Kaja Kwiatkowska
Inżynieria Obliczeniowa
286114

Podstawy sztucznej inteligencji
Sprawozdanie ćwiczenie 5
Budowa i działanie sieci Kohonena dla WTA

Cel:

Celem ćwiczenia było poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTA (*Winner Takes All*) do odwzorowania istotnych cech kwiatów.

Wiedza:

Sieć Kohonena jest podstawowym przykładem typu sieci samoorganizujących się – *Self-Organizing Maps* (SOM). Daje to możliwości adaptacji do wcześniej nieznanych danych wejściowych. Nie są definiowane żadne wzorce, tylko musza krystalizować się w trakcie procesu uczenia wraz z normalnym funkcjonowaniem, działanie zbliżone do ludzkiego mózgu.
Uczenie sieci tego typu polega na podawaniu na wejściach sygnałów, a następnie wybraniu w drodze konkurencji zwycięskiego neuronu, który najlepiej odpowiada wektorowi wejściowemu.
Inicjalizacja wag sieci Kohonena jest losowa. Wektory wejściowe stanowią próbę uczącą.

Działanie sieci samoorganizujących:

- Konstrukcja – sygnały wejściowe opisujące charakterystyczne cechy zjawisk zachodzących w otoczeniu, aby możliwe było ich pogrupowanie przez sieć.
- Uczenie – mechanizm określający dla każdego neuronu podobieństwo wag do danego sygnału wejściowego oraz wyznaczenie zwycięzcy.
- Rozpoznawanie – zdolność do adaptacji wartości wag neuronu zwycięzcy i jego sąsiadów w zależności od siły, z jaką odpowiedział na dane wejściowe.

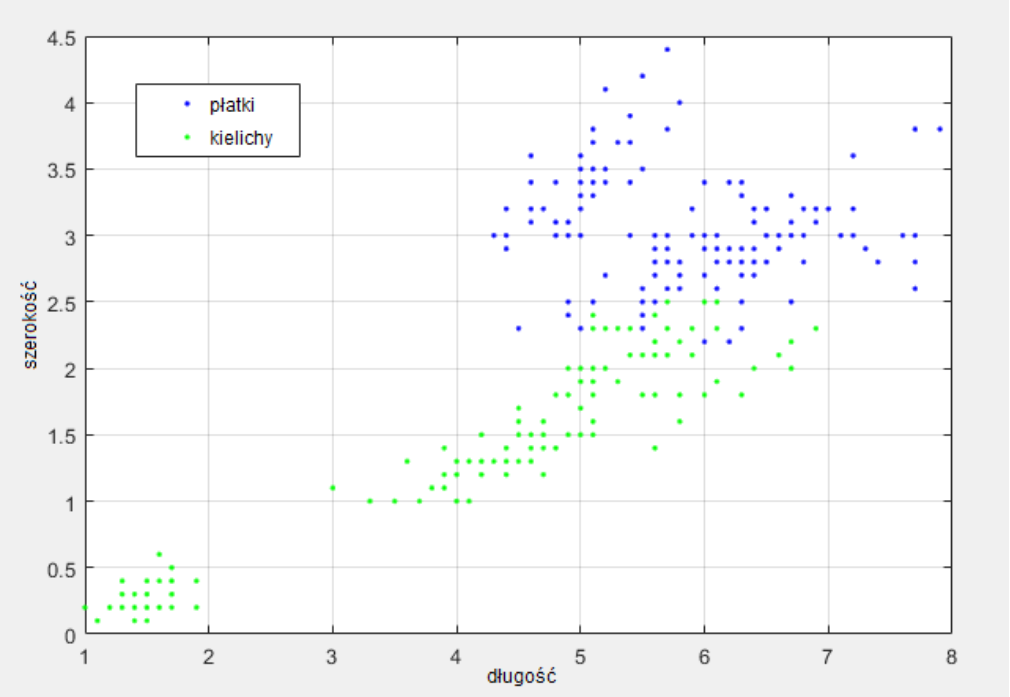
Reguła WTA (Winner Takes All – zwycięzca bierze wszystko) jest jedną z podstawowych strategii sieci typu SOM. Polega na modyfikacji wag neuronu najbardziej podobnego do elementu prezentowanego, tak aby jego wagi były najbardziej zbliżone do wektora wejściowego.

Algorytm uczenia:

- Wygenerowanie losowo znormalizowanych wektorów wag
- Losowanie wektora X oraz obliczanie dla niego aktywacji Y dla wszystkich neuronów
- Wyszukiwanie neuronu zwycięzcy
- Zmodyfikowanie wektora wag neuronu zwycięzcy
- Zatrzymanie algorytmu po odpowiednio dużej ilości iteracji

Zadania, które wykonałam w ramach ćwiczenia:

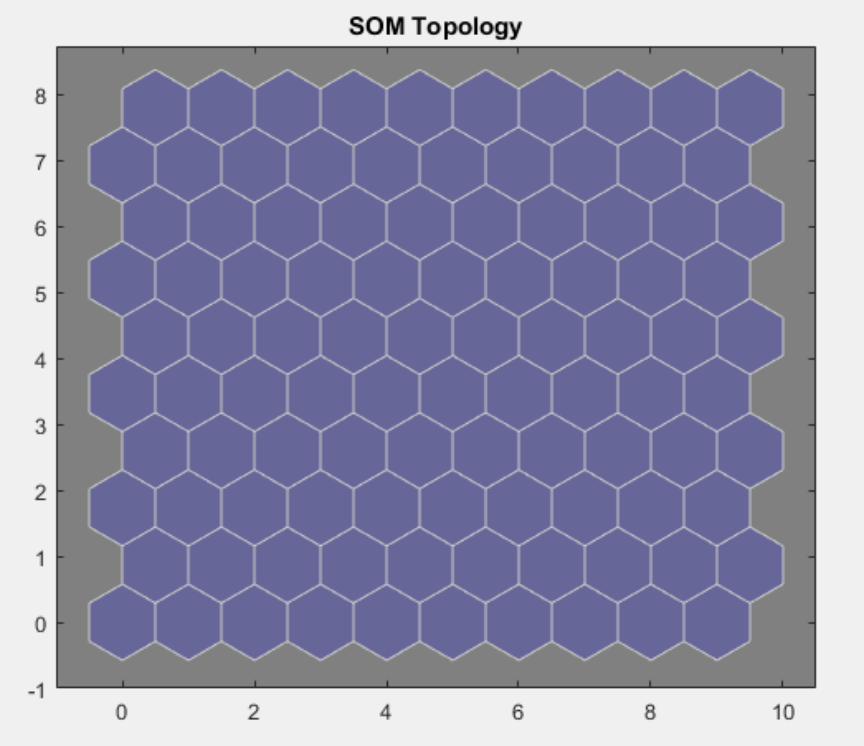
- Za pomocą programu *Matlab* oraz biblioteki *Neutral Network Toolbox* wygenerowałam dane uczące przy użyciu zaimplementowanego oprogramowania *iris_dataset*, zawierającego numeryczny zapis czterech cech kwiatów irysa – długość, szerokość płatków oraz kielichów kwiatów.



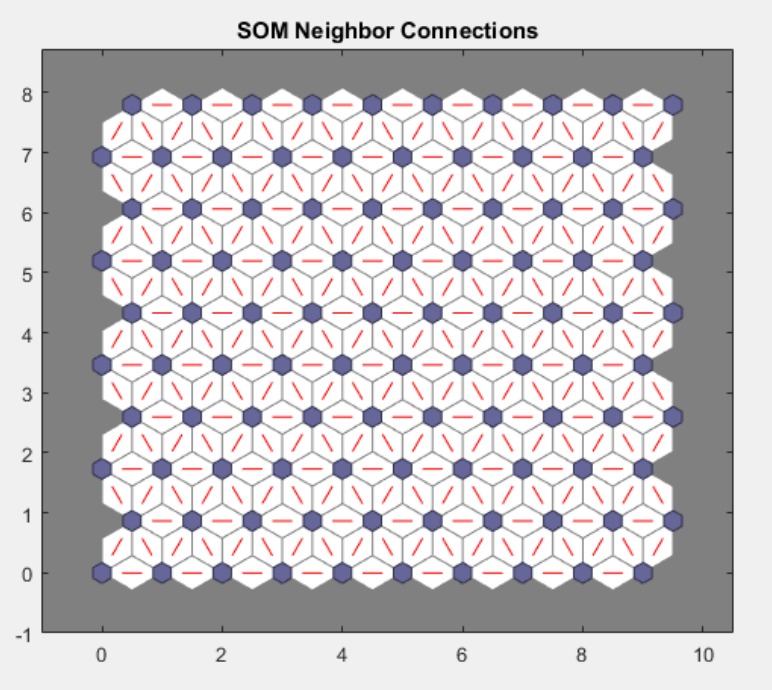
Wykres przedstawiający punkty płatków oraz kielichów dla wszystkich danych uczących

- Implementacja – użyte metody i zmienne:
  - Zdefiniowanie dane uczące, jako *WEJSCIE = irys_dataset*, będące wartościami zaimplementowanymi przez użyte oprogramowanie, zawierające numeryczny zapis czterech cech kwiatu irysa umieszczonych w tablicy 4x150.
  - Określenie rozmiaru tablicy przypisując dane wejściowe *size(WEJSCIE).*
  - Funkcja *hold on* do nadpisywania wykresów.
  - Funkcja *grid on* do wyświetlania głównych linii siatki dla bieżących osi lub wykresu.
  - Zmienna *dimensions* zawierająca wektor rzędów wymiarów.
  - Zmienna *coverSteps* z liczbą kroków szkoleniowych dla początkowego pokrycia przestrzeni wejściowej, domyślna wartość ustawiona na 100.
  - Zmienna *initNeighbor* z początkowym rozmiarem sąsiedztwa, domyślnie 3.
  - Funkcja topologii warstw *topologyFcn*, domyślnie *hextop*, czyli topologia sześciokątna.
  - Funkcja odległości neuronowej *distanseFcn*.
  - Utworzenie zmiennej, do której będzie przypisywana nowa sieć neuronowa za pomocą algorytmu Kohonena z wykorzystaniem wcześniej zdefiniowanych parametrów *net = selforgmap()*.

- Przeprowadziłam proces uczenia i przetestowałam działanie algorytmu. Poniżej zamieściłam otrzymane wyniki wraz z wnioskami.



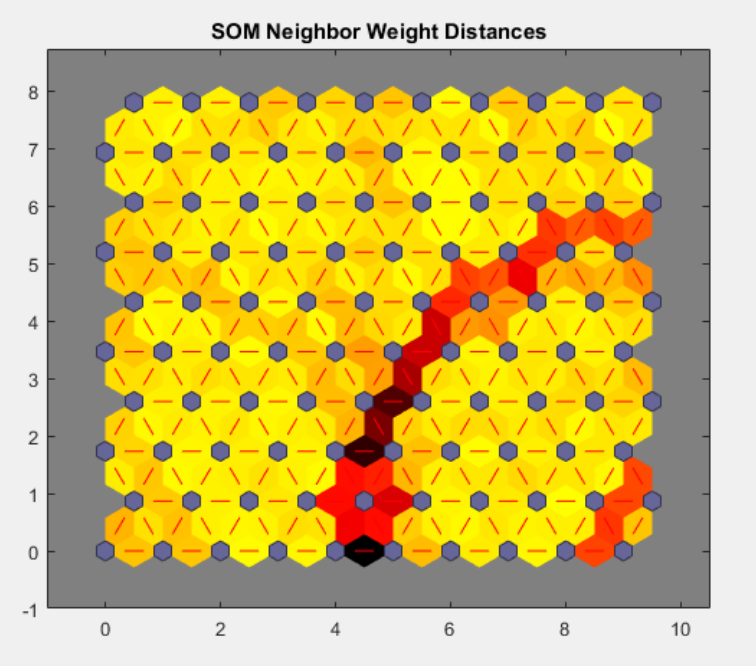
Topologia sieci Kohonena (SOM)
Dla maksymalnej liczby epok równej 500.



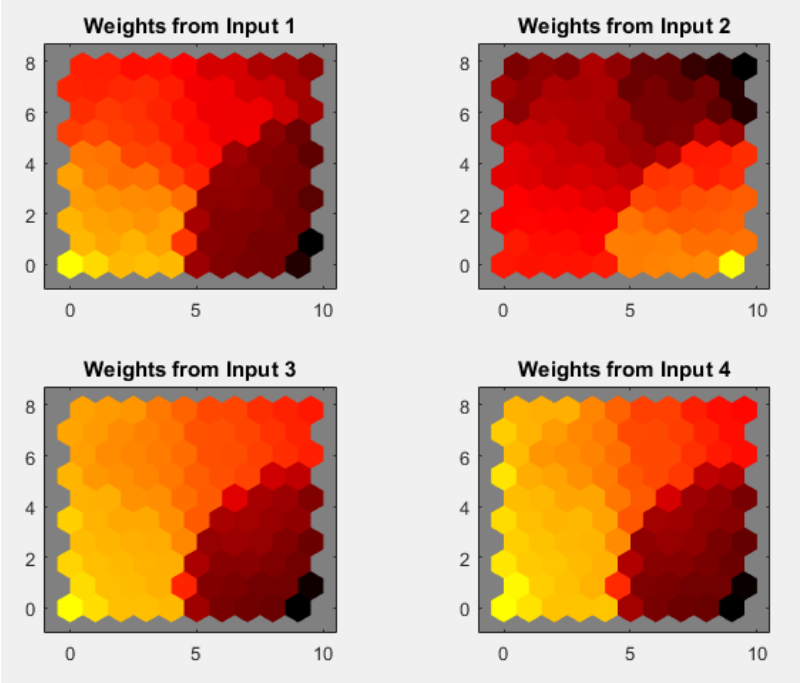
Połączenia pomiędzy poszczególnymi neuronami w sieci Kohonena (SOM).

Wniosek:

Dzięki heksagonalnej siatce neuronów możliwe jest tworzenie sieci o większej liczbie połączeń pomiędzy neuronami niż w przypadku sieci prostokątnej, co daje sieci więcej możliwości w doborze odpowiednich wag dla poszczególnych neuronów.



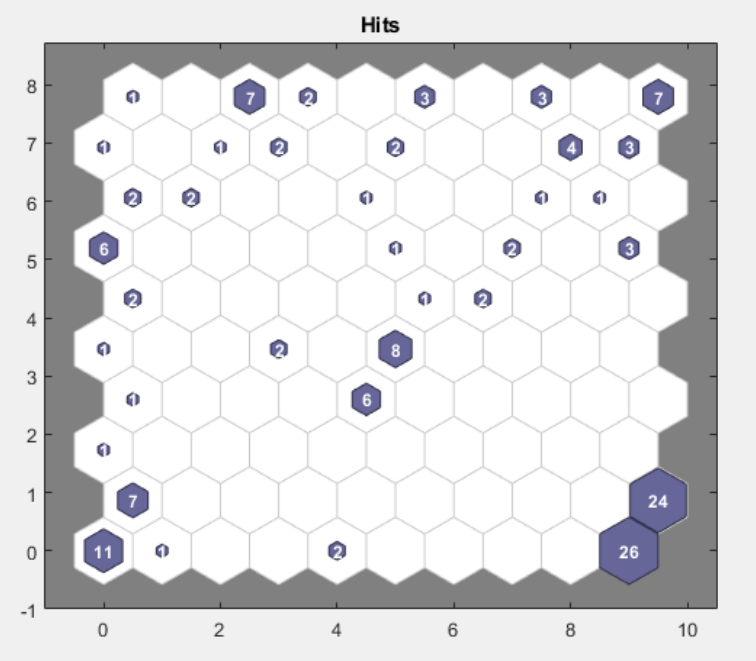
Odległości pomiędzy wagami poszczególnych neuronów.
Większe wagi oznaczone ciemniejszym kolorem.



Wyniki dla poszczególnych wejść (długość, szerokość płatka, długość, szerokość kielicha).

Wniosek:

Analizując rozkład barw można określić jak może wyglądać irys według uczonej sieci neuronowej. Im ciemniejszy kolor, tym bliższe jest położenie wyglądu irysa. Obrazowo widać, że kwiat ma zarówno długie jak i szerokie płatki oraz kielich, czyli odpowiada to rzeczywistemu wyglądowi irysa.

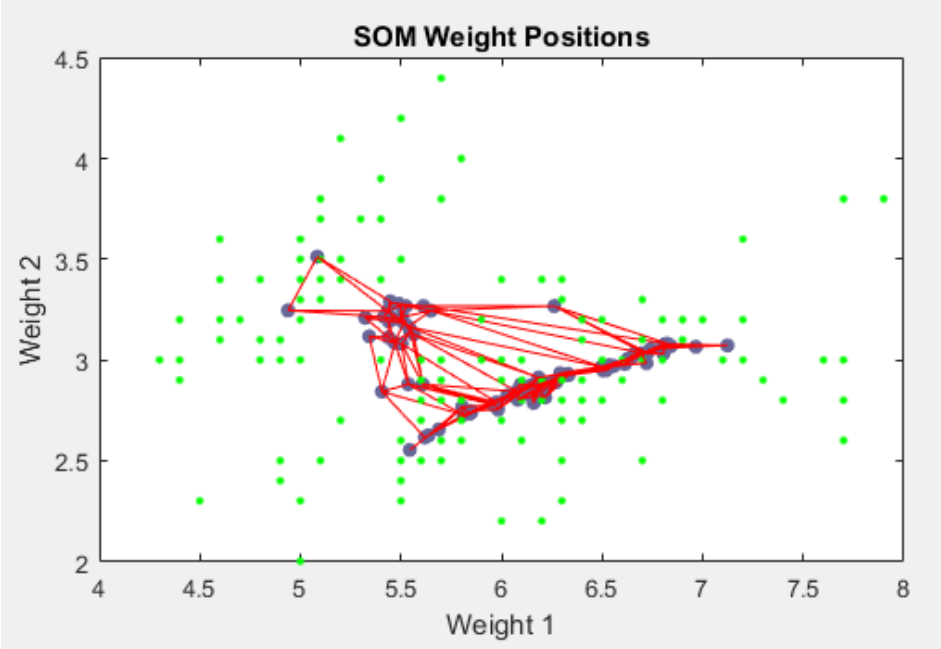


Rozkład sił neuronów.
Ilości zwycięstw danego neuronu w strategii WTA.

Wniosek:

Po analizie rozkładu sił neuronów widać, że sieć wykorzystała mechanizm WTA. Sieć zastosowała również normalizację, aby umożliwić zwycięstwa innym neuronom poza dwoma najsilniejszymi w prawym dolnym rogu.
Brak normalizacji w takim przypadku może doprowadzić do nierównomiernego rozkładu sił, czyli sytuacji, w której na niewielkim obszarze będzie znajdować się kilka silnych neuronów lub stref wpływów, a na pozostałych obszarach żadnego silnego neuronu.

Można również zauważyć, że wykres rozkładu sił jest związany z wykresem odległości pomiędzy wagami poszczególnych neuronów. Ciemniejsze kolory na wykresie odległości pokrywają się z silnymi neuronami.



Efekt końcowy nauki sieci.
Niebieskie punkty przedstawiają kwiaty, wyznaczone przez sieć jako bliskie irysowi.

Wniosek:

Punkty zdefiniowane jako irysy, posiadały parametry bliskie lub równe średnim wartościom. Średnie wartości interpretowane są przez sieć, jako typowe dla irysów.

Wnioski końcowe:

Algorytm zastosowany do uczenia sieci Kohonena jest algorytmem uczenia bez nauczyciela, mimo to sieć prawidłowo odwzorowała cechy typowe dla podanych danych, ze stosunkowo małym nakładem wydajności i czasu (przy około 500 epok wyniki są poprawne).

Ważne jest dobranie odpowiednich parametrów do manipulowania czasem nauki oraz dokładnością wyników. Im większa liczba neuronów tym dokładniejsze wyniki, ale dłuższy czas nauki.

Listing kodu programu wraz z komentarzami:

```
close all; clear all; clc;

WEJSCIE = iris_dataset;    %dane wejsciowe
size(WEJSCIE);             %rozmiar dla tablicy z danymi wejściowymi

%wyświetlenie wykresu
plot(WEJSCIE(1, :), WEJSCIE(2, :), 'b.', WEJSCIE(3, :), WEJSCIE(4, :), 'g.');

hold on;
grid on;


%parametry dla sieci Kohonena
dimensions = [10 10];     %wymiar wektora
coverSteps = 100;         %liczba kroków szkoleniowych dla początkowego pokrycia
przestrzeni wejściowej
initNeighbor = 0;         %początkowy rozmiar sąsiedztwa
topologyFcn = 'hextop';   %funkcja topologii warstw
distanceFcn = 'dist';     %funkcja odległości neuronowej


%tworzenie sieci Kohonena
net = selforgmap(dimensions, coverSteps, initNeighbor, topologyFcn, distanceFcn);
net.trainParam.epochs = 500;%ustalenie maksymalnej liczby epok treningowych
utworzonej sieci


%trening sieci
[net, tr] = train(net, WEJSCIE);
y = net(WEJSCIE);

grid on
```