Final Project

Names: Aris Podotas, Rafail Adam, George Leventis

ID's: 7115152400040, 7115152400009, 711572100024

University: National and Kapodistrian University of Athens

Program: Data Science and Information Technologies

Specialization: Bioinformatics - Biomedical Data

Lesson: Machine Learning in Computational Biology

Date: July 2, 2025

Contents

1 Abstract **1**

2 Introduction **1**
2.1 Overview Of The Publication . . . 1
2.2 Flaws Of The Paper . . . 2
2.3 Proposals . . . 2
2.4 Reasoning . . . 2
2.5 Limitations . . . 3

3 Methods **3**
3.1 Programming . . . 3
3.2 Models and Metrics . . . 4
3.3 Generative Modeling . . . 4

4 Results and Discussion **5**
4.1 Cell Generator . . . 5
4.2 Optimal K . . . 6
4.3 Optimal Algorithm . . . 7

5 Disclaimer **9**
5.1 LLM . . . 9
5.2 Purpose . . . 9

1 Abstract

In recent years the hematopoietic progenitor cells have been a focal point of research for the key insights to the immune system and cell differentiation they provide. Much of this research has to do with the details of the differentiation of these progenitors to downstream cell types, since this is a key concept that has many intricacies yet to be discovered. Within the reference publication the authors try to deliver some key insights in this process with results that suggest some of the current understanding of this process may be incorrect. Here we hone in on a key issue found in this publication with aim to definitively decide what algorithms should be used in a key step of the pipeline.

2 Introduction

2.1 Overview Of The Publication

In recent years, many studies have been focusing on unraveling the differentiation trajectories of the hematopoietic lineage. One key aspect of this task is the early differentiation of these cells whose structure sheds light into the origins of each differentiated cell type. The challenges that govern this task include the high variability of the human hematopoietic progenitors whose isolation has been mostly based on the marker CD34. One caveat that comes from this, is the fact that hematopoietic progenitors are so diverse that some subtypes include cells with lowly expressed CD34 or none at all, which compromises any studies that are solely based on this marker. This prompted the authors of this study to use an additional marker , CD164, which they found to be indicative of differentiation according to preliminary data. After obtaining two samples, one containing the CD34+ fraction of cells and another the whole bone marrow fraction of cells from two healthy donors, they proceeded with the computational analysis from the single-cell RNA-seq experiment.

Following the filtering out of dying cells, and gene filtering, the expression values were z-scaled and PCA Pearson, 1901 was fitted on the sample coming from the whole bone-marrow fraction sample keeping only 40 components and was then used to transform the sample containing only the CD34+ cells. A K-nearest-neighbor (KNN) graph was constructed with $k = 4$ for every sample and structure aware filtering followed. During this step, consolidation points are fitted on the graph whose position is determined by a loss function that changes the structure by using the consolidation points to bring cells closer together within a radius and spread them across the first 2 Principal components, convergence is achieved if the consolidation points do not move more than a predefined threshold.

Branch reconstruction is performed by the use of the minimum spanning tree algorithm and branch association takes place based on the Euclidean distance Ballantine and Jerbert, 1952 of each cell from the axis defined by the previous step. Finally, the distances between the different branches are measured with the goal of ordering them, giving rise to the final structure.

From these two differentiation structures that are shown, one can clearly observe the diffence in resolution obtained by sampling the whole bone-marrow fraction as well as the finding that the baseophilic branch of cells comes more closely resembles the megakaryocyte progenitors rather than the lymphoid ones

2.2 Flaws Of The Paper

In the methods section of the paper Pellin et al., 2019, there is mention of one of the initial steps being that of the creation of a K-nearest-neighbor graph (commonly KNN graph). This graph is then mentioned to have been made using a K value $k = 4$. No more mention into the reasoning of the value was made, so two key questions arose. One is why the value of 4 specifically? Considering that odd values are prefers by default. Two is why only the KNN algorithms was used for the generation of the graph with the purpose of partitioning or labeling the cells?

From an otherwise excellently written paper Pellin et al., 2019, this one methodology choice stuck out.

2.3 Proposals

Our proposals start with the use of a generative model to estimate the gene expression domains for each gene individually across all cell types. This sounds strange since the cell data is available, however after reading the publication carefully there are two available datasets that lead to very different projects. The first dataset is of the cells as they were extracted from the patients. The second is of the results of the analysis in the publication, in a matrix where rows represent unique genes, columns represent groups isolated and values are of gene expression z-scaled. Seeing this, since the second dataset is cleaned (no missing data), pre processed and clear we decided to use that data. Then since we do not have individual cells to cluster as the publication did we would need to generate them using the domain of each gene's expression to estimate and sample expressions. The gene domain was the gene expression along groups (so each row of the matrix). More on the generative model in the methods 3 and the results 4.

We then proposed alternative methods that can output labels to mitigate the flaws found in section 2.2. These methods do not need to use a graph, however some methods that expand on the KNN with more steps were used to try and generate outputs that simply try to better the results of the original graph.

The algorithms chosen:

1. KNN
2. Single Link
3. Ward Algorithm
4. Complete Link
5. Spectral Clustering
6. The Chameleon

The algorithms are mostly clustering algorithms for the reproduction of the cell labels into groups (clusters).

The reasons elaborated on the relative section 2.4.

2.4 Reasoning

The choices of the algorithms 2.3 were based on the following few assumptions about the data and the problem at hand.

Assumptions:

1. The clusters are not compact
2. The data is high in dimensionality
3. The number of clusters is unknown and expensive to estimate
4. Clusters form abstract shapes
5. Cluster bounds will not be clear

6. There exist rare sub populations of cells

The reason the KNN is included is for a direct comparison with the original work. The reason we use the Chameleon and the Spectral Clustering is because they have in initialization using a KNN graph, also the spectral clustering is a subspace clustering algorithm to mitigate the dimensionality of the data. The reason the Single link was chosen is because it outputs elongated clusters and has a graph based approach. The same is true for the Ward and Complete link as far as the graph based approach is involved however they do not output elongated clusters, they were added for reference and to validate this idea of our clusters not being compact.

An added note about one of the assumptions is that since we cannot easily estimate the number of clusters this has removed some algorithms from being viable candidates, so questions of why a specific algorithms was *not* used can be traced to that.

2.5 Limitations

The key limitations are of the generative step, since we estimate the domain of the means over the groups the interpolation between the discreet values will affect the project significantly. To add to this since the generators sample from the distribution based on likely samples, the under represented groups in the data will be hard to generate.

3 Methods

3.1 Programming

The project was implemented in Python "3.13.2 Documentation", n.d. using the following libraries:

1. numpy "NumPy", n.d.
2. pandas "pandas documentation — pandas 2.3.0 documentation", n.d.
3. the chameleon Lin, 2025
4. sklearn "scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation", n.d.
5. scipy "SciPy - Installation", n.d.
6. matplotlib "Matplotlib documentation — Matplotlib 3.10.1 documentation", n.d.

The implementations is in multiple files, each for a different part of the results (such as the optimal k search and the optimal algorithms search). Each file contains the same command line input format and it is recommended to read the help menu.

Each scrip ran for the generation of the results used 100 cells and 100 runs (-cells 100 -runs 100), which is the default for the command line inputs.

Keep in mind that the script was made to run on a directory input that contains files and will use a listing of the directory (will *not* walk down the directory) to find all files (won't look at extension) to use all DSV (deliminator separated value) files for appending to one gene × groups matrix. Also keep in mind that DSV files with multiple sheets will append all sheets to the same matrix. The only file that has the actual gene expression values is "CD34

CD164 GroupLevelExpression" so it was isolated in a separate folder.

Finding the best K in KNN:

```
python knn.py -out ../knnRun1/ -inp
../data/CDgroup/expression/
CD34_CD164_GroupLevelExpression
.xlsx
```

The output path (after `-out`) can be any desired output directory. The input file (after `-inp`) can have any preceding path to the file.

Finding the best algorithm:

```
python main.py -out ../testOutput/
-inp ../data/CDgroup/expression/
CD34_CD164_GroupLevelExpression
.xlsx
```

Where the output (after -out) can be wherever you want the output files and the input file (after -inp) could have any preceding path to the file mentioned

Keep in mind that the file paths may be renamed or moved in the github repository.

Disclaimer: The runs take around 45 seconds each, keep in mind that the scrip runs for extended periods of time with the default parameters. Program is overly verbose due to no options being set in one of the dependencies.

Source files can be found in the relative repository.

3.2 Models and Metrics

As was mentioned in the proposals the Models were:

1. KNN
2. Single Link
3. Ward Algorithm
4. Complete Link
5. Spectral Clustering
6. The Chameleon

The corresponding metrics were:

1. Calinski-Harabasz score, also known as the Variance Ratio Criterion Caliński and Harabasz, 1974
2. Silhouette score Rousseeuw, 1987
3. Davies-Bouldin score Davies and Bouldin, 1979

These metrics were used to get a view of the clusters output by a method considering we cannot visualize this high dimensional set even with feature selection. The Calinski-Harabasz score is a measure that will compare the within cluster dispersion to the overall cluster dispersion between clusters, in this way it gives us a view of the clusterings variance with higher values being an indicator of a better fit. The silhouette score will compare how well points from cluster A are described by cluster A as opposed to all other clusters, in this way giving us a measure of how well a fit our clustering is to the data with values ranging from -1 to 1 with higher values being indicative of a better fit. The Davies-Bouldin score is a measure of cluster compactness, in this way it tells us the shape of our clusters with lower values indicating more compactness. Actually our clusters are expected to be arbitrary in shape so the Davies Bouldin score does not necessarily need to be minimized.

3.3 Generative Modeling

Five different models were developed for the purpose of generating the new cells from the gene domain distributions. Out of these five only two can be seen as true

generative models and the other three were for reference or as a baseline to improve in key directions.

1. Random sampling
2. Discreet selection of a pre existing expression
3. Gaussian modeling
4. Gaussian mixture modeling
5. Gaussian Kernel Density Estimates

The final results shown here were made using the Gaussian mixture models. Reasons include faster computation time and non discreetized step along any one axis (since the KDE takes small steps to compensate for non continuous functions. It's simply a model fit for another application).

4 Results and Discussion

4.1 Cell Generator

An example of what the generators should estimate:

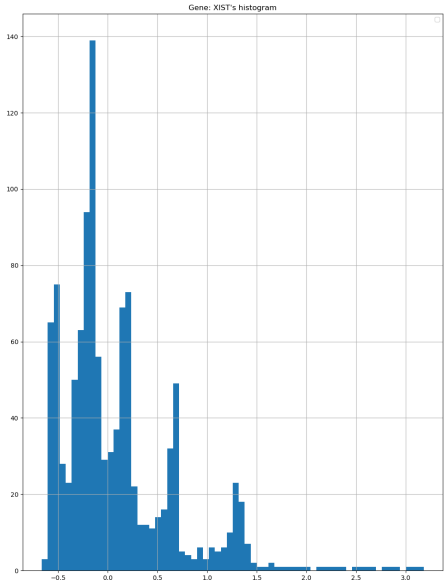

Figure 1: Isolated gene domain (histogram) for a random gene (XIST). The X axis contains values of expression for the gene in question and Y axis values are the frequency of that range of expression in the data.

Notice that in figure 1 the distribution of the gene expression domain is similar to a Gaussian mixture distribution. This is the general form, some genes may look like a standard Gaussian some like other function like $\frac{1}{x}$, some like the figure 1 and some other may be discreet in a way that makes it hard to estimate in general.

An example of an output:

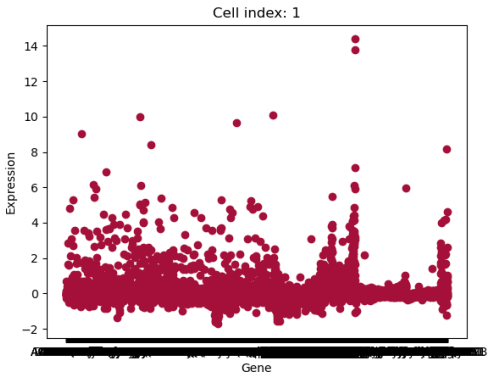

Figure 2: Visualization of a generated cell. The x axis has the discreet genes, the y the generated expression value. The X axis is of the discreet genes and the Y is of the generated expression value for that gene.

No rigorous way to determine the optimal sampler is done since no metrics exist for this task and thus it was chosen that a Gaussian mixture model should estimate all the distributions used in the analysis as was mentioned in the methods 3.

More figures like figure 1 and 2 can be found in the repository.

4.2 Optimal K

It was mentioned in the introduction 2 that the KNN graph used in the citation used a K of 4 with no further explanation. We wanted to test the idea that $k = 4$ is the optimal one by running a KNN graph with K values ranging from 3 to 7.

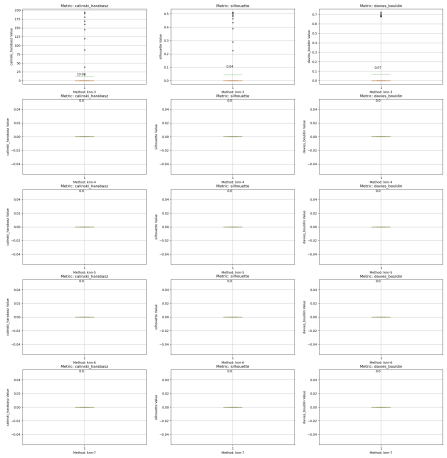

Figure 3: Metrics per KNN run with increasing k. Green dotted lines represent the mean, brown lines represent the median, data labels are of the mean value

It is advised to view images in the original file for greater detail.

One note about the figure 3 is that the values of K above 3 are all of a single 0 value. This is because once a graph unifies the whole data space (cells) into one cluster the metrics all default to 0. This is actually not as bad as it would seem since for instance the silhouette score ranges from -1 to 1 and the Davies-Bouldin score score is best once lower in value.

It is pretty conclusive that from the figure 3 that a value of 3 is optimal for the KNN graph. This would imply that the clusters are fairly close since the value of K is so low and any higher unifies the whole dataset.

In the rest of the assignment (since the k search was the first thing done) the assumed K for any KNN run will be of the optimal K found in 4.2.

4.3 Optimal Algorithm

Upon running the algorithms:

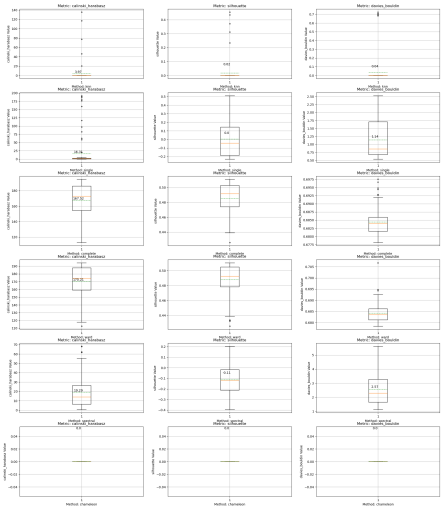

Figure 4: Metrics per algorithm ran. Green dotted lines represent the mean, brown lines represent the median, data labels are of the mean value

It is advised to view images in the original file for greater detail. Keep in mind that the axis is not common before making conclusions.

In figure 4, we see a few interesting and easy to notice first results. The chameleon has clustered all points to a single cluster and can be seen as generally performing worse than the KNN of $K = 3$ and the same as all other K values. The KNN has the most compact output of all algorithms with very low variance distributions (not to be mixed up with variance clustering) and many outliers. The clusters of the KNN are high in variance and the points of one cluster do not fit neatly within one cluster as is evident by the silhouette score.

In figure 4, the single link has output clusters of not very compact shape as is expected of the algorithm with many outliers evident by the difference between the mean and the median. The points of one cluster do not fit neatly within each cluster as the silhouette score would indicate however with a very high variance. The variance of any one cluster (the variance ratio) is better than the KNN but the worst of the remaining algorithms with a very narrow distribution around the mean value and many outliers.

In figure 4, the Complete link has done better than all mentioned algorithms thus far in nearly all metrics (depends on how you want to score the Davies-Bouldin score). Metrics are high in variance yet centered around the best mean and median values with considerable differences in the mean and median. Despite the shape of the variance of the metric distribution it would seem that the complete link performs better than the previous two algorithms on it's worst predictor.

In figure 4, the Ward algorithms narrowly beats out the complete link with very similar distribution in shape. The edges of the distribution reach the same values as the complete link but the mean and median lie just slightly better for all metrics (except the Davies-Bouldin which is just for cluster shape reference). The Ward algorithm is the best performer on this dataset yet.

In figure 4, the spectral clustering is not too dissimilar to the complete link in the metrics, just slightly better in the variance ration and slightly worse in the silhouette score. The clusters have the most non-compact shape of all algorithms. Let us make a note that in the test the spectral clustering outputs the same as the Ward algorithms but obviously the sampling has had a major effect on the algorithm here.and this is a results that if we had not done as many tests as we have we would not have found. This would suggest that the spectral clustering performs the same as the Ward algorithm in most cell types but there is a subset that differentiates the two algorithms in favor of the Ward algorithm. We overwrote previous results however there is an image the shows this in figure 5, the only issue is that the plotting function at the time was not the same and the axis is such that distributions for all metrics are not visible.

In figure 4, compared to the single link the spectral clustering does have more dispersed distributions of metrics.

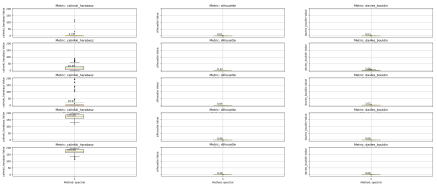

Figure 5: Metrics per algorithm ran while testing to show the difference of the spectral clustering to the Ward algorithm in other runs. Green dotted lines represent the mean, brown lines represent the median, data labels are of the mean value. Keep in mind that the chameleon had not yet been implemented as this is a old version of the script. This image is for illustrating a point made in the previous paragraph

It is advised to view images in the original file for greater detail.

In figure 5 we can see that the distribution for at least the variance ratio criterion and the mean values of the rest of the metrics are comparable between the Ward algorithm and the Spectral clustering. This figure is from a version of the script that had a common axis for all plots and thus most metrics are not visible.

Finally, after all consideration it would seem that the Ward algorithms has performed the best of all our proposals based on the metrics.

5 Disclaimer

Large language models (LLM's) were used during the assignment.

5.1 LLM

Open AI: Chat GPT 4.0 "ChatGPT", n.d.

5.2 Purpose

1. To ask if concepts already exist in some dependency.
2. For acquisition of the relative documentation.
3. To validate the idea about the generative model application
4. To help with the "makeBoxPlots" function
5. To help convert the graph to labels in the "knn" method of the "ClusterPipeline" class

References

3.13.2 documentation. (n.d.). Retrieved March 28, 2025, from https://docs.python.org/3/

Ballantine, J. P., & Jerbert, A. R. (1952). Distance from a line, or plane, to a poin [Publisher: [Taylor & Francis, Ltd., Mathematical Association of America]]. *The American Mathematical Monthly*, *59*(4), 242–243. https://doi.org/10.2307/2306514

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis [Publisher: Taylor & Francis]. *Communications in Statistics*, *3*(1), 1–27. https://doi.org/10.1080/03610927408827101

ChatGPT. (n.d.). Retrieved March 29, 2025, from https://chatgpt.com

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224–227. https://doi.org/10.1109/TPAMI.1979.4766909

Lin, H. (2025, March 21). *Moonpuck/chameleon_cluster* [original-date: 2018-11-03T02:59:12Z]. Retrieved June 23, 2025, from https://github.com/Moonpuck/chameleon_cluster

Matplotlib documentation — matplotlib 3.10.1 documentation. (n.d.). Retrieved March 29, 2025, from https://matplotlib.org/stable/

NumPy. (n.d.). Retrieved April 29, 2025, from https://numpy.org/

Pandas documentation — pandas 2.3.0 documentation. (n.d.). Retrieved July 2, 2025, from https://pandas.pydata.org/docs/index.html

Pearson, K. (1901). *On lines and planes of closest fit to systems of points in space* [Google-Books-ID: uGt_YgEACAAJ]. University College.

Pellin, D., Loperfido, M., Baricordi, C., Wolock, S. L., Montepeloso, A., Weinberg, O. K., Biffi, A., Klein, A. M., & Biasco, L. (2019). A comprehensive single cell transcriptional landscape of human hematopoietic progenitors [Publisher: Nature Publishing Group]. *Nature Communications*, *10*(1), 2395. https://doi.org/10.1038/s41467-019-10291-0

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Scikit-learn: Machine learning in python — scikit-learn 1.6.1 documentation. (n.d.). Retrieved March 28, 2025, from https://scikit-learn.org/stable/

SciPy - installation. (n.d.). Retrieved April 28, 2025, from https://scipy.org/install/