Introducing NNSOM: A New Python Package for Enhanced Self-Organizing Maps

Ei Tanaka
Data Science Department, Columbian College of Arts and Sciences
George Washinton University
Washington, DC
ei.tanaka@gwu.edu

Lakshmi Sravya Chalapati
Data Science Department, Columbian College of Arts and Sciences
George Washinton University
Washington, DC
sravyach24@gwu.edu

***Abstract*—** Self-Organizing Maps (SOMs) have emerged as a valuable tool for analyzing and visualizing complex, high-dimensional data. In this research paper, we introduce NNSOM, a Python package that implements SOMs with a focus on efficiency, flexibility, and educational value.

I. Introduction

Self-Organizing Maps (SOMs) has become a pivotal tool in the exploration and analysis of high-dimensional data. These networks facilitate mapping complex, nonlinear statistical relationships into simple, comprehensible two-dimensional representations. The newly developed NNSOM Python package stands out as a significant advancement in this domain, leveraging the power of popular libraries like NumPy and CuPy to enhance the implementation and accessibility of SOMs.

NNSOM is crafted with the dual objectives of extensibility and educational utility. Providing a robust foundation encourages researchers to tailor and enhance the library to meet their unique investigative needs. Simultaneously, its clear and logical structure makes it an invaluable educational resource, enabling students to demystify the sophisticated mechanisms underpinning SOMs.

This introduction of NNSOM aims to illuminate its core functionalities, outline its structural advantages, and discuss its potential applications, ranging from data visualization and clustering to dimensionality reduction. By harnessing the capabilities of NNSOM, researchers and students can achieve more profound insights and more effective results in their data analysis endeavors

II. Background and Theorical Foundation

A. *SOMs Overview*

SOMs, known as Kohonen maps, are an unsupervised learning algorithm that Teuvo Kohonen developed in the 1980s. This neural network technique produces a two-dimensional, discretized representation of input data while preserving the topological structure of the data set. SOMs are particularly well-suited for visualizing low-dimensional views of high-dimensional data, akin to reducing data dimensions through principal component analysis.

B. *Algorithm*

The NNSOM package utilizes a streamlined algorithm for implementing SOMs, prioritizing efficiency and precision. The algorithm kicks off with the initialization of the network's weight vectors, typically with small random values. For each iteration, a sample from the dataset is randomly chosen and presented to the network. The algorithm swiftly identifies the Best Matching Unit (BMU), the neuron whose weight vector is closest to the input vector according to a distance metric (usually Euclidean distance). Following the identification of the BMU, the weights of the BMU and its neighboring neurons are adjusted to make them more similar to the input vector. This adjustment is governed by the learning rate and the neighborhood function, which decrease over time. The neighborhood function determines the radius of the neighborhood around the BMU that will be affected by the update, ensuring that the learning process incorporates local smoothing over the map.

C. *Training Process*

The training process of NNSOM is characterized by two main phases: the ordering and convergence phases. The ordering phase takes place over the initial iterations, where significant weight adjustments are made, and the map's topological structure begins to form. During this phase, the neighborhood radius is large, and the learning rate is higher, allowing for significant network modifications. In the convergence phase, the learning rate is reduced, and the neighborhood radius is decreased gradually. This phase is critical for refining the feature mappings and stabilizing the network. The changes made during this phase are finer, focusing on perfecting the representation of the input data in the network. The process continues until the weight changes are minimal, indicating that the network has converged and the training is complete. Throughout both phases, it is crucial to maintain a balance between learning rate and neighborhood size to ensure that the SOM effectively learns the underlying structure of the data without overfitting. This balanced approach helps achieve a robust and reliable model that accurately reflects the complexities of the input data.

III. Pakage Overview

The NNSOM package is a robust implementation of SOMs designed for Python. It emphasizes usability, flexibility, and educational value. As you begin integrating NNSOM into your projects, it is vital to approach the package with a clear organizational strategy.

A. *Archtecture*

The architecture of the NNSOM package is thoughtfully designed to facilitate both ease of use and extendibility, making it ideal for users ranging from academic researchers to industry professionals exploring data patterns through Self-Organizing Maps. Here is a more detailed and clear explanation of its components:

1) *Core Engine:*

The Core Engine of the NNSOM package comprises two main components: the Initialization Module and the Learning Algorithm. The Initialization Module is crucial for setting up the SOM's initial parameters and weight matrices. It offers a strategy for initialization, including principal component analysis (PCA). These strategies can have a significant impact on the convergence speed and the quality of the final mapping. The Learning Algorithm is central to NNSOM, where it meticulously adjusts the network weights through iterative training sessions. This includes competitive learning, where neurons compete to be closest to the input vector, and cooperative learning, where the winning neuron and its neighbors are adjusted to represent the data topology better.

2) Utility Functions:

NNSOM includes a range of utility functions designed to prepare and process data effectively. Data handling is one of the primary utilities necessary for preprocessing the data before input into the SOM visualization. This includes normalization, scaling, and handling data types and structure to ensure the data is well-suited for SOM visualization.

3) Visualization Tools:

After training the SOM, NNSOM provides powerful visualization tools that allow users to interpret and analyze the results effectively. Mapping visualization lets the user view the topological map produced by the SOM, with tools available for creating hit histograms, heat maps, distance maps, U-matrices, and items in each cluster with basic plots such as pie charts, scatter plots, stem plots, and so on. These visualizations are instrumental in identifying clusters and anomalies within the data. Furthermore, the component planes feature offers insights into the feature responses of the SOM neurons by visualizing individual weight vectors as component planes, which can elucidate what features the neurons are most strongly reacting to.

IV. Usage and Examples

A) Example Dataset

A. 1 Introduction to the Iris Dataset

The Iris dataset is a classic dataset in the field of machine learning. It consists of 150 samples of iris flowers, each with four features: sepal length, sepal width, petal length, and petal width. The dataset is commonly used for testing clustering algorithms due to its simple structure and well-defined classes.

A.2. Brief Overview

The Iris dataset is widely used in machine learning and pattern recognition research for its simplicity and clarity. It is often used as a benchmark dataset to test the performance of clustering algorithms, including Self-Organizing Maps (SOMs).

A.3. Dataset Description

- Sepal Length: The length of the iris flower's sepal in centimeters.
- Sepal Width: The width of the iris flower's sepal in centimeters.
- Petal Length: The length of the iris flower's petal in centimeters.
- Petal Width: The width of the iris flower's petal in centimeters.

A.4. Relevance

The Iris dataset is relevant in the context of machine learning because it represents a simple yet effective way to test and compare different clustering algorithms. Its well-defined classes and easily interpretable features make it ideal for educational purposes as well.

A .5. Objective

The objective of applying Self-Organizing Maps (SOMs) to the Iris dataset is to explore the capabilities of SOMs in clustering and pattern recognition. By training a SOM on the Iris dataset, we aim to visualize how the SOM organizes the data into clusters based on the four features provided.

B) Using iris dataset

B.1 Data loading

The Iris dataset is loaded using the load_iris function from sklearn.datasets.

B.2 SOM Configuration

The SOM was configured with a grid size of 4x4 to accommodate the dimensions of the Iris dataset. This grid size was chosen based on the nature of the dataset and the desired level of granularity in the clustering. The total number of neurons is equal to the number of points in the grid, in this example, there are 16 neurons.

B.3 SOM Training Parameters

The SOM was trained using the following parameters:

Number of Epochs: 500

Number of Steps per Epoch: 100

Initial Neighborhood Radius: 3

These parameters were selected to balance the computational efficiency of the training process with the quality of the resulting SOM representation.

B.4 Initialization Method

som.init_w(X,norm_func=scaler.fit_transform): This method initializes the weight vectors of the SOM using the input data X and a normalization function scaler.fit_transform.

B.5 Training the SOM

The SOM is trained using the train method, with parameters such as the initial neighborhood size (Init_neighborhood), number of epochs (Epochs), and steps (Steps).

B.6 Visualization of Training

The training progress of the Self-Organizing Map (SOM) using the Iris dataset was monitored through various visualizations and metrics to assess its development over iterations. One crucial visualization utilized was the U-matrix, a 2D grid where each neuron is depicted by a color representing its similarity to its neighboring neurons. This visualization enabled the identification of clusters and patterns within the Iris dataset, aiding in the understanding of how the SOM was learning and organizing the data.

In addition to the U-matrix, metrics such as the quantization error and the topological error were employed to evaluate the performance during training. The quantization error calculated the average distance between each input vector and its best-matching neuron, providing insight into how well the SOM represented the Iris dataset. The topological error measured the preservation of the topological relationships between neurons in the input space and the SOM's map, indicating the extent to which the SOM retained the original structure of the data.

By monitoring these visualizations and metrics throughout the training process, we were able to assess the SOM's learning progress and make necessary adjustments to improve its performance, ensuring that it effectively clustered and represented the Iris dataset

C) *Analyzing SOM Output*

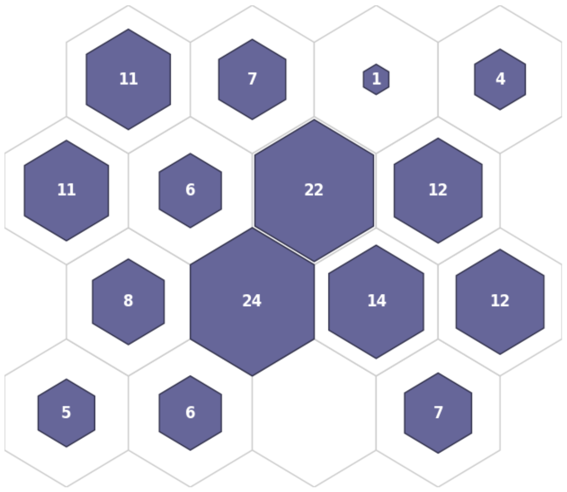
C.1 Data Mapping

In the mapping process, each data point in the Iris dataset, which consists of four features (sepal length, sepal width, petal length, and petal width), is associated with the neuron in the SOM grid that is most similar to it. This similarity is determined using a distance metric, such as Euclidean distance, between the data point and the weight vectors (representing neurons) in the SOM. The neuron with the closest weight vector is considered the "best-matching unit" (BMU) for that data point, and the data point is mapped to this neuron on the SOM grid. This mapping process results in each neuron on the SOM grid being associated with one or more data points from the Iris dataset, forming clusters that represent similar patterns in the data.

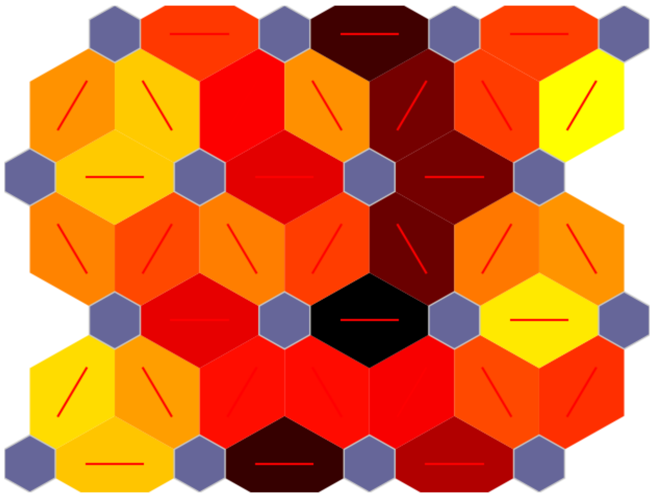
C.2 *Analyze the results*

To analyze the training results, generate plots. For SOM training, the weight vector associated with each neuron moves to become the center of a cluster of input vectors. In addition, neurons that are adjacent to each other in the topology should also move close to each other in the input space, therefore it is possible to visualize a high-dimensional inputs space in the two dimensions of the network topology. The default topology of the SOM is hexagonal.

One visualization tool for the SOM is the Hit Histogram



In the Hit Histogram of the Iris dataset, each neuron in the 4-by-4 grid represents a cluster center. The histogram shows the number of observations associated with each neuron.



Another visualization tool for the SOM is the *neuron distance matrix* (also called the *U-matrix*).
In this figure, the blue hexagons represent the neurons. The red lines connect neighboring neurons.
The colors in the regions containing the red lines indicate the distances between neurons.
The darker colors represent larger distances, and the lighter colors represent smaller distances.
A band of dark segments crosses the map.
The SOM network appears to have clustered the flowers into two distinct groups.

C.3 *Discussion*

The results of applying the Self-Organizing Map (SOM) algorithm to the Iris dataset reveal interesting insights into the clustering and distribution of the Iris flower species based on their features. By mapping the four-dimensional feature space onto a two-dimensional grid, the SOM provides a visual representation that helps identify patterns and relationships within the dataset.

The SOM successfully clustered the Iris dataset into distinct groups corresponding to the three Iris species: Setosa, Versicolor, and Virginica. Each neuron in the SOM grid represents a cluster, and the organization of these clusters reflects the similarities and differences between the Iris species based on their feature vectors.

One of the key findings from the SOM analysis is the clear separation of the Setosa species from the other two species. This separation indicates that the Setosa species is distinct in terms of its feature characteristics compared to Versicolor and Virginica. On the other hand, Versicolor and Virginica show some overlap in their clusters, suggesting a closer relationship in their feature space.

Overall, the SOM analysis of the Iris dataset demonstrates the algorithm's effectiveness in clustering and visualizing high-dimensional data. The results provide valuable insights into the structure of the dataset and can aid in further analysis and classification tasks.

V. INSIGHTS AND IMPLICATIONS

Cluster Analysis: SOM effectively clustered the Iris dataset into distinct groups based on the features of the flowers. This demonstrates the ability of SOM to uncover underlying patterns and structures within the data.
Dimensionality Reduction: By representing the dataset in a lower-dimensional space (2D grid), SOM provides a visually interpretable representation of the data. This can be useful for understanding complex datasets and identifying relationships between variables.
Visualization: The U-matrix and hit histograms provided visualizations that helped interpret the clustering result.These visualizations can be valuable for exploratory data analysis and communication of findings.

VI. CONCLUSION

Overall, the NNSOM package has the potential to significantly impact the way researchers and practitioners approach data analysis and visualization. Its user-friendly interface, coupled with its powerful features, makes it a valuable addition to the toolkit of anyone working with high-dimensional data.

ACKNOWLEDGMENT

We would like to express our deepest gratitude to Professor Amir Jafari and Professor Martin Hagan for their invaluable guidance and mentorship throughout the development of this project. Their expertise and insights have significantly contributed to the success of our research and the creation of the NNSOM package. We are truly thankful for their continuous support and encouragement.

REFERENCES

[1] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesús, "Neural Network Design (2nd Edition)", pp. 616 - 638.

[2] T. Kohonen, “Correlation matrix memories,” IEEE Transactions on Computers, vol. 21, pp. 353–359, 1972.

[3] T. Kohonen, Self-Organization and Associative Memory, 2nd Ed., Berlin: Springer-Verlag, 1987.