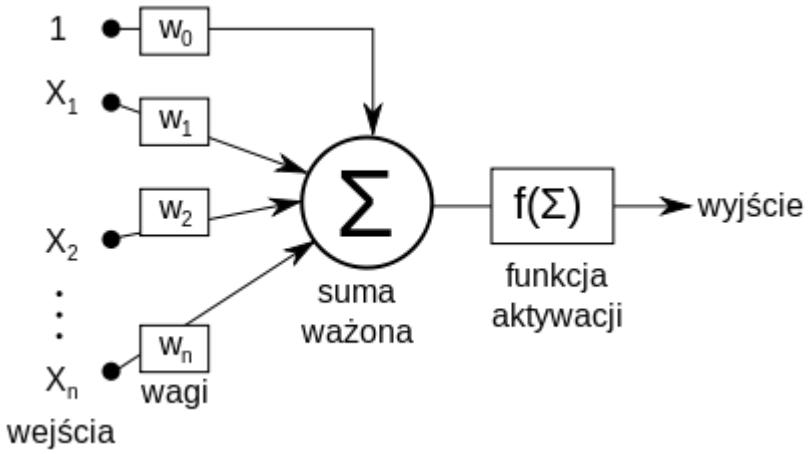
Tomasz Tomala
Podstawy Sztucznej Inteligencji
Sprawozdanie z projektu nr 1

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:

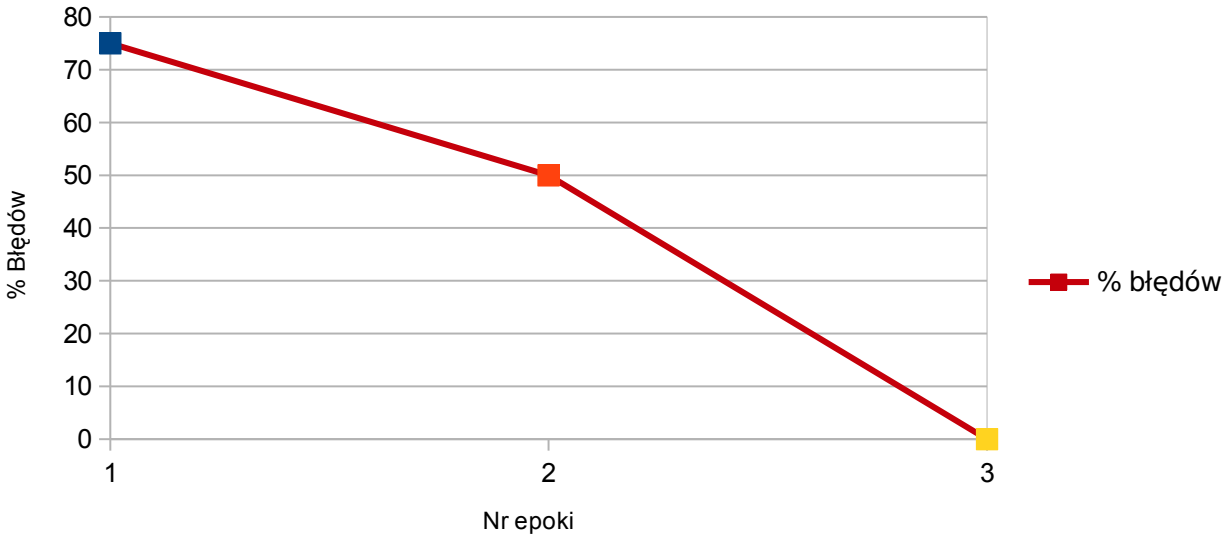


Ilustracja 1: Model perceptronu McCullocha-Pittsa

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. Do poprawnego nauczenia wystarczyły 3 takie powtórzenia uczenia. Poniżej prezentacja na wykresie:



Następnie zacząłem eksperymentować z różnymi wagami początkowymi. Poniżej wyniki:

Waga początkowa	Ilość powtórzeń
0	3
0.2	3
0.5	3
0.7	4
1	6
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
0.1	3
0.15	3
0.2	2
0.25	5
0.3	1

Następnie sprawdziłem jak będą wyglądać wyniki dla różnej ilości danych uczących. Tym razem jednak ustaliłem stałą wartość dla wszystkich wag początkowych równą 0.5.
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, mimo to w głównej mierze to od danych uczących zależy czy perceptron będzie w stanie się nauczyć czy też nie. 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 danych.
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;

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] = 0.5;
    }

    //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;

public class Main {
    public static void main ( String[] args ) {

        int number_of_inputs = 3;

        Perceptron perc = new Perceptron( number_of_inputs );

        int n = 1; //ilość powtórzeń uczenia się
        int r = 4; //rozmiar tablic danych wejściowych
        double learning_rate = 0.3; //krok uczenia się
        int x0 = 1; //bias

        //dane wejściowe do AND i OR
        int[] x1 = { 0, 0, 1, 1 };
        int[] x2 = { 0, 1, 0, 1 };

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

        //uczenie perceptronu
        for ( int j = 0; j < n; j++ )
            for ( int i = 0; i < r; i++ )
                perc.learn( new int[] { x0, x1[i], x2[i] }, y[i], learning_rate );

        //wyświetlenia wag po zakończeniu uczenia
        System.out.println( "WAGI:" );
        for ( int i = 0; i < number_of_inputs; i++ )
            System.out.println( "w" + i + " = " + perc.getW( i ) );
        System.out.println();

        //testowanie perceptronu
        System.out.println( perc.process( new int[] { x0, 0, 0 } ) );
        System.out.println( perc.process( new int[] { x0, 0, 1 } ) );
        System.out.println( perc.process( new int[] { x0, 1, 0 } ) );
        System.out.println( perc.process( new int[] { x0, 1, 1 } ) );
    }
}
```

Bibliografia:

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