Akademia Górniczo – Hutnicza

WIMiIP Inżynieria Obliczeniowa

18.12.2017, Kraków

Podstawy Sztucznej Inteligencji

Budowa i działanie sieci Kohonena dla WTM

sprawozdanie z projektu 6:

Agnieszka Majkut

nr indeksu 286116

Wprowadzenie:

Celem projektu było poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTM do odwzorowania istotnych cech liter alfabetu.

Zadania do wykonania:

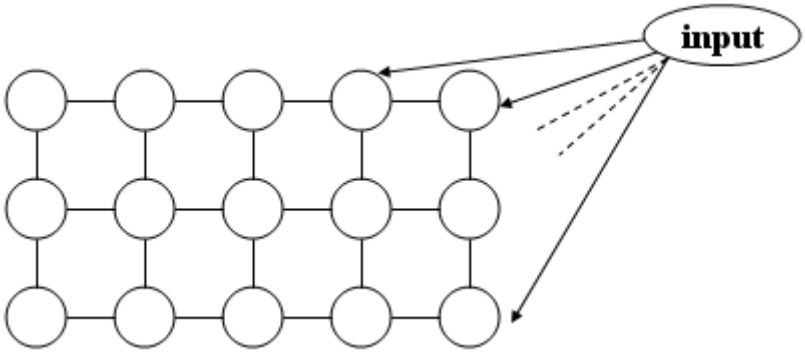
- wygenerowanie danych uczących i testujących
- przygotowanie sieci Kohonena i algorytmu opartego o regułę Winner Takes Most (WTM)
- uczenie sieci dla różnych współczynników uczenia
- testowanie sieci

Sieć neuronowa (sztuczna sieć neuronowa) – ogólna nazwa struktur matematycznych i ich programowych lub sprzętowych modeli, realizujących obliczenia lub przetwarzanie sygnałów poprzez rzędy elementów, zwanych sztucznymi neuronami, wykonujących pewną podstawową operację na swoim wejściu. Oryginalną inspiracją takiej struktury była budowa naturalnych neuronów, łączących je synaps, oraz układów nerwowych, w szczególności mózgu.

Uczenie sieci neuronowych - wymuszenie na niej określonej reakcji na zadane sygnały wejściowe. Uczenie jest konieczne tam, gdzie brak jest informacji doświadczalnych o powiązaniu wejścia z wyjściem lub jest ona niekompletna, co uniemożliwia szczegółowe zaprojektowanie sieci. Uczenie może być realizowane krok po kroku lub poprzez pojedynczy zapis. Istotnym czynnikiem przy uczeniu jest wybór odpowiedniej strategii (metody) uczenia. Wyróżnić możemy dwa podstawowe podejścia: uczenie z nauczycielem (supervised learning) i uczenie bez nauczyciela (unsupervised learning).

Sieć Kohonena - jest szczególnym przypadkiem algorytmu realizującego uczenie się bez nadzoru. Jej 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. Sieć Kohonena znana jest też pod nazwami Self-Organizing Maps, Competitive Filters.

Topologia sieci Kohonena odpowiada topologii docelowej przestrzeni. Jeśli np. chcemy prezentować wynik na ekranie, rozsądnym modelem jest prostokątna siatka wezłów (im więcej, tym wyższą rozdzielczość będzie miała mapa):



Topologia sieci Kohonena

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

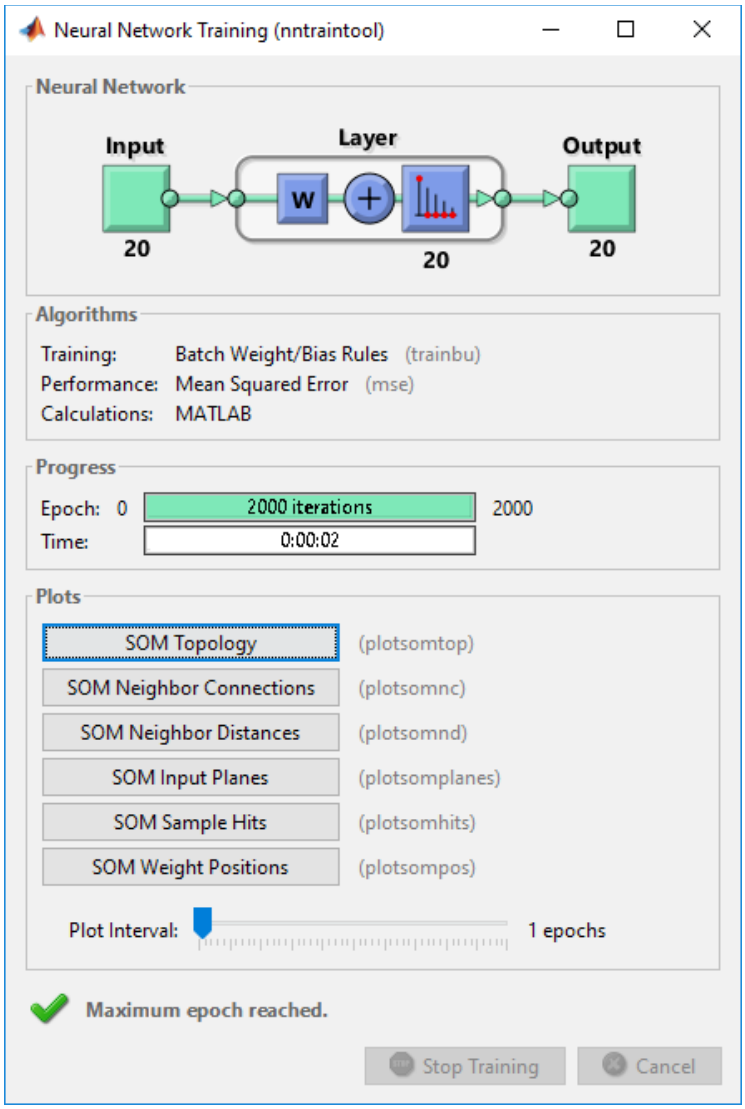
- Każdy węzeł oblicza swój poziom aktywacji jako iloczyn skalarny wektora wag i wektora wejściowego (podobnie jak w zwykłym neuronie)
- Ten węzeł, który dla danego wektora wejściowego ma najwyższy poziom aktywacji, zostaje zwycięzcą i jest uaktywniony
- Wzmacniamy podobieństwo węzła-zwycięzcy do aktualnych 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.

Winner Takes Most (WTM) - 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.

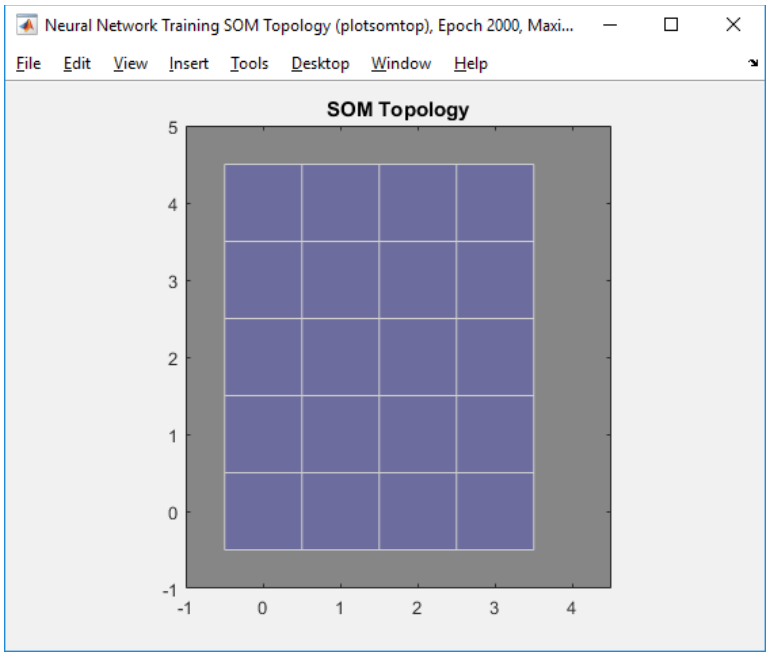
W programie Matlab, za pomocą biblioteki Neural Network Toolbox, zaimplementowałam sztuczną sieć neuronową. Danymi wejściowymi są litery A, B, C, D, E, F, G, H, I, J, K, L, N, O, P, R, S, T, U, Y są one reprezentowane za pomocą wartości binarnych (0 i 1) w tablicy o rozmiarze 4x5.

Otrzymane wyniki:

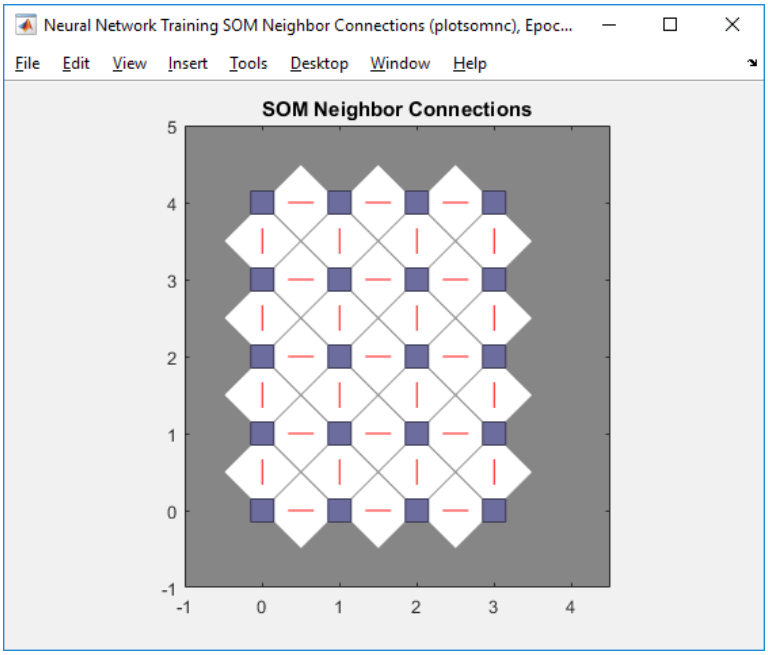


rys1. Wynik działania sieci

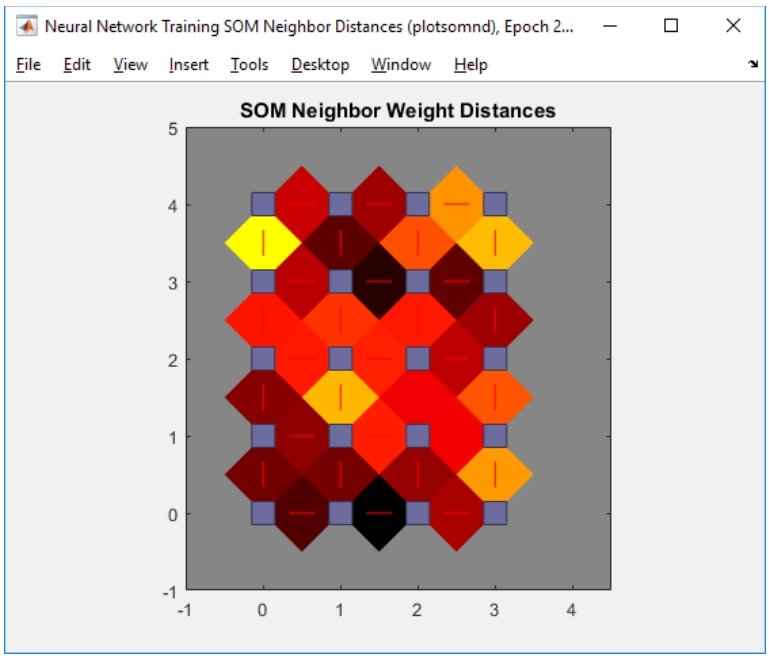
Wykorzystałam prostokątną siatkę neuronów. Uczenie sieci nastąpiło wg reuguły Kohonena oraz WTM. Program odwzorowuje istotne cech liter alfabetu na podstawie danych wejściowych.



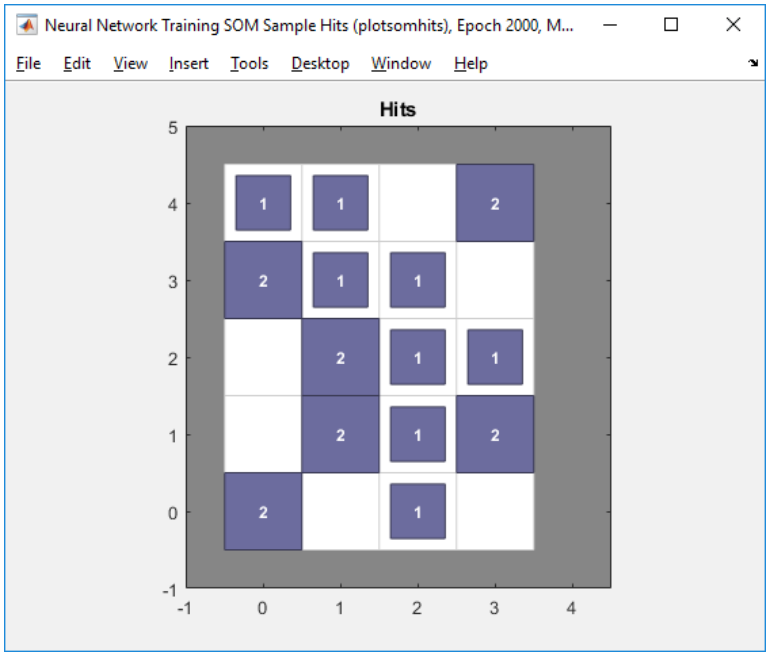
rys2. Topologia sieci



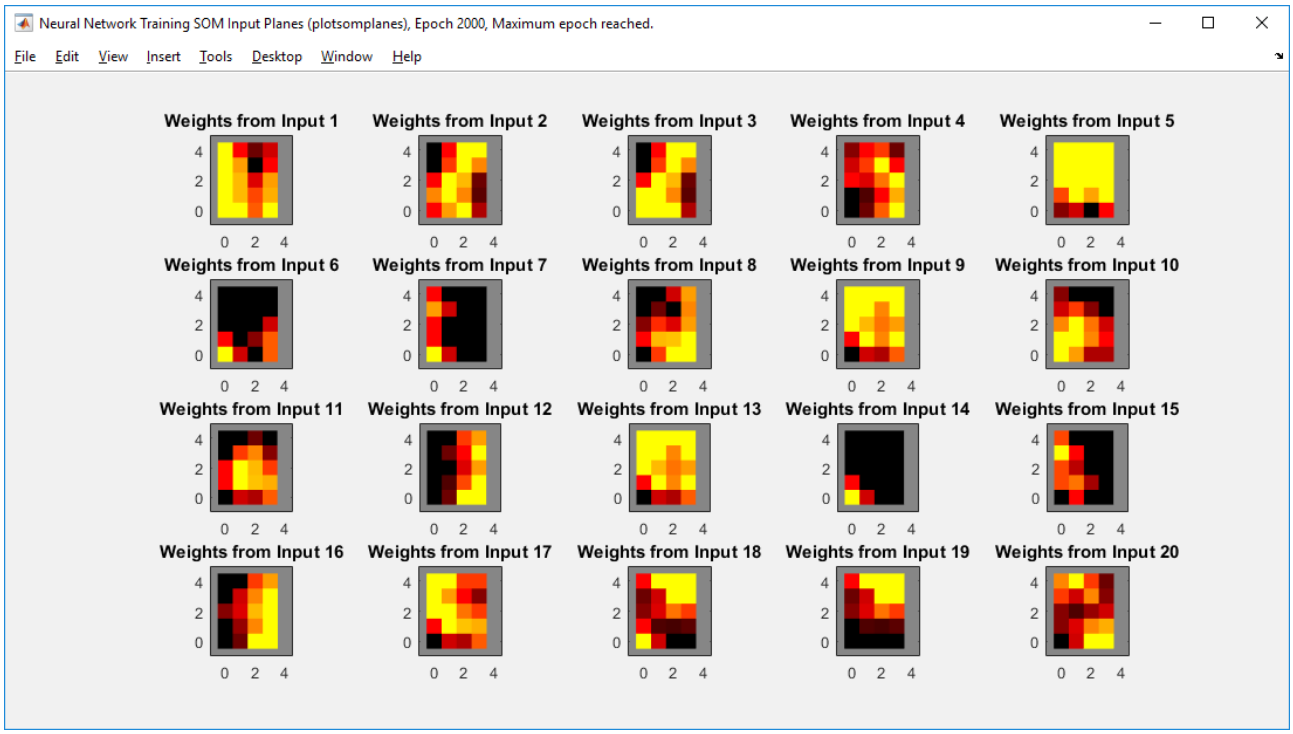
rys3. Połączenia pomiędzy sąsiadującymi neuronami



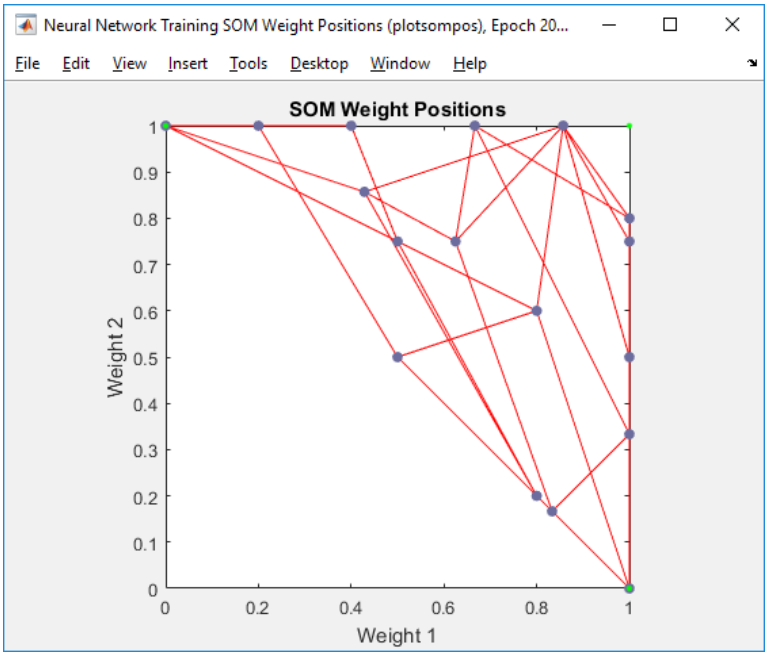
rys4. Dystans pomiędzy neuronami – im kolor ciemniejszy tym dystans jest większy



rys5. Wynik działania zasady WTM – obraz pokazuje ile razy zwyciężyły poszczególne neurony podczas rywalizacji



rys6. Rozkład wag dla każdego z wejść – kolor ciemniejszy oznacza wyższe wagi



rys8. Efekt końcowy – fiotetowe kropki to neurony, a czerwone linie to połączenia pomiędzy nimi

Na wykresie obserwujemy rozkład wag dla stworzonej sieci neuronowej.

Wnioski:

Na rys5 można zauważyć, że algorytm WTM równomiernie rozłożył zwyciężców, inaczej niż algorytm WTA. Poprawność działania programu jest spowodowana tym, że zwycieżcy 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 na istotne znaczenie dla efektywności dzialania sieci. Mniejsza ilość neuronów skutkowała, tym że wagi neuronów mogą być do siebie bardzo zbliżone.

Na rys6 możemy zauważyć jak wygląda rozkład wag dla poszczególnych wejść. Dla każego z nich wygląda inaczej, ponieważ każda liczba, choć są do siebie podobne to mają pewne różnice, które można zauważyć na rysunku.

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

Metoda WTA działa lepiej niż WTA. Gdy mamy do czynienia ze złożonymi sieciami z duża ilością neuronów dla lepszej efektywności należy zastosować metodę WTM. Ale efektywność zależy też od jakości danych jakie dostarczamy do danych uczących.

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 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,distanceFcn);
net.trainParam.epochs = 2000;

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

Cały kod znajduje się również na Githubie.