Fabiola Dąbroś , IO gr.1 rok III

Sprawozdanie z zagadnienia nr 6.

Temat ćwiczenia:
Budowa i działanie sieci Kohonena dla WTM.
Cel ćwiczenia:
Celem ćwiczenia jest poznanie budowy i działanie sieci Kohonena przy wykorzystaniu reguły WTM do odwzorowania istotnych cech liter alfabetu.

1. Opis budowy użytej sieci oraz wykorzystanego algorytmu uczenia.

Do wykonania tego ćwiczenia korzystałam z sieci Kohonena – sieci samoorganizujących się. Należą one go grupy sieci uczących się w sposób rywalizacyjny bez nauczyciela. Oznacza to, że nie ma żadnego zewnętrznego źródła wiedzy, dostarczającego gotowe widomości, które wystarczy tylko sobie przyswoić. Samoucząca się sieć neuronowa, musi najpierw sama odkryć wiedzę, którą następnie zapamięta. Sieć Kohonena wyróżnia się tym od innych sieci, że zachowuje odwzorowanie sąsiedztwa przestrzeni wejściowej. Wynikiem działania sieci jest klasyfikacja przestrzeni w sposób grupujący zarówno przypadki ze zbioru uczącego, jak i wszystkie inne wprowadzenia po procesie uczenia. Polega ona na podawaniu na wejścia sieci sygnałów, a następnie wybraniu w drodze konkurencji zwycięskiego neuronu, który najlepiej odpowiada wektorowi wejściowemu. W sieci tej neurony wyjściowe są uporządkowane. Istotne jest również właściwe zdefiniowanie sąsiedztwa neuronów.
SOMy dokonują redukcji wymiarów danych przez tworzenie map, zazwyczaj jedno lub dwuwymiarowych, które przedstawiają graficzne podobieństwo analizowanych danych, grupując podobne dane razem. Jest to sieć złożona z jednej warstwy, w której neurony ułożone są w pewnym porządku topologicznym. Mapa jednowymiarowa to jeden wiersz lub jedna kolumna neuronów. W strukturze sieci Kohonena istotne jest to, że każdy neuron warstwy wejściowej komunikuje się z każdym neuronem warstwy topologicznej –natomiast neurony w warstwach nie komunikują się pomiędzy sobą.

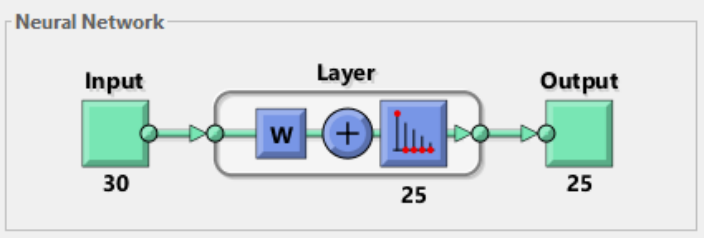
Etapy tworzenia algorytmu:
1. Inicjacja-przyjęcie losowych wartości wag wszystkich połączeń.
2. Współzawodnictwo-dla każdego z sygnałów wejściowych, wyliczana jest wartość sygnału wyjściowego. W efekcie wyznaczany jest neuron zwycięski(najbliższy sygnałowi wejściowemu).
3. Współpraca – neuron zwycięski wyznacza swoje topologiczne sąsiedztwo neuronów, które będą aktywowane w ramach współpracy.
4. Adaptacja-neuron zwycięski modyfikuje wagi w zależności od sygnału wejściowego. W efekcie kolejna prezentacja podobnego sygnału wejściowego spowoduje silniejszą reakcję tego neuronu.

W przypadku tego ćwiczenia działanie sieci oparte jest o zasadę WTAM – Winner Takes Most. W przeciwieństwie do algorytmów WTA, w których tylko jeden neuron może podlegać adaptacji w każdej iteracji w algorytmach WTM neuron zwycięski oraz neurony sąsiadujące z nim, aktualizują swoje wagi według zasady:

$$w_i = w_i + \eta_i G(i,x)[x - w_i]$$

Gdzie funkcja G oznacza tutaj wpływ sąsiedztwa .

Stopień uaktywnienia neuronów z sąsiedztwa zależy od odległości ich wektorów wagowych od wag neuronu wygrywającego. Im dalsza jest odległość od zwycięzcy, tym mniejsza jest zmiana wartości wag neuronu.



Do wykonania danego zadania użyłam funkcji z pakietu Matlab - selforgmap(). Tworzy ona samoorganizującą się mapę do grupowania zestawu danych. Samoorganizujące się mapy używane są zarówno do grupowania danych jak i do zmniejszania wymiarów danych, ilustrują również graficzne podobieństwo między nimi. Dzięki niej możliwe jest stworzenie mapy neuronów z odpowiednimi parametrami.

```
net = selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn);
```

gdzie:

- dimensions – wektor rzędów wymiarów sieci
- coverSteps – liczba kroków szkoleniowych
- initNeighbor – początkowy rozmiar sąsiedztwa
- topologyFcn – funkcja topologii warstw
- distanceFcn – funkcja odległości neuronowej

Parametry te zostały ustawione następująco:

```
dimensions   = [5 5];
coverSteps   = 100;
initNeighbor = 1;
topologyFcn  = 'hextop';
distanceFcn  = 'linkdist';
```

gdzie:

- ‘hextop’ – funkcja topologii sześciokątnej ; oblicza położenia neuronów dla warstw, których neurony są ułożone w n-wymiarowy sześciokątny wzór
- ‘linkdist’ – funkcja odległości odległości między neuronami warstwy biorąc pod uwagę ich położenie

Jako funkcje topologii warstw możliwe jest użycie również opcji ‘gridtop’, która ukazuje topologię przy użyciu prostokątów (oblicza położenia neuronów dla warstw, których neurony są ułożone w n-wymiarową siatkę) , oraz ‘randtop’ , która prezentuje losowo ułożone neurony (oblicza położenia neuronów dla warstw, których neurony są ułożone w n-wymiarowy losowy wzór).

W przypadku funkcji odległości neuronowej mamy do dyspozycji również opcję ‘dist’ , która jest funkcją używana do znalezienia odległości między neuronami warstwy biorąc pod uwagę ich położenie używając przy tym odległości euklidesowej.

Jak widać powyżej początkowy rozmiar sąsiedztwa ustawiony jest na 1, ponieważ korzystamy tutaj z reguły WTM, która bierze pod uwagę neurony sąsiednie przy wyznaczaniu zwycięzcy.

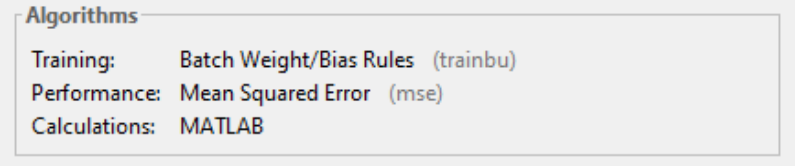
W celu wykonania zadania został stworzony zestaw danych uczących, który składa się z 20 dużych liter alfabetu. Przedstawione są one jako matryca 5x6 o wartościach 0 lub 1 co daje nam 30 pól.

```
      % A  B C D E F G H I  J  K L M N O P R S T U
dane =[0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1;
       1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0;
       1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0;
       1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 0 1 0;
       0 0 1 0 1 1 0 1 0 1 0 1 1 1 0 0 0 0 1 1;
       1 1 1 1 1 1 1 1 0 1 1 0 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 1 0 0 1 0 0 0 0 1 0 0;
       1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 1;
       1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1;
       0 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0;
       0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 0 1 0;
       0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
       1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1;
       1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1;
       1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0;
       1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0;
       1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 0;
       1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 0 0 0 0 1;
       1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1;
       0 0 0 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 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 1 0 0;
       1 1 0 1 0 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0;
       1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0;
       0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1;
       0 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1;
       0 1 1 1 1 1 1 0 1 0 0 1 0 0 1 0 0 0 0 1;
       1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 0;
       ];
```

Ponadto dla funkcji selforgmap() wywołane zostały następujące procedury:

```
net.trainParam.epochs = 400;
net.trainFcn = 'trainbu';
[net,tr] = train(net,dane);
```

Rozpoczynamy od określenia długość treningu ustalając liczbę epok. Na końcu wywołujemy funkcję ‘train’ dokonującą treningu sieci. Pozwala nam uczyć daną sieć na podstawie zadanych parametrów w net.trainParam za pomocą odpowiedniej funkcji uczącej net.trainFcn. Jej działanie dla sieci jednokierunkowych polega na obliczaniu metodą iteracyjną wartości współczynników wagowych.

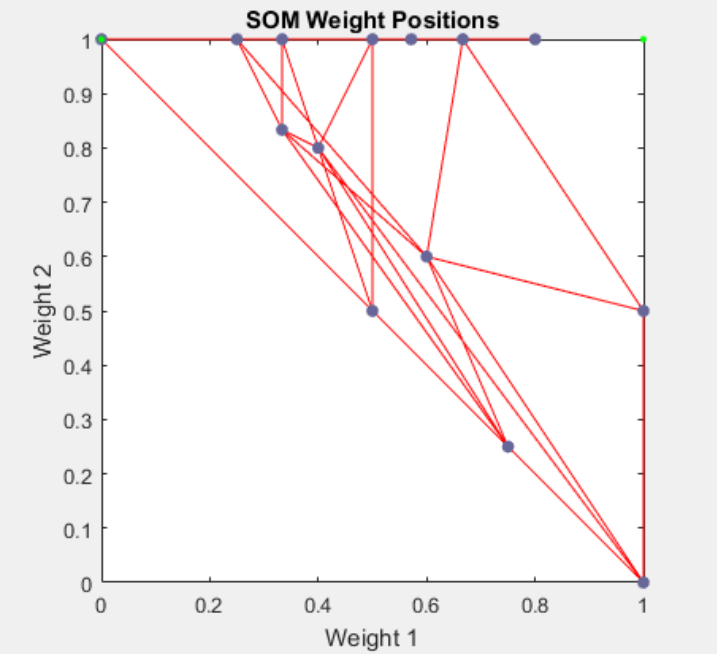


Algorytm ‘trainbu’ szkoli sieć z regułami ważenia i obciążania z aktualizacjami wsadowymi. Aktualizacje wag i błędów pojawiają się na końcu całego przebiegu danych wejściowych. Jest to funkcja szkolenia dla samoorganizujących się map.

Po treningu została wywołana procedura:

```
plotsompos(net,dane);
```

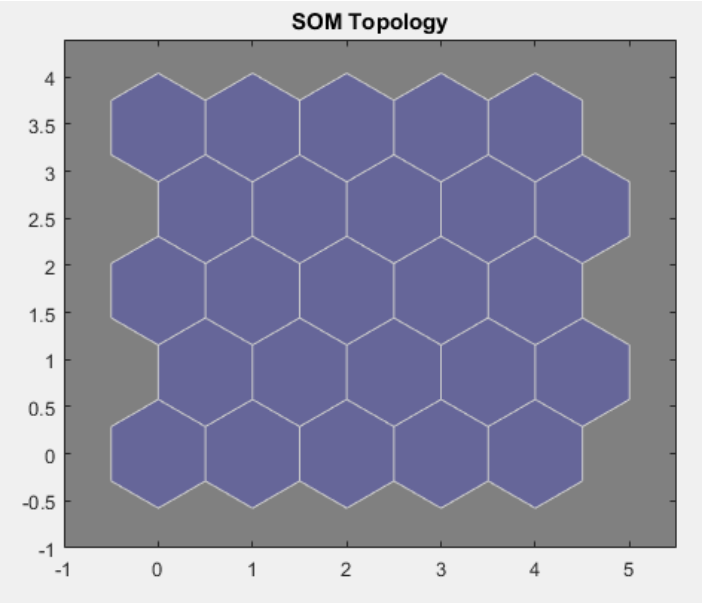
Dzięki niej wyświetlone zostały pozycje wag mapy samoorganizującej się.



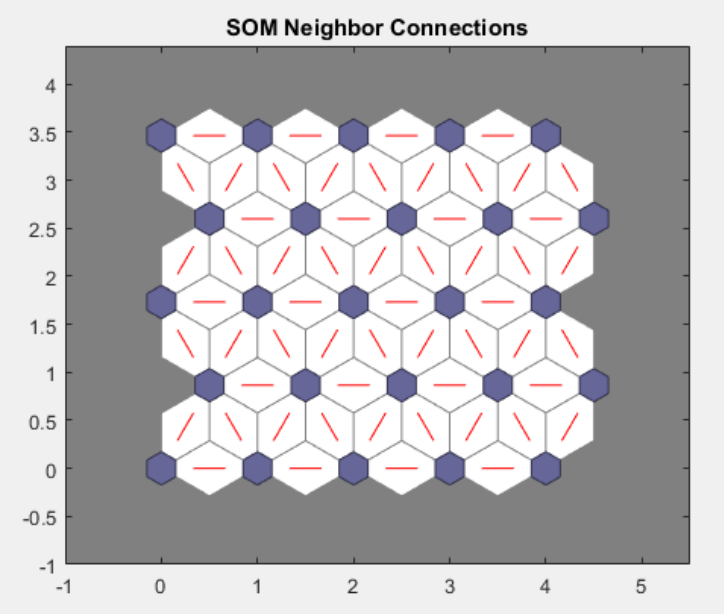
Wektory wejściowe ukazane są jako zielone kropki. Funkcja ta pokazuje w jaki sposób SOM klasyfikuje przestrzeń wejściową. W tym celu wyświetla niebiesko-szare kropki jako neurony oraz łączy sąsiednie neurony czerwonymi liniami.

2. Zestawienie otrzymanych wyników.

Samoorganizująca się mapa w topologii sześciokątnej:

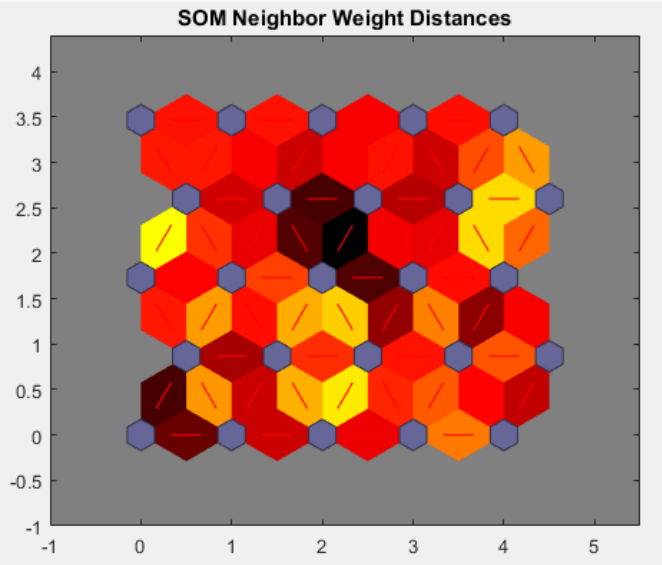


Samoorganizująca się mapa ukazująca połączenia sąsiedzkie:



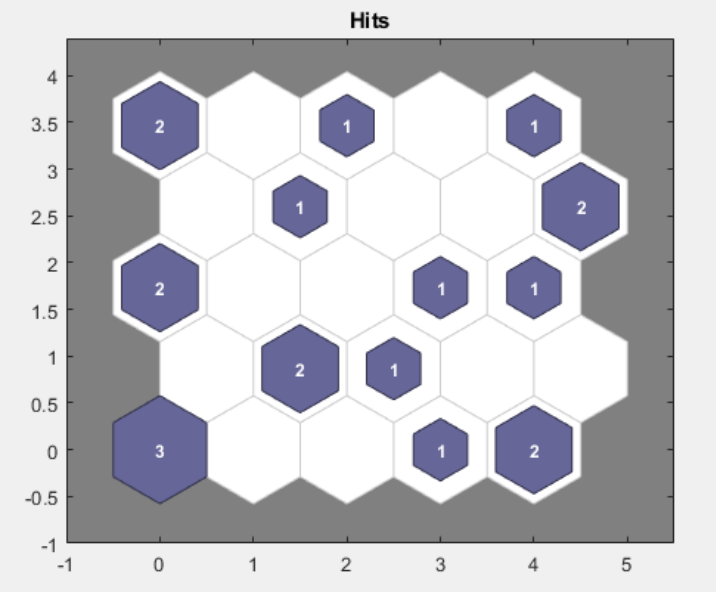
Na mapie widoczne są neurony przedstawione jako szaro-niebieskie łaty oraz ich bezpośrednie relacje z sąsiadującymi neuronami ukazane poprzez czerwone linie.

Samoorganizująca się mapa ukazująca dystans między neuronami:



Sąsiadujące łaty są zabarwione od koloru czarnego do żółtego, aby pokazać, jak blisko każdy neuron jest swojego sąsiada. Im ciemniejszy kolor tym dystans jest większy.

Samoorganizująca się mapa prezentująca trafienia neuronu:



Każdy neuron pokazuje liczbę wektorów wejściowych, które klasyfikuje. Przedstawia ile razy był zwycięzcą i jakie grupy stworzył.

3. Analiza i dyskusja błędów uczenia i testowania oraz wyłonionych cech dla wyników opracowanej sieci w zależności od wartości współczynnika uczenia

Z przedstawionych powyżej wykresów możemy zauważyć, że ważne jest dobranie odpowiedniej liczby neuronów w celu grupowania sieci. Na ostatniej mapie prezentującej trafienia zostały ukazane grupy. W tym przypadku podział ten jest dużo bardziej uśredniony. Świadczy to o tym, że wagi były modyfikowane zgodnie z regułą WTM, gdzie uaktualnianie następowało nie tylko dla zwycięzcy, ale również dla neuronów z nim sąsiadujących. Dzięki temu nie dochodzi do dominacji jednego neuronu.

Podczas testowania różnych funkcji topologii sieci okazało się, że 'randtop' nie spełnia naszych oczekiwań. Nie można rozwiązać tego zadania przy jej użyciu. Jednak 'gridtop' oraz 'hextop' dają podobne rezultaty.

Analiza wyników opracowanej sieci w zależności od wartości współczynnika uczenia nie jest możliwa do zrealizowania w tej wersji programu Matlab. Pojawia się on jednak w regule Kohonena. Na ogół jest on malejącą funkcją wraz z iteracjami. Dzięki temu kontroluje rozmiar wektora wag, a więc jest kluczowy do osiągnięcia prawidłowego rozwiązania.

4. Wnioski.

W celu poznania działania sieci Kohonena przy wykorzystaniu reguły WTM korzystałam z funkcji dostępnych w pakiecie Matlab. Wybrałam funkcję selforgmap() oraz podczas testowania sieci zauważyłam, że idealnie się nadaje do odwzorowania istotnych cech liter alfabetu . Podczas analizy wyników zauważamy, że zasada WTM nie powoduje rywalizacji między neuronami. Z racji tego, że wagi są uaktualniane są zarówno dla zwycięskiego neuronu jak i neuronów z nim sąsiadujących nie dochodzi do sytuacji, że gromadzi on większość lub nawet całość sygnałów wejściowych. W tym przypadku grupy są znacznie mniejsze oraz więcej neuronów jest w stanie je grupować. Ponad to istotne jest dobranie odpowiedniej liczby neuronów w celu grupowania sieci. Biorąc pod uwagę współczynnik uczenia wnioskujemy, że kontroluje on rozmiar wektora wag, przez co musi być odpowiednio dokładny aby wynik był prawidłowy.

5. Listing całego kodu.

```
close all; clear all; clc;

% Zestaw danych składający się z 20 dużych liter.
Przedstawione są jako matryca 5x6 o wartościach 0 lub 1.

        %A B C D E F G H I K L J M N O P R S T U
  dane =[0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1;
         1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0;
         1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0;
         1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 0 1 0;
         0 0 1 0 1 1 0 1 0 1 0 1 1 1 0 0 0 0 1 1;
         1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1;
         0 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 1 0 0 0 0 0 0 0 0 0 1 0;
0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0;
1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 1;
1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1;
0 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0;
0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 0 1 0;
0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1;
1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1;
1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0;
1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0;
1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 0;
1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 0 0 0 0 1;
1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1;
0 0 0 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 1 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 1 0 0;
1 1 0 1 0 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0;
1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0;
0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1;
0 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1;
0 1 1 1 1 1 1 0 1 0 0 1 0 0 1 0 0 0 0 1;
1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 0;
];
```

```
[m, n] = size(dane);
 Określenie parametrów SOM
% wektor rzędów wymiarów sieci
dimensions   = [5 5];
% liczba kroków szkoleniowych
coverSteps   = 100;
% początkowy rozmiar sąsiedztwa
initNeighbor = 1;
% funkcja topologii warstw  - sześciokątna
topologyFcn  = 'hextop';
% - funkcja odległości neuronowej
distanceFcn  = 'linkdist';

Tworzenie sieci Kohonena przy pomocji funkcji selforgmap()  o rozmiarze 5x5
net = selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn);
% Określenie długości treningu
net.trainParam.epochs = 400;
% Określenie algorytmu treningu
net.trainFcn = 'trainbu';
% Trening sieci
[net,tr] = train(net,dane);
wyjscie = net(dane);
```

```
% Skonwertowanie wektorów na indeksy
classes = vec2ind(wyjscie);
 % Wyświetlenie pozycji wag
plotsompos(net,dane);
```