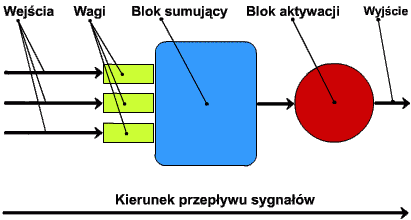
Sprawozdanie

Małgorzata Bień IS gr.1

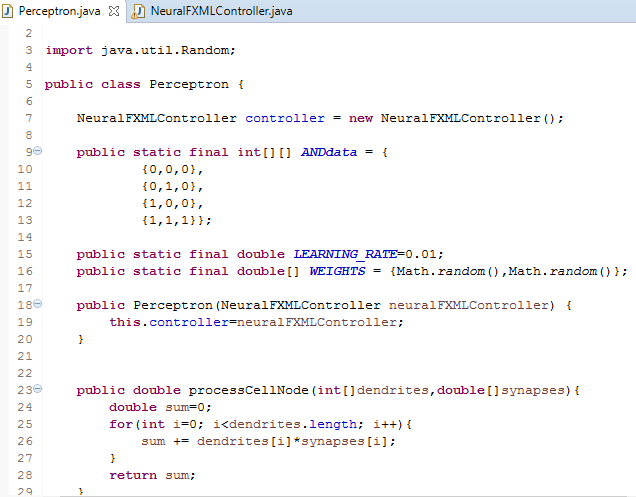
Temat ćwiczenia: Budowa i działanie perceptronu

W moim programie zaimplementowałam sztuczny neuron według modelu McCullocha-Pittsa.

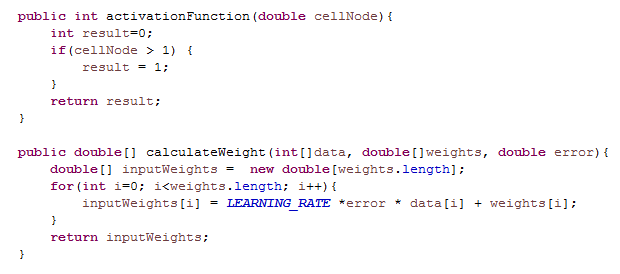


W dendrytach(wejścia) umieszcza się dane wejściowe, które zostają przemnożone przez wartości wag w odpowiadających im synapsach. Dla każdego powiązania dendryt<→synapsa zachodzi: R=d\*w, gdzie R to wynik połączenia, d informacja w dendrycie a w to wartość wagi. Wyniki wszystkich połączeń trafiają do bloku sumującego. Rezultat tej operacji przechodzi przez blok aktywacyjny . W tym miejscu zostaje poddany funkcji aktywacyjnej, która może mieć charakter liniowy lub przyjmować postać bipolarnej funkcji skoku jednostkowego. Na wyjście zostaje oddana wartość zmodyfikowana w bloku aktywacyjnym.

Aby przeprowadzić uczenie potrzebny jest nam zestaw wzorców. Czyli pary danych wejściowych i pożądanych odpowiedzi od neuronu. Dla każdego wzorca liczymy błąd, który z każdą iteracją będzie coraz mniejszy, jest on równy różnicy oczekiwanej wartości i wartości która pojawiła się na wyjściu sieci. Kiedy mamy policzony błąd neuronu dla każdego wzorca wtedy trzeba nanieść na synapsy(wagi) odpowiednie poprawki. Aby to zrobić musimy wcześniej ustalić jeszcze jedną składową procesu uczenia – współczynnik uczenia z zakresu (0-1). Odpowiada on za prędkość uczenia sieci. Dokonujemy poprawek w synapsach neuronu: do wag na poszczególnych synapsach neuronu dodajemy iloczyn współczynnika uczenia, błędu neuronu i wartości wejściowej na dendrycie.

Opis kodu programu:

W klasie Perceptron na początku ustalam w tablicy ANDdata zestaw wzorców, wejścia i odpowiedzi funkcji logicznej AND. Ustawiam współczynnik uczenia w zmiennej LEARNING\_RATE oraz umieszczam losowe wartości wag w synapsach neuronu. W funkcji processCellNode() obliczam sumę połączenia dendryt-synapsa.



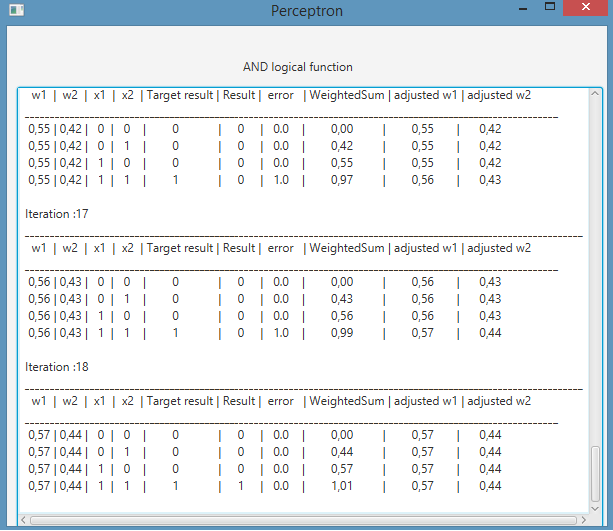
Suma tych połączeń jest przekazywana do funkcji aktywacji – activationFunction(). Metoda calculateWeight() służy do obliczenia korekty wag.



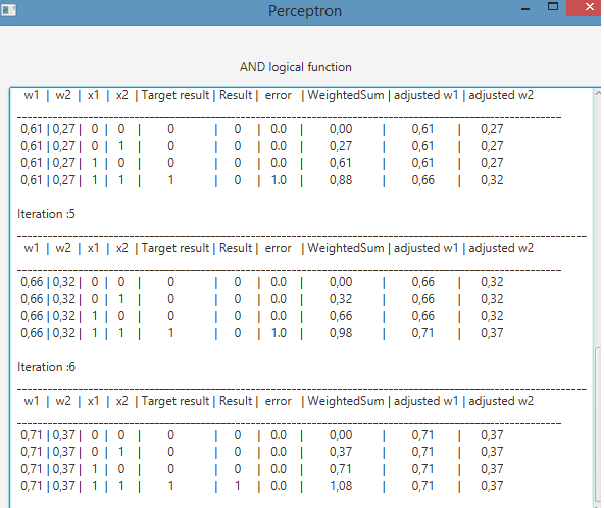
W funkcji logicalFunction() zachodzi cały proces uczenia. Wypisujemy numer iteracji by zobaczyć w którym kroku bład neuronu wyniósł wartość 0.

Przykładowe działania programu:

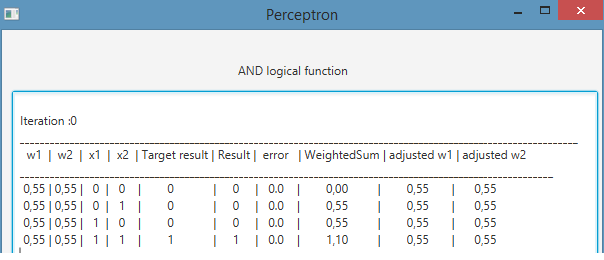
LEARNING\_RATE=0.01

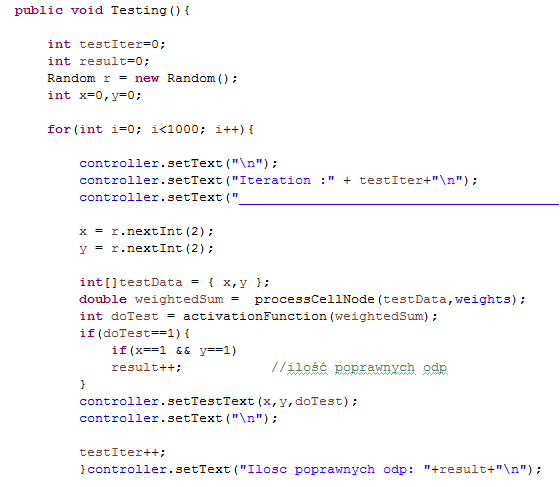


LEARNING\_RATE=0.05

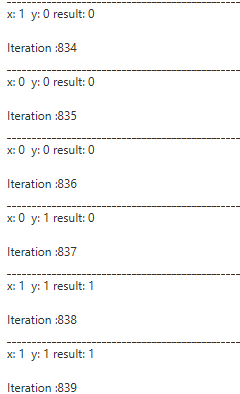


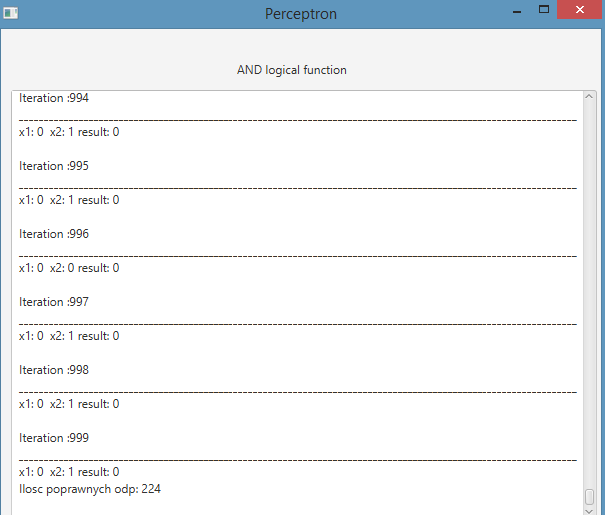
LEARNING\_RATE = 0.1





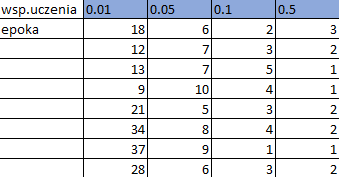
Funkcja Testing() służy do testowania neuronu dla większej ilości danych wejściowych z zakresu 0 lub 1, już bez pożądanej odpowiedzi.

Przykładowy wynik testowania dla 1000 iteracji:



\*poprawnych odpowiedzi tj. poprawnego wyniku neuronu dla wartości x1=1 i x2=1

Analiza:



Dane do wykresu: Liczba iteracji a współczynnik uczenia.

Analiza:

Jak widać na powyższych przykładach uczenie neuronu przebiegało poprawnie. Neuron trenowany na zadanym zestawie danych potrafi podać poprawną odpowiedź już w pierwszej iteracji-epoce. Uczenie neuronu trwa dopóki nie wskaże on błędu równego 0. Na wykresie przedstawiających liczbę iteracji od współczynnika uczenia pokazałam zależność między tymi dwoma wartościami.

Gdy ustawiłam współczynnik na 0.01 błąd neuronu osiągał wartość 0 najpóźniej w 37 epoce. Po zmianie na 0.05 liczba iteracji wahała się od 5 do 10. Gdy współczynnik ustawiony był na 0.1 lub 0.5 neuron uzyskiwał poprawność już w 1 epoce. Z wyników testowani pokazanych na screenach pod opisem funkcji testing() widzimy że uzyskana została 100% poprawność.

Wnioski:

Z otrzymanych wyników można łatwo zauważyć, że prędkość uczenia zależy od współczynnika uczenia. Im większy tym uczenie przebiega szybciej. Przy wyższym współczynniku neuron potrzebował niewiele iteracji, by wyliczane dane wyjściowe były prawidłowe. Podczas testowania dla różnej ilości danych nie wykryłam żadnego błędu obliczeniowego, być może dlatego że funkcja logiczna AND jest bardzo prostą operacją i prawdopodobieństwo otrzymania dobrego wyniku jest duże. Sztuczny neuron jest w stanie szybko nauczyć się podstawowych funkcji logicznych. Z łatwością można zmodyfikować program by działał dla innych operacji np. OR. Wystarczy zmienić zestaw danych uczących.