Małgorzata Bień IS gr. 1

Sprawozdanie do scenariusza nr 5

Temat ćwiczenia: Budowa i działanie sieci Kohonena dla WTA.

1. Cel: Celem ćwiczenia jest poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTA do odwzorowywania istotnych cech kwiatów.

2. Syntetyczny opis budowy sieci oraz algorytmu uczenia

- Sieci Kohonena są jednym z podstawowych typów sieci samoorganizujących się.
- Podstawę samoorganizacji sieci neuronowych stanowi prawidłowość, że globalne uporządkowanie sieci jest możliwe przez działania samoorganizujące prowadzone lokalnie w różnych punktach sieci, niezależnie od siebie.
- W wyniku przyłożonych sygnałów wejściowych następuje w różnym stopniu aktywacja neuronów, dostosowująca się wskutek zmiany wartości wag synaptycznych do zmian wzorców uczących.
- Uczenie odbywa się metodą samoorganizującą typu konkurencyjnego. Polega ona na podawaniu na wejścia sieci sygnałów, a następnie wybraniu w drodze konkurencji zwycięskiego neuronu, który najlepiej odpowiada wektorowi wejściowemu.

Zasady działania sieci Kohonena:

- w sieciach Kohonena, przeważnie jednowarstwowych, każdy neuron połączony jest ze wszystkimi składowymi wektora wejściowego x,
- wagi połączeń synaptycznych neuronów tworzą wektor w,
- każdy węzeł oblicza swój poziom aktywacji jako iloczyn wektora wag i wektora wejściowego
- przy założeniu normalizacji wektorów wejściowych po pobudzeniu sieci wektorem x zwycięża we współzawodnictwie neuron, którego wagi najmniej różnią się od odpowiednich składowych tego wektora (neuron, który dla danego wektora wejściowego ma najwyższy poziom aktywacji)

  Zwycięzca, neuron w-ty, spełnia relacje: $$d(\mathbf{x}, \mathbf{w}_w) = \min_{1 \leqslant i \leqslant n} d(\mathbf{x}, \mathbf{w}_i)$$

  gdzie d(x,w) oznacza odległość w sensie
  wybranej metryki między wektorem x i wektorem w, a n jest liczbą neuronów.

- W uczeniu stosuje się strategię WTA(Winner Takes All) (użytą w projekcie) i WTM(Winner Takes Most)

- Zwycięski neuron, ten dla którego odległość między wektorem wag a wektorem wejściowym jest najmniejsza podlega adaptacji, zmieniając swój wektor wag w kierunku wektora x, zgodnie z regułą Kohonena:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \eta_i(k)[\mathbf{x} - \mathbf{w}_i(k)]$$

  gdzie η(k) jest współczynnikiem uczenia neuronu z sąsiedztwa S(k) w k-tej chwili.

3. Dane uczące i testujące

Dane uczące i testujące to to dane zawierające numeryczny opis cech kwiatów.

Fisher's *Iris* Data [hide]

Dataset Order	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	*I. setosa*
2	4.9	3.0	1.4	0.2	*I. setosa*
3	4.7	3.2	1.3	0.2	*I. setosa*
4	4.6	3.1	1.5	0.2	*I. setosa*
5	5.0	3.6	1.4	0.3	*I. setosa*
6	5.4	3.9	1.7	0.4	*I. setosa*
7	4.6	3.4	1.4	0.3	*I. setosa*
8	5.0	3.4	1.5	0.2	*I. setosa*
9	4.4	2.9	1.4	0.2	*I. setosa*
10	4.9	3.1	1.5	0.1	*I. setosa*
11	5.4	3.7	1.5	0.2	*I. setosa*
12	4.8	3.4	1.6	0.2	*I. setosa*
13	4.8	3.0	1.4	0.1	*I. setosa*
14	4.3	3.0	1.1	0.1	*I. setosa*
15	5.8	4.0	1.2	0.2	*I. setosa*
16	5.7	4.4	1.5	0.4	*I. setosa*
17	5.4	3.9	1.3	0.4	*I. setosa*
18	5.1	3.5	1.4	0.3	*I. setosa*
19	5.7	3.8	1.7	0.3	*I. setosa*
20	5.1	3.8	1.5	0.3	*I. setosa*

Wykorzystano zestaw z: https://en.wikipedia.org/wiki/Iris_flower_data_set

W programie liczba danych uczących i testujących wyniosła po 75 danych, po 25 z każdego gatunku kwiatu w zestawie.

4. Analiza

Sieć uczyłam i testowałam z wykorzystaniem 6 współczynników uczenia. Uczenie przebiegało podczas 1000 iteracji. Podczas testowania wyświetlałam wyniki dla całego zestawu testującego. 25 kolejnych wyników należy do jednej grupy. Jeżeli wszystkie z tych kolejnych wyników należą do danej grupy to otrzymujemy tą samą liczbę, Jeśli otrzymaliśmy różne liczby, oznacza to błąd uczenia. Został aktywowany niewłaściwy neuron. W żadnym teście nie otrzymałam 100% poprawności, zdarzały się poszczególne przypadki o obrębie jednego gatunku kwiatu. W niżej przedstawionych tabelkach zebrałam informacje: ile wystąpiło błędnych wskazań dla każdego z gatunków przy różnych współczynnikach uczenia oraz liczbę powstałych grup(liczba ta powinna wynosić 3 ponieważ tyle gatunków kwiatów mamy). Najczęściej występującym błędem był problem z przyporządkowaniem kwiatów z gatunku *I.virginica.* Najmniej błędnych wskazań otrzymałam dla gatunku *I.setosa*. Najbardziej optymalne wyniki, najmniej błędów i najmniejszą liczbę grup otrzymałam dla współczynników uczenia 0,2 i 0,5.

Przykładowe zrzuty podczas testowania:

```
Uczenie:
Liczba iteracji: 100
Testowanie
1. Flower: I.setosa  result: 11
2. Flower: I.setosa  result: 9
3. Flower: I.setosa  result: 11
4. Flower: I.setosa  result: 11
5. Flower: I.setosa  result: 11
6. Flower: I.setosa  result: 11
7. Flower: I.setosa  result: 9
8. Flower: I.setosa  result: 11
9. Flower: I.setosa  result: 9
10. Flower: I.setosa  result: 11
11. Flower: I.setosa  result: 11
12. Flower: I.setosa  result: 11
13. Flower: I.setosa  result: 11
14. Flower: I.setosa  result: 11
15. Flower: I.setosa  result: 11
16. Flower: I.setosa  result: 9
17. Flower: I.setosa  result: 11
18. Flower: I.setosa  result: 9
19. Flower: I.setosa  result: 9
20. Flower: I.setosa  result: 9
21. Flower: I.setosa  result: 11
22. Flower: I.setosa  result: 9
23. Flower: I.setosa  result: 11
24. Flower: I.setosa  result: 11
25. Flower: I.setosa  result: 11
```

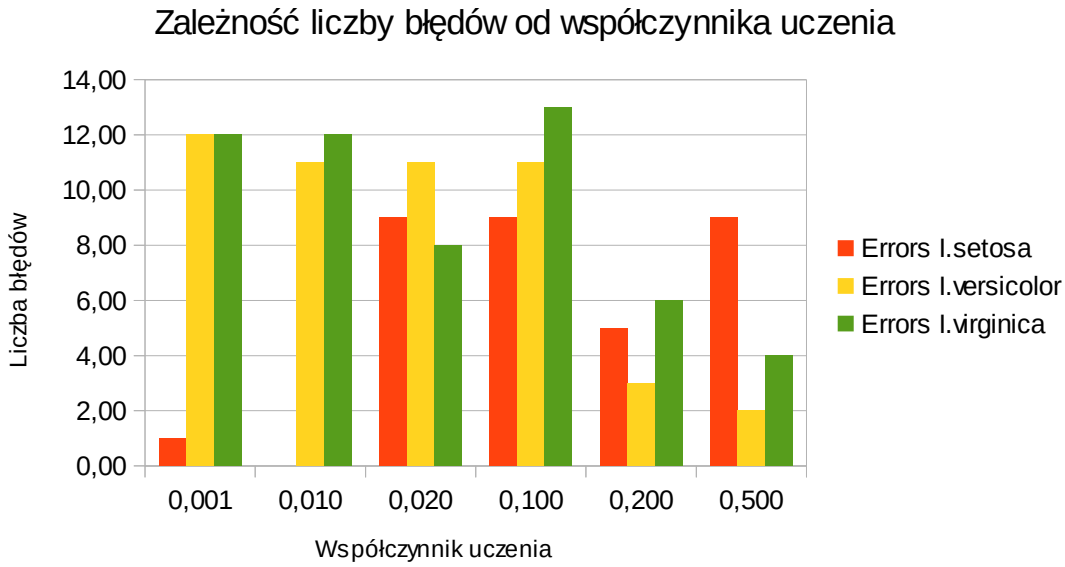
```
26. Flower: I.versicolor  result: 12
27. Flower: I.versicolor  result: 1
28. Flower: I.versicolor  result: 8
29. Flower: I.versicolor  result: 8
30. Flower: I.versicolor  result: 12
31. Flower: I.versicolor  result: 12
32. Flower: I.versicolor  result: 12
33. Flower: I.versicolor  result: 12
34. Flower: I.versicolor  result: 8
35. Flower: I.versicolor  result: 8
36. Flower: I.versicolor  result: 12
37. Flower: I.versicolor  result: 12
38. Flower: I.versicolor  result: 1
39. Flower: I.versicolor  result: 12
40. Flower: I.versicolor  result: 12
41. Flower: I.versicolor  result: 8
42. Flower: I.versicolor  result: 8
43. Flower: I.versicolor  result: 12
44. Flower: I.versicolor  result: 12
45. Flower: I.versicolor  result: 8
46. Flower: I.versicolor  result: 8
47. Flower: I.versicolor  result: 8
48. Flower: I.versicolor  result: 12
49. Flower: I.versicolor  result: 12
50. Flower: I.versicolor  result: 12
51. Flower: I.versicolor  result: 8
```

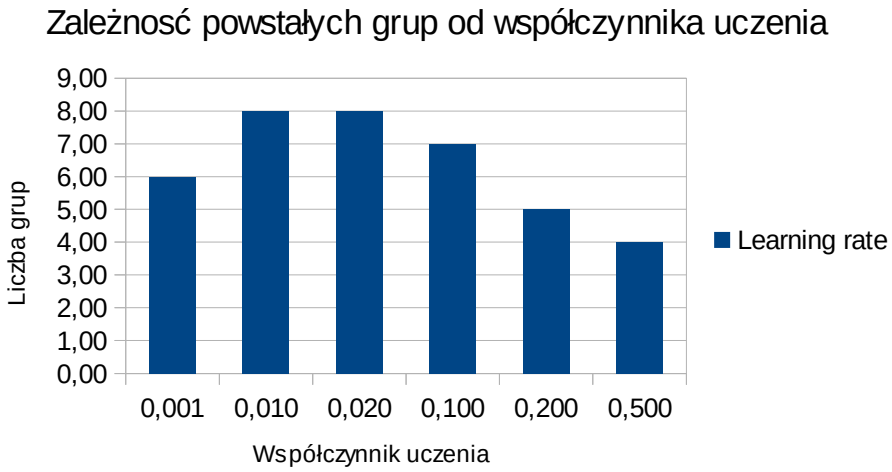
```
52. Flower: I.virginica  result: 12
53. Flower: I.virginica  result: 8
54. Flower: I.virginica  result: 10
55. Flower: I.virginica  result: 1
56. Flower: I.virginica  result: 1
57. Flower: I.virginica  result: 8
58. Flower: I.virginica  result: 10
59. Flower: I.virginica  result: 8
60. Flower: I.virginica  result: 1
61. Flower: I.virginica  result: 1
62. Flower: I.virginica  result: 10
63. Flower: I.virginica  result: 8
64. Flower: I.virginica  result: 8
65. Flower: I.virginica  result: 10
66. Flower: I.virginica  result: 10
67. Flower: I.virginica  result: 12
68. Flower: I.virginica  result: 10
69. Flower: I.virginica  result: 10
70. Flower: I.virginica  result: 10
71. Flower: I.virginica  result: 10
72. Flower: I.virginica  result: 1
73. Flower: I.virginica  result: 10
74. Flower: I.virginica  result: 10
75. Flower: I.virginica  result: 8
```

```
Testowanie
1. Flower: I.setosa  result: 7
2. Flower: I.setosa  result: 7
3. Flower: I.setosa  result: 7
4. Flower: I.setosa  result: 7
5. Flower: I.setosa  result: 7
6. Flower: I.setosa  result: 7
7. Flower: I.setosa  result: 7
8. Flower: I.setosa  result: 7
9. Flower: I.setosa  result: 7
10. Flower: I.setosa  result: 7
11. Flower: I.setosa  result: 7
12. Flower: I.setosa  result: 7
13. Flower: I.setosa  result: 7
14. Flower: I.setosa  result: 7
15. Flower: I.setosa  result: 7
16. Flower: I.setosa  result: 7
17. Flower: I.setosa  result: 7
18. Flower: I.setosa  result: 7
19. Flower: I.setosa  result: 7
20. Flower: I.setosa  result: 7
21. Flower: I.setosa  result: 7
22. Flower: I.setosa  result: 7
23. Flower: I.setosa  result: 7
24. Flower: I.setosa  result: 7
25. Flower: I.setosa  result: 7
```

Learning rate	Errors I.setosa	Errors I.versicolor	Errors I.virginica
0,001	1,00	12,00	12,00
0,01	0,00	11,00	12,00
0,02	9,00	11,00	8,00
0,10	9,00	11,00	13,00
0,20	5,00	3,00	6,00
0,50	9,00	2,00	4,00

Learning rate	Liczba grup
0,001	6,00
0,01	8,00
0,02	8,00
0,10	7,00
0,20	5,00
0,50	4,00





8. Wnioski

Na podstawie analizy i wykresów można stwierdzić że najbardziej optymalnymi współczynnikami uczenia były 0,2 i 0,5. Błędy w rozpoznawaniu mogły wynikać z niepoprawnej normalizacji danych. Powodem błędów mógł być również sposób rozmieszczenia gatunków w danych wejściowych. Sieć ma również problem z odróżnieniem podobnych danych, gatunki *I.versicolor* i *I.virginca* często były przyporządkowywane do jednej grupy.

9. Listing kodu

```
public class KohonenWTA {
    private int noNeurons = 20;
    private int noInputs = 75;      //liczba danych wejściowych uczących i testujących
    private String[] flowerType;
    public static double LEARNING_RATE=0.2; //współczynnik uczenia
    private static int maxIter = 1000;
    //private String[] testflowerType;
    private double[][]weights;      //tablica wag

    public double[][] learnFlowerSet = {{5.1,3.5,1.4,0.2},      //dane uczące
            {4.9,3.0,1.4,0.2},
            {4.7,3.2,1.3,0.2},
            {4.6,3.1,1.5,0.2},
            {5.0,3.6,1.4,0.3},
            {5.4,3.9,1.7,0.4},
            {4.6,3.4,1.4,0.3},
            {5.0,3.4,1.5,0.2},



   public double[][] testFlowerSet = {{5.0,3.0,1.6,0.2},       //dane testujące
            {5.0,3.4,1.6,0.4},
            {5.2,3.5,1.5,0.2},
            {5.2,3.4,1.4,0.2},
            {4.7,3.2,1.6,0.2},
            {4.8,3.1,1.6,0.2},
            {5.4,3.4,1.5,0.4},
            {5.2,4.1,1.5,0.1},
            {5.5,4.2,1.4,0.2},
            {4.9,3.1,1.5,0.2},
            {5.0,3.2,1.2,0.2},
            {5.5,3.5,1.3,0.2},
            {4.9,3.6,1.4,0.1},


     //funkcja w której następuje uczenie
     public void startKohonen(){

     //tablica z nazwami kwiatów z zestawu uczącego i testującego
         flowerType = new String[noInputs];
         for(int i=0; i<noInputs; i++){
             if(i<25){
                 flowerType[i] = "I.setosa";
             }
             if(i>=25 && i<51){
                 flowerType[i] = "I.versicolor";
             }
             if(i>=51){
                 flowerType[i] = "I.virginica";
             }
         }
```

```
            weights = new double[noNeurons][4];
            randWeight();
            normalizeLearningSet(); //normalizacja danych
            int iter = 0;
            double[] result = new double[noNeurons];
            System.out.println("Uczenie: ");
            do{
                for(int i=0; i<noInputs; i++){
                    for(int j=0; j<noNeurons; j++){
                        //obliczanie sumy połączenia na wejściu
                        result[j] = calculateInput(j,i);
                    }
                    //korekta wag
                    updateWeights(winner(result),i);
                }
                iter++;

            }while(iter<maxIter);
            System.out.println("Liczba iteracji: " + iter);
        }


    //funkcja w której następuje testowanie
    public void testKohonen() {

        System.out.println("Testowanie");
        normalizeTestSet();
        double[] result = new double[noNeurons];
        int winner;

        for(int i=0; i< noInputs; i++) {
            for(int j=0;j<noNeurons;j++)
            {
                result[j]=calculateTestInput(j,i);
            }
            winner = winner(result); //zwycięzca, otrzymał największy wynik przy aktywacji

            System.out.println(i+1 + ". Flower: " + flowerType[i] + "  result: " + winner );
        }

//ustawienie poczatkowych wartości wag
public void randWeight(){
    Random random = new Random();
    for(int i =0;i<noNeurons;i++) {
        for (int j = 0; j < 4; j++)
            weights[i][j] = random.nextDouble();
    }
}

//funkcja normalizująca
public void normalizeFlowerSet(){
    double min_elem = 0.0;
    double max_elem = 0.0;
    min_elem = min(learnFlowerSet);
    max_elem = max(learnFlowerSet);

    for(int i =0 ; i<noInputs ; i++){
        for(int j=0;j<4;j++)            {
            learnFlowerSet[i][j] = (learnFlowerSet[i][j] - min_elem)/(max_elem - min_elem);
        }
    }
}
```

```
  //normalizuja danych uczących
    public void normalizeLearningSet() {
        double suma = 0.0;
        double Sqrt = 0.0;
        for(int i =0 ; i<noInputs ; i++){
            for(int j=0;j<4;j++){
                suma +=learnFlowerSet[i][j]*learnFlowerSet[i][j];
            }
           Sqrt=Math.sqrt(suma);
           for(int j=0;j<4;j++){
                learnFlowerSet[i][j]/=Sqrt;
            }
            suma =0.0;
        }
    }
    //normalizuja danych testujących
    public void normalizeTestSet() {
        double suma = 0.0;
        double Sqrt = 0.0;
        for(int i =0 ; i<noInputs ; i++){
            for(int j=0;j<4;j++){
                suma +=testFlowerSet[i][j]*testFlowerSet[i][j];
            }
           Sqrt=Math.sqrt(suma);
           for(int j=0;j<4;j++){
                testFlowerSet[i][j]/=Sqrt;
            }
            suma =0.0;
        }
```

```
public double calculateInput(int j, int k){
    double activate = 0;
    for(int i = 0; i < 4; i++) {
        activate += learnFlowerSet[k][i] * weights[j][i];
    }
    return activate;
}

public double calculateTestInput(int j, int k){
    double activate = 0;
    for(int i = 0; i < 4; i++) {
        activate += testFlowerSet[k][i] * weights[j][i];
    }
    return activate;
}

//zwraca id elementu tablicy -> ten neuron ma największą wartość aktywacji
public int winner(double[] result){
    double input = result[0];
    int id = 0;
    for(int i=1; i<result.length; i++){
        if(input<result[i]){
            input = result[i];
            id = i;
        }
    }
    return id;
}
```

```
public double max(double[][] result){
    double input = result[0][0];
    for(int i=1; i<noInputs; i++){
        for(int j=1; j<4;j++)
        if(input<result[i][j]){
            input = result[i][j];

        }
    }
    return input;
}

public double min(double[][] result){
    double input = result[0][0];
    for(int i=1; i<noInputs; i++){
        for(int j=1; j<4;j++)
        if(input>result[i][j]){
            input = result[i][j];
        }
    }
    return input;
}

public void updateWeights(int j,int l){
    for (int i = 0; i < 4; i++) {
        weights[j][i] = weights[j][i] + LEARNING_RATE * (learnFlowerSet[l][i] - weights[j][i]);
}
}
```