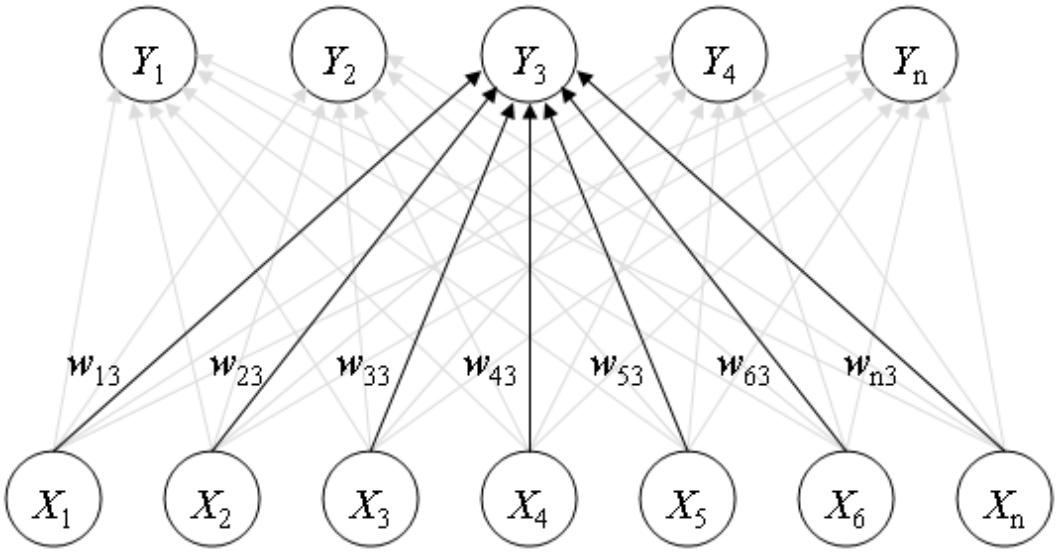
Daniel Wenecki
Podstawy Sztucznej Inteligencji
Sprawozdanie – Projekt nr 6

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

2. Syntetyczny opis budowy użytej sieci i algorytmu uczenia
Do wykonania ćwiczenia użyłem sieci Kohonena przy użycie reguły WTM.

Schemat sieci:



Schemat modyfikacji wag:

$$W_v(s+1) = W_v(s) + \theta(u,v,s) \cdot \alpha(s) \cdot (D(t) - W_v(s)),$$

Dokładniej funkcja modyfikacji wag i funkcja sąsiedztwa w matlabie ma postać:

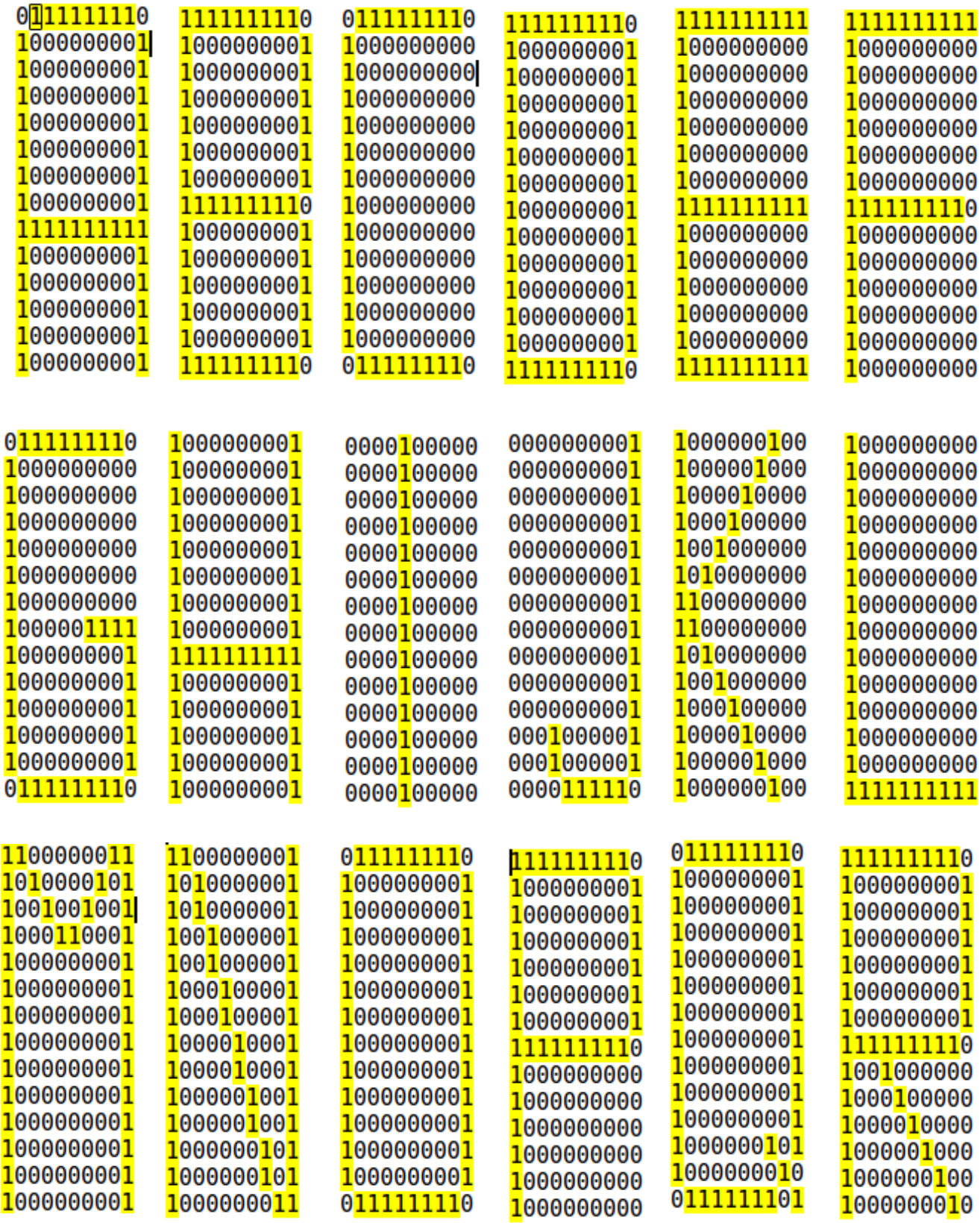
```
dw = lr*a2*(p'-w)

a2(i,q) = 1,  if a(i,q) = 1
              = 0.5, if a(j,q) = 1 and D(i,j) <= nd
              = 0, otherwise
```

Odległość pomiędzy punktami znajdowałem za pomocą funkcji:

$$d(A,B) = \sqrt{(x_{1A} - x_{1B})^2 + (x_{2A} - x_{2B})^2 + \ldots + (x_{nA} - x_{nB})^2} = \sqrt{\sum_{i=1}^{n}((x_{iA} - x_{iB})^2)}$$

Jako dane uczące użyłem 20 dużych liter alfabetu odwzorowanych na tablicach 14x10 w postaci zer i jedynek.

Dane uczące:

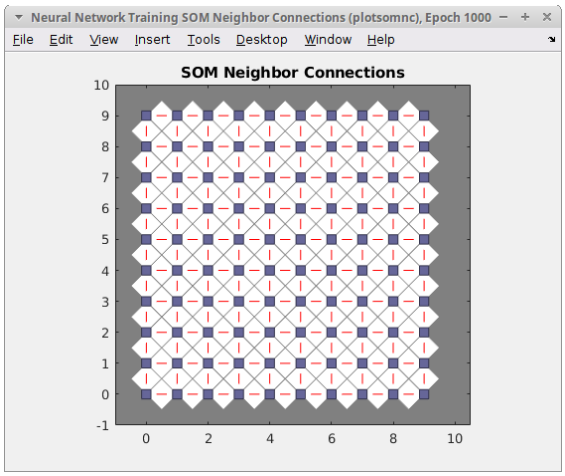

0111111110 1111111110 0111111110 1111111110 1111111111 1111111111
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1111111110 1000000000 1000000001 1111111111 1111111110
1111111111 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1000000001 1000000000 1000000001 1000000000 1000000000
1000000001 1111111110 0111111110 1111111110 1111111111 1000000000
0111111110 1000000001 0000100000 0000000001 1000000100 1000000000
1000000000 1000000001 0000100000 0000000001 1000001000 1000000000
1000000000 1000000001 0000100000 0000000001 1000010000 1000000000
1000000000 1000000001 0000100000 0000000001 1000100000 1000000000
1000000000 1000000001 0000100000 0000000001 1001000000 1000000000
1000000000 1000000001 0000100000 0000000001 1010000000 1000000000
1000001111 1000000001 0000100000 0000000001 1100000000 1000000000
1000000001 1111111111 0000100000 0000000001 1100000000 1000000000
1000000001 1000000001 0000100000 0000000001 1010000000 1000000000
1000000001 1000000001 0000100000 0000000001 1001000000 1000000000
1000000001 1000000001 0000100000 0000000001 1000100000 1000000000
1000000001 1000000001 0000100000 0001000001 1000010000 1000000000
1000000001 1000000001 0000100000 0001000001 1000001000 1000000000
0111111110 1000000001 0000100000 0000111110 1000000100 1111111111
1100000011 1100000001 0111111110 1111111110 0111111110 1111111110
1010000101 1010000001 1000000001 1000000001 1000000001 1000000001
1001001001 1010000001 1000000001 1000000001 1000000001 1000000001
1000110001 1001000001 1000000001 1000000001 1000000001 1000000001
1000000001 1001000001 1000000001 1000000001 1000000001 1000000001
1000000001 1000100001 1000000001 1000000001 1000000001 1000000001
1000000001 1000100001 1000000001 1000000001 1000000001 1000000001
1000000001 1000010001 1000000001 1111111110 1000000001 1111111110
1000000001 1000010001 1000000001 1000000000 1000000001 1001000000
1000000001 1000001001 1000000001 1000000000 1000000001 1000100000
1000000001 1000001001 1000000001 1000000000 1000000001 1000010000
1000000001 1000000101 1000000001 1000000000 1000000001 1000001000
1000000001 1000000101 1000000001 1000000000 1000000010 1000000100
1000000001 1000000011 0111111110 1000000000 0111111101 1000000010

```
1111111111   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   1000000001
0000110000   0111111110
```

Zestaw testowy, polegał na dodanie jednego losowego piksela w wolnym miejscu w tablicy dla każdej litery.
Do stworzenia sieci użyłem programu Matlab i narzędzia nntool.

3. Zestawienie otrzymanych wyników oraz ich analiza

Do testów użyłem siatki 10 na 10, w której modyfkiowałem współczynniki uczenia i promień początkowy.
Użyta siatka:



Notowałem ilość błędnych przypasowań liter po 1000 iteracji.

Uzyskane wyniki:

r\lr	0,5	0,1	0,01
5	2,1	3,2	6,7
1	0	0,1	3,5
0,1	0	0,3	10,3

Jak widzimy odpowiednie dostosowanie początkowego promienia i współczynnika uczenia ma bardzo duży wpływ na to jak sieć się nauczy.

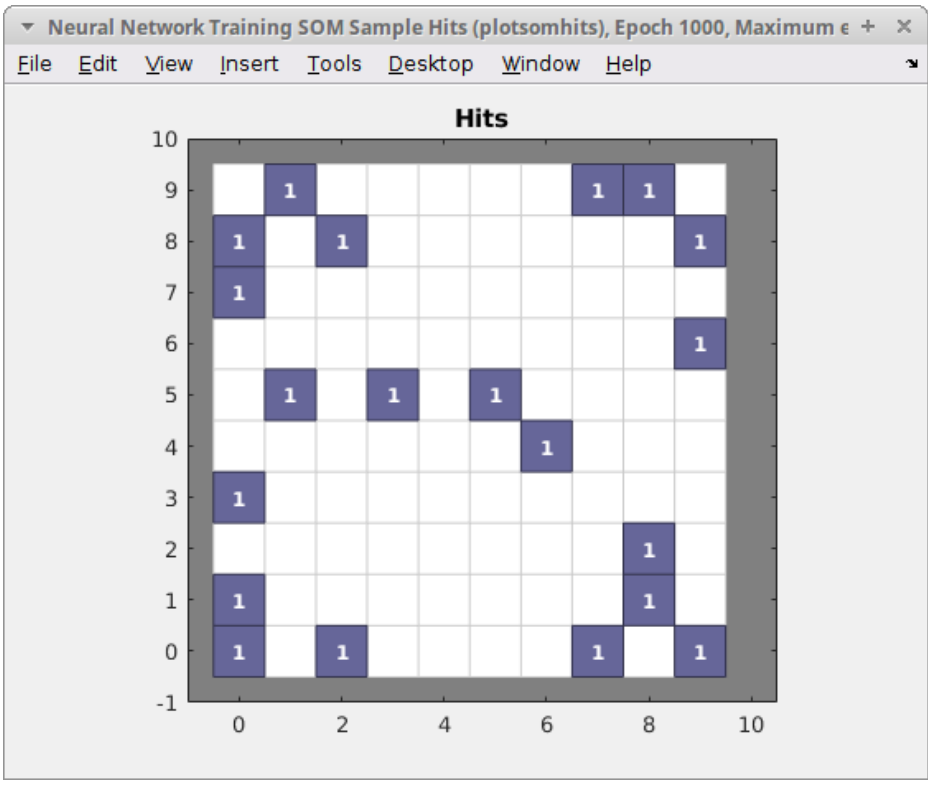
Dobranie zbyt małego współczynnika uczenia oznacza że sieć się będzie długo uczyła, dlatego uzyskaliśmy takie błędy w przypadku lr ustawionego na 0,01 – sieć się nie zdążyłą nauczyć.

Z drugiej strony dobranie tego współczynnika zbyt dużego może sprawić że sieć zacznie oscylować i nie znajdzie dobrego wyniku.

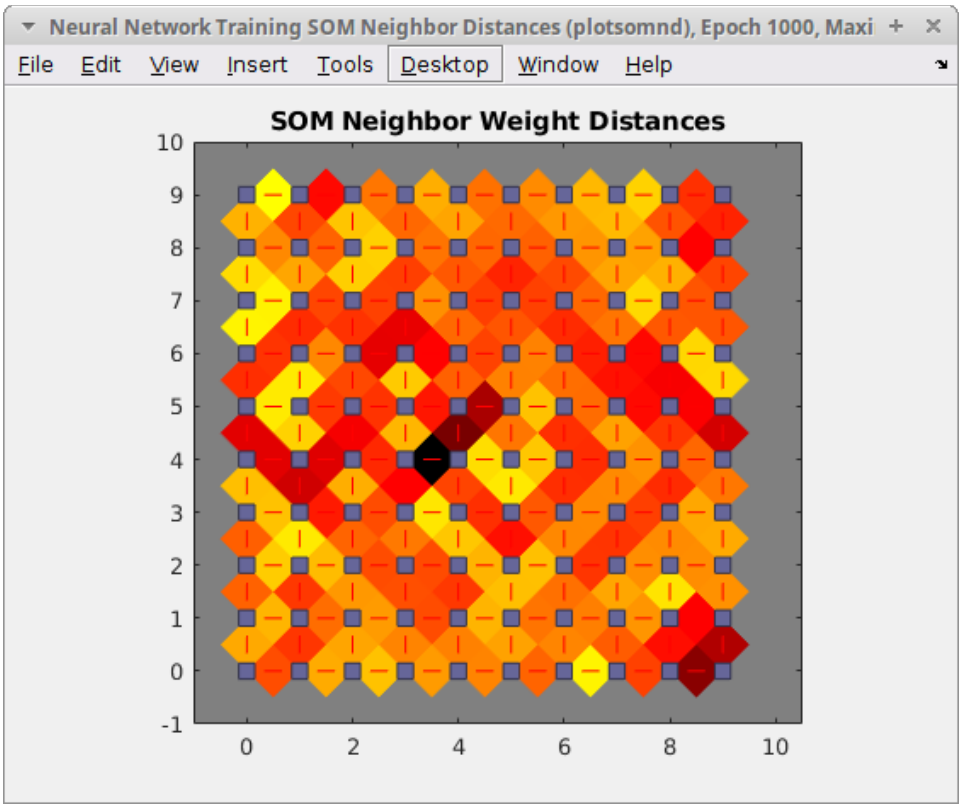
Bardzo ważnym czynnikiem jest też promień początkowy. Jak widzimy przy wartości zbyt dużej tego promienia sieć nie jest w stanie się prawidłowo rozróżniać litery, wagi zbyt wielu neuronów są modyfikowane. Przy wartości zbyt małej błąd odpowiedniego ułożenia sieci również zwrasta.

Dla poprawnie nauczonej sieci użyłem wcześniej wygenerowanych danych testowych, sieć nie pomyliła się w nich nigdzie.

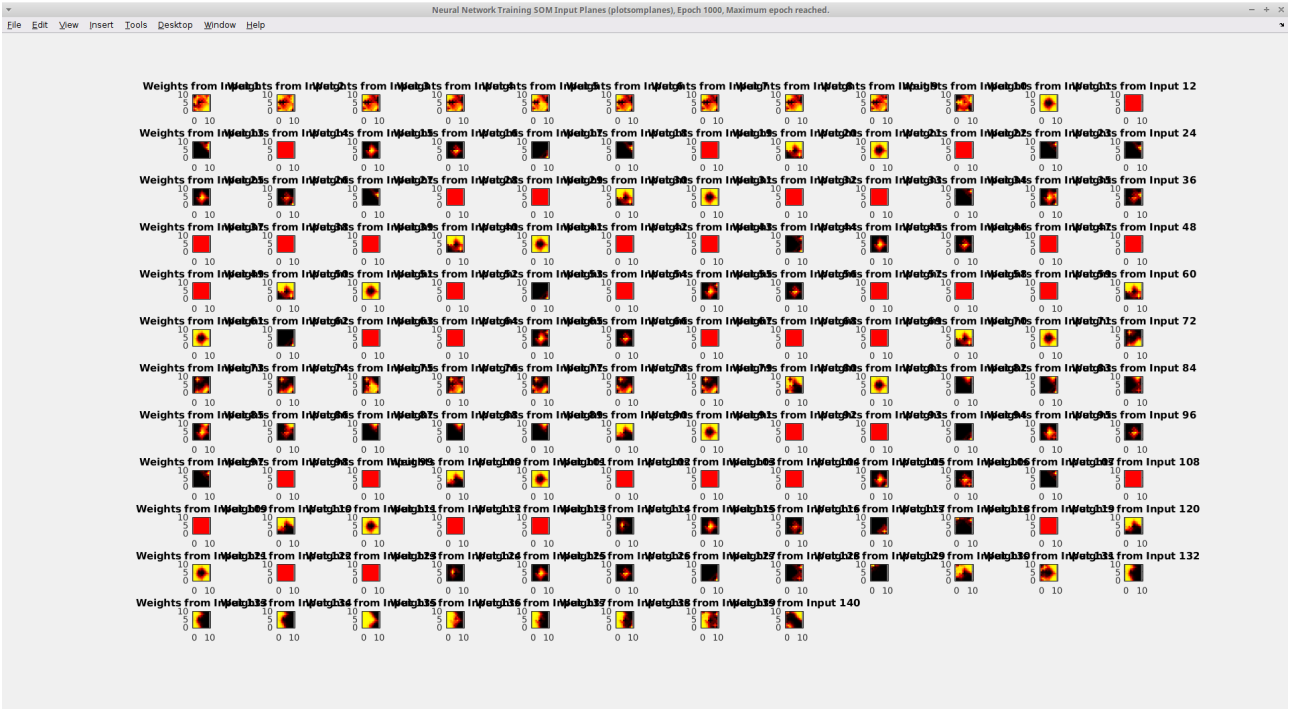
Nauczona sieć:



Jak widać w nauczonej sieci każda litera jest rozpoznawana przez inny neuron.



Powyzsza mapa obrazuja nam odleglości pomiędzy neuronami, jeśli dwie litery rozpoznawane przez neurony maja między soba dość male odleglosci mozemy to interpretować tym, ze te litery są podobne.



Na powyższych wykresach możemy zaobserwować znaczenie każdych wag dla każdego neuronu. Jak widać wagi dla pól w tablicy, które były puste dla każdej litery pozostają jednakowe dla wszystkich neruonów.

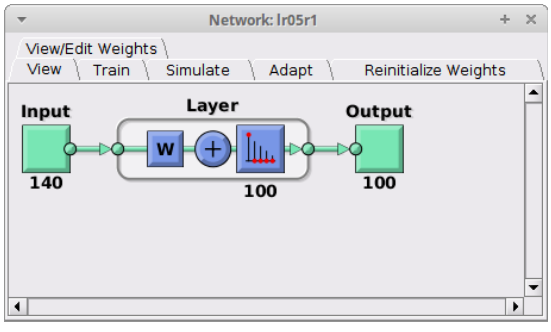
4.Wnioski

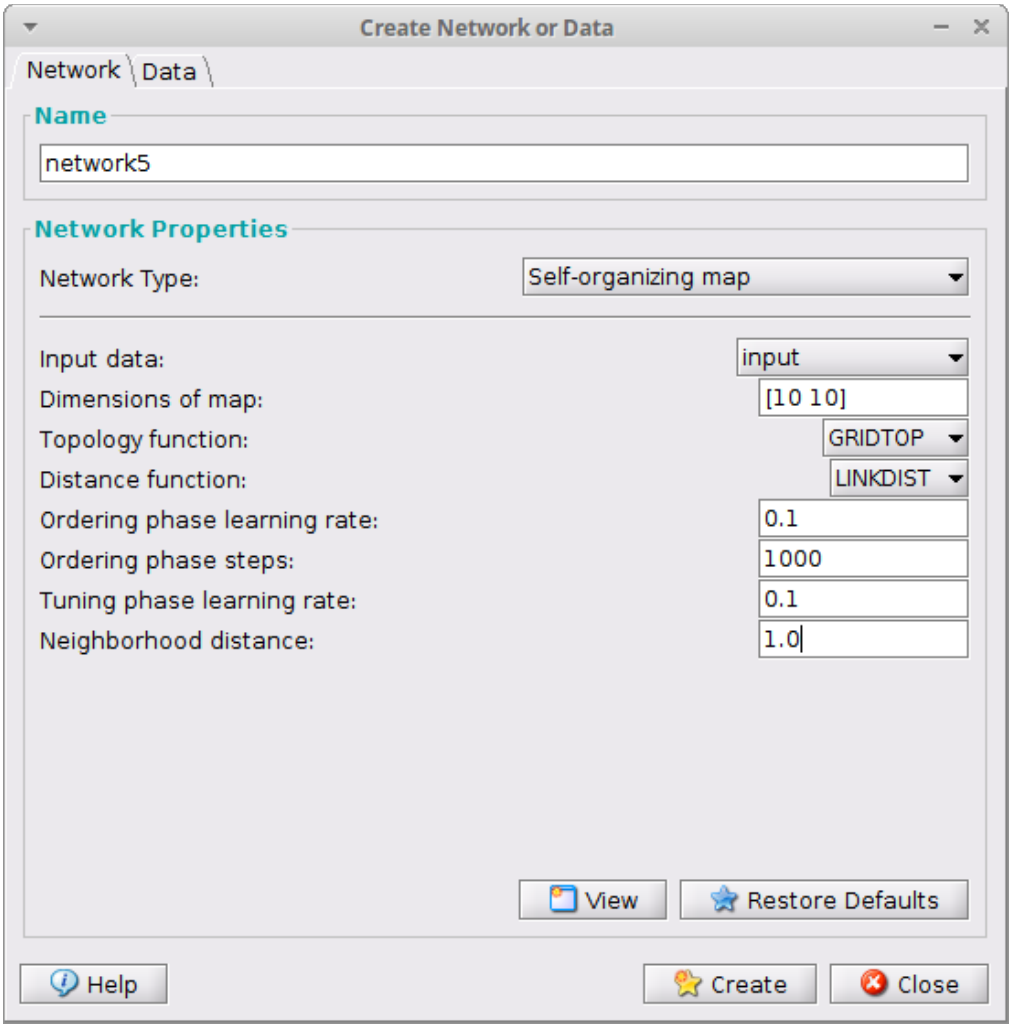
Wybierając learning rate, powinniśmy wybrać optymalną wartość. Wartość lr zbyt duża spowoduje niedokładność lub w ogóle nie możliwość nauczenia się sieci, wartość zbyt mała bardzo wydłuży czas uczenia.

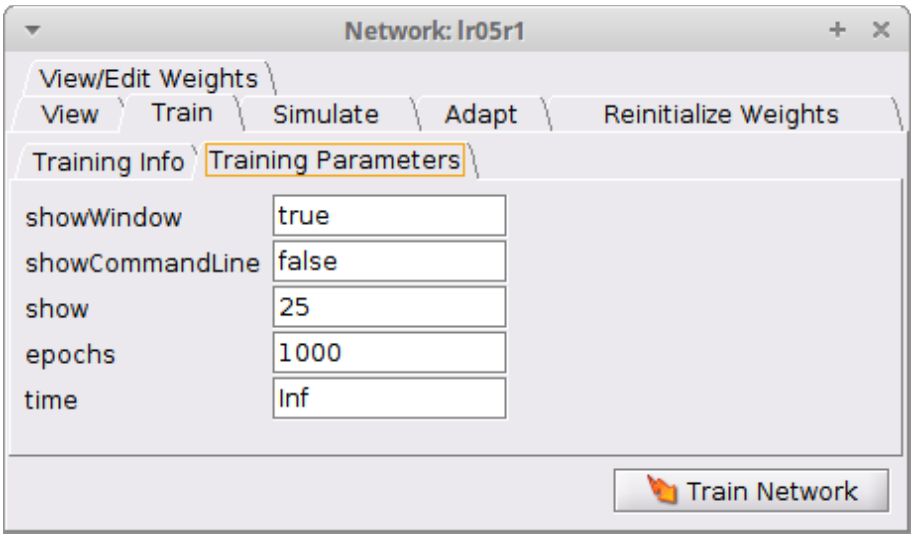
Dobranie odpowieniego promieniu poczatkowego jest również bardzo ważne. Dobranie go zbyt dużego spowoduje że modyfikacja wag obejmie zbyt wiele neuronów i sieć nie będzie się w stanie nauczyć. Dobranie zbyt małej wartości sprawi, że sieć będzie się uczyła niemal w taki sam sposób jak przy metodzie WTA.

Jak widać po wartości istotności każdej wagi dla neuronów poprawność sieci po zaszumieniu danych zależała by od pól w których to zaszumienie występowało. Dla zaszumienia w niektórych polach, wynik praktycznie by się nie zmienił, lecz dla zaszumieniach w polach istotnych sieć mogłaby pomylić.

5. Screeny konfiguracji programu i listing kodu:








```java
import java.util.ArrayList;
import java.util.List;
/**
 * Created by Daniel on 2017-10-03.
 */
public class Alfabet {//tworzenie danych wejsciowych w postaci liter
    Litera litA = new Litera();
    Litera litB = new Litera();
    Litera litC = new Litera();
```

```
Litera litD = new Litera();
Litera litE = new Litera();
Litera litF = new Litera();
Litera litH = new Litera();
Litera litI = new Litera();
Litera litL = new Litera();
Litera litO = new Litera();
Litera litG = new Litera();
Litera litJ = new Litera();
Litera litK = new Litera();
Litera litM = new Litera();
Litera litN = new Litera();
Litera litP = new Litera();
Litera litQ = new Litera();
Litera litR = new Litera();
Litera litT = new Litera();
Litera litU = new Litera();
List<Litera> litery = new ArrayList<Litera>();
Alfabet()
{
    //A
    for(int i=1;i<14;i++)
        litA.tab[i][0] = 1;
    for(int i=1;i<14;i++)
        litA.tab[i][9] = 1;
    for(int i=1;i<9;i++)
        litA.tab[0][i] = 1;
    for(int i=1;i<9;i++)
        litA.tab[8][i] =1;
    //B
    for(int i=1;i<14;i++)
        litB.tab[i][0] = 1;
    for(int i=1;i<14;i++)
        litB.tab[i][9] = 1;
    for(int i=0;i<9;i++)
        litB.tab[0][i] = 1;
    for(int i=1;i<9;i++)
        litB.tab[7][i] =1;
    for(int i=1;i<9;i++)
        litB.tab[13][i] =1;
    litB.tab[7][9] =0;
    litB.tab[13][9] = 0;
    //C
    for(int i=1;i<13;i++)
        litC.tab[i][0] = 1;
    for(int i=1;i<9;i++)
        litC.tab[0][i] = 1;
    for(int i=1;i<9;i++)
        litC.tab[13][i] = 1;
    //D
    for(int i=0;i<14;i++)
        litD.tab[i][0] = 1;
    for(int i=1;i<9;i++)
```

```
    litD.tab[0][i] = 1;
for(int i=1;i<9;i++)
    litD.tab[13][i] = 1;
for(int i=1;i<13;i++)
    litD.tab[i][9] = 1;
//E
for(int i=0;i<14;i++)
    litE.tab[i][0] = 1;
for(int i=1;i<10;i++)
    litE.tab[0][i] = 1;
for(int i=1;i<10;i++)
    litE.tab[13][i] = 1;
for(int i=1;i<10;i++)
    litE.tab[7][i] = 1;
//F
for(int i=0;i<14;i++)
    litF.tab[i][0] = 1;
for(int i=1;i<10;i++)
    litF.tab[0][i] = 1;
for(int i=1;i<9;i++)
    litF.tab[7][i] = 1;
//G
for(int i=1;i<13;i++)
    litG.tab[i][0] = 1;
for(int i=7;i<13;i++)
    litG.tab[i][9] = 1;
for(int i=1;i<9;i++)
    litG.tab[13][i] = 1;
for(int i=1;i<9;i++)
    litG.tab[0][i] = 1;
litG.tab[7][8] =1;
litG.tab[7][7] =1;
litG.tab[7][6] =1;
//H
for(int i=0;i<14;i++)
    litH.tab[i][0] = 1;
for(int i=0;i<14;i++)
    litH.tab[i][9] = 1;
for(int i=1;i<10;i++)
    litH.tab[8][i] = 1;
//I
for(int i=0;i<14;i++)
    litI.tab[i][4] = 1;
//J
for(int i=0;i<13;i++)
    litJ.tab[i][9] =1;
for(int i=4;i<9;i++)
    litJ.tab[13][i] =1;
litJ.tab[11][3] =1;
litJ.tab[12][3] =1;
//K
for(int i=0;i<14;i++)
    litK.tab[i][0] =1;
```

```
for(int i=0;i<7;i++)
{
    litK.tab[6-i][i+1] = 1;
    litK.tab[7+i][i+1] = 1;
}
//L
for(int i=0;i<14;i++)
    litL.tab[i][0] = 1;
for(int i=1;i<10;i++)
    litL.tab[13][i] = 1;
//M
for(int i=0;i<14;i++) {
    litM.tab[i][0] = 1;
    litM.tab[i][9] = 1;
}
for(int i=0;i<4;i++)
{
    litM.tab[i][i+1] = 1;
    litM.tab[i][8-i] = 1;
}
//N
for(int i=0;i<14;i++) {
    litN.tab[i][0] = 1;
    litN.tab[i][9] = 1;
}
litN.tab[0][1] = 1;
litN.tab[13][8] = 1;
for(int i=0;i<6;i++){
    litN.tab[2*i+1][i+2] =1;
    litN.tab[2*i+2][i+2] =1;
}
//O
for(int i=1;i<13;i++)
    litO.tab[i][0] = 1;
for(int i=1;i<13;i++)
    litO.tab[i][9] = 1;
for(int i=1;i<9;i++)
    litO.tab[13][i] = 1;
for(int i=1;i<9;i++)
    litO.tab[0][i] = 1;
//P
for(int i=0;i<14;i++)
    litP.tab[i][0] = 1;
for(int i=0;i<9;i++)
    litP.tab[0][i] = 1;
for(int i=1;i<9;i++)
    litP.tab[7][i] =1;
for(int i=1;i<7;i++)
    litP.tab[i][9] =1;
//Q
for(int i=1;i<13;i++)
    litQ.tab[i][0] = 1;
for(int i=1;i<12;i++)
```

```
    litQ.tab[i][9] = 1;
for(int i=1;i<8;i++)
    litQ.tab[13][i] = 1;
for(int i=1;i<9;i++)
    litQ.tab[0][i] = 1;
litQ.tab[12][8] =1;
litQ.tab[13][9] =1;
litQ.tab[11][7] =1;
//R
for(int i=0;i<14;i++)
    litR.tab[i][0] = 1;
for(int i=0;i<9;i++)
    litR.tab[0][i] = 1;
for(int i=1;i<9;i++)
    litR.tab[7][i] =1;
for(int i=1;i<7;i++)
    litR.tab[i][9] =1;
for(int i=2;i<9;i++)
    litR.tab[i+5][i] = 1;
//T
for(int i=0;i<14;i++)
    litT.tab[i][5] = 1;
for(int i=0;i<14;i++)
    litT.tab[i][4] = 1;
for(int i=0;i<10;i++)
    litT.tab[0][i] = 1;
//U
for(int i=0;i<13;i++)
    litU.tab[i][0] = 1;
for(int i=0;i<13;i++)
    litU.tab[i][9] = 1;
for(int i=1;i<9;i++)
    litU.tab[13][i] = 1;
litery.add(litA);
litery.add(litB);
litery.add(litC);
litery.add(litD);
litery.add(litE);
litery.add(litF);
litery.add(litG);
litery.add(litH);
litery.add(litI);
litery.add(litJ);
litery.add(litK);
litery.add(litL);
litery.add(litM);
litery.add(litN);
litery.add(litO);
litery.add(litP);
litery.add(litQ);
litery.add(litR);
litery.add(litT);
litery.add(litU);
```

```
    }
    void wyswietlAlfabet() {
        for (int i = 0; i < litery.size(); i++) {
            litery.get(i).wyswietl();
            System.out.println();
        }
    }
}

public class Litera {//litera z odpowiednimi metodami
    public int[][] tab = new int[14][10];
    int[] input = new int[6];
    Litera()
    {
        for(int i=0; i<6;i++)
            input[i] = 0;
        for(int i=0; i<14;i++)
            for(int j=0;j<10;j++)
                tab[i][j] = 0;
    }
//    public void wyswietl()
//    {
//        for(int i =0;i<14;i++) {
//            for (int j = 0; j < 10; j++)
//                System.out.print(tab[i][j]+ "\t");
//            //System.out.println();
//        }
//    }
    public void wyswietl()
    {
        for(int i =0;i<14;i++) {
            for (int j = 0; j < 10; j++)
                System.out.print(tab[i][j]);
            System.out.println();
        }
    }
    public void update()
    {
        for(int i = 0;i<2;i++)
            for(int j=0;j<7;j++)
                input[0] += tab[j][i];
        for(int i = 2;i<8;i++)
            for(int j=0;j<7;j++)
                input[1] += tab[j][i];
        for(int i = 8;i<10;i++)
            for(int j=0;j<7;j++)
                input[2] += tab[j][i];
        for(int i = 0;i<2;i++)
            for(int j=7;j<14;j++)
                input[3] += tab[j][i];
        for(int i = 2;i<8;i++)
            for(int j=7;j<14;j++)
                input[4] += tab[j][i];
        for(int i = 8;i<10;i++)
```

```
            for(int j=7;j<14;j++)
                input[5] += tab[j][i];
    }
}
```

Bibliografia:

https://en.wikipedia.org
https://www.mathworks.com/help/nnet/self-organizing-maps.html
http://mnemstudio.org/neural-networks-kohonen-self-organizing-maps.htm