Marcin Damek
Inżynieria obliczeniowa, gr 1
Nr. albumu 285952

Sprawozdanie numer 5

Budowa i działanie sieci Kohonena dla WTA

Celem ćwiczenia jest poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTA do odwzorowywania istotnych cech kwiatów.

Zadania do wykonania:

a) Przygotowanie danych uczących zawierających numeryczny opis cech kwiatów. przykładowy zestaw: https://en.wikipedia.org/wiki/Iris_flower_data_set

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

c) Uczenie sieci dla różnych współczynników uczenia.

d) Testowanie sieci.

Część teoretyczna

Sieci Kohonena są jednym z podstawowych typów sieci samoorganizujących się. Właśnie dzięki zdolności samoorganizacji otwierają się zupełnie nowe możliwości - adaptacja do wcześniej nieznanych danych wejściowych, o których bardzo niewiele wiadomo. Wydaje się to naturalnym sposobem uczenia, który jest używany chociażby w naszych mózgach, którym nikt nie definiuje żadnych wzorców, tylko muszą się one krystalizować w trakcie procesu uczenia, połączonego z normalnym funkcjonowaniem. Sieci Kohonena stanowią 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 neuronowej
Bardzo istotną kwestią jest struktura sieci neuronowej. Pojedynczy neuron jest mechanizmem bardzo prostym i przez to niewiele potrafiącym. Dopiero połączenie wielu neuronów ze sobą umożliwia prowadzenie dowolnie skomplikowanych operacji. Ze względu na raczej niewielką wiedzę o faktycznych zasadach funkcjonowania ludzkiego mózgu, powstało wiele różnych architektur, które starają się naśladować budowę i zachowanie poszczególnych fragmentów układu nerwowego. Najczęściej stosuje się w tego typu sieciach architekturę jednokierunkową jednowarstwową. Jest to podyktowane faktem, że wszystkie neurony muszą uczestniczyć w konkurencji na równych prawach. Dlatego każdy z nich musi mieć tyle wejść ile jest wejść całego systemu.

Algorytm uczenia
Algorytmem, od którego nazwę wzięła cała klasa sieci są samoorganizujące się mapy Kohonena. Zostały one opisane przez ich twórcę w publikacji "The Self Organising Map". Kohonen zaproponował dwa rodzaje sąsiedztwa: prostokątne i gaussowskie.
Pierwsze ma postać:

$$G(i,x)=\begin{cases}1 & dla \quad d(i,w)\le\lambda\\ 0 & dla \quad d(i,w)>\lambda\end{cases}$$

A drugie:

$$G(i,x)=\exp\left(-\frac{d^2(i,w)}{2\lambda^2}\right)$$

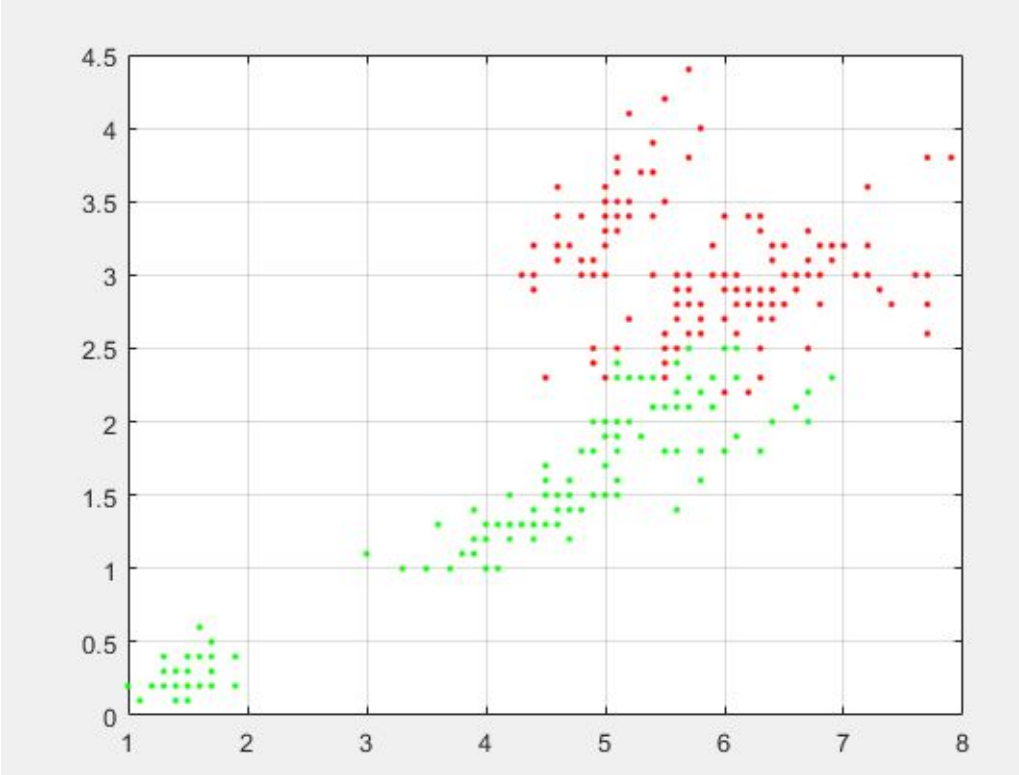
Winner Takes All - WTA
W konkurencyjnej metodzie uczenia sieci, tylko jeden element wyjciowy moze znajdować się w stanie aktywnym. Nazywany jest on zwycięzcą, a schemat takiej reguły aktywacji neuronów określany jest mianem “zwycięzca bierze wszystko”

Zasady działania sieci Kohonena:

- Wejścia (tyle, iloma parametrami opisano obiekty) połączone są ze wszystkimi węzłami sieci
- Każdy węzeł przechowuje wektor wag o wymiarze identycznym z wektorami wejściowymi
- Każdy węzeł oblicza swój poziom aktywacji jako iloczyn skalarny wektora wag i wektora wejściowego (podobnie jak w zwykłym neuronie)
- Ten węzeł, który dla danego wektora wejściowego ma najwyższy poziom aktywacji, zostaje zwycięzcą i jest uaktywniony
- Wzmacniamy podobieństwo węzła-zwycięzcy do aktualnych danych wejściowych poprzez dodanie do wektora wag wektora wejściowego (z pewnym współczynnikiem uczenia)
- Każdy węzeł może być stowarzyszony z pewnymi innymi, sąsiednimi węzłami - wówczas te węzły również zostają zmodyfikowane, jednak w mniejszym stopniu.

Część praktyczna

Do wykonania zadania użyłem narzędzia Neural Network Training Tool z pakietu Matlab. Dzięki niemu możemy w stosunkowo łatwy sposób tworzyć i uczyć sieci neuronowe.
Wizualizacja danych płatka i działki kielicha.

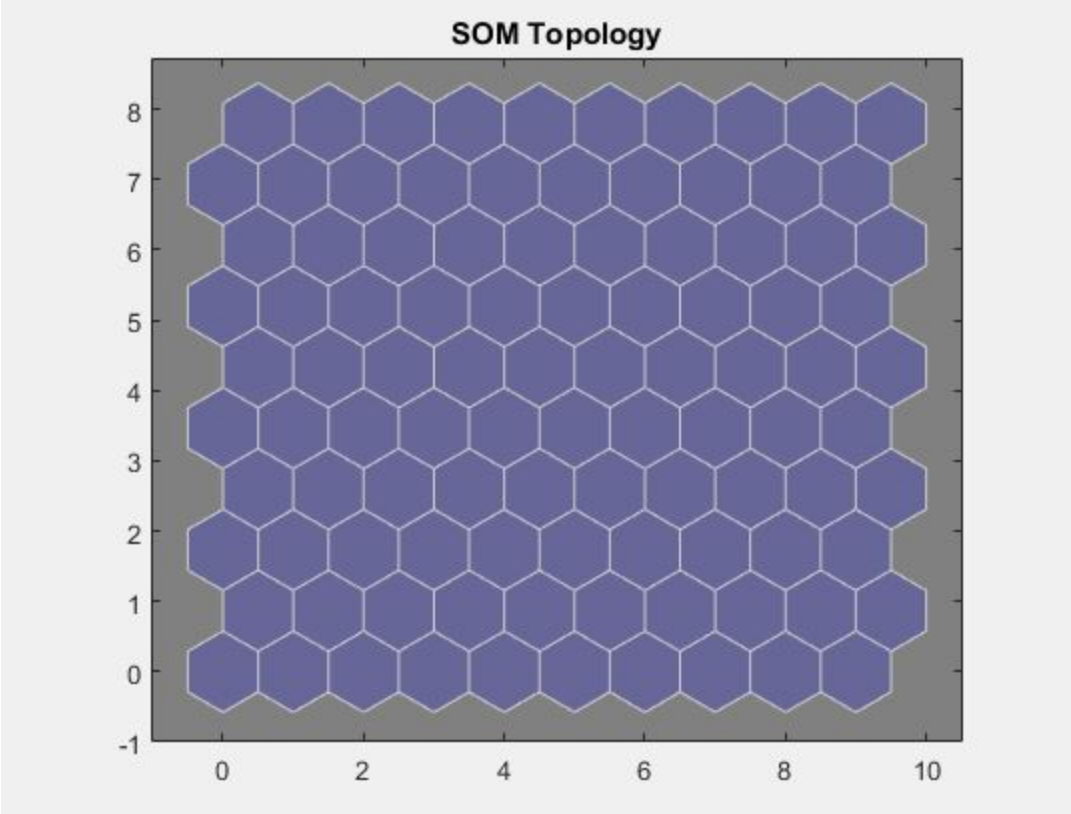


W zadaniu najważniejsza okazała się funkcja:

`selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn)`

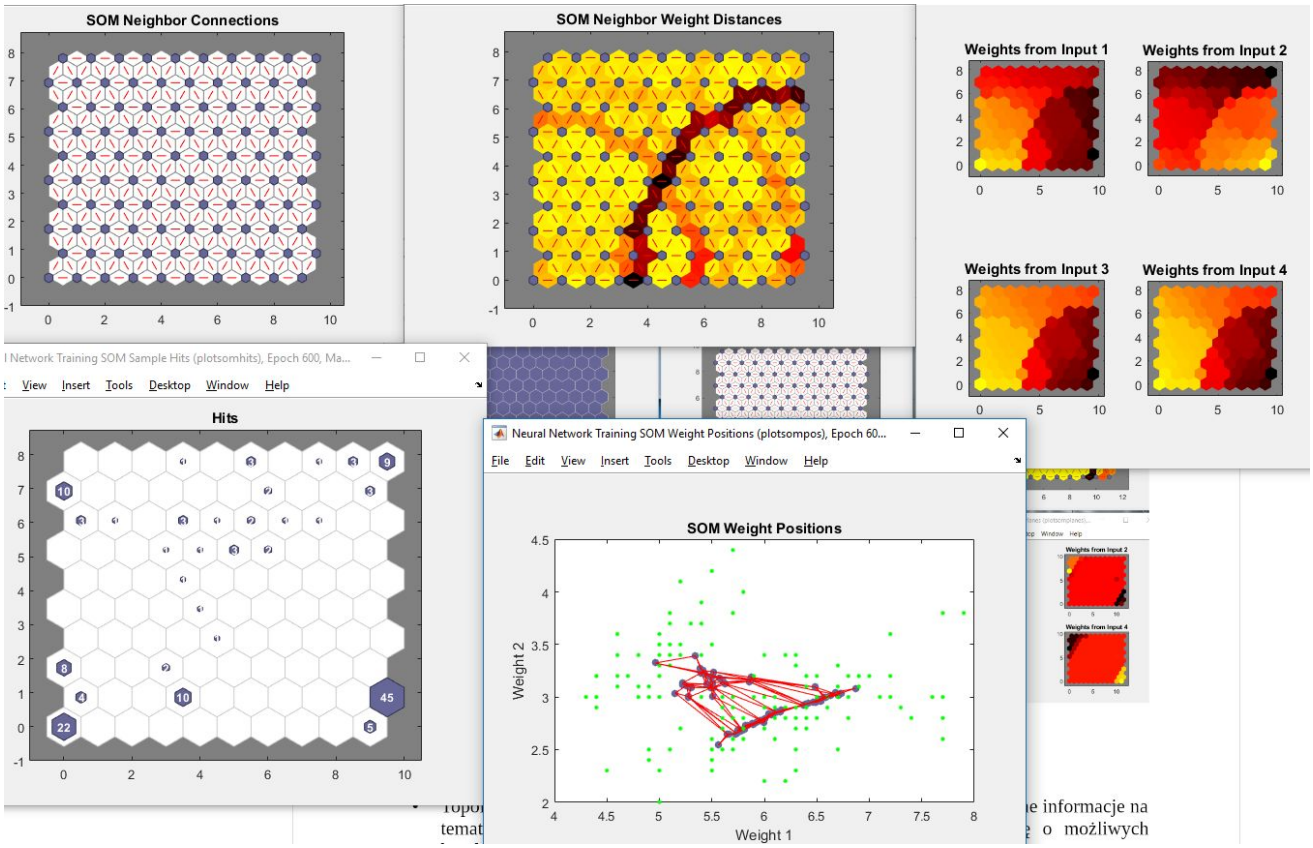
Dimensions - rozmiar wektora
coverSteps - liczba kroków szkoleniowych
initNeighbor - początkowy rozmiar sąsiedztwa
topologyFcn - funkcja topologii warstw
distanceFcn - odległość neuronów

Wszystkie próby wykonywałem dla topologii warstwy sześciokąta określonej w parametrze topologyFcn równej 'hextop' ponieważ w tej konfiguracji były lepiej widoczne wyniki. Poniżej wygląd zastosowanej topologii.

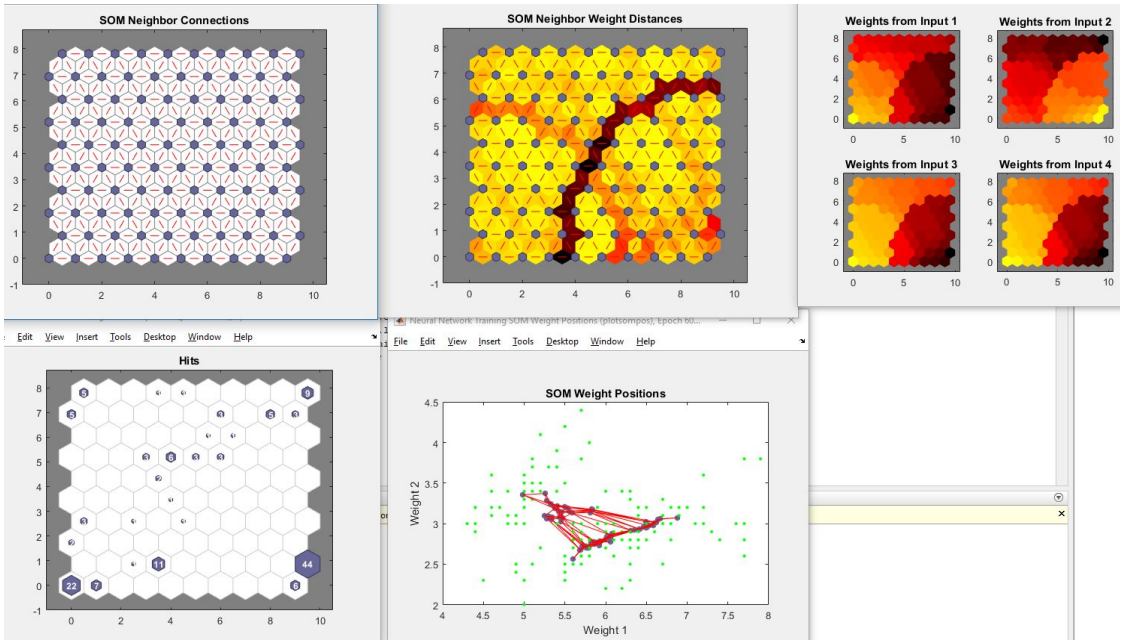


Analizy dokonałem dla 4 współczynników uczenia: 0.2, 0.4, 0.6, 0.8
Wymiar wektora ustaliłem na 10x10, oraz etapy szkolenia w celu pokrycia przestrzeni wejściowej ustawiłem na 100. Maksymalna liczna epok treningowych to 600.

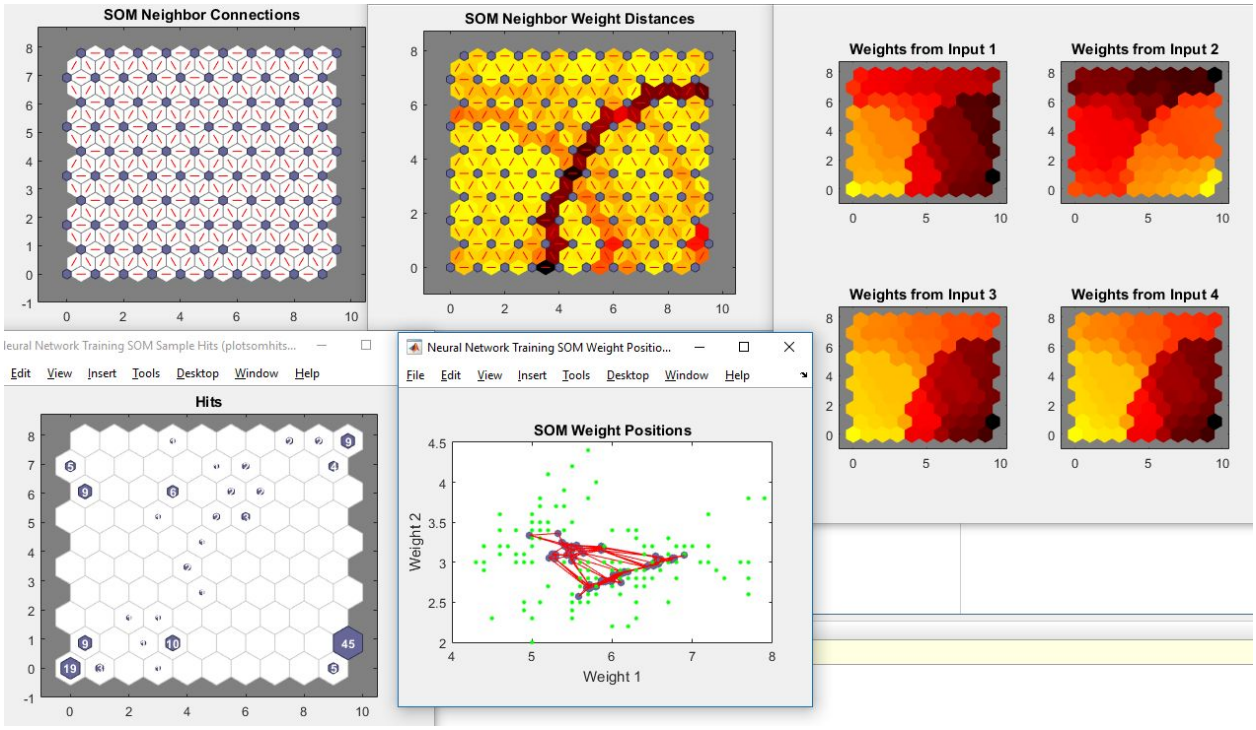
Współczynnik uczenia 0,2



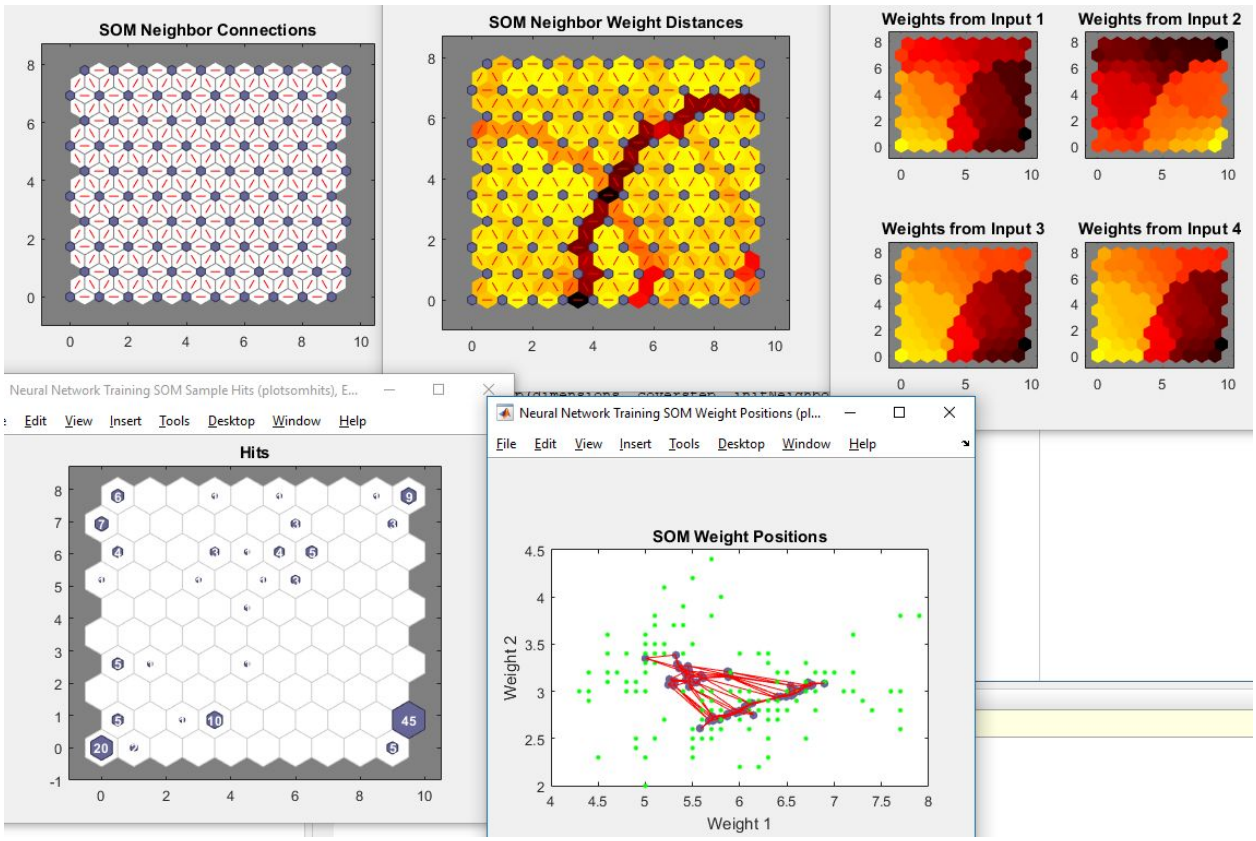
Współczynnik uczenia 0,4



Współczynnik uczenia 0,6



Współczynnik uczenia 0,8



Wnioski

W moim przypadku, każdy z neuronów reprezentowany jest poprzez jeden sześciokąt na wizualizacji. Algorytmy uczenia nienadzorowanego dobrze wykorzystuje się w miejscach, w których grupujemy obiekty przy pomocy danych cech. Dzięki funkcji iris_dataset możemy uzyskać dostęp do danych zaimplementowanych w pakiecie Matlab, które zawierając numeryczny zapis czterech cech kwiatu irysa, umieszczonych w tablicy.
Wykres hits w każdej próbie oznacza liczbę, ile razy dany neuron zwyciężył podczas rywalizacji. Topologia heksagonalna udostępnia bardziej szczegółowe i łatwiejsze w odczytaniu informacje niż druga której podgląd możemy wygenerować w matlabie - kwadratową. W moim przypadku współczynnik uczenia nie miał bardzo dużego wpływu na zaobserwowane wyniki. Im bardziej zwiększałem współczynnik uczenia, czas potrzebny na naukę modelu zmniejszał się. Na wykresach dystansu, możemy zaobserwować odległości między neuronami - im ciemniejszy kolor tym większa odległość.

Kod programu

```
close all; clear all; clc;

input = iris_dataset; %Standardowe dane wejsciowe dostarczane przez Matlab
% I i II kolumna dane o długości i szerokości działki kielicha,
% III i IV kolumna dlugosc i szerokosc płatka
plot(input(1, :) ,input(2, :), 'r.', input(3, :), input(4, :), 'g.');
hold on; grid on;

dimensions = [10 10]; % wymiary wektora
coverstep = 100; %etapy szkolenia w celu pokrycia przestrzeni wejściowej
initNeighbor = 0; % wejściowy rozmiar sąsiedztwa
topologyFcn = 'hextop'; %ksztalt funkcji topologicznej
distanceFcn = 'dist'; %odleglosc miedzy neuronami

net = selforgmap(dimensions, coverstep, initNeighbor, topologyFcn, distanceFcn); %tworzenie
mapy samoorganizacji
net.trainFcn = 'trainbu'; %uczenie bez nauczyciela
net.trainParam.epochs = 600;
net.trainParam.lr = 0.8; %współczynnik uczenia
[net, tr] = train(net, input); %trening
y = net(input); %testowanie
```