Jakub Siemaszko, gr. 2

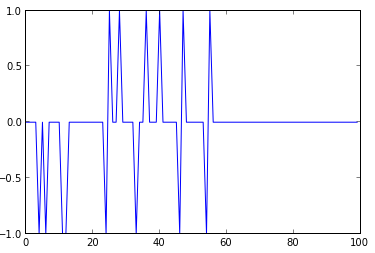
Scenariusz 1

1) syntetyczny opis budowy oraz wykorzystanego algorytmu uczenia

Perceptron to najprostsza sieć neuronowa, składająca się z jednego bądź wielu niezależnych neuronów McCullocha-Pittsa, implementująca algorytm uczenia nadzorowanego klasyfikatorów binarnych. Perceptron jest funkcją, która potrafi określić przynależność parametrów wejściowych do jednej z dwóch klas. Może być wykorzystywany tylko do klasyfikowania zbiorów liniowo separowalnych.



2) zestawienie otrzymanych wyników



Jak widać błędy stabilizują się po ok. 60 iteracji programu.

3) analiza i dyskusja błędów uczenia i testowania opracowanego perceptronu w zależności od wartości współczynnika uczenia oraz liczby danych uczących

W celu znalezienia idealnych wartości wagi „w”, staramy się zmniejszyć wielkość błędu do zera. W tym prostym przypadku n = 100 iteracji jest wystarczające; dla większego i prawdopodobnie bardziej złożonego zestawu danych wejściowych należy używać znacznie większych liczb.

4) sformułowanie wniosków

Tak jak zostało to wcześniej napisane, po ok. 60 iteracjach programu możliwe jest „nauczenie” perceptronu funkcjonowania jak logiczna bramka OR bez większej ilości błędów.

5) listing całego kodu programu

**from** random **import** choice  
**from** matplotlib.pyplot **import** ylim, plot  
**from** numpy **import** array, dot, random  
  
funkcja\_skokowa = **lambda** x: 0 **if** x < 0 **else** 1  
  
dane\_uczace = [(array([0, 0, 1]), 0), (array([0, 1, 1]), 1), (array([1, 0, 1]), 1), (array([1, 1, 1]), 1), ]  
  
r = random.rand(3)  
bledy = []  
eta = 0.2  
n = 100  
  
**for** i **in** range(n): x, oczekiwane = choice(dane\_uczace)  
wynik = dot(r, x)  
error = oczekiwane - funkcja\_skokowa(wynik)  
bledy.append(error)  
r += eta \* error \* x  
  
**for** x, \_ **in** dane\_uczace: wynik = dot(x, r)  
print(**"{}: {} -> {}"**.format(x[:2], wynik, funkcja\_skokowa(wynik)))  
  
ylim([-1,1])  
plot(bledy)