Paweł Pajor

PSI

Sprawozdanie ze scenariusza nr 1

Cel ćwiczenia:

Ćwiczenie miało na celu poznanie budowy i implementację Perceptrony oraz uczenie perceptronu którego zadaniem było wykonanie pewnej wybranej funkcji logicznej (W moim przypadku AND).

Wykonane kroki:

1. **Wybieram do implementacji język obiektowy Java.**
2. Wybieram wagi o losowych wartościach.
3. Obliczam odpowiedzi Perceptronu dla każdego przykładu uczącego
4. Modyfikuję wagi do czasu, aż perceptron nie udzieli prawidłowej odpowiedzi.
5. W przypadku, gdy po skończeniu iteracji przechodzącej przez wszystkie dane wejściowe ani razu nie trzeba było zmieniać czasu oznacza to, że pętla może się skończyć a perceptron umie już to, co trzeba

Wyniki testów:

Maksymalna liczba iteracji: 1000, Próg błędu: 0.3 Współczynnik nauczania: 0.1

Loop ended after iterations : 1

Loop ended after iterations : 7

Loop ended after iterations : 5

Loop ended after iterations : 6

Loop ended after iterations : 2

Maksymalna liczba iteracji: 1000, Próg błędu: 0.3 Współczynnik nauczania: 0.01

Loop ended after iterations : 22

Loop ended after iterations : 65

Loop ended after iterations : 44

Loop ended after iterations : 39

Loop ended after iterations : 66

Max iteracji: 1000, współczynnik nauczania: 0.001, próg błędu: 0.3

Loop ended after iterations : 686

Loop ended after iterations : 648

Loop ended after iterations : 379

Loop ended after iterations : 465

Loop ended after iterations : 527

Wnioski:

Z przytoczonych powyżej wyników testów jasno wynika, że im mniejszy współczynnik uczenia tym więcej prób potrzebuję perceptron aby nauczyć się swojego zadania. Warto jednak brać też pod uwagę fakt, że liczby zostały dobierane losowo przez funkcję Random dlatego wyniki pomiarów mogą różnić się dla każdego pojedynczego, wykonanego testu.

Kod:

Main.java

```
/**
 * Created by elgha_000 on 22.10.2017.
 */
public class Main

{

    public static void main(String[] args)
    {
        Perceptron perceptron = new Perceptron();
        double inputs[][] = {{0, 0}, {0, 1}, {1, 0}, {1, 1}};

        int outputs[] = {0, 0, 0, 1};
        for (int i = 0; i < 5; i++)
        {
            perceptron.Learn(inputs, outputs, 0.3, 0.001, 1000);
        }
    }
}
```

Perceptor.java

```
/**
 * Created by elgha_000 on 22.10.2017.
 */
```

```
import java.util.Random;
import java.util.ArrayList;


public class Perceptron

{
    int proby=0;

    double prog;
    double[] wagi;
    public int Output(double[] input)
    {
        double suma = 0;
        for(int i = 0; i < input.length; i++)

        {
            suma += wagi[i] * input[i];
        }

        if(suma > prog)
            return 1;
        else
            return 0;
    }

    public void Learn(double[][] inputs, int[] outputs, double prog, double
learningRate, int numberOfIterations)

    {
        this.prog = prog;
        wagi = new double[inputs[0].length];
        Random random = new Random();


        for(int i=0;i<inputs[0].length;i++)
            wagi[i] = random.nextDouble();


        for(int i = 0; i < numberOfIterations; i++)
        {

            int Errorsuma = 0;
            for(int j = 0; j < outputs.length; j++)
            {

                int output = Output(inputs[j]);

                int error = outputs[j] - output;

                Errorsuma += error;

                for(int k = 0;k < inputs[0].length; k++)
                {

                    double delta = learningRate * inputs[j][k] * error;


                    wagi[k] += delta;
                }
            }
```

```
            if(Errorsuma == 0)


            {
                System.out.println("Loop ended after iterations : " + i);
                break;
            }
        }
    }

}
```