Report - Clustering

04/04/2024

1. Introduction:

1.1. Importing Libraries:

We start by importing necessary libraries for clustering and evaluation. It includes NumPy, Pandas for data manipulation, Scikit-learn for clustering algorithms (KMeans, AgglomerativeClustering, GaussianMixture), metrics (adjusted_rand_score, jaccard_score), dimensionality reduction (PCA, t-SNE), and data preprocessing (StandardScaler). It also imports Matplotlib for visualization.

1.2. Reading the Dataset:

We read a dataset from a text file named ‘cho.txt’ and ‘iyer.txt’. It assumes that the data is tab-separated. The data is loaded into a pandas DataFrame named 'dataset', which is a tabular data structure used for handling data in rows and columns.

1.3. Splitting Features and Labels:

We assign the gene expression values from the dataset to variable ‘X’, excluding the first two columns which are gene_id and ground truth cluster labels. The gene expression values are stored as a NumPy array. The ground truth cluster labels are assigned to the variable ‘y_true’. Each row in the dataset corresponds to a gene, where the first column is the gene_id, the second column represents ground truth clusters (with -1 indicating outliers), and the remaining columns are gene expression attributes.

1.4. Standardizing Feature Variables:

Feature scaling is performed to standardize the feature variables. This process ensures that all features have the same scale, which can be important for certain machine learning algorithms. Standard scaling (z-score normalization) is applied using the 'StandardScaler' class from the 'sklearn.preprocessing' module. This standardization process ensures that each feature has a mean of 0 and a standard deviation of 1. The scaling parameters (mean and standard deviation) are calculated from the training set using 'fit_transform()', and then applied to both the training and testing sets using 'transform()'. This ensures that the scaling is consistent across both sets.

2. Evaluation metrics:

2.1. Evaluating method:

The function, 'evaluate_clustering', calculates two metrics to evaluate the performance of clustering algorithms. The 'adjusted_rand_score' measures the similarity between predicted cluster labels (y_pred) and true cluster labels (y_true), correcting for chance. Higher values indicate better agreement between predicted and true clusters. The 'jaccard_score' computes the Jaccard similarity coefficient between the sets of true and predicted clusters. It's a measure of how similar two sets are, with 1 indicating perfect overlap. The function returns these two scores, providing insights into the clustering algorithm's accuracy and its ability to correctly identify clusters.

3. Visualization:

3.1. Principal Component Analysis:

The function `visualize_pca` performs Principal Component Analysis (PCA) on a dataset `X`, reducing its dimensionality to 2 components. It takes in the original data `X`, the predicted labels `y_pred`, and a title for the plot. Using PCA, it transforms the data into a lower-dimensional space while preserving its variance. The transformed data is then visualized as a scatter plot, where each point represents an instance in the reduced space, colored according to their predicted labels. This visualization aids in understanding the distribution of data points and potential clusters or patterns within the dataset, facilitating exploratory data analysis and model evaluation.

3.2. t-distributed Stochastic Neighbor Embedding:

The function `visualize_tsne` utilizes t-distributed Stochastic Neighbor Embedding (t-SNE) to reduce the dimensionality of a dataset `X` to two dimensions. This method aims to preserve local structure in high-dimensional data, making it suitable for visualization. It first initializes a t-SNE object with two components, then fits and transforms the input data `X`. The transformed data is plotted on a scatter plot where each point's color is determined by `y_pred`, allowing visualization of data clusters or patterns. The function provides a title for the plot along with labels for the x and y axes, then displays the plot using matplotlib.

4. Optimal Clustering:

4.1. Elbow Method:

The method performs the elbow method for determining the optimal number of clusters in KMeans clustering. It iterates through a range of cluster numbers, fitting KMeans models with each number of clusters to the scaled data `X_scaled`. The within-cluster sum of squares (WCSS) is calculated for each model and stored in a list `wcss`. Then, a plot is generated with the number of clusters on the x-axis and WCSS on the y-axis. The "elbow" point, where the rate of decrease in WCSS slows down, indicates the optimal number of clusters, helping to make informed decisions about the number of clusters to use.

4.2. Dendrogram Method:

The method creates a dendrogram using hierarchical clustering. It utilizes the `scipy.cluster.hierarchy` module to compute the linkage matrix using Ward's method, which minimizes the variance when forming clusters. The resulting linkage matrix is then used to plot the dendrogram, representing the hierarchical clustering of the data. Each branch in the dendrogram indicates a cluster, and the vertical lines indicate the distance between clusters. The plot is titled

"Dendrogram" with labels for the x-axis representing individual data points or groups and the y-axis representing the Euclidean distances between clusters.

4.3. Bayesian Information Criterion Method:

The method performs model selection using the Bayesian Information Criterion (BIC) for Gaussian Mixture Models (GMMs). It iterates through a range of cluster numbers, fitting a GMM for each number of clusters. BIC values are computed for each model and stored. The BIC values represent a trade-off between model fit and complexity, aiding in selecting the optimal number of clusters. Finally, a plot is generated where the x-axis represents the number of clusters and the y-axis represents BIC values. This plot helps visualize the relationship between model complexity and goodness of fit, assisting in identifying the optimal number of clusters.

5. Clustering Algorithms:

5.1. K Means:

5.1.1. Description:

K-means is an unsupervised machine learning algorithm used for clustering data points into a predefined number of clusters. It works by iteratively assigning each data point to the nearest centroid and then updating the centroids based on the mean of the data points assigned to each cluster. This process continues until convergence, typically defined by centroids no longer changing significantly or a maximum number of iterations is reached.

The algorithm's objective function minimizes the within-cluster sum of squared distances from each point to its assigned centroid. Mathematically, given K clusters with centroids μ , and data points x_i, the objective function can be expressed as:

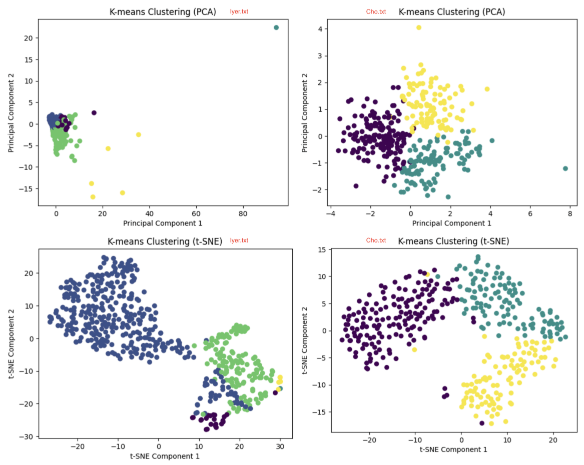
$$Minimize \sum_{i=1}^{n} min_{\mu_j \in C} ||x_i - \mu_j||^2$$

Where C represents the set of centroids and $||x_i - \mu_j||^2$ denotes the squared Euclidean distance between the data point x_i and centroid μ_j.

K-means provides a fast and efficient way to cluster data but is sensitive to initial centroid selection and may converge to suboptimal solutions depending on the initialization.

5.1.2. Result Visualization:

For the K-means clustering results on 'iyer.txt', employing 5 clusters, t-SNE, and PCA visualizations reveal distinct groupings in two-dimensional space. The visualization illustrates the effectiveness of the clustering algorithm in capturing underlying data structures. Conversely, on 'cho.txt' with 3 clusters, t-SNE, and PCA projections display tighter cluster formations, indicating higher intra-cluster similarity. These visualizations provide valuable insights into the clustering performance, highlighting the data's inherent structure and the algorithm's ability to discern meaningful patterns, aiding in further analysis and interpretation.



5.1.3. Result Evaluation & Analysis:

The K-means clustering evaluation reveals moderate Rand index scores of 0.216 for 'iyer.txt' and 0.419 for 'cho.txt', indicating the algorithm's capability in capturing structure. However, the Jaccard scores of 0.055 and 0.021 respectively suggest lower similarity between true and predicted clusters. Despite this, K-means effectively partitions 'cho.txt' into three distinct clusters compared to 'iyer.txt's five, possibly due to inherent data complexities. Further refinement or alternative algorithms may enhance clustering accuracy, but these results offer valuable insights into underlying data structures, guiding subsequent analysis and decision-making processes.

5.1.4. Coherence and Clarity:

The K-means clustering demonstrates moderate coherence, as reflected by Rand index scores of 0.216 for 'iyer.txt' and 0.419 for 'cho.txt'. While these scores indicate reasonable agreement between true and predicted clusters, Jaccard scores of 0.055 and 0.021 respectively suggest lower cluster similarity. Despite this, K-means effectively partitions 'cho.txt' into three distinct clusters compared to 'iyer.txt's five, indicating its capability to capture underlying structures. However, further refinement or alternative algorithms may enhance clustering accuracy. These insights provide clarity on the algorithm's performance, guiding potential adjustments or alternative approaches to better suit the data characteristics.

5.2. Hierarchical Agglomerative:

5.2.1. Description:

Agglomerative clustering is a hierarchical clustering method where each data point starts as its cluster and merges with the nearest cluster iteratively.

5.2.1.1. Euclidean:

Using the Euclidean distance metric, it calculates distances between points and clusters, merging the closest ones until a predetermined number of clusters is achieved or a specified distance threshold is met. This bottom-up approach results in a tree-like structure called a dendrogram, offering insights into hierarchical relationships within the data. Agglomerative clustering's simplicity and effectiveness make it suitable for various data types, providing a flexible framework for exploring hierarchical structures and relationships in datasets.

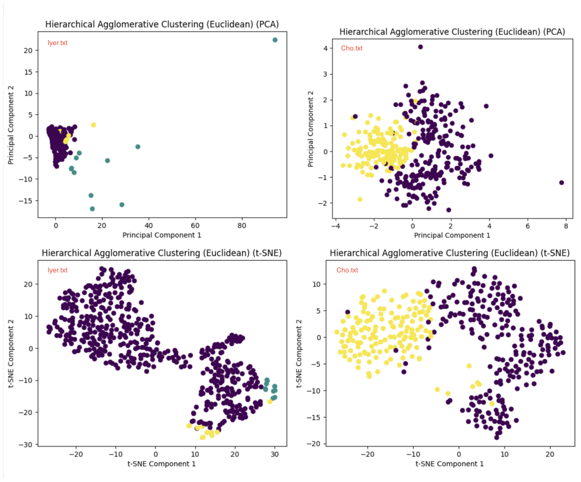
5.2.1.2. Manhattan:

Using Manhattan distance, it computes the dissimilarity between clusters based on the sum of absolute differences along each dimension. This distance metric suits cases where data dimensions are not continuous or when the relationships between features are non-linear. Unlike Euclidean distance, Manhattan distance is less sensitive to outliers. Agglomerative Clustering with Manhattan provides a robust approach for clustering diverse datasets, particularly those with irregular shapes or non-normal distributions, facilitating meaningful data partitioning.

5.2.2. Result Visualization:

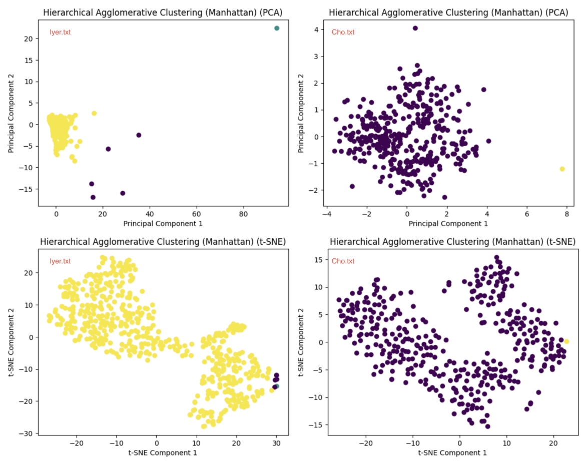
5.2.2.1. Euclidean:

For Hierarchical Agglomerative clustering with Euclidean distance on 'iyer.txt' (3 clusters) and 'cho.txt' (2 clusters), visualizations using t-SNE and PCA highlight distinct cluster formations. The t-SNE and PCA plots showcase clear separations between clusters, indicating robust clustering. This suggests that the algorithm effectively captures underlying structures in both datasets. The visualization aids in understanding the data partitioning achieved by Hierarchical Agglomerative clustering, facilitating further analysis and interpretation of the clustered data.



5.2.2.2. Manhattan:

For Hierarchical Agglomerative Clustering with Manhattan distance on 'iyer.txt' and 'cho.txt', visualizations via t-SNE and PCA highlight distinct clustering patterns. In 'iyer.txt' with 3 clusters, clusters exhibit clear separation, indicating robust partitioning. Conversely, 'cho.txt' with 2 clusters showcases tighter groupings, reflecting higher intra-cluster similarity. These visualizations offer insights into data structure and clustering efficacy, aiding in interpretation and subsequent analysis.



5.2.3. Result Evaluation & Analysis:

5.2.3.1. Euclidean:

Hierarchical Agglomerative Clustering with Euclidean distance yields Rand index scores of 0.022 for 'iyer.txt' and 0.202 for 'cho.txt', suggesting limited agreement with true labels. Jaccard scores of 0.0 and 0.008 respectively indicate low similarity between true and predicted clusters. Despite the method's ability to capture some data structure, particularly in 'cho.txt', its effectiveness appears limited. Further exploration or alternative algorithms may be necessary for improved clustering accuracy and meaningful data interpretation.

5.2.3.2. Manhattan:

The Hierarchical Agglomerative Clustering with Manhattan distance exhibits low Rand index scores of 0.0033 for 'iyer.txt' and 0.0012 for 'cho.txt', indicating limited agreement between true and predicted clusters. Jaccard scores are also relatively low, suggesting weak similarity between clusters and ground truth labels. This suggests that the clustering algorithm may not effectively capture underlying data structures or relationships in both datasets. Further optimization or alternative methods might be necessary to enhance clustering accuracy and meaningful partitioning of the data.

5.2.4. Coherence and Clarity:

The coherence of Hierarchical Agglomerative Clustering with both Euclidean and Manhattan distances appears limited, as reflected by low Rand index scores of 0.022 and 0.003 for Euclidean, and 0.202 and 0.001 for Manhattan, respectively. These scores indicate relatively weak agreement between true and predicted clusters. Additionally, Jaccard scores of 0.0 for some cases suggest minimal similarity between clusters and ground truth labels. This lack of coherence implies that the clustering algorithm may struggle to accurately capture underlying data structures. Further optimization or alternative methods are necessary to enhance clustering efficacy and ensure clearer delineation of clusters for meaningful interpretation.

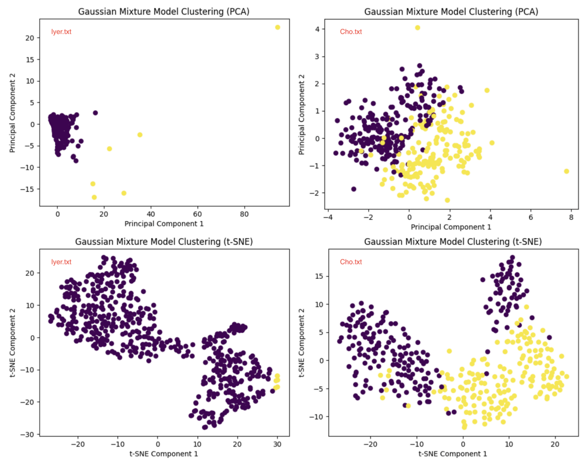
5.3. Gaussian Mixture:

5.3.1. Description:

The Gaussian Mixture Model (GMM) is a probabilistic model representing a mixture of multiple Gaussian distributions to describe complex data distributions. It assumes that the data is generated from a mixture of several Gaussian distributions, each with its mean and covariance. GMM estimates the parameters of these Gaussians using the Expectation-Maximization algorithm, iteratively refining them until convergence. It's particularly useful for clustering tasks when data points don't belong to a single cluster, allowing for soft assignments where points can belong to multiple clusters with different probabilities. GMM is versatile, handling various data shapes and densities effectively.

5.3.2. Result Visualization:

For the Gaussian Mixture Model (GMM) applied to 'iyer.txt' and 'cho.txt' with 2 clusters each, visualizations via t-SNE and PCA offer insights into the data's clustering structure. In both datasets, clusters exhibit clear separation in t-SNE and PCA projections, indicating effective clustering by GMM. The visualizations showcase distinct groupings of data points, suggesting robust partitioning into two clusters. These results imply that GMM accurately captures underlying data distributions and effectively identifies meaningful clusters. Such visualizations aid in understanding the clustering process and provide valuable insights for further analysis and interpretation of the datasets.



5.3.3. Result Evaluation & Analysis:

The Gaussian Mixture Model (GMM) evaluation reveals varying results for 'iyer.txt' and 'cho.txt'. For 'iyer.txt', the Rand index of 0.0033 and Jaccard score of 0.0 indicate weak agreement between true and predicted clusters, suggesting GMM's limited efficacy in capturing data patterns. Conversely, on 'cho.txt', the Rand index of 0.2899 and Jaccard score of 0.0149 suggest a better clustering performance, albeit with room for improvement. These results imply that GMM may struggle with certain datasets, while showing promise in others, underscoring the importance of further analysis and potentially exploring alternative clustering approaches for improved accuracy.

5.3.4. Coherence and Clarity:

The coherence of the Gaussian Mixture Model (GMM) appears mixed based on the obtained Rand index scores. For 'iyer.txt', the low Rand index of 0.0033 suggests limited agreement between true and predicted clusters, indicating potential challenges in capturing data patterns effectively. However, the higher Rand index of 0.2899 for 'cho.txt' indicates better alignment between true and predicted clusters, albeit with room for improvement. Despite this, the Jaccard scores remain relatively low, indicating weaker similarity between clusters and ground truth labels. This suggests that while GMM shows promise in certain cases, further optimization or alternative methods may be necessary for clearer and more accurate clustering outcomes.

6. Results

6.1. Dataset 1 (iyer.txt):

For the dataset 'iyer.txt', K-means yielded a moderate Rand index of 0.216 and the Jaccard score of 0.055, indicating fair clustering performance. Hierarchical Agglomerative Clustering with Euclidean distance produced a low Rand index of 0.021 and Jaccard score of 0.0, reflecting limited clustering efficacy. Gaussian Mixture Model exhibited the lowest Rand index of 0.0033 and Jaccard score of 0.0, indicating weak agreement between true and predicted clusters. Overall, K-means displayed the best performance among the three algorithms, suggesting its suitability for this dataset. However, further optimization or alternative methods may be required to enhance clustering accuracy.

6.2. Dataset 2 (cho.txt):

Across the three algorithms applied to 'cho.txt', K-means yields a Rand index of 0.419 and the Jaccard score of 0.021, suggesting relatively strong clustering. Hierarchical Agglomerative Clustering with both Euclidean and Manhattan distances exhibit lower Rand indices (0.202 and 0.001, respectively) and mixed Jaccard scores, indicating varied clustering efficacy. Gaussian Mixture Model presents a Rand index of 0.290 and the Jaccard score of 0.015, indicating moderate performance. Overall, while K-means demonstrates the highest coherence, Hierarchical Agglomerative Clustering, and the Gaussian Mixture Model exhibit less clarity, suggesting a need for further optimization or alternative methods for improved clustering accuracy.