Anna Budzoń

Sprawozdanie do scenariusza 3.

23.11.17r.

1. Cel ćwiczenia:

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

1. Opis syntetyczny:
2. Sieć neuronowa wielowarstwowa typu Feedforward:

* Budowa sieci perceptronowej:



* Opis sieci:

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

Typ ***feedforward*** oznacza, że istnieje ściśle określony kierunek przepływu danych – od wejścia, przez warstwy ukryte, do wyjścia sieci neuronowej. Każda warstwa 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.

W naszym przypadku przyjmujemy 2 neurony warstwy wejściowej (x1 i x2) oraz jedno wyjście (y jako wartość funkcji rastrigin). Funkcją aktywacji jest tangens hiperboliczny.

1. Algorytm wstecznej propagacji błędu:

* Metoda uczenia z nauczycielem.
* Uczenie:

Pierwszą czynnością w procesie uczenia jest przygotowanie dwóch ciągów danych: ***uczącego*** i ***weryfikującego***. ***Ciąg uczący*** jest to zbiór takich danych, które w miarę dokładnie charakteryzują dany problem. Jednorazowa porcja danych nazywana jest ***wektorem uczącym***. W jego skład wchodzi ***wektor wejściowy*** czyli te dane wejściowe, które podawane są na wejścia sieci i ***wektor wyjściowy*** czyli takie dane oczekiwane, jakie sieć powinna wygenerować na swoich wyjściach.

W naszym przypadku wektorem uczącym jest zestaw dwóch wartości wejściowych x1 i x2, wylosowanych z przedziału <-2,2> , a wektorem wyjściowym jest wartość oczekiwana, czyli wynik funkcji Rastrigin.

**Wagi początkowe**, z którymi sieć rozpoczyna naukę z reguły stanowią liczby wygenerowane przypadkowo.

Wykorzystanie mechanizmów dających nauczycielowi możliwość regulacji szybkości i jakości uczenia, czyli ***współczynnika*** ***uczenia*** i ***momentum***. Wpływają one na stromość funkcji aktywacji i regulują szybkość wpływu zmiany wag na proces uczenia.

Po przetworzeniu wektora wejściowego, nauczyciel porównuje wartości otrzymane z wartościami oczekiwanymi i informuje sieć czy odpowiedź jest poprawna, a jeżeli nie, to jaki powstał ***błąd średniokwadratowy*** odpowiedzi. Błąd ten jest następnie propagowany do sieci ale **w odwrotnej niż wektor wejściowy kolejności** (od warstwy wyjściowej do wejściowej) i na jego podstawie następuje taka korekcja wag w każdym neuronie, aby ponowne przetworzenie tego samego wektora wejściowego spowodowało zmniejszenie błędu odpowiedzi.

Procedurę taką powtarza się do momentu wygenerowania przez sieć błędu mniejszego niż założony. Wtedy na wejście sieci podaje się kolejny wektor wejściowy i powtarza te czynności. Po przetworzeniu całego ***ciągu uczącego*** (proces ten nazywany jest ***epoką***) oblicza się błąd dla ***epoki*** i cały cykl powtarzany jest do momentu, aż błąd ten spadnie poniżej dopuszczalnego.

Dla naszego przypadku przeprowadzane zostało 20 epok uczenia dla 20 losowych wektorów wejściowych.

* **Wzór uaktualniania wag:**

Wi – wagi

Delta – uaktualniana wartość dla każdego neuronu na podstawie błędu kolejnej warstwy oraz wag

f’ – pochodna funkcji aktywacji

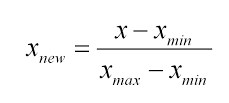
* Testowanie:

Jeżeli mamy już nauczoną sieć, musimy zweryfikować jej działanie. W tym momencie ważne jest podanie na wejście sieci wzorców z poza zbioru treningowego w celu zbadania czy sieć może efektywnie generalizować zadanie, którego się nauczyła. Do tego używamy ***ciągu weryfikującego***, który ma te same cechy co ***ciąg uczący*** tzn. dane dokładnie charakteryzują problem i znamy dokładne odpowiedzi. Ważne jest jednak, aby dane te nie były używane uprzednio do uczenia. Dokonujemy zatem prezentacji ciągu weryfikującego z tą różnicą, że w tym procesie nie rzutujemy błędów wstecz a jedynie rejestrujemy ilość odpowiedzi poprawnych i na tej podstawie orzekamy, czy sieć spełnia nasze wymagania czyli jak została nauczona.

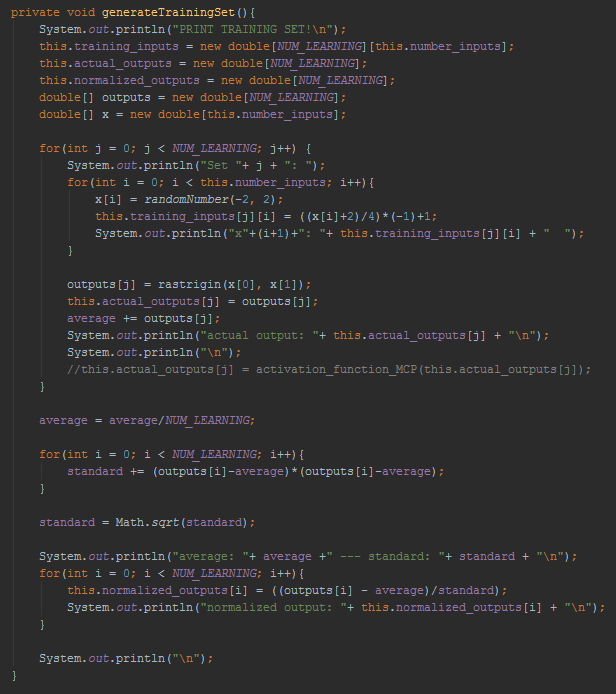
1. **Funkcja Rastrigin:**



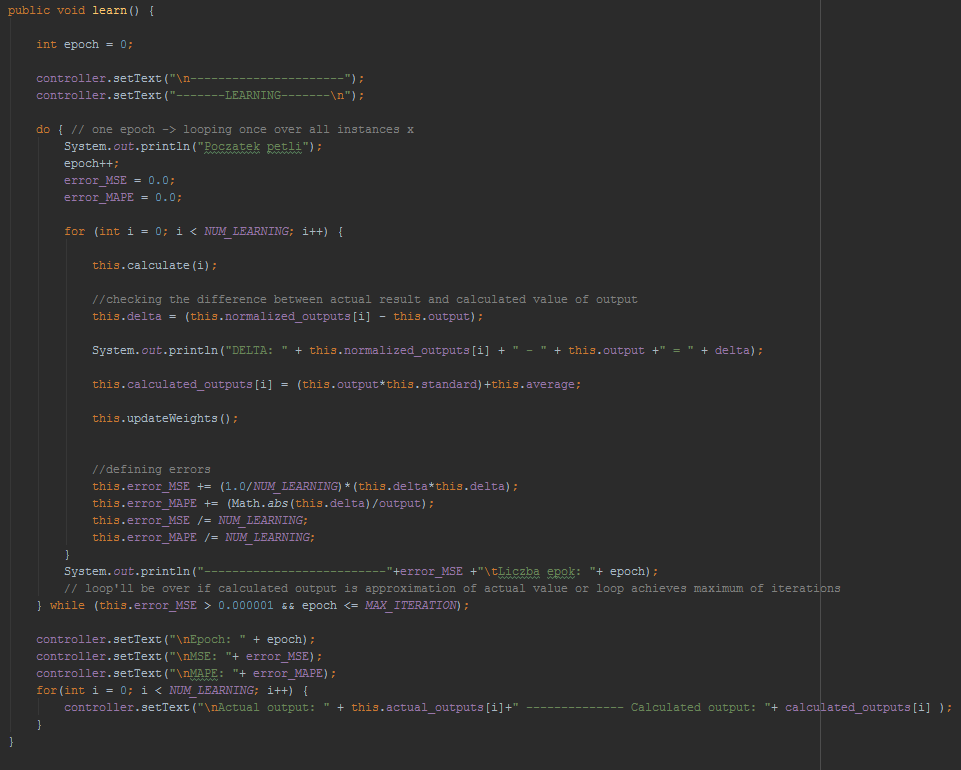
1. Normalizacja danych, by znajdowany się w przedziale <-1,1>:



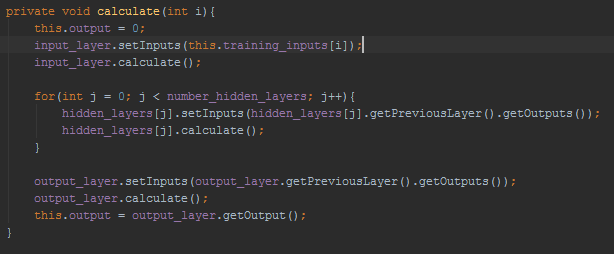
1. Listing kodu:
2. Generowanie danych uczących:



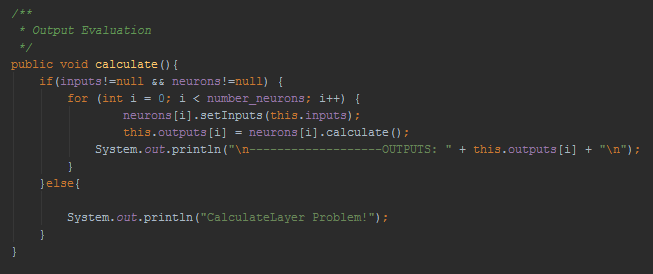
1. Proces uczenia się:



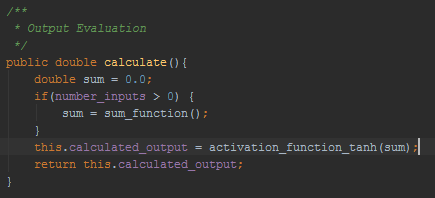
1. Przeliczanie wyniku w pojedynczej epoce:



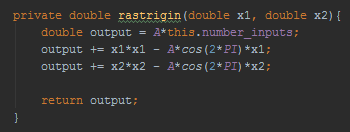
- W warstwie:



- W neuronie:

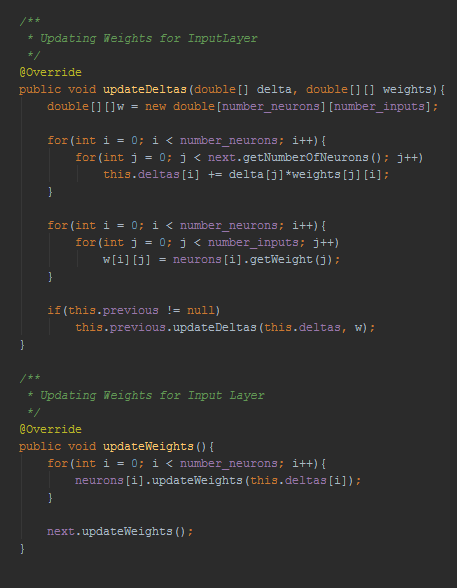


1. Funkcja Rastrigin:

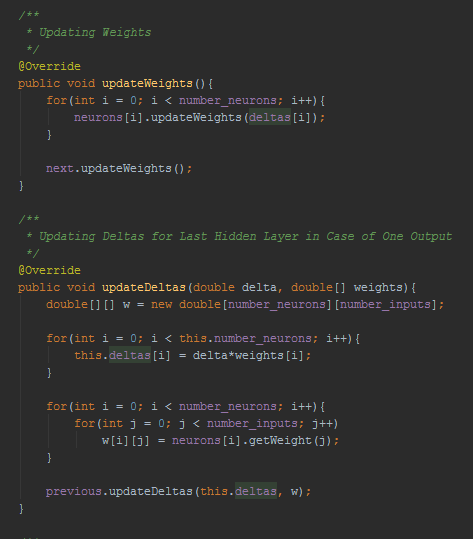


1. Uaktualnianie wag i delt:

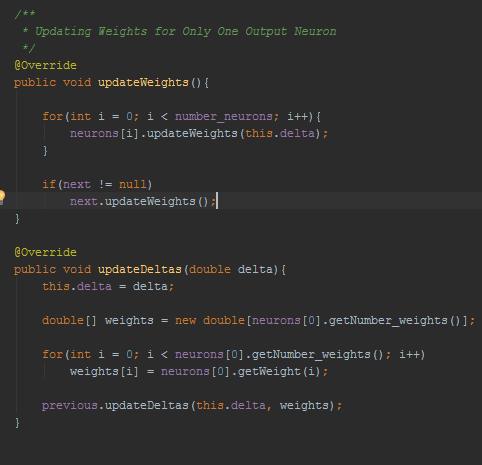
- w warstwie wejściowej:



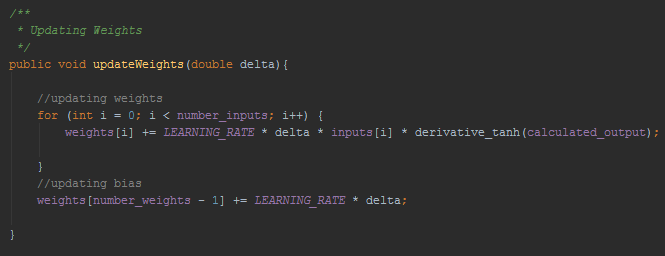
- w warstwach ukrytych (inaczej dla ostatniej warstwy ukrytej)



- w warstwie wyjściowej:



- w neuronie:



1. Wyniki:

EXCEL

1. Analiza wykresów:

Uczenie przeprowadzono dla 3 różnych współczynników uczenia: 0.001, 0.01 i 0,5 oraz przy 3 różnych monumentach: 0.7, 0.8, 0.9. Ponadto wykorzystano trzy sieci: 3-warstwową (1 warstwa ukryta), 5-warstwową (3 warstwy ukryte) i 10-warstwową (8 warstw ukrytych).

1. Przeprowadzone testy sieci odbyły się na małym zakresie współczynnika uczenia, ale mimo to możemy zaobserwować znaczący jego wpływ na epoki uczenia. Przy bardzo małym learning rate (0.001), sieci neuronowe uczyły się bardzo długo -1800, 4500, 2250 iteracji. Dla 3 sieci przy mniejszym monumentum (0.5), a dużym współczynniku uczenia (0.5) ilość epok znacząco się zwiększyła. Nie jesteśmy jednak w stanie stwierdzić czy to monumentum wywołało ten przyrost czy nastąpiło to z powodu innych czynników. To pokazuje, że należałoby wykonać więcej testów dla większego zakresu wartości współczynników uczenia oraz osobno dla różnych monumentów, by zaobserwować dokładniejszy ich wpływ na otrzymane wyniki.
2. Ilość warstw oraz neuronów nie jest wprost proporcjonalna ani odwrotnie proporcjonalna do ilości epok. Jednak możemy zauważyć, że również mogą te wartości mieć wpływ na szybkość uczenia się sieci, ponieważ zaobserwowaliśmy odmienną ilość epok dla tych samych współczynników uczenia oraz monumentum. Najwolniej uczyła się sieć II przy małym współczynniku uczenia się – aż 4500 epok. Najszybciej nauczona została sieć I o dużym współczynniku uczenia.
3. Z wykresów przedstawiających porównanie między wartością oczekiwaną, a otrzymaną, możemy zaobserwować, że sieć popełniała dosyć widocznie błędy, gdyż połączenie otrzymanych punktów powinno dać nam funkcję y=x. Najbardziej odbiegającymi od wzorca wykresami są konfiguracje II i VIII.
4. Przy VIII konfiguracji, tj. sieci III z 10 warstwami, możemy również zaobserwować nietypowy kształt funkcji błędu od ilości iteracji. Możemy podejrzewać, że przy tym procesie uczenia sieci neuronowa nie została prawidłowo nauczona. Kolejnym argumentem jest również to, że największy całkowity błąd wystąpił właśnie przy tej konfiguracji i wyniósł 0.003981139031447607
5. Najmniejszy całkowity błąd uczenia otrzymaliśmy przy uczeniu sieci I dla współczynnika uczenia równego 0.5 i monumentum 0.5 – error: 7.28156867570213E-4. Wykres w tym przypadku jest prawie symetryczny, więc możemy stwierdzić, że sieć nauczyła się.
6. Wnioski
7. Porównanie otrzymanych wyników z sieci neuronowej z aktualnymi wartościami funkcji:

Charakter liniowy wykresów świadczy o tym, że uczenie sieci neuronowej przebiegło pomyślnie – oba wyniki są do siebie zbliżone. Jednak poszczególne wyniki są obarczone dość znaczącym błędem, co prawdopodobnie mogło wyniknąć z źle znormalizowanych danych lub nieodpowiednio dobranych współczynników uczenia i monumentów.

1. Błąd otrzymanych wartości mógł również wyniknąć z odwrócenia procesu normalizacji wyliczonych danych wyjściowych, podczas którego otrzymany błąd został zwielokrotniony.
2. Ilość warstw i neuronów ściśle wpływa na szybkość uczenia się sieci wielowarstwowej – sieć uczy się szybciej przy większej ilości neuronów.