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

George Washington University
Columbian College of Arts and Science
Data Science Department

DATS 6501 Capstone
Project Presentation

Group 3
Ei Tanaka, Lakshmi Sravya Chalapati

Contents

Project Objective

The goal of this project is to build a python Self Organizing Map (SOM) package can be used for Neural Network community.

SOMs Overview

- Developed by Teuvo Kohonen in the 1980s
- Unsupervised learning algorithm
- Produces a 2D representation of high-dimensional data
- Preserves the topological structure of the dataset
- Well-suited for visualizing low-dimensional views of high-dimensional data

Algorithm

- NNSOM package uses batch algorithm for SOM implementation
- Initialization of weight vectors with small random values
- Random selection of dataset sample for each iteration
- Identification of Best Matching Unit (BMU) - neuron with closest weight vector to input vector
- Adjustment of BMU and neighboring neurons' weights based on learning rate and neighborhood function
- Neighborhood function determines affected radius around BMU for update, ensuring local smoothing over the map

Training Process

- Two main phases: ordering and convergence
- Ordering phase: initial iterations, significant weight adjustments, formation of map's topological structure
- Convergence phase: learning rate and neighborhood radius reduction, focus on refining feature mappings and stabilizing network
- Continues until weight changes are minimal, indicating convergence and completion of training
- Balance between learning rate and neighborhood size crucial for effective learning without overfitting

Package Overview

1) Core Engine
Initialization Module: Sets up initial parameters and weight matrices
Learning Algorithm: Adjusts network weights through iterative training sessions
Competitive learning: Neurons compete to be closest to input vector
Cooperative learning: Winning neuron and neighbors adjusted to represent data topology

2) Utility Functions
Data handling utilities for preprocessing data before input into SOM visualization
Normalization, scaling, handling data types and structure

3) Visualization Tools
Mapping visualization to view topological map produced by SOM
Tools for creating hit histograms, heat maps, distance maps, U-matrices
Basic plots (pie charts, scatter plots, stem plots) for analyzing clusters and anomalies
Component planes feature to visualize individual weight vectors as component planes, revealing features correlation

Usage and Examples

1. Training
2. Plots
3. Post Training Analysis
4. GPU Usage

https://github.com/amir-jafari/SOM/tree/main/examples/Tabular/Iris/notebook

Documentation

The detailed documentation of NNSOM is generated using Sphinx pages.
These pages were deployed to Git using GitActions.

Check out - https://amir-jafari.github.io/SOM