Jarosław Dutka

Inżynieria Obliczeniowa, rok III

WIMIiIP, AGH

**Sprawozdanie nr 6**

„Budowa i działanie sieci Kohonena dla WTM”

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

1. **Przebieg ćwiczenia**

Do realizacji ćwiczenia użyto środowiska **MATLAB** z wtyczką **Neural** **Network** **Toolbox**. Wybrano to środowisko ze względu na prostotę implementacji sieci Kohonena..

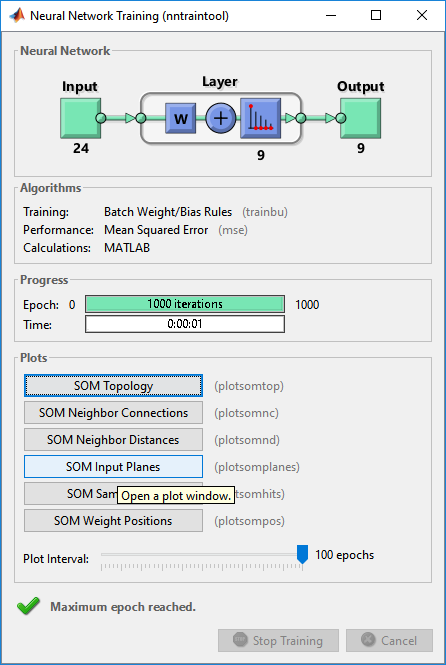
Zestaw danych uczących pobrano został stworzony do realizacji poprzednich scenariuszy. Zawierał on 10 małych i 10 dużych liter alfabetu łacińskiego. Litery zostały zapisane w postaci tablic dwuwymiarowych, o rozmiarze 6 x 4 znaków w sposób, jaki pokazuje grafika obok.

Dane te zostały zapisane w tablicy o nazwie **WE**(24x20).

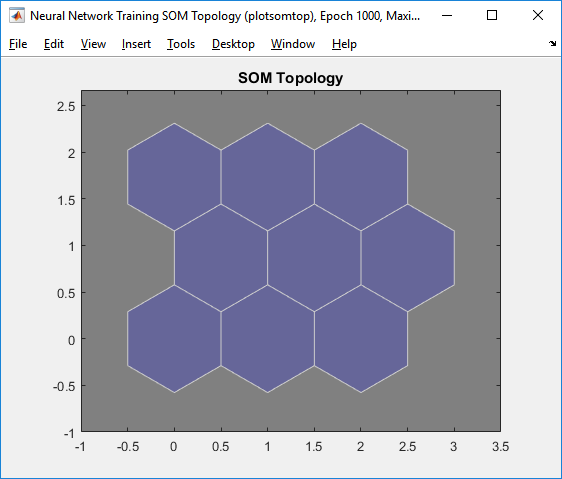
W przeciwieństwie do algorytmu WTA, algorytm WTM użyty w tym ćwiczeniu modyfikuje nie tylko neuron który wygrywa, ale również jego otoczenie.

Do stworzenia mapy neuronów użyto funkcji selforgmap(), która jest częścią biblioteki **Neural Network Toolbox**. Funkcja ta przyjmuje następujące argumenty:

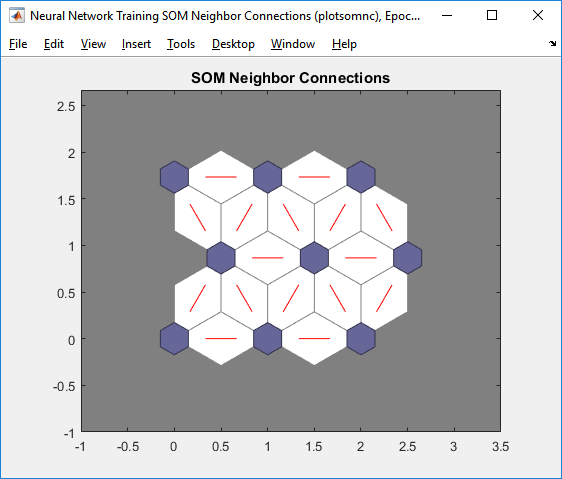
* Dimensions – wymiar sieci (np. 6x6);
* InitNeighbor – ilość neuronów tworzących sąsiedztwo;
* topologyFcn – funkcja topologii:
* Gridtop – topologia przy użyciu prostokątów;
* Hextop – przy użyciu sześciokątów;
* Randtop – losowo rozrzucone neurony;
* distanceFcn – funkcja obliczająca dystans pomiędzy neuronami:
* dist - odległość Euklidesowa.
* linkdist – kilka sposobów liczenia, w tym Euklidesowa, zależnych od ilości wprowadzanych wektorów.
* W ćwiczeniu sieć zainicjalizowano poprzez wywołanie następującego kodu:

net = selforgmap(wymiary,coverSteps,initNeighbor,topologyFcn,distanceFcn);

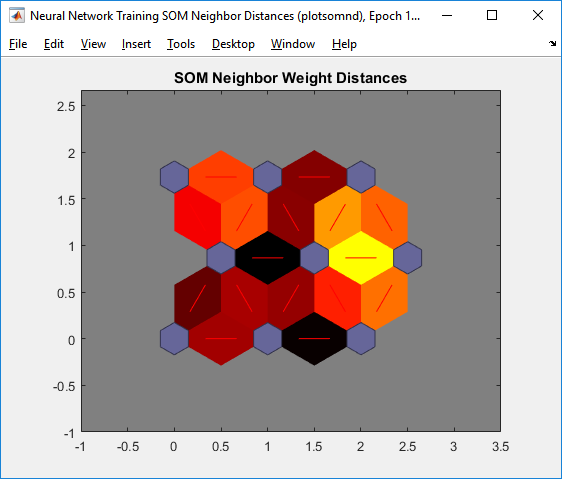
**Uczenie sieci**

Poniższe wykresy są wynikiem działania mapy o wymiarach 3x3 neurony, dla 24 wejść po 1000 iteracjach.

SOM Topology – przedstawia poglądowy wygląd topologii mapy użytej w ćwiczeniu.

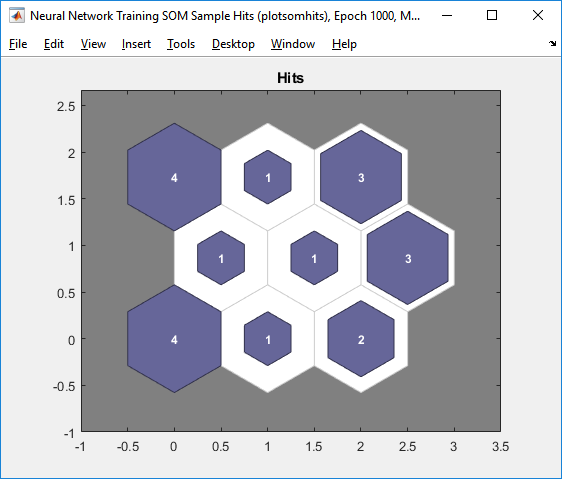


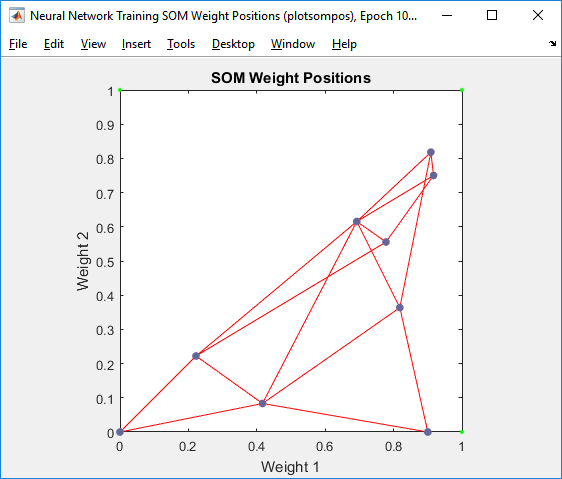
SOM Neighbor Connections ilustruje w jaki sposób poszczególne neutrony w sieci są połączone. Czerwone linie stanowią połączenia (o odpowiednich wagach) a niebieskie sześciokąty neurony.



Wykres SOM Neighbor Weight Distances pokazuje odległości poszczególnych neuronów w danej sieci od siebie. Podobnie jak na powyżyszym wykresie, niebieskie sześciokąty to neurony a czerwone linie to połączenia. Linie są narysowane na kolorowych sześciokątach, które w zależności od barwy mają ilustrować dystanse dzielące poszczególne komórki. Czarne sześciokąty stanowią największe odległości. Im kolor jaśniejszy tym odległości są krótsze.

 SOM Input Planes ma na celu pokazanie odległości od siebie neuronów w mapie dla każdego wejścia. Kolorystyka jest analogiczna do tej, która została użyte w wykresie SOM Neighbor Weight Distances.

Z wykrsu SOM Samples Hits można się dowiedzieć, ile razy dany neuron zwyciężył. Każdy sześciokąt na rysunku ilustruje jeden neuron a liczba w nim zapisana informuje o ilości zwycięstw danej komórki.

SOM Weight Positions pokazuje jak sieć klasyfikuje przestrzeń wejściową. Niebieskie kropki to neurony a czerwone linie to połączenia między nimi.

**Wnioski**

1. W porównaniu do WTA, algorytm WTM powoduje równomierny rozkład sygnałów wejściowych. Dzięki modyfikacji nie tylko neuronu, ale także jego otoczenia, nie zachodzi zjawisko dominacji jednego neuronu.
2. Współczynnik uczenia jest samoistnie zmieniany w trakcie procesu nauki.
3. Sieć działa poprawnie dla topologii sześcianów jak również kwadratów.
4. Aby sieć działała poprawnie, należy się upewnić, że mapa jest wystarczająco duża. Dla każdego problemu istnieje minimalna liczba neuronów, która jest niezbędna do jego rozwiązania. Po osiągnięciu tej minimalnej liczby neuronów, dalsze zwiększanie ich ilości, w żaden sposób nie wpływa na zmianę wyniku pracy sieci.

**Listing kodu**

close all; clear all; clc;

%A a B b C c D d E e Z z R r K k H h L l

WE=[0 0 1 1 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 ;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

0 0 0 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 0 1 0 0 0 0 0 ;

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

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

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

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

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

];

% SOM parametry

wymiary = [3,3];

coverSteps = 100;

initNeighbor = 1;

topologyFcn = 'hextop';

distanceFcn = 'linkdist';

% Tworzenie SOM

net = selforgmap(wymiary,coverSteps,initNeighbor,topologyFcn,distanceFcn);

net.trainParam.epochs = 200;

net.trainFcn = 'trainbu';

% Trenowanie sieci

[net,tr] = train(net,WE);