CS 451: Computational Intelligence

Assignment 03

Syed Hasan Faaz Abidi (sa06195), Syed Hammad Ali (sa04324)

17 April 2022

1. Reinforcement Learning

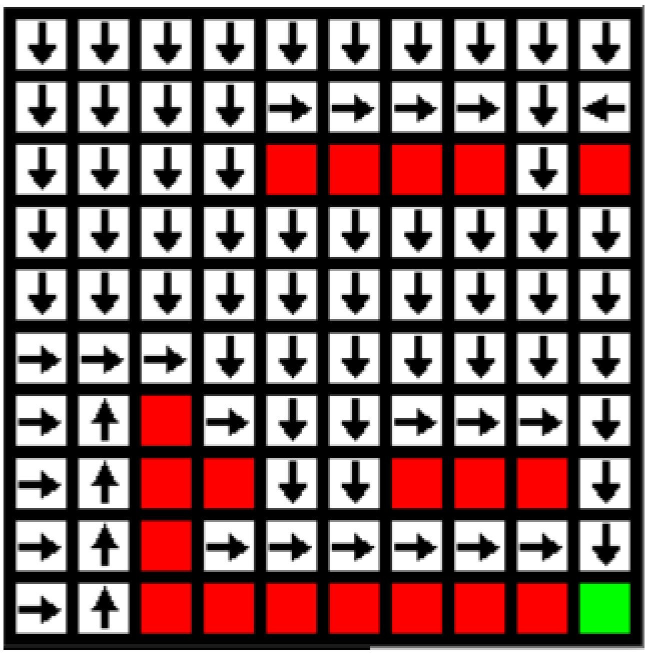
Our task was to train an agent using Reinforcement Learning to navigate through a 10x10 grid world and reach those blocks which offer the maximum reward. A 10x10 default grid was hard-coded as well as a function (initialize_grid) to initialize a 10x10 space and another function (place_objects) was written to populate the empty 10x10 grid with obstacle and reward blocks. The red block i.e. the obstacles carried a reward of -100 points while the green boxes carried a reward of 100 points, while all other plain blocks carried no reward whatsoever. For every block the agent was on, a value was calculated for every possible move in either of the 4 directions possible (North/ Up, South/ Down, East/ Right, West/ Left). The reward for a move was calculated by multiplying alpha (learning/ discount rate), which was taken to be 0.9, with the max of all the values in the possible directions i.e,

Reward = max(value of moving North, value of moving South, value of moving East, value of moving West) * 0.9.

The agent would navigate the grid-world for several iterations and the value function would eventually converge to an optimal solution where it was able to navigate its way to the green blocks from any non-red (obstacle) block without making any unnecessary moves.

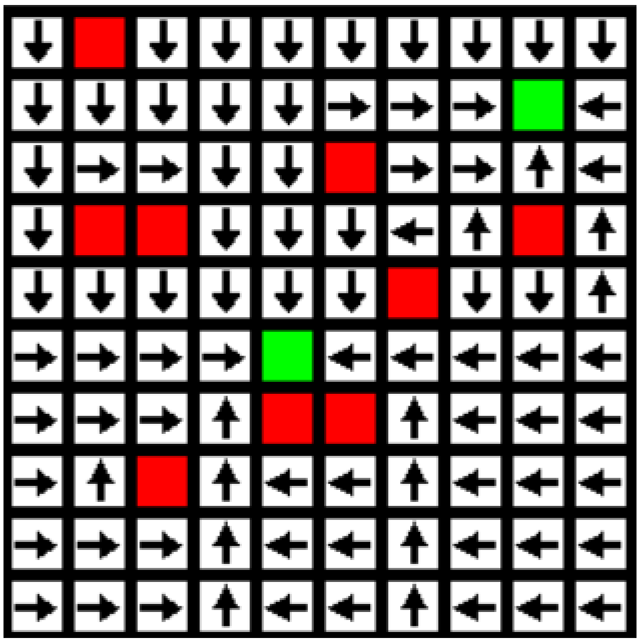
The simulation of the algorithm was tested several times on various configurations of the 10x10 grid and the agent was found to converge to an optimal solution every time. Some of the simulated examples of the solutions, on both the default and randomly generated grids, are shown below;

Default Grid:

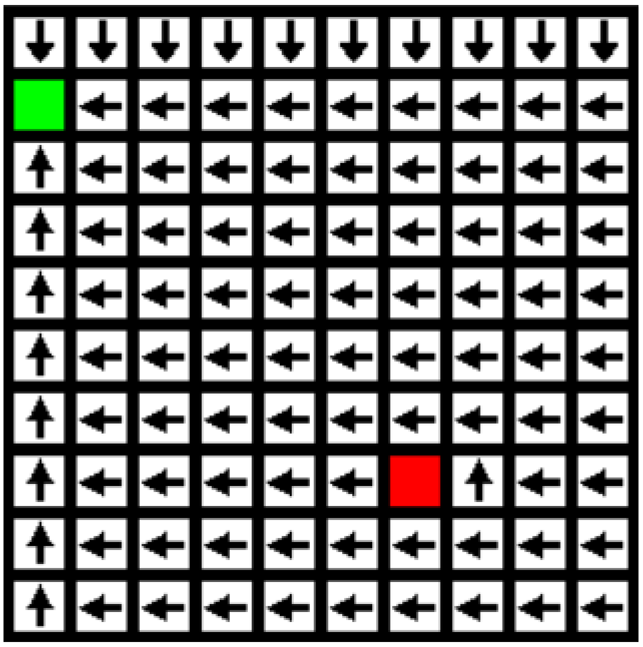


Randomly Generated Grids:

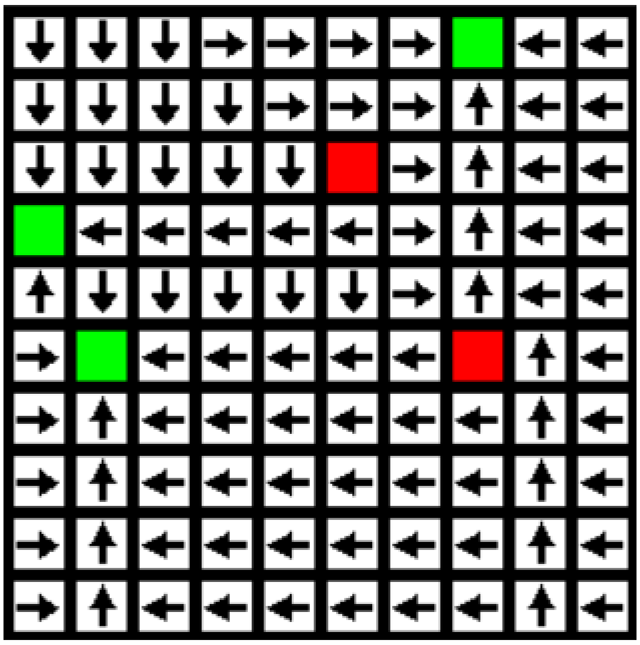
(1)



(2)



(3)



2. Clustering World dataset using SOM

2.1. Problem Statement

In this part of the assignment, we were supposed to cluster any world data of our choice and apply Self Organizing Map to it for clustering and visualization. SOMs are known for clustering and dimension reduction and in this question, we used it for both of these applications.

2.2. Dataset

Choosing the right dataset was really crucial for this task. After a lot of searching, I decided to go with a dataset of **Global Carbon Dioxide Emissions**. This is time-series data that includes more than 200 years of data for every country. For every country, there are **three** parameters. This adds more depth and dimensions to the data. The three parameters are Annual CO2 emissions, Share of Global CO2 emissions, and Annual CO2 emissions per capita. All three attributes contributes to overall CO2 by a country. There were a few issues with the data initially which we solved in the preprocessing.

2.3. Algorithm

I applied the classic SOM algorithm with almost the same methods that are generally used. One major modification was the use of NumPy arrays. I tried to minimize the use of for/while loops as much as possible. Our implementation is very generic in every aspect. We can give any size grid and it will map colors to that grid. However, for better results, grid size must be chosen appropriately. One good approach that I found by trial and error is that grid size must be bigger than the rows and columns by a very small amount. For example. If I have 120 rows (labels) then a 12x12 grid is sufficient because there are 12*12=144 patches available for every label.

Moreover, as our implementation generates a hexagonal plot grid with labels, the input vector must have labels to it. Below is the acceptable format for an input vector:

Input = [[‘A’, 1, 2, 3], [‘B’, 1, 2, 3], [‘B’, 1, 2, 3],]

In this example, **‘A’‘B’‘C’** are labels and this vector has 3 rows and 4 columns.

In addition to it, I implemented the Gaussian Neighborhood function which decreases the radius of the neighborhood as time increases. The learning rate was also dynamic and it was also dependent on the time constant (alpha) and the number of iterations.

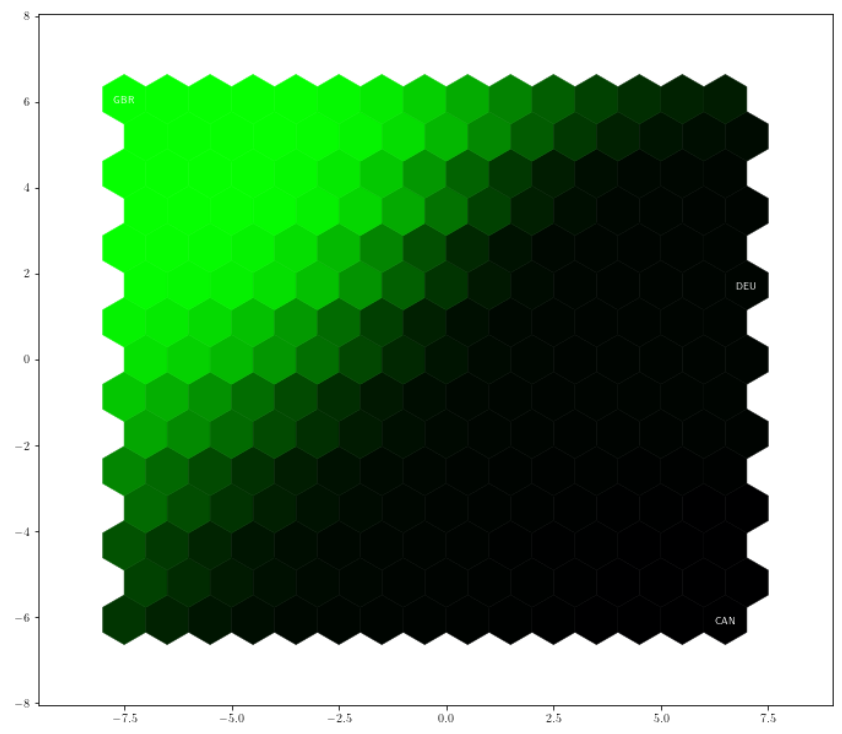
For the visualization part, I have generated hexagonal plots exactly as shown in the assignment pdf. For plotting, I used a python package called ‘hexalattice’. I am not directly importing it rather I modified the package a little according to my needs and the kind of result I wanted. There’s a file called “hexalattice.py” in the question 02 folder of our assignment, that file is from that package that I modified a little.

2.4. Results

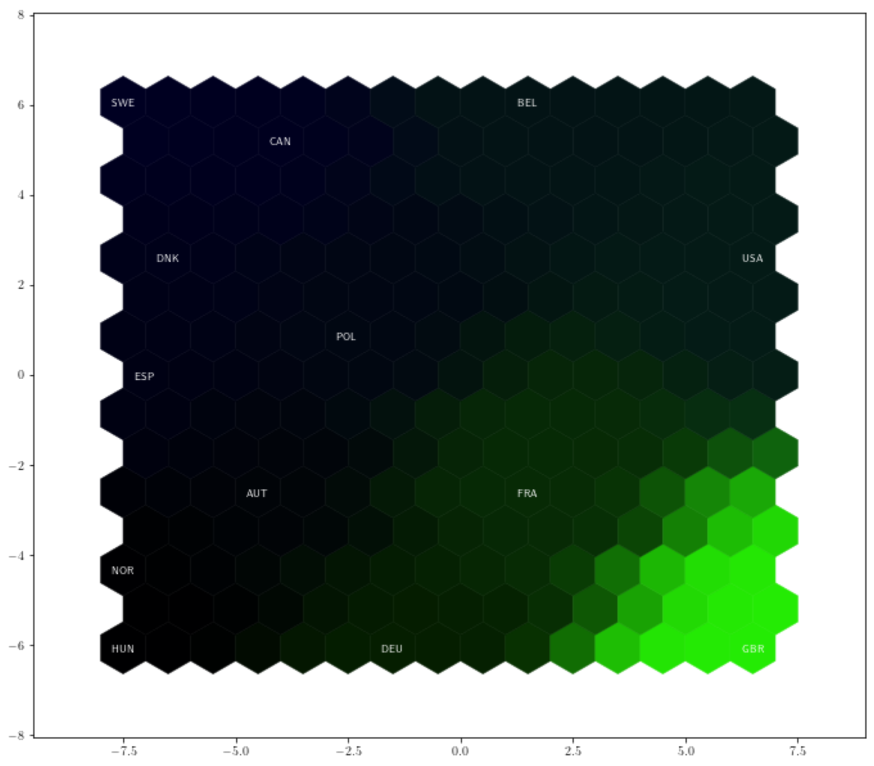
Results were the most exciting yet disappointing part of this assignment. Since we were using quite a large dataset, I ran our algorithm for about 150 years of data.

SOMs plots for all 150 years of data are in the Map folder inside Question 02 but below are a few worth mentioning plots:

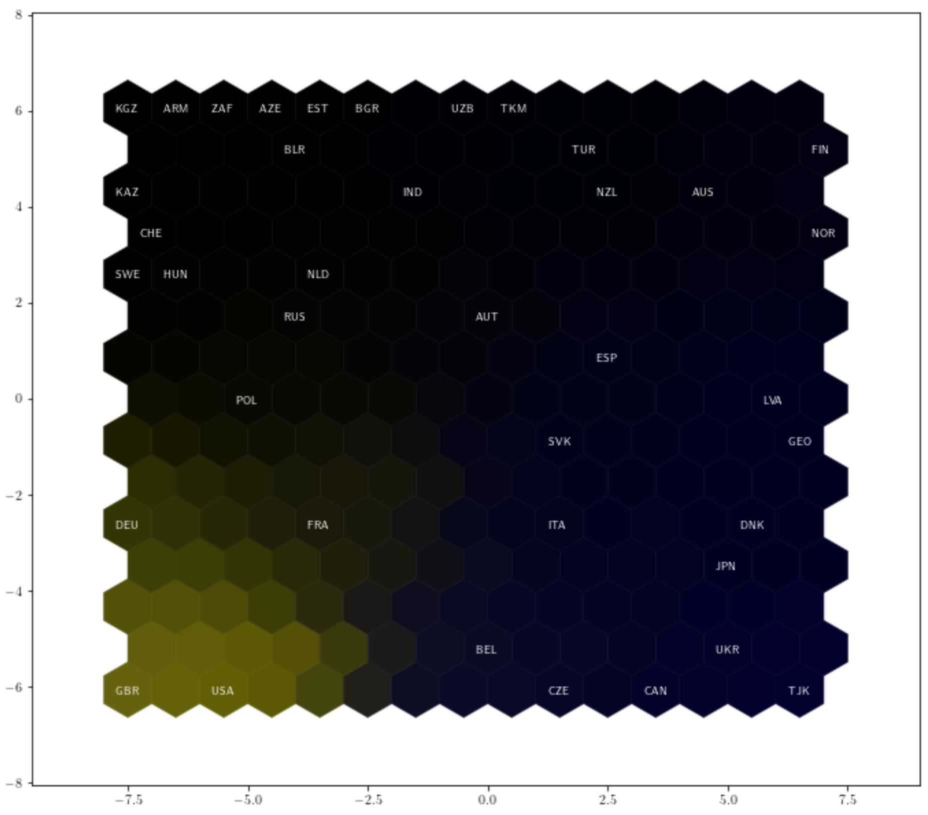
Year: **1794**



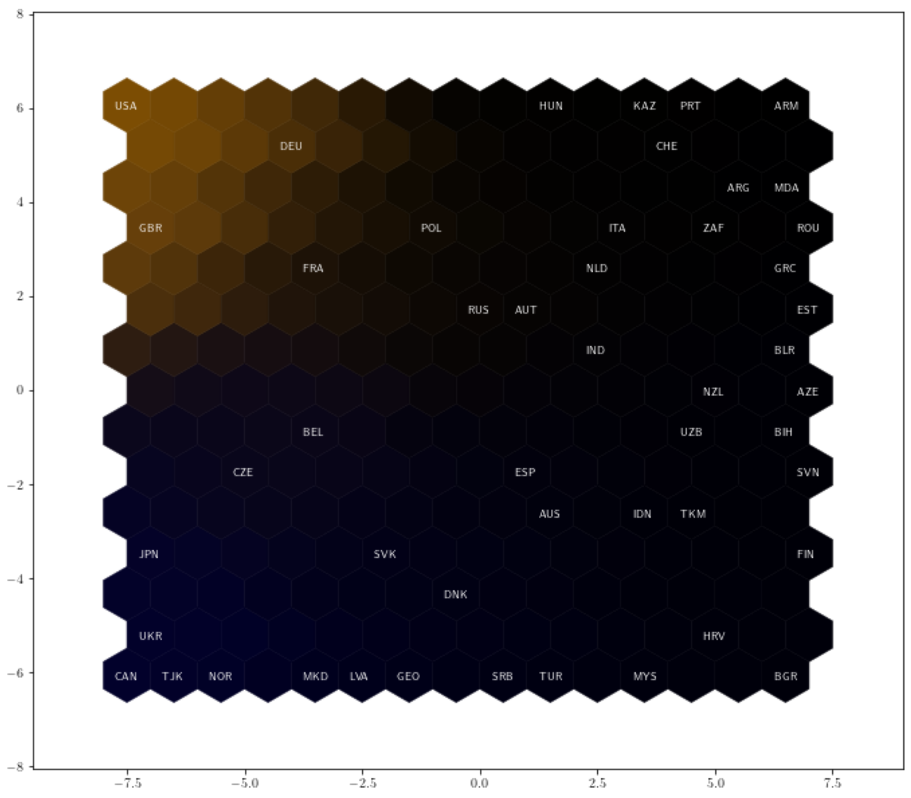
Year: **1845**



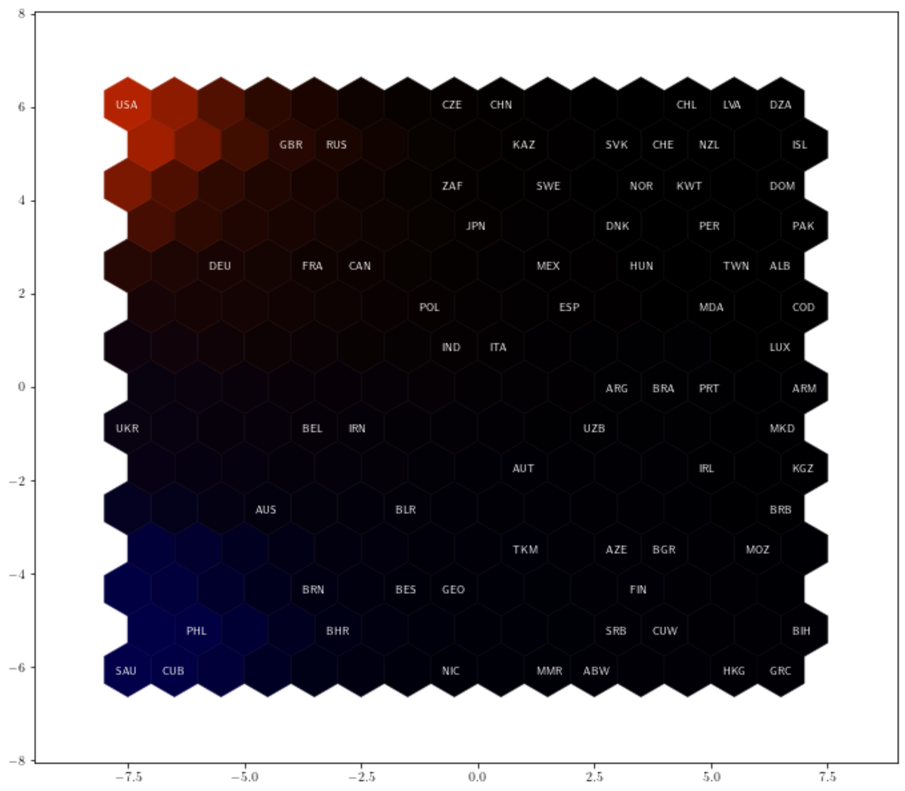
Year: **1886**



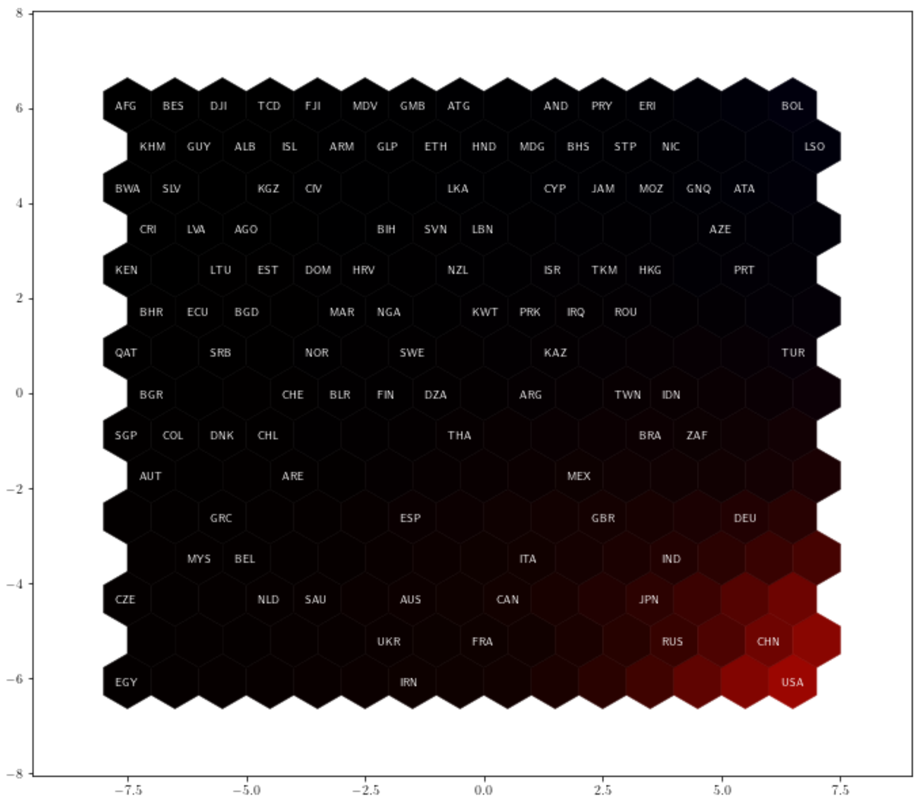
Year: **1897**



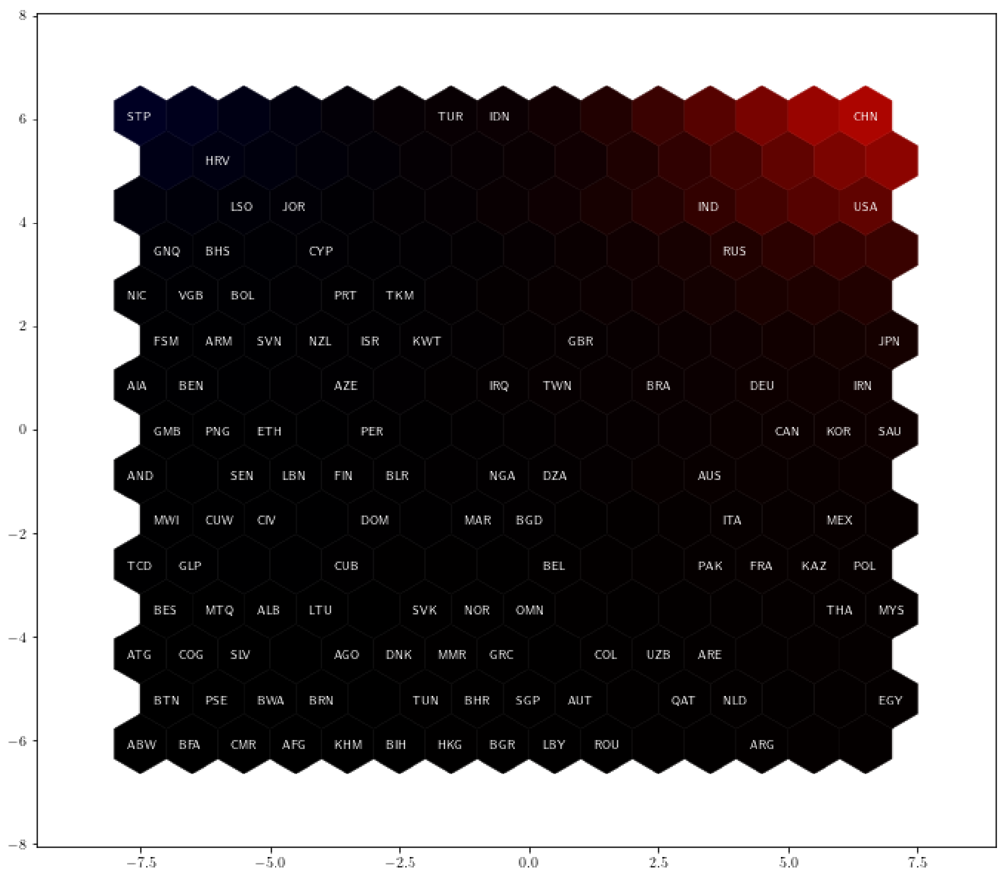
Year: **1947**



Year: **1999**



Year: **2020**



There's so much we can talk about looking at these clusters and how are they changing over time. There are many underlying factors here. In the initial year, about 1770s, we can see bright green and there are only a handful of countries which makes sense since there is not enough data of that time and carbon emissions all around the world were neglible. Also, in the initial years, there was only one major country contributing which was GBR (Great Britain). As years passed, clusters changed. In the 20th century, China was no way near the USA but in the recent year China is very close to the USA and in the SOM of 2020 China replaced the position of the USA. Moreover, these graphs reflect historical events. Russia was in a similar cluster to the USA but around the 1980s it went far from the USA lattice. Also, the USA massively increased its CO2 emissions in the years when the industrial revolution took place.

It's interesting to see the insights we can get by only using data. Also, please note that all the countries are labeled in the grid because many countries belonged to the same patch. For better visualization, I have labeled just one country from that patch. But all the countries are used for training. There was a total of 207 countries in our dataset.

2.5. Limitations

We faced many challenges while doing this. Firstly, I struggled a lot with writing a proper neighborhood function. After that, finding the right dataset was really tiresome. There were many resources for public data but all of them had single attributes. Basically, the data was 2D. And I was looking for multidimensional with more than one parameter for every country.

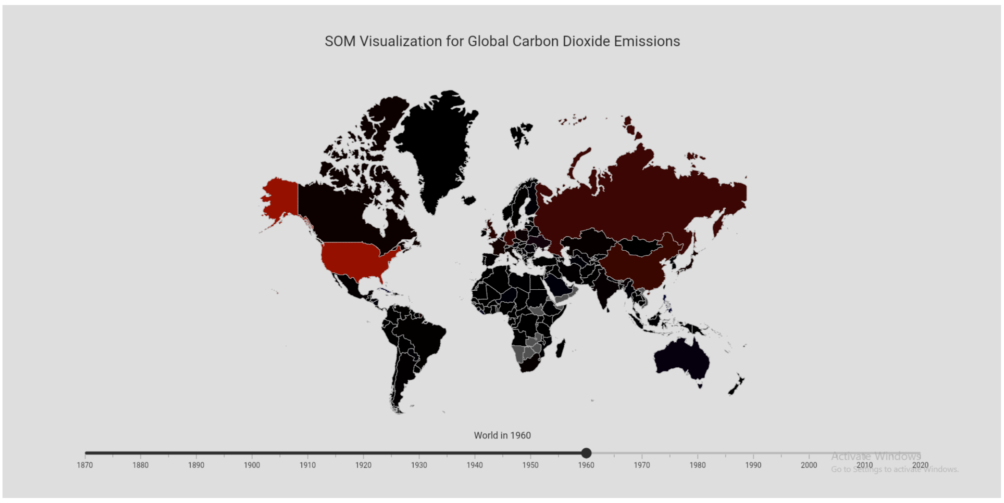
2.6. Bonus Visualization - Web App

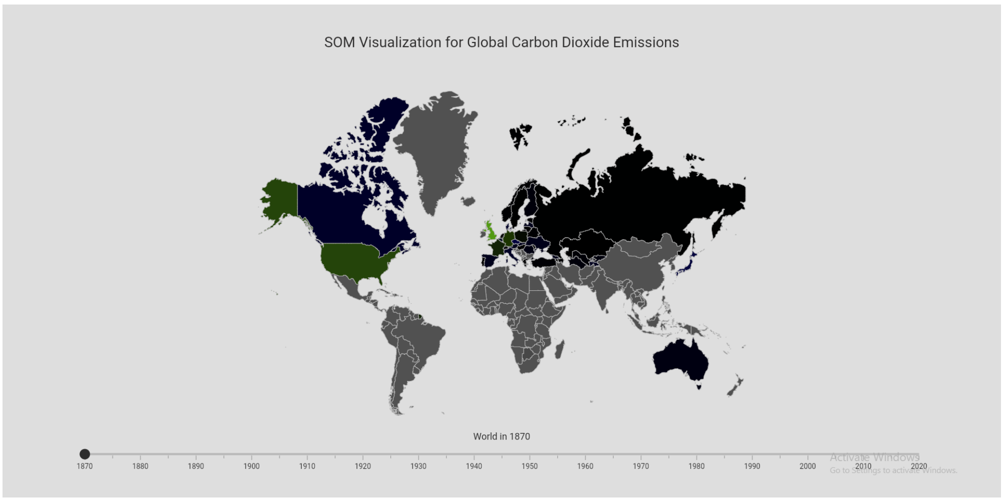
Now, for the bonus part, we had to visualize the SOM on a world map with progression. So, I created a web app for visualizing almost 150 years of Self Organizing Maps of our dataset.

Here's the link for the web app: https://som-historical-co2.netlify.app/#/

Please wait a few seconds. The app first fetches all of the data (which is pretty large) and then visualizes it on the world map.

Screenshot:


SOM Visualization for Global Carbon Dioxide Emissions
World in 1960
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
Activate Windows
Go to Settings to activate Windows.


SOM Visualization for Global Carbon Dioxide Emissions
World in 1870
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
Activate Windows
Go to Settings to activate Windows.