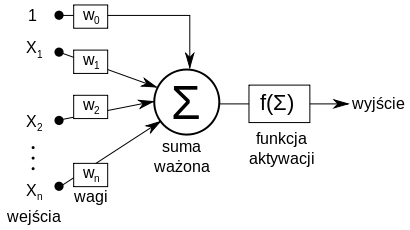
Tomasz Tomala

Podstawy Sztucznej Inteligencji

Sprawozdanie z projektu nr 1

Celem ćwiczenia buło poznanie budowy i działania perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwóch zmiennych.

**1) Syntetyczny opis budowy wykorzystanego algorytmu uczenia:**

  
Ilustracja 1: Model perceptronu McCullocha-Pittsa

Do budowy perceptronu wykorzystałem podany na wykładzie model McCullocha-Pittsa. Zaimplementowana przeze mnie klasa Perceptron składa się z trzech metod: active, process oraz learn.

Metoda ***active*** wykorzystuje unipolarną funkcję progową która zwraca wynik 0 lub 1 dla podanego argumentu.

Metoda ***process*** sumuje iloczyn sygnałów wejściowych i odpowiadających im wag. Uruchamia metodę activate z otrzymaną sumą iloczynów jako parametrem, oraz zwraca wynik tej metody.

Metoda ***learn*** wywołuje metodę process dla otrzymanych wejść jako parametrów, po czym na podstawie otrzymanego wyniku modyfikuje wszystkie wagi dla odpowiednich wejść.

**2) Zestawienie otrzymanych wyników:**

Do uczenia perceptronu wybrałem funkcję logiczną AND.

Na początku zacząłem uczyć perceptron w taki sposób, że przesyłałem mu wszystkie kombinacje danych wejściowych taką samą ilość razy. Wszystkie wagi początkowe ustawiłem na wartość równą 0.5. Z kolei współczynnik uczenia się ustawiłem na wartość 0.01. Proces uczenia wymagał 28 powtórzeń. Poniżej prezentacja na wykresie:

Następnie zacząłem eksperymentować z różnymi wagami początkowymi. W każdym kolejnym teście sprawdzałem jak długo zajmie nauka w zależności od wag początkowych. Warunki testu identyczne jak do tego powyżej. Poniżej wyniki:

|  |  |
| --- | --- |
| **Waga początkowa** | **Ilość powtórzeń** |
| 0 | 5 |
| 0.05 | 3 |
| 0.08 | 5 |
| 0.13 | 8 |
| 0.17 | 10 |
| 0.2 | 11 |

Kolejnym krokiem było sprawdzenie jak będzie wyglądał proces uczenia się dla różnych współczynników uczenia się. Test taki sam jak powyżej, jednak wagi początkowe są równe 0.5. Poniżej przedstawiam wyniki:

|  |  |
| --- | --- |
| **Współczynnik uczenia** | **Ilość powtórzeń** |
| 0.01 | 28 |
| 0.1 | 3 |
| 0.5 | 5 |
| 1.2 | 1 |
| 5 | 5 |

Następnie sprawdziłem jak będą wyglądać wyniki dla różnej ilości danych uczących. Tym razem jednak ustaliłem stałą wartość dla wszystkich wag początkowych równą 0.5 a współczynnik uczenia wynosił 0.1. Poniżej przedstawiam wyniki dla różnych kombinacji danych wejściowych i ilości ich powtórzeń:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **x1** | **x2** | **Ilość powtórzeń** | **wynik** | **x1** | **x2** | **Ilość powtórzeń** | **wynik** |
| 0 | 0 | 1 | błędny | 0 | 0 | 2 | błędny |
| 0 | 1 | 1 | 0 | 1 | 2 |
| 1 | 0 | 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 1 | 1 | 2 |

| **x1** | **x2** | **Ilość powtórzeń** | **wynik** | **x1** | **x2** | **Ilość powtórzeń** | **wynik** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 4 | poprawny | 0 | 0 | 4 | poprawny |
| 0 | 1 | 3 | 0 | 1 | 3 |
| 1 | 0 | 2 | 1 | 0 | 2 |
| 1 | 1 | 1 | 1 | 1 | 0 |

| **x1** | **x2** | **Ilość powtórzeń** | **wynik** |
| --- | --- | --- | --- |
| 0 | 0 | 6 | poprawny |
| 0 | 1 | 3 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

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

Uczenie się perceptronu zależy od trzech czynników: wag początkowych, współczynnika uczenia oraz od danych uczących. Jak widać na powyższych danych, wagi początkowe wpływają na szybkość uczenia się jednak współczynnik uczenia wpływa na to o wiele bardziej, ponieważ wagi i tak ulegają ciągłej modyfikacji. Jednak w głównej mierze to od danych uczących zależy czy perceptron będzie w stanie się nauczyć czy też nie, co wyraźnie widać na ostatnim teście. W przypadku funkcji logicznej AND nie musiałem podawać perceptronowi do nauczenia się danych wejściowych [1,1] ani razu, jednak musiałem go nauczyć pozostałych trzech kombinacji odpowiednią ilość razy, w przeciwnym wypadku wynik był błędny.

**4) Sformułowanie wniosków:**

Dla pojedynczego perceptronu proces uczenia jest bardzo prosty. Jesteśmy w stanie nauczyć go prostej funkcji logicznej w zaledwie kilku krokach. Całość opiera się o wykonywanie tych samych operacji dla różnych danych wejściowych i odpowiednim modyfikowaniu wag.

W całym procesie uczenia się najważniejsze są dane uczące. Należy dobrać je w odpowiedni sposób i powtórzyć uczenie się na ich podstawie odpowiednią ilość razy. W przeciwnym wypadku wyniki będą niepoprawne. Wagi początkowe również wpływają na proces uczenia, jednak mają one raczej marginalne znaczenie. O wiele większe znaczenie ma współczynnik uczenia. Jeżeli wybierzemy za mały, to perceptronowi zajmie bardzo długo, aż uzyskamy odpowiednie wyniki. Z kolei za duża wartość tego współczynnika może sprawić, że nie będziemy w stanie w ogóle trafić w odpowiednie wagi, lub też zajmie to bardzo dużo czasu. Dlatego właśnie musimy dokonać optymalnego wyboru współczynnika uczenia.

**5) Listing całego kodu**

**package perceptron;**

import java.util.Random;

public class Perceptron {

private int noi; //ilość wejść

private double[] w; //wagi

public Perceptron ( int numbers\_of\_inputs ) {

noi = numbers\_of\_inputs;

w = new double[noi];

for ( int i = 0; i < noi; i++ )

w[i] = new Random().nextDouble(); //wagi początkowe są losowane

}

//funkcja aktywująca

private int active ( double y\_p ) {

return y\_p < 0 ? 0 : 1;

}

//sumator

public int process ( int[] x ) {

double y\_p = 0;

for ( int i = 0; i < noi; i++ )

y\_p += x[i] \* w[i];

return active( y\_p );

}

//uczenie

public void learn ( int[] x, double y, double lr ) {

double y\_p = process( x );

for ( int i = 0; i < noi; i++ )

w[i] += ( y - y\_p ) \* lr \* x[i];

}

public double getW ( int i ) {

return w[i];

}

}

**package perceptron;**

import java.util.Arrays;

public class Main {

public static void main ( String[] args ) {

int number\_of\_inputs = 3;

Perceptron perc = new Perceptron( number\_of\_inputs );

int n = 0; //licznik ilości epok uczenia się

int r = 4; //rozmiar tablic danych wejściowych

double learning\_rate = 0.1; //krok uczenia się

int x0 = 1; //bias

int[] x1 = { 0, 0, 1, 1 }; //dane wejściowe do AND

int[] x2 = { 0, 1, 0, 1 };

int[] y = { 0, 0, 0, 1 }; // dane oczekiwane do AND

int[] wyj = new int[4]; //tablica przechowująca wyniki testowania perceptronu

//uczenie perceptronu

while ( ! Arrays.*equals*( y, wyj ) ) {

for ( int i = 0; i < r; i++ )

perc.learn( new int[] { x0, x1[i], x2[i] }, y[i], learning\_rate );

for ( int i = 0; i < r; i++ )

wyj[i] = perc.process( new int[] { x0, x1[i], x2[i] } );

n++;

}

System.*out*.println( "Ilość kroków do nauczenia się = " + n );

}

}

**Bibliografia:**

<https://pl.wikipedia.org/wiki/Neuron_McCullocha-Pittsa>