Rafał Jaroszek, 286103

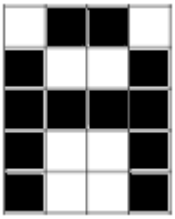
Podstawy sztucznej inteligencji
Sprawozdanie do scenariusza 6

Zadanie:

Przygotowanie sieci Kohonena przy wykorzystaniu reguły Winner Takes Most, która będzie odwzorowywać istotne cechy liter alfabetu.
Podczas realizacji scenariusza będę korzystał z oprogramowania MATLAB 2016a oraz biblioteki Neural Network Toolbox.

Dane uczące:

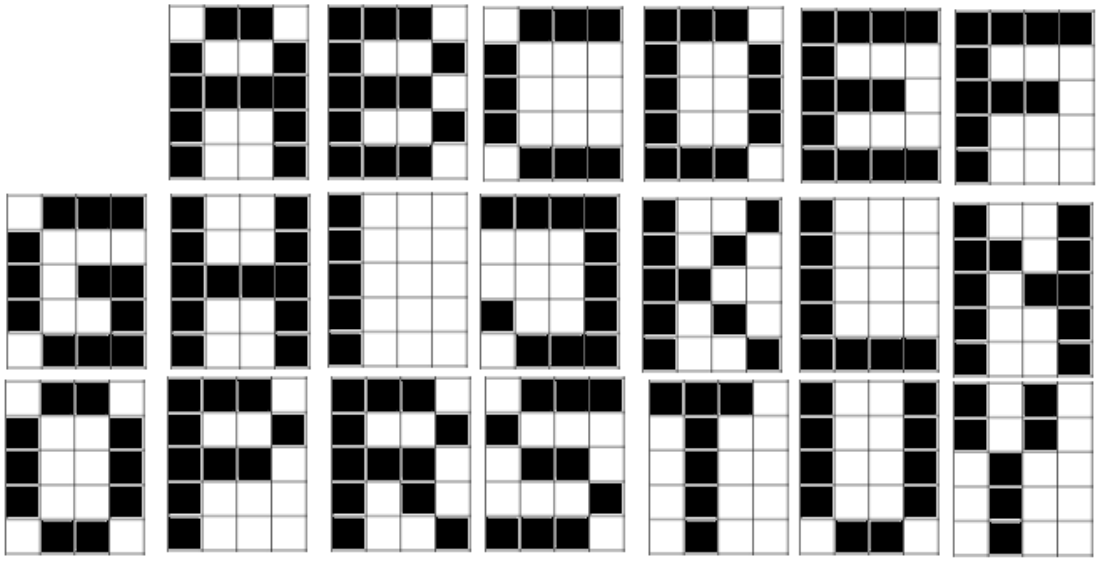
celu realizacji scenariusza przygotowałem dane uczące składające się z 20 dużych liter: A, B, C, D, E, F, G, H, I, J, K, L, N, O, P, R, S, T, U, Y. Litery są zapisywane na tablicy 4x5 wg takiego samego sposobu jak w poprzednich scenariuszach:



===> 0 1 1 0
1 0 0 1
1 1 1 1
1 0 0 1
1 0 0 1

===> A=0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1

Taki ciąg znaków zostaje wpisany jako kolumna w wartościach uczących.
Poniższa grafika obrazuje kształty liter, które wybrałem:



Takie litery skutkują następującymi danymi uczącymi, gdzie każda kolumna odpowiada kodowi

innej litery:

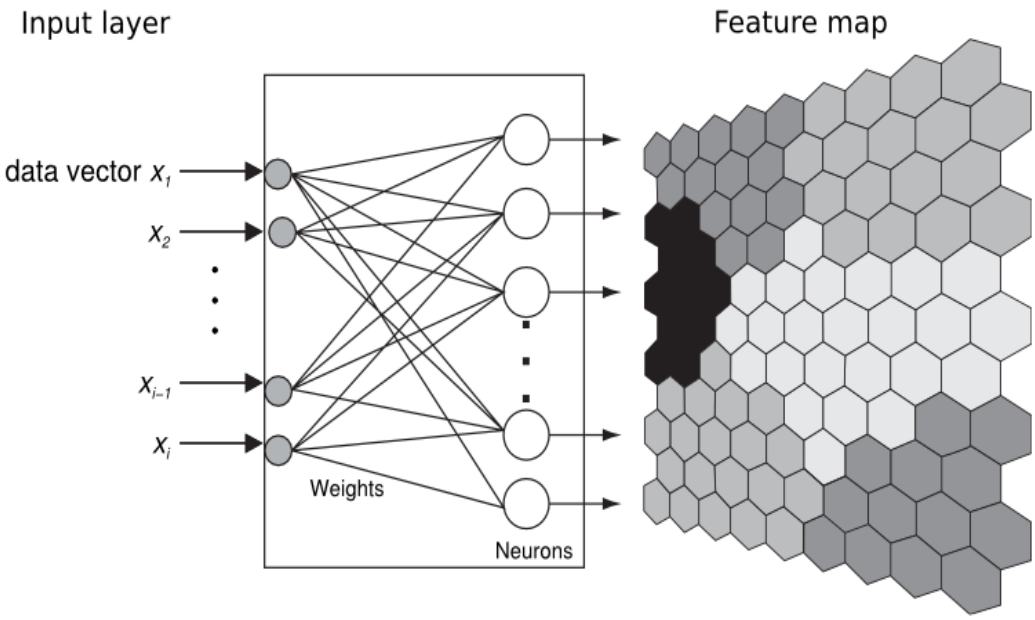
```
%A B C D E F G H I J K L N O P R S T U Y
[0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1;
 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 0;
 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
 0 0 1 0 1 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0;%
 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1;
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0;
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
 1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0;%
 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0;
 1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1;
 1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0;
 1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0;%
 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0;
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0;
 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0;%
 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0;
 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1;
 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0;
 1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0;%
 ];
%A B C D E F G H I J K L N O P R S T U Y
```

Sieć Kohonena:

Sieć Kohonena jest szczególnym przypadkiem algorytmu realizującego uczenie bez nauczyciela. Jej głównym zadaniem jest organizacja wielowymiarowej informacji (np. obiektów opisanych 20 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.

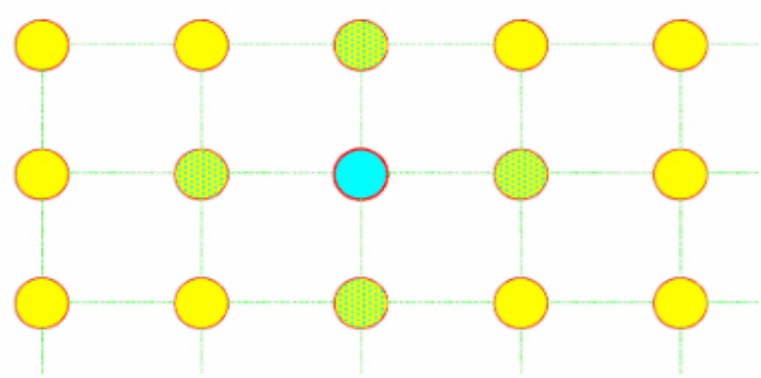
Funkcjonowanie sieci Kohonena odbywa się w trzech etapach:
-konstrukcja
-uczenie
-rozpoznawanie

Sieci Kohonena opierają się na uczeniu metodą samoorganizacyjną typu konkurencyjnego – oznacza to że neurony współzawodniczą między sobą o to, by zwiększyć wartość swojej wagi. Istnieje wiele różnych metod współzawodnictwa, które w zróżnicowany sposób wybierają zwycięski neuron. Ponieważ sieć Kohonena opiera się na uczeniu bez nauczyciela dane wyjściowe są opracowywane samodzielnie przez sieć na podstawie obserwacji danych wejściowych i wyławianiu zależności pomiędzy nimi. W sieciach SOM każdy neuron warstwy wejściowej komunikuje się z neuronami warstwy topologicznej ale neurony z tej samej warstwy nie komunikują się między sobą.

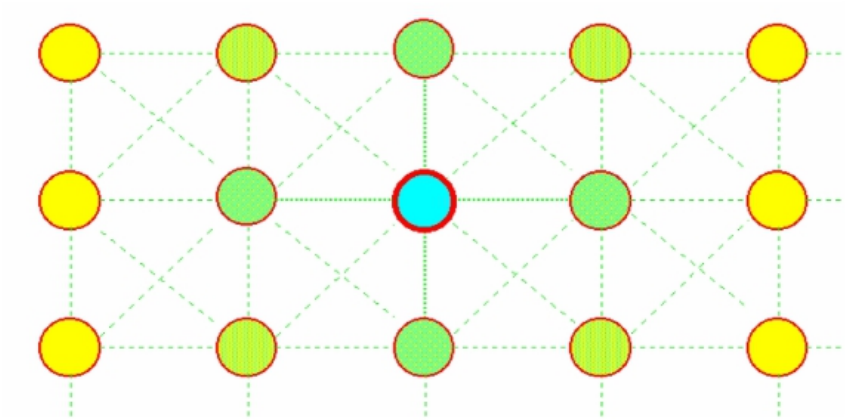


Zasada działania sieci Kohonena – dane wejściowe trafiają do neuronów a następnie są odwzorowywane na warstwę topologiczną, co daje siatkę neuronów z efektem działania sieci. Neurony mogą być ułożone w siatkę heksagonalną lub prostokątną.

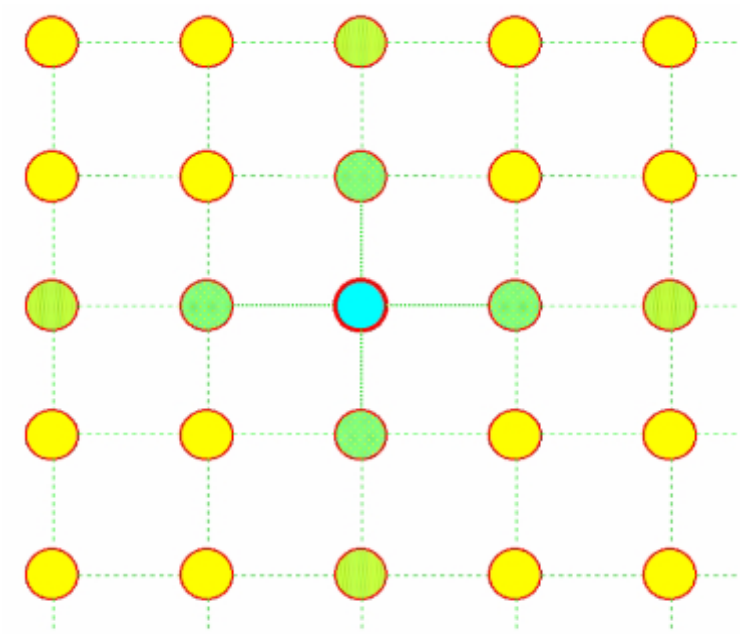
Reguła Winner Takes Most:
Ponieważ reguła Winner Takes All jest metodą słabo zbieżną, szczególnie dla dużej liczby neuronów, w praktyce częściej stosuje się regułę Winner Takes Most. Jest ona oparta na takiej samej zasadzie jak WTA z tą różnicą, że oprócz zwycięzcy wagi modyfikują również neurony z jego sąsiedztwa, przy czym im dalsza jest odległość od zwycięzcy, tym mniejsza jest zmiana wartości wag neuronu.



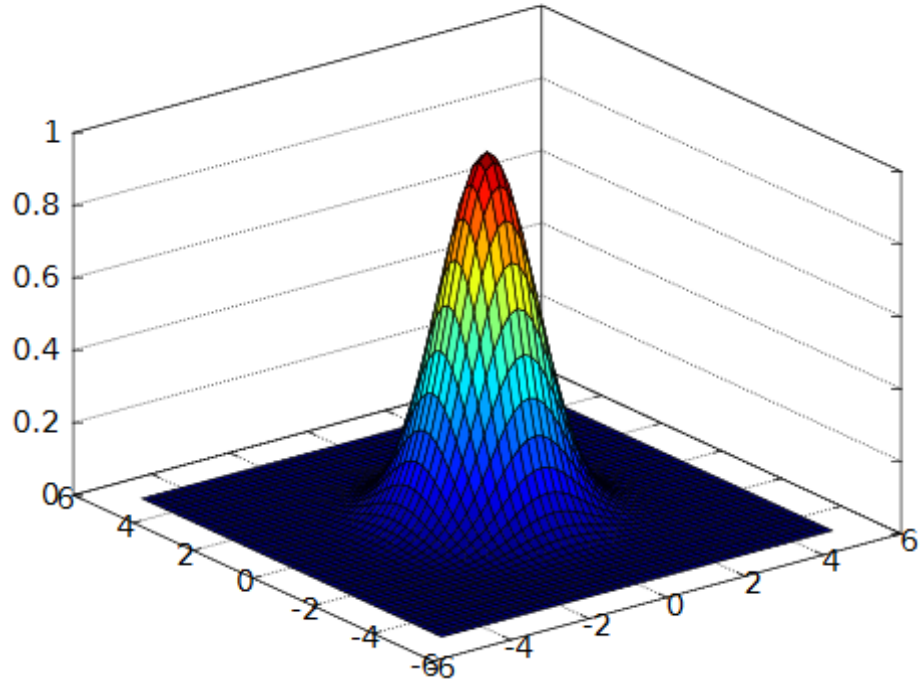
W powyższym przykładzie zostaną zwiększone wagi neuronu zwycięskiego (niebieski) a także jego sąsiadów (zielone).



Sąsiedztwo jest pojęciem umownym, można także zdefiniować sąsiadów bliższych i dalszych. Ich wagi będą modyfikowane w zależności od odległości od neuronu zwycięskiego: najmniejsza modyfikacja przypadnie jasnozielonym neuronom ponieważ ich odległość od zwycięzcy jest większa niż neuronów ciemnozielonych.



Sąsiedztwo nie oznacza że neurony muszą być bezpośrednio połączone ze zwycięzcą. W powyższym przykładzie mamy sąsiadów bliższych, którzy bezpośrednio graniczą ze zwycięskim neuronem, a także sąsiadów dalszych, którzy są sąsiadami wyłącznie dla sąsiadów bliższych a same nie są połączone ze zwycięzcą.

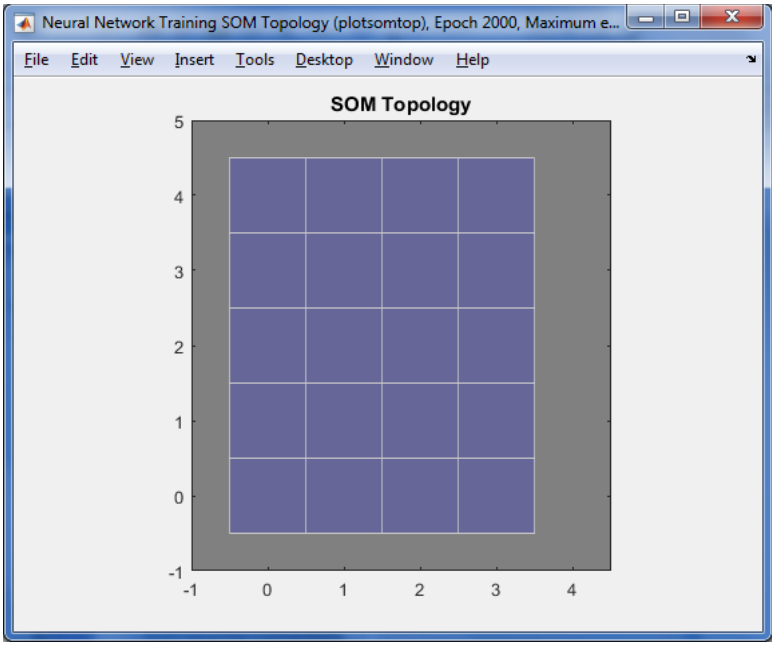


Zasada modyfikowania wag w sąsiedztwie: im dalej neuron znajduje się od zwycięzcy (ciemnoczerwony punkt) tym mniejsza będzie modyfikacja jego wagi.

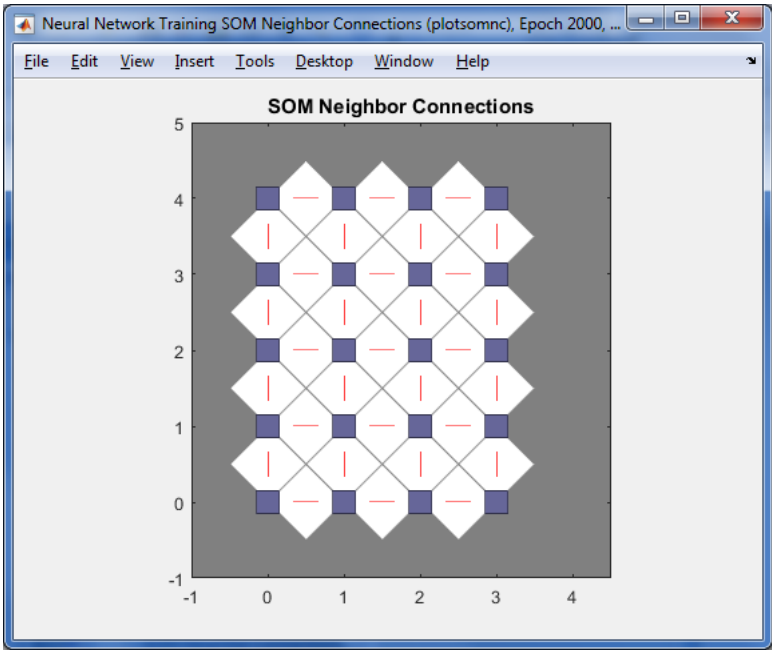
Aby uniknąć tego, że jeden neuron będzie zawsze wygrywał podobnie jak w metodzie WTA stosuje się mechanizm zmęczenia który polega na tym, że jeśli jakiś neuron wygrywa zbyt często to na pewien czas przestaje być brany pod uwagę w rywalizacji.

Do uczenia sieci używamy następującego algorytmu:
-generujemy losowo znormalizowane wektory wag
-losujemy wektor x oraz liczymy dla niego aktywację y dla wszystkich neuronów
-szukamy neuronu zwycięzcy
-modyfikujemy wektory wag zwycięzcy oraz sąsiedztwa a następnie normalizujemy zwycięzcę (sprawdzamy czy nie wygrywa zbyt często, jeśli tak to jest na chwilę usypiany)
-zatrzymujemy algorytm po odpowiednio dużej ilości iteracji

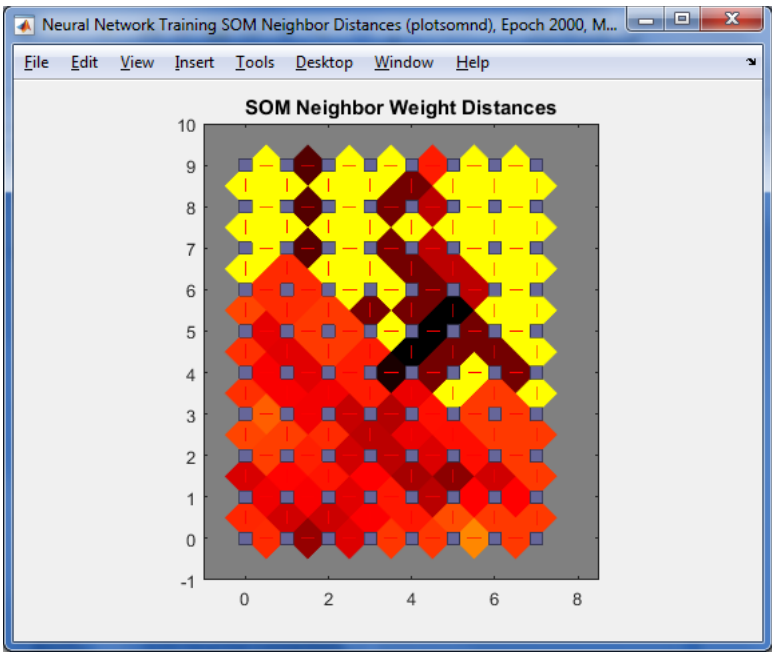
Wyniki działania programu:
W programie wykorzystałem prostokątną siatkę neuronów, uczenie wg reguły Kohonena i WTM. Program w efekcie powinien odwzorować istotne cechy liter alfabetu na podstawie otrzymanych danych. Jako wymiar sieci wybrałem siatkę 4x5 czyli odpowiadającą wymiarom liter. Sąsiedztwo w podanym przykładzie wynosi 1 tzn. sąsiadami są wyłącznie neurony graniczące bezpośrednio ze zwycięzcą.

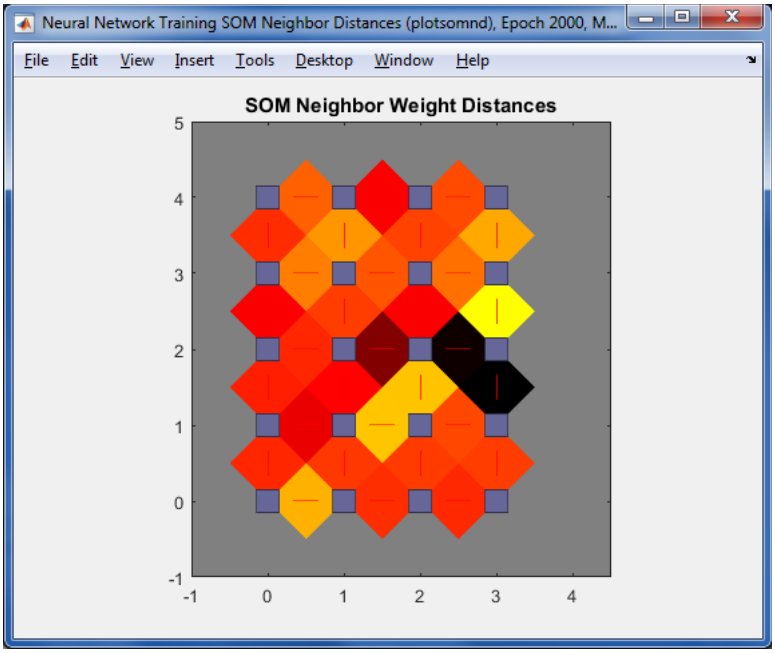


Topologia sieci

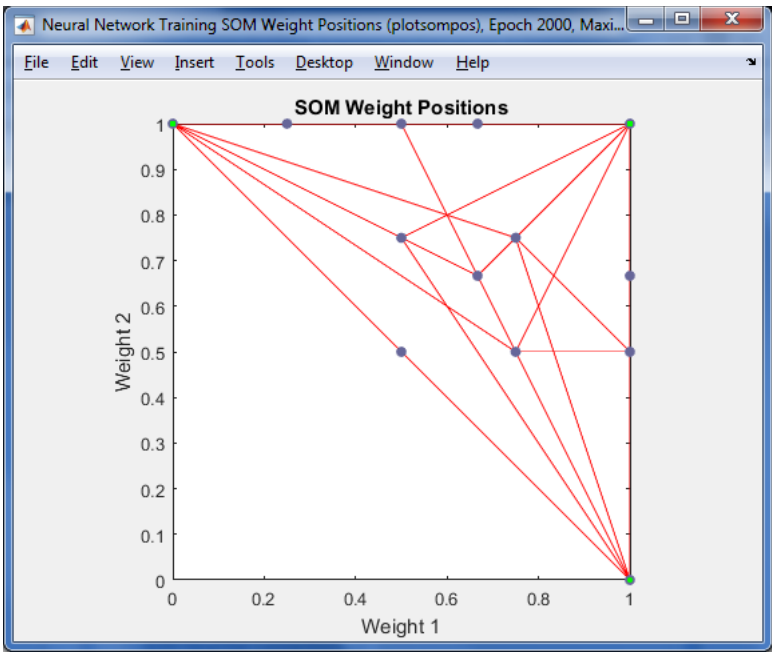


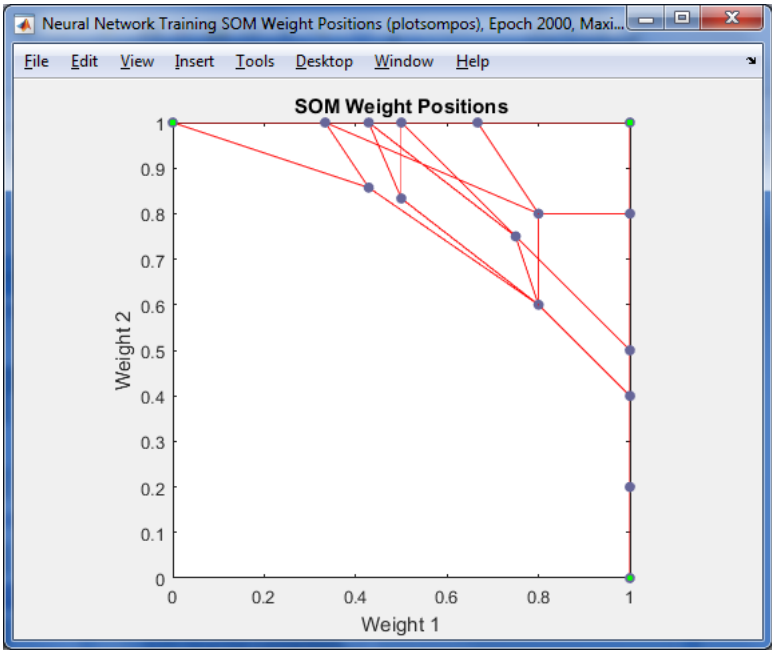
Połączenia między neuronami



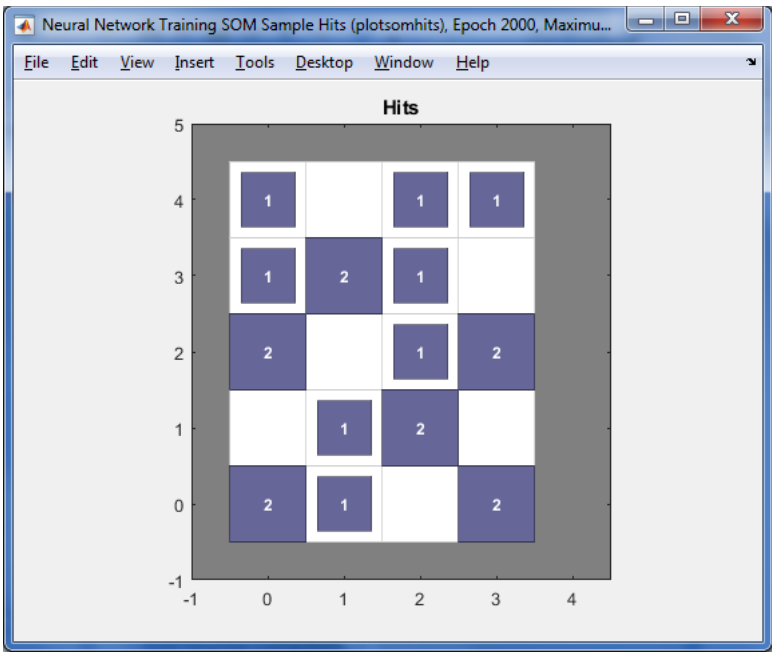


Odległości pomiędzy wagami – im ciemniejszy kolor tym większa odległość. Dodatkowo umieściłem wykres również dla sieci o rozmiarach 8x10 aby możliwe było porównanie różnic pomiędzy nimi.





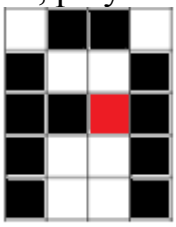
Rozkład wag: czerwone linie to połączenia między neuronami, kropki to neurony. U góry rozkład dla sieci 8x10, niżej dla 4x5. Widać wyraźnie że rozkłady wag różnią się pomiędzy sobą aczkolwiek zachowują pewne podobieństwa.



Przykładowa grafika pokazująca jak często i jaki neuron zostawał zwycięzcą.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	1	0	1	0	0	0
13	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	1	0	0	0	0	1	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fragment tablicy przechowującej wynik działania programu – 1 oznacza najbardziej „typową” kratkę dla każdej z liter, przykładowo dla A jest to pole 13 czyli



Analiza oraz wnioski:

Porównując wyniki WTM z wynikami WTA z poprzedniego laboratorium można zauważyć kilka istotnych różnic w działaniu sieci. Analizując rozkład wag, ilość

zwycięstw w rywalizacji i ich rozłożenie w sieci widać, że algorytm Winner Takes Most skutkował niemalże równomiernym rozłożeniem zwycięzców po całej sieci. W przeciwieństwie do Winner Takes All gdzie neurony wygrywające były zebrane mniej więcej w jednej okolicy WTM rozłożył zwycięzców nieco bardziej „sprawiedliwie", co w efekcie mogło sprzyjać poprawności działania sieci gdyż nie skupiała się ona na jednym, konkretnym swoim fragmencie tylko brała podczas działania pod uwagę całą sieć.
Analiza schematów przedstawiających rozkład wag pozwala zauważyć że wagi są różnie rozłożone w zależności od ilości neuronów w sieci: mniejsza ilość neuronów skutkowała wypchnięciem neuronów w kierunku prawego górnego rogu wykresu tzn. że wagi neuronów miały wartości wysokie i mocno do siebie zbliżone. Z kolei 4 razy większa ilość neuronów prowadziła do nieco większych różnic w wagach. Jednak warto zauważyć że pomimo różnic w rozmiarze obie sieci zachowały się podobnie w rozłożeniu wag – oba wykresy nie mają wartości zbliżonych do lewego dolnego rogu. Warto zauważyć że zwiększenie ilości neuronów oczywiście wpłynęło negatywnie na czas obliczeń, wydłużając go kilkukrotnie, należy się więc zastanowić nad tym, czy tworzenie sieci złożonej z wielu neuronów jest zawsze opłacalne.
Próby uruchomienia sieci dla dużej wartości sąsiedztwa spowodowały niepoprawne działanie programu. Z kolei jeśli dla tej samej wartości sąsiedztwa zwiększyłem rozmiar sieci to jej działanie uległo niewielkiej poprawie – wynika z tego że sąsiedztwo powinno być dostosowane do rozmiarów sieci ponieważ przy małej sieci i dużym sąsiedztwie neurony sąsiadujące będą na siebie wzajemnie nachodzić co w efekcie da nam sieć w której wagi mogą być na bardzo podobnym poziomie.
Podobnie jak w przypadku Winner Takes All ilość zwycięstw poszczególnych neuronów wpływa na odległości wag, jednak w przypadku Winner Takes Most unika się zgrupowania zwyciężających neuronów w jednym miejscu co w efekcie daje nieco lepsze rozłożenie niż w przypadku WTA.
Wydłużanie treningu nie dawało znaczenie lepszych efektów działania ale jest to zapewne spowodowane tym, ze sieć uczy się bez nauczyciela – aby możliwe było zaobserwowanie znacznych różnic należało by dać sieci bardzo dużo czasu na naukę.
Jeśli chodzi o efektywność sieci to sądzę że wiele zależy także od danych wejściowych – ponieważ litery które dostarczyłem sieci składały się tylko z 20 pól każda trudno oczekiwać, że sieć byłaby w stanie bezproblemowo rozróżniać takie litery – niektóre z nich różnią się zaledwie jedną kratką a ta różnica może zaniknąć w wyniku modyfikacji wag sąsiednich neuronów.

Wnioski:
-WTM nie pozwala na dominację małej ilości neuronów w sieci, rozkłada wagi bardziej równomiernie niż WTA
-ilość neuronów w sieci ma istotne znaczenie dla efektywności jej działania ale bez względu na liczbę neuronów sieć będzie mieć typowe cechy powiązane z danymi uczącymi
-należy postarać się dostosować ilość neuronów w sieci do zapotrzebowania aby otrzymać wystarczająco dokładne wyniki przy maksymalnie krótkim czasie obliczeń
-współczynnik uczenia kontroluje przydział wag dla neuronów

-sąsiedztwo ma bardzo istotny wpływ na efekt działania sieci, zbyt duże sąsiedztwo w małej sieci doprowadzi do niepoprawnego rozłożenia wag
-reguła WTM w ogólnej ocenie sprawuje się lepiej od reguły WTA, szczególnie w przypadku sieci złożonych z dużej ilości neuronów ale w przypadku sieci o bardzo małej liczbie neuronów może powodować niepoprawne działanie sieci ze względu na modyfikację wag sąsiadów
-sieć uczy się powoli ze względu na to że jest to uczenie bez nauczyciela więc dla znacznej poprawy jakości sieci należy umożliwić jej uczenie się przez wiele epok
-efektywność działania sieci w dużym stopniu zależy od jakości dostarczonych danych uczących

Listing programu:

```
close all; clear all; clc;
   %A B C D E F G H I K L J M N O P R S T U
WE=[0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 0;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
    0 0 1 0 1 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0;%
    1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1;
    0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
    1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0;%
    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1;
    1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0;
    1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0;%
    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0;
    1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0;%
    1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0;
    1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0;%
    ];


% parametry
dimensions   = [4 5];
coverSteps   = 100;
initNeighbor = 1;
topologyFcn  = 'gridtop';
distanceFcn  = 'dist';


% Tworzenie SOM
net = selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn);
net.trainParam.epochs = 2000;

% Trenowanie sieci
[net,tr] = train(net,WE);
y = net(WE);
classes = vec2ind(y);
```