Krzysztof Sekuła

Sprawozdanie z projektu nr 5

Celem ćwiczenia było poznanie budowy działania sieci Kohonena przy wykorzystaniu reguły Winner Takes All do odwzorowywania istotnych cech kwiatów.

Opis budowy algorytmu:

Sieci Kohonena są przykładem sieci uczących się bez nadzoru. Oznacza to że sieci nie są prezentowane wzorcowe odpowiedzi a jedynie dane wejściowe. Neurony w sieci Kohonena są uporządkowane względem siebie przestrzennie. Często realizowane jest to za pomocą siatki dwuwymiarowej. Uczenie polega na wyznaczeniu zwycięskiego neuronu i modyfikacji jego wektora wag w kierunku prezentowanego wektora wejściowego x. zwycięskim neuronem jest ten neuron którego wagi synaptyczne są najbardziej zbliżone do prezentowanego wektora danych wejściowych. Zatem wektor wag zwycięskiego neuronu wyznacza się za pomocą:

$$d(x, w_w) = \min_{1 \le i \le n} d(x, w_i)$$

Norma d może być dowolną normą. W trakcie uczenia zwycięzca oraz jego sąsiedztwo podlega adaptacji wag zgodnie z regułą Kohonena:

$$w_i(k+1) = w_i(k) + \eta_i(k)[x - w_i(k)]$$

Parametr eta krok uczenia, poważnie malejący w trakcie kolejnych iteracji. W przypadku danych wejściowych o małych wymiarach istnieje potrzeba normalizacji wektorów wejściowych.

Algorytm WTA:

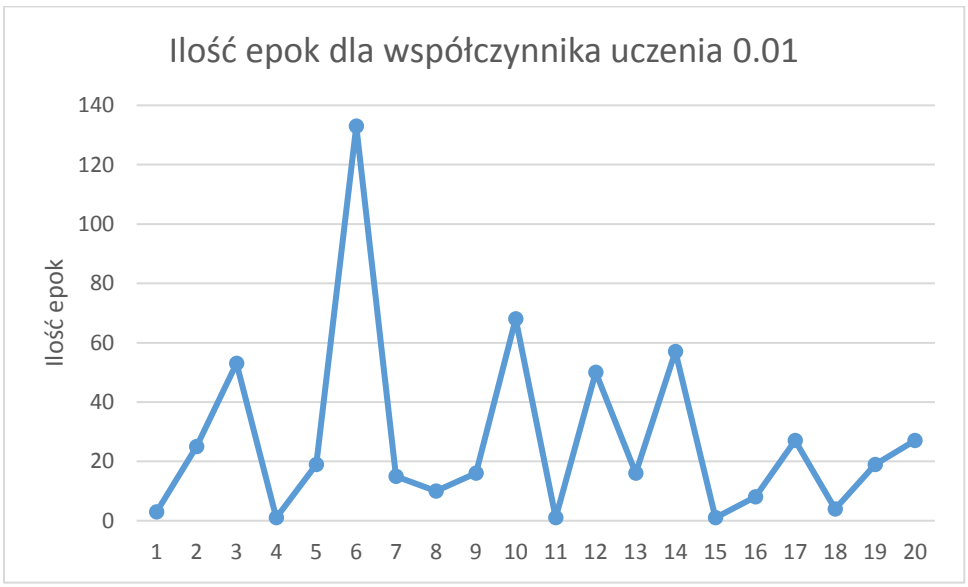
W metodzie tej tylko zwycięski neuron ma prawo adaptacji swoich wag. Aby uniknąć przypadku, w którym niektóre neurony nie mają szans wygrać z najsilniejszymi neuronami, często wprowadza się mechanizm zmęczenia. Polega on na tym że zwycięski neuron pauzuje na pewien czas nie biorąc udziału w konkurencji, dając tym samym szanse na wygranie innym neuronom.

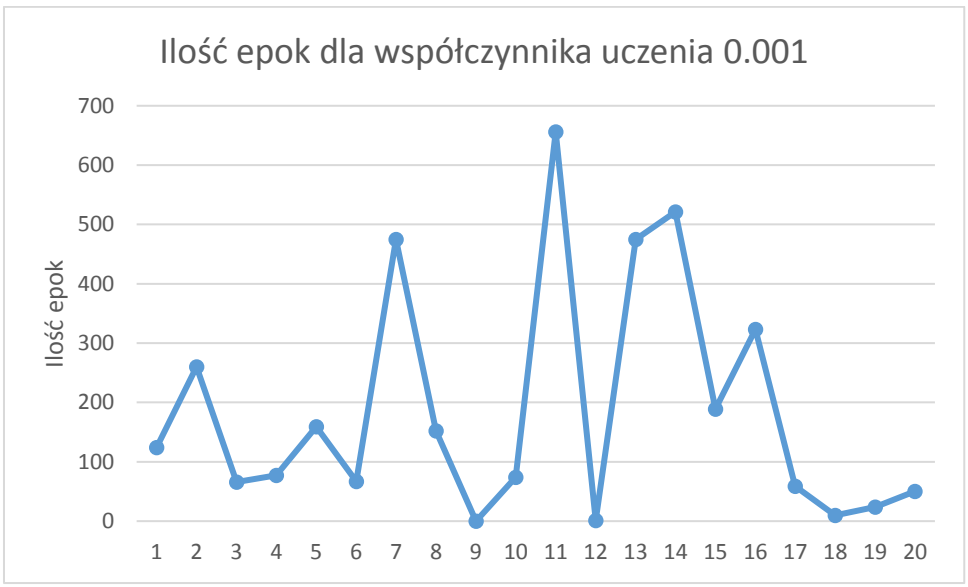
Zestawienie wyników:

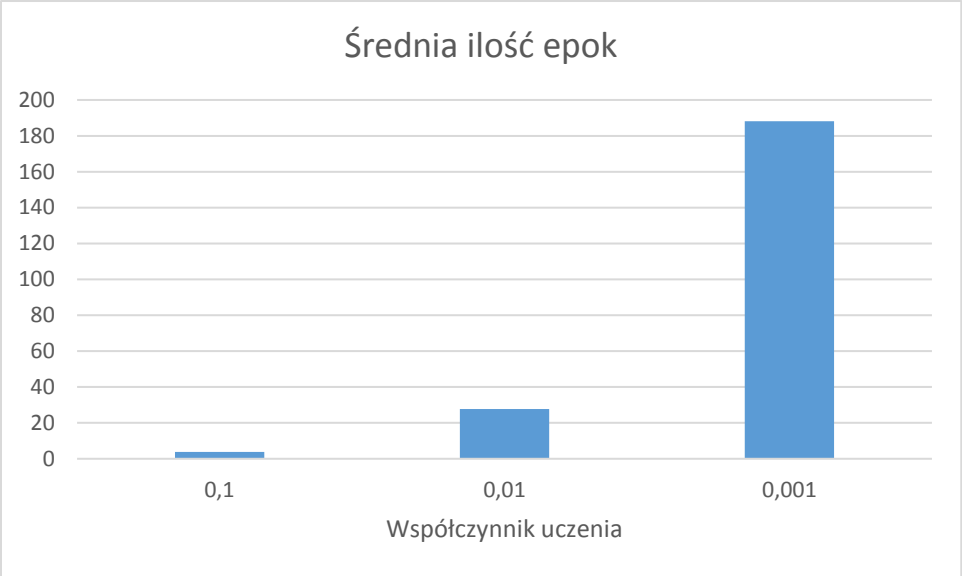
Jako dane uczące posłużył numeryczny opis cech kwiatów. Każdy z 3 rodzajów kwiatów składa się z 4 składowych. Dla każdego gatunku wybrałem do nauki 10 kwiatów a do testów 5. Testy przeprowadziłem dla 3 współczynników uczenia.

LP	współczynnik uczenia:	0,1	0,01	0,001
1	poprawność:	100%	100%	100%
	ilość epok:	10	3	124
2	poprawność:	100%	100%	100%
	ilość epok:	1	25	260
3	poprawność:	100%	100%	100%
	ilość epok:	2	53	66
4	poprawność:	100%	100%	100%
	ilość epok:	4	1	77
5	poprawność:	100%	100%	100%
	ilość epok:	1	19	159
6	poprawność:	100%	100%	93%
	ilość epok:	1	133	67
7	poprawność:	100%	100%	100%
	ilość epok:	9	15	475
8	poprawność:	100%	100%	93%
	ilość epok:	3	10	152
9	poprawność:	100%	100%	100%
	ilość epok:	1	16	1%
10	poprawność:	100%	100%	100%
	ilość epok:	5	68	74
11	poprawność:	100%	93%	100%
	ilość epok:	11	1	656
12	poprawność:	100%	100%	100%
	ilość epok:	1	50	137%
13	poprawność:	100%	100%	100%
	ilość epok:	11	16	475
14	poprawność:	100%	100%	100%
	ilość epok:	4	57	521
15	poprawność:	100%	93%	100%
	ilość epok:	4	1	189


Ilość epok dla współczynnika uczenia 0.1
Ilość epok
0
2
4
6
8
10
12
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Ilość epok dla współczynnika uczenia 0.01
Ilość epok
0
20
40
60
80
100
120
140
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Ilość epok dla współczynnika uczenia 0.001
Ilość epok
0
100
200
300
400
500
600
700
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20



Wnioski:

Na podstawie powyższych wyników możemy stwierdzić że współczynnik uczenia ma wpływ na poprawność otrzymanych wyników jak i również na ilość epok potrzebnych do uczenia. Dla współczynnika równego 0.1 poprawność wyników zawsze wynosiła 100%, a ilość epok wahała się od 1 do 11. W momencie zmniejszenia współczynnika uczenia pojawiły się błędy oraz liczba epok znacznie wzrosła. Podczas testowania można zauważyć że procent błędów był stosunkowo niewielki. Zawdzięczamy to odpowiedniej ilości danych uczących. Jeśli danych byłoby za mało to zwycięskim neuronem mógłby się okazać neuron, który podczas uczenia został wyłoniony jako zwycięzca dla innego zbioru kwiatów, co mogłoby spowodować znaczny wzrost ilości błędów. Ważnym aspektem również była odpowiednia ilość neuronów. Zbyt duża liczba neuronów mogłaby spowodować zły podział danych na zbiory. Natomiast zbyt mała liczba mogłaby doprowadzić do tego że tylko jeden neuron byłby zwycięzcą a reszta pozostałaby martwa.

Listing kodu:

```java
import java.util.Random;

public class WTA {

    private int noi;
    private double[] w;

    public WTA ( int numbers_of_inputs ) {
        noi = numbers_of_inputs;
        w = new double[noi];

        for ( int i = 0; i < noi; i++ )
            w[i] = new Random().nextDouble();
    }

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

        for ( int i = 0; i < noi; i++ )
            w[i] += lr * ( x[i] - w[i] );
    }


    public double[] getW () {
        return w;
    }
}
```

```java
public class Main{

    private static double learningRate = 0.001;
    private static int nots = 5;       //liczba danych testujacych
    private static int noi = 4;          //liczba wejsc
    private static int nols = 5;    //liczba danych uczących
    private static int nof = 3;          //liczba kwiatow
    private static int ll = 1000;         //maksymalna ilosc epok uczenia
    private static int non = 200;        //liczba neuronow
    public static void main ( String[] args ) {

        int successCounter = 0;        //licznik prób uczenia zakończonych
powodzeniem
        int failCounter = 0;    //licznik prób uczenia zakończonych
niepowodzeniem

        while ( successCounter != 15 && failCounter != 100 ) {

            WTA[] kohonens = new WTA[non];
            for ( int i = 0; i < non; i++ )
                kohonens[i] = new WTA( noi );

            int ages = learn( kohonens );

            if ( ages != ll ) {
                successCounter++;

                int winner;
                System.out.println( "Nr"+successCounter );
                System.out.println( "PO UCZENIU" );
                for ( int i = 0; i < nof; i++ ) {
```

```
                    winner = getWinner( kohonens, Flower.flowerLearn[i][0]
);
                    System.out.println( "Flower[" + i + "] winner = " +
winner );
                }
                System.out.println();

                System.out.println( "PO TESTOWANIU" );
                for ( int i = 0; i < nof; i++ ) {
                    for ( int j = 0; j < nots; j++ ) {
                        winner = getWinner( kohonens,
Flower.flowerTest[i][j] );
                        System.out.println( "Flower[" + i + "][" + j + "]
test winner = " + winner );
                    }
                    System.out.println();
                }
                System.out.println();


                System.out.println( "Ilość epok = " + ages + "\n\n\n" );
            }
            else failCounter++;
        }
        System.out.println( "\nIlość niepowodzeń = " + failCounter );
    }


    //uczenie sieci
    private static int learn ( WTA[] kohonens ) {

        int counter = 0;
        int winner;

        int[][] winners = new int[nof][nols];
        for ( int i = 0; i < nof; i++ )
            for ( int j = 0; j < nols; j++ )
                winners[i][j] = - 1;

        while ( ! isUnique( winners ) ) {


            for ( int i = 0; i < nof; i++ ) {
                for ( int j = 0; j < nols; j++ ) {
                    winner = getWinner( kohonens, Flower.flowerLearn[i][j]
);
                    kohonens[winner].learn( Flower.flowerLearn[i][j],
learningRate );
                }
            }

            for ( int i = 0; i < nof; i++ )
                for ( int j = 0; j < nols; j++ )
                    winners[i][j] = getWinner( kohonens,
Flower.flowerLearn[i][j] );


            if ( ++ counter == ll )
                break;
        }
```

```
        return counter;
    }


    private static boolean isUnique ( int[][] winners ) {


        for ( int i = 0; i < nof; i++ )
            for ( int j = 1; j < nols; j++ )
                if ( winners[i][0] != winners[i][j] )
                    return false;


        for ( int i = 0; i < nof; i++ )
            for ( int j = 0; j < nof; j++ )
                if ( i != j )
                    if ( winners[i][0] == winners[j][0] )
                        return false;

        return true;
    }



    private static int getWinner ( WTA[] kohonens, double[] vector ) {

        int winner = 0;
        double minDistance = distanceBetweenVectors( kohonens[0].getW(),
vector );


        for ( int i = 0; i < non; i++ ) {
            if ( distanceBetweenVectors( kohonens[i].getW(), vector ) <
minDistance ) {
                winner = i;
                minDistance = distanceBetweenVectors( kohonens[i].getW(),
vector );
            }
        }

        return winner;
    }



    public static double distanceBetweenVectors ( double[] vector1,
double[] vector2 ) {

        double suma = 0.0;

        for ( int i = 0; i < vector1.length; i++ ) {

            suma += Math.abs( vector1[i] - vector2[i] );
        }

        return Math.sqrt( suma );
    }

}
```

Źródła:

http://www.michalbereta.pl/dydaktyka/WdoSI/lab_neuronowe_II/Sieci_Neuronowe_2%20Sieci%20Kohonena.pdf

https://en.wikipedia.org/wiki/Iris_flower_data_set