Marcelina Bałamut, IO, gr. 1

Temat ćwiczenia:

Budowa i działanie sieci Kohonena dla WTA.

Cel ćwiczenia

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

Opis budowy oraz wykorzystanych sieci i algorytmów uczenia:

System realizujący funkcjonowanie sieci samoorganizującej powinien składać się z kilku podstawowych elementów. Pierwszym z nich jest macierz neuronów pobudzanych przez sygnały wejściowe. Sygnały te powinny opisywać pewne charakterystyczne cechy zjawisk zachodzących w otoczeniu, tak, aby na ich podstawie sieć była w stanie je pogrupować. Informacja o zdarzeniach jest przekładana na bodźce pobudzające neurony. Zbiór sygnałów przekazywanych do każdego neuronu nie musi być identyczny, nawet ich ilość może być różna. Muszą one jednak spełniać pewien warunek, a mianowicie jednoznacznie określać dane zdarzenia. Kolejną częścią składową sieci jest mechanizm, który dla każdego neuronu określa stopień podobieństwa jego wag do danego sygnału wejściowego oraz wyznacza jednostkę z największym dopasowaniem - zwycięzcę. Obliczenia zaczynamy dla wag równych małym liczbom losowym, przy czym ważne jest, aby nie zachodziła żadna symetria. W trakcie uczenia wagi te są modyfikowane w taki sposób, aby najlepiej odzwierciedlać wewnętrzną strukturę danych wejściowych. Istnieje jednak niebezpieczeństwo, że zwiążą się one z pewnymi wartościami zanim jeszcze grupy zostaną prawidłowo rozpoznane i wtedy trzeba ponawiać uczenie z innymi wagami. Wreszcie konieczne do przeprowadzenia samoorganizacji jest, aby sieć była wyposażona w zdolność do adaptacji wartości wag neuronu zwycięzcy w zależności od siły, z jaką odpowiedział on na dane wejście. Załóżmy, że jednostkę, której odpowiedź na dane pobudzenie jest maksymalna, będziemy nazywali "obrazem" tego pobudzenia. Wtedy możemy przyjąć, że sieć jest uporządkowana, jeśli topologiczne relacje między sygnałami wejściowymi i ich obrazami są takie same. Zatem sieć SOM jest to sieć jednokierunkowa składająca się w dwóch warstw: wejściowej i wyjściowej, która zawiera mapę topologiczną. Natomiast każdy neuron połączony jest ze wszystkimi składowymi n-wymiarowego wektora wejściowego, a wagi połączeń neuronów tworzą wektor wi=[wi1, wi2…win]^T. W strukturze sieci Kohonena istotne jest to, że każdy neuron warstwy wejściowej komunikuje się z każdym neuronem warstwy wyjściowej, a neurony w warstwach nie komunikują się między 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. Adaptacja-neuron zwycięski modyfikuje wagi w zależności od sygnału wejściowego. W efekcie kolejna prezentacja podobnego sygnału wejściowego spowoduje silniejszą reakcję tego neuronu.

Reguła WTA:

Algorytm WTA polega na obliczaniu aktywacji każdego neuronu, a następnie wyborze zwycięzcy o największym sygnale wyjściowym. Zwycięski neuron, a więc ten dla którego odległość między wektorem wag *w* a wektorem wejściowym *x* jest najmniejsza podlega uczeniu, polegającym na adaptacji swoich wag w kierunku wektora *x*:

$$w(k+1)=w(k)+n(x-w(k))$$

Jeżeli wektory wejściowe są normalizowane, wówczas minimalna odległość między wektorem wag *w* a wektorem wejściowym *x* odpowiada równocześnie maksymalnej wartości iloczynu skalarnego: *w* **x*. W ścisłym algorytmie WTA pozostałe neurony nie podlegają uczeniu, co prowadzi do słabej zbieżności.

Algorytm:

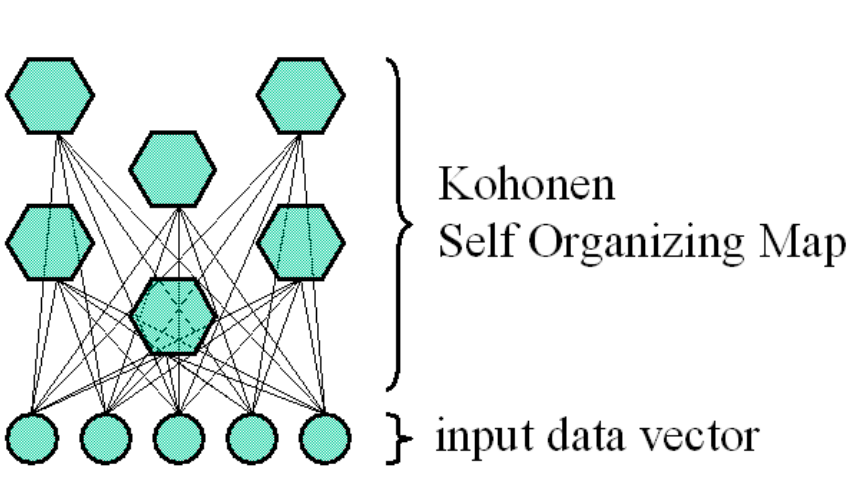
1. Przyjęcie losowych znormalizowanych wartości wag poszczególnych neuronów.
2. Po podaniu pierwszego wektora wejściowego *x* wyłaniany jest zwycięzca o mierze *k*.

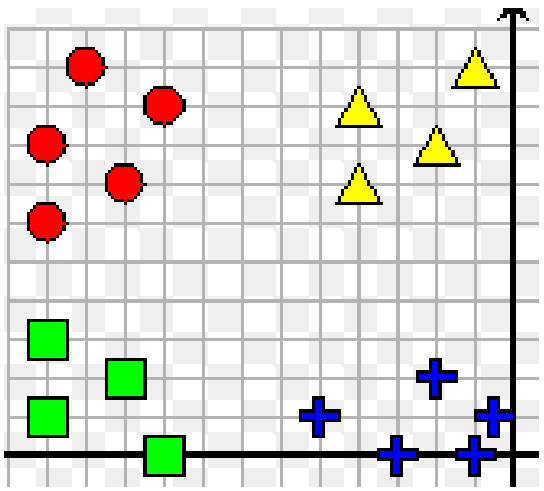
$$\mathbf{w}_k^T\mathbf{x} = \max_{i=1,2,\ldots,K}(\mathbf{w}_i^T\mathbf{x})$$

3. Aktualizacja wag neuronu zwycięzcy (neurony przegrywające maja na wyjściu stan 0, co blokuje proces aktualizacji ich wag).

$$w_C(k+1) = w_C(k) + \eta(k)[x(k) - w_C(k)]$$

4. Wektor wag zwycięzcy jest zwiększany o ułamek różnicy *x*-*w*.





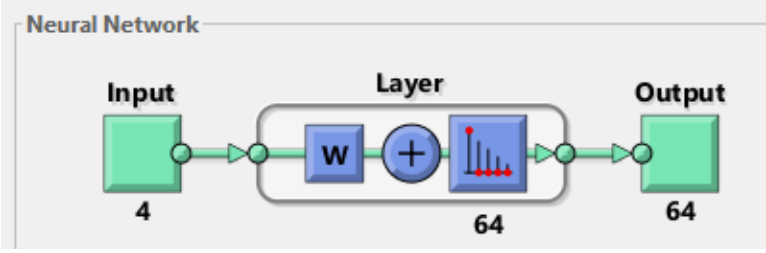
W wykonywanym zadaniu jako dane uczące została wykorzystana macierz 4x150, która opisuje konkretne właściwości kwiatów. Parametry identyfikują 150 różnych kwiatów pogrupowanych na 3 gatunki. Pierwsza kolumna oznacza długość płatka, druga -szerokość płatka, trzecia - długość kielicha i ostatnia szerokość kielicha, czyli łącznie 600 pól, gdzie każde z pól przedstawione jest jako wartość liczbowa identyfikująca odpowiednią własność. Dane zostały wczytane do zmiennej x. W zestawie iris_dataset znajdują się trzy gatunki kwiatów: setosa, versicolor oraz virginica.

```
x=iris_dataset;
```

Przykładowe dane:

```
4.6000    5.1000    4.8000    5.0000    5.0000
3.6000    3.3000    3.4000    3.0000    3.4000
1.0000    1.7000    1.9000    1.6000    1.6000
0.2000    0.5000    0.2000    0.2000    0.4000
```

Do implementacji danego problemu użyłam wbudowanej funkcji pakietu Matlab o nazwie selforgmap().



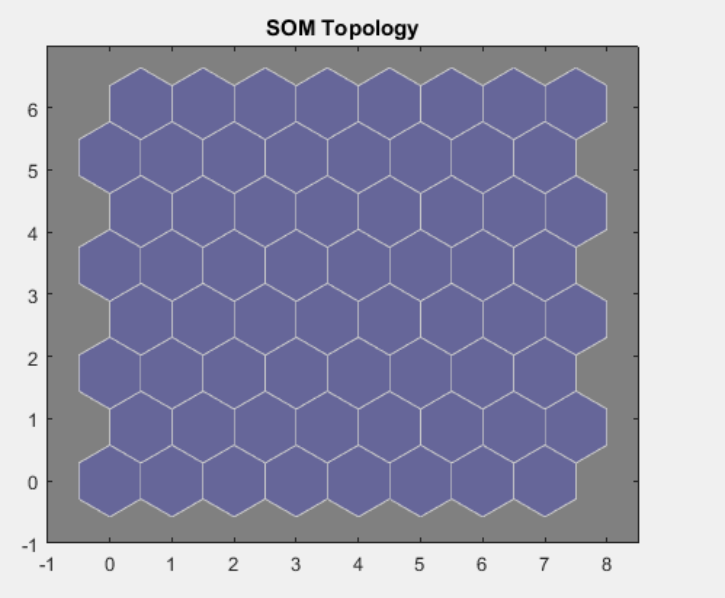
Funkcja selforgmap() tworzy samoorganizującą się mapę do grupowania zestawu danych. Samoorganizujące się mapy uczą się klastra danych na podstawie podobieństwa przypisując tę samą liczbę wystąpień do każdej klasy. Są używane zarówno do grupowania danych jak i do zmniejszania wymiarów danych. Po wywołaniu funkcji tworzony jest w przestrzeni obiekt net w którym zapisane są wszystkie informacje na temat utworzonej sieci.

```
dimensions= [8 8];
coverSteps=100;
initNeighbor=0;
topologyFcn='hextop';
distanceFcn='dist';
net= selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn);
```

gdzie:
demensions- wektor rzędów wymiarów, domyślnie [8 8]
coverSteps- liczba kroków szkoleniowych dla początkowego pokrycia przestrzeni wejściowej, domyślnie 100
initNeighbor-początkowy rozmiar sąsiedztwa
topologyFcn-finkcja topologii warstw, domyślnie 'hextop'
distanceFcn-funkcja odległości neuronowej, ustawiona na 'dist'

Jak widać powyżej promień sąsiedztwa ustawiony jest na 0, ponieważ korzystamy tutaj z reguły WTA, która nie bierze pod uwagę neuronów sąsiednich przy wyznaczaniu zwycięzcy.

Do analizowanego problemu została wykorzystana funkcja topologii sześciokątnej. Hextop oblicza położenia neuronów dla warstw, których neurony są ułożone w n-wymiarowy sześciokątny wzór.



Natomiast dist jest to funkcja odległości warstw używana do znalezienia odległości pomiędzy neuronami warstwy, biorąc pod uwagę ich położenie używając odległości euklidesowej. Funkcje wagowe stosują wagi do danych wejściowych aby uzyskać ważone dane wejściowe.

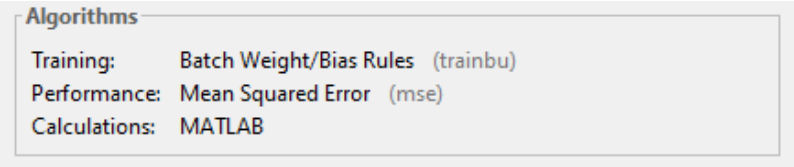
Wzór na miarę euklidesową:

$$d(x, W_i) = \|x - W_i\| = \sqrt{\sum_{j=1}^{N} \left(x_j - W_j^{(i)}\right)^2}$$

Dla funkcji selforgmap została wywołana procedura train, dokonująca treningu zbudowanej sieci. Jest ona uniwersalna, wywoływana w jednolity sposób dla wszystkich typów sieci neuronowych. Działa ona dla sieci jednokierunkowych obliczając wartości współczynników wagowych metodą iteracyjną. Jako parametry przyjmuje obiekt net i wektor wejścia. Przed wywołaniem ustalona została maksymalna liczba kroków uczących(epochs) oraz funkcja trenująca.

```
net.trainParam.epochs = 300;
net.trainFcn='trainbu';
[net,tr] = train(net,x);
```

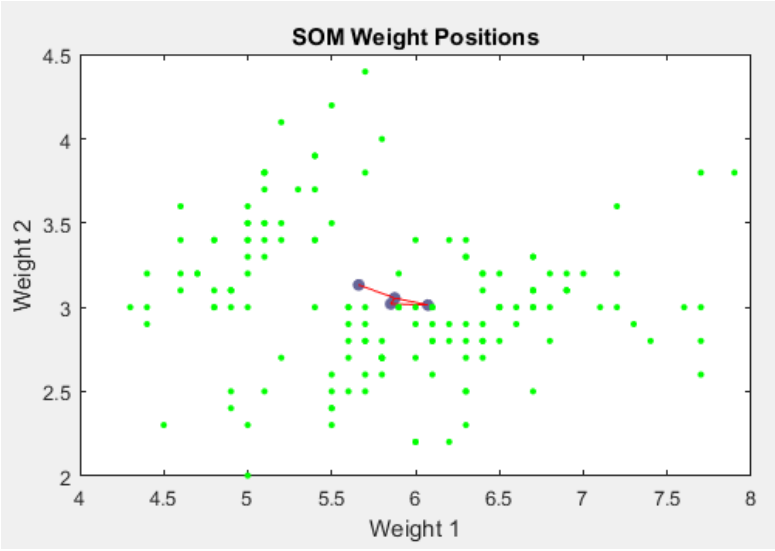
Do treningu sieci została wykorzystana funkcja ‘trainbu’, która realizuje szkolenie w zakresie wagowym i obciążeniowym z aktualizacjami wsadowymi. Aktualizacja wag i błędów następuje na końcu całego przebiegu danych wejściowych. Jest to funkcja szkolenia dla samoorganizujących się map. Sieć identyfikuje zwycięski neuron dla każdego wektora wejściowego. Każdy wektor wagowy przesuwa się do średniej pozycji wszystkich wektorów wejściowych, dla których jest zwycięzcą lub znajduje się w jego pobliżu.



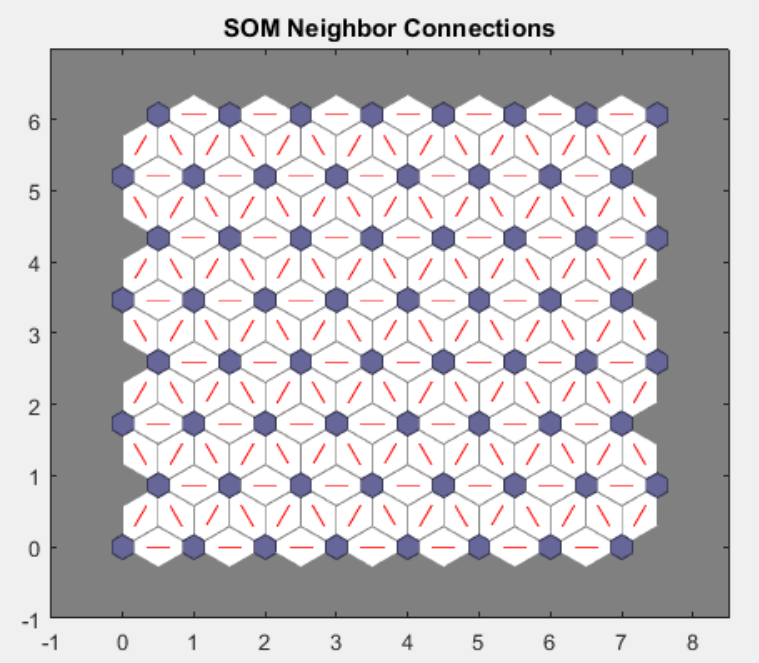
Zestawienie otrzymanych wyników

W analizowanym problemie wyniki przeprowadzonych testów najlepiej odzwierciedlają wykresy określające różne parametry procesu, generowane przez pakiet Matlab.

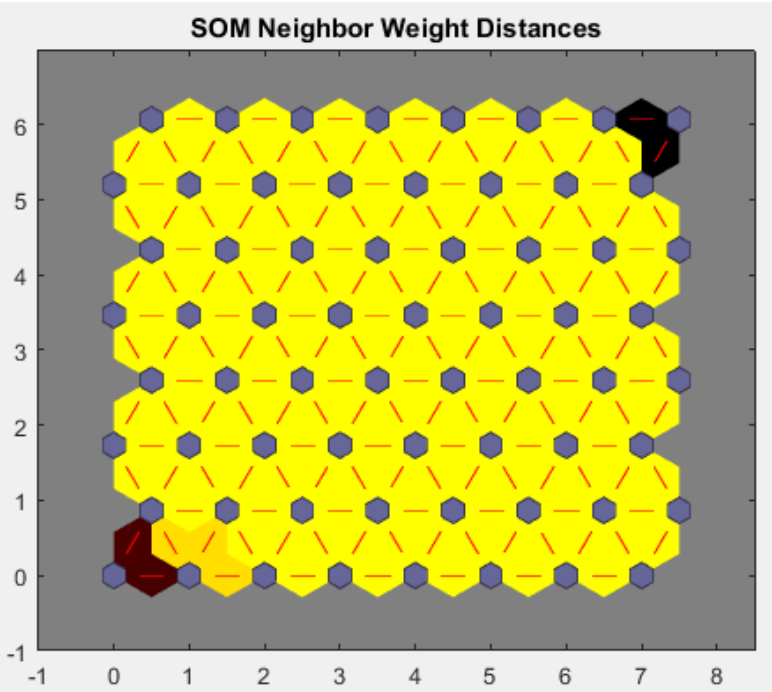
- Na poniższym wykresie kolorem zielonym zostały zaznaczone punkty które odzwierciedlają rozkład danych wejściowych za pomocą wag. Punkty o kolorze niebieskim identyfikują neurony, które są połączone czerwoną linią. Wykres ten pokazuje w jaki sposób SOM klasyfikuje przestrzeń wejściową.



- Poniższy wykres przedstawia neurony jako niebieskie sześciokąty oraz połączenia między nimi jako czerwone linie.

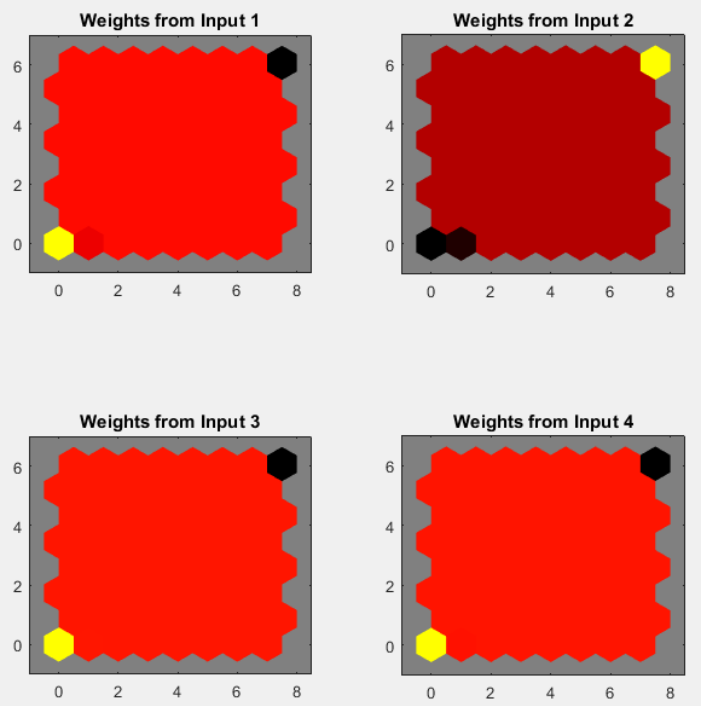


- Odległości między neuronami wyznaczone metodą euklidesową przedstawią się następująco:



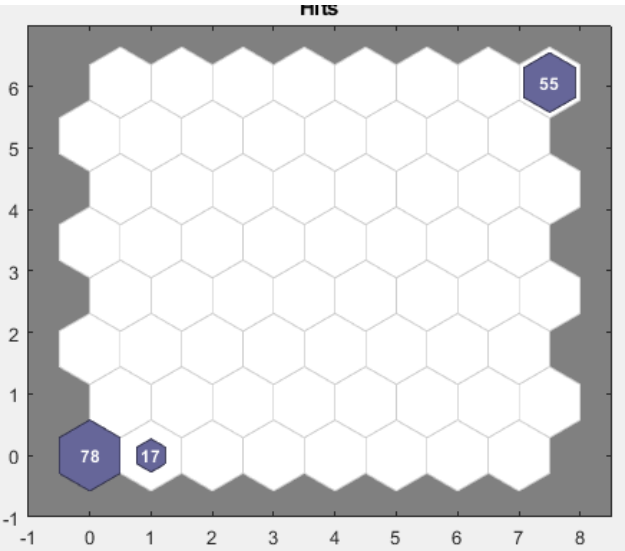
Sąsiadujące elementy przybierają barwy od czarnego do żółtego. Im kolor jest ciemniejszy, tym odległość między neuronami jest większa.

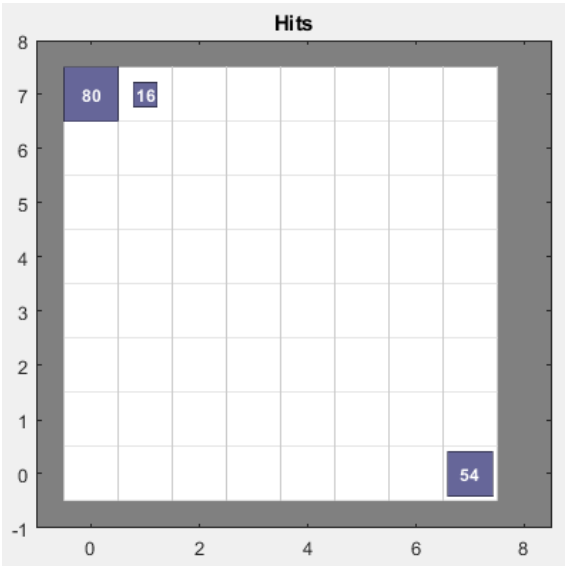
- Poniższy wykres przedstawia rozkład wag dla każdego z wejścia, czyli dla każdej z własności kwiatu. Im kolor jest ciemniejszy tym waga jest większa.



Ujemne połączenia są oznaczone kolorem niebieskim, połączenia czarne oznaczają zero, natomiast czerwone są to połączenia najsilniejsze i najbardziej pozytywne.

- Kolejne wykresy przedstawiają rozkład trafień neuronu, tzn. dla każdego neuronu przypisują pewną liczbę, która odpowiada za to ile razy dany neuron został zwycięzcą oraz jakie stworzył grupy.





Analiza i dyskusja błędów uczenia i testowania opracowanej sieci.

Z analizy powyższych wykresów oraz testowania różnych wariantów wynika wiele zależności. Najważniejszym chyba aspektem jest fakt, iż sieć prawidłowo wyodrębniła 3 grupy. Jednak zadanie nie zostało do końca prawidłowo wykonane. Jak widać z ostatniego wykresu grupy nie są jednakowo liczebne. Niektóre grupy zawierają zbyt wiele danych, przez to inne grupy są mniej liczebne niż powinny. Wynika to z faktu, iż zastosowana została reguła WTA, która nie bierze pod uwagę sąsiedztwa i pozostałe neurony maja stan 0, natomiast waga zwycięzcy jest zwiększana. Dla sieci samoorganizującej się można zastosować funkcję 'linkdist', która liczy odległość w zależności od ilości danych wejściowych oraz wykorzystać różne topologie np. prostokątną - 'gridtop', lub losową - 'randtop'. Analizując wyniki pod kątem topologii sieci widać, że niezależnie czy użyjemy topologii prostokątnej czy sześciokątnej zawsze otrzymujemy 3 grupy, natomiast rozkład neuronów w grupach różni sie, jednak nieznacząco. Cały czas mamy dwie grupy bardziej liczne i jedna mniej liczną. Natomiast używając topologii losowej nie da się przeprowadzić stosownych testów. Można zauważyć również, że w celu grupowania sieci należy dobrać odpowiednią ilość neuronów w sieci. Niestety użycie współczynnika uczenia nie było możliwe w tej wersji pakietu Matlab, dlatego nie udało się przeprowadzić testów oraz ich analizy. Jednak w algorytmie Kohohena współczynnik uczenia jest bardzo ważny do uzyskania pożądanego wyniku i na ogół jest malejącą funkcją wraz z iteracjami, kontrolując przy tym rozmiar wektora wag.

Wnioski

Sieć samoorganizująca się to jednokierunkowa której wyjściem jest mapa. W rozpatrywanym zadaniu sieć posiada wejście w postaci 4 własności dla 150 kwiatów. Sygnały przesyłane są od warstwy wejściowej do warstwy wyjściowej. Do implementacji został użyty algorytm Kohonena z regułą WTA, wykorzystujący zasadę, że wyznaczany jest neuron zwycięski najbardziej podobny do danych wejściowych i aktualizowane są tylko jego wagi, nie bierze pod uwagę sąsiedztwa. Wykorzystując funkcje wbudowane do programu Matlab udało mi się przeprowadzić stosowne testy oraz na ich podstawie przeprowadzić analizę z której wynika wiele wniosków. Analizując wyniki działania funkcji można stwierdzić, iż ustawiane parametry mają znaczący wpływ na działanie sieci. W kodzie programu można wprowadzać różne topologie sieci oraz różne miary odległości między neuronami co nieznacznie wpływa na wynik sieci. Niezależnie czy wybierzemy topologię prostokątną, czy sześciokątną sieci widać, ze powstają trzy grupy jednak mają ona różną liczebność. Wynika

stąd wniosek, iż wykorzystując regułę WTA nie udało się podzielić kwiaty na równe grupy ze względu na brak sąsiedztwa oraz to że aktualizowane są wagi jedynie neuronu zwycięskiego. Natomiast topologią losową nie da się rozwiązać zadanego problemu. Ważnym wnioskiem jest fakt, że ilość neuronów ma wpływ na wynik grupowania. Niestety nie udało się uzyskać wyników w zależności od zmiany wartości współczynnika uczenia.

Listing kodu wykonanego programu

```
close all; clear all; clc;
% wczytanie danych wejściowych
x=iris_dataset;
size(x);
% wyświetlanie danych w postaci punktów
plot( x(1, :), x(2, :), '.b');

 dimensions= [8 8];       % rozmiar mapy
 coverSteps=100;          % liczba kroków szkoleniowych
 initNeighbor=0;          % promień sąsiedztwa
 topologyFcn='hextop';    % sześciokątna topologia warstw
 distanceFcn='dist';      % euklidesowa funkcja odległości neuronowej
 % tworzenie samoorganizującej się mapy w parametrami opisanymi wyżej
 net=
selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn);

 net.trainParam.epochs = 300;  % maksymalna liczba epok
 net.trainFcn='trainbu';       % funkcja uczenia
% trening sieci
 [net,tr] = train(net,x);
 wy = net(x);
 classes = vec2ind(wy);         % skonwertowanie wektorów na indeksy

 % wyświetlanie pozycji wag
 plotsompos(net, x);
```