Podstawy sztucznej inteligencji

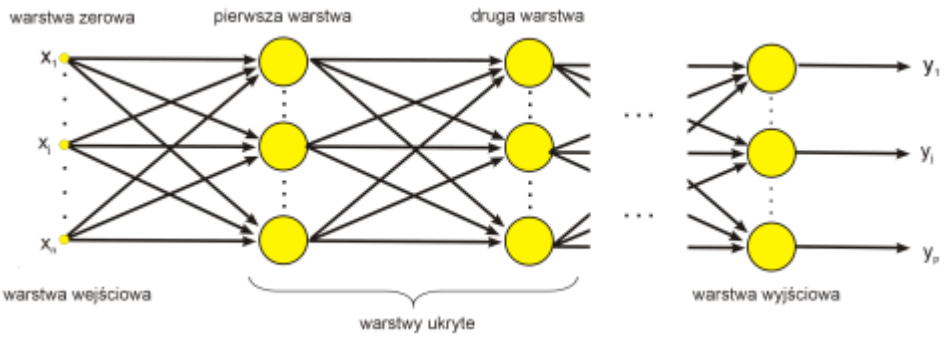
Kinga Synowiec, Inżynieria obliczeniowa, grupa 2

Sprawozdanie nr 3

- **Temat ćwiczenia:**
  Budowa i działanie sieci wielowarstwowej typu feedforward.

- **Cel ćwiczenia:**
  Poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie kształtu funkcji matematycznej z użyciem algorytmu wstecznej propagacji błędu.

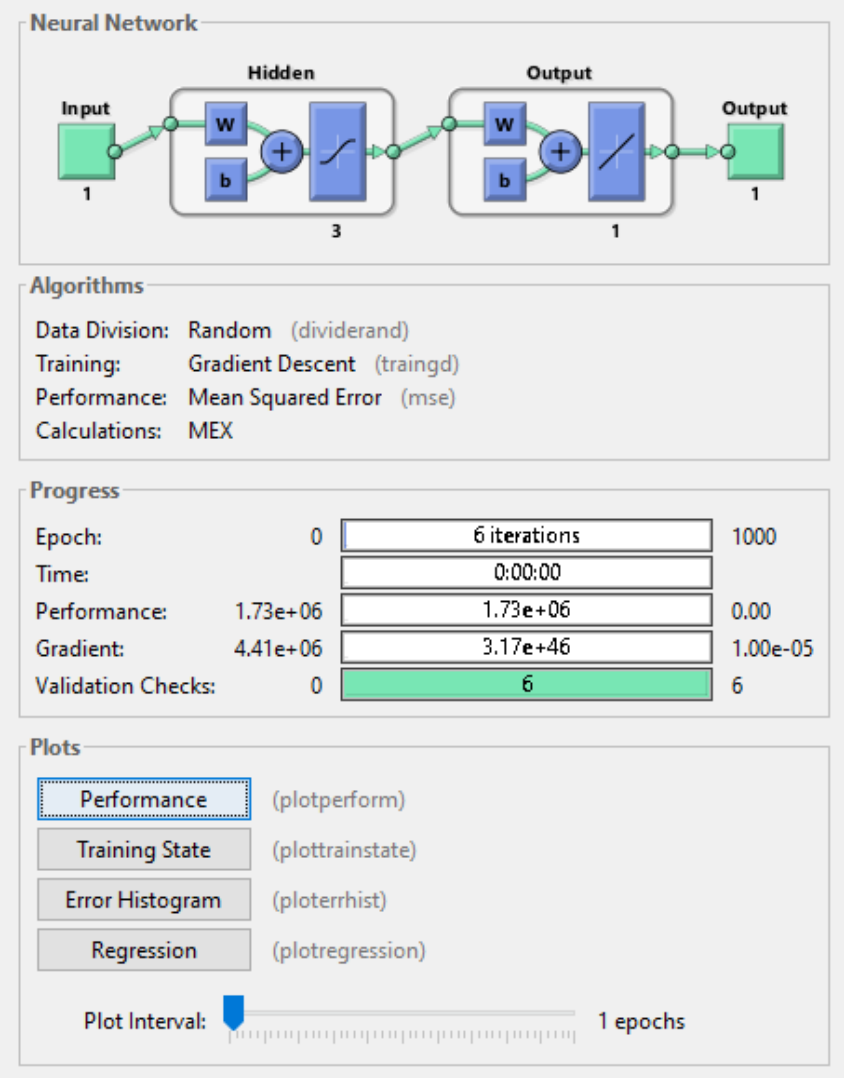
- **Wstęp teoretyczny:**
  Sieci jednokierunkowe wielowarstwowe (feedforward) – sieci neuronowe posiadające prostą, łatwą do opisania strukturę oraz łatwe do realizacji metody uczenia. Struktura ta jest zbliżona do budowy mózgu. W sieciach jednokierunkowych można wyróżnić uporządkowane warstwy neuronów (w tym warstwę wejściową i warstwę wyjściową). Liczba neuronów w każdej z warstw może być różna, przy czym w danej warstwie wszystkie neurony maja taką sama funkcję przejścia – neurony z różnych warstw mogą mieć różne funkcje przejścia. Połączenia występują tylko pomiędzy neuronami z sąsiednich warstw, wg zasady „każdy z każdym" i mają one charakter asymetryczny. Sygnały przesyłane są od warstwy wejściowej poprzez warstwy ukryte (jeśli występują) do warstwy wyjściowej (w jednym kierunku). Neurony warstwy wejściowej posiadają tylko jedno wejście i uproszczoną funkcję przejścia (umownie jest to warstwa zerowa sieci). Zadaniem neuronów z warstwy wejściowej jest wstępna obróbka sygnału (np.: normalizacja, kodowanie itp.). Z kolei za przetwarzanie decyzyjne odpowiedzialne są neurony warstw ukrytych i warstwy wyjściowej, a odpowiedź udzielana jest przez neurony warstwy wyjściowej.

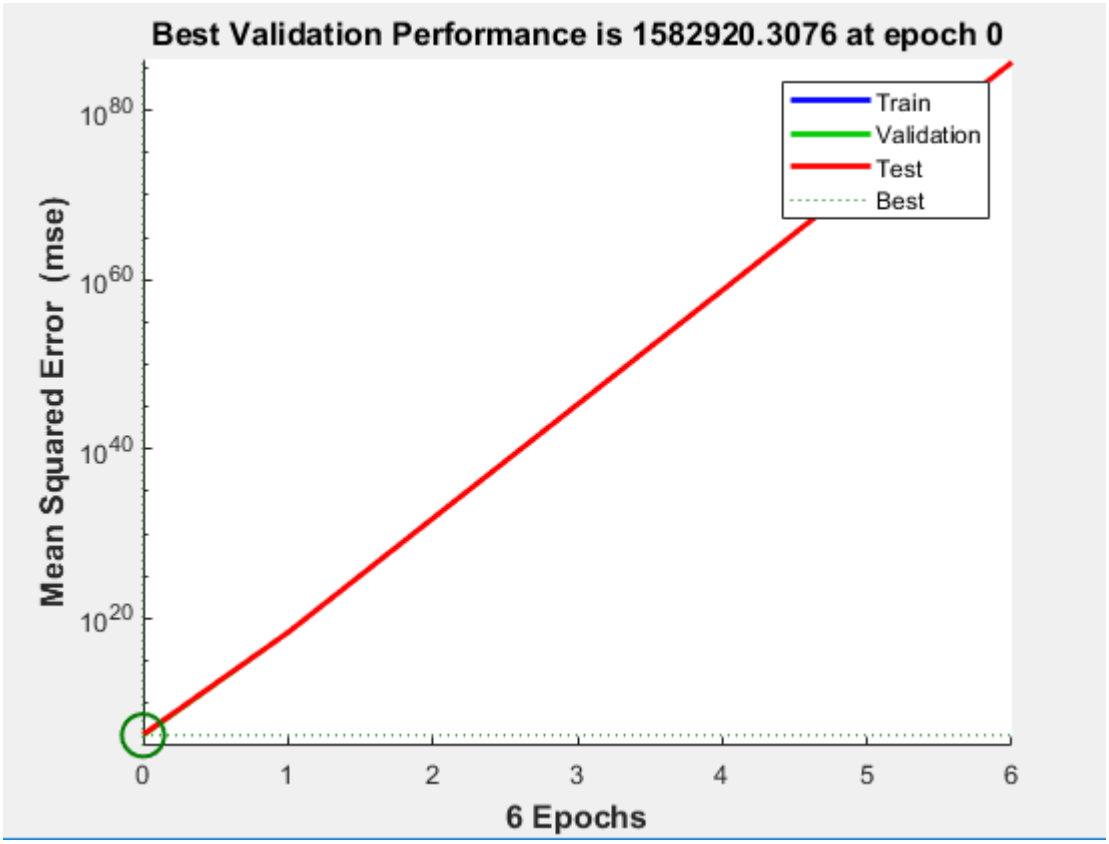


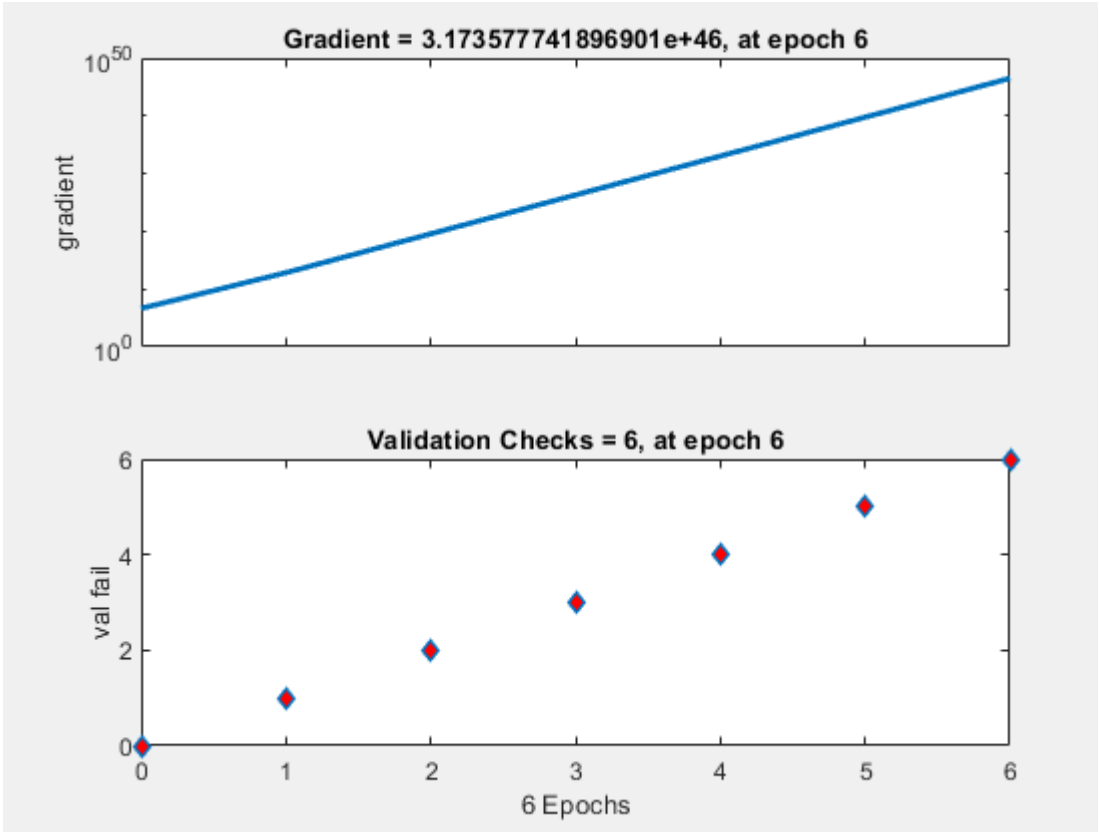
// źródło :
http://www.neurosoft.edu.pl/media/pdf/jbartman/sztuczna_inteligencja/NTI%20cwiczenie_5.pdf

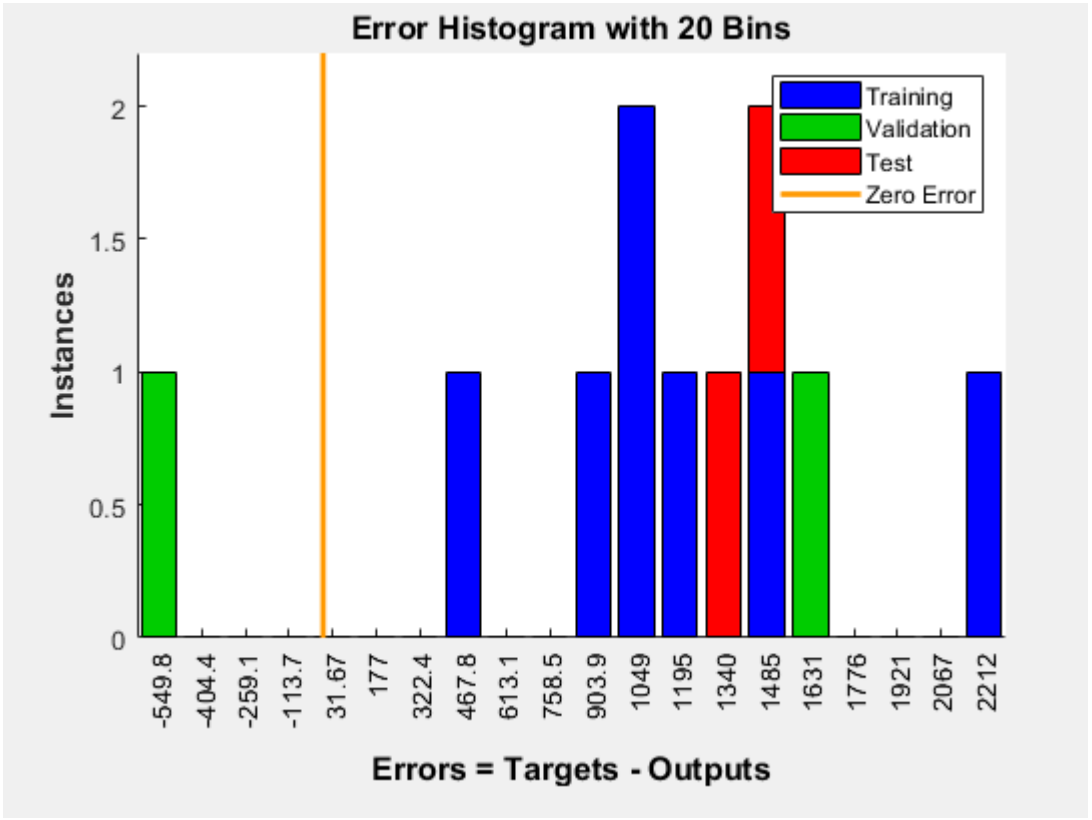
- **Opis użytych funkcji i metod zaczerpniętych z pakietu MATLAB:**
  - wejście – tablica danych wejściowych z przedziału od -2 do 2, z krokiem zmieniania co 0.4
  - wyjście – tablica o takiej samej liczbie elementów jak wejście, której elementami są wyniki uzyskane dzięki napisanej funkcji Rastigin3D dla liczb z tablicy wejścia
  - testowe – tablica pomocnicza uzupełniona zerami
  - feedforwardnet() – funkcja służąca do tworzenia sieci z ukrytymi warstwami, ilość wszystkich warstw wskazana jest jako wartość argumentu
  - trainFcn – algorytm służący do przeprowadzania wstecznej propagacji
  - trainParam – pod tą zmienną przechowywane są wszelkie parametry potrzebne do wykonania zadania; w naszym przypadku są to współczynnik uczenia oraz bezwładność
  - train() – przeprowadzenie procesu uczenia sieci z wykorzystaniem danych wejściowych i wyjściowych
  - RastrignTest3D() – funkcja napisana w osobnym pliku, dla której należało wygenerować dane uczące i testujące

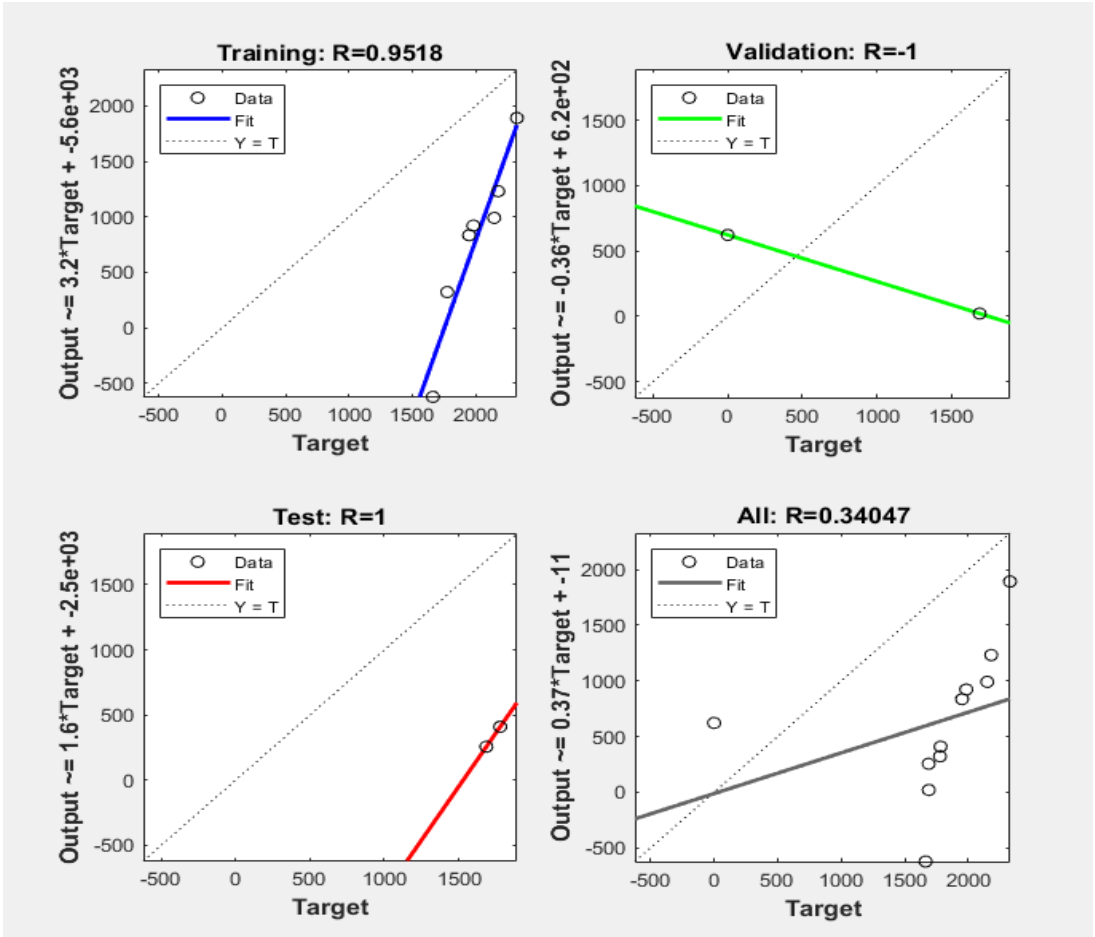
- **Zrzuty ekranu przedstawiające przykładowe rezultaty po uruchomieniu programu:**
  - wersja dla parametrów : współczynnik uczenia = 0.5, bezwładność = 0.5



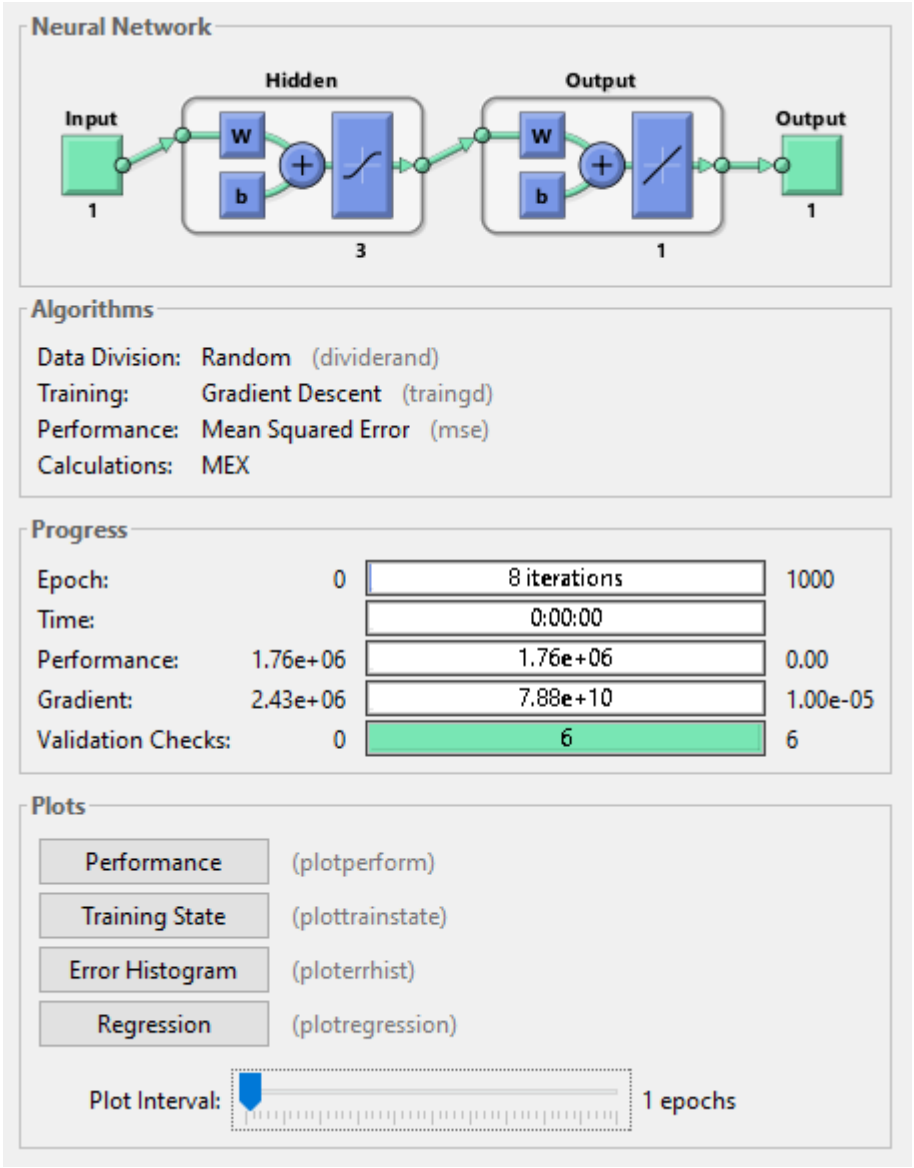

Best Validation Performance is 1582920.3076 at epoch 0
Mean Squared Error (mse)
10^{80}
10^{60}
10^{40}
10^{20}
Train
Validation
Test
Best
0
1
2
3
4
5
6
6 Epochs

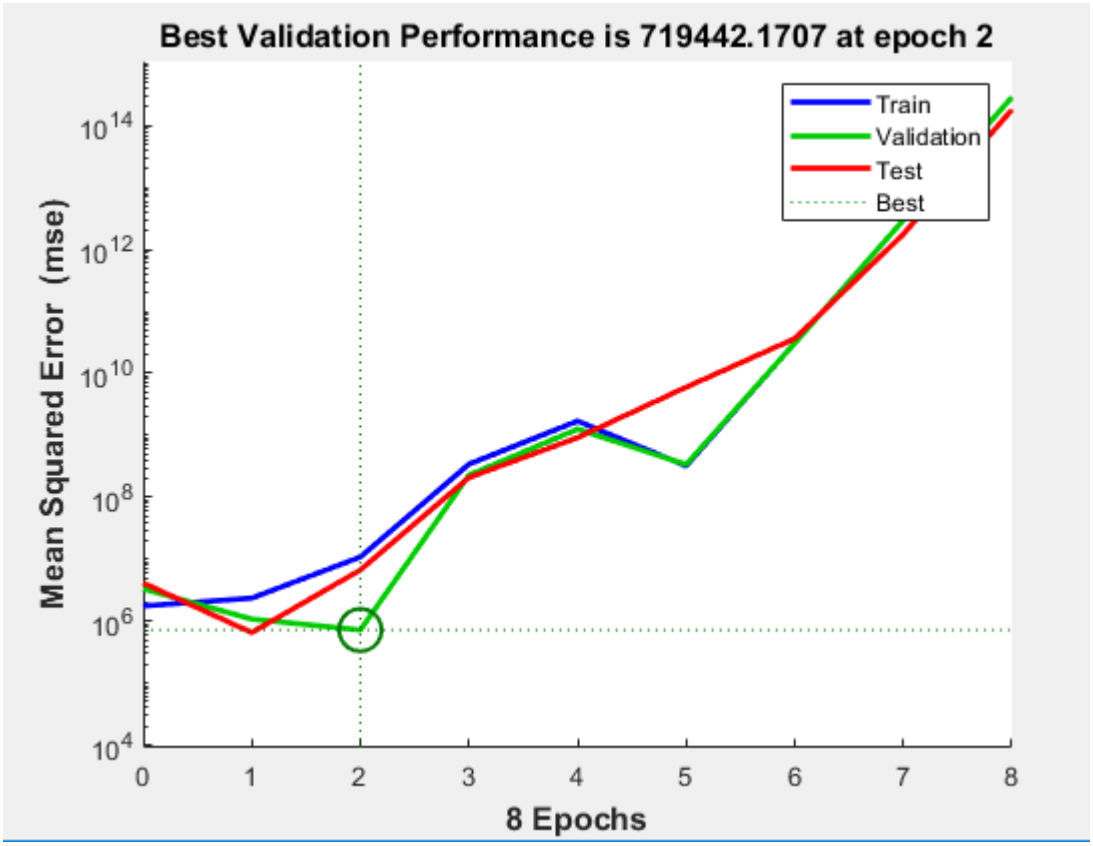

Gradient = 3.173577741896901e+46, at epoch 6
gradient
10^{50}
10^{0}
Validation Checks = 6, at epoch 6
val fail
6
4
2
0
0
1
2
3
4
5
6
6 Epochs

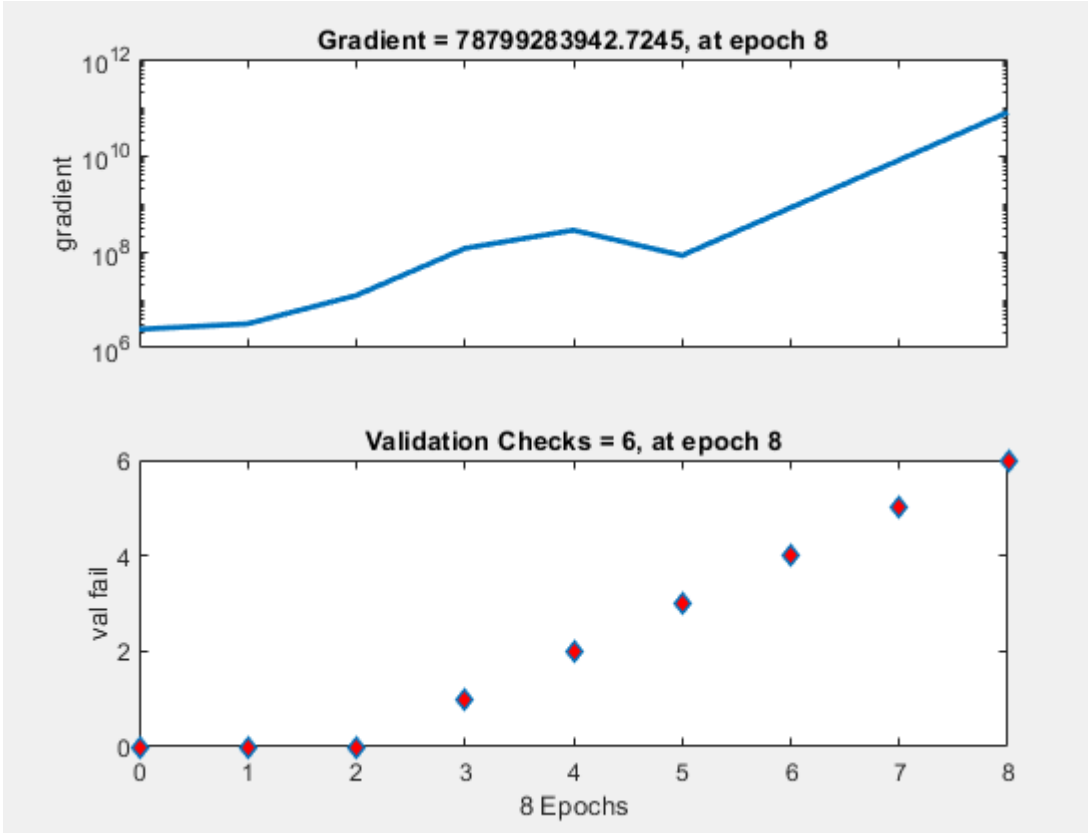

Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
0
0.5
1
1.5
2
-549.8
-404.4
-259.1
-113.7
31.67
177
322.4
467.8
613.1
758.5
903.9
1049
1195
1340
1485
1631
1776
1921
2067
2212
Errors = Targets - Outputs

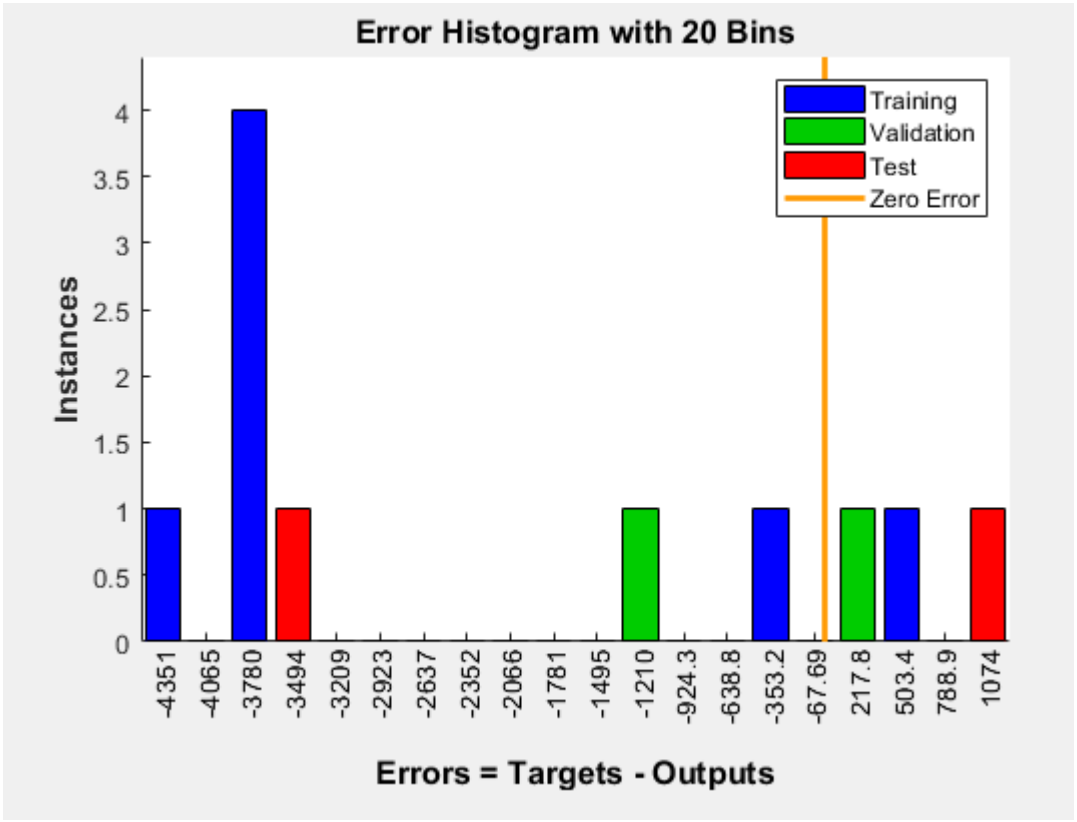

Training: R=0.9518
Data
Fit
Y = T
Output ~= 3.2*Target + -5.6e+03
Target
Validation: R=-1
Data
Fit
Y = T
Output ~= -0.36*Target + 6.2e+02
Target
Test: R=1
Data
Fit
Y = T
Output ~= 1.6*Target + -2.5e+03
Target
All: R=0.34047
Data
Fit
Y = T
Output ~= 0.37*Target + -11
Target

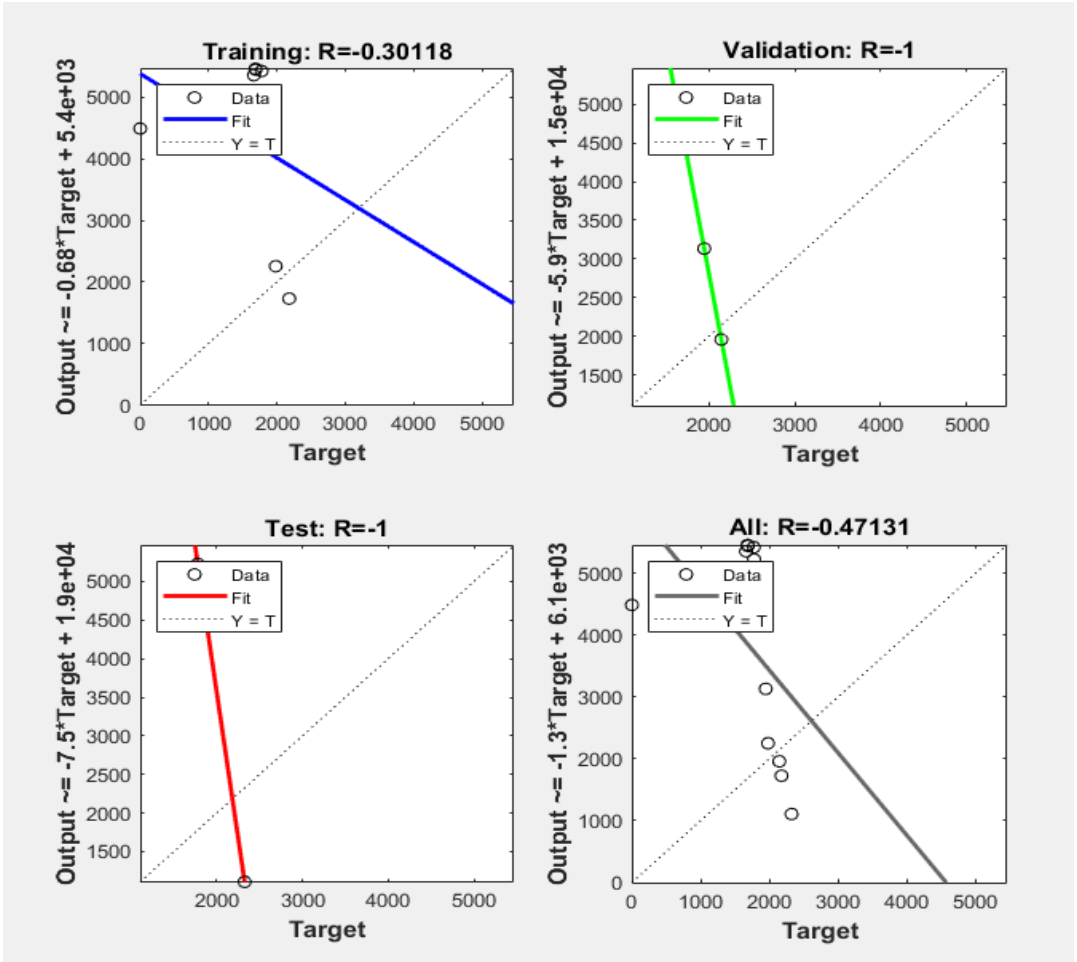
- wersja dla parametrów : współczynnik uczenia = 0.000001, bezwładność = 0.1




Best Validation Performance is 719442.1707 at epoch 2
Train
Validation
Test
Best
Mean Squared Error (mse)
8 Epochs


Gradient = 78799283942.7245, at epoch 8
gradient
Validation Checks = 6, at epoch 8
val fail
8 Epochs


Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
0
0.5
1
1.5
2
2.5
3
3.5
4
-4351
-4065
-3780
-3494
-3209
-2923
-2637
-2352
-2066
-1781
-1495
-1210
-924.3
-638.8
-353.2
-67.69
217.8
503.4
788.9
1074
Errors = Targets - Outputs


Training: R=-0.30118
Output ~= -0.68*Target + 5.4e+03
Data
Fit
Y = T
Target
Validation: R=-1
Output ~= -5.9*Target + 1.5e+04
Data
Fit
Y = T
Target
Test: R=-1
Output ~= -7.5*Target + 1.9e+04
Data
Fit
Y = T
Target
All: R=-0.47131
Output ~= -1.3*Target + 6.1e+03
Data
Fit
Y = T
Target

- **Wyniki:**

- dla pierwszego przypadku:

```
1.6633e+03
1.6880e+03
1.6867e+03
1.7764e+03
1.7800e+03
1.9495e+03
1.9825e+03
2.1477e+03
2.1791e+03
2.3262e+03
```

- dla drugiego przypadku:

```
1.6633e+03
1.6880e+03
1.6867e+03
1.7764e+03
1.7800e+03
1.9495e+03
1.9825e+03
2.1477e+03
2.1791e+03
2.3262e+03
```

- **Wnioski:**

- w ramach projektu udało mi się zrealizować wszystkie wymagane kroki
- sieciami wielowarstwowymi łatwo jest sterować, zmieniając poszczególne parametry, takie jak bezwładność czy współczynnik uczenia
- tempo uczenia sieci jest zmienne, z reguły im większy współczynnik uczenia, tym sieć uczy się szybciej
- w takiej sytuacji jednak generowana jest większa ilość błędów, gdyż tracona jest precyzja
- bardzo małe wartości współczynnika uczenia wpływają nie tylko na spowolnienie uczenia, ale też na zwiększenie liczby epok
- w ogólnym wniosku można stwierdzić, że algorytm z propagacją danych znacznie przyspiesza proces uczenia, jednak wykryć można więcej błędów
- co ciekawe, dla wartości 0 funkcja Rastrigin3D przyporządkowała również 0

- **Listing całego kodu programu wykonanego w MATLABie:**

- funkcja RastrignTest3D:

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

        fx = A * n;

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

- główny program:

```
close all; clear all; clc;

wejscie = [-2 -1.6 -1.2 -0.8 -0.4 0 0.4 0.8 1.2 1.6 2];
wyjscie = [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(3);
net.trainFcn = 'traingd';
net.trainParam.lr = 0.000001;
net.trainParam.mc = 0.1;
net = train(net, wejscie, wyjscie);
efekty = zeros(size(net));
for i = 1:11
    testowe(i) = RastrignTest3D(wejscie(i));
    efekty(i) = sim(net, wejscie(i));
end
```