Perceptron – Podstawy Sztucznej Inteligencji

Dawid Wełna (Sprawozdanie 1)

Informacje wstępne:

Klasyfikator perceptronowy jest prostym modelem sztucznego neuronu. Ma on różne dane

wejściowe (X1....Xn) z różnymi wagami (W1...Wn)



Ważona suma s tych danych wejściowych jest potem wprowadzana do funkcji jednostkowej(zazwyczaj funkcja Heaviside)

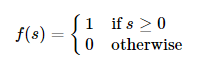
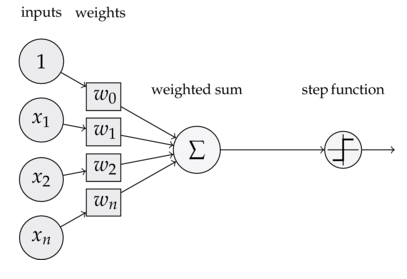


Diagram perceptronu:



Inputs – dane wejściowe

Weights - wagi

Weighted sum – suma ważona

Step function – funkcja aktywacji

Program w Pythonie

Oto prosta wersja takiego perceptronu przy użyciu języka programowania Python oraz biblioteki NumPy.

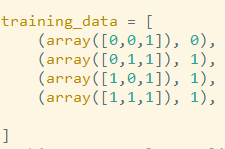
Na początku trzeba zaimportować odpowiednie moduły z bibliotek:



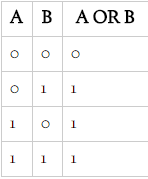
Teraz tworzę funkcje aktywującą, w tym przypadku funk. skoku jednostkowego



Następnie muszę zmapować możliwe dane wejściowe wraz ze spodziewanymi danymi wyjściowymi. Dwa pierwsze wpisy w tablicy NumPy w każdej krotce są dwiema wartościami wejściowymi. Drugi element każdej krotki to właśnie oczekiwany rezultat. Zaś trzeci wpisy w każdej z tablic numpy’ia to dana wejściowa będąca tzw. „atrapą”(z ang. „dummy”), która jest potrzebna aby przesuwać próg(znany jako granica decyzyjna) w górę lub w dół w razie potrzeby funkcji jednostkowej. Wartość tej trzeciej danej „atrapy” jest ustawiony zawsze na 1, a jej wpływ na końcowy wynik może być kontrolowany przez jej wagi.



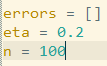
Jak można zobaczyć, ten trenujący ciąg danych mapuje definicje funkcji logicznej OR



Następnie wybieramy trzy losowe liczby w przedziale [0,1] jako początkowe wagi

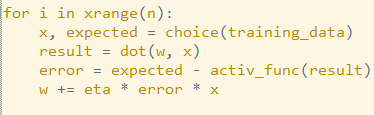


Przejdźmy do inicjalizjacji potrzebnych zmiennych. Errors- to lista, która będzie przechowywać wartości blędów, tak aby móc je przedstawić potem na wykresie. Zmienna eta kontroluje prędkość uczenia. Zmienna n to liczba iteracji.

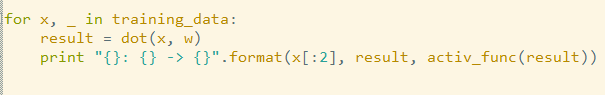


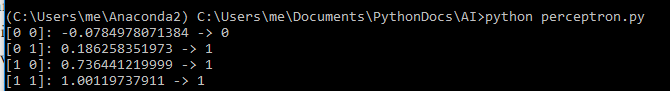
W celu znalezienia idealnych wartości dla wag „w”, próbujemy zredukować wielkość błędu do zera. W takim prostym przypadku jak ten wystarczy liczba n=100 iteracji.

Najpierw otrzymujemy losowe dane wejściowe z danych trenujących. Potem obliczamy iloczyn skalarny danych wejściowych oraz wag. Ten skalarny rezultat porównujemy z oczekiwaną wartością. Jeśli oczekiwana wartość jest większa, musimy zwiększyć wagi, jeśli jest zaś mniejsza, to analogicznie wagi zmniejszamy. Ten czynnik korygujący jest obliczany w ostatniej linijce, gdzie błąd mnożony jest przez eta oraz wektor danych wejściowych x. Następnie ta wartość dodawana jest to wektora wag, w c elu poprawienia rezultatu w kolejnej iteracji.

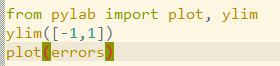


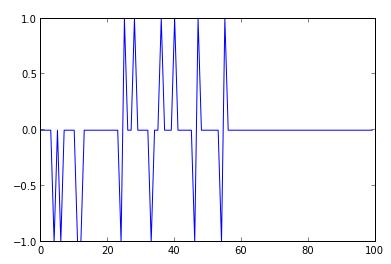
Perceptron jest już „nauczony”. Przetestujmy go:





Możemy teraz wyświetlić wykres błędów. To świetny sposób aby zobaczyć jak wyglądał proces uczenia:





Błędy stabilizują si w okolicach 60 iteracji. A o tutaj poniżej prezentuję cały kod w Pythonie.

