Identifying Cell Types with Single Cell Sequencing Data

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial pulfillment of the requirements
for the degree of Master of Science
in the Division of Biomedical Engineering
University of Saskatchewan
Saskatoon

By

Wenjing Zhang

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Disclaimer

Reference in this thesis to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this thesis in whole or part should be addressed to:

Head of the Division of Biomedical Engineering
University of Saskatchewan
57 Campus Drive
Saskatoon, Saskatchewan S7N 5A9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

Single-cell RNA sequencing (scRNA-seq) techniques, which examine the genetic information of individual cells, provide an unparalleled resolution to discern deeply into cellular heterogeneity. On the contrary, traditional RNA sequencing technologies (bulk RNA sequencing technologies), measure the average RNA expression level of a large number of input cells, which are insufficient for studying heterogeneous systems. Hence, scRNA-seq technologies make it possible to tackle many inaccessible problems, such as rare cell types identification, cancer evolution and cell lineage relationship inference.

Cell population identification is the fundamental of the analysis of scRNA-seq data. Generally, the workflow of scRNA-seq analysis includes data processing, dropout imputation, feature selection, dimensionality reduction, similarity matrix construction and unsupervised clustering. Many single-cell clustering algorithms rely on similarity matrices of cells, but many existing studies have not received the expectant results. There are some unique challenges in analyzing scRNA-seq data sets, including a significant level of biological and technical noise, so similarity matrix construction still deserves further study.

In my study, I present a new method, named Learning Sparse Similarity Matrices (LSSM), to construct cell-cell similarity matrices, and then several clustering methods are used to identify cell populations respectively with scRNA-seq data. Firstly, based on sparse subspace theory, the relationship between a cell and the other cells in the same cell type is expressed by a linear combination. Secondly, I construct a convex optimization objective function to find the similarity matrix, which is consist of the corresponding coefficients of the linear combinations mentioned above. Thirdly, I design an algorithm with column-wise learning and greedy algorithm to solve the objective function. As a result, the large optimization problem on the similarity matrix can be decomposed into a series of smaller optimization problems on the single column of the similarity matrix respectively, and the sparsity of the whole matrix can be ensured by the sparsity of each column. Fourthly, in order to pick an optimal clustering method for identifying cell populations based on the similarity matrix developed by LSSM, I use several clustering methods separately based on the similarity matrix calculated by LSSM from eight scRNA-seq data sets. The clustering results show that my method performs the best when combined with spectral clustering (Laplacian eigenmaps + k-means clustering). In addition, compared with five state-of-the-art methods, my method outperforms most competing methods on eight data sets. Finally, I combine LSSM with t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize the data points of scRNA-seq data in the two-dimensional space. The results show that for most data points, in the same cell types they are close, while from different cell clusters, they are separated.

Keywords Clustering, Cell type identification, Sparse similarity learning, Single cell

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Fang-Xiang Wu, for his help and guidance in my study. Since my undergraduate background is biological engineering, lack of knowledge in mathematics and computer science has brought many difficulties to my research. Professor Wu's guidance makes me more confident and easier to overcome them.

I would like to thank my advisory committee members: Professor Francis Bui and Professor Lingling Jin. They gave me very useful comments and suggestions. I also want to thank Natural Science and Engineering Research Council of Canada (NSERC) for their support.

Besides, I am grateful to study and work with my cute and funny colleagues: Ping Luo, Hongyu Guo, Lingkai Tang, Fei Wang, Pijing Wei, Yulian Ding, Rayyan Khan, Sakib Mostafa, Ali Jamali Beyrami, Yuting Tan and Ming Fang. Thanks for their help in my life and study.

Finally, I would like to say thanks to my family and my boyfriend, Zexi Yu for their endless love and unreserved support.

Contents

List of Tables

List of Figures

List of Abbreviations

ADMM	alternating direction method of multipliers
ARI	adjusted rand index
cDNA	complementary DNA
DBSCAN	density-based spatial clustering of applications with noise
ESS	Error Sum of Squares
FACS	fuorescence-activated cell sorting
FN	false negative
FP	false positive
GMM	Gaussian mixture model
HCG	human chorionic gonadotropin
HVG	highly variable genes
H7 hESCs	H7 human embryonic stem cells
Isomap	Isometric Mapping
KL divergence	Kullback–Leibler divergence
KNN	k-nearest-neighbor
LLE	Locally Linear Embedding
LP	linear programs
LSH	locally sensitive hashing
MAD	median absolute deviation
NGS	next-generation sequencing
NMI	normalized mutual information
OPTICS	ordering points to identify the clustering structure
PBMCs	peripheral blood mononuclear cells
PCA	principal component analysis
PCC	Pearson correlation coefficient
PCR	polymerase chain reaction
PCs	principal components
PSCs	pluripotent stem cells
QP	quadratic programs
RNA-seq	RNA sequencing
RI	rand index
scRNA-seq	single-cell RNA sequencing
SDP	semidefinite programs

SNE	stochastic neighbor embedding
TN	true negative
TP	true positive
t-SNE	t-Distributed Stochastic Neighbor Embedding

1 Introduction

With the tremendous increase in single-cell RNA sequencing (scRNA-seq) data, algorithmic analysis techniques are significant for studying cell heterogeneity. Single-cell sequencing, which is a sequencing technology to obtain genetic information of individual cells, has been applied to broad biological fields over the past few decades, such as microbiology, neurobiology, embryonic development, tissue mosaicism, immunology, and cancer research [118]. In these researches, cell type identification is essential and necessary for clarifying the complexity of the organization, including the number of cell types and the transcriptome characteristics of each cell population [1]. Generally, unsupervised clustering methods are used for grouping cells into clusters. The cluster labels can then be assigned to their corresponding biological cell types by several ways, such as the differential expression of marker genes (i.e. [108]), development time of cells (i.e. [29]), directed cell differentiation in vitro (i.e. [65]) and so on. However, there still are many unique challenges in scRNA-seq data analysis because of high dropout rates, low and uneven read coverage, high variability and complex batch effects of scRNA-seq data [89, 31, 86]. Therefore, in order to tackle these challenges in scRNA-seq data analysis, appropriate algorithmic analysis techniques should be taken.

1.1 Previous research

1.1.1 Literature review

In recent years, many methods for identifying cell populations have been proposed. Generally, the workflow of scRNA-seq analysis includes data pre-processing, normalization, dropout imputation, dimension reduction, similarity matrix construction and clustering. Due to the high dimensionality of scRNA-seq data, dimentionality reduction is usually performed, and commonly used methods include principal component analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), Laplacian eigenmaps and so on. Furthermore, data points are grouped by unsupervised clustering methods, such as k-means clustering, hierarchical clustering and Gaussian mixture model (GMM) clustering. I summary several popular single-cell clustering analysis tools in Table 1.1.

Majority of the exciting single-cell clustering methods rely on building similarity matrices between cells.

Table 1.1: Summary of nine single-cell clustering analysis tools

Name	Language	Method type	Download
ascend[99]	R	PCA + hierarchical clustering	https://github.com/powellgenomicslab/ascend
TSCAN [50]	R	PCA + model-based clustering	https://github.com/zji90/TSCAN
Seurat [98]	R	PCA + graph-based clustering	https://github.com/satijalab/seurat
monocle [113]	R	t-SNE + density-based clustering	http://monocle-bio.sourceforge.net/
scanpy [120]	Python	PCA + graph-based clustering	https://github.com/theislab/scanpy
SC3 [60]	R	PCA + k-means clustering+hierarchical clustering	https://github.com/hemberg-lab/SC3
SINCERA [44]	R	hierarchical clustering	https://github.com/xu-lab/SINCERA
RaceID [43]	R	Pearson correlation dissimilarities + K-medoids clustering	https://github.com/dgrun/RaceID
CIDR [68]	R	zero-imputed similarities + PCA + hierarchical clustering	https://github.com/VCCRI/CIDR

For example, Jiang et al. construct cell-cell similarity matrices based on the 'differentiability correlation', which is calculated based on the relationship of every two cells deriving from gene differential patterns. The algorithm can also automatically determine the optimal cluster number for hierarchical clustering [51]. Grün et al. build cell-cell similarity matrices based on Pearson correlation coefficients, and further use k-means clustering to reveal rare intestinal cell types [43]. Jeong et al. introduce a single-cell clustering framework, consisting of discerning the set of potential features, estimating the similarity between cells through multiple feature sampling, and using a network-based clustering method to cluster cells [48]. Xu and Su calculate Euclidean distances between cells based on gene expression profile, and then utilize k-nearest-neighbor (KNN) to construct cell-cell similarity matrices for a quasi-clique-based clustering algorithm [123].

1.1.2 Previous research at the University of Saskatchewan

Li et al. propose a single-cell clustering method, Improved Spectral Clustering (ISC), through constructing cell-cell incidence matrices by integrating similarity matrices and dissimilarity matrices between cells. Then classic spectral clustering method is used to cluster cells, which mainly consists of three steps, as illustrated in Figure 1.1 [67].

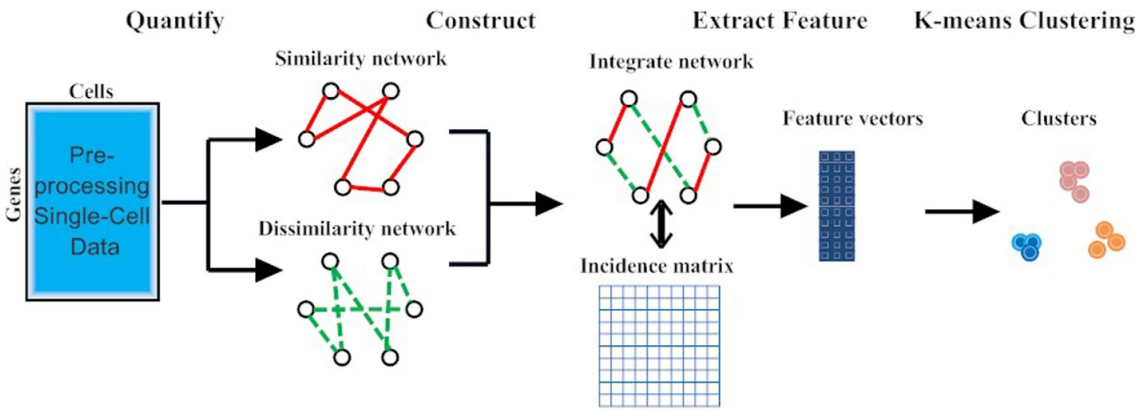

Figure 1.1: The ISC workflow for clustering scRNA-seq data

The three steps can be briefly summarized as follows.

1. Calculating similarity matrices and dissimilarity matrices between cells.

The authors construct a cell-cell similarity matrix, denoted as matrix A, and a cell-cell dissimilarity matrix, denoted as matrix B, base on Pearson correlation coefficient (PCC) [6] to measure the strength and direction of the linear correlation between two variables.

2. Integrating cell-cell similarity with dissimilarity matrices

The authors compute the adjacency matrix W by integrating matrix A with B as: $W = (1\text{-}\alpha)\text{A}+\alpha B$, where α is a constant parameter.

3. Identifying cells by spectral clustering

The authors calculate the eigenstructure of the Laplacian matrix (L), which can be calculated based on adjacency matrix W. The eigenvectors of L are sorted according to the corresponding eigenvalues. Finally, they perform k-means clustering technique on the first k eigenvectores of L, where k is the number of cell types.

Although ISC improves the accuracy and robustness of clustering qualities with single-cell data, there are still potential improvements. Firstly, the authors do not explain clearly how to get the appropriate value of α to calculate the adjacency matrix W. Secondly, they do not give more details about how to determine the neighbors of similar cells or dissimilar cells to construct cell-cell similarity or dissimilarity matrices. Thirdly, the sample sizes of their tested scRNA-seq data sets are too low. The number of cells, used in the paper, is from 251 to 704, while a typical scRNA-seq data set contains thousands of cells. In my study, to address the problems mentioned above, I clearly point out how to adjust the parameters used in my algorithms. Besides, I test my method with scRNA-seq data sets, which contain much more data points than the data sets in ISC study.

1.2 Overview of the study

In this study, I propose a novel method, called Learning Sparse Similarity Matrices (LSSM), to construct sparse cell-cell similarity matrices based on scRNA-seq data. I further apply several clustering methods to identify cell populations based on the similarity matrices. The main steps are briefly summarized as follows.

Firstly, I preprocess the data, including removing zeros, logarithmic transformation, feature selection and dimensionality reduction. Removing zeros and logarithmic transformation can reduce computation load and alleviate data heterogeneity. Feature selection and dimensionality reduction of high dimensional data can also reduce computational complexity well. For details, please refer section 3.2.

Secondly, I construct an objective function and design algorithms to calculate sparse cell-cell similarity matrices. According to the sparse subspace theory, I assume that each cell can be represented by the other cells in the same cell type with a linear combination, and the similarity matrices are composed by these linear combinations. Furthermore, I decompose the large optimization problem into n easier-to-solve optimization problems by column-wise learning. Combining with greedy algorithm, each column of the similarity matrix can be calculated by adding only one non-zero value to the corresponding column at each iteration. For details, please refer section 4.

Thirdly, in order to select an appropriate clustering method based on the cell-cell similarity matrices, I apply two spectral clustering methods (including Laplacian eignmaps + k-means clustering and Laplacian eignmaps + Gassian mixture model (GMM) clustering), a hierarchical clustering method and a Louvain clustering method to identify cell groups with eight scRNA-seq data sets separately. For details, please refer section 5.1.

Finally, I combine LSSM with t-SNE to reveal the visualization of the data points in two-dimensional space from high-dimensional space. For details, please refer section 5.2.

The framework of this study is as Figure 1.2, and the specific process is detailed in the following chapters.

1.3 Motivation and contributions

1.3.1 Motivation

In single-cell data analysis, one of the problems with most existing methods of constructing cell-cell similarity matrices is that, the similarity matrices are calculated based on the gene expression of two cells only. As a result, they have the following issues.

- Firstly, constructing similarity matrices based on pairwise relationships cannot effectively reduce the

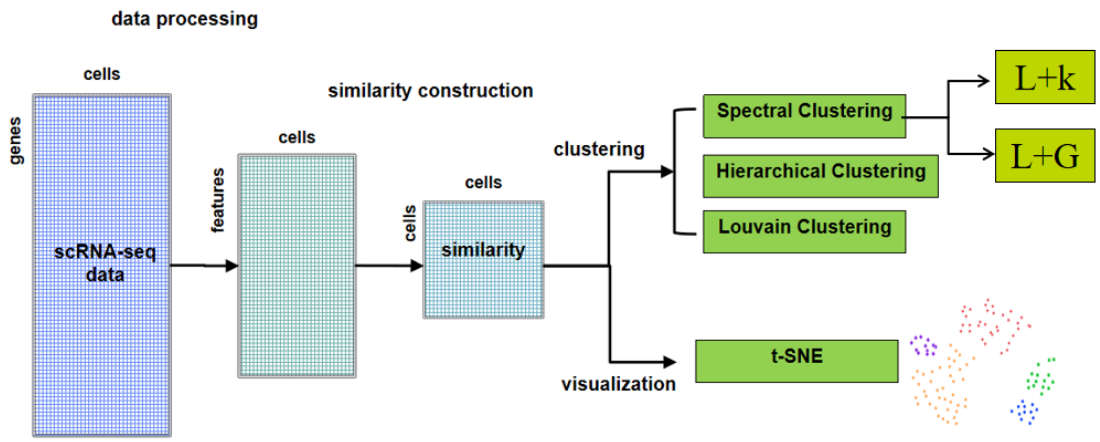

Figure 1.2: The framework of this study

The framework includes: 1) data processing 2) sparse cell-cell similarity matrix construction 3) clustering 4) visualization. L+k represents Laplacian eignmaps + k-means clustering. L+G represents Laplacian eignmaps + Gassian mixture model clustering.

impact of noise in single cell gene expression data [68], because in high-dimensional scRNA-seq data, some relationships may be caused by the noise. Hence, such similarity matrix tends to be more dense due to noise and reduces the accuracy and efficiency of clustering.

- Secondly, most researchers do not take the information of individual cells into the context of the whole data to explore similarity relationships among cells. As a result, constructing pair-wise similarity matrices simplifies the problem by failing to consider the relationship among multiple cells. Therefore, the clustering methods based on pair-wise cell similarity may be ineffective in cell type identification task.

Due to the above problems, in this research, I want to leverage the idea of putting the information of a single cell into the convex of the whole data to construct the similarity matrices. More specifically, I need to address the following tasks.

- I need to look for a calculation method capable of building the relationships between one cell and multiple cells.
- The method also needs to keep the similarity matrix to be sparse.
- After cell-cell similarity matrices are constructed, I need to find an appropriate clustering method to identify cell populations according to the clustering qualities.
- Visualization in scRNA-seq data analysis is essential for biological interpretation and evaluating cell populations. I want to combine visualization methods with the sparse cell-cell similarity matrices to

visualize data points in two-dimensional space.

1.3.2 Contributions

Based on the motivations above, in this study, I propose a novel framework of single-cell data clustering. More specifically, I make the following contributions.

- According to sparse subspace theory [33], I assume that each cell can be represented by linear combination of the other cells in the same population. Therefore, the relationships between a cell and other cells in the same subspace can be built. Based on the assumption, I construct a convex optimisation objective function and a convex hull to calculate the similarity matrix between cells.
- To solve the convex optimization objective function, I design an effective and convergent algorithm by combining column-wise learning with greedy algorithm. In later section, I will present that using column-wise learning, the original large and complex problem of finding the similarity matrices can be recomposed into easier-to-solve smaller optimization problems through finding each column of the similarity matrices. In order to guarantee each column of the cell-cell similarity matrix to be sparse, I utilize greedy algorithm to add only one non-zero value to the corresponding column at each iteration.
- Based on the constructed sparse cell-cell similarity matrices, I apply several clustering methods, including two spectral clustering methods (Laplacian eigenmaps + k-means clustering, Laplacian eignmaps + GMM clustering), a hierarchical clustering method and a Louvain clustering method to identify cells over eight scRNA-seq data sets respectively. I will present it in the later section, and the results show that my method (LSSM) combining with spectral clustering, especially Laplacian eignmaps + k-means clustering, outperforms to most competing methods on the multiple data sets.
- For utilization, I combine the cell-cell similarity matrix, computed by LSSM, with t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize scRNA-seq data sets in two-dimensional space. The results show that for most data points, if they are in the same cell populations, they are grouped nicely. At the same time, from different cell populations, they are clear separation.

1.4 Organization

The thesis includes 8 chapters. Apart from the 1st chapter, the remaining chapters are organized as follows. In chapter 2, some background knowledge is introduced from three major aspects: 1) single cell sequencing technologies, 2) unsupervised clustering and 3) convex optimization. In chapter 3, I describe eight scRNA-seq data sets and data preprocessing. For these data sets, I explain how the ground truths (cell labels) are

annotated by the original papers and where to get the data sets. In chapter 4, I demonstrate the details about constructing an convex optimization objective function and an algorithm to compute cell-cell sparse similarity matrices with their derivations and proofs in detail. Subsequently, in chapter 5, based on the constructed sparse similarity matrices, I introduce several clustering methods (two spectral clustering methods (Laplacian eigenmaps + k-means clustering, Laplacian eignmaps + GMM clustering), a hierarchical clustering method and a Louvain clustering method, which are denoted as LSSM-EK, LSSM-EG, LSSM-H and LSSM-L for identifying cell populations. Also, I use t-SNE to visualize scRNA-seq data based on the similarity matrices. In chapter 6, I depict three clustering evaluation metrics: Purity, normalized mutual information (NMI), and adjusted rand index (ARI). In chapter 7, I show the results of clustering and visualization. I compare the clustering results between the five state-of-the-art methods (NMF, SIMLR, MPSSC, DropClust, SC3) and my methods (LSSM-EK, LSSM-EG, LSSM-H and LSSM-L) based on the three clustering evaluation metrics. Besides, I present the visualization performance by using t-SNE. Finally, in chapter 8, I summarize the whole research in this thesis and discuss several potential future improvement directions. In the appendix, I list the Python code used in this study.

A paper based on the main results of this thesis has been accepted as a regular conference paper by IEEE International Conference on Bioinformatics and Biomedicine 2021 (IEEE BIBM 2021) with the acceptance rate of 19.6%.

2 Background

2.1 Single cell sequencing

2.1.1 A brief overview of the development of sequencing technologies

The generation of first-generation sequencing technology

First-generation sequencing technologies include Sanger and Maxam-Gilbert sequencing technologies. In the 1970s, Maxam and Gilbert [80] and Sanger et al. [96] developed methods for determining nucleotide sequences in DNA by fragmentation and chain termination severally, which provided tools for deciphering complete genes and later, entire genomes. While the approach developed by Sanger et al., usually called Sanger sequencing, required less processing of toxic chemicals and radioisotopes compared with Maxam and Gilbert's approach, so Sanger sequencing became the most popular DNA sequencing method for the next 30 years [100]. With the demand for ever-increasing throughput, Sanger method was automated, which can be used to analyze large-scare metagenomic libraries [94, 61].

The generation of next-generation sequencing (NGS) technology

Next-generation sequencing technologies, also called high throughput sequencing technologies, have mostly replaced the first generation sequencing technologies lately. The Human Genome Project formally launched in 1990 and was declared complete in 2003 [77]. It requires much faster, cheaper and larger scale sequencing technologies, that provoked the development and commercialization of NGS technologies [58, 94]. For the entire human genome sequencing, NGS technologies can finish in one day, while Sanger sequencing needs more than a decade to complete. Compared to Sanger sequencing method, NGS technologies have lower sample input requirements, higher accuracy, and detecting variants at lower allele frequencies features [5]. Therefore, NGS technologies have opened new perspectives for genomics research and diagnostic applications, because of their high speed and throughput of data generation [5, 20].

2.1.2 The workflow of single-cell RNA sequencing

To clearly understand single-cell RNA sequencing (scRNA-seq) technology, we should know RNA sequencing (RNA-seq) first. RNA sequencing (RNA-seq), which is mainly applied to examine the quantity and sequences of RNA in a sample, is one of the most widely using NGS technology. RNA-seq needs reverse transcription of RNA into complementary DNA (cDNA), which is DNA synthesized from a single-stranded RNA, and then examine the qualities and sequences of these cDNAs by using NGS technology [25, 84], because cDNA is more stable and resistant to be degraded than RNA. Genes with higher expression, can generate more sequence reads of DNA, RNA and cDNA. Therefore, RNA-seq technology provides a digital read of gene expression, and the number of DNA sequence reads is consistent with the biological expression level of specific genes in a sample [90]. Traditionally, researchers obtain RNA-seq data from bulk of cells, which measure the average gene expression level in a large number of input cells [79, 107]. However, these measurements may cover key differences of individual cells in the bulk of cells. Therefore, it is insufficient for studying cellular heterogeneity, which measures the variability among cells.

Single-cell RNA sequencing (scRNA-seq) technologies, which depict the sequence information from individual cells with a range of technologies for sensitive, highly multiplexed or combinatorially barcoded profiling, are now becoming a powerful tool for mapping cell-cell variation on a genome-wide scale [106, 93, 46]. ScRNA-seq technologies have been employed in many disparate fields of biology, such as microbiology, neurobiology, tissue mosaicism, immunology, and cancer research over the past few decades [118].

Apart from the fact that scRNA-seq technologies require cells to be isolated first, the general scRNA-seq workflow (Figure 2.1) is similar with bulk RNA-seq technologies. The steps to generate scRNA-seq data from biological samples are as follows.

- Firstly, solid tissues need to be dissociated for producing a monodispersed suspension of viable cells by using mechanical force and/or biological enzymes, such as collagenase, hyaluronidase and deoxyribonuclease. It is important to ensure that the cell surface epitopes are not affected.

- Secondly, single cells are captured and isolated. There are various methods, including low-throughput single-cell isolationapproach (e.g. laser capture microdissection, fluorescence-activated cell sorting (FACS)), and high-throughput single-cell isolation approach (e.g. microfluidics circuits, droplet fluidics platforms), to capture and isolate single cells.

- Thirdly, library construction is performed. Library construction is the process of capturing intracellular mRNA, transcribing mRNA into cDNA and amplifying cDNA molecules. After library construction, cDNA libraries are labelled with cellular barcodes. And then they are pooled together for sequencing

by NGS technologies.

- Fourthly, bioinformatic and /or computational approaches can be used to analyze scRNA-seq data, once the scRNA-seq data is produced. Generally, the common steps of analyzing scRNA-seq data includes quality control, normalization, feature selection, dimensionality reduction, cell-cell distance/similarity matrix calculation and unsupervised clustering for identifying cell types [59].

More detailed explanations of the workflow to do scRNA-seq experiments could be found in [74, 82].

2.2 Unsupervised clustering

Clustering (or cluster analysis) is a common unsupervised machine learning approach, which learns patterns from a data set without classified or labeled responses[66]. Unsupervised clustering aims to divide the populations or data points into a number of clusters, such that the data points in the same group are more similar to each other than those in other groups [42]. Figure 2.2 shows a clustering diagram.

According to different notions of a cluster, more than 100 different clustering algorithms have been developed [88]. Each clustering method follows a diverse set of rules to define the 'similarity' between data points. I introduce several typical cluster models (connectivity models, centroid models, distribution models, graph-based models, density models), including their definition, some clustering examples, and their pros and cons.

Connectivity models Connectivity models are based on the core idea that the closer data points are to each other in data space, the more similar they are than the data points lying farther away [81, 88]. For example, hierarchical clustering algorithms, which hunts for building a hierarchy of clusters, cluster data points based on distance matrices [52]. Although connectivity model clustering methods are robust with input parameters and less affected by cluster shapes, when the data set has too many outliers, theses methods are not very suitable, since plotting these outliers is complex and time consuming [87, 95].

Centroid models In centriod model clustering methods, the clusters is derived by the closeness of data points to the centroids of the clusters, and these algorithms usually run iteratively to find the local optimal [88]. The most widely-used centroid-based clustering algorithm is k-means clustering. Most centroid model clustering algorithms are simple and fast, but they require specify cluster numbers in advance [56]. There are several methods to determine the optimal number of clusters, such as elbow method [62], average silhouette method [35] and gap statistic method [111].

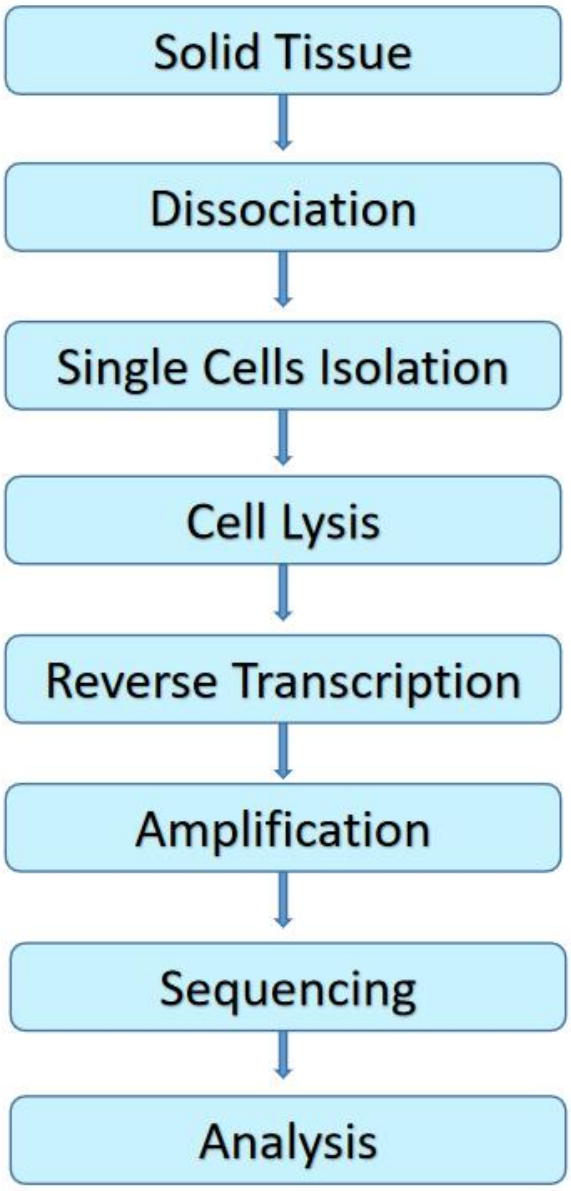

Figure 2.1: The workflow of single cell RNA sequencing experiments

The steps of a typical scRNA-seq experiment workflow are: 1) dissociate biological solid tissue samples 2) isolate single cells 3) lysis cells for getting mRNA 4) mRNA is converted into cDNA with reverse transcription 5) cDNA amplication (often by polymerase chain reaction (PCR)) 6) sequence 7) use bioinformatic and /or computational methods to analyse scRNA-seq data.

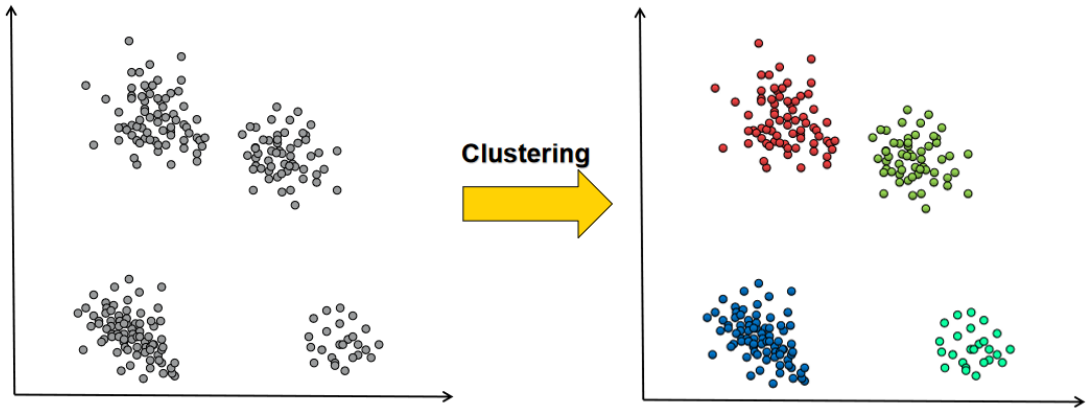

Figure 2.2: A clustering diagram

In the first sub-graph, the data points are in one single group, while after clustering, the points are identified into four groups. Different colors represent different clusters.

Distribution models The clustering algorithms based on distribution models assume that all objectives in one cluster belong to the same distribution [88]. Hence, the use of distribution models, such as Gaussian distribution, plays a key role in distribution-based clustering methods. A good property of distribution-based models is that they are very similar to the way that a artificial data set is generated: by sampling random objectives from a distribution [102]. Therefore, these models enable us to know other information beyond cluster assignment of objectives, like revealing the correlation and dependence between data points. However, these models suffer from a key problem: over-fitting [49].

Graph-based models In graph-based models, data points are treated as the vertices of a graph and edges are constructed according to the distance or similarity between data points so that a weighted graph can be formed by theses vertices and edges. And then, the clustering methods are achieved through graph cutting. In graph cutting, the graph is divided into multiple sub-graphs, and these sub-graphs are the corresponding clusters. Usually, these models employ sparsification of similarity (or distance) matrices under diverse heuristics to extract similarity graphs for maintaining the main properties of data sets [8]. Sometimes these models invoke the idea of spectrum theory, that is, applying the spectral decomposition of the Laplacian matrices of graphs [72]. Graph structures can represent the relationships of data points in a clear and manageable way, but the methods of graph construction and the choices of parameters (i.e. sparsity) impact the performance of clustering results greatly [26, 76].

Density models The purpose of density-based clustering approaches is to identify clusters as areas with high point density, which are separated by areas with low point density, so that they can be arbitrarily shaped in data spaces [64]. More specifically, density-based models discover clusters by contiguous regions

of low density of objects, while these data points in low-density areas are usually considered to be noise or border points [17]. The most obvious advantage of the clustering methods is that they allow any shape of distribution as long as dense regions can be connected. However, it is not easy for them to deal with data with varying densities and high dimensions. In addition, these algorithms do not assign border points to clusters. Popular examples are density-based spatial clustering of applications with noise (DBSCAN) [34] and ordering points to identify the clustering structure (OPTICS) [3].

2.3 Convex optimization

Convex optimization is a typical class of optimization problems, which is a process to seek out the optimal in the set of feasible solutions. There are many benefits to recognize or formulate a problem as a convex optimization problem [11]. The most basic one is that this problem can be solved more reliably and effectively [11], because of a good property of convex optimization: its local optimal solution is the global optimal solution. In convex optimization, there are three common solving methods: gradient method, dual method and alternating direction method of multipliers (ADMM) method. In the following, I will introduce several basic definitions and concepts of convex set, convex combination, convex hull, convex function and convex problem. If readers is interested in more details of convex optimization, he/she can read this publication [11].

2.3.1 Convex set, convex combination, convex hull

Convex set C is a convex set, if the line segment between any two points in C lies in C. It can be represented as:

$$\theta x_1 + (1-\theta)x_2 \in C, \tag{2.1}$$

where $x_1, x_2 \in C, 0 \leq \theta \leq 1$.

According to the definition above, several common examples of convex sets can be shown easily, like any straight line, any line segment, sphere, ellipsoid, hyperplane and simplex. Figure 2.3 shows two examples: one convex set and one non-convex set to further help readers to understand convex set.

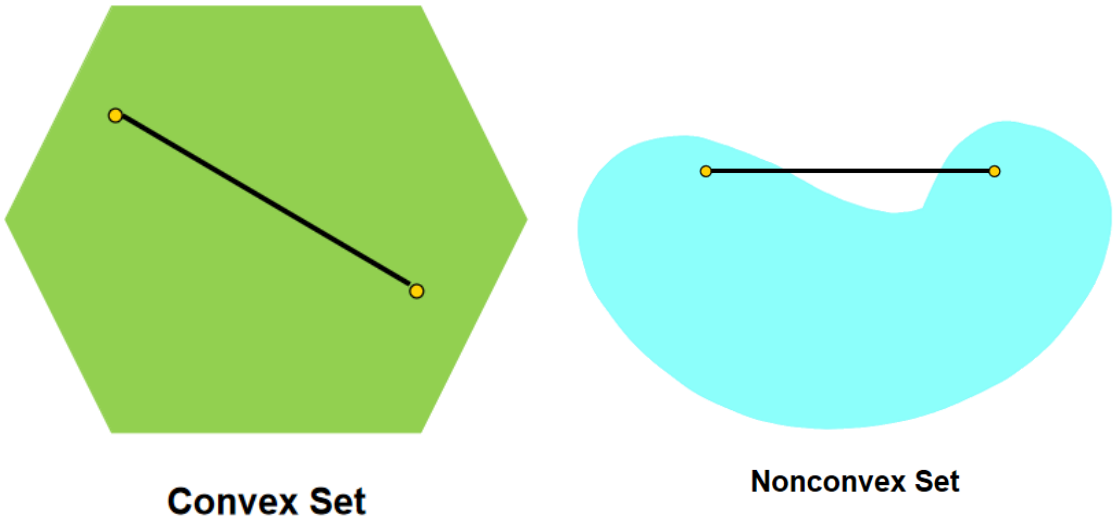

Figure 2.3: A convex set and a non-convex set

Convex combination Point x is a convex combination of points $x_1, x_2, \cdots, x_k$, if it can be represented as:

$$x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k \tag{2.2}$$

with $\theta_1 + \theta_2 + \cdots + \theta_k = 1$, and $\theta_i \geq 0$, $i = 1, 2, \cdots, k$. A convex combination of points can be thought as a mixture or weighted average of the points.

Convex hull The convex hull of a convex set C, denoted as $convC$, is defined as the set of all convex combinations of points in C, specifically:

$$convC = \theta x_1 + \cdots + \theta x_k \quad | \quad x_i \in C, \theta_1 + \cdots + \theta_k = 1, \theta_i \geq 0, i = 1, \cdots, k \tag{2.3}$$

By the definition, it can be proved that the convex hull $convC$ is always convex, which is the smallest convex set that contains C, which can be expressed as: $C \subseteq convC$. Figure 2.4 is an example of a convex hull.

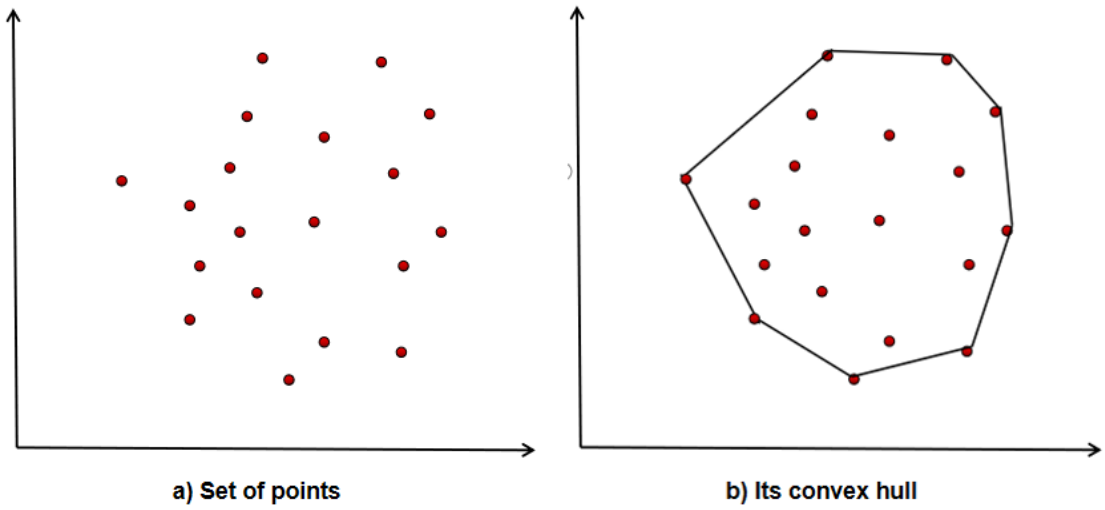

Figure 2.4: A convex set of points and its convex hull

2.3.2 Convex function

Definition A function $f : R^n \to R$ is a convex function, if the domain of f, denoted as $domf$, is a convex set, and for $\forall x, y \in domf$, $\theta \in [0, 1]$, there is:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \tag{2.4}$$

Geometrically, this inequality means that the line segment between $(x, f(x))$ and $(y, f(y))$, which is the chord from $(x, f(x))$ to $(y, f(y))$, lies above the graph of f. There is an simple example in Figure 2.5.

Figure 2.5: Graph of an example for convex function
The chord from $(x, f(x))$ to $(y, f(y))$, lies above the graph of f.

Judgment criteria of convex function After introducing the definition of convex function, it is obvious that we can use the definition of convex function to determine whether a function is convex or not. However, in practice, there are simpler and more intuitive approaches to determine convex functions. I will introduce two of these approaches: the first-order and second-order convexity conditions in the following.

1. **First-order convexity conditions**

Suppose $f : R^n \to R$ is a differentiable function, that is, its gradient ∇f exists at each point in the domain of f, denoted as $domf$. Then f is convex if and only if :

1) $domf$ is a convex domain;

2) $f(y) \geq f(x) + \nabla f^T(x)(y - x)$, for all $x, y \in domf$, where $\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \cdots, \frac{\partial f(x)}{\partial x_n}\right)$.

2. **Second-order convexity conditions**

Suppose $f : R^n \to R$ is twice differentiable over an open domain. Then f is convex if and only if :

1) $domf$ is a convex domain;

2) $\nabla^2 f(x) \succeq 0$, for all $x \in domf$, where $\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$, $i, j = 1, \cdots, n$.

The Hessian matrix ($\nabla^2 f(x)$) is positive semi-definite, represented as: $\nabla^2 f(x) \succeq 0$, which means the curvature constant of f is non-negative for all $x \in domf$. The details of curvature constant can be seen in section 4.2.3.

Common convex functions Here, some common convex functions, divided into two categories: scalar convex functions and vector convex functions are introduced as below.

1. **Scalar convex functions:**

- affine: $ax + b$
- exponential: e^{ax}
- powers of absolute value: $|x|^p$ on $\mathbf{R}$, for $p \geq 1$
- negative entropy: $xlogx$ on $\mathbf{R}_{++}$

2. **Vector convex functions:**

- affine function on $\mathbf{R}^n$: $a^T x + b$
- maximum function on $\mathbf{R}^n$: $max\{x_1, x_2, \cdots, x_n\}$
- affine function on $\mathbf{R}^{m \times n}$: $f(X) = tr(A^T A) + b = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} X_{ij} + b$
- log-determinant on $\mathbf{R}^{m \times n}$: for a symmetric positive-definite matrix (X), $f(X) = logdetX = log(\prod_i \lambda_i) = \sum_i log\lambda_i$, where $\lambda = \lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of X

2.3.3 Convex problem

Generally, for s convex problem, the objective function is a convex function, and the domain of the objective function is a constrained convex set. The standard form of a convex optimization problem can be written as:

$$\begin{aligned} &\underset{x}{minimize} \quad f_0(x) \\ &s.t. \quad f_i(x) \leq 0, \qquad i = 1, \cdots, m \\ &s.t. \quad h_j(x) = 0, \qquad j = 1, \cdots, p \end{aligned} \tag{2.5}$$

In the function 2.5, $x = [x_1, \cdots, x_n]$ is the variable to be optimized. The function $f_0 : R^n \to R$ is the objective function. The functions $f_i(x) \leq 0$, $i = 1, 2, \cdots, m$ are inequality constraints. And the function $h_j = 0$, $j = 1, 2, \cdots, p$ are equality constraints. The goal is to find the optimal over the feasible set, while satisfy the constraints.

Common convex problems Common convex problems includes: linear program (LP), quadratic program (QP) and semidefinite program (SDP).

A LP can be described as follows:

$$\begin{aligned} &\underset{x}{minimize} \quad c^T x \\ &s.t. \quad Dx \leq d \\ &s.t. \quad Ax = b, \end{aligned} \tag{2.6}$$

where c is a vector, D and A are matrices, d and b are real numbers.

A QP can be described as follows:

$$\begin{aligned} &\underset{x}{minimize} \quad \frac{1}{2} x^T Q x + c^T x \\ &s.t. \quad Dx \leq d \\ &s.t. \quad Ax = b, \end{aligned} \tag{2.7}$$

where c is a vector, Q is a positive semi-definite matrix, D and A are matrices d and b are real numbers.

A SDP can be described as follows:

$$\begin{aligned} &\underset{X}{minimize} \quad CT \\ &s.t. \quad A_i X = b_i, \quad i = 1, 2, \cdots, m \\ &s.t. \quad X \succeq 0, \end{aligned} \tag{2.8}$$

where the variable is the matrix X, A and C are symmetric matrices, b_i is a vector.

3 Data Sources and Data Preprocessing

3.1 Data sources

A problem, I notice with many public scRNA-seq data sets, is that cell type labels in those data sets are usually inferred by clustering methods. For example, in [16], after sequencing cells using scRNA-seq technique, the authors use spectral clustering method to cluster cells. Then, they assign the labels of each cell cluster according to the expression patterns of cell type specific marker genes. As a result, any assessment based on these labels are potentially biased towards methods similar to those used to derive the labels originally [31]. To avoid such artificial risk, in this study, I select eight data sets whose labels are derived from the scRNA-seq data sets themselves. Specifically, the ground truth labels of the chosen data sets are defined independently of any unsupervised clustering methods.

Yan's data [125] Yan et al. get the data set from pre-implantation embryos and embryonic stem cells of human at several key embryonic stages, including zygotes, 2-cell-stage embryos, 4-cell-stage embryos, 8-cell-stage embryos, Morula and early hatching blastocysts. The authors label these cells of different stages according to embryonic development time [109]. For this data set, I access it via SC3 software package (https://github.com/hemberg-lab/SC3).

Deng's data [29] Deng et al. collect the data set from mouse preimplantation embryos. There are 7 cell types including zygote, early 2-cell-stage, mid 2-cell-stage, late 2-cellstage, 4-cell-stage, 8-cell-stage, and 16-cell-stage. The researchers collect mouse embryonic cells at diverse stages of preimplantation development at defined time periods after human chorionic gonadotropin (hCG) administration. This data set is used to reveal independent and stochastic allelic transcription gene expression in mammalian cells. For this data set, I download the processed data from SC3 software package (https://github.com/hemberg-lab/SC3).

Kumar's data [65] Kumar et al. design experiments to characterize transcriptional heterogeneity in mouse characterize transcriptional heterogeneity of mouse pluripotent stem cells (PSCs) through the analysis of single cell expression profiles under different chemical and genetic perturbations. The labels of cells are

defined by the genetic perturbation and the medium where they were grown. For this data set, I download it from Bioconductor (https://bioconductor.org/ packages/DuoClustering2018).

Koh's data [63] Koh et al. culture undifferentiated H7 human embryonic stem cells (H7 hESCs) and induce them to differentiate directly into 8 cell types in a defined medium. Therefore, this data set has 9 subpopulations including H7 hESCs and 8 kinds of H7-derived downstream early mesoderm progenitors. For this data set, I download it from conquer [104]: SRP073808.

Tian's data [110] Tian et al. design a serious of credible gold-standard scRNA sequencing experiments. They identify the ground-truth of three human lung adenocarcinoma cell lines (HCC827, H1975 and H2228) based on known genetic variation. For this data set, I get it from GEO (GSE118767).

Xin's data [122] Xin et al. use stringent inclusion criteria to identify specific genes of four human pancreatic islet cells: α cells, β cells, δ cells and PP cells. For α cells, the specific genes are: GCG, DPP4, FAP, PLCE1, LOXL4, IRX2, TMEM236, IGFBP2, COTL1, SPOCK3, and ARRDC4). For β cells, the specific genes are:INS, ADCYAP1, IAPP, RGS16, DLK1, MEG3, INS-IGF2, and MAFA. For δ cells, the specific genes are: SST, BCHE, HHEX, and RPL7P19), and for PP cells, it is PPY. This data set can be found in GEO under ID code: GSE81608.

Zheng's data [129] Zheng et al. purify 11 cell populations from peripheral blood mononuclear cells (PBMCs), which are purified with fluorescence-activated cell sorting (FACS) [112]. I get these 11 pre-sorted cell types from the 10x Genomics GemCode protocol (https://support.10xgenomics.com/ single-cell-gene - expression/datasets). Then, I combine them into two data sets (denoted as **Zheng11uneq** and **Zheng8eq**). For **Zheng11uneq** data set, I combine all 11 cell types in unequal proportions (200-300 cells per cell population), which are randomly selected from CD14+ monocytes, CD19+ B-cells, CD34+ cells, CD56+ Natural Killer cells, 293T cells, CD4+ T-helper cells, regulatory T-cells, memory T-cells, naive T-cells, naive cytotoxic T-cells and CD8+ cytotoxic T-cells. For **Zheng8eq** data set, I combine 8 cell types in equal proportions (700 cells per cell subpopulation), which are randomly selected from CD14+ monocytes, CD19+ B-cells, CD34+ cells, CD56+ Natural Killer cells, 293T cells, CD4+ T-helper cells, regulatory T-cells, memory T-cells.

More details of all these eight data sets are shown in Table 3.1.

Table 3.1: Summary of the details of eight single-cell data sets

Data sets	No. of cells	No. of genes	No. of clusters
Yan [125]	90	20,214	6
Deng [29]	135	12,548	7
Kumar [65]	246	45,159	3
Koh [63]	651	65,218	10
Tian [110]	902	16,468	3
Xin [122]	1,600	39,851	4
Zheng11uneq [129]	2,750	32,738	11
Zheng8eq [129]	5,600	32,738	8

3.2 Data preprocessing

Once single cells are sequenced, it is usually shown as an expression matrix. In the matrix, each row corresponds to a gene and each column corresponds to a cell. Data preprocessing is essential due to the high dropout rates, low and uneven read coverage and high variability of scRNA-seq data [89]. Figure 3.1 can intuitively show the workflow of data processing.

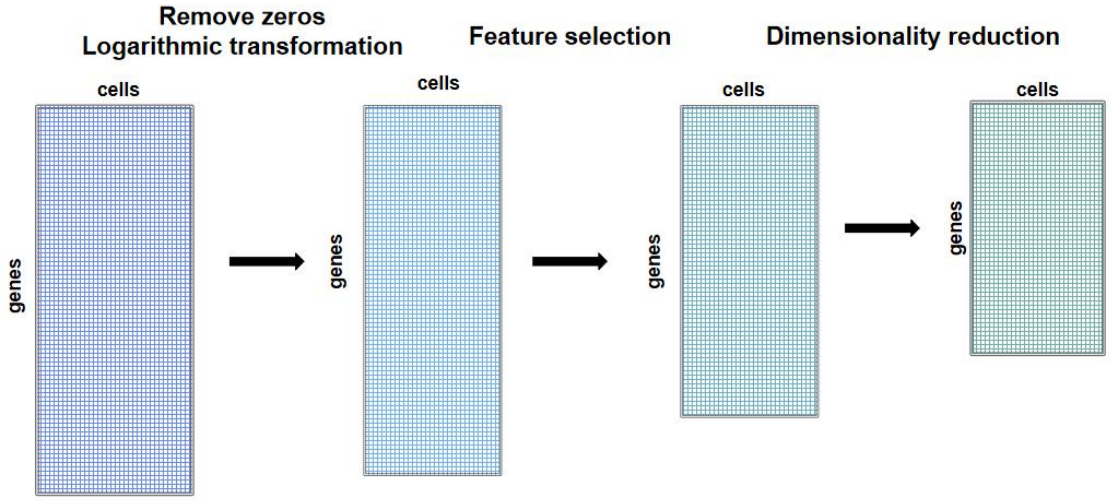

Figure 3.1: Overview of the workflow for the scRNA-seq data processing

3.2.1 Removing zeros

Once raw scRNA-seq data sets are downloaded, I first remove genes with zero expression values in all cells, and I also remove cells with zero expression values in all genes. From data processing perspective, this means I delete all columns or rows where all elements are zero of the matrices. Usually, scRNA-seq data sets are sparse with many biological zeros (i.e. not expressed) and technical zeros (i.e. expressed but not detected)

[69]. These zero values would not help downstream analysis but occupy computing space. Because of this, I remove these zeros.

3.2.2 Logarithmic transformation

After removing zeros, I apply the logarithmic transformation: $\log_{10}(X + 1)$ [117] to transform single cell expression data, where X is the scRNA-seq data set. For scRNA-seq data sets, the expression of different genes often has a large variance, specifically, the values of variance increase monotonically with the value of mean. Because of this, genes highly expressed have high variance values. On the other hand, genes barely expressed or detected have almost zero variance. As a result, it is much easier in the data processing to lose information on genes with smaller expression values. To address this, logarithmic transformations can reduce the information loss of genes with small expression values, especially when the original median absolute deviation (MAD) value is equal to or greater than the median value [75].

3.2.3 Feature selection

Feature selection, or gene filter, is used to select genes with useful biological information, remove genes with high-dimensional random noise, and improve calculation efficiency for the whole algorithm. The choice of genes during the feature selection has a big impact on constructing cell-cell similarity matrices and the performance of downstream analysis. In order to select enough genes with useful biological information while still removing genes with high-dimensional random noise, I select 2000 highly variable genes (HVG). Specifically, it means I select genes which strongly contribute to the variation among cells in a heterogeneous cell population [127]. For simplicity, the HVG selection method I used is to take the genes with the top 2000 largest values of the correlative variances.

3.2.4 Dimensionality reduction

Using a dimensionality reduction method can further improve the calculation efficiency of the whole algorithm. In this study, I use two dimensionality reduction methods: principal components analysis (PCA) [39] and t-distributed stochastic neighbor embedding (t-SNE) [131]. PCA is used for improving the computing efficiency of the workflow of single cell clustering algorithm, while t-SNE is solely applied for scRNA-seq data visualization.

PCA is a method to project data points only on the first few principal components (PCs) to obtain lower-dimensional data but still keep the data's variation as much as possible. PCA is defined as an orthogonal linear transformation which transforms data points to a new coordinate system [54]. The first PC is equivalently

taken as the direction maximizing the variance of the projected data. The second PC is taken as a direction orthogonal to the first PC to maximize the variance of the projection data, and so on.

Specifically, the process of calculating PCs is the calculation of the eigenvectors, which are selected by the corresponding eigenvalues of the correlation matrix $X^T X$ [53], where X is is the scRNA-seq data set. The eigenvalues of $X^T X$ are ranked in descending order to retain as much variance as possible when choosing the first few PCs. However, choosing the exact number of PCs is tricky in single cell analysis. In some studies, the researchers choose the number as a fixed value (i.e. 30, 50), such as [19] and [121].

Given that the sizes of scRNA-seq data sets varies greatly, especially the number of cells spans widely from dozens to tens of thousands, choosing a fixed number may not be appropriate. In this study, inspired by the theory of maximum variance expansion [27], for each data set, I find the largest difference variance value between every two neighbor PCs from the 30th PC to the 50th PC. I set the number of PC to the latter neighbor. Below shows an example of this selection process for Deng's data set (see Figure 3.2).

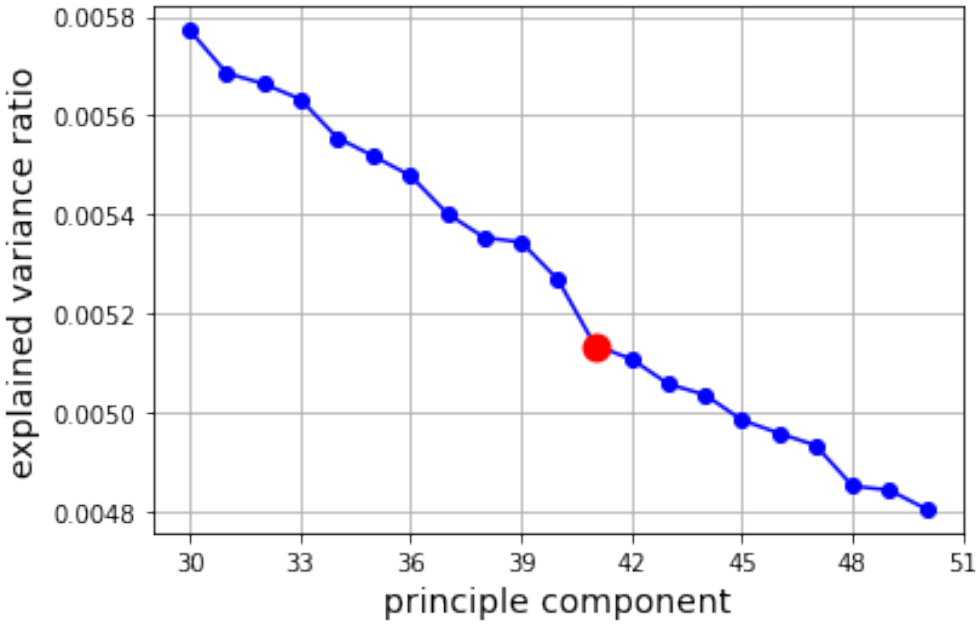

Figure 3.2: Explained variance ratios vs principle components of Deng's data set with PCA
The PCs are ranked by descending order, for example, the 30th PC represents the PC with 30th highest explained variance. The x axis represents the order of the PC, and the y axis represents the corresponding explained variance. We can see the variance difference between the 40th and 41st is larger than any other two PCs from 30th PC to 50th PC, thus I choose 41 as the reserved number of PCs for the algorithm.

Apart from PCA, I also use t-SNE solely for scRNA-seq data visualization. T-SNE maps data points to probability distributions through affinity transformation mainly includes three steps:

- Construct a probability distribution (Gaussian distribution) between high-dimensional data points.

- Construct a probability distribution (t-distribution) of these points in low-dimensional space.
- Minimize the Kullback–Leibler divergence (KL divergence) between these two probability distributions.

I will show more details at section 5.2.

4 Cell-cell Sparse Similarity Matrix Representation

4.1 Problem formulation

A number of studies have been proposed on cell type identification, and many of which rely on the cell-cell similarity construction. Many methods for calculating pairwise similarity or distance matrices have been proposed, such as Pearson correlation coefficient [6], Euclidean squared distance [21] and Gaussian kernel similarity [73]. In this study, inspired by sparse subspace theory, which declares that every data point in one subspace can be represented as a linear combination of the other points in the same subspace [33], I decide to construct sparse cell-cell similarity matrices for identifying cell types.

The process for constructing sparse cell-cell similarity matrices is as follows. Given a set of data points $X = \{x_1, x_2, ...x_n\} \in R^{p \times n}$, where p represents the number of genes and n represents the number of cells. According to sparse subspace theory, I assume that for the cells in the same cell sub-type, a cell could be represented as a linear combination of the other cells. Then X can be represented as $X = XS$, where $S = (S_{ij}) \in T^{n \times n}$ is the similarity matrix of cells. Note that only when cell i and cell j are in the same cell type, meaning cell i can be partly represented by cell j, expressed as $S_{ij} > 0$. On the other hand, if cell i and cell j are in different cell types, meaning cell j cannot explain any part of cell i, expressed as $S_{ij} = 0$. As a result, the cell-cell similarity matrix S should be a sparse matrix because the number of cell types is much less than the number of cells.

To find the similarity matrix S, the optimization objective function should contain two parts:

1. Minimizing the reconstruction loss of gene expression matrix X.

Here, I use linear least squares regression model [119] to represent this loss function: $\min ||X - XS||_F^2$. Because matrix S is a symmetrical matrix, the loss function can be represented as: $\min_{S \in T^{n \times n}} \frac{1}{2}||X - XS||_F^2$.

2. Keeping the sparsity (i.e., minimizing the number of non-zero entries) of the similarity matrix S.

Here, I use ℓ_0 norm, which calculates the number of non-zero entries of a matrix, to guarantee the sparsity of S: $||S||_0$.

To realize the above ideas, the optimization objective function is as follows:

$$\begin{aligned} &\min_{S\in T^{n\times n}} J(S) = \min_{S\in T^{n\times n}} \frac{1}{2}||X - XS||_F^2 + \lambda||S||_0 \\ &s.t. \qquad diag(S) = 0, \quad S_{ij} \geq 0, \end{aligned} \tag{4.1}$$

where λ is a regularization parameter, $||\cdot||_F$ denotes the Frobenius norm which calculates the square root of the sum of all squared elements, $||S||_0$ is the ℓ_0 norm which counts non-zero entries of S, S_{ij} is the element of the ith row and the jth column of S, $T^{n\times n}$ is the set of all $n \times n$ symmetrical matrices.

Although the problem can be described accurately by function (4.1), it is intractable. In order to make the optimization problem (4.1) computationally tractable, in practice, researchers usually relax ℓ_0 norm to ℓ_1 norm, like [24] [41]. Following the similar idea, I relax the objective function in (4.1) as follows.

$$\begin{aligned} &\min_{S\in T^{n\times n}} J(S) = \min_{S\in T^{n\times n}} \frac{1}{2}||X - XS||_F^2 + \lambda||S||_1 \\ &s.t. \qquad diag(S) = 0, \quad s_{ij} \geq 0, \end{aligned} \tag{4.2}$$

where $||S||_1 = \sum_{i,j}|S_{ij}|$ expresses the ℓ_1 norm of the similarity matrix S.

In order to reduce the complexity of the problem, while still retaining sparsity of S, I reform the objective function with column-wise learning. Specifically, for a given cell i_0, the objective function in (4.2) can be recast as:

$$\begin{aligned} \min_{S\in T^{n\times n}} J(S) &= \sum_{i_0} \min_{S_{\cdot i_0}} \left\{ \frac{1}{2}||x_{i_0} - \sum_i S_{ii_0}x_i||^2 + \lambda||S_{\cdot i_0}||_1 \right\} \\ &= \sum_{i_0} \min_{S_{\cdot i_0}} J(S_{\cdot i_0}) \\ s.t. \quad S_{ii} &= 0, \quad S_{ii_0} \geq 0, \end{aligned} \tag{4.3}$$

where $S_{\cdot i_0}$ is the i_0th column of the similarity matrix S, x_{i_0} is the i_0th column of the data set X, S_{ii_0} is the element of the ith row and i_0th column of S, x_{ii_0} is the element of the ith row and i_0th column of X.

To achieve the shift invariant similarities, I can set the sum of each column of matrix S equals to 1[45], and thus have $\lambda||S_{\cdot i_0}||_1 = \lambda$, so that the regularization term λ has no impact on the objective function (4.3).

As a result, I eliminate $\lambda||S_{.i_0}||_1$ and get the following optimization problem.

$$\begin{aligned} &\min J(S_{.i_0}) = min\frac{1}{2}||x_{i_0} - \sum_i S_{ii_0}x_i||^2 \\ &s.t. \qquad S_{ii} = 0, \quad S_{ii_0} \geq 0, \quad \sum_i S_{ii_0} = 1 \end{aligned} \tag{4.4}$$

By now, the large optimization problem (4.2) can be solved by optimizing n small optimization problems (4.4), where n is the total number of cells of a scRNA-seq data set.

To make the notation simple, for a given i_0, let $\alpha \in \mathbf{R}^n$ with $\alpha_{i_0} = -1$, $\sum \alpha_i = 0$, $\alpha_i \in [0, 1]$ and $i \neq i_0$, so the objective function in (4.4) can be further rewritten as:

$$J = \min_{\alpha \in \Omega_{i_0}} J(\alpha) = min\frac{1}{2}||X\alpha||^2 \tag{4.5}$$

where $\Omega_{i_0} = \left\{ \sum \alpha_i = 0, \alpha_{i_0} = -1, \underset{i \neq i_0}{\alpha_i} \in [0, 1], \alpha \in \mathbf{R}^n \right\}$ is the optimization domain of $J(\alpha)$. According to the definition of convex set, it is easy to observe that for a given i_{i_0}, Ω_{i_0} is a convex set in $\mathbf{R}^n$.

From the function: $J(\alpha) = \frac{1}{2}||X\alpha||^2$, we can get: $\nabla^2 f(\alpha) \succeq 0$. According to second-order convexity conditions, mentioned in section 2.3.2, $J(\alpha) = \frac{1}{2}||X\alpha||^2$ is a convex function of α on Ω_{i_0}. Therefore, the problem (4.5) is a convex optimization problem.

4.2 A convergent algorithm for sparse similarity learning

4.2.1 Related works

Greedy algorithm Greedy algorithm is a common method to find the optimal solution of a problem. This method divides the solution process piece by piece, and select the best/optimal in the current state. Specifically, it selects a locally optimal solution in every piece, hoping that the final stacked result is the globally best/optimal solution.

Greedy on a convex set The published paper [47] presents a basic algorithm, called "greedy on a convex set". The algorithm is showed below (see Algorithm 1). The algorithm is used to solve the convex function f: $\underset{x \in D}{min} f(x)$, where D is the domain of f, and also a convex set. Based on this algorithm, I design a new algorithm detailed in Section 4.2.2, to calculate sparse cell-cell similarity matrix.

Note: $ExactLinear(\bigtriangledown f(\alpha^{(k)}), D)$ is a method that minimizes the linear function $\left\langle x, f(\alpha^{(k)} \right\rangle$ over the compact convex domain D, where $\langle \cdot \rangle$ represents inner product. The algorithm returns a point $s \in D$.

Algorithm 1: Greedy on a convex set

Input: Convex function f, convex set D, target accuracy ε $X \in R^{p\times n}, iter, \varepsilon$;
Output: ε-approximate solution for $\underset{x\in D}{min} f(x)$;
Pick an arbitrary starting point $x^(0) \in D$;
for $k = 0 \cdots \infty$ **do**
    Let $\alpha := \frac{2}{k+2}$;
    Compute $s :=$ ExactLinear$(\bigtriangledown f(\alpha^{(k)}), D)$;
    Update $x^{(k+1)} := x^{(k)} + \alpha(s - x^{(k)})$;
end

4.2.2 Main algorithm for sparse similarity learning

Observation 1 The domain $\Omega_{i_0} = \left\{ \sum \alpha_i = 0, \alpha_{i_0} = -1, \underset{i\neq i_0}{\alpha_i} \in [0,1], \alpha \in \mathbf{R}^n \right\}$ of function 4.5 is a convex hull. It can be described as follows:

$$conv(\Omega_{i_0}) = conv\{e_i - e_{i_0} \quad | \quad i \neq i_0\}$$

Note that $conv(\Omega_{i_0})$ is a convex hull with $n-1$ vertices. For given i and i_0, $(e_i - e_{i_0})$ is an n-dimensional vector, whose i_0th element is -1 and $i(\neq i_0)$th element is 1, while its other elements are 0.

Proof According to the definition of convex hull. My goal is to show: $\Omega_{i_0} \subseteq conv(\Omega_{i_0})$.

To do so, take any $\alpha_i \in [0,1]$ with $i \neq i_0$, $\sum\limits_{i\neq i_0} \alpha_i = 1$, and $\alpha \in \mathbf{R}^n$. For a given i_0, we have:

$$\begin{aligned}
\alpha &= (\alpha_1, \alpha_2, \cdots, \underset{\uparrow}{-1}_{i_0}, \cdots, \alpha_n) \\
&= (\alpha_1, \alpha_2, \cdots, \underset{\uparrow}{0}_{i_0}, \cdots, \alpha_n) - (0, 0, \cdots, \underset{\uparrow}{1}_{i_0}, \cdots, 0) \\
&= \sum_{i\neq i_0} \alpha_i (0, 0, \cdots, \underset{\uparrow}{1}_{i}, 0, \cdots, 0) - (0, 0, \cdots, \underset{\uparrow}{1}_{i_0}, \cdots, 0) \\
&= \sum_{i\neq i_0} \alpha_i e_i - e_{i_0}
\end{aligned} \tag{4.6}$$

$\alpha_i \in [0,1]$, so $\alpha = \sum\limits_{i\neq i_0} \alpha_i e_i - e_{i_0} \subseteq e_i - e_{i_0}$with $i \neq i_0$, which means $\Omega_{i_0} \subseteq conv(\Omega_{i_0})$.

The following algorithm (see algorithm 2) is the main algorithm for the optimization problem 4.5 to calculate sparse cell-cell similarity matrices based on scRNA-seq data sets.

Parameter setting of algorithm 2: 1) The choice of initial i for α_0: although any i from 1 to n and $i \neq i_0$ does not affect the convergence of Algorithm 1, it may affect the sparsity of α. To make α more sparse, for a given i_0, I choose initial i by the following formula: $\underset{i}{arg}(X^T x_{i_0})$, which indicates the ith cell is the most

Algorithm 2: Sparse similarity learning

Input: data matrix $X \in R^{p\times n}, iter, \varepsilon$;
Output: sparse similarity matrix (S);
initialization: $S \in R^{n\times n}$ to be a zero matrix;
for i_0 *in [1 : n]* **do**
  choose $i \neq i_0$, let $\alpha_0 = e_i - e_{i_0}$, $\beta_0 = X\alpha_0$, $a_0 = 1$, $k = 0$;
  while *True* **do**
    k = k+1;
    Let $a_k = \frac{1+\sqrt{4a_{k-1}^2+1}}{2}$;
      Compute $i_+ := \text{ExactLinear}(\bigtriangledown J(\alpha^{(k-1)}), \Omega_{i_0})$;
    Update $\alpha^{(k)} := \alpha^{(k-1)} + \frac{1}{a_{k-1}}(e_{i+} - e_{i0} - \alpha^{(k-1)})$;
    Update $\beta^{(k)} := \beta^{(k-1)} + \frac{1}{a_{k-1}}(x_{i+} - x_{i0} - \beta^{(k-1)})$;
    if $k > iter$ **then**
      break;
      $err = (||\beta^{(k-1)}||^2 - ||\beta^{(k)}||^2)/||X||^2$;
    end
    if $err < \varepsilon$ **then**
      $S[:, i_0] = \alpha^{(k)}$;
    end
  end
end
$diag(S) = 0$;
$S = \frac{S+S^T}{2}$;

similar to the i_0th cell of the scRNA-seq data set. 2) The value of parameter *iter* represents the maximum iterations, which also determines the sparsity of α, as each iteration of updating $\alpha^{(k)}$ only adds one nonzero element of α. It is empirically set as $40\% \times n$ in this study, where n is the total cell number of a sc-RNA seq data set. 3) The value of ε controls the maximum difference between two consecutive iterations, which is set to 1×10^{-7} in this study.

The following explanations can help to understand Algorithm 2.

1. $ExactLinear(\bigtriangledown J(\alpha^{k-1}), \Omega_{i_0})$ in Algorithm 2 is used to find a column index i_+ such that the linear function $\langle \bigtriangledown J(\alpha^{k-1}), \Omega_{i_0} \rangle$ is minimized over the optimization domain Ω_{i_0} at the vertex of convex hall Ω_{i_0}.

2. In terms of the smooth convex optimization literature (i.e. [13]) the vectors $(e_{i_+} - e_{i_0})$ have negative scalar product with the gradient, i.e. $\left\langle e_{i_+} - e_{i_0}, \bigtriangledown J(\alpha^{k-1} \right\rangle < 0$, which is called descent direction [18]. As a result, in Algorithm 2, it always choose descent steps staying in the domain D, which is different with the traditional gradient descend techniques, which usually use arbitrary directions and need to project back onto D after each step.

3. From the definition of $J(\alpha) = \frac{1}{2}||X\alpha||^2$, we have $\nabla J(\alpha) = X^T X\alpha$. Then we can get:

$$\begin{aligned} &\text{ExactLinear}(\bigtriangledown J(\alpha^{k-1}), \Omega_{i_0}) \\ &\quad = \underset{0\leq i\leq n}{argmin} \left\{ (e_i - e_{i_0})^T \nabla J(\alpha^{(k-1)}) \right\} \\ &\quad = \underset{0\leq i\leq n}{argmin} \left\{ (x_i - x_{i_0})^T X\alpha^{(k-1)} \right\}, \end{aligned} \tag{4.7}$$

where x_i is the ith column of data matrix X, x_{i_0} is the i_0th column of data matrix X. Note that for given i and i_0, the vector $(e_i - e_{i_0})$ have only two non-zero values (the ith element is 1, and the i_0th element is -1), so at each iteration of computing $ExactLinear(\bigtriangledown J(\alpha^{k-1}), \Omega_{i_0})$, updating $\alpha^{(k)}$ and updating $\beta^{(k)}$, only one non-zero value is added to the corresponding column of similarity matrix S.

4. In the algorithm 2, I replace $X\alpha^{(k)}$ with $\beta^{(k)}$ to reduce the computational effort in solving the optimization problem of $J(\alpha)$.

5. To keep cell-cell similarity matrix (S) to be symmetry, S is calculated as $S = \frac{(S+S^T)}{2}$.

The derivation and convergence of algorithm 2 is in 4.2.3

4.2.3 The derivation of the algorithm for sparse similarity learning

In this part, I consider to solve the following convex optimization problem:

$$\underset{x\in D}{min} f(x), \tag{4.8}$$

where $D \subset \mathbf{R}^n$ is a convex set, and $f(x)$ is a convex function on D.

Dual gap as a measure of approximate quality

For given points $x, y \in D$, where $D \subset \mathbf{R}^n$ is a compact set, $\nabla f(x)$ is the gradient of $f(x)$, the dual function value $\omega(x, \nabla f(x))$ is defined as follows:

$$\omega(x, \nabla f(x)) := \underset{y\in D}{min} \left\{ f(x) + \langle y - x, \nabla f(x)\rangle \right\}, \tag{4.9}$$

where $\langle\cdot\rangle$ represents inner product.

According to weak duality theorem [40], for each pair $x, y \in D$, there is:

$$\omega(x, \nabla f(x)) \leq f(y) \tag{4.10}$$

At any point $x \in D$, the duality gap $g(x, \nabla f(x))$, which is the difference between the original solution $f(x)$ and the dual solution $w(x, \nabla f(x))$, is defined as:

$$\begin{aligned} g(x, \nabla f(x)) &:= f(x) - w(x, \nabla f(x)) \\ &= f(x) - \min_{y \in D} \{f(x) + \langle y - x, \nabla f(x) \rangle\} \\ &= \max_{y \in D} \{\langle x - y, \nabla f(x) \rangle\} \end{aligned} \tag{4.11}$$

Assuming that $x^* \in D$ is the optimal solution of function 4.8, according to weak duality theorem, which declares the value of duality gap is always equal or greater than 0, I get:

$$g(x, \nabla f(x)) \geq f(x) - f(x^*) \geq 0, \quad \forall x \in D \tag{4.12}$$

The quantity $f(x) - f(x^*)$ is called the primal error at point x. While the quantity is usually impossible to calculate because x^* is unknown, the duality gap $g(x, \nabla f(x))$, in case like defining, an upper bound on the primal error is easier to compute. This property makes it an useful measure such as being a stopping criterion in the practical optimizers or heuristics.

Convergence of the algorithm

Curvature constant The curvature constant C_f of a convex and differential function f with compact domain D is defined as follows.

$$C_f = sup \frac{1}{\theta^2} \{f(y) - f(x) - \langle y - x, \nabla f(x) \rangle\}, \tag{4.13}$$

where $x, y \in D$, $\theta \in (0, 1]$.

Imaging that the optimization procedure at the current state $x = x^{(k-1)}$, the next iteration can be expressed as : $y := x^{(k)} = x^{(k-1)} + \theta(s - x^{(k-1)})$, and θ determines the step size of gradient descent.

In the case of our study, combining the algorithm 2, $x^{(k-1)} := \alpha^{(k-1)}$, $\theta := \frac{1}{a_{k-1}}$, $s := e_{i+} - e_{i_0}$, $y := \alpha^{(k)} = \alpha^{(k-1)} + \frac{1}{a_{k-1}}(e_{i+} - e_{i_0} - \alpha^{(k-1)})$, the domain is Ω_{i_0}. Here, bounded C_f means the deviation of f at $\alpha^{(k-1)}$ from the "best" linear prediction given by $\nabla f(\alpha^{(k-1)})$ is bounded, where the acceptable deviation is a_{k-1}^2.

Lemma 4.1 For $\forall f$, f is a quadratic differentiable convex function, over a compact convex domain D,

it has:

$$C_f \leq \frac{1}{2} diam(D)^2 \sup_{x \in D} \lambda_{max}(\nabla^2 f(x)), \tag{4.14}$$

where $\nabla^2 f(x)$ is the Hessian matrix of function $f(x)$. From this limitation, it can be concluded that the upper bound of C_f is the largest eigenvalue, denoted as $\lambda_{max}(\nabla^2 f(x))$ of the Hessian matrix $\nabla^2 f(x)$, which is scaled or formalized by the Euclidean diameter of the field.

Proof Firstly, according to Quadratic (the 2nd Order) Taylor approximation [30], for $f(y)$ at point x, we have:

$$\begin{aligned} f(y) &= f(x + \theta(s - x)) \\ &= f(x) + \theta(s - x)^T \nabla f(x) + \frac{\theta^2}{2}(s - x)^T \nabla^2 f(x)(s - x) \end{aligned} \tag{4.15}$$

Combining function 4.13 and function 4.15, we can get:

$$\begin{aligned} C_f = & \sup_{\substack{x,y \in D \\ \theta \in (0,1] \\ y = x + \theta(s-x)}} \frac{1}{\theta^2} \{f(y) - f(x) - \langle y - x, \nabla f(x) \rangle\} \\ &= \frac{1}{2} \sup_{\substack{x,y,s \in D \\ \theta \in (0,1] \\ y = x + \theta(s-x)}} (s - x)^T \nabla^2 f(x)(s - x) \end{aligned} \tag{4.16}$$

According to Cauchy-Schwarz inequality [7], we can further get:

$$\begin{aligned} C_f &= \frac{1}{2} \sup_{\substack{x,s \in D \\ \theta \in (0,1] \\ z \in [x,y] \subseteq D}} (s - x)^T \nabla^2 f(x)(s - x) \\ &\leq ||s - x||_2 \nabla^2 f(x) ||s - x||_2 \\ &\leq ||s - x||_2^2 \nabla^2 f(x) \\ &\leq \frac{1}{2} diam(D)^2 \sup_{x \in D} \lambda_{max}(\nabla^2 f(x)) \end{aligned} \tag{4.17}$$

From the definition of convex hull Ω_{i_0}, we have that $diam(\Omega_{i_0}) = \sqrt{2}$. From the definition of $J(\alpha)$, we can calculate that $\nabla^2 f(\alpha) = X^T X$. Finally, we can get that $C_f \leq \lambda_{max}(X^T X)$.

Let the gradient $\nabla f(\alpha^{(k-1)}) = \frac{1}{a_{k-1}} \in (0, 1]$. According to the definitions of C_f and $g(x, \nabla f(x))$, we can

get the following formula.

$$\begin{aligned} f(\alpha^{(k)}) &= f(\alpha^{(k-1)} + \frac{1}{a_{k-1}}(s - \alpha^{(k-1)})) \\ &\leq f(\alpha^{(k-1)}) + \frac{1}{a_{k-1}}\nabla f(\alpha^{(k-1)})^T(\alpha^{(k)} - \alpha^{(k-1)}) + \frac{1}{a_{k-1}^2}C_f \\ &\leq f(\alpha^{(k-1)}) - \frac{1}{a_{k-1}}g(\alpha^{(k-1)}, \nabla f(\alpha^{(k-1)})) + \frac{1}{a_{k-1}^2}C_f \end{aligned} \tag{4.18}$$

Let $h(\alpha^{(k)}) := f(\alpha^{(k)}) - f(\alpha^*)$ for the primal error at $f(\alpha^{(k)})$. Based on the former formula 4.18, we can get:

$$\begin{aligned} h(\alpha^{(k)}) &\leq h(\alpha^{(k-1)} - \frac{1}{a_{k-1}}h(\alpha^{(k-1)}) + \frac{1}{a_{k-1}^2}C_f \\ a_{k-1}^2 h(\alpha^{k-2}) &\leq (a_{k-1}^2 - a_{k-1})h(\alpha^{(k-1)}) + C_f \end{aligned} \tag{4.19}$$

Ask $a_{k-1}^2 - a_{k-1} = a_{k-2}^2$, then function 4.19 can be rewritten as: $a_{k-1}^2 h(\alpha^{(k-2)}) \leq a_{k-2}^2 h(\alpha^{(k-1)}) + C_f$. Expanding this function, we can get:

$$\begin{aligned} a_{k-1}^2 h(\alpha^{(k)}) - a_{k-2}^2 h(\alpha^{(k-1)}) &\leq C_f \\ a_{k-2}^2 h(\alpha^{(k-1)}) - a_{k-3}^2 h(\alpha^{(k-2)}) &\leq C_f \\ \cdots & \\ \cdots & \\ a_1^2 h(\alpha^{(2)}) - a_0^2 h(\alpha^{(1)}) &\leq C_f \end{aligned} \tag{4.20}$$

Adding up all the above functions together, we have:

$$a_{k-1}^2 h(\alpha^{(k)}) - a_0^2 h(\alpha^{(1)}) \leq (k-1)C_f \tag{4.21}$$

As $a_0 = 1$, from formula 4.19, we can calculate: $h(\alpha^{(1)}) \leq C_f$, so:

$$h(\alpha^{(k)} \leq \frac{kC_f}{a_{k-1}^2} \tag{4.22}$$

As $a_k = \frac{1+\sqrt{4a_{k-1}^2+1}}{2} > 0.5 + a_{k-1}$, we have $a_k > 0.5(k+2)$. Therefore, we can get:

$$f(\alpha^{(k)}) - f(\alpha^*) = h(\alpha^{(k)}) \leq \frac{kC_f}{a_{k-1}^2} < \frac{4kC_f}{(k+1)^2} < \frac{4C_f}{k+2} < \frac{4\lambda_{max}(X^TX)}{k+2}, \tag{4.23}$$

where λ_{max} is the maximum eigenvalue of X^TX.

4.2.4 Optimizing over convex hull

In this study, the optimization domain Ω_{i_0} is given as the convex hull of a finite subset $\{e_i - e_{i_0} \in R^n | i \neq i_0\}$. In my algorithm, the optimizer consists of greedily selecting the "vertex" of the convex hull, which promises best improvement. It makes the linear optimization sub-problems as needed in my algorithm particularly easy to be solved.

Lemma 4.2 (Linear optimization over convex hull) Let $D = conv(V)$, where $V \in \mathbf{R}^n$, D is a convex hull, a compact domain. At some "vertex" v ($v \in V$), for any linear function $y \to c^Ty$, the linear function can attain its local minimum or maximum over D, while the local optimal are also global optimal, according to the proper of convex optimization problems [4].

Proof Omitted, interested readers can refer to the proof of Lemma 2.8 in paper [47].

5 Clustering and Visualization

Once similarity matrices are constructed, they can be used for clustering and visualization. In this study, based on the sparse cell-cell similarity matrices, I use four different clustering methods and one visualization method for analyzing scRNA-seq data.

5.1 Clustering

Clustering is an unsupervised method. It aims to divide a data set into different clusters according to a certain stand (i.e. distance between data points), so similar data points are placed in the same cluster and dissimilar data points are placed in different clusters [91]. In single cell data analysis, clustering plays a key role on identifying cell populations.

In this study, based on the sparse cell-cell similarity matrix constructed in section 4, I further apply four clustering methods including two spectral clustering methods, one hierarchical clustering method and one Louvain clustering method, to find an optimal appropriate clustering method.

5.1.1 Spectral clustering

Spectral clustering is an algorithm evolved from graph theory [70]. Its main idea is to treat all data objects as points in space, and these points can be connected by edges. Further distance between two points means smaller edge weight. By cutting the graph composed of points and edges, this algorithm tries to make the sum of edge weights between different sub-graphs is as small as possible, and the sum of edge weights within the sub-graphs is as large as possible. because of this, it is called minimum cut. Spectral clustering uses this idea to achieve the purpose of clustering. Figure 5.2 shows an example of minimum cut of a graph.

Spectral clustering combines a dimentionality reduction method, Laplacian eigenmap, with a clustering method, such as k-means clustering and Gaussian mixture model clustering, to cluster data points. There are three main steps of spectral clustering.

Firstly, Laplacian matrix L should be calculated using $L = D - S$, where S is the cell-cell sparse similarity matrix, calculated in section 4.1, D is a diagonal matrix, whose ith diagonal element is the sum of the ith column of S.

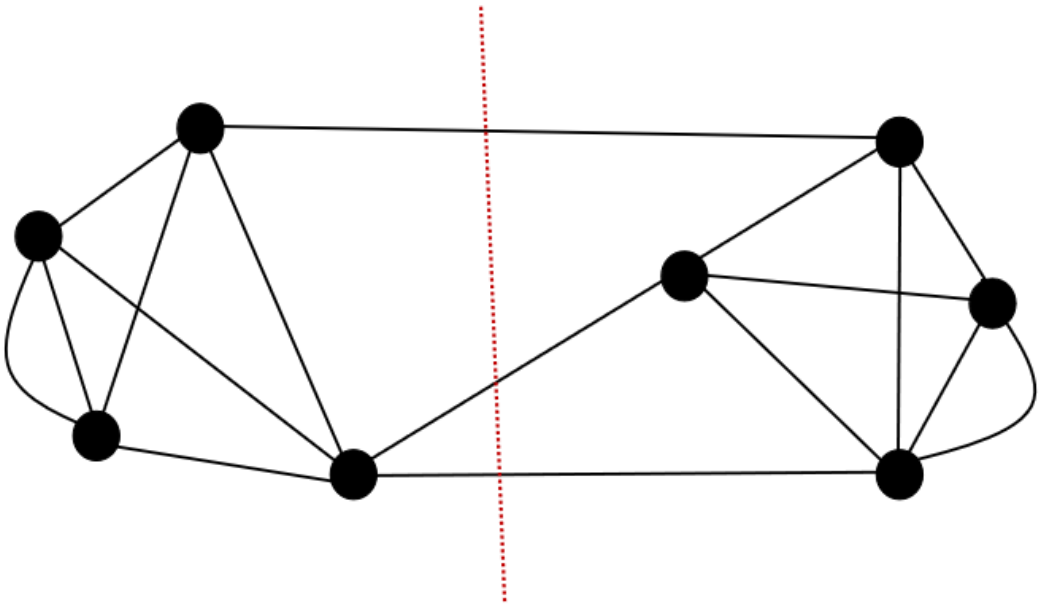

Figure 5.1: An example for minimum cut of an undirected graph, in which the weight of each edge is equal

The minimum cut is the smallest set of edges whose removal breaks the graph into two pieces.

Secondly, feature matrix $F \in \mathbf{R}^{n \times k}$ is constructed, which consists of k eigenvectors $(v_1, ..., v_k)$ corresponding to the k smallest nonzero eigenvalues $(0 \leqslant \lambda_1 \leqslant \lambda_2 ... \leqslant \lambda_k)$ of Laplacian matrix L. λ and v are calculated by the eigenproblem: $Lv = \lambda Dv$.

Thirdly, a clustering method is applied to the feature matrix F for clustering data points. In this study, I use k-means clustering and Gaussian mixture clustering respectively on F to group cells. K-means clustering is used to divide data points into k clusters, by repetitively assigning points to clusters with the nearst controids, and update the controids by averaging all the points in that cluster[116]. While Gaussian mixture model clustering [10] is a probabilistic clustering method, which assumes that all data points are generated by a mixture distribution composed of k mixed multivariate Gaussian distributions. In the section 7.1, I will compare the performance between these two methods in order to select the most appropriate method.

5.1.2 Hierarchical clustering

Hierarchical clustering [52] is a type of clustering algorithm that creates a clustering dendrogram by merging or splitting different clusters. Specifically, the approaches are called agglomerative approach (bottom-up) and divisive approach (top-down).

In this study, I use an agglomerative algorithm, which iterates two processes: 1) calculating the distance

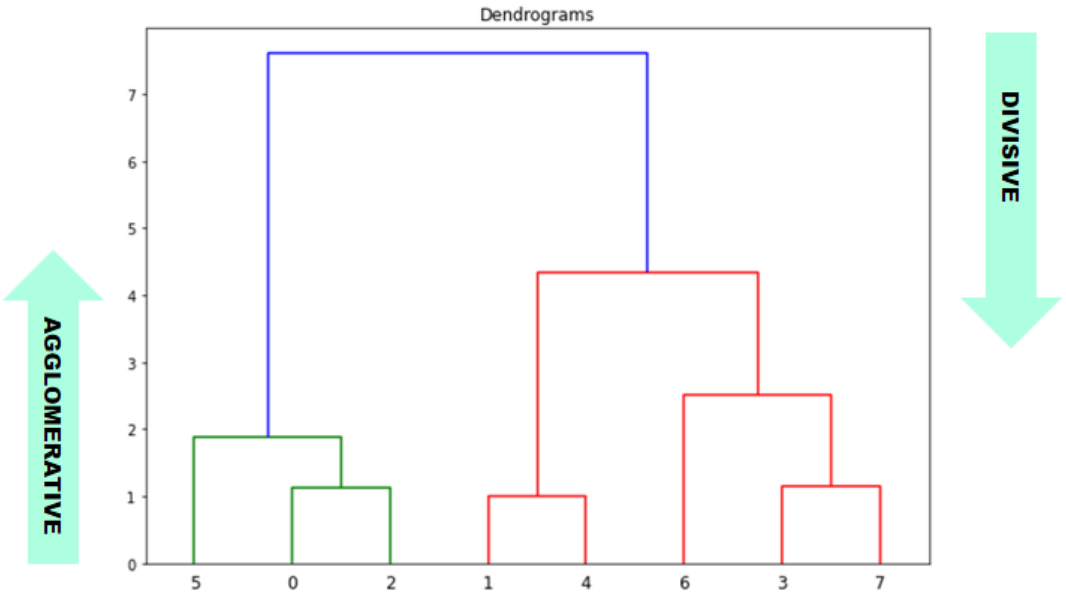

Figure 5.2: Agglomerative approach vs divisive approach of hierarchical clustering
From top to down, it is divisive approach, while from bottom to up, it is agglomerative approach.

between two clusters, and 2) combining the two most close clusters across all clusters, until all clusters are merged together. Note that in sparse cell-cell similarity matrix, the higher the value is, the similar the corresponding cells are. This is the opposite of distance matrix, so I transform similarity matrices into distance matrices using:

$$\begin{aligned} D_{ij} &= 1 - S_{ij} \\ s.t. D_{ii} &= 0, \end{aligned} \tag{5.1}$$

where D_{ij} is the value of the ith row and the jth column of distance matrix D, S_{ij} is the value of the ith row and the jth column of sparse cell-cell similarity matrix S.

Linkage methods work by calculating the distance between all data points. Then the closest pair of clusters are merged into a single cluster. According to the diverse criterion of linkage methods, generally, in agglomerative approach, there are four linkage methods including single-linkage, complete-linkage, average-linkage and Ward-linkage [83]. Specifically, for single-linkage, two clusters with the closest distance are merged. For complete linkage, two clusters with the farthest distance are merged. While, average-linkage merges clusters based on the average distances of data points in two clusters. Ward-linkage merges two clusters based on their error sum of square (ESS). The two clusters with the lowest ESS are merged. Ward-linkage is different with the other three strategies mentioned above, which are based on distance.

For Ward-linkage, the loss function of two clusters is calculated by Error Sum of Squares (ESS) [36]. ESS can be calculated as follows.

$$ESS = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{i\cdot})^2, \tag{5.2}$$

where y_{ij} is the jth data point in the ith cluster, k is the number of clusters, $\bar{y}_{i\cdot}$ is the mean of all data points of the ith cluster, and n_i is the number of data points in the ith cluster.

In this study, I choose Ward-linkage, because it can minimize the information loss of each merger to link clusters. Ward-linkage is less susceptible to noise and outliers compared with the other three strategies[36], as a result, it is more suitable to analyze scRNA-seq data, which contains a lot of technical and biological noise [12].

5.1.3 Louvain clustering

In order to explain Louvain clustering clearly, we need to know modularity first. Modularity is a measure of the structures of networks or graphs which measures the strength of partition of networks into modules. It measures the density of both the connections within modules and the connections between different modules. The modularity is defined as follows.

$$Q = \frac{1}{2m} \sum_{ij} [S_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j), \tag{5.3}$$

where S is the cell-cell similarity matrix in this study, and S_{ij} is the element of the ith row and the jth column of S; $m = \frac{1}{2} \sum_{ij} S_{ij}$, meaning it is the sum of all edge weights of the network; k_i and k_j are the sum of elements of ith and jth column of matrix S, respectively. Based on that, $\frac{k_i k_j}{2m}$ is the expected valued of the weight between node i and node j; δ is Kronecker delta function defined as $\delta(c_i c_j) = \begin{cases} 1 & if\ c_i = c_j \\ 0 & if\ c_i \neq c_j \end{cases}$, and c_i and c_j are the communities of data points.

The original formula for modularity is complicated, so I simplify it to make it easier to understand:

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{ij} [S_{i,j} - \frac{k_i k_j}{2m}] \delta(c_i, c_j) \\ &= \frac{1}{2m} [\sum_{ij} S_{i,j} - \frac{\sum_i k_i \sum_j k_j}{2m}] \delta(c_i c_j) \\ &= \frac{1}{2m} \sum_{i=1}^{k} [\sum\nolimits_{in} - \frac{(\sum_{tot})^2}{2m}], \end{aligned} \tag{5.4}$$

where $\sum_{in}$ is the sum of weights of all edges within the community; $\sum_{tot}$ is the sum of weights of edges between communities; k is the number of communities of the network. Obviously, the larger the value of modularity Q is, the better the clustering quality is. Louvain clustering for community detection, also called graph partition, is an algorithm based on multilevel modularity optimization [9]. It is becoming more and more popular because of its great efficiency, accuracy and the ability to discover high modularity community structures [37]. The optimization goal of Louvain clustering is to maximize the modularity of the entire data, which will be explained further in the next few paragraphs.

In order to effectively maximize the value of modularity Q, Louvain clustering has two-phase iterations.

At the beginning, each node is regarded as an independent community, and the weights of connected edge between communities are 0.

Phase1 For each node i, through removing node i from its own community j out and adding it into the community of every neighbor of node i, the modularity benefit ΔQ for the community j can be calculated as:

$$\Delta Q = [\frac{\sum_{in}+k_{i,in}}{2m} - (\frac{\sum_{tot}+k_i}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k_i}{2m})^2], \tag{5.5}$$

where $\sum_{in}$ is the sum of weights of all edges within the community that node i is moving to (denoted as community c); $k_{i,in}$ is the sum of weights of edges between node i with the nodes in community c; $\sum_{tot}$ is the sum of weights of edges between community c and the other communities in the graph; k_i is the weighted degree of node i, which can be calculated as the sum of elements of ith column of matrix S; m is the sum of all edge weights of the network, which can be calculated as: $m = \frac{1}{2}\sum_{i,j} S_{ij}$, and S_{ij} is the element of the ith row and the jth column of cell-cell similarity matrix S. Node i is placed to the community which results in the largest modularity benefit ΔQ. The process is repeated until ΔQ equals zero.

Phase2 After phase 1, every community is regarded as merging into one big node (see the middle subgraph of Figure 5.3). This node is a self-loop node, and its weight is the sum of the edge weights of all nodes within the original community constructed in phase 1. The edge weights between these big nodes are the sum of the edge weights of the corresponding communities constructed in phase 1. Once the new network is constructed, phase 2 is over. Figure 5.3 shows an example of these two phases.

Phase 1 and Phase 2 are repeated until there is no increase of the value of modularity Q or reaching the number of iterations set by users. In this study, the iteration is controlled by the first case.

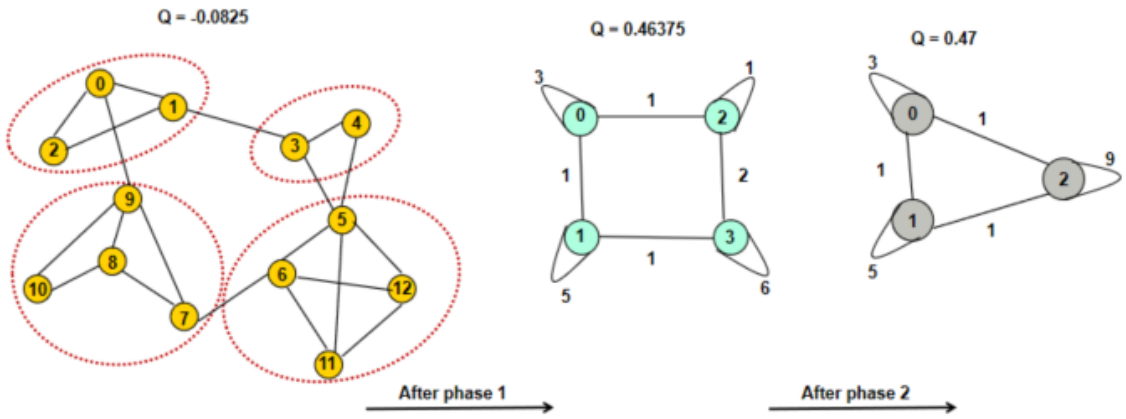

Figure 5.3: An example of louvain clustering for modularity optimization

Q is the value of modularity of each sub-graph. The whole process represents two-phase iterations of louvain clustering

5.2 Visualization with t-SNE

Visualization is essential of biological interpretation for single-cell RNA sequencing data analysis. [130]. Generally, scRNA data visualization is in two-dimensional space, where each cell is given a pair of X-Y coordinates to define its position [15]. Through highlighting cell metadata according to the information of cells in given experiments, such as batch processing, donors, etc. or the differential expression of specific genes of different cell types, visualization can be used to evaluate the cell types obtained [15].

In general, the methods for visualization can be divided into two categories: linear transformations (i.e. principal component analysis (PCA)) and nonlinear transformations (i.e. t-distributed stochastic neighbor embedding (t-SNE)). However, linear transformation methods cannot faithfully capture the nonlinear relationships of features, which are ubiquitous in scRNA-seq data sets [2]. And in visualizing scRNA-seq data, t-SNE performs better than many nonlinear transformation methods (i.e. Isometric Mapping (Isomap), Locally Linear Embedding (LLE)). Hence, in this study, I choose to combine sparse cell-cell similarity matrices with t-SNE to visualize scRNA-seq data.

T-distributed stochastic neighbor embedding (t-SNE) is an improved statistical method for visualization based on stochastic neighbor embedding (SNE). The main idea of these two methods is that the more the data points are closer in high-dimensional space, the more closer the distances mapped to low-dimensional spaces are. The conventional methods use Euclidean distance between data points to express the similarity, while SNE and t-SNE converts this distance relationship into a conditional probability to express the similarity. Compared to SNE, t-SNE have two improvements. Firstly, t-SNE uses t-distribution to replace the original Gaussian distribution in the low-dimensional space, while in the high-dimensional sparse, t-SNE still use

Gaussian distribution like SNE. Secondly, t-SNE turn SNE into symmetrical SNE, which means, in low and high dimensional spaces, the conditional probabilities of every two data points are equal. These two improvements make t-SNE a better visualization method. Compared with SNE, the distances of data points within clusters become closer, and the distances of different clusters become farther.

The t-SNE algorithm has two main stages to reduce data from high dimension to low dimension. Firstly, t-SNE constructs a probability distribution over pairs of high-dimensional data points in such a way that similar data points are assigned a higher probability while dissimilar points are assigned a lower probability. Secondly, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map. While the original t-SNE algorithm uses the Euclidean distance between data points to construct similarity metric, this can be changed as appropriate. The details are as follows.

Given the data points $x_1, x_2, \cdots, x_n$ in high-dimensional space, the joint probability p_{ij} of point x_i and point x_j $(i \neq j)$ can be defined as:

$$\begin{aligned} p_{i|j} =& \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_j||^2/2\sigma_i^2)} \\ & s.t. \quad p_{i|i} = 0, \end{aligned} \tag{5.6}$$

where σ_i^2 is the variance of the Gaussian distribution centered on data point x_i.

In order to keep the symmetry of the joint probability between i and j, $p_{ij} = p_{ji} = \frac{1}{2n}(p_{j|i} + p_{i|j})$, where n is the number of data points.

Given the data points $y_1, y_2, \cdots, y_n$ in low-dimensional space, the joint probability $q_{j|i}$ of point y_i and point y_j $(i \neq j)$ can be defined as:

$$\begin{aligned} q_{i|j} =& \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \\ & s.t. \quad q_{i|i} = 0 \end{aligned} \tag{5.7}$$

Similarly, to keep the symmetry of the joint probability between data points i and j, $q_{ij} = q_{ji} = \frac{1}{2}(q_{j|i} + q_{i|j})$.

Then, KL divergence is used to construct the following cost function between joint probability distributions

P and Q.

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{5.8}$$

Furthermore, gradient descent algorithm [92], an iterative optimization algorithm which is used to find the local minimum of a function, is used to minimize KL divergence with respect to the points $y_1, y_2, \cdots, y_n$ in low-dimensional space.

In this study, I combine my method, LSSM with t-SNE to visualize scRNA-seq data sets in two-dimensional space. During the process, S_{ij} $(0 \leq S_{ij} \leq 1, \sum_i S_{ij} = 1)$, the ith row and the jth column element of sparse cell-cell similarity matrix S, is used to express the joint probability of point i and point j in low-dimensional spaces. Hence, in low-dimensional space, the joint probability p_{ij} is calculated as: $p_{ij} = \frac{S_{ij}}{\sum_{k \neq j} S_{kj}}$. The other parts of algorithms remain the same as t-SNE.

6 Evaluation Metrics

Unsupervised clustering evaluation approaches play a key role on evaluating the success of clustering algorithms. Generally, they can be divided into two categories: internal cluster evaluation and external cluster evaluation [71].

Internal cluster evaluation assesses a clustering result based on the clustered data itself. Hence, it does not need to know the ground truth of the data in advance. This evaluation is based on the following two criteria: 1) compactness, which measures how similar (close) the data points are within a cluster, 2) separation, which measures how different (well-separated) of a cluster from the other clusters [71]. However, one shortcoming of internal cluster evaluation is that good results over internal evaluation measures do not necessarily lead to high performance in the application[78]. Besides, these internal evaluation methods may be biased to the algorithms which are based on the same model. For example, both k-means clustering and Davies-Bouldin index [28] (an internal evaluation) are distance-based methods. Therefore, using Davies-Bouldin index to evaluate the results, which are generated by k-means clustering, may be biased towards k-means clustering. Hence, this internal cluster evaluation method may not give good results for the clustering methods which are not distance-based.

On the other hand, external cluster evaluation needs the external data information such as class labels. Ideally, the ground truth is decided by human with great inter-judge agreements. External evaluation methods measure the similarity between cluster results and predetermined class true labels.

Usually, for single cell gene expression data, the cells are labeled by researchers in advance, so external cluster evaluation methods are often used to evaluate clustering results. There are a number of external cluster evaluation metrics can be used here, such as Purity [114], normalized mutual information (NMI) [55], adjusted rand index (ARI) [22] and F-measure [23]. In order to assess clustering results more comprehensively, I choose three external cluster evaluation methods, Purity, NMI and ARI, to assess the clustering results.

Let $T = (t_1, t_2, \cdots, t_k)$ represent the set of ground truth labels, where $t_i \subseteq T$ is the set of data points in class t_i, $i \in [1, 2, \cdots, k]$. $C = (c_1, c_2, \cdots, c_v)$ denotes the set of predicted clustering labels, where $c_j \subseteq C$ is the set of data points in cluster c_j, $j \in [1, 2, \cdots, v]$.

6.1 Purity

Purity is a simple method of clustering evaluation. It is the percent of the total number of data points that were classified correctly. The formal expression is as follows.

$$Purity(T, C) = \frac{1}{N} \sum_{i=1}^{k} \underset{j}{max} |t_i \cap c_j|, \tag{6.1}$$

where $|t_i \cap c_j|$ represents the number of data points in both t_i and c_j.

The advantage of using the Purity method is that it is easy to calculate. The value of Purity is between 0 and 1. When clustering is completely wrong, it is 0. On the contrary, when clustering is totally correct, the value is 1. However, the shortcoming of purity method is also obvious. It cannot give a correct evaluation for the degenerate clustering method. Assuming that if the clustering algorithm clusters all data points into a single category, the value of purity is 1. This is obviously not the desired results.

6.2 NMI

Normalized mutual information (NMI) [55] can be explained by information theory. It is normalized, so we can use NMI values to compare with different clustering methods, even with different predicted cluster numbers. Using entropy of the true cluster labels and the predicted cluster result, NMI is defined as follows.

$$NMI(T, C) = \frac{2 \times I(T; C)}{[H(T) + H(C)]}, \tag{6.2}$$

where $I(T; C)$ is the mutual information between group T (the ground truth labels) and group C (the predicted clustering labels). $H(\cdot)$ represents entropy.

I is mutual information function, which measures the level of these two splittings (T and C) agreeing to each other. It is defined as:

$$\begin{aligned} I(T; C) &= \sum_{i=1}^{k} \sum_{j=1}^{v} P(t_i \cap c_j) log \frac{P(t_i \cap c_j)}{P(t_i)P(c_j)} \\ &= \sum_{i=1}^{k} \sum_{j=1}^{v} \frac{|t_i \cap c_j|}{N} log \frac{N|t_i \cap c_j|}{|t_i||c_j|} \\ &= \sum_{i=1}^{k} \sum_{j=1}^{v} \frac{m_{ij}}{N} log \frac{N m_{ij}}{a_i b_j} \end{aligned} \tag{6.3}$$

where $P(t_i)$, $P(c_j)$ represent the marginal probabilities of a data point belonging to t_i, c_j respectively, $|t_i \cap c_j| = m_{ij}$ represents the number of data points in both t_i and c_j, $|t_i| = a_i$ and $|c_j| = b_j$ denote the number of data points in t_i and c_j respectively, $P(t_i \cap c_j)$ is the joint probability of a data point being in the intersection of t_i and c_j.

$H(\cdot)$ is entropy, for example, $H(T)$ is defined as:

$$\begin{aligned} H(T) &= -\sum_{i=1}^{k} P(t_i) log P(t_i) \\ &= -\sum_{i=1}^{k} \frac{|t_i|}{N} log \frac{|t_i|}{N} \\ &= -\sum_{i=1}^{k} \frac{a_i}{N} log \frac{a_i}{N} \end{aligned} \tag{6.4}$$

The range of NMI is from 0 to 1. The value is closer to 1, the better the clustering quality is.

6.3 ARI

In order to explain adjusted rand index (ARI) clearly, rand index (RI) should be introduced first. RI, based on information theory interpretation, is considered as a series of decisions. It is defined as:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{6.5}$$

Between ground truth groups and predicted label groups, TP, TN, FP, FN are explained as follows:

• True positive (TP) decision is the number of pairs of similar data points, which are assigned to the same cluster between T and C.

• True negative (TN) decision is the number of pairs of dissimilar data points, which are assigned to two different clusters between T and C.

• False positive (FP) decision is the number of pairs of dissimilar data points, which are assigned to the same cluster between T and C.

• False negative (FN) decision is the number of pairs of similar data points, which are assigned to different clusters between T and C.

Intuitively, $TP + TN$ can be regarded as the number of agreements of two groups, while $FP + FN$ can be regarded as the number of disagreements of two groups.

RI has the following disadvantages: 1) RI is not being punitive enough, 2) RI cannot compare to random sets. Because of these, adjusted rand index (ARI), the revised Rand index, is proposed. ARI increases penalty and considers the randomness. It is defined as follows:

$$ARI = \frac{RI - E(RI)}{Max(RI) - E(RI)} \tag{6.6}$$

More specifically:

$$\begin{aligned} ARI(T,C) &= \frac{\sum_{i=1}^{k}\sum_{j=1}^{v}\binom{m_{ij}}{2}/\binom{N}{2} - \left[\sum_{i=1}^{k}\binom{a_i}{2}\sum_{j=1}^{v}\binom{b_j}{2}\right]/\binom{N}{2}}{\frac{1}{2}\left[\sum_{i=1}^{k}\binom{a_i}{2}/\binom{N}{2} + \sum_{j=1}^{v}\binom{b_j}{2}/\binom{N}{2}\right] - \left[\sum_{i=1}^{k}\binom{a_i}{2}\sum_{j=1}^{v}\binom{b_j}{2}\right]/\binom{N}{2}/\binom{N}{2}} \\ &= \frac{\sum_{i=1}^{k}\sum_{j=1}^{v}\binom{m_{ij}}{2} - \left[\sum_{i=1}^{k}\binom{a_i}{2}\sum_{j=1}^{v}\binom{b_j}{2}\right]/\binom{N}{2}}{\frac{1}{2}\left[\sum_{i=1}^{k}\binom{a_i}{2} + \sum_{j=1}^{v}\binom{b_j}{2}\right] - \left[\sum_{i=1}^{k}\binom{a_i}{2}\sum_{j=1}^{v}\binom{b_j}{2}\right]/\binom{N}{2}} \end{aligned} \tag{6.7}$$

where m_{ij} represents the number of data points in both t_i and c_j, a_i and b_j denote the number of data points in t_i and c_j respectively. N is the total number of cells in the data, and $\binom{N}{2} = \frac{N(N-1)}{2}$.

$RI = \sum_{i=1}^{k}\sum_{j=1}^{v}\binom{m_{ij}}{2}/\binom{N}{2}$ is the number of pairs of data points which are in the same ground truth group in T and in the same predicted label group in C.

$Max(RI) = \frac{1}{2}\left[\sum_{i=1}^{k}\binom{a_i}{2} + \sum_{j=1}^{v}\binom{b_j}{2}\right]/\binom{N}{2}$ is calculated assuming that the clustering is totally correct.

In order to explain $E(RI)$ more clearly, I need to introduce the contingency table (Table 6.1), which can intuitively display the relationship between two variables of sets T and C (see Table 6.1)

Table 6.1: The contingency table of ground truth labels set T and predicted labels set C

T \ C	c_1	c_2	...	c_v	sums
t_1	m_{11}	m_{12}	...	m_{1v}	a_1
t_2	m_{21}	m_{22}	...	m_{2v}	a_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_k	m_{k1}	m_{k2}	...	m_{kv}	a_k
sums	b_1	b_2	...	b_v	

From Table 6.1 we can know that there are $\binom{N}{2} = \frac{N(N-1)}{2}$ pairs of data points between sets C and T in total. Therefore, the expected value of RI is: $E(RI) : \left[\sum_{i=1}^{k}\binom{a_i}{2}\sum_{j=1}^{v}\binom{b_j}{2}\right]/\binom{N}{2}$.

The value of ARI is between 0 and 1. The value is 1 if only the predicted results are completely identical to the true cluster labels. While the value is close to 0 for a random partition [126], which is not the case with RI.

7 Results and Discussion

In this part, the results of my method on scRNA-seq data sets and the results of some state-of arts methods as comparison are shown. Furthermore, the visualization performance of t-SNE with the similarity matrices constructed by my method also is shown.

7.1 Clustering

To evaluate my method's performance, I apply my method, Learning Sparse Similarity Matrices (LSSM), which constructs a sparse cell-cell similarity matrix, on eight scRNA-seq data sets. Based on the similarity matrix, I further use four clustering methods to identify cell populations: 1) spectral clustering based on Laplacian eigenmap and k-means clustering, denoted as LSSM-EK, 2) spectral clustering based on Laplacian eigenmap and Gassian mixture model clustering, denoted as LSSM-GK, 3) hierarchical clustering, denoted as LSSM-H, and 4) Louvain clustering, denoted as LSSM-L. I use three common clustering evaluation metrics: Purity, normalized mutual information (NMI) and adjusted rand index (ARI) to evaluate the clustering methods.

Using the metrics and data sets described above, I compare LSSM-EK, LSSM-EG, LSSM-H and LSSM-L with five state-of-the-art methods: NMF [101], SIMLR [117], MPSSC [85], DropClust [103] and SC3 [60]. In these compared methods, NMF decomposes a large target matrix into two smaller non-negative matrices (a basis matrix and a coefficient matrix), and a dimension of the latent space represents a particular cell type in one of composed matrices (coefficient matrix); MPSSC and SIMLR focus on considering multiple different resolution kernel function for constructing cell-cell similarity matrices; DropClust uses locally sensitive hashing (LSH), a logarithmic-time algorithm, for searching approximate k nearest neighbors (KNN) of scRNA data, and Louvain clustering is employed for this KNN network to cluster cells. SC3 is a popular tool for single-cell researchers, and its clustering accuracy is improved by integrating two consensus clustering solutions (k-means clustering and hierarchical clustering). More details are show in Table 7.1.

For the input parameters of the competing methods, the number of clusters are set as the number of the pre-annotated cell types (except for DropClust which does not need this parameters as input). Other parameters are set to the default values recommended by their original papers or downloadable programs.

Table 7.1: Summary of the compared studies

Compared methods	Programming language	Download source
NMF	Python	https://github.com/ccshao/nimfa
SIMLR	MATLAB/R	https://github.com/BatzoglouLabSU/SIMLR
MPSSC	MATLAB	https://github.com/ishspsy/project/tree/master/MPSSC
DropClust	R	https://github.com/debsin/dropClust
SC3	R	https://github.com/hemberg-lab/SC3

On the other hand, LSSM-EK, LSSM-EG and LSSM-H, the number of the pre-annotated cell types is also needed as an input. However, LSSM-L algorithm does not need this input.

Figure 7.1, Figure 7.2 and Figure 7.3 depict the values of Purity, NMI and ARI of all methods with eight scRNA-seq data sets. I compare four methods (LSSM-EK, LSSM-EG, LSSM-H and LSSM-L) with five other methods respectively. In order to determine the best clustering method based on LSSM among these four clustering methods, I further compare the clustering results of LSSM-EK, LSSM-EG, LSSM-H and LSSM-L.

Firstly, I compare the clustering results of the five state-of-the-art methods (NMF, SIMLR, MPSSC, DropClust, SC3) with LSSM-EK, LSSM-EG, LSSM-H and LSSM-L separately.

1. Comparing LSSM-EK with NMF, SIMLR, MPSSC, DropClust, SC3 based on Purity, NMI and ARI.

From Figure 7.1, Figure 7.2 and Figure 7.3, it can be seen that LSSM-EK outperforms most compared methods over eight data sets.

- For Purity, LSSM-EK is ranked first or in joint first place on the remaining data sets except for **Deng** data set, **Zheng11uneq** data set and **Zheng8eq** data set.
- For NMI, LSSM-EK achieves the best or tied first on six data sets and places second on the other two data sets: **Xin** data set and **Zheng8eq** data set.
- For ARI, LSSM-EK ranks first or tied first on seven data sets and ranks second on **Zheng11uneq** data set.

2. Comparing LSSM-EG with NMF, SIMLR, MPSSC, DropClust, SC3 based on Purity, NMI and ARI.

- For Purity, the clustering quality of LSSM-EG is best on the remaining data sets except for **Deng** data set, **Zheng11uneq** data set and **Zheng8eq** data set.
- For NMI, LSSM-EG is the first or tied for first place on six data sets apart from **Xin** data set and **Zheng8eq** data set.
- For ARI, LSSM-EG ranks first or ties first on six of the eight data sets, and ranks second on the largest two data sets: **Zheng11uneq** data set and **Zheng8eq** data set.

Hence, my method LSSM combining with the spectral clustering (Laplacian eigenmap + k-means clustering or Laplacian eigenmap + Gassian mixture model clustering), works well in scRNA-seq data analysis. Considering all the values of Purity, NMI and ARI of the eight data sets, LSSM-EK outperforms LSSM-EG.

3. Comparing LSSM-H with NMF, SIMLR, MPSSC, DropClust, SC3 based on Purity, NMI and ARI.

- For Purity, LSSM-H surpasses these five methods on four data sets: **Yan** data set, **Kumar** data set, **Koh** data set and **Tian** data set.
- For both NMI and ARI, LSSM-H performs best on **Yan** data set, **Deng** data set, **Kumar** data set and **Tian** data set.

Hence, the clustering qualities of LSSM-H are not good as LSSM-EK and LSSM-EG based on the results evaluated on the eight scRNA-seq data sets.

4. Comparing LSSM-L with NMF, SIMLR, MPSSC, DropClust, SC3 based on Purity, NMI and ARI.

- For Purity, LSSM-L surpasses the five methods on **Yan** data set, **Deng** data set and **Zheng11uneq** data set, while on **Kumar** data set, LSSM-L ties first place with SC3 and SIMLR.
- For NMI and ARI, LSSM-L is ranked first or tied for first place on five data sets (**Yan** , **Deng** , **Kumar** , **Koh** and **Zheng11uneq** data sets) and four data sets (**Yan** , **Kumar** , **Koh** and **Zhneg11uneq** data sets) respectively.

However, it is important to note that LSSM-L algorithm does not need the number of clusters as an input, which is the same as DropClust. Comparing these two methods, for Purity, LSSM-L surpasses DropClust on six data sets, apart from **Xin** data set and **Zheng8eq** data set. For NMI, LSSM-L performs better than DropClust except for **Xin** data set and **Deng** data set. And for ARI, except for **Xin** data set, LSSM-L outperforms DropClust. Hence, if the number of cell groups is unknown, Louvain clustering can be chosen to identify cell types with LSSM.

I would like to point out that these eight data sets varies in the data sizes, experiment apparatus, cell types, collected tissues/species, etc. This indicates that the reliable results achieved by LSSM methods are not affected by the scale and biological conditions of scRNA-seq data sets.

Additionally, I compare and analyze the clustering results of LSSM-EK, LSSM-EG, LSSM-H and LSSM-L. From Figure 7.1, Figure 7.2 and Figure 7.3. Given all the eight data sets, it can be seen that LSSM-EK is superior when compared to the other three methods. LSSM-EK gets the best or tied for first place over 4 data sets, 7 data sets and 6 data sets on Purity, NMI and ARI respectively. And LSSM-EG is tied for first with LSSM-EK or ranked right behind LSSM-EK over these 8 scRNA-seq data sets. Hence, only considering

the performance of clustering results, LSSM combined with spectral clustering methods (that are LSSM-EK and LSSM-GK) performs better, and LSSM-EK performs better than LSSM-EG. I can then conclude that, if the number of clusters is given, we can choose the spectral clustering (Laplacian eigenmap + k-means clustering) to achieve optimal performance on scRNA-seq data analysis.

Furthermore, note that in the three figures (Figure 7.1, Figure 7.2 and Figure 7.3), MPSSC has no results for **Zheng8eq** data set. This is due to the fact that MPSSC needs a lot of computational resource to deal with large scale data sets. In my case, it cannot process **Zheng8eq** data set on the computing cluster (Plato HPC cluster in University of Saskatchewan) with 32 CPUs, 1600 GB of memory. Actually I have to utilize such a computing cluster to run some methods such as MPSSC and SIMLR on the large scale data sets such as **Zheng11uneq** and **Zheng8eq**. On contrary, I can use a personal laptop, with 1 CPU, 8 GB of memory, to run LSSM-EK, LSSM-EG, LSSM-H and LSSM-L, even on large scale data sets mentioned above. So LSSM-EK, LSSM-EG, LSSM-H and LSSM-L have a highly computational efficiency, especially compared with some of the state-of-art methods such as MPSSC and SIMLR.

Besides, from the clustering results, we note that NMF performs much worse on the two largest scale data sets (**Zheng11uneq** and **Zheng8eq**). Actually NMF originally aims at finding a low-rank representation of a matrix with non-negative elements. As a competing method in this study, NMF can improve the estimation of the selected low-rank values. For a large scale data set, however, it is easy to lose information if the selected rank is improper. Therefore, the clustering performance of NMF may not be good on large scale data sets.

7.2 Visualization

Visualization is essential to provide biological explanation of cell populations of single-cell RNA sequencing data. LSSM can also contribute in this area. As detailed in section 5.2, I combine t-distributed stochastic neighbor embedding (t-SNE) with LSSM to show the visualization of data points in 2-dimensional space from high-dimensional space. Specifically, I calculate the joint probability in low-dimensional space based the cell-cell sparse similarity matrices, computed by LSSM, and the other parts are the same with t-SNE.

Figure 7.4 shows the visualization of four scRNA-seq data sets with t-SNE based on the similarity matrix computed by LSSM. For these four data sets, it is easy to observe that, in general, the points in the same cell types are close, while the points in the different cell types are clearly distinguished. Specifically, in **Kumar** data set, the cells are clustered into 3 mouse pluripotentstem cell types: TwoiLI, Dgcr8K and FBSLIF. In **Koh** data set, it has 9 subpopulations including H7hESCs and 8 kinds of H7-derived downstream early mesodermprogenitors, which are also clearly separated. And in **Tian** data set, the cells of 3 human lung adenocar-cinoma cell lines: HCC827, H1975 and H2228 are divided well too. The same thing can be said to

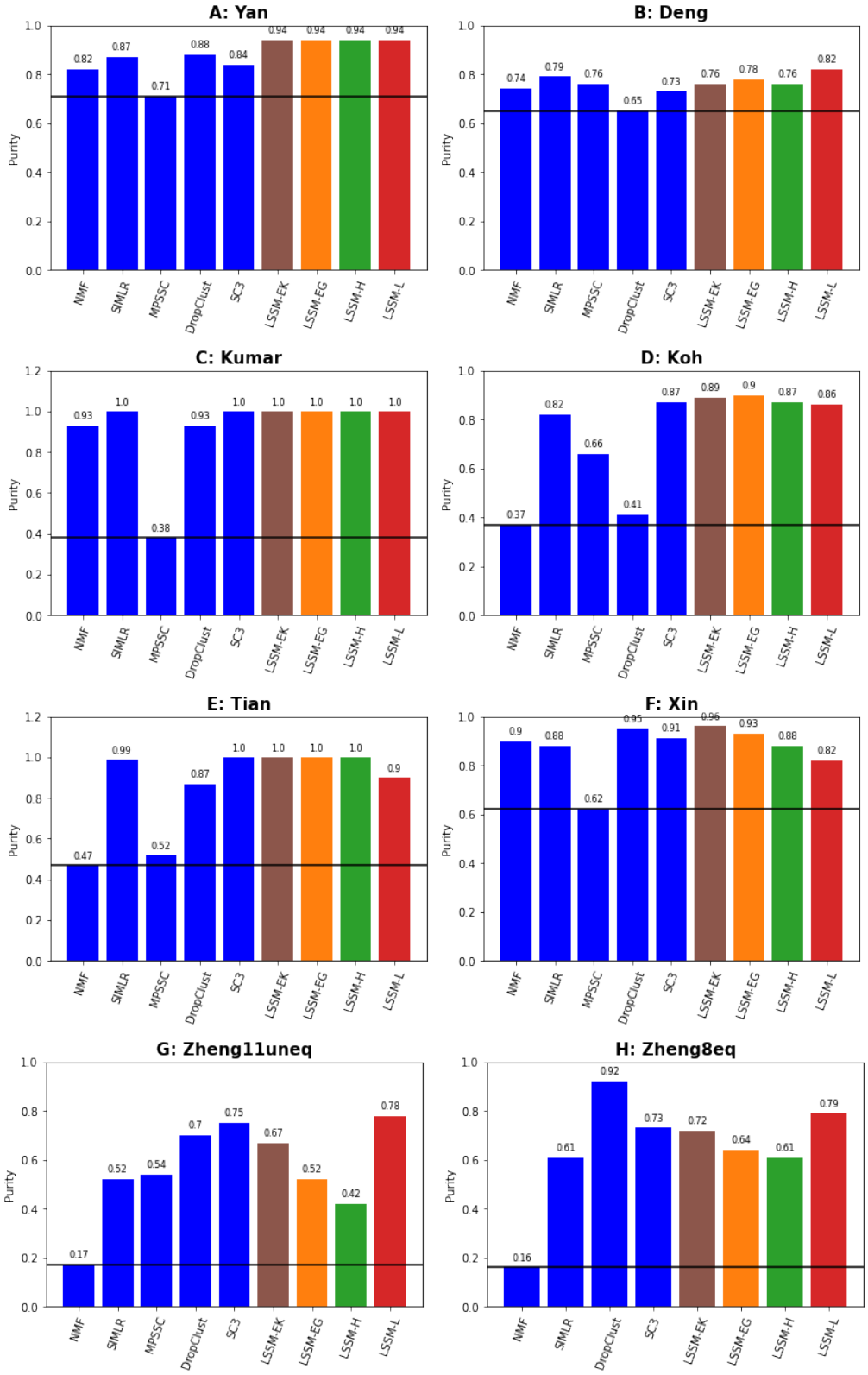

Figure 7.1: Purity values of nine methods with eight scRNA-seq data sets

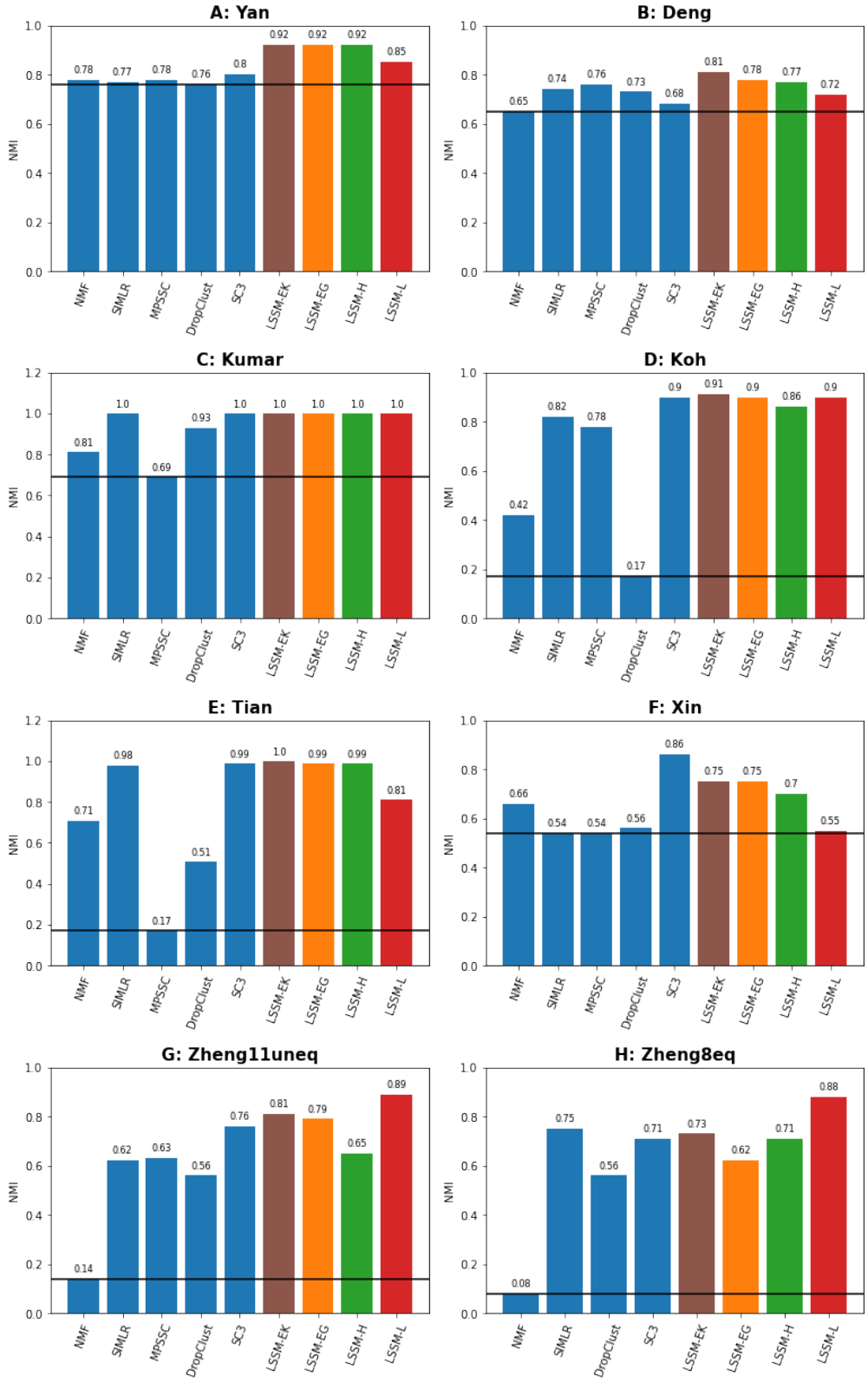

Figure 7.2: NMI values of nine methods with eight scRNA-seq data sets

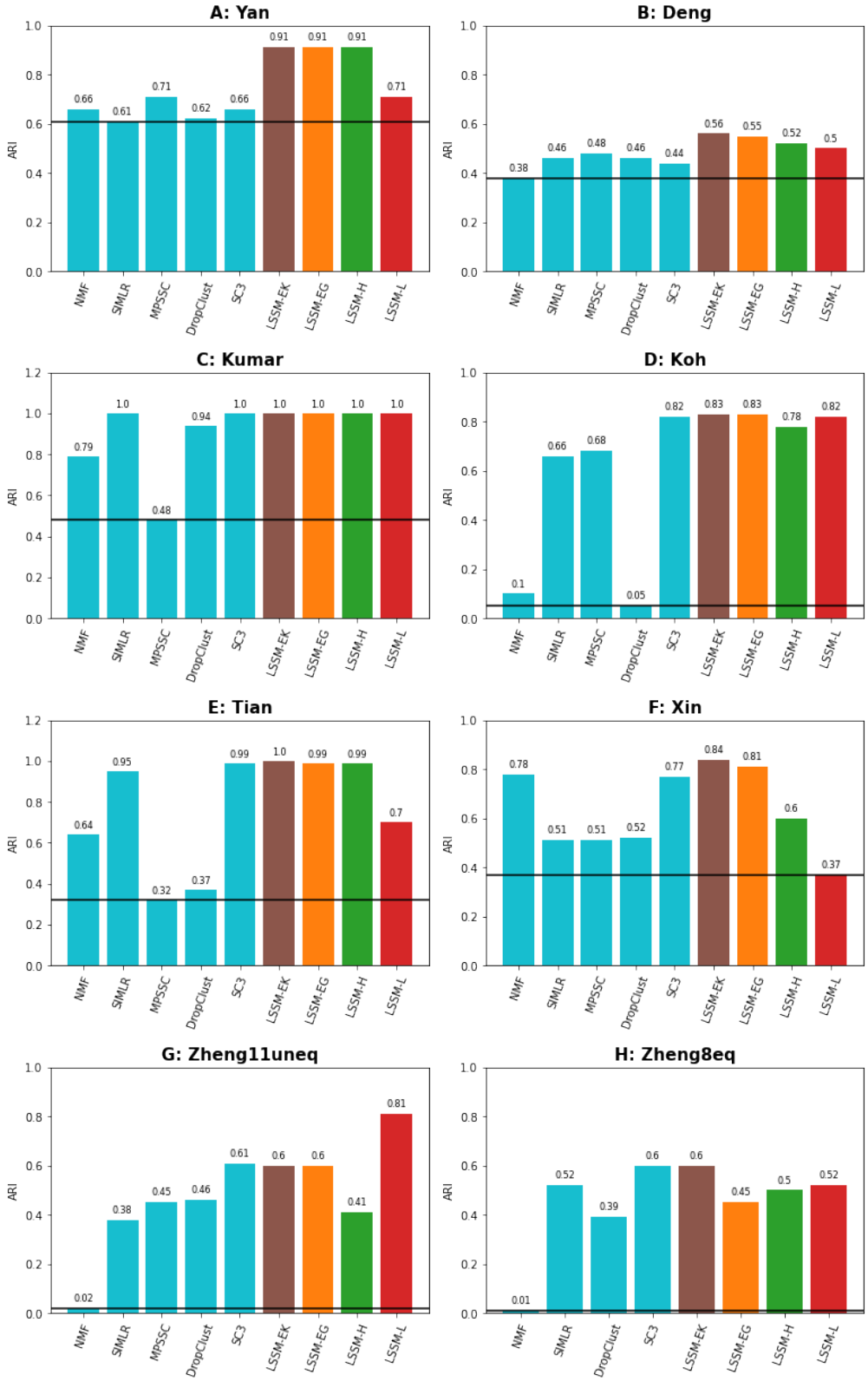

Figure 7.3: ARI values of nine methods with eight scRNA-seq data sets

Xin data set, which contains four human pancreatic islet cells including α cells, β cells, δ cells and PP cells.

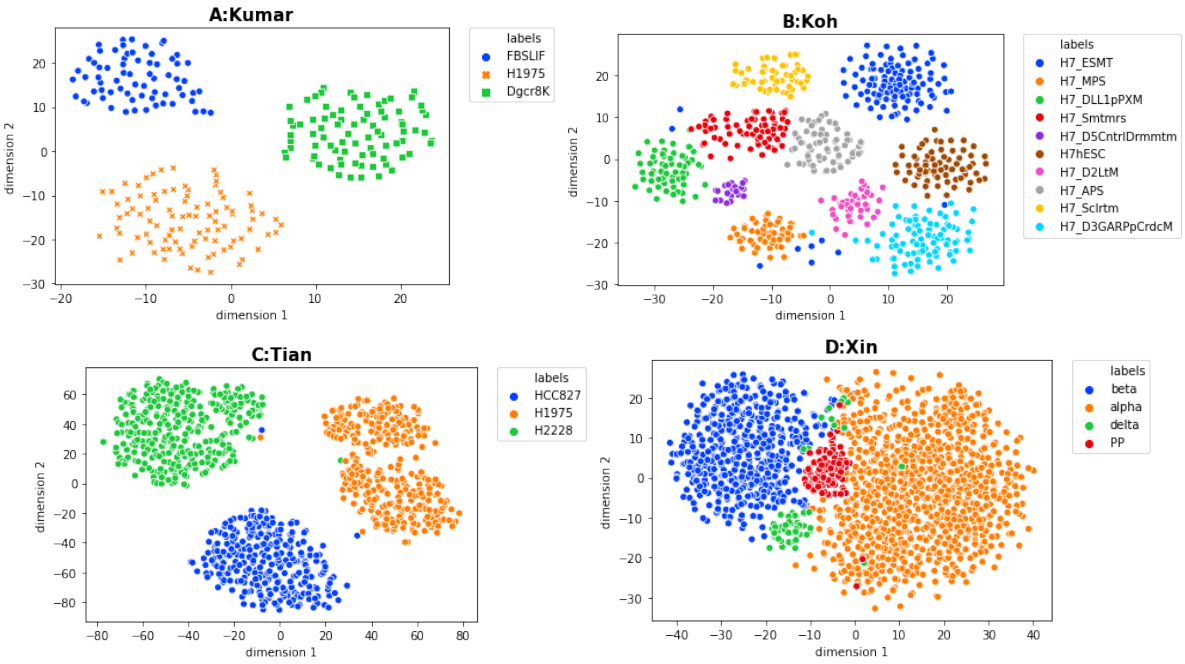

Figure 7.4: Visualization of four scRNA-seq data sets combining LSSM and t-SNE for (A) Kumar data set, (B) Koh data set, (C) Tian data set and (D) Xin data set. Different colors represent different cell sub-types. Labels represent diverse cell types.

8 Summary and Future Work

8.1 Summary

Single cell RNA sequencing (scRNA-seq) provides an unparalleled resolution to explore cellular heterogeneity. It determines the differential gene expression of individual cells by measuring the amount of RNA molecules. With the development of scRNA-seq technologies, a huge number of data sets have been generated. Nowadays, identifying cell populations through unsupervised clustering methods based on constructing similarity matrices between cells has become one of the most common applications of scRNA-seq data. However, there are many unique challenges in analyzing scRNA-seq because the data contains a lot of technical and biological noises. As a result, the performances of many scRNA-seq data clustering algorithms are not as good as expected.

In this study, I propose a novel method, LSSM, to construct sparse cell-cell similarity matrices for scRNA-seq data, and further apply several clustering methods to identify cell populations based on the similarity matrices.

Firstly, I construct a convex optimization objective function to calculate sparse cell-cell similarity matrices. According to sparse subspace theory, I assume that each cell can be represented by the other cells from the same subgroup with a linear combination, and the similarity matrices are composed by these linear combinations. In the cell-cell similarity matrices, the values are nonzero if the cells are in the same type, while the values are zero if the situation is opposite.

Secondly, I design an effective and convergent algorithm to compute the similarity matrices. Using column-wise learning, I decompose the large optimization problem into n easier-to-solve optimization problems. Besides, in order to guarantee the sparsity of each column of the similarity matrix, I use a greedy algorithm to add only one non-zero value to the corresponding column at each iteration. Hence, the number of sparsity of each column of the matrix can be controlled by setting the number of iteration and convergence parameters.

Thirdly, for purpose of selecting an appropriate clustering method, I apply two spectral clustering methods (Laplacian eigenmap + k-means clustering and Laplacian eigenmap + Gassian mixture model clustering), hierarchical clustering method and Louvain clustering method separately on the sparse cell-cell similarity matrices with multiple data sets. The clustering results show if the number of cell types is given, my method

LSSM performs best when combining with spectral clustering, especially Laplacian eigenmap + k-means clustering. If the number of cell types is unknown, Louvain clustering method can be chosen to identify cell populations. Furthermore, I combine my method LSSM with t-SNE to visualize scRNA-seq data.

My method, Learning Sparse Similarity Matrices (LSSM), has several advantages. First of all, constructing sparse cell-cell similarity matrices can reduce the effect of noise of scRNA-seq data, since some relationships in the dense similarity matrix may be caused by high-dimensional noise. Besides, LSSM assumes each cell can be represented by the other cells of the data set, which is a good way to build the relationships of multiple cells. In addition, if the number of cell types is given, the clustering result is pretty good for LSSM combining with the spectral clustering (Laplacian eigenmap + k-means clustering), no matter the size of the scRNA-seq data set is. Moreover, LSSM has high computational efficiency. Last but not least, the visualization of t-SNE based on my method also is pretty great.

8.2 Future work

Although the accuracy of identifying cell populations is improved with my method, there are several potential directions for my future improvements.

1. Automatically determine the number of clusters with spectral clustering.

LSSM combining with spectral clustering has the best clustering performance, but the number of cell types is needed as an input. So if I can find a good way to automatically decide the number of clusters, this framework will be more practical in single cell RNA sequencing data analysis. There are many excellent methods to determine the cluster number, such as gap statistic [115], elbow method [105] and average silhouette method [57], which may provide inspiration for my future research.

2. Improve the method of gene selection.

Some studies show that in scRNA-seq data analysis, the prior knowledge (i.e. genome annotation) of the data sets can be very helpful in determining genes. More specifically, a number of studies, like [128], [32], [14], have verified that some related and functional gene sets play a key role in scRNA-seq data analysis [124]. Hence, utilizing prior biological knowledge to better filter genes may help improve my method.

3. Combine several clustering methods to improve the clustering performance.

In this study, based on the similarity matrices constructed by LSSM, I use several clustering methods to identify cell populations respectively. One further improvement direction is to combine multiple clustering methods [38] to improve clustering quality. There are many consensus function for integrating multiple clustering methods, such as voting based approach, mixture models, information theory approach, and so on [97]. Some studies have combined several clustering methods in single-cell clustering algorithms. For example,

SC3, mentioned in section 7.1, combines the clustering solutions of k-means clustering and hierarchical clustering in their algorithms.

References

[1] Tamim Abdelaal, Lieke Michielsen, Davy Cats, Dylan Hoogduin, Hailiang Mei, Marcel JT Reinders, and Ahmed Mahfouz. A comparison of automatic cell identification methods for single-cell rna sequencing data. *Genome biology*, 20(1):1–19, 2019.

[2] El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe'er. visne enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature biotechnology*, 31(6):545–552, 2013.

[3] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[4] Alexander Barvinok. *A course in convexity*, volume 54. American Mathematical Soc., 2002.

[5] Sam Behjati and Patrick S Tarpey. What is next generation sequencing? *Archives of Disease in Childhood-Education and Practice*, 98(6):236–238, 2013.

[6] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[7] Rajendra Bhatia and Chandler Davis. A cauchy-schwarz inequality for operators with applications. *Linear algebra and its applications*, 223:119–129, 1995.

[8] Swapan Bhattacharya, Ananya Kanjilal, and Sabnam Sengupta. Tools and techniques for model based testing. In *Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization*, pages 226–249. IGI Global, 2010.

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[10] Charles A Bouman, Michael Shapiro, GW Cook, C Brian Atkins, and Hui Cheng. Cluster: An unsupervised algorithm for modeling gaussian mixtures, 1997.

[11] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[12] Philip Brennecke, Simon Anders, Jong Kyoung Kim, Aleksandra A Kołodziejczyk, Xiuwei Zhang, Valentina Proserpio, Bianka Baying, Vladimir Benes, Sarah A Teichmann, John C Marioni, et al. Accounting for technical noise in single-cell rna-seq experiments. *Nature methods*, 10(11):1093–1095,

2013.

[13] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *arXiv preprint arXiv:1405.4980*, 2014.

[14] Florian Buettner, Naruemon Pratanwanich, Davis J McCarthy, John C Marioni, and Oliver Stegle. f-sclvm: scalable and versatile factor analysis for single-cell rna-seq. *Genome biology*, 18(1):1–13, 2017.

[15] Batuhan Cakir, Martin Prete, Ni Huang, Stijn Van Dongen, Pinar Pir, and Vladimir Yu Kiselev. Comparison of visualization tools for single-cell rnaseq data. *NAR Genomics and Bioinformatics*, 2(3):lqaa052, 2020.

[16] John N Campbell, Evan Z Macosko, Henning Fenselau, Tune H Pers, Anna Lyubetskaya, Danielle Tenen, Melissa Goldman, Anne MJ Verstegen, Jon M Resch, Steven A McCarroll, et al. A molecular census of arcuate hypothalamus and median eminence cell types. *Nature neuroscience*, 20(3):484–496, 2017.

[17] Ricardo JGB Campello, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1343, 2020.

[18] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[19] Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M Ibrahim, Andrew J Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J Steemers, et al. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, 2019.

[20] MR Capobianchi, E Giombini, and G Rozera. Next-generation sequencing technology in clinical virology. *Clinical Microbiology and Infection*, 19(1):15–22, 2013.

[21] Randy L Carter, Robin Morris, and Roger K Blashfield. On the partitioning of squared euclidean distance and its applications in cluster analysis. *Psychometrika*, 54(1):9–23, 1989.

[22] Michael Chau, Reynold Cheng, Ben Kao, and Jackey Ng. Uncertain data mining: An example in clustering location data. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 199–204. Springer, 2006.

[23] Tsong Yueh Chen, Fei-Ching Kuo, and Robert Merkel. On the statistical properties of the f-measure. In *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.*, pages 146–153. IEEE, 2004.

[24] X Chen, J Lam, P Li, and Z Shu. L1-induced performance analysis and sparse controller. *Science*, 1:257–262, 2013.

[25] Yongjun Chu and David R Corey. Rna sequencing: platform selection, experimental design, and data interpretation. *Nucleic acid therapeutics*, 22(4):271–274, 2012.

[26] Andrea Danyluk. *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009.

[27] Herbert A David and Haikady N Nagaraja. *Order statistics*. John Wiley & Sons, 2004.

[28] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[29] Qiaolin Deng, Daniel Ramsköld, Björn Reinius, and Rickard Sandberg. Single-cell rna-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science*, 343(6167):193–196, 2014.

[30] Javier Díaz-Giménez. Linear quadratic approximation: An introduction. *Computational methods for the study of dynamic economies*, pages 13–29, 1999.

[31] Angelo Duò, Mark D Robinson, and Charlotte Soneson. A systematic performance evaluation of clustering methods for single-cell rna-seq data. *F1000Research*, 7, 2018.

[32] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[33] Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

[34] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[35] Abdelilah Et-taleby, Mohammed Boussetta, and Mohamed Benslimane. Faults detection for photovoltaic field based on k-means, elbow, and average silhouette techniques through the segmentation of a thermal image. *International Journal of Photoenergy*, 2020, 2020.

[36] Laura Ferreira and David B Hitchcock. A comparison of hierarchical methods for clustering functional data. *Communications in Statistics-Simulation and Computation*, 38(9):1925–1949, 2009.

[37] Richard Forster. Louvain community detection with parallel heuristics on gpus. In *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)*, pages 227–232. IEEE, 2016.

[38] Ana LN Fred and Anil K Jain. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):835–850, 2005.

[39] H Dunternan George et al. Principal components analysis. *Sage*, (69), 1989.

[40] Teofilo F Gonzalez. *Handbook of approximation algorithms and metaheuristics*. CRC Press, 2007.

[41] Susanne Graf, Roberto Passerone, and Sophie Quinton. Contract-based reasoning for component systems with rich interactions. In *Embedded systems development*, pages 139–154. Springer, 2014.

[42] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content*, 1:9–16, 2004.

[43] Dominic Grün, Anna Lyubimova, Lennart Kester, Kay Wiebrands, Onur Basak, Nobuo Sasaki, Hans Clevers, and Alexander Van Oudenaarden. Single-cell messenger rna sequencing reveals rare intestinal cell types. *Nature*, 525(7568):251–255, 2015.

[44] Minzhe Guo, Hui Wang, S Steven Potter, Jeffrey A Whitsett, and Yan Xu. Sincera: a pipeline for single-cell rna-seq profiling analysis. *PLoS computational biology*, 11(11):e1004575, 2015.

[45] Jin Huang, Feiping Nie, and Heng Huang. A new simplex sparse learning model to measure data similarity for clustering. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[46] Byungjin Hwang, Ji Hyun Lee, and Duhee Bang. Single-cell rna sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine*, 50(8):1–14, 2018.

[47] Martin Jaggi. *Sparse convex optimization methods for machine learning*. PhD thesis, ETH Zurich, 2011.

[48] Hyundoo Jeong and Navadon Khunlertgit. Effective single-cell clustering through ensemble feature selection and similarity measurements. *Computational Biology and Chemistry*, 87:107283, 2020.

[49] TR JeraldBeno and M Karnan. Dimensionality reduction: rough set based feature reduction. *Int. J. Sci. Res. Publ*, 2:1–6, 2012.

[50] Zhicheng Ji and Hongkai Ji. Tscan: Pseudo-time reconstruction and evaluation in single-cell rna-seq analysis. *Nucleic acids research*, 44(13):e117–e117, 2016.

[51] Hao Jiang, Lydia L Sohn, Haiyan Huang, and Luonan Chen. Single cell clustering based on cell-pair differentiability correlation and variance analysis. *Bioinformatics*, 34(21):3684–3694, 2018.

[52] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[53] Ian Jolliffe. Principal component analysis. *Encyclopedia of statistics in behavioral science*, 2005.

[54] Ian T Jolliffe. Springer series in statistics. *Principal component analysis*, 29, 2002.

[55] Nezamoddin N Kachouie and Meshal Shutaywi. Weighted mutual information for aggregated kernel clustering. *Entropy*, 22(3):351, 2020.

[56] K Kameshwaran and K Malarvizhi. Survey on clustering techniques in data mining. *International Journal of Computer Science and Information Technologies*, 5(2):2272–2276, 2014.

[57] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[58] Mehdi Kchouk, Jean-Francois Gibrat, and Mourad Elloumi. Generations of sequencing technologies: from first to next generation. *Biology and Medicine*, 9(3), 2017.

[59] Vladimir Yu Kiselev, Tallulah S Andrews, and Martin Hemberg. Challenges in unsupervised clustering of single-cell rna-seq data. *Nature Reviews Genetics*, 20(5):273–282, 2019.

[60] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, et al. Sc3: consensus clustering of single-cell rna-seq data. *Nature methods*, 14(5):483–486, 2017.

[61] Martin G Klotz. *Research on Nitrification and Related Processes, Part B*. Academic Press, 2011.

[62] Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[63] Pang Wei Koh, Rahul Sinha, Amira A Barkal, Rachel M Morganti, Angela Chen, Irving L Weissman, Lay Teng Ang, Anshul Kundaje, and Kyle M Loh. An atlas of transcriptional, chromatin accessibility, and surface marker changes in human mesoderm development. *Scientific data*, 3(1):1–15, 2016.

[64] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[65] Roshan M Kumar, Patrick Cahan, Alex K Shalek, Rahul Satija, A Jay DaleyKeyser, Hu Li, Jin Zhang, Keith Pardee, David Gennert, John J Trombetta, et al. Deconstructing transcriptional heterogeneity in pluripotent stem cells. *Nature*, 516(7529):56–61, 2014.

[66] Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F Stewart, and Adam Perer. Clustervision: Visual supervision of unsupervised clustering. *IEEE transactions on visualization and computer graphics*, 24(1):142–151, 2017.

[67] Yuanyuan Li, Ping Luo, Yi Lu, and Fang-Xiang Wu. Identifying cell types from single-cell data based on similarities and dissimilarities between cells. *BMC bioinformatics*, 22(3):1–18, 2021.

[68] Peijie Lin, Michael Troup, and Joshua WK Ho. Cidr: Ultrafast and accurate clustering through imputation for single-cell rna-seq data. *Genome biology*, 18(1):1–11, 2017.

[69] George C Linderman, Jun Zhao, and Yuval Kluger. Zero-preserving imputation of scrna-seq data using low-rank approximation. *BioRxiv*, page 397588, 2018.

[70] Jialu Liu and Jiawei Han. Spectral clustering. In *Data Clustering*, pages 177–200. Chapman and Hall/CRC, 2018.

[71] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. Understanding of internal clustering validation measures. In *2010 IEEE international conference on data mining*, pages 911–916. IEEE, 2010.

[72] Zijing Liu and Mauricio Barahona. Graph-based data clustering via multiscale community detection.

Applied Network Science, 5(1):1–20, 2020.

[73] David G Lowe. Similarity metric learning for a variable-kernel classifier. *Neural computation*, 7(1):72–85, 1995.

[74] Malte D Luecken and Fabian J Theis. Current best practices in single-cell rna-seq analysis: a tutorial. *Molecular systems biology*, 15(6):e8746, 2019.

[75] Aaron TL Lun, Davis J McCarthy, and John C Marioni. A step-by-step workflow for low-level analysis of single-cell rna-seq data with bioconductor. *F1000Research*, 5, 2016.

[76] Markus Maier, Ulrike Von Luxburg, and Matthias Hein. How the result of graph clustering methods depends on the construction of the graph. *ESAIM: Probability and Statistics*, 17:370–418, 2013.

[77] Vitruvian Man and Leonardo da Vinci. Human genome project.

[78] Christopher D Manning and Prabhakar Raghavan. utze, introduction to information retrieval, 2008.

[79] Georgi K Marinov, Brian A Williams, Ken McCue, Gary P Schroth, Jason Gertz, Richard M Myers, and Barbara J Wold. From single-cell to cell-pool transcriptomes: stochasticity in gene expression and rna splicing. *Genome research*, 24(3):496–510, 2014.

[80] Allan M Maxam and Walter Gilbert. A new method for sequencing dna. *Proceedings of the National Academy of Sciences*, 74(2):560–564, 1977.

[81] Daniel L Moody. Entity connectivity vs. hierarchical levelling as a basis for data model clustering: An experimental analysis. In *International Conference on Database and Expert Systems Applications*, pages 77–87. Springer, 2003.

[82] Quy H Nguyen, Nicholas Pervolarakis, Kevin Nee, and Kai Kessenbrock. Experimental considerations for single-cell rna sequencing approaches. *Frontiers in cell and developmental biology*, 6:108, 2018.

[83] Frank Nielsen. Hierarchical clustering. In *Introduction to HPC with MPI for Data Science*, pages 195–211. Springer, 2016.

[84] Fatih Ozsolak and Patrice M Milos. Rna sequencing: advances, challenges and opportunities. *Nature reviews genetics*, 12(2):87–98, 2011.

[85] Seyoung Park and Hongyu Zhao. Spectral clustering based on learning similarity matrix. *Bioinformatics*, 34(12):2069–2076, 2018.

[86] Youngjun Park, Anne-Christin Hauschild, and Dominik Heider. Transfer learning compensates limited data, batch-effects, and technical heterogeneity in single-cell sequencing. *bioRxiv*, 2021.

[87] Sakshi Patel, Shivani Sihmar, and Aman Jatain. A study of hierarchical clustering algorithms. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 537–541. IEEE, 2015.

[88] S. Paul, P. Bhattacharya, and A. Bit. *Early Detection of Neurological Disorders Using Machine Learning Systems*. Advances in Medical Technologies and Clinical Practice. IGI Global, 2019.

[89] Raphael Petegrosso, Zhuliu Li, and Rui Kuang. Machine learning and statistical methods for clustering single-cell rna-sequencing data. *Briefings in bioinformatics*, 21(4):1209–1223, 2020.

[90] S Steven Potter. Single-cell rna sequencing for the study of development, physiology and disease. *Nature Reviews Nephrology*, 14(8):479–492, 2018.

[91] Ren Qi, Anjun Ma, Qin Ma, and Quan Zou. Clustering and classification methods for single-cell rna-sequencing data. *Briefings in bioinformatics*, 21(4):1196–1208, 2020.

[92] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[93] Antoine-Emmanuel Saliba, Lei Li, Alexander J Westermann, Silke Appenzeller, Daphne AC Stapels, Leon N Schulte, Sophie Helaine, and Joerg Vogel. Single-cell rna-seq ties macrophage polarization to growth rate of intracellular salmonella. *Nature microbiology*, 2(2):1–8, 2016.

[94] Antoine-Emmanuel Saliba, Alexander J Westermann, Stanislaw A Gorski, and Jörg Vogel. Single-cell rna-seq: advances and future challenges. *Nucleic acids research*, 42(14):8845–8860, 2014.

[95] Jörg Sander, Xuejie Qin, Zhiyong Lu, Nan Niu, and Alex Kovarsky. Automatic extraction of clusters from hierarchical clustering representations. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 75–87. Springer, 2003.

[96] Frederick Sanger, Steven Nicklen, and Alan R Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467, 1977.

[97] S Sarumathi, N Shanthi, and G Santhiya. A survey of cluster ensemble. *International Journal of Computer Applications*, 65(9), 2013.

[98] Rahul Satija, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. Spatial reconstruction of single-cell gene expression data. *Nature biotechnology*, 33(5):495–502, 2015.

[99] Anne Senabouth, Samuel W Lukowski, Jose Alquicira Hernandez, Stacey B Andersen, Xin Mei, Quan H Nguyen, and Joseph E Powell. ascend: R package for analysis of single-cell rna-seq data. *GigaScience*, 8(8):giz087, 2019.

[100] Elif Funda Sener, Halit Canatan, and Yusuf Ozkul. Recent advances in autism spectrum disorders: applications of whole exome sequencing technology. *Psychiatry investigation*, 13(3):255, 2016.

[101] Chunxuan Shao and Thomas Höfer. Robust classification of single-cell transcriptome data by nonnegative matrix factorization. *Bioinformatics*, 33(2):235–242, 2017.

[102] NK Sharma, RC Jain, and Manoj Yadav. A survey on data mining algorithms and future perspective. *IJCSIT) International Journal of Computer Science and Information Technologies*, 3(5):5149–5156,

2012.

[103] Debajyoti Sinha, Akhilesh Kumar, Himanshu Kumar, Sanghamitra Bandyopadhyay, and Debarka Sengupta. dropclust: efficient clustering of ultra-large scrna-seq data. *Nucleic acids research*, 46(6):e36–e36, 2018.

[104] Charlotte Soneson and Mark D Robinson. Bias, robustness and scalability in single-cell differential expression analysis. *Nature methods*, 15(4):255–261, 2018.

[105] MA Syakur, BK Khotimah, EMS Rochman, and Budi Dwi Satoto. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. In *IOP Conference Series: Materials Science and Engineering*, volume 336, page 012017. IOP Publishing, 2018.

[106] Fuchou Tang, Catalin Barbacioru, Yangzhou Wang, Ellen Nordman, Clarence Lee, Nanlan Xu, Xiaohui Wang, John Bodeau, Brian B Tuch, Asim Siddiqui, et al. mrna-seq whole-transcriptome analysis of a single cell. *Nature methods*, 6(5):377–382, 2009.

[107] Xiaoning Tang, Yongmei Huang, Jinli Lei, Hui Luo, and Xiao Zhu. The single-cell sequencing: new developments and medical applications. *Cell & bioscience*, 9(1):1–9, 2019.

[108] Bosiljka Tasic, Vilas Menon, Thuc Nghi Nguyen, Tae Kyung Kim, Tim Jarsky, Zizhen Yao, Boaz Levi, Lucas T Gray, Staci A Sorensen, Tim Dolbeare, et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nature neuroscience*, 19(2):335–346, 2016.

[109] Manuela Tavian and Bruno Peault. Embryonic development of the human hematopoietic system. *International Journal of Developmental Biology*, 49(2-3):243–250, 2003.

[110] Luyi Tian, Xueyi Dong, Saskia Freytag, Kim-Anh Lê Cao, Shian Su, Abolfazl JalalAbadi, Daniela Amann-Zalcenstein, Tom S Weber, Azadeh Seidi, Jafar S Jabbari, et al. Benchmarking single cell rna-sequencing analysis pipelines using mixture control experiments. *Nature methods*, 16(6):479–487, 2019.

[111] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[112] F William Townes, Stephanie C Hicks, Martin J Aryee, and Rafael A Irizarry. Feature selection and dimension reduction for single-cell rna-seq based on a multinomial model. *Genome biology*, 20(1):1–16, 2019.

[113] Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology*, 32(4):381–386, 2014.

[114] Olga Vechtomova. Introduction to information retrieval christopher d. manning, prabhakar raghavan, and hinrich schütze (stanford university, yahoo! research, and university of stuttgart) cambridge: Cambridge university press, 2008, xxi+ 482 pp; hardbound, isbn 978-0-521-86571-5, 2009.

[115] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[116] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[117] Bo Wang, Junjie Zhu, Emma Pierson, Daniele Ramazzotti, and Serafim Batzoglou. Visualization and analysis of single-cell rna-seq data by kernel-based similarity learning. *Nature methods*, 14(4):414–416, 2017.

[118] Yong Wang and Nicholas E Navin. Advances and applications of single-cell sequencing technologies. *Molecular cell*, 58(4):598–609, 2015.

[119] Geoffrey S Watson. Linear least squares regression. *The Annals of Mathematical Statistics*, pages 1679–1699, 1967.

[120] F Alexander Wolf, Philipp Angerer, and Fabian J Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19(1):1–5, 2018.

[121] Samuel L Wolock, Romain Lopez, and Allon M Klein. Scrublet: computational identification of cell doublets in single-cell transcriptomic data. *Cell systems*, 8(4):281–291, 2019.

[122] Yurong Xin, Jinrang Kim, Haruka Okamoto, Min Ni, Yi Wei, Christina Adler, Andrew J Murphy, George D Yancopoulos, Calvin Lin, and Jesper Gromada. Rna sequencing of single human islet cells reveals type 2 diabetes genes. *Cell metabolism*, 24(4):608–615, 2016.

[123] Chen Xu and Zhengchang Su. Identification of cell types from single-cell transcriptomes using a novel clustering method. *Bioinformatics*, 31(12):1974–1980, 2015.

[124] Yunpei Xu, Hong-Dong Li, Yi Pan, Feng Luo, and Jianxin Wang. Biorank: A similarity assessment method for single cell clustering. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 157–162. IEEE, 2018.

[125] Liying Yan, Mingyu Yang, Hongshan Guo, Lu Yang, Jun Wu, Rong Li, Ping Liu, Ying Lian, Xiaoying Zheng, Jie Yan, et al. Single-cell rna-seq profiling of human preimplantation embryos and embryonic stem cells. *Nature structural & molecular biology*, 20(9):1131, 2013.

[126] Yun Yang. *Temporal data mining via unsupervised ensemble learning*. Elsevier, 2016.

[127] Shun H Yip, Pak Chung Sham, and Junwen Wang. Evaluation of tools for highly variable gene discovery from single-cell rna-seq data. *Briefings in bioinformatics*, 20(4):1583–1589, 2019.

[128] Matthew D Young, Matthew J Wakefield, Gordon K Smyth, and Alicia Oshlack. Gene ontology analysis for rna-seq: accounting for selection bias. *Genome biology*, 11(2):1–12, 2010.

[129] Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8(1):1–12, 2017.

[130] Bo Zhou and Wenfei Jin. Visualization of single cell rna-seq data using t-sne in r. In *Stem Cell Transcriptional Networks*, pages 159–167. Springer, 2020.

[131] Wenbo Zhu, Zachary T Webb, Kaitian Mao, and Jose Romagnoli. A deep learning approach for process data visualization using t-distributed stochastic neighbor embedding. *Industrial & Engineering Chemistry Research*, 58(22):9564–9575, 2019.

Appendix A

Data preprocessing

A.1 Data load

```
import pandas
import numpy as np

data = pd.read_csv('yan_data.txt',sep='\t')
data_lab = pd.read_csv('yan_lab.txt',sep='\t',header=None)
x = np.matrix(data).astype(float)

print(data_lab.head())
print(x)
```

A.2 Removing zeros

```
import math

m, n = x.shape
for i in range(m):
    for j in range(n):
        x[i,j] = math.log10(x[i,j]+1)

x = pd.DataFrame(x)
print(x.shape)
```

A.3 Feature selection

```
a,b = x.shape
gene_list = []
for i in range(b):
    gene_var = np.var(x.loc[:,i])
    gene_list.append(gene_var)

print(gene_list)

x_length = len(x)
x.loc[x_length] = gene_list
x_sort= x.sort_values(by=a, ascending=False, axis=1)
print(x_sort)

x_selct = x_sort.iloc[:,0:2000]
x_selct = np.matrix(x_selct)

print(x_selct.shape)
```

A.4 PCA

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA

scaler = MinMaxScaler()
data_rescaled = scaler.fit_transform(x_selct)
pca = PCA(n_components = 34)
pca.fit(data_rescaled)
x_redc = pca.transform(data_rescaled)
x_redc = np.matrix(x_redc)

print(x_redc.shape)
```

Appendix B

Sparse cell-cell similarity matrix contribution

B.1 Initialization

```
from numpy.linalg import norm

sim = np.zeros((n,n))
norm_x = norm(x_redc,2)**2
cov=np.array(x_redc.T*x_redc)

print(cov.shape)

for i in range(n):
    for j in range(n):
        if i==j:
            cov[i,j]=0
cov_maxlist = list(cov.argmax(axis=1))

print(len(cov_maxlist))
```

B.2 Similarity matrix calculation

```
import time
time_start = time.time()

for i_0 in range(n):
    print(i_0)
    a_0 = 1
    k = 0
    alpha_0 = np.matrix(np.zeros((1,n)))
    alpha_0[0,i_0] = -1
    b = cov_maxlist[i_0]
    alpha_0[0,b] = 1
    beta_0 = x_redc*alpha_0.T
    alpha_k = np.matrix(np.zeros((1,n)))
    beta_k = np.matrix(np.zeros((m,1)))

    from numpy.linalg import norm

    e_0 = np.matrix(np.zeros((n,n)))
    e_0[:,i_0] = -1
    e = np.matrix(np.eye((n)))
    e_1 = e+e_0   # construct conv of omega 0

    while True:
        k += 1
        a_k = (1+np.sqrt(4*a_0**2+1))/2

    ## calculating i+
        opt = []
```

```
        for j in range(n): # j reprents i
            if i_0 == j:
                b = np.inf
            else:
                beta_0 = x_redc*alpha_0[:,:].T
                b = -(x_redc[:,i_0]-x_redc[:,j]).T*beta_0[:,:]
            opt.append(b)
    #print(opt)
        c = opt.index(min(opt))
    #print(c)

        alpha_k[:,:] = alpha_0[:,:] + 1/a_0*(e_1[c,:]-alpha_0[:,:])
        beta_k[:,:] = beta_0[:,:] + 1/a_0*(x_redc[:,i_0]-x_redc[:,c]-beta_0[:,:])
        a_0 = a_k
        conveg =
             np.abs(norm((x_redc*alpha_k[:,:].T),2)**2-norm((x_redc*alpha_0[:,:].T),2)**2)/norm_x
        #print(conveg)
        if k>0.4*n:

            print("iterations are arrived")
            sim[i_0,:] = alpha_k[:,:]
            break
        #conveg = np.abs(norm((x*alpha_k[:,:].T),2)**2-norm((x*alpha_0[:,:].T),2)**2)
        if conveg<0.0000001:
            sim[i_0,:] = alpha_k[:,:]
            break
        alpha_0[:,:] = alpha_k[:,:]
        #print(conveg)
print(sim)

time_end = time.time()
time_cost = time_end-time_start
print(time_cost)

# keep the similarity matrix to be symmetrical, and the sum of every column to be 1.
sim = (sim+sim.T)/2
for i in range(n):
    for j in range(n):
        if i == j:
            sim[i,j] = 0
colum=list(sim.sum(axis=1))
for i in range(n):
    for j in range(n):
            sim[i,j]=sim[i,j]/colum[i]
print(sim)
```

Appendix C

Clustering and visualization

C.1 Spectral clustering

```
D = np.sum(sim,axis=1)
L = np.diag(D) - sim
# normailze
sqrtDegreeMatrix = np.diag(1.0 / (D**(0.5)))

Nor = np.dot(np.dot(sqrtDegreeMatrix,L),sqrtDegreeMatrix)
L = Nor
print(L)

eigval,eigvec = np.linalg.eig(L)
cluster_num = 6

dim = len(eigval)
dictEigval = dict(zip(eigval,range(0,dim)))
kEig = np.sort(eigval)[0:cluster_num]
ix = [dictEigval[k] for k in kEig]
print(eigval[ix])
print(eigvec[:,ix])

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=cluster_num,random_state=0)
pred_y = kmeans.fit_predict(eigvec[:,ix])
y=np.array(data_lab.T)

from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=cluster_num,random_state=0)
pred_y_G = gm.fit_predict(eigvec[:,ix])
```

C.2 Louvain clustering

```
pip install python-louvain

import community as community_louvain
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import networkx as nx

G = nx.from_numpy_matrix(np.array(sim))
partition.items()
pred_y_L = partition.values()
```

C.3 Hierarchical clustering

```
for i in range(n):
    for j in range(n):
        distce[i,j] = 1-sim[i,j]
print(distce)

for i in range(n):
    for j in range(n):
        if i == j:
            distce[i,j] = 0
print(distce)

from scipy.cluster.hierarchy import dendrogram, linkage, cut_tree
from matplotlib import pyplot as plt

# in this case, I use 'ward'
Z = linkage(distce, 'ward')

print(cut_tree(Z, n_clusters=cluster_num))

pred_y_H = cut_tree(Z, n_clusters=cluster_num)
print(pred_y_H)
```

C.4 NMI, ARI, Purity

```
from sklearn.metrics import adjusted_rand_score

ari=adjusted_rand_score(y[0], pred_y)
print(' ARI %f'%(ari))
from sklearn.metrics.cluster import normalized_mutual_info_score
nmi = normalized_mutual_info_score(y[0], pred_y)
print(' NMI %f'%(nmi))

# purity
from collections import Counter

def purity(labels, clustered):

    # find the set of cluster ids
    cluster_ids = set(clustered)

    N = len(clustered)
    majority_sum = 0
    for cl in cluster_ids:

        # for this cluster, I compute the frequencies of the different human labels I encounter
        # the result will be something like { 'camera':1, 'books':5, 'software':3 } etc.
        labels_cl = Counter(l for l, c in zip(labels, clustered) if c == cl)

        # I select the *highest* score and add it to the total sum
        majority_sum += max(labels_cl.values())

    # the purity score is the sum of majority counts divided by the total number of items
    return majority_sum / N

purity = purity(y[0], pred_y)
```

```
print(purity)
```

C.5 t-SNE

```
import numpy as np
import matplotlib.pyplot as plt

realmin = 2.2251e-308
def tsne_l (P, labels, n_dims=2):
    #P is a similarity matrix
    # Check initial solution
    if not isinstance(n_dims, int) and len(np.reshape(n_dims, -1))>1:
        initial_solution = True
        ydata = n_dims
        n_dims = y_data.shape[1]
    else:
        initial_solution = False
    n = P.shape[0]
    momentum = 0.08;                                  # initial momentum
    final_momentum = 0.1;                             # value to which momentum is changed
    mom_switch_iter = 250;                            # iteration at which momentum is changed
    stop_lying_iter = 100;                            # iteration at which lying about P-values is
        stopped
    max_iter = 1000;                                  # maximum number of iterations
    epsilon = 500;                                    # initial learning rate
    min_gain = 0.01;

    for i in range(n):
        P[i,i] = 0
    P = 0.5*(P+P.T)
    P /= P.sum()
    P = np.where(P>=realmin, P, np.ones(P.shape)*realmin)
    const = (P*np.log(P)).sum()
    if not initial_solution:
        P = P * 4

    if not initial_solution:
        ydata = 0.0001 * np.random.randn(n, n_dims)

    y_incs = np.zeros(ydata.shape)
    gains = np.ones(ydata.shape)

    for itera in range(max_iter):
        sum_ydata = (ydata**2).sum(axis=1)

        # column add
        denomi = sum_ydata.T.reshape((-1,1)) - 2* ydata.dot(ydata.T)
        # row add
        denomi = denomi + sum_ydata
        denomi += 1
        num = 1/denomi

        # reset the diagal
        for i in range(num.shape[0]):
            num[i,i] = 0
        Q = num / num.sum()
```

```
        Q = np.where(Q>=realmin, Q, np.ones(Q.shape)*realmin)

        L = (P-Q) * num
        y_grads = 4 * (np.diag(L.sum(axis=0)) - L).dot(ydata)

        gains = (gains + 0.2) * (np.sign(y_grads) != np.sign(y_incs)) + (gains * 0.8) *
            (np.sign(y_grads)==np.sign(y_incs))
        gains = np.where(gains>=min_gain, gains, np.ones(gains.shape)*min_gain)
        y_incs = momentum * y_incs - epsilon * (gains * y_grads)
        ydata = ydata + y_incs
        ydata = ydata - ydata.mean(axis=0)
        ydata = np.where(ydata>=-100, ydata, np.ones(ydata.shape)*(-100))
        ydata = np.where(ydata<=100, ydata, np.ones(ydata.shape)*100)

        if itera == mom_switch_iter:
            momentum = final_momentum
        if itera == stop_lying_iter and not initial_solution:
            P = P / 4

    colors = np.random.rand(11)
    plt.scatter(ydata[:,0],ydata[:,1],c=labels)
    plt.show()

    return ydata,labels
```