Michalina Nikiel
WIMiIP, Inżynieria Obliczeniowa
rok 3, grupa laboratoryjna nr 2

Podstawy sztucznej inteligencji – projekt nr 6 – sprawozdanie

Budowa i działanie sieci Kohonena dla WTM.

Cel ćwiczenia:

Celem ćwiczenia jest poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTM do odwzorowywania istotnych cech liter alfabetu.

Zrealizowane kroki:

- 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.

Sprawozdanie zawiera kolejno:

- Zagadnienia teoretyczne: opis budowy użytej sieci i odpowiednich algorytmów.
- Zestawienie otrzymanych wyników wraz z zrzutami ekranu i przebiegiem ćwiczenia.
- Wnioski wraz z analiza i dyskusją błędów uczenia opracowanej sieci.
- Listing całego kodu programu.

Zagadnienia teoretyczne:

Działanie sieci Kohonena:
Sieci Kohonena są szczególnym przypadkiem algorytmu realizującego uczenie się bez nadzoru. Ich głównym zadaniem jest organizacja wielowymiarowej informacji (np. obiektów opisanych 50 parametrami w taki sposób, żeby można ją było prezentować i analizować w przestrzeni o znacznie mniejszej liczbie wymiarów, czyli mapie (np. na dwuwymiarowym ekranie)).
Warunek: rzuty "podobnych" danych wejściowych powinny być bliskie również na mapie. Sieci Kohonena znane są też pod nazwami Self-Organizing Maps (SOM) lub Competitive Filters.

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.

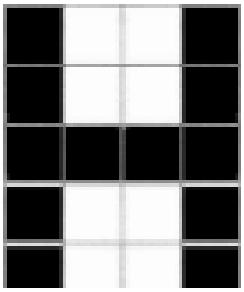
Inicjalizacja wag sieci Kohonena jest losowa. Wektory wejściowe stanowią próbę uczącą, podobnie jak w przypadku zwykłych sieci rozpatrywaną w pętli podczas budowy mapy. Wykorzystanie utworzonej w ten sposób mapy polega na tym, że zbiór obiektów umieszczamy na wejściu sieci i obserwujemy, które węzły sieci się uaktywniają. Obiekty podobne powinny trafiać w podobne miejsca mapy.

Mechanizm WTM (Winner Takes Most):
Jest to 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.
Podobnie jak w metodzie WTA, aby uniknąć tego, że jeden neuron będzie zawsze wygrywał stosuje się mechanizm zmęczenia, który w przypadku, gdy jeden neuron wygrywa zbyt często, to na pewien czas przestaje on być brany pod uwagę w rywalizacji.

Użyty algorytm uczenia sieci:
1) Generowanie losowo znormalizowanych wektorów wag.
2) Losowanie wektora X oraz obliczanie dla niego aktywację Y dla wszystkich neuronów.
3) Szukanie neuronu zwycięzcy.
4) Modyfikacja wektora wag neuronu zwycięzcy oraz sąsiedztwa, a następnie jego normalizacja (sprawdzenie warunków WTM).
5) Zatrzymanie algorytmu po odpowiednio dużej ilości iteracji.

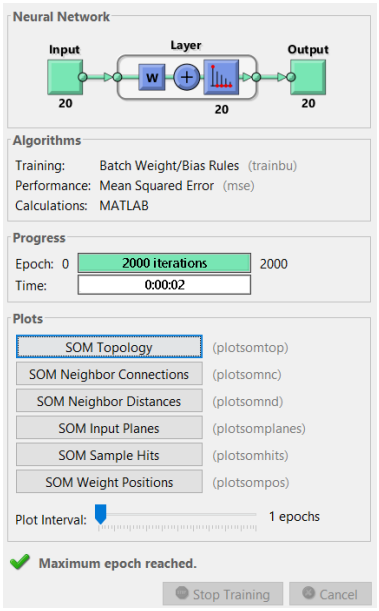
Zestawienie otrzymanych wyników:

Umieszczanie liter w tablicy polegało na zinterpretowaniu danego pola przez sieć. Tablica składa się z 20 pól, które odpowiednio są czarne - 1, lub białe - 0. Czarne pole oznaczało, że w danym miejscu występował fragment litery, np. dla litery H wygląda to następująco:

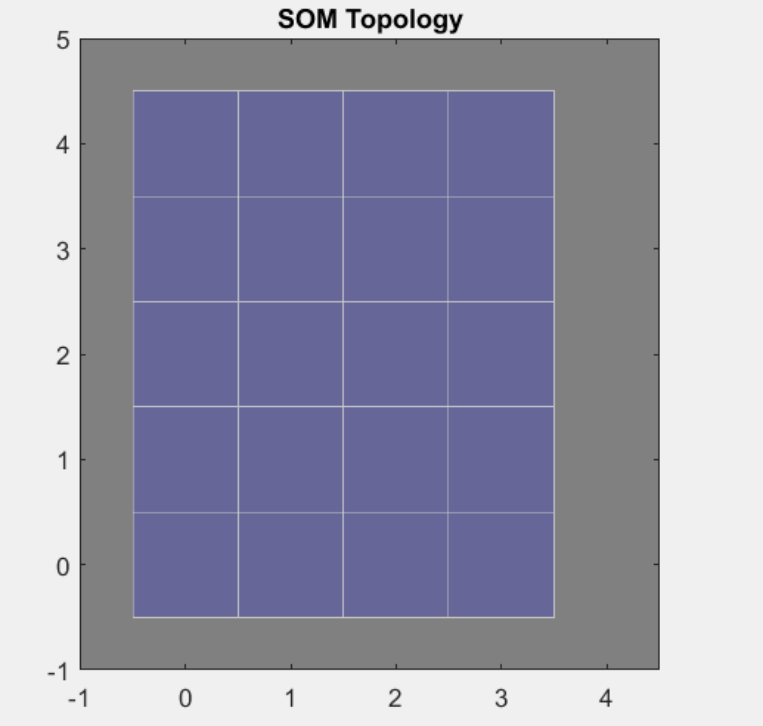


1 0 0 1
1 0 0 1
1 1 1 1
1 0 0 1
1 0 0 1

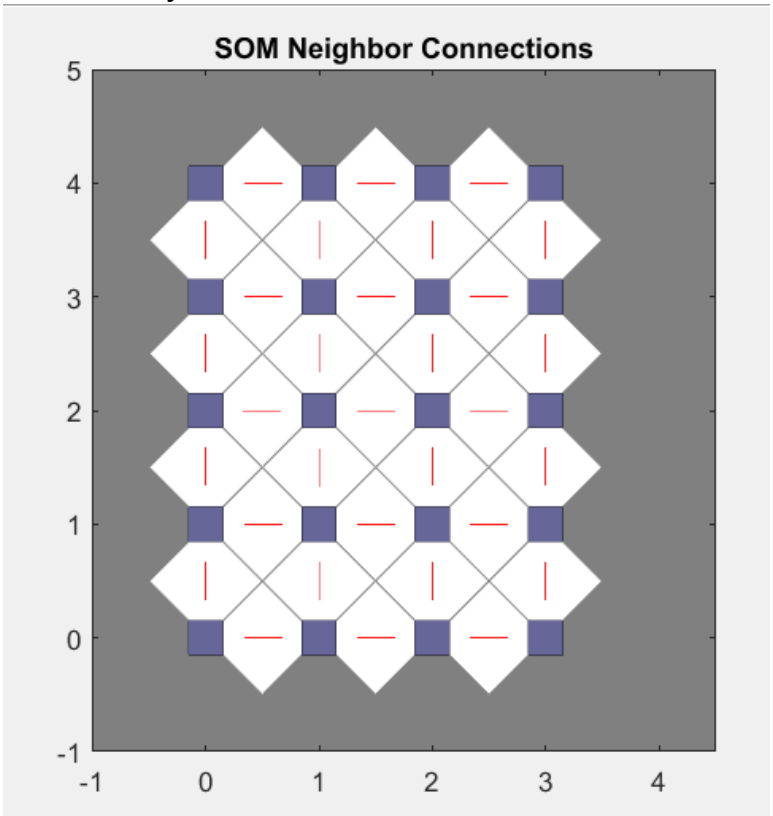
Dla litery H wygenerowana interpretacja jest zapisywana w postaci ciągu zer i jedynek:
H = [1 0 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1];
Tak stworzone litery zostały zapisane kolumnowo przy zmiennej „**WEJŚCIE**”, są to dane uczące dla sieci neuronowej.



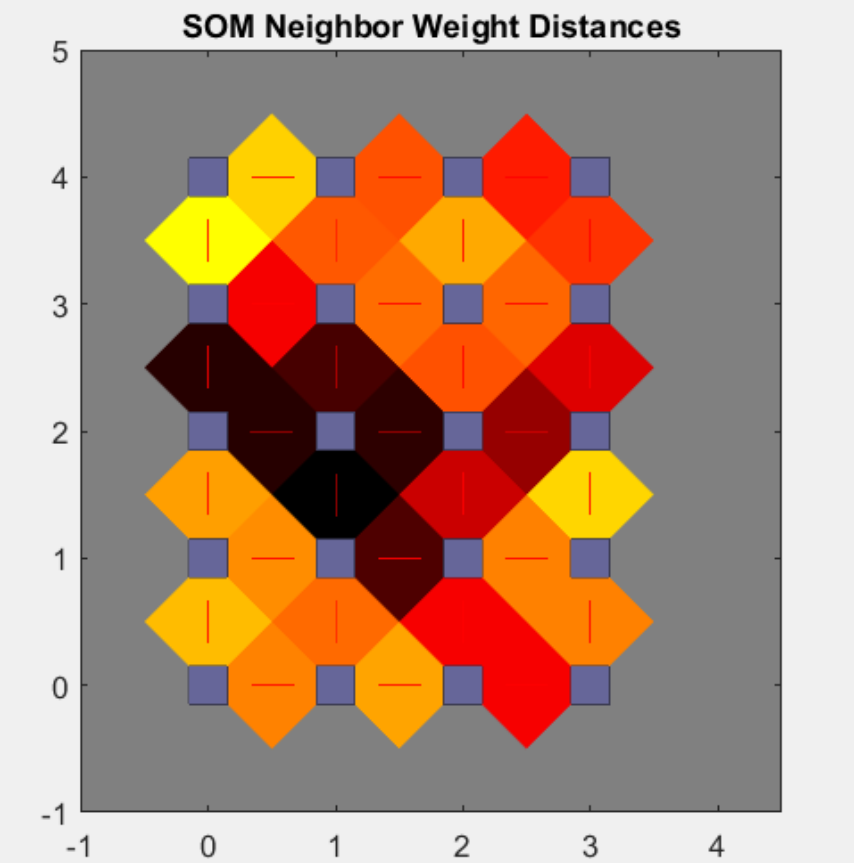
SOM Topology:
Zrzut ekranu przedstawia topologię sieci Kohonena dla wcześniej określonej maksymalnej liczby epok treningowych równej 2000 (w programie została użyta prostokątna siatka neuronów odpowiadająca rozmiarowi liter 4x5).



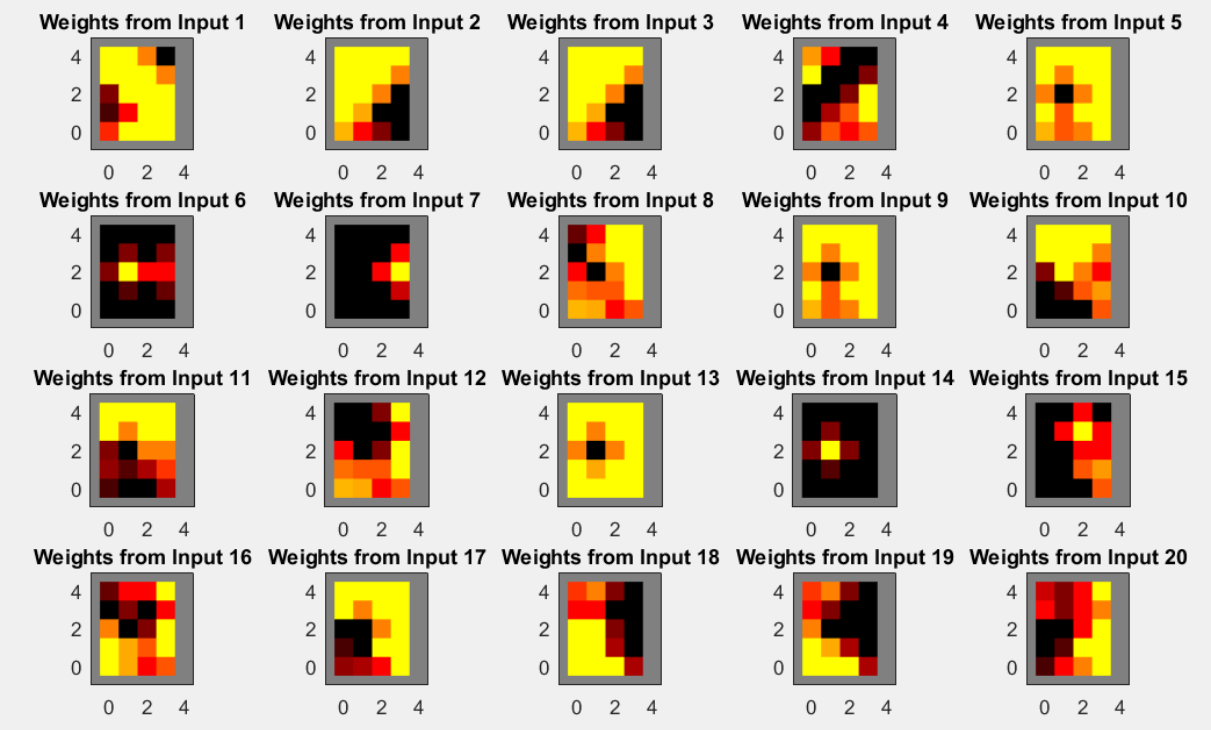
SOM Neighbor Connections:
Zrzut ekranu przedstawia połączenia pomiędzy poszczególnymi neuronami w utworzonej sieci.



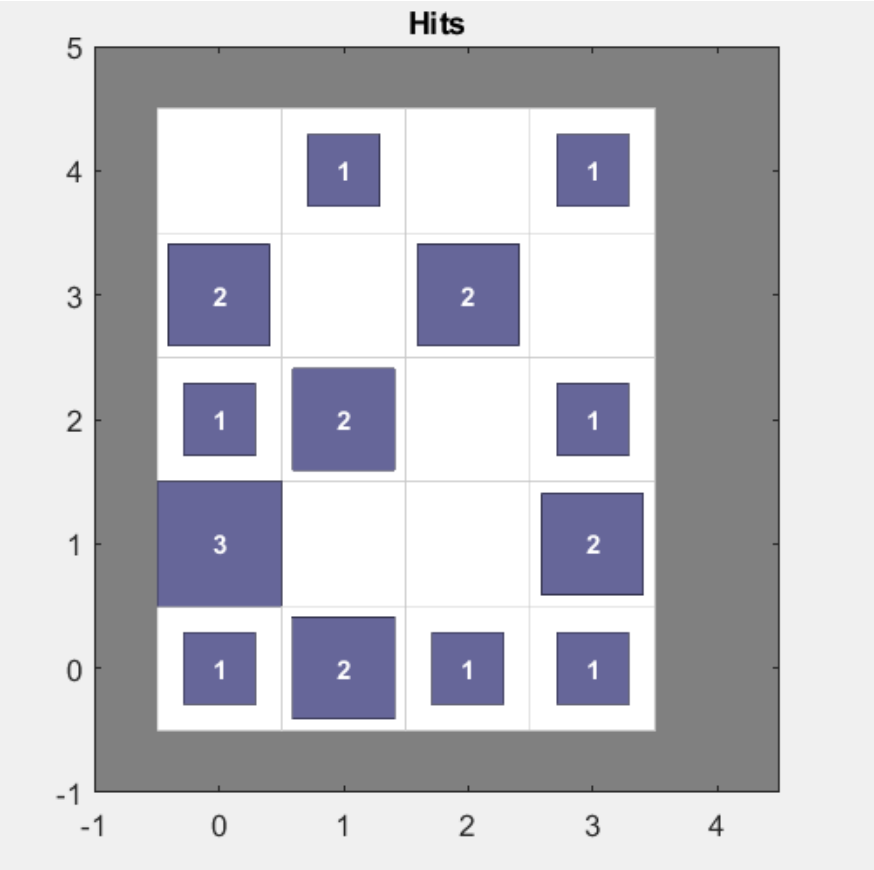
SOM Neighbor Distance:
Zrzut ekranu przedstawia odległości pomiędzy wagami poszczególnych neuronów (im kolor ciemniejszy, tym oznacza większą wartość wagi).



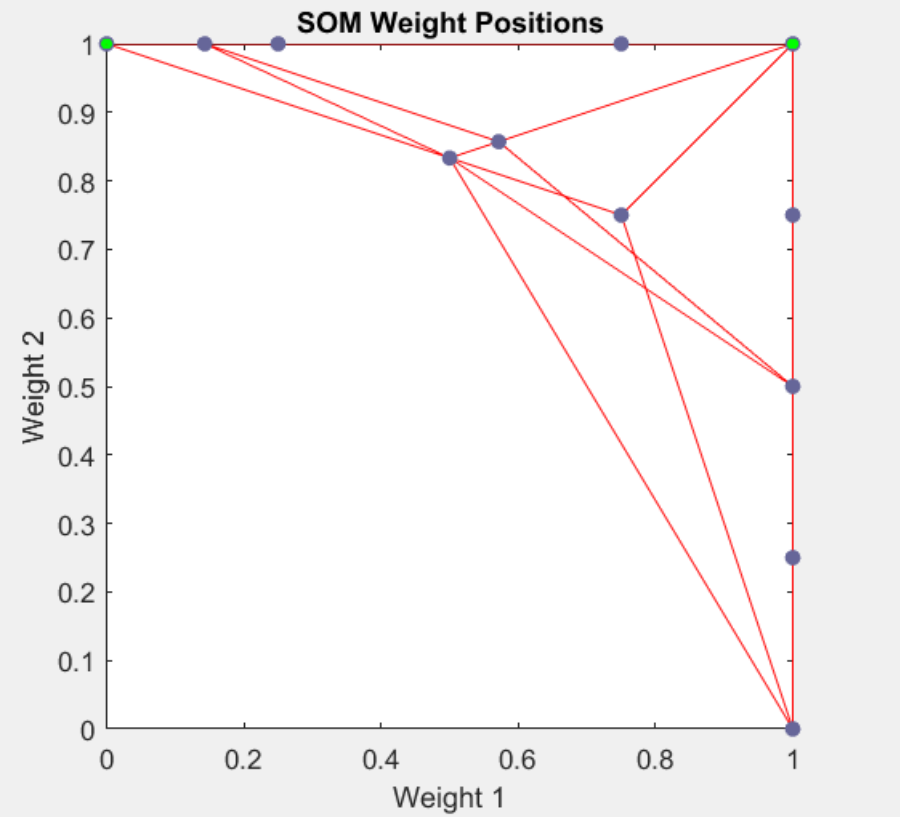
SOM Input Planes:
Zrzut ekranu przedstawia kolejne wejścia. Podobnie jak wcześniej - im ciemniejszy kolor, tym wyższa waga.



SOM Sample Hits:
Zrzut ekranu przedstawia rozkład sił neuronów (ilość zwycięstw neuronów w konkurencji do WTM).



SOM Weight Positions:
Zrzut ekranu przedstawia efekt końcowy nauki SOM (niebieskie kropki to neurony, natomiast linie to połączenia między nimi).



Opis użytych funkcji:
– **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 (domyślnie = 100),
– **initNeighbor** – początkowy rozmiar sąsiedztwa (ustawiony na najbliższe = 1),
– **topologyFcn** – funkcja topologii warstw (domyślnie = 'hextop'),
– **gridtop** – funkcja topologii siatki prostokątnej,
– **distanseFcn** – funkcja odległości neuronowej (domyślnie = "linkdist"),
– **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 kodu:

```
close all; clear all; clc;
%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];
           %A B C D E F G H I J K L M N O P R S T U

% 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 = 2000; %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
```