

Paweł Nowak

Temat ćwiczenia:

Budowa i działanie sieci Kohonena dla WTA.

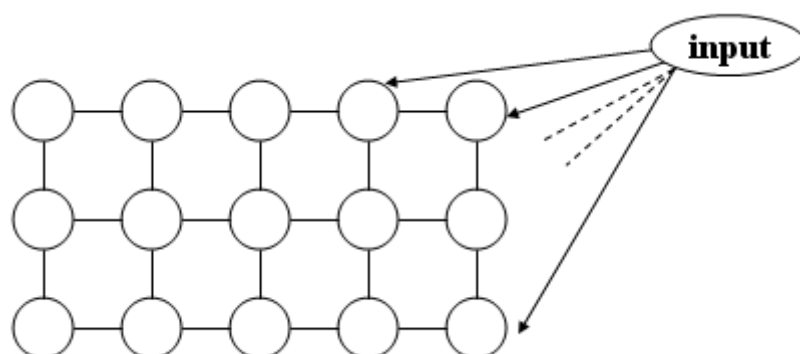
Cel ćwiczenia

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

Sieci Kohonena są szczególnym przypadkiem algorytmu realizującego uczenie się bez nadzoru. Ich głównym zadaniem jest organizacja wielowymiarowej informacji (np. obiektów opisanych 50 parametrami w taki sposób, żeby można ją było prezentować i analizować w przestrzeni o znacznie mniejszej liczbie wymiarów, czyli mapie (np. na dwuwymiarowym ekranie). Warunek: rzuty "podobnych" danych wejściowych powinny być bliskie również na mapie. Sieci Kohonena znane są też pod nazwami Self-Organizing Maps, Competitive Filters.

Topologia sieci Kohonena odpowiada topologii docelowej przestrzeni. Jeśli np. chcemy prezentować wynik na ekranie, rozsądnym modelem jest prostokątna siatka węzłów (im więcej, tym wyższą rozdzielczość będzie miała mapa):

Topologia
Kohonena



sieci
(przykład)

Zasady działania sieci Kohonena:

- Wejścia (tyle, iloma parametrami opisano obiekty) połączone są ze wszystkimi węzłami sieci
- Każdy węzeł przechowuje wektor wag o wymiarze identycznym z wektorami wejściowymi
- Każdy węzeł oblicza swój poziom aktywacji jako iloczyn skalarny wektora wag i wektora wejściowego (podobnie jak w zwykłym neuronie)
- Ten węzeł, który dla danego wektora wejściowego ma najwyższy poziom aktywacji, zostaje zwycięzcą i jest uaktywniony
- Wzmacniamy podobieństwo węzła-zwycięzcy do aktualnych danych wejściowych poprzez dodanie do wektora wag wektora wejściowego (z pewnym współczynnikiem uczenia)
- Każdy węzeł może być stowarzyszony z pewnymi innymi, sąsiednimi węzłami - wówczas te węzły również zostają zmodyfikowane, jednak w mniejszym stopniu.

Inicjalizacja wag sieci Kohonena jest losowa. Wektory wejściowe stanowią próbę uczącą, podobnie jak w przypadku zwykłych sieci rozpatrywaną w pętli podczas budowy mapy. Wykorzystanie utworzonej w ten sposób mapy polega na tym, że zbiór obiektów umieszczamy na wejściu sieci i obserwujemy, które węzły sieci się uaktywniają. Obiekty podobne powinny trafiać w podobne miejsca mapy.

Ciekawym zastosowaniem jest próba wykorzystania sieci Kohonena w eksploracji Internetu. Podobne pod względem treści dokumenty możemy rozłożyć na dwuwymiarowej mapie tak, by leżały w pobliżu siebie - prowadzi to do powstania mapy, na której można wyróżnić obszary tematyczne.

Do zadania zostały użyte dane o kwiatach jest to problem nazwany The Iris flower data set. Zbiór danych kwiatu irysa lub zestaw danych Iris Fisher to wielowymiarowy zestaw danych wprowadzony przez brytyjskiego statystyka i biologa, Ronalda Fishera w jego pracy z 1936 r. Wykorzystanie wielu pomiarów w problemach taksonomicznych jako przykładu liniowej analizy dyskryminacyjnej. Czasami nazywany jest zbiorem danych Irisona Iris, ponieważ Edgar Anderson zebrał dane w celu oszacowania morfologicznej zmienności kwiatów tęczówki trzech pokrewnych gatunków. Dwa z trzech gatunków zostały zebrane na półwyspie Gaspé "wszystkie z tego samego pastwiska i zebrane w tym samym dniu i zmierzone w tym samym czasie przez tę samą osobę za pomocą tego samego urządzenia" .

Tabela Danych użytych:

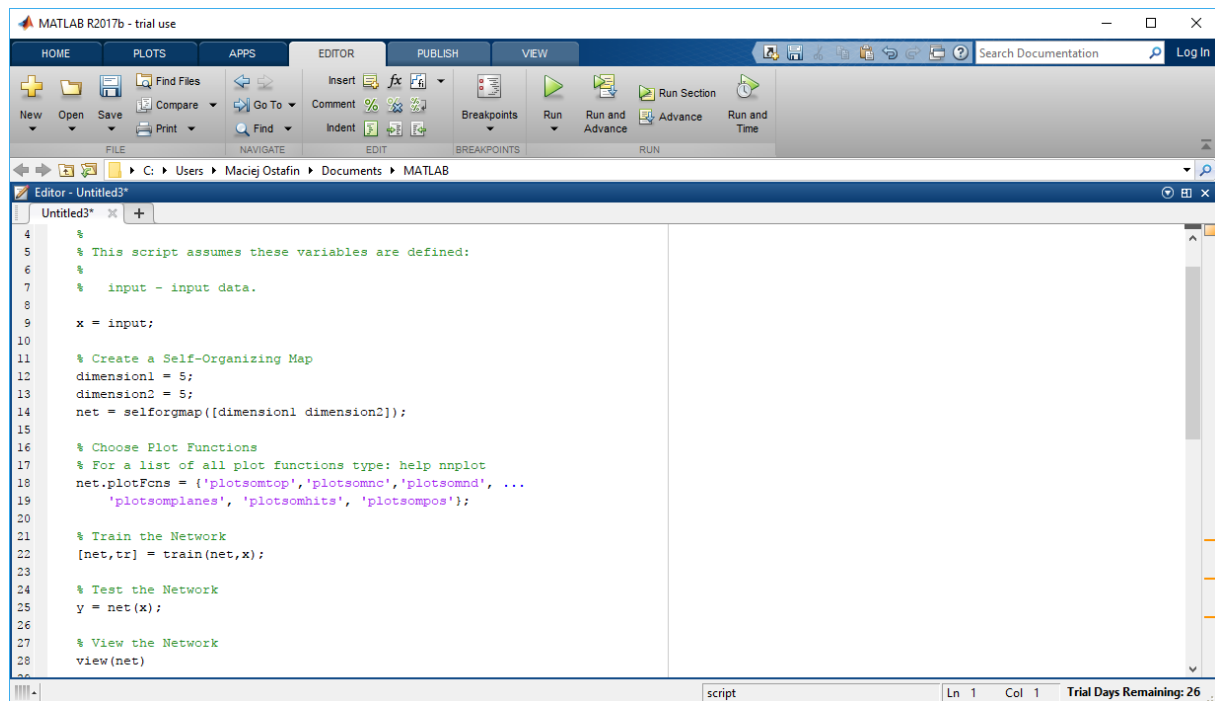
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
4.3	3.0	1.1	0.1	Iris-setosa
5.8	4.0	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa
5.7	3.8	1.7	0.3	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.4	3.4	1.7	0.2	Iris-setosa
5.1	3.7	1.5	0.4	Iris-setosa
4.6	3.6	1.0	0.2	Iris-setosa
5.1	3.3	1.7	0.5	Iris-setosa
4.8	3.4	1.9	0.2	Iris-setosa

5.0	3.0	1.6	0.2	Iris-setosa
5.0	3.4	1.6	0.4	Iris-setosa
5.2	3.5	1.5	0.2	Iris-setosa
5.2	3.4	1.4	0.2	Iris-setosa
4.7	3.2	1.6	0.2	Iris-setosa
4.8	3.1	1.6	0.2	Iris-setosa
5.4	3.4	1.5	0.4	Iris-setosa
5.2	4.1	1.5	0.1	Iris-setosa
5.5	4.2	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.0	3.2	1.2	0.2	Iris-setosa
5.5	3.5	1.3	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
4.4	3.0	1.3	0.2	Iris-setosa
5.1	3.4	1.5	0.2	Iris-setosa
5.0	3.5	1.3	0.3	Iris-setosa
4.5	2.3	1.3	0.3	Iris-setosa
4.4	3.2	1.3	0.2	Iris-setosa
5.0	3.5	1.6	0.6	Iris-setosa
5.1	3.8	1.9	0.4	Iris-setosa
4.8	3.0	1.4	0.3	Iris-setosa
5.1	3.8	1.6	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1.0	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5.0	2.0	3.5	1.0	Iris-versicolor
5.9	3.0	4.2	1.5	Iris-versicolor
6.0	2.2	4.0	1.0	Iris-versicolor
6.1	2.9	4.7	1.4	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor
6.7	3.1	4.4	1.4	Iris-versicolor
5.6	3.0	4.5	1.5	Iris-versicolor
5.8	2.7	4.1	1.0	Iris-versicolor
6.2	2.2	4.5	1.5	Iris-versicolor
5.6	2.5	3.9	1.1	Iris-versicolor

5.9	3.2	4.8	1.8	Iris-versicolor
6.1	2.8	4.0	1.3	Iris-versicolor
6.3	2.5	4.9	1.5	Iris-versicolor
6.1	2.8	4.7	1.2	Iris-versicolor
6.4	2.9	4.3	1.3	Iris-versicolor
6.6	3.0	4.4	1.4	Iris-versicolor
6.8	2.8	4.8	1.4	Iris-versicolor
6.7	3.0	5.0	1.7	Iris-versicolor
6.0	2.9	4.5	1.5	Iris-versicolor
5.7	2.6	3.5	1.0	Iris-versicolor
5.5	2.4	3.8	1.1	Iris-versicolor
5.5	2.4	3.7	1.0	Iris-versicolor
5.8	2.7	3.9	1.2	Iris-versicolor
6.0	2.7	5.1	1.6	Iris-versicolor
5.4	3.0	4.5	1.5	Iris-versicolor
6.0	3.4	4.5	1.6	Iris-versicolor
6.7	3.1	4.7	1.5	Iris-versicolor
6.3	2.3	4.4	1.3	Iris-versicolor
5.6	3.0	4.1	1.3	Iris-versicolor
5.5	2.5	4.0	1.3	Iris-versicolor
5.5	2.6	4.4	1.2	Iris-versicolor
6.1	3.0	4.6	1.4	Iris-versicolor
5.8	2.6	4.0	1.2	Iris-versicolor
5.0	2.3	3.3	1.0	Iris-versicolor
5.6	2.7	4.2	1.3	Iris-versicolor
5.7	3.0	4.2	1.2	Iris-versicolor
5.7	2.9	4.2	1.3	Iris-versicolor
6.2	2.9	4.3	1.3	Iris-versicolor
5.1	2.5	3.0	1.1	Iris-versicolor
5.7	2.8	4.1	1.3	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
6.3	2.9	5.6	1.8	Iris-virginica
6.5	3.0	5.8	2.2	Iris-virginica
7.6	3.0	6.6	2.1	Iris-virginica
4.9	2.5	4.5	1.7	Iris-virginica
7.3	2.9	6.3	1.8	Iris-virginica
6.7	2.5	5.8	1.8	Iris-virginica
7.2	3.6	6.1	2.5	Iris-virginica
6.5	3.2	5.1	2.0	Iris-virginica
6.4	2.7	5.3	1.9	Iris-virginica
6.8	3.0	5.5	2.1	Iris-virginica
5.7	2.5	5.0	2.0	Iris-virginica
5.8	2.8	5.1	2.4	Iris-virginica

6.4	3.2	5.3	2.3	Iris-virginica
6.5	3.0	5.5	1.8	Iris-virginica
7.7	3.8	6.7	2.2	Iris-virginica
7.7	2.6	6.9	2.3	Iris-virginica
6.0	2.2	5.0	1.5	Iris-virginica
6.9	3.2	5.7	2.3	Iris-virginica
5.6	2.8	4.9	2.0	Iris-virginica
7.7	2.8	6.7	2.0	Iris-virginica
6.3	2.7	4.9	1.8	Iris-virginica
6.7	3.3	5.7	2.1	Iris-virginica
7.2	3.2	6.0	1.8	Iris-virginica
6.2	2.8	4.8	1.8	Iris-virginica
6.1	3.0	4.9	1.8	Iris-virginica
6.4	2.8	5.6	2.1	Iris-virginica
7.2	3.0	5.8	1.6	Iris-virginica
7.4	2.8	6.1	1.9	Iris-virginica
6.4	2.8	5.6	2.2	Iris-virginica
6.3	2.8	5.1	1.5	Iris-virginica
6.1	2.6	5.6	1.4	Iris-virginica
7.7	3.0	6.1	2.3	Iris-virginica
6.3	3.4	5.6	2.4	Iris-virginica
6.4	3.1	5.5	1.8	Iris-virginica
6.0	3.0	4.8	1.8	Iris-virginica
6.9	3.1	5.4	2.1	Iris-virginica
6.7	3.1	5.6	2.4	Iris-virginica
6.9	3.1	5.1	2.3	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
6.8	3.2	5.9	2.3	Iris-virginica
6.7	3.3	5.7	2.5	Iris-virginica
6.7	3.0	5.2	2.3	Iris-virginica
6.3	2.5	5.0	1.9	Iris-virginica
6.5	3.0	5.2	2.0	Iris-virginica
6.2	3.4	5.4	2.3	Iris-virginica
5.9	3.0	5.1	1.8	Iris-virginica

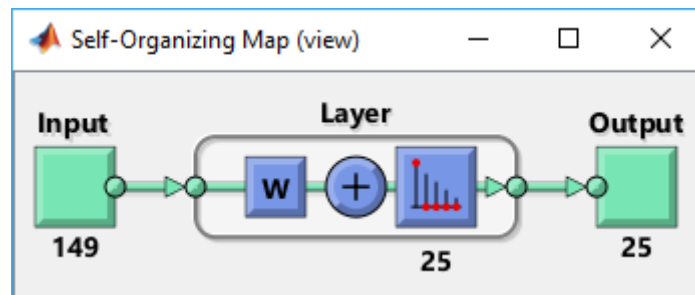
Do Stworzenie sieci Kohonena posłużyłem się Programem Matlab.



```
4 %  
5 % This script assumes these variables are defined:  
6 %  
7 % input - input data.  
8  
9 x = input;  
10  
11 % Create a Self-Organizing Map  
12 dimension1 = 5;  
13 dimension2 = 5;  
14 net = selforgmap([dimension1 dimension2]);  
15  
16 % Choose Plot Functions  
17 % For a list of all plot functions type: help nnplot  
18 net.plotFcns = {'plotsomtop','plotsomnc','plotsomnd', ...  
19               'plotsomplanes', 'plotsomhits', 'plotsompos'};  
20  
21 % Train the Network  
22 [net,tr] = train(net,x);  
23  
24 % Test the Network  
25 y = net(x);  
26  
27 % View the Network  
28 view(net)
```

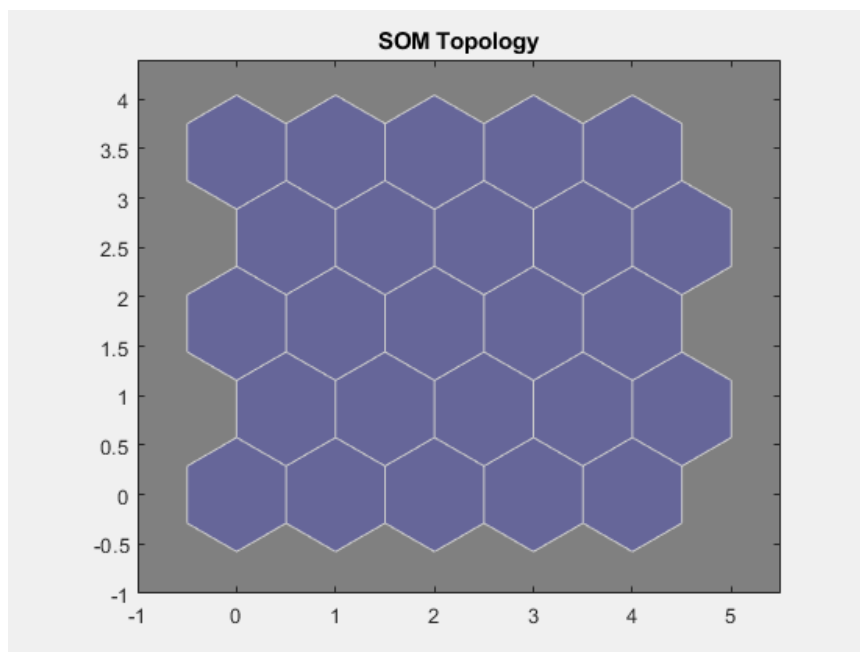
Przykładowy zrzut ekranu

Schemat działania

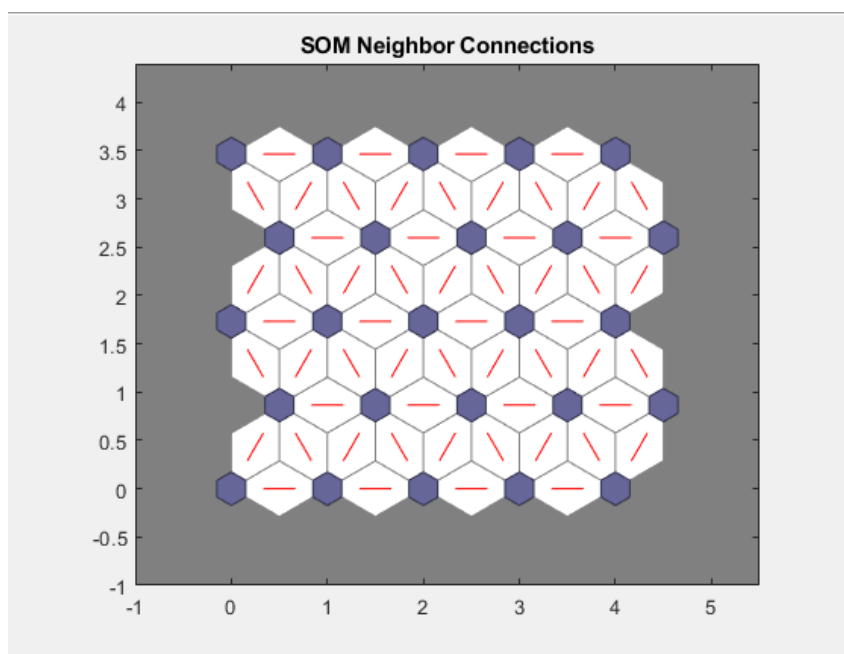


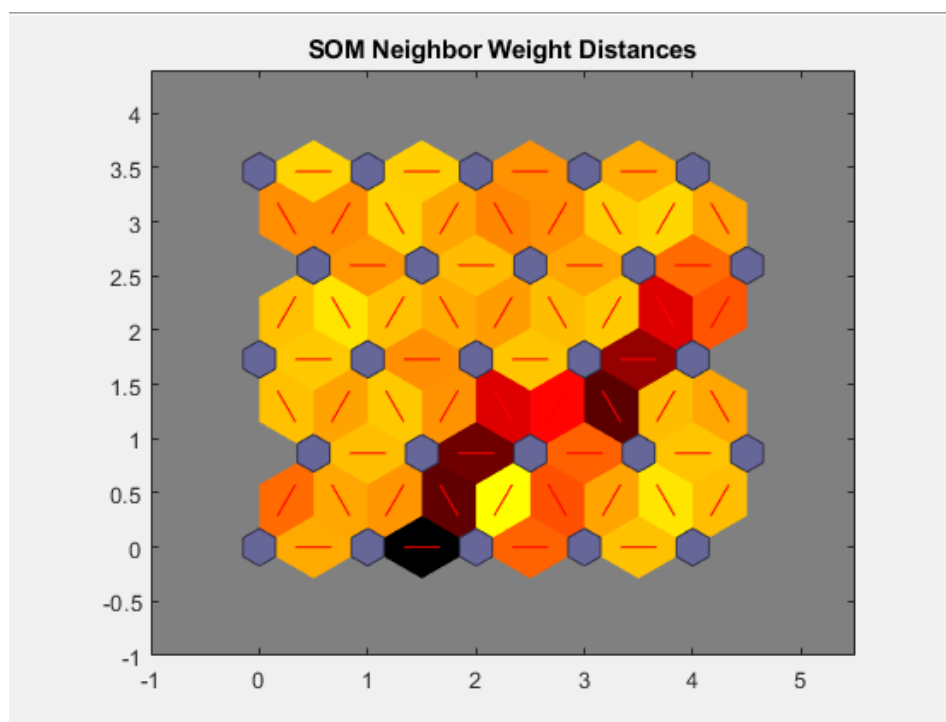
Wyniki Dla Stworzonej samoorganizującej się 2 wymiarowej Sieci Kohonena.
Każdy wymiar zawiera 4 neurony.

Topologia 2 wymiarowej sieci 5x5



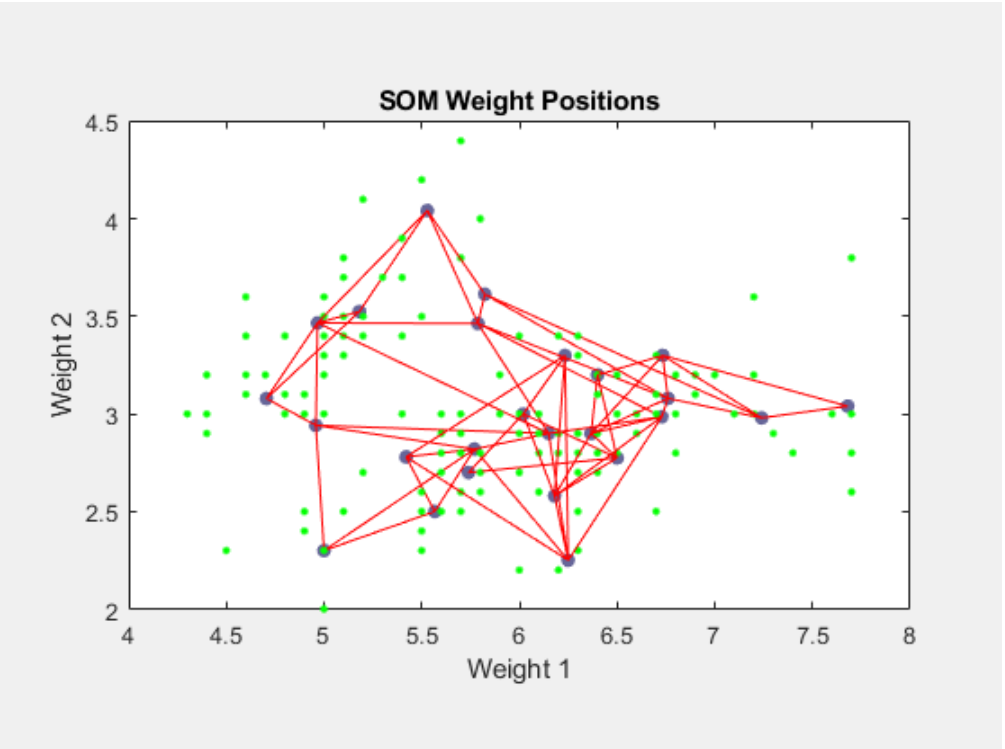
Polaczenia miedzy sąsiednimi neuronami





Odległości między neuronami po stworzeniu sieci

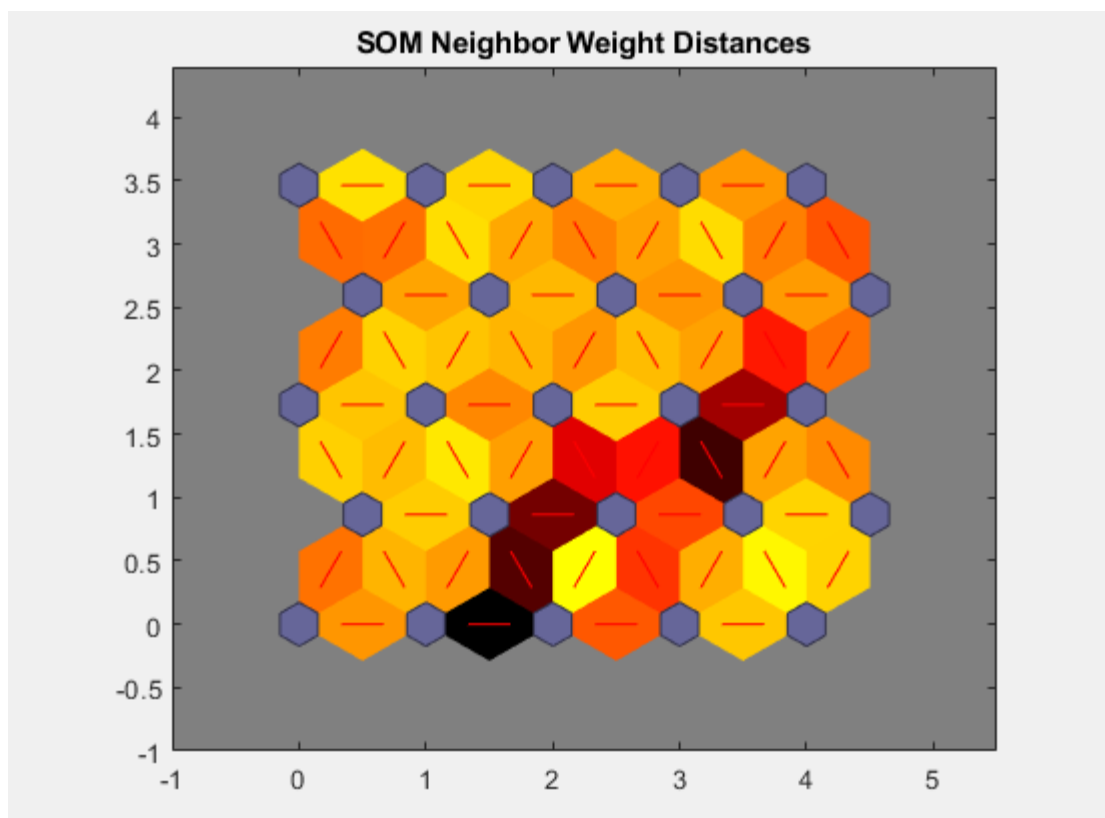
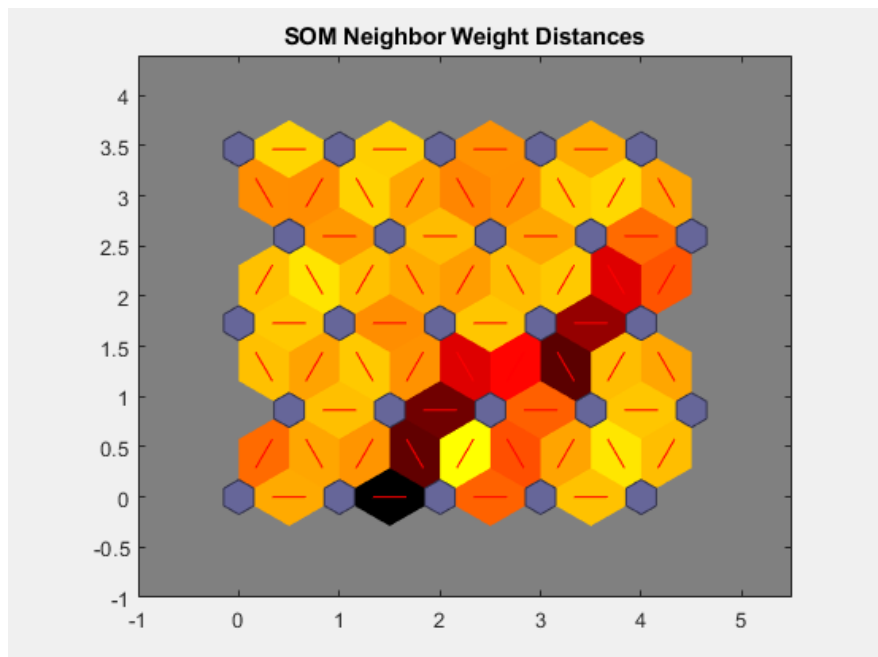
Wygląd całej sieci wraz z neuronami , danymi oraz połączeniami

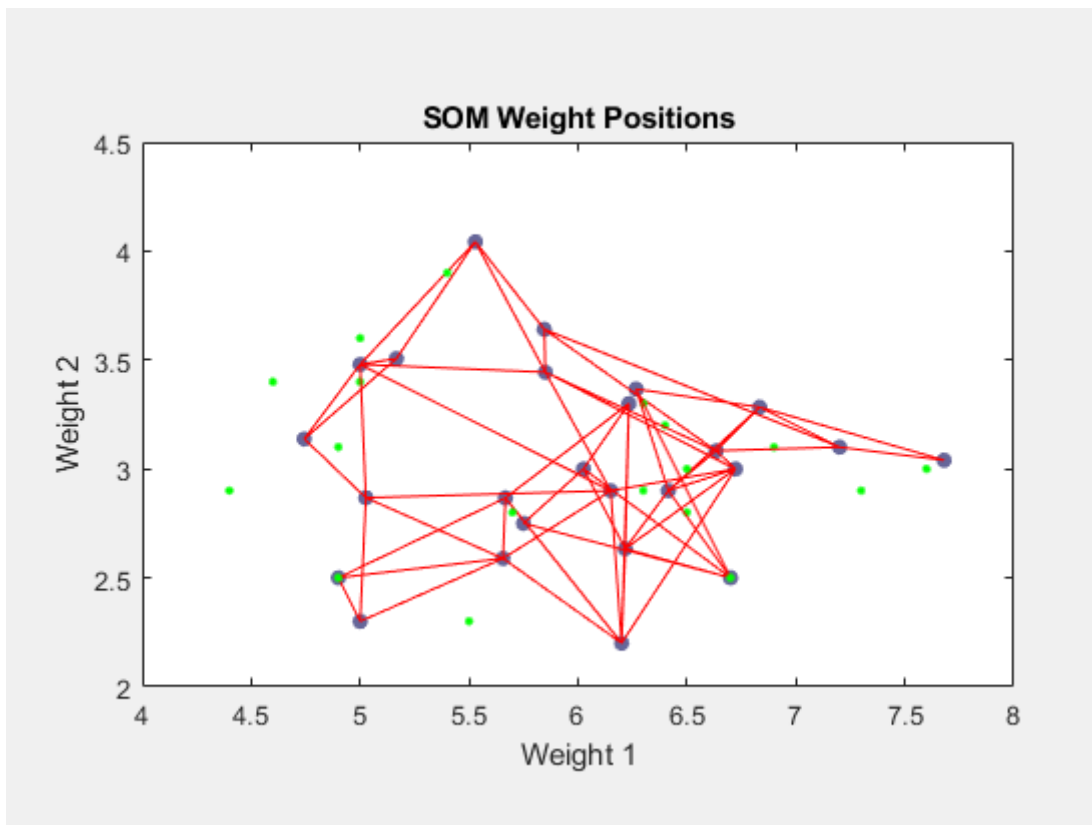


Dane użyte do testowania:

5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4
4.6	3.4	1.4	0.3
5.0	3.4	1.5	0.2
4.4	2.9	1.4	0.2
4.9	3.1	1.5	0.1
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5
5.7	2.8	4.5	1.3
6.3	3.3	4.7	1.6
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2
7.6	3.0	6.6	2.1
4.9	2.5	4.5	1.7
7.3	2.9	6.3	1.8
6.7	2.5	5.8	1.8

Wyniki :





Wnioski:

Po porównaniu wyników dla sieci stworzonej z wszystkich danych i tej przetestowanej na podstawie 18 przykładów można zauważyć iż sieć potrafiła dopasować wygląd sieci na podstawie danych testowych na przyzwoitym poziomie. Odległości między poszczególnymi neuronami nie różnią się w znacznym stopniu od siebie tylko na kilku można zobaczyć nieco większą zmianę koloru co świadczy o zmniejszonej lub zwiększonej odległości między neuronami a co za tym idzie dopasowaniem danych testujących do nauczonych. Na wykresie całej sieci można również zauważyć iż niektóre połączenia są nieco odmiennie długości a położenie neuronów nieco odbiega od pierwotnej pozycji. Patrz na cały wynik można powiedzieć iż algorytm WTA sprawdził się tutaj w miarę satysfakcjonująco a trafność wyników testowania jest na dobrym poziomie.

KOD PROGRAMU:

```
from
Kohonen
import
Kohonen

from math import sqrt
from Data import flower_to_learn
from Data import flower_to_test
def learn(kohonens):
    "Funkcja uczenia"
    eras = 0
    winners = []
    for i in range(0, FLOWERS):
        winners.append([])
        for j in range(0, LEARN_SAMPLES):
            winners[i].append(-1)
    while unique(winners) != True:
        for i in range(0, FLOWERS):
            for j in range(0, LEARN_SAMPLES):
                winner = get_winner(kohonens, flower_to_learn[i][j])
                kohonens[winner].learn(flower_to_learn[i][j], LEARNING_RATE)
        for i in range(0, FLOWERS):
            for j in range(0, LEARN_SAMPLES):
                winners[i][j] = get_winner(kohonens, flower_to_learn[i][j])
        eras += 1
        if eras == LIMIT:
            break
    return eras
def unique(winners):
    "Funkcja sprawdza czy siec juz sie nauczyla rozpoznawac kwiaty"
    "Petla sprawdza czy kazdy typ ma jednego zwyciezce"
    for i in range(0, FLOWERS):
        for j in range(1, LEARN_SAMPLES):
            if winners[i][0] != winners[i][j]:
                return False
    "Petla sprawdzajaca wszystkie gatunki, czy jest rozny od zwyciezcow pozostalych gatunkow"
    for i in range(0, FLOWERS):
        for j in range(0, FLOWERS):
            if i != j:
                if winners[i][0] == winners[j][0]:
                    return False
    return True
def get_winner(kohonens, vector):
    "Zwraca wygranego"
    winner = 0
    minimum_distance = vector_distance(kohonens[0].weights, vector)
    for i in range(1, NEURONS):
        current_distance = vector_distance(kohonens[i].weights, vector)
```

```

        if current_distance < minimum_distance:
            winner = i
            minimum_distance = current_distance
    return winner

def vector_distance(vector, vector2):
    "Zwraca odleglosc pomiedzy dwoma wektorami (Manhattan distance)"
    sum = 0
    for i in range(0, len(vector)):
        sum += abs(vector[i] - vector2[i])
    return sqrt(sum)

### Dane wejściowe
LEARNING_RATE = 0.01
INPUTS = 4
NEURONS = 220
FLOWERS = 3
LEARN_SAMPLES = 20
TEST_SAMPLES = 8
LIMIT = 10000
success = 0
fail = 0
while success != 10 and fail != 100:
    kohonens = []
    for i in range(0, NEURONS):
        kohonens.append(Kohonen(INPUTS))
    eras = learn(kohonens)
    if eras != LIMIT:
        success += 1
        print("Po nauce:\n")
        for i in range(0, FLOWERS):
            winner = get_winner(kohonens, flower_to_learn[i][0])
            print("Kwiat [", i, "] wygral =", winner)
        print("\nPoczekaj na testowanie:")
        for i in range(0, FLOWERS):
            print("\n")
            for j in range(0, TEST_SAMPLES):
                winner = get_winner(kohonens, flower_to_test[i][j])
                print("Kwiat [", i, "][", j, "]", "winner =", winner)
            print("Ilosc epok:", eras)
    else:
        fail += 1
print("Ilosc niepowodzen w nauce:", fail)

```

```

from
random
import
uniform

```

```

class Kohonen:
    def __init__(self, inputs):

```

```

        "Konstruktor"
        self.inputs = inputs
        self.weights = []
        for i in range(0, inputs):
            self.weights.append(uniform(0, 1))
    def learn(self, vector, learning_rate):
        "Metoda uczenia"
        for i in range(0, self.inputs):
            self.weights[i] += learning_rate * (vector[i] - self.weights[i])

```

```

from openpyxl
import
load_workbook

from math import pow
from math import sqrt
### rekordy w excelu ###
### 1-50 ---> setosa
### 51-100 ---> versicolor
### 101-150 ---> virginica
workbook = load_workbook('./Iris.xlsx', data_only=True)
sheet = workbook.get_sheet_names()[0]
worksheet = workbook.get_sheet_by_name(sheet)
text_file = open("flowers.txt", "a")
def normalize(array, number):
    sum = 0
    for i in range(0, len(array)):
        sum += pow(array[i], 2)
    return number / sqrt(sum)
list = []
for row in worksheet.iter_rows():
    for i in range(0, len(row)):
        list.append(row[i].value)
    text_file.write("[ ")
    for i in range(0, 4):
        result = normalize(list, list[i])
        text_file.write(str(result))
        if i != 3:
            text_file.write(", ")
    text_file.write(" ],\n")
    list.clear()
text_file.close()

```