An Exploratory Study on a Clustering Algorithm for Single-cell RNA-seq Data

Wenjing Ma*
wenjing.ma@emory.edu
Computer Science Department, Emory University
Atlanta, Georgia

ABSTRACT

In this report, I explored applying KMeans algorithms on single-cell RNA-sequencing (scRNA-seq) data to check how this unsupervised clustering algorithm performs on the high-dimensional data. I implemented the KMeans based on Expectation Maximization, and compared to Scikit Learn package, it is slower due to the usage of Pandas dataframe and for-loop. The dataset is from the Frontal Cortex region in the male adult brain which consists large proportion of neuronal cells along with small proportion of glial cells. The dataset is very sparse and imbalanced. I applied the self-implemented KMeans to multiple scenarios such as the original high-dimension, PCA projected and feature selected data.

KEYWORDS

scRNA-seq, KMeans, cell type annotation, UMAP

ACM Reference Format:
Wenjing Ma. 2020. An Exploratory Study on a Clustering Algorithm for Single-cell RNA-seq Data. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

With the rapid development of the advanced sequencing technologies, single-cell RNA-sequencing (scRNA-seq) data, with a high-dimension data structure, carries more information compared to bulk RNA-seq data which is considered as an average sum of gene expressions from different cell types. For scRNA-seq data analysis, pipelines mostly include dimension reduction algorithms to visualize the single-cell data in a 2-dimension (2D) view, clustering algorithms to indicate cell-cell distance and other machine learning algorithms to assist in annotating clusters to cell types. However, due to the high-dimensional feature of the data, results of different clustering algorithms may vary according to the choice of certain features and distance metrics and the clustering result may largely affect the final cell type annotation.

In this exploratory study, I applied a self-implemented KMeans algorithm on a well-annotated datasets from the Frontal Cortex region in male adult mice [2]. In the original paper, Saunders et al. adopted a two-stage Independent Components Analysis (ICA) method to extract signal and applied a shared nearest neighbor (SNN) graph with the smart local moving algorithm (SLM) to detect communities. Saunders et al. claimed that they could remove technical variations by applying the two-stage ICA. SNN is based on K Nearest Neighbors (KNN), however, it utilizes more information from the surrounding cells to help with the clustering [3]. Saunders et al. annotated the clusters according to cell-type specific marker genes and divided them into 9 cell types including: Inter-neurons, Neurons, Astrocytes, Endothelials, Oligodendrocytes, Oligodendrocytes Precursor Cells, Mural cells, Fibroblast cells and Microglia/Macrophage.

Because KMeans is an unsupervised classification algorithm and SNN is, in contrast, a supervised learning model, I expect the result to be worse with KMeans application than SNN. In this report, I took the cell type annotation from the original paper as the golden standard and tested how KMeans works on the dataset.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

2 RESULTS

This section covers the datasets description, data pre-processing, KMeans methodology, code structure analysis and the results compared to the original paper.

2.1 Datasets and Pre-processing

According to the raw data downloaded from GEO (GSE: 116470), there are in total 19,4027 cells with 29,463 genes. However, due to the sparsity of scRNA-seq datasets, most entries in the sparse matrix are 0, which makes it difficult to indicate whether the gene from that cell does not express or is not correctly measured. To remove those misleading signals, the two-stage ICA was performed in Saunders et al. After the removal, there are 71,639 cells with 15,976 genes remained.

For this high-dimension data, it would be slow to perform KMeans algorithm on the complete dataset. Thus, I randomly selected 4,000 cells from the data maintaining the original cell type proportions. Although this might lose some level of information, when applying Uniform Manifold Approximation and Projection (UMAP) to visualize the 2-D datasets, we can still find that Inter-neurons(red) and Neurons(green) are clustered together while part of the Oligodendrocytes Precursor Cells(purple) and Oligodendrocytes(blue) are aggregated in Figure 1. The result indicates that the sampled cells still maintain biological meanings from the original data.

2.2 Methodology

Before implementing the algorithm, I ran the KMeans algorithm provided by Scikit-Learn to ascertain my algorithm works properly. During the implementation, I applied the KMeans algorithm which

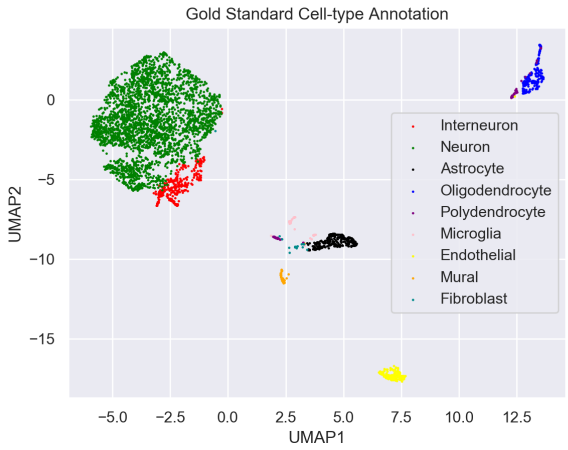

Figure 1: UMAP on sampled 4,000 cells as golden standard.

can be separated in the following steps. First, the most important parameter in KMeans is the number of cluster, which is already known in our case as 9 cell types. Then, 9 cells are randomly selected as the initial centroids. I iterated the dataset and calculated the distance between each data entry with the pre-selected centroids. For the nearest centroid, I assigned the cluster label to that data entry. After label assignments, I re-computed the centroids according to the average of data with the same labels. Then by repeating calculating distance and label assignment, the algorithm converges at a certain point when the centroids are stable.

I mimicked the data structure used in the Scikit-Learn by creating a *My_Kmeans* class and initialize with the number of cluster, maximum iterations and the converge tolerance criteria. Then I created a function named *distance* to calculate the distance metrics between the centroids and other data points. I offered Manhattan distance and Euclidean distance as options. Finally, I implemented the *fit* function which takes the input data and conducts the algorithm.

I compared the clustering results between the Scikit Learn KMeans which used the Euclidean distance and my implemented KMeans with the same distance metric to test whether my algorithm work well. Although my algorithm takes much longer time (793.84s) than Scikit Learn packages (87.64s) to converge, the clustering results are quite similar as compared in Figure 2.

In the result section, I summarized the KMeans results on the original high-dimensional data and with different distance metrics.

2.3 KMeans on sampled cells

In addition Euclidean distance, I also provided the Manhattan distance as a metric, which is more appropriate for high-dimensional data. For these two distance metrics, the algorithm both takes around 30 iterations to converge. However, when comparing the trend of differences per iteration, KMeans with Manhattan distance seems to be more fluctuated as indicated in Figure 3. This might be caused by the l1 norm form, which is a non-convex problem.

I compared the clustering result from KMeans using Manhattan distance and Euclidean distance to the original cell-type annotation

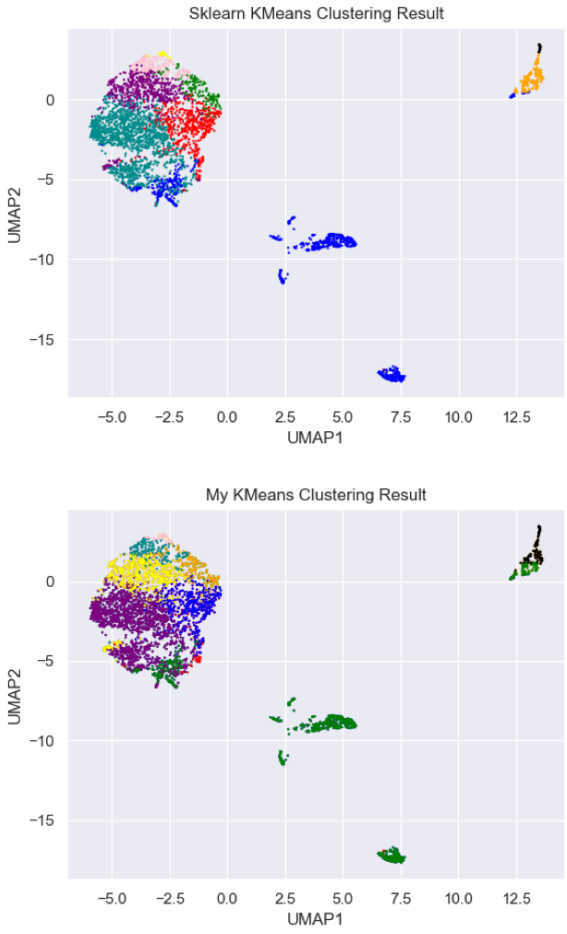

Figure 2: Clustering result between Scikit Learn KMeans and self-implemented KMeans.

and found that neither of them could capture the patterns in the original dataset. The reason may be due to too many genes with large sparsity. Thus, I try to select certain significant genes from the 15,976 genes and check the result.

2.4 KMeans on feature selection

I selected genes which express in over %50 percent of cells and after filtering, 727 genes are remained. Then I applied KMeans with different distance metrics to the filtered dataset and it partitions the largest neuronal group into different clusters as shown in Figure 4.

2.5 KMeans on PCA projection

I also tried to project the dataset onto a 15-dimension space by applying Principal Components Analysis (PCA). PCA could extract the signal with reducing the dimension of the dataset. However, as shown in Figure 5, it also partitions the neuronal group into sub-groups but the dots seem to be even more blurry compared to the golden standard.

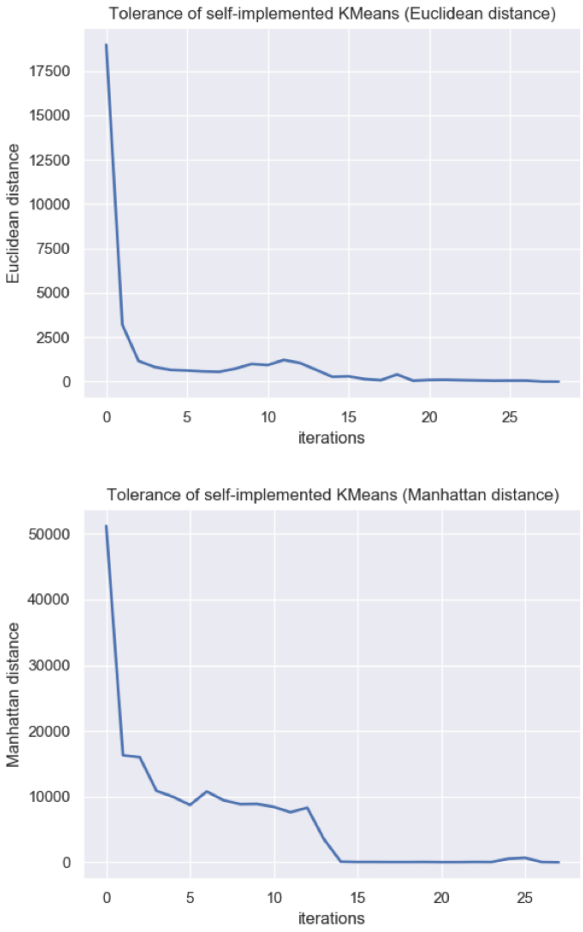

Figure 3: KMeans tolerance pattern between Manhattan distance and Euclidean distance.

3 DISCUSSION

After comparing the KMeans result with the golden standard, we may simply infer that KMeans does not fit scRNA-seq problem since it only captures the information for the distance between points regardless of utilizing information from neighbors. Also, because of the sparse data, the distance calculated between points may not reflect the underlying truth in reality. In addition, the high-dimensional data may need a better representation or projection.

As for the code, my implementation takes around 10 times compared to the Scikit Learn package. The reason might be caused by the for-loop when calculating the nearest distance to assign labels. It may be accelerated by changing the calculation into list comprehensions. Also, according to the source code for the Scikit Learn package, by setting the algorithm as 'auto', it used triangle inequality to accelerate KMeans [1]. However, in my implementation, which is a more Expectation-Maximization style. Another issue that may affect the execution time is the usage of Pandas dataframe which is slower than Numpy Array operations.

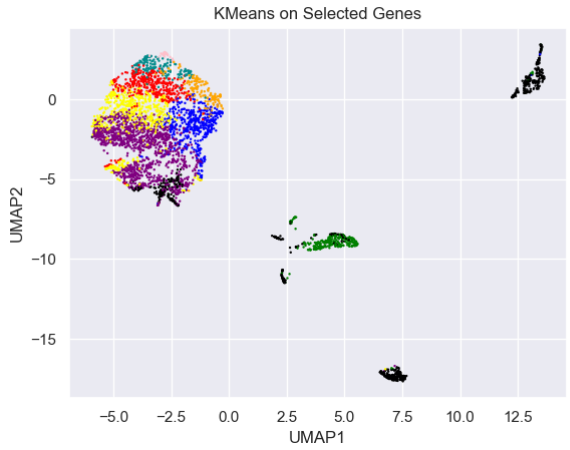

Figure 4: KMeans on 727 genes with different distance metrics.

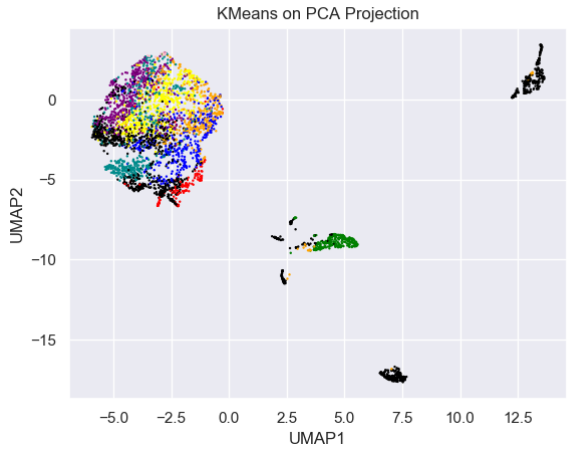

Figure 5: KMeans on PCA Projection (top 15 PCs).

REFERENCES

[1] Charles Elkan. 2003. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)*. 147–153.

[2] Arpiar Saunders, Evan Z Macosko, Alec Wysoker, Melissa Goldman, Fenna M Krienen, Heather de Rivera, Elizabeth Bien, Matthew Baum, Laura Bortolin, Shuyu Wang, et al. 2018. Molecular diversity and specializations among the cells of the adult mouse brain. *Cell* 174, 4 (2018), 1015–1030.

[3] Xiaoshu Zhu, Jie Zhang, Yunpei Xu, Jianxin Wang, Xiaoqing Peng, and Hong-Dong Li. 2020. Single-Cell Clustering Based on Shared Nearest Neighbor and Graph Partitioning. *Interdisciplinary Sciences: Computational Life Sciences* (2020), 1–14.