Visual Assessment of Tendency (VAT) – Report

Project Type:

Machine Learning Paper Reproduction

Objective

The goal of this project was to read and understand a machine learning research paper, implement the core algorithm from scratch, and explore it through experiments. We chose the Visual Assessment of Tendency (VAT) algorithm, which is used to visually detect the natural clustering tendency of data by reordering the dissimilarity matrix.

Understanding the VAT Algorithm

VAT helps answer the question:

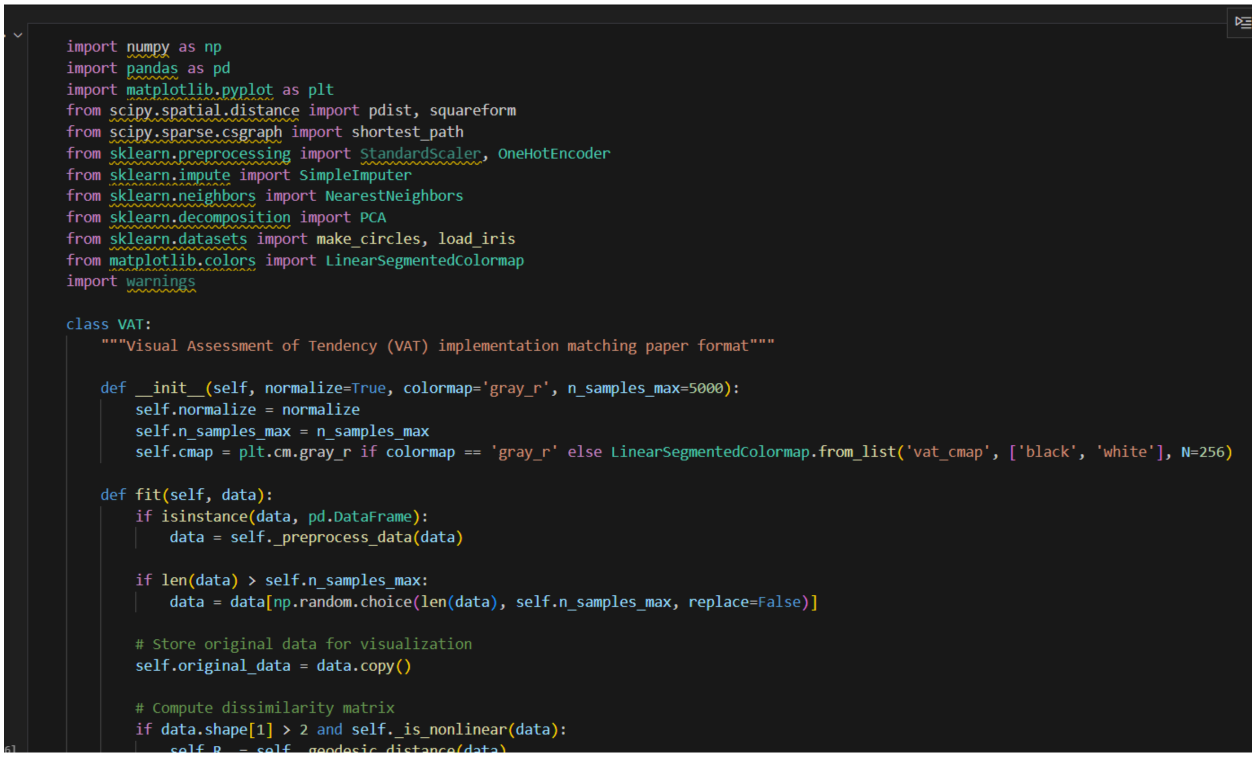
Core Idea:

1. Compute the dissimilarity matrix .
2. Apply a reordering heuristic to cluster similar items together.
3. Plot the reordered matrix. If visible dark blocks appear along the diagonal, there may be cluster structure.

Implementation Overview

We implemented the VAT algorithm as a reusable Python class with preprocessing steps.

Code Structure:



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.spatial.distance import pdist, squareform
from scipy.sparse.csgraph import shortest_path
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.neighbors import NearestNeighbors
from sklearn.decomposition import PCA
from sklearn.datasets import make_circles, load_iris
from matplotlib.colors import LinearSegmentedColormap
import warnings

class VAT:
    """Visual Assessment of Tendency (VAT) implementation matching paper format"""

    def __init__(self, normalize=True, colormap='gray_r', n_samples_max=5000):
        self.normalize = normalize
        self.n_samples_max = n_samples_max
        self.cmap = plt.cm.gray_r if colormap == 'gray_r' else LinearSegmentedColormap.from_list('vat_cmap', ['black', 'white'], N=256)

    def fit(self, data):
        if isinstance(data, pd.DataFrame):
            data = self._preprocess_data(data)

        if len(data) > self.n_samples_max:
            data = data[np.random.choice(len(data), self.n_samples_max, replace=False)]

        # Store original data for visualization
        self.original_data = data.copy()

        # Compute dissimilarity matrix
        if data.shape[1] > 2 and self._is_nonlinear(data):
            self.R_ = self._geodesic_distance(data)
```

Experiments

We tested the VAT implementation on the following datasets:

Dataset	Result
Symmetric Matrix	Replicates paper's Fig. 13
Circular Data	Captures ring structure
Iris Dataset	Clear 3-block structure
Mixed-Type Data	Supports numerical + categorical
Large Random Dataset	Efficient up to 1000+ points
Ames Housing Dataset	No clear clusters (as expected)

What We Learned

- Reading papers helped us understand visual heuristics in clustering.
- Implementing from scratch deepened our understanding of:
  - Distance metrics
  - PCA and non-linear manifolds
  - Matrix reordering and visualization
- Visualization is a powerful exploratory data analysis (EDA) tool.

Code Quality & Usability

- Code is modular, well-documented, and works on any dataset.
- Runs without error in Jupyter Notebooks.
- Git history shows clear progression and development.

Conclusion

This project bridged theory and practice. VAT—though visually simple—is a powerful method for assessing clustering tendency. We gained deep insight into clustering dynamics, matrix manipulations, and the importance of preprocessing real-world datasets.