Konrad Tabiś

Sprawozdanie - Podstawy Sztucznej Inteligencji – Scenariusz 1

Temat ćwiczenia: Budowa i działanie perceptronu.

Źródło algorytmu wg którego zaimplementowałem sztuczny neuron:
http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/w3.htm

Na początku inicjujemy wagi losowo, a następnie porównujemy wagi z wynikiem oczekiwanym i modyfikujemy je przy pomocy wzorów:

w_1+=n*(d-y) *x_1

w_2+=n*(d-y) *x_2

b +=n*(d-y)
gdzie w_1, w_2 to wagi, n jest niewielkim współczynnikiem uczenia(n>0), d-oczekiwaną odpowiedzią, y-odpowiedzią neuronu, b-progiem wzbudzenia perceptronu, a x_1, x_2 wartościami wejściowymi.

```
Wyniki uczenia perceptonu funkcji AND

Przeprowadzono 16 prób uczenia się
Dla 0,0 przeprowadzono 4próby
Dla 1,0 przeprowadzono 4próby
Dla 0,1 przeprowadzono 4próby
Dla 1,1 przeprowadzono 4próby

WAGI:
W0 = -0.6063312043857374
W1 = 0.5545901562959658
W2 = 0.09817600800264537

Wynik dla x1=0 i x2=0
0

Wynik dla x1=0 i x2=1
0

Wynik dla x1=1 i x2=0
0

Wynik dla x1=1 i x2=1
1
```

Dla 16 prób, po 4 próby dla każdej danej nadal zdarzają się błędy:

```
Wyniki uczenia perceptonu funkcji AND

Przeprowadzono 16 prób uczenia się
Dla 0,0 przeprowadzono 4 próby
Dla 1,0 przeprowadzono 4 próby
Dla 0,1 przeprowadzono 4 próby
Dla 1,1 przeprowadzono 4 próby

WAGI:
W0 = -0.8465243531355882
W1 = 0.9384721888569619
W2 = 0.16500538048493119

Wynik dla x1=0 i x2=0
0

Wynik dla x1=0 i x2=1
0

Wynik dla x1=1 i x2=0
1

Wynik dla x1=1 i x2=1
1
```

Testując dalej możemy zauważyć, że nawet dla 40 prób ciągle pojawiają się błędy

```
Przeprowadzono 40 prób uczenia się
Dla 0,0 przeprowadzono 10 próby
Dla 1,0 przeprowadzono 10 próby
Dla 0,1 przeprowadzono 10 próby
Dla 1,1 przeprowadzono 10 próby

WAGI:
W0 = -0.7125387079617952
W1 = 0.020713117723189733
W2 = 0.680739593867906

Wynik dla x1=0 i x2=0
0

Wynik dla x1=0 i x2=1
0

Wynik dla x1=1 i x2=0
0

Wynik dla x1=1 i x2=1
0
```

Dopiero przy 400 próbach, po 100 dla każdej danej przeprowadzenie 1000 prób daje pozytywny rezultat.

```
Wyniki uczenia perceptonu funkcji AND

Przeprowadzono 400 prób uczenia się
Dla 0,0 przeprowadzono 100 próby
Dla 1,0 przeprowadzono 100 próby
Dla 0,1 przeprowadzono 100 próby
Dla 1,1 przeprowadzono 100 próby

WAGI:
W0 = -0.2544011493925638
W1 = 0.24345517626288593
W2 = 0.23978909833733109

Wynik dla x1=0 i x2=0
0

Wynik dla x1=0 i x2=1
0

Wynik dla x1=1 i x2=0
0

Wynik dla x1=1 i x2=1
1
```

Wszystkie powyższe wyniki są dla współczynnika uczenia się równego 0.1, oraz losowych wag początkowych z zakresu [0,1]. Dlatego możemy tutaj stwierdzić że dla losowych wag początkowych dopiero przy 400 próbach dostaniemy zawsze dobre wyniki.

Po ustawieniu wagi początkowej na 0.5 wystarczyły 3-krotne powtórzenie uczenia się dla każdej danej aby otrzymać poprawne wyniki.

Następnie zacząłem zmieniać współczynnik uczenia i tutaj nie da się jednoznacznie stwierdzić czy zwiększeni lub zmniejszenie go ulepsza nam szybkość uczenia się perceptronu.

Z moich prób wynika że najlepszy wynik otrzymujemy dla wagi początkowej równej 0.5 i współczynnika uczenia się 0.3, ponieważ wystarczyło raz przeprowadzić proces uczenia dla każdej danej aby dostać poprawne wyniki.

Wszystkie trzy czynniki wpływają na uczenie się perceptronu: wagi początkowe, współczynnik uczenia się a także ilość powtórzeń danych uczących.

Listing kodu:

```
public class Main {
   public static void main(String[] args) {
      Perceptron perceptron = new Perceptron();

      int a = 2, b = 0, c = 3, d = 3, s = a + b + c + d;
      int bias = 1;
      double learning_factor = 0.3;

      //tablice wejściowe
      int[] x1 = {0, 1, 0, 1};
      int[] x2 = {0, 0, 1, 1};

      //tabela wynikowa dla AND
      int[] y = {0, 0, 0, 1};

      //j-ilość wykonanych pętli na konkretnym zestawie danych
      //pętla dla pierwszej danej [0,0|0]
      for (int i = 0, j = 0; j < a; j++) {
         perceptron.learn_perceptron(bias, x1[i], x2[i], y[i], learning_factor);
      }
      //pętla dla drugiej danej [1,0|0]
      for (int i = 1, j = 0; j < b; j++) {
         perceptron.learn_perceptron(bias, x1[i], x2[i], y[i], learning_factor);
      }
      //pętla dla trzeciej danej [0,1|0]
      for (int i = 2, j = 0; j < c; j++) {
         perceptron.learn_perceptron(bias, x1[i], x2[i], y[i], learning_factor);
      }
      //pętla dla czwartej danej [1,1|1]
      for (int i = 1, j = 0; j < d; j++) {
         perceptron.learn_perceptron(bias, x1[i], x2[i], y[i], learning_factor);
      }

      System.out.println("Wyniki uczenia perceptonu funkcji AND\n");
      System.out.println("Przeprowadzono " + s + " prób uczenia się\nDla 0,0 przeprowadzono " + a + "
próby\nDla 1,0 przeprowadzono " + b +
            " próby\nDla 0,1 przeprowadzono " + c + " próby\nDla 1,1 przeprowadzono " + d + "
próby\n");

      System.out.println("WAGI:");
      System.out.println("W0 = " + perceptron.weight0);
      System.out.println("W1 = " + perceptron.weight1);
      System.out.println("W2 = " + perceptron.weight2);

      System.out.println("\nWynik dla x1=0 i x2=0");
      System.out.println(perceptron.process(bias, 0, 0));
```

```
        System.out.println("\nWynik dla x1=0 i x2=1");
        System.out.println(perceptron.process(bias, 0, 1));
        System.out.println("\nWynik dla x1=1 i x2=0");
        System.out.println(perceptron.process(bias, 1, 0));
        System.out.println("\nWynik dla x1=1 i x2=1");
        System.out.println(perceptron.process(bias, 1, 1));
    }
    }


import java.util.Random;
public class Perceptron {
    Random generator =new Random();

    //wagi poczatkowe
    //1)ustawione randomowo
    public double weight0=generator.nextDouble(),weight1=generator.nextDouble()
,weight2=generator.nextDouble();
    //2)ustawione na wartość stałą
    //public double weight0=0.5,weight1=0.5,weight2=0.5;

    public int perceptronOut (double perceptronOut)
    {
        if(perceptronOut<0)return 0;
        else return 1;
    }
    public int process ( int x0, int x1, int x2 ) {
        double y_p = x0 * weight0 + x1 * weight1 + x2 * weight2;
        return perceptronOut( y_p );
    }

    //algorytm uczenia perceptronu
    public void learn_perceptron(int x0, int x1, int x2, double y, double  learning_factor)
    {
        //double perceptronOut = x0 * weight0 + x1 * weight1 + x2 * weight2;
        //perceptronOut = perceptronOut(perceptronOut);
        double percOut=process(x0,x1,x2);
        weight0 += ( y - percOut ) * learning_factor * x0;
    }
}
```