Project mid-way report

● Graded

Student

Leila Michal

Total Points

9 / 10 pts

Question 1

Assessment **9** / 10 pts

1.1 Introduction **4** / 4 pts

+ 4 pts Your intro is exemplary and could serve as the intro to a conference or journal paper in your field.Correct

✓ **+ 4 pts** Your intro tackles all of the points above, but the prose could use some polishing. *(4pt for mid-way report but 3pt for this quality in final report)*

+ 3 pts Your intro either lacks a good description of the problem, or your motivation is insufficiently fleshed out to convince a disinterested reader to care about your project. *(3pt for mid-way report but 2pt for this quality in final report)*

+ 2 pts You wrote an intro. *(2pt for mid-way report but 1pt for this quality in final report)*

+ 0 pts You did not write an intro.

Nice intro. The goal of your project is clear. It'd be nice if you could mention any additional aspects of the cancer cell data (besides dimensionality) that might make clustering a challenge. The specifics of the clustering metrics you use can likely be moved to the metrics section

1.2 Methods **3** / 4 pts

+ 4 pts Your Methods are exemplary and publishable at a top conference/journal in your field.

+ 4 pts Your Methods do the tasks above, but with some minor shortcomings or imperfections in wording or organization. *(4pt for mid-way report but 3pt for this quality in final report)*

✓ **+ 3 pts** Your Methods do not fully explain what you're doing. The reader cannot figure out your exact plans. *(3pt for mid-way report but 2pt for this quality in final report)*

+ 2 pts You wrote a Methods. *(2pt for mid-way report but 1pt for this quality in final report)*

+ 0 pts You did not write a Methods.

You have a nice run-down of the different dimensionality reduction and clustering techniques. However, in the methods section, you should also emphasize how the different techniques were applied in your specific experiment. You should describe the data you're using, the input for the different techniques, and any parameters you set to initialize the clusters. You also need to include the metrics you want to use when analyzing your results.

1.3 Implementation Details **2** / 2 pts

✓ **+ 2 pts** You wrote down enough implementation details that we (the graders) believe you have thought about how you will implement stuff.

+ 2 pts You wrote down any implementation details at all. *(2pt for mid-way report but 1pt for this quality in final report)*

+ 0 pts You wrote down nothing.

~ helpful to include where you got the dataset from

Question assigned to the following page: 1.1

A Comparative Analysis of Clustering Gene Expression Data for Cancer Subtypes

Leila Michal (Andrew ID: lmichal)

Due Date: April 4th, 2025

1 Introduction

Gene expression profiling has revolutionized our understanding of cancer biology by allowing for the classification of cancers into distinct molecular subtypes. These subtypes are crucial for determining patient-specific treatments, as they may exhibit different responses to therapies and varying prognoses. Traditional approaches to cancer subtyping often rely on expert-curated gene sets or predefined molecular pathways, which, while informative, can miss hidden patterns or subtypes that are not yet known. Conversely, clustering methods offer a data-driven approach to identify novel subtypes by grouping samples based on the similarity of their gene expression profiles.

However, clustering gene expression data presents significant challenges due to the high dimensionality of the data, noise, and technical variation. High-dimensional datasets, often involving thousands of genes per sample, require careful preprocessing to ensure meaningful clusters are identified. Dimensionality reduction techniques like Principal Component Analysis (PCA), t-SNE, and UMAP have been widely used to mitigate these challenges by reducing the number of variables while preserving the key structures in the data [3].

The K-means algorithm is one of the most commonly used clustering techniques [2]. It partitions the data into k clusters by minimizing within-cluster variance. Despite its popularity, K-means has limitations, including

Questions assigned to the following page: 1.1 and 1.2

its sensitivity to the initial placement of centroids and the need to specify k beforehand. Hierarchical clustering, on the other hand, constructs a dendrogram of nested clusters and does not require the number of clusters to be predefined. This method is especially useful for exploring the hierarchical relationships between samples [1]. More recently, Leiden clustering, a graph-based algorithm that maximizes modularity, has shown promise in identifying complex structures that other clustering methods may overlook.

The goal of this project is to evaluate the performance of K-means, hierarchical clustering, and Leiden clustering on cancer gene expression data to identify cancer subtypes. The clustering results will be assessed by using both internal metrics (stability and coherence) and external metrics (Jaccard Index and Adjusted Rand Index), which will allow us to compare the algorithms' ability to identify biologically meaningful subtypes. The approach will leverage two publicly available cancer gene expression datasets: the Gene Expression Cancer RNA-Seq dataset and the Arcene dataset from the UC Irvine Machine Learning Repository. These datasets will be used to compare and contrast the clustering methods, providing insight into which algorithm offers the most effective and reliable subtyping of cancer samples.

Methods

Computational Approach

The computational approach employed in this study consists of several distinct stages: data preprocessing, dimensionality reduction, clustering, and validation. These stages are designed to optimize the clustering process and ensure the robustness of the results. Each stage plays a crucial role in preparing the data, reducing noise, and improving the interpretability of the final clusters.

Data preprocessing is the first and most crucial step in preparing the gene expression data for clustering. Gene expression data is inherently noisy, often suffering from issues like missing values, technical variation, and systematic biases. To address these issues, the data undergoes normalization, log transformation, and standardization. Normalization accounts for differences in sequencing depth across samples, ensuring that the data is comparable. Log transformation is applied to reduce skewness and compress the range of gene expression values, making the data more suitable for clustering algorithms.

Standardization ensures that all genes contribute equally to the analysis by adjusting for differences in variance between genes. Additionally, any missing values are imputed using appropriate techniques, and genes with low variability across samples are filtered out, as they are unlikely to be informative for identifying subtypes.

Dimensionality reduction is an essential step due to the high dimensionality of gene expression data. Gene expression datasets often contain thousands of genes per sample, which can make clustering computationally expensive and challenging. Dimensionality reduction methods are employed to reduce the number of features while preserving the underlying structure of the data. The three different methods for dimensionality reduction are Principal Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP), and t-distributed Stochastic Neighbor Embedding (t-SNE). PCA is a linear method that captures the maximum variance in the data, making it useful for identifying the most significant sources of variation. UMAP is a non-linear technique that is well-suited for preserving both local and global structures in the data, making it a powerful tool for high-dimensional data visualization and clustering. t-SNE is another non-linear method that emphasizes local neighborhood structure and is commonly used for visualizing high-dimensional data in lower dimensions.

After dimensionality reduction, the next step is clustering. Three clustering algorithms are used in this study: K-means, hierarchical clustering, and Leiden clustering. K-means clustering is a centroid-based algorithm that partitions the dataset into k clusters, where each cluster is represented by the mean of the data points assigned to it. The objective function for K-means is to minimize the within-cluster variance, which can be expressed as:

$$J = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

where k is the number of clusters, C_i is the set of points in cluster i, and μ_i is the centroid of cluster i. This method aims to assign samples to clusters in a way that minimizes the sum of squared distances between points and their respective centroids.

Hierarchical clustering, in contrast, does not require the number of clusters to be specified in advance. It builds a tree-like structure, known as a dendrogram, that represents nested groupings of samples. This method starts by treating each sample as its own cluster and progressively merges

Questions assigned to the following page: 1.3 and 1.2

the closest clusters based on a chosen distance metric, such as Euclidean distance or Pearson correlation. The resulting dendrogram can be cut at a particular level to define the desired number of clusters.

Leiden clustering is a more recent approach that is based on graph theory. It aims to maximize modularity, a measure of the quality of a clustering that compares the number of edges within clusters to what would be expected in a random graph. The modularity objective function is given by:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where A_{ij} represents the edge weight between nodes i and j, k_i and k_j are the node degrees, m is the total edge weight, and $\delta(c_i, c_j)$ is 1 if nodes i and j belong to the same cluster and 0 otherwise. Leiden clustering is particularly effective at identifying more complex structures in the data, which makes it a promising method for cancer subtype discovery.

Implementation Details

The first step in the implementation pipeline is data preprocessing. The Gene expression data is normalized using the `scikit-learn` library, and missing values are imputed using the `SimpleImputer` class. Low-variance genes are filtered out to focus on the most informative features for clustering. These preprocessing steps ensure that the data is well-suited for clustering algorithms and that the results are not affected by irrelevant or noisy features.

For dimensionality reduction, PCA, UMAP, and t-SNE are used from the `scikit-learn` and `umap-learn` libraries. The number of principal components for PCA is selected based on the proportion of variance explained by the components. For UMAP and t-SNE, hyperparameters such as perplexity and the number of components are tuned based on empirical validation.

The clustering algorithms are implemented using `scikit-learn` for K-means and hierarchical clustering, and the `leidenalg` library for Leiden clustering. K-means is run multiple times with different initializations to determine the optimal number of clusters using the elbow method. Hierarchical clustering is performed using agglomerative methods, and the number of clusters is selected using silhouette scores. Leiden clustering is performed using the Leiden algorithm, and the modularity objective is maximized to identify the most meaningful clusters.

Datasets and Code Implementation

The following steps are used to load and preprocess the datasets:

1. **Reading the Datasets:**
   - The gene expression data for both datasets is loaded using the `read_snp` function, which reads SNP data from tab-delimited files and processes it into a Pandas DataFrame. The function uses `sep=r'+'` to account for whitespace-separated columns.
   - Gene expression data for specific populations is handled using the `read_snp_pop` function, which allows for loading SNP data for a particular population by passing the population's name.
2. **Handling Genotype Data:**
   - The genotype data for each dataset is read using the `read_geno_pop` function. This function loads the genotype data (typically in `.geno` files) into a NumPy masked array, with missing data denoted by `9` and handled using the `usemask=True` argument.
   - Specific chromosome slices can be read using the `read_geno_pop_chr` function, which opens the `.geno` file for a population and extracts a subset of the data corresponding to a specific chromosome.
3. **Preprocessing:**
   - Missing values in the gene expression data are imputed using the `SimpleImputer` class from `scikit-learn`, ensuring that all samples are complete.
   - Low-variance genes are filtered out using a custom filtering method to focus on the most informative genes for clustering.
4. **Dimensionality Reduction and Clustering:**
   - **PCA** is applied using `PCA` from `scikit-learn` to reduce the dimensionality of the dataset while retaining the most significant variance in the data.

- **UMAP** and **t-SNE** are also employed for dimensionality reduction using the respective libraries: `umap-learn` for UMAP and `scikit-learn` for t-SNE. Hyperparameters like perplexity and number of components are tuned to optimize clustering performance.
- **Clustering Algorithms:**
  - **K-means clustering** is implemented using `KMeans` from `scikit-learn`. The algorithm is run with different initializations to determine the optimal number of clusters using the elbow method.
  - **Hierarchical clustering** is implemented using the `AgglomerativeClustering` class from `scikit-learn`.
  - **Leiden clustering** is performed using the `leidenalg` library, which optimizes modularity to find the best clusters.

No questions assigned to the following page.

References

[1] Eisen, M. B., et al. ”Cluster analysis and display of genome-wide expression patterns.” *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, 1998, pp. 14863–14868.

[2] MacQueen, J. ”Some Methods for Classification and Analysis of Multivariate Observations.” *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1967, pp. 281-297.

[3] van der Maaten, L., & Hinton, G. ”Visualizing Data using t-SNE.” *Journal of Machine Learning Research*, vol. 9, 2008, pp. 2579-2605.