

Sprawozdanie Scenariusz 5 Aleksandra Karaś

Budowa i działanie sieci Kohonena dla WTM

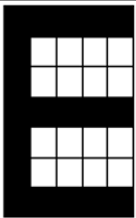
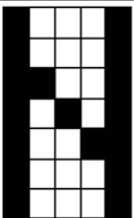
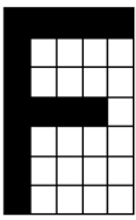
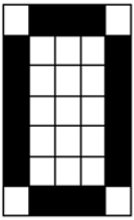
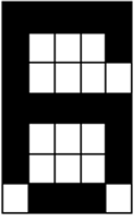
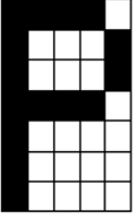
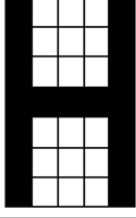
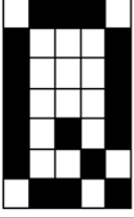
1. Cel ćwiczenia

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

2. Opis budowy sieci i algorytmów uczenia.

Celem zbudowanej sieci jest podział danych uczących na określone grupy i przyporządkowanie danej grupie danego elementu wyjściowego. Podział na grupy polega na tym, żeby elementy w danej grupie były jak najbardziej podobne do siebie jednocześnie będąc zupełnie inne od elementów z innych grup.

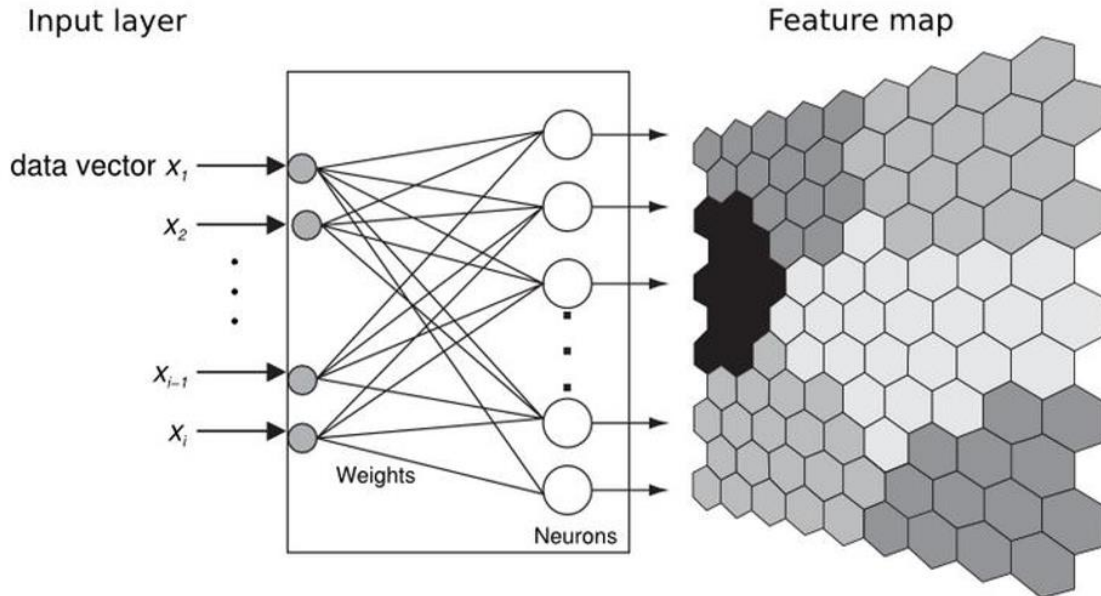
	0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1		1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0
	1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0		1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1
	0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0		1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1
	1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0		1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1

	1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1		1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 0 0 1
	1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0		0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 1 1 0
	1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 1 1 1 0		1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
	1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1		0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 0 0 1 1 0 1

Litery i ich reprezentacja

Syntetyczny opis sieci

Nauka sieci odbywa się za pomocą uczenia rywalizującego (metoda uczenia sieci samoorganizujących). Podczas procesu uczenia neurony są nauczane rozpoznawania danych i zbliżają się do obszarów zajmowanych przez te dane. Po wejściu każdego wektora uczącego wybierany jest tylko jeden neuron (neuron będący najbliższemu prezentowanemu wzorcowi). Wszystkie neurony rywalizują między sobą, gdzie zwycięża ten neuron, którego wartość jest największa. Zwycięski neuron przyjmuje na wyjściu wartość 1, pozostałe 0. Różnica pomiędzy WTM, a WTA polega na tym, że podczas procesu uczenia wykorzystywany jest promień, który pozwala na zaktualizowanie wag neuronów, które nie zwyciężyły. Jednakże z każdą iteracją (krokiem czasowym) promień ten zmniejsza się, żeby na samym końcu mógł zaktualizować swoje wartości tylko i wyłącznie zwycięski neuron.



Rys. 1 Sieć samoorganizująca

Proces uczenia przebiega według następującego schematu:

1. Normalizacja wszystkich danych
2. Wybór współczynnika uczenia η z przedziału $(0; 1)$
3. Wybór początkowych wartości wag z przedziału $(0; 1)$
4. Dla danego zbioru uczącego obliczamy odpowiedź sieci – dla każdego pojedynczego neuronu obliczana jest suma ilorazów sygnałów wejściowych oraz wag
5. Wybierany jest neuron, którego odległość euklidesowa. Tylko dla tego neuronu następuje aktualizacja wag. Wzór na zaktualizowanie wagi jest następujący:

$$w_{i,j}(t + 1) = w_{i,j}(t) + \eta * \theta(t) * (x_i * w_{i,j}(t))$$

gdzie: $\theta(t)$ – funkcja sąsiedztwa (wg Gaussa), obliczana ze wzoru:

$$\theta(t) = e^{\frac{-d^2}{2 * R^2}}$$

gdzie: d – jest to odległość pomiędzy zwycięskim neuronem oraz każdym dowolnym innym neuronem R – promień sąsiedztwa

$$d(i, w) = \sqrt{\sum_{i=1}^n (i_i - w_i)^2} \quad R(t) = R_0 * e^{-\frac{t}{\lambda}}$$

gdzie: i – wektor wejściowy w – waga neuronu t – obecna iteracja λ – stała czasowa

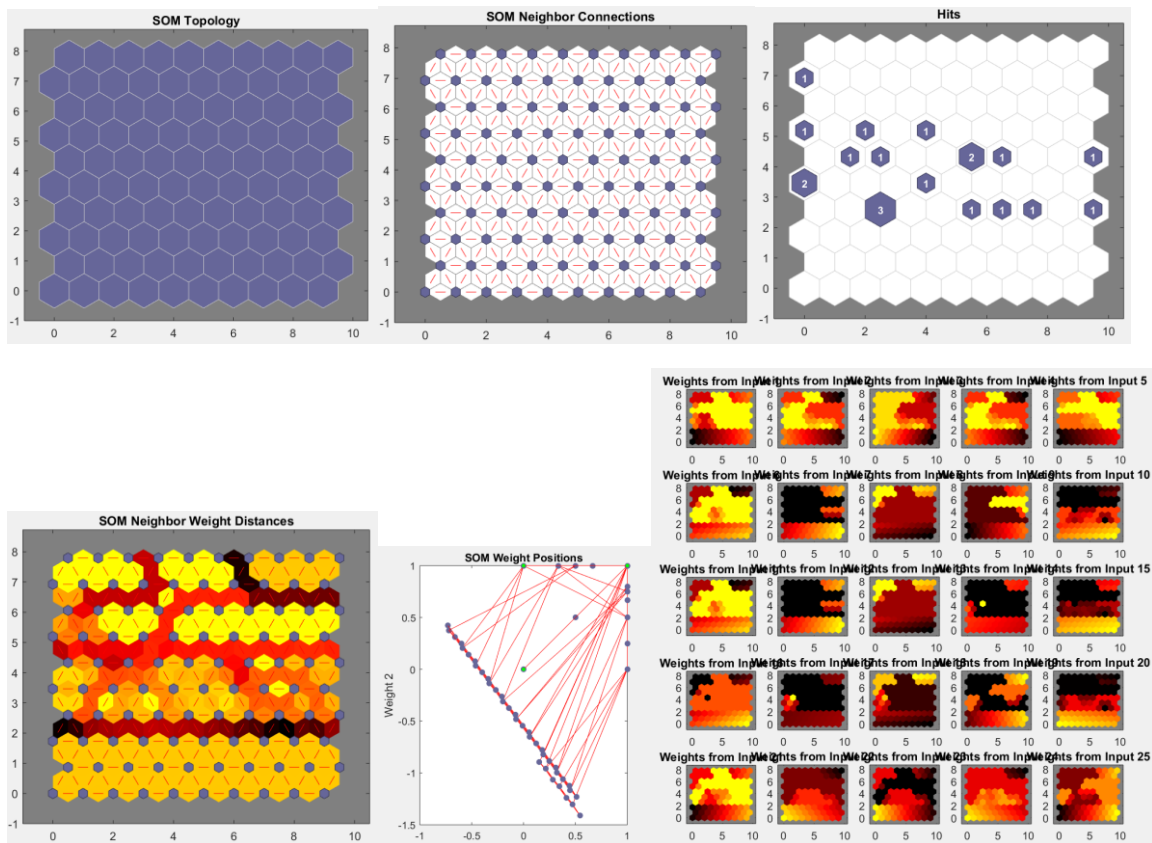
$$\lambda = \frac{x}{R_0}$$

gdzie: x – liczba iteracji R_0 – początkowy promień sąsiedztwa

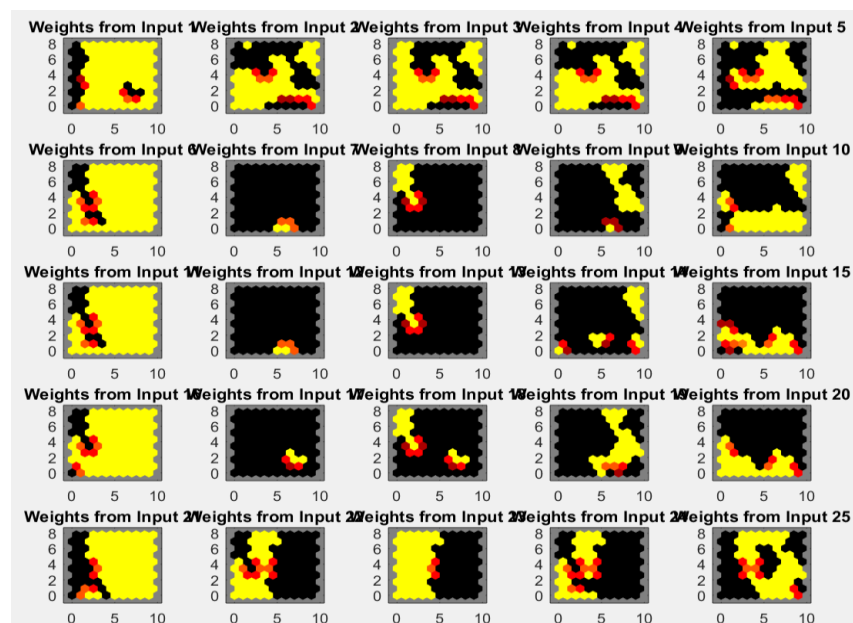
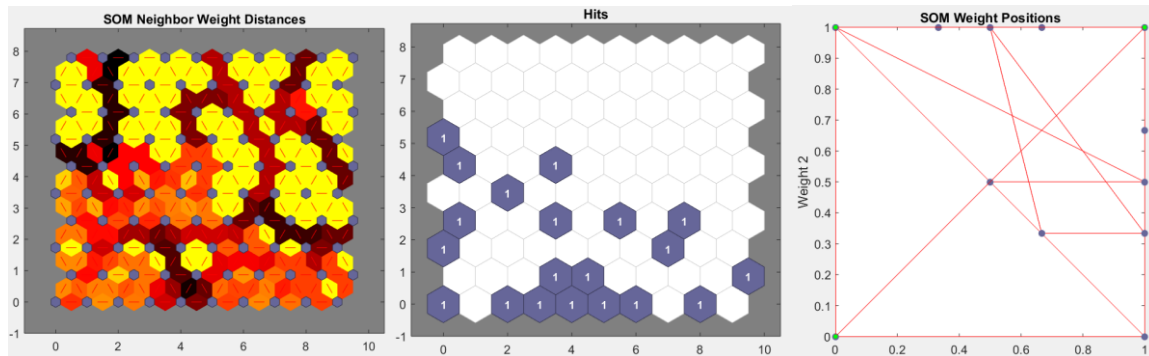
6. Znormalizowanie wartości nowego wektora wag 7. Zwycięski neuron daje odpowiedź na swoim wyjściu równą 1, a pozostałe 0. 8. Wczytanie kolejnego wektora uczącego.

3. Wyniki

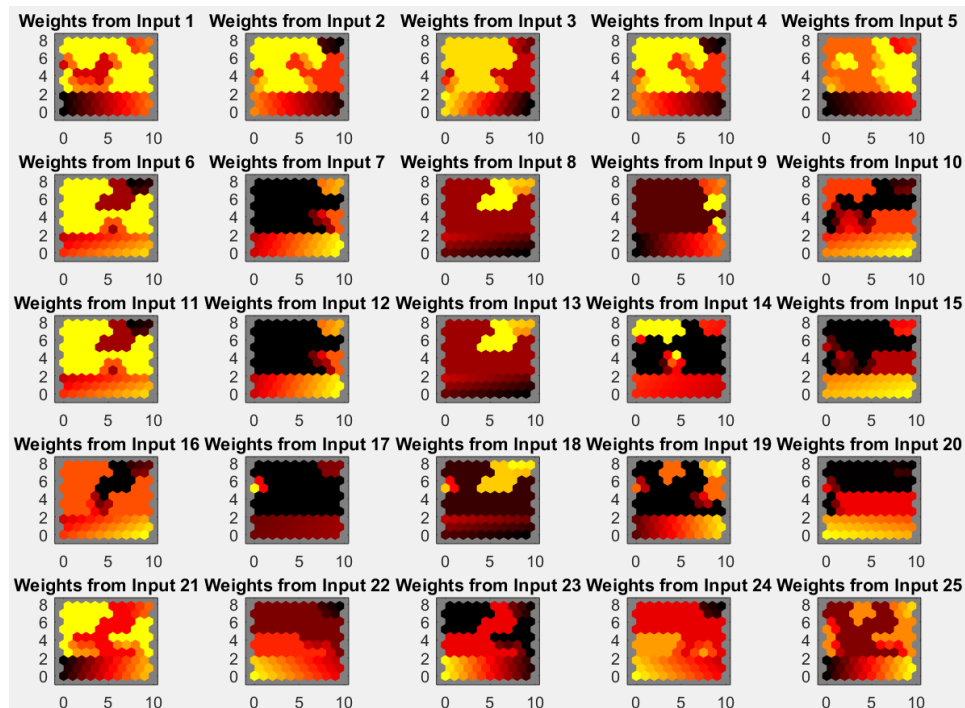
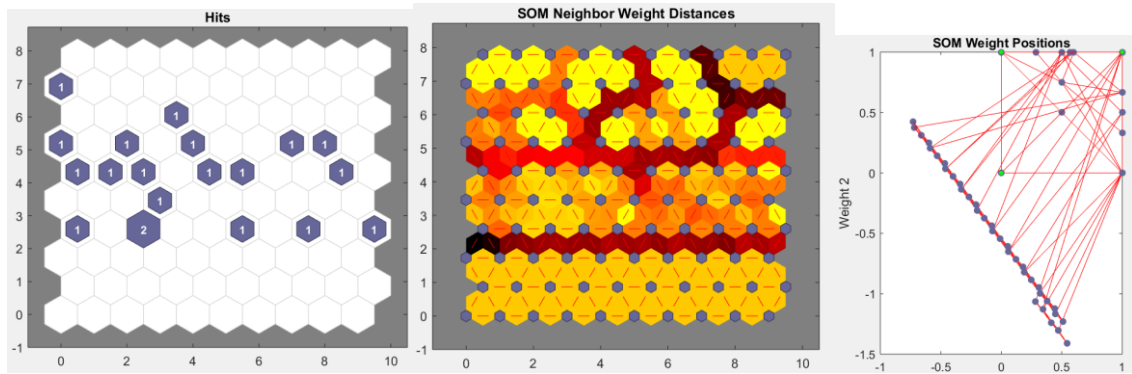
Wykresy dla topologii heksagonalnej o współczynniku uczenia 0.5 i sąsiedztwie 1



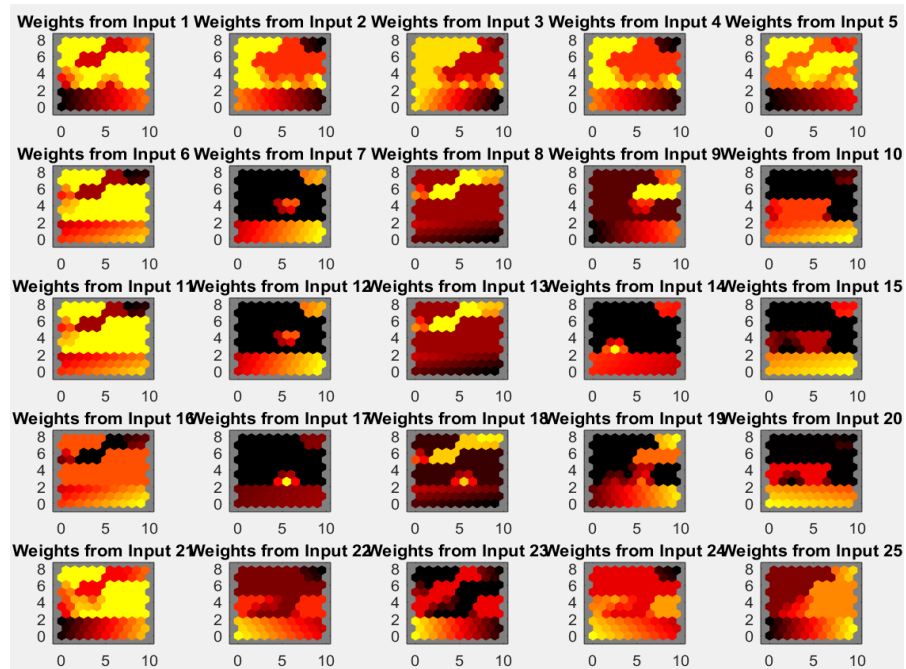
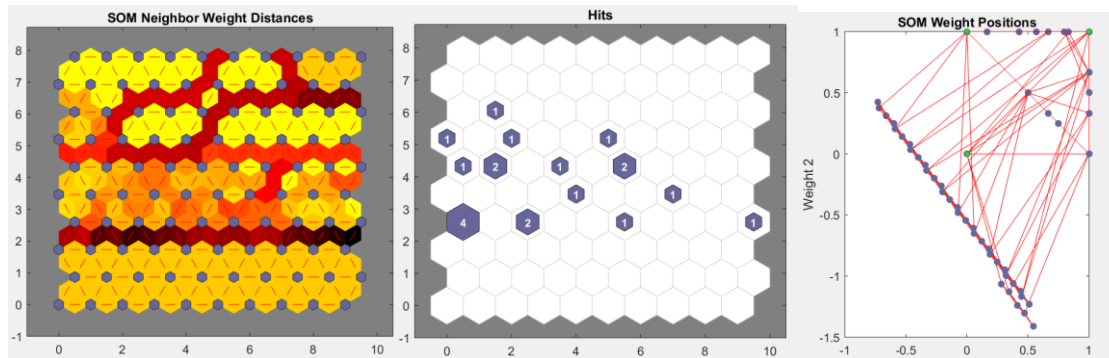
Wykresy dla topologii heksagonalnej o współczynniku uczenia 0.5 i sąsiedztwie 3



Wykresy dla topologii heksagonalnej o współczynniku uczenia 0.75 i sąsiedztwie 1



Wykresy dla topologii heksagonalnej o współczynniku uczenia 0.25 i sąsiedztwie 1



4.Wnioski:

Sieć Kohonena cechuje umiejętność podziału danych, które posiadają różne wartości dla poszczególnych cech, ponieważ jest to sieć samoorganizująca. Powoduje to, że odpowiedni podział na grupy może być wykonywane bez podawania wartości oczekiwanych (uczenie bez nauczyciela). Cały proces uczenia (jego efektywność) zależy od współczynnika uczenia. Im większy jest ten współczynnik, tym sieć uczy się szybciej. Jednakże wzrost efektywności nie jest wprost proporcjonalny do współczynnika uczenia. Dla wzrostu (efektywności) przy małych współczynnikach uczenia następuje większa różnica w ilości potrzebnych epok aniżeli dla wzrostu przy dużych współczynnikach uczenia – występuje stabilizacja uczenia. W odróżnieniu od metody WTA, metoda WTM pozwala na wiele sposobów implementacji. Można stosować w niej różne metryki (np. euklidesową lub miejską). Ponadto, wykorzystując metodę WTM otrzymamy lepsze rezultaty, ponieważ sieć jest bardziej uporządkowana (większa zbieżność algorytmu). Wadą WTM w porównaniu do WTA jest większy narzut – potrzeba więcej czasu i miejsca w pamięci, ponieważ aktualizowane są nie tylko zwycięskie neurony, ale także te będące w ich sąsiedztwie. Sieć dla odpowiednich współczynników uczenia pozwala uzyskać 100% poprawnych odpowiedzi, co powoduje, że sieć Kohonena sprawdza się przy dużej ilości różniących się pomiędzy sobą danych.

5.Kod

```
close all; clear all; clc;
```

```
data = [ 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 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 = [10 10];  
coverstep = 50;  
initNeighbor = 1;  
topologyFcn = 'hextop';  
distanceFcn = 'dist';  
  
net = selforgmap(dimensions, coverstep, initNeighbor, topologyFcn, distanceFcn);  
net.trainFcn = 'trainbu';  
net.trainParam.epochs = 600;  
net.trainParam.lr = 0.25;  
[net, tr] = train(net, data);  
y = net(data);  
indexOfOutput = vec2ind(y);
```