## Paulina Reichel, WIMiIP Inżynieria Obliczeniowa Grupa 2

### Cel projektu:

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

### Realizacja projektu:

- ✓ Wygenerowanie danych uczących i testujących, zawierających 20 dużych liter dowolnie wybranego alfabetu w postaci dwuwymiarowej tablicy.
- ✓ Przygotowanie sieci Kohonena i algorytmu uczenia opartego o regułę Winner Takes Most (WTM). Uczenie sieci dla różnych współczynników uczenia.
- ✓ 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**:

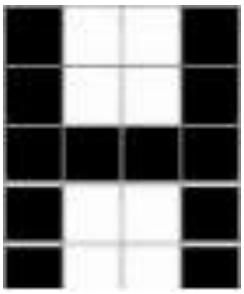
- ✓ Konstrukcja
- ✓ Uczenie
- ✓ Rozpoznawanie

**WTM („winner takes most")** - reguła aktywacji neuronów w sieci neuronowej, która jest oparta na zasadzie działania WTA z tą różnicą, że oprócz zwycięzcy wagi modyfikują również neurony z jego sąsiedztwa, przy czym im dalsza odległość od zwycięzcy, tym mniejsza jest zmiana wartości wag neuronu. Metoda WTA jest metodą słabo zbieżną - w szczególności dla dużej liczby neuronów. Sąsiedztwo jest pojęciem umownym - można definiować sąsiadów bliższych i dalszych, sąsiedztwo nie oznacza również, że neurony muszą być bezpośrednio połączone ze zwycięzcą.

### 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 WTM)
- ✓ Zatrzymanie algorytmu po odpowiednio dużej ilości iteracji

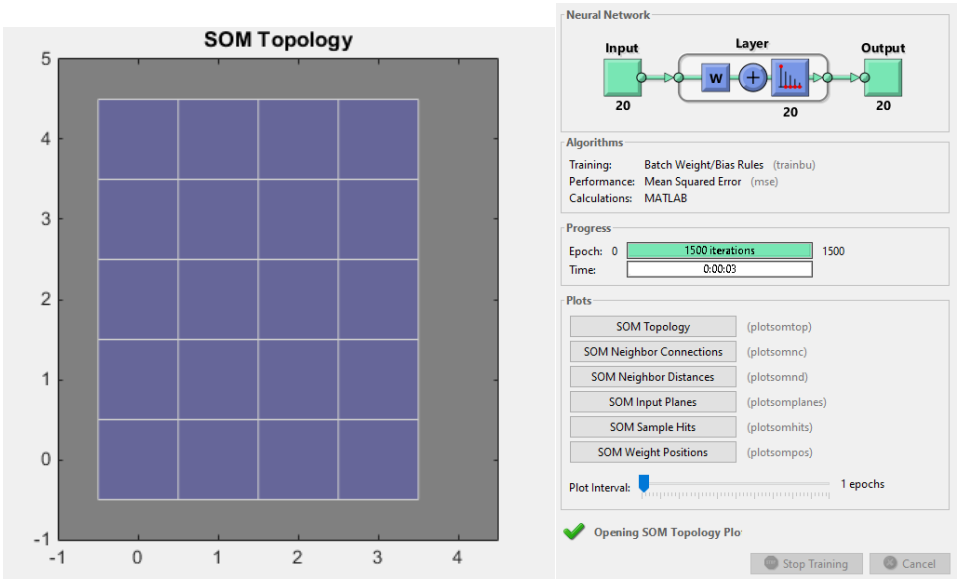
W programie, litery zostały przedstawione w postaci tablicy. Tablica o wymiarze 4x5, składa się z pól czarnych lub białych. Czarne pole, interpretowane jako 1, zawiera fragment litery, natomiast białe czyli 0 – nie zawiera litery.

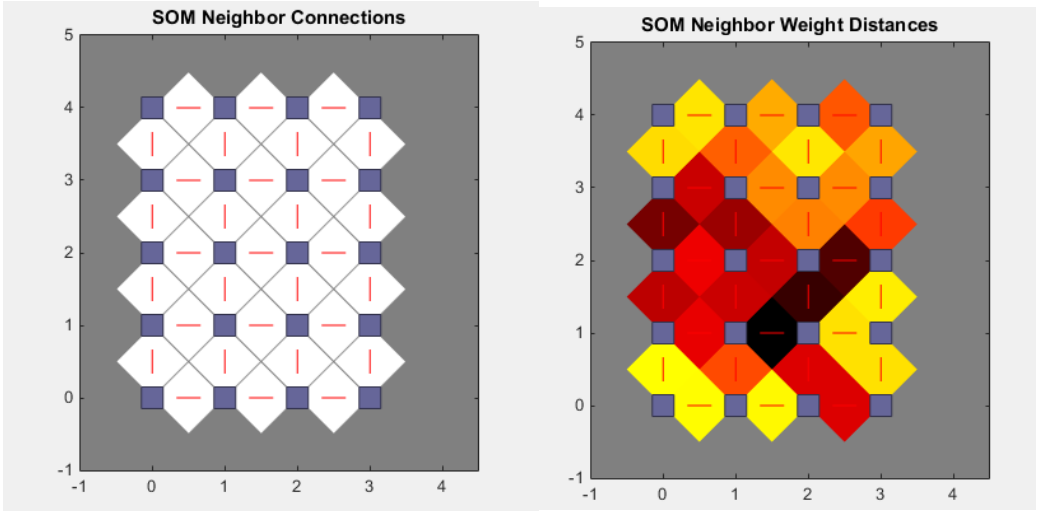


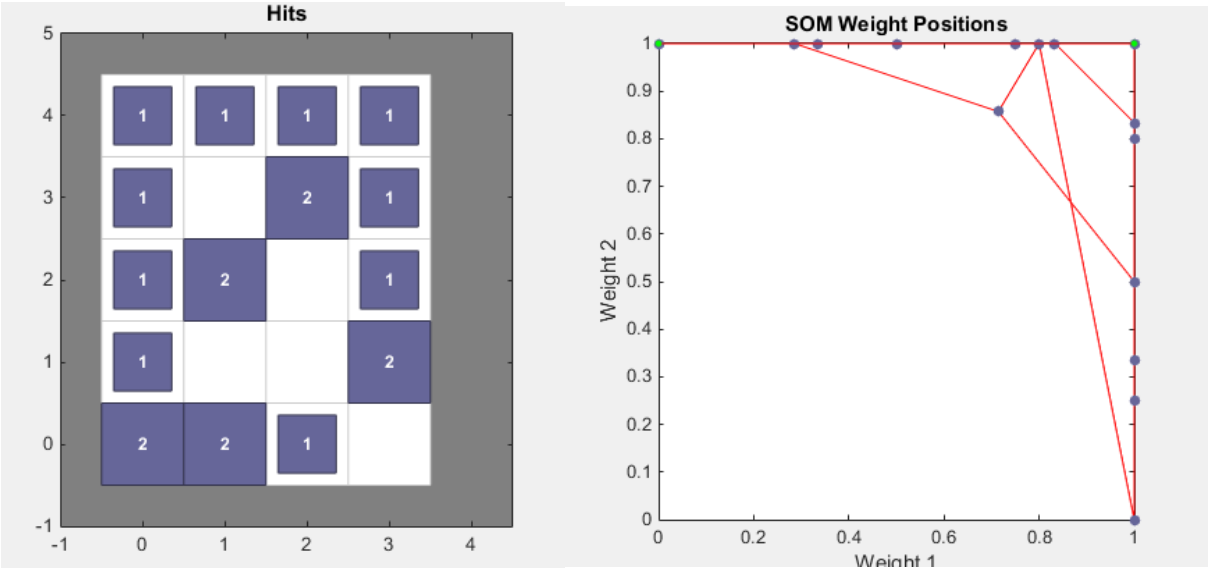
**1** 0 0 **1**
**1** 0 0 **1**
**1** **1** **1** **1**
**1** 0 0 **1**
**1** 0 0 **1**

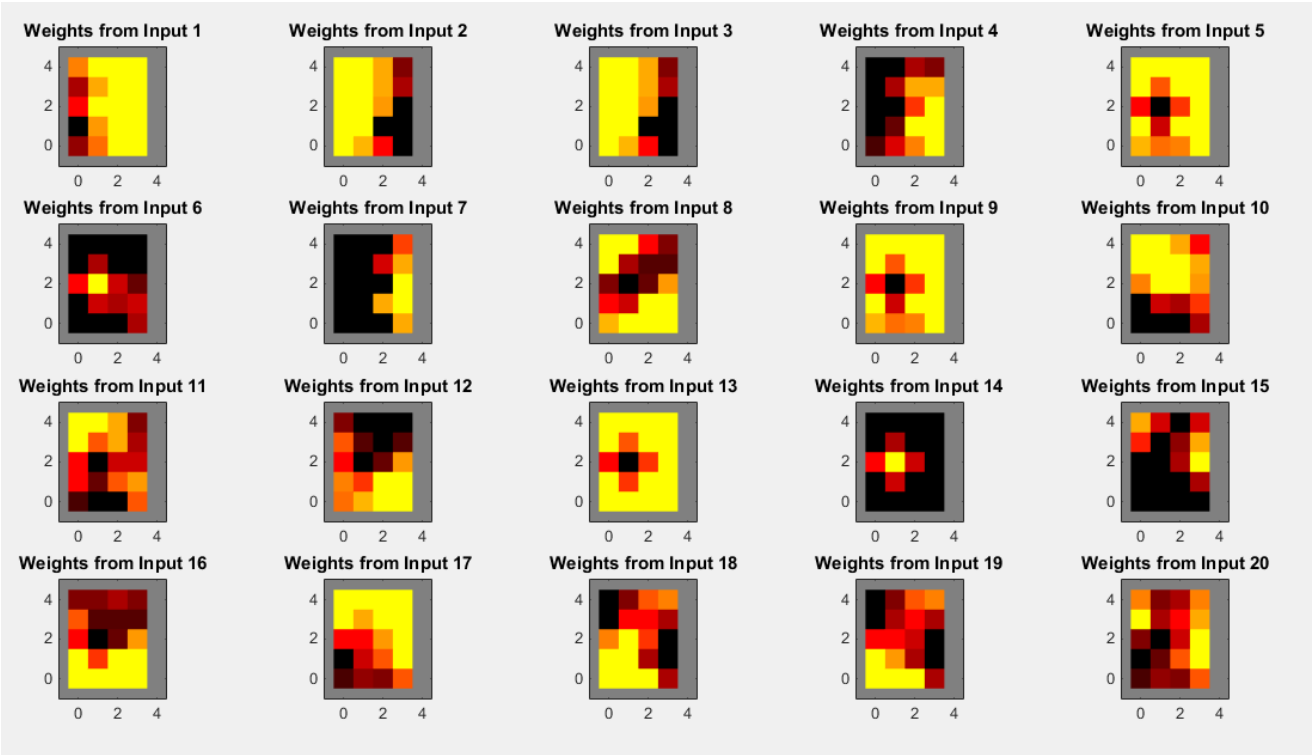
H = [1 0 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1];

**Zrzuty ekranu wykonania programu:**




Hits
SOM Weight Positions
Weight 2
Weight 1


Weights from Input 1
Weights from Input 2
Weights from Input 3
Weights from Input 4
Weights from Input 5
Weights from Input 6
Weights from Input 7
Weights from Input 8
Weights from Input 9
Weights from Input 10
Weights from Input 11
Weights from Input 12
Weights from Input 13
Weights from Input 14
Weights from Input 15
Weights from Input 16
Weights from Input 17
Weights from Input 18
Weights from Input 19
Weights from Input 20

**Objaśnienie przedstawionych wykresów oraz użytych funkcji:**

- ✓ SOM Topology - w programie została użyta prostokątna siatka neuronów odpowiadająca rozmiarowi liter 4x5.
- ✓ SOM Neighbor Connections - mapa przedstawia powiązanie sąsiedzkie.
- ✓ SOM Neighbor Distance - Kolory w przedstawionych obszarach wskazują na odległości między neuronami. Ciemniejsze kolory reprezentują większe wartości wag, a jaśniejsze – mniejsze.
- ✓ SOM Sample Hits - przedstawia rozkład sił neuronów (ilość zwycięstw neuronów w konkurencji do WTM).
- ✓ SOM Input Planes - przedstawia kolejne wejścia. Podobnie jak wcześniej - im ciemniejszy kolor, tym wyższa waga.
- ✓ SOM Weight Position - przedstawia efekt końcowy nauki SOM (niebieskie kropki to neurony, natomiast linie to połączenia między nimi).
- ✓ WEJSCIE - dane uczące, wartości implementowane przez oprogramowanie, litery zapisane binarnie oraz kolumnowo
- ✓ 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.
- ✓ **net.trainParam.epochs** – ustalenie maksymalnej liczby epok treningowych utworzonej sieci,
- ✓ **vec2ind** – konwertowanie wektorów uczonej sieci na indeksy.

**Wnioski:**

- ✓ Prostokątna siatka neuronów umożliwia utworzonej sieci stworzenie bezpośrednich powiązań pomiędzy najbliższymi neuronami.
- ✓ Algorytm Winner Takes Most pokazywał prawie równomierne rozłożenie zwycięstw na całej wygenerowanej wcześniej sieci. W poprzednim scenariuszu metoda Winner Takes All koncentrowała wygrane neurony głównie w jednym miejscu sieci heksagonalnej. Rozkład zwycięstw metody WTM zwiększała poprawność działania sieci ze względu na równomierny rozkład zwycięstw.
- ✓ Wagi poszczególnych neuronów są rozłożone w zależności od ilości neuronów w sieci. Zwiększanie liczby neuronów wpływało na czas obliczeń znacznie go wydłużając.
- ✓ Zwiększanie sąsiedztwa powodowało błędy w działaniu sieci neuronowej. Wielkość sąsiedztwa jest uzależniona od wielkości sieci - jeśli oba te parametry rosną jednocześnie wtedy działanie programu jest poprawne.
- ✓ Na podstawie wykresu pokazującego rozkład sił neuronów można zauważyć, że sieć korzystała z mechanizmu WTM (wykres pokazuje prawie równomierny rozkład zwycięstw neuronów - z pewnością otrzymane wyniki mniej koncentrują się na jednym punkcie sieci - jak w przypadku WTA).
- ✓ Pomimo treningu bez nadzoru sieć Kohonena (wraz z mechanizmem WTM) prawidłowo odwzorowała cechy typowe dla wybranej litery, przy stosunkowo niewielkiej liczbie

narzuconych epok treningowych (w moim przypadku już przy zmniejszeniu liczby do 2000 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).

- ✓ Bardzo 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.
- ✓ Metoda WTM daje lepsze wyniki w przypadku sieci z dużą liczbą neuronów niż metoda WTA.

**Listing programu:**

```
%kolumnowa reprezentacja binarna pierwszych 20 duzych liter alfabetu
dla tablicy 4x5
%dane wejsciowe:
           %A B C D E F G H I J K L M N O P R S T U
WEJSCIE = [0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1;
           1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0;
           1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0;
           0 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1;
           1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1;
           0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0;
           0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0;
           1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1;
           %A B C D E F G H I J K L M N O P R S T U
           1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1;
           1 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0;
           1 1 0 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 0;
           1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 0 0 1;
           1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1;
           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 1 0 0 1 0 0 1 0 0 0;
           1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 1;
           1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 0 0;
           0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 1;
           0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1;
           1 0 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0];
% PARAMETRY SIECI KOHONENA
dimensions = [4 5]; %wymiar wektora
coverSteps = 100; %liczba kroków szkoleniowych dla początkowego
pokrycia przestrzeni wejściowej
initNeighbor = 1; %początkowy rozmiar sąsiedztwa
topologyFcn = 'gridtop'; %funkcja topologii warstw
distanceFcn = 'dist'; %funkcja odległości neuronowej
% TWORZENIE SIECI KOHONENA (SOM)
net = selforgmap(dimensions, coverSteps, initNeighbor,topologyFcn,
distanceFcn);
net.trainParam.epochs = 1500; %ustalenie maksymalnej liczby epok
treningowych utworzonej sieci
% TRENING SIECI
[net, tr] = train(net, WEJSCIE);%uczenie sieci
y = net(WEJSCIE); %przypisanie sieci do wyjścia Y
classes = vec2ind(y); %konwertowanie wektorów uczonej sieci na
indeksy
```