

Przemysław Bartkowski gr.1

Podstawy Sztucznej Inteligencji

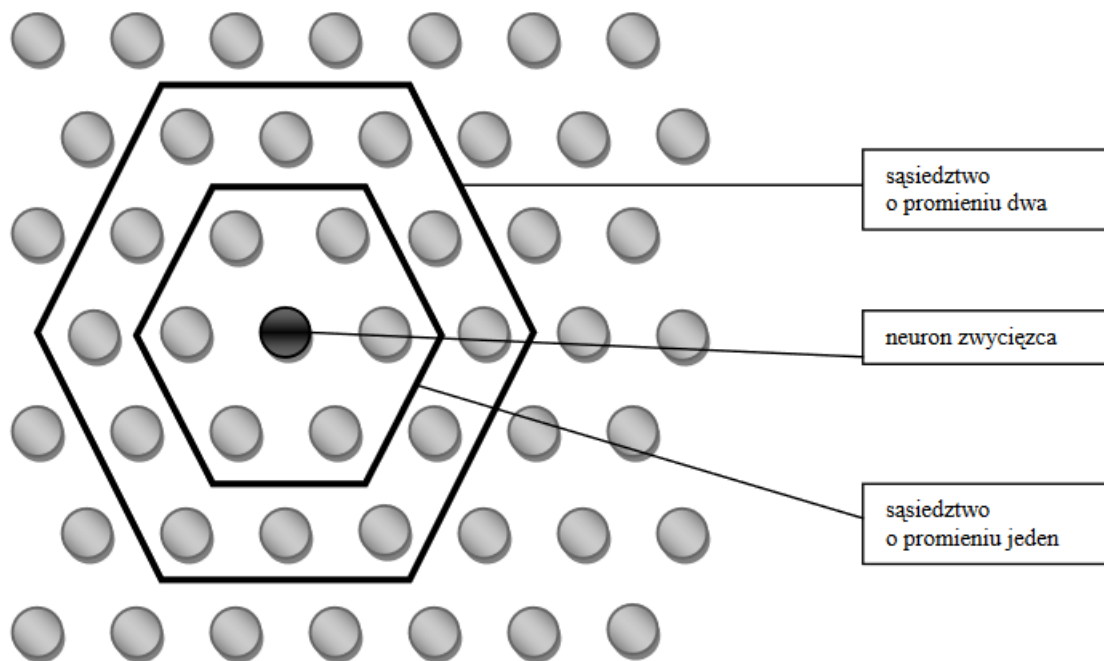
Sprawozdanie z Projektu nr 5 : Budowa i działanie sieci Kohonena dla WTM

Cel ćwiczeń:

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

Zagadnienia teoretyczne:

Ponieważ schemat budowy sieci Kohonena został poruszony w poprzednim sprawozdaniu, dlatego w części teoretycznej poruszę zagadnienie WTM. Ważnym pojęciem w tej dziedzinie jest sąsiedztwo – jeśli x_1 i x_2 są dwoma wektorami wejściowymi, a y_1 i y_2 są wyjściami odpowiadającymi zwycięskiemu węzłom wyjściowym, wówczas y_1 i y_2 powinny być położone blisko siebie, jeżeli x_1 i x_2 są do siebie podobne.

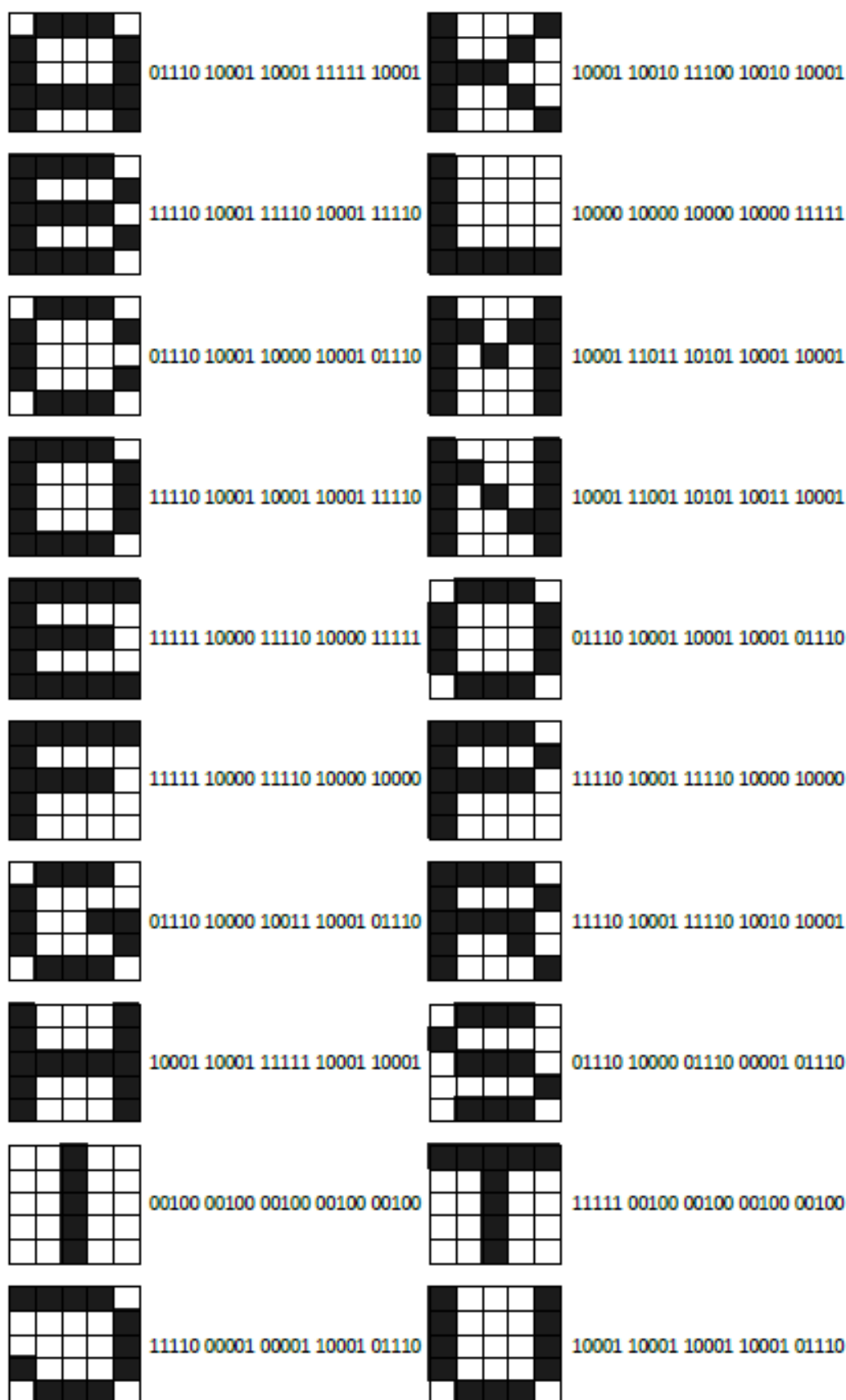


Rys. 3. Sąsiedztwo w sieci SOM o topologii heksagonalnej

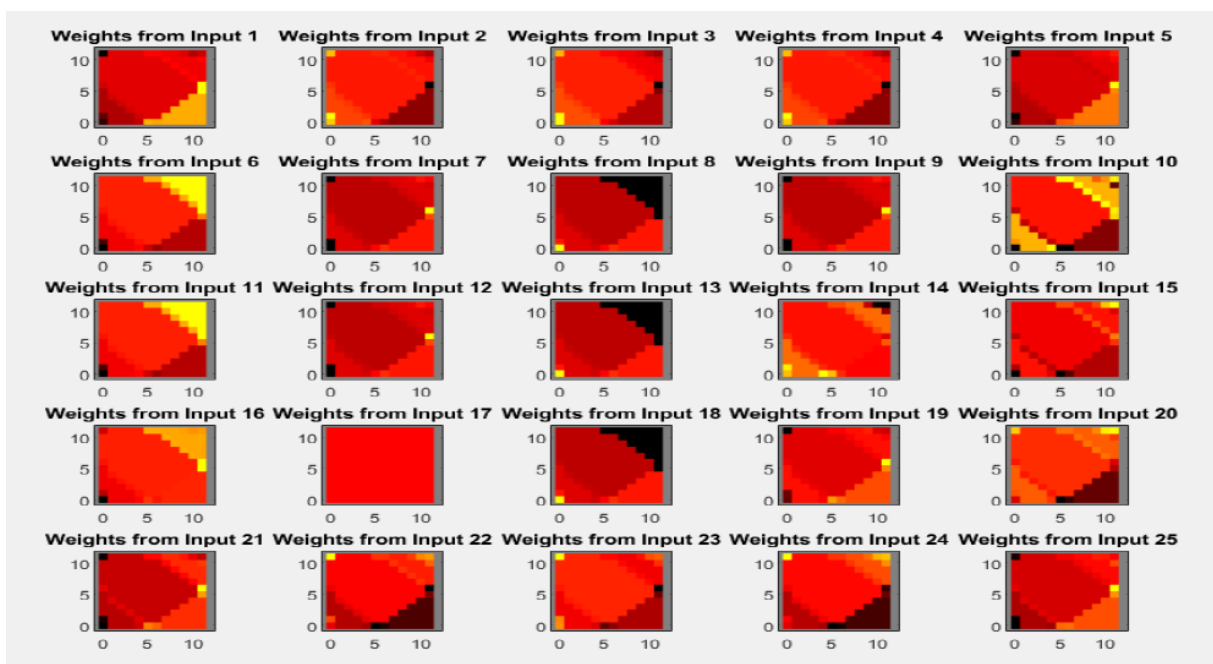
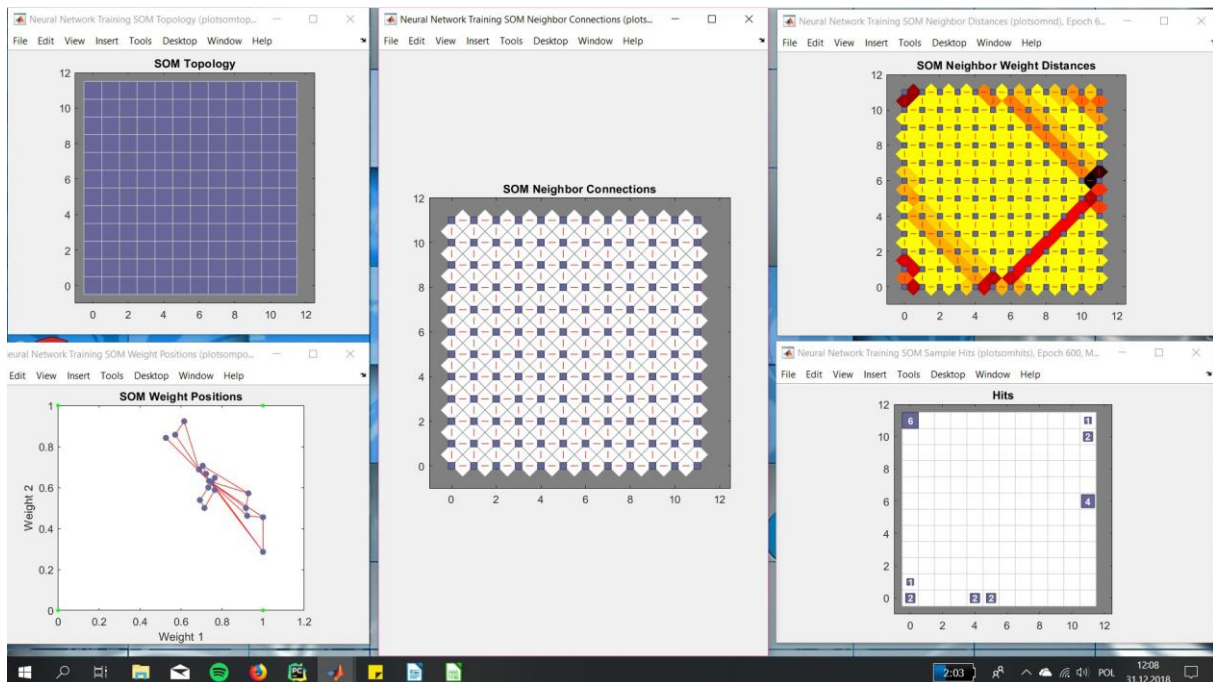
Zasada WTM (*Winner Takes Most*) jest to zasada, gdzie neuron zwycięski oraz neurony sąsiadujące z neuronem zwycięskim, czyli należące do sąsiedztwa $N_c(k)$, aktualizują swoje wagi według zasady: $w_i(k+1) = w_i(k) + \eta(k)G(i, c)[x(k) - w_i(k)]$, gdzie $i \in N_c(k)$, G – wpływ sąsiedztwa.

Zagadnienia praktyczne:

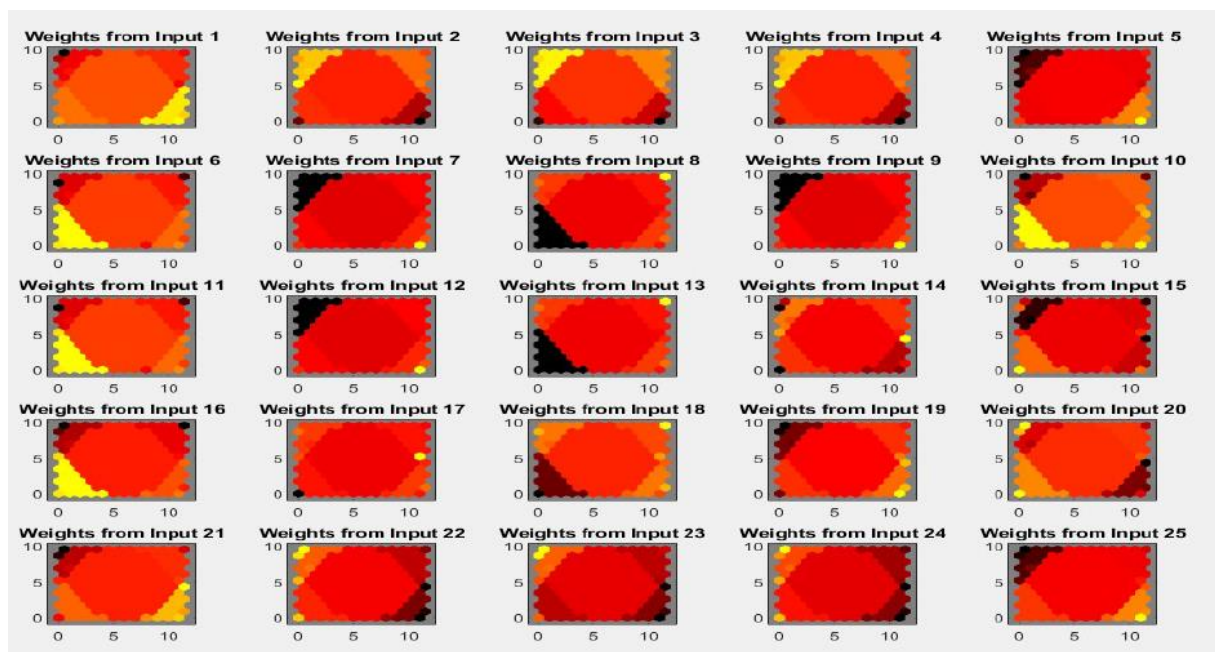
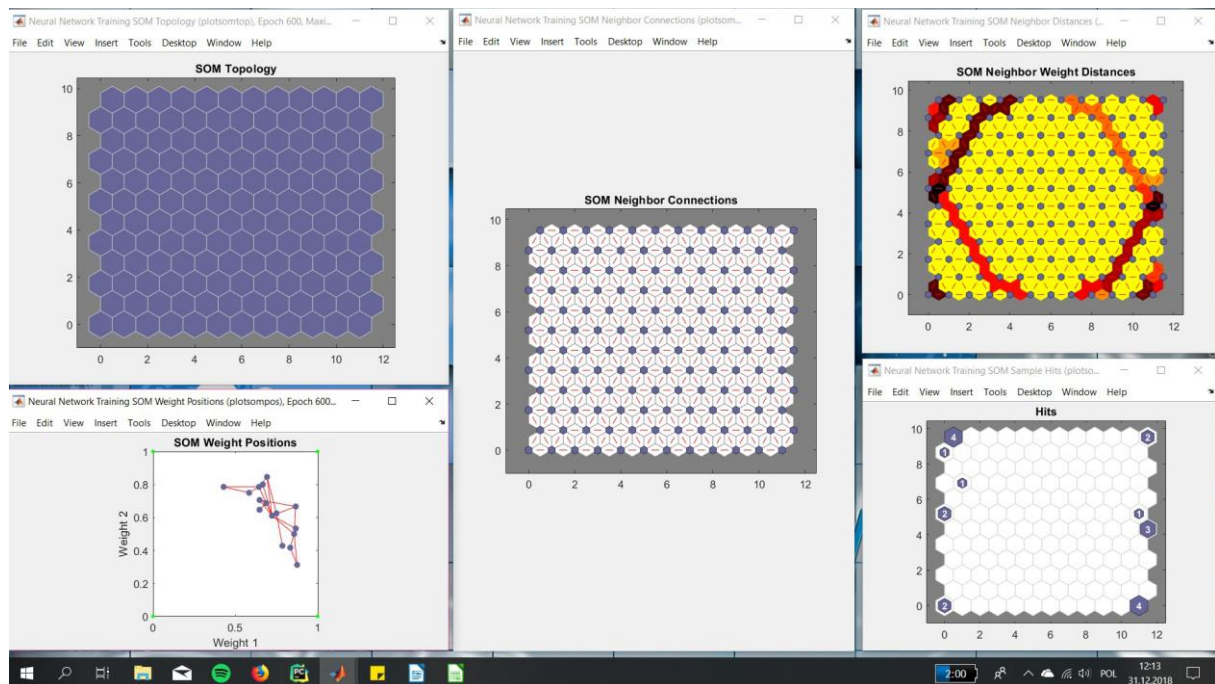
Do wykonania tego ćwiczenia wykorzystaliśmy pakiet **MATLAB** oraz narzędzie **Neural Network Training Tool**. Następnie utworzyłem dane uczące, którymi były 20 pierwszych, wielkich liter alfabetu łańskiego. Każda tablica miała wymiar 5x5 pikseli, gdzie kolor biały oznaczał wartość 0, a kolor czarny oznaczał wartość 1.



Aby lepiej porównać cechy między WTA a WTM posłużę się wektorem wyjściowym wektorów o wymiarach 12x12.

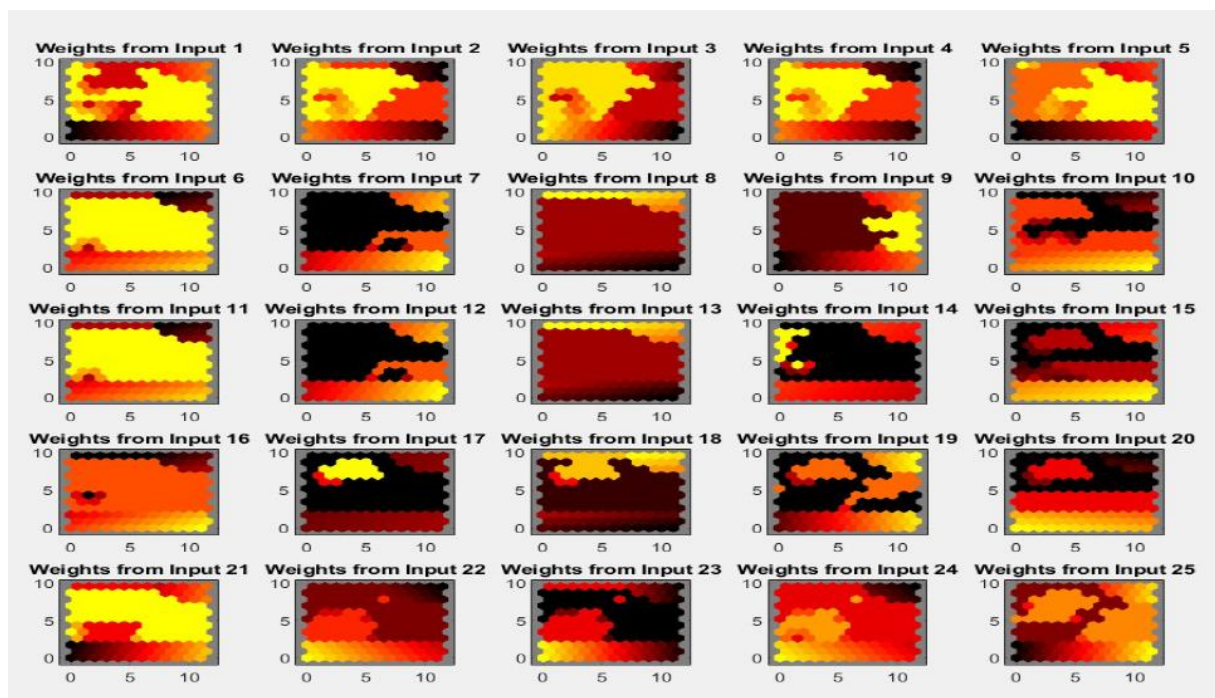
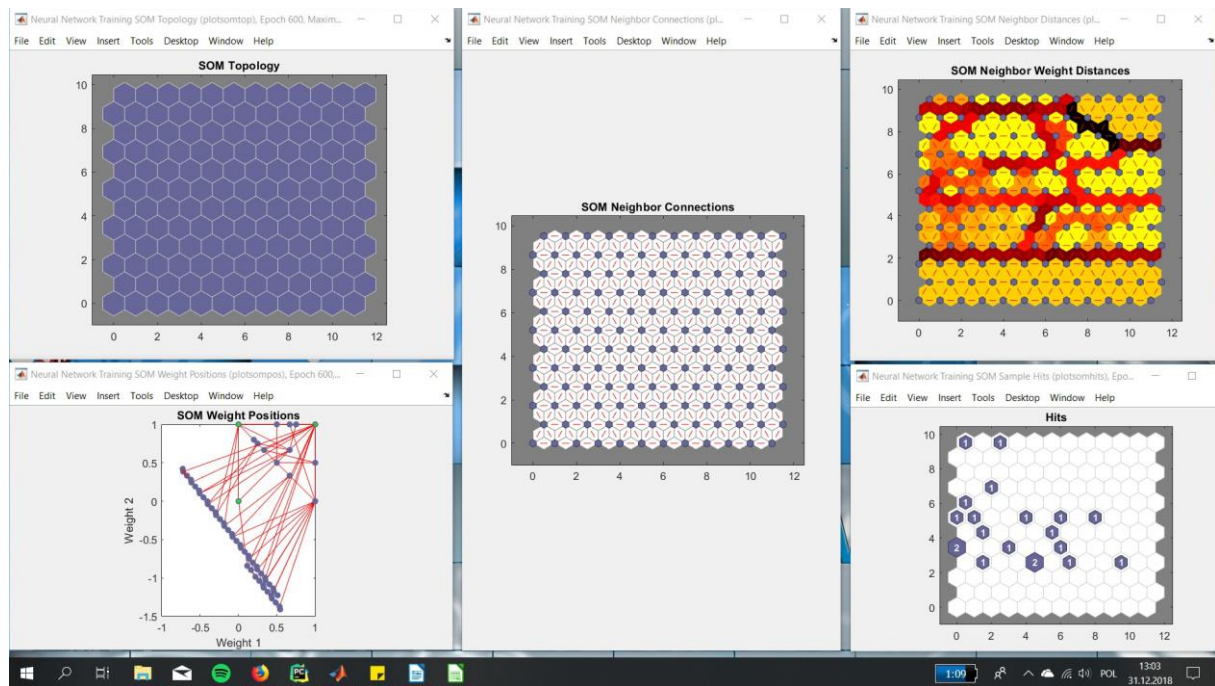


Wykresy dla topologii kwadratowej

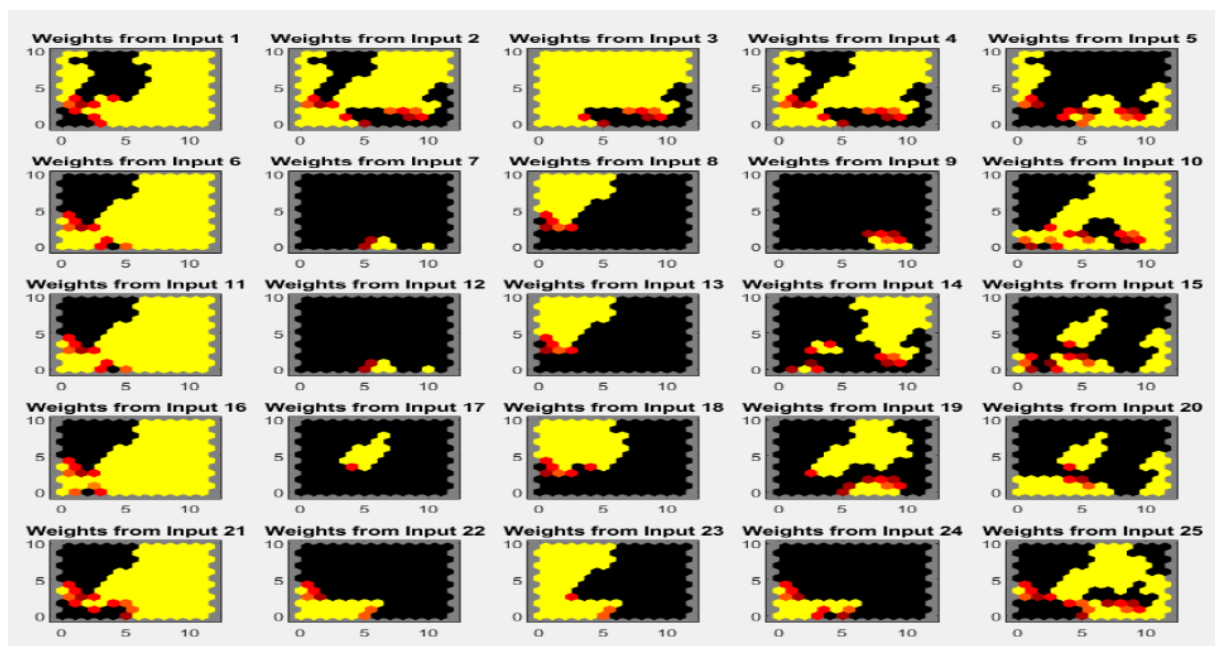
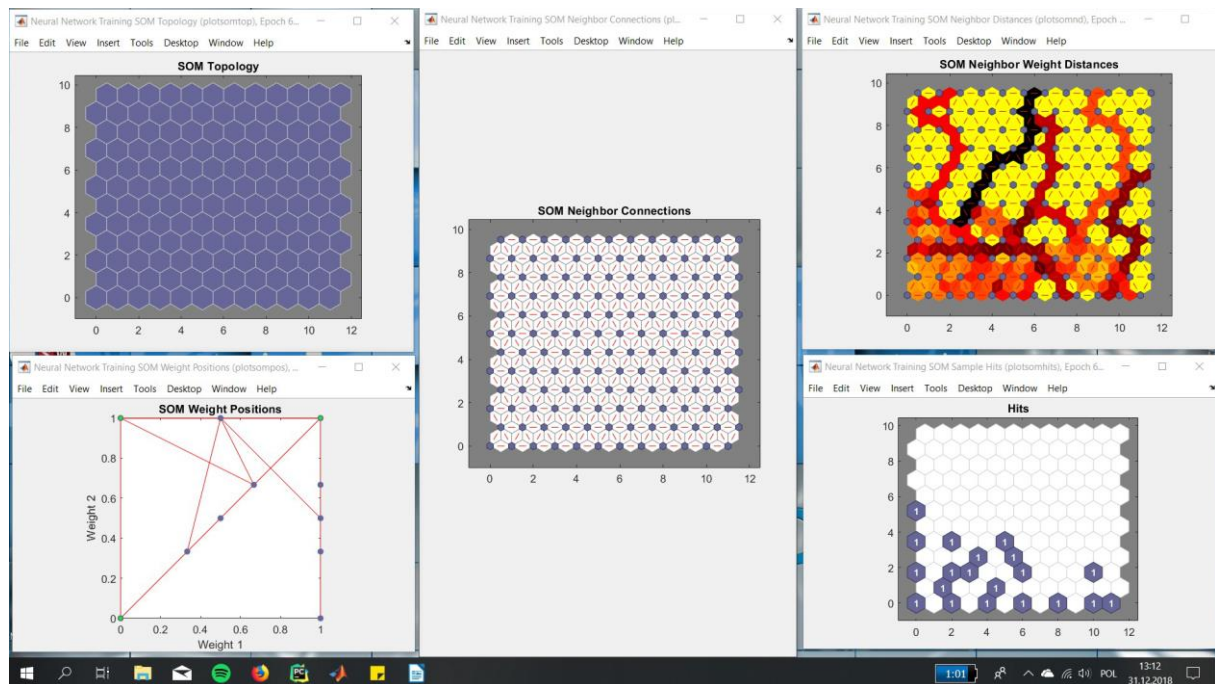


Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,5, sąsiedztwo = 0)

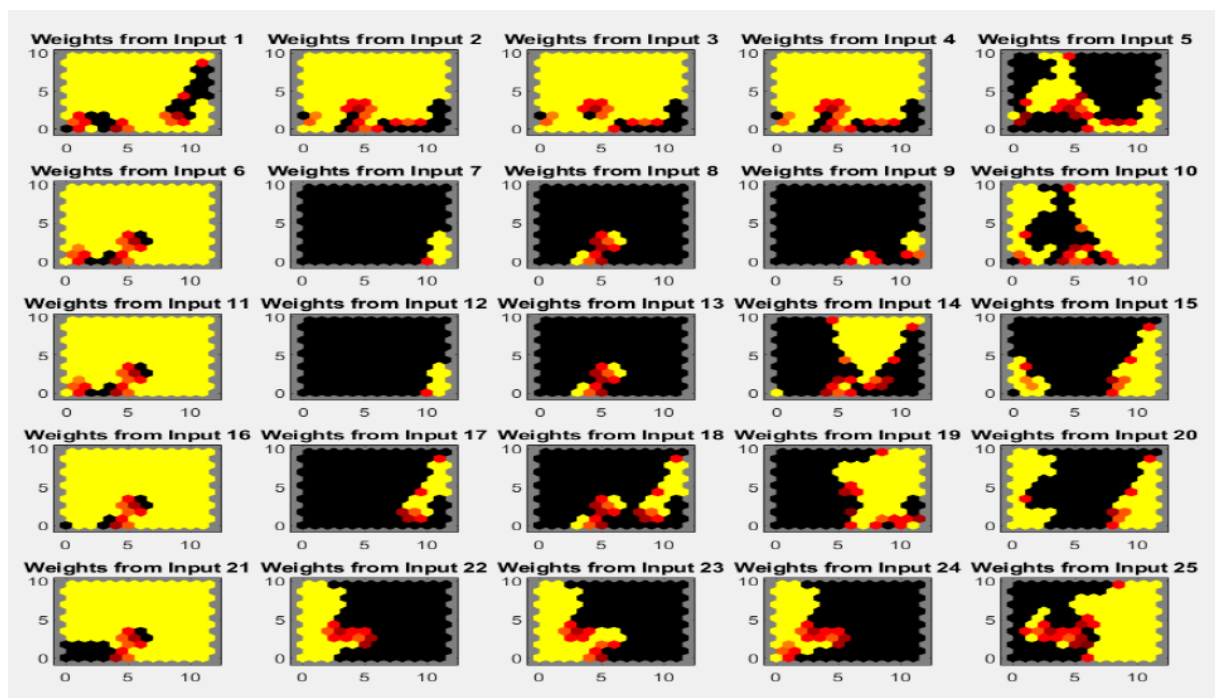
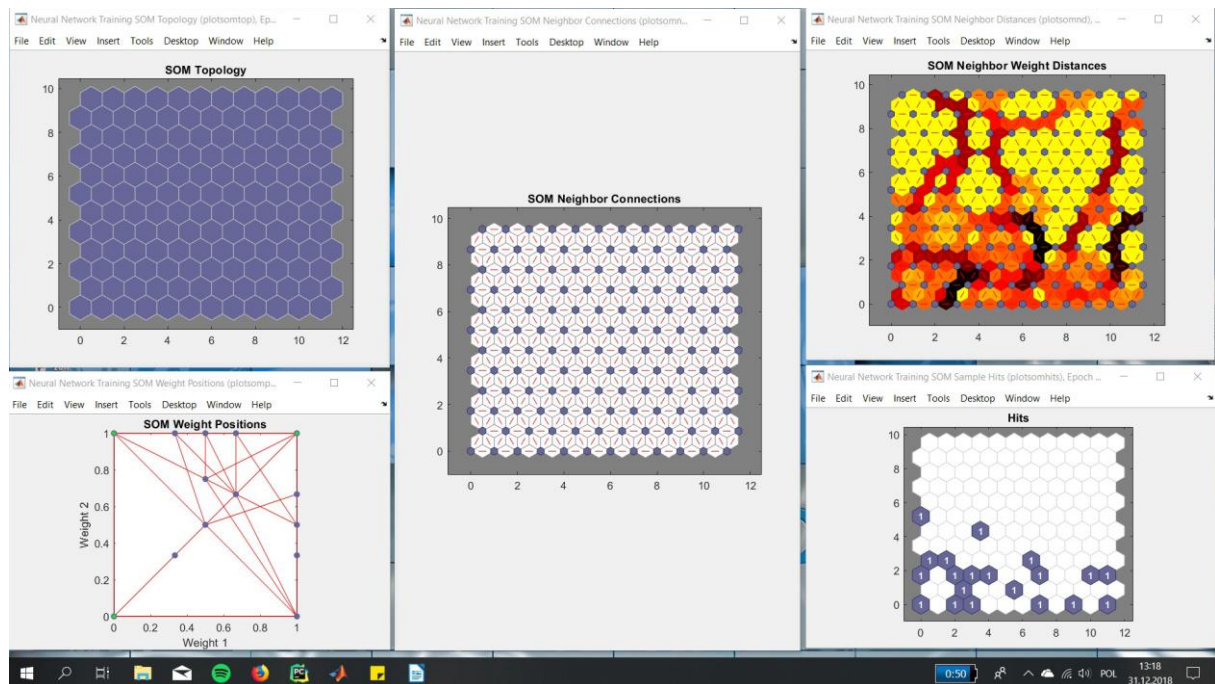
Pewne tendencje są bardziej uwidocznione w topologii heksagonalnej, dlatego pozostałe analizy i testy przeprowadziłem dla tej właśnie takiej topologii. Jednym z parametrów istotnych dla algorytmu WTM jest sąsiedztwo. Dlatego sprawdziłem działanie algorytmu dla 4 różnych początkowych sąsiedztw (0, 1, 3, 5) dla współczynnika uczenia równego 0,5.



Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,5, sąsiedztwo = 1)

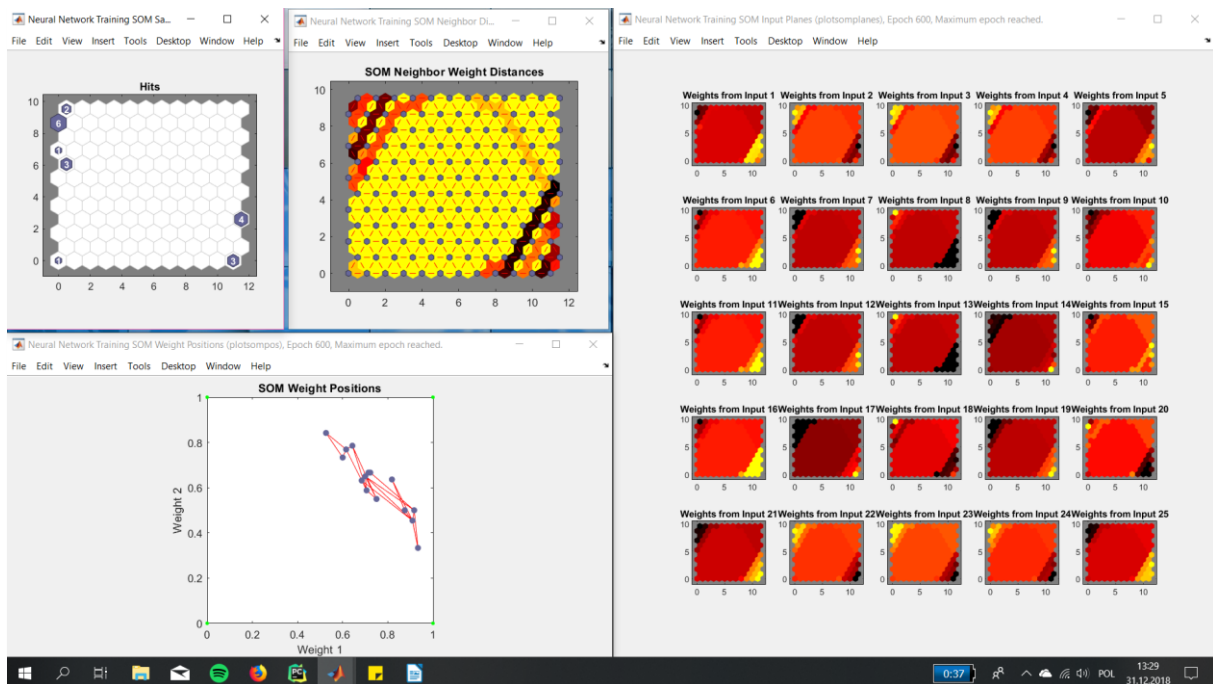


Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,5, sąsiedztwo = 3)

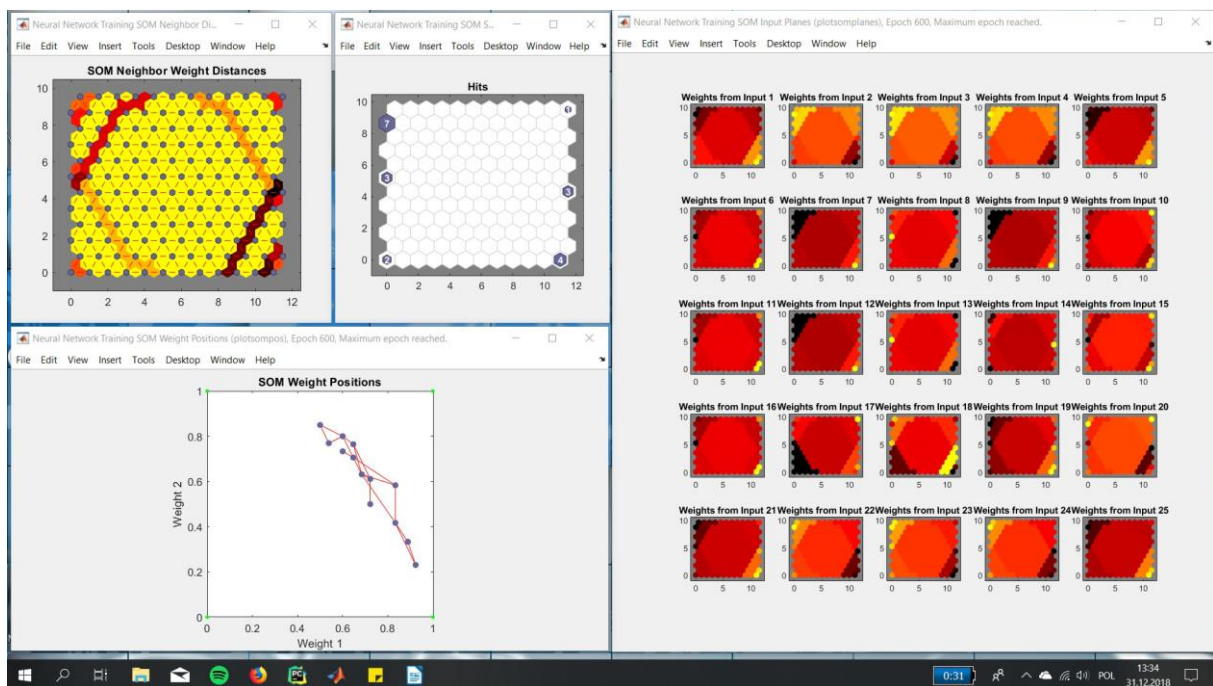


Wykresy dla topologii heksagonalnej (wsp. uczenia = 0,5, sąsiedztwo = 5)

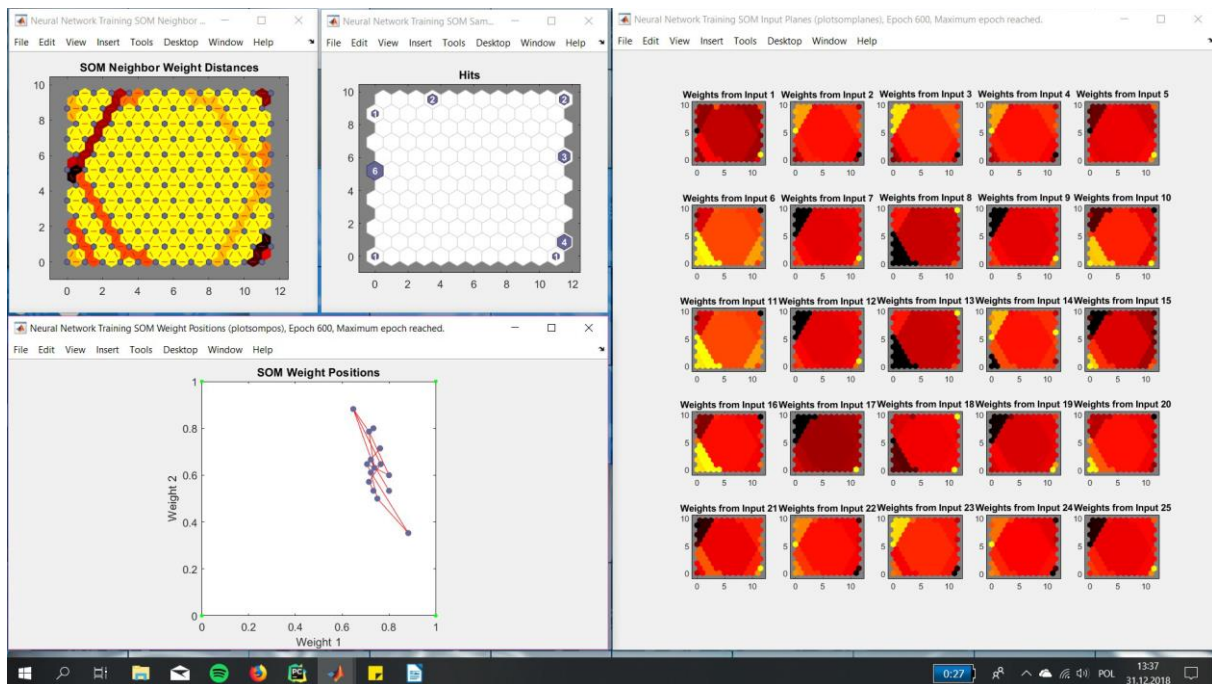
Innym istotnym czynnikiem jest współczynnik uczenia. Dlatego dla sąsiedztwa równego 0 sprawdziłem cztery współczynniki uczenia (0.1, 0.5, 0.75, 0.9).



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



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



Wykres dla topologii heksagonalnej (wsp. uczenia = 0.9, sąsiedztwo = 0)

Wnioski:

- W zależności od współczynnika uczenia pewne podziały na klastry i tendencje stają się silniejsze.
- 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 neuronów martwych (niepowiązanych, niepodobnych do siebie), co zdecydowanie odróżnia system WTM od WTA.
- W metodzie WTM sąsiedztwo gra niebagatelną rolę i determinuje ono kształt korelacji i zależności – im jest ono wyższe tym podziałów jest mniej, ale bardziej porozrzucone po całej siatce i poszczególne neurony znajdują się w innych kategoriach.
- Rozkłady sił 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.
- Metodologia ta powinna być stosowana wszędzie tam, gdzie istnieje potrzeba znalezienia jakichkolwiek powiązań z daną cechą.

Listing kodu:

```
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 1 1 0 0 0 0 0 0;
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 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 1 1 0 0 0 0 0 0;
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 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 0 1 0;
1 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 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 neuronów - miara
euklidesowa (znormalizowana)
% domyślnym parametrem jest odległość między neuronami
warstwy z % uwzględnieniem 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
```