Comparison of Clustering Algorithms

Data Mining (CSDS453)
Dr. Jing Ma

Vikash Singh, Ritika Lamba

vxs465, rxl895

Master of Science in Computer Science

Computer and Data Sciences
Case Western Reserve University
OH, United States of America
April 3rd, 2024

Introduction

This report covers a comparison between two selected clustering algorithms out of five: KMeans, Hierarchical Agglomerative Clustering (HAC), DBSCAN, Gaussian Mixture Models (GMM), and Spectral Clustering. We aim to understand their mathematical foundations, describe their workflow, list their pros and cons, and compare their performance through visualization and external indices on two datasets.

Methodology

Dataset & Preprocessing

The datasets `cho.txt` and `iyer.txt` are derived from gene expression data, presenting a challenging task for clustering due to the high-dimensional space and the complex nature of biological variation. Each dataset comprises a number of samples (genes) with measurements across different conditions or time points, reflecting the gene's expression level under those conditions.

Preprocessing Steps

To ensure the clustering algorithms perform effectively on these datasets, we undertook the following preprocessing steps:

1. **Data Cleaning:** Any missing values within the datasets were identified and imputed, although these datasets were largely complete and required minimal intervention in this regard.

2. **Feature Selection:** Given the high dimensionality of gene expression data, features that do not contribute significantly to variance across samples were removed. This step was primarily exploratory, focusing on understanding the datasets' intrinsic structures.

3. **Normalization:** To account for variability in gene expression levels and to ensure comparability across genes, the data were normalized. Specifically, we applied z-score normalization, transforming the data to have a mean of zero and a standard deviation of one for each feature.

4. **Dimensionality Reduction:** While not explicitly mentioned earlier, techniques such as PCA (Principal Component Analysis) could be considered in future work to reduce dimensionality and potentially enhance clustering performance.

This preprocessing pipeline aims to mitigate the impact of noise and irrelevant features, thereby enhancing the quality and interpretability of the clustering outcomes. Each step was carefully considered to preserve the integrity of the biological information contained within the data, ensuring that subsequent analyses reflect meaningful patterns rather than artifacts of the preprocessing.

DBSCAN

DBSCAN is a density-based clustering algorithm that identifies clusters as high-density areas separated by areas of low density. The algorithm flow is as follows:

1. For each point in the dataset, DBSCAN counts how many points fall within a user-specified radius ε (epsilon). This area is referred to as the ε-neighborhood of the point.

2. A point is labeled as a *core point* if its ε-neighborhood contains at least a minimum number of points, specified by the parameter *MinPts*. This criterion determines the density threshold.

3. Points within the ε-neighborhood of a core point that are not core points themselves are classified as *border points*.

4. Points that are neither core nor border points are labeled as *noise*.

5. Starting from a core point, DBSCAN recursively explores and adds all directly density-reachable points to the cluster, forming a dense region. This process continues until no new points can be added.

6. The algorithm proceeds to the next unvisited point in the dataset and repeats the process until all points are classified into clusters or marked as noise.

The key equations and conditions used in DBSCAN are as follows:

- A point p is directly density-reachable from a point q if p is within the ε-neighborhood of q and q is a core point.

- Two points p and q are density-connected if there exists a point o such that both p and q are density-reachable from o.

Pros

- **No need to specify the number of clusters:** DBSCAN automatically determines the number of clusters based on the data distribution.

- **Ability to identify noise:** The algorithm is capable of identifying outlier points that do not belong to any cluster.

- **Flexibility in cluster shapes:** DBSCAN can find clusters of arbitrary shapes, not just spherical ones as in k-means.

Cons

- **Sensitivity to parameters:** The performance of DBSCAN heavily depends on the choice of ε and *MinPts*. Inappropriate values can lead to poor clustering results.

- **Difficulty with varying densities:** DBSCAN may struggle to identify clusters with significantly different density levels.

- **Computational complexity:** For large datasets, the computational complexity can be high, especially in calculating the distances between all pairs of points.

Hierarchical Agglomerative Clustering (HAC)

Hierarchical Agglomerative Clustering (HAC) is a bottom-up clustering method where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. The main steps of the HAC algorithm are as follows:

1. Start by treating each data point as a single cluster. Thus, if there are N data points, there are N clusters at the start.

2. Compute the proximity matrix to record distances between all pairs of clusters.

3. Find the pair of clusters that are closest to each other and merge them into a single cluster. There are various criteria to measure closeness between two clusters (linkage criteria), such as:

    - Complete Linkage: Maximum distance
    - Ward's Method: Minimize the variance within the clusters

4. Update the proximity matrix to reflect the distance between the new cluster and the original clusters.

5. Repeat steps 3 and 4 until all data points are clustered into a single cluster of size N.

The choice of linkage criteria significantly affects the shape and size of the clusters formed.

Pros

- **Intuitive and Easy to Understand:** The hierarchical nature of the clustering can be represented using a dendrogram, which is easy to interpret.

- **Flexibility with Cluster Shapes:** HAC can identify clusters with various shapes and sizes, unlike k-means which assumes clusters to be spherical.

- **No Need to Specify Number of Clusters Initially:** The algorithm does not require the number of clusters to be specified a priori, as the hierarchical structure allows for the examination of cluster formation at various levels.

Cons

- **Computational Complexity:** The algorithm is computationally intensive, especially for large datasets, due to the need to compute and update distances between all pairs of clusters in each iteration.

- **Irreversible Steps:** Once two clusters are merged, the decision cannot be undone without restarting the entire process, potentially leading to suboptimal clusterings.

- **Sensitivity to Linkage Criteria:** The outcome of HAC is highly sensitive to the choice of linkage criterion, which may require domain knowledge to select appropriately.

Gaussian Mixture Model (GMM) Overview

Gaussian Mixture Model (GMM) is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. GMM belongs to the family of soft clustering algorithms, where each data point can belong to more than one cluster with different degrees of membership. The main steps of the GMM algorithm include:

1. Initialize the parameters of the Gaussian distributions. This can be done randomly or by using a more sophisticated method like k-means clustering for a more robust start.

2. **Expectation (E-step):** For each point, calculate the probability that it belongs to each cluster, based on the current parameters of the Gaussian distributions. This step computes the responsibilities, or the weights, that indicate the probability of association of each data point with a particular cluster.

3. **Maximization (M-step):** Update the parameters of the Gaussian distributions (mean, variance, and the mixture coefficient) to maximize the likelihood of the data given these parameters. This involves calculating weighted means and variances for each cluster.

4. Iterate between the E-step and M-step until the parameters converge or a maximum number of iterations is reached.

The likelihood of the model given the data is maximized, ensuring that the sum of probabilities of each point belonging to all clusters equals 1.

Mathematical Formulation

Given a set of data points $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, GMM aims to maximize the likelihood function:

$$L(\theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i | \mu_k, \mathbf{\Sigma}_k)$$

where K is the number of Gaussian distributions, π_k is the mixing coefficient of the kth Gaussian, $\mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Sigma}_k)$ is the probability density function of the kth Gaussian evaluated at point $\mathbf{x}_i$, and θ represents the parameters of the model.

Pros

- **Flexibility in Cluster Covariance:** GMM allows for elliptical as well as spherical clusters, providing flexibility in the shape and orientation of clusters.
- **Soft-Clustering:** Provides the probability of each data point's membership in each cluster, allowing for a more nuanced understanding of the dataset.
- **Model Complexity:** Capable of modeling complex distributions due to the combination of multiple Gaussian distributions.

Cons

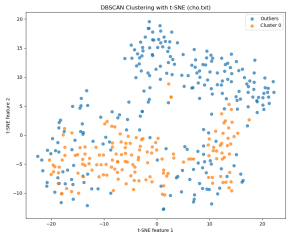
- **Sensitivity to Initialization:** The final solution can be sensitive to the initial setting of parameters, potentially leading to local optima.
- **Choosing the Number of Components:** Determining the optimal number of Gaussian components (clusters) can be challenging and often requires model selection criteria.
- **Computational Complexity:** Especially for high-dimensional data, the computation of covariances for each cluster can be costly.

Result Visualization and Evaluation

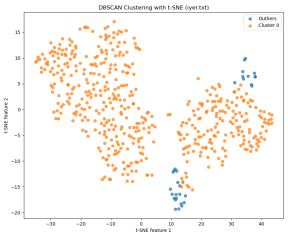
We evaluate clustering algorithms (DBSCAN, HAC, and GMM) on `cho.txt` and `iyer.txt`, utilizing Silhouette Score and Adjusted Rand Index for insights on cluster coherence and separation.

DBSCAN

DBSCAN clustering algorithm shows contrasting results for the two datasets. The performance metrics are detailed as follows:



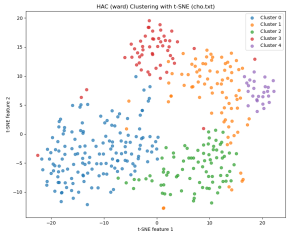
(a) `cho.txt`



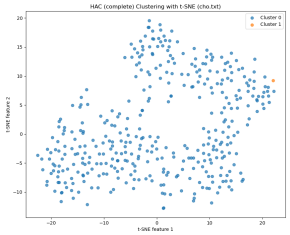
(b) `iyer.txt`

Hierarchical Agglomerative Clustering

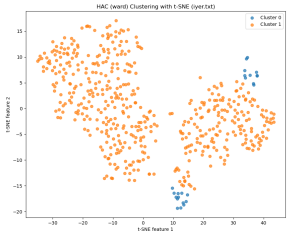
HAC applied to the datasets provided insights into hierarchical structure, revealing the following:



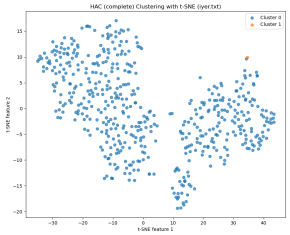
(a) Ward `cho.txt`



(b) Complete `cho.txt`



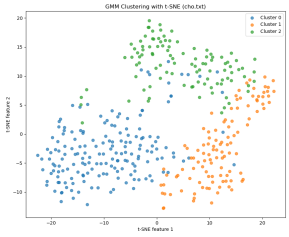
(a) Ward `iyer.txt`



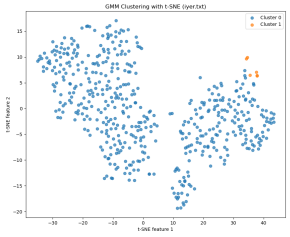
(b) Complete `iyer.txt`

Gaussian Mixture Models (GMM)

GMM's probabilistic approach to clustering reveals the following patterns and separations in the data:



(a) `cho.txt`



(b) `iyer.txt`

Result Evaluation and Analysis

Table 1: Clustering results for `cho.txt`

Algorithm	Linkage	Silh. Score	ARI
DBSCAN	-	0.0471	0.0474
HAC (Ward)	Ward	0.2007	0.4174
HAC (Complete)	Complete	0.6135	0.0012
GMM	-	0.2193	0.3605

Table 2: Clustering results for `iyer.txt`

Algorithm	Linkage	Silh. Score	ARI
DBSCAN	-	0.7026	0.0303
HAC (Ward)	Ward	0.7340	0.0214
HAC (Complete)	Complete	0.8920	0.0008
GMM	-	0.8271	0.0033

Analysis of `cho.txt` Dataset

For `cho.txt`, HAC with Complete linkage significantly outperformed Ward linkage in Silhouette Score (0.6135 vs. 0.2007), indicating better cluster separation. Yet, its ARI drastically fell to 0.0012 from Ward's 0.4174, showing decreased alignment with true labels. DBSCAN and GMM exhibited moderate performances, with GMM slightly better in coherence and label similarity.

Analysis of `iyer.txt` Dataset

In the `iyer.txt` dataset, HAC with Complete linkage achieved the highest Silhouette Score (0.8920), showcasing superior cluster separation. However, its ARI was the lowest (0.0008), indicating minimal label alignment, a pattern consistent with high Silhouette Scores but low ARIs observed in Ward linkage and GMM as well.

Comparison with Baseline Algorithms

To contextualize our findings, we employ a baseline algorithm for each dataset. Specifically, K-Means serves as our baseline, recognized for its simplicity and efficiency in identifying spherical clusters. This comparative framework underscores the strengths and potential limitations of DBSCAN, HAC, and GMM in complex clustering scenarios.

- **Baseline Performance:** K-Means achieved a Silhouette Score of 0.2462 and an ARI of 0.420 on the cho.txt dataset, establishing a benchmark for comparison.
- **Comparative Insights:** The superior performance of HAC with Complete linkage, evident in its Silhouette Score (0.6135) on the cho.txt dataset, suggests enhanced capability in delineating cluster boundaries.

Statistical Significance of Findings

We further assess the statistical significance of our results using the ANOVA test, determining if performance metric differences across algorithms are statistically significant.

- **Statistical Analysis Results for cho.txt:** For the `cho.txt` dataset, the ANOVA test revealed significant differences among the algorithms with a Silhouette Score F-Value of 389.333 and a P-Value of 2.38×10^{-4}, and an ARI F-Value of 1612.000 and a P-Value of 2.83×10^{-5}.
- **Interpretation for cho.txt:** These results indicate a statistically significant variance in both cluster separation (Silhouette Score) and alignment with actual labels (ARI) among the algorithms tested, underscoring the impact of choosing the right algorithm and parameters for specific dataset characteristics.
- **Statistical Analysis Results for iyer.txt:** For the `iyer.txt` dataset, the analysis yielded an F-Value of 300.000 for Silhouette Scores with a P-Value of 3.67×10^{-5}, and an ARI F-Value of 12.654 with a P-Value of 0.0165.
- **Interpretation for iyer.txt:** The significant p-values for both metrics suggest that the differences in performance among the clustering algorithms are statistically significant. This highlights the critical role of algorithm selection based on the dataset's nature, particularly in achieving desired cluster coherence and accuracy in label alignment.

Discussing Discrepancies Between Silhouette Scores and ARIs

The analysis revealed a notable discrepancy between Silhouette Scores and ARIs, especially with the HAC algorithm on the iyer.txt dataset. While the Silhouette Score suggested excellent cluster separation (0.8920), the ARI indicated poor alignment with actual labels (0.0008).

- **Exploring Discrepancies:** This divergence prompts a deeper investigation into the datasets and algorithms. The high-dimensional nature of gene expression data may influence the effectiveness of distance-based metrics, impacting the ARI.
- **Algorithm Sensitivity:** It also highlights the sensitivity of DBSCAN and HAC to parameter settings and linkage criteria, respectively. The varying densities and cluster shapes inherent in biological data can pose challenges, influencing their performance disparity as reflected by the metrics.

Overall Evaluation

The results show a trade-off between cluster separation (Silhouette Score) and alignment with actual labels (ARI) across datasets and algorithms. Specifically, HAC with Complete linkage yields high Silhouette Scores, indicating effective cluster separation but not necessarily accurate label alignment, as the lower ARI values suggest. This discrepancy may stem from dataset characteristics or the algorithms' inherent properties, underscoring the need for careful algorithm and parameter selection to meet the specific needs of the dataset and task.

Appendices

Future Directions

Future work could explore additional clustering algorithms and parameters, potentially incorporating domain knowledge or alternative metrics for evaluating clustering outcomes. Moreover, investigating the underlying reasons for the discrepancies between Silhouette Scores and ARIs could provide further insights into improving clustering methodologies for complex datasets.

Acknowledgment

We extend our heartfelt gratitude to Dr. Jing Ma for her invaluable instruction and mentorship in the field of Data Mining. Her expertise and insightful guidance have been pivotal in the successful completion of this project. Her dedication to fostering a deep understanding of complex concepts has profoundly enhanced my learning experience.