Klaudia Pakos

IO, rok 3, 267561

PODSTAWY SZTUCZNEJ INTELIGENCJI

SPRAWOZDANIE NR 5

BUDOWA I DZIAŁANIE SIECI KOHONENA DLA WTA

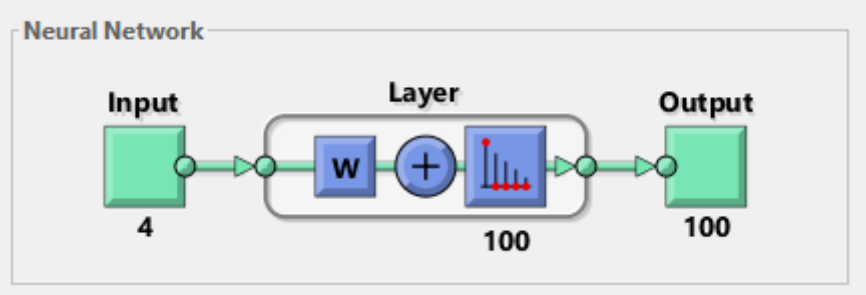
1. Cel ćwiczenia

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

2. Zadania do wykonania

- Przygotowanie danych uczących zawierających numeryczny opis cech kwiatów.
- Przygotowanie sieci Kohonena i algorytmu uczenia opartego o regułę Winner Takes All (WTA).
- Uczenie sieci dla różnych współczynników uczenia.
- Testowanie sieci.

3. Opis budowy sieci



Działanie sieci Kohonena

Sieci Kohonena są szczególnym przypadkiem algorytmu realizującego uczenie się bez nadzoru. Ich głównym zadaniem jest organizacja wielowymiarowej informacji w taki sposób, żeby można ją było prezentować i analizować w przestrzeni o znacznie mniejszej liczbie wymiarów, czyli mapie.

Warunkiem jest to, że rzuty "podobnych" danych wejściowych powinny być bliskie również na mapie.

Zasady działania sieci Kohonena:

- Wejścia 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.

Mechanizm WTA (Winner Takes All)

Reguła aktywacji neuronów w sieci neuronowej, która pozwala na aktywację tylko jednego neuronu w danej chwili. W konsekwencji w jednym momencie zostaje zmodyfikowana waga tylko jednego neuronu. Aby uniknąć dominacji jednego neuronu często stosuje się mechanizm zmęczenia, który polega na tym, że jeśli jakiś neuron wygrywa zbyt często, to przez pewien czas przestaje być brany pod uwagę podczas rywalizacji (jest odsuwany/usypiany w celu wylosowania innego neuronu).

Stosowanie tego mechanizmu prowadzi do podzielenia mapy neuronów na tzw. „strefy wpływów”. Są to obszary, które znajdują się pod dominacją konkretnego silnego neuronu, który uniemożliwia modyfikację wag neuronów z jego otoczenia (brak normalizacji może doprowadzić do sytuacji, w której na niewielkim obszarze znajduje się kilka silnych neuronów lub kilka niewielkich stref wpływów, natomiast pozostały obszar nie posiada żadnego silnego neuronu - nierównomierny rozkład sił).

Użyty algorytm uczenia sieci

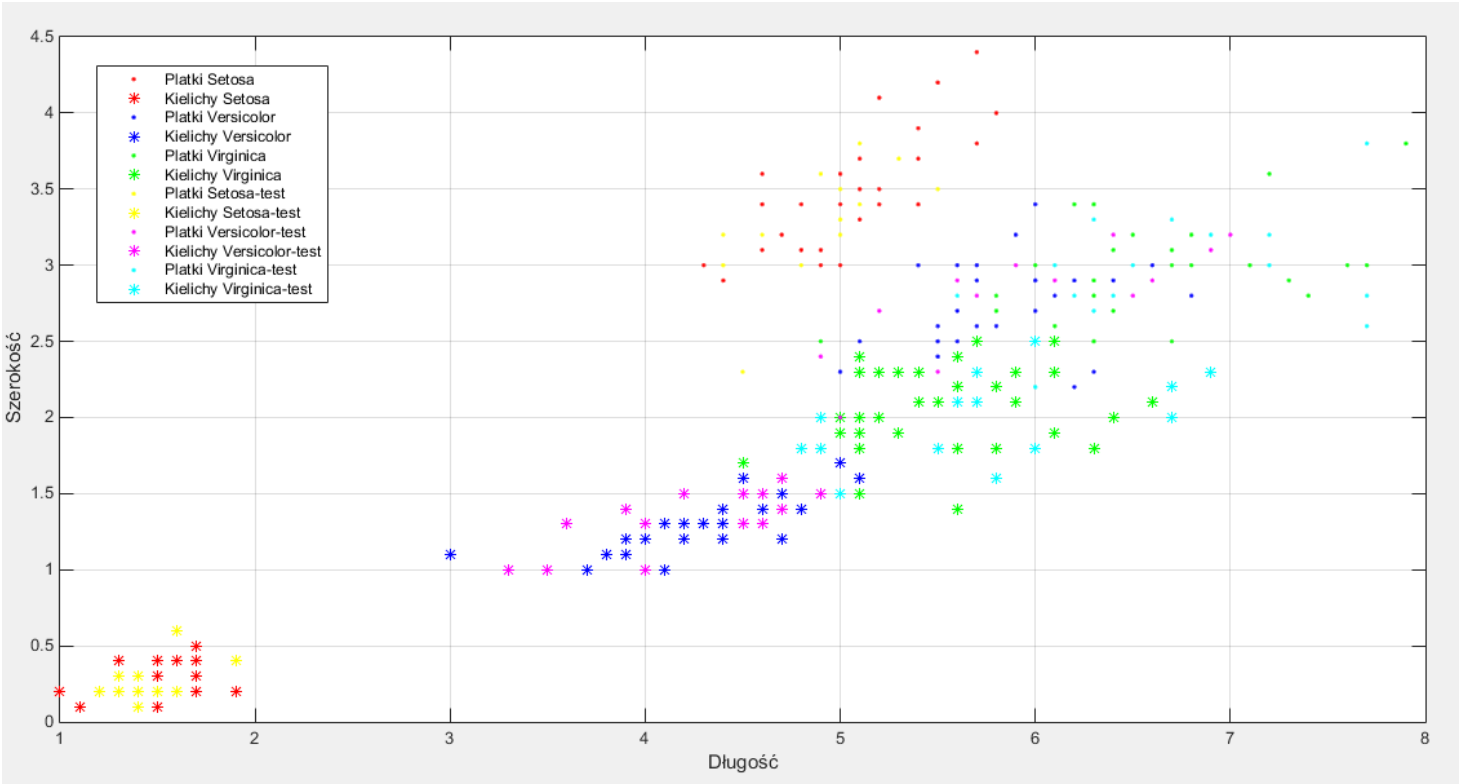
- Generowanie losowo znormalizowanych wektorów wag.
- Losowanie wektora X oraz obliczanie dla niego aktywację Y dla wszystkich neuronów.
- Szukanie neuronu zwycięzcy.
- Modyfikacja wektora wag neuronu zwycięzcy, a następnie jego normalizacja (sprawdzenie warunków WTA).
- Zatrzymanie algorytmu po odpowiednio dużej ilości iteracji.

Dane wejściowe:

Dane uczące zostały wygenerowane za pomocą zaimplementowanych w oprogramowaniu Matlab zbioru danych: iris_dataset. Zawiera on numeryczny zapis czterech cech kwiatów irysa - długość i szerokość płatków oraz długość i szerokość kielichów kwiatów). Opracowany zbiór danych wskazuje na przyporządkowanie irysów do trzech różnych rodzajów, w zależności od wielkości ich płatków i kielichów.

Jako dane testowe, przyjęto macierz 45 elementów przedstawioną w listingu kodu, w której pierwsze 15 reprezentowało irysy gatunku Setosa, kolejne 15 gatunku Versicolor i ostatnie 15 gatunku Virginica.

Wygenerowano w ten sposób następujące dane:



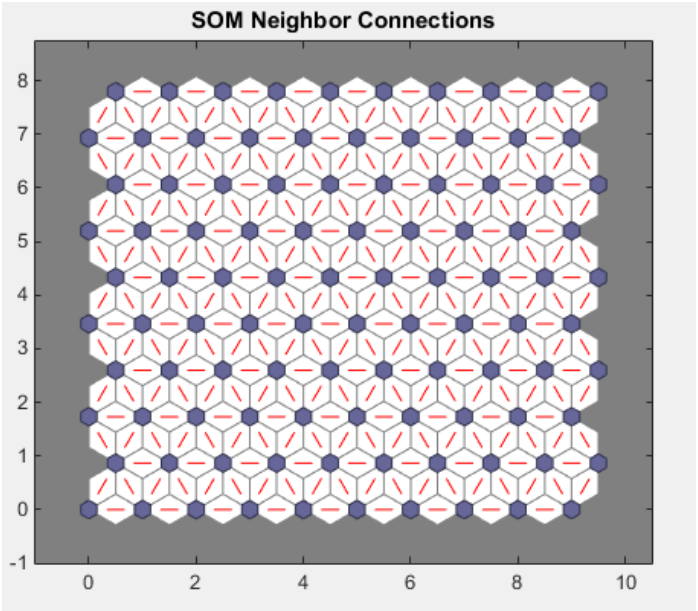
4. Przebieg zadania-listing kodu

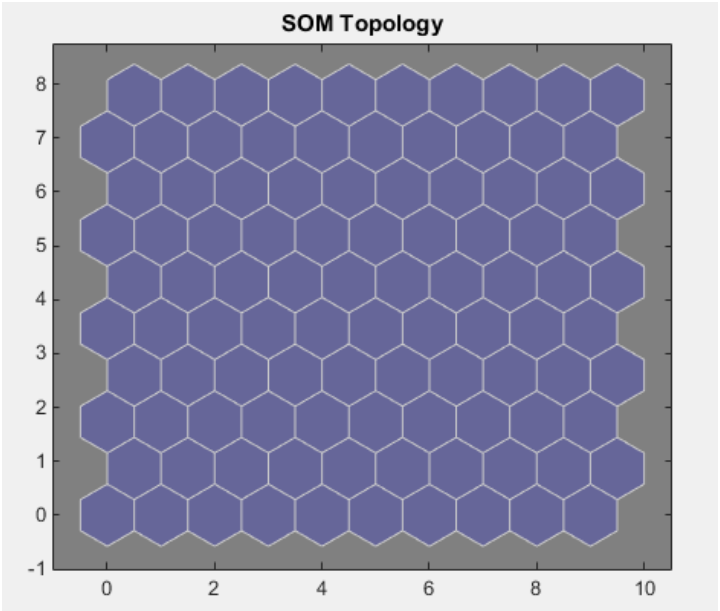
```
close all; clear all; clc;

WEJSCIE = iris_dataset;      %dane wejsciowe
size(WEJSCIE);               %okreslenie rozmiaru tablicy
x=1:50;
plot(WEJSCIE(1, x), WEJSCIE(2, x), 'r.', WEJSCIE(3, x), WEJSCIE(4,
x), 'r*');
hold on; grid on;
x=51:100;
plot(WEJSCIE(1, x), WEJSCIE(2, x), 'b.', WEJSCIE(3, x), WEJSCIE(4,
x), 'b*');
hold on; grid on;
x=101:150;
plot(WEJSCIE(1, x), WEJSCIE(2, x), 'g.', WEJSCIE(3, x), WEJSCIE(4,
x), 'g*');
legend('Platki Setosa','Kielichy Setosa','Platki
Versicolor','Kielichy Versicolor','Platki Virginica','Kielichy
Virginica'); %legenda
hold on; grid on;            % wyświetlenie danych wejściowych

% PARAMETRY SIECI KOHONENA
dimensions    = [8, 8];     %wymiar wektora
coverSteps    = 100;         %liczba kroków szkoleniowych dla
początkowego pokrycia przestrzeni wejściowej
initNeighbor = 0;            %początkowy rozmiar sąsiedztwa
topologyFcn   = 'hextop';    %funkcja topologii warstw
distanceFcn   = 'dist';      %funkcja odległości neuronowej
```

```
% TWORZENIE SIECI KOHONENA (SOM)
net = selforgmap(dimensions, coverSteps, initNeighbor, topologyFcn,
distanceFcn);
net.trainParam.epochs = 500;%ustalenie maksymalnej liczby epok
treningowych utworzonej sieci

% TRENING SIECI
[net, tr] = train(net, WEJSCIE);
%wartości testowe
test=[  5.0     3.2     1.2     0.2 ;
        5.5     3.5     1.3     0.2 ;
        4.9     3.6     1.4     0.1 ;
        4.4     3.0     1.3     0.2 ;
        5.1     3.4     1.5     0.2 ;
        5.0     3.5     1.3     0.3 ;
        4.5     2.3     1.3     0.3 ;
        4.4     3.2     1.3     0.2 ;
        5.0     3.5     1.6     0.6 ;
        5.1     3.8     1.9     0.4 ;
        4.8     3.0     1.4     0.3 ;
        5.1     3.8     1.6     0.2 ;
        4.6     3.2     1.4     0.2 ;
        5.3     3.7     1.5     0.2 ;
        5.0     3.3     1.4     0.2 ;
        7.0     3.2     4.7     1.4 ;
        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 ;
        4.9     2.4     3.3     1.0 ;
        6.6     2.9     4.6     1.3 ;
        5.2     2.7     3.9     1.4 ;
        5.0     2.0     3.5     1.0 ;
        5.9     3.0     4.2     1.5 ;
        6.0     2.2     4.0     1.0 ;
        6.1     2.9     4.7     1.4 ;
        5.6     2.9     3.6     1.3;
        6.3     3.3     6.0     2.5 ;
        6.5     3.0     5.5     1.8 ;
        7.7     3.8     6.7     2.2 ;
        7.7     2.6     6.9     2.3 ;
        6.0     2.2     5.0     1.5 ;
        6.9     3.2     5.7     2.3 ;
        5.6     2.8     4.9     2.0 ;
        7.7     2.8     6.7     2.0 ;
        6.3     2.7     4.9     1.8 ;
        6.7     3.3     5.7     2.1 ;
        7.2     3.2     6.0     1.8 ;
        6.2     2.8     4.8     1.8 ;
        6.1     3.0     4.9     1.8 ;
        6.4     2.8     5.6     2.1 ;
        7.2     3.0     5.8     1.6 ;   ];
    test=test';
```

```
%wyswietlenie wartości testowych
x=1:15;
plot(test(1, x), test(2, x), 'y.', test(3, x), test(4, x), 'y*');
hold on; grid on;
x=16:30;
plot(test(1, x), test(2, x), 'm.', test(3, x), test(4, x), 'm*');
hold on; grid on;
x=31:45;
plot(test(1, x), test(2, x), 'c.', test(3, x), test(4, x), 'c*');
legend('Platki Setosa','Kielichy Setosa','Platki
Versicolor','Kielichy Versicolor','Platki Virginica','Kielichy
Virginica','Platki Setosa-test','Kielichy Setosa-test','Platki
Versicolor-test','Kielichy Versicolor-test','Platki Virginica-
test','Kielichy Virginica-test'); %legenda
xlabel('Długość'),ylabel('Szerokość');

hold on; grid on;            % wyświetlenie danych wejściowych


y = net(test); %testowanie sieci dla danych testowych
y2=net(WEJSCIE);%testowanie sieci dla danych wejściowych

figure, pcolor(y); title('Dane testowe'); %wyświetlenie mapy dla
danych testowych
figure, pcolor(y2); title('Dane wejściowe'); %wyświetlenie mapy dla
danych początkowych
początkowych
```

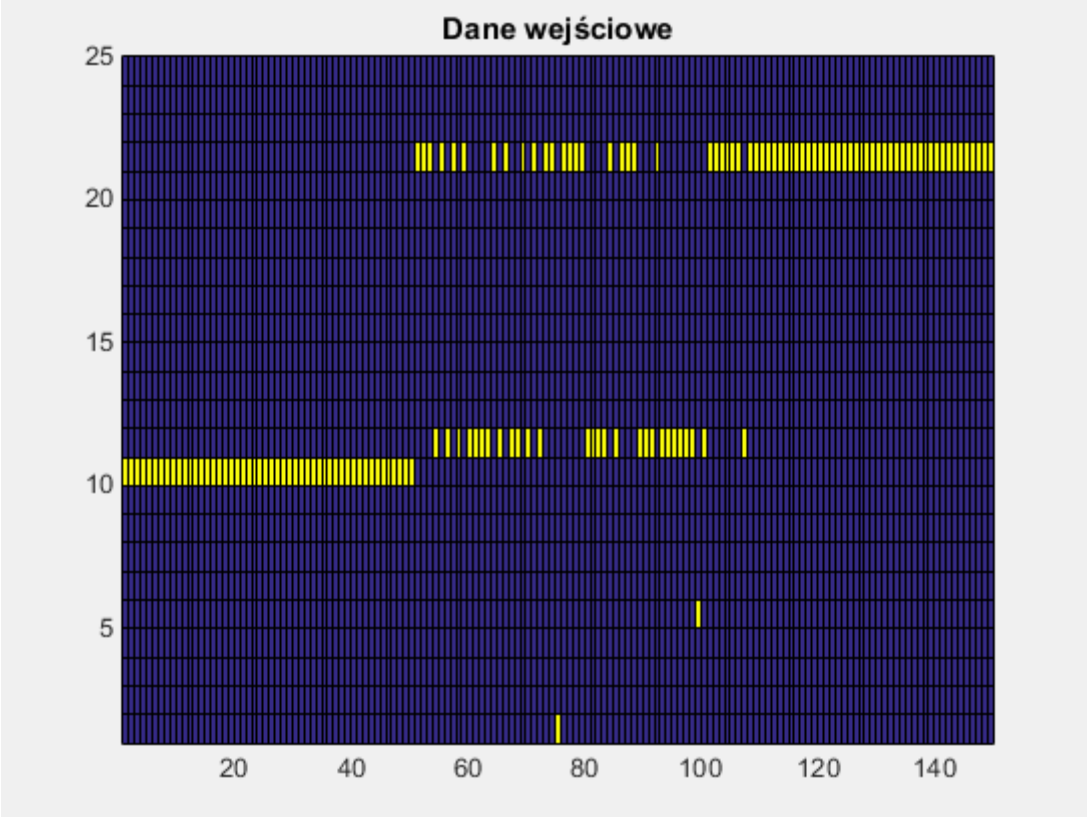
5. Przedstawienie wyników

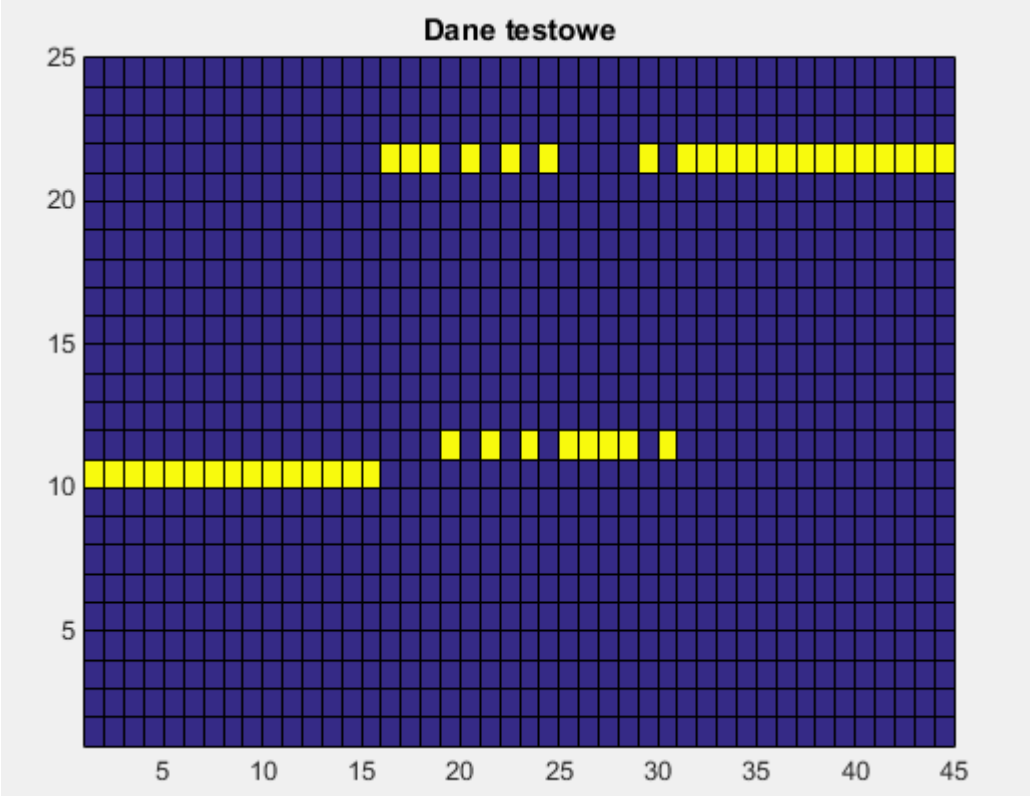


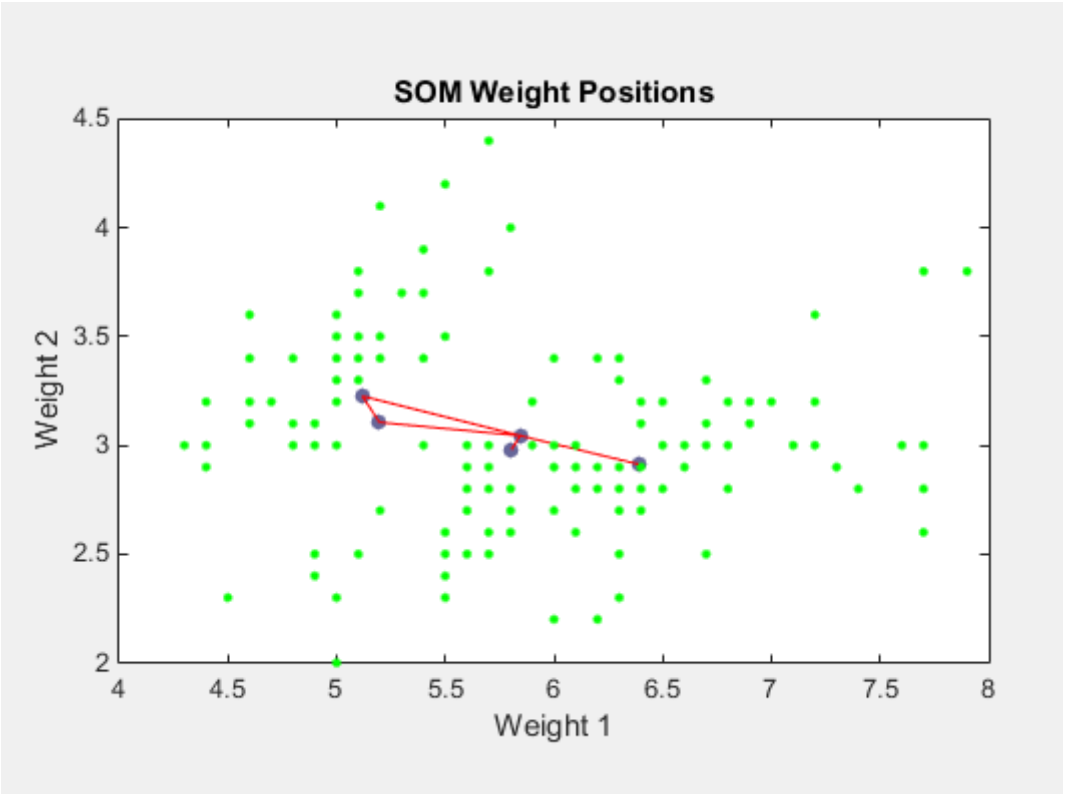


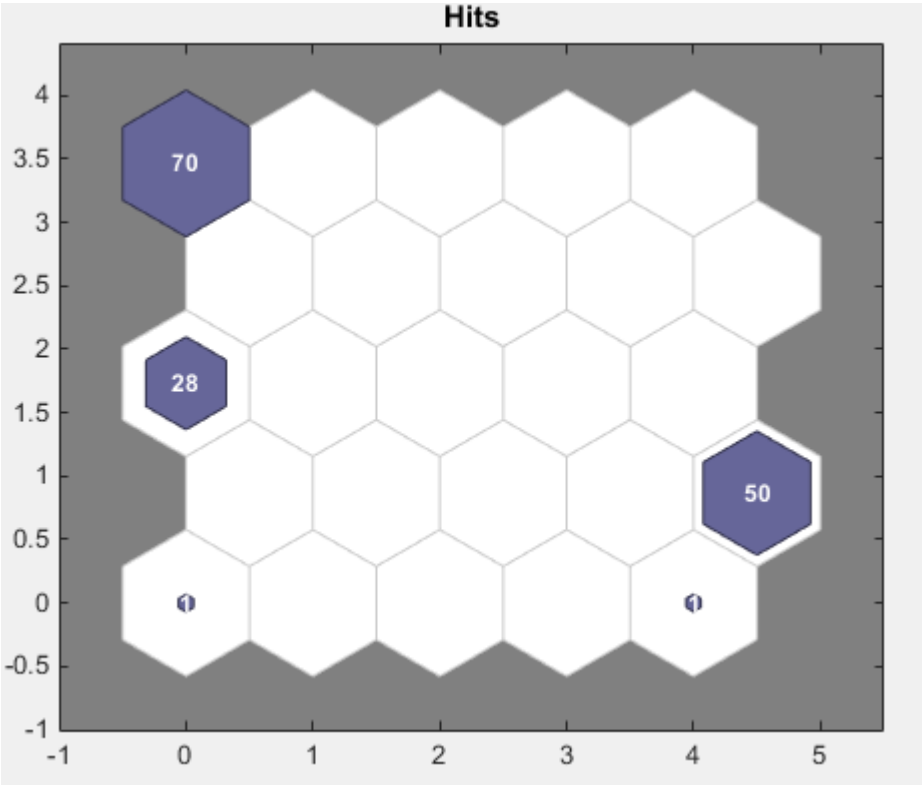
Sprawdzono 3 różne wymiary wektora do sieci Kohenena: [5,5], [8,8], [10,10], otrzymano następujące wyniki:

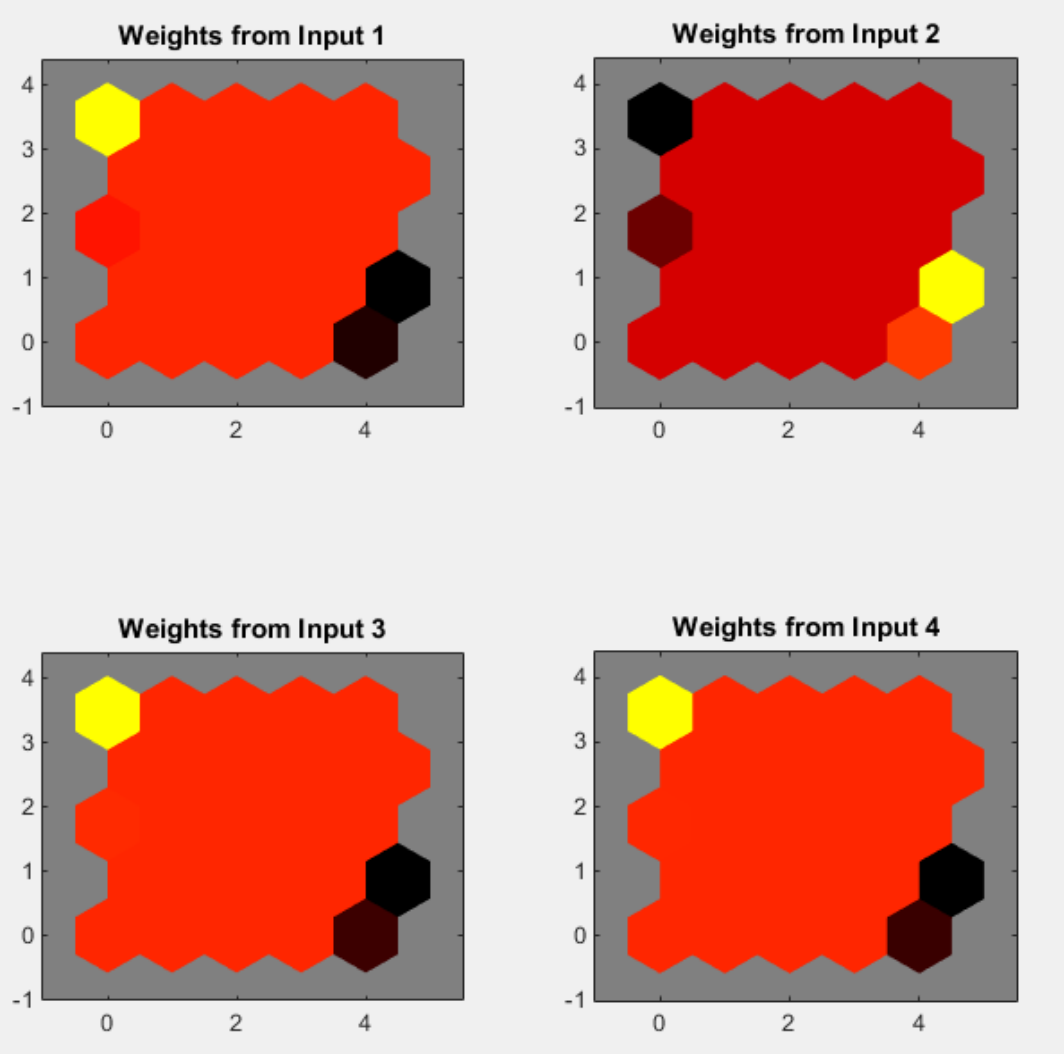
Dla [5,5]



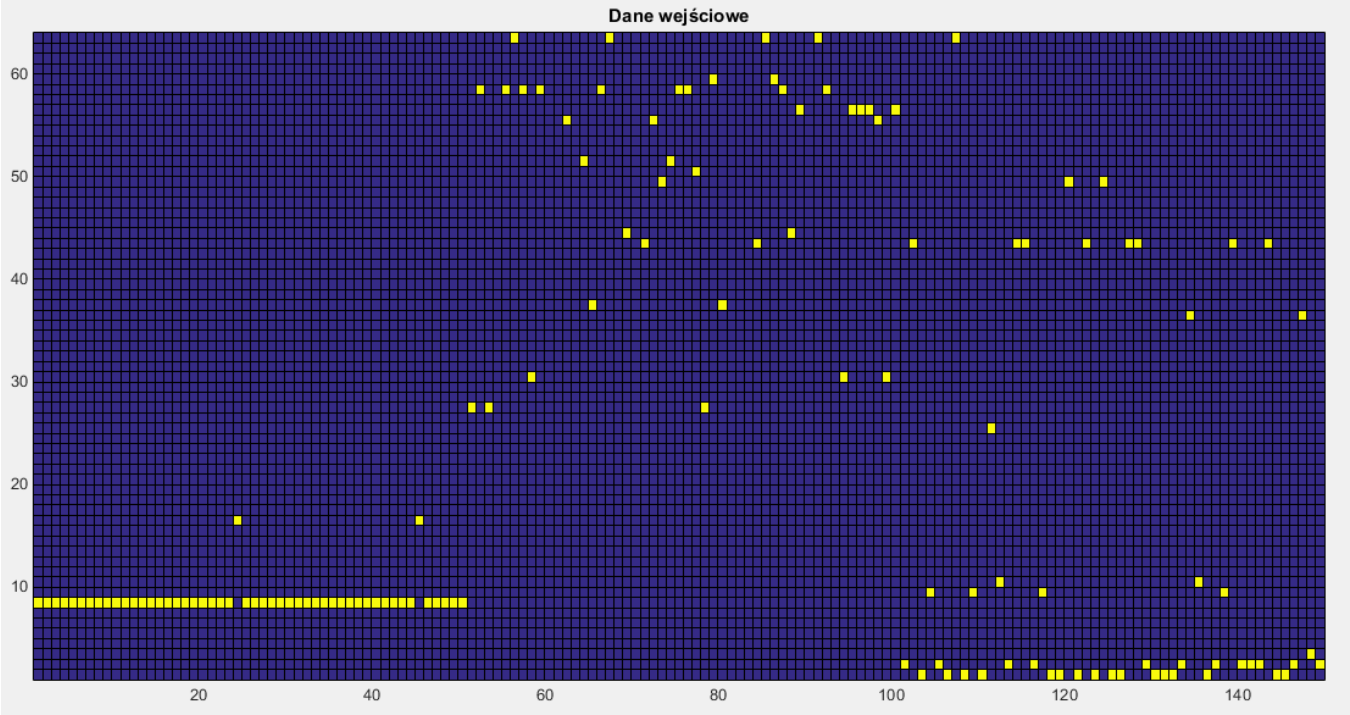


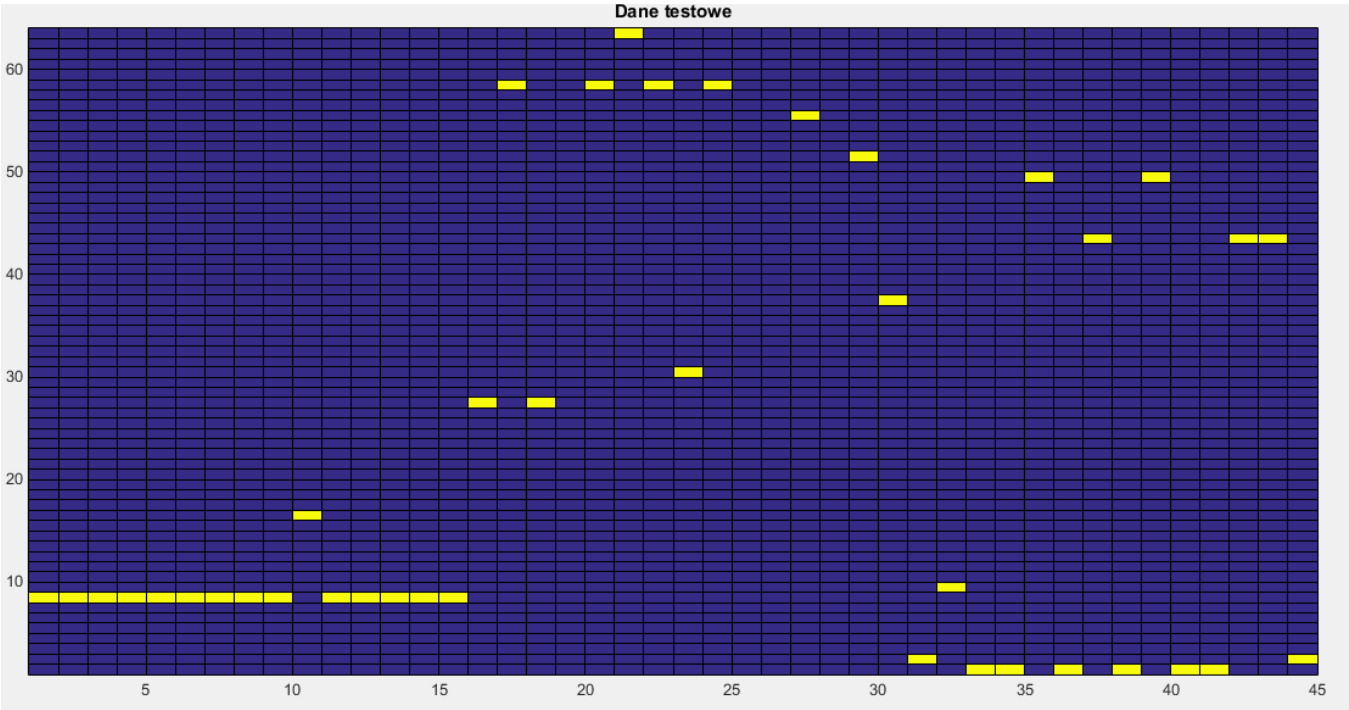

SOM Weight Positions
Weight 2
Weight 1
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

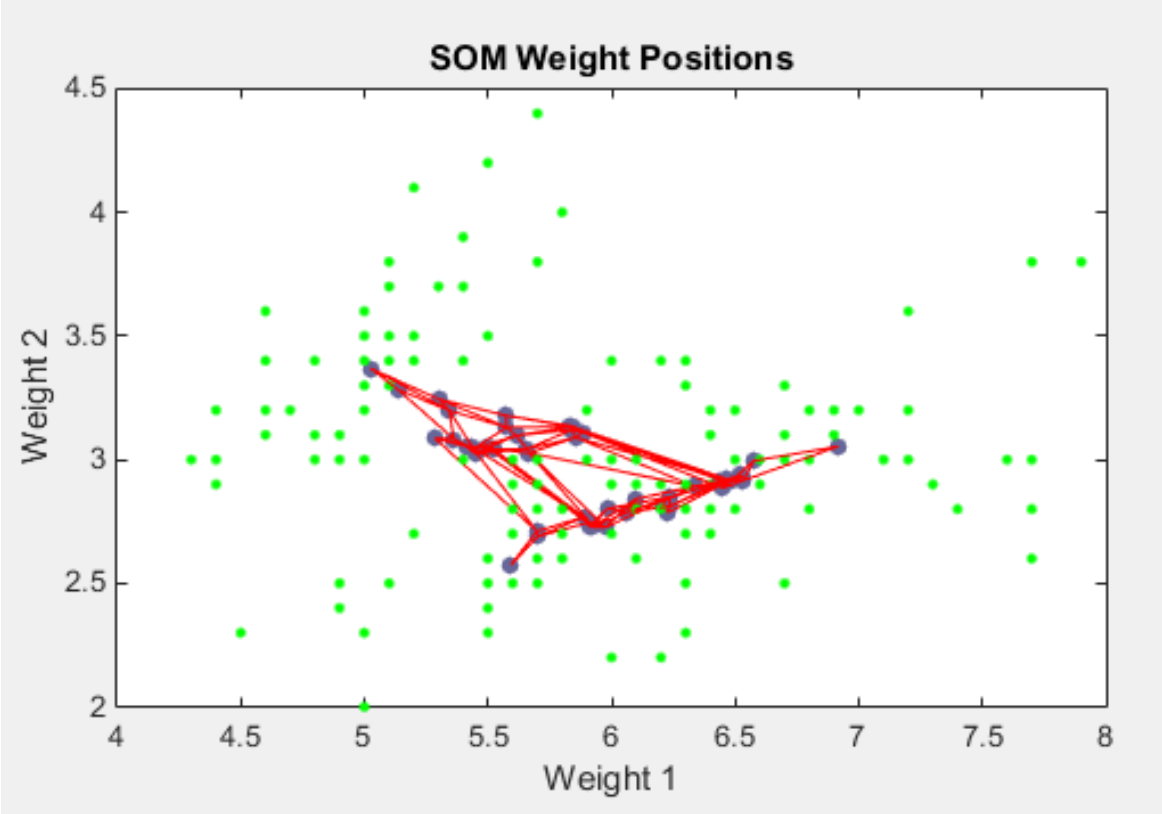

Hits
70
28
50
1
1
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1
0
1
2
3
4
5

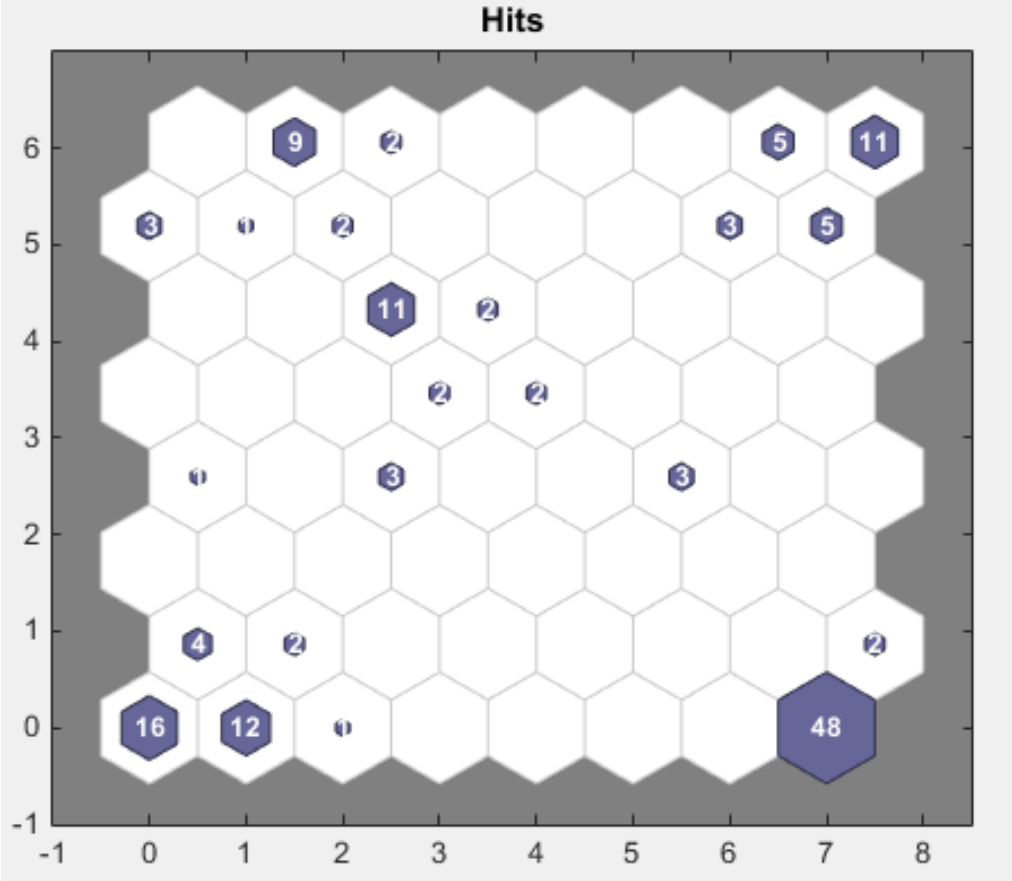

Weights from Input 1
Weights from Input 2
Weights from Input 3
Weights from Input 4

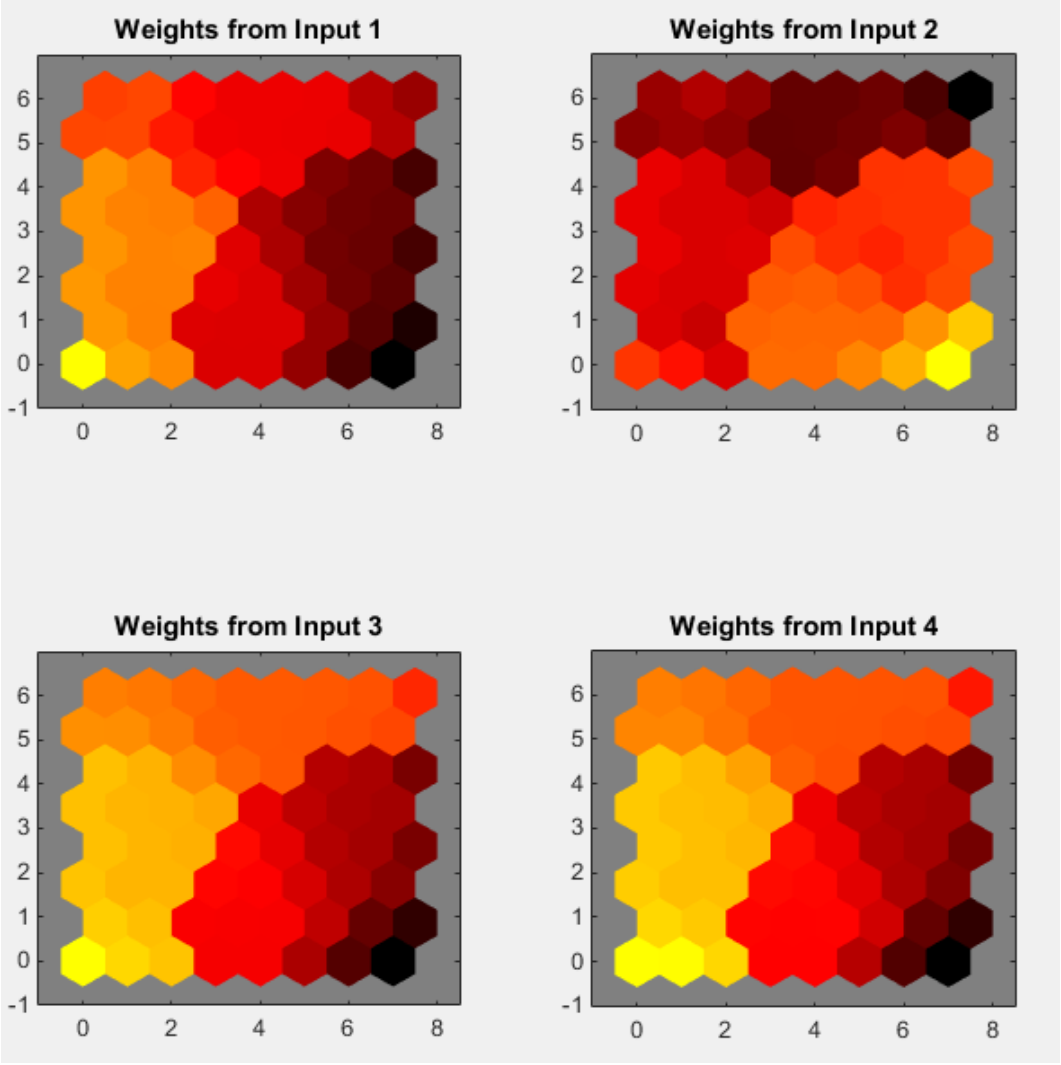
Dla [8,8]


Dane wejściowe

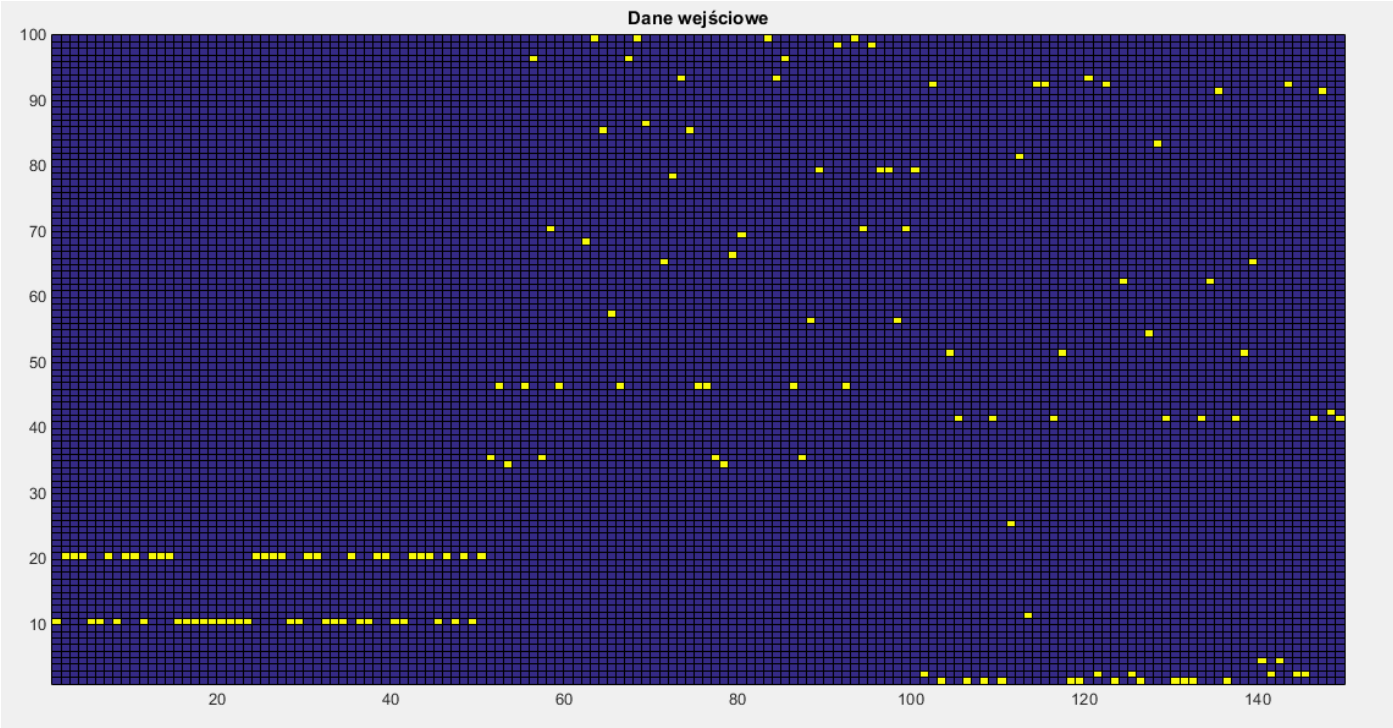

Dane testowe
60
50
40
30
20
10
5
10
15
20
25
30
35
40
45

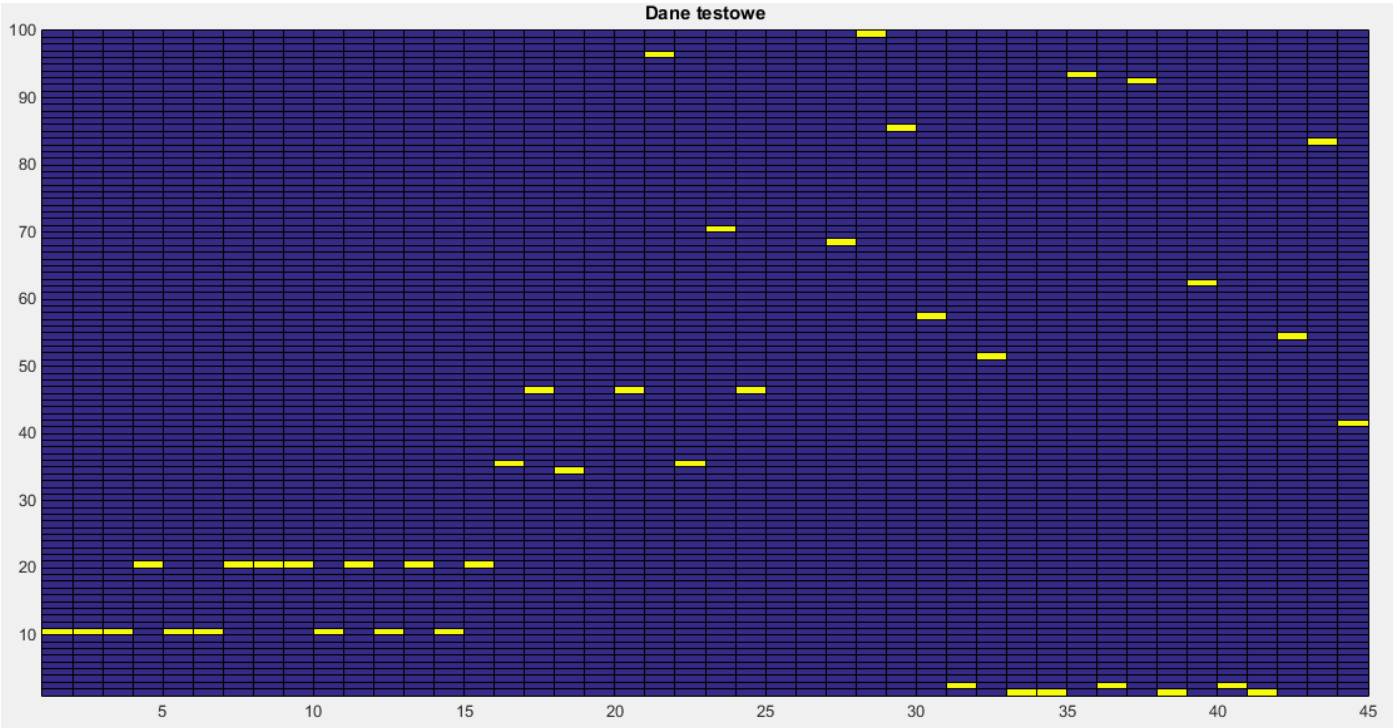

SOM Weight Positions
Weight 2
Weight 1
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

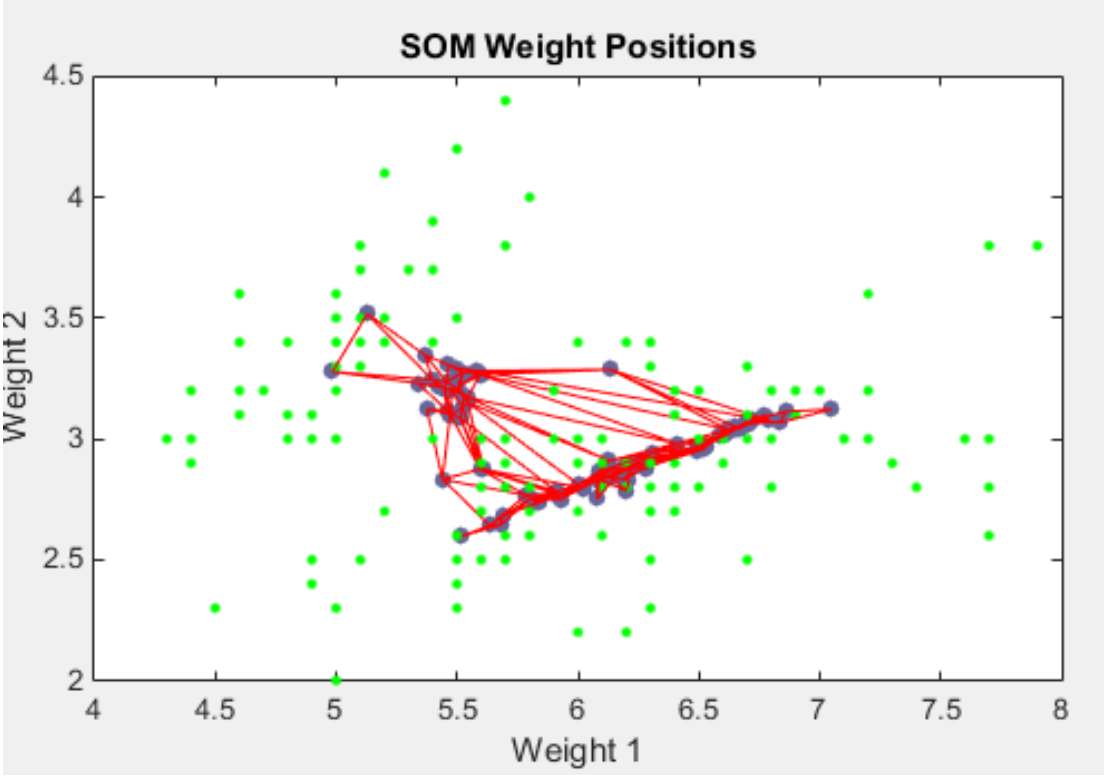

Hits
9
2
5
11
3
0
2
3
5
11
2
2
2
0
3
3
4
2
2
16
12
0
48

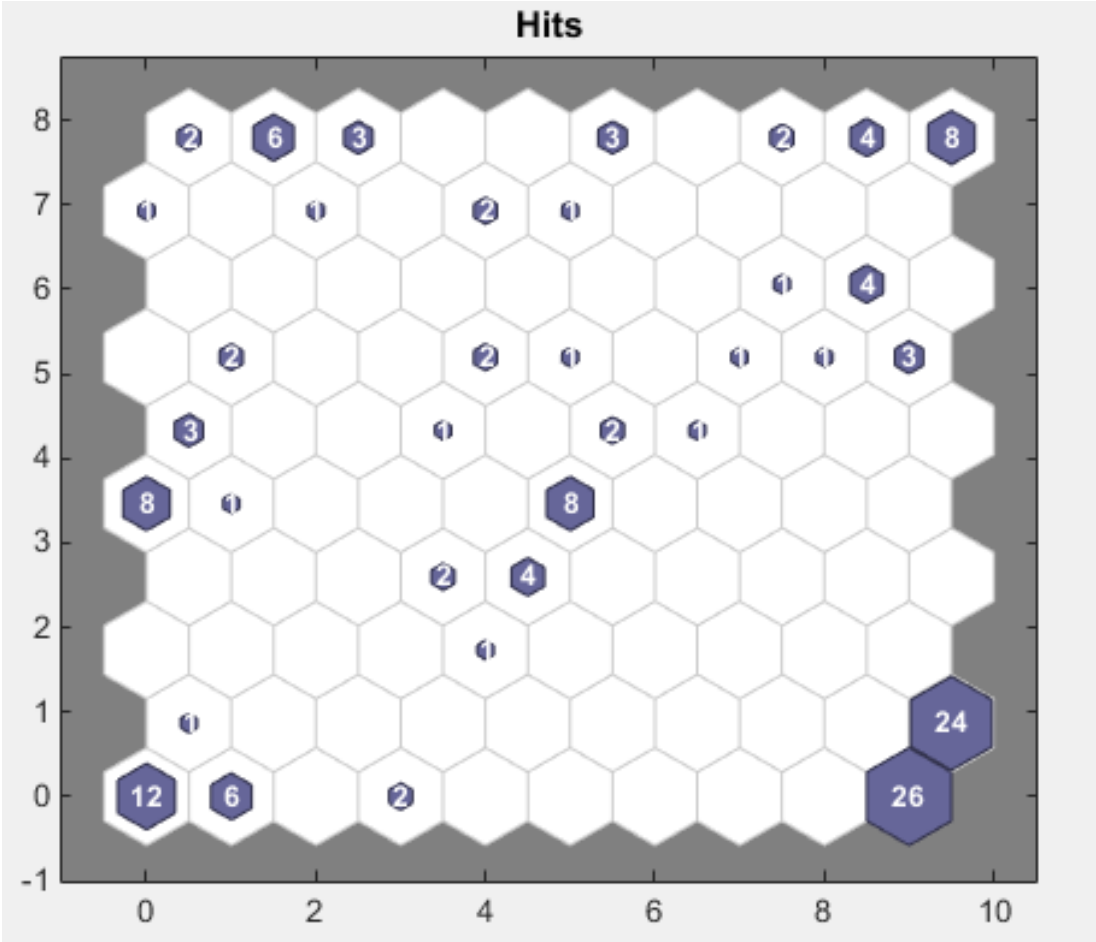

Weights from Input 1
Weights from Input 2
Weights from Input 3
Weights from Input 4

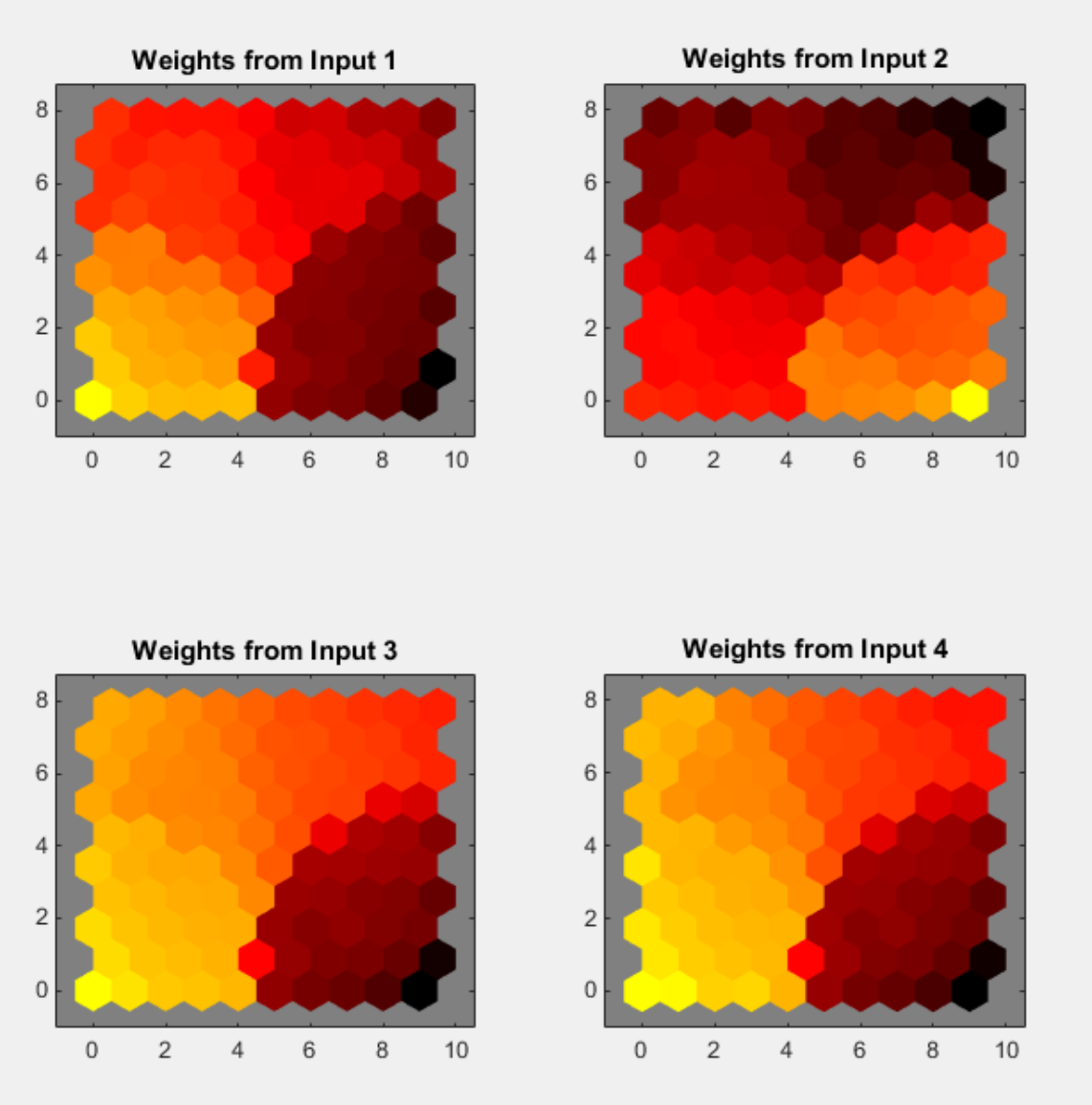
Dla [10,10]


Dane wejściowe
100
90
80
70
60
50
40
30
20
10
20
40
60
80
100
120
140


Dane testowe
100
90
80
70
60
50
40
30
20
10
5
10
15
20
25
30
35
40
45


SOM Weight Positions
Weight 2
Weight 1
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Hits
8
7
6
5
4
3
2
1
0
-1
0
2
4
6
8
10
2
6
3
3
2
4
8
1
1
2
1
1
4
2
2
1
1
1
3
3
1
2
1
8
1
8
2
4
1
1
24
12
6
2
26



6. Wnioski

Sieć Kohonena pozwala na analizę i prezentację wielowymiarowych informacji w przestrzeni o mniejszej liczbie wymiarów. Dzięki temu sieć po treningu potrafiła stworzyć mapę, na której można było umieścić kolejne testowane dane w celu dopasowania ich do odpowiednich grup.

Heksagonalna siatka neuronów, zastosowana w przedstawionym algorytmie umożliwia utworzonej sieci stworzenie większej liczby połączeń pomiędzy neuronami, niż w przypadku sieci prostokątnej (w konsekwencji sieć ma więcej możliwości w doborze odpowiednich wag dla poszczególnych neuronów).

Na podstawie wykresu pokazującego rozkład sił neuronów można zauważyć, że sieć korzystała z mechanizmu WTA (sieć nie modyfikowała wag tylko jednego neuronu, co wynika to z zastosowania normalizacji).

Kwiaty rodzaju Setosa, zostały rozpoznane nawet przy małym wymiarze wektora [5,5], jednakże sieć nie potrafiła rozróżnić pozostałych dwóch rodzajów irysów. Dla wymiaru wektora [8,8] również dobrze rozpoznaje ten charakterystyczny rodzaj kwiatów, oprócz tego całkiem nieźle radzi sobie z rozpoznawaniem odmiany Versicolor, jednakże dla kwiatów odmiany Virginica nie zawsze udaje jej się dobrze przyporządkować kwiaty. Tak samo wygląda to dla wymiaru wektora [10,10] , tutaj sieć również ma problemy z rozpoznawaniem tej odmiany. Wynika to prawdopodobnie z podobieństwa do siebie dwóch odmian, nie są one tak diametralnie różne od pozostałych jak odmiana Setosa.