Akademia Górniczo-Hutnicza
WIMIiP, Inżynieria Obliczeniowa
grupa laboratoryjna 01
Agnieszka Kępka

28.12.2018,Kraków

Podstawy Sztucznej Inteligencji

Sprawozdanie numer 6

Budowa i działanie sieci Kohonena dla WTM

1. Cel ćwiczenia:

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

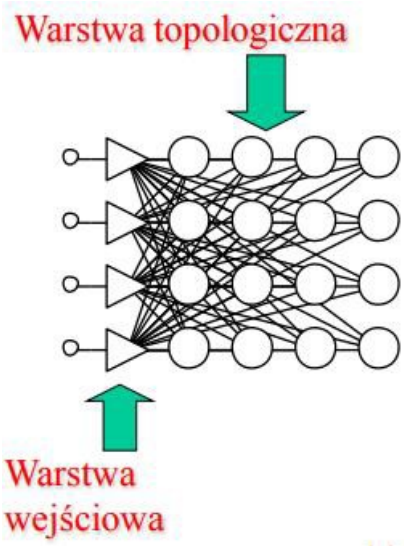
2. Zadania do wykonania:

- Wygenerowanie danych uczących i testujących, zawierających 20 dużych liter dowolnie wybranego alfabetu w postaci dwuwymiarowej tablicy np. 4x5 pikseli dla jednej litery.
- Przygotowanie sieci Kohonena i algorytmu uczenia opartego o regułę Winner Takes Most (WTM).
- Uczenie sieci dla różnych współczynników uczenia.
- Testowanie sieci.

3. Zagadnienia teoretyczne:

Sieć Kohonena uczy się całkiem sama, wyłącznie obserwując przesyłane do niej dane, których wewnętrzna struktura i ukryta w tej strukturze nieznana logika będą decydować o końcowym obrazie klasyfikacji i o ostatecznym działaniu sieci.
Można założyć, że w sieci Kohonena poszczególne neurony będą identyfikowały i rozpoznawały poszczególne skupienia danych.
Przez skupienie danych rozumieć tu będziemy grupę danych: dane należące do skupienia są między sobą w jakimś stopniu podobne, natomiast dane należące do różnych skupień różnią się pomiędzy sobą.
Sieci Kohonena realizują pewne odwzorowania, stąd ich powszechnie używany symbol: SOM (Self Organizing Maps)
Sieci uczące się bez nauczyciela w trakcie uczenia opierają się wyłącznie na obserwacji danych wejściowych, nikt im natomiast nie mówi, co z tych danych wejściowych powinno wynikać to sieci muszą wykryć i ustalić same.

Struktura sieci Kohonena



Cechy charakterystyczne:

- sieć uczy się bez nauczyciela
- uporządkowane neurony wyjściowe
- jest konkurencja i wyłaniany jest neuron „zwycięski”
- ważną rolę odgrywa „sąsiedztwo”
- w wyniku uczenia powstaje mapa topologiczna
- aprioryczna interpretacja wartości wyjściowych jest niemożliwa
- po uczeniu można ustalić, jakie znaczenie mają poszczególne rejony mapy topologicznej - ale wyłącznie na podstawie analizy konkretnych Warstwa przykładów danych wejściowych.

Sieci samouczące się:

Samouczenie tym się głównie różni od uczenia, że podczas samouczenia nie ma żadnego zewnętrznego źródła wiedzy (bazy danych uczących, nauczyciela itp.), dostarczającego gotowe widomości, które wystarczy tylko sobie przyswoić. Umysł samouczącego się człowieka, a także samoucząca się sieć neuronowa, musi najpierw sama odkryć wiedzę, którą następnie zapamięta.

WTM - Winner Takes Most

WTM - Winner Takes Most Zwycięzca bierze najwięcej. W tej strategii nie tylko neuron najbardziej podobny, ale także jego otoczenie zostają zmodyfikowane. Najczęściej ta modyfikacja jest zależna od odległości sąsiada od zwycięzcy.

W metodzie tej oprócz wag zwycięskiego neuronu zmianie podlegają również wagi jego sąsiadów według reguły:

$$w_i(k+1) = w_i(k) + \eta_i(k)G(i,x)[x - w_i(k)]$$

Gdzie G(x,i) jest funkcją sąsiedztwa. W klasycznym algorytmie Kohenena funkcja G(x,i) zdefiniowana jest jako:

$$G(i,x) = \begin{cases} 1 & dla \quad d(i,w) \le \lambda \\ 0 & dla \quad d(i,w) > \lambda \end{cases}$$

Gdzie d(i,w) jest odległością pomiędzy neuronami w przestrzeni sieci tzn. na siatce (a nie w przestrzeni danych wejściowych!), a lambda jest promieniem sąsiedztwa malejącym do zera w trakcie nauki. Jest to tzw. sąsiedztwo prostokątne. Innym rodzajem sąsiedztwa jest sąsiedztwo gaussowskie:

$$G(i,x) = \exp(-\frac{d^2(i,w)}{2\lambda^2})$$

W tym przypadku stopień aktywacji neuronów jest zróżnicowany w zależności od wartości funkcji Gaussa. Sąsiedzi neuronu zwycięskiego są modyfikowani w różnym stopniu.

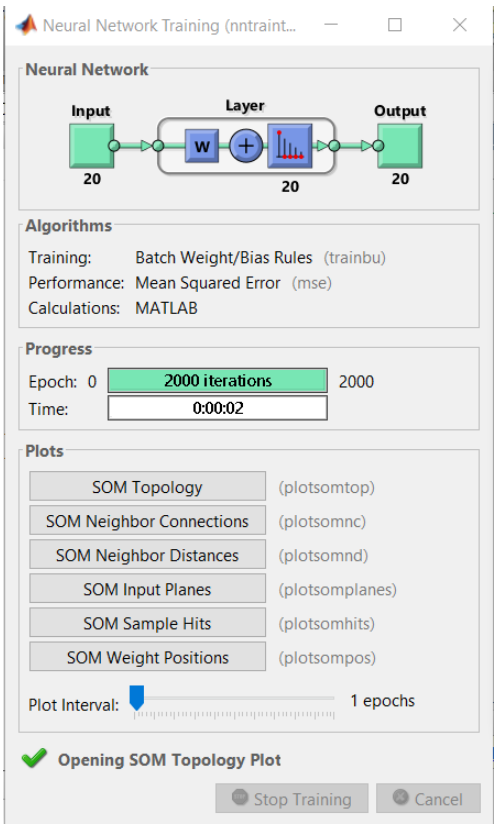
Często stosowanym podejściem jest wyróżnienie dwóch faz uczenia sieci Kohenena:

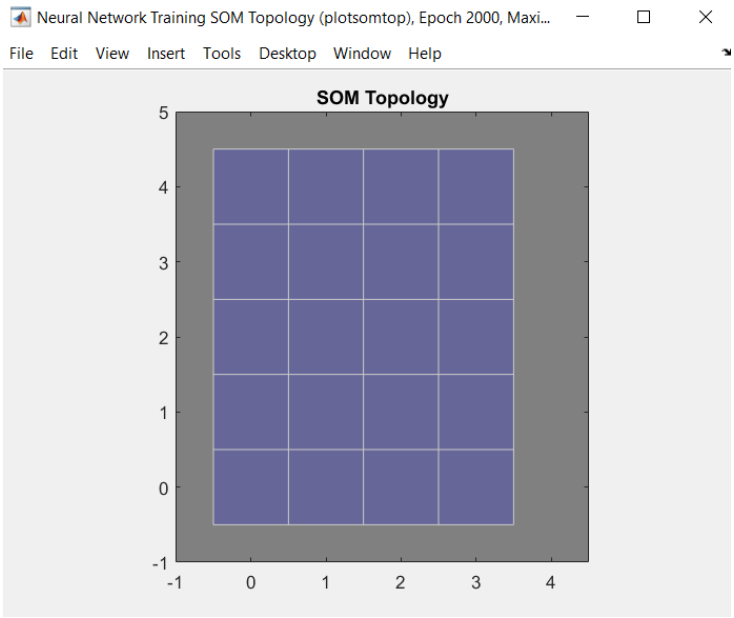
- **wstępna**, podczas której modyfikacji ulegają wagi neuronu zwycięzcy oraz jego sąsiedztwa malejącego z czasem
- **końcowa**, podczas której modyfikowane są wagi jedynie neuronu zwycięzcy

Mają one na celu najpierw wstępne nauczenie sieci, a następnie jej dostrojenie.

4.Wyniki przeprowadzonego ćwiczenia:

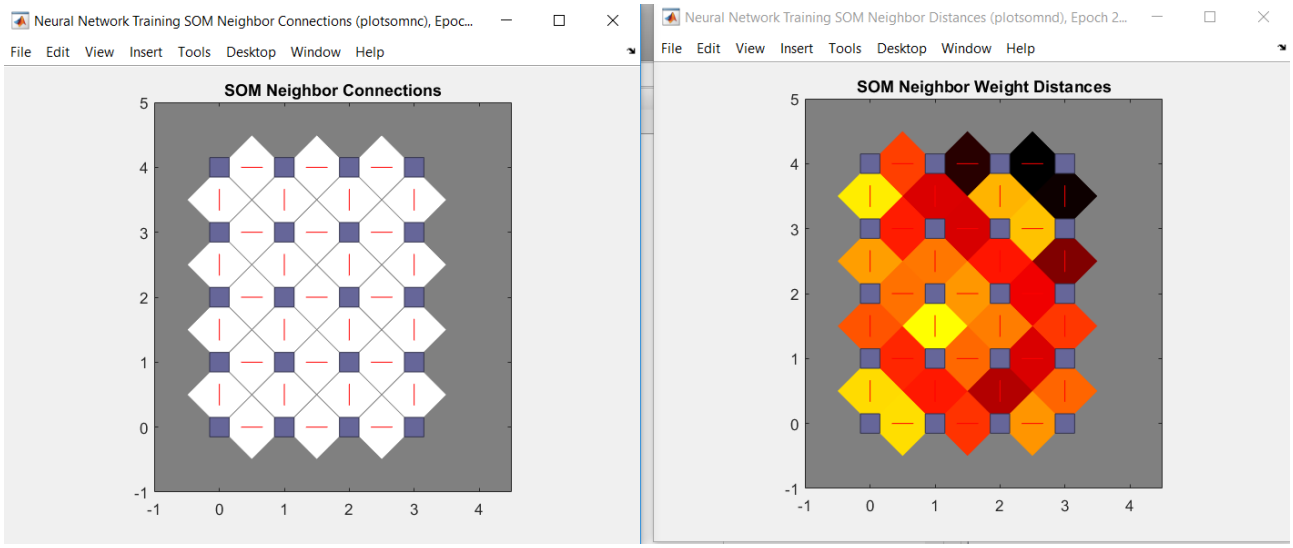
W programie Matlab, za pomocą biblioteki Neural Network Toolbox, zaimplementowałam sztuczną sieć neuronową. Danymi wejściowymi są litery A, B, C, D, E, F, G, H, I, J, K, L, N, O, P, R, S, T, U, Y są one reprezentowane za pomocą wartości binarnych (0 i 1) w tablicy o rozmiarze 4x5.





Trenowanie obejmuje maksymalną liczbę epok, która wynosi 2000, czas wykonania zadania to 0:00:02.

Wykorzystałam prostokątną siatkę neuronów. Uczenie sieci nastąpiło wg reuguły Kohonena oraz WTM. Program odwzorowuje istotne cechy liter alfabetu na podstawie danych wejściowych.

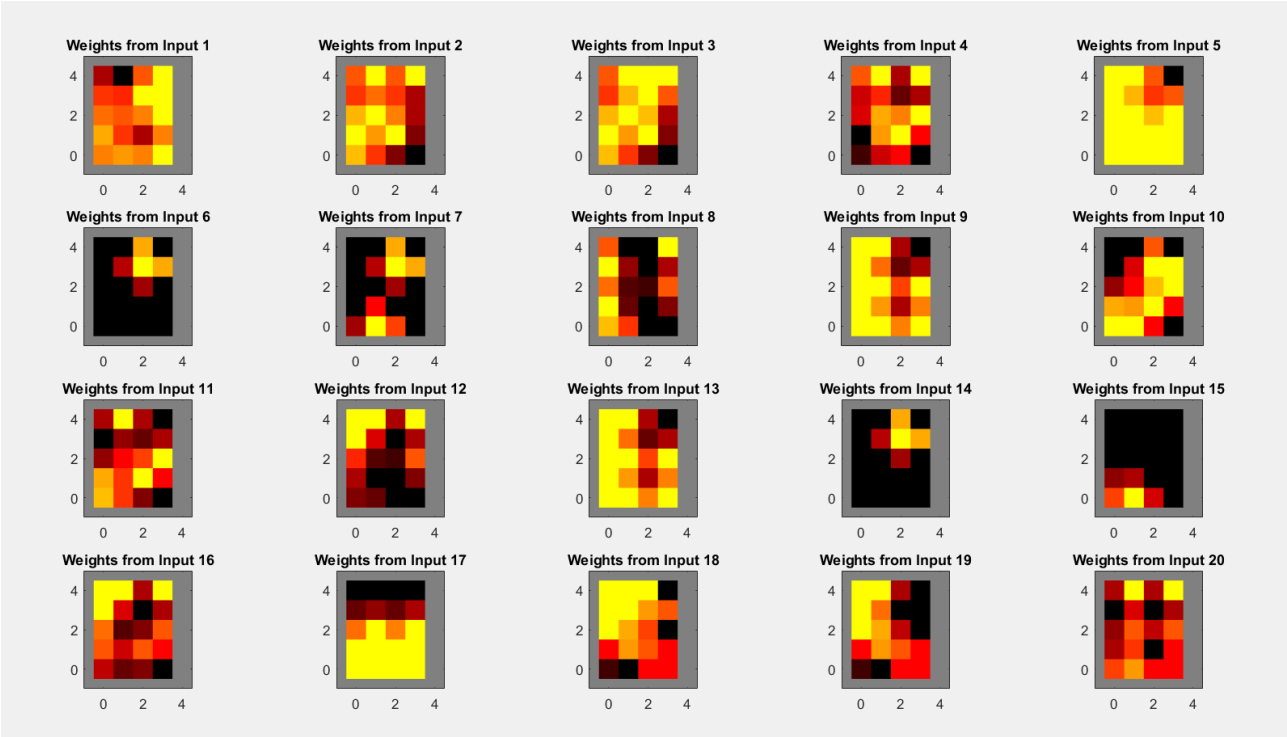


SOM Neighbor Connections:

Wyżej umieszczona mapa przedstawia powiązanie sąsiedzkie.

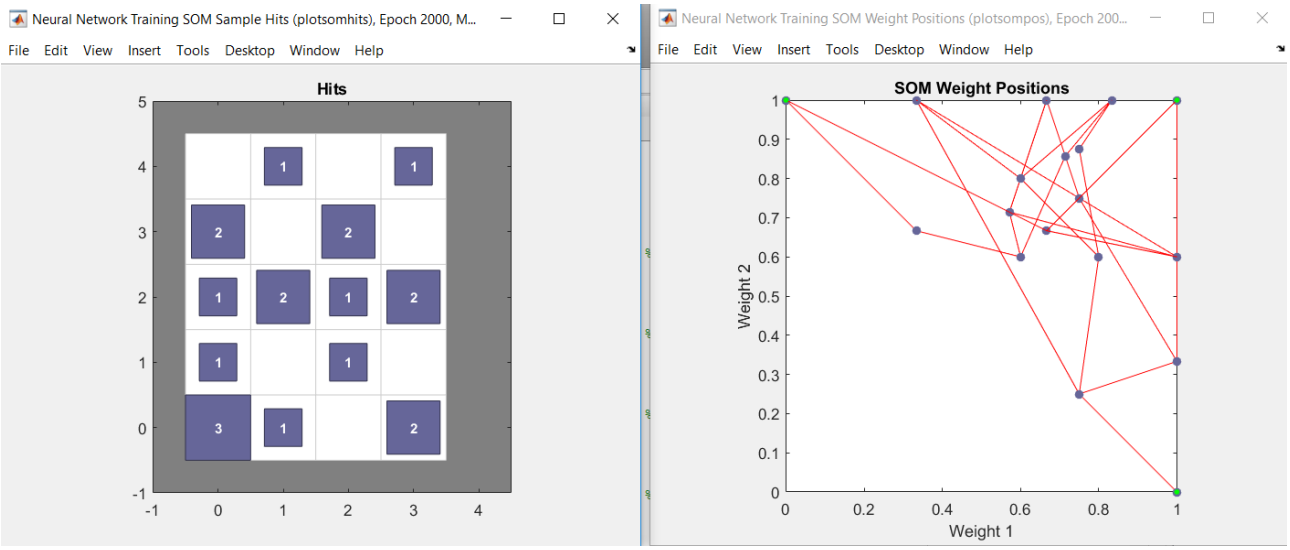
SOM Neighbor Distance:

- na tej figurze niebieskie sześciokąty reprezentują neurony.
- czerwone linie łączą sąsiednie neurony
- kolory w obszarach zawierających czerwone linie wskazują odległości między neuronami
- ciemniejsze kolory reprezentują większe odległości
- jaśniejsze kolory oznaczają mniejsze odległości
- ciemny segment znajduje się wyłącznie w środkowo dolnym regionie



SOM Input Planes:

Ta figura przedstawia płaszczyznę ciężkości dla każdego elementu wektora wejściowego (w tym przypadku 20). 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.



SOM Sample Hits:

Domyślna topologia SOM jest sześciokątna. Ta figura pokazuje lokalizacje neuronów w topologii i wskazuje, ile danych treningowych jest powiązanych z każdym z neuronów (centrami klastra).

SOM Weight Positions:

Dodatkowo samoorganizacja prowadzi do tego, że neurony „obstawiające” podobne obiekty wejściowe ulokowane są w warstwie topologicznej sieci jako neurony sąsiednie

5. Wnioski:

- Głównymi cechami odróżniającymi algorytm WTA od algorytmu WTM, jest to, że w WTA tylko jeden neuron może podlegać adaptacji w każdej iteracji, są algorytmami słabo zbieżnymi, szczególnie przy dużej liczbie neuronów. Natomiast algorytm WTM, który zastąpił w praktyce wcześniej omawiany algorytm, charakteryzuje się tym, że oprócz zwycięzców, również neurony z jego sąsiedztwa uaktualniają swoje wagi.

- W strukturze sieci Kohonena istotne jest to, że każdy neuron warstwy wejściowej komunikuje się z każdym neuronem warstwy topologicznej – natomiast neurony w warstwach nie komunikują się pomiędzy sobą.

- Algorytm WTM równomiernie rozłożył zwycięzców, inaczej niż algorytm WTA. Poprawność działania programu jest spowodowana tym, że zwycięzcy nie znajdują się w jednym miejscu tylko są „rozproszeni”. Skutkiem tego jest fakt, że sieć nie skupiała się na jednym fragmencie, ale na całej sieci.

- Ilość neuronów ma istotne znaczenie dla efektywności działania sieci. Mniejsza ilość neuronów skutkowała, tym że wagi neuronów mogą być do siebie bardzo zbliżone.

- Aby program działał poprawnie trzeba przyjąć odpowiednią liczbę sąsiadujących neuronów. Gdy jest ona zbyt duża program nie wyświetla poprawnych wyników.

- Zwiększenie ilości epok w których sieć musiała się nauczyć nie dawało lepszych wyników. Zapewne jest to spowodowane, tym że sieć uczy się bez nauczyciela, czyli sieć dłużej się uczy niż z nauczycielem. Aby zaobserwować znaczące wyniki trzeba by dać sieci bardzo dużo czasu na naukę.

6. Kod 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 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 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 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 1 0 1;
    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,distance
Fcn);
net.trainParam.epochs = 2000;

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