École pour l'informatique et les techniques avancées, Le Kremlin-Bicêtre

MSc Computer Science (Data Science and Analytics)

DSA Foundations of Statistical Analysis and Machine Learning

Project:

Visual Assessment of Tendency (VAT) Algorithm

Submitted by:

Ganpat Patel
Adnan Ali
Jatinkumar Parmar
Musa Ummar

Submitted to:

Professor Ismael Lachheb

Index

1. Introduction

Clustering is an essential technique in data analysis and pattern recognition. It used to group similar objects together based on shared characteristics. Identifying the natural grouping of data is a fundamental step in various domains such as machine learning, image processing, bioinformatics, and market segmentation. However, before applying formal clustering algorithms like K-Means or Hierarchical Clustering, it is crucial to assess whether the dataset exhibits a clustering tendency.

The Visual Assessment of Tendency (VAT) algorithm is a method designed to visually determine the presence of clustering structures in a dataset before performing clustering. The VAT algorithm achieves this by computing a dissimilarity matrix and reordering it using a Minimum Spanning Tree (MST)-based approach. The reordered dissimilarity matrix provides a visual representation where clusters appear as darker square-shaped blocks along the diagonal, offering insight into the natural partitions within the data.

VAT is particularly useful for unsupervised learning tasks, where there is no prior knowledge about the number of clusters present in the dataset. Unlike direct clustering methods, which may impose assumptions on the number and shape of clusters, VAT provides a data-driven approach to evaluate whether clustering is meaningful. This preliminary assessment can guide practitioners in choosing appropriate clustering algorithms and parameters.

This report provides an in-depth exploration of the VAT algorithm, detailing its conceptual framework, step-by-step implementation, and experimental application on multiple datasets, including synthetic and real-world data. The results demonstrate the algorithm's effectiveness in identifying clustering tendencies and improving the interpretability of data distributions.

2. VAT Algorithm

2.1 Concept

The VAT algorithm is based on the principle that a well-ordered dissimilarity matrix can reveal underlying cluster structures in a dataset. The algorithm leverages Prim's Minimum Spanning Tree (MST) approach to determine an optimal reordering of the data points that enhances cluster visualization. By transforming the original dissimilarity matrix into a structured format, VAT highlights natural clusters in the dataset, making it easier to interpret potential groupings before applying clustering algorithms.

The primary goal of VAT is to identify clustering tendencies without making explicit assumptions about the number of clusters or their distribution. Unlike traditional clustering algorithms that partition data directly, VAT provides an exploratory visualization tool to assess whether clustering is an appropriate technique for the given dataset.

2.2 Steps of the Algorithm

1. **Compute the Pairwise Distance Matrix:**
    - The Euclidean distance between each pair of data points is calculated to construct a dissimilarity matrix.
    - The dissimilarity matrix represents how different each data point is from every other point.
2. **Determine an Initial Point:**
    - The algorithm selects an initial point that is the most distant from an arbitrarily chosen starting point.
    - This ensures that the ordering begins with a well-separated data point, helping to enhance cluster differentiation.
3. **Iteratively Reorder Points:**
    - Using a Minimum Spanning Tree (MST) approach, the algorithm iteratively selects and adds the next closest point to the reordered sequence.
    - The goal is to minimize the maximum dissimilarity at each step, effectively restructuring the dataset for better visualization.
4. **Reorder the Dissimilarity Matrix:**
    - The original dissimilarity matrix is reordered according to the new sequence of points.
    - Clusters appear as dark square blocks along the diagonal, making them visually distinguishable.
5. **Visualize Results:**
    - The reordered dissimilarity matrix is plotted as a heatmap to assess the clustering structure.
    - A scatter plot of the original and reordered data points is also generated to illustrate the impact of VAT on the dataset.

3. Implementation

The VAT algorithm is implemented in Python using **NumPy, Matplotlib, and SciPy**. Below is the function implementation:

1. Function Definition & Docstring

```
def vatManual(X):
    """
    Visual Assessment of Tendency (VAT) implementation:

    Parameters:
    X : ndarray
        The input data matrix (objects as rows, features as columns).

    Returns:
    R : ndarray
        The reordered dissimilarity matrix.

        Displays 4 graphs:
        1. Scatter Plot of Data (Before VAT)
        2. Scatter Plot of Data (After VAT)
        3. Dissimilarity Matrix (Before VAT)
        4. Dissimilarity Matrix (After VAT)
    """
```

The function vatManual takes a 2D NumPy array X as input, where rows represent objects (data points), and columns represent features.

- The function returns R, which is the reordered dissimilarity matrix.
- The function plots four graphs to visually compare the data and dissimilarity matrices before and after VAT.

2. Compute the Pairwise Distance Matrix

```
# Step 1: Compute the pairwise distance matrix
D = dist.squareform(dist.pdist(X, metric='euclidean'))  # Convert the pairwise distances into a square matrix.
# 'dist.pdist' computes the pairwise distance between each pair of points in X using the Euclidean metric.
# 'dist.squareform' converts the condensed distance matrix (1D) into a square matrix (2D).
```

What happens here?

1. dist.pdist(X, metric='euclidean')
    - Computes pairwise Euclidean distances between all data points in X.
    - Returns a condensed 1D array of pairwise distances.
2. dist.squareform(...)
    - Converts the 1D condensed distance array into a 2D square matrix D.
    - This square matrix has dimensions (n × n), where n is the number of data points.
    - Each entry D[i, j] represents the Euclidean distance between data points i and j.

3. Initialize VAT Ordering Using Prim's MST Approach

```
# Step 2: VAT ordering (Prim's Minimum Spanning Tree-based algorithm)
n = D.shape[0]  # Get the number of points (objects) in the dataset (size of the square dissimilarity matrix).
P = np.zeros(n, dtype=int)  # Initialize an array P to store the reordered indices (of size n).
J = np.arange(n)  # Create an array J of available indices (0, 1, 2, ..., n-1).
i = np.argmax(D[0])  # Find the index of the point that is most distant from the first point in the dataset.
P[0] = i  # Set the first index of the reordered list as the most distant point.
J = np.delete(J, i)  # Remove this point from the list of available indices J.
```

What happens here?

- n = D.shape[0]
  - Stores the number of objects (data points).
- P = np.zeros(n, dtype=int)
  - Initializes an array P to store the reordered indices.
- J = np.arange(n)
  - Creates a list J of all object indices from 0 to n-1.
- i = np.argmax(D[0])
  - Finds the index of the most distant point from the first row of D.
- P[0] = i
  - Assigns this point as the starting point for the VAT ordering.
- J = np.delete(J, i)
  - Removes this point from J so that it won't be selected again.

4. Iterative Reordering Using MST Approach

```
# Step 3: Iteratively reorder the points using Prim's Minimum Spanning Tree (MST) approach
for r in range(1, n):
    # For each remaining point, find the one that is most similar (minimizing the maximum dissimilarity) to the reordered points.
    i = J[np.argmin(D[P[:r]][:, J].max(axis=0))]  # Find the next point with the minimum dissimilarity to the existing reordered points.
    P[r] = i  # Assign this point to the r-th position in the reordered list P.
    J = np.delete(J, np.where(J == i))  # Remove the selected point from the list of available points J.
```

What happens here?

- For each remaining point, the algorithm finds the next most similar point based on dissimilarity.
- How does this work?
  1. D[P[:r]][:, J] extracts a submatrix of distances between already selected points (P[:r]) and remaining points (J).
  2. .max(axis=0) computes the maximum dissimilarity for each candidate point.
  3. np.argmin(...) finds the index of the point that minimizes the maximum dissimilarity.
  4. The selected point is added to P and removed from J.

5. Reorder the Dissimilarity Matrix

R = D[np.ix_(P, P)]

What happens here?

- Uses advanced NumPy indexing to rearrange rows and columns of D based on the new order stored in P.
- The resulting matrix R reveals potential clusters as darker square blocks along the diagonal.

6. Visualization (Plot 4 Graphs)

fig, axes = plt.subplots(2, 2, figsize=(10, 8))

- Creates a 2×2 grid of subplots for the four visualizations.

6.1 Plot: Dissimilarity Matrix (Before VAT)

```
# Left plot: Show dissimilarity matrix before VAT
axes[1, 0].imshow(D, cmap='gray', aspect='equal')  # Display the dissimilarity matrix as an image in grayscale.
axes[1, 0].set_title("Dissimilarity Matrix (Before VAT)")  # Title for the left subplot.
axes[1, 0].set_xlabel("Objects")  # Label for the x-axis.
axes[1, 0].set_ylabel("Objects")  # Label for the y-axis.
fig.colorbar(axes[1, 0].imshow(D, cmap='gray'), ax=axes[1, 0], label="Dissimilarity")  # Add a color bar to indicate the scale of dissimil

# Right plot: Scatter plot of the data
```

- Displays the original dissimilarity matrix (D) using a grayscale heatmap.
- Darker values represent closer points, and lighter values represent distant points.

6.2 Plot: Scatter Plot of Data (Before VAT)

```
# Right plot: Scatter plot of the data
axes[0, 0].scatter(X[:, 0], X[:, 1], s=10, alpha=0.7)  # Create a scatter plot of the first two features (columns) of the data X.
axes[0, 0].set_title("Scatter Plot of Data (Before VAT)")  # Title for the right subplot.
axes[0, 0].set_xlabel("Feature 1")  # Label for the x-axis.
axes[0, 0].set_ylabel("Feature 2")  # Label for the y-axis.
```

- Plots the original dataset using a scatter plot.

6.3 Plot: Dissimilarity Matrix (After VAT)

```
# Bottom row:
# Left plot: VAT-reordered dissimilarity matrix
im = axes[1, 1].imshow(R, cmap='gray', aspect='equal')  # Display the VAT-reordered dissimilarity matrix as an image.
axes[1, 1].set_title("Dissimilarity Matrix (After VAT)")  # Title for the left subplot.
axes[1, 1].set_xlabel("Objects")  # Label for the x-axis.
axes[1, 1].set_ylabel("Objects")  # Label for the y-axis.
fig.colorbar(axes[1, 1].imshow(R, cmap='gray'), ax=axes[1, 1], label="Dissimilarity")  # Add a color bar to indicate the scale of dissimilari
```

- Displays the reordered dissimilarity matrix (R).
- If clusters exist, darker square blocks appear along the diagonal.

6.4 Plot: Scatter Plot of Data (After VAT)

```
# Right plot: Scatter plot of the reordered data
axes[0, 1].scatter(R[:, 0], R[:, 1], s=10, alpha=0.7)  # Create a scatter plot of the first two features of the reordered dissimilarity matri
axes[0, 1].set_title("Scatter Plot of Data (After VAT)")  # Title for the right subplot.
axes[0, 1].set_xlabel("Feature 1")  # Label for the x-axis.
axes[0, 1].set_ylabel("Feature 2")  # Label for the y-axis.
```

- Plots the reordered data (though this may not always be meaningful in 2D).

7. Adjust Layout & Show the Plots

```
plt.tight_layout()  # Automatically adjust the spacing between subplots for better layout.
plt.show()
```

- plt.tight_layout() ensures clear spacing between subplots.
- plt.show() displays all the visualizations simultaneously.

8. Return the Reordered Dissimilarity Matrix

```
return R  # Return the reordered dissimilarity matrix.
```

- The function returns R, which can be used for further analysis.

4. Experimental Results

The VAT algorithm is applied to four datasets: Synthetic Blobs, Iris, Wine, and Breast Cancer. The results demonstrate the effectiveness of VAT in visualizing clustering tendencies before applying a formal clustering method.

4.1 Synthetic Data

The synthetic dataset, generated using make_blobs(), consists of three well-separated clusters. The original scatter plot shows an apparent separation, which is further confirmed by the VAT algorithm. The initial dissimilarity matrix appears random, but after VAT reordering, the reordered matrix displays distinct dark square blocks along the diagonal, indicating strong cluster structures. This confirms the ability of VAT to enhance cluster visualization before applying clustering methods like K-Means.

4.2 Real-World Datasets

VAT is also applied to Iris, Wine, and Breast Cancer datasets, each with distinct characteristics. The results are analyzed as follows:

- **Iris Dataset:** This dataset consists of three known clusters (corresponding to three species of iris flowers). VAT effectively reveals this natural structure by reordering the dissimilarity matrix, making the cluster boundaries clearer.
- **Wine Dataset:** The Wine dataset contains 13 features used to classify wine into three different classes. The reordered VAT matrix shows clustering tendencies, suggesting that clustering methods can be meaningfully applied.
- **Breast Cancer Dataset:** This dataset contains data on malignant and benign tumors. The VAT algorithm helps in visually confirming the presence of two main clusters, which aligns with the classification of tumors as malignant or benign.

Overall, the experimental results confirm that VAT is an effective tool for assessing clustering tendency before applying formal clustering algorithms. The ability to visualize cluster structures in an intuitive manner allows data analysts to make informed decisions on appropriate clustering techniques.

5. Conclusion

The Visual Assessment of Tendency (VAT) algorithm is a valuable tool for pre-clustering analysis, allowing for an intuitive assessment of the presence of natural clusters within a dataset. By leveraging pairwise distance calculations and Minimum Spanning Tree (MST)-based reordering, VAT transforms the dissimilarity matrix into a structured form, revealing potential clustering patterns before applying formal clustering methods.

Through experimentation on synthetic and real-world datasets (Iris, Wine, and Breast Cancer), VAT effectively demonstrated its ability to highlight inherent cluster structures. The reordered dissimilarity matrices clearly exhibited distinct block formations along the diagonal, reinforcing the presence of well-separated clusters. These findings confirm VAT's usefulness as a preliminary diagnostic step for guiding the selection of appropriate clustering techniques, such as K-Means, Hierarchical Clustering, or DBSCAN.

Despite its strengths, VAT has scalability limitations, particularly with large datasets where the computational complexity of pairwise distance calculations can become prohibitive. Future advancements could involve optimizing VAT for big data applications, integrating it with deep learning methods for improved feature extraction, or enhancing its efficiency through parallel computing techniques.

In conclusion, VAT serves as a simple yet effective visualization tool that enhances the interpretability of clustering tendencies in data. Its ability to provide a data-driven insight into the natural grouping of data points makes it a valuable addition to the toolkit of data scientists and analysts working on unsupervised learning tasks.

6. Referneces

1. https://www.researchgate.net/publication/3950332_VAT_A_tool_for_visual_assessment_of_cluster_tendency
2. https://www.scirp.org/journal/paperinformation?paperid=17956