Katarzyna Mlicka
Inżynieria Obliczeniowa WIMiIP
Nr indeksu 286120

Podstawy Sztucznej Inteligencji
Scenariusz 6:

Budowa i działanie sieci Kohonena dla WTM.

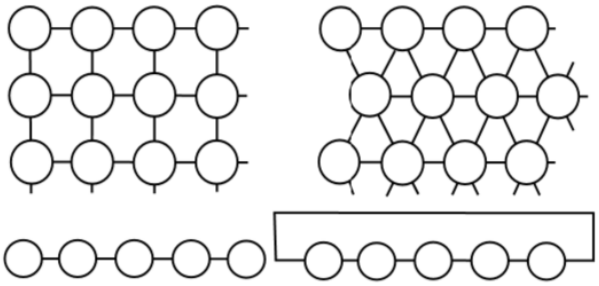
Celem ćwiczenia jest poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTM do odwzorowywania istotnych cech liter alfabetu.

WINNER TAKES MOST

WTA nie jest uznawana za wystarczającą ze względu na niską zbieżność dla dużej liczby neuronów. WTM (ang. Winner Takes Most) opiera się nie na modyfikacji wyłącznie wagi zwycięzcy ale modyfikacji wag sąsiadów, z tym że im większa odległość od zwycięzcy tym mniejsza modyfikacja. W praktyce reguła WTA została zastąpiona regułą WTM.

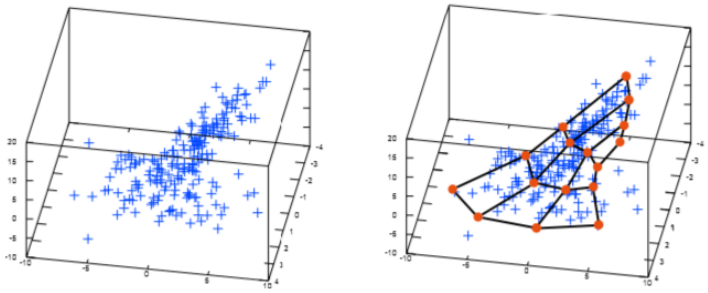
Reprezentacja sieci. Dana jest sieć neuronowa modelowana przy pomocy grafu nieskierowanego H = (V,E)gdzie V jest zbiorem wierzchołków (neuronów), zaś E zbiorem krawędzi. Dla każdego wierzchołka v ∈V określamy tak zwane sąsiedztwo s(v) = {u ∈V|(v,u) ∈E}. Załóżmy ponadto, że mamy pewną ilość obiektów (przykładów) opisywanych punktami przestrzeni R^d.

Topologie sieci:



Stawiamy sobie następujące cele:

- każdemu neuronowi chcemy przypisać d wag reprezentujących współrzędne pewnego punktu w R^d),
- powyższe przypisanie ma odpowiadać rozkładowi przestrzennemu przykładów (tzn. jeżeli w danej okolicy jest dużo przykładów, powinno do niej zostać przypisanych wiele neuronów),
- przypisanie ma odpowiadać strukturze topologicznej grafu, tzn. sąsiadujące ze sobą neurony powinny tra#ać w niedaleko od siebie po^{a}o»one obszary R^d.



ALGORYTM

Niech dany będzie zbiór przykładów $P = \{P_1,...P_N\} \in \epsilon R^d$. Każdemu neuronowi $v \epsilon \in V$ przypisujemy wektor wag $\pi(v) \in \epsilon R_d$.

1. Wybieramy pewną liczbę T oznaczającą liczbę powtórzeń algorytmu
2. Losujemy wagi przy wszystkich neuronach
3. Dla t = 1, ... ,T wykonujemy:
   a) Losujemy przykład P
   b) Znajdujemy jednostkę $v \epsilon \in V$, której zestaw wag $\pi(v)$ leży najbliżej P
   c) Dla neuronu zwycięzcy i wszystkich neuronów z jego sąsiedztwa s(v) wykonujemy: $\pi(w) = \pi(w) + \alpha(t)*(P - \pi(w))$

      $\alpha(t)$ jest funkcją o wartościach dodatnich, malejąca w kolejnych iteracjach algorytmu, na przykład $\alpha(t) = 1 - (t - 1)/T$

Do algorytmu można wprowadzać modyfikacje, które w znacznym stopniu mogą poprawić jego działanie. Bardzo często inaczej defniuje się sąsiedztwo s(v) i wprowadza drobne zmiany w sposobie uaktualniania wag.

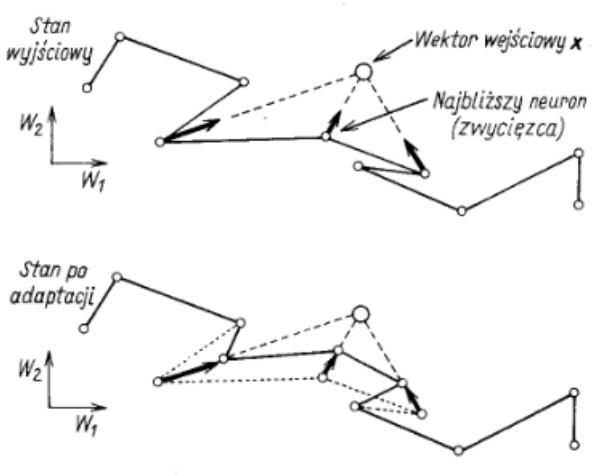
Algorytmy WTA, w których tylko jeden neuron może podlegać adaptacji w każdej iteracji, są algorytmami słabo zbieżnymi, szczególnie przy dużej liczbie neuronów.

W praktyce zostały one zastąpione algorytmami WTM (ang. Winner Takes Most) w których oprócz zwycięzcy uaktualniają swoje wagi również neurony z jego ząsiedztwa:

$$w_i = w_i + \eta_i * G(i,x)[x - w_i]$$

dla wszystkich neuronów i należących do sąsiedztwa S_{j*} zwycięzcy.
Definiując G(i,x) = 1 dla i = j* oraz G(i,x) = 0 dla i ≠ j* otrzymujemy algorytm WTA jako szczególny przypadek WTM.

Przykład adaptacji wag WTM: (ilustracja procesu adaptacji wag)

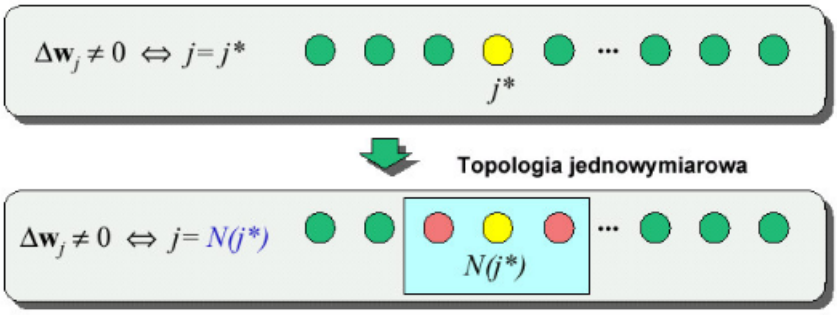


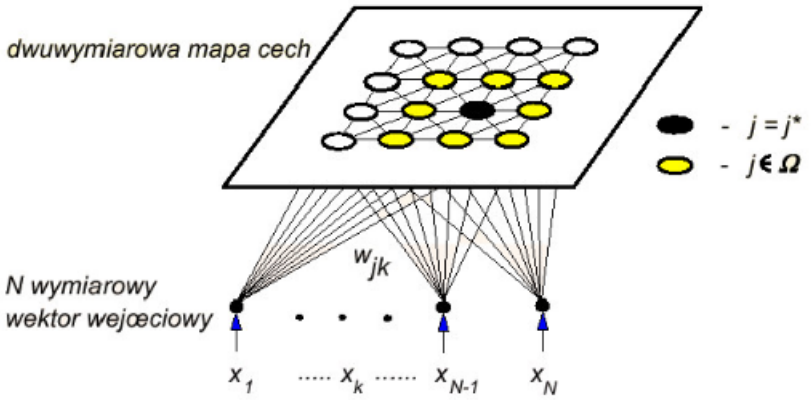
MAPA CECH KOHONENA

Kohnen zaproponował regułę uczenia konkurencyjnego, w której w odróżnieniu od reguły standardowej, modyfikacji wag dokonuje się nie tylko dla neuronu zwycięskiego, lecz również dla pewnej liczby neuronów z jego otoczenia.

Reguła ta wymaga wprowadzenia tzw. Topologicznego uporządkowania neuronów w warstwie wyjściowej (zwanej też mapą cech), dlaktórej można wprowadzić pojęcie otoczenia neuronu zwycięskiego np. Przez zaliczenie do tego otoczenia neuronów, których indeksy porządkujące nie różnią się o większą niż złożona wartość, która wyznacza to otoczenie.

Cel jest taki że neurony położone blisko mają pełnić podobną funkcję. Modyfikacje reguły WTA tworząca obszary wrażliwe na podobne wzorce.



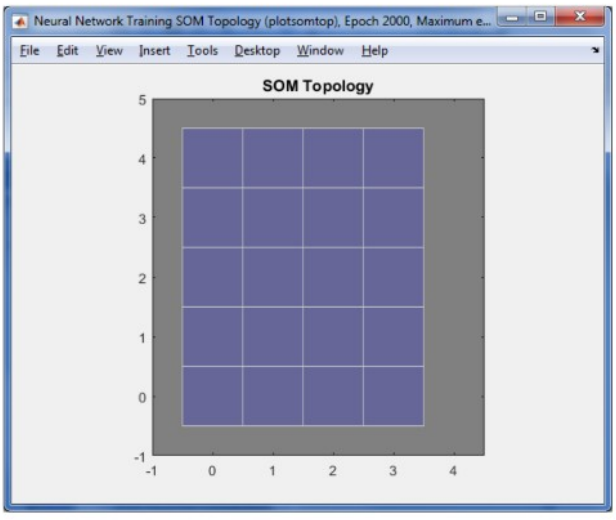


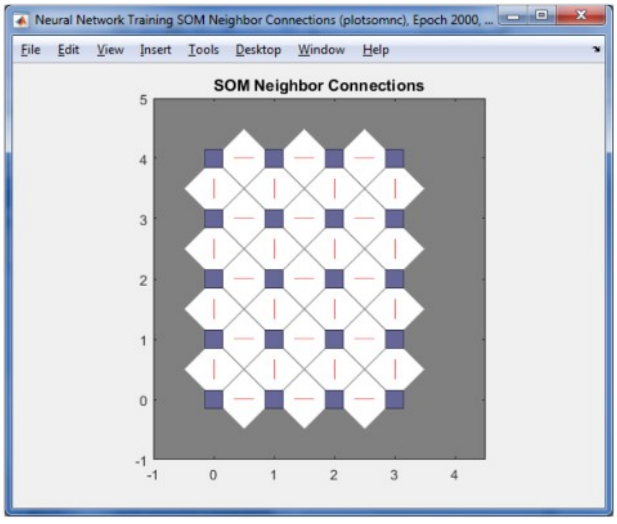
Algorytm uczenia konkurencyjnego Kohnena jest dany zależnością:

$$w_j(k+1) = \begin{cases} w_j(k) + \eta(k)[x - w_j(k)] & j \in \Omega(k) \\ w_j(k) & j \notin \Omega(k) \end{cases}$$

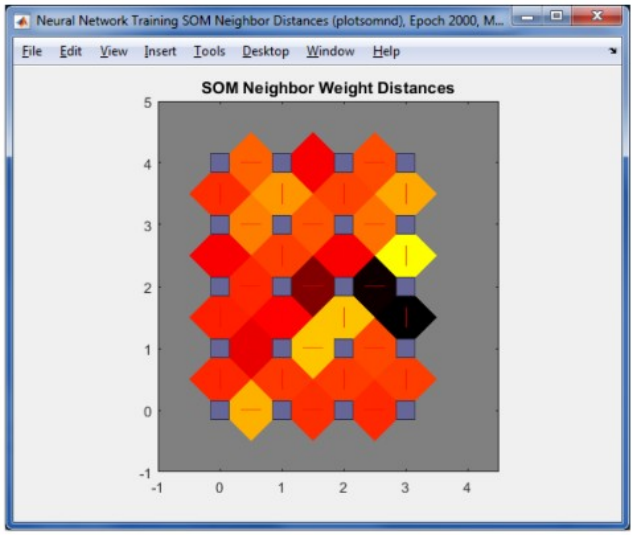
WYNIKI DZIAŁANIA PROGRAMU ORAZ WNIOSKI

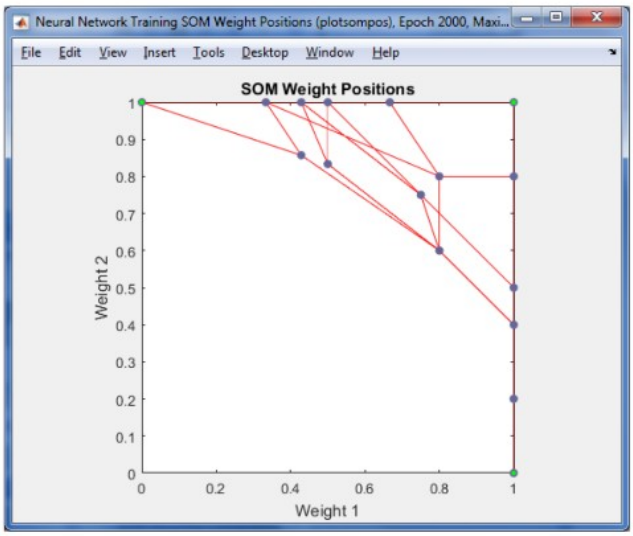
Topologia sieci i połączenia między neuronami:





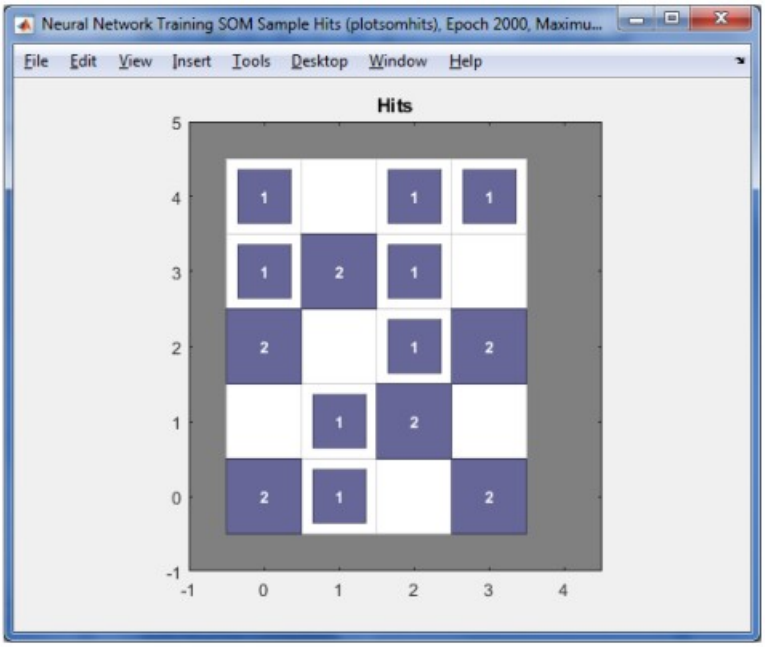
Odległości między wagami oraz rozkład wag:





(Po lewej): im ciemniejszy kolor, tym większa odległość pomiędzy wagami. (Po prawej): WTM rozkłada wagi równomiernie, nie pozwalając na dominację żadnego z neuronów.

Kto i jak często został zwycięzcą:



Nie tworzą sie strefy wpływów – zwycięzcy są rozłożeni równomiernie. Sieć nie skupia się na pojedynczych neuronach lub ich skupiskach, a ogarnia swoim zasięgiem całość.
Zbyt duża wartość sąsiedztwa sprawiała że sieć nieprawidłowo rozłożyła wagi. Mogła mieć na to wpływ wielkość sieci, stąd wniosek że sąsiedztwo powinno być dostosowane do jej rozmiarów.

LISTING KODU MATLAB

```
close all; clear all; clc;
%od A do U (20x20)[4,5]
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 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 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 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;
    ];
%parametry
dimensions   = [4 5];
coverSteps   = 100;
initNeighbor = 1;
%topologie
topologyFcn  = 'gridtop';%ustala wartość w rzędach do góry
distanceFcn  = 'dist';%ustala wartość w rzędach w stylu heksagonalnym
%tworzenie SOM (self organisation-map):
net = selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn);
net.trainParam.epochs = 2000;
%trenowanie sieci
[net,tr] = train(net,WE);
y = net(WE);
classes = vec2ind(y);
```