PROJECT TITLE: Implementation of Kohonen Self Organizing Maps for Color Recognition

Kohonen SOMs is an unsupervised machine learning technique discovered by Professor Teuvo Kohonen in the 1980s. Kohonen SOMs are generally used for taking n-dimensional information and mapping it down to a 2D representation of the input space $\mathbb{R}n$. In this project, this machine learning technique will be applied to a clustering a set of n-dimensional input and clustering the input together by common characteristics. Then categorizing the input to have a way to interact with the program. The application will be a form of color recognition utilizing kohonen Self Organizing Maps. The colors can be represented as vectors of (red, green, blue) components and the SOM will categorize the colors and you can then ask what color a given color vector best represents.

Decomposition:

1. **TASK 1:** Symbolically visualizing the map: Every vector in the map will be initialized with three values corresponding to the RGB values. An empty map should be an n by n grid consisting of -. A filled map should be an n by n grid consisting of RGB vectors. The map will consist of a list of lists.

2. **TASK 2:** Initializing the weight vector map: Every vector in the map will be initialized with three values corresponding to the RGB values. SOMs are computationally expensive and should be initialized carefully so less iterations are needed. Three weight vector maps will be tested, one map having all randomly dispersed values, another having red, green, blue, and black initialized at the corners of the map fading toward the center, and lastly having red, green, and blue initialized equidistant from the center.

3. **TASK 3:** Finding the winning weight vector: Find the weight vector with the shortest distance to the sample vector. This can be done using similarity metrics, and a common one to use is euclidean distance.

4. **TASK 4:** Finding neighboring weight vectors: Find the weight vectors that are neighbors to the winning weight vector. Common techniques for finding the neighboring vectors, include others hexagons, concentric squares, and the gaussian function.

5. **TASK 5:** Creating the learning function: Given that the winning weight vector is scaled to be more like the sample vector. The neighboring weight vectors are also affected; whereas the further away the neighbors are the less they will learn and over time the neighbors of the winning weight vector will be less and less affected. A common technique for the decreasing rate of change for each weight vector, is to use the gaussian function.

6. **TASK 6:** Scaling the neighboring weight vectors: Scale the weight vectors that are neighbors to the winning weight vector, to be more similar to the sample vector. The function used for scaling, simply has decrease the radius of what gets affected over time.

7. **TASK 7:** Visualizing the grade of the SOMs: A common problem with SOMs and many other machine learning techniques is being able to assess the quality or performance of the program. Here similarities between high dimensional information might become difficult to visualize and thus assigning performance measurements be eye. A solution for this to calculate the distance between weight vectors and their neighbors; whereas the average of all these distances being a color which is then set to the respective location.

8. **TASK 8:** Color recognition through classification: After the SOMs have been trained, the weighted vectors should be clustered together by their similarities. Given this to be the case, craft a way to label all the similar data by asking the program what color is "and then a specified RGB color vector". If the system doesn't already have a classification for the given vector, supply one to the program to label all the similar ones and to get an accurate color classification. A common post-clustering technique is to use k-means clustering.

9. **TASK 9:** Similarity visualization: Create a program to represent the similarities in the SOMs using a black and white coloring scheme.

10. **TASK 10:** SOMs color visualization: Create a program to create a video visualization of the Self Organizing Map adjusting throughout all the iterations.

11. **TASK 11:** Paint a canvas from colors in two different clusters: Create a program to alternatingly paint from each cluster, by selecting different colors in those clusters at random.

12. **TASK 12:** Write all clusters to a file: Create a program to write all the clusters from the program to a file.