Self-Organizing Maps (SOMs) for Dimensionality Reduction and Visualization: The Impact of Neighborhood Functions on Topology Preservation

Abstract

This tutorial explores Self-Organizing Maps (SOMs), an unsupervised neural network technique for dimensionality reduction and visualization of high-dimensional data. We focus specifically on how different neighborhood functions affect the topology preservation capabilities of SOMs. Through implementation from scratch and extensive experimentation, we demonstrate the unique characteristics of Gaussian, Mexican Hat, and Bubble neighborhood functions, analyzing their impact on clustering quality, convergence behavior, and visualization properties. This tutorial provides practical insights for applying SOMs effectively in data analysis tasks.

1. Introduction

High-dimensional data visualization remains a significant challenge in machine learning and data analysis. While techniques like Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are popular, Self-Organizing Maps (SOMs) offer unique advantages in preserving topological relationships within data.

SOMs, introduced by Teuvo Kohonen in the 1980s, provide a way to project high-dimensional data onto a low-dimensional (typically 2D) grid while preserving the topological properties of the input space. Unlike other dimensionality reduction techniques, SOMs create a structured representation that maintains spatial relationships between data points, making them particularly useful for visualization and exploratory data analysis.

This tutorial focuses on a critical but often overlooked aspect of SOMs: the neighborhood function. We'll explore how different neighborhood functions—Gaussian, Mexican Hat, and Bubble—affect the SOM's ability to preserve topology and represent data clusters effectively.

2. Understanding Self-Organizing Maps

2.1 Core Concepts

A Self-Organizing Map consists of a grid of neurons, each with a weight vector of the same dimensionality as the input data. The SOM learning process involves:

1. **Competitive learning**: For each input sample, find the neuron with the closest weight vector (Best Matching Unit or BMU)
2. **Cooperative learning**: Update the BMU and its neighbors to make their weight vectors more similar to the input
3. **Adaptive process:** Repeat this process until the map converges

The key to SOM's topology preservation lies in the cooperative learning step, where the neighborhood function determines how much influence the current input has on neurons surrounding the BMU.

2.2 Neighborhood Functions

The neighborhood function defines the region of influence around the BMU during weight updates. We explore three common neighborhood functions:

1. **Gaussian:** A smooth, bell-shaped function that gradually decreases with distance from the BMU
2. **Mexican Hat (Ricker wavelet):** Has a positive central region surrounded by a negative region, creating a more complex influence pattern
3. **Bubble (step function):** A simple binary function where neurons within a certain radius are updated equally, and those outside receive no update

These functions significantly impact how SOMs learn and represent data structures, as we'll demonstrate through our experiments.

3. Implementation from Scratch

To fully understand SOMs and the impact of neighborhood functions, we've implemented a SOM from scratch in Python. Our implementation includes:

1. A flexible `SelfOrganizingMap` class supporting different neighborhood functions
2. Customizable learning parameters (learning rate, sigma, decay functions)
3. Comprehensive visualization tools for U-Matrix and component planes
4. Quality metrics for evaluating topology preservation

The core training algorithm follows these steps:

```
python
for iteration in range(n_iterations):
    # 1. Select a random input vector
    x = data[random_index]

    # 2. Find the Best Matching Unit (BMU)
    bmu_idx = find_bmu(x)

    # 3. Calculate the neighborhood influence based on the chosen function
    influence = calculate_influence(bmu_idx, neighborhood_function)

    # 4. Update weights according to learning rate and influence
    weights += learning_rate * influence * (x - weights)

    # 5. Decay learning rate and neighborhood radius
    decay_learning_rate()
    decay_sigma()
```

The key difference between neighborhood functions lies in the `calculate_influence` method, where we implement different mathematical formulations for each function type.

4. Experimental Results

4.1 Dataset Selection

For our experiments, we use three datasets:

1. **Iris dataset:** A classic 4-dimensional dataset with 3 classes
2. **Synthetic 3D clusters:** Custom-generated data with clear cluster structures
3. **Olivetti faces:** A high-dimensional (4096 features) dataset of face images

These datasets provide a range of dimensionality and structural complexity to test our SOM implementations.

4.2 Visual Comparison of Neighborhood Functions

When applying SOMs with different neighborhood functions to the Iris dataset, we observe distinct patterns in the resulting maps. Figure 1 shows the U-Matrices generated using the Gaussian, Mexican Hat, and Bubble neighborhood functions.

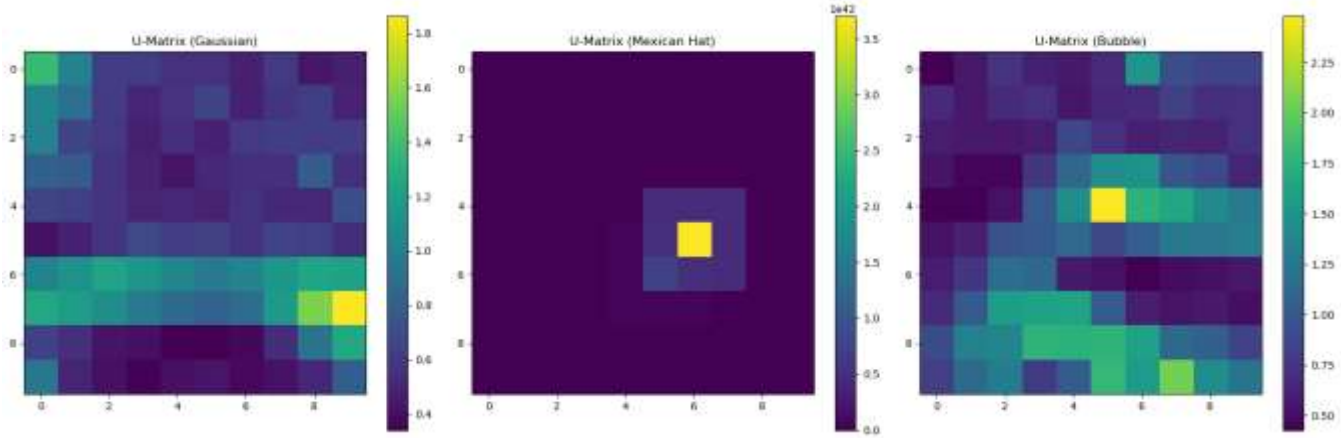

Figure 1

The Gaussian neighborhood (Figure 1a – Picture1 (Right to left)) creates smooth transitions between clusters, preserving global topology effectively, as evidenced by the gradual changes in the U-Matrix.

In contrast, the **Mexican Hat neighborhood** (Figure 1a – Picture2 (Right to left)) produces more distinct cluster boundaries but also some 'repulsion zones' which can be seen as darker areas between clusters.

The Bubble neighborhood (Figure 1a – Picture3 (Right to left)) forms sharp cluster boundaries with less smooth transitions, potentially sacrificing some topological preservation for clearer cluster separation.

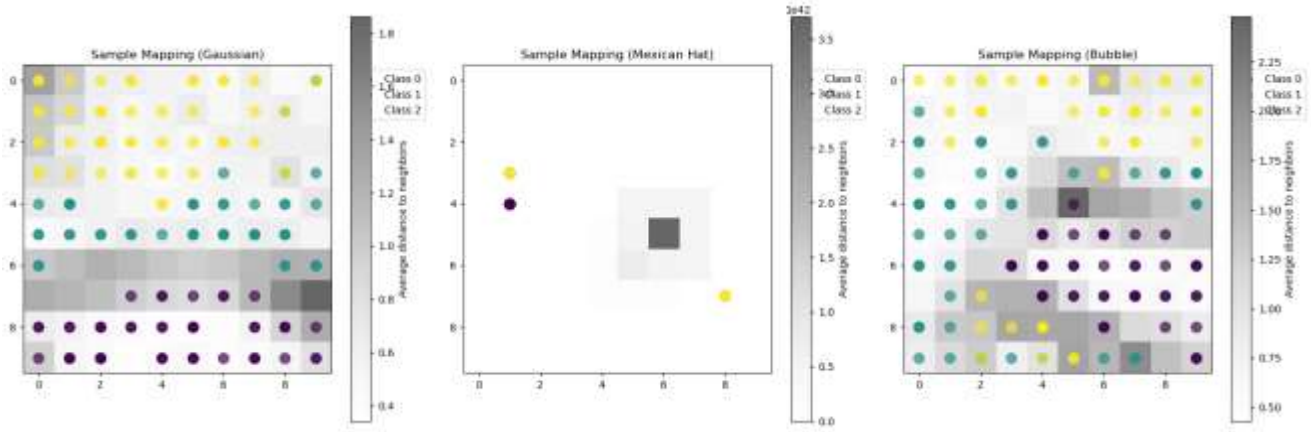

Figure 1a

The U-Matrix visualizations reveal these differences clearly, with Gaussian neighborhood generally showing smoother distance gradients compared to the more abrupt changes in Bubble neighborhood maps.

4.3 Quantitative Evaluation

To quantitatively assess the impact of neighborhood functions, we use two key metrics: **topographic error** and **quantization error.**

1. **Topographic Error** measures how well the SOM preserves the topology (neighborhood relationships) of the input data.
    - **Definition**: It calculates the proportion of input samples for which the second-best matching unit (2nd BMU) is not adjacent to the best matching unit (BMU) on the map grid.
    - **Interpretation**: A low value (closer to 0) indicates better topology preservation, meaning similar data points in the input space remain close in the SOM grid.
    - **Formula**:

      Topographic Error=1/N∑i=1NII(2nd BMU of xi not adjacent to BMU), where II is the indicator function, and NN is the number of samples.

2. **Quantization Error** evaluates how accurately the SOM represents the input data by measuring the average distance between data points and their BMUs.
    - **Definition**: It is the mean squared error between input vectors and their corresponding BMU weight vectors.
    - **Interpretation**: A lower value indicates better representation fidelity (less distortion). However, it does not account for topological relationships.
    - **Formula**:

      Quantization Error=1N∑i=1N‖xi−wBMU‖2

      where wBMUwBMU is the weight vector of the BMU for xixi.

.

Our experiments show that

1. The Gaussian neighborhood generally achieves the lowest topographic error 0.1933, indicating better topology preservation.
2. The Mexican Hat neighborhood can sometimes achieve lower quantization error 0.8658, suggesting better data representation in some cases.
3. The Bubble neighborhood often shows higher topographic error 0.3000, but comparable quantization error to Gaussian 0.1908.

These results confirm that the choice of neighborhood function involves a trade-off between smooth topology preservation and clear cluster delineation.

4.4 Convergence Behavior

Tracking the quantization error during training reveals different convergence patterns (Figure2):

1. Gaussian neighborhood shows steady, smooth convergence
2. Mexican Hat exhibits more oscillations during early training
3. Bubble function converges rapidly initially but may plateau at a higher error level

This suggests that Mexican Hat neighborhoods might benefit from more careful learning rate tuning, while Bubble neighborhoods might require fewer iterations but potentially sacrifice some precision.

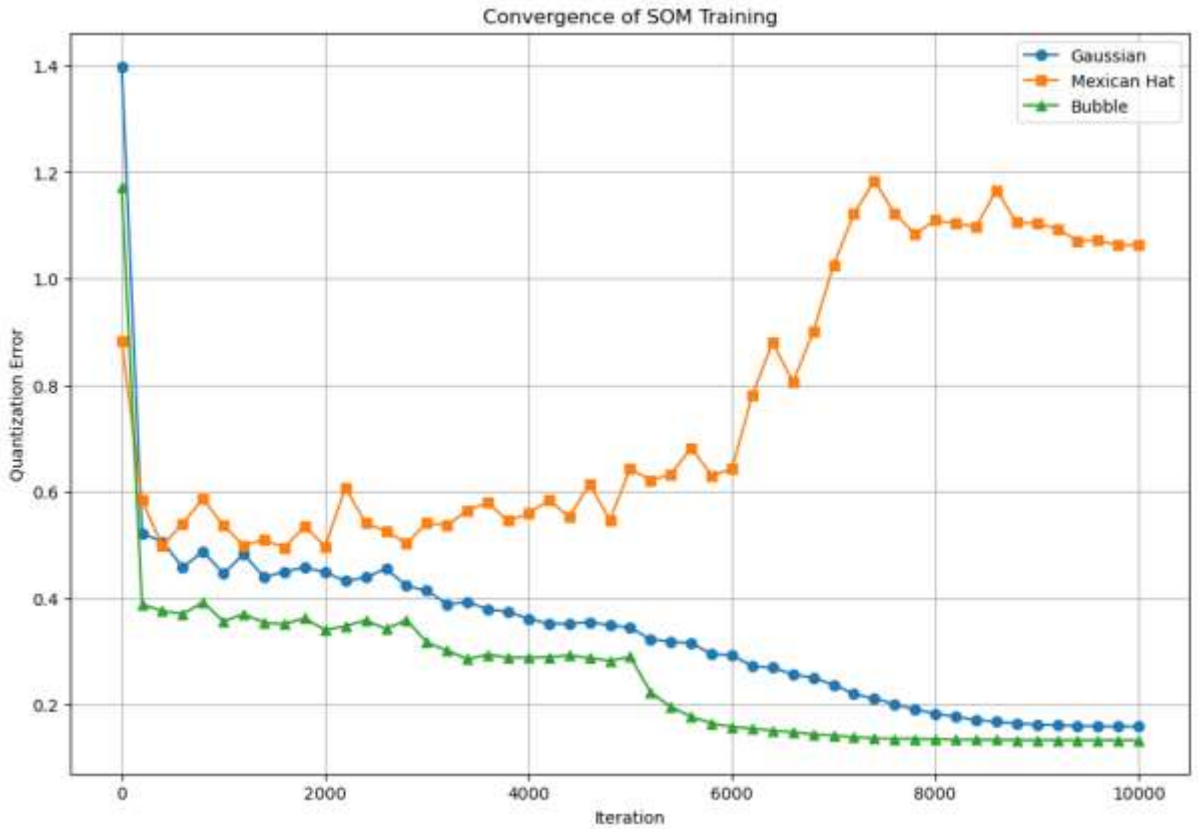

Figure 2

5. Parameter Sensitivity Analysis

The effectiveness of SOMs depends critically on several parameters. Our experiments examine:

5.1 Learning Rate Impact

Testing learning rates from 0.1 to 0.9 reveals:
- Higher learning rates (0.7-0.9) lead to faster initial convergence but may cause instability
- Lower learning rates (0.1-0.3) provide more stable but slower learning
- Moderate learning rates (0.4-0.6) often provide the best balance

5.2 Neighborhood Radius (Sigma)

The sigma parameter controls how wide the influence spreads from the BMU:
- Small sigma values (0.5-1.0) create tight, localized updates, emphasizing local structures
- Larger sigma values (3.0-5.0) spread influence widely, better preserving global topology
- Optimal values depend on both the dataset complexity and the chosen neighborhood function

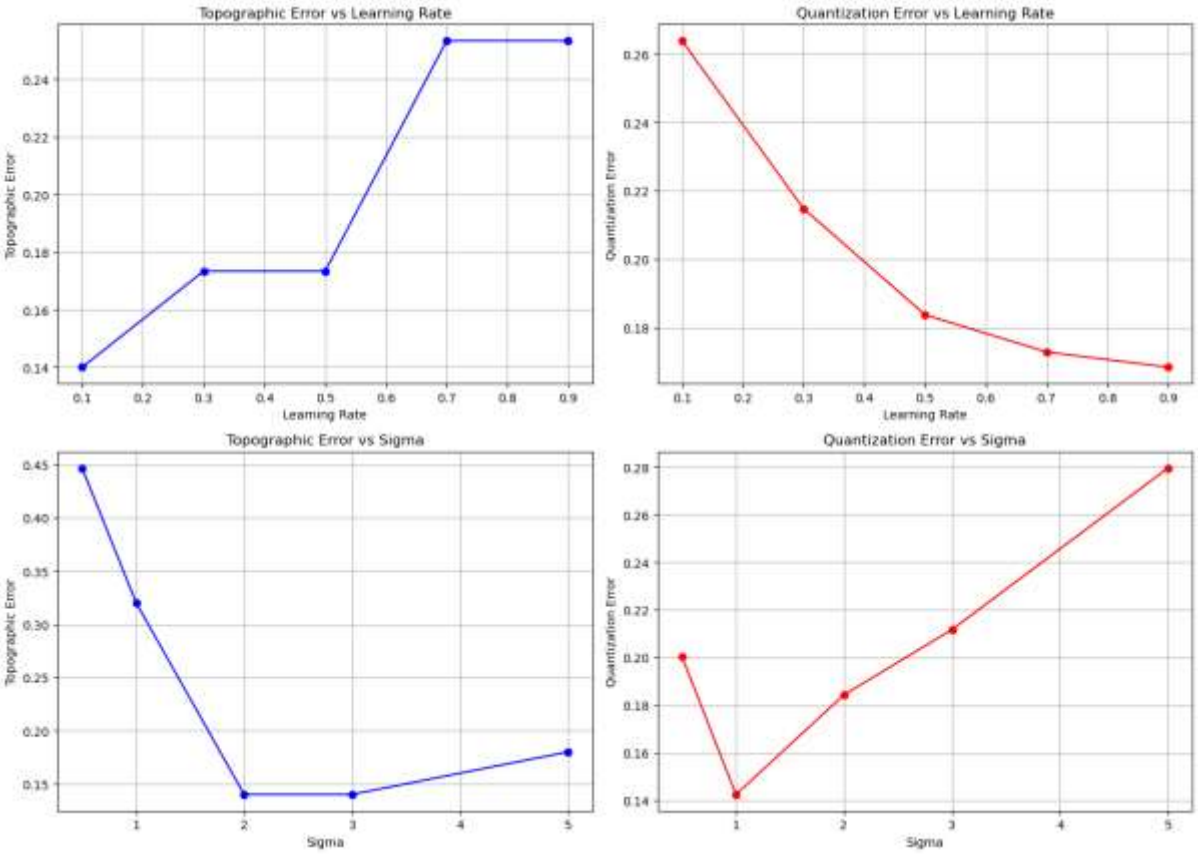

Figure: 3 Learning Rate Impact & Sigma Impact

5.3 Map Size

The dimensions of the SOM grid affect both resolution and computational requirements:
- Smaller maps (5×5) provide faster training but may not capture fine details
- Larger maps (20×20) offer higher resolution but may have empty regions
- Quantization error generally decreases with map size, but topographic error may not

Our results suggest starting with a moderate-sized map (10×10 to 15×15) and adjusting based on specific needs.

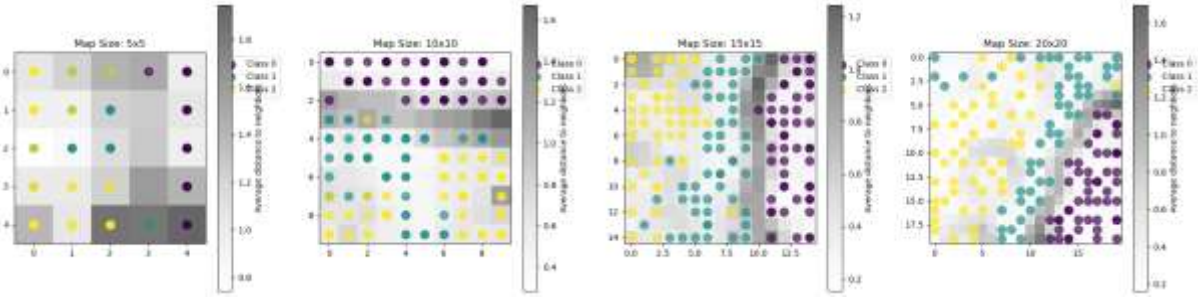

Figure: 3a

6. Practical Applications and Insights

6.1 Comparison with t-SNE

While t-SNE is a popular dimensionality reduction technique, SOMs offer distinct advantages:

1. SOMs provide a structured grid layout that facilitates consistent visualization
2. t-SNE excels at preserving local similarities but loses global structure
3. SOMs allow direct interpretation of neuron weights as prototype vectors

A visual comparison on the Iris dataset (Figure 4) shows t-SNE creating tighter clusters, while SOMs better preserve relative distances between clusters. This aligns with the general characteristics of t-SNE excelling at preserving local similarities but losing global structure, while SOMs provide a structured grid layout that facilitates consistent visualization

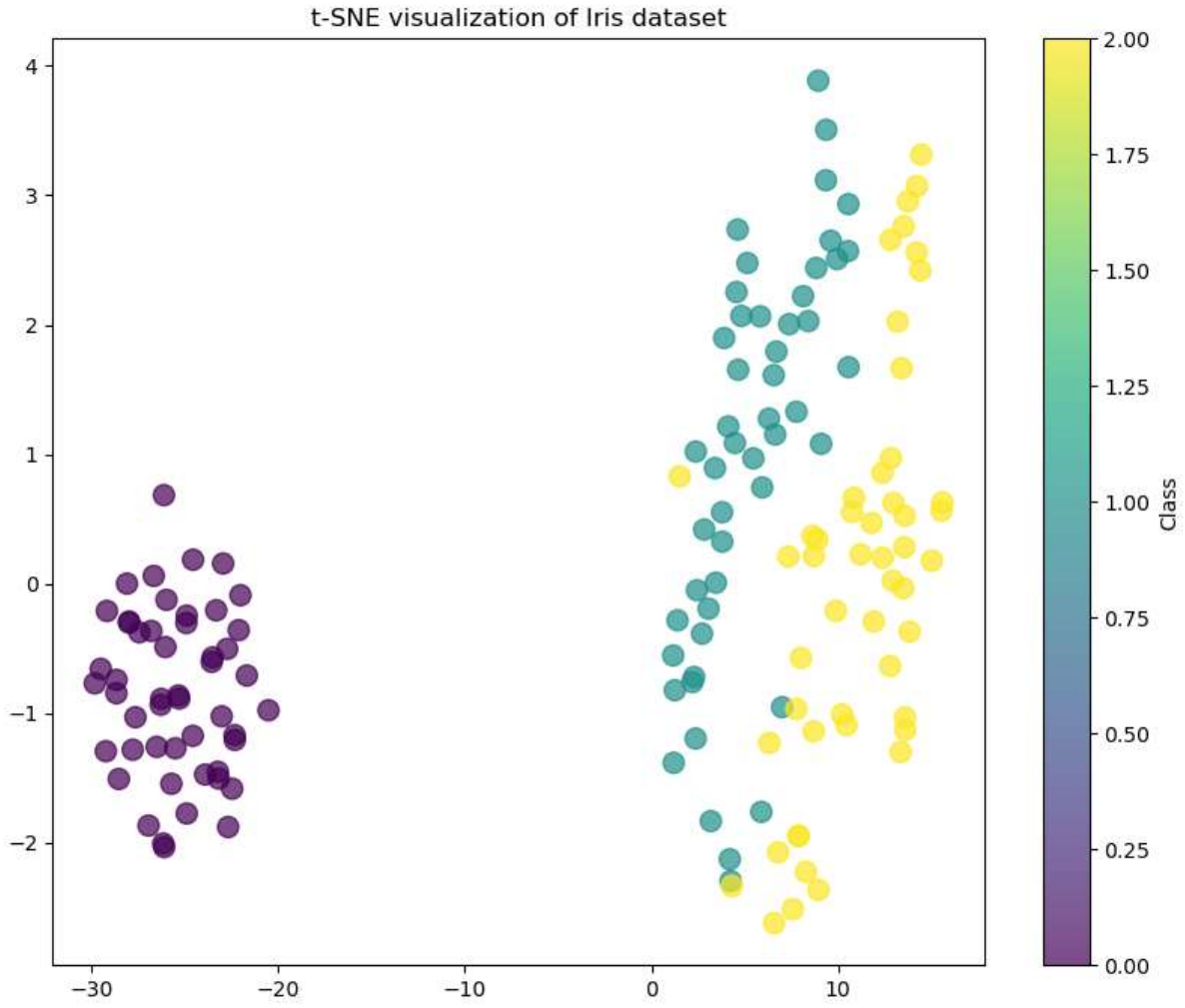

Figure 4

6.2 Choosing the Right Neighborhood Function

Based on our experiments, we recommend:

1. **Gaussian neighborhood:** For general-purpose use and when topology preservation is critical, as demonstrated by its low topographic error and smooth U-Matrix.
2. **Mexican Hat:** When more distinct cluster boundaries are needed and data has clear group structure, but be mindful of potential 'repulsion zones'.
3. **Bubble:** For faster training and when computational efficiency is important, but be aware of the trade-off with topology preservation.

6.3 Visualizing High-Dimensional Data

Our application to the Olivetti faces dataset demonstrates how SOMs can organize face images based on visual similarity (Figure 5), creating an intuitive map where similar faces

are positioned near each other. This highlights SOM's ability to visualize complex, high-dimensional datasets

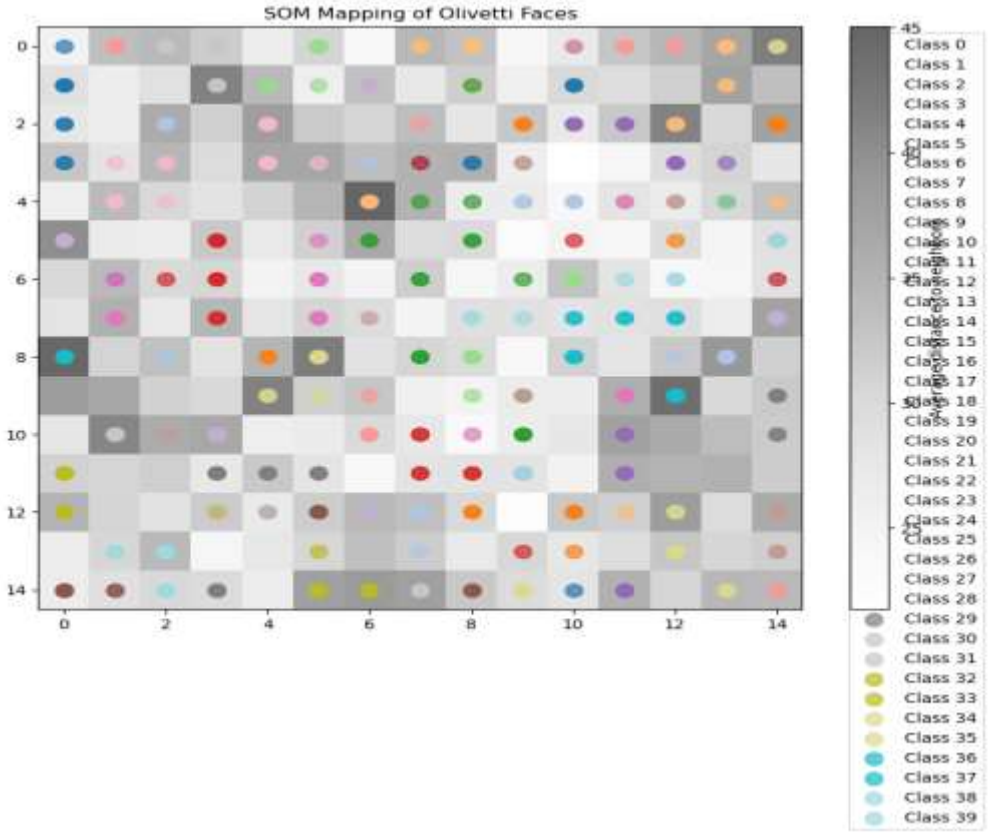

Figure 5

7. Conclusion

Self-Organizing Maps provide a powerful approach to dimensionality reduction and data visualization, with neighborhood functions playing a crucial role in their behavior. Our experiments confirm that the choice of neighborhood function significantly impacts topology preservation, cluster boundary definition, and convergence properties.

The Gaussian neighborhood function generally provides the best topology preservation, while Mexican Hat and Bubble functions may offer advantages in specific scenarios. Understanding these differences enables practitioners to select the most appropriate function for their particular data analysis needs.

Future work could explore adaptive neighborhood functions that change during training or hybrid approaches that combine the strengths of different functions.

References

1. Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480.
2. Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map. IEEE Transactions on Neural Networks, 11(3), 586-600.
3. Ultsch, A., & Siemon, H. P. (1990). Kohonen's self organizing feature maps for exploratory data analysis. In Proceedings of the International Neural Network Conference (pp. 305-308).
4. Pölzlbauer, G. (2004). Survey and comparison of quality measures for self-organizing maps. In Proceedings of the Fifth Workshop on Data Analysis (WDA'04) (pp. 67-82).
5. Van Der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(11).

The link to git repository: **https://github.com/Junaid62-code/SOM-Tutorial---J.Ghaffar**