Akademia Górniczo-Hutnicza

WIMiIP, Inżynieria Obliczeniowa

G01, Małgorzata Kusik

Nr. indeksu 293103

Podstawy Sztucznej Inteligencji

Sprawozdanie z Projektu 3

Budowa i działanie sieci wielowarstwowej typu feedforward

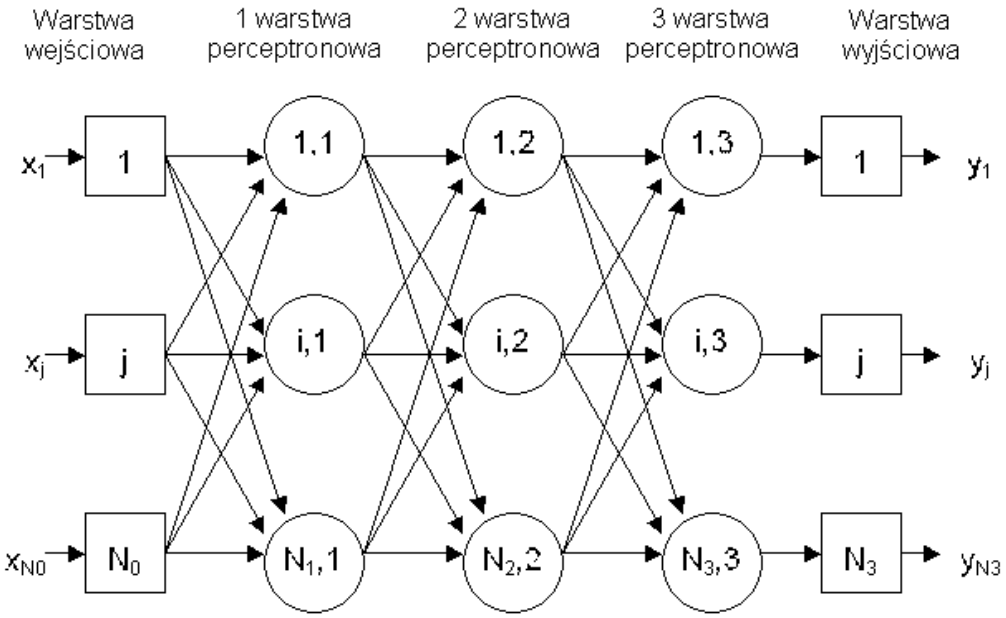
1. Cel projektu:

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

Zadania do wykonania:
– wygenerowanie danych uczących i testujących zawierających 20 wielkich dowolnie wybranych liter alfabetu
– przygotowanie wielowarstwowej sieci oraz algorytmu wstecznej propagacji błędu
– uczenie sieci dla różnych współczynników uczenia
– testowanie sieci

2. Część teoretyczna

Sieć neuronowa (sztuczna sieć neuronowa) – ogólna 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 wielowarstwowa - składa się zwykle z jednej warstwy wejściowej, kilku warstw ukrytych oraz jednej warstwy wyjściowej. Warstwy ukryte składają się najczęściej z neuronów McCullocha-Pittsa.



Uczenie sieci neuronowych - wymuszenie na niej określonej reakcji na zadane sygnały wejściowe. Uczenie jest konieczne tam, gdzie brak jest informacji doświadczalnych o powiązaniu wejścia z wyjściem lub jest ona niekompletna, co uniemożliwia szczegółowe zaprojektowanie sieci. Uczenie może być realizowane krok po kroku lub poprzez pojedynczy zapis. Istotnym czynnikiem przy uczeniu jest wybór odpowiedniej strategii (metody) uczenia. Wyróżnić możemy dwa podstawowe podejścia: uczenie z nauczycielem (supervised learning) i uczenie bez nauczyciela (unsupervised learning).

Algorytm uczenia sieci wielowarstwowych metodą propagacji błędu:

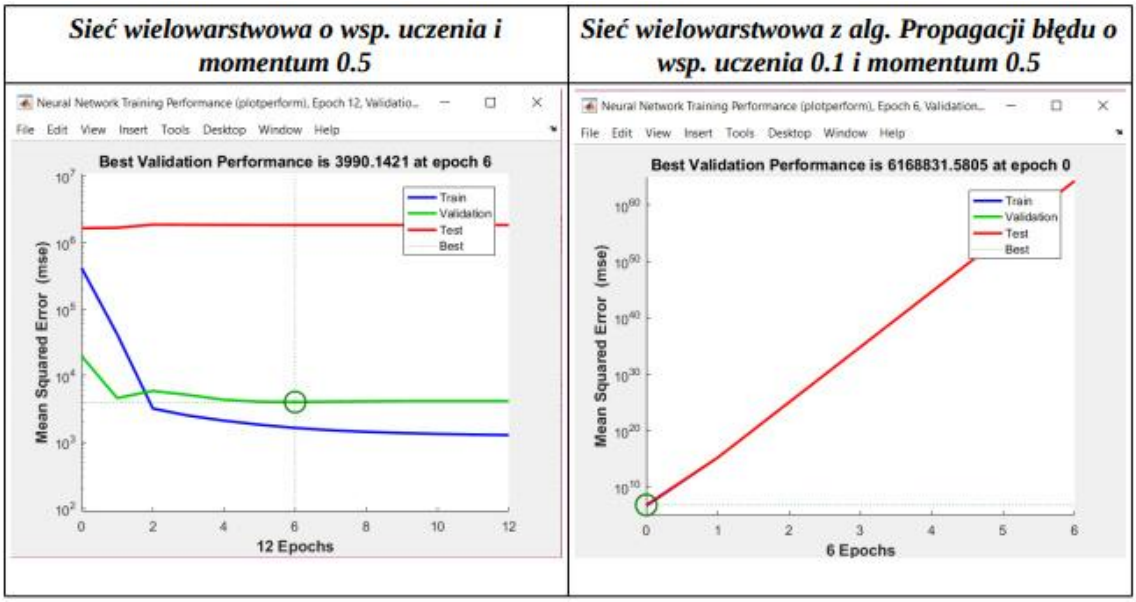
1. Podajemy dane wejściowe.
2. Inicjujemy losowo, odpowiednio małe wagi.
3. Dla danego wektora uczącego obliczamy odpowiedź sieci (warstwa po warstwie).
4. Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią y oraz poprawną odpowiedzią t. Błędy propagowane są do wcześniejszych warstw.
5. Każdy neuron modyfikuje wagi na podstawie wartości błędu i wielkości przetwarzanych w tym kroku sygnałów.
6. Dla kolejnych wektorów uczących powtarzamy operacje od kroku 3. Gdy wszystkie wektory zostaną użyte, losowo zmieniamy ich kolejność i zaczynamy wykorzystywać powtórnie.
7. Jeśli średni błąd na danych treningowych przestanie maleć, przerywamy działanie algorytmu.

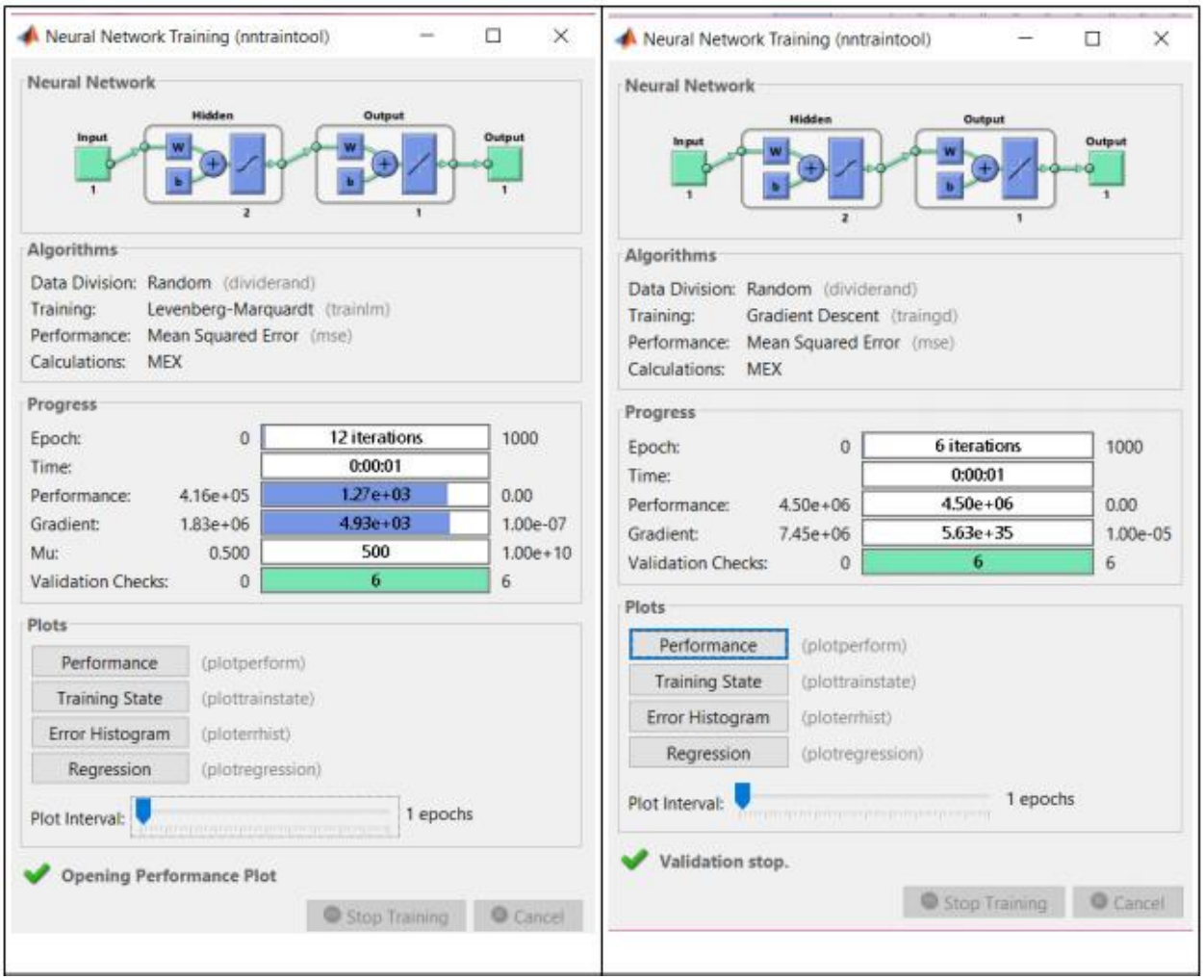
3. Wyniki:

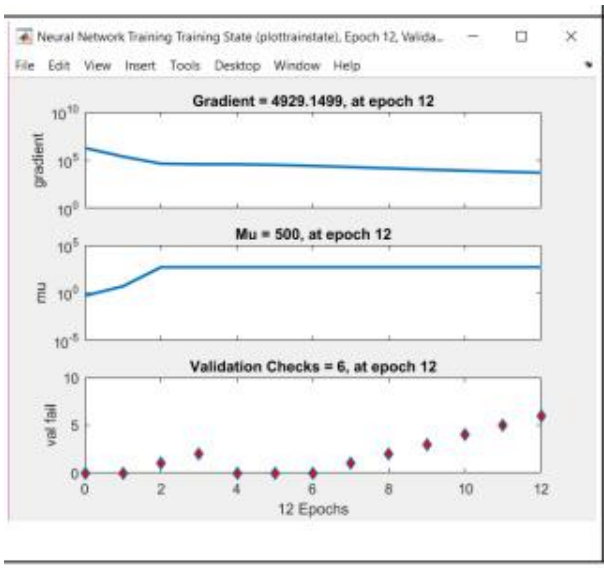
Do wykonania tego zadania wykorzystałam pakiet MATLAB, a szczególnie narzędzie Neural Networking Training Tool. Pozwala ono nie tylko na utworzenie prostych sieci jednowarstwowych, ale również tych bardziej złożonych, korzystających z modelu sieci wielowarstwowych. Można również, za pomocą parametrów wykorzystać algorytm

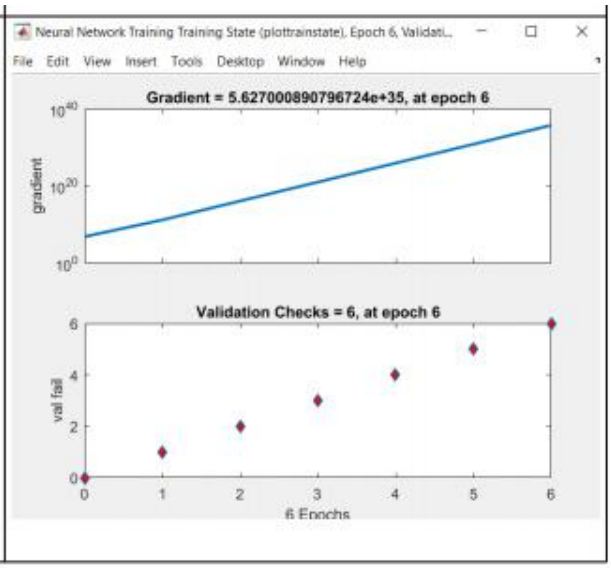
propagacji błędów. Zadanie polegało na wygenerowaniu danych uczących i testujących dla funkcji Rastrign 3D dla danych wejściowych z przedziału [-2; 2].

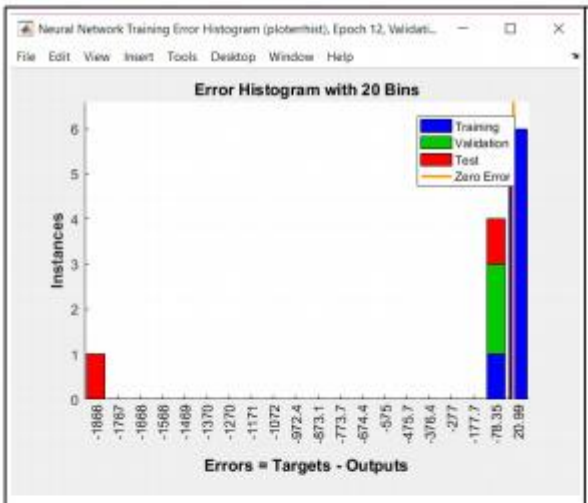
Poniżej zamieszczone są zrzuty wykresów dla wariantów najgorzej i najlepiej działających.

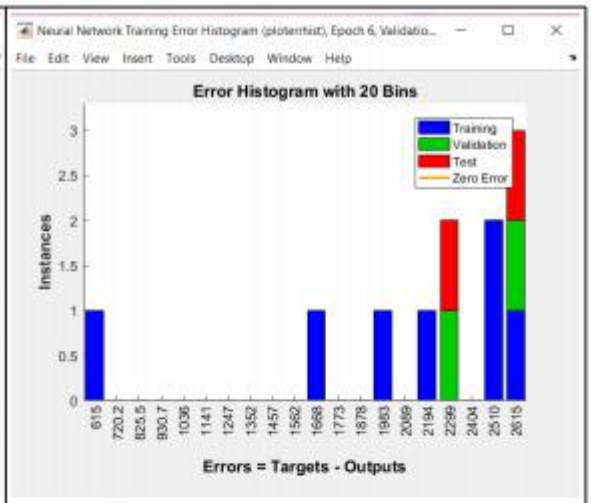


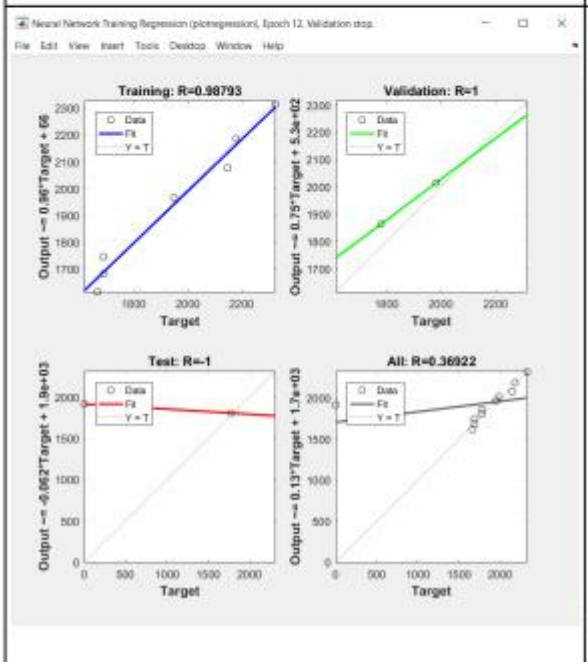


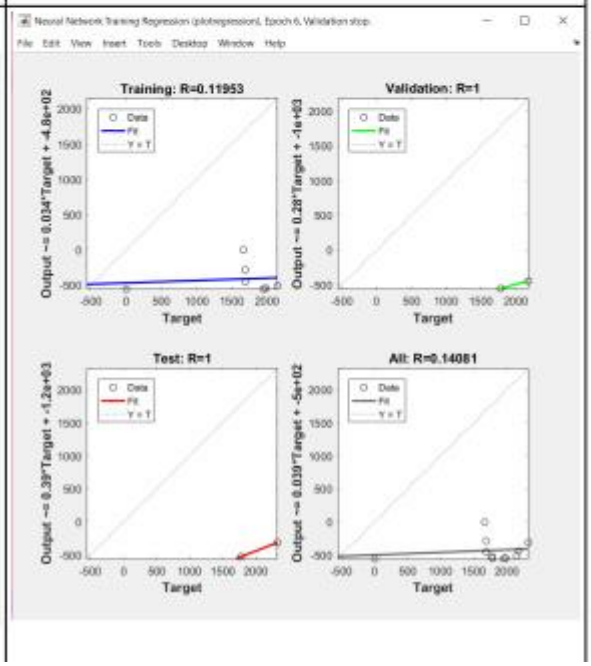

Neural Network Training Training State (plottrainstate), Epoch 12, Valida...
File Edit View Insert Tools Desktop Window Help
Gradient = 4929.1499, at epoch 12
gradient
Mu = 500, at epoch 12
mu
Validation Checks = 6, at epoch 12
val fail
12 Epochs


Neural Network Training Training State (plottrainstate), Epoch 6, Validati...
File Edit View Insert Tools Desktop Window Help
Gradient = 5.627000890796724e+35, at epoch 6
gradient
Validation Checks = 6, at epoch 6
val fail
6 Epochs


Neural Network Training Error Histogram (ploterrhist), Epoch 12, Validati...
File Edit View Insert Tools Desktop Window Help
Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
Errors = Targets - Outputs


Neural Network Training Error Histogram (ploterrhist), Epoch 6, Validatio...
File Edit View Insert Tools Desktop Window Help
Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
Errors = Targets - Outputs


Training: R=0.98793
Validation: R=1
Test: R=-1
All: R=0.36922
Target


Training: R=0.11953
Validation: R=1
Test: R=1
All: R=0.14081
Target

4. Wnioski:

• Sieciami wielowarstwowymi możemy sterować dopasowując poszczególne parametry uczenia, takie jak współczynnik uczenia, czy bezwładność. Współczynniki te wpływają bardzo mocno na otrzymywane wyniki.
• Podczas nie korzystania z algorytmów propagacji danych należy stosować duże różnice między współczynnikami: uczenia i bezwładności.
• Najmniej dokładne wyniki osiągają sieci których współczynnik uczenia jest równy bezwładności.
• Sieci wielowarstwowe posiadające algorytm propagacji danych wykazują duże odsetki błędów. Im większy jest współczynnik uczenia od bezwładności, tym częściej mogą zdarzać się błędy.
• Algorytmy z propagacją danych przyspieszają proces uczenia nawet o połowę epok w stosunku do sieci niezawierających tego algorytmu.
• Należy pamiętać, że sieci tylko modelują pewne dane na bazie pewnych danych uczących. Nie tworzą one idealnych obrazów między danymi wejściowymi, a danymi wyjściowymi.

5. Kod programu:

Funkcja pomocnicza

```
function fx = RastrignTest3D(x)
    if x == 0
        fx = 0;
    else
        A = 10;
        n = 100;
        x1 = x;
        dx = (5.12-x)/n;

        fx = A * n;

        for i = 1:n
            x = x1 + (i * dx);
            fx = fx + (x^2) - (A * cos(2 * pi * x));
        end
        %disp(fx)
    end
end
```

Funkcja ucząca i testująca dane

```
close all; clear all; clc;

wej_in = [-2 -1.6 -1.2 -0.8 -0.4 0 0.4 0.8 1.2 1.6 2];
wej_out = [1.6633e+03 1.6880e+03 1.6867e+03 1.7764e+03 1.7800e+03 0.0
1.9495e+03 1.9825e+03 2.1477e+03 2.1791e+03 2.3262e+03];

testowe = zeros(1);

net = feedforwardnet(2); %tworzenie sieci z 2 warstwami ukrytymi
net.trainFcn = 'traingd'; %algorytm wstecznej propagacji
net.trainParam.lr = 0.1; %wsp. uczenia
net.trainParam.mc = 0.5; %bezwladnosc
net = train(net, wej_in, wej_out);
efekty = zeros(size(net));

for i = 1:11
    testowe(i) = RastrignTest3D(wej_in(i)); %wygenerowanie prawidłowych
wyników działania funkcji RastrignTest3D
    efekty(i) = sim(net, wej_in(i)); %test działania sieci
end
```