Sprawozdanie Scenariusz 3 Aleksandra Karaś

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 błędu

2. Zadania do wykonania

a) Wygenerowanie danych uczących i testujących dla funkcji Rastrigin 3D dla danych wejściowych z przedziałów od -2 do 2. formuła funkcji: https://en.wikipedia.org/wiki/Rastrigin_function

b) Przygotowanie (implementacja lub wykorzystanie gotowych narzędzi) wielowarstwowej sieci oraz algorytmu wstecznej propagacji błędu.

c) Uczenie sieci dla różnych współczynników uczenia (np. 0.5, 0.1, 0.01) i bezwładności (np. 0, 0.5, 1). d) Testowanie sieci.

3. Opis budowy użytej sieci i algorytmu 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.

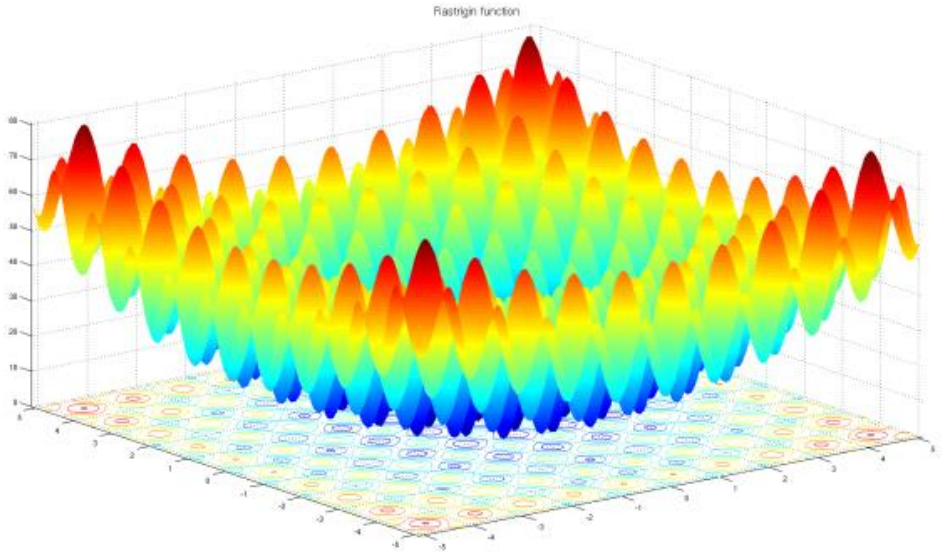
Wzór funkcji rastrigin:

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

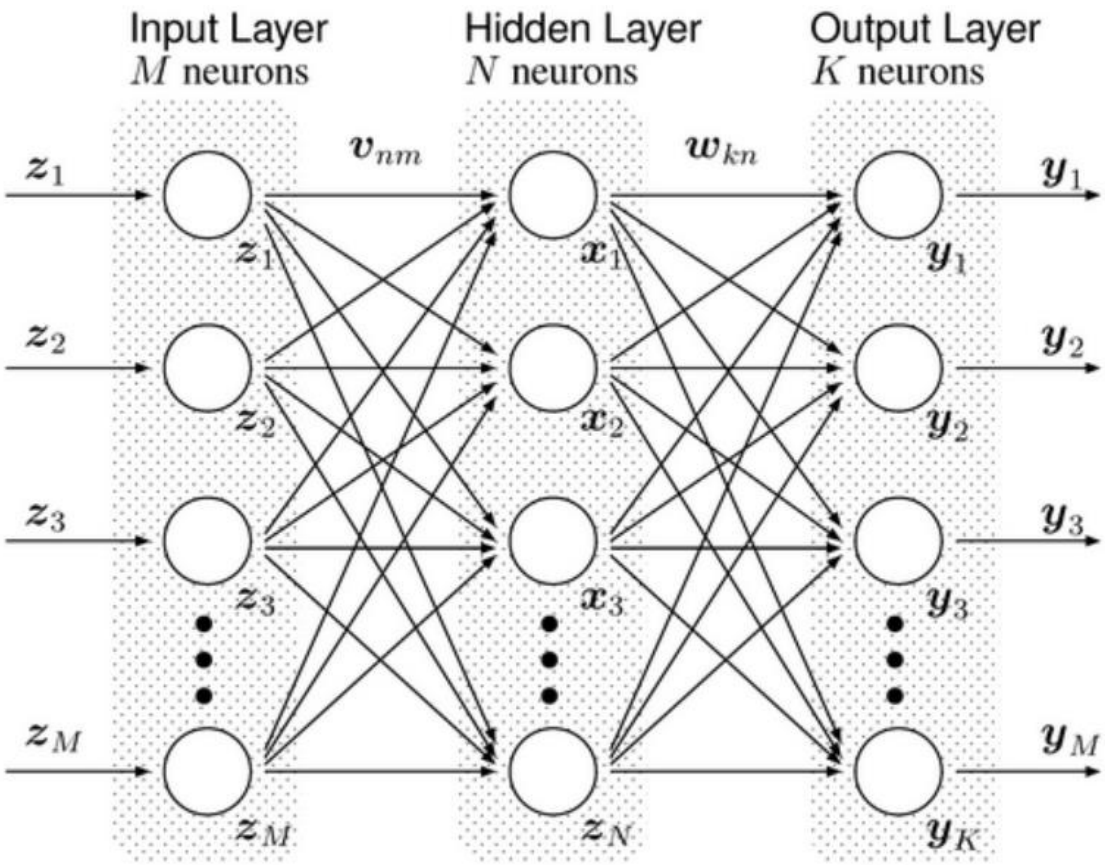
Gdzie $A = 10$ oraz $x_i \in [-5.12, 5.12]$

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

Wykres funkcji Rastrigin



Ogólny schemat sieci neuronowej wielowarstwowej



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

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 seci). 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.

4. Przebieg programu

Dane wejściowe: tablica 21 liczb zmiennoprzecinkowych z przedziału [-2;2] z krokiem 0.2. Uwzględnione zostały końce przedziału.

Dany wyjściowe: 21 elementowa tablica zawierająca wyniki działania funkcji Rastring dla danych wejściowych.

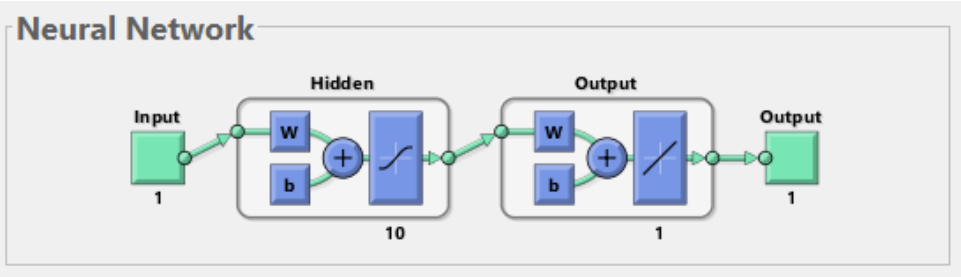
Do zaimplementowania sieci wielowarstwowej użyłam polecenia feedforwardnet(). Polecenie to tworzy sieć wielowarstwową z 10-cioma warstwami ukrytymi. Można utworzyć sieć o mniejszej bądź większej ilości warstw. Domyślnie funkcja ta bierze 10 warstw ukrytych.

Nasz program nie jest na tyle złożony, żeby używać większej ilości warstw ukrytych.

Następnie wprowadziłam testowanie sieci, poprzez dodanie parametru treningu net.trainFcn = 'traingd'

Działanie uzależnione było głównie od współczynnika uczenia oraz współczynnika bezwładności.

Schemat działania sieci w programie MATLAB



5. Wyniki

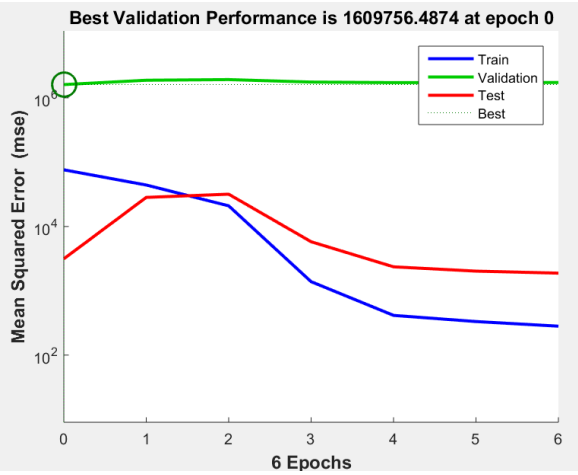
	1.0e+03 *	Bez propagacji wstecznej							
współczynnik uczenia	0.5			0.1			0.01		
współczynnik bezwładności	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0
-2.0	1.6689	1.9218	1.3886	1.8437	1.9265	1.6621	1.6626	1.8401	3.0728
-1.6	1.6772	1.9090	1.3849	1.7724	1.8996	1.6918	1.6687	1.8434	2.6332
-1.2	1.6974	1.9015	1.3833	1.6438	1.8698	1.7236	1.6688	1.8444	2.3738
-0.8	1.7408	1.8984	1.3832	1.4516	1.8367	1.7596	1.6688	1.8448	2.2650
-0.4	1.8081	1.8972	1.3863	1.2479	1.8000	1.8033	1.6688	1.8449	2.2251
0.0	1.8771	1.8967	1.4052	1.1863	1.7596	1.8565	1.6688	1.8449	2.2083
0.4	1.9375	1.8963	1.4991	1.5020	1.7150	1.9209	1.6688	1.8449	2.1928
0.8	2.0026	1.8954	1.8051	2.0025	1.6662	1.9982	1.6688	1.8449	2.1579
1.2	2.0857	1.8929	2.1761	2.2223	1.6130	2.0899	1.6688	1.8449	2.0636
1.6	2.1911	1.8870	2.3237	2.2670	1.5626	2.1972	1.6688	1.8449	1.8392
2.0	2.3120	1.8769	2.3562	2.2710	1.6631	2.3208	1.6688	1.8449	1.4588

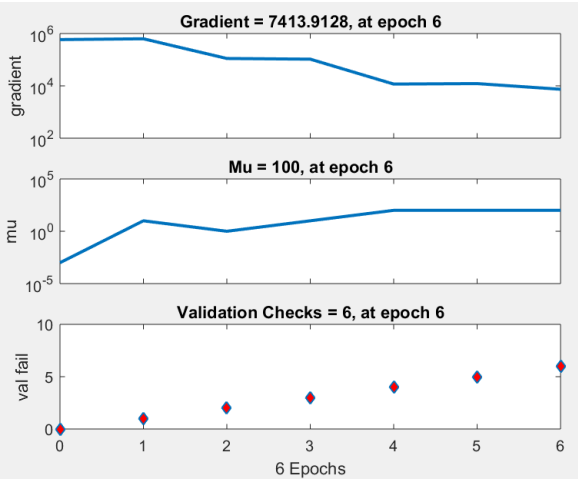
	1.0e+03 *	Z propagacją wsteczną							
współczynnik uczenia	0.5			0.1			0.01		
współczynnik bezwładności	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0
-2.0	1.9441	2.2150	1.2266	1.1264	1.1595	0.0277	1.0295	0.	1.9127	1.8402
-1.6	1.9082	2.2631	1.4738	1.3294	1.2489	0.5210	0.7934	2.3762	1.9250
-1.2	1.8871	2.2915	1.6196	1.4492	1.3016	0.8120	0.6541	2.6496	1.9748
-0.8	1.8783	2.3032	1.6802	1.4996	1.3236	0.9340	0.5961	2.7640	1.9954
-0.4	1.8752	2.3072	1.7009	1.5182	1.3314	0.9784	0.5761	2.8051	2.0017
0.0	1.8745	2.3076	1.7049	1.5267	1.3336	0.9960	0.5713	2.8198	2.0006
0.4	1.8750	2.3052	1.6971	1.5361	1.3331	1.0098	0.5759	2.8273	1.9906
0.8	1.8775	2.2968	1.6676	1.5586	1.3295	1.0381	0.5955	2.8375	1.9587
1.2	1.8849	2.2729	1.5830	1.6203	1.3187	1.1136	0.6524	2.8625	1.8688
1.6	1.9026	2.2156	1.3802	1.7674	1.2925	1.2930	0.7889	2.9212	1.6534
2.0	1.9326	2.1184	1.0363	2.0167	1.2482	1.5969	1.0204	3.0206	1.2882

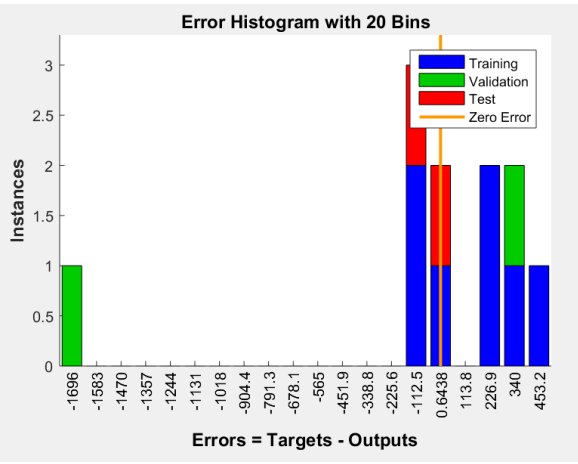
Widać, że przy współczynniku uczenia 0.5 i współczynniku bezwładności wynik jest dużo mniejszy. To samo można powiedzieć o współczynniku uczenia 0.1 dla wszystkich odpowiadających mu współczynników bezwładności. Przy współczynniku uczenia 0.01 dużo lepiej wypadł algorytm z propagacją wsteczną.

Porównanie działania sieci neuronowej dla współczynnika uczenia 0.5 oraz współczynnika bezwładności 0.5

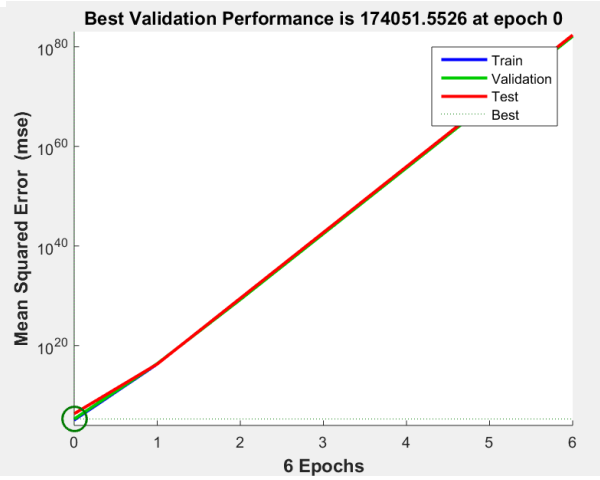
Bez algorytmem wstecznej propagacji

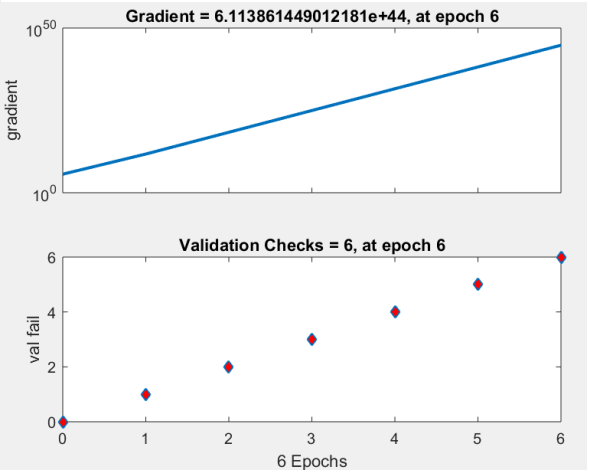


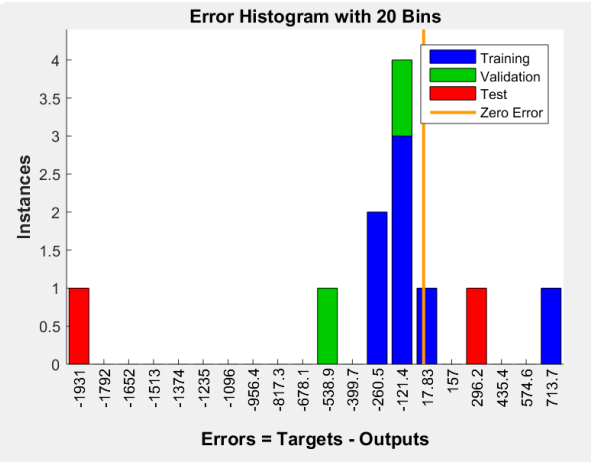


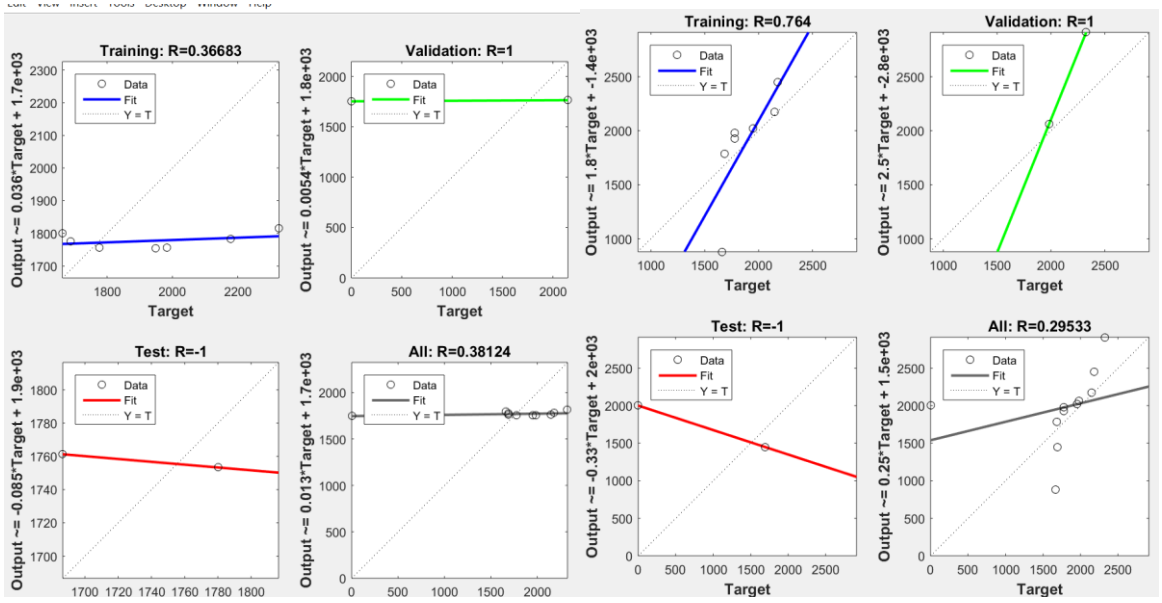


Z algorytmem wstecznej propagacji









6. Wnioski

Podczas tworzenia sieci wielowarstwowej trzeba zwrócić największą uwagę na jej strukturę oraz na współczynnik uczenia. Sieci wielowarstwowe posiadające większą liczbę neuronów oraz warstw ukrytych na ogół uczą się bardziej efektywnie od sieci z ich mniejszą ilością. Jednakże w samej strukturze sieci ważne jest rozmieszczenie danych neuronów. Im więcej neuronów w początkowych warstwach ukrytych, tym sieć zaczyna uczyć się gorzej aniżeli w przypadku, gdy miałaby ich mniej.

Ponadto trzeba zwrócić uwagę na współczynnik uczenia, który pełni także olbrzymią rolę w procesie uczenia. W przypadku sieci wielowarstwowej jego wpływ jest wyraźny, więc lepiej dobierać większe współczynniki uczenia, które spowodują w miarę szybkie nauczenie sieci aniżeli oscylujące w granicy wartości 0.1, gdzie na wytrenowanie sieci trzeba było poczekać kilka, kilkanaście minut.

Dla dużych sieci i ciągów uczących składających się z wielu tysięcy wektorów uczących ilość obliczeń wykonywanych podczas całego cyklu uczenia jest gigantyczna, a więc i czasochłonna. Nie zdarza się także, aby sieć została dobrze zbudowana od razu. Zawsze jest ona efektem wielu prób i błędów. Ponadto nigdy nie mamy gwarancji, że nawet prawidłowa sieć nie utknie w minimum lokalnym podczas gdy interesuje nas znalezienie minimum globalnego. Dlatego algorytmy realizujące SSN wyposaża się w mechanizmy dające nauczycielowi możliwość regulacji szybkości i jakości uczenia. Są to współczynniki: uczenia i momentum. Wpływają one na stromość funkcji aktywacji i regulują szybkość wpływu zmiany wag na proces uczenia.

7. Listing programu

```
close all; clear all; clc;

input = [-2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0]
output = [1663.3144032672324 1690.1132374370839 1688.034998342087 1674.099059643066 1686.7210309127113 1732.9225480289
test = zeros(1);

net = feedforwardnet();%tworzymy siec z 8 warstwami ukrytymi

net.trainFcn = 'traingd';%algorytm wstecznej propagacji

net.trainParam.lr = 0.1;%wspolczynnik uczenia 0.5 0.1 0.01

net.trainParam.mc = 0.5;% wspolczynnik bezwladnośsci 0.0 0.5 1

net = train(net, input, output);
result = zeros(size(net));
% genereujemy wyniki dla funkcji rstrigin oraz testujemy dzialanie sieci.
for k = 1:21
    test(k) = rastriginFunction(in(k));
    result(k) = sim(net, in(k));
end

function fx = rastriginFunction(x)
    if x == 0
        fx = 0;
    else
        x1 = x; A = 10; n = 100;
        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
```