Krzysztof Kaflowski

Podstawy Sztucznej Inteligencji

Scenariusz 1

**Perceptron**

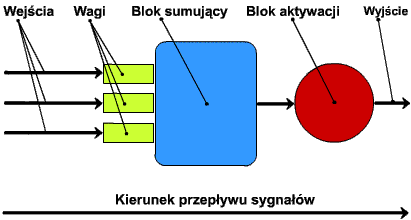
1. Celem ćwiczenia jest poznanie budowy i działanie perceptronu poprzez implementację oraz uczenie perceptronu. Wybranym przeze mnie problemem jest czy suma dwóch liczb jest większa niż 10. Aplikację napisałem samodzielnie na podstawie znalezionych źródeł w Internecie.

**2a)**

Korzystałem ze źródła z linku poniżej:

<http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/w3.htm>  
  
Krótkie streszczenie teorii zawartej na tej stronie:

Budowa perceptronu:



**Perceptronem** nazywamy prosty element obliczeniowy, który sumuje ważone sygnały wejściowe i porównuje tę sumę z progiem aktywacji - w zależności od wyniku perceptron może być albo wzbudzony (wynik 1), albo nie (wynik 0).

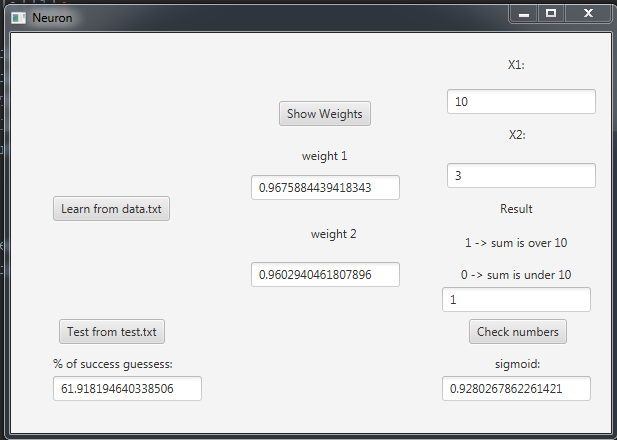
Schemat działania perceptronu:

* Inicjujemy wagi losowo.
* Dla każdego przykładu uczącego obliczamy odpowiedź perceptronu.
* Jeśli odpowiedź perceptronu jest nieprawidłowa, to modyfikujemy wagi:

w1 += n \* (d-y) \* x1   
w2 += n \* (d-y) \* x2   
b   += n \* (d-y)

gdzie n jest niewielkim współczynnikiem uczenia (n > 0), d - oczekiwana odpowiedź a y - odpowiedź neuronu.

Struktura i wygląd wykonanego programu:



Program został wykonany w wersji okienkowej dla czytelniejszych wyników.

Zastosowanie przycisków:

* Learn from data.txt – uruchamia metodę odpowiedzialną za naukę neuronu
* Test from test.txt – uruchamia metodę odpowiedzialną za testowanie neuronu i wyświetla jaki był procent dobrze wytypownych odpowiedzi przez program
* Show Weights – uruchamia metodę odpowiedzialną za wyświetlanie aktualnych wartości wag
* Check Numbers - uruchamia metodę odpowiedzialną za porównanie dwóch liczb z uwzględnieniem aktualnych wag, wyświetla wynik zero-jedynkowy oraz dokładną wartość funkcji sigmoid dla tych liczb

Kluczowe metody:

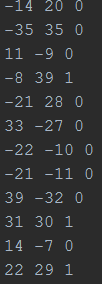
* learn2() – zczytuje wszystkie dane pochodzące z zestawu uczącego w pliku tesktowym, następnie oblicza odpowiedź perceptronu, na końcu sprawdzania każdej pary liczb, poprawia wagi
* adjustWeights() – poprawia aktualne wagi na podstawie otrzymanego błędu obliczania wyniku
* weightSum2() – oblicza średnią ważona używając aktualnych wag oraz pary liczb z danych wejściowych
* test() – oblicza wynik odpowiedzi perceptronu dla każdej pary liczb z plik służacego do walidacji neuronu, następnie wyświetla ile procent estymacji zakończyło się sukcesem
* sigmoid() - oblicza wartości funkcji sigmoidalnej, wzór poniżej

Funkcja sigmoidalna:

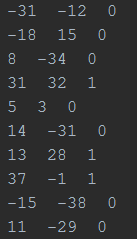
Funkcja sigmoidalna jest stosowana do obliczenia końcowej wartości odpowiedzi perceptronu. Wartość zawiera się w przedziale <0,1>. Jeżeli wartość jest bliska 0 to znaczy, że suma pary danych liczb jest mniejsza od 0, a jeśli jest bliska 1 to znaczy że suma pary danych liczb jest większa od 10.

**2b)**

W celu nauczenia perceptronu rozpoznawania czy suma dwóch liczb jest większa 10 wygenerowano plik tesktowy o nazwie „data.txt”, który w każdej linijce ma dwie zmienne oraz wartość 1 lub 0 w zależności od tego czy suma tych dwóch liczb jest większa od 10 (wynik 1) czy jest mniejsza lub równa od 10 (wynik 0). Zawiera on 8645 rekordy. Fragment pliku:



W celu przetestowania poprawności wyników wygenerowano plik o nazwie „test.txt”, który w swojej budowie jest identyczny do zestawu uczącego. Fragment dla przykładu:



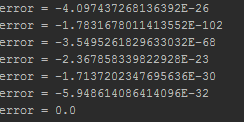
Zestawienie wyników:

Perceptron testowano dla 3 współczynników uczenia: 0,1; 0,01; 0,001.

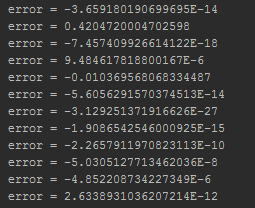
Najlepsze wyniki oraz najszybciej perceptron uczył się przy współczynniku 0,01 oraz 0,001. Wagi stabilizowały się już po pierwszym przejściu metody uczącej. Wagi ostatecznie przyjmowały zawsze wartość z przedziału <0,9;1>.

Wyniki błędów predykcji wyniku dla różnych współczynników uczenia:

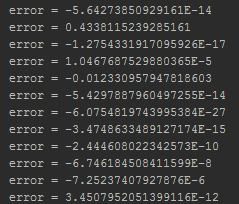
0,1:



0,01:



0,001:



Po tym jak wagi się ustabilizowały do swoich wartości uruchomiono metodę testującą. Próby dla różnej ilości danych uczącej oraz różnej ilości danych testujących wykazały, że trafność algorytmu stoi na poziomie 61%-64%.

**2c)**

Analiza i dyskusja błędów uczenia:

Średni wynik testu pokazuje około 62% trafności programu. Nie jest to niska wartość, lecz też nie do końca satysfakcjonująca. Może to być spowodowane zbyt małą ilością danych wejściowych. Innym powodem może być również źle dobrany współczynnik uczenia. Jest taka możliwość, że żaden z wyżej podanych nie jest optymalny dla zadanego problemu.

**2d)**

Wnioski:

* Budowa perceptronu jest dość elastyczna, dlatego może on być stosowany do rozwiązywania róznych problemów
* Perceptron pozwala na szybkie estymowanie wyników przy stałej ilości obliczeń na jedno rozwiązanie
* Otrzymane wyniki zależą od tego jakie dane zostaną wprowadzone w celu nauczenia neuronu
* Ważne jest dostosowanie współczynnika uczenia, aby uzyskać optymalny czas nauki neuronu

**2e)**

Listing całego kodu znajduje się w folderze „src”, w serwisie github pod linkiem:

<https://github.com/kkaflows/Perceptron-Proj1>

Znajdują się tam również pliki użyte do uczenia oraz testowania neuron.