Gabriela Ciołek

Nr indeksu: 293083

Inżynieria Obliczeniowa

Wydział Inżynierii Metali i Informatyki Przemysłowej

Akademia Górniczo-Hutnicza w Krakowie

**Sprawozdanie 6**

Przedmiot: Podstawy sztucznej inteligencji

Temat: Budowa i działanie sieci Kohonena dla WTM

**CEL:**

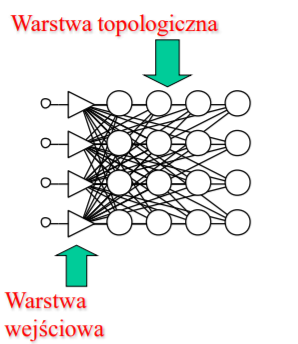
Celem ćwiczenia jest poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTM do odwzorowywania istotnych cech liter alfabetu.

**TEORIA:**

*SIECI KOHONENA*

Sieci Kohonena są jednym z podstawowych typów sieci samoorganizujących się. Stanowią one synonim całej grupy sieci, w których uczenie odbywa się metodą samoorganizującą typu konkurencyjnego. Polega ona na podawaniu na wejścia sieci sygnałów, a następnie wybraniu w drodze konkurencji zwycięskiego neuronu, który najlepiej odpowiada wektorowi wejściowemu. Dokładny schemat konkurencji i późniejszej modyfikacji wag synaptycznych może mieć różną postać. Wyróżnia się wiele podtypów sieci opartych na konkurencji, które różnią się dokładnym algorytmem samoorganizacji.

STRUKTURA SIECI KOHONENA



Cechy charakterystyczne:

• sieć uczy się bez nauczyciela,

• uporządkowane neurony wyjściowe,

• jest konkurencja i wyłaniany jest neuron „zwycięski”,

• ważną rolę odgrywa „sąsiedztwo”,

• w wyniku uczenia powstaje mapa topologiczna,

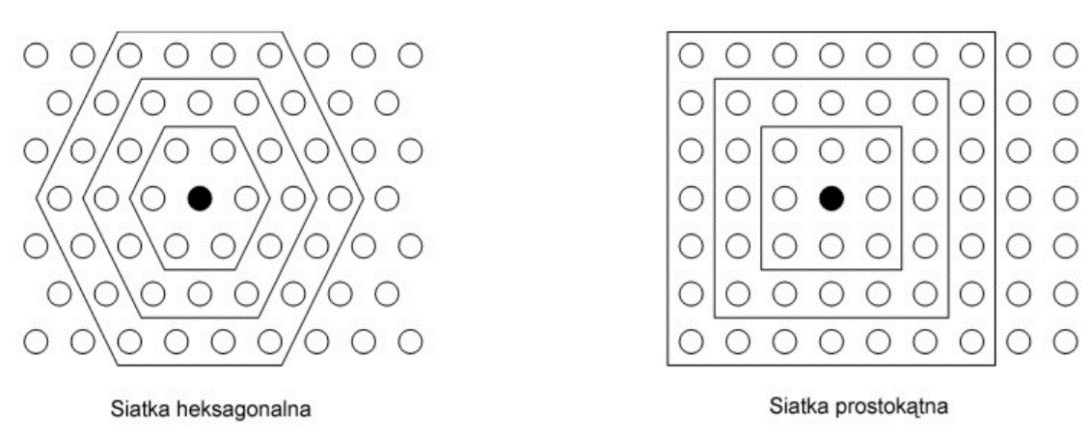
• po uczeniu można ustalić, jakie znaczenie mają poszczególne rejony mapy topologicznej - ale wyłącznie na podstawie analizy konkretnych przykładów danych wejściowych.

*ALGORYTM WTM*

• Algorytmy WTA, w których tylko jeden neuron może podlegać adaptacji w każdej iteracji, są algorytmami słabo zbieżnymi, szczególnie przy dużej liczbie neuronów

• W praktyce zostały one zastąpione algorytmami WTM (ang. Winner Takes Most), w których oprócz zwycięzcy uaktualniają swoje wagi również neurony z jego sąsiedztwa: dla wszystkich neuronów i należących do sąsiedztwa Sj\* zwycięzcy. Wagi są aktualizowane według wzoru:





**LISTING KODU WRAZ Z KOMENTARZAMI:**

close all; clear all; clc;

%WYGENEROWANE LITERY W KOLUMNOWEJ REPREZENTACJI

% A B C D E F G H I J K L M N O P R S T U

input = [0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1;

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

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

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

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

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

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

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

1 0 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0];

dimensions = [10 10]; % wymiary wektora

coverstep = 50; %etapy szkolenia

initNeighbor = 2; % wejściowy rozmiar sąsiedztwa

topologyFcn = 'hextop'; %dane przyjmuja ksztalt szescianow

distanceFcn = 'dist'; %funkcja dystansu neuronów

% TWORZENIE SIECI KOHONENA

net = selforgmap(dimensions, coverstep, initNeighbor, topologyFcn, distanceFcn);

% USTALENIE WSPÓŁCZYNNIKÓW

net.trainParam.epochs = 500; % maksymalna liczba epok treningowych

net.trainParam.lr = 0.5; %współczynnik uczenia

%TRENING SIECI

[net, tr] = train(net, input); %trening sieci

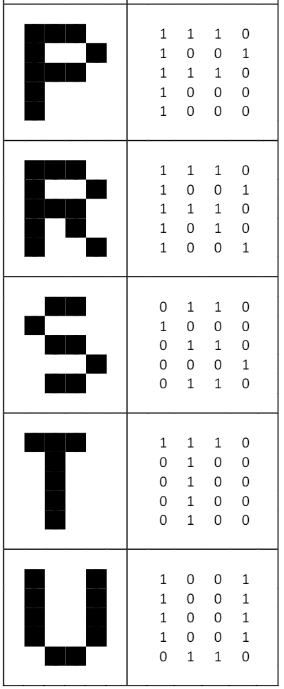
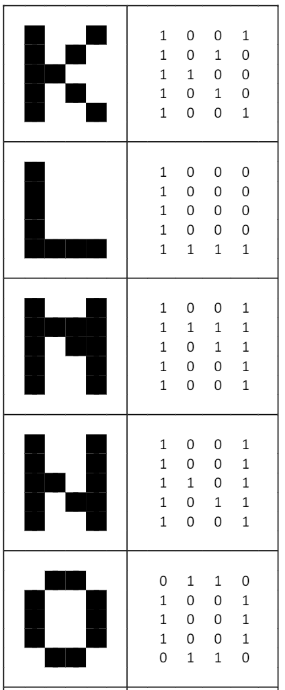
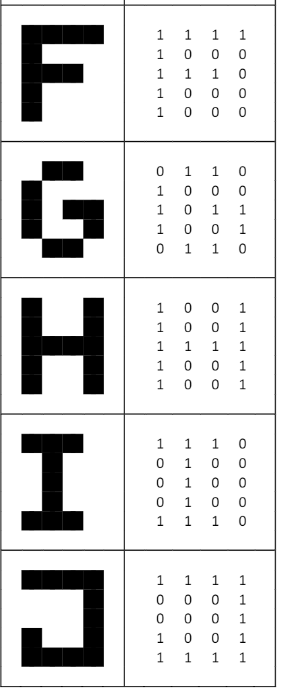
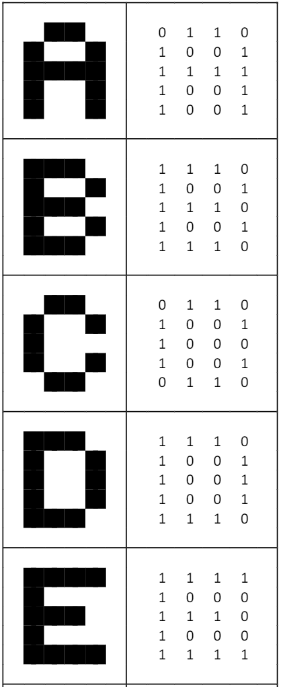
y = net(input); %testowanie i zapis wyników osiągniętych przez sieć

**OPIS WYKONANEGO ZADANIA:**

Do moich zadań należało:

1. Wygenerowanie danych uczących i testujących, zawierających 20 dużych liter dowolnie wybranego alfabetu w postaci dwuwymiarowej tablicy np. 4x5 pikseli dla jednej litery

Wygenerowałam tablicę 20 wielkich liter z polskiego alfabetu. Tablica ma rozmiar 4x5 pikseli dla jednej litery. Tablica zawiera 20 pierwszych liter alfabetu (tj. A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, R, S, T, U). Tablicę tą przerobiłam w sposób zero-jedynkowy. Tam gdzie znajduje się fragment litery pole przyjmuje wartość 1, dla pola wolnego jest to 0.



1. Przygotowanie (implementacja lub wykorzystanie gotowych narzędzi) sieci Kohonena i algorytmu uczenia opartego o regułę Winner Takes All (WTA).

Do stworzenia sieci Kohonena, która jest siecią samoorganizującą się wykorzystałam zawartą w pakiecie Matlab funkcję *selforgmap().* Przyjmuje ona następujące argumenty:

(dimensions, coverstep, initNeighbor, topologyFcn, distanceFcn)

gdzie:

dimension-wymiary wektora,

coverstep – etapy szkolenia,

initNeighbor – wejściowy rozmiar sąsiedztwa,

topologyFcn – kształt, który przyjmą dane,

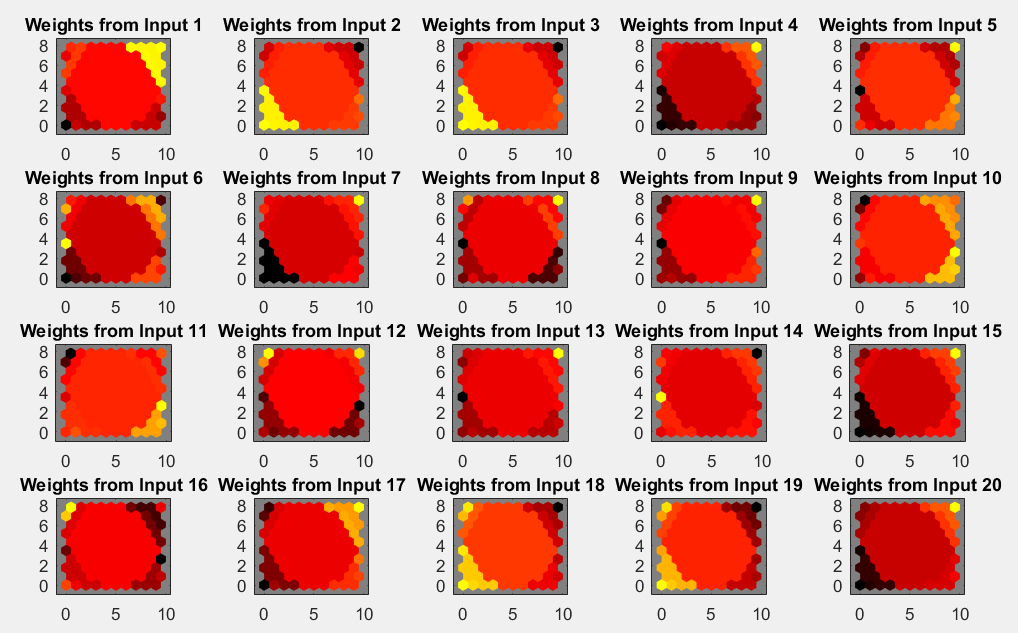
distanceFcn – dystans neuronów

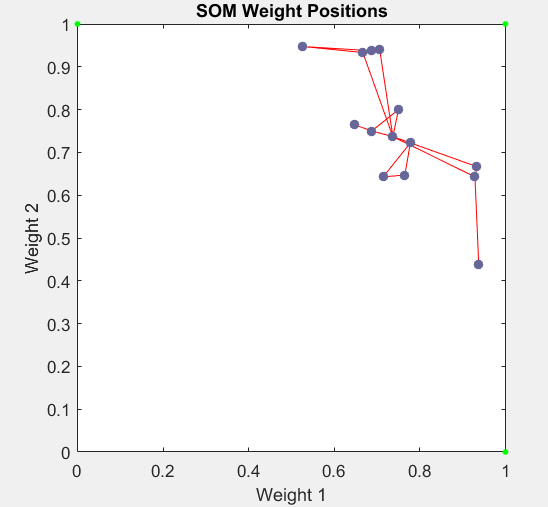
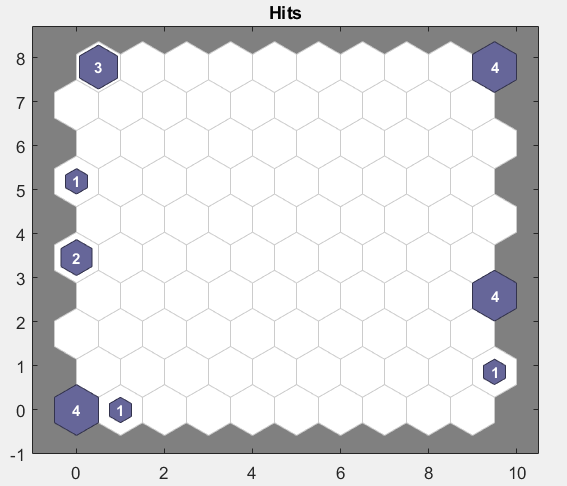
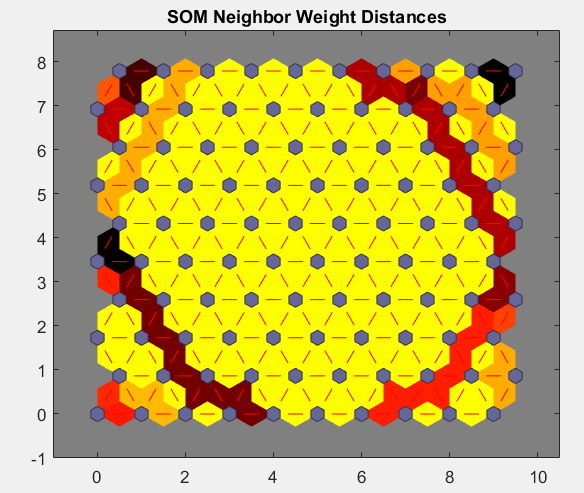
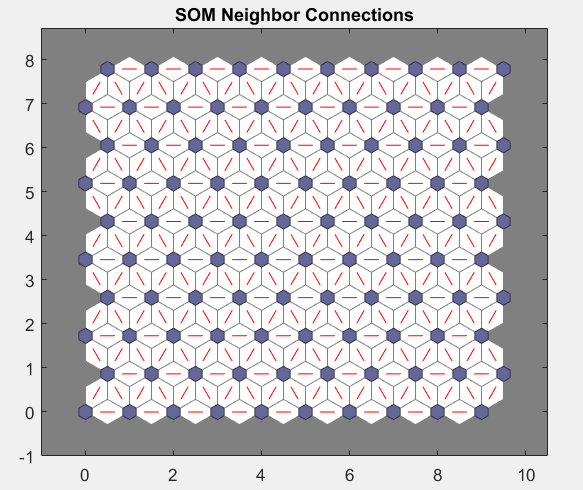
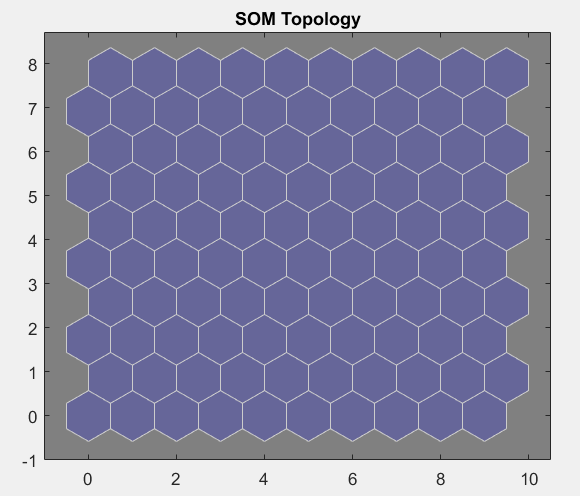
c) Uczenie sieci dla różnych współczynników uczenia.

Sieć uczyłam dla współczynnika uczenia równego 0.5. Zmieniałam tylko wejściowy rozmiar sąsiedztwa, aby sprawdzić czy ma to wpływ na działanie programu. Wyniki umieściłam w podpunkcie d).

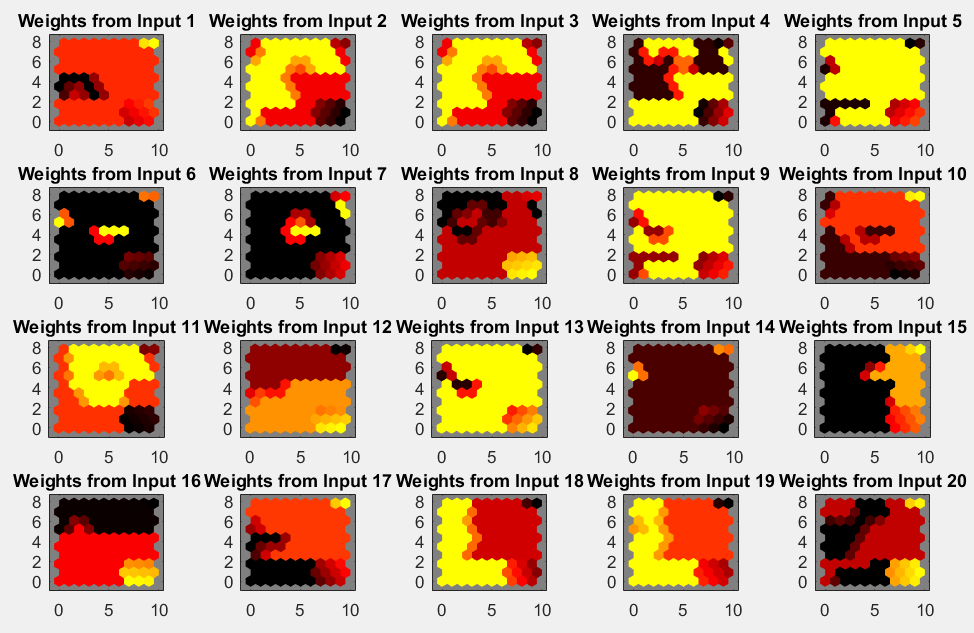
d) Testowanie sieci.

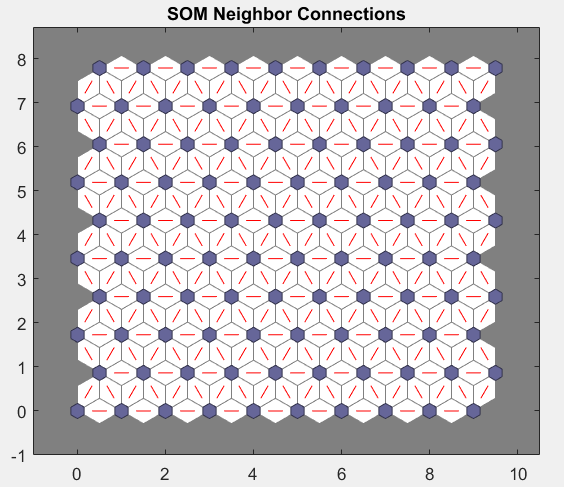
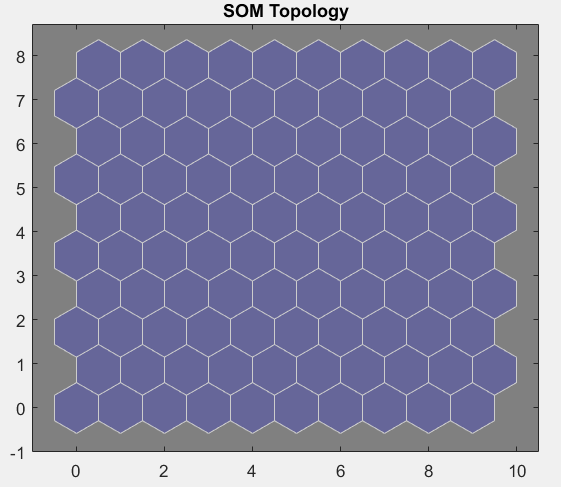
**ROZMIAR SĄSIEDZTWA = 0**

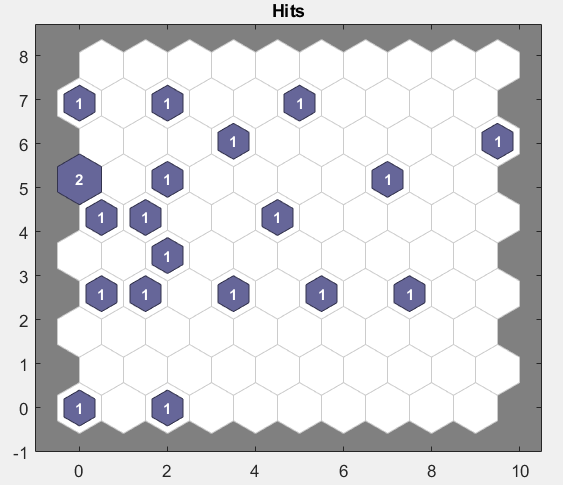
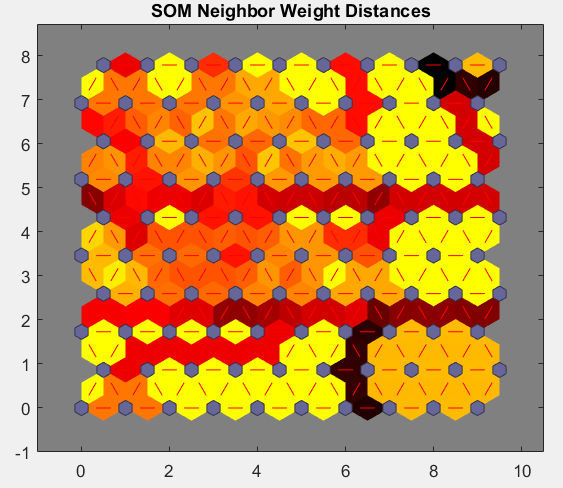
****

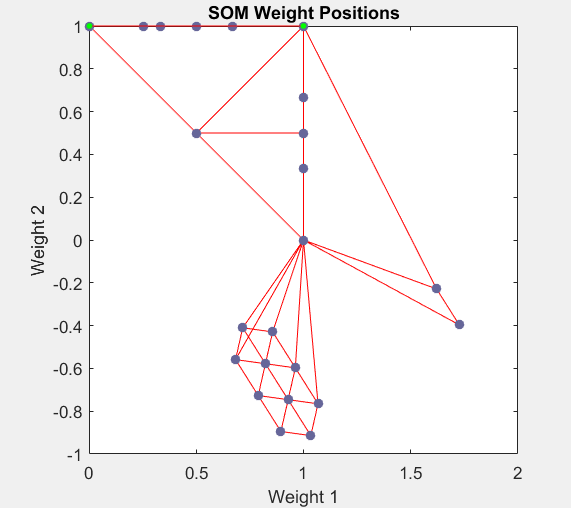
****

**ROZMIAR SĄSIEDZTWA = 1**

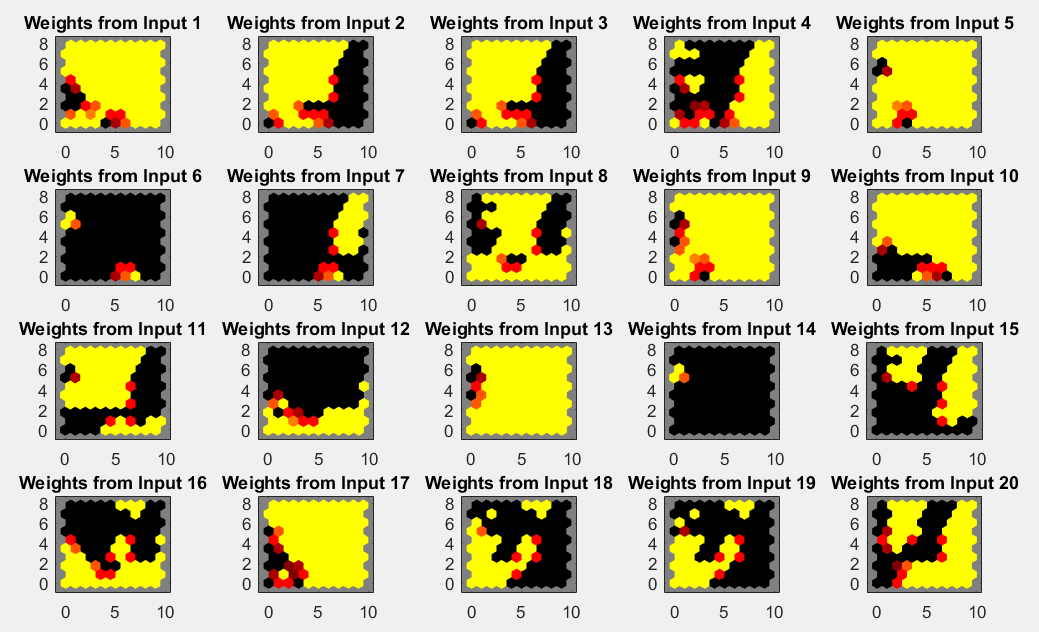
****

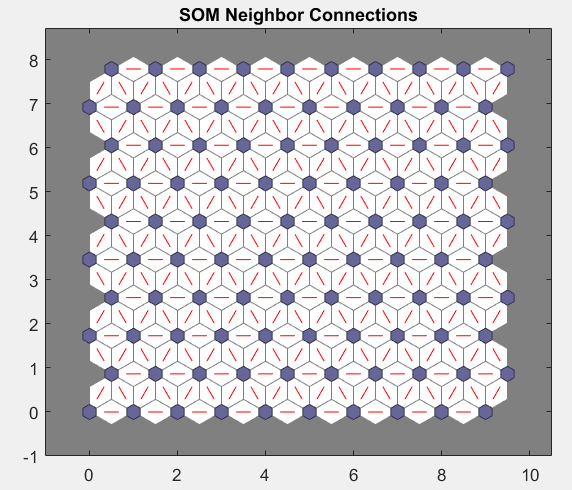
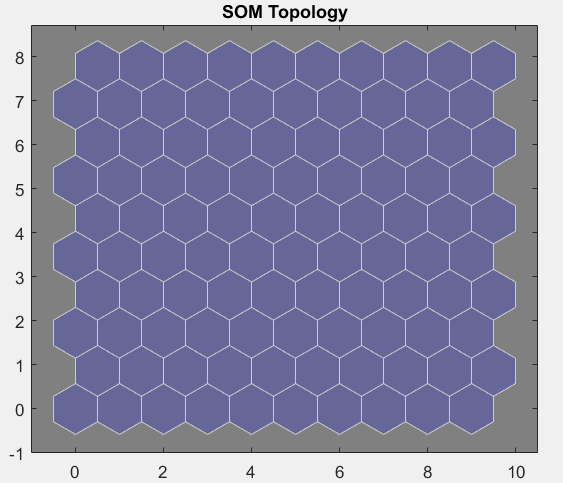
****

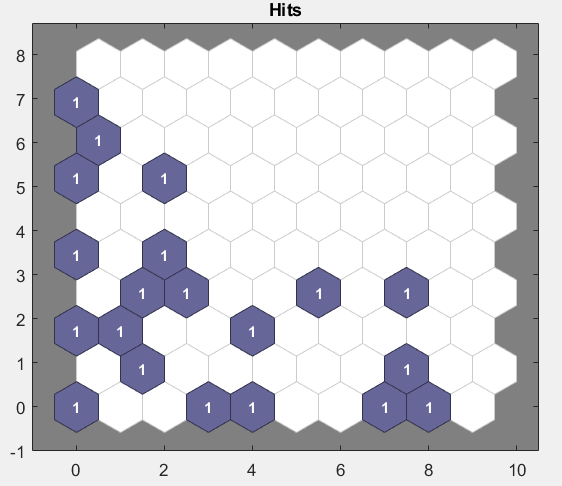
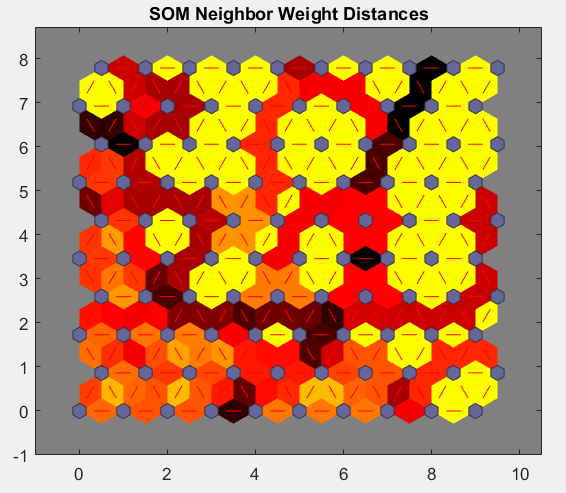
****

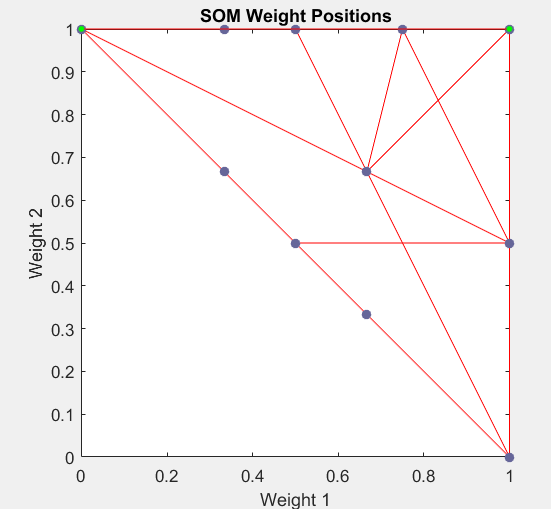
****

**ROZMIAR SĄSIEDZTWA = 3**

****

****

****

****

SOM Topology – topologia sieci Kohonena. Każdy z sześciokątów reprezentuje neuron. Sąsiedztwo dwóch figur może wskazywać na ich podobieństwo.

SOM Neighbor Connections - reprezentują połączenia pomiędzy poszczególnymi neuronami.

SOM Neighbor Distance – reprezentują jak silne są powiązania między poszczególnymi neuronami. Większe wagi są oznaczone ciemniejszym kolorem.

SOM Weights – wskazuje na rozkład wag poszczególnych neuronów w sieci

SOM Hits – pokazuje ile trafień znalazło się w naszym uruchomieniu programu. Czyli ile podobnych wyników zostało uzyskanych.

SOM Weight Position – zielone kropki to dane wejściowe, połączone ze sobą liniami wykazują korelację.

**WNIOSKI:**

**-** Im większy jest rozmiar sąsiedztwa, tym większa jest liczba poprawnych trafień.

- Algorytm WTM jest o wiele bardziej wydatniejszy niż algorytm WTA, co powoduje zmniejszenie występowania błędnych lub mało znaczących neuronów w sieci.

- Zwiększenie ilości epok znacząco poprawia jakość uczenia sieci, jednakże spowalnia w znacznym stopniu czas pracy programu.

- Im większa jest liczba neuronów występujących w sieci, tym wyniki są poprawniejsze.

- Algorytm WTM sprawdza się o wiele bardziej w przypadku bardzo rozbudowanych sieci, algorytm WTA nie jest aż tak dobry jak WTM.