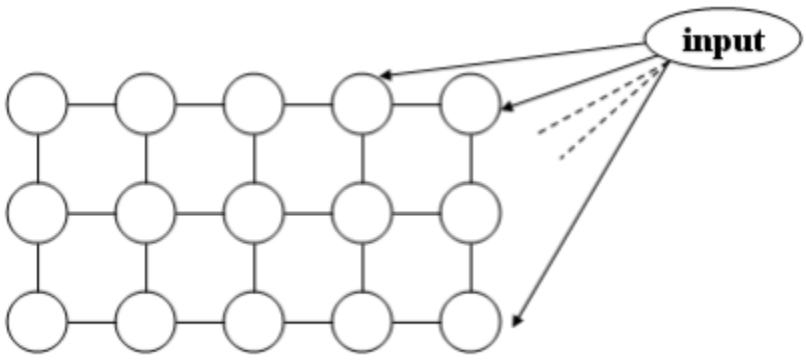
Akademia Górniczo-Hutnicza WIMiIP,
Inżynieria Obliczeniowa G01,
Małgorzata Kusik Nr. indeksu 293103

Podstawy Sztucznej Inteligencji
Sprawozdanie z Projektu 6
Budowa i działanie sieci Kohonena dla WTM

1. **Cel ćwiczenia:**
   Celem projektu było poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTM do

2. **Część teoretyczna:**

   Sieć Kohonena - jest szczególnym przypadkiem algorytmu realizującego uczenie się bez nadzoru. Jej 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. Sieć Kohonena znana jest 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 wezłów (im więcej, tym wyższą rozdzielczość będzie miała mapa).



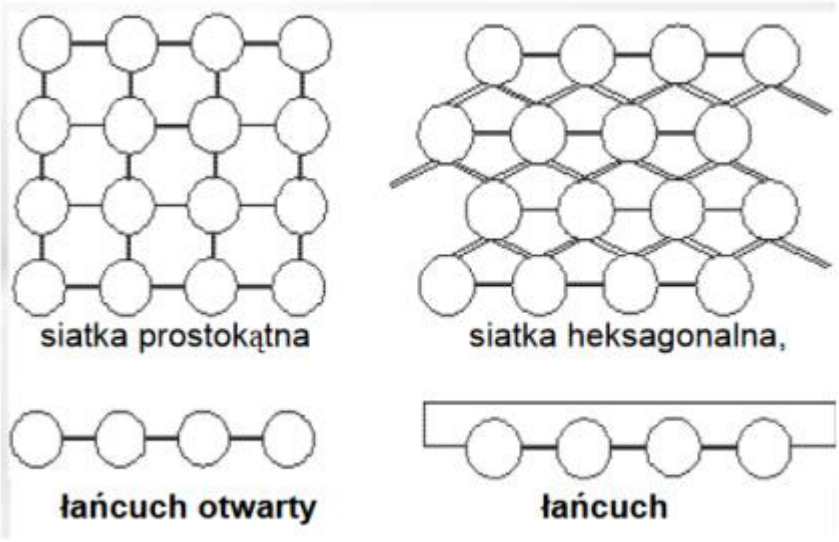
Topologia sieci Kohonena

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.

Winner Takes All (WTA) - zwycięzca bierze wszystko. Po przedstawieniu sieci wektora wejściowego, neuron najbardziej podobny do elementu prezentowanego (którego wagi są najbardziej podobne składowym wektora wejściowego) zostaje zmodyfikowany zgodnie z funkcją f tak, aby jego wagi były jak najbardziej zbliżone do wektora wejściowego. W programie Matlab, za pomocą biblioteki Neural Network Toolbox, zaimplementowałam sztuczną sieć neuronową. Danymi wejściowymi jest zestaw zaimplementowany w oprogramowaniu MATLAB o nazwie iris_dataset. Zawiera on opis 4 cech kwiatów irysa, tj. długość i szerokość płatka oraz długość i szerokość działki kielicha.

Topologię sieci Kohonena można określić poprzez zdefiniowanie sąsiadów dla każdego neuronu. Jednostka, kórej odpowiedź na nasze pobudzenie jest maksymalna nazywany obrazem pobudzenia. Sieć jest uporządkowana, jeśli topologiczne relacje między sygnałami wejściowymi i ich obrazami są takie same.

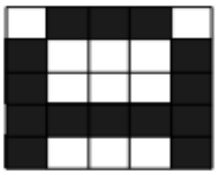


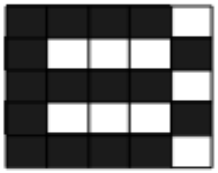
Topologie sieci Kohonena

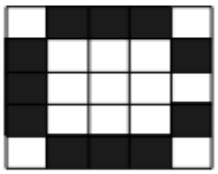
3. Opis wykonanego zadania:

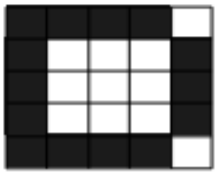
Do stworzenia naszej sieci wykorzystaliśmy pakiet MATLAB, narzędzie Neural Networking Training Tool. Następnie utworzyłam dane uczące, którymi były 20 pierwszych, wielkich liter

alfabetu łacińskiego. Każda tablica miała wymiar 5x5 pikseli, gdzie kolor biały oznaczał wartość 0, a kolor czarny oznaczał wartość 1.

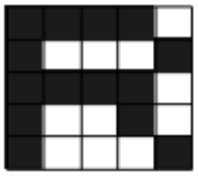

01110 10001 10001 11111 10001

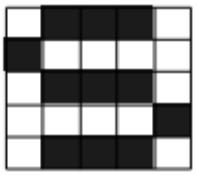

11110 10001 11110 10001 11110

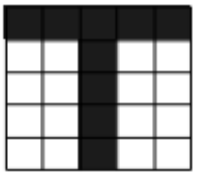

01110 10001 10000 10001 01110

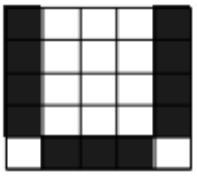

11110 10001 10001 10001 11110

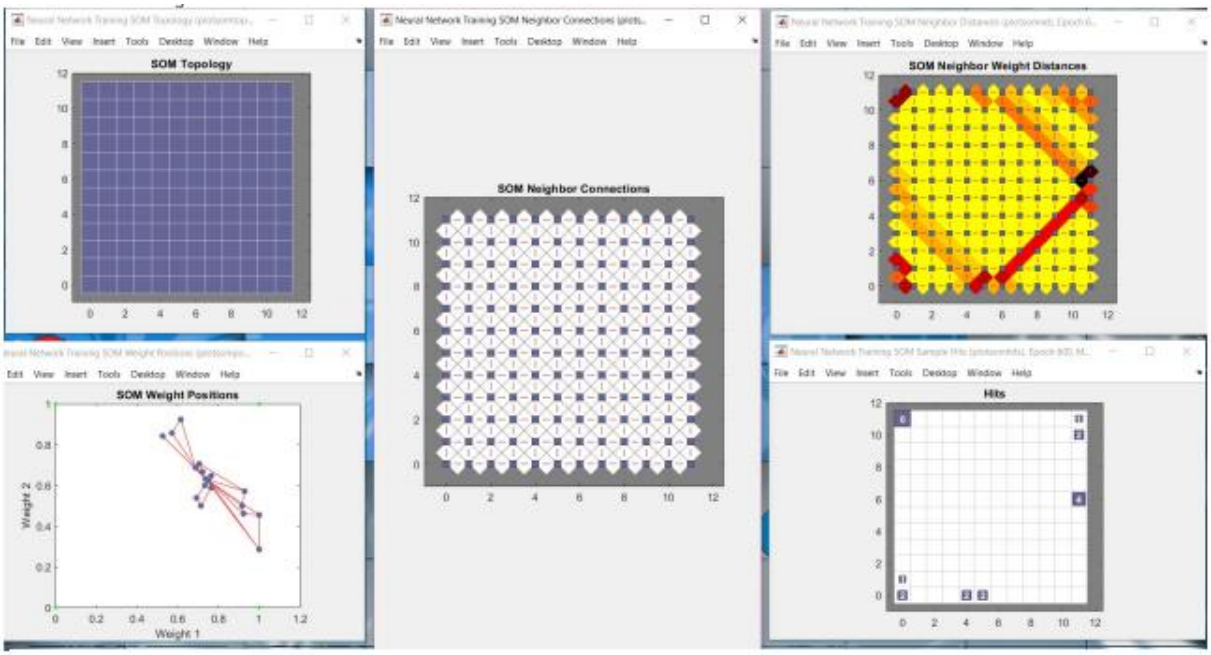
(…)

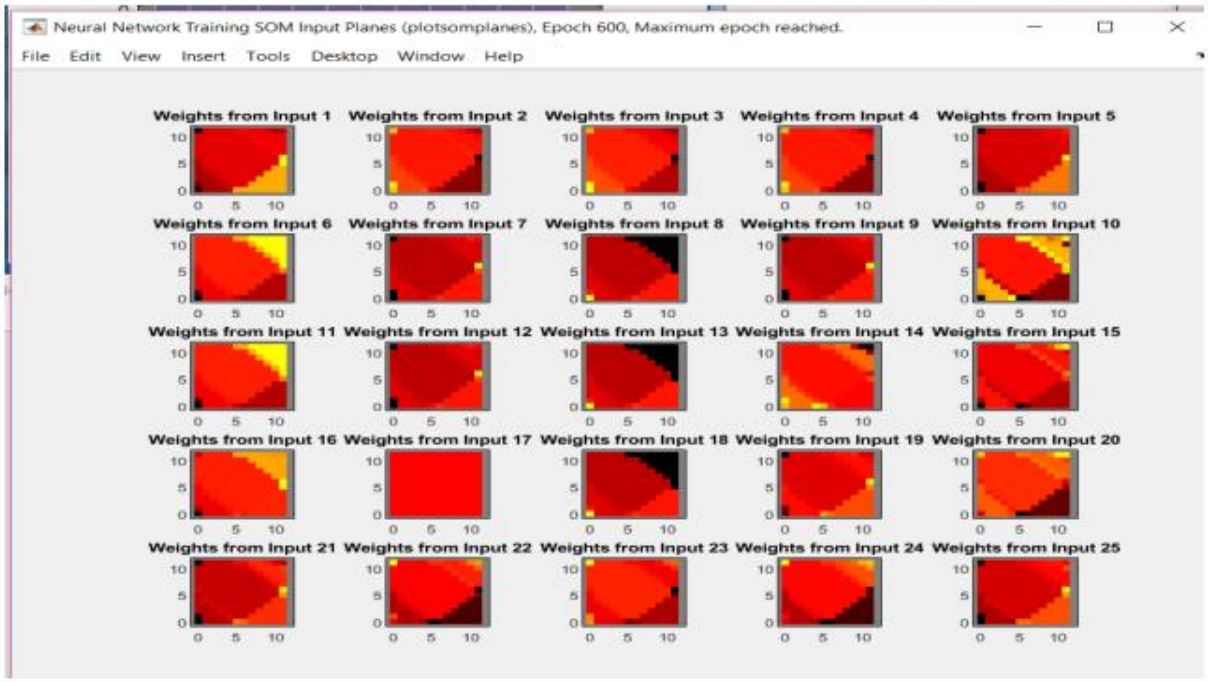

11110 10001 11110 10010 10001


01110 10000 01110 00001 01110

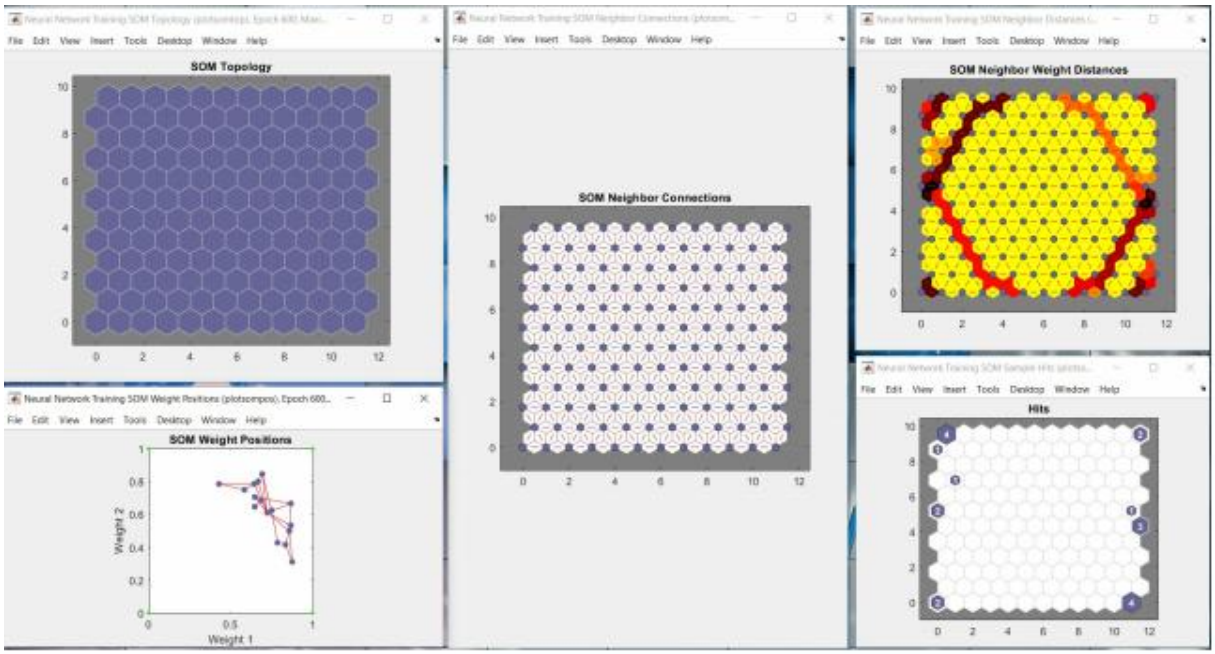

11111 00100 00100 00100 00100

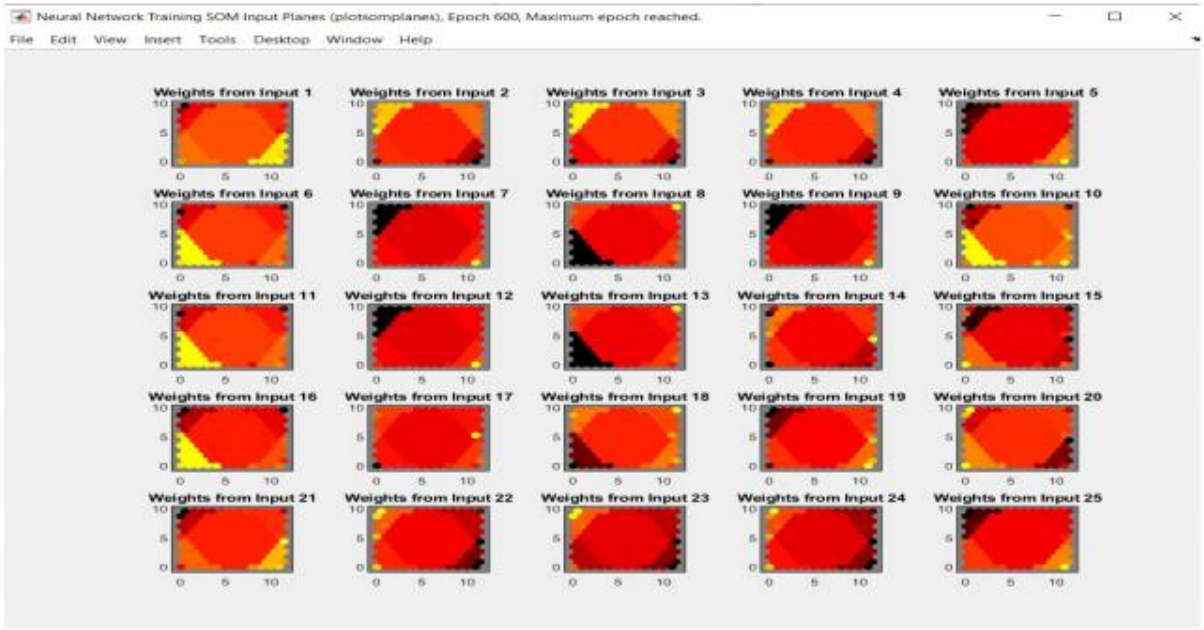

10001 10001 10001 10001 01110



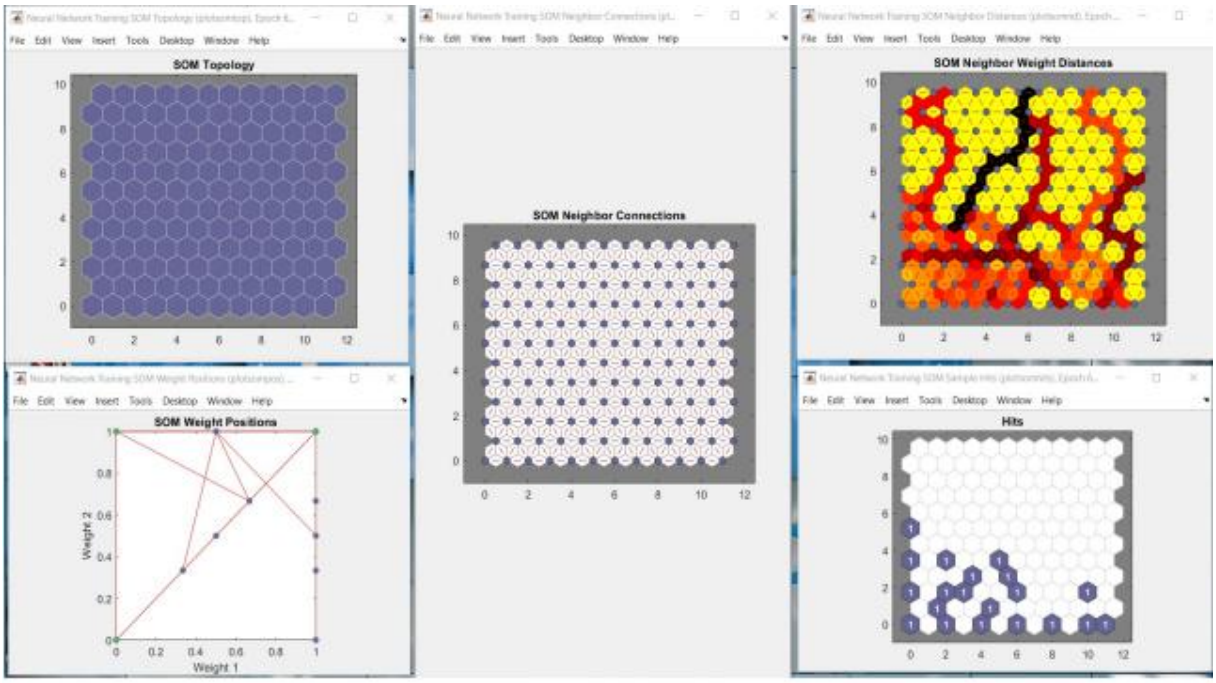


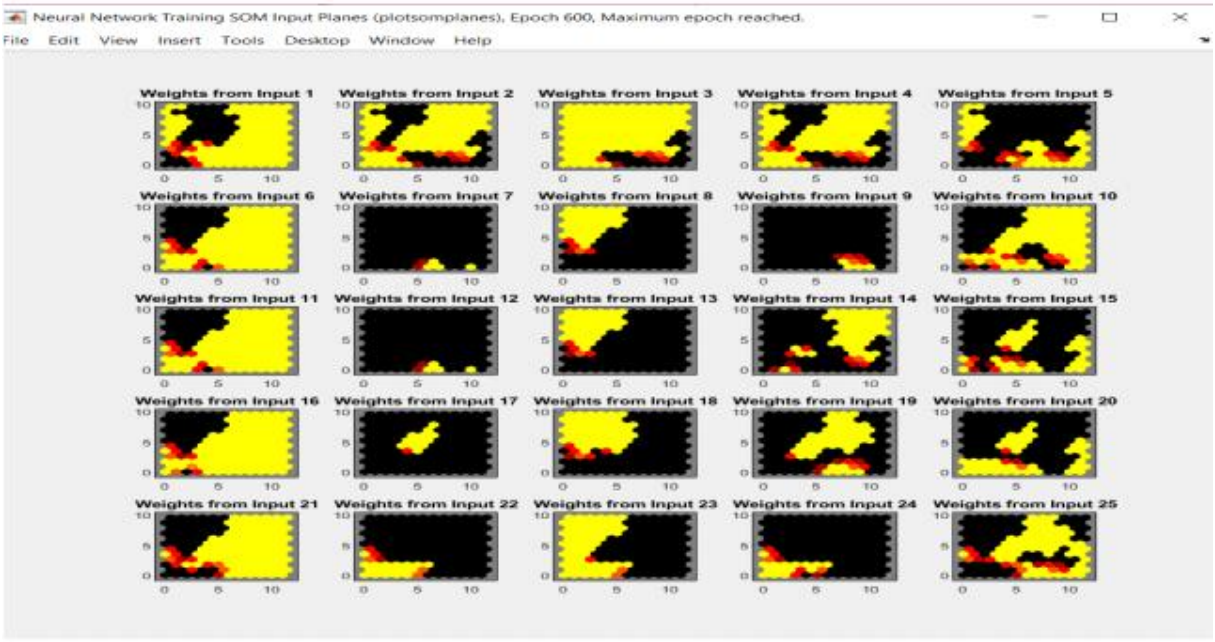
Wykresy dla topologii kwadratowej



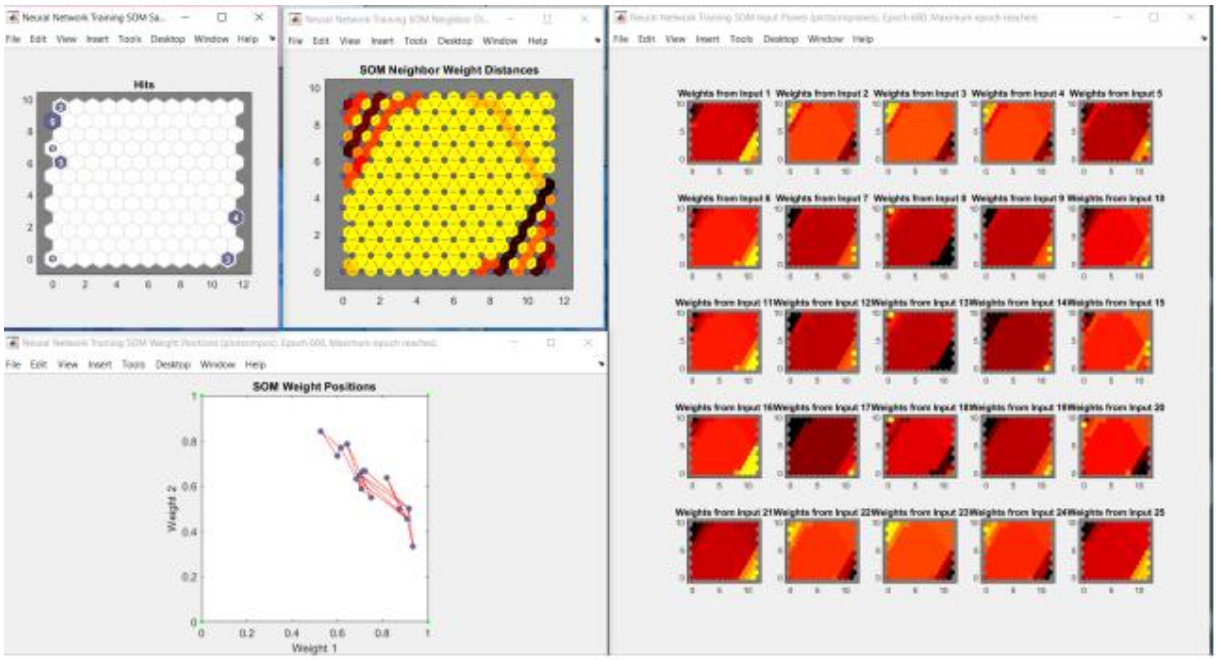


Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,5, sąsiedztwo = 0)

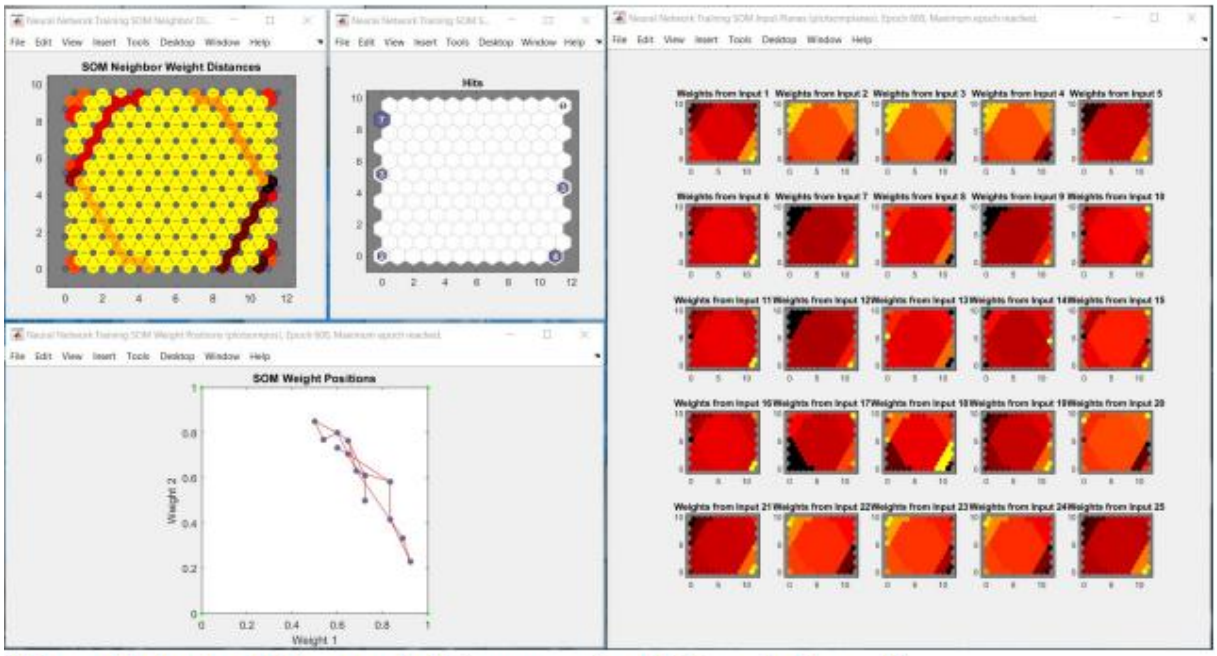




Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,5, sąsiedztwo = 3)



Wykresy dla topologii heskagonalnej (wsp. uczenia = 0.1, sąsiedztwo = 0)



Wykresy dla topologii heksagonalnej (wsp. uczenia = 0.75, sąsiedztwo = 0)

4. Wnioski:

Rozkłady sił koncentrują się wzdłuż brzegów siatki topologii w zależności od współczynnika uczenia – im jest on wyższy, tym siły bardziej skupiają się wzdłuż brzegów siatki i tylko w tych miejscach.

W zależności od współczynnika uczenia zmienia się rozkład wag i kształt ich powiązań, ale co ciekawe jest niska ilość neutronów martwych (niepowiązanych, niepodobnych do siebie), co zdecydowanie odróżnia system system WTM od WTA.

W metodyce WTM sąsiedztwo gra niebagatelną rolę i determinuje ono kształt korelacji i zależności – im jest ono wyższe tym podziałów jest mniej, ale bardziej porozrzucane po całej siatce i poszczególne neutrony znajdują się w innych kategoriach.

Rozkłady sil stają się równomierne, im sąsiedztwo jest wyższe. Również rozkład sił na siatce koncentruje się w poziomym kierunku, równolegle bądź współliniowo do dolnego brzegu siatki Kohonena.

Im wyższe sąsiedztwo, tym obiekty stają się bardziej do siebie podobne w pewnych cechach i zanika ilość podziałów.

W metodologii WTM obiekty są bardziej ze sobą powiązane, mimo zmian i w sąsiedztwie i we wsp. uczenia. Jedyne co się zmienia to ilość wag i kształt tych powiązań.

5. Listing kodu wraz z komentarzami:

```
close
all;
clear
all;
clc;
                % A B C D E F G H I J K L M N O P R S T U
        in_value = [  0 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1;
                      1 1 1 1 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0;
                      1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0;
                      1 1 1 1 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0;
                      0 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1;

                      1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1;
                      0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0;
                      0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0;
                      0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0;
                      1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1;

                      1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1;
                      0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0;
                      0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0;
                      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0;
                      1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 0 0 1;

                      1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1;
                      1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
                      1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0;
                      1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0;
                      1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 1;

                      1 1 0 1 1 1 0 1 0 0 1 1 1 1 0 1 1 0 0 0;
                      0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1;
                      0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 1;
                      0 1 1 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1;
                      1 0 0 0 1 0 0 1 0 0 1 1 1 1 0 0 1 0 0 0];
        dimensions = [12 12]; % wymiary wektora wyjściowego -> ilość neuronów i możliwości cech
        coverstep = 50; %etapy szkolenia w celu pokrycia przestrzeni wejściowej
        initNeighbor = 3; % wejściowy rozmiar sąsiedztwa
```

```
topologyFcn = 'hextop'; %funkcja topologiczna -> kształt, jaki będą przyjmować nasze dane
% mogą przyjmować kształt trójkąta, siatek kwadratowych, sześciokątów, itp.
distanceFcn = 'dist'; %funkcja dystansu neronów - miara euklidesowa (znormalizowana)
% domyślnym parametrem jest odległość między neuronami warstwy z
% uwzglednieniem ich położenia
net = selforgmap(dimensions, coverstep, initNeighbor, topologyFcn, distanceFcn); %tworzenie mapy
samoorganizacji
net.trainFcn = 'trainbu'; %uczenie bez nauczyciela
net.trainParam.epochs = 600;
net.trainParam.lr = 0.5; %współczynnik uczenia
[net, tr] = train(net, in_value); %trening sieci
y = net(in_value); %testowanie i zapis wyników osiągniętych przez sieć
indexOfOutput = vec2ind(y); % wskaźniki, do którego neuronu podobny jest dany neutron
```