Paulina Reichel, WIMiIP Inżynieria Obliczeniowa Grupa 2

Cel projektu:

Poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTA do odwzorowywania istotnych cech kwiatów.

Realizacja projektu:

- ✓ Przygotowanie danych uczących zawierających numeryczny opis cech kwiatów
- ✓ Przygotowanie sieci Kohonena i algorytmu uczenia opartego o regułę Winner Takes All (WTA) Uczenie sieci
- ✓ Testowanie sieci

Wstęp teoretyczny:

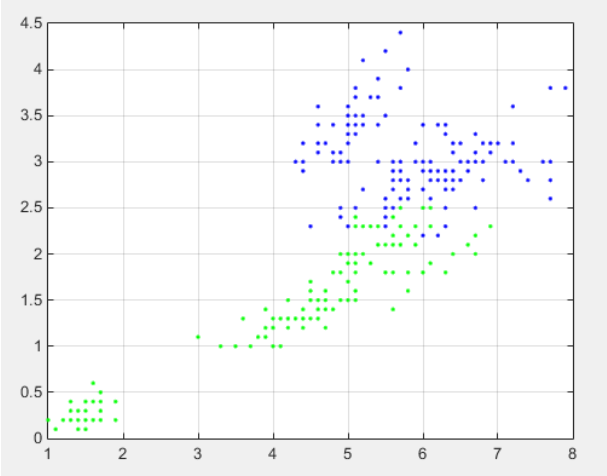
Sieć Kohonena – sieć neuronowa uczona w trybie bez nauczyciela w celu wytworzenia niskowymiarowej (przeważnie dwuwymiarowej) zdyskretyzowanej reprezentacji przestrzeni wejściowej. Sieć Kohonena wyróżnia się tym od innych sieci, że zachowuje odwzorowanie sąsiedztwa przestrzeni wejściowej. Wynikiem działania sieci jest klasyfikacja przestrzeni w sposób grupujący zarówno przypadki ze zbioru uczącego, jak i wszystkie inne wprowadzenia po procesie uczenia.

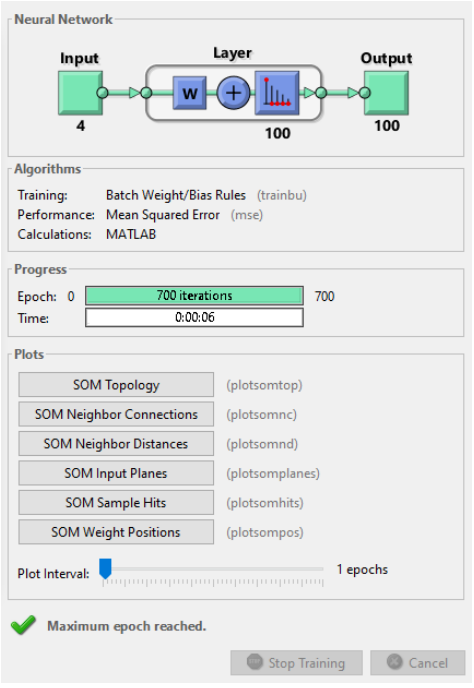
Etapy działania sieci:

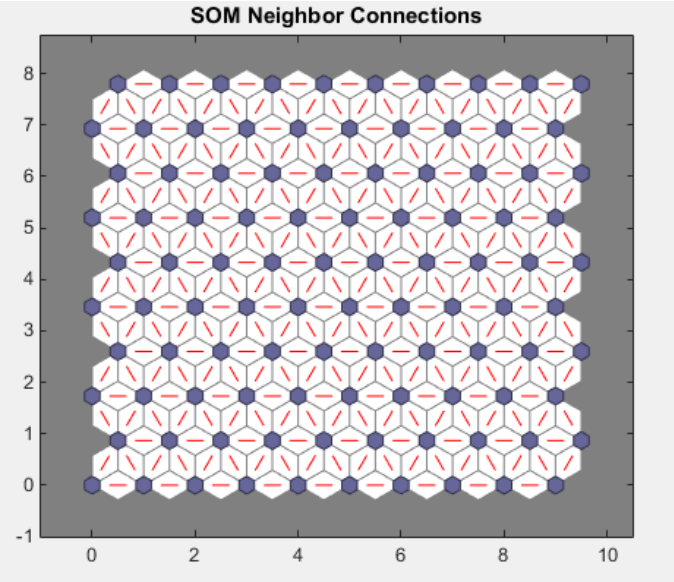
- ✓ Konstrukcja
- ✓ Uczenie
- ✓ Rozpoznawanie

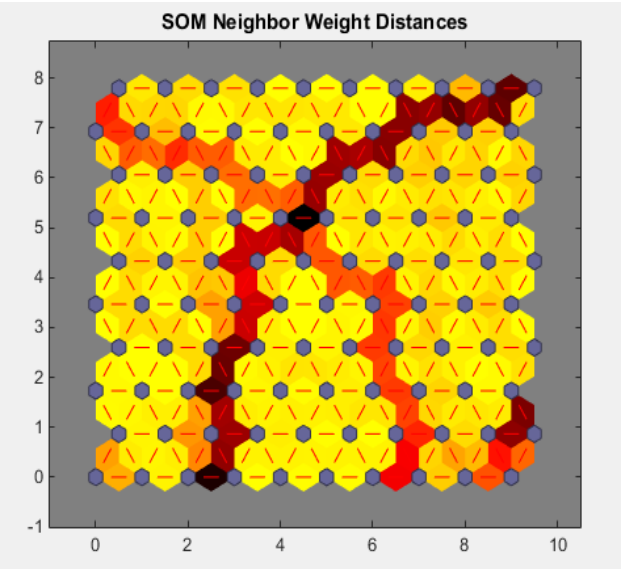
WTA („winner takes all") - reguła aktywacji neuronów w sieci neuronowej, która pozwala na aktywację tylko jednego neuronu w danej chwili. W konsekwencji w jednym momencie zostaje zmodyfikowana waga tylko jednego neuronu. Aby uniknąć dominacji jednego neuronu często stosuje się mechanizm zmęczenia, który polega na tym, że jeśli jakiś neuron wygrywa zbyt często, to przez pewien czas przestaje być brany pod uwagę podczas rywalizacji (jest odsuwany/usypiany w celu wylosowania innego neuronu).

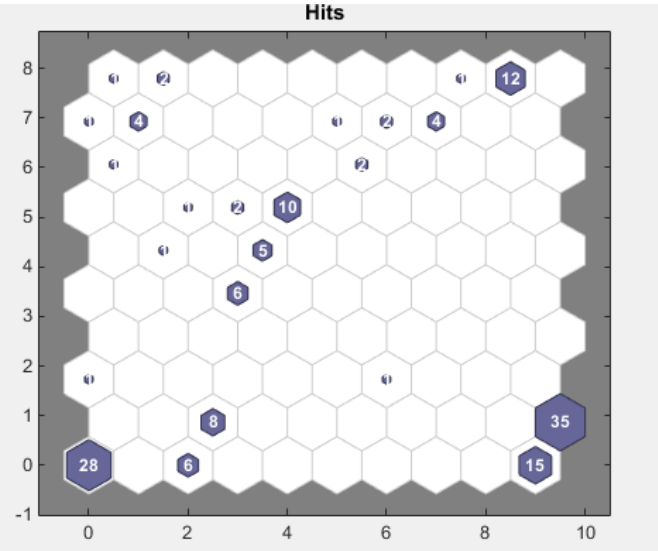
Zrzuty ekranu wykonania programu:

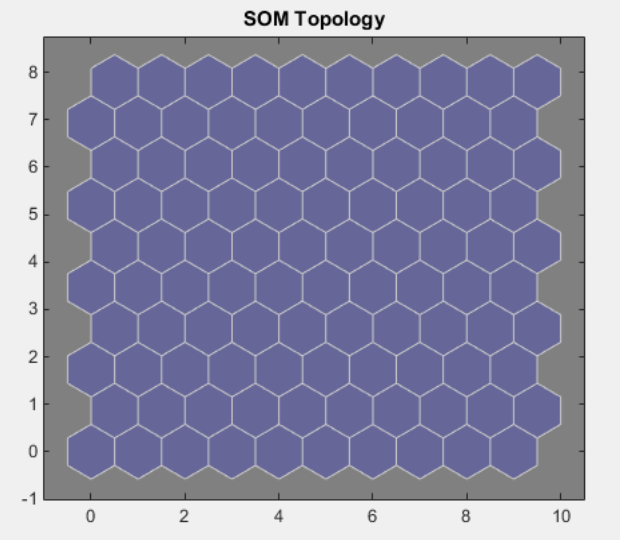


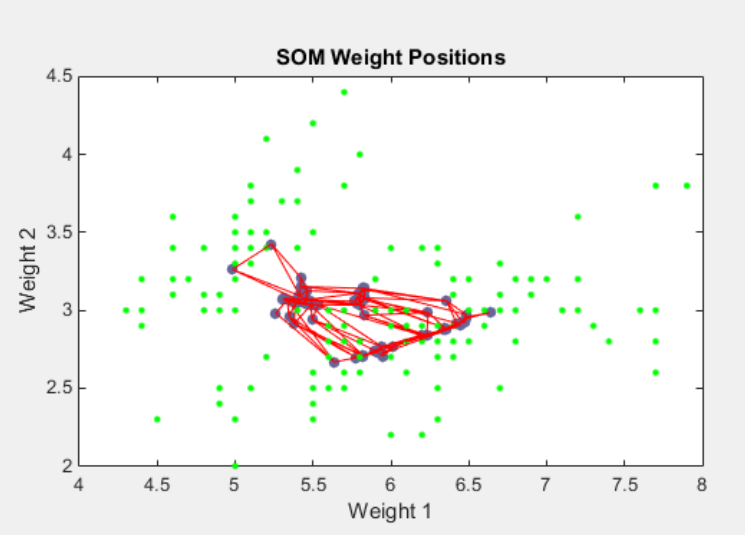






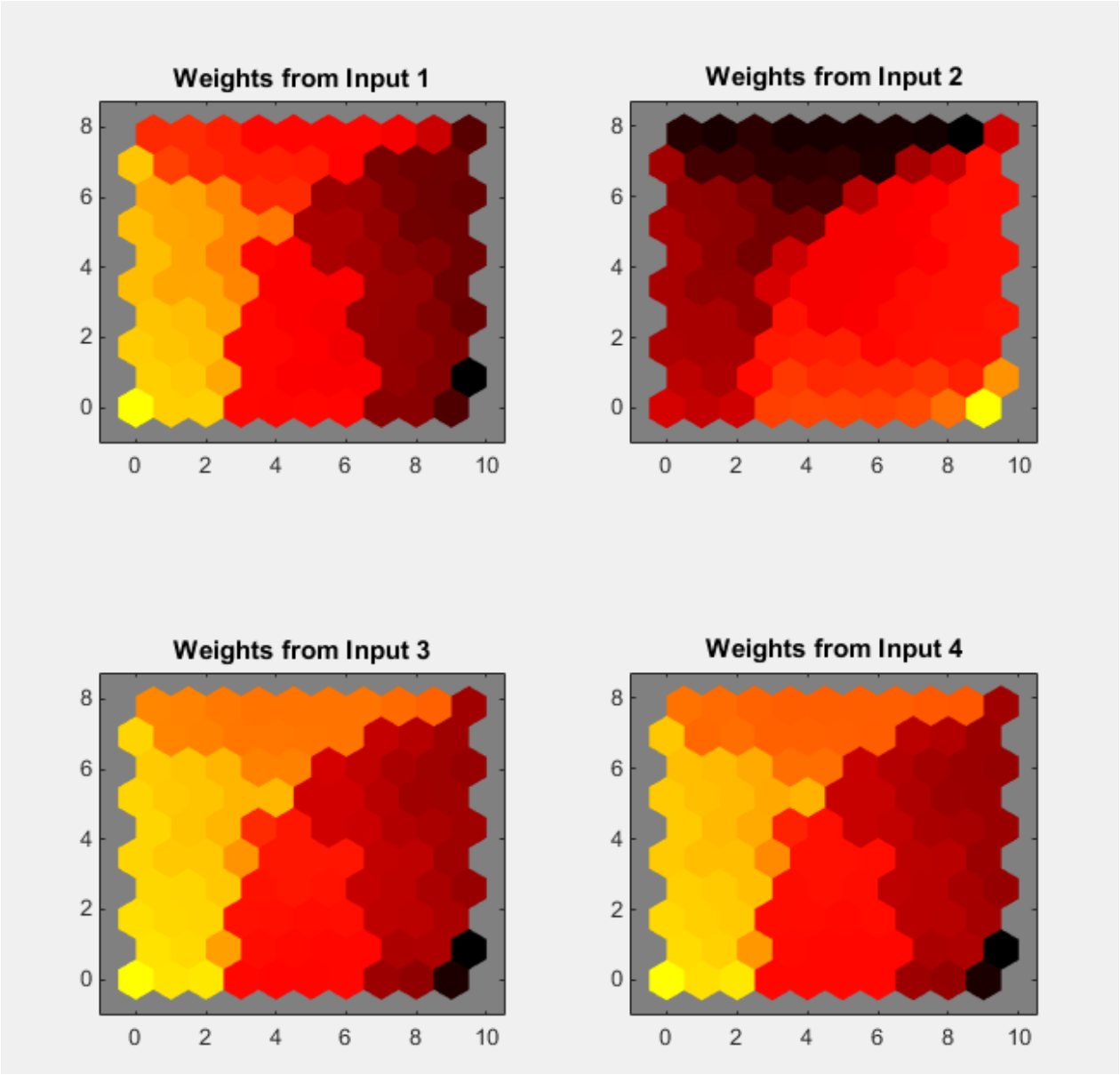






Program wykonany dla:

- ✓ Ilości epok =700
- ✓ Początkowego rozmiaru sąsiedztwa=0



Opis użytego algorytmu:

- ✓ Generowanie losowo znormalizowanych wektorów wag.
- ✓ Losowanie wektora X oraz obliczanie dla niego aktywację Y dla wszystkich neuronów
- ✓ Szukanie neuronu zwycięzcy
- ✓ Modyfikacja wektora wag neuronu zwycięzcy, a następnie jego normalizacja (sprawdzenie warunków WTA)
- ✓ Zatrzymanie algorytmu po odpowiednio dużej ilości iteracji

Objaśnienie przedstawionych wykresów oraz użytych funkcji:

- ✓ SOM Topology - każdy neuron reprezentowany jest przez sześciokąt. Siatka przedstawia 80 neuronów. W każdym wektorze wejściowym znajdują się cztery elementy, więc przestrzeń wejściowa jest czterowymiarowa. Wektory masy (centra skupień) mieszczą się w tej przestrzeni.
- ✓ SOM Neighbor Connections - mapa przedstawia powiązanie sąsiedzkie.
- ✓ SOM Neighbor Distance - niebieskie sześciokąty reprezentują neurony. Czerwone linie łączą sąsiednie neurony. Kolory w obszarach zawierających czerwone linie wskazują odległości między neuronami. Ciemniejsze kolory reprezentują większe odległości, a jaśniejsze kolory oznaczają mniejsze odległości. Pasmo ciemnych segmentów przechodzi od górnego w dół.
- ✓ SOM Sample Hits - przedstawia ile razy dany neuron zwyciężył podczas rywalizacji. Wektor wagowy związany z każdym neuronem przesuwa się, aby stać się centrum skupienia wektorów wejściowych. Neurony, które sąsiadują ze sobą w topologii, powinny również poruszać się blisko siebie w przestrzeni wejściowej, dlatego możliwe jest zwizualizowanie wielowymiarowej przestrzeni wejściowej w dwóch wymiarach topologii sieci.
- ✓ SOM Input Planes - przedstawia płaszczyznę ciężkości dla każdego elementu wektora wejściowego (w tym przypadku cztery). Są to wizualizacje wag, które łączą każde wejście z każdym z neuronów. (Ciemniejsze kolory oznaczają większe wagi.) Jeśli schematy połączeń dwóch wejść były bardzo podobne, można założyć, że dane wejściowe są wysoce skorelowane.
- ✓ WEJSCIE = iris_dataset – dane uczące, wartości implementowane przez oprogramowanie, zawiera numeryczny zapis czterech cech kwiatu irysa umieszczonych w tablicy 4x150,
- ✓ size(WEJSCIE) – określenie rozmiaru poprzez przypisanie danych wejściowych
- ✓ dimensions – wektor rzędów wymiarów,
- ✓ coverSteps – liczba kroków szkoleniowych dla początkowego pokrycia przestrzeni wejściowej,
- ✓ initNeighbor – początkowy rozmiar sąsiedztw,
- ✓ topologyFcn – funkcja topologii warstw,
- ✓ hextop – funkcja topologii sześciokątnej,
- ✓ distanseFcn – funkcja odległości neuronowej,
- ✓ dist – funkcja wagi odległości euklidesowej,
- ✓ net = selforgmap(…) – zmienna, do której będzie przypisywana nowo tworzona sieć neuronowa za pomocą algorytmu Kohonena z wykorzystaniem wcześniej zdefiniowanych parametrów sieci.

Wnioski:

- ✓ Heksagonalna siatka neuronów umożliwia utworzonej sieci stworzenie większej liczby połączeń pomiędzy neuronami, niż w przypadku sieci prostokątnej (w konsekwencji sieć ma więcej możliwości w doborze odpowiednich wag dla poszczególnych neuronów).
- ✓ Na podstawie ostatniego wykresu można zauważyć, że sieć neuronowa zdefiniowała te punkty jako irysy, które posiadały parametry bliskie lub równe średnim wartościom.
  Średnie wartości były interpretowane jako typowe wartości dla irysów.
- ✓ Wykres przedstawiający rozkład wag kolejnych wejść doskonale obrazuje wcześniej opisane zjawisko strefy wpływów. Analiza rozkładu barw pozwala określić jak może wyglądać irys według szkolonej sieci neuronowej. Dla przypomnienia im kolor jest ciemniejszy, tym jest bardziej bliski cechy irysa (według wyznaczonych danych przez sieć można stwierdzić, że zdefiniowany kwiat posiada zarówno długie i szerokie płatki jak i kielich, co pokrywa się z rzeczywistym wyglądem kwiatu).
- ✓ Na podstawie wykresu pokazującego rozkład sił neuronów można zauważyć, że sieć korzystała z mechanizmu WTA (sieć nie modyfikowała wag tylko jednego neuronu, wynika to z zastosowania normalizacji, gdyby nie ona zapewne dwa najmocniejsze neurony w prawym dolnym rogu zwyciężałyby za każdym razem).
- ✓ Zestawiając wyniki można zauważyć, że wykres przedstawiający rozkład sił neuronów jest powiązany z wykresem pokazującym odległości pomiędzy wagami poszczególnych neuronów. Ciemniejsze kolory na wykresie drugim pokrywają się z występowaniem neuronów zwyciężających w wykresie pierwszym.
- ✓ Pomimo treningu bez nadzoru sieć Kohonena (wraz z mechanizmem WTA) prawidłowo odwzorowała cechy typowe dla wybranego kwiatu, przy stosunkowo niewielkiej liczbie narzuconych epok treningowych (w moim przypadku już przy zmniejszeniu liczby do 500 epok wyniki stawały się coraz bardziej klarowne, przy odpowiednio krótkim czasie działania programu - więcej epok zapewniało na pewno większą dokładność, jednak zdecydowanie wydłużało czas nauki).
- ✓ Ważnym czynnikiem przy tworzeniu takiej sieci jest dobór odpowiedniej liczby neuronów - dla małej liczby rośnie ryzyko wystąpienia błędu, natomiast zbyt duża liczba neuronów znacznie wydłuży czas potrzebny na naukę sieci.

Listing programu:

```
WEJSCIE = iris_dataset; %dane wejsciowe
size(WEJSCIE); %okreslenie rozmiaru tablicy
plot(WEJSCIE(1, :), WEJSCIE(2, :), 'b.', WEJSCIE(3, :),WEJSCIE(4, :), 'g.');
hold on; grid on; %do wyświetlenia (trzymanie wykresu i siatka)
% PARAMETRY 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 = 700;% ustalenie maksymalnej liczby epok treningowych utworzonej sieci
% TRENING SIECI
[net, tr] = train(net, WEJSCIE);
y = net(WEJSCIE);
grid on
```