Manifold Learning - Self Organizing Maps

Lucas de Morais Tramasso
Florida Institute of Technology
Melbourne, FL 32901
ldemoraistra2019@my.fit.edu

Rosalin Dash
Florida Institute of Technology
Melbourne, FL 32901
rdash2018@my.fit.edu

Abstract

In this project, we focused on classifying countries using various economic data. We have generated the input data set for 124 countries with 26 different indicators. Using the Self Organizing Map Algorithm we can see that the 26-dimensional data can be represented in 2D and the samples formed clusters of similar countries. This can be better visualized by using the the Income Level classification to label the countries. As the Self Organizing Map Algorithm is unsupervised learning and it learns with respect to the input data, we planned in taking a higher dimension data with more countries and we have achieved our desired goal as the input data set seemed to be perfectly clustered. We also focused on the time taken for the code execution and we have come up with an algorithm which obtains the desired result with a minimal time of execution which is roughly around 23 seconds for 10000 iterations and with a five sigma value, where sigma represents the spread of the neighborhood function. We have run the code with variations which include sigma ranging from 1, 3, 5 and iterations from 100, 500, 1000, 2000, 10000. After observing the various output we can conclude that the more sigma and the higher the number of iterations we can expect a better clustering of the input data.

1 Manifold Learning

Manifold in mathematics is defined as a collection of points forming a certain kind of set. In the computer world, we can define it as mapping images or objects into a lower dimensional space. By representing into the lower dimensional space, we try to preserve only the properties what is necessary for us. Hence, manifold learning is basically a technique knowing which property is to be reserved to obtain the optimal result we desire.

2 Self-organizing Map

Self-organizing Map is a popular neural network model and is classified under the unsupervised training technique where no human intervention is needed during the learning and needs the least knowledge about the characteristics of the input data. It is a type of manifold learning as we represent a higher dimensional data set in 2D by preserving the properties needed.

2.1 Stages of Self Organizing Maps

The stages to create a Self Organizing Map can be defined as below.

2.1.1 Initialization

Network initialization can be done in three different ways. First, assign random values to the neurons in the map. Second, initialize the network by randomly picking samples from the input data and assigning them to neurons. Lastly, initialize vectors to lie in the same input space spanned by two eigenvectors corresponding to the largest eigenvalues of the input data, also called PCA Initialization. For the purposes of this project, the last two ones were used.

2.1.2 Training

This is an iterative process. It draws sample vectors from the input data set, chooses a winner (weight vector closest to the input vector) and updates the vector values of the neighbors of the winner neuron. The sample data can be drawn either randomly or sequentially.

2.1.3 Visualization

The distance between the adjacent nodes can be represented in different coloring using a U-Matrix representation. A dark coloring between the neurons corresponds to a large distance and thus a gap between the values in the input space. A light coloring between the neurons signifies that the values are close to each other in the input space. Light areas can be thought of as clusters and dark areas as cluster separators.

2.1.4 Validation

Models can be created as we desire but to ensure the model to be working, we need to follow validation on sample data. This validation has to be done by running the algorithm on a large sample of data inputs and check whether the model gives a reasonable and accurate value. The data has been labeled for better visualization of accuracy in the output

3 Dynamic Self Organizing Map (DSOM)

This is a interesting variation of the self-organizing map algorithm where the original time-dependent (learning rate and neighborhood) learning function is replaced by a time-invariant one. It ensures a coupled environment while performing on-line and continuous learning. By replacing the learning and neighborhood functions, the neighborhood in a DSOM is dynamic, resulting in different self-organization controlled by the free elasticity parameter.[8]

3.1 Dynamic Neighborhood

The neighborhood in a DSOM becomes dynamic as a consequence of the new learning and neighborhood functions. These new functions are responsible for changing how neurons learn new data. In a DSOM, if a neuron is close to the data, it will be the only one to learn the new data. When there is no neuron able to learn the data than any neuron can learn based on its distance to the data. A consequence of this new behavior is that a DSOM maps the structure of the distribution and not the density.

3.2 Elasticity

It is necessary to control when a neuron is close enough to learn new data. This is important because when a neuron learns new data, it prevents its neighbors from learning the same data. The elasticity parameter is responsible for modulating the connection between neurons. Having a very high or low elasticity may result in an algorithm that does not converge, or self-organization may not happen, respectively. Although there might be an optimal elasticity for a given distribution, it was not investigated in the paper.

The DSOM algorithm is different from other algorithms because it does not map density. While this can be a drawback in cases where data density is needed, it can also be a benefit depending on the task. The DSOM is a very interesting algorithm, however, for the purposes of this project, a **regular SOM** algorithm was used in every test.

4 U-Matrix (Unified Distance Matrix)

U-Matrix is a representation of Self Organizing Map where the Euclidean distance between the values of neighboring neurons is displayed in a gray scale image. This image is used to visualize the data in higher dimensional space using a 2D image. Each neuron in the U-matrix contains the Euclidean distance in the input space between the neighboring cells. The distance value in the U-matrix is represented as colors in a gray scale i.e. the darker it is the neighbors are apart and the lighter it is the neighbors are closer with respect to the neuron data.

5 Input Data Set

We have generated the data set of 124 countries with 26 indicators taken from the worldbank.org [6] and added them in a CSV file named countries_data.csv without any indicator as the algorithm deals with unsupervised learning. We have appended the country codes along with the income level for better visualization of the neurons. The countries_data.csv is located in the AI_Final/src folder.

6 Source Code Execution

Below are the necessary requirements for running the source code.

- Python 3
- Source Code folder named AI_Final which is present inside the zip folder named Sp19_Group5_ManifoldLearning
- Input data set named countries_data.csv which is present inside AI_Final folder
- All the other dependent libraries are numpy and Tkinter.

6.1 Execution Steps

1. Unzip the folder Sp19_Group5_ManifoldLearning.
2. Go into the AI_Final *cd Sp19_Group5_ManifoldLearning/AI_Final* and read the README file
3. Execute the *somCountries.py* file using python 3

7 Output

After the Self Organizing Map is created and trained, it is possible to obtain a map given a set of samples. The output is a hexagonal grid representing the Self Organizing Map. Each hexagon represents a neuron from the map and samples from the input data set are mapped to positions on the map after it has been trained. Each neuron can contain any number of samples, represented by their respective country code. The distance matrix (u-matrix) is used as the background color for the map. Each color in the u-matrix represents a range of distances described in Figure 1

Figure 1: U-matrix colors and respective values

After creating the map, the countries are labeled (colored) according to their income group for better visualization of relationships between countries. The labels are created according to the Input Group classification, which is based on the GNI of a country. These labels are described in Figure 2.

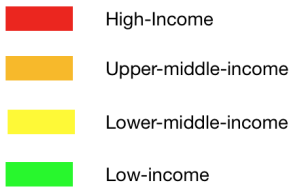

Figure 2: Labels for each country

8 Results

The following results were obtained using a 25x15 Self Organizing Map using the four different configurations below.

1. Random sample initialization and random training
2. Random sample initialization and sequential training
3. PCA initialization and random training
4. PCA initialization and sequential training

For every configuration, the map was trained using sigma = 5 and number of iterations = 100, 2000, and 10000.

8.1 Random sample initialization and random training

8.1.1 Number of Iterations: 100 and Sigma: 5

Time Elapsed: 0.41151 s

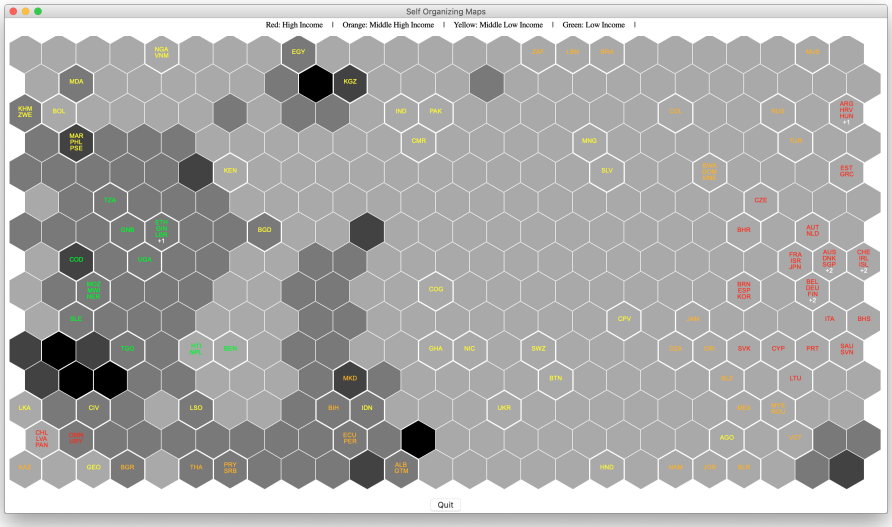

Figure 3: Random sample initialization and random training, 100 iterations

8.1.2 Number of Iterations: 2000 and Sigma: 5

Time Elapsed: 8.57447 s

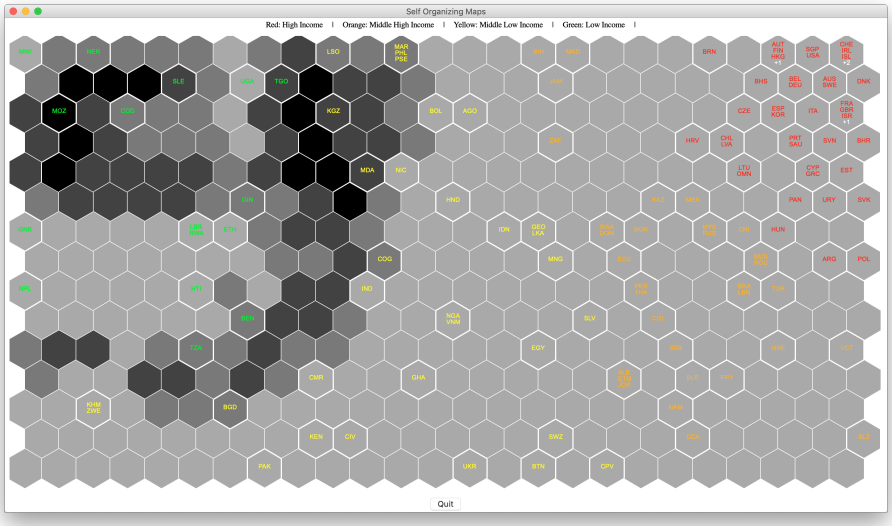

Figure 4: Random sample initialization and random training, 2000 iterations

8.1.3 Number of Iterations: 10000 and Sigma: 5

Time Elapsed: 42.64751 s

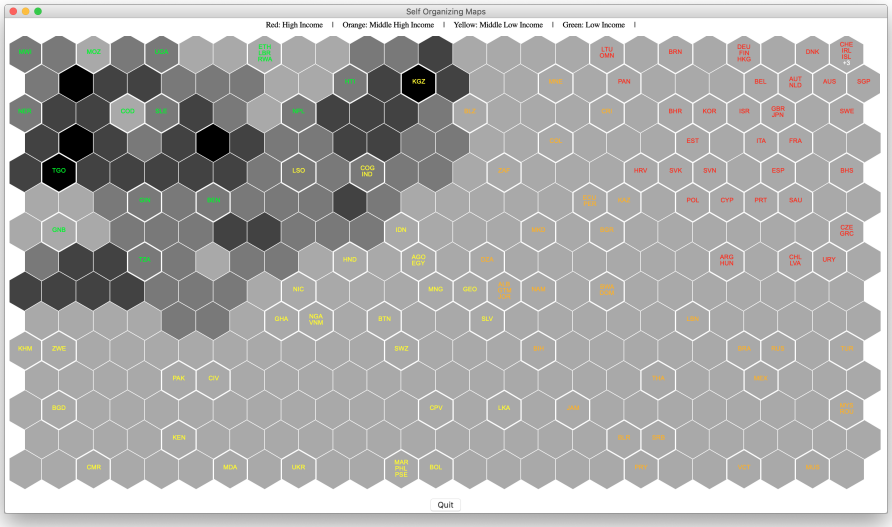

Figure 5: Random sample initialization and random training, 10000 iterations

8.2 Random sample initialization and sequential training

8.2.1 Number of Iterations: 100 and Sigma: 5

Time Elapsed: 0.40916 s

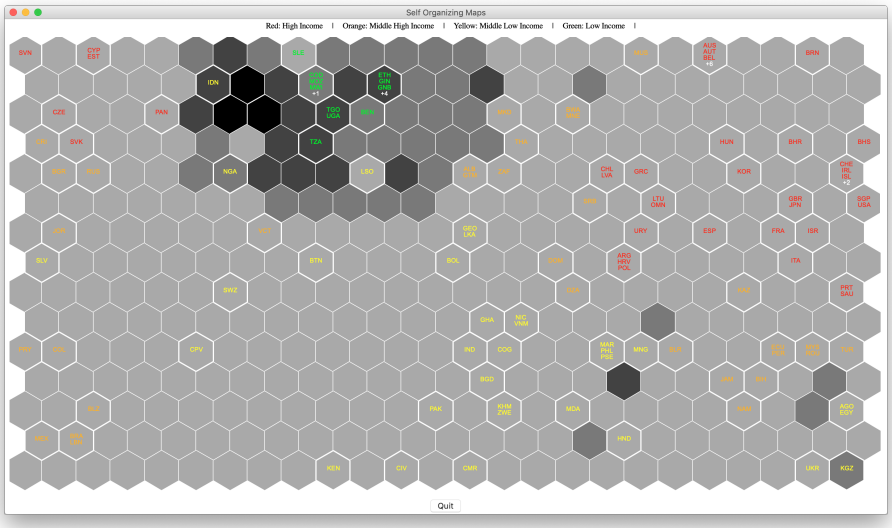

Figure 6: Random sample initialization and sequential training, 100 iterations

8.2.2 Number of Iterations: 2000 and Sigma: 5

Time Elapsed: 8.62996 s

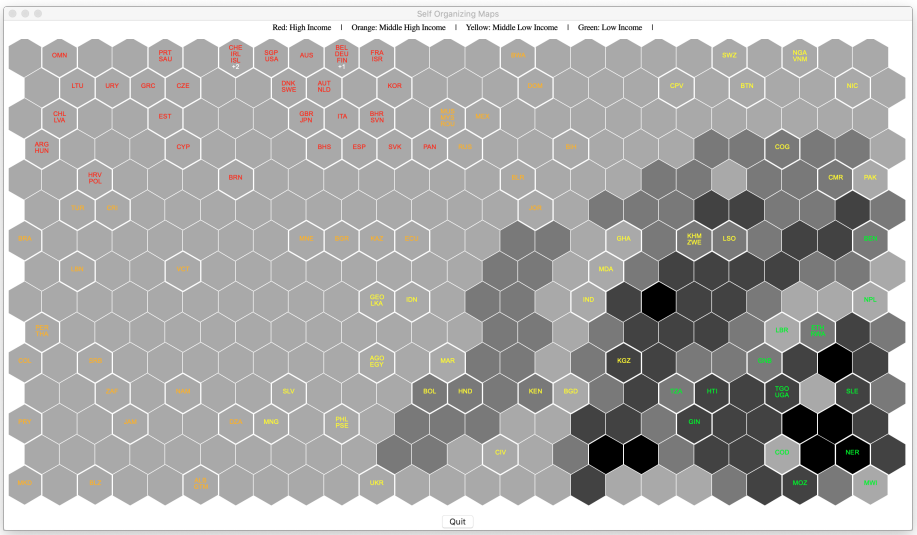

Figure 7: Random sample initialization and sequential training, 2000 iterations

8.2.3 Number of Iterations: 10000 and Sigma: 5

Time Elapsed: 42.71395 s

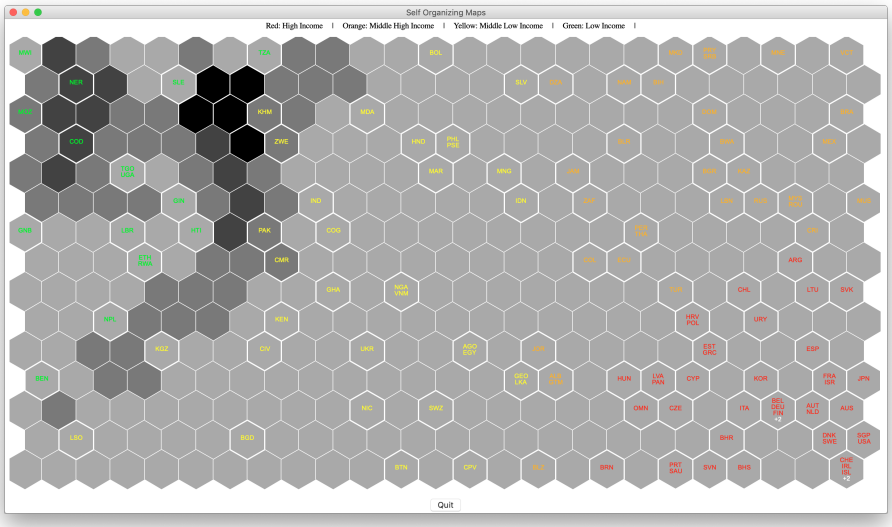

Figure 8: Random sample initialization and sequential training, 10000 iterations

8.3 PCA initialization and random training

8.3.1 Number of Iterations: 100 and Sigma: 5

Time Elapsed: 0.45273 s

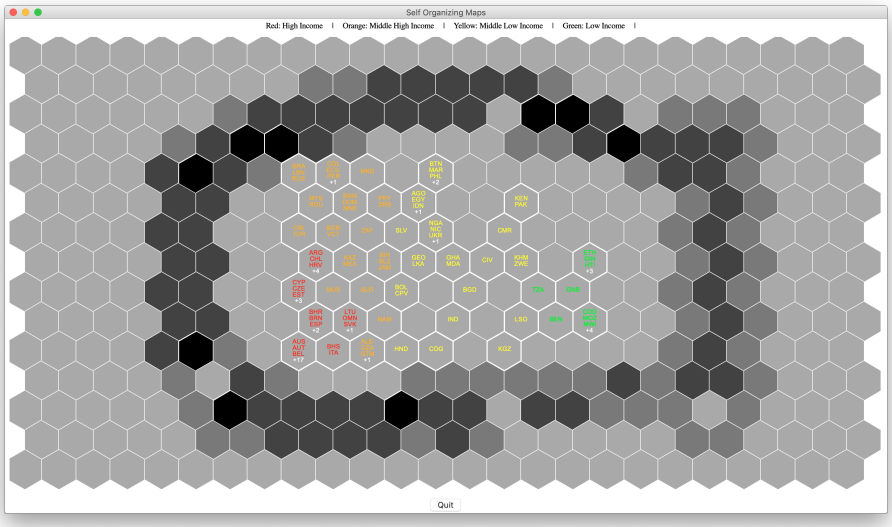

Figure 9: PCA initialization and random training, 100 iterations

8.3.2 Number of Iterations: 2000 and Sigma: 5

Time Elapsed: Time Elapsed: 8.52247 s

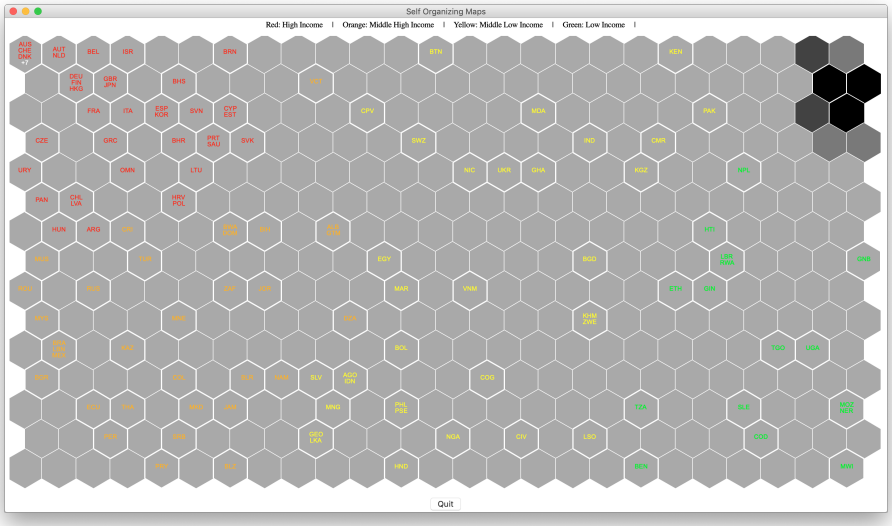

Figure 10: PCA initialization and random training, 2000 iterations

8.3.3 Number of Iterations: 10000 and Sigma: 5

Time Elapsed: 41.51154 s

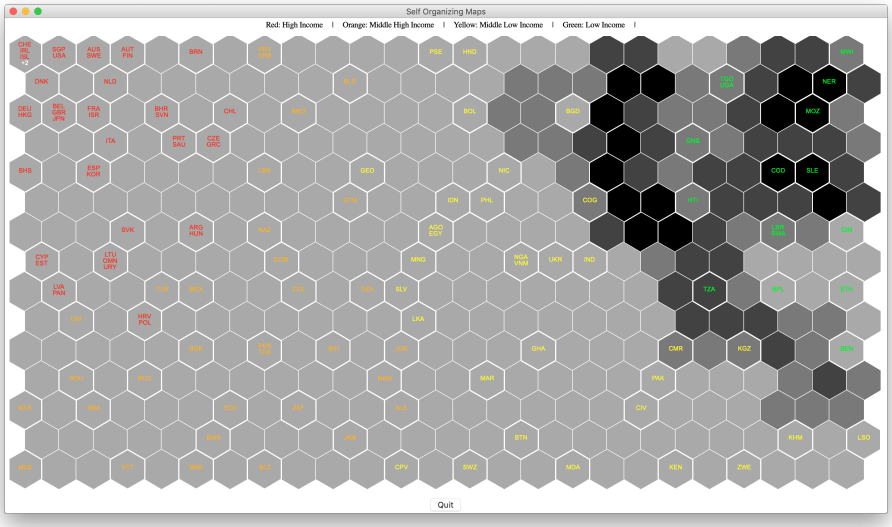

Figure 11: PCA initialization and random training, 10000 iterations

8.4 PCA initialization and sequential training

8.4.1 Number of Iterations: 100 and Sigma: 5

Time Elapsed: 0.41838 s

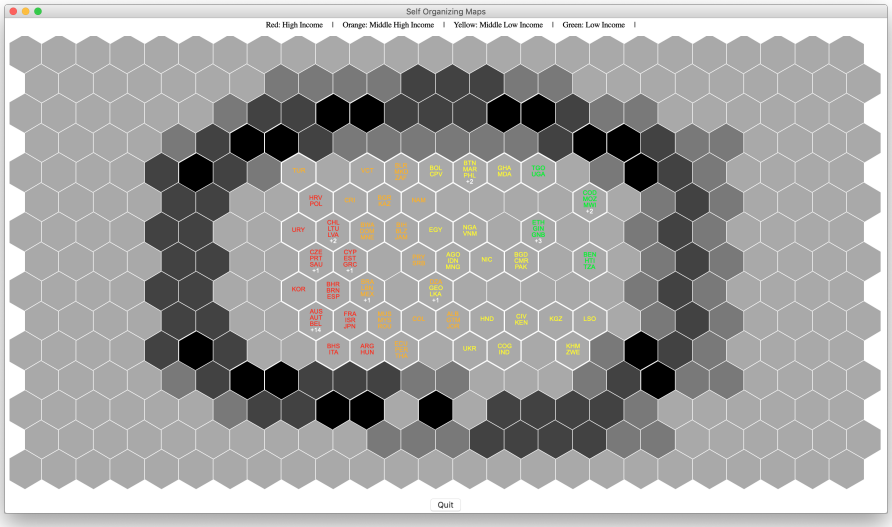

Figure 12: PCA initialization and sequential training, 100 iterations

8.4.2 Number of Iterations: 2000 and Sigma: 5

Time Elapsed: 8.55840 s

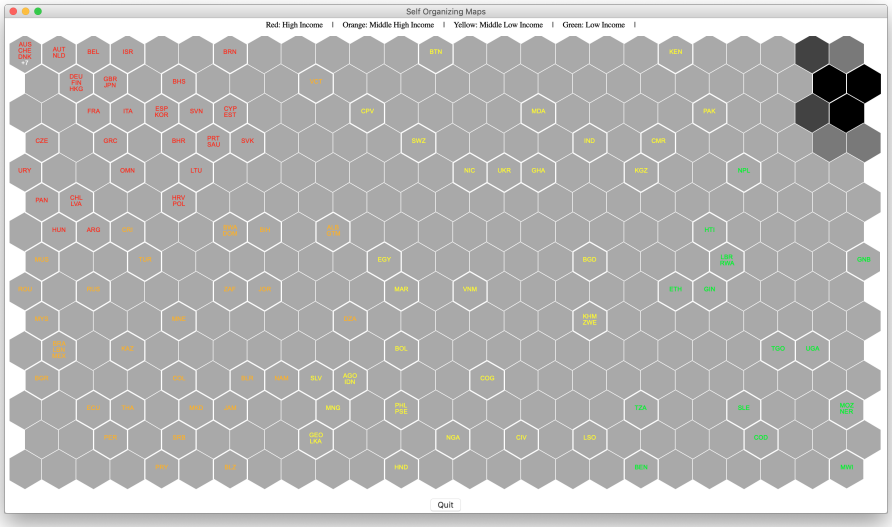

Figure 13: PCA initialization and sequential training, 2000 iterations

8.4.3 Number of Iterations: 10000 and Sigma: 5

Time Elapsed: 41.77348 s

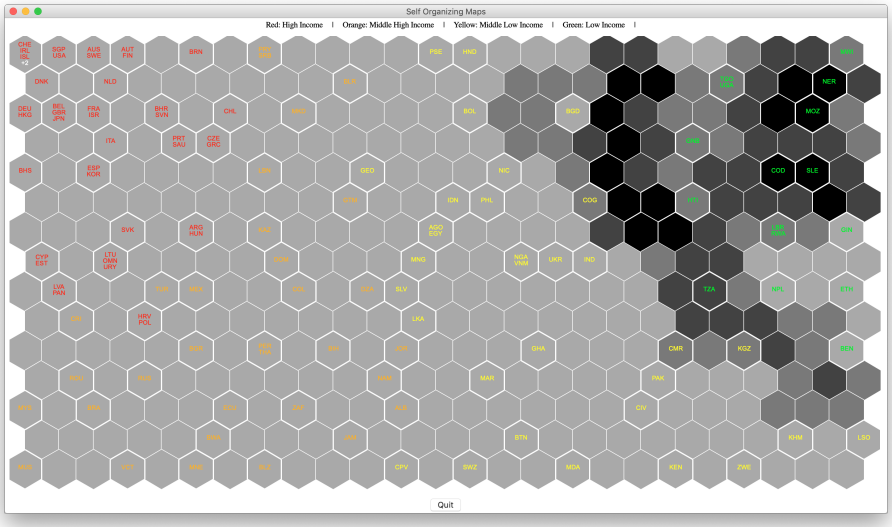

Figure 14: PCA initialization and sequential training, 10000 iterations

8.5 Terminal Output Example

```
# ================== Matrix Data ================== #
(0,0)     -->     5 sample(s):    ['CHE', 'IRL', 'ISL', 'LUX', 'NOR']
(0,1)     -->     1 sample(s):    ['USA']
(0,2)     -->     2 sample(s):    ['AUS', 'SWE']
(0,4)     -->     1 sample(s):    ['BHS']
(0,6)     -->     1 sample(s):    ['BRN']
(0,8)     -->     1 sample(s):    ['MNE']
(0,11)    -->     2 sample(s):    ['PER', 'THA']
(0,14)    -->     1 sample(s):    ['VCT']
(1,1)     -->     2 sample(s):    ['DNK', 'SGP']
(1,9)     -->     1 sample(s):    ['BWA']
(2,1)     -->     2 sample(s):    ['AUT', 'NLD']
(2,2)     -->     5 sample(s):    ['BEL', 'DEU', 'FIN', 'GBR', 'HKG']
(2,3)     -->     1 sample(s):    ['ITA']
(2,5)     -->     1 sample(s):    ['SVN']
(2,7)     -->     1 sample(s):    ['RUS']
(2,14)    -->     1 sample(s):    ['MKD']
(3,0)     -->     3 sample(s):    ['FRA', 'ISR', 'JPN']
(3,2)     -->     1 sample(s):    ['KOR']
(3,3)     -->     1 sample(s):    ['BHR']
(3,4)     -->     2 sample(s):    ['PRT', 'SAU']
(3,5)     -->     2 sample(s):    ['EST', 'GRC']
(3,6)     -->     1 sample(s):    ['HRV']
(3,8)     -->     2 sample(s):    ['BRA', 'LBN']
(3,12)    -->     1 sample(s):    ['PRY']
(4,0)     -->     1 sample(s):    ['ESP']
(4,3)     -->     1 sample(s):    ['CZE']
(4,5)     -->     1 sample(s):    ['LTU']
(4,9)     -->     1 sample(s):    ['BGR']
(4,12)    -->     1 sample(s):    ['ZAF']
(4,14)    -->     1 sample(s):    ['BLR']
(5,2)     -->     1 sample(s):    ['CYP']
(5,14)    -->     1 sample(s):    ['JAM']
(6,0)     -->     1 sample(s):    ['SVK']
(6,1)     -->     2 sample(s):    ['OMN', 'URY']
(6,2)     -->     2 sample(s):    ['LVA', 'PAN']
(6,3)     -->     1 sample(s):    ['CHL']
(6,6)     -->     1 sample(s):    ['MEX']
(6,8)     -->     1 sample(s):    ['ECU']
(6,9)     -->     1 sample(s):    ['COL']
(6,11)    -->     1 sample(s):    ['NAM']
(7,0)     -->     1 sample(s):    ['POL']
(7,1)     -->     2 sample(s):    ['ARG', 'HUN']
(7,4)     -->     1 sample(s):    ['CRI']
(7,5)     -->     3 sample(s):    ['MUS', 'MYS', 'ROU']
(7,6)     -->     1 sample(s):    ['KAZ']
(7,8)     -->     1 sample(s):    ['SRB']
(7,12)    -->     1 sample(s):    ['GTM']
(8,3)     -->     1 sample(s):    ['TUR']
(8,8)     -->     1 sample(s):    ['BIH']
(9,7)     -->     1 sample(s):    ['JOR']
(9,9)     -->     1 sample(s):    ['DZA']
(9,12)    -->     1 sample(s):    ['GEO']
(9,14)    -->     1 sample(s):    ['LKA']
(10,2)    -->     1 sample(s):    ['DOM']
(10,5)    -->     1 sample(s):    ['ALB']
(10,6)    -->     1 sample(s):    ['MNG']
(10,7)    -->     2 sample(s):    ['AGO', 'EGY']
(10,10)   -->     1 sample(s):    ['SLV']
(11,0)    -->     1 sample(s):    ['BLZ']
(11,8)    -->     1 sample(s):    ['IDN']
(11,12)   -->     1 sample(s):    ['CPV']
(12,5)    -->     1 sample(s):    ['BOL']
```

Figure 15: Terminal Output with Matrix Data and the countries for each position

9 Conclusion

From the above observations we can conclude that SOM helps with data visualization , dimensionality reduction and clustering data set. The initialization technique used for neuron i.e the PCA and the random initialization is not same i.e. PCA converge faster compared to random initialization. Considering all the 26 different economic indicators, by labeling the countries with the income group classification (GNI per capita) we can see that the resulting clusters contain countries that are very similar economically. Hence, the label facilitates the visualization of the clusters, and the resulting maps were what we expected.

10 Executive Summary

10.1 Learning Accomplishments

- Understanding of unsupervised learning and how it can achieved on a random set of input data without having any idea about their labels.
- Understanding of neighbourhood functions, learning rate and how it affects the neurons arrangement
- Understanding of different ways of initializing neurons for the entire data set
- Understanding of various python libraries like numpy, pandas and Tkinter, used for coding

10.2 Achievements

- Generated an input data with 26 economic indicators for 124 countries
- Minimize the time taken for code execution
- Generated U-matrix in grey-scale
- Output Data clustered with different configuration and with different training patterns

11 References

[1] https://users.ics.aalto.fi/jhollmen/dippa/node9.html

[2] Bullinaria, J.A. (2004) Self Organizing Maps: Fundamentals (Introduction to Neural Networks: Lecture 16).

[3] https://github.com/JustGlowing/minisom

[4] Nicolas P. Rougier, Yann Boniface. Dynamic Self-Organizing Map. Neurocomputing, Elsevier, 2011, 74 (11), pp.1840-1847. <10.1016/j.neucom.2010.06.034>

[5] http://www.ai-junkie.com/ann/som/som1.html

[6] http://datatopics.worldbank.org/world-development-indicators/themes/economy.html#income-and-savings_1

[7] https://stackoverflow.com/questions/26583602/displaying-data-in-a-hexagonal-grid-using-python

[8] https://hal.inria.fr/inria-00495827/document