Marcin Baszak

IS WIMiIP rok 3 grupa 1

**SPRAWOZDANIE**

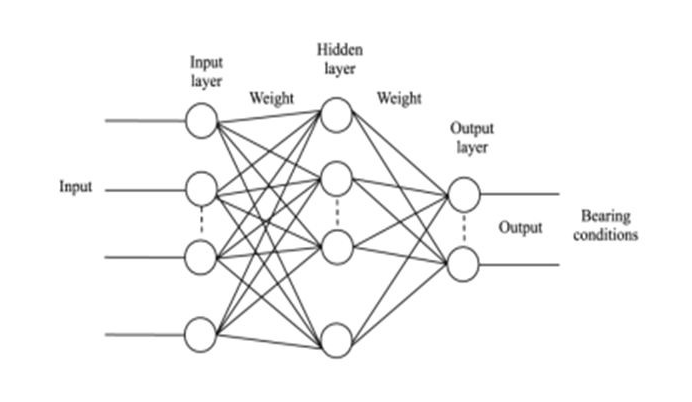
**PSI Scenariusz 3**

**Cel ćwiczenia:**

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

**Opis syntetyczny:**

**Algorytm wstecznej propagacji** - podstawowy 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. Dzięki zastosowaniu specyficznego sposobu propagowania błędów uczenia sieci powstałych na jej wyjściu, tj. przesyłania ich od warstwy wyjściowej do wejściowej, algorytm propagacji wstecznej stał się jednym z najskuteczniejszych algorytmów uczenia sieci.



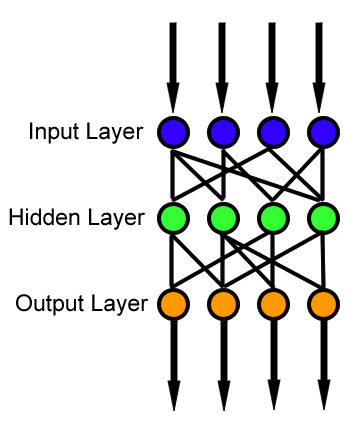
**Sieć neuronowa typu feedforward** - Każdy neutron składa się z elementu sumującego iloczyny wag i sygnałów wejściowych oraz z elementu nieliniowego, którego charakterystyka przejściowa jest funkcją aktywacji neuronu. Proces uczenia przebiega iteracyjnie dla kolejnych danych uczących, przy czym w każdym kroku współczynniki wagowe połączeń ulegają modyfikacji. Propagacja sygnałów następuje przez

warstwę wejściową, warstwę ukrytą i ostatecznie wyjściową. W kolejnym kroku porównujemy

wartość oczekiwaną z wartością sygnału wyjściowego. Różnica obu wartości to sygnał błędu

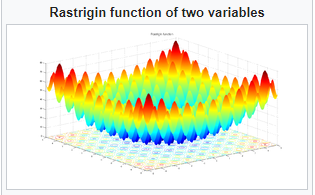
warstwy wyjściowej. Do neuronów leżących w wewnętrznych warstwach sieci (ukrytych) nie ma

możliwości bezpośredniego określenia błędu, ponieważ nie są znane wartości oczekiwane sygnałów wyjściowych z tych neuronów. Korzysta się więc z wyznaczonego wcześniej sygnału błędu rzutując go wstecz do wszystkich tych neuronów, których sygnały stanowiły wejścia dla rozważanego neuronu. Mając określony błąd sygnału przystępuje się do modyfikacji wag jego połączeń wejściowych.



**Funkcja Rastrigin 3D:**

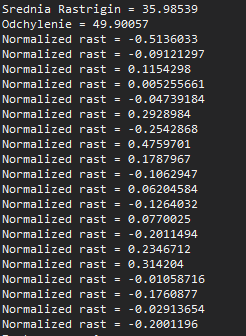
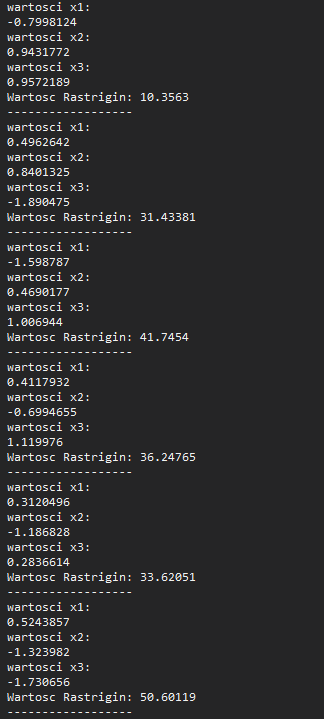
Typowy przykład nieliniowej multimodalnej funkcji. Z początku była to funkcja dwuwymiarowa, jednak z biegiem czasu stała się ona wielowymiarową. Głównym problemem tej funkcji jest znalezienie jej minimum ze względu na duży obszar oraz istniejące w dużej ilości minimum lokalne.

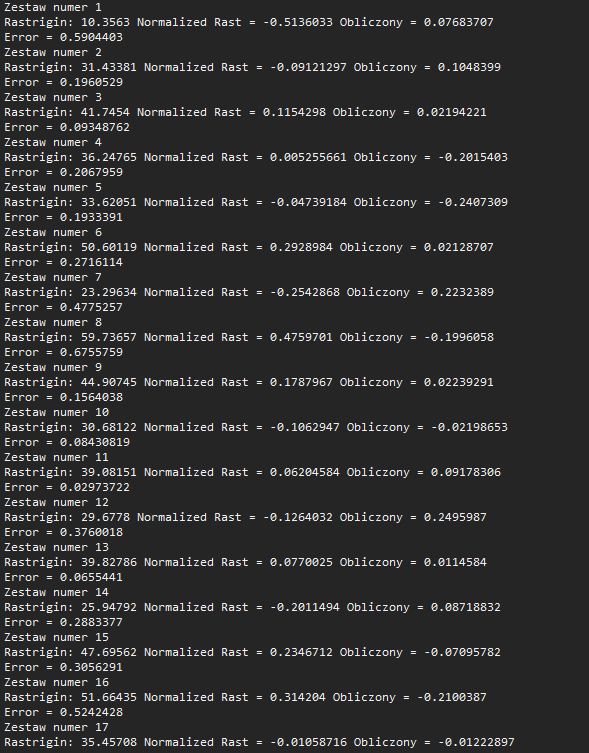


**Zestawienie wyników:**

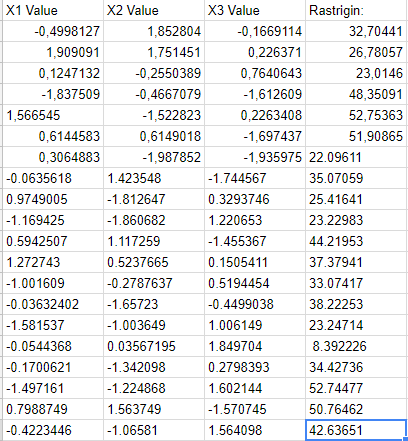
Poniżej screeny przedstawiające przykładowe wyniki działania programu:

Liczba iteracji 1000, współczynnik nauczania 0.5.



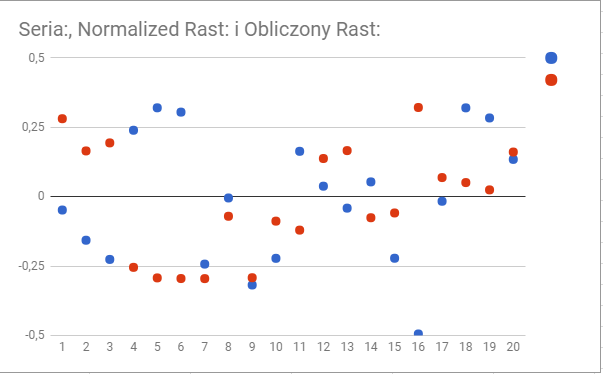
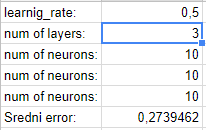


Tabelka zawierająca wygenerowane dane wejściowe oraz obliczona wartość funkcji Rastrigin:

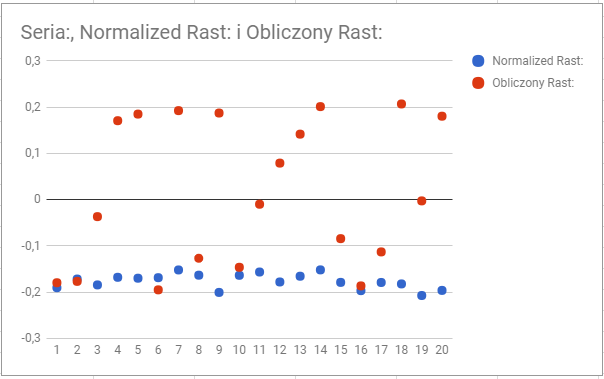
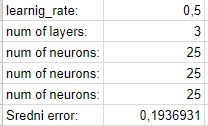


**Analiza i dyskusja błędów:**

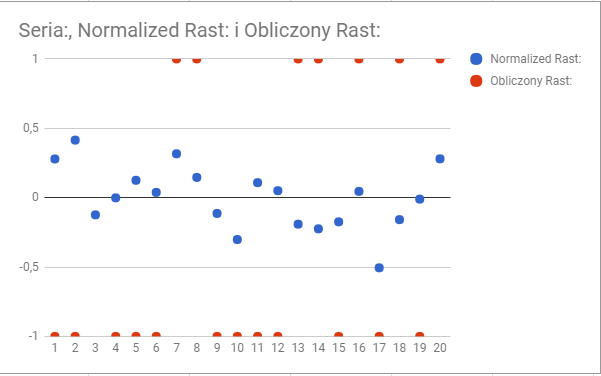
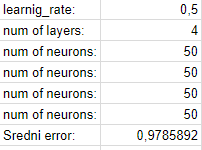
Jako liczbę iteracji w fazie testów przyjąłem 1000. Manipulowane były: ilość warstw, liczba neuronów w warstwie oraz współczynnik nauczania.



Dla współczynnika nauczania 0.5, 3 warstw oraz liczbie neuronów odpowiednio 10, 10 i 10 sieć nauczyła się dostatecznie dobrze, dla niektórych zestawów danych testujących oraz uczących wynik jest bardzo zbliżony, a średni błąd ogółem wyniósł 0,27.

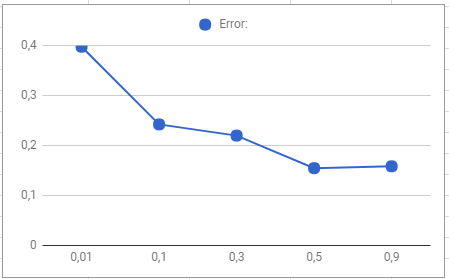
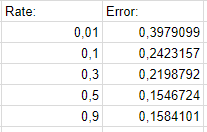


Po zwiększeniu liczby neuronów w każdej z warstw średni błąd zmalał, jednak nadal był widoczny. Więcej wyników danych testujących i uczących jest do siebie zbliżonych, a błąd średni wyniósł 0,19.



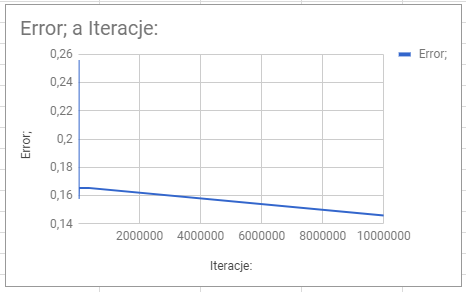
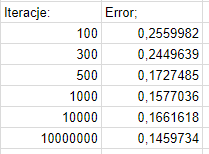
W kolejnym przypadku użyłem bardzo dużej ilości neuronów w każdej warstwie, których zarazem było 4. Okazało się, że sieć się “przeuczyła” i tym razem wyniki były bardzo odległe od poprawnych. Średni error wyniósł prawie 1, co jest już wyraźną anomalią.

W kolejnej fazie testów sprawdziłem wpływ zmiany współczynnika nauczania na błędy.



Zgodnie z oczekaniami, wzrost współczynnika nauczania wpłynął korzystnie na poprawność nauki. Im większy współczynnik, tym błędy mniejsze.

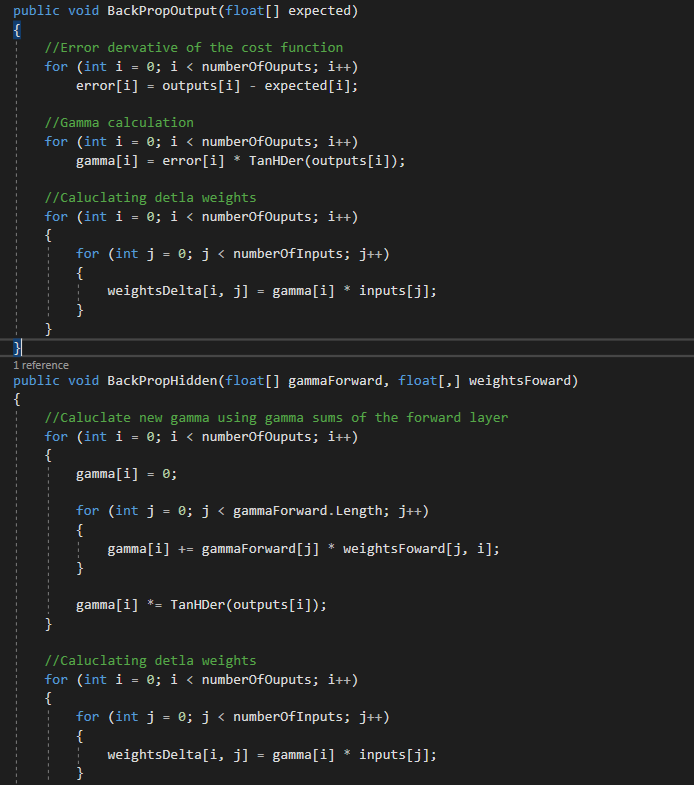
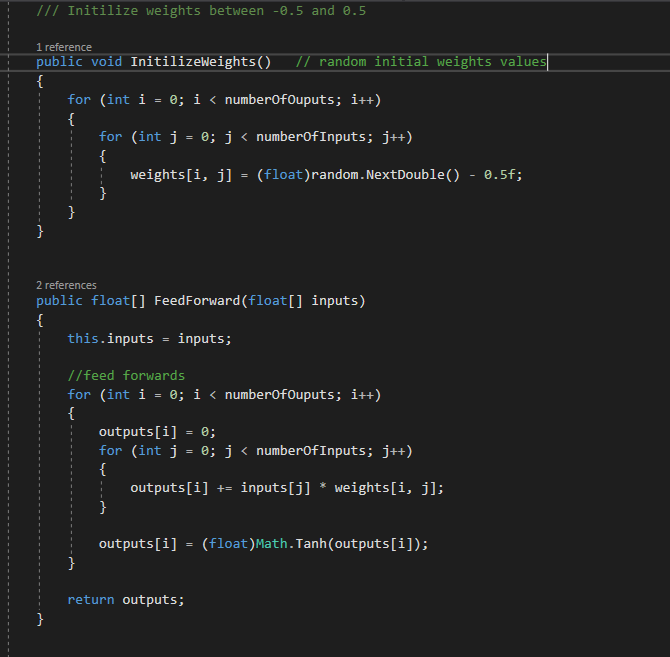
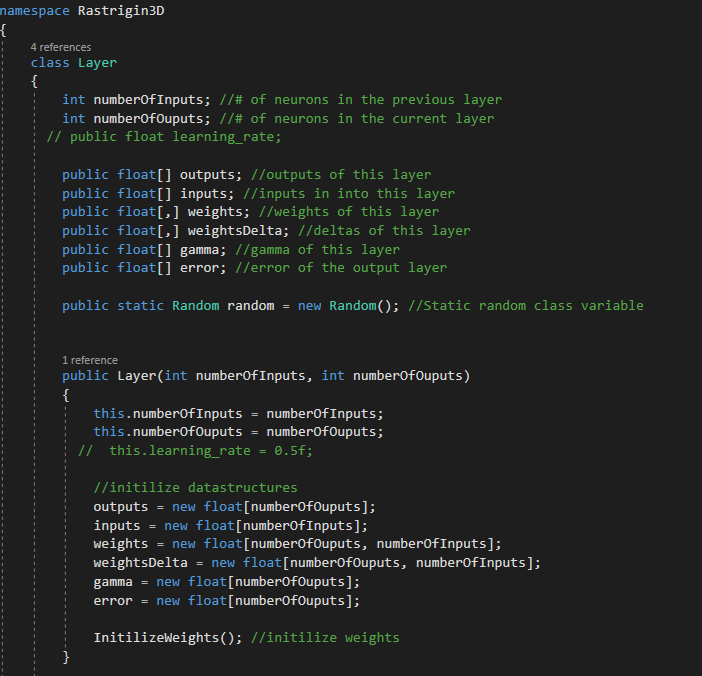
Następnie przetestowałem wpływ ilości iteracji na błędy uczenia.



Zwiększenie liczby iteracji zmniejszało błędy, jednak w dość mało znaczącym stopniu, jednak przy zbyt małej ilości sieć mogłaby się w ogóle nie nauczyć i działać źle. Należy jednak zauważyć, że zwiększenie iteracji zdecydowanie wydłuża działanie programu i przy liczbach większych od 1 000 000, program działa już bardzo długo.

**Wnioski:**

Algorytm wstecznej propagacji spełnił oczekiwania i zdołał nauczyć się bardzo skomplikowanej funkcji jaką jest Rastrigin, co jest na pewno imponującym osiągnięciem. Tak jak w przypadku poprzednich programów (Adaline oraz Perceptron) wraz z liczbą iteracji oraz wzrostem współczynnika nauczania rośnie efektywność programu i błędy są mniejsze. Ciekawszym aspektem jest zmiana liczby warstw oraz neuronów. Można zauważyć, że zwiększenie ich liczby ma początkowo pozytywny wpływ na efektywność, jednak po przekroczeniu pewnej granicy program “przeucza” się i jego efektywność bardzo wyraźnie maleje, a błędy są zdecydowanie większe.

****