NEURAL NETWORKS HOMEWORK 3

Self-Organizing Maps

ENPM 808Y University of Maryland College Park
Timothy Werder

Table of Contents

SOM Design

For my Neural network I designed a simple SOM that was modified and adapted for each problem. I began by initializing the net based on the input size multiplied by a modifier (pivot). The algorithm then randomly selects an input data and performs the Euclidian distance operation between the net and the selected values and returns the minimum coordinates and Euclidian mapping (Activation).

$$\mathrm{E}\Delta = \sqrt{(\mathrm{q1}\text{-}\mathrm{p1})^2 + (\mathrm{q2}\text{-}\mathrm{p2})^2}$$

The algorithm then performs a nearest neighbors' operation using the minimum coordinates resulting in a local gaussian distribution.

$$\mathrm{d}\Delta = \min|\mathrm{shape}\text{-}\mathrm{q1}, \mathrm{q2}|$$

$$\mathrm{Bij} = \mathrm{e}^{-\frac{\mathrm{d}\Delta^2}{2(\sigma^2)}}$$

The local gaussian distribution is then multiplied by the learning rate and the difference between the selected values and the network (Deactivation). The learning rate and the n value (neighboring sigma) are then diminished. The SOM training operation exits once the number of runs has completed or the learning rate or number of neighbors have diminished below a threshold.

Problem 1: Traveling Salesman

Results

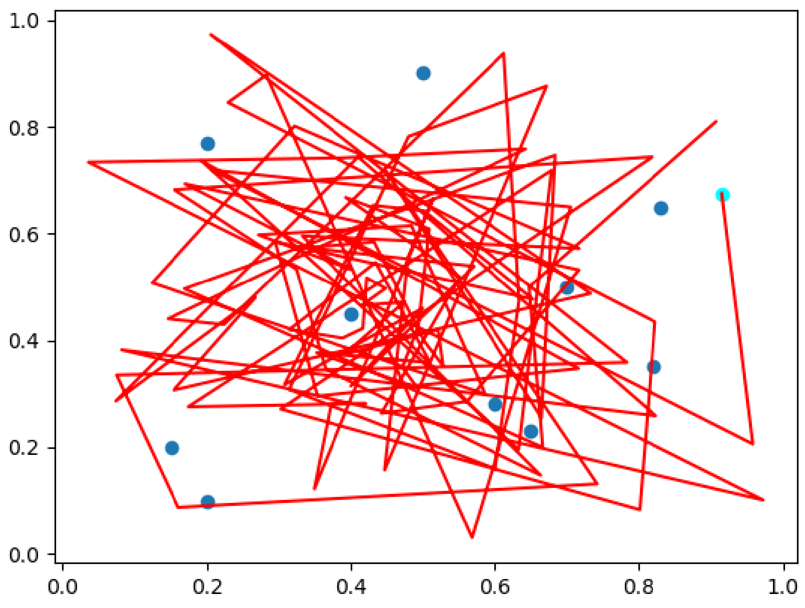

Figure 1: TSP Epoch=1

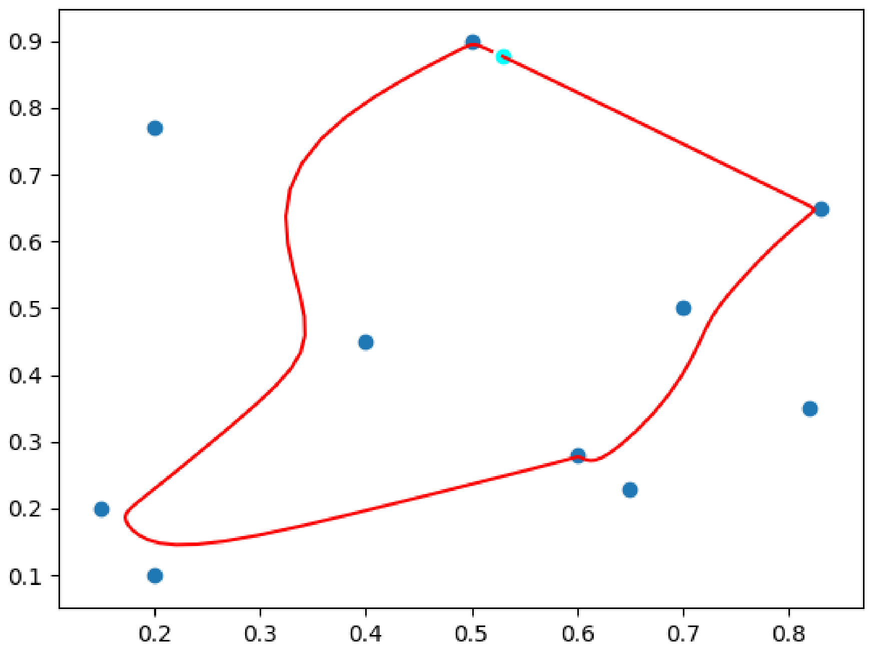

Figure 2: TSP Epoch=50

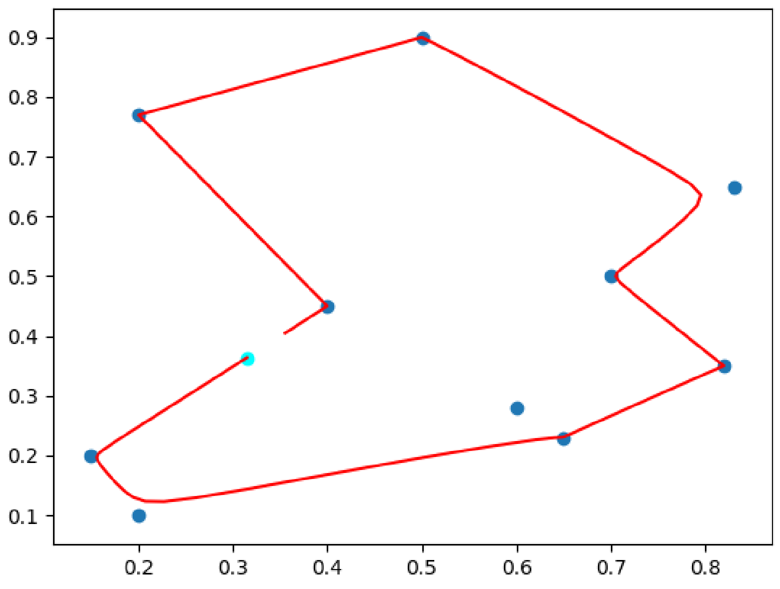

Figure 3: TSP Epoch=150

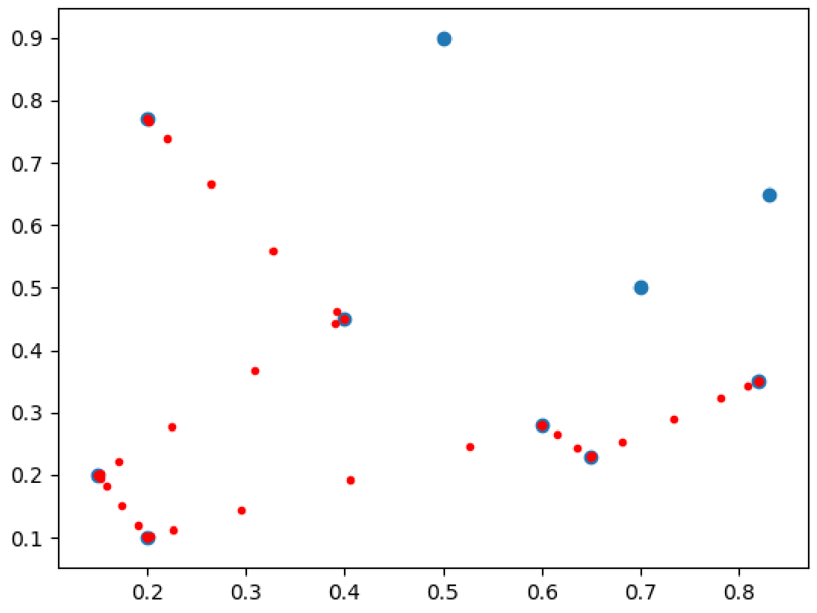

Figure 4: TSP Final Animated

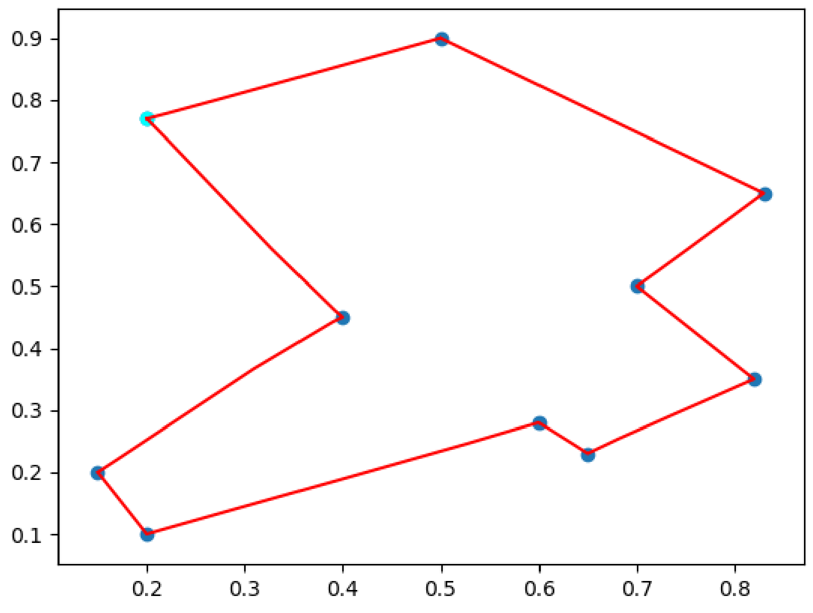

Figure 5: TSP Final Epoch=458

Hyperparameters

Learning Rate= 0.8	Neurons: 100	Sigma: 6	Layering:
Time to Train: 0.78s	Epochs: 458	N: 10	Euclidean, Gaussian

Discussion

The TSP problem is a relatively easy problem of ordering distances. For most algorithms this constitutes checking a point to all neighbors and ordering the minimum distances from start to finish. With SOMs we roughly do the same procedure, with the difference being the gaussian distribution across a network that distributes the distances and eventually culminates in a pathing formed by a linear interpretation of the network. Overall my algorithm performs very well within a short learning period.

Problem 2: Characteristic Mapping

Results

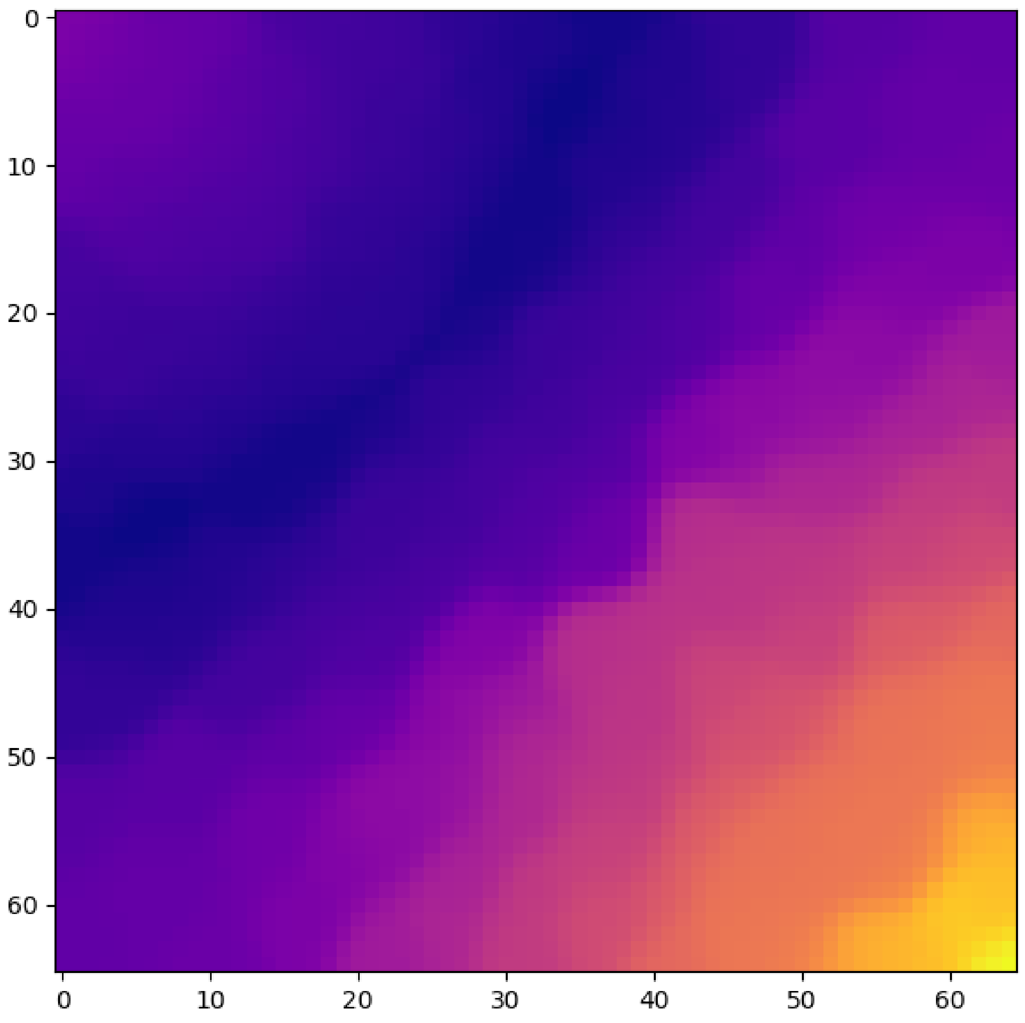

Figure 6: Euclidean gradient mapping

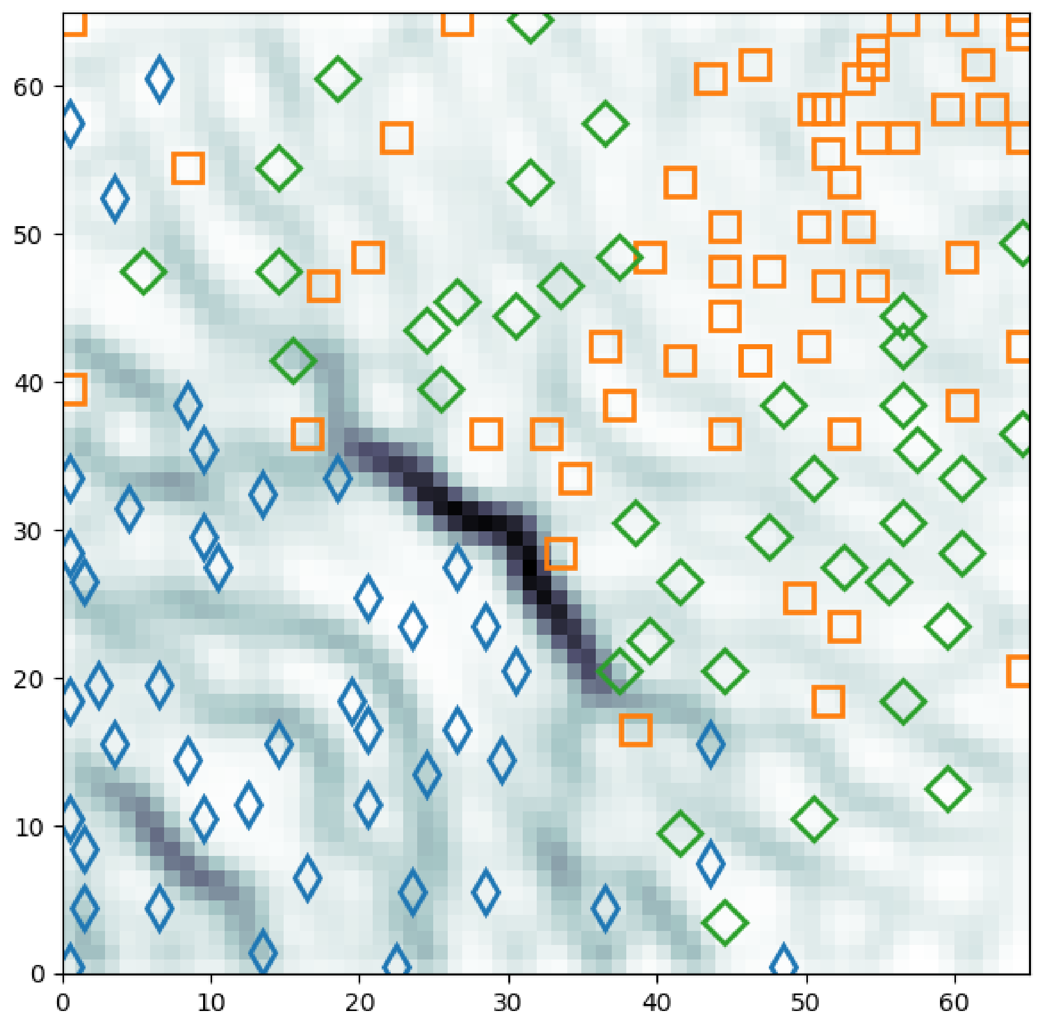

Figure 7: Cluster Mapping

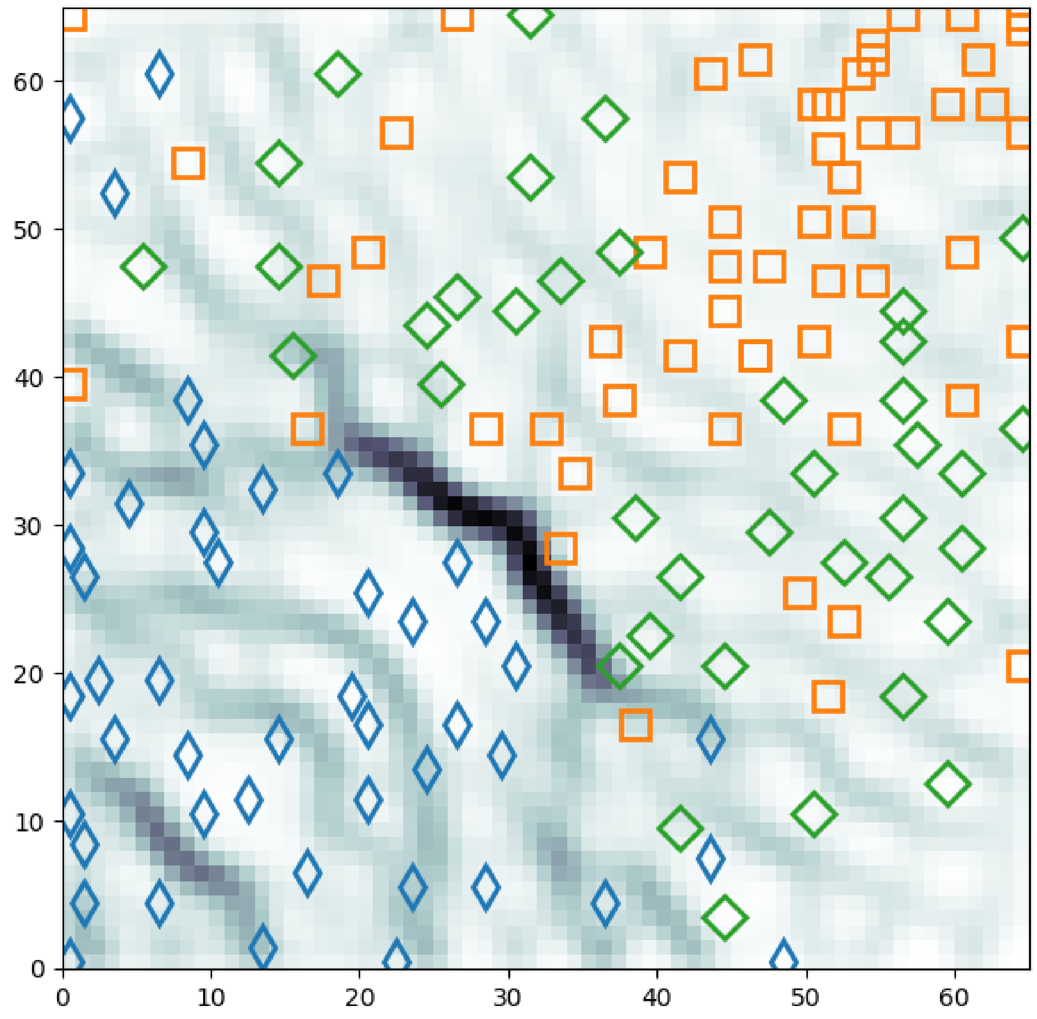

Figure 8: Cluster Mapping Segmentation

Hyperparameters

Learning Rate= 0.7	Neurons: 65	Sigma: 7	Layering:
Time to Train: 3.23s	Epochs: 4172	N: 5	Euclidean, Gaussian 2D

Discussion

The 2D mapping of clusters of data from a multi-dimensional space to a 2D Kohen map is one of the bigger advantages of SOMs. In this problem 13 aspects of data were classified into three classes, these classes are represented by the different colored shapes. For this I shuffled the data and trained. The data was then run against the activation function and plotted along the map after transposing the map and determining the distance map. Overall my algorithm computes a fairly good cluster in a short amount of time. It would have been better if there was more data to split into a train and test segment so that testing could show a better approximation of the map.

Problem 3: Internal Polygon mapping

Results

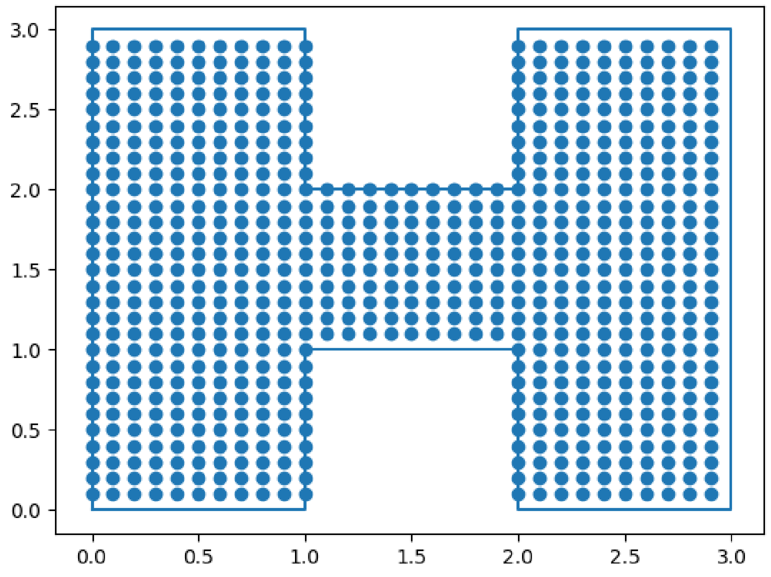

Figure 9: Internal area polygon point representation

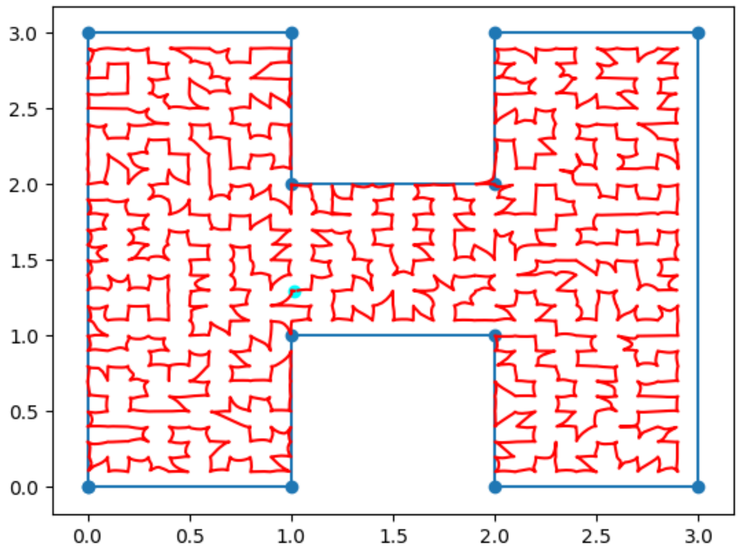

Figure 10: Internal mapping of Polygon

Hyperparameters

Learning Rate= 0.8	Neurons: 9786	Sigma: 6	Layering:
Time to Train: 17.44s	Epochs: 9184	N: 14	Euclidean, Gaussian

Discussion

Similar to the TSP problem the SOM attempts to map out a region. For this to work an additional aspect of data preparation must be made, a distribution of points internal to the polygon. This is done by creating a mask against the polygon and selecting only points internal to the walls. This requires that a larger number of neurons be used to accurately map out each of the points and a higher sigma value to get a finer gaussian distribution. Overall my algorithm performs well with the occasional external mapping to complete the route and does this within a short period of time given the number of neurons and neighbors needing to be accounted for.

Key takeaways

This project helped me understand the power of SOMs. They are very powerful and very lightweight ways to do self-organizing based on large or limited quantities of data. Overall, my code worked well, but given more time it could be adapted such that it could act more as a library to be called for any dimension rather than tuning the internal parameters for each problem but I am overall happy with the results.