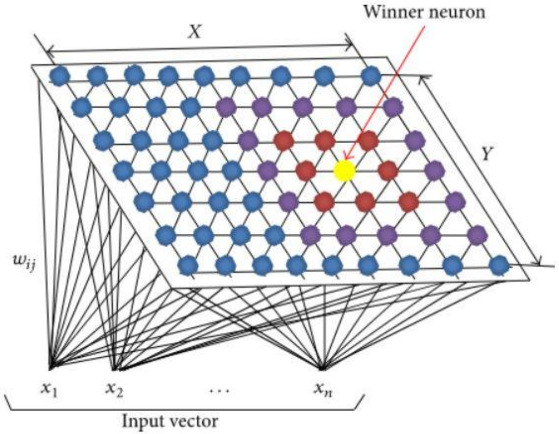
Self Organizing Map



Prajjwal Pandey (077BCT055)

Pradeep Bhattarai (077BCT100)

Reetwiz Amatya (077BCT067)

Table of contents

MACHINE LEARNING

A branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

Types Of Machine Learning

- **Supervised Learning**: Training machines on a labelled dataset for output prediction
- **Unsupervised Learning:** Training machine on an unlabelled dataset and predicts output without any supervision.
- **Reinforcement Learning:** Works on feedback process.

Unsupervised Learning

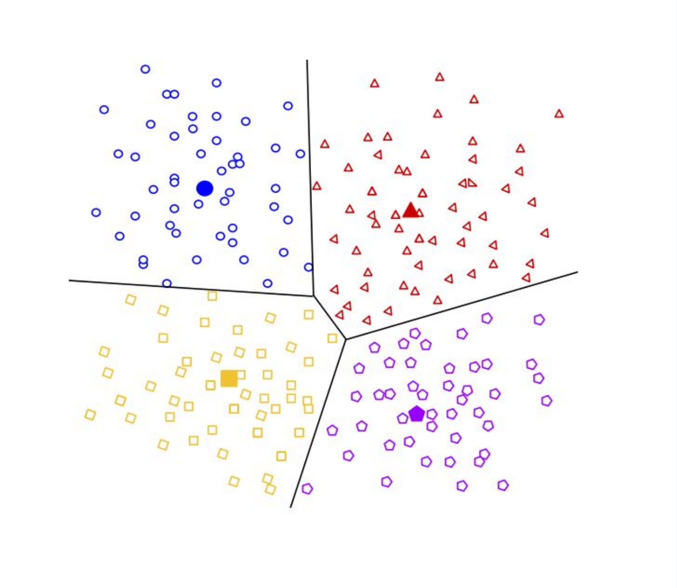
Main aim of grouping or categorizing the unsorted dataset according to the similarities, patterns, and differences.

- **Clustering**
- **Association**

CLUSTERING

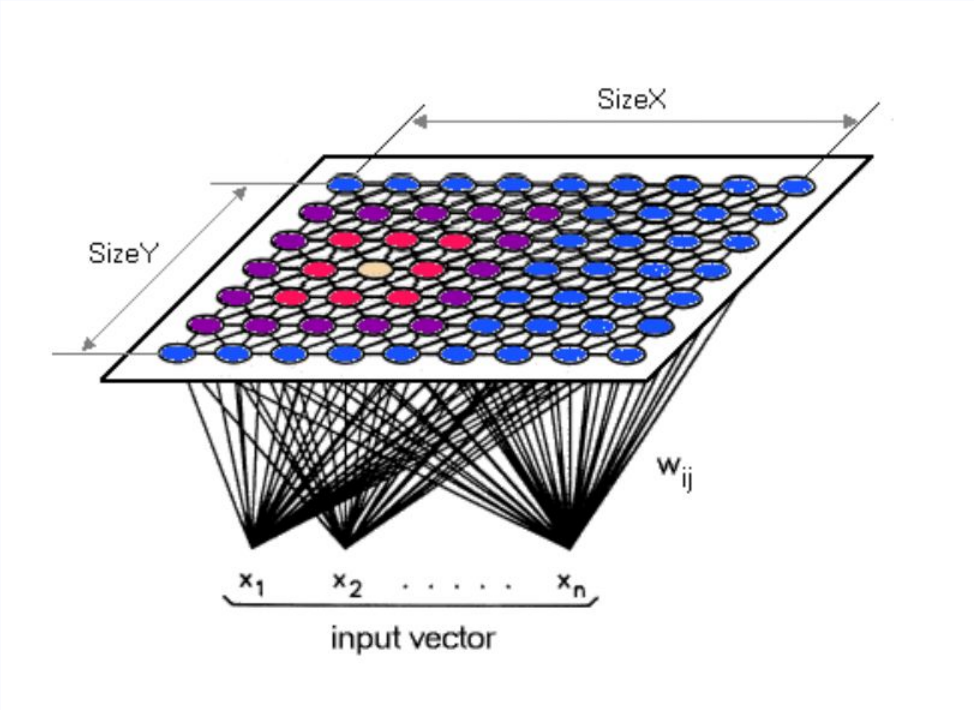
Clustering is the process of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups.

- Self Organized Maps
- K Means Clustering
- Singular Value Decomposition
- Principal Component Analysis



SELF ORGANIZING MAPS

Data visualization machine learning technique



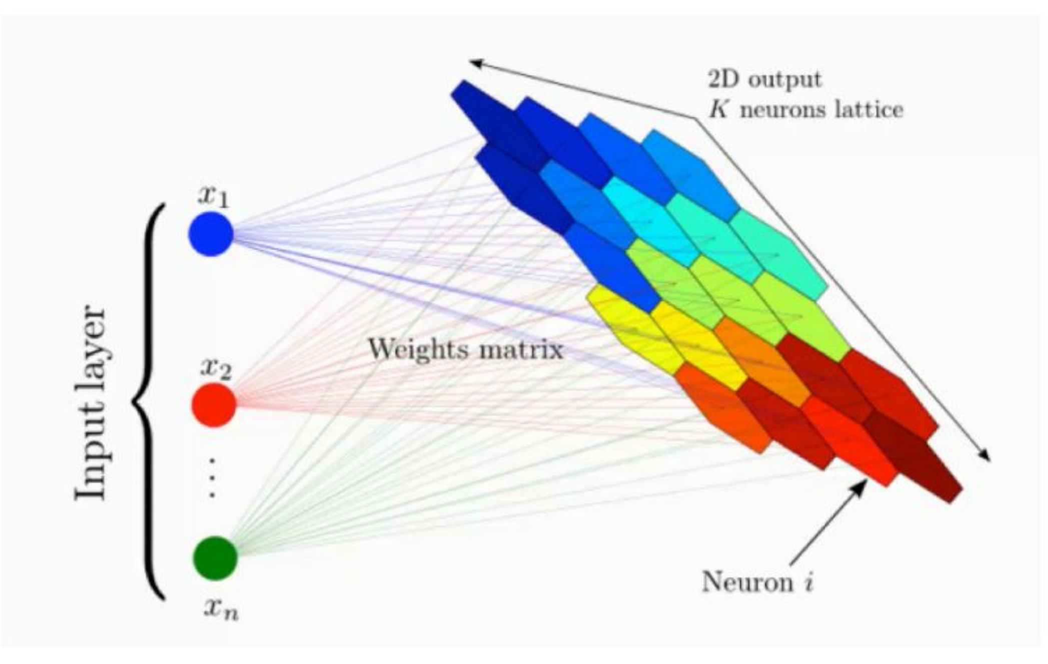
TYPES OF SELF ORGANIZING MAPS

Von-Der Malsburg model

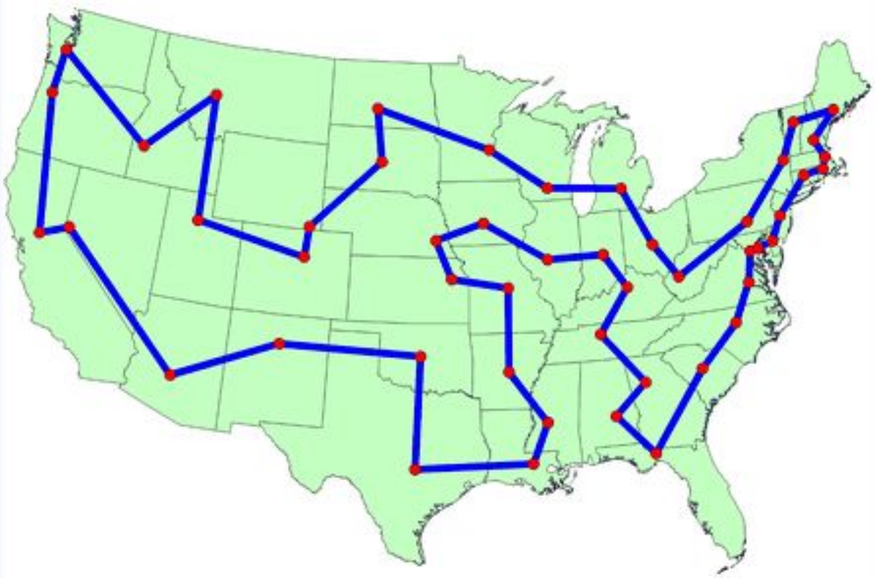
Kohonen Model

Kohonen Model

is particularly useful for visualizing and understanding the structure of high-dimensional data by mapping it onto a lower-dimensional grid.



Application



Traveling Salesman Problem

is a problem of finding the shortest possible route that visits each city in a given list exactly once and returns to the original city.

Training Process

1. **Initialization**
2. Normalization of Input Data
3. Iterative Training
4. Weight Update
5. Animation Update
6. Shortest Path Calculation
7. Stopping Criterion
8. Animation Termination
9. Visualization

Initialize the SOM with random weights for each neuron. These weights typically have the same dimensionality as the input data (in this case, 2D for x and y coordinates).

Training Process

1. Initialization
2. **Normalization of Input Data**
3. Iterative Training
4. Weight Update
5. Animation Update
6. Shortest Path Calculation
7. Stopping Criterion
8. Animation Termination
9. Visualization

Normalize the input data to a common range, typically [0, 1], to ensure uniformity. This step is important as it prevents any particular dimension from dominating the training process.

```
x_n = (x - np.amin(x)) / (np.amax(x) - np.amin(x))
y_n = (y - np.amin(y)) / (np.amax(y) - np.amin(y))
```

Training Process

1. Initialization
2. Normalization of Input Data
3. **Iterative Training**
4. Weight Update
5. Animation Update
6. Shortest Path Calculation
7. Stopping Criterion
8. Animation Termination
9. Visualization

The training process iterates over a fixed number of epochs. Each epoch represents a complete pass through the entire dataset.

Training Process

1. Initialization
2. Normalization of Input Data
3. Iterative Training
4. **Weight Update**
5. Animation Update
6. Shortest Path Calculation
7. Stopping Criterion
8. Animation Termination
9. Visualization

Update the weights of the BMU and its neighboring neurons. The update is done based on a learning rate and a neighborhood function. Neurons closer to the BMU have their weights adjusted more than those farther away.

$$w_{ji}(t+1) = w_{ji}(t) + \eta(t) \cdot h_{ji}(t) \cdot (x(t) - w_{ji}(t))$$

Training Process

1. Initialization
2. Normalization of Input Data
3. Iterative Training
4. Weight Update
5. **Animation Update**
6. Shortest Path Calculation
7. Stopping Criterion
8. Animation Termination
9. Visualization

Update the visualization of the SOM after each epoch to observe how the solution evolves over time.

Training Process

1. Initialization
2. Normalization of Input Data
3. Iterative Training
4. Weight Update
5. Animation Update
6. **Shortest Path Calculation**
7. Stopping Criterion
8. Animation Termination
9. Visualization

Calculate the distance of the current path formed by the neuron weights.

Keep track of the shortest path found during the training process.

Training Process

1. Initialization
2. Normalization of Input Data
3. Iterative Training
4. Weight Update
5. Animation Update
6. Shortest Path Calculation
7. **Stopping Criterion**
8. Animation Termination
9. Visualization

The training process continues for a fixed number of epochs. Alternatively, it can stop when certain conditions are met, such as convergence of the solution or reaching a predefined number of epochs.

Training Process

1. Initialization
2. Normalization of Input Data
3. Iterative Training
4. Weight Update
5. Animation Update
6. Shortest Path Calculation
7. Stopping Criterion
8. **Animation Termination**
9. Visualization

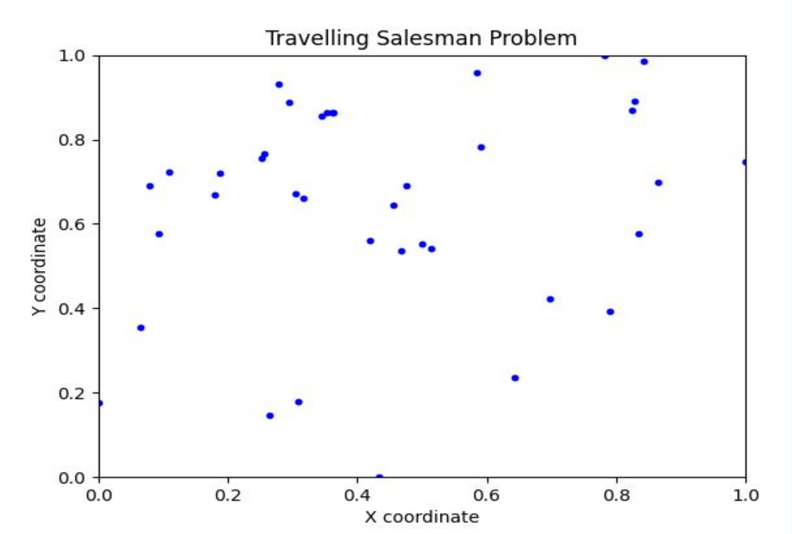
Once the training process is complete, stop the animation and display the final solution.

Training Process

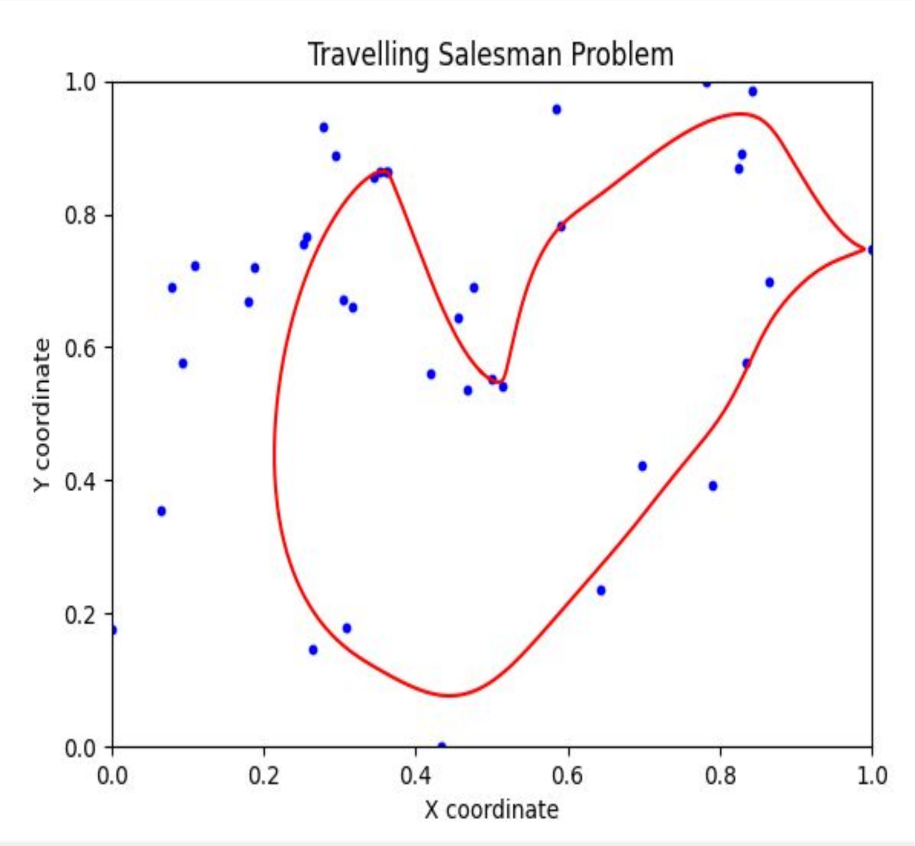
1. Initialization
2. Normalization of Input Data
3. Iterative Training
4. Weight Update
5. Animation Update
6. Shortest Path Calculation
7. Stopping Criterion
8. Animation Termination
9. **Visualization**

Visualize the final solution, which represents the shortest path found through the dataset.

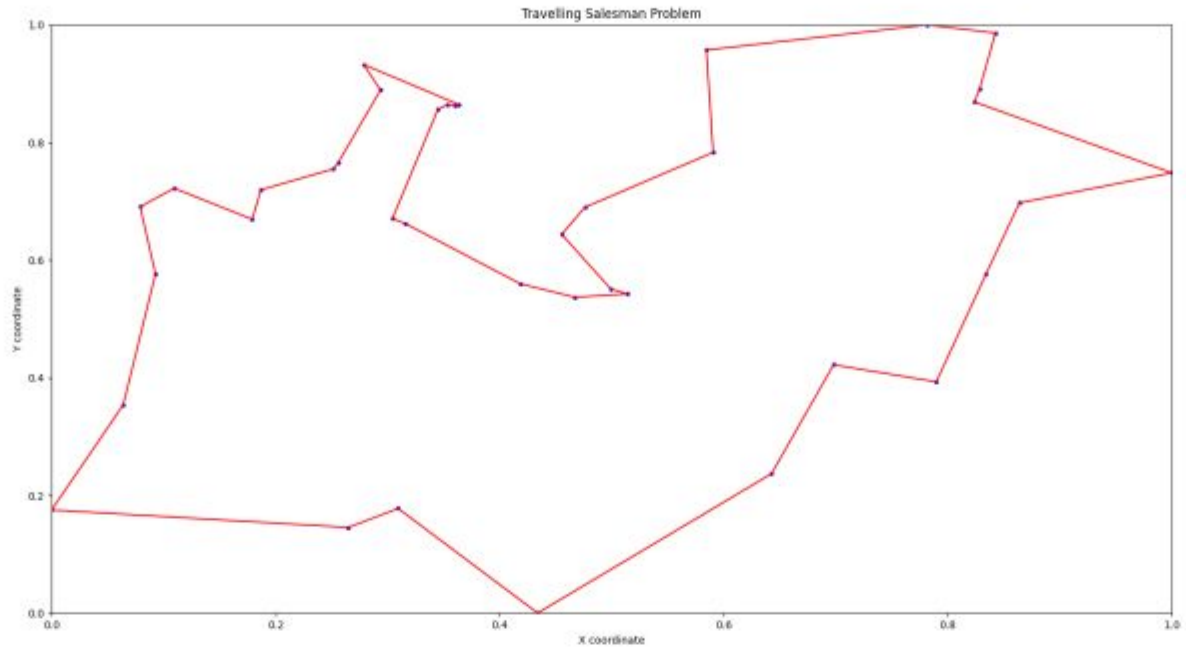
Results




```
Shortest Path: [[0.95817884 0.76096147]
 [0.95831318 0.76101076]
 [0.95844734 0.76105949]
 ...
 [0.95777476 0.76081028]
 [0.95790963 0.76086123]
 [0.95804433 0.76091163]]
```



Results



Travelling Salesman Problem
Y coordinate
X coordinate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

Thank You