Paulina Reichel, IO gr.2

Temat projektu:

Budowa i działanie sieci wielowarstwowej typu feedforward.

Cel:

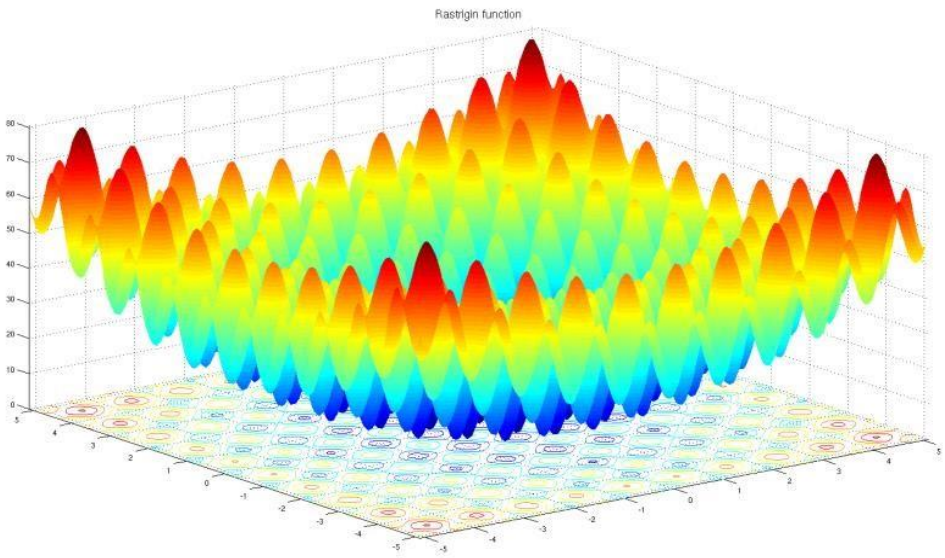
Poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie kształtu funkcji matematycznej z użyciem algorytmu wstecznej propagacji błędu.

Opis budowy sieci i algorytmów uczenia:

Celem budowanej sieci jest rozpoznawanie funkcji rastrigin. Funkcja ta w naszym przyjmuje za wejście współrzędne x, y z przedziału <-2; 2> i zwraca współrzędną z.

Wzór funkcji rastrigin: $f(x) = 10 * n + \sum [xi\ n\ 2\ i=1 - 10 * \cos(2\pi xi)$

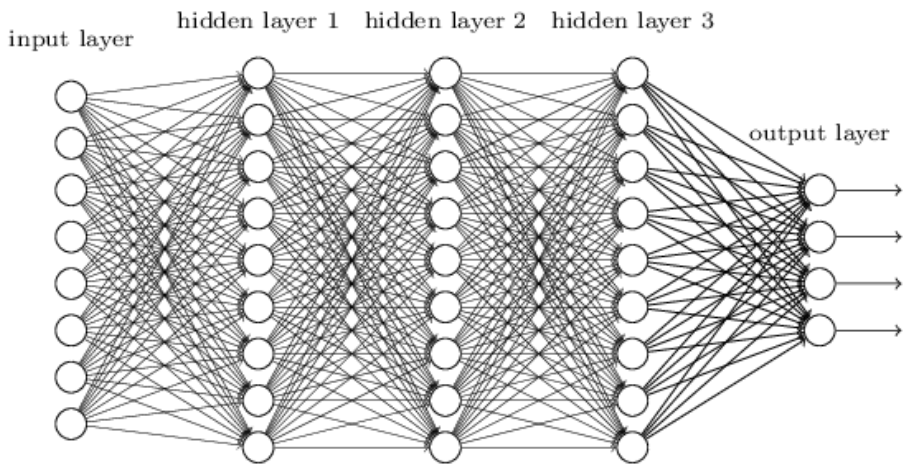
Rastrigin ma minimum globalne w punkcie (0, 0, 0).



Sieć wielowarstwowa składa się z warstwy wejściowej, co najmniej jednej warstwy ukrytej oraz warstwy wyjściowej. Warstwy ukryte służą do przetwarzania sygnałów w sieci neuronowej.

Perceptron wielowarstwowy w przeciwieństwie do perceptronu jednowarstwowego może być wykorzystywany do klasyfikowania zbiorów, które nie są liniowo separowalne .

Sieć MLP w swojej podstawowej wersji jest siecią, w której nie ma sprzężenia zwrotnego, w przeciwieństwie do sieci zwanych sieciami rekurencyjnymi . Na bazie sieci MLP zbudowane są splotowe sieci neuronowe, służące do rozpoznawania obrazów.



Zastosowany typ ***feedforward*** oznacza, że w sieci istnieje z góry określony kierunek przepływu danych – dane przechodzą od warstwy wejściowej przez wszystkie warstwy ukryte kończąc na warstwie wyjściowej (dane nie mogą się „cofać” w użytej sieci). Każda z warstw jest powiązana tylko z warstwą poprzednią i następną. Dane wyjściowe każdego neuronu w jednej warstwie są jednocześnie danymi wejściowymi dla neuronów w kolejnej warstwie na zasadzie każdy z każdym. Sygnał wyjściowy nie jest dzielony, więc jest podawany taki sam na wejścia wszystkich neuronów kolejnej warstwy. Natomiast neurony w jednej warstwie nie są ze sobą w żaden sposób połączone.

Algorytm wstecznej propagacji błędu zdecydowanie dominuje wśród metod uczenia jednokierunkowych sieci wielowarstwowych. Działanie polega na „przenoszeniu" błędu, jaki popełniła sieć, w kierunku od warstwy wyjściowej do warstwy wejściowej (a więc wstecz w stosunku do kierunku przepływu informacji).

Cykl uczenia metodą wstecznej propagacji błędu (backpropagation) składa się z etapów:

1. Wyznaczenie odpowiedzi neuronów warstwy wyjściowej oraz warstw ukrytych na zadany sygnał wejściowy.

2. Wyznaczenie błędu popełnianego przez neurony znajdujące się w warstwie wyjściowej i przesłanie go w kierunku warstwy wejściowej.

3. Adaptacja wag. Algorytm wstecznej propagacji błędu (backpropagation) określa procedurę korekty wag w sieci wielowarstwowej przy wykorzystaniu gradientowych metod optymalizacji. Korekta wektora wag sieci oparta jest na minimalizacji funkcji miary błędu (funkcji celu), którą określono jako sumę kwadratów błędów na wyjściach sieci. Jeśli aktualizacja wag uczonych neuronów odbywać się będzie po prezentacji każdego elementu wówczas funkcja celu ma postać:

$$E = \frac{1}{2}\sum_{k=1}^{m}\left(z_k(t) - y_k(t)\right)^2$$

Realizacja projektu:

Projekt został zrealizowany w programie Matlab, ponieważ ten program dostarcza użytkownikowi wiele gotowych funkcji m.in. do tworzenia sieci wielowarstwowych czy algorytmu propagacji błędu. Jako dane wejściowe do programu została przyjęta 11-elementowa tablica, zawierająca liczby z przedziału [-2,2], natomiast jako dane wyjściowe zostały wykorzystane wyniki działania funkcji Rastring 3D dla danych wejściowych.

Funkcja Rastring 3D została utworzona w oddzielnym pliku, co umożliwiło pozyskanie odpowiednich wyników.

Przy wykorzystaniu *funkcji feedforwardnet()* z biblioteki *Neural Networking Training Tool* została utworzona sieć wielowarstwowa. Jako argument funkcji użytkownik podaje liczbę warstw ukrytych sieci.

Funkcja *net.trainFcn= 'traingd'* umożliwiła przetestowanie działania sieci z wykorzystaniem algorytmu wstecznej propagacji błędu.

Podczas testowania programu modyfikowane były również:

- ✓ Współczynnik uczenia (zmieniany kolejno na 0.5, 0.1, 0.01)
- ✓ Współczynnik bezwładności (zmieniany kolejno na . 0, 0.5, 1)

Wnioski:

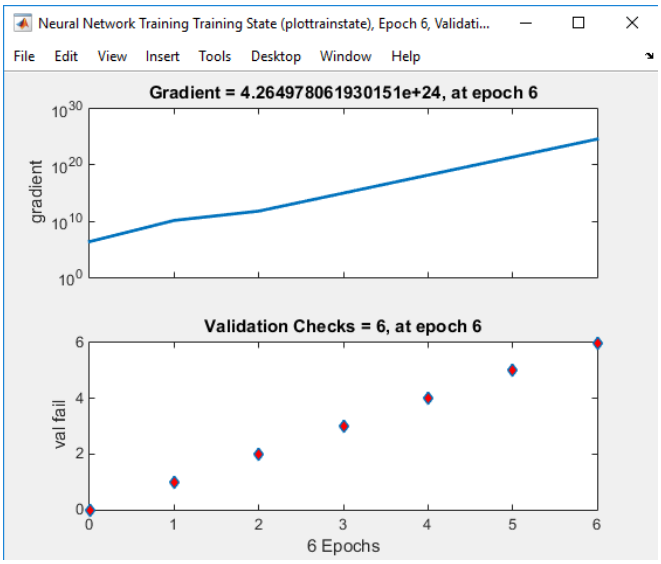
- ✓ Wszystkie wymagania zawarte w scenariuszu, zostały zrealizowane
- ✓ Przetestowanie programu przy różnej kombinacji parametrów, pozwoliło na stwierdzenie poprawnego działania kodu, tworzącego sieć wielowarstwową z propagacją wsteczną
- ✓ Funkcja Rastrigin 3D zdecydowanie nie jest optymalną funkcją do wyuczenia, ponieważ jej duża ilość minimum i maksimum lokalnych, znacznie zmniejsza efektywność uczenia
- ✓ Przy budowie sieci największą uwagę należy zwrócić na strukturę sieci oraz współczynnik uczenia
- ✓ Efektywniej uczą się sieci wielowarstwowe z większą liczbą neuronów oraz warstw ukrytych
- ✓ Przy bardziej skomplikowanych strukturach sieciowych istnieje ryzyko przeuczenia sieci
- ✓ Wraz ze wzrostem współczynnika uczenia, wzrasta tempo uczenia sieci, lecz prowadzi do również do zwiększenia błędów, ponieważ precyzja działania spada
- ✓ Dobór współczynnika uczenia ma duży wpływ na uczenie się sieci
- ✓ Im mniejszy współczynnik uczenia, tym wzrasta ilość epok potrzebnych do wytrenowania perceptronów (wartości ok. 0.001 powodują dłuższy czas obliczeń)

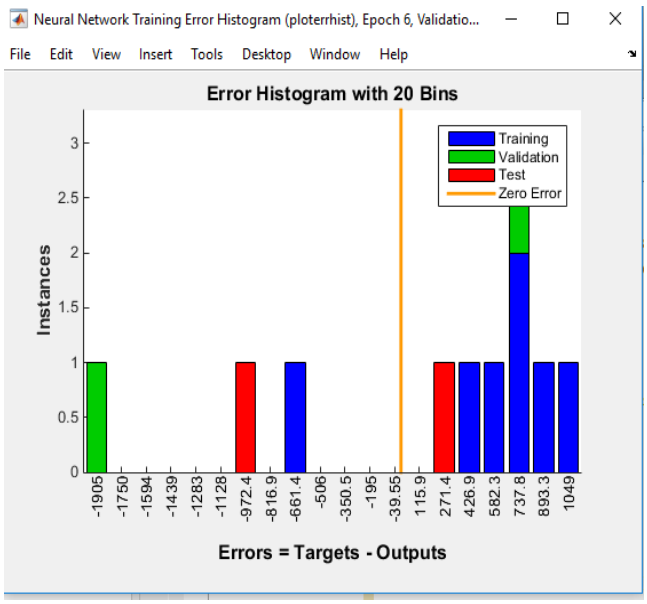
Wyniki działania sieci oraz zrzuty ekranów wybranych kombinacji:

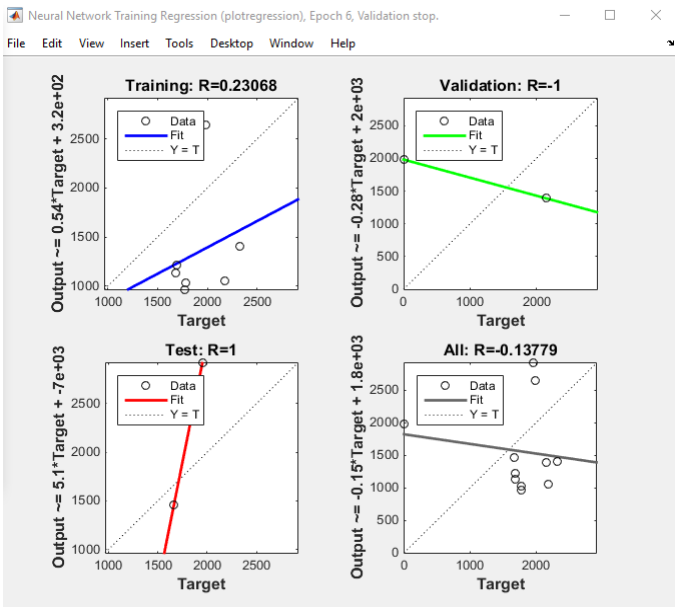
- ✓ Liczba ukrytych warstw – 5
- ✓ Współczynnik uczenia - 0.0001
- ✓ Bezwładność – 0

Wyniki:

	1	2	3	4	5	6	7	8	9	10	11
1	2.5980e+03	3.0076e+03	3.0519e+03	3.0345e+03	2.9684e+03	2.4626e+03	1.9420e+03	1.6672e+03	961.2265	815.9346	1.3685e+03



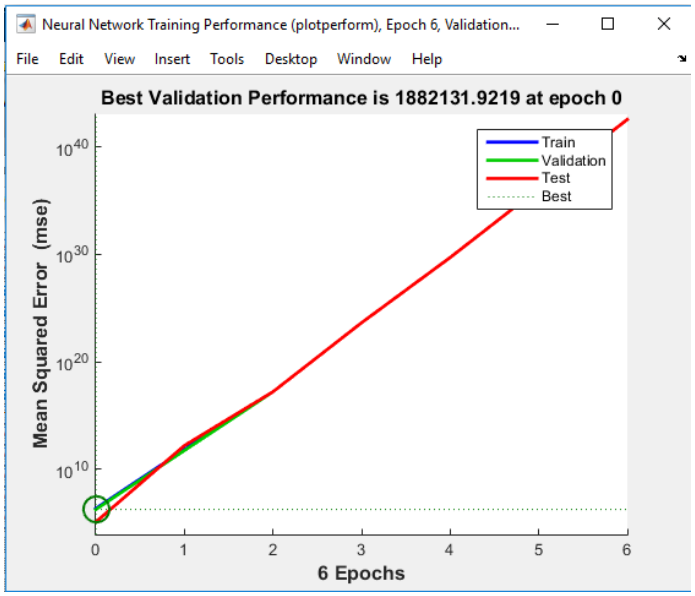


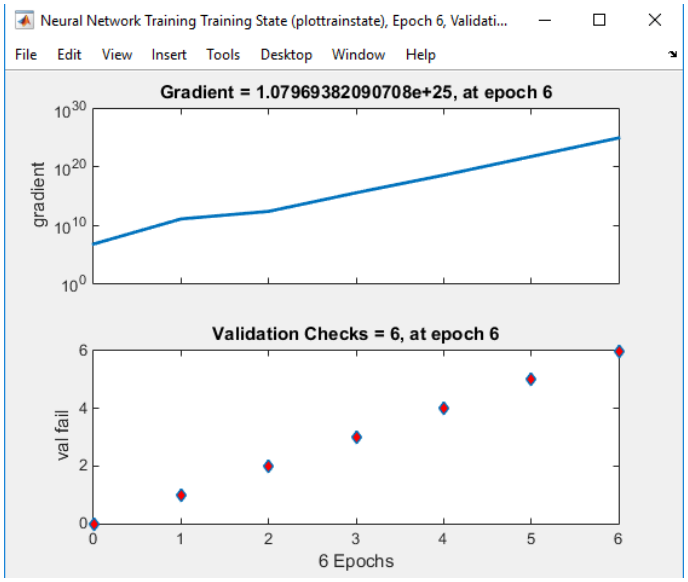


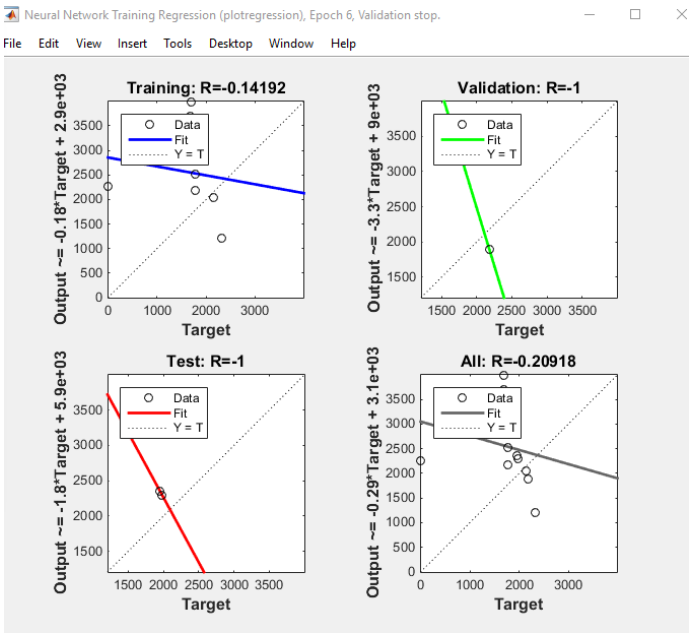
- ✓ Liczba ukrytych warstw – 5
- ✓ Współczynnik uczenia - 0.0001
- ✓ Bezwładność – - 0.0001
- ✓

Wyniki:

	1	2	3	4	5	6	7	8	9	10	11
1	3.5824e+03	3.9965e+03	3.6947e+03	2.5259e+03	2.1833e+03	2.2584e+03	2.3537e+03	2.2930e+03	2.0488e+03	1.8936e+03	1.2068e+03



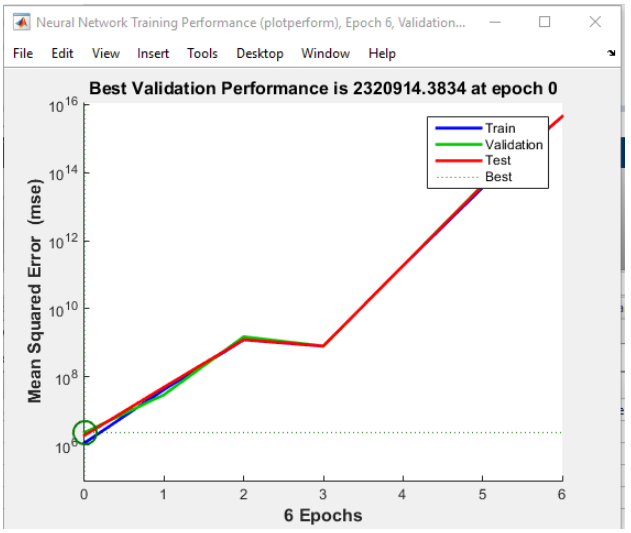


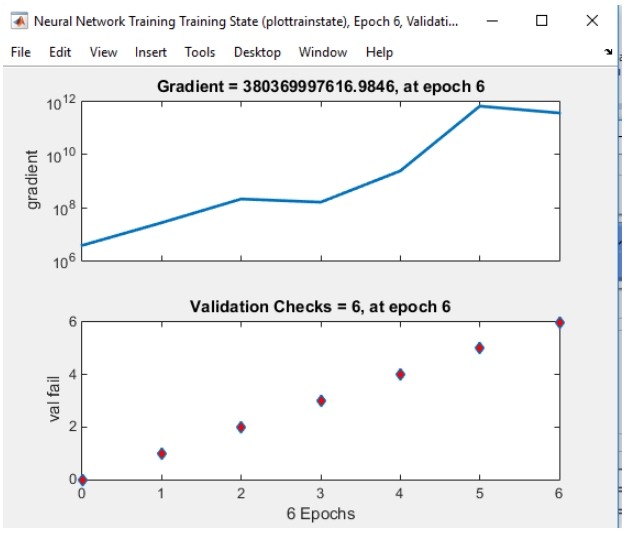


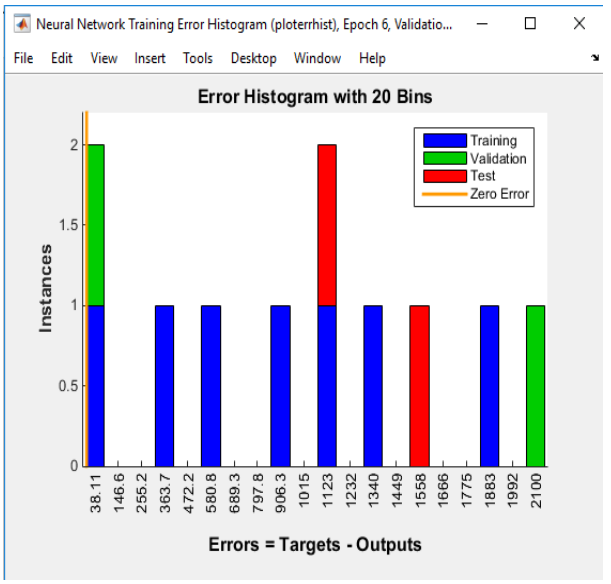
- ✓ Liczba ukrytych warstw – 5
- ✓ Współczynnik uczenia - 0.000001
- ✓ Bezwładność – 0.5

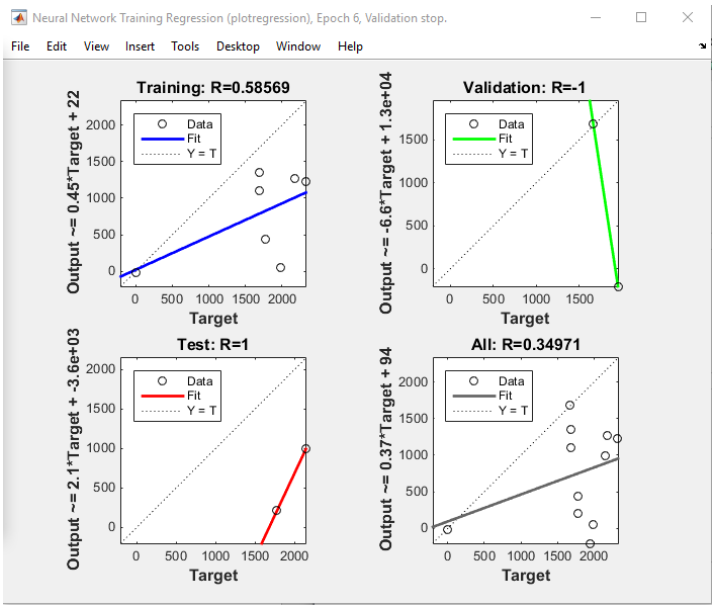
Wyniki:

	1	2	3	4	5	6	7	8	9	10	11
1	1.6795e+03	1.3468e+03	1.1064e+03	435.8526	208.4821	-15.5160	-204.9298	54.2632	992.2749	1.2698e+03	1.2326e+03









Listing programu głównego:

```
wejscie = [-2 -1.6 -1.2 -0.8 -0.4 0 0.4 0.8 1.2 1.6 2];
weyjscie = [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(5); %tworzenie sieci z 2 warstwami ukrytymi
net.trainFcn = 'traingd'; %algorytm wstecznej propagacji
net.trainParam.lr = 0.000001; %wsp. uczenia
net.trainParam.mc = 0.5; %bezwladnosc
net = train(net, wejscie, wejyjscie);
wyniki = zeros(size(net));

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

Funkcja pomocnicza do wygenerowania danych wyjściowych:

```
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
      end
end
```