Grupa lab. ***1***	*Przedmiot* ***Podstawy Sztucznej Inteligencji***	*Data wykonania.* ***29.12.2018***
Nr ćwicz. ***5***	*Temat ćwiczenia.* ***Budowa i działanie sieci Kohonena dla WTA***	
Imię i nazwisko. ***Katarzyna Giądła***		*Ocena i uwagi*

Część teoretyczna

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

W naszym modelu przyjęliśmy numeryczny opis cech kwiatów pod nazwą *Iris flower data set* (inaczej tez zwn. zestawem danych o irysach Fishera bądź zbiorem danych o irysach Andersona). Jest to wielowymiarowy zbiór danych stworzony przez brytyjskiego statystyka i biologa Ronalda Fishera w 1936 r. Zestaw danych składa się z 50 próbek z każdego z trzech specyficznych gatunków irysów. Próbki te były badane pod kątem długości i szerokości płatka i działki kielicha. Fisher opracował liniowy model pozwalający odróżniać gatunki między sobą.

Istotnym pojęciem z punktu widzenia sieci samoorganizujących się jest klaster. Jest to grupa podobnych obiektów. Analiza klastrowa odnosi się do tworzenia grup obiektów, które są do siebie bardzo podobne, ale bardzo się różnią od obiektów w innych skupiskach.

T Tworzenie map samoorganizacji - uczenie danych klastra w oparciu o podobieństwo, topologię z preferencją (bez gwarancji) przypisania takiej samej liczby instancji do każdej klasy. Mapy samoorganizujące się można wykorzystać również do zmniejszania wymiarowości danych oraz do tworzenia klastrów danych.

Sieć Kohonena to sieć samoorganizująca się korzystająca z uczenia rywalizacyjnego (*competitive learning*). Na początku neurony mają przypadkowe wartości wag, a następnie uczą się rozpoznawać dane prezentowane i zbliżają się odpowiednio do obszarów zajmowanych przez te dane. Po każdej prezentacji wzorca *x(k)* zwycięzcą zostaje tylko ten jeden neuron, którego metryka jest najbliższa prezentowanemu wzorcowi. Wygrany neuron zostaje jedynym pobudzanym przy konkretnej obserwacji wejściowej – na wyjściu przyjmie wartość *1*. Wagi zwycięzcy będą dostrajane w danym kroku uczenia.

Funkcjonowanie samoorganizujących się sieci neuronowych odbywa się w 3 etapach:

1. konstrukcja
2. uczenie
3. rozpoznawanie

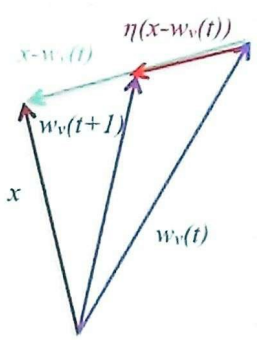
Neurony dopasowują swoje wagi w ten sposób, że przy prezentacji grup wektorów wejściowych zbliżonych do siebie zwycięża zawsze ten sam neuron. Mapy samoorganizujące się mogą być tworzone z dowolnym poziomem szczegółowości.

Algorytm uczenia WTA:

1. Przyjmujemy losowe, znormalizowane wartości wag poszczególnych neuronów

2. Po podaniu pierwszego wektora wejściowego *x* wyłaniany jest zwycięzca o numerze *v*.

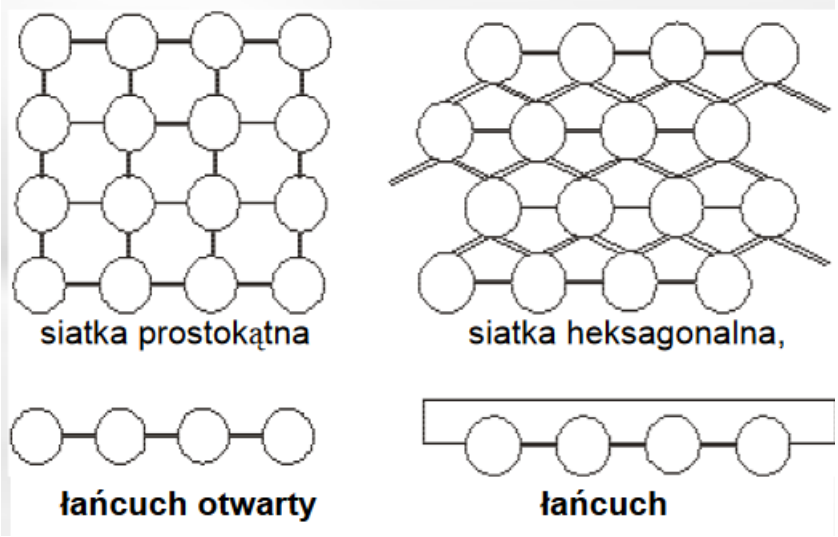
$$w_v^T x = \max_{i=1,2\ldots,k} (w_i^T x)$$



3. Aktualizacja wag neuronu zwycięzcy (neurony przegrywające mają na wyjściu stan *0*, co blokuje proces aktualizacji ich wag). Aktualizacja ta odbywa się według reguły Kohenena: $w_v(t+1)=w_v(t)+\eta[x-w_v(t)]$, gdzie η – wsp. uczenia ($0<\eta<1$) i maleje w miarę postępu nauki.

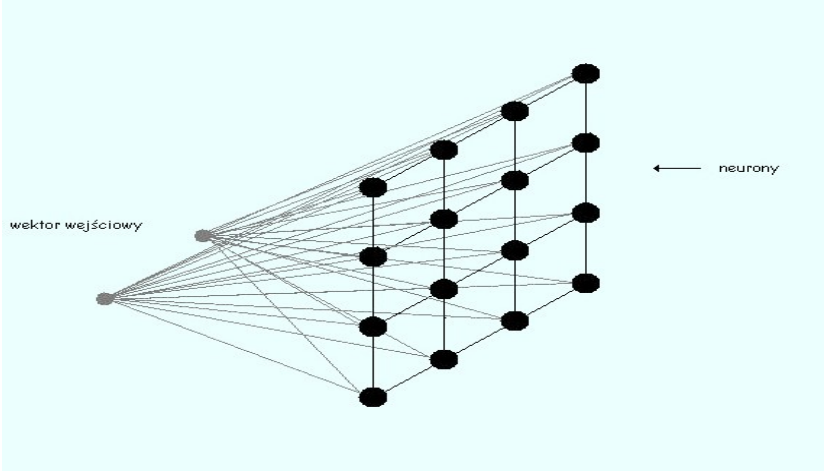
Wektor wag neuronu zwycięzcy jest zwiększany o ułamek różnicy *x*-*w*, w wyniku czego w następnych krokach lepiej odtwarza rozpatrywany wektor wejściowy.

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

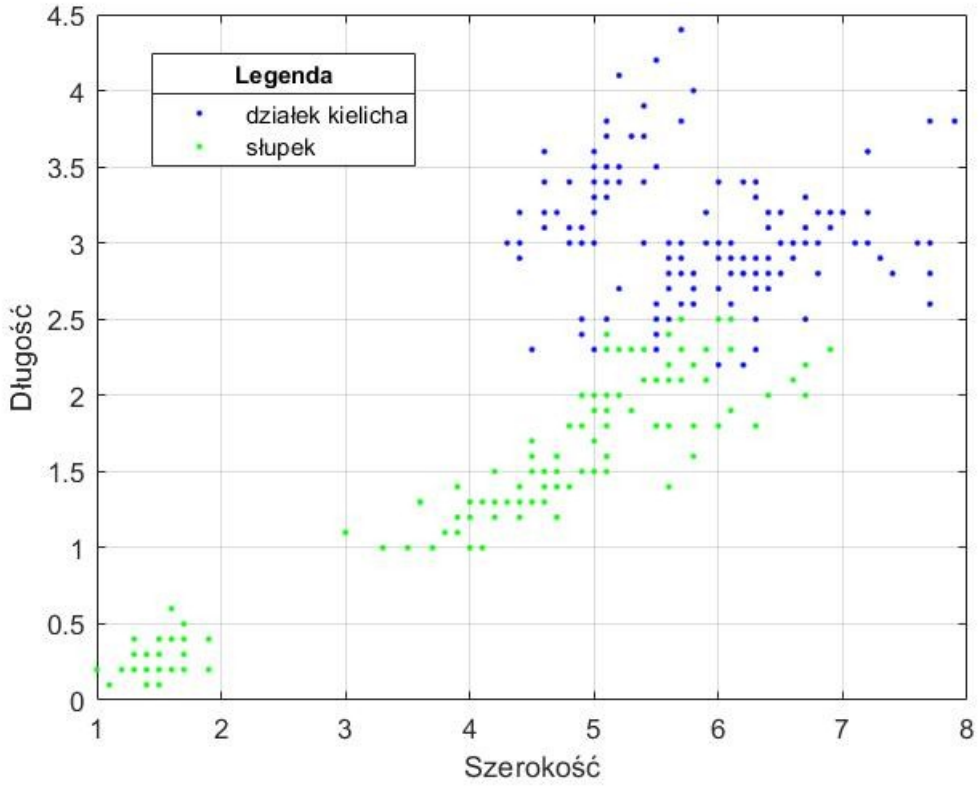
Ważnym parametrem jest określenie ile neuronów obok ma podlegać uczeniu w przypadku zwycięstwa danego neuronu.



Powstawanie prostokątnej dwuwymiarowej sieci Kohonena

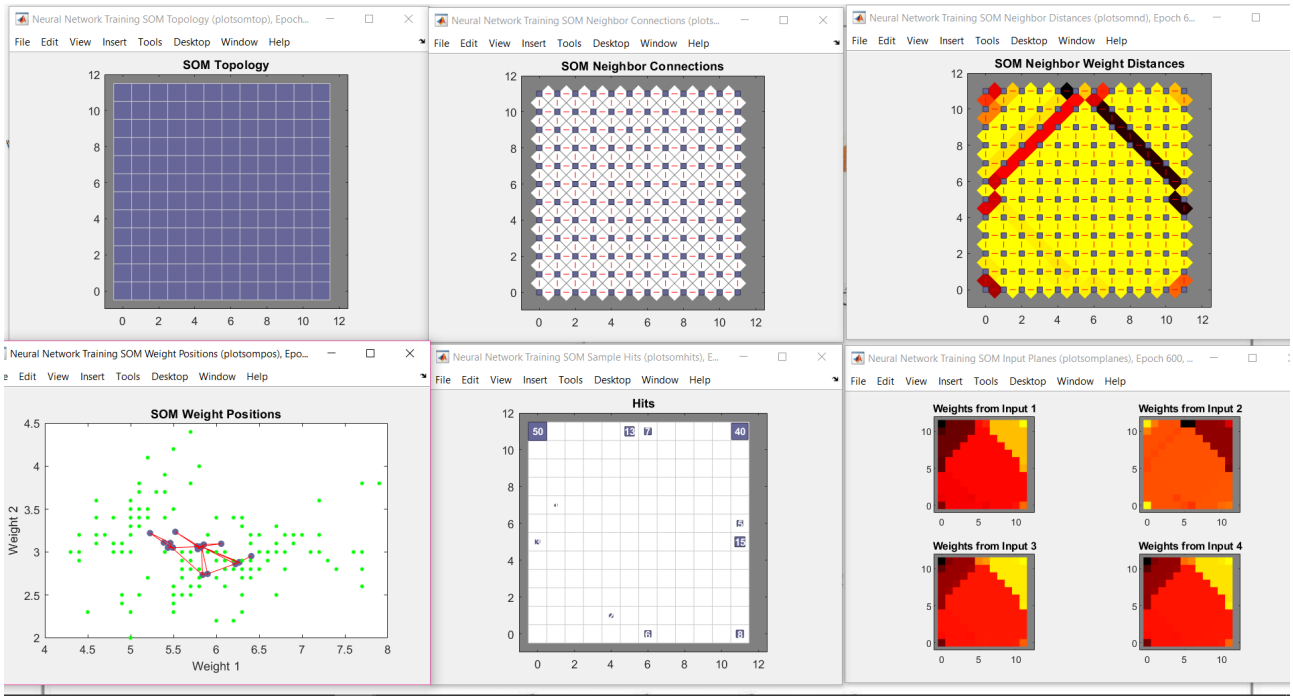
Część praktyczna

Do stworzenia naszej sieci wykorzystaliśmy pakiet *MATLAB*, w szczególności narzędzie *Neural Networking Training Tool.* Naszymi danymi wejściowymi był wbudowany w pakiecie zbiór danych *iris_dataset.* Udało się również wygenerować dwuwymiarowy wykres dla parametrów płatka i działki kielicha.

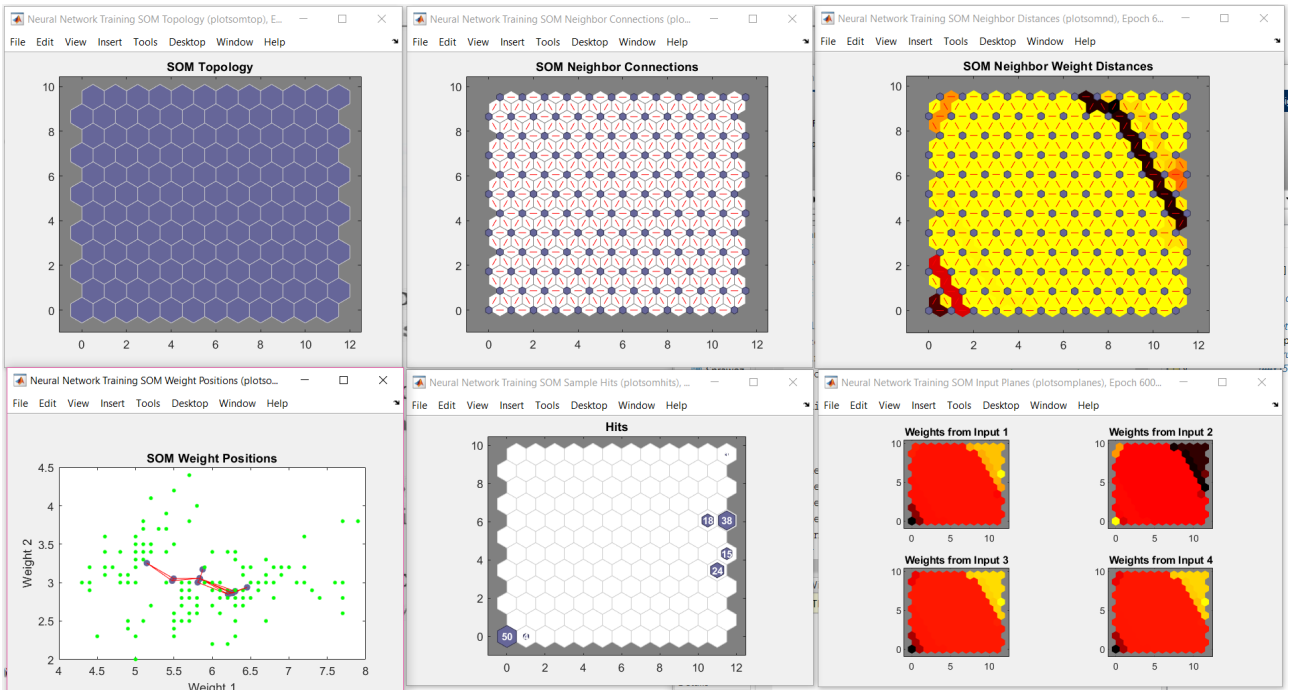


Wykorzystaliśmy funkcję do tworzenia map samoorganizacji *selforgmap.* W ramach wykonywania programu otrzymaliśmy 6 różnych wykresów, które mają swoje znaczenie:

- *SOM Topology* – jeden kształt symbolizuje jeden neutron. Są one rozmieszczone w siatce o określonych wymiarach. Ułożenie jest nieprzypadkowe – ich sąsiedztwo może wskazywać na ich podobieństwo.
- *SOM Neighbor Distances* – Na tej mapie otrzymamy możliwość sprawdzenia, jak bardzo połączone są ze sobą poszczególne neutrony – czyli jak silne podobieństwo one wykazują. Im jaśniejsze połączenie, tym bardziej te dane są do siebie podobne. Ciemne linie mogą zatem oznaczać granice klas.
- *SOM Neighbor Connections* – na tym wykresie umieszczone są połączenia między sąsiadami. Sąsiadami są zwykle próbki do siebie podobne.
- *SOM Weight Planes* – jest to zestaw wykresów, który wskazuje na rozkład wag poszczególnych neutronów w zależności od cechy. Im ciemniejszy kolor, tym większą wagę dany neuron skupia. Im więcej neuronów o podobnych kolorach w danym sąsiedztwie, tym te neurony są bardziej ze sobą skorelowane
- *SOM Sample Hits* – ten wykres pokazuje nam, ile podobnych wyników otrzymaliśmy dla danej klasy– im wyższa liczba, tym więcej obiektów o podobnych cechach można wykazać
- *SOM Weight Positions* – zielone kropki oznaczają dane wejściowe, a linie je łączące wskazują na korelacje pomiędzy poszczególnymi neuronami.



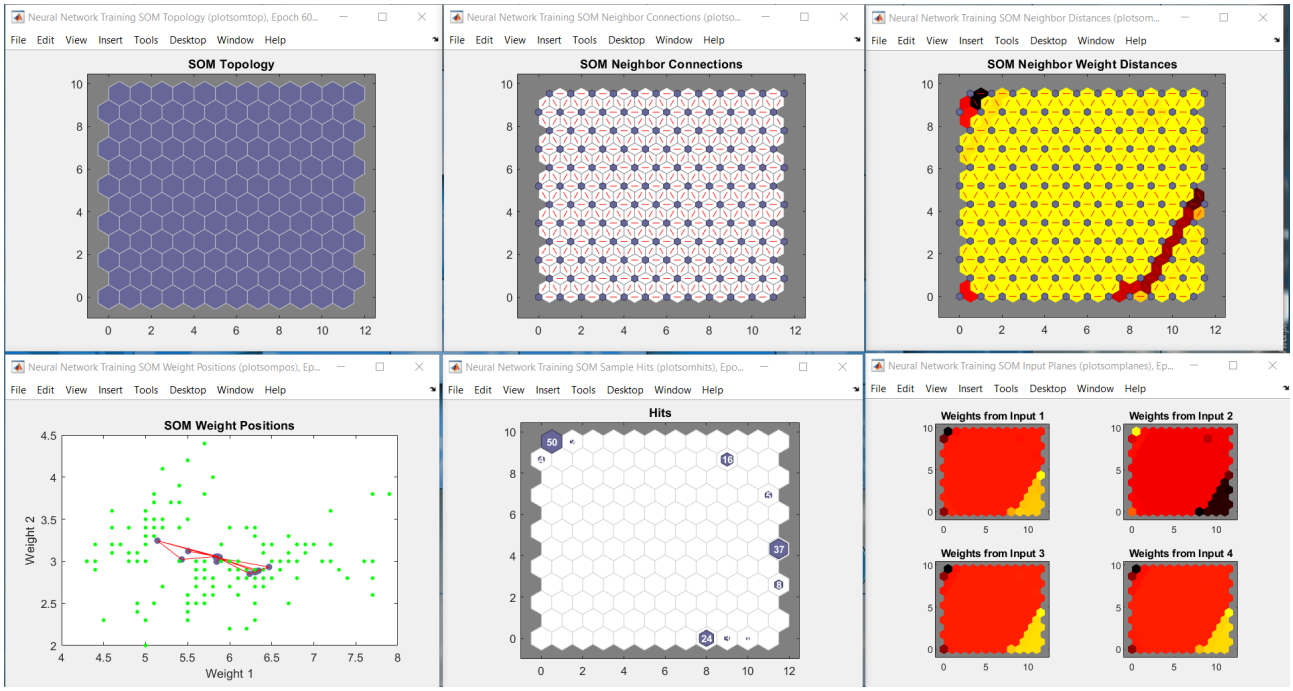
Wykresy dla topologii kwadratowej (wsp. uczenia = 0,5)



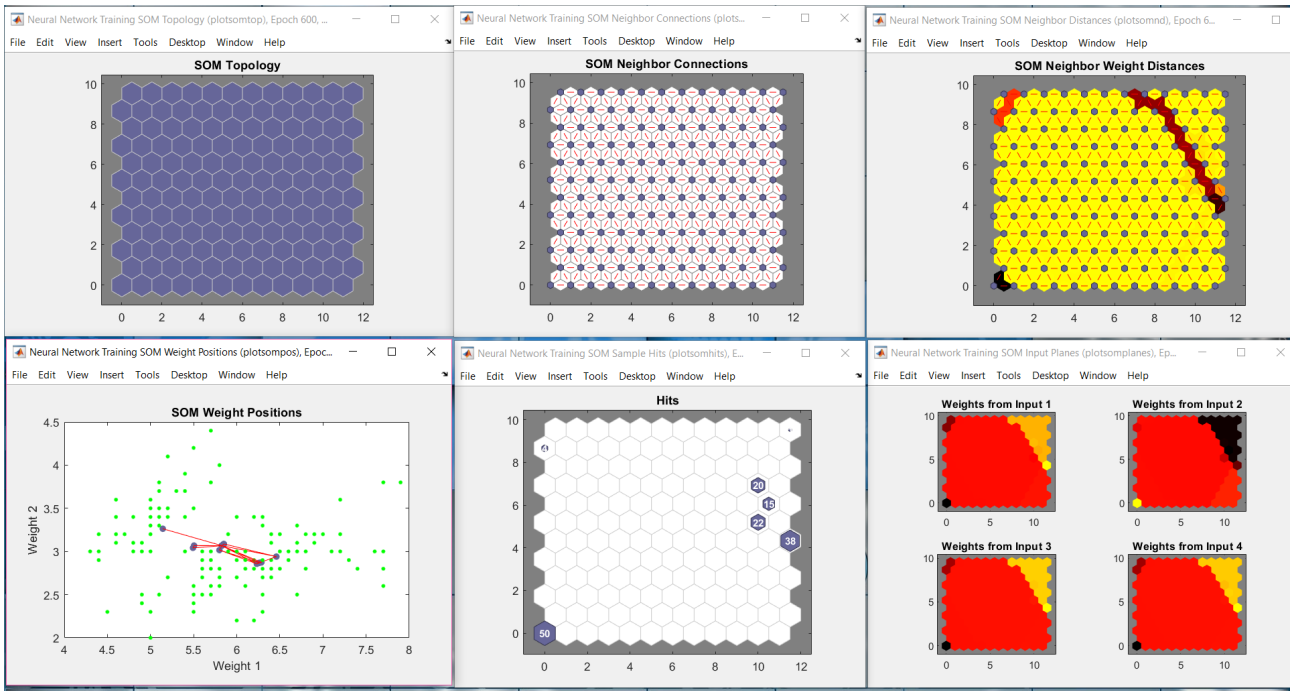
Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,5)

Poniższe wykresy różnią się zastosowaną topologią. Pakiet *MATLAB* daje podgląd tylko dla topologii heksagonalnej i kwadratowej. Przyjęłam, wymiary siatki neuronów 12x12 i 50 etapów szkolenia w celu pokrycia siatki oraz maksymalną liczbę epok uczenia równą 600.

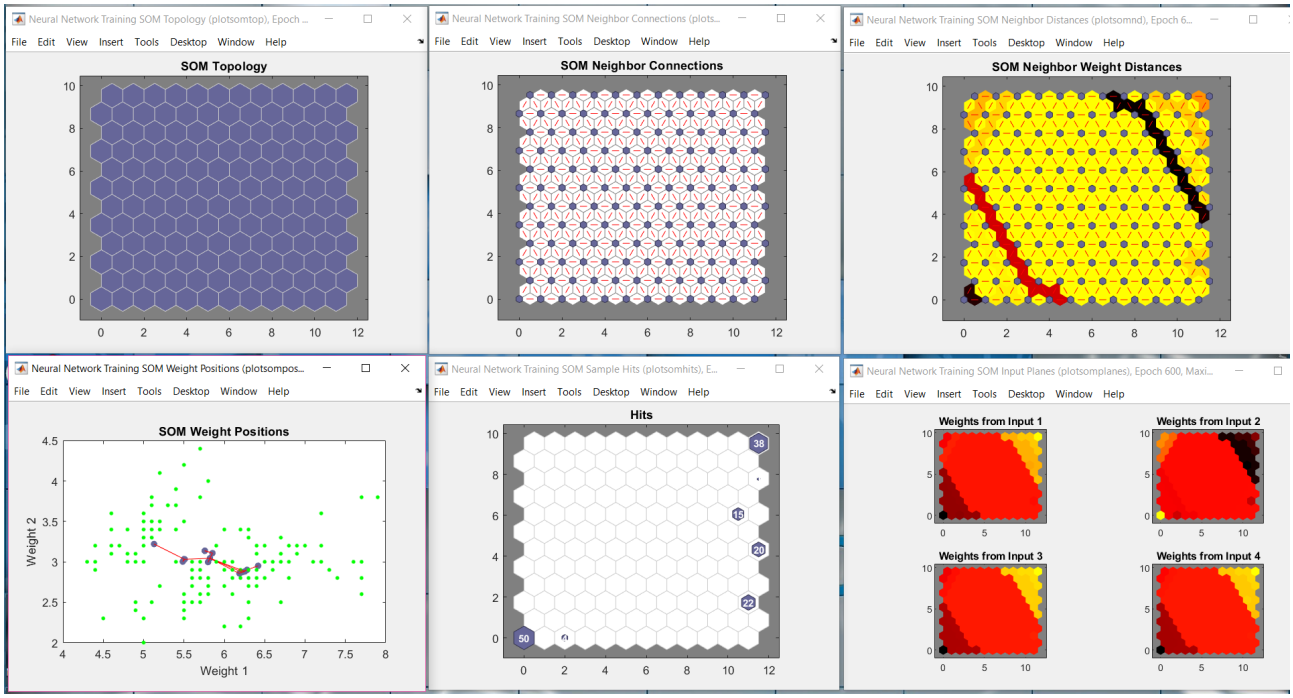
Ponieważ pewne cechy i tendencje lepiej są naświetlone poprzez topologię heksagonalną, dlatego moje dalsze rozważania będę prowadzić względem tej właśnie topologii. Jednym z parametrów uczenia, od którego zależy algorytm uczenia jest wsp. uczenia. Sprawdziłam dla pięciu współczynników uczenia (*0, 0.25, 0.5, 0.75, 0.9*).



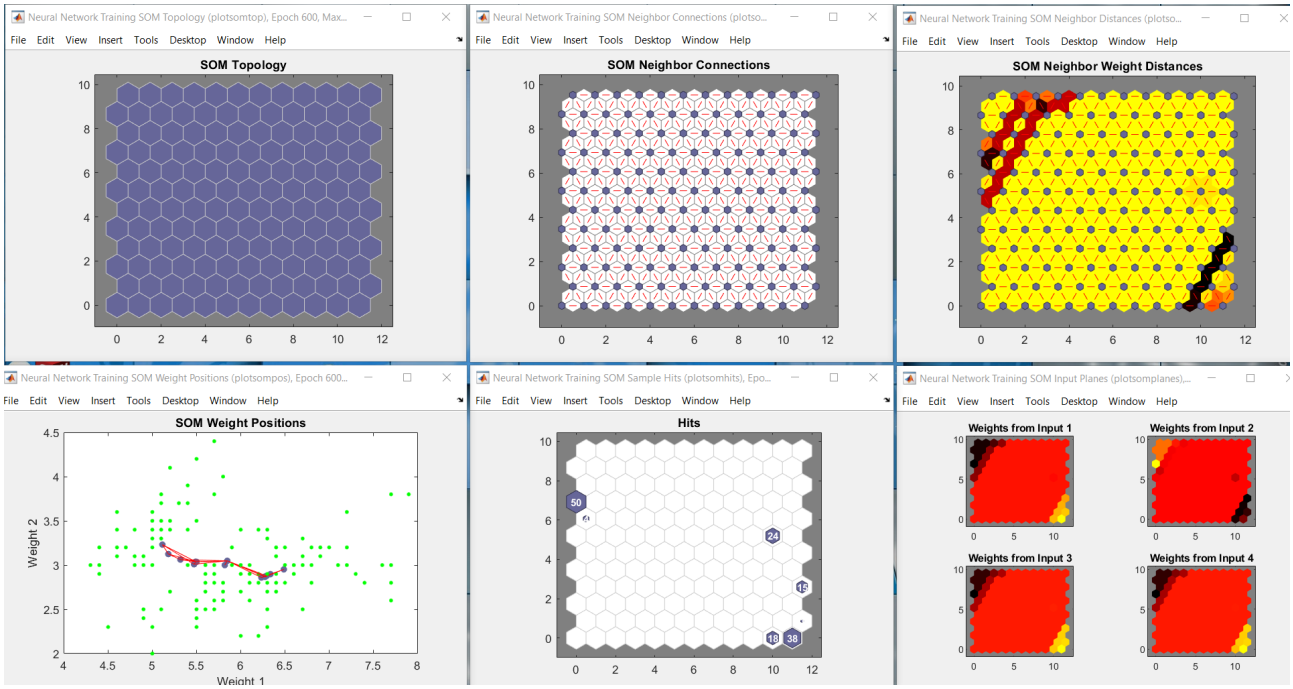
Wykresy dla topologii heksagonalnej (wsp. uczenia = 0)



Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,25)



Wykresy dla topologii heksagonalej (wsp. uczenia = 0,75)



Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,9)

Wnioski

- Topologia kwadratowa pozwala na podgląd pewnych tendencji, jednak pewne informacje na temat zależności są zbyt uogólnione. Bardziej szczegółową informację o możliwych korelacjach i podobieństwach daje nam topologia heksagonalna.
- Wynik działania topologii jest podobny, jednak w naszym przypadku topologia heksagonalna doliczyła się mniejszej ilości podobieństw, a co za tym idzie mniej klas danego obiektu.
- Algorytmy uczenia bez nauczyciela można wykorzystać tam, gdzie istnieje potrzeba grupowania elementów pod względem pewnych cech.

- Algorytm WTA nie zwraca uwagi na sąsiedztwo – dla niego najważniejsza jest aktualizacja wag. Istotnym czynnikiem może okazać się współczynnik uczenia – w pewnym zakresie kształtują się pewne tendencję – jednak za niski, bądź za wysoki powoduje wytworzenie się zbyt szczegółowego podziału na grupy, co nie zawsze jest naszym celem.
- Im wyższy był współczynnik uczenia, tym czas uczenia był wyższy.
- W zależności od współczynnika uczenia otrzymujemy też różne połączenia korelacyjne – im był on wyższy, tym te korelacje zachodziły rzadziej.
- W pewnych przedziałach rozkład wag dla poszczególnych klas jest podobny, co pozwala domniemywać, że w tych neuronach kumulują się pewne dominujące cechy.

Listingi kodów źródłowych

```
close all; clear all; clc;

in_value = iris_dataset; %danymi wejściowymi jest
% zbiór danych, gdzie w I i II kolumnie mamy dane o długości
% i szerokości działki kielicha, a w III i IV kolumnie
% o długości i szerokości płatka (wszystkie wymiary podane są
w cm)

plot(in_value(1, :) ,in_value(2, :), 'b.', in_value(3, :),
in_value(4, :), 'g.');
% narysowanie dwuwymiarowego wykresu zależności
% długości od szerokości
hold on; grid on; %hold - zapamiętanie aktualnego wykresu,
% kiedy tworzą się inne wykresy
% grid - wyświetlanie linii siatek wykresu0

dimensions = [12 12]; % wymiary wektora
coverstep = 50; %etapy szkolenia w celu pokrycia przestrzeni
wejściowej
initNeighbor = 0; % 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 = 0.9; %współczynnik uczenia
[net, tr] = train(net, in_value); %trening sieci
y = net(in_value); %testowanie i zapis wyników osiągniętych
przez sieć
```