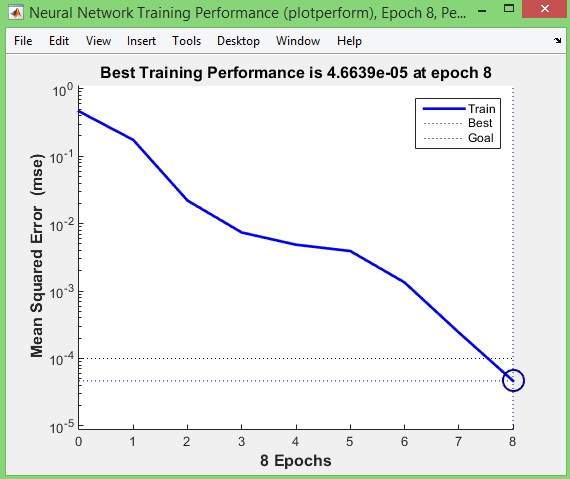
Dla sieci 35– 20 i train:



Rysunek 11- Testowana litera



Rysunek 12- Otrzymany wynik

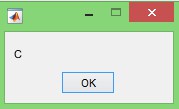


Rysunek 13- Wykres uczenia się

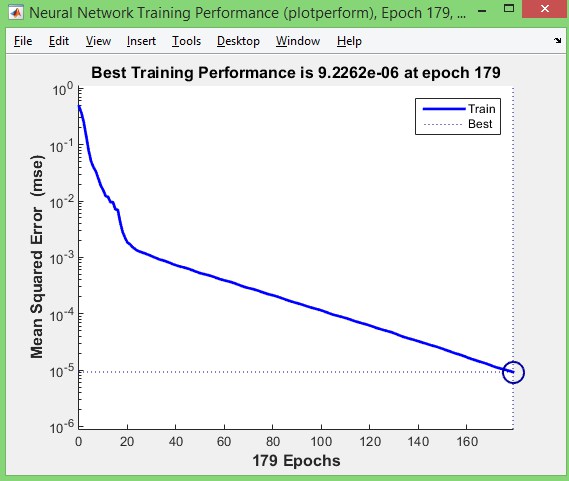
Dla sieci 35 – 19 – 20 , trainrp i zmiana parametru lr:

**C:\Users\km64\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot 2017-11-19 09-39-30.jpg**

Rysunek 8- Testowana litera

****

Rysunek 9 - Otrzymany wynik



Rysunek 14 - Wykres uczenia się

Tabelka pomiarów znajduje się w wynikach.

**Analiza:**

Proces uczenia odbywał się dla trainrp oraz train. Można było zauważyć ze train wykonuje się w małej liczbie epok ale jego wyniki zmierzają do 1 nie było nigdy 1 przez co nie nauczył się w 100% danej literki natomiast podczas stosowania trainrp z propagacja wsteczną prawie wszędzie występowały 1 dla literki A i E były bliskie 1 co daje lepsze wyniki niż w train.

Testy przeprowadziłem dla 2 sieci z których pierwsza nie posiadała warstwy ukrytej 35-20 oraz druga która już ją miała 35-19-20. Obserwując procesy uczenia można było stwierdzić ze dużo lepiej działała sieć pierwsza gdyż nauczyła się w ok 100-200 epokach oraz dawała znakomite wyniki natomiast sieć druga potrzebowała ponad 1000, nie dostawaliśmy dobrego przybliżenia oraz można było zaobserwować przeuczenie się sieci.

Zmieniając wskaźnik uczenia się lr otrzymywaliśmy lepsze przybliżenie wyników ale w efekcie siec musiała się uczyć dłużej. Raz była to różnica kilku epoka a innym kilkunastu.

**Wnioski:**

Na podstawie otrzymanych wyników można było zauważyć ze dobór ukrytych warstw sieci ma duże znaczenie na otrzymywanie wyniki. Jeśli chodzi o funkcję która testuje nam siec jest ona w tym momencie kluczowym elementem gdyż może ona nam zmniejszyć liczbę epok lub zwrócić precyzyjniejszy wynik. Natomiast przez mniejszy lr wzrosła liczba epok lecz wyniki stały się dużo lepsze.

Tworząc tego typu sieci nie można skupić się na jednym czynniku trzeba pamiętać o doborze odpowiednich trenerów, stworzenie odpowiedniej sieci neuronów oraz doborze parametrów lr, gdyż jeśli zbudujemy większą siec i lr również będzie wysoki to możne nam się nie nauczyć albo otrzymamy mało satysfakcjonujący efekt.

**Bibliografia:**

<http://www.ai.c-labtech.net/sn/pod_prakt.html>

<https://www.mathworks.com/help/nnet/ref/traingda.html>

Wykłady

**Listing:**

close all; clear all; clc;

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 1 0 0 0 1 1 0 0 0 1];

B=[1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0];

C=[0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0];

D=[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 1 0];

E=[1 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 1 1 1 1];

F=[1 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];

G=[0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 1 1 1 0];

H=[1 0 0 0 1 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 1 0 0 0 1];

I=[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 0 1 0 0 0 1 1 1 0];

J=[1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0];

K=[1 0 0 0 1 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 1 0 0 0 1];

L=[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 1 0 0 0 0 1 1 1 1 1];

M=[1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1];

N=[1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 0 0 1];

O=[0 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 0 1 1 1 0];

P=[1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0];

Q=[0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1];

R=[1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1];

S=[0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0];

T=[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 0 0 1 0 0];

AA=A'; BB=B'; CC=C'; DD=D'; EE=E'; FF=F'; GG=G'; HH=H'; II=I'; JJ=J';

KK=K'; LL=L'; MM=M'; NN=N'; OO=O'; PP=P'; QQ=Q'; RR=R'; SS=S'; TT=T';

wejscie=[ AA(:) BB(:) CC(:) DD(:) EE(:) FF(:) GG(:) HH(:) II(:) JJ(:) KK(:) LL(:) MM(:) NN(:) OO(:) PP(:) QQ(:) RR(:) SS(:) TT(:)];

wyjscie = eye(20);

testA=[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; 1 ;0; 0; 0; 1; 1 ;0; 0; 0; 1;];

testB=[1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 0; 0;];

testC=[0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0;];

testD=[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; 1; 0;];

testE=[1; 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; 1; 1; 1; 1;];

testF=[1; 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;];

testG=[0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 0; 1; 1; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0;];

testH=[1; 0; 0; 0; 1; 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; 1; 0; 0; 0; 1;];

testI=[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; 0; 1; 0; 0; 0; 1; 1; 1; 0;];

testJ=[1; 1; 1; 1; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0;];

testK=[1; 0; 0; 0; 1; 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; 1; 0; 0; 0; 1;];

testL=[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; 1; 0; 0; 0; 0; 1; 1; 1; 1; 1;];

testM=[1; 0; 0; 0; 1; 1; 1; 0; 1; 1; 1; 0; 1; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1;];

testN=[1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 0; 0; 1; 1; 0; 1; 0; 1; 1; 0; 0; 1; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1;];

testO=[0; 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; 0; 1; 1; 1; 0;];

testP=[1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0;];

testQ=[0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 1; 0; 1; 1; 0; 0; 1; 1; 0; 1; 1; 1; 1;];

testR=[1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 0; 1; 0; 1; 0; 0; 1; 0; 0; 1; 0; 1; 0; 0; 0; 1;];

testS=[0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 0; 1; 1; 1; 0; 0; 0; 0; 0; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0;];

testT=[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; 0; 0; 1; 0; 0;];

net=newff( minmax(wejscie), 20, {'logsig', 'purelin', 'traingda'});

net.trainParam.epochs = 10000;

net.trainParam.goal = 0.001; %Cel wydajności

net.trainParam.lr=0.08; % wskażnik uczenia sie

net=trainrp(net, wejscie, wyjscie); % z metoda propagacji wstecznej

%net=train(net, wejscie, wyjscie); % zwykły trener

wynik=sim(net, wejscie);

litera=input('podaj litere do rozpoznania : ', 's');

switch litera % Sprawdzenie czy jest taka literka ktora podalismy

case 'A'

test=testA;

case 'B'

test=testB;

case 'C'

test=testC;

case 'D'

test=testD;

case 'E'

test=testE;

case 'F'

test=testF;

case 'G'

test=testG;

case 'H'

test=testH;

case 'I'

test=testI;

case 'J'

test=testJ;

case 'K'

test=testK;

case 'L'

test=testL;

case 'M'

test=testM;

case 'N'

test=testN;

case 'O'

test=testO;

case 'P'

test=testP;

case 'Q'

test=testQ;

case 'R'

test=testR;

case 'S'

test=testS;

case 'T'

test=testT;

otherwise

disp('nie ma takiej litery')

end

test1=sim(net,test)

for i=1:20

if(max(test1)== test1(i)) % Sprawdzenie czy nauczyła sie podanej przez nas literki

if(i==1)

msgbox('A');

end

if(i==2)

msgbox('B');

end

if(i==3)

msgbox('C');

end

if(i==4)

msgbox('D');

end

if(i==5)

msgbox('E');

end

if(i==6)

msgbox('F');

end

if(i==7)

msgbox('G');

end

if(i==8)

msgbox('H');

end

if(i==9)

msgbox('I');

end

if(i==10)

msgbox('J');

end

if(i==11)

msgbox('K');

end

if(i==12)

msgbox('L');

end

if(i==13)

msgbox('M');

end

if(i==14)

msgbox('N');

end

if(i==15)

msgbox('O');

end

if(i==16)

msgbox('P');

end

if(i==17)

msgbox('Q');

end

if(i==18)

msgbox('R');

end

if(i==19)

msgbox('S');

end

if(i==20)

msgbox('T');

end

end

end