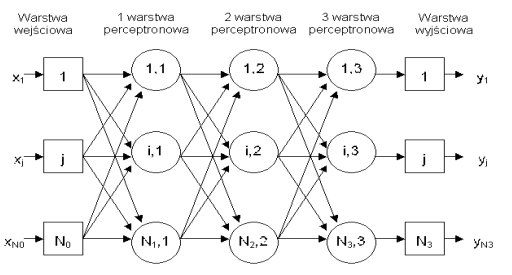
Aleksandra Spiecha

Podstawy sztucznej inteligencji

**Sprawozdanie nr 3**

**Cel**: Celem ćwiczenia jest poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie kształtu funkcji matematycznej z użyciem algorytmu wstecznej propagacji błędu oraz poznanie budowy i działania sieci wielowarstwowej typu feedforward​.

Sieć wielowarstwowa jest siecią połączonych ze sobą neuronów, w której neurony  
przesyłają sygnały wszystkim neuronom kolejnej warstwy (na zasadzie „każdy z  
każdym”). Pozwala ona na tworzenie sieci niemal o dowolnej charakterystyce. Działanie  
takiej sieci polega na liczeniu odpowiedzi neuronów w kolejnych warstwach – najpierw w  
pierwszej, do której trafiają sygnały z wejść sieci, potem korzystając z wyników  
poprzedniej warstwy, liczymy odpowiedzi kolejnych warstw neuronów. Odpowiedzi  
znajdujące się w ostatniej warstwie są wynikami sieci.



Do uczenia perceptronu użyłam algorytmu RPROP (od ang. Resilient backPROPagation). Jest to algorytm przeznaczony dla pełnego (wsadowego) trybu korekcji parametrów (wag). Oznacza to, że jedno skorygowanie parametrów (wag) następuje dopiero po przeglądnięciu przez sieć całego zbioru uczącego i obliczeniu sumarycznego a tym samym dokładnego gradientu. Kluczowymi elementami algorytmu RPROP są: wykorzystywanie jedynie samego znaku każdej składowej gradientu (natomiast wartości są pomijane), a także modyfikowanie współczynnika (współczynników) uczenia w każdym kroku. Współczynnik uczenia jest zwiększany, gdy znaki kolejnych gradientów pozostają zgodne, natomiast zmniejszany (a dokładnie połowiony), gdy są różne.

**Opis działania**  
a) Wzór dla funkcji Rastrign 3D:

  
Gdzie: A = 10,  
xi∈ [-5.12; 5.12]  
f(0) = 0  
b) Dane wejściowe - tablica 11-elementowa, liczby z przedziału [-2; 2] z krokiem  
0.4.  
c) Dane wyjściowe - tablica 11- elementowa zawierającą wyniki działania funkcji  
Rastrign 3D dla danych wejściowych.  
d) Do zaimplementowania sieci wielowarstwowej polecenie feedforwardnet(2), które  
tworzy sieć wielowarstwową zawierającą 2 warstwy ukryte.  
e) Testowanie działania sieci, z użyciem algorytmu wstecznej propagacji i bez niego  
– aby wprowadzić ten algorytm należy dodać parametr treningu  
net.trainFcn = 'traingd'.

**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

**Listing kodu:**

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

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

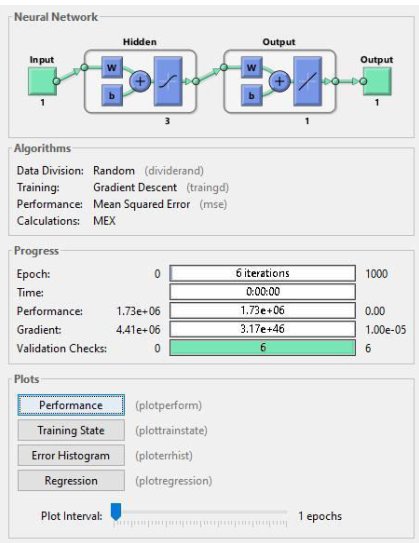
testowe(i) = RastrignTest3D(wejscie(i));

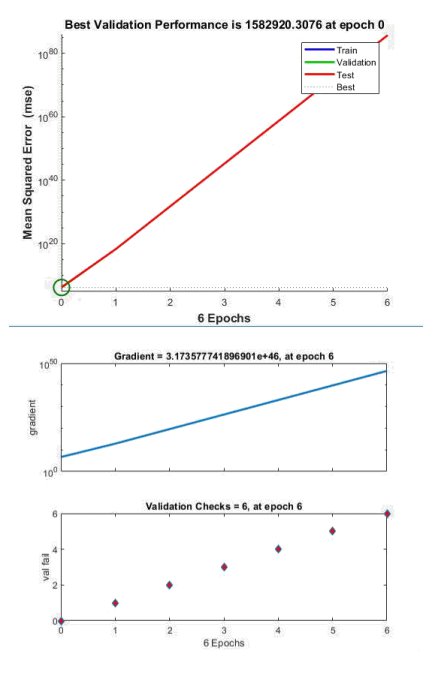
efekty(i) = sim(net, wejscie(i));

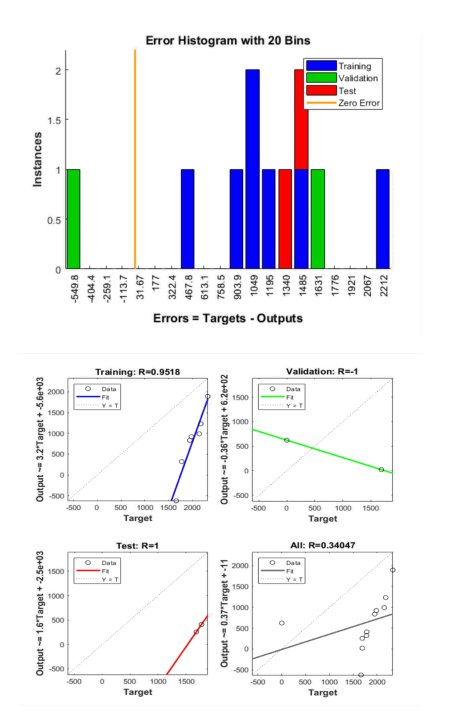
end

Wyniki po uruchomieniu programu:

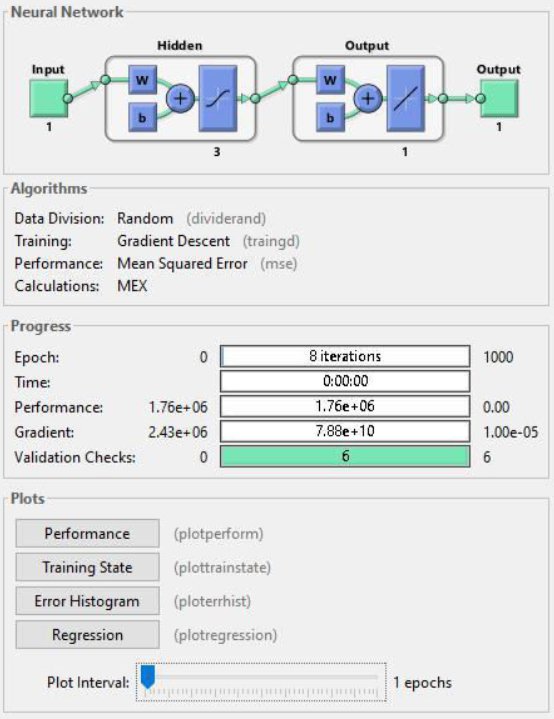
wersja dla parametrów : współczynnik uczenia = 0.5, bezwładność = 0.5

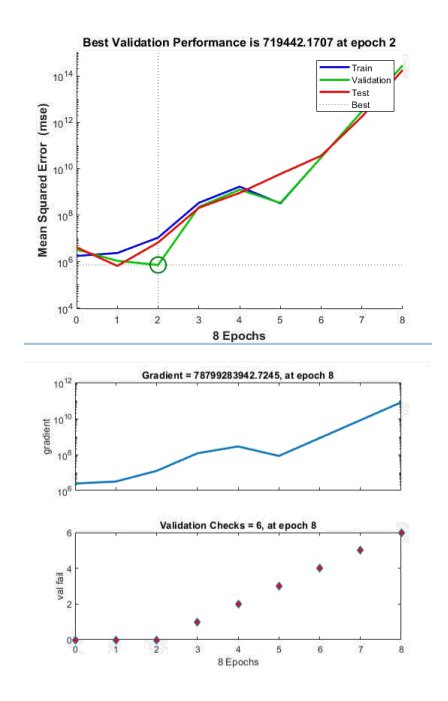


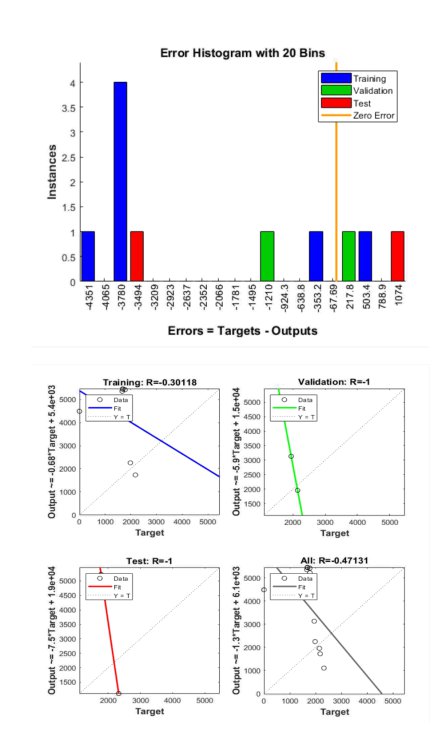




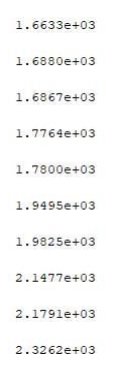
- wersja dla parametrów : współczynnik uczenia = 0.000001, bezwładność = 0.1



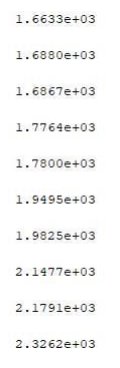




**Wyniki dla 1 przypadku:**



**Wyniki dla drugiego przypadku:**



**Wnioski**

* sieciami wielowarstwowymi, podobnie jak sieciami jednowarstwowymi, można  
  sterować dopasowując poszczególne parametry uczenia, takie jak współczynnik  
  uczenia, czy bezwładność (momentum).
* sieci wielowarstwowe bez algorytmu propagacji błędu działają lepiej niż sieci  
  posiadające ten algorytm.
* najgorsze wyniki osiągają te sieci, których współczynnik uczenia jest równy  
  bezwładności
* im mniejszy współczynnik uczenia tym dokładniejszy wynik. Aby otrzymać wiarygodne dane musimy odpowiednio dobrać współczynniki uczenia a także bezwładności
* wraz ze zwiększaniem wartości współczynnika uczenia zmniejsza się liczba iteracji, co ma wpływ na działanie programu i powoduje uzyskanie błędnych wyników. Aby zredukować liczbę błędnych wyników można zwiększyć liczbę iteracji.
* dobór wag ma bezpośredni wpływ na działanie perceptronów w sieci, w zależności od ich wartości zmienia się poprawność wyników i liczba iteracji. Wartości wag mają największy wpływ na efekt końcowy, często wagi są ustalane losowo.
* dane uczące wpływają na poprawność uczenia perceptronu, przy zbyt małej ilości otrzymujemy błędne wyniki. Aby uzyskać lepsze wyniki powinniśmy dostarczyć wystarczającą liczbę danych wejściowych
* bardzo ważny jest odpowiedni dobór współczynnika uczenia, wag oraz liczby danych uczących, gdyż mają bezpośredni wpływ na działanie sieci.
* zastosowanie sieci wielowarstwowych umożliwia uzyskanie dokładniejszych wyników w krótszym czasie obliczeń.