

Michalina Nikiel

WIMiIP, Inżynieria Obliczeniowa

rok 3, grupa laboratoryjna nr 2

Podstawy sztucznej inteligencji – projekt nr 3 – sprawozdanie

Budowa i działanie sieci wielowarstwowej typu *feedforward*.

Cel ćwiczenia:

Celem ćwiczenia jest poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie kształtu funkcji matematycznej z użyciem algorytmu wstecznej propagacji błędów.

Zrealizowane kroki:

- Wygenerowanie danych uczących i testujących dla funkcji Rastrigin 3D dla danych wejściowych z przedziałów od -2 do 2.
- Przygotowanie wielowarstwowej sieci oraz algorytmu wstecznej propagacji błędów.
- Uczenie sieci dla różnych współczynników uczenia i bezwładności
- Testowanie sieci.

Sprawozdanie zawiera kolejno:

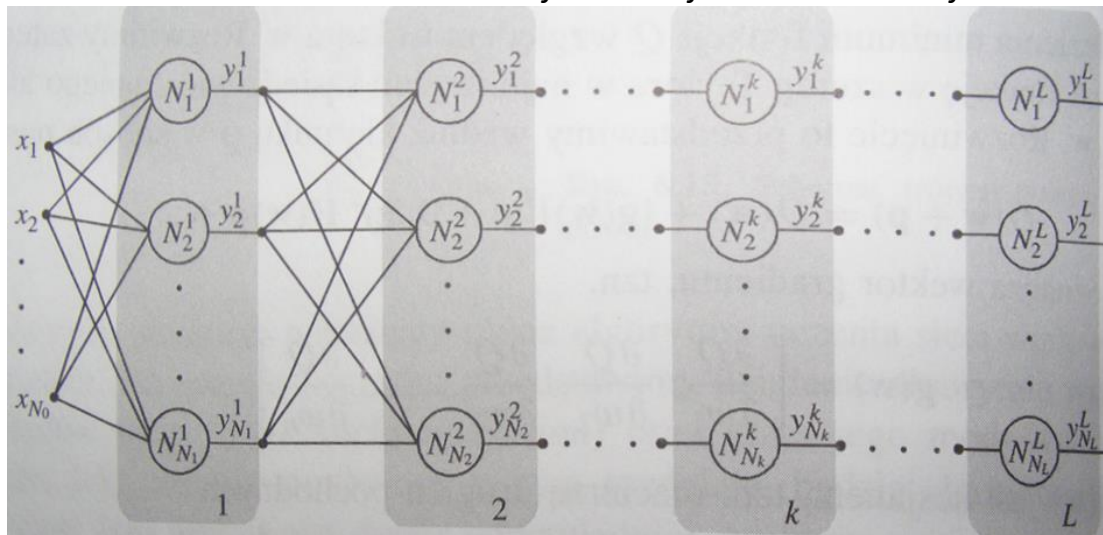
- Zagadnienia teoretyczne: opis budowy użytej sieci i algorytmu uczenia.
- Zestawienie otrzymanych wyników.
- Wnioski wraz z analizą i dyskusją błędów uczenia opracowanej sieci.
- Listing całego kodu programu wraz funkcją Rastrigin 3D.

Zagadnienia teoretyczne:

Sieć neuronowa jest systemem dokonującym określonych obliczeń na zasadzie równoczesnej pracy wielu połączonych ze sobą elementów zwanych neuronami.

Sieć wielowarstwowa jest to najpopularniejszy typ sztucznych sieci neuronowych. Sieć tego typu składa się zwykle z jednej warstwy wejściowej, kilku warstw ukrytych oraz jednej warstwy wyjściowej.

Schemat wielowarstwowej sztucznej sieci neuronowej:



Sieci feedforward to struktury, w których istnieje ściśle określony kierunek przepływu sygnałów – od pewnego ustalonego wejścia (na którym podaje się sieci sygnały będące danymi wejściowymi, precyzującymi zadania, które mają być rozwiązywane), do wyjścia, na którym sieć podaje ustalone rozwiązanie. Takie sieci są najczęściej stosowane i najbardziej użyteczne.

Propagacja wsteczna to algorytm uczenia nadzorowanego wielowarstwowych, jednokierunkowych sieci neuronowych. Podaje on przepis na zmianę wag dowolnych połączeń elementów przetwarzających rozmieszczonych w sąsiednich warstwach sieci. Oparty jest on na minimalizacji sumy kwadratów błędów (lub innej funkcji błędu) uczenia z wykorzystaniem optymalizacyjnej metody największego spadku.

Algorytm wstecznej propagacji błędów polega na takiej zmianie wag sygnałów wejściowych każdego neuronu w każdej warstwie, by wartość błędu dla kolejnych par uczących zawartych w zbiorze uczącym była jak najmniejsza. W tym celu wykorzystywana jest metoda gradientowa najszybszego spadku.

Schemat krokowy można przedstawić następująco:

1. Inicjalizacja sieci i algorytmu
2. Obliczanie wartości wyjściowej sieci na podstawie danych
3. Obliczanie błędu sieci
4. Korekcja wag
5. Czy sieć nauczona?
 - A. TAK – przejdź dalej
 - B. NIE – wróć do punktu 2
6. Koniec

Zestawienie otrzymanych wyników w postaci zrzutów ekranu i opisu:

Zadanie zostało wykonane z wykorzystaniem pakietu Matlab. Utworzona została sieć typu feedforward. Za pomocą zmian parametrów wykorzystany został algorytm propagacji błędów. W programie wygenerowano dane uczące i testujące z wykorzystaniem funkcji Rastrigin 3D. Uwzględniono dane z wejściowe zgodnie z poleceniem czyli z przedziału $[-2;2]$. Funkcja Rastrigin 3D została specjalnie stworzona w oddzielnym pliku. Jej działanie opiera się na obliczaniu wzoru:

$$f(\mathbf{x}) = An + \sum_{i=1}^n [x_i^2 - A \cos(2\pi x_i)]$$

Kod programu posiada tablicę złożoną z jedenastu elementów do których zapisywane są liczby z przedziału $[-2,2]$ z krokiem 0.4. Z kolei danymi wyjściowymi, które również są zapisane w tablicy złożonej z jedenastu elementów, są wyniki działania funkcji Rastrigin 3D odpowiednio dla danych wejściowych z przedziału $[-2,2]$.

Sieć wielowarstwowa została zaimplementowana z użyciem polecenia `feedforwardnet(2)`, gdzie 2 jest ilością warstw ukrytych w stworzonej sieci wielowarstwowej. Liczba ta została wybrana celowo, aby uniknąć ryzyka przeuczenia sieci. Przeuczenie pojawia się w przypadku zbyt długiego uczenia (działania algorytmu uczącego) lub wówczas, gdy zastosowana sieć jest zbyt złożona w porównaniu ze złożonością problemu lub liczbą dostępnych danych uczących.

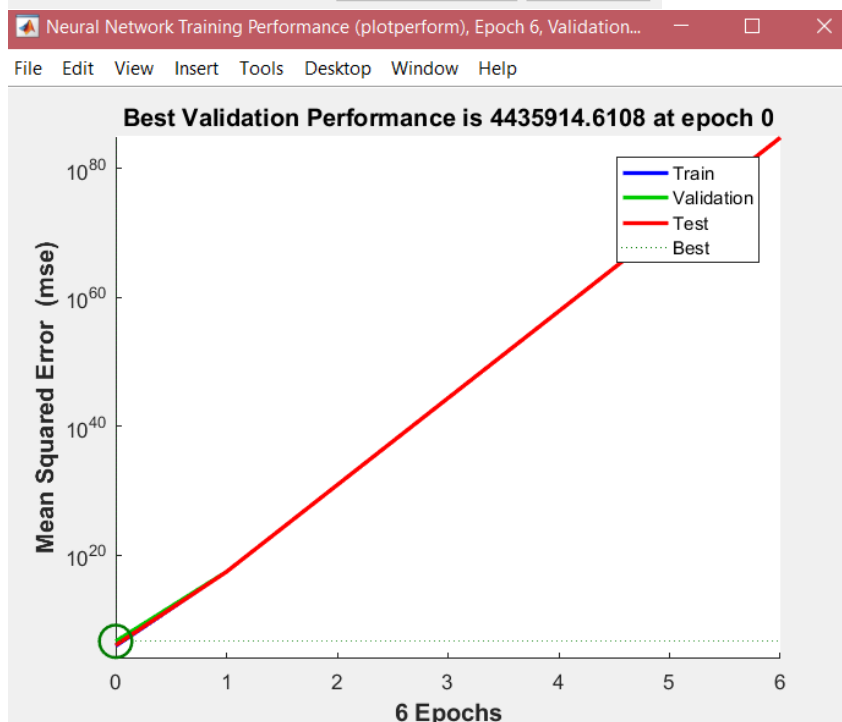
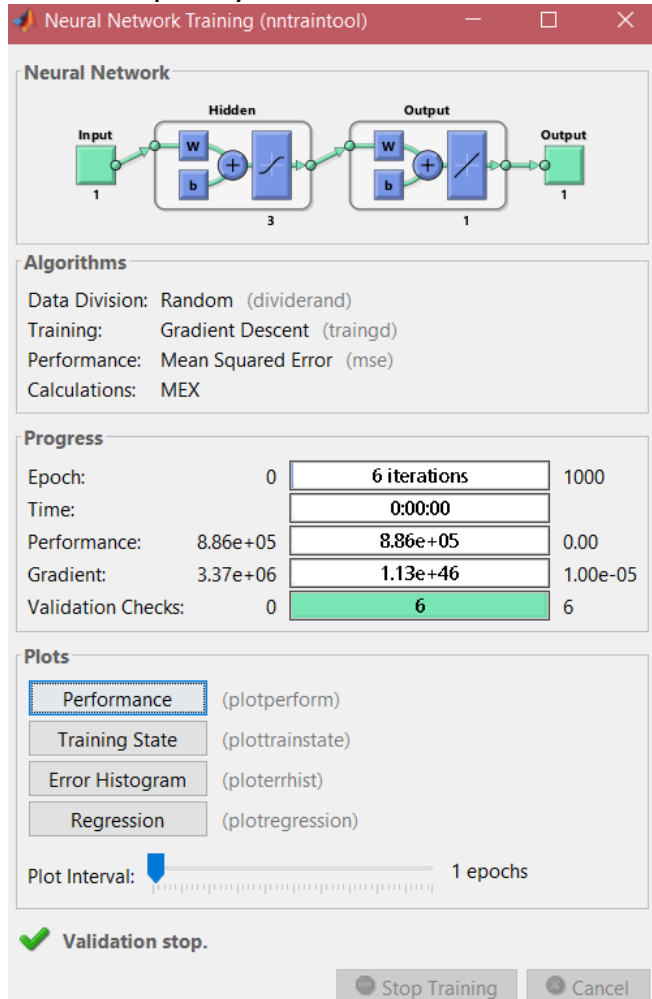
Kolejno, odbywa się testowanie działania sieci, która uczy z wykorzystaniem parametru treningu wstecznej propagacji :

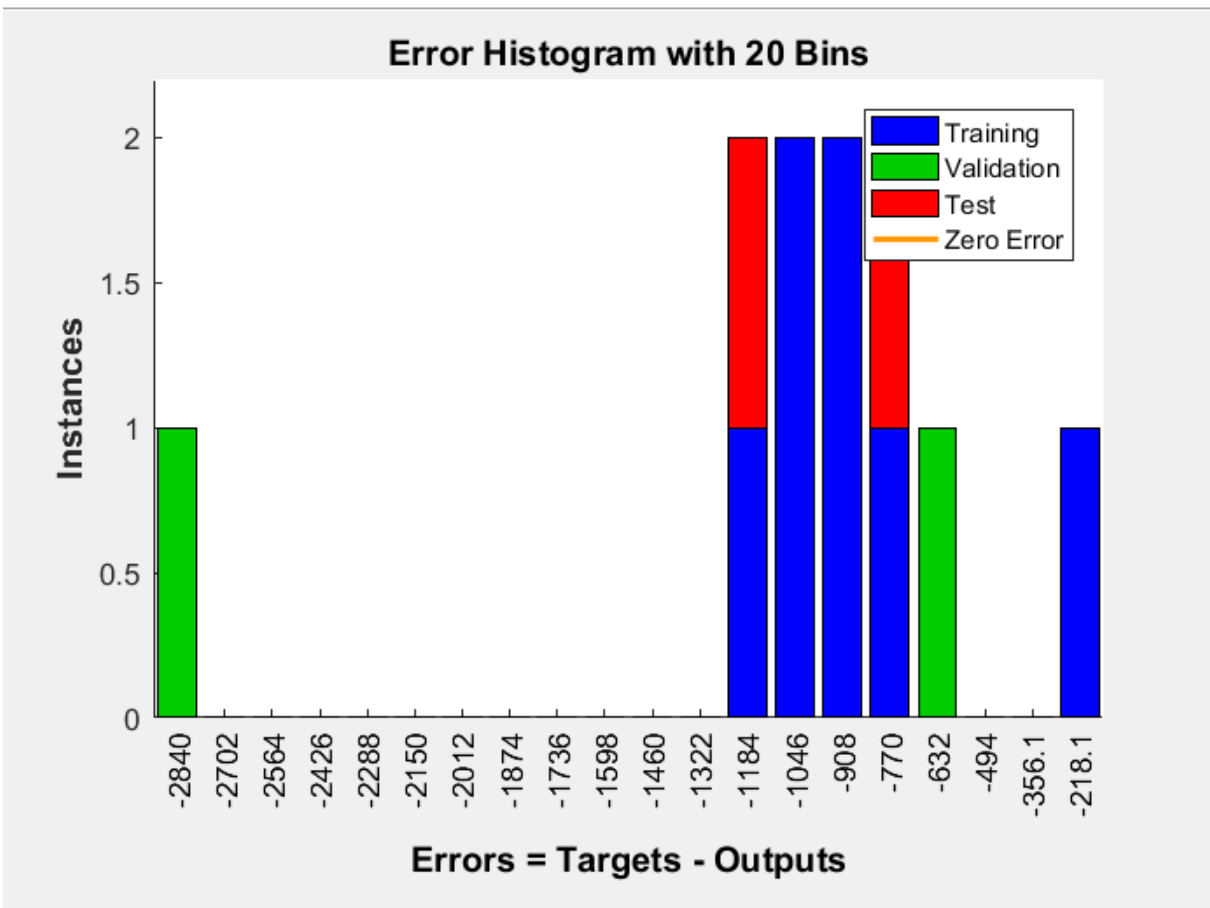
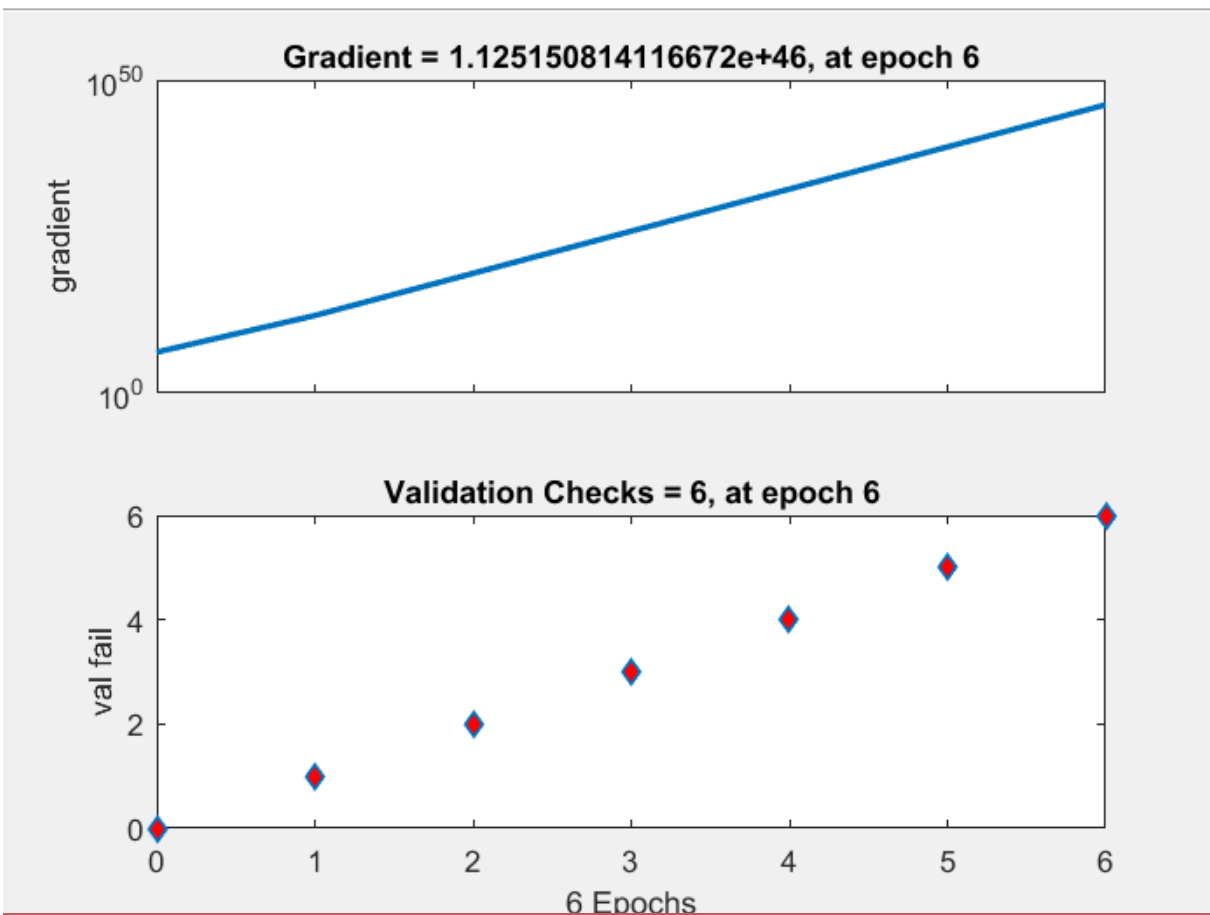
```
net.trainFcn = 'traingd'
```

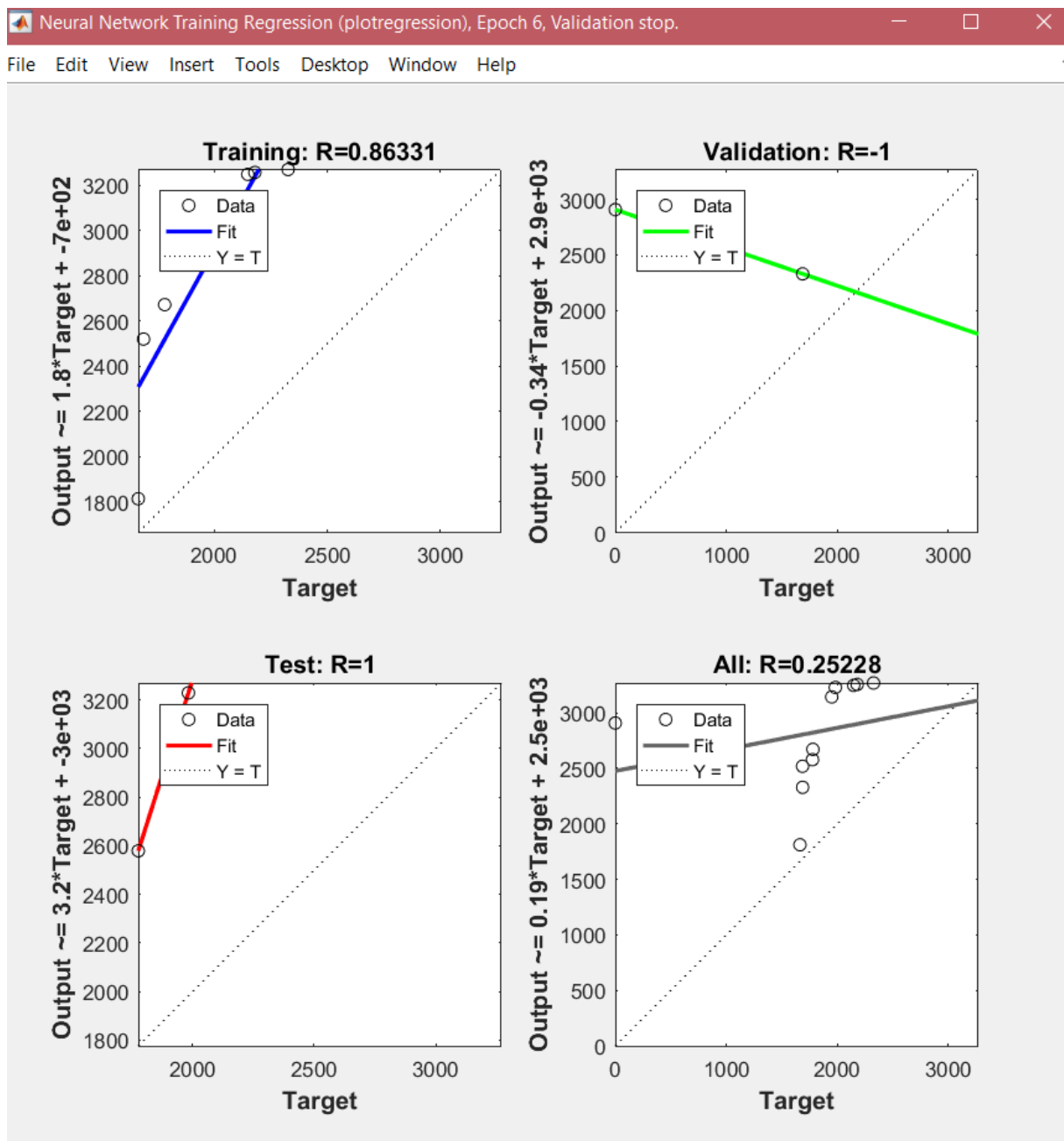
Dodane zostały również współczynnik uczenia oraz bezwładności. Dzięki zmianie tych parametrów otrzymałam wyniki, których zestawienie znajduje się poniżej w formie zrzutów ekranu i odpowiednich wniosków.

Zrzuty ekranu dla sieci wielowarstwowej typu feedforward o parametrach:

- Współczynnik uczenia: 0.5
- Współczynnik bezwładności: 0.5







Wyniki zmiennej efekt:


	1	2	3	4	5	6	7	8	9	10	11
1	21.5079	21.1516	255.6636	323.1981	409.1168	622.4606	835.8448	922.0272	990.9397	1.2317e+03	1.8918e+03

Zrzuty ekranu dla sieci wielowarstwowej typu feedforward o parametrach:

- Współczynnik uczenia: 0.000001
- Współczynnik bezwładności: 0

Neural Network Training (nntraintool)

Neural Network



Algorithms

Data Division: Random (dividerand)
Training: Gradient Descent (traingd)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	6 iterations	1000
Time:		0:00:00	
Performance:	1.13e+06	1.13e+06	0.00
Gradient:	2.58e+06	1.25e+09	1.00e-05
Validation Checks:	0	6	6

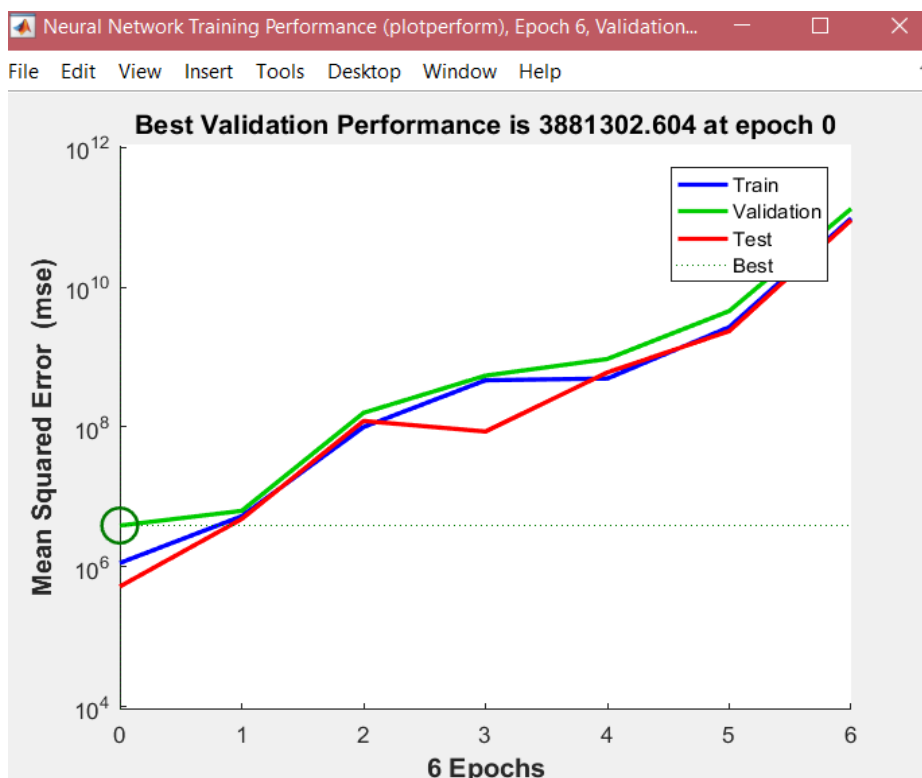
Plots

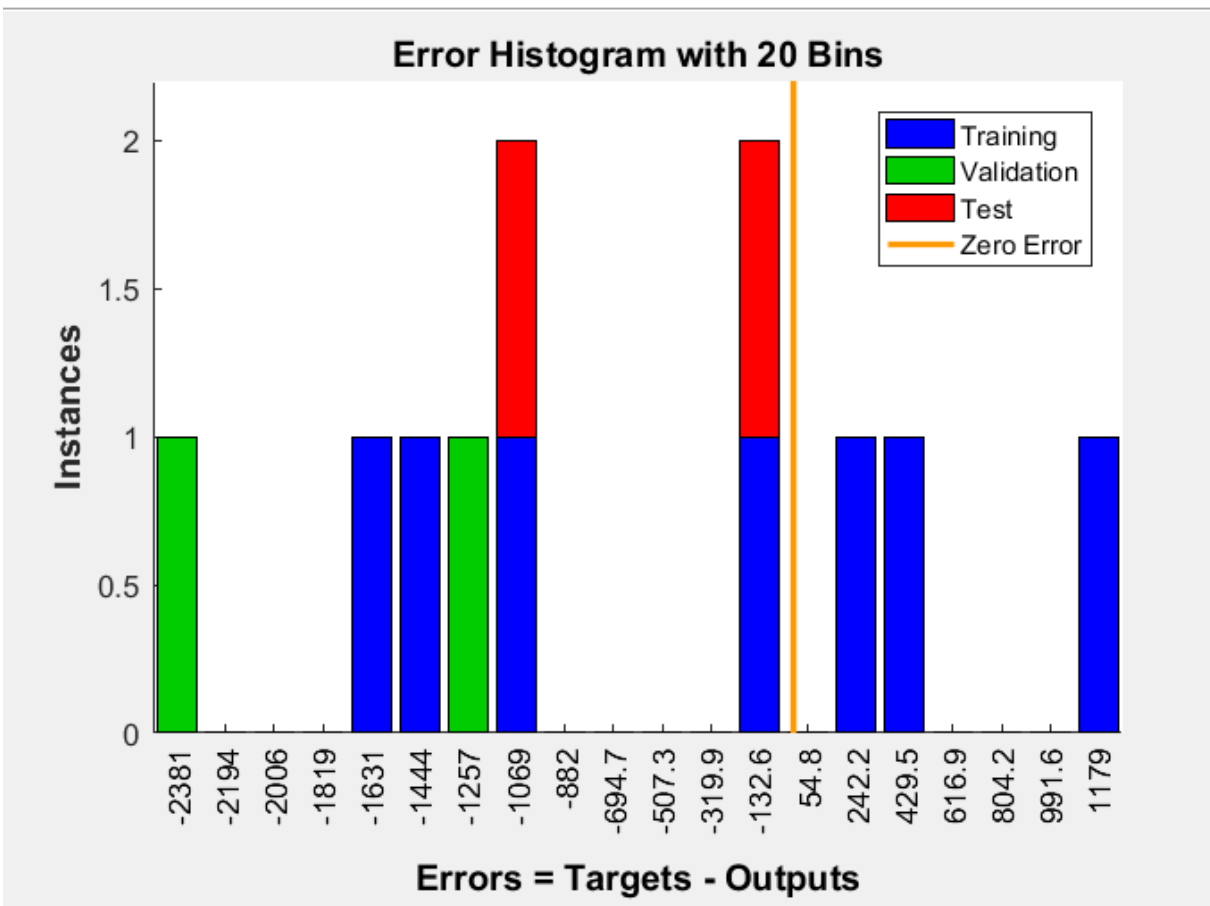
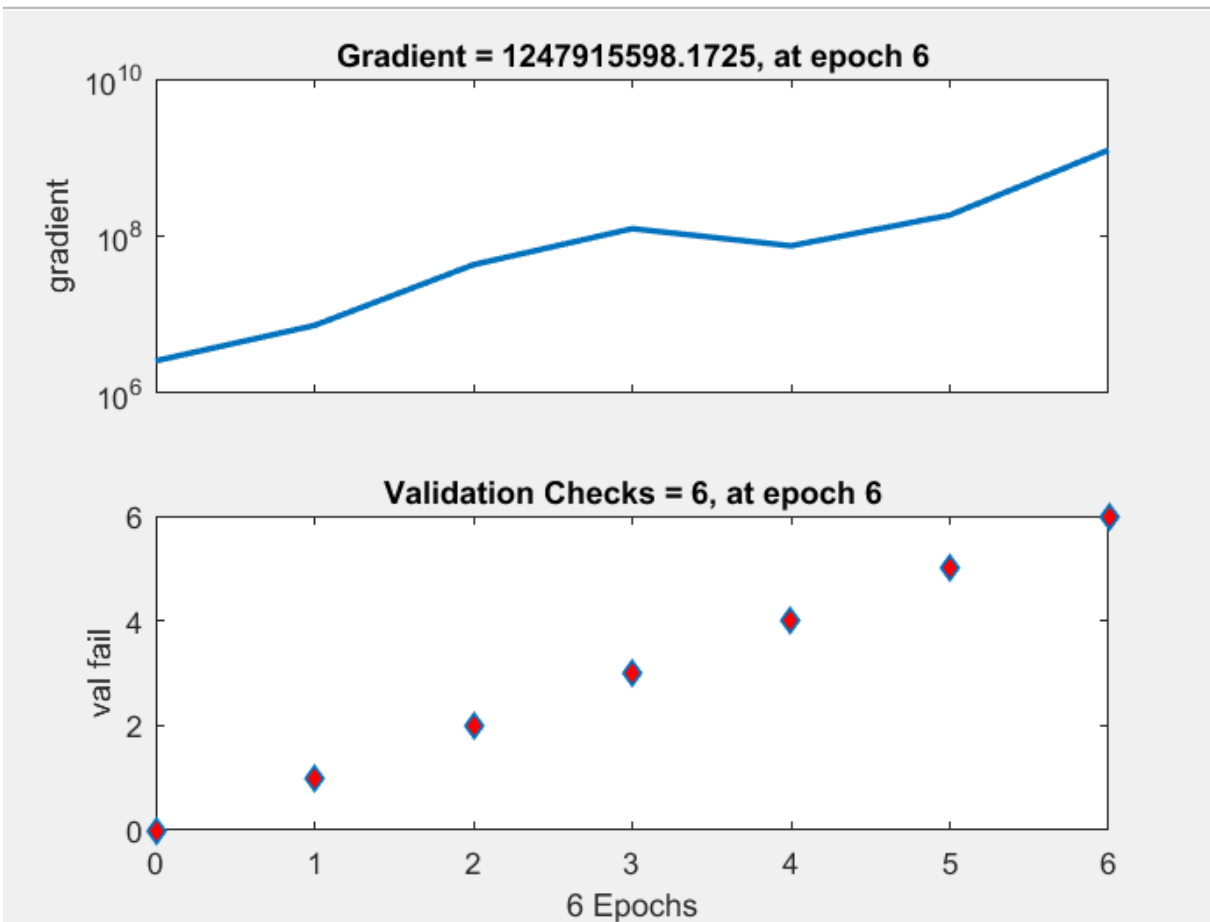
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)

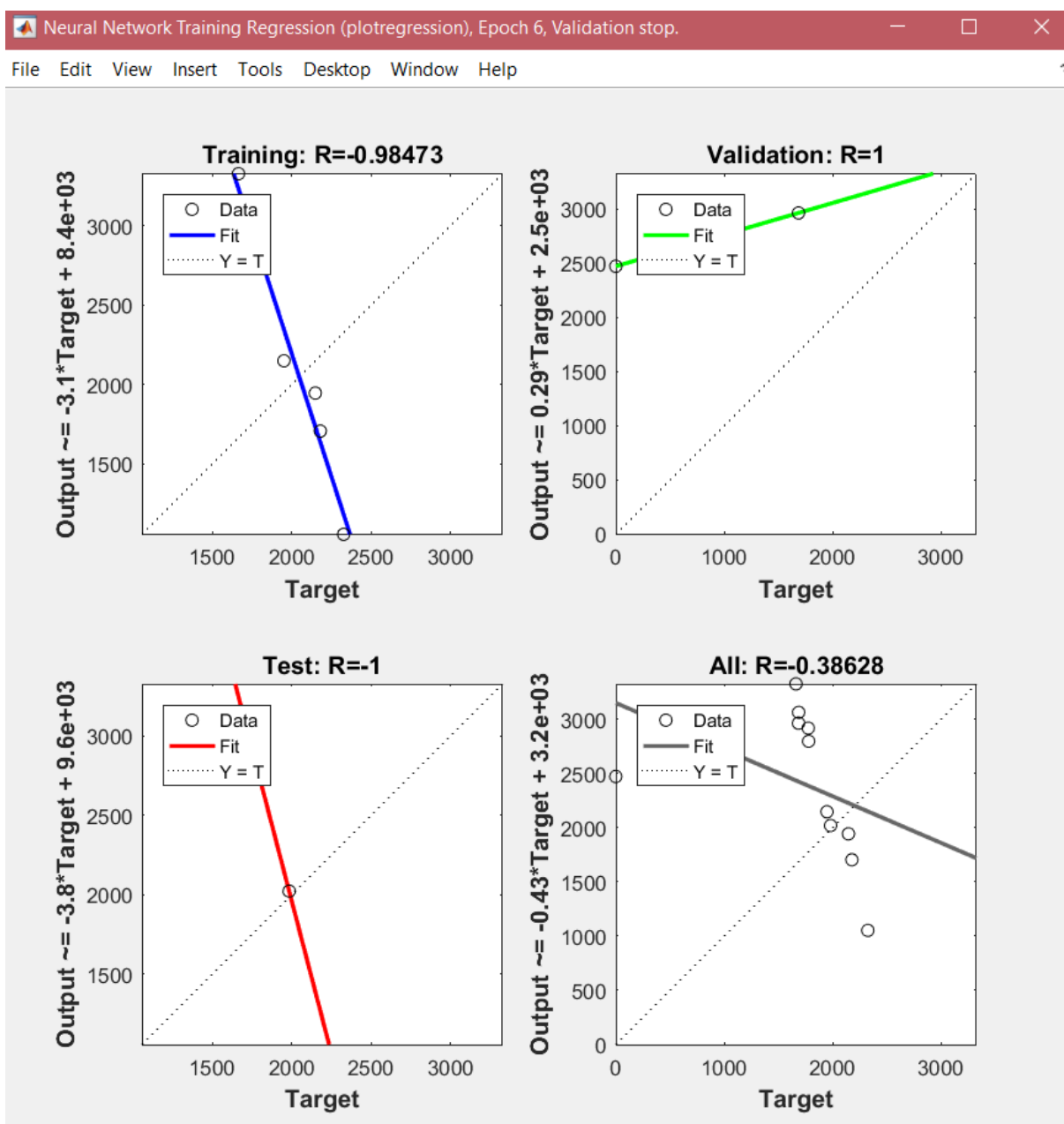
Plot Interval: 1 epochs

Opening Performance Plot

Stop Training Cancel







Wyniki zmiennej efekt:

	1	2	3	4	5	6	7	8	9	10	11
1	2.7213e+03	2.5326e+03	2.4660e+03	2.4588e+03	2.4910e+03	2.5878e+03	2.6833e+03	2.7072e+03	2.6561e+03	2.3901e+03	1.6477e+03

Wnioski:

- Udało się zrealizować wszystkie wymagane kroki, które składały się na projekt.
- Program tworzy sieć wielowarstwową z wykorzystaniem propagacji wstecznej i działa poprawnie. Został przetestowany z wykorzystaniem różnych parametrów.
- Podczas tworzenia sieci wielowarstwowej trzeba zwrócić największą uwagę na jej strukturę oraz na współczynnik uczenia.
- Wysoki współczynnik uczenia znacznie przyspiesza tempo uczenia sieci, ale należy brać pod uwagę, że w tym przypadku dana metoda może stracić na precyzji działania, co skutkuje większą ilością generowanych błędów.
- W przypadku sieci wielowarstwowej wpływ współczynnika uczenia jest wyraźny, więc lepiej dobierać większe współczynniki uczenia, które spowodują w miarę szybkie nauczenie sieci aniżeli oscylujące w granicy wartości 0.001, gdzie na wytrenowanie sieci trzeba było dłużej poczekać. Jeszcze mniejsze wartości powodują zwiększenie liczby epok.
- Mniej skomplikowane struktury perceptronów są lepszym rozwiązaniem, gdyż nie ma ryzyka wystąpienia przeuczenia.
- Funkcja Rastrigin 3D jest trudną do wyuczenia funkcją, ponieważ posiada bardzo mnóstwo minimów i maksimów lokalnych co znacznie utrudnia skuteczną naukę.

Listing kodu głównego i funkcji Rastrigin 3D:

```
1 close all; clear all; clc;
2
3 wej_in = [-2 -1.6 -1.2 -0.8 -0.4 0 0.4 0.8 1.2 1.6 2];
4 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];
5
6 testowe = zeros(1);
7
8 net = feedforwardnet(3); %tworzenie sieci z 2 warstwami ukrytymi
9 net.trainFcn = 'traingd'; %algorytm wstecznej propagacji
10 net.trainParam.lr = 0.5; %wsp. uczenia
11 net.trainParam.mc = 0.5; %bezwładność
12 net = train(net, wej_in, wej_out);
13 efekty = zeros(size(net));
14
15 for i = 1:11
16     testowe(i) = RastrignTest3D(wyj_in(i)); %wygenerowanie prawidłowych wyników działania funkcji RastrignTest3D
17     efekty(i) = sim(net, wej_in(i)); %test działania sieci
18 end
```

```
1  function fx = RastrignTest3D(x)
2      if x == 0
3          fx = 0;
4      else
5          A = 10;
6          n = 100;
7          x1 = x;
8          dx = (5.12-x)/n;
9          fx = A * n;
10
11         for i = 1:n
12             x = x1 + (i * dx);
13             fx = fx + (x^2) - (A * cos(2 * pi * x));
14         end
15     end
16 end
```