TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

ARTIFICIAL INTELLIGENCE (CT653)

PROJECT REPORT

ON

# SELF ORGANIZING MAP

**SUBMITTED BY:**

PRAJJWAL PANDEY (PUL077BCT055)

REETWIZ AMATYA (PUL077BCT067)

PRADEEP BHATTARAI(PUL077BCT100)

**SUBMITTED TO:**

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

March, 2024

# Acknowledgments

We would like to express our sincere appreciation to Dr.Basanta Joshi and the AI Lab Instructors at the Department of Electronics and Computer Engineering, Pulchowk Campus, for their invaluable support and guidance throughout the development of this AI project report. Their expertise and resources have been instrumental in shaping the proper direction of our project.

In conclusion, we want to thank all the teachers in the department who have shared their knowledge and expertise, helping us build a strong foundation in Electronics and Computer Engineering. Their dedication to teaching and commitment to excellence have helped us understand the subjects better.

# Abstract

This work shows how a modified Kohonen Self-Organizing Map with two dimensional neighborhood is used to approach the symmetrical Traveling Salesperson Problem (TSP). Solution generated by the Kohonen network is improved by the 2-opt algorithm. The paper describes briefly self-organization in neural networks, 2opt algorithm and modifications applied to Self-Organizing Map.

# Table of Contents

# List of Figures

# 1. Introduction

## 1.1 Problem statements

The Traveling Salesman Problem (TSP) is a classic optimization problem in which the goal is to find the shortest possible route that visits each city exactly once and returns to the original city.

## 1.2 Objectives

- **Implement a Kohonen SOM:**
  - Define Network Architecture: Create a two-dimensional grid of neurons.
  - Weight Initialization: Randomly initialize weights of neurons, ensuring normalization.
  - Training Process: Implement SOM training process using TSP dataset, including learning rate schedule and neighborhood function.
- **Incorporate Two-Dimensional Neighborhood:**
  - Modify Weight Update Rule: Adjust weight update rule to incorporate two-dimensional neighborhood function.
- **Integrate 2-opt Algorithm:**
  - Implement 2-opt Algorithm: Develop efficient implementation to optimize SOM solution.
  - Integration with SOM: Integrate 2-opt algorithm with SOM solution to further refine TSP solution.

# 2. Literature Review

In 1975 Teuvo Kohonen introduced a new type of neural network that uses competitive, unsupervised learning [1]. This approach is based on Winner Takes All (WTA) and Winner Takes Most (WTM) algorithms. Therefore, these algorithms will be explained here briefly. The most basic competitive learning algorithm is WTA. When an input vector (a pattern) is presented, the distance to each neuron's synaptic weights is calculated. The neuron whose weights are most correlated to the current input vector is the winner. Correlation is equal to the scalar product of the input vector and the considered synaptic weights. Only the winning neuron modifies its synaptic weights to the point presented by the input pattern. The learning process can be described by the following equation:

$$W_i \leftarrow W_i + \eta \cdot (x - W_i)$$

where $i \in [0, \text{number of neurons}]$, $W_i$ represents all synaptic weights of the winning neuron, $\eta$ is the learning rate, and $x$ stands for the current input vector. This simple algorithm can be extended. The most common extension is giving more chance of winning to neurons that are rarely activated. However, the WTM strategy has better convergence than WTA. The difference between those two algorithms is that many neurons in WTM strategy adapt their synaptic weights in one learning iteration. In this case, not only the winner but also its neighborhood adapts. The farther the neighboring neuron is from the winner, the smaller modification is applied to its weights. This adaptation process can be described as:

$$W_i \leftarrow W_i + \eta \cdot N(i, x) \cdot (x - W_i)$$

for all neurons $i$ that belong to the winner's neighborhood. $W_i$ stands for synaptic weights of neuron $i$, $x$ is the current input vector, $\eta$ stands for the learning rate, and $N(i, x)$ is a function that defines the neighborhood. A classical Self-Organizing Map (SOM) can be created when the function $N(i, x)$ is defined as:

$$N(i, x) = \begin{cases} 1 & \text{for others } d(i, w) \leq \lambda \\ 0 & \text{otherwise} \end{cases}$$

where $d(i, w)$ is the Euclidean distance between the winning and $i$-th neuron, and $\lambda$ is the neighborhood radius. To train Kohonen SOM, the Euclidean distance between the input vector and all neural weights has to be calculated. The neuron that has the shortest distance to the input vector (the winner) is chosen, and its weights are slightly modified to the direction represented by the input vector. Then neighboring neurons are taken, and their weights are modified in the same direction. $\eta$ and $\lambda$ are multiplied with $\Delta\eta$ and $\Delta\lambda$ respectively during each learning iteration. These two last parameters are always less than one. Therefore, $\eta$ and $\lambda$ become smaller during the learning process. At the beginning, SOM tries to organize itself globally and with following iterations it performs more and more local organization, because learning rate and neighborhood get smaller.

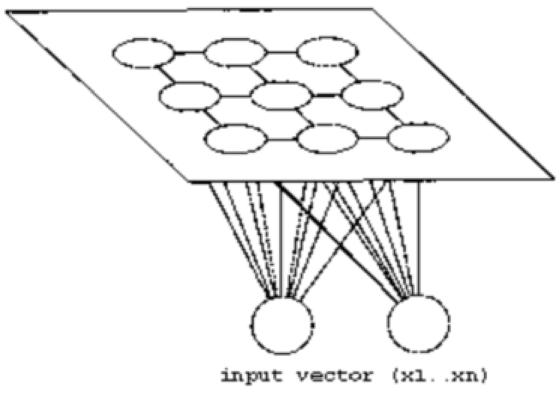


Figure 2.1: Kohonen SOM with two-dimensional neighborhood and input vector.

Kohonen SOM is shown in Figure 1. It maps input vectors of any dimension onto a map with one, two, or more dimensions. Input patterns, which are similar to one another in the input space, are put close to one another on the map. The input vector is passed to every neuron. A Kohonen SOM is made of a vector or matrix of output neurons. If a vector representation is chosen, each neuron has two neighbors (on the left and on the right). It is called one-dimensional neighborhood:

If a two-dimensional matrix representation is used, neurons have four neighbors (left, right, top, and bottom). This is classical two-dimensional neighborhood:

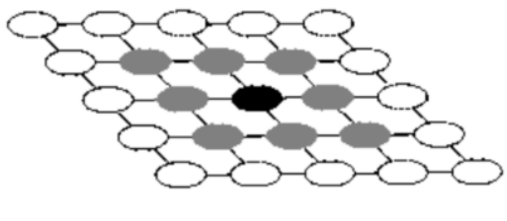


Figure 2.2: Extended two-dimensional neighborhood of Kohonen SOM

The neighborhood can be expanded. Instead of taking four nearest neurons, 8 or more can be taken.

As many dimensions can be used as needed: 1D, 2D, 3D, or more. However, 2D neighborhood is most common.

# 3. Algorithm for SOM

**Algorithm 1** Kohonen Self Organizing Map

```
0: procedure TRAINSOM(iteration_number)
0:   Randomize weights for all neurons
     for $i = 1$ to $iteration\_number$ do
0:
     Take one random input pattern
0:   Find the winning neuron
0:   Find neighbors of the winner
0:   Modify synaptic weights of these neurons
0:   Reduce the learning rate ($\eta$) and neighborhood radius ($\lambda$)
0:
0: end procedure=0
```

Explanation:

- **Initialization:**
  - Randomize weights for all neurons.
- **Training Loop:**
  - For each iteration from 1 to the specified number:
    - Take one random input pattern.
    - Find the winning neuron.
    - Find neighbors of the winner.
    - Modify synaptic weights of these neurons.
    - Reduce the learning rate ($\eta$) and neighborhood radius ($\lambda$).

## 3.1 Experiments on self-organization

Most interesting results of self-organization can be achieved in networks that have two dimensional input vector and two dimensional neighborhood. In this case input to network consists of two values: x and y, which represent a point in two dimensional space. This kind of network can map two dimensional objects in such a way that a mesh which covers this object is created. This process is illustrated in Figure 5. Each example consists of six squares. First one shows object that should be learned. Second square illustrates network just after randomization of all neural weights. Following squares describe learning process. Please note that each neuron (a circle) represents a point whose coordinates are equal to neuron's weights.

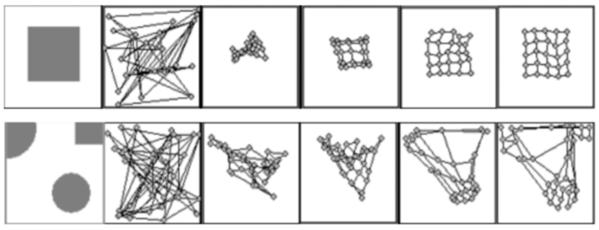


Figure 3.1: Self-organization of a network with two dimensional neighborhood.

These figures illustrate that Kohonen neural network is a powerful self-organizing and clustering tool. However, it is also possible to create a network with one dimensional neighborhood and two dimensional input. Learning process of this is shown in Figure 6. It can be observed that this network tries to organize it's neurons in such a way, that a

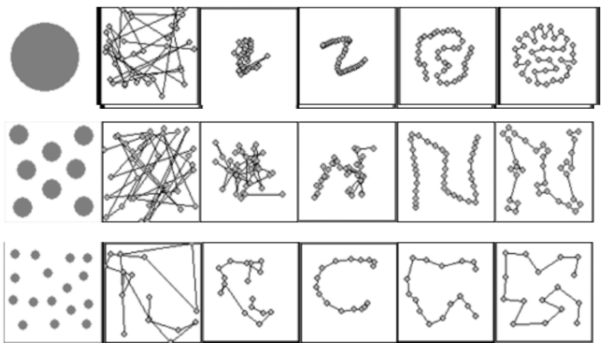


Figure 3.2: Self-organization of a network with one dimensional neighborhood.

relatively short route between all neurons emerges. These experiments were a stimulus to build a system based on Kohonen one dimensional SOM that would solve TSP problems.

# 4. SOM based TSP solver

To solve TSP problem a one dimensional network must be created. Number of neurons must be equal to the number of cities. If the weights of a neuron are equal to some city's coordinates this neuron represents that city. In other words a neuron and a city are assigned to each other. All neurons are organized in a vector. This vector represents sequence of cities that must be visited. Unfortunately, Kohonen SOM without some modifications is unable to solve TSP. The reason for this is that if neural weights are taken as coordinates of cities, they may not equal the coordinates of cities that were given. To solve the problem an algorithm that would modify Kohonen solution to one that is valid has been created. Positions of cities and positions of neurons may not equal. However, adequate neural weights and cities' coordinates are very close to each other. An algorithm that modifies neural weights so they equal to cities' coordinates has been applied.

```
procedure repair
begin
Iterate through all neurons
begin
nearest_city = find the nearest city to current neuron
if (nearest_city is not assigned to any neuron)
assign nearest_city and current neuron
else
delete this neuron
end
Iterate through all cities
begin
if (current city is not assigned to any neuron)
begin
create a new_neuron and assign it to current city
nearest_neuron = find the nearest neuron to current city
insert new_neuron before or after nearest_neuron, depending
on which tour is locally shorter
end
end
end
```

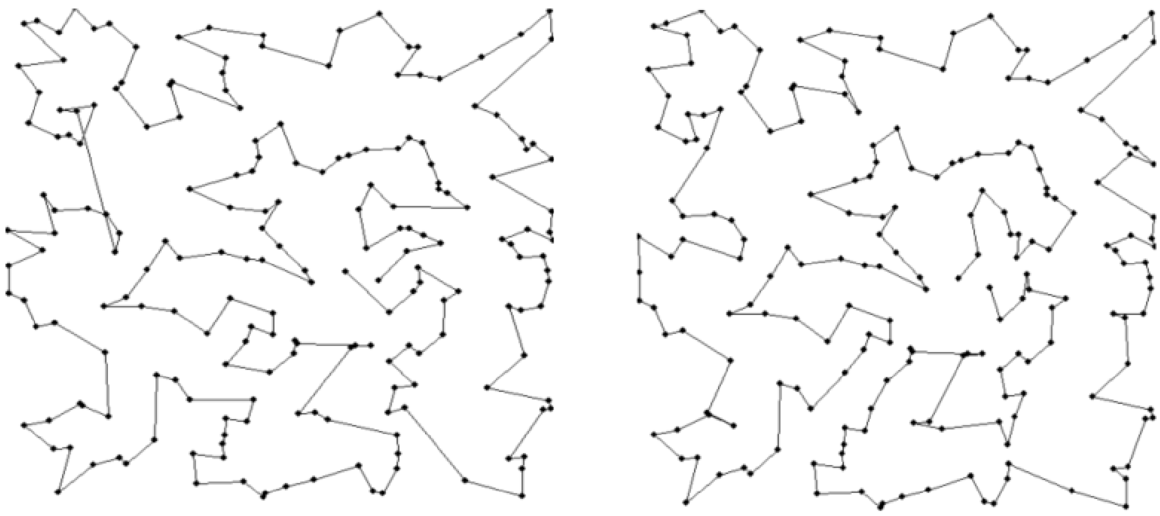


Figure 4.1: SOM solution without 2-opt optimization (left). There are two local loops on the left. First and last neuron can be seen in the middle. They are not connected it the picture, but distance between them is also computed. The same solution improved by 2-opt (right). Loops on the left have been erased. Additional changes can be observed.

HostelHub is being built as a MERN (MongoDB, Express.js, React, Node.js) stack application. The backend, powered by Node.js and Express.js, handles data storage, retrieval, and business logic through RESTful APIs, providing a seamless interaction between the frontend and the database, which is managed by MongoDB. The frontend, developed using React, offers a user-friendly interface with features such as authentication, dashboard views, and interactive components, ensuring a smooth user experience.

## 4.1 Theoretical Overview of Clustering and Unsupervised Learning

## 4.2 Introduction

Clustering and unsupervised learning are fundamental concepts in machine learning and data analysis. They play a crucial role in discovering patterns, structures, and relationships within unlabeled datasets. While supervised learning requires labeled data for training, unsupervised learning methods, including clustering, can operate on raw, unlabeled data. This makes unsupervised learning particularly valuable in scenarios where labeled data is scarce or costly to obtain.

## 4.3 Clustering

Clustering is a process of grouping similar data points into clusters, where points within the same cluster are more similar to each other than to those in other clusters. The objective of clustering is to partition the data in such a way that data points within the same cluster exhibit high intra-cluster similarity and low inter-cluster similarity. Clustering algorithms are widely used in various fields, including pattern recognition, image analysis, and customer segmentation.

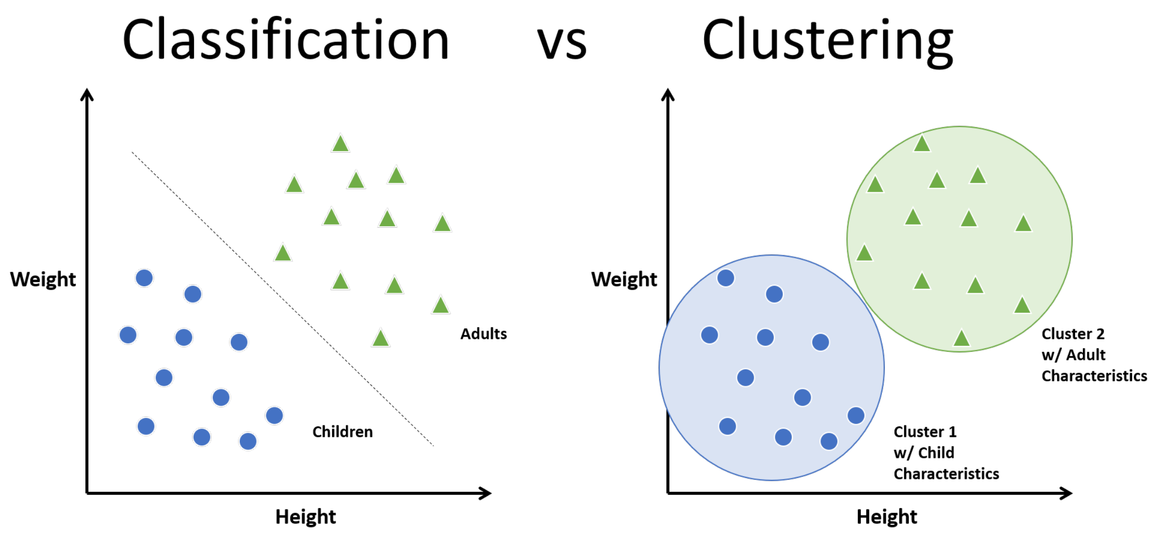


Figure 4.2: Clustering

### 4.3.1 Definition

Clustering is an unsupervised learning technique that aims to organize data points into groups based on their similarities. In a clustering task, the algorithm groups data points into clusters such that points within the same cluster are more similar to each other than to those in other clusters. The goal is to discover underlying patterns and structures in the data without prior knowledge of the groupings.

### 4.3.2 Types of Clustering Algorithms

There are several types of clustering algorithms, each with its own characteristics and suitability for different types of data. K-means clustering is one of the most popular partitioning algorithms, where the data is partitioned into a predefined number of clusters based on the mean of the data points. Hierarchical clustering, on the other hand, builds a hierarchy of clusters by recursively merging or splitting clusters based on a distance metric. Density-based clustering algorithms, such as DBSCAN, identify clusters as regions of high density separated by regions of low density.

## 4.4 Unsupervised Learning

Unsupervised learning is a type of machine learning where the model is trained on unlabeled data. Unlike supervised learning, where the model learns from labeled examples, unsupervised learning algorithms operate on raw, unstructured data and seek to discover patterns or structures within the data without explicit guidance. Clustering is one of the most common tasks in unsupervised learning, but other tasks such as dimensionality reduction and density estimation also fall under the umbrella of unsupervised learning.

### 4.4.1 Definition

Unsupervised learning is a machine learning paradigm where the model is trained on a dataset without any explicit supervision or labeled targets. The goal of unsupervised learning is to extract meaningful information or patterns from the input data without the need for labeled examples. Unlike supervised learning, where the model learns to predict outputs based on labeled inputs, unsupervised learning algorithms focus on uncovering hidden structures or relationships within the data.

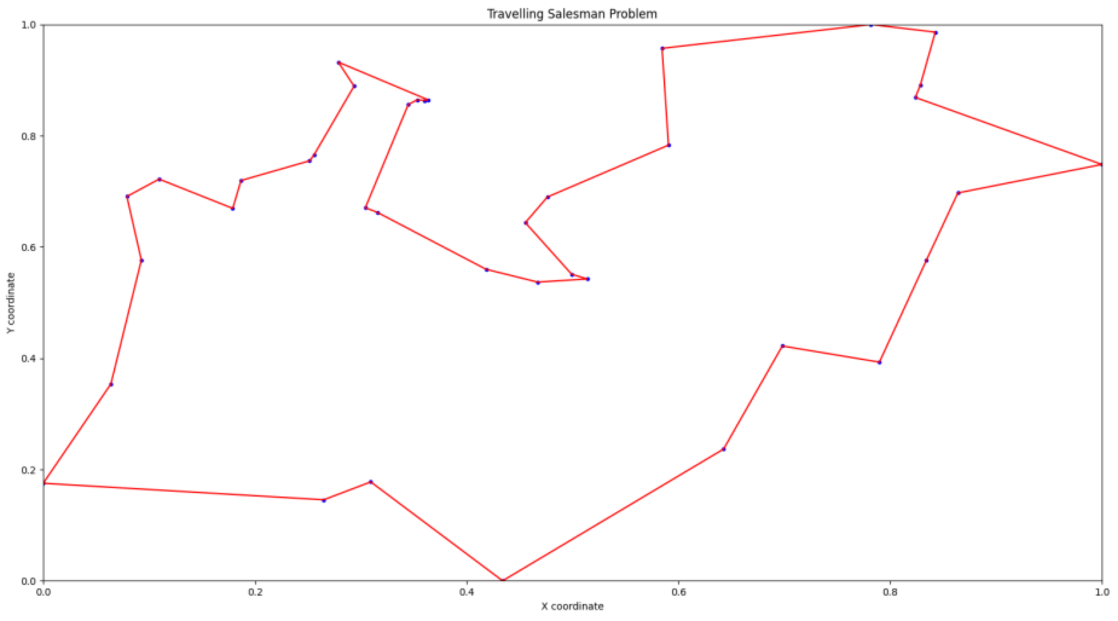


Figure 4.3: Unsupervised Learning

### 4.4.2 Internal Evaluation

Internal evaluation metrics assess the quality of clustering results based solely on the characteristics of the data and the clustering structure itself. Examples of internal evaluation metrics include the silhouette score, Davies–Bouldin index, and Dunn index. These metrics measure properties such as compactness, separation, and cohesion of clusters without reference to external labels.

### 4.4.3 External Evaluation

External evaluation metrics compare clustering results against known ground truth labels or reference partitions, if available. Examples of external evaluation metrics include purity, Rand index, and adjusted mutual information (AMI).

# 5. Results

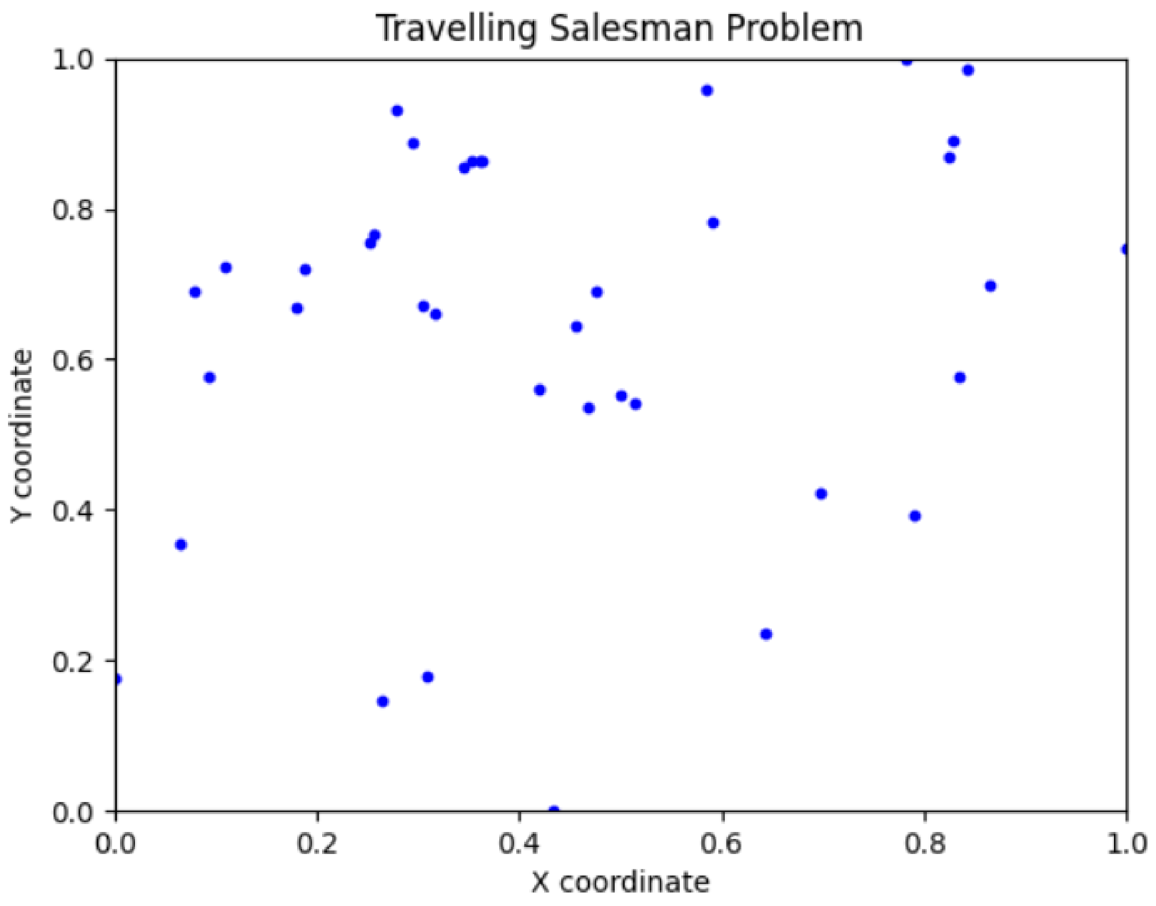


Figure 5.1: Starting cities displayed as Neurons

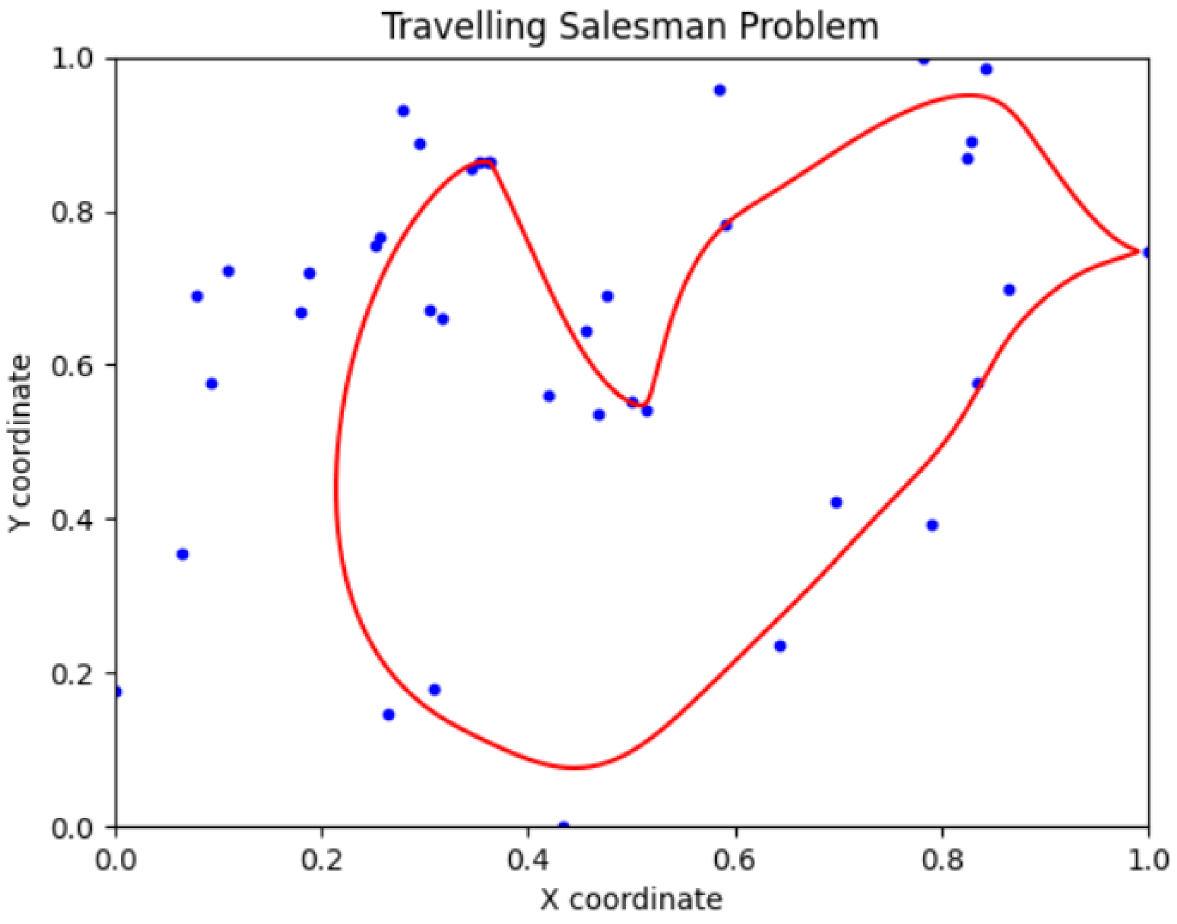


Figure 5.2: Process of Competition in SOM

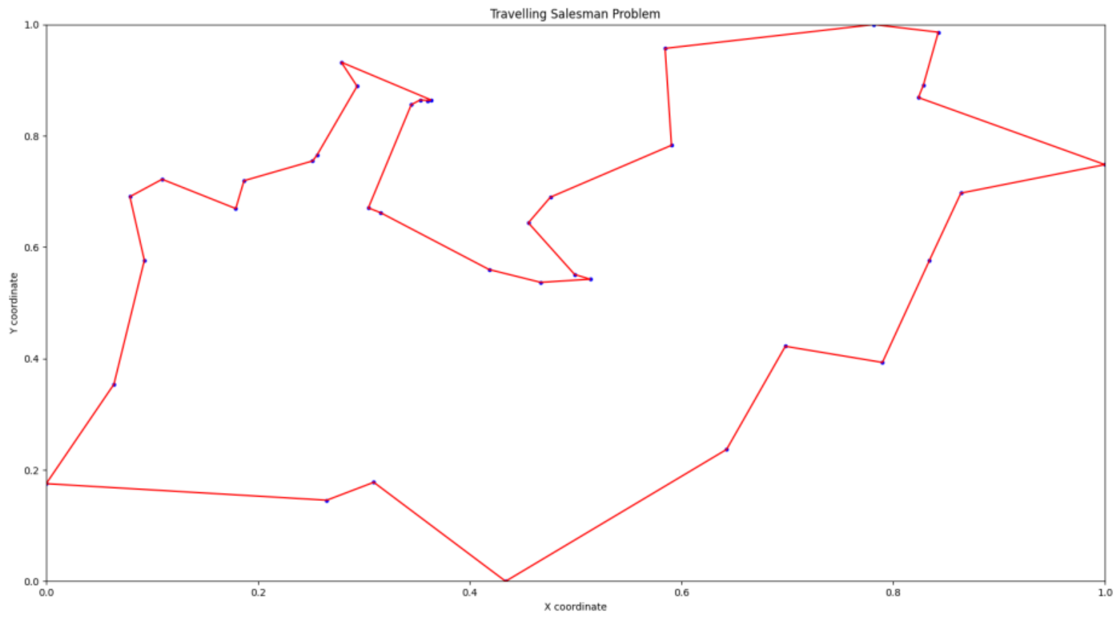
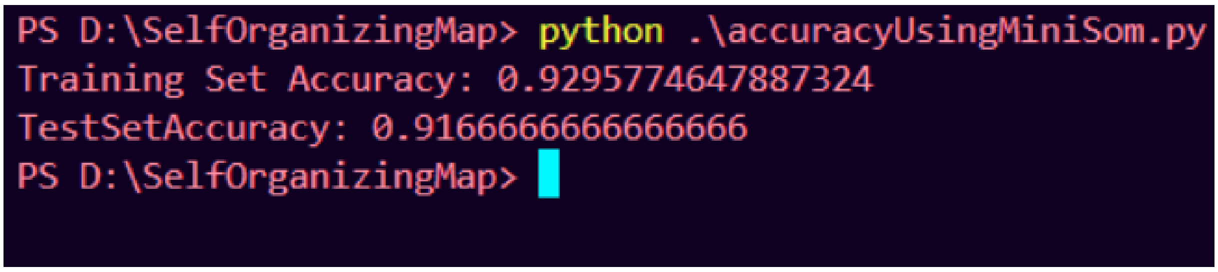


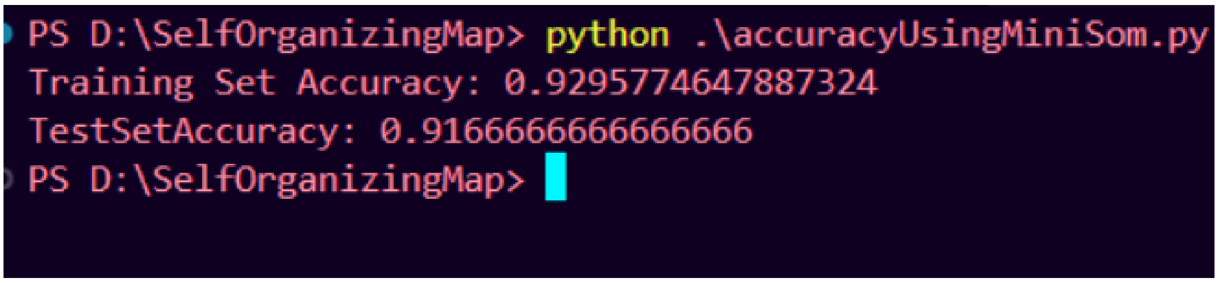
Figure 5.3: Final Mapping of Neurons(Cities) in a 2-Dimensional graph using Kohonen Self Organizing Map

# 6. Testing



```
PS D:\SelfOrganizingMap> python .\accuracyUsingMiniSom.py
Training Set Accuracy: 0.9295774647887324
TestSetAccuracy: 0.9166666666666666
PS D:\SelfOrganizingMap>
```

Figure 6.1: Testing of our implementation of our SOM algorithm on WINE dataset



```
PS D:\SelfOrganizingMap> python .\accuracyUsingMiniSom.py
Training Set Accuracy: 0.9295774647887324
TestSetAccuracy: 0.9166666666666666
PS D:\SelfOrganizingMap>
```

Figure 6.2: Final Accuracy

We compared our implementation of SOM algorithm to the test data set and obtained a reasonable accuracy of 91 percent. For the solution of Travelling Salesman Problem we used the following dataset:

**Dataset Name:** dj38

**Type:** TSP

**Dimension:** 38

**Edge Weight Type:** EUC2D

**Node Coordinates:** The dataset includes the node coordinates for each location in Djibouti

# 7. Applications of SOM

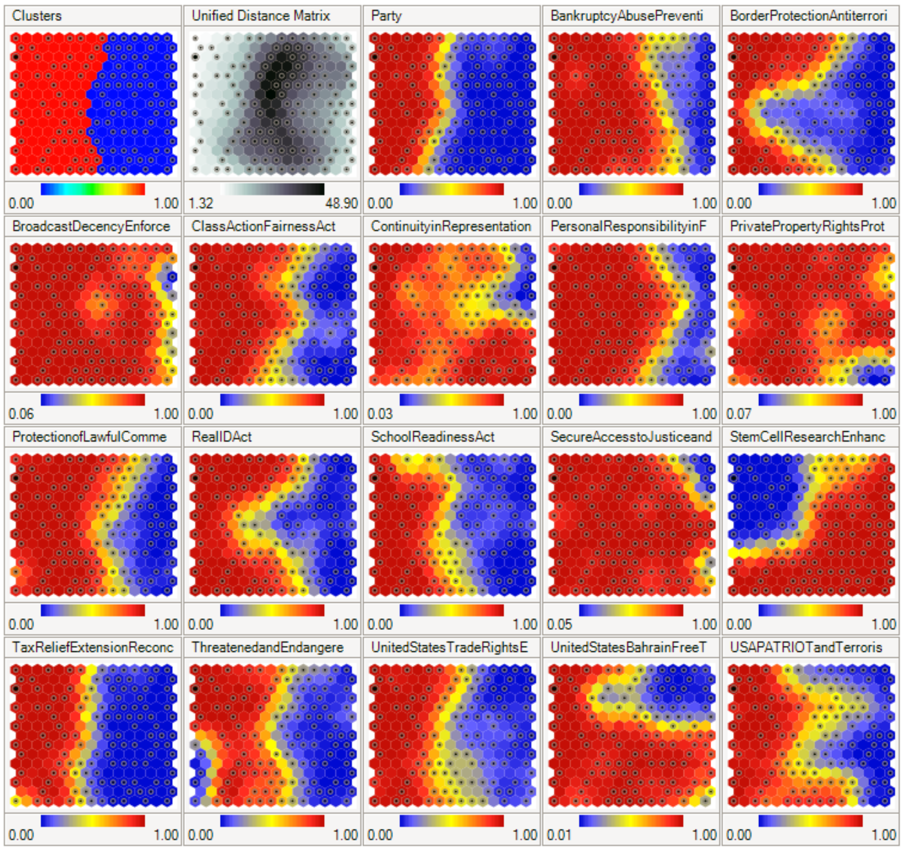


Figure 7.1: Applications of SOM

- **Pattern Recognition**: Self-Organizing Maps are widely used for pattern recognition tasks in various fields such as image processing, speech recognition, and bioinformatics.
- **Data Visualization**: SOMs are effective tools for visualizing high-dimensional data in a lower-dimensional space. By organizing data points on a map based on their similarities,
- **Customer Segmentation**: SOMs are extensively used in marketing and customer relationship management for segmenting customers based on their purchasing behavior, preferences, and demographics. By clustering customers into distinct groups.
- **Anomaly Detection**: SOMs can be employed for anomaly detection tasks, where the goal is to identify rare or abnormal instances in a dataset that deviate from the typical patterns.

# 8. Conclusion

Based on our results there are a couple of things that can be optimized. Here are some of them:

- An optimal network parameter settings should be found ($\eta$, $\Delta\eta$, $\Delta\lambda$, number of iterations).
- Experiments with other self-organizing networks should be performed, Gaussian neighborhood and "conscience mechanism" may be applied. Conscience mechanism can improve TSP solutions generated by neural networks, as reported in [2].
- 2-opt algorithm is not very sophisticated. Some other optimization method may be better.

# 9. References

1. Kohonen T. (2001), Self-Organizing Maps, Springer, Berlin
2. Burke Laura I., (1993), Neural Methods for the Traveling Salesman Problem: Insights From Operations Research