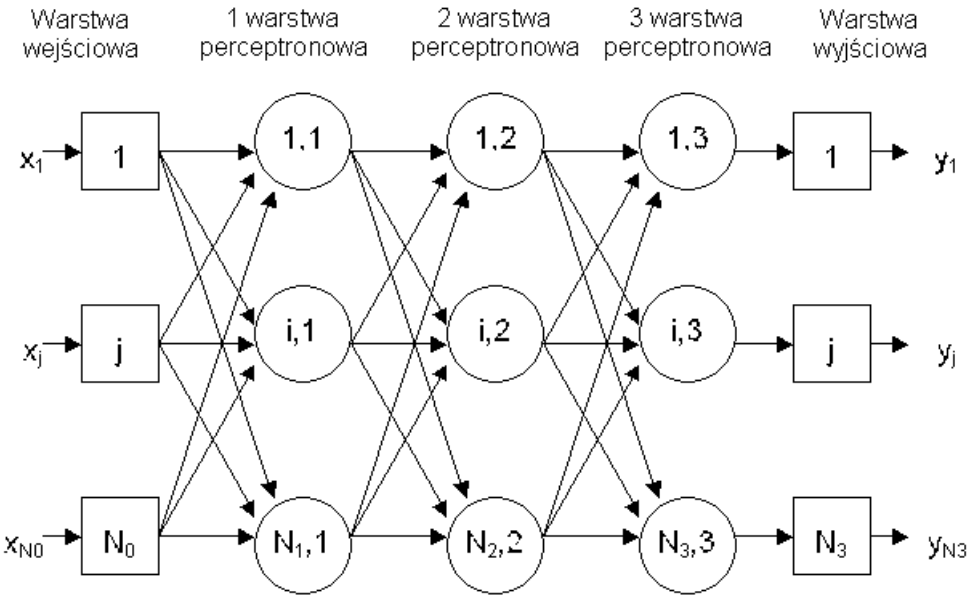
Grupa proj. ***1***	*Przedmiot* ***Podstawy Sztucznej Inteligencji***	*Data wykonania.* ***30.11.2018***
Nr ćwicz. ***3***	*Temat ćwiczenia.* ***Budowa i działanie sieci wielowarstwowej typu feedforward***	
Imię i nazwisko. ***Katarzyna Giądła***		*Ocena i uwagi*

Część teoretyczna

Celem ćwiczenia było zapoznanie się z budową oraz działaniem wielowarstwowych sieci neuronowych poprzez uczenie kształtu funkcji matematycznej z użyciem algorytmu wstecznej propagacji błędu.

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.



Warstwy (minimum jedna), które znajdują się się między warstwą wejściową a warstwą wyjściową nazywamy warstwami ukrytymi.

W sieciach wielowarstwowych nie stosuje się nieliniowych funkcji aktywacji (np.: funkcje sigmoidalne, czy radialne). Często jednak można znaleźć funkcję liniową w warstwie wyjściowej. Najczęściej sieci wielowarstwowe korzystają z uczenia nadzorowanego.

Jedną z zalet sieci neutronowej jest fakt, że nie musimy wyszukiwać osobno dobierać wag. Można tak wytrenować wagi tak, by znaleźć optymalny zestaw za pomocą algorytmu wstecznej propagacji błędu. Działanie tego algorytmu oparte jest na regule delta.

Algorytm uczenia sieci wielowarstwowych metodą propagacji błędu:

1. Podajemy dane wejściowe.
2. Inicjujemy losowo, odpowiednio małe wagi.
3. Dla danego wektora uczącego obliczamy odpowiedź sieci (warstwa po warstwie).
4. Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią *y* oraz poprawną odpowiedzią *t*. Błędy propagowane są do wcześniejszych warstw.

5. Każdy neuron modyfikuje wagi na podstawie wartości błędu i wielkości przetwarzanych w tym kroku sygnałów.
6. Dla kolejnych wektorów uczących powtarzamy operacje od kroku 3. Gdy wszystkie wektory zostaną użyte, losowo zmieniamy ich kolejność i zaczynamy wykorzystywać powtórnie.
7. Jeśli średni błąd na danych treningowych przestanie maleć, przerywamy działanie algorytmu.

Funkcja obliczająca błąd:

$$d(w_1,\ldots,w_K)=\frac{1}{2}\left(f(w_1z_1+\ldots+w_Kz_K)-t\right)^2$$

Część praktyczna

Do wykonania tego zadania wykorzystałam pakiet MATLAB, a szczególnie narzędzie *Neural Networking Training Tool*. Pozwala ono nie tylko na utworzenie prostych sieci jednowarstwowych, ale również tych bardziej złożonych, korzystających z modelu sieci wielowarstwowych. Można również, za pomocą parametrów wykorzystać algorytm propagacji błędów.

Zadanie polegało na wygenerowaniu danych uczących i testujących dla funkcji Rastrign 3D dla danych wejściowych z przedziału *[-2; 2]*. Funkcja Rastrign przyjmowała następującą postać:

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

gdzie: $A = 10$, $x_i \in [-5.12; 5.12]$ oraz dla $f(0) = 0$.

Dla naszych potrzeb obliczeniowych utworzyłam funkcję *RastrignTest3D*. Ze względu na ograniczenia związane z wersją oprogramowania funkcja została utworzona w pliku *RastrignTest3D.m*. Kod źródłowy został umieszczony w listingach kodu.

Przyjęłam jako dane wejściowe tablicę 11-elementową, gdzie zapisałam liczby z przedziału *[-2; 2]* z krokiem 0.4, wraz z końcami przedziału. Jako dane wyjściowe utworzyłam tablicę 11-elementową zawierającą wyniki działania funkcji Rastrign 3D dla danych wejściowych. Do zaimplementowania sieci wielowarstwowej użyłam polecenia *feedforwardnet(2),* które tworzy sieć wielowarstwową zawierającą 2 warstwy ukryte. Można jednak utworzyć sieć o większej ilości warstw. Użyliśmy dwóch, ponieważ przeważnie buduje się sieci o maksymalnie 2 warstwach ukrytych, gdyż istnieje ryzyko przeuczenia sieci. Nasz problem nie jest na tyle złożony, by korzystać z dużej ilości warstw ukrytych.

Następnie postanowiłam przetestować działanie sieci, która uczyła bez i z algorytmem wstecznej propagacji – aby wprowadzić ten algorytm należy dodać parametr treningu *net.trainFcn = 'traingd'*. Działanie sieci było uzależnione również od współczynnika uczenia oraz bezwładności (tzw. *momentum*).

Wszelkie testy i wyniki zostały umieszczone w poniższych tabelach.

Zbiór danych wyjściowych:
wej_out = [1.6633e+03 1.6880e+03 1.6867e+03 1.7764e+03 1.7800e+03 1.8672e+03 1.9495e+03 1.9825e+03 2.1477e+03 2.1791e+03 2.3262e+03];

W poniższej tabeli przedstawiono wyniki zmiennej *efekt*, czyli wyniki działania sieci po podaniu danych wejściowych.

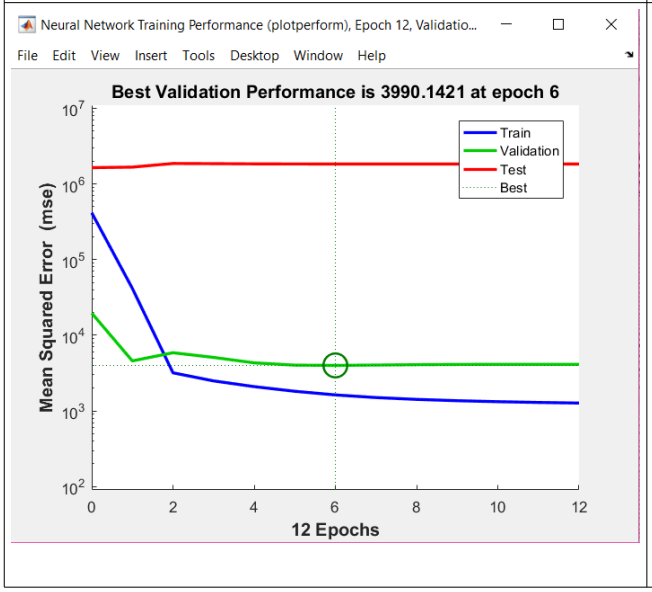
Dane wejściowe	Sieć wielowarstwowa									Sieć wielowarstwowa z alg. Propagacji błędu									Dane testowe
Wsp. uczenia	0,5			0,1			0,01			0,5			0,1			0,01			
Momentum	0	0,5	1	0	0,5	1	0	0,5	1	0	0,5	1	0	0,5	1	0	0,5	1	
-2	1,667E+03	1,664E+03	1,633E+03	1,669E+03	1,661E+03	2,169E+03	1,657E+03	2,479E+03	1,776E+03	2,396E+03	2,461E+03	1,931E+03	1,923E+03	2,287E+03	1,650E+03	1,886E+03	1,898E+03	2,190E+03	1,663E+03
-1,6	1,668E+03	1,682E+03	1,664E+03	1,676E+03	1,679E+03	2,102E+03	1,674E+03	2,637E+03	1,612E+03	2,532E+03	2,374E+03	1,824E+03	1,821E+03	2,341E+03	1,706E+03	1,844E+03	1,845E+03	2,200E+03	1,688E+03
-1,2	1,671E+03	1,708E+03	1,704E+03	1,693E+03	1,706E+03	2,063E+03	1,699E+03	2,731E+03	1,515E+03	2,613E+03	2,322E+03	1,761E+03	1,761E+03	2,373E+03	1,739E+03	1,820E+03	1,814E+03	2,206E+03	1,687E+03
-0,8	1,682E+03	1,747E+03	1,754E+03	1,729E+03	1,744E+03	2,046E+03	1,736E+03	2,770E+03	1,474E+03	2,646E+03	2,300E+03	1,735E+03	1,735E+03	2,387E+03	1,753E+03	1,809E+03	1,801E+03	2,208E+03	1,776E+03
-0,4	1,720E+03	1,799E+03	1,810E+03	1,792E+03	1,797E+03	2,040E+03	1,788E+03	2,783E+03	1,460E+03	2,658E+03	2,292E+03	1,726E+03	1,726E+03	2,391E+03	1,758E+03	1,805E+03	1,797E+03	2,209E+03	1,780E+03
0	1,372E+03	1,876E+03	1,597E+03	1,863E+03	8,730E+02	1,145E+03	1,639E+03	1,718E+03	1,686E+03	2,974E+03	2,189E+03	3,322E+02	2,851E+02	2,423E+03	2,857E+03	1,804E+03	3,590E+03	2,173E+03	1,867E+03
0,4	1,998E+03	1,929E+03	1,933E+03	1,963E+03	1,947E+03	2,033E+03	1,937E+03	2,785E+03	1,452E+03	2,660E+03	2,284E+03	1,726E+03	1,718E+03	2,394E+03	1,763E+03	1,802E+03	1,798E+03	2,208E+03	1,950E+03
0,8	2,142E+03	1,996E+03	2,007E+03	2,033E+03	2,042E+03	2,025E+03	2,026E+03	2,777E+03	1,447E+03	2,654E+03	2,275E+03	1,735E+03	1,707E+03	2,394E+03	1,769E+03	1,797E+03	1,805E+03	2,205E+03	1,983E+03
1,2	2,206E+03	2,068E+03	2,095E+03	2,098E+03	2,144E+03	2,000E+03	2,112E+03	2,752E+03	1,436E+03	2,637E+03	2,248E+03	1,761E+03	1,680E+03	2,395E+03	1,786E+03	1,785E+03	1,824E+03	2,197E+03	2,148E+03
1,6	2,226E+03	2,179E+03	2,191E+03	2,180E+03	2,246E+03	1,942E+03	2,189E+03	2,692E+03	1,410E+03	2,594E+03	2,185E+03	1,825E+03	1,614E+03	2,397E+03	1,825E+03	1,757E+03	1,869E+03	2,177E+03	2,179E+03
2	2,233E+03	2,326E+03	2,274E+03	2,295E+03	2,341E+03	1,844E+03	2,250E+03	2,591E+03	1,365E+03	2,522E+03	2,079E+03	1,932E+03	1,502E+03	2,401E+03	1,892E+03	1,708E+03	1,947E+03	2,144E+03	2,326E+03
Epoki	14	13	8	8	14	4	19	7	8	6	6	6	6	6	6	6	6	6	

W poniższej tabeli umieszczono rozkład odchylenia standardowego dla poszczególnych wyników zmiennej efekt. Kolorem zielonym oznaczono relatywnie małe błędy - przyjęłam wartość 20, kolorem żółtym odchylenie w okolicach 60, a czerwonym błędy o odchyleniu powyżej 100 (błędy grube).

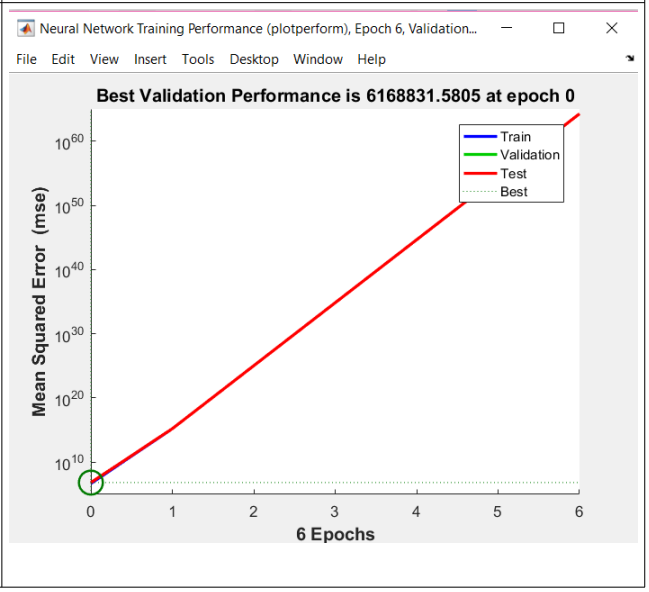
Dane wejściowe	Sieć wielowarstwowa									Sieć wielowarstwowa z alg. Propagacji błędu								
Wsp. uczenia	0,5			0,1			0,01			0,5			0,1			0,01		
Momentum	0	0,5	1	0	0,5	1	0	0,5	1	0	0,5	1	0	0,5	1	0	0,5	1
-2	2,39E+00	4,03E-01	2,14E+01	3,85E+00	1,62E+00	3,58E+02	4,62E+00	5,76E+02	7,98E+01	5,18E+02	5,64E+02	1,90E+02	1,84E+02	4,41E+02	9,36E+00	1,57E+02	1,66E+02	3,72E+02
-1,6	1,45E+01	4,59E+00	1,72E+01	8,27E+00	6,69E+00	2,93E+02	1,01E+01	6,71E+02	5,38E+01	5,97E+02	4,85E+02	9,64E+01	9,40E+01	4,62E+02	1,28E+01	1,10E+02	1,11E+02	3,62E+02
-1,2	1,13E+01	1,53E+01	1,25E+01	4,72E+00	1,33E+01	2,66E+02	8,84E+00	7,38E+02	1,21E+02	6,55E+02	4,49E+02	5,28E+01	5,23E+01	4,85E+02	3,72E+01	9,40E+01	9,02E+01	3,67E+02
-0,8	6,67E+01	2,06E+01	1,58E+01	3,35E+01	2,27E+01	1,91E+02	2,83E+01	7,02E+02	2,14E+02	6,15E+02	3,70E+02	2,92E+01	2,91E+01	4,31E+02	1,64E+01	2,32E+01	1,77E+01	3,05E+02
-0,4	4,23E+01	1,35E+01	2,12E+01	8,57E+00	1,18E+01	1,84E+02	5,76E+00	7,09E+02	2,26E+02	6,21E+02	3,62E+02	3,82E+01	3,82E+01	4,32E+02	1,53E+01	1,80E+01	1,20E+01	3,03E+02
0	3,51E+02	6,39E+00	1,91E+02	2,90E+00	7,03E+02	5,11E+02	1,61E+02	1,05E+02	1,28E+02	7,83E+02	2,27E+02	1,09E+03	1,12E+03	3,93E+02	7,00E+02	4,49E+01	1,22E+03	2,16E+02
0,4	3,45E+01	1,44E+01	1,17E+01	9,86E+00	1,91E+00	5,94E+01	8,79E+00	5,91E+02	3,52E+02	5,03E+02	2,37E+02	1,58E+02	1,64E+02	3,14E+02	1,32E+02	1,04E+02	1,07E+02	1,83E+02
0,8	1,13E+02	9,56E+00	1,70E+01	3,61E+01	4,20E+01	2,97E+01	3,05E+01	5,62E+02	3,79E+02	4,75E+02	2,07E+02	1,75E+02	1,95E+02	2,91E+02	1,51E+02	1,31E+02	1,26E+02	1,57E+02
1,2	4,09E+01	5,60E+01	3,71E+01	3,50E+01	2,59E+00	1,04E+02	2,49E+01	4,27E+02	5,03E+02	3,46E+02	7,12E+01	2,73E+02	3,31E+02	1,75E+02	2,56E+02	2,56E+02	2,29E+02	3,46E+01
1,6	3,32E+01	2,09E-01	8,71E+00	2,85E-01	4,75E+01	1,68E+02	6,96E+00	3,63E+02	5,44E+02	2,93E+02	4,42E+00	2,51E+02	4,00E+02	1,54E+02	2,50E+02	2,99E+02	2,19E+02	1,42E+00
2	6,61E+01	3,18E-01	3,68E+01	2,19E+01	1,07E+01	3,41E+02	5,36E+01	1,87E+02	6,80E+02	1,38E+02	1,75E+02	2,79E+02	5,83E+02	5,26E+01	3,07E+02	4,37E+02	2,68E+02	1,29E+02

Poniżej zamieszczone są zrzuty wykresów dla wariantów najgorzej i najlepiej działających.

Sieć wielowarstwowa o wsp. uczenia i momentum 0.5

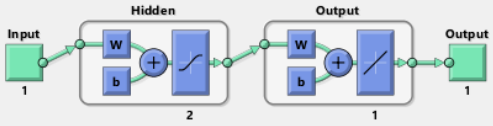


Sieć wielowarstwowa z alg. Propagacji błędu o wsp. uczenia 0.1 i momentum 0.5



Neural Network Training (nntraintool)

Neural Network



Algorithms

Data Division:	Random (dividerand)
Training:	Levenberg-Marquardt (trainlm)
Performance:	Mean Squared Error (mse)
Calculations:	MEX

Progress

Epoch:	0	12 iterations	1000
Time:		0:00:01	
Performance:	4.16e+05	1.27e+03	0.00
Gradient:	1.83e+06	4.93e+03	1.00e-07
Mu:	0.500	500	1.00e+10
Validation Checks:	0	6	6

Plots

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

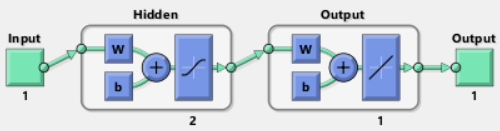
Plot Interval: 1 epochs

Opening Performance Plot

Stop Training Cancel

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:		0:00:01	
Performance:	4.50e+06	4.50e+06	0.00
Gradient:	7.45e+06	5.63e+35	1.00e-05
Validation Checks:	0	6	6

Plots

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

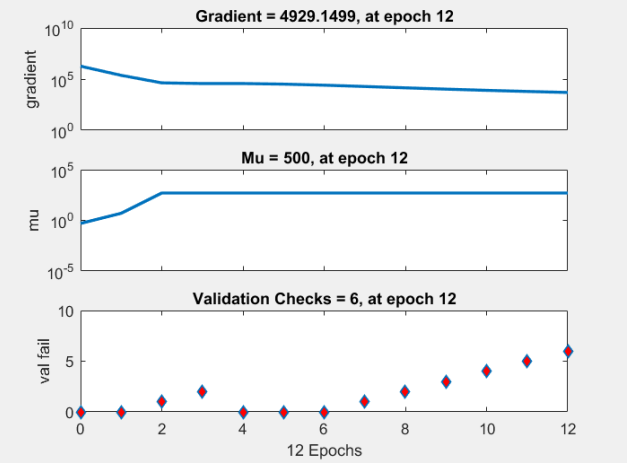
Plot Interval: 1 epochs

Validation stop.

Stop Training Cancel

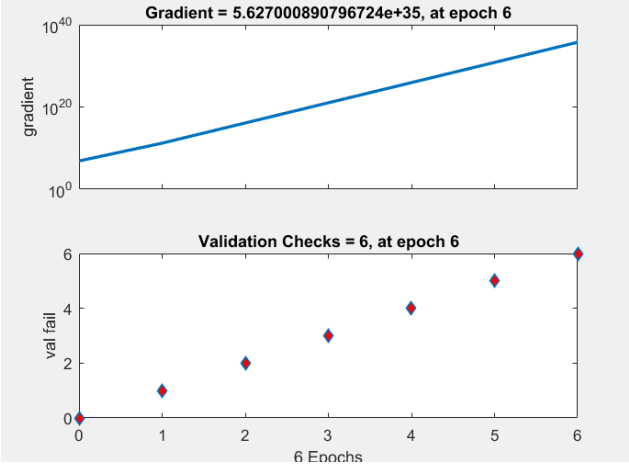
Neural Network Training Training State (plottrainstate), Epoch 12, Valida...

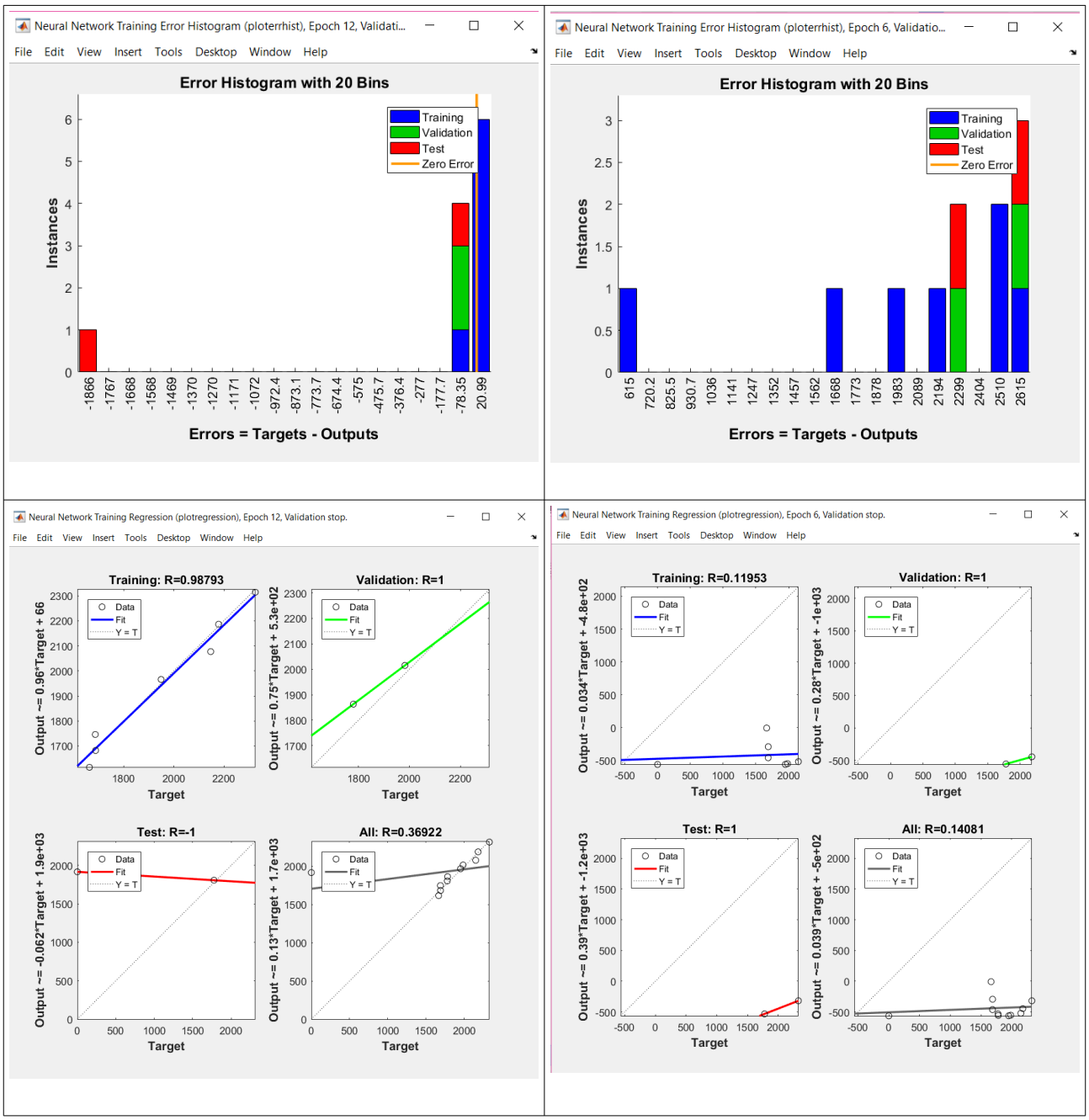
File Edit View Insert Tools Desktop Window Help



Neural Network Training Training State (plottrainstate), Epoch 6, Validati...

File Edit View Insert Tools Desktop Window Help





Wnioski

- Sieci wielowarstwowe nie są pozbawione błędów, podobnie w stosunku do swoich prostszych, jednowarstwowych odpowiedników.
- Podobnie jak sieciami jednowarstwowymi, możemy sterować dopasowując poszczególne parametry uczenia, takie jak współczynnik uczenia, czy bezwładność (*momentum*). Współczynniki te mają niebagatelny wpływ na wyniki działania sieci.
- Sieć wielowarstwowa bez algorytmu propagacji błędu działają lepiej niż sieci posiadające ten algorytm mimo, że błędy, które na wejściu i w trakcie działania wydają się być pomijalne mogą na końcu spowodować duży rozrzut między tym, co zostało zasymulowane a tym, co powinna dać sieć w swoim wyniku.
- Jeżeli nie korzystamy z algorytmu propagacji danych należy pamiętać, by były duże różnice między wsp. uczenia a bezwładnością. Najgorsze wyniki osiągają te sieci, których wsp.

uczenia jest równy bezwładności. Tylko wówczas nasze wyniki osiągają akceptowalne wartości.

- Sieci wielowarstwowe posiadające algorytm propagacji danych wykazują praktycznie duże odsetki błędów. Nie należy jednak nie zwracać uwagi na wymienione wcześniej parametry. W tym przypadku im większy jest wsp. uczenia od bezwładności, tym częściej mogą zdarzać się błędy.
- Algorytmy z propagacją danych przyspieszają proces uczenia nawet o połowę epok w stosunku do sieci niezawierających tego algorytmu.
- Gradienty uczenia przy sieciach z algorytmem propagacji błędów posiadają charakterystykę niemalże liniową, dzięki czemu można łatwiej przewidzieć, jakich współczynników do nauki sieci najlepiej użyć, by otrzymane rezultaty były zadowalające.
- Najtrudniejszym zadaniem dla sieci było przyporządkowanie argumentowi 0 wartości 0. Niestety żadna z sieci nie podołała temu zadaniu – sieci tylko modelują pewne dane na bazie pewnych danych uczący, a nie tworzą idealnych obrazów między danymi wejściowymi a danymi wyjściowymi.

Załączniki – listingi kodów i działanie kodów

Funkcja pomocnicza – RastrignTest3D.m

```
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
```

Funkcja ucząca i testująca dane – P3_Katarzyna_Giadla.m

```
close all; clear all; clc;

wej_in = [-2 -1.6 -1.2 -0.8 -0.4 0 0.4 0.8 1.2 1.6 2];
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];

testowe = zeros(1);

net = feedforwardnet(2); %tworzenie sieci z 2 warstwami ukrytymi
net.trainFcn = 'traingd'; %algorytm wstecznej propagacji
net.trainParam.lr = 0.1; %wsp. uczenia
net.trainParam.mc = 0.5; %bezwladnosc
net = train(net, wej_in, wej_out);
efekty = zeros(size(net));

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