|  |  |  |
| --- | --- | --- |
| 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.

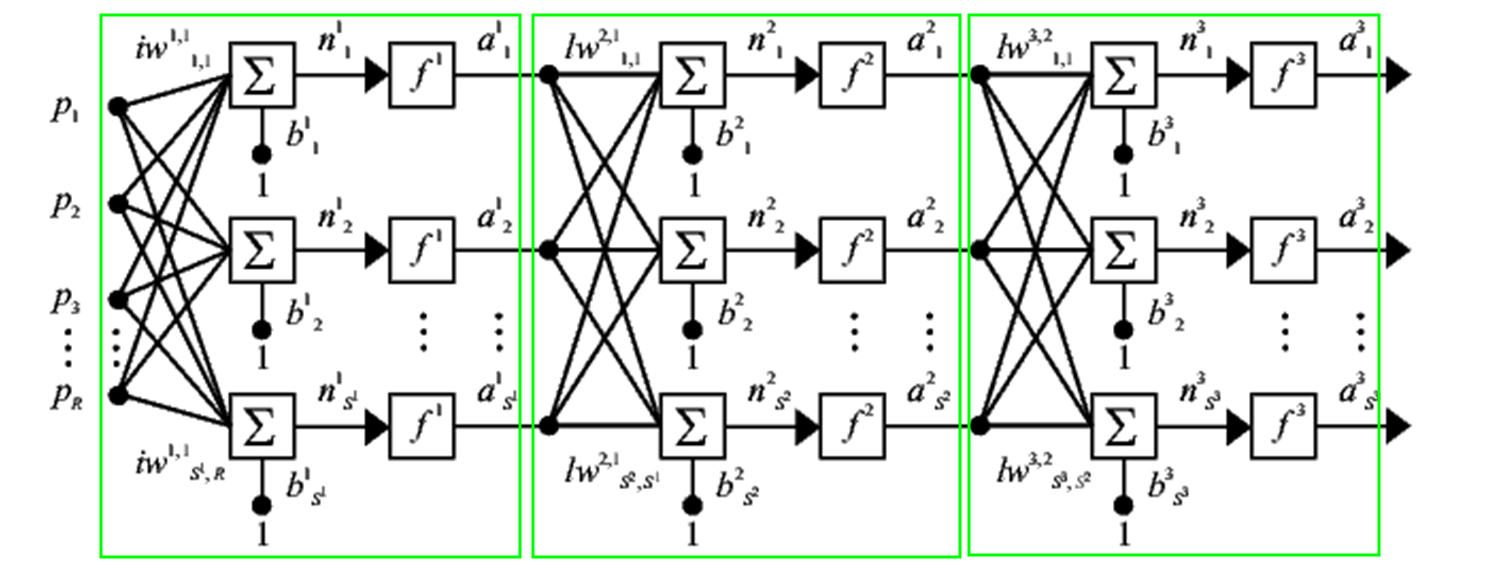
**Traingda**- metoda propagacji wstecznej błędu z adaptacyjną zmianą stałej szybkości

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:**

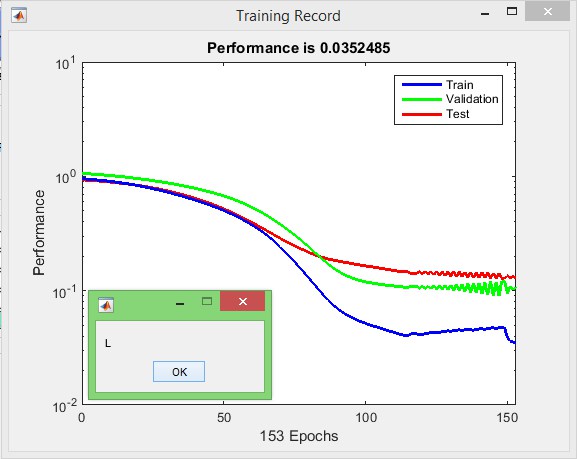
Do wykonania zadania użyłem zastawu danych z 20 literami. W sieci do rozpoznawania liter użyłem następująca ilość neuronów : 35 – 20, 35 – 19 – 20

**Wyniki:**

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



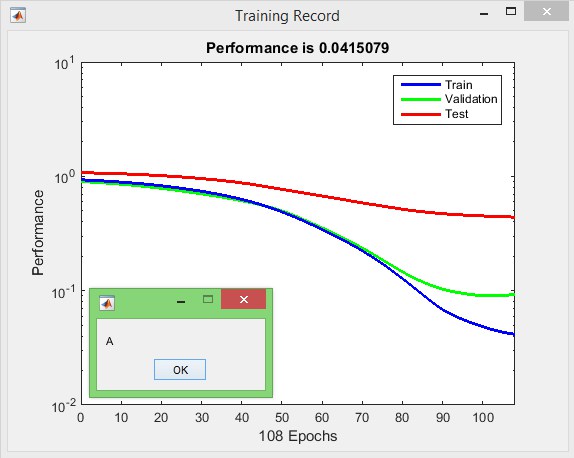
Rysunek 2 - Testowana litera



Rysunek 3 - Otrzymane wyniki



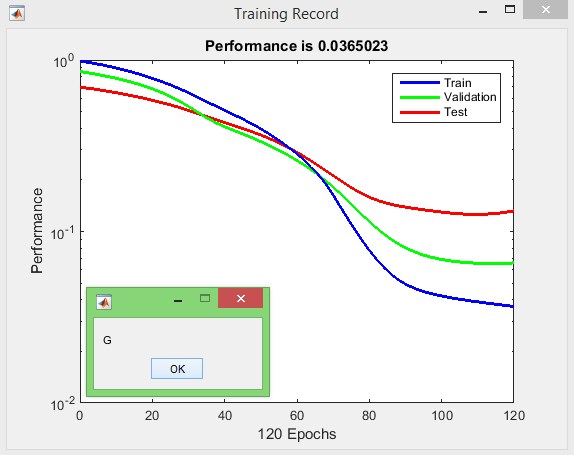
Rysunek 4 - Testowana litera



Rysunek 5 - Otrzymane wyniki



Rysunek 6 - Testowana litera



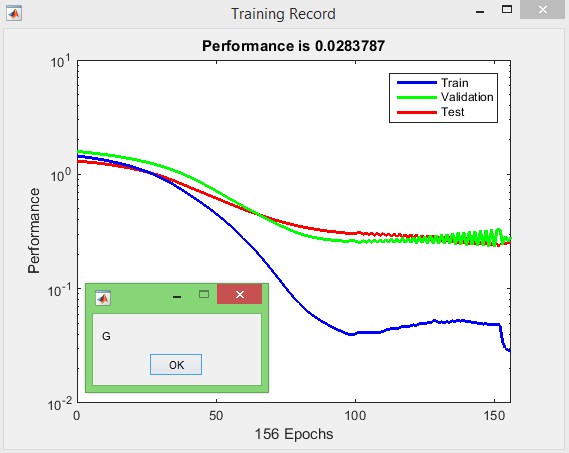
Rysunek 7 - Otrzymane wyniki

Tabelka z reszta pomiarów znajduje się w wynikach.

Dla sieci 35 – 19 – 20:

C:\Users\km64\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot 2017-11-17 13-37-11.jpg

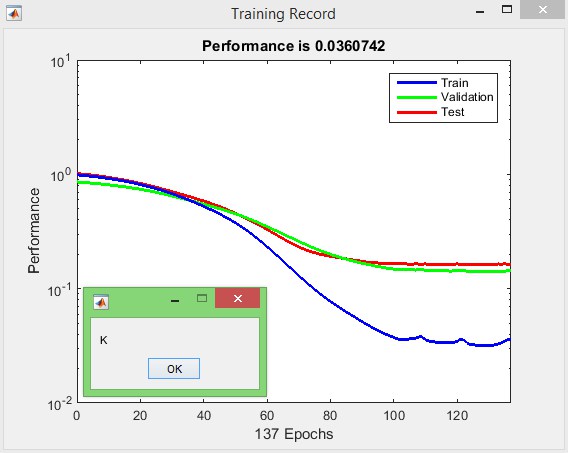
Rysunek 8 - Testowana litera



Rysunek 9 - Otrzymane wyniki



Rysunek 10 - Testowana litera



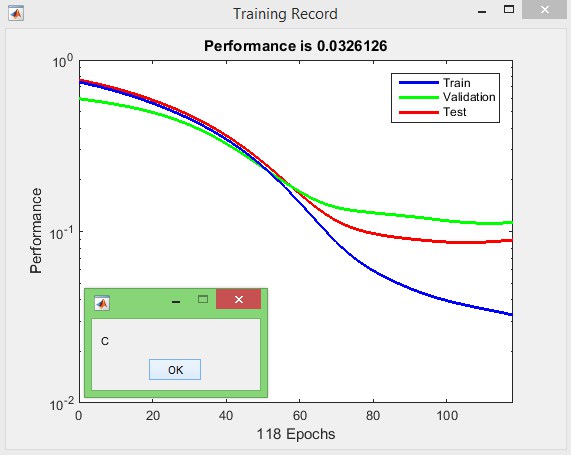
Rysunek 11 - Otrzymane wyniki

Tabelka z pomiarów znajduje się w wynikach.

Kilka błędów podczas uczenia:



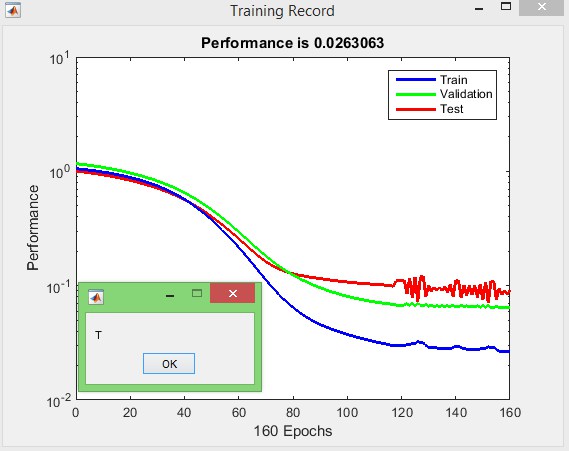
Rysunek 12 - Testowana litera



Rysunek 13 - Otrzymane wyniki



Rysunek 14 - Testowana litera



Rysunek 15 - Otrzymane wyniki

**Analiza:**

Proces uczenia odbywał się na początku dla kilkunastu tranów ale po dłuższych testach okazało się to zgubne bo sieć się przyuczała i otrzymywałem większości jakieś śmieci. Po zmniejszeniu do kilku udało się otrzymać już zadowalający efekt. Sieć zwracała mi literki jakie chciałem. Podczas przeprowadzania testów używałem traingda która jest metoda propagacji wstecznej błędu z adaptacyjną zmianą stałej szybkości uczenia. Sieć miała czasami problemy z nauczeniem i nie zawsze otrzymywałem zadowalający efekt natomiast podczas zamiany na zwykłego tarin efekt był gorszy spadła ilość epeok i otrzymywaliśmy różne wyniki. Przy zmianie parametru lr z 0,4 na 0,08 wyniki stały się bardziej precyzyjnie ale było widać wzrost liczby epok do wcześniejszych.

Podczas testów dla sieci 35 – 19 – 20 można było zauważyć mniej precyzyjne wyniki niż podczas testów sieci 35– 20 tu różnice pomiędzy literkami były większe.

Testując taką literkę jak C zauważyłem ze sieć bardzo często myli ja z literką G gdyż są bardzo podobne i otrzymujemy wyniki dość zbliżone do siebie. Podobny problem był z literka I oraz T raz na ta literkę mówił T a następnym razem I. Jeżeli literki były bardzo podobne można było zauważyć takie błędy.

**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 ma ona ogromne znaczenie gdyż może ona nam zmniejszyć liczbę epok oraz zwrócić precyzyjniejszy wynik. Natomiast przez mniejszy lr może i wzrosła liczba epok ale 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:**

clear all;close 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];

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];

Z = [1 1 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 1 1 1];

inputs=[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;0 1;0 1;0 1;0 1;0 1;0 1;0 1];

targets=[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];

net=newff(inputs,targets, [20]);

litery=[A;B;C;D;E;F;G;H;I;J;K;L;M;N;O;P;R;S;T;Z];

litery=litery';

spr =i'; % litera ktora bedziemy sprawdzac czy nauczył sie okreslac wielkosc liter

wy=eye(20);

wy=wy';

Y=sim(net,litery);

net.trainParam.max\_fail=8;%Maksymalne niepowodzenia sprawdzania poprawności

% net.adaptParam.passes = 3; %3 pętle doboru wag i przesunięcia

% [net,z,x,Pf,Af,TR] = adapt(net, litery, wy);

%net.trainParam.goal=20;%kryterium stopu (sumę kwadratów błędów wyjść sieci)

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

[net tr] =traingda(net,litery,wy) % Trening sieci neuronowej.

plotperf(tr)

[net tr] =traingda(net,litery,wy)

[net tr] =traingda(net,litery,wy)

Y=sim(net,litery);

te=sim(net,D' )

wynik = te'

max(te);

for i=1:20

if(max(te)== te(i))

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('R');

end

if(i==18)

msgbox('S');

end

if(i==19)

msgbox('T');

end

if(i==20)

msgbox('Z');

end

end

end