# Kamil Wieczorek

Kierunek: Informatyka
Specjalizacja: PSI
Przedmiot: Systemy uczące się
kamil.wieczorek1@edu.uekat.pl

# Wstęp

W ramach projekty zaimplementowano sieć Kohonena (SOM) w celu rozwiązania problemu transportowego TSP. Algorytm został napisany w języku Python bez zaawansowanych bibliotek (oferujących gotowe implementacje). Program na wejściu otrzymuje listę współrzędnych geograficznych (długość i szerokość) miast, które mają zostać odwiedzone. Na wyjściu zwracana jest wizualizacja z optymalnym przebiegiem trasy.

# Opis teoretyczny

Sieci Kohonena są rodzajem sztucznej sieci neuronowej realizującej uczenie nienadzorowane, które pozwalają na analizę danych i ich klastrowanie. Zwane również również mapą Kohonena lub samoorganizującą się mapą (Self-Organizing Maps). Sieć składa się z warstwy neuronów, które są połączone z wejściową warstwą danych. W procesie uczenia, neurony są tak modyfikowane, aby ich reprezentacje odpowiadały charakterystyce danych wejściowych. Algorytmy SOM, oparte na sieciach Kohonena, znajdują zastosowanie w wielu dziedzinach, w tym w rozwiązaniu problemu komiwojażera (Travelling Salesman Problem). TSP polega on na znalezieniu najkrótszej ścieżki, która pozwoli odwiedzić wszystkie wierzchołki grafu (miasta) z danego zbioru dokładnie jeden raz, zaczynając i kończąc podróż w tym samym punkcie. W celu wykorzystania sieci Kohonena do rozwiązania takiego problemu każdy punkt na mapie SOM jest przypisywany do najbliższego wierzchołka grafu, a następnie wyznaczana jest trasa, która przechodzi przez te wierzchołki w kolejności określonej przez ich przypisanie na mapie Kohonena. Dzięki takiemu podejściu, możliwe jest znalezienie efektywnego rozwiązania problemu TSP

# Rozwiązanie

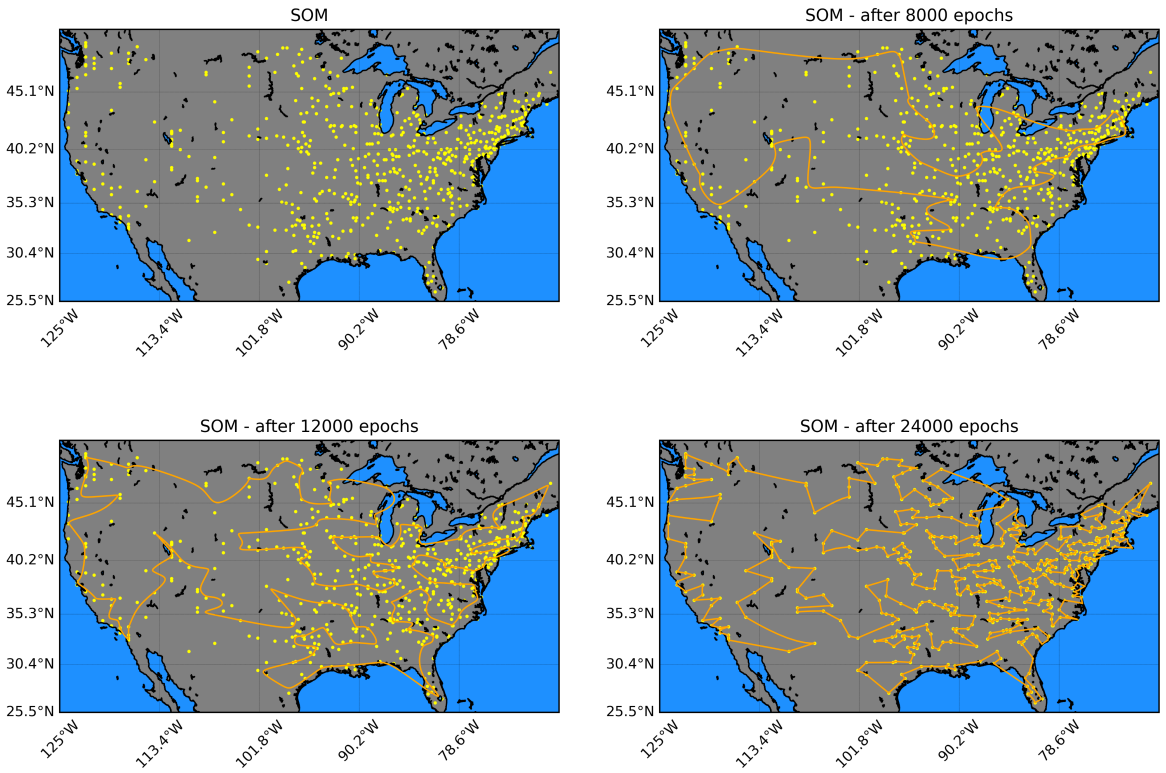
Napisany w języku Python program implementuje algorytm uczenia sieci Kohonena, a jego kroki w uproszczeniu wyglądają następująco:

1. Inicjalizacja wag sieci:
    - Losowe wartości wag początkowych są przypisywane do każdej jednostki sieci.
2. Przypisanie wejścia do najbliższej jednostki sieci:

- Dla każdego wejścia obliczana jest odległość Euklidesowa od każdej jednostki sieci.
- Jednostka sieci, której wagi są najbliższe wejściu, zostaje wybrana jako zwycięzca.

3. Aktualizacja wag zwycięzcy i jego sąsiadów:
   - Wagi zwycięzcy i jego sąsiadów są dostosowywane do wejścia.
   - Zwycięzca otrzymuje większą zmianę wag niż jego sąsiedzi, którzy znajdują się dalej od zwycięzcy.
4. Powtarzanie procesu uczenia:
   - Kroki 2-3 są powtarzane dla każdego wejścia w danych treningowych.

Jak wspomniano na wstępie, wynikiem działania programu jest wizualizacja trasy odpowiednio w każdej określonej iteracji uczenia (epoce), aż do satysfakcjonującego wyniku. Poniżej zaprezentowano efekt działania programu w formie progresu dopasowywania się trasy do zadanych punktów wejściowych (współrzędnych 600 miast w USA).




*Rysunek 1. Efekt działania programu. Samoorganizująca się trasa dla zadanych punktów wejściowych*

# Kod programu

Cały kod stworzony w ramach projektu dostępny jest na GitHub'ie autora: https://github.com/vieczorkamil/SOM-Kohonen-neural-network, a także udostępniony na platformie classroom.