1. \*\*Initialize the Network\*\*

a. Define number of cities (N\_Cities) and generate their coordinates.

b. Define number of neurons (N\_Neurons), typically greater than N\_Cities.

c. Initialize neurons in a circular topology with random weights in 2D space.

d. Set initial parameters:

- Learning rate (alpha)

- Neighborhood radius (sigma)

- Number of iterations (T)

2. \*\*Training Loop (for each iteration t = 1 to T)\*\*

a. Select a random city from the dataset.

b. \*\*Find the Best Matching Unit (BMU):\*\*

- Compute Euclidean distance between the city and each neuron.

- Identify the neuron with the minimum distance (BMU).

c. \*\*Update Neuron Weights Using Neighborhood Function:\*\*

- Compute the decayed learning rate: α(t) = α\_initial \* (1 - t/T)

- Compute the decayed neighborhood radius: σ(t) = σ\_initial \* (1 - t/T)

- For each neuron i:

1. Compute distance from BMU (considering circular topology).

2. Compute the neighborhood function using a Gaussian:

`theta = exp(-distance\_to\_BMU^2 / (2 \* σ(t)^2))`

3. Update neuron weight:

`W\_i = W\_i + α(t) \* theta \* (city - W\_i)`

d. Reduce learning rate and neighborhood radius over time.

3. \*\*Extract Final Tour Order\*\*

a. Sort neurons based on their final 2D positions in a circular order.

b. Match neurons to the closest city in the dataset.

c. Return the sequence as the approximated TSP route.