Visualization and Analysis of Single-Cell Data using Diffusion Maps

Lisa Mariah Beer

Born 23rd December 1994 in Suhl, Germany

24th September 2018

Master's Thesis Mathematics

Advisor: Prof. Dr. Jochen Garcke

Second Advisor: Prof. Dr. Martin Rumpf

Institute for Numerical Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

Symbols and Abbreviations

Symbols

$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$m \ll d$	m is much smaller than d
$\|\cdot\|_2$	Euclidean norm
$G = (V, E)$	A graph consisting of a set of vertices V and (weighted) edges E
$\mathbb{1}$	A vector with ones
$diag(v)$	A diagonal matrix with v being the diagonal
$\rho(A)$	The spectral radius of a matrix A
$\mathcal{O}(g)$	Big O notation
χ_I	The indicator function of a set I
erfc	The Gaussian complementary error function
$\mathbb{E}[X]$	The expectation of a random variable X
$Var[X]$	The variance of a random variable X
$\|X\|$	The cardinality of a set X
$\mathcal{X}$	A given data set
$\mathcal{M}$	The manifold, the data is lying on
n	The number of data points in $\mathcal{X}$
d	The dimension of a data point $x \in \mathcal{X}$
m	The dimension of the manifold $\mathcal{M}$

Abbreviations

DBSCAN	Density-based spatial clustering of applications with noise
DM	Diffusion maps
DPT	Diffusion pseudotime
ISOMAP	Isometric feature mapping
LE	Laplacian eigenmaps
LLE	Locally linear embedding
MDS	Multidimensional scaling
PCA	Principal component analysis
PCR	Polymerase chain reaction
tSNE	t-distributed stochastic neighbour embedding

1 Introduction

Single-cell data analysis is nowadays an indispensable part of biological research. Biologists aim to extract valuable information out of data in order to gain new insights about the development of single cells from one cell type to another, the so-called cell differentiation. However, due to new and better data collection techniques, the size of biological data sets has increased immensely in recent years. When data is of high dimensionality, the direct receipt of information is usually no longer possible. Analyzing data in high-dimensional spaces becomes computationally too expensive. This phenomenon is well known in the literature as the *curse of dimensionality*. In order to reveal hidden structure of high-dimensional biological data, it is therefore essential to apply *machine learning* methods.
One useful approach is to tranform the data into a more compact form, which is known as *dimensionality reduction*. Biologists use dimensionality reduction methods to visualize the data in order to obtain an overall picture of the data set. Afterwards, they can do further analysis, for example detecting different cell groups, representing specific differentiation stages. However, classical dimensionality reduction methods, such as *principal component analysis* (PCA), often fail to reveal the special structure of differentiation data. For instance, PCA has the restricted assumption that the data is lying on a linear subspace, which is not suitable for the mostly nonlinear single-cell data.
In recent years, a more appropriate dimensionality reduction method was proposed: *Diffusion maps*. Diffusion maps is a nonlinear dimensionality reduction method, established by Coifman and Lafon in 2004-2006 [CL06]. In [HBT15], diffusion maps were firstly introduced in combination with single-cell differentiation data and revealed promising results.

Objective of this work

In this work, we carry out further investigations of diffusion maps as a dimensionality reduction method for biological applications. In addition to data visualization, the focus is on finding a learning method for partitioning the single-cell data into meaningful cell groups using diffusion maps as preprocessing step. For this, we compare the performance of *spectral clustering* [BK17] to the *diffusion pseudotime* (DPT) analysis, especially developed for biological data in [HBW$^+$16].
Moreover, we deal with so-called *censored* biological data. Censored data is a special form of data with missing values, where a certain range of numbers is not detected due to experimental reasons. So far, censored values are not specifically treated in the data processing. In [HBT15], a procedure for considering censored data is presented which estimates the kernel function used in diffusion maps. We investigate this approach and compare it to the performance of an alternative method, based on [EDVL13].
Great emphasis in this work is placed on the selection and influence of various parameters.

Own contributions

- Investigation of the influence of different parameters and Gaussian kernel functions on the performance of diffusion maps for data visualization.
- Examination of the performance of spectral clustering in combination with diffusion maps and several common clustering techniques for biological data.
- Extension of the diffusion pseudotime analysis for allowing directly to specify any number of groups and proposal for a method to find the correct number of groups.
- Proposal of using the procedure, based on [EDVL13], to consider censored biological data.

Outline of this work

In *chapter 2*, we give an introduction into the biological single-cell data processing and present the used data in this work. After giving an overview of dimensionality reduction, *chapter 3* describes the diffusion maps method in detail. The influence of different parameters and kernel functions used in diffusion maps on the data visualization is examined in the end of this chapter. In *chapter 4*, we introduce the two group detection techniques, spectral clustering and diffusion pseudotime analysis, including an extension of it, and investigate and compare the two methods. Subsequently, *chapter 5* deals with two approaches for considering censored data. Finally, we summarize the results of this work in *chapter 6* and give a brief outlook for possible further research.

2 Single-Cell Data

New and better technologies for measuring gene levels in single cells have ushered in a new era in biological research. Biologists hope for gaining new detailed insights into how single cells, in particular stem cells, differentiate.
During the differentiation process of a stem cell to a specific cell type, the cell passes many different stages of development. Depending on the target state, each stem cell develops individually. Two stem cells, differentiating to similar cell types, e.g. to two different muscle cells, go through the same stages first, e.g. to develop into a general muscle cell, before they individually differentiate into their respective target cell types. To biologists, this point in time representing a specific differentiation stage at which two or more of such development branches are formed, is of particular interest.
In order to study the temporal evolution of single cells, cell measurements are collected for different differentiation stages and are united into a single data set. For each cell, gene expression analysis is done by measuring a certain number of genes. In mathematical notation, for n being the number of cells and d the number of genes, the data set is given by

$$\mathcal{X} = \{x_i^t \mid i = 1, \dots, n; t \in I\} \subset \mathbb{R}^d$$

where $I = \{t_1, \dots, t_T\} \subset \mathbb{R}$ denote the $T \in \mathbb{N}$ differentiation stages or rather time points when the cell was measured. In other words, the data set represents a matrix with the rows being the cells and the columns being the genes. In particular, the number of measured genes represents the dimension of the cell data points we want to reduce using diffusion maps. The differentiation stages are used for labeling the cells.
However, in order to ensure accurate and meaningful data analysis, the raw data sets often require preprocessing techniques first, such as data cleaning, normalization and handling missing or uncertain values. The data is normalized in order to obtain more accurate results. A common strategy in biology is the normalization via reference genes or so-called *housekeeping genes*, e.g. by subtracting for each cell the mean expression of the control genes. Cells with undetectable gene data are usually excluded from analysis.
As a special form of data with undetectable values, *censored* data often have to be considered in biological applications:

Definition 1 (Censored data)**.** *For a data point $x \in \mathcal{X}$, let $M_x \subset \{1, \dots, d\}$ be the set of missing components, i.e. undetectable gene entries of x. We call a data set $\mathcal{X}$ censored, if the following two properties hold for all data points $x \in \mathcal{X}$:*

1. *For all detected entries $g \notin M_x$, we have $x_g < L$ and*
2. *for all undetected entries $g \in M_x$, we know that the real values fulfill $x_g \geq L$*

for a $L \in \mathbb{R}$. If $\mathcal{X}$ is censored, we call a undetectable value x_g with $g \in M_x$ for $x \in \mathcal{X}$ a censored value.

This means, in particular, that we do not have any knowledge about the distribution of the missing values. The so-called PCR (polymerase chain reaction) data sets usually contain a high amount of censored values. In a PCR, in order to measure the DNA or gene concentration of a cell, several cycles are conducted. At each cycle, the amount of fluorescence is measured. Subsequently, the so-called Ct value (abbreviation for threshold cycle value) is determined as a measure of the gene concentration. A Ct-value is defined as the cycle number at which the fluorescence significantly exceeds the background-fluorescence, i.e. at which a clear fluorescence signal is first detected. Thus, a higher Ct value means a lower DNA or gene concentration. However, there are cases, where no clear fluorescence signal is measured for a gene in any cycle of the PCR. This means, theoretically, we would need to conduct more cycles to measure the value. However, biologists usually set all these censored values to a fixed value, called *limit of detection* (LoD), which is defined as the maximum value that can be measured in the PCR (i.e. $L = \text{LoD}$).
For the experiments in this thesis, we consider one toy data set and seven real biological data sets, in total. The data vary in the amount of cells and measured genes per cell. We give for each biological data set a description about the differentiation process, the data set should point out. This will give us a clue about the accuracy of the diffusion maps analysis. Besides, we describe the used preprocessing techniques for the biological data, in particular the treatment of the censored values for the experiments in chapter 3 and 4. Notice that for handling censored values in chapter 5, only the PCR data sets are relevant.

Toy data
As toy data, we use a set containing 5 branches (600 cells and 100 genes), artificially made by Moon et al. in [MvDW$^+$17]. The algorithm for creating the data is taken from the supplementary implementation package of [MvDW$^+$17].

Guo data
The single-cell qRT-PCR[1] data set from [GHT$^+$10] contains Ct values for 48 genes of 442 mouse embryonic stem cells at seven different developmental time points, from zygote to blastocyst (1-cell stage to 64-cell stage). Starting at the 1-cell stage, cells transition smoothly either towards the trophectoderm (TE) lineage or the inner cell mass (ICM). Subsequently, cells transition from the inner cell mass either towards the primitive endoderm (PE) or the epiblast (EPI) lineage. The data is normalized by the mean of the reference genes *Actb* and *Gapdh* apart from the censored values, i.e. gene expression values with baseline 28. They are set to the ceiling value of the normalized data set. Gene expression values bigger than the baseline 28 point out undetectable data, i.e. cells with such values are removed (5 in total). Cells from the 1-cell stage embryos are excluded from analysis, as well, since they were treated differently in the experimental procedure (9 in total).

Moignard data
The qRT-PCR data set from [MMS$^+$13] contains Ct values for 24 genes of 620 mouse haemotopoietic stem and progenitor cells from five different cell

[1] real-time quantitative polymerase chain reaction

types: Haematopoietic stem cell (HSC), lymphoid-primed multipotent progenitor (LMPP), common lymphoid progenitor (CLP), granulocyte-monocyte progenitor (GMP) and megakaryocyte-erythroid progenitor (PreMegE). HSC cells can differentiate either towards PreMegE or LMPP cells. Subsequently, LMPP cells transition either towards GMP or the CLP lineage. All gene expression values are normalized by the mean of the two housekeeping genes *Polr2a* and *Ubc*. Cells, where one of the housekeeping genes is not expressed, are excluded from the data (23 in total). Censored values are assigned to a Ct-value of 15 after normalization. Afterwards, the housekeeping genes and gene *Kit* are removed.

Goettgens data
The qRT-PCR data set from [MWH$^+$15] contains Ct values for 46 genes of 3934 single cells at four distinct embryonic stages focusing on early blood development of mouse embryos: Primitive streak (PS), neural plate (NP), head fold (HF) and four somite (4SG and 4SFG-). The haematopoietic cells move either towards the endothelial branch or the erythroid branch. For normalization, all gene expression values are subtracted by the limit of detection 25 and normalized by the mean of the four reference genes *Eif2b1*, *Mrpl19*, *Polr2a* and *Ubc*. Censored values are assigned to a value of -14. Subsequently, the reference genes are removed from the data.

Yan data
The RNA-seq[2] data set from [YYG$^+$13] contains 90 human early embryonic cells from 7 different developmental time points, from oocyte to late blastocyst. 20214 genes were measured per cell.

Mass data
The MARS-seq[3] data from [MBF$^+$16] contains 408 cells with 8657 measured genes per cell. It illustrates the differentiation of erythro-myeloid progenitors (EMP) towards macrophage precursors (pMacs) and macrophages.

Klein data
The single-cell RNA-seq data from [KMA$^+$15] contains 2717 mouse embryonic stem cells with 2047 genes from four different timepoints, namely at 0, 2, 4 and 7 days after leukemia inhibitory factor (LIF) withdrawal.

Paul data
The MARS-seq data from [PAG$^+$15] contains 2730 bone marrow stem cells with 3451 informative genes. The data is subdivided into 19 groups. For better visualization, we regrouped the data into 10 cell types. The data set contains the branch of granulocyte-macrophage progenitors (GMP) and the branch of megakaryocyte-erythrocyte progenitors (MEP).

[2]Single-cell RNA sequencing (RNA-seq) is a method for measuring the concentration of RNA in single cells.

[3]Massively parallel RNA single-cell sequencing (MARS-seq) is an extension of the RNA-seq technology.

3 Data Visualization

In this chapter, the aim is to create meaningful two-dimensional data visualizations for single-cell data using diffusion maps. Initially, we introduce the topic of dimensionality reduction and present the diffusion maps method. For this, let $\mathcal{X} = \{x_i^t \mid i = 1, \dots, n; t \in I\} \subset \mathbb{R}^d$ be a given (single-cell) data set. In biological applications, n is the number of cells and d the number of measured genes. Since the measuring times $t \in I$ are only for labeling the data points, we leave the variable t out and write from now on $\mathcal{X} = \{x_1, \dots, x_n\}$.

3.1 Dimensionality Reduction

The goal of dimensionality reduction is to embed high-dimensional data in a lower-dimensional space. In most applications, it is justified to assume, that the most dimensions are redunant, i.e. the important information of the data can be described by a few dimensions. In mathematical notation, this means, that $\mathcal{X}$ is lying on a m-dimensional manifold $\mathcal{M}$ of $\mathbb{R}^d$ with $m \ll d$. We call d the *extrinsic dimension* and m the *intrinsic dimension* of the data set. For each data point $x \in \mathcal{X}$, dimensionality reduction tries to find a suitable representation $\hat{x} \in \mathbb{R}^m$ in the embedding space. These lower-dimensional representations are denoted as the *latent variables* of the data. If we reduce the dimensionality to 2 or 3 dimensions, it is possible to visualize the embedded data.
The greatest challenge in dimensionality reduction is the fact, that, in general, any knowledge of the manifold $\mathcal{M}$ (in particular the intrinsic dimension m) is not given. Thus, many approaches for dimensionality reduction were introduced in the last years, based on different assumptions about the structure of the unknown manifold. In the following, we will present some of the most important dimensionality reduction methods [HTF09, LV07]:

***Principal component analysis* (PCA)** [Pea01, Hot33] is the oldest and most popular dimensionality reduction method. It is based on the assumption, that $\mathcal{M}$ is a linear subspace. Therefore it belongs to the *linear dimensionality reduction* methods. It finds an embedding by preserving the variance in the data as best as possible. This is done by computing the eigenvectors of the data covariance matrix. The eigenvectors become then the *principal components* of the data set, which are used for the embedding. Although most data sets do not contain any linear structure, PCA is the most used dimensionality reduction method due to its simplicity.

***Multidimensional scaling* (MDS)** is a family of dimensionality reduction methods. In contrast to PCA, the main idea is to preserve pairwise distances or similarities between points, i.e. close data points should stay close in the embedding space. For this, a stress function is minimized, whose form differ depending on the method.

***Isometric feature mapping* (ISOMAP)** [TdSL00] belongs to the field of *nonlinear dimensionality reduction* or *manifold learning*, i.e. it is not based on a linear model. It uses the same criterion as MDS, namely distance preservation. Rather than the Euclidean distance, ISOMAP tries to approximate the geodesic distance on $\mathcal{M}$, measuring the length of the shortest curve between two points in $\mathcal{M}$. This is realized by constructing a graph with the data points being its nodes and then approximating the geodesic distance by the so-called graph distance, measuring the shortest path between points in the graph.

***Locally linear embedding* (LLE)** [RS00] proposes another approach to embed the data. It preserves the local structure of the data set by approximating each data point by a linear combination of its neighbouring points. Thus, LLE takes sparse and dense regions of the data set into consideration.

***Laplacian eigenmaps* (LE)** [BN01] preserves the local structure just as LLE, but in a different way. It constructs a graph and uses a similarity measure for the weights of the graph edges. Usually, a kernel is applied for defining the weights. The embedding is obtained by computing the eigenvectors of the graph Laplacian.

***t-distributed stochastic neighbour embedding* (tSNE)** [vdMH08] defines the similarity between two points by conditional probabilities. The same is done for the lower-dimensional space using the Student-t-distribution. Subsequently, the Kullback-Leibler divergence of the distribution of the lower-dimensional points from the one of the higher-dimensional points is minimized to get the embedding.

Another nonlinear dimensionality reduction method was proposed by Coifman and Lafon in 2006 [CL06]: *Diffusion maps*. Diffusion maps combine local structure and distance preservation by defining a new distance notion, the so-called *diffusion distance*. In the next section, we will consider the concept of diffusion maps in more detail.

3.2 Diffusion Maps

To reveal the geometry of a given data set $\mathcal{X}$ on a manifold $\mathcal{M}$, we first define a notion of affinity or similarity between points in $\mathcal{X}$ using a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, that for all $x, y \in \mathcal{X}$ satisfies:

- K is symmetric: $K(x, y) = K(y, x)$
- K is positivity-preserving: $K(x, y) \geq 0$.

One can think of the data points as being the nodes of an (undirected) weighted graph $(\mathcal{X}, K)$ whose edge weighting function is specified by K. A common choice for K is the Gaussian kernel

Definition 2 (Gaussian kernel)**.**

$$K_\sigma(x, y) \coloneqq \exp\left(-\frac{\|x - y\|_2^2}{2\sigma^2}\right),$$

based on Euclidean distances and a bandwidth $\sigma > 0$. The main idea of diffusion maps is to construct a random walk Markov chain on $\mathcal{X}$, where walking to a nearby data point is more likely than walking to another that is far away. For $x, y \in \mathcal{X}$ we set

$$D(x) \coloneqq \sum_{z \in \mathcal{X}} K(x, z) \tag{3.1}$$

and define the transition matrix

$$P(x, y) \coloneqq \frac{K(x, y)}{D(x)},$$

specifying the Markov chain. The new matrix P inherits the positivity-preserving property of K, but it is no longer symmetric. However, we have gained

$$\sum_{z \in \mathcal{X}} P(x, z) = 1 \tag{3.2}$$

for all $x \in \mathcal{X}$. This means that the matrix entry $P(x, y)$ can be viewed as the one-step transition probability from x to y. For a time parameter $t \in \mathbb{N}$, the power P^t gives the t-step transition matrix, i.e. the entry $P^t(x, y)$ represents the transition probability from x to y in t time steps. Thus, running the chain forward in time describes the diffusion process of the data $\mathcal{X}$ at various scales.
The Markov chain now allows us to define a time-dependent distance measure on $\mathcal{X}$, the so-called *diffusion distance*.

Definition 3 (Diffusion distance)**.** *For $t \in \mathbb{N}$, the diffusion distance $D_t : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$D_t^2(x, y) \coloneqq \sum_{z \in \mathcal{X}} (P^t(x, z) - P^t(y, z))^2 \frac{1}{\pi(z)}$$

for all $x, y \in \mathcal{X}$, where $\pi(z)$ denotes the stationary distribution[1] of the Markov chain.

The diffusion distance $D_t(x, y)$ sums over all paths of length $2t$ connecting x to y. Thus, a small value for $D_t(x, y)$ corresponds with a high transition probability between x and y.
It is useful to rewrite the diffusion distance D_t by means of an eigendecomposition of P^t. A spectral analysis of the Markov chain is indeed possible under mild assumptions on K [Sch13]: There exists a symmetric matrix P_{sym}, given by

$$P_{sym}(x, y) \coloneqq P(x, y) \sqrt{\frac{D(x)}{D(y)}} = \frac{K(x, y)}{\sqrt{D(x) D(y)}} \tag{3.3}$$

sharing the same eigenvalues with P. Since P_{sym} is symmetric, P_{sym} has n real eigenvalues $\{\lambda_l\}$ with corresponding orthonormal eigenvectors $\{\phi_l\}$[2]. Moreover,

[1] The stationary distribution is the limiting distribution of the Markov chain, i.e. $\lim_{t \to \infty} P^t(x, y) = \pi(y)$. For existence, we can assume $K(x, y) > 0$ for all $x, y \in \mathcal{X}$. Then, π can be explicitly given by the left eigenvector of P corresponding to the eigenvalue 1, i.e. $\pi P = \pi$, normed with respect to the 1-norm. If the graph $(\mathcal{X}, K)$ is connected, the stationary distribution is unique.

[2] $\{\phi_l\}$ form an orthornormal basis with respect to the Euclidean inner product $\langle v, w \rangle_{\ell^2(\mathcal{X})} = \sum_{z \in \mathcal{X}} v(z) w(z)$.

if the graph $(\mathcal{X}, K)$ is connected[3], applying the Perron-Frobenius Theorem, we have for sorted eigenvalues:

$$1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n.$$

Note, that the connectivity of the graph depends on the kernel function K. For the Gaussian kernel, the graph is theoretically connected for all $\sigma > 0$. However, in practice, the bandwidth σ has to be chosen big enough.
The corresponding right and left eigenvectors $\{\psi_l\}$ and $\{\chi_l\}$ of P can be expressed in terms of the eigenvectors $\{\phi_l\}$ of P_{sym}:

$$\psi_l(x) = \frac{\phi_l(x)}{\sqrt{\pi(x)}}, \qquad \chi_l(x) = \phi_l(x)\sqrt{\pi(x)}.$$

Subsequently, the eigendecomposition of P^t is given by

$$P^t(x, y) = \sum_{l=1}^{n} \lambda_l^t \psi_l(x) \chi_l^T(y). \tag{3.4}$$

Using this representation of P^t, the diffusion distance can be rewritten[4] as

$$D_t^2(x, y) = \sum_{l=2}^{n} \lambda_l^{2t} (\psi_l(x) - \psi_l(y))^2.$$

Due to the fact, that the entries of the eigenvector ψ_1 corresponding to the eigenvalue $\lambda_1 = 1$ are all identical[5], the term for $l = 1$ is omitted in the sum.
As seen above, the eigenvalues $\{\lambda_l\}$ become smaller and smaller (they tend to zero with bigger n). Thus, the diffusion distance can be approximated by the first terms of the sum. For $s \leq n$, we therefore introduce the approximated diffusion distance

Definition 4.

$$D_{t,s}^2(x, y) \coloneqq \sum_{l=2}^{s} \lambda_l^{2t} (\psi_l(x) - \psi_l(y))^2$$

and the family of *diffusion maps* $\{\Psi_t\}_{t \in \mathbb{N}}$:

Definition 5 (Diffusion maps)**.** *For $s \leq n$, we define the family of diffusion maps $\{\Psi_t : \mathcal{X} \to \mathbb{R}^{s-1}\}_{t \in \mathbb{N}}$:*

$$\Psi_t(x) \coloneqq \begin{pmatrix} \lambda_2^t \psi_2(x) \\ \lambda_3^t \psi_3(x) \\ \vdots \\ \lambda_s^t \psi_s(x) \end{pmatrix}.$$

Each component $\lambda_l^t \psi_l$ is called diffusion coordinate.

[3] A graph is called connected, if there exists a path between any two vertices of the graph, this means that for all $x, y \in \mathcal{X}$ it holds $P^t(x, y) > 0$ for some t.

[4] Here we use, that $\{\chi_l\}$ form an orthonormal basis with respect to the inner product $\langle v, w \rangle_{\ell^2(\mathcal{X}, \frac{1}{\pi})} = \sum_{z \in \mathcal{X}} v(z) w(z) \frac{1}{\pi(z)}$.

[5] This follows by equation 3.2.

We now can connect the diffusion distance with the diffusion map.

Theorem 1. *For $s \leq n$, $t \in \mathbb{N}$ and $x, y \in \mathcal{X}$ we have*

$$D_{t,s}(x, y) = \|\Psi_t(x) - \Psi_t(y)\|_2.$$

Theorem 1 implies, that the diffusion distance D_t is equal to the Euclidean distance in the diffusion map space, up to a relative accuracy depending on s. Thus, the diffusion map Ψ_t is an embedding into the Euclidean space $\mathbb{R}^{s-1}$.

Anisotropic Diffusion

So far, we used so-called isotropic kernels such as the Gaussian kernel in definition 2. These can be good enough if the data points are drawn from a uniform data distribution. In this case, the eigenvectors of the transition matrix P discretely approximate the eigenvectors of the Laplace-Beltrami operator Δ and are therefore well suited to recover the manifold structure of the data. However, in most applications, the data is affected by distribution heterogeneities, such as dense or sparse regions. When using an isotropic kernel for non-uniform distributed data, the eigenvectors of P yield an approximation of the operator

$$\Delta - \frac{\Delta q}{q},$$

where q is the underlying probability density of the data. As a consequence, the data distribution in the non-uniform case, has a great influence on the embedding. To recover the manifold structure regardless of the data distribution, it is useful to consider anistropic kernels. To this end, let K be a given kernel, e.g. from definition 2. Using K, we define a new, generalized kernel

$$K^{(\alpha)}(x, y) \coloneqq \frac{K(x, y)}{D^\alpha(x) D^\alpha(y)},$$

for some $\alpha \in [0, 1]$, where D is defined by (3.1). Note, that for $\alpha = 0$ the new kernel corresponds with the old one. Now, we can proceed as above by setting

$$D^{(\alpha)}(x) \coloneqq \sum_{z \in \mathcal{X}} K^{(\alpha)}(x, z)$$

and defining the transition matrix

$$P^{(\alpha)}(x, y) \coloneqq \frac{K^{(\alpha)}(x, y)}{D^{(\alpha)}(x)}.$$

Coifman and Lafon showed in [CL06], that, using $\alpha = 1$, the eigenvectors of the transition matrix $P^{(1)}$ discretely approximate the eigenvectors of the Laplace-Beltrami operator, like in the case of isotropic kernels for uniform distributed data. This means, that setting $\alpha = 1$ removes all the influence of the data distribution and recovers the geometry of the data set, regardless of the distribution.
Finally, we give the complete diffusion maps algorithm:

Algorithm 1 Diffusion maps algorithm

Input: data $\mathcal{X}$, $\alpha \in [0,1]$, $s \leq n$
Output: embedding Ψ
$K \leftarrow [K(x_i, x_j)]_{i,j=1}^n$ with kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
$D \leftarrow diag(K\mathbb{1})$ with $\mathbb{1} = (1, \ldots, 1)$
$K^{(\alpha)} \leftarrow D^{-\alpha} K D^{-\alpha}$
$D^{(\alpha)} \leftarrow diag(K^{(\alpha)}\mathbb{1})$
$P^{(\alpha)} \leftarrow (D^{(\alpha)})^{-1} K^{(\alpha)}$
Compute the first s eigenvalues $\{\lambda_l\}_{l=1}^s$ and the corresponding eigenvectors $\{\psi_l\}_{l=1}^s$ of $P^{(\alpha)}$.
$\Psi \leftarrow \{\lambda_l \psi_l\}_{l=2}^s$

Note that the k-th coordinate of the l-th diffusion coordinate $\lambda_l \psi_l$ is the l-th coordinate of the k-th data point in the embedding space. From now on, we will set α to 1. For simplicity, we therefore write $P = P^{(1)}$. In the end, we give some final remarks on the diffusion maps algorithm:

- If the data is lying on a m-dimensional manifold, we need to set $s = m + 1$. For visualization, we need to assume m to be 2 or 3.
- Instead of using the eigenvectors of P, we could use the eigenvectors of the symmetric matrix P_{sym} for embedding, as well[6]. This has the advantage, that one can achieve a faster and more robust computation of the eigendecompositon of the transition matrix. Furthermore, a symmetric transition matrix can make sense in practical applications, where it is useful to have symmetric transition probabilities, i.e. $P(x, y) = P(y, x)$ [HBW+16].
- In [HBT15], it is proposed to set the diagonal of $K^{(\alpha)}$ to 0. In fact, for recovering the structure of the data set, based on relations between data points, it could be disturbing to have nonzero entries on the diagonal of the transition matrix. In this way, the relations between data points are better weighted in the embedding.

3.2.1 Bandwidth Selection

Using the Gaussian kernel $K_\sigma(x, y) = \exp(-\|x - y\|_2^2/(2\sigma^2))$ for diffusion maps, leads to the question of how selecting the bandwidth σ. In fact, parameter selection is commonly a challenging task in machine learning algorithms. We already stated, that for a robustly computable eigendecomposition, the bandwidth σ has to be chosen "big enough", in order to have the graph $(\mathcal{X}, K)$ connected. Indeed, if the bandwidth is too small, the most entries of the transition matrix P tend to zero, which can lead to slow or even absent convergence in the computation of the eigendecomposition. Moreover, if the transition probabilities are too small and in particular almost identical for all data points, the data points tend to accumulate in one place in the embedding space and the structure of the data can not be revealed.

[6]Note, that in the symmetric case, the rescaling parameter π^{-1} can be dropped in the definition of the diffusion distance.

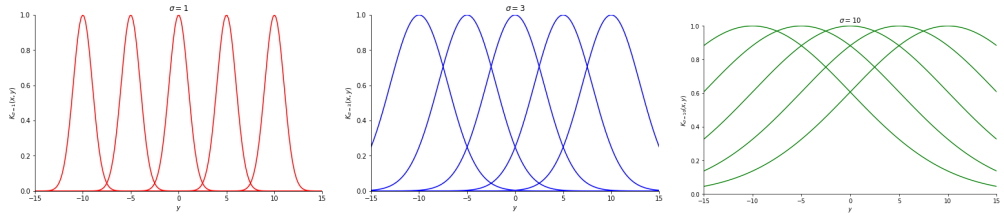


Figure 3.1: Gaussian kernel functions $K_\sigma(x, y) = exp(-(x-y)^2/(2\sigma^2))$ for bandwidths $\sigma = 1$, 3 and 10. With smaller σ, the kernel functions tend faster to zero outside their centering points, i.e. neighbouring points become less similar. The bandwidth σ has to be chosen in such a way, that the non-zero regions of the kernel functions $K(x, \cdot)$ for $x \in \mathcal{X}$ overlap in a reasonable fashion.

Surely, the term of a "big enough" bandwidth is too vague for our purpose. Choosing simply a very big bandwidth does not lead to an appropriate embedding, as well. More precise, if the bandwidth is too big, the transition probabilities become too sensitive to distances and the embedding is blurred.
Thus, a bandwidth, that is not too big and not too small, is aspired. One idea is to set $\epsilon := 2\sigma^2$ to the smallest value, such that the underlying graph is connected (enough). The figure 3.1 illustrates what connectivity means: The non-zero regions of the kernel functions $K(x, \cdot)$ for $x \in \mathcal{X}$ should overlap in a reasonable fashion. Data sets with bigger distances between points require consequently larger bandwidths. Lafon suggested in [Laf04] to take the average of all (1-)nearest neighbour distances in the data set:

$$\epsilon_{Laf} := \frac{1}{n} \sum_{i=1}^{n} \min_{j \in \{1,\dots,n\} \setminus \{i\}} \{\|x_i - x_j\|_2^2\}. \tag{3.5}$$

Such a rule has the advantage, that it is fast and easy to compute and give good intuitions about the size of the optimal parameter. However, in the case of missing values for instance (as we will see in chapter 5), we need other methods, which are independent of measuring distances. To this end, we present an approach suggested by [HBT15]:
We consider

$$D_\sigma(x) := \sum_{z \in \mathcal{X}} K_\sigma(x, z)$$

for $x \in \mathcal{X}$. $D_\sigma(x)$ is a quantity for measuring the amount of data points accessible from point x. The average of the logarithmic values

$$\bar{D}(\sigma) := \frac{1}{n} \sum_{x \in \mathcal{X}} \log D_\sigma(x)$$

is then proportional to the average size of the neighbourhood of a data point in $\mathcal{X}$ with respect to σ. Considering the behaviour of $\bar{D}(\sigma)$ against $\log \sigma$, the bandwidth can be chosen as the parameter σ with the highest slope of $\bar{D}(\sigma)$ (see figure 3.2). This point can be interpreted as the first point, where the connectivity of all data points in $\mathcal{X}$ is reached.

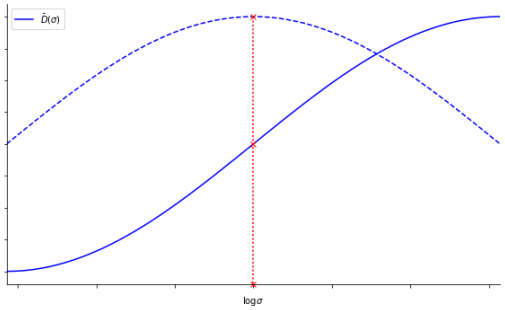


Figure 3.2: Illustration of the bandwidth selection technique, proposed in [HBT15]. The bandwidth is selected as the parameter with the highest slope of $\bar{D}(\sigma)$ (——). The highest slope can be determined by means of the first derivative of $\bar{D}(\sigma)$ (- - -), namely by determining the maximum of the derivative.

In order to take density heterogeneities into consideration, Haghverdi et al. propose in [HBT15] a normalized version of $\bar{D}(\sigma)$:

$$\bar{D}_{norm}(\sigma) := \frac{\frac{1}{n}\sum_{x\in\mathcal{X}} \frac{\log D_\sigma(x)}{D_\sigma(x)}}{\sum_{x\in\mathcal{X}} \frac{1}{D_\sigma(x)}}.$$

In order to compute the optimal parameter $\hat{\sigma}$, $\bar{D}_{norm}(\sigma)$ is evaluated for a discrete set of parameters $\{\sigma_k\}_{k=1}^K$ (sorted in ascending order) for $K \in \mathbb{N}$ (e.g. $K = 10$). The slope of $\bar{D}_{norm}(\sigma_k)$ can be computed by difference quotients. Hence, the optimal parameter is approximated by

$$\hat{\sigma} \approx \underset{\sigma_k : 1\le k\le K-1}{\arg\max} \frac{\bar{D}_{norm}(\sigma_{k+1}) - \bar{D}_{norm}(\sigma_k)}{\sigma_{k+1} - \sigma_k}.$$

3.2.2 Gaussian Kernel Selection

So far, we fixed a global bandwidth for all data points. This choice can be appropriate for some data sets. However, when data sets contain sparse data regions or high density dissimilarities, a global bandwidth could be not enough anymore to reveal the data structure. Instead of a global bandwidth for all data points, one can use local kernel widths σ_x for each cell $x \in \mathcal{X}$ [HBW+16]. To this end, we interpret the Gaussian kernel $K_\sigma(x, y) = \exp(-\|x - y\|_2^2/(2\sigma^2))$ from definition 2 as an interference of two Gaussian wave functions

$$K_\sigma(x, y) = \int_{-\infty}^{\infty} Y_x(z)Y_y(z)dz \tag{3.6}$$

with

$$Y_x(z) := \left(\frac{2}{\pi\sigma^2}\right)^{1/4} \exp\left(-\frac{\|z - x\|_2^2}{\sigma^2}\right) \tag{3.7}$$

fulfilling the normalization $\int_{-\infty}^{\infty} Y_x^2(z)dz = 1$. Replacing the global bandwidth σ by a local bandwidth σ_x for $x \in \mathcal{X}$ in 3.7 yields a new locally-scaled Gaussian kernel:

Definition 6 (Local Gaussian kernel)**.** *Let $k \in \mathbb{N}$. The local Gaussian kernel $K_k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$K_k(x, y) := \left(\frac{2\sigma_x \sigma_y}{\sigma_x^2 + \sigma_y^2} \right)^{1/2} \exp\left(-\frac{\|x - y\|_2^2}{\sigma_x^2 + \sigma_y^2} \right),$$

where for each $x \in \mathcal{X}$, the squared local bandwidth σ_x^2 is chosen to be the half of the k-th nearest neighbour squared distance of data point x.

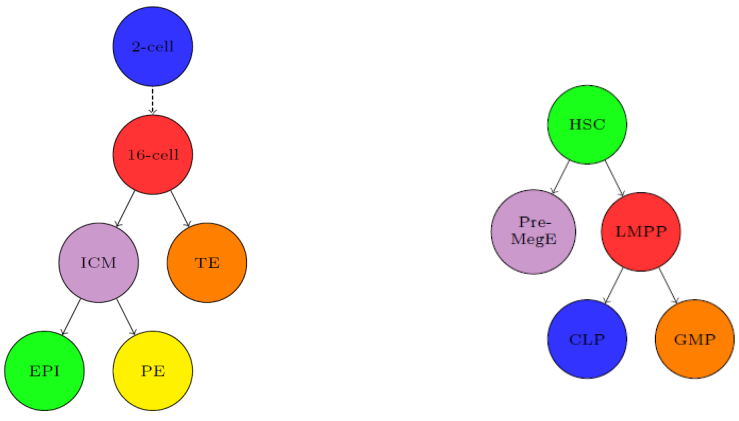
A further useful kernel is

$$K_{k,a}(x, y) := \frac{1}{2} \exp\left(-\left(\frac{\|x - y\|_2}{\sqrt{2}\sigma_x} \right)^a \right) + \frac{1}{2} \exp\left(-\left(\frac{\|x - y\|_2}{\sqrt{2}\sigma_y} \right)^a \right) \tag{3.8}$$

where the additional parameter a controls the rate of decay of the kernel [MvDW+17].

3.3 Experiments

The aim and challenge of data visualization is the depiction of *relevant* information. For single-cell data, relevant information primarily includes cell lineages; this means we strive to visualize the cell differentiation development, described by the data. In figure 3.3, you can see two examples for cell differentiation structures we strive to depict using diffusion maps embedding, one of the *Guo* data and one of the *Moignard* data. The circles represent cell stages in the course of the differentiation progress, which can be seen as branches of a tree, similar to a genealogical tree.
Theoretically, a visualization of such trees is possible in 2 dimensions. Therefore, we focus on a 2-dimensional representation of the data in our experiments. In order to embed the data into a 2-dimensional space, the first embedding vector is mapped against the second one. In the case of diffusion maps, the embedding vectors are the diffusion coordinates $\lambda_2\psi_2$ and $\lambda_3\psi_3$. Note, that the multiplication with the eigenvalues λ_2 and λ_3 is only a scaling factor and does not influence the visualization.



(a) *Guo* data (b) *Moignard* data

Figure 3.3: Cell differentiation development of the *Guo* and the *Moignard* data.

Comparison to other Dimensionality Reduction Methods

In comparison to other dimensionality reduction methods, diffusion maps prove to be an appropriate dimensionality reduction method revealing relevant information of biological data. We compare the diffusion maps embedding in 2 dimensions to five other dimensionality reduction methods, namely PCA, ISOMAP, LE, LLE and tSNE. For diffusion maps, we use the Gaussian kernel with a global bandwidth σ, chosen by the rule of Lafon in 3.5. Considering the *Guo* data in figure 3.4 as an example, diffusion maps succesfully reveal the four lineages, including the two subbranches PE and EPI, and the relations of the lineages to each other. In the PCA and LE embedding, the subbranches PE and EPI coincide in a region and LLE completely fails to reveal any structure. Comparing, in particular, diffusion maps with the PCA embedding, one can observe the denoising effect of the diffusion maps method. This can be observed for other data sets, as well (cf. appendix A.2.1). Considering the ISOMAP and tSNE embedding, it is difficult to recognize the relations of the lineages[7]. Especially, tSNE displays the lineages as clear separable clusters without temporal order[8]. In contrast to that, diffusion maps point out the temporal development of the cells from the 2-cell to the 16-cell stage and the ensuing separation into the three different differentiation states.

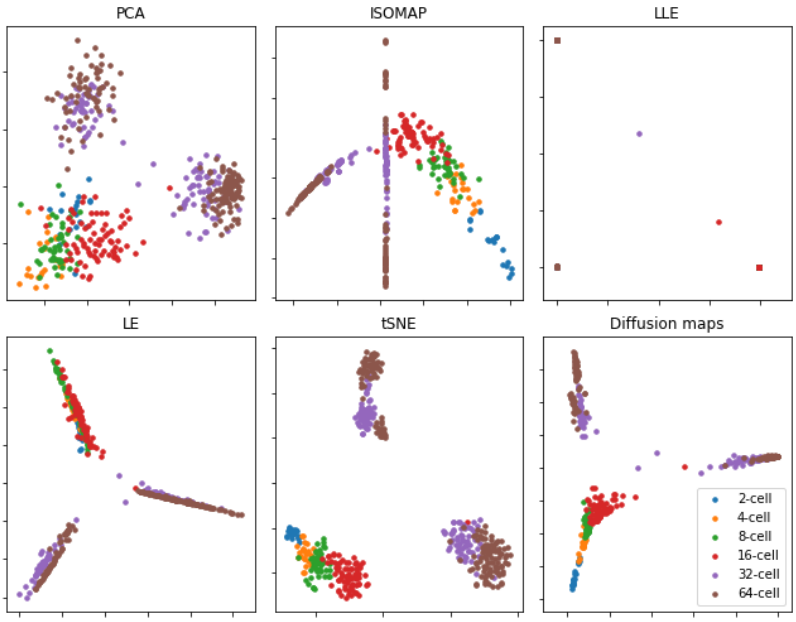


Figure 3.4: Embedding in 2 dimensions of the (normalized) *Guo* data set by means of six different dimensionality reduction methods.

[7]Notice, that ISOMAP would show a more understandable picture, if we do a 3-dimensional visualization of the data.

[8]However, for clustering the lineages, tSNE appears here to be the better choice.

Bandwidth Selection

With the *Guo* data, we saw a data set, where diffusion maps revealed all relevant information using the Gaussian kernel with a bandwidth chosen by the rule of Lafon. However, in general, it is difficult to concentrate all important information of the data in only two diffusion components. In figure 3.5, several 2-dimensional plots and a 3-dimensional plot for the *Moignard* data is presented, using the first three diffusion components. Comparing the pictures with figure 3.3, showing the desired cell differentiation development of the *Moignard* data, one can observe, that the CLP lineage can only be revealed using the third diffusion component, this means, that a 2-dimensional visualization using only the first two diffusion components is not enough for revealing all desired information.
In order to achieve an appropriate visualization in 2 dimensions, the bandwidth σ in the Gaussian kernel needs to be appropriately selected. It plays an important role in the resulting embedding. The figure 3.6 demonstrates the influence of the bandwidth σ on the resulting visualization using the example of the *Guo* data set. If the bandwidth is too small, the data points accumulate in a few places, such that we cannot see any structure. If the bandwidth is too big, the data points are rather blurred and noisy, such that lineages like the PE and EPI lineage are harder to recognize, as well. Hence, a bandwidth in the middle range is preferable, as already considered in theory.
A useful criterion for assessing how much information is hidden in a diffusion

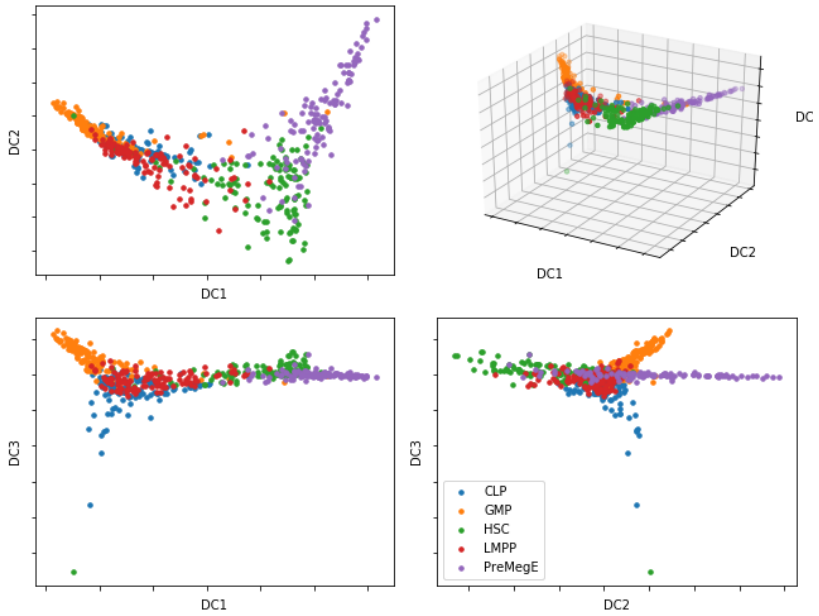


Figure 3.5: Diffusion maps embedding of the (normalized) *Moignard* data set using the Gaussian kernel with global bandwidth σ chosen by the rule of Lafon. At the top, a 2- and 3-dimensional visualization of the data is presented, where the first 2 or 3 diffusion components are mapped against each other. At the bottom, 2-dimensional plots can be seen, where the first and the second diffusion component are each mapped against the third.

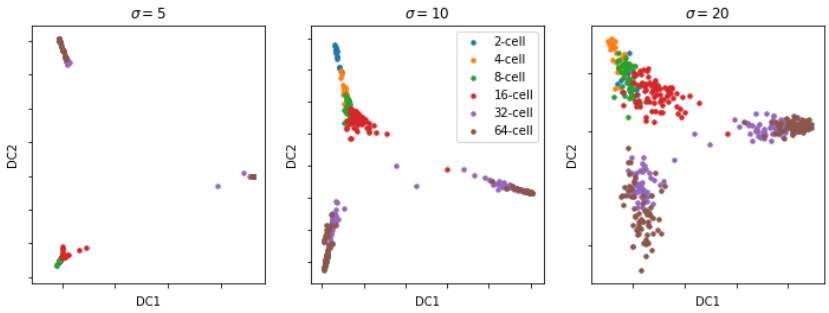


Figure 3.6: Comparison of different bandwidths ($\sigma = 5, 10$ and 20) for diffusion maps embedding of the *Guo* data in 2 dimensions.

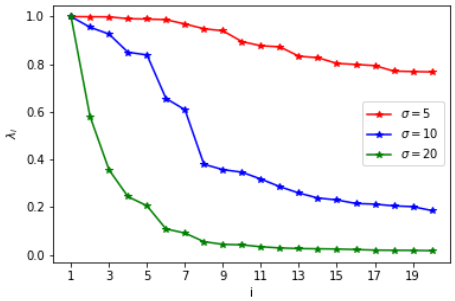


Figure 3.7: The first 20 eigenvalues of the transition matrix of the diffusion maps algorithm for the *Guo* data, using different bandwidths.

component is the size of the corresponding eigenvalue. The larger the eigenvalue, the more information the corresponding diffusion component contains. Hence, for a 2-dimensional visualization, we aim to have the first two non-trivial eigenvalues of the transition matrix to be large and the other eigenvalues as small as possible. Thus, the diffusion components with corresponding large eigenvalues contain all important information of the data and the diffusion components with corresponding small eigenvalues represent data noise, i.e. they do not include valueable information. In figure 3.7, the first 20 eigenvalues of the transition matrix of the *Guo* data are plotted, using three different bandwidths as in figure 3.6. For a small bandwidth, here for $\sigma = 5$, all eigenvalues are rather large, this means that the information is spread over many diffusion components so that a useful visualization of the data is not possible (cf. figure 3.6). If the bandwidth is too large, namely $\sigma = 20$ for the *Guo* data, the eigenvalues tend fast to zero; especially those of the first two non-trivial diffusion components are comparatively small. This means that we lost relevant information in the course of the embedding process. A bandwidth of $\sigma = 10$ gives the most appropriate visualization. Considering the course of the corresponding first 20 eigenvalues in figure 3.7, one can observe several jumps, particularly after the third eigenvalue. These jumps indicate that the corresponding next diffusion components carry significantly less information than those with the bigger eigenvalues. Thus, the

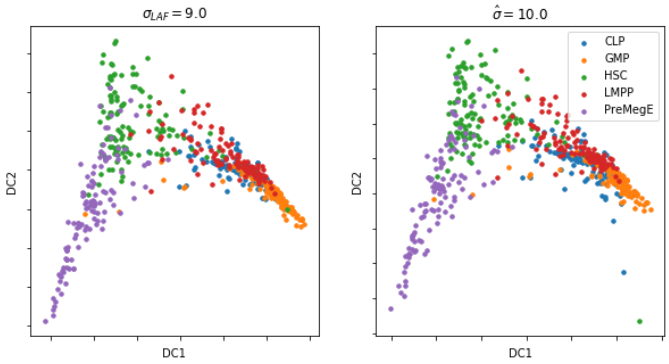


Figure 3.8: Diffusion maps embedding in 2 dimensions for the *Moignard* data, using Lafon's rule (left) and the selection technique by [HBT15] (right) for the bandwidth σ.

most important information is bundled in the first two diffusion components[9].
For bandwidth selection, we introduced two different techniques: The rule by Lafon in 3.5, denoted by σ_{LAF}, and the selection technique by [HBT15], which we indicate with $\hat{\sigma}$. The rule by Lafon has the great advantage that the bandwidth can be easy and directly determined, supposing we can compute distances between data points. However, in the case of censoring values, distances between censored data need to be defined first before calculating the rule by Lafon (cf. chapter 5). In particular, the rule by Lafon is only applicable to the Gaussian kernel in definition 2.
The selection technique by [HBT15] can be theoretically applied to general kernel functions depending on a global parameter. Particularly, we do not need to define a distance measure on the data set, which is useful for handling censored values. However, this selection technique finds the best parameter among a set of bandwidth candidates, which need to be chosen by the user. This means, the user need to have an idea about the approximate size of the best bandwidth. In fact, for some data sets, we had to re-select the candidates several times to find an appropriate parameter. In contrast to that, Lafon's rule suggested adequate bandwidths for almost all data sets.
For the *Moignard* data, the bandwidth chosen by the selection technique by [HBT15] results in a more suitable 2-dimensional data visualization than by Lafon's rule. We used 19 different bandwidths to compute the optimal parameter $\hat{\sigma}$. Comparing the two visualizations in figure 3.8, one can observe, that with the selection technique by [HBT15], the CLP lineage becomes visible, in contrast to the visualization with Lafon's rule. However, for the most other data sets, both selection techniques result in similar visualizations (cf. appendix A.2.2).

[9]Note, that the third to sixth diffusion components contain further subordinate information, which will be useful for spectral clustering.

Runtime Analysis

The computation time of diffusion maps depends on two components, the number of cells, denoted by n, and the number of genes which is d. The genes are only used for the calculation of the pairwise cell distances. For these, we have to perform a total of $n^2 d$ computations. Having the distances, the remaining construction of the transition matrix can be done in $\mathcal{O}(n^2)$. The algorithm is dominated by the final computation of the first eigenvalues and eigenvectors with a worst-case complexity of $\mathcal{O}(n^3)$. However, in practice, the worst-case scenario is usually not achieved.

	σ_{LAF}	$\hat{\sigma}$
Guo (428 cells × 48 genes)	0,03	0,91
Moignard (597 cells × 21 genes)	0,07	1,06
Goettgens (3934 cells × 42 genes)	2,62	15,72
Yan (90 cells × 20214 genes)	0,43	1,19
Mass (408 cells × 8657 genes)	2,25	3,11
Klein (2717 cells × 2047 genes)	23,42	30,77
Paul (2730 cells × 3451 genes)	39,17	46,33

Figure 3.9: Runtime analysis of the diffusion maps algorithm using two different bandwidth selection techniques (in seconds).

In figure 3.9, the computation times for the diffusion maps embedding in three dimensions using the two different bandwidth selection techniques for all biological data sets are listed. For each embedding, we took the smallest time of several runs. For the selection of $\hat{\sigma}$, we used a total of 20 different bandwidths. As expected, the selection technique by [HBT15] needs more computation time than the rule of Lafon. However, in order to choose $\hat{\sigma}$, the distances have to be calculated only once for all bandwidth candidates. Therefore, the runtime of data sets with a large number of genes, but a small number of cells, such as *Yan* or *Mass*, is comparatively small. Moreover, the calculation times increase with growing number of cells and genes. In fact, the amount of cells seems to have a stronger impact on the runtime as one can see in comparing *Guo* and *Moignard* or *Goettgens* and *Mass*. However, no computation exceeds one minute.

Local Gaussian Kernels

So far, we used the classical Gaussian kernel with a global bandwidth. However, for some data sets, a global bandwidth is not enough to reveal the structure of the data. Particularly, in the case of the *Klein* data set, the Gaussian kernel with global bandwidth, using any selection technique, seems not to be appropriate, as can be seen in figure 3.10.
Instead of a global bandwidth, Gaussian kernels with local bandwidths, which can be computed by means of the k-th nearest neighbour distances of the data points with $k \in \mathbb{N}$, can be useful in these cases. We introduced a generalized local Gaussian kernel (definition 6), denoted by K_k depending on $k \in \mathbb{N}$ and a kernel $K_{k,a}$ in 3.8, depending additionaly on a parameter a. In [HBW$^+$16] and [MvDW$^+$17], it is proposed to set $k = 5$ and $a = 10$. In fact, it is useful

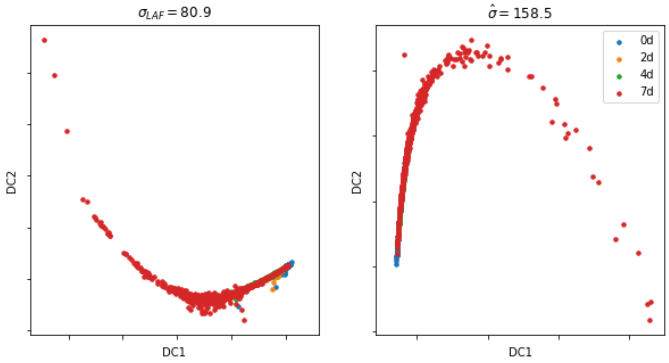


Figure 3.10: Diffusion maps embedding in 2 dimensions for the *Klein* data, using Lafon's rule (left) and the selection technique by [HBT15] (right) for the bandwidth σ.

to choose k not too high and not too low, similar to the bandwidth selection. Furthermore, choosing the parameter a high enough, guarantees to account for sparse or dense data regions. However, a rule for selecting these two parameters is difficult to find. In figure 3.11, a 2- and 3-dimensional visualization of the *Klein* data, using the local Gaussian kernel K_k with $k = 5$ is presented. In comparison with the results using global bandwidths in figure 3.10, the second branch, dominated by cells at 0 and 2 days after LIF withdrawal, can be seen more clearly in the 2-dimensional visualization using the first two diffusion components. Nevertheless, the third diffusion component carries the most information for this branch, as one can observe in the lower left picture of figure 3.11. Taking the kernel $K_{k,a}$ with $k = 5$ and $a = 10$ instead, both branches can be visualized in 2 dimensions using only the first two diffusion components. Considering figure 3.12, we can observe that the information for the second branch is carried by the second diffusion component whereas the third diffusion component does not represent important information anymore.
Furthermore, we detected an improvement of the visualization using local Gaussian kernels in the case of the *Paul* data, as well (cf. appendix A.2.3).

Conclusion

In summary, we see a strong dependance of the choice of the kernel function on the resulting visualization. However, we found, that taking a Gaussian kernel with global bandwidth selected by Lafon's rule achieves an appropriate overall picture of the data set in most cases. In some cases, it is useful to select other parameters or kernels with local bandwidths, to transport relevant information from the third (and more) diffusion components to the first two components for a 2-dimensional visualization. Naturally, a 3-dimensional visualization of biological data is possible and useful, as well. The size of the eigenvalues as a criterion for the information content of the diffusion components will be particularly relevant in the following chapter.

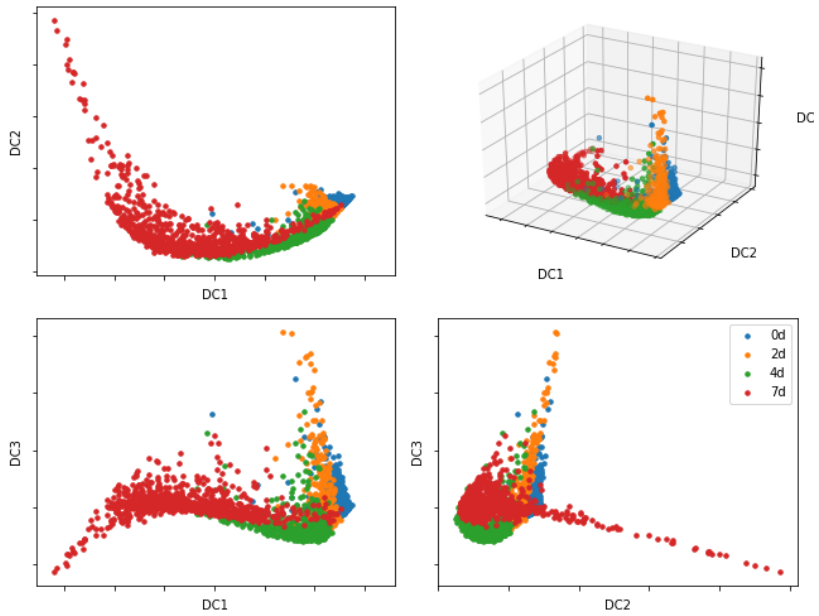


Figure 3.11: Diffusion maps embedding for the *Klein* data, using the local Gaussian kernel K_k with $k = 5$.

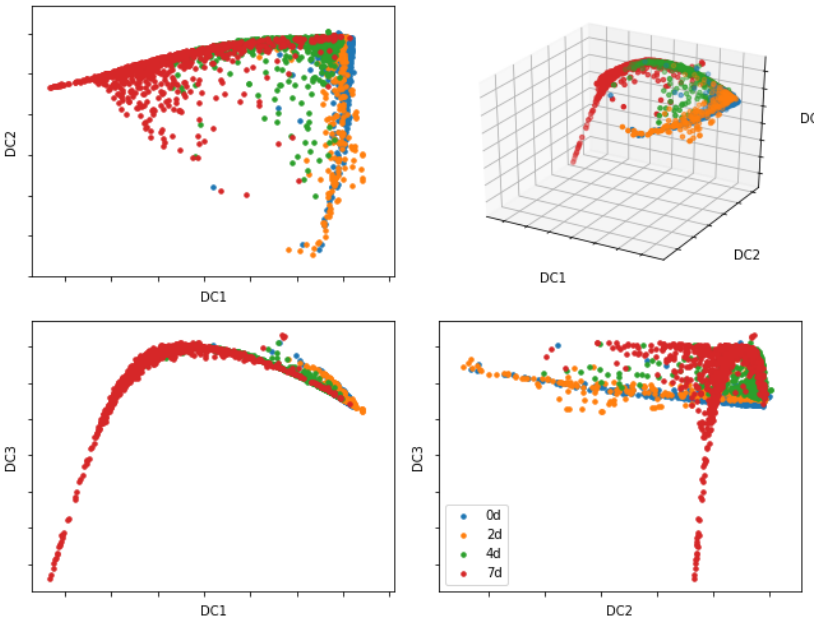


Figure 3.12: Diffusion maps embedding for the *Klein* data, using the kernel $K_{k,a}$ with $k = 5$ and $a = 10$.

4 Group Detection

Apart from data visualization, biologists aim to detect cell groups in the data. In single-cell differentiation data, these cell groups are developmental cell stages during the differentiation process, which can be imagined as branches in a genealogical tree. In the following, we will present two different approaches for cell group detection using diffusion maps as preprocessing step.

4.1 Spectral Clustering

The goal of clustering is to partition a data set $\mathcal{X}$ into groups, the so-called *clusters*, based on a notion of similarity (or dissimilarity) measure on the points [HTF09]. Data points within one cluster should be more similar to each other than to data points within another cluster. The most popular clustering algorithm is k-means [HTF09]. Choosing k cluster centers at the beginning, the sum of squared Euclidean distances of all data points to their closest cluster center is minimized. Iteratively, k-means determines for each data point the closest cluster center and updates each center by taking the average of all data points closest to the old one. The iteration is repeated until the centers no longer change (up to an accuracy).
Another clustering technique is the so-called *hierarchical clustering* [HTF09]: The clusters are gradually built up by defining at first each data point as its own cluster and then merging similar clusters to new bigger clusters or reversely, defining the whole data set as one cluster at the beginning and split it afterwards iteratively into smaller clusters.
Finally, another approach, the density-based clustering, finds clusters as areas of high data density. The best known method for this is DBSCAN (*density-based spatial clustering of applications with noise*) [EKSX96]. In contrast to k-means, it does not need the number of clusters as input.
Let now $G = (\mathcal{X}, W)$ be an undirected *similarity graph* of the data set $\mathcal{X}$ whose data points being the vertices of G. The matrix $W = (W_{ij})_{i,j=1,\dots,n}$ is the weighted adjacency matrix of the graph, i.e. each edge (x_i, x_j) is weighted by a non-negative weight $W_{ij} \geq 0$ [1] representing the similarity of the vertices x_i and x_j. Since G is undirected, the matrix W is symmetric. Clustering the data set $\mathcal{X}$ can now be viewed as partitioning the graph G such that edges within a group have high weight and edges between groups have low weight. Setting

$$D_{ij} \coloneqq \begin{cases} \sum_{l=1}^{n} W_{il} & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i, j \in \{1, \dots, n\}$, we define the *unnormalized graph Laplacian* as $L \coloneqq D - W$. We can now use the graph Laplacian for clustering the data. This leads to the so-called *spectral clustering* algorithm [vL07]:

[1] $W_{ij} = 0$ means that the vertices x_i and x_j are not connected by an edge.

Algorithm 2 Classical spectral clustering algorithm

Input: the graph Laplacian L, number k of clusters
Output: clusters $C_1, \dots, C_k$
Compute k eigenvectors $\{u_l\}_{l=1}^k$ corresponding to the k smallest eigenvalues of L.
$U \leftarrow [u_1, \dots, u_k]$
for $l = 1, \dots, n$ **do**
    $y_l \leftarrow l$-th row of U
end for
Extract clusters $C_1, \dots, C_k$ from $\{y_l\}_{l=1}^n$ in $\mathbb{R}^k$ (e.g. with the k-means algorithm).

Instead of the unnormalized graph Laplacian L, it is possible to use the symmetric *normalized graph Laplacian*

$$L_{sym} \coloneqq I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

or the *random walk normalized graph Laplacian*

$$L_{rw} \coloneqq I - D^{-1} W = D^{-1} L$$

in the spectral clustering algorithm, as well. The transition matrix P in diffusion maps is highly related to the random walk normalized graph Laplacian L_{rw}: Using a kernel function K for the weights W_{ij}, we get

$$P = D^{-1} W = 1 - L_{rw}.$$

Consequently, u is an eigenvector of L_{rw} corresponding to the eigenvalue λ if and only if u is an eigenvector of P corresponding to the eigenvalue $1 - \lambda$. Thus, the spectral clustering algorithm using diffusion maps is given as follows:

Algorithm 3 Spectral clustering algorithm with diffusion maps

Input: transition matrix P computed in algorithm 1, number k of clusters
Output: clusters $C_1, \dots, C_k$
Compute k eigenvectors $\{u_l\}_{l=1}^k$ corresponding to the k largest eigenvalues of P.
$U \leftarrow [u_2, \dots, u_k]$
for $l = 1, \dots, n$ **do**
    $y_l \leftarrow l$-th row of U
end for
Extract clusters $C_1, \dots, C_k$ from $\{y_l\}_{l=1}^n$ in $\mathbb{R}^{k-1}$ (e.g. with the k-means algorithm).

In order to choose an appropriate number k of clusters, one can use the following approach [BK17]: For some not too large M, we compute the M largest eigenvalues $\{\lambda_l\}_{l=1}^M$ of P. Then, we set k to the index, where $\lambda_k - \lambda_{k+1}$ is large. This is known as the *spectral gap* indicating how many eigenvectors we need to represent the main information of the data. Note that, as in the diffusion maps algorithm,

the first trivial eigenvector is not considered in the cluster analysis. This means, that for k clusters we consider $k-1$ eigenvectors. Another possibility is to use clustering algorithms (e.g. DBSCAN), which do not need the number of clusters as input, but which find k during the clustering procedure.

4.2 Diffusion Pseudotime Analysis

In the following, we will present and extend another approach for group detection using the so-called *diffusion pseudotime* [HBW+16]. Let P be the transition matrix from the diffusion maps algorithm. For a time parameter $t \in \mathbb{N}$, the power P^t describes a random walk of length t on the data set. Summing over all powers

$$\sum_{t=1}^{\infty} P^t \tag{4.1}$$

hence contains information about the time development of the diffusion process. The following theorem shows when this series converges:

Theorem 2. *For a matrix A, the geometric series $\sum_{t=0}^{\infty} A^t$ converges if and only if $\rho(A) < 1$, i.e. $|\lambda| < 1$ for all eigenvalues λ of A. If $\rho(A) < 1$, the series converges to $(I-A)^{-1}$.*

Since P is a stochastic matrix, the largest eigenvalue is 1, this means $\rho(P) = 1$. Thus, theorem 2 implies that the series 4.1 does not converge. To make the series converge, we need to remove the part which causes the non-convergence, namely the eigenspace of the eigenvalue 1. Considering the eigendecomposition of P in 3.4, it is clear, that we need to remove the first term of the sum, namely $\psi_1 \chi_1^T$. Thus, we define the new matrix

$$M := \sum_{t=1}^{\infty} (P - \psi_1 \chi_1^T)^t = (I - (P - \psi_1 \chi_1^T))^{-1} - I. \tag{4.2}$$

Analogously to the diffusion distance, we now define the *diffusion pseudotime* (DPT) distance measure:

Definition 7 (Diffusion pseudotime distance)**.** *For a transition matrix P describing a Markov chain, let M be given as in 4.2. Then, the diffusion pseudotime distance $dpt : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$dpt^2(x,y) := \sum_{z \in \mathcal{X}} (M(x,z) - M(y,z))^2 \frac{1}{\pi(z)}$$

for all $x, y \in \mathcal{X}$, where $\pi(x)$ denotes the stationary distribution of the Markov chain.

As in the diffusion maps theory, we can rewrite[2] the diffusion pseudotime distance as

$$dpt^2(x,y) = \sum_{l=2}^{n} \left(\frac{\lambda_l}{1-\lambda_l} \right)^2 (\psi_l(x) - \psi_l(y))^2. \tag{4.3}$$

[2] Using the eigendecomposition of P, one can show, that the eigendecomposition of M is given by $M(x,y) = \sum_{l=2}^{n} \frac{\lambda_l}{1-\lambda_l} \psi_l(x) \chi_l^T(y)$.

In contrast to the diffusion distance D_t summing over random walks of a specific length depending on t, the DPT distance considers random walks of arbitrary length. Therefore, the DPT measure better captures the temporal distance between two points in the random walk setup. Since the eigenvalues $\{\lambda_l\}$ tend to zero, the factors $\{\lambda_l/(1-\lambda_l)\}$ tend to zero, as well. Thus, in order to compute the diffusion pseudotime distance, it can be approximated, similar to the diffusion distance D_t in diffusion maps, by the first terms of the sum.
Using this, the idea of diffusion pseudotime analysis is to identify branches in the (biological) data set. A branch is a set of points, lying on one straight line (or submanifold) on the manifold. In contrast to clear saparable clusters, branches are usually linked to each other. This makes it harder for clustering algorithms to separate the branches from each other. DPT analysis uses the idea, that a branch describes a straight temporal development with a start and ending state.
We assume at the beginning, that we have only three branches to identify. First of all, we aim to determine the endpoints or tips of the branches, which indicate the start or the end of a differentiation process in biological applications. Defining a specific data point $r \in \mathcal{X}$ as a root point, the diffusion pseudotime $dpt(r,x)$ indicates how long it takes, starting in root r, to reach the point $x \in \mathcal{X}$ in the random walk. Maximizing $dpt(r,x)$ with respect to $x \in \mathcal{X}$ gives the data point, which is the most distant from r. This point, denoted by r_1, can be identified as the tip of a branch. In order to get the tip r_2 of a second branch, we can now maximizing the DPT distance $dpt(r_1,x)$ of the first tip with respect to x. Having the first two tips r_1 and r_2, it is possible to determine a third tip by means of the triangle inequality

$$dpt(r_1,r_2) \leq dpt(r_1,x) + dpt(x,r_2). \tag{4.4}$$

Maximizing the length of the path from r_1 to r_2 via point x leads to the endpoint of a third branch:

$$r_3 = \underset{x\in\mathcal{X}\setminus\{r_1,r_2\}}{\arg\max}\ dpt(r_1,x) + dpt(x,r_2) = \underset{x\in\mathcal{X}\setminus\{r_1,r_2\}}{\arg\max} \sum_{j=1}^{2} dpt(r_j,x)^3.$$

Having the endpoint of a branch, we aim to determine the other end of the branch, as well. We call this end a starting point[4] or a branching point. It is usually at a fork of several branches. All the data points, which are on the path between endpoint and branching point, can then be assigned to this branch.
For this, we first order the data set $\mathcal{X}$ for each $i \in \{1,2,3\}$ according to the DPT distance with respect to tip r_i. In mathematical notation, we define the ordering

$$\mathcal{X}^{(i)} \coloneqq [x_1^{(i)}, \ldots, x_n^{(i)}] \tag{4.5}$$

with $x_j^{(i)} \in \mathcal{X}$ such that

$$dpt(r_i, x_j^{(i)}) \leq dpt(r_i, x_l^{(i)})$$

[3]For the last equation, we used the symmetry of the DPT distance.

[4]The name of a starting point or endpoint of a branch is confusing here. In fact, we do not know where the differentiation process starts and where it ends by considering solely the data. A temporal sorting of the cells in the correct direction is not possible via a data visualization or a DPT analysis. However, we usually have preliminary information about the developmental stages of the cells, so that we can correctly date the two ends of the branches, afterwards.

for all $j \leq l$, $j, l \in \{1, \ldots, n\}$[5]. Then, for each $j \in \{1,2,3\} \setminus \{i\}$, we define a vector of DPT distances with respect to r_j, sorted according to the ordering $\mathcal{X}^{(i)}$:

$$O_{i,j} \coloneqq [dpt(r_j, x_1^{(i)}), \ldots, dpt(r_j, x_n^{(i)})].$$

In order to identify a starting point of a branch i, we make use of the correlations between the pseudotime orderings $O_{i,j}$ of the other two branches $j \neq i$: The orderings of the two branches will only correlate on the third branch i. As correlation measure we use *Kendall's tau*:

Definition 8 (Kendall's tau[6])**.** *Let $(x, y) = \{(x_i, y_i)\}_{i=1}^n$ a set of pairs with $x_1 \leq x_2 \leq \cdots \leq x_n$ be given. Let*

- *C be the number of concordant pairs, i.e. $x_i < x_j$ and $y_i < y_j$,*
- *D be the number of discordant pairs, i.e. $x_i < x_j$ and $y_i > y_j$,*
- *T_Y be the number of ties in y, i.e. $x_i < x_j$ and $y_i = y_j$ and*
- *T_X be the number of ties in x, i.e. $x_i = x_j$ and $y_i \neq y_j$*

for $j \in \{i+1, \ldots, n\}$, $i \in \{1, \ldots, n-1\}$[7]. Then, Kendall's tau $\tau \in [-1, 1]$ is defined by

$$\tau(x, y) \coloneqq \frac{C - D}{\sqrt{(C + D + T_X)(C + D + T_Y)}}.$$

For $j_1, j_2 \in \{1,2,3\} \setminus \{i\}$, $j_1 \neq j_2$ and $x \in \mathcal{X}$ we define

$$\begin{aligned} C_{j_1,j_2}^{(i)}(x) \coloneqq \tau &\left(O_{i,j_1}[1 : ind^{(i)}(x)], O_{i,j_2}[1 : ind^{(i)}(x)]\right) \\ -\tau &\left(O_{i,j_1}[ind^{(i)}(x) + 1 : n], O_{i,j_2}[ind^{(i)}(x) + 1 : n]\right), \end{aligned} \tag{4.6}$$

where $ind^{(i)}(x)$ denotes the index of x in the vector $\mathcal{X}^{(i)}$[8]. The starting point c_i of branch i can then be determined by

$$c_i = \arg\max_{x \in \mathcal{X}} C_{j_1,j_2}^{(i)}(x)^{9}.$$

Finally, having branching and ending point of the branch i, the branch can be given by the points

$$\begin{aligned} B_i &= \{x \in \mathcal{X} | dpt(r_i, x) \leq dpt(r_i, c_i)\} \\ &= \mathcal{X}^{(i)}[1 : ind^{(i)}(c_i)]. \end{aligned}$$

Having determined all three branches, there are surely data points, which remain unassigned. These data points are denoted in [HBW+16] as *undecided*, which, in biological applications, are cells nearing a decision for a differentiation path.

[5] Note that, by definition of $\mathcal{X}^{(i)}$, it holds for the first entry $x_1^{(i)} = r_i$.

[6] This is the tau-b-version of Kendall's tau which accounts for ties.

[7] The last case, where a tie occurs in x and in y, i.e. $x_i = x_j$ and $y_i = y_j$, is not accounted in τ.

[8] The notation $O_{i,j}[k_1 : k_2]$ means, that we take the k_1-th entry up to the k_2-th entry of the vector $O_{i,j}$.

[9] Note that for branch i, j_1 and j_2 are uniquely specified in the case of three branches, apart from transposition.

4.2.1 Direct Extension

In order to separate more than three branches, Haghverdi et al. in [HBW+16] propose to determine iteratively subbranches of already found branches. However, this approach has one decisive disadvantage: The user has to choose manually which branches to separate further. This means, that the user needs information on the number and the location of meaningful subbranches. We strive to alter the DPT analysis in such a way, that we can directly determine $k \geq 3$ branches. To this end, we need first to extend the approach for finding endpoints of more than three branches, i.e. we search for tips $r_1, \ldots, r_k$ for $k \geq 3$. Let the first tips r_1 and r_2 computed as above. To determine tip r_i for $i \geq 3$, we consider the DPT distances between the already found tips and use the triangle ineqaulity for a point $x \in \mathcal{X}$:

$$\begin{aligned}\sum_{j=1}^{i-1}\sum_{l=j+1}^{i-1} dpt(r_j, r_l) \leq & \sum_{j=1}^{i-1}\sum_{l=j+1}^{i-1} dpt(r_j, x) + dpt(x, r_l) \\ = & \sum_{j=1}^{i-1}\sum_{l=j+1}^{i-1} dpt(r_j, x) + dpt(r_l, x) \\ = & (i-2)\sum_{j=1}^{i-1} dpt(r_j, x).\end{aligned}$$

Note, that this is a generalization of equation 4.4. Analogously, we determine tip r_i as

$$r_i = \underset{x \in \mathcal{X} \setminus \{r_1, \ldots, r_{i-1}\}}{\arg\max} \sum_{j=1}^{i-1} dpt(r_j, x).$$

The complete algorithm for endpoint identification is given below.

Algorithm 4 Endpoint identification

Input: DPT distance $dpt : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, number k of endpoints, root point $r \in \mathcal{X}$
Output: endpoints $r_1, \ldots, r_k$
$r_1 \leftarrow \underset{x \in \mathcal{X}}{\arg\max}\, dpt(r, x)$
$r_2 \leftarrow \underset{x \in \mathcal{X} \setminus \{r_1\}}{\arg\max}\, dpt(r_1, x)$
for $i = 3, \ldots, k$ **do**
  $r_i \leftarrow \underset{x \in \mathcal{X} \setminus \{r_1, \ldots, r_{i-1}\}}{\arg\max} \sum_{j=1}^{i-1} dpt(r_j, x)$
end for

In order to receive the starting point of a branch i for $i \in \{1, \ldots, k\}$, we define possible candidates

$$c_{i,j_1,j_2} \coloneqq \underset{x \in \mathcal{X}}{\arg\max}\, C^{(i)}_{j_1,j_2}(x)$$

for $j_1, j_2 \in \{1, \ldots, k\} \setminus \{i\}$, $j_1 \neq j_2$, where $C^{(i)}_{j_1,j_2}(x)$ is defined as in 4.6. Subsequently, we choose the candidate with the smallest index according to the

ordering $\mathcal{X}^{(i)}$:

$$c_i = \underset{c_{i,j_1,j_2}}{\arg\min}\, ind^{(i)}\left(c_{i,j_1,j_2}\right).$$

Finally, we give the complete extended DPT analysis algorithm:

Algorithm 5 DPT analysis algorithm

Input: data set $\mathcal{X}$, number k of branches, root point $r \in \mathcal{X}$, $s \leq n$
Output: branches $B_1, \ldots, B_k$
Compute the first s eigenvalues $\{\lambda_l\}_{l=1}^{s}$ and eigenvectors $\{\psi_l\}_{l=1}^{s}$ of transition matrix P computed in algorithm 1.
$dpt^2(x,y) \leftarrow \sum_{l=2}^{s} \left(\frac{\lambda_l}{1-\lambda_l}\right)^2 (\psi_l(x) - \psi_l(y))^2$ for $x, y \in \mathcal{X}$
Compute the endpoints $r_1, \ldots, r_k$ with algorithm 4,
for $i = 1, \ldots, k$ **do**
    Compute the ordering $\mathcal{X}^{(i)}$ according 4.5.
    for all $j_1, j_2 \in \{1, \ldots, k\} \setminus \{i\}$, $j_1 \neq j_2$ **do**
        Compute $C^{(i)}_{j_1,j_2}(x)$ for all $x \in \mathcal{X}$ according to 4.6.
        $c_{i,j_1,j_2} \leftarrow \underset{x \in \mathcal{X}}{\arg\max}\, C^{(i)}_{j_1,j_2}(x)$
    end for
    $c_i \leftarrow \underset{c_{i,j_1,j_2}}{\arg\min}\, ind^{(i)}\left(c_{i,j_1,j_2}\right)$
    $B_i \leftarrow \mathcal{X}^{(i)}[1 : ind^{(i)}(c_i)]$
end for

4.3 Experiments

Spectral Clustering

Since diffusion maps is a spectral embedding method, spectral clustering seems to be a natural choice for finding clusters in the data. In the classical version of spectral clustering, one uses k eigenvectors in order to find k clusters. This means, that the method expects for each eigenvector to represent a cluster. However, in practical applications, this does not have to be the case. Particularly, in diffusion maps, the first trivial eigenvector does not contain any information and can be neglected in the clustering method. Theoretically, a more meaningful modification of the spectral clustering algorithm would be to choose the number of clusters and the number of eigenvectors, separately. However, it is useful to keep the number of eigenvectors still small, since the most eigenvectors rather represent noise. Thus, nevertheless, taking $k-1$ eigenvectors for k clusters as in algorithm 3 results in practice to appropriate cell clustering.
As an example, we consider the *Guo* data. For the diffusion maps embedding, we use the classical Gaussian kernel with global bandwidth $\sigma = 10$. To apply the spectral clustering algorithm with k-means, we first need to determine the number of clusters. For this, we use the presented method from [BK17], considering the largest eigenvalues of the transition matrix and determining the spectral gap. In figure 4.1, the 20 largest eigenvalues of the transition matrix are shown for the *Guo* data set. The largest gap is between the 7-th and 8-th eigenvalue, therefore we

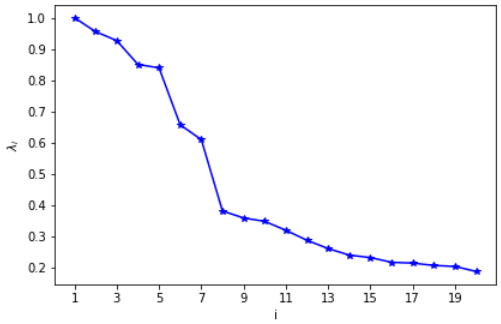


Figure 4.1: The first 20 eigenvalues of the transition matrix for the *Guo* data, using the Gaussian kernel with $\sigma = 10$.

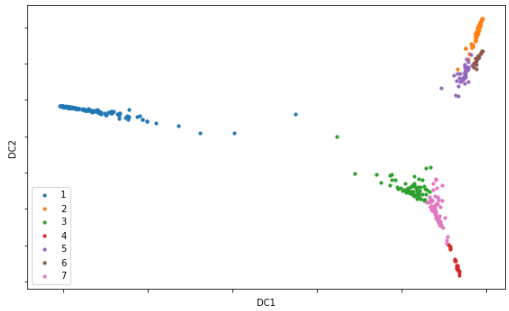


Figure 4.2: Result of the spectral clustering algorithm 3 using k-means for the *Guo* data.

choose $k = 7$ clusters to find. Considering the result in figure 4.2 and comparing the clustering to the differentiation development of the *Guo* data in figure 3.3, we can recognize the correctly found lineages TE (the blue cluster), ICM (the violet cluster), EPI and PE (the orange and brown cluster)[10]. The green cluster can be interpreted as a decision-making stage for the TE and ICM branch. However, the segregation of data points into a red and a pink cluster is biologically less meaningful. Here, the cells represent a repeated cell division from the 2-cell to the 16-cell stage. One could summarize this process under one cluster. Consequently, determining 6 clusters instead of 7, seems biologically to be the best choice. In fact, the trivial eigenvector corresponding to the largest eigenvalue, that is always 1, does not contain any information about the clusters. Therefore, it would be more meaningful not to count the first eigenvalue, which would lead to 6 clusters instead of 7. Besides, as an alternative, one could consider the second gap in figure 4.1, which leads to 5 clusters using the original count.
The result of spectral clustering for 5 and 6 clusters can be found in figure 4.3. On the left plot in figure 4.3, one can see, that the splitting of the ICM, PE and EPI lineage is not succesfully revealed (considering the orange and violet cluster). Thus, 6 is the minimal number of clusters to reveal the division of the ICM cell group into the two subbranches. The result, using 6 clusters, is therefore the most

[10]Which of the two clusters does represent the PE or the EPI lineage cannot be determined considering only the plot.

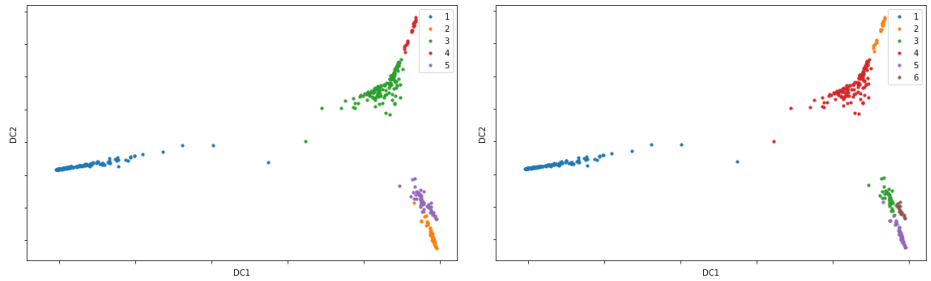


Figure 4.3: Spectral clustering with 5 and 6 clusters for the *Guo* data.

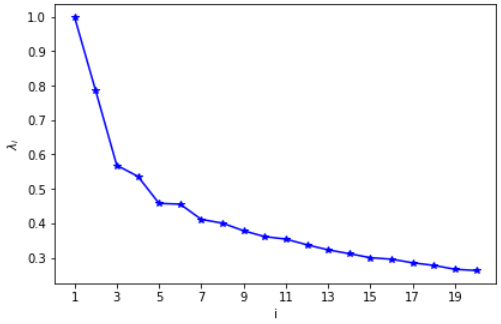


Figure 4.4: The first 20 eigenvalues of the transition matrix for the *Moignard* data, using the Gaussian kernel with $\sigma = 10$.

appropriate.
We noticed, that choosing the correct number of clusters is a difficult task. In the case of the *Moignard* data, determining the largest gap does not lead to a reasonable result, at all. The largest gap is between the second and the third eigenvalue, as one can see in figure 4.4. Further gaps are between the first and second eigenvalue, between the 4-th and 5-th eigenvalue and finally between the 6-th and 7-th eigenvalue. Classifying the data set into 2 groups, according to the largest gap, is too vague for our purpose.
In figure 4.5, we performed spectral clustering using different numbers of clusters for the *Moignard* data set. For the evaluation of the plots for the *Moignard* data, we take the expected cell development and the desired cell grouping in 3.3 and in the last picture of figure 4.5 into account. In the first plot, where three clusters are determined, we can interpret the three clusters as the HSC (the green cluster), PreMegE (the orange cluster) and a third branch (the blue cluster), summarizing the LMPP, CLP and GMP lineages. In the second plot, where four clusters are determined, the CLP cluster (the green cluster) is additionally identified. Subsequently, choosing five clusters to determine, leads to the separation of the LMPP and CLP lineage. The red cluster represents the CLP subbranch. Finally, the additional sixth cluster (the red cluster) in the fourth plot can be interpreted as a decision-making stage between the PreMegE (the orange cluster) and LMPP lineage (the brown cluster). Thus, taking 5 or 6 clusters is, from a biological point of view, the most appropriate choice. This corresponds to the small gap between the 6-th and 7-th eigenvalue. Consequently, taking the last clear gap proves to be

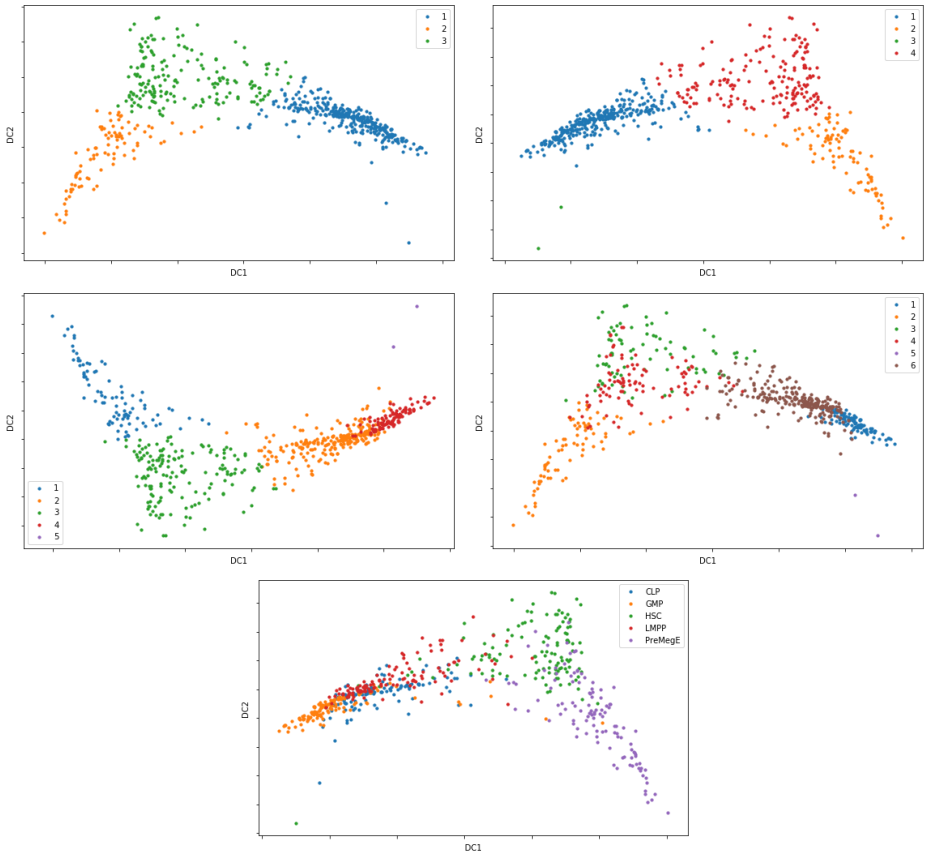


Figure 4.5: Spectral clustering using k-means with 3, 4, 5 and 6 clusters for the *Moignard* data. The last picture is a visualization of the *Moignard* data, labeled according to the original cell types .

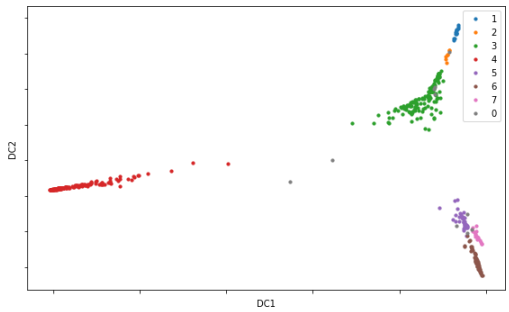


Figure 4.6: Spectral clustering with DBSCAN for the *Guo* data, using 10 non-trivial eigenvectors and $\epsilon = 0,03$ for clustering.

more appropriate for our purposes.
In order to overcome the difficulty of choosing the correct number of clusters, it is useful to consider clustering methods, which find the number of clusters, automatically. A common algorithm for this is DBSCAN, which is a data density-based approach. In order to use algorithm 3 with DBSCAN, we only need to choose the number of eigenvectors for clustering. Since the most eigenvectors contain disturbances, it is useful to take a small number of eigenvectors, e.g. 10. Figure 4.6 demonstrates the performance of spectral clustering for the *Guo* data, using DBSCAN as clustering method. In order to obtain this result, we had to carefully choose the parameter ϵ, defining the maximum distance between two data points for them to be considered as in the same neighbourhood. Using $\epsilon = 0,03$, DBSCAN determines seven clusters and identifies all desired lineages of the *Guo* data. Moreover, DBSCAN points out noisy data points that are marked in grey here (and form the zeroth cluster).
In the *Guo* data set, the different cell groups are spatially separated, such as the TE lineage, and form areas of high data density. Thus, in the case of the *Guo* data, DBSCAN leads to a reasonable result. However, naturally, cell stages in differentiation data move smoothly towards other cell groups, i.e. without spatial separation. Consequently, we did not find a choice of parameters to obtain a reasonable result for the *Moignard* data, when using DBSCAN. For the entire data set, the density of the points is similar, and there are no areas of low density between the cell groups. Thus, DBSCAN cannot separate the correct clusters.
In fact, clusters in differentiation data are not characterized by their data density, but by their direction in space. In particular, the cell groups have rather the shape of branches than of round point clouds. Most clustering methods, such as k-means, determine clusters as areas where the distance of each pair of samples is small. However, the distance between two data points in a branch-shaped group can be very large.
We illustrate the problem using the *Goettgens* data set. It describes the differentiation of cells towards two different development branches. For the diffusion maps embedding, we use the local Gaussian kernel K_k with $k = 5$. As clustering method, we choose k-means. The spectral gap in the plot of the eigenvalues in figure 4.7 is obviously between the third and the fourth eigenvalue. Thus, depending on the count of the eigenvalues, we need to determine 2 or 3 clusters.

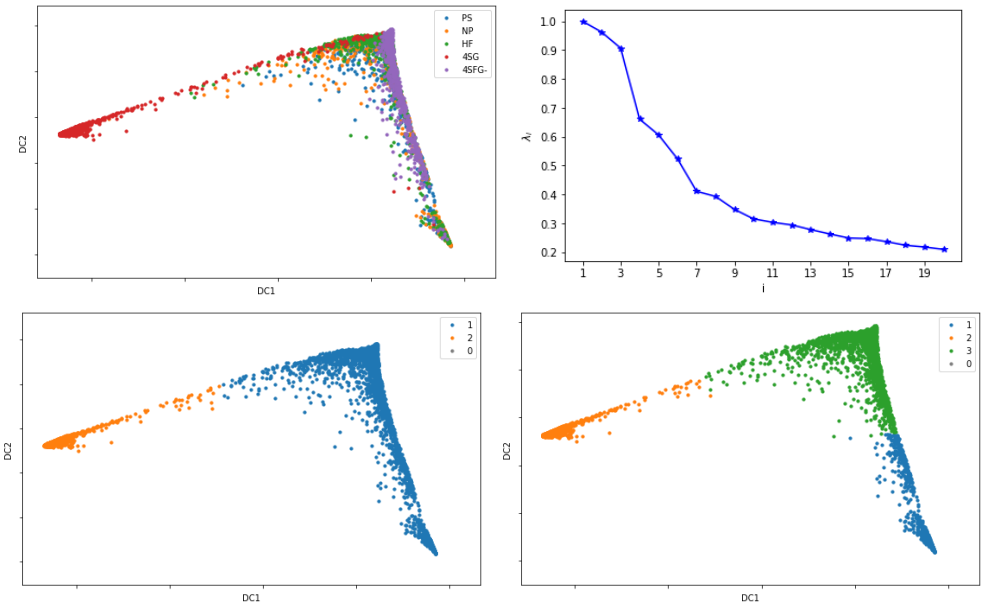


Figure 4.7: Spectral clustering with k-means for the *Goettgens* data, using the local Gaussian kernel K_k with $k = 5$. The upper left plot is a visualization of the *Goettgens* data, labeled according to the original cell types. The upper right picture shows the first 20 eigenvalues of the transition matrix. The lower plots demonstrates the clustering result with 2 and 3 clusters.

From a biological point of view, each cluster must represent a differentiation branch. However, this is not the case, considering the resulting plots in figure 4.7. Instead of identifying a branch as a cell group, the separation of two clusters is done in the middle of the branches. Thus, distance-based as well as density-based clustering methods are not appropriate enough for determining branches in cell differentiation data. In comparison, we now consider another method for group detection, using diffusion maps as preprocessing step, that has been developed especially for branch identification, called diffusion pseudotime analysis.

Diffusion Pseudotime Analysis

First of all, we consider the classical DPT analysis from [HBW$^+$16], determining three data branches. For this, the diffusion pseudotime distance $dpt(x, y)$ has to be approximately computed for all cells $x, y \in \mathcal{X}$. For this, we need to choose an appropriate number of summands for approximating the sum in 4.3. Similar to the spectral clustering methods, it is useful to take a small number of terms in order to avoid disturbances, e.g. the first 10 summands.
The figure 4.8 shows the result of the DPT method, determining three branches, for the *Goettgens* data. In the left picture, one can see the correctly determined endpoints of the three branches. In contrast to the result of spectral clustering in figure 4.7, DPT identified the branches as cell groups correctly. The green branch represents the less differentiated cells moving either towards the blue or the orange cell branch. Moreover, DPT determined unassigned cells (the grey

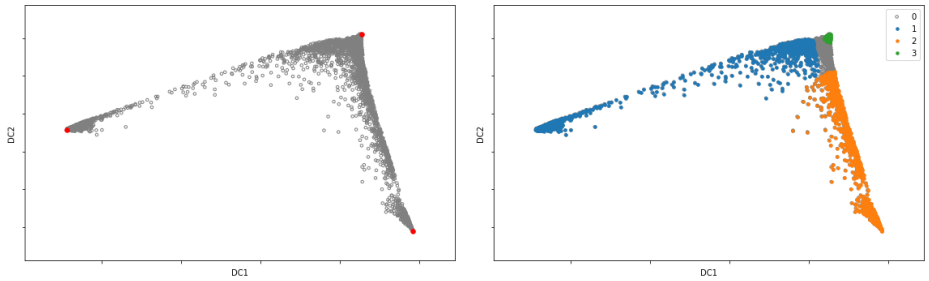


Figure 4.8: DPT analysis for the *Goettgens* data. We determined three branches including undecided cells, using the same settings as in figure 4.7 for the initial diffusion maps embedding. The left plot demonstrates the calculated tips as red points.

cells in the right plot), which is meaningful from a biological point of view. They represent cells in a decision-making stage.

Runtime Analysis

The two most time-consuming steps in the DPT analysis algorithm is the computation of the DPT distances and the correlations. Our implementation is based on [HBW+16], where an efficient recursive calculation of the correlations is presented. For data sets with more than 1000 cells, the computation of the DPT distance is reduced by using principal component analysis (PCA): Instead of using the complete matrix $M \in \mathbb{R}^{n \times n}$ for calculating the DPT distances, we reduce the number of columns to 50 using PCA and obtain an approximation $\tilde{M} \in \mathbb{R}^{n \times 50}$.

	DM	DPT	DPT with PCA
Guo (428 cells × 48 genes)	0,04	0,31	-
Moignard (597 cells × 21 genes)	0,10	0,57	-
Goettgens (3934 cells × 42 genes)	3,24	53,50	10,99
Yan (90 cells × 20214 genes)	0,43	0,46	-
Mass (408 cells × 8657 genes)	2,28	2,52	-
Klein (2717 cells × 2047 genes)	23,76	41,32	27,81
Paul (2730 cells × 3451 genes)	39,32	57,06	43,38

Figure 4.9: Runtime analysis of the DPT analysis algorithm for determining three branches using 10 components for calculating the DPT distances (in seconds).

In order to compare the computation times, we determined three branches for all data sets using the classical Gaussian kernel with σ_{LAF} as bandwidth and 10 components for calculating the diffusion pseudotime distance. In figure 4.9, the total computations times for the DPT analysis without or with the use of PCA (only for data sets with more than 1000 cells) is listed. Moreover, in the first column, the runtime of diffusion maps embedding in 10 dimensions is given, which is included in the total runtime of the DPT analysis in the other columns. Since the genes are only used for the Euclidean distance computations in diffusion maps, the remaining runtime of the DPT analysis strongly depends on the number of

cells. For instance, the *Goettgens* data set with a high amount of cells, but a low amount of genes, takes about 20 seconds less for the diffusion maps embedding than the *Klein* data. However, the *Klein* data needs less computation time for the complete DPT analysis than the *Goettgens* data, due to the lower amount of cells. Considering the *Yan*, *Mass*, *Klein* and *Paul* data, the computation time of the DPT analysis is dominated by the calculation of the diffusion maps embedding, taking more than the half of the total runtime of the DPT analysis. Using the PCA preprocessing step for matrix M has a significant impact on the runtime of the DPT analysis. In particular, the calculation time of the *Goettgens* data is significantly improved. The improvement for the *Klein* and *Paul* data is lower than for the *Goettgens* data, since the runtime of diffusion maps dominates the total computation time, due to the high amount of genes.

Extended DPT Analysis

For data sets with more than three branches, we proposed an extension of the diffusion pseudotime method, summarized in algorithm 5, for directly determining the branches. However, for using the algorithm 5, we need to pass the number of branches as input. In order to choose the correct number of branches, we use the same idea as for spectral clustering: Instead of determining the spectral gap in the sequence of eigenvalues $\{\lambda_l\}_{l=1}^n$ of the transition matrix P, used in spectral clustering, we consider the eigenvalues of the matrix M from 4.2, used for defining the diffusion pseudotime distance. The eigenvalues of the matrix M can be explicitly given by $\{\lambda_l/(1-\lambda_l)\}_{l=2}^n$[11].
To test the extended method, we perform a DPT analysis for the *Guo* data set. Initially, we consider the first 20 eigenvalues of the matrix M in figure 4.10. Notice that the minimal number of branches to determine with algorithm 5 is 3. Thus, we search the largest gap from the third eigenvalue. For the *Guo* data, it is between the 4-th and 5-th eigenvalue, i.e. we choose 4 branches. In fact, choosing 4 branches to compute, is the correct number of branches for the *Guo* data, as can be seen in figure 4.11: All calculated endpoints are actually at the top of a branch. This means, the extended algorithm 4 for endpoint identification succesfully determined the fourth tip.
The identified branches of the *Guo* data by algorithm 5 is given in figure 4.12. The blue, red and green branch can be identified as the TE, PE and EPI lineages. The grey undecided cells near the red and green branch can be understood as the ICM cell group. In fact, the ICM stage can be interpreted as a decision-making stage for the EPI and PE branch. The remaining orange branch represents the initial cell division of the mouse embryonic stem cells from the 2-cell stage to the 16-cell stage. In contrast to spectral clustering using k-means and DBSCAN, in figure 4.2 and 4.6, DPT summarizes this cell development as one differentiation branch instead of several clusters. Thus, from a biological point of view, the DPT analysis is the most meaningful method for group detection in the case of the *Guo* data.
As another example, we consider the *Moignard* data set. For the diffusion maps embedding, we use, as before, the Gaussian kernel with global bandwidth $\sigma = 10$.

[11] Note that the first eigenvalue is $\lambda_2(1-\lambda_2)$ due to the construction of the matrix M.

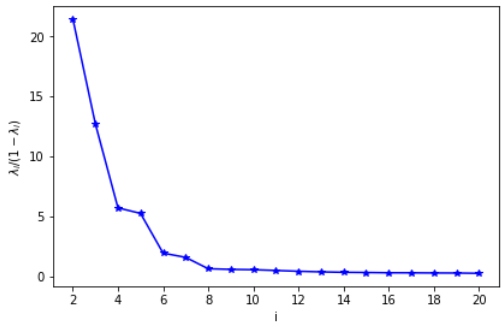


Figure 4.10: The first 20 eigenvalues of the matrix M from 4.2 for the *Guo* data.

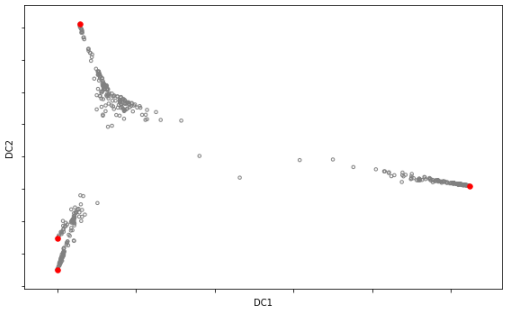


Figure 4.11: The 4 endpoints, calculated with algorithm 4 for the *Guo* data.

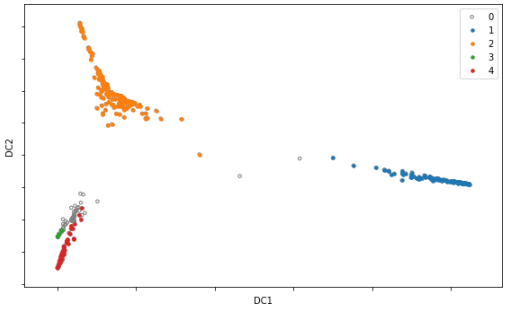


Figure 4.12: The result of the DPT analysis algorithm for the *Guo* data.

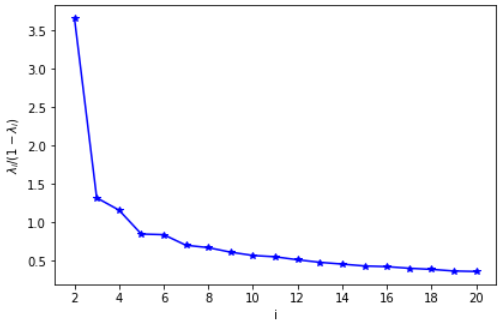


Figure 4.13: The first 20 eigenvalues of the matrix M from 4.2 for the *Moignard* data.

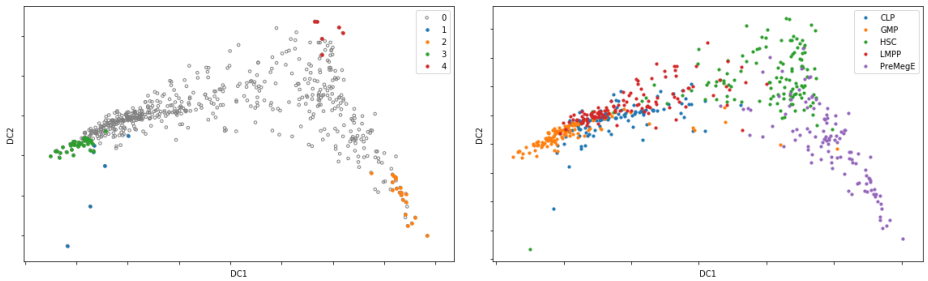


Figure 4.14: DPT analysis for the *Moignard* data. The left plot shows the resulted branch identification by DPT. The right plot is a visualization of the *Moignard* data, labeled according to the original cell types, for comparison.

We expect DPT to determine four groups (cf. figure 3.3): the PreMegE, CLP and GMP branch as well as the HSC cell group. The LMPP group is a decision-making stage and is expected to be represented by undecided cells. However, figure 4.13, showing the first 20 eigenvalues of the matrix M, proposes to determine only three branches: The largest gap from the third eigenvalue is between the third and the fourth eigenvalue while the gap between the 4-th and 5-th eigenvalue is rather small. One reason for that could be the greater distribution of the data (cf. the right picture of figure 4.14 showing a visualization of the *Moignard* data). The HSC cell group represents a data cloud rather than a clear branch. Thus, a clear branch identification of the *Moignard* data is more difficult than for the *Guo* data set.
In order to obtain a reasonable result for the *Moignard* data, we used the symmetric transition matrix P_{sym} in 3.3 instead of P and only 5 terms for constructing the diffusion pseudotime distance. The figure 4.14 shows the result of the DPT analysis for 4 branches. The red cluster can be identified as the HSC group, the green and blue branch represent GMP and CLP, and the orange branch is PreMegE (cf. figure 3.3). Noteworthy is the majority of unassigned cells. Compared to the right data visualization, labeled according to the original cell types, the amount of data points assigned to the red and orange branch, i.e. the HSC and PreMegE cell group, is too small. The reason lies in the determination of the branching points by the DPT algorithm: Branching points are determined as data points,

where the orderings of two branches switch from an anticorrelated to correlated behaviour. When dealing with more distributed data or data affected by noise, the switching point can occur much earlier in the data ordering according to the branch. Thus, DPT is unfortunately affected by disturbances in the case of the *Moignard* data.
DPT analysis for further data sets including a toy data set with 5 branches can be found in the appendix A.3.

Conclusion

All in all, we found that, for group detection in cell differentiation data, DPT analysis is more useful than spectral clustering. Distance-based as well as density-based clustering techniques are in general not suitable to determine differentiation branches. In order to directly identify more than 3 branches, we introduced a modification of the DPT analysis with appropriate results. Moreover, we proposed an approach to choose the number of branches, similar to the approach from [BK17] for spectral clustering. However, the DPT method seems to perform well only for clearly delineated branches.

5 Consideration of Censored Data

In biological data, we need to account for so-called *censored* values. In censored data, a certain range of numbers is missing in the data set due to experimental reasons. In PCR data sets, values above a certain limit of detection (LoD) cannot be measured, i.e. the values, which would be above the LoD, remain undetected. Censored values have to be distinguished from real missing values, where the experiment fails to detect the value during the procedure. So far, biologists usually remove cells with real missing values from analysis and set all censored values to the fixed value LoD. However, the high amount of censored values in a PCR data set, all fixed to one number, can disturb the analysis.
Thus, we will present two approaches to account for censored values in the diffusion maps method. The first approach, proposed in [HBT15], directly estimates the classical Gaussian kernel for censored data. The second approach, based on [EDVL13], estimates Euclidean distances instead and can be applied to general distance-based kernel functions.
Notice, that the approaches can be applied to real missing values, as well. However, they do not use information from the detected data values, which, in the case of real missing values, would be relevant.

5.1 Gaussian Kernel Estimation

Here, we present an approach from [HBT15] for directly estimating the kernel function $K_\sigma(x,y) = \exp(-\|x-y\|_2^2/(2\sigma^2))$ for a global parameter σ for a censored data set. To this end, we use the intepretation of the kernel as the interference of two Gaussian wave functions, as we presented in 3.6. For $x \in \mathcal{X}$, let $x_g \in \mathbb{R}$ be the entry of gene $g \in \mathcal{G}$ and $\mathcal{G}$ be the set of all measured genes in the data set $\mathcal{X}$. We can write the Gaussian kernel K_σ as the product of Gaussian kernels of the gene entries:

$$\begin{aligned} K_\sigma(x,y) &= \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right) \\ &= \prod_{g\in\mathcal{G}} \exp\left(-\frac{(x_g-y_g)^2}{2\sigma^2}\right) \\ &= \prod_{g\in\mathcal{G}} \int_{-\infty}^{\infty} Y_{x_g}(z) Y_{y_g}(z) dz \end{aligned}$$

[1]

When x_g (or y_g, respectively) is a censored value, the idea is to approximate the Gaussian wave function $Y_{x_g}(z)$ by an indicator function. To this end, we assume the value x_g to be between the limit of detection M_1 and a maximal value $M_2 > M_1$. Consequently, we approximate the wave function by

$$Y_{x_g}(z) \approx \frac{1}{\sqrt{M_2 - M_1 + 2\sigma}} \chi_{[M_1-\sigma, M_2+\sigma]} =: \tilde{Y}_{x_g}(z)$$

[1] Notice that, according to the dimension of x_g and y_g, Y_{x_g} and Y_{y_g} are here defined on $\mathbb{R}$.

where the factor in front is for normalization. Subsequently, we have to consider three different cases:

1. When the entries x_g and y_g are definite, then the interference can be given as the original one:

$$\int_{-\infty}^{\infty} Y_{x_g}(z) Y_{y_g}(z) dz = \exp\left(-\frac{(x_g - y_g)^2}{2\sigma^2}\right).$$

2. When both entries x_g and y_g are not detected, then, due to the normalization, the interference of the two approximate wave functions is 1[2]:

$$\int_{-\infty}^{\infty} \tilde{Y}_{x_g}(z) \tilde{Y}_{y_g}(z) dz = 1.$$

3. When one entry, e.g. x_g, is known and the other entry y_g is not detected, then we get

$$\int_{-\infty}^{\infty} Y_{x_g}(z) \tilde{Y}_{y_g}(z) dz =$$

$$\int_{M_1-\sigma}^{M_2+\sigma} \frac{1}{\sqrt{M_2 - M_1 + 2\sigma}} \left(\frac{2}{\pi\sigma^2}\right)^{1/4} \exp\left(-\frac{(z - x_g)^2}{\sigma^2}\right) dz$$

$$= \frac{1}{\sqrt{M_2 - M_1 + 2\sigma}} \left(\frac{\pi\sigma^2}{8}\right)^{1/4} \left(\operatorname{erfc}\left(\frac{M_1 - \sigma - x_g}{\sigma}\right) - \operatorname{erfc}\left(\frac{M_2 + \sigma - x_g}{\sigma}\right)\right).$$

We denote the resulted (approximated) interference for any two entries x_g and y_g as I_{x_g,y_g}. The complete Gaussian kernel estimation algorithm is given as follows:

Algorithm 6 Gaussian kernel estimation algorithm

Input: Censored data set $\mathcal{X}$, M_1, M_2, bandwidth $\sigma > 0$
Output: Estimation $\tilde{K}$ for the Gaussian kernel $K_\sigma(x, y)$
for all $x \in \mathcal{X}$ **do**
  for all $y \in \mathcal{X}$ **do**
    $\tilde{K}(x, y) \leftarrow 1$
    for all $g \in \mathcal{G}$ **do**
      Compute I_{x_g,y_g} according to one of the three cases as above.
      $\tilde{K}(x, y) \leftarrow \tilde{K}(x, y) I_{x_g,y_g}$
    end for
  end for
end for

Notice, that the algorithm 6 can be used in combination with the bandwidth selection technique by [HBT15].

[2]Notice, that, for definite entries, the interference is 1, if and only if both entries are identical. This means, that here we indirectly assume the censored entries to be (approximately) identical.

5.2 Euclidean Distance Estimation

The previously presented approach directly estimates the Gaussian kernel for censored data. In particular, the method does not need to define Euclidean distances on censored data. However, for computing Lafon's rule or local bandwidths, a notion of Euclidean distances on censored data is needed. Thus, we present an approach for estimating Euclidean distances between data points with censored (or missing) values, based on [EDVL13].
For any $x, y \in \mathcal{X}$, we aim to compute the Euclidean distance $\|x - y\|_2$. Let $M_x \subset \{1, \dots, d\}$ be the set of indices of missing components (i.e. the set of missing gene entries) for $x \in \mathcal{X}$. Then, we can split the squared Euclidean distance in the following way:

$$\begin{aligned}\|x - y\|_2^2 = &\sum_{g \notin M_x \cup M_y} (x_g - y_g)^2 + \sum_{g \in M_x \setminus M_y} (x_g - y_g)^2 \\ &+ \sum_{g \in M_y \setminus M_x} (x_g - y_g)^2 + \sum_{g \in M_x \cup M_y} (x_g - y_g)^2.\end{aligned}$$

The idea is now to model the missing components of a cell data point $x \in \mathcal{X}$ as random variables X_g for $g \in M_x$. Then, the expectation of the squared distance can be given by[3]

$$\begin{aligned}\mathbb{E}[\|x - y\|_2^2] = &\sum_{g \notin M_x \cup M_y} (x_g - y_g)^2 + \sum_{g \in M_x \setminus M_y} (\mathbb{E}[X_g] - y_g)^2 + Var[X_g] \\ &+ \sum_{g \in M_y \setminus M_x} (x_g - \mathbb{E}[Y_g])^2 + Var[Y_g] \\ &+ \sum_{g \in M_x \cup M_y} (\mathbb{E}[X_g] - \mathbb{E}[Y_g])^2 + Var[X_g] + Var[Y_g].\end{aligned}$$

If we now define the variables

$$x'_g \coloneqq \begin{cases} \mathbb{E}[X_g] & \text{if } g \in M_x \\ x_g & \text{otherwise} \end{cases} \tag{5.1}$$

for all $g \in \mathcal{G}$ and

$$s_x \coloneqq \sum_{g \in M_x} Var[X_g]$$

for all $x \in \mathcal{X}$, we can rewrite the expectation as

$$\mathbb{E}[\|x - y\|_2^2] = \|x' - y'\|_2^2 + s_x + s_y.$$

Consequently, it suffices to estimate the expectation $\mathbb{E}[X_g]$ and the variance $Var[X_g]$ of the censored values. Unfortunately, we do not have any knowledge about the distribution of the missing components. If we assume X_g to be uniformly distributed between M_1 and M_2, the expectation and variance can be explicitly given by

$$\mathbb{E}[X_g] = \frac{1}{2}(M_1 + M_2), \qquad Var[X_g] = \frac{1}{12}(M_2 - M_1)^2.$$

[3] Here we use the linearity of the expectation and the definition of the variance. For details, we refer to [EDVL13].

Since the variance is the same for all censored values, we can rewrite the sum in s_x as the variance $V := Var[X_g]$ multiplied with the number of missing components in x:

$$s_x = V|M_x|.$$

Finally, the algorithm, assuming uniformly distributed censored values, is given as follows:

Algorithm 7 Euclidean distance estimation algorithm

Input: Censored data set $\mathcal{X}$, M_1, M_2
Output: Estimation $d(x, y)$ for the Euclidean distances $\|x-y\|_2$ for all $x, y \in \mathcal{X}$

$\mathbb{E} \leftarrow \frac{1}{2}(M_1 + M_2)$
$V \leftarrow \frac{1}{12}(M_2 - M_1)^2$
for all $x \in \mathcal{X}$ **do**
    $s_x \leftarrow V|M_x|$
    Define $x' \in \mathbb{R}^{|\mathcal{G}|}$ according to 5.1.
end for
for all $x \in \mathcal{X}$ **do**
    for all $y \in \mathcal{X}$ **do**
        $d^2(x, y) \leftarrow \|x' - y'\|_2^2 + s_x + s_y$
    end for
end for

5.3 Experiments

For the analysis of the two methods, we consider three PCR data sets, the *Guo* data, the *Moignard* data and the *Goettgens* data set. All three data sets have a high amount of censored values. After data cleaning, as described in chapter 2, the *Guo* data set has 5.087 censored values out of 20.544 (24,8 %), the *Moignard* data 3.269 censored values out of 12.537 (26,1 %) and the *Goettgens* data 73.460 censored values out of 165.228 (44,5 %). In figure 5.1, the positions of censored values in the data sets are illustrated. Notice that almost each cell contains

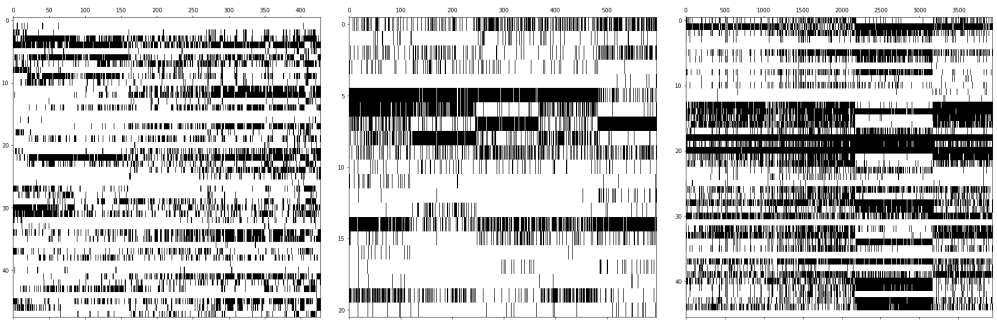


Figure 5.1: Illustration of the positions of the censored values for the *Guo* data (left), the *Moignard* data (middle) and the *Goettgens* data (right). The vertical axis represents the cells, and the horizontal axis the genes. The white boxes are detected values and the black ones censored values.

undetermined gene values. In particular, there are genes, where very few cells have detected a value.

Gaussian Kernel Estimation

In order to account for censored values, we consider the Gaussian kernel estimation approach, first. For this, we renounced the data normalization in each case and applied the kernel estimation method on the unnormalized data. We chose the limit of detection for M_1 ($M_1 = 28$ for *Guo* and *Moignard* and $M_1 = 25$ for *Goettgens*) and set $M_2 = 40$ for all data sets, which is a meaningful upper limit, from a biological point of view. Now, we compare three cases: First, we performed a diffusion maps embedding without accounting for censored values, i.e. the censored values were fixed to one value and the data was normalized, as described in chapter 2. There, we selected the bandwidth σ by the approach from [HBT15] for the classical Gaussian kernel. Note that for all three PCR data sets, the bandwidth $\sigma = 10$ was selected. Afterwards, we used the Gaussian kernel estimation approach on the unnormalized data using the fixed bandwidth $\sigma = 10$. Finally, we applied the approach from [HBT15] for selecting the bandwidth σ in combination with kernel estimation.
The resulted data visualizations can be seen in figure 5.2 and 5.3. In the case of the *Moignard* and *Goettgens* data, $\sigma = 10$ was suggested by the approach from [HBT15], using kernel estimation. Therefore, we left out one plot for each data set. The plots show that the kernel estimation approach generally results in appropriate data visualizations for all three data sets. Some structures are slightly more stressed when considering the censored data, for instance the two subbranches PE and EPI of the *Guo* data. In particular, when using a selected bandwidth, in the right plot, the two subbranches are clearly separated in space. In the case of the *Moignard* data, the CLP and GMP branch are more highlighted in contrast to the HSC cell group, when using kernel estimation. However, the *Goettgens* data set has the same structure with or without kernel estimation.
Despite appropriate results in data visualization, the Gaussian kernel estimation has several disadvantages. One problem is the high runtime of the approach in contrast to the remaining diffusion maps embedding. For estimating the kernel for a fixed bandwidth in the case of the *Goettgens* data, for instance, we need about two and a half minutes. However, just approximately 2 seconds are needed for the remaining computation. Using additionally the bandwidth selection technique from [HBT15], where for each candidate a kernel estimation has to be performed, a diffusion maps embedding with kernel estimation for the *Goettgens* data takes about one hour (cf. table 5.5). The main reason is that we need to consider several cases for each data pair, which costs a lot of time. Another problem is the low flexibility of the approach. It is developed only for estimating the classical Gaussian kernel using a global bandwidth. Applying Lafon's rule or translating the kernel estimation approach to Gaussian kernels with local bandwidths is not (directly) possible. For defining local bandwidths using k-th nearest neighbour distances, a notion of Euclidean distances on censored data is needed. Therefore, we proposed an Euclidean distance estimation method, based on [BK17], that we will investigate in the following.

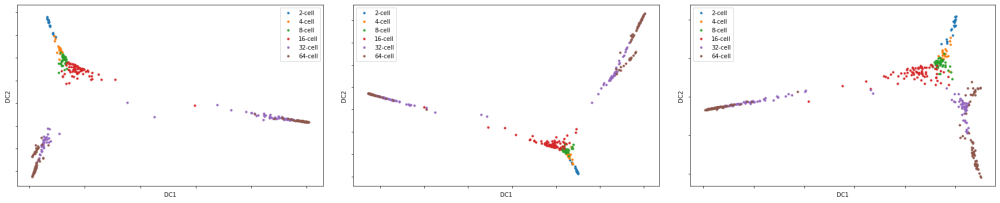


Figure 5.2: Comparison of the data visualization using the Gaussian kernel estimation approach to the visualization without accounting for censored values for the *Guo* data. On the one hand, the censored values are fixed to one value and diffusion maps embedding is performed on the normalized data, using the bandwidth $\sigma = 10$ for the classical Gaussian kernel (left plot). On the other hand, kernel estimation is performed on the unnormalized data with a fixed bandwidth $\sigma = 10$ (middle plot) and with σ selected by the approach from [HBT15] (right plot).

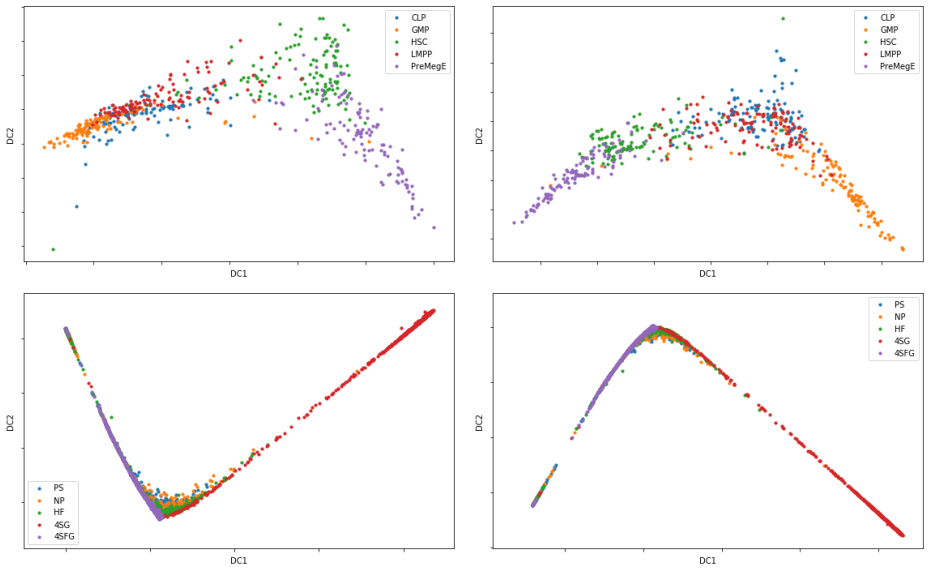


Figure 5.3: Comparison of the data visualization using the Gaussian kernel estimation approach to the visualization without accounting for censored values for the *Moignard* data (above) and the *Goettgens* data (below). The left plots show the result without considering censored values, the right plots demonstrate the result using the kernel estimation approach. The approach from [HBT15] suggested the bandwidth $\sigma = 10$ for all plots.

Euclidean Distance Estimation

We compare the performance of the distance estimation approach to the kernel estimation method with respect to data visualization. For the Euclidean distance estimation algorithm we use the same values for M_1 and M_2 as for the kernel estimation approach. After estimating the Euclidean distances, we calculate the classical Gaussian kernel, using a bandwidth, selected by the approach from [HBT15]. In figure 5.4, the results are presented. The data visualizations in the

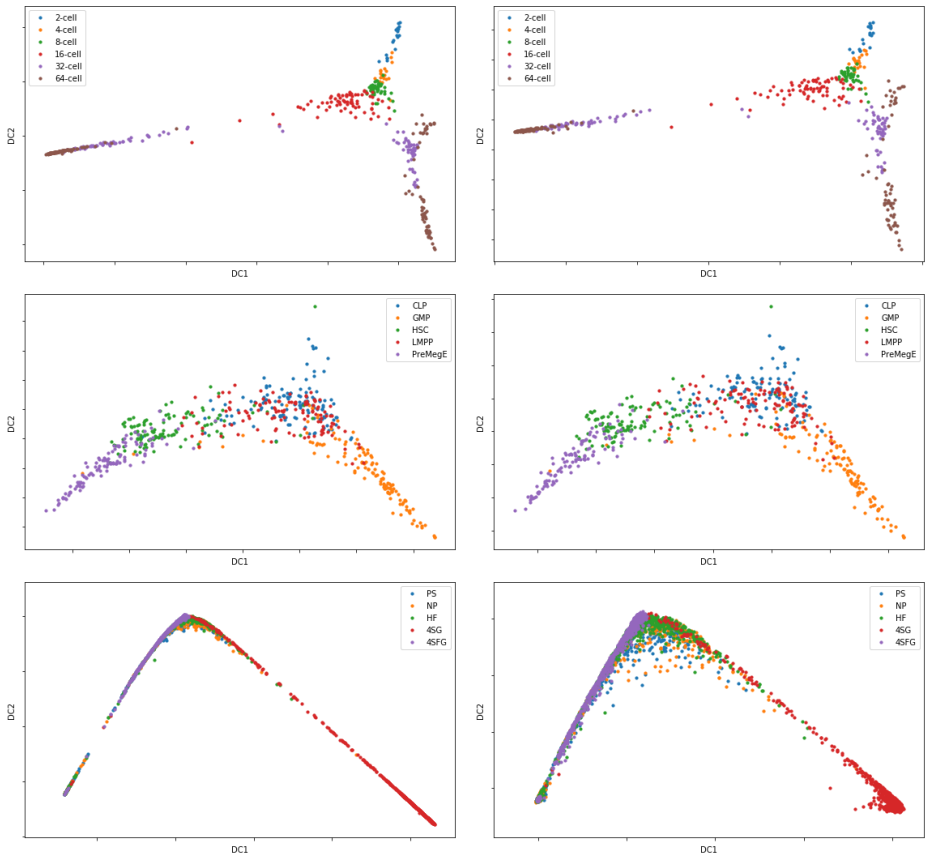


Figure 5.4: Comparison between kernel estimation (left) and distance estimation approach (right) for the (unnormalized) *Guo* data (above), *Moignard* data (middle) and *Goettgens* data (below). For all plots, we have used a global bandwidth, selected by the approach from [HBT15] for the classical Gaussian kernel.

left plots, where kernel estimation is used, and in the right plots, where distance estimation is applied, look rather similar. In the case of the *Guo* and *Goettgens* data set, the data points are more affected by small disturbances using the distance estimate. Nevertheless, the results of the distance estimation approach are comparable to the kernel estimation approach.

Comparison of the Computation Times

In figure 5.5, the computation times for a diffusion maps embedding in three dimensions using the kernel estimation approach and the distance estimation

	KE σ fix	KE $\hat{\sigma}$	DE σ fix	DE $\hat{\sigma}$
Guo	2,56 sec.	54,48 sec.	0,10 sec.	1,12 sec.
Moignard	2,23 sec.	47,13 sec.	0,19 sec.	1,47 sec.
Goettgens	2,78 min.	64,08 min.	8,12 sec.	34,55 sec.

Figure 5.5: Runtime analysis of the kernel estimation (KE) and distance estimation (DE) algorithm for diffusion maps embedding in three dimensions using a fixed bandwidth σ and a bandwidth $\hat{\sigma}$ selected by the approach from [HBT15].

method are listed for the three data sets. On the one hand, we measured the time using a fixed bandwidth ($\sigma = 10$). On the other hand, we performed the bandwidth selection technique by [HBT15] to choose an appropriate parameter ($\hat{\sigma}$). In fact, the distance estimation procedure takes much less computation time than the kernel estimation method. For the *Goettgens* data, we needed approximately half a minute for the embedding with bandwidth selection using distance estimation in contrast to an hour with kernel estimation. Notice, that the distance estimation algorithm has to be performed only once for selected bandwidth, whereas the kernel estimation has to be computed for each bandwidth candidate, separately. Nevertheless, for fixed bandwidth, diffusion maps embedding using distance estimation is faster than using kernel estimation, as well.

Conclusion

In summary, we proposed a distance estimation approach for considering censored data as an alternative method to the introduced kernel estimation technique by [HBT15], which is faster and more flexible than the old one. With the estimated Euclidean distances, it is now possible to calculate Lafon's rule for selecting the bandwidth or apply Gaussian kernels with local bandwidths for embedding. A DPT analysis using a distance estimate in order to account for censored values can be done, as well (cf. appendix A.4).

6 Conclusion and Outlook

Conclusion

In this work, we investigated, how diffusion maps as a nonlinear dimensionality reduction method can be used for the analysis of biological data. We were able to confirm that diffusion maps is well suited to depict the particular structure of biological data. In order to detect and separate cell groups in the data, we considered two group detection techniques, based on diffusion maps. Since diffusion maps is a spectral embedding method, we first investigated spectral clustering as the most natural method which directly uses the embedding vectors to find clusters in the data. However, spectral clustering in combination with common clustering techniques was not able to determine the branch structure of biological data, correctly. Therefore, we considered the diffusion pseudotime analysis which first uses the diffusion maps embedding to define a distance measure and afterwards applies a correlation-based approach in order to identify branches in biological data. This gave more meaningful results.
As our own contribution, we extended the method of DPT to make it possible to set the number of branches as input to the algorithm. With this extension, the DPT analysis algorithm now directly determines a specific number of branches without manual control by the user. Additionally, we proposed an approach to find the correct number of branches in the data that can be used as input to the algorithm.
Finally, we dealt with censored biological data. We proposed a method for estimating Euclidean distances for censored data which can be inserted in the kernel function of diffusion maps. This approach proved to be more flexible than the proposed kernel estimation approach by Haghverdi et al.. Moreover, the calculation time of the distance estimation method proved to be much lower than the kernel estimation approach with comparable results in the data visualization.

Outlook

Since the performance of diffusion maps strongly depends on the used kernel function, it is useful to investigate the influence of other kernel functions on data visualization and group detection. Parameter-free kernels are of particular interest as the difficult selection of optimal parameters can be omitted.
So far, we set up the transition matrix of diffusion maps as a completely filled matrix for all pairs of cells. However, the n^2d computational cost can be too high for biological data with a high number of cells. Computing only the transition probabilities to the k-th nearest neighbours and set up a sparse transition matrix can solve this problem. Since the most transition probabilities are anyway very small, we do not expect a significant deterioration of the results (for a k big enough) when setting these to zero.
Furthermore, diffusion maps can be combined with multidimensional scaling

(MDS) by using the diffusion distance for the definition of a stress function to minimize. A possible realization, called PHATE and investigated for biological data, can be found in [MvDW+17].
Finally, for further improvement of cell group detection in biological data, *single-cell topological data analysis* (scTDA) could be relevant [RCK+17]. Based on dimensionality reduction, scTDA attempts to reconstruct the tree structure of differentiation data by building a topological representation of the data.

A Appendix

A.1 Implementation

The implementation of all algorithms was realized using the *python* programming language, version 3.4.4, including numerous python packages, e.g. *numpy*, *scipy*, *matplotlib* and *sklearn*. For all dimensionality reduction methods except diffusion maps and for the clustering techniques k-means and DBSCAN, we used the algorithms from the machine learning package *sklearn*. The realization of our DPT analysis algorithm is based on the implementation by [HBW$^+$16]. For the kernel estimation approach, we applied the *cython* language to speed up the computation.

A.2 Appendix to Chapter 3

In this section, all further plots to chapter 3 can be found.

A.2.1 Comparison to other Dimensionality Reduction Methods

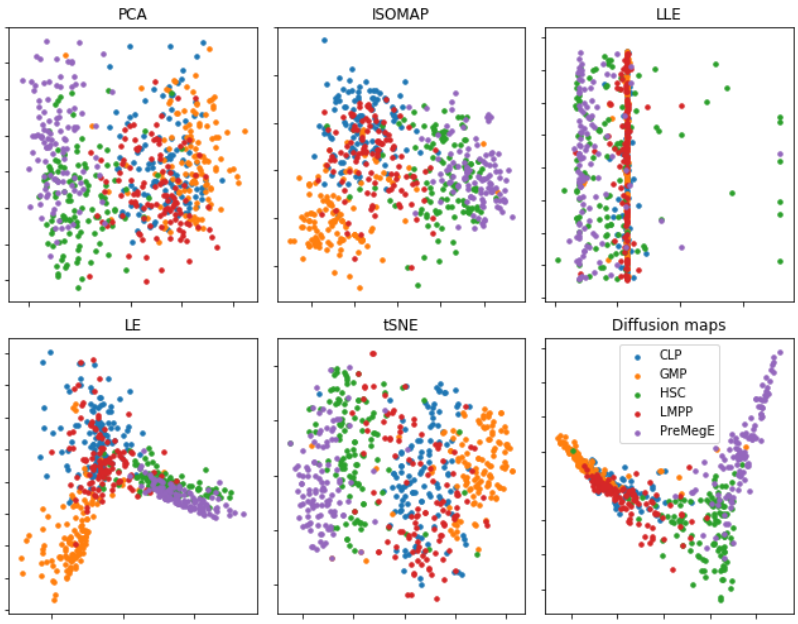


Figure A.1: Embedding of the *Moignard* data. Diffusion maps and LE are the only methods, which reveal a clear structure of the data set. However, the diffusion maps embedding points out the correct differentiation process of the cells in contrast to the LE embedding, namely the transition of HSC cells into either PreMegE or LMPP. The subbranch CLP is not revealed by means of any method (cf. figure 3.8).

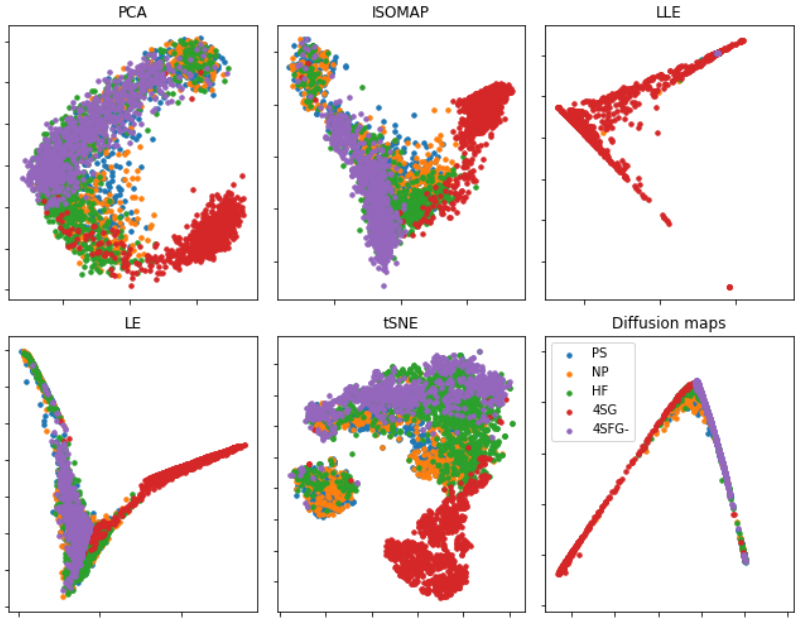


Figure A.2: Embedding of the *Goettgens* data.

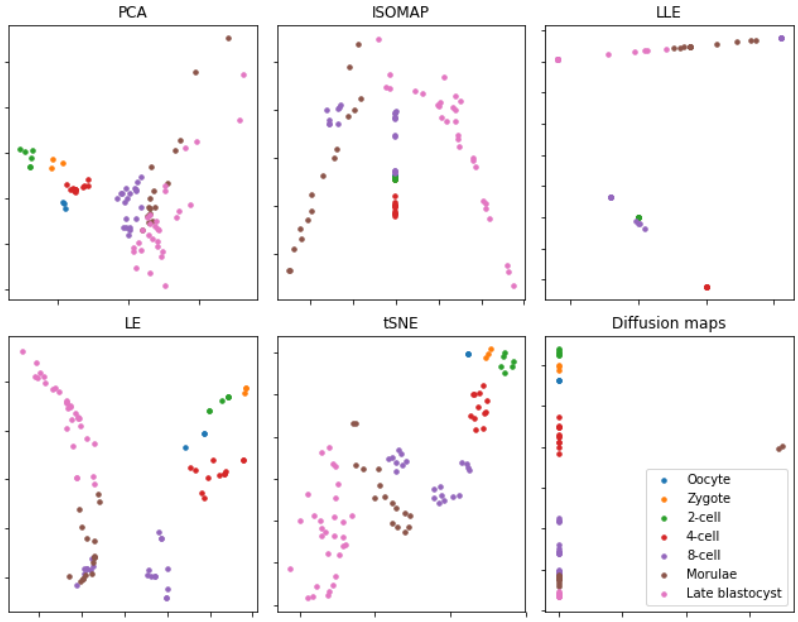


Figure A.3: Embedding of the *Yan* data. The remarkable observation here is that diffusion maps is the only method showing two outlier cells from the morulae cell group, which could indicate a differentiation lineage. However, we need to remark, that, with only 90 cells and a high amount of genes, the *Yan* data is the most susceptible to perturbation.

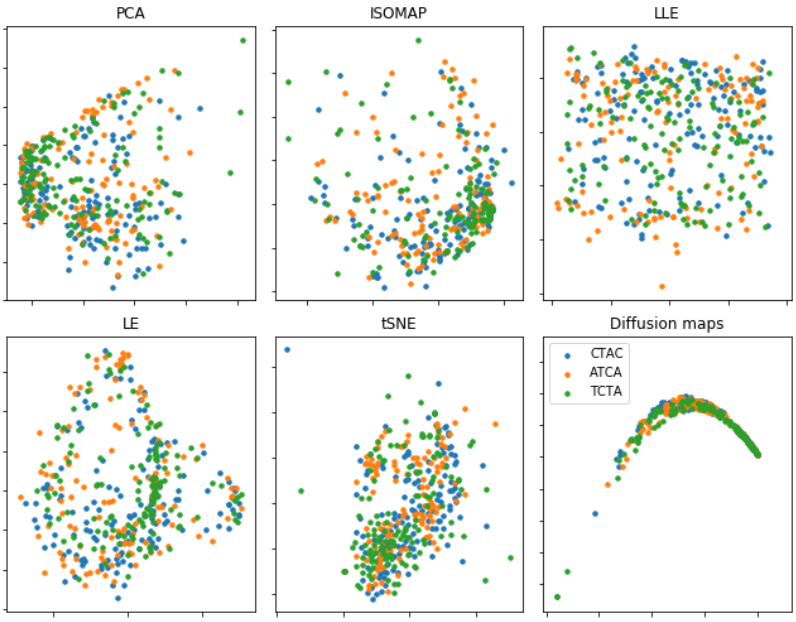


Figure A.4: Embedding of the *Mass* data. The diffusion maps visualization is the only one with a clear structure due to the denoising effect.

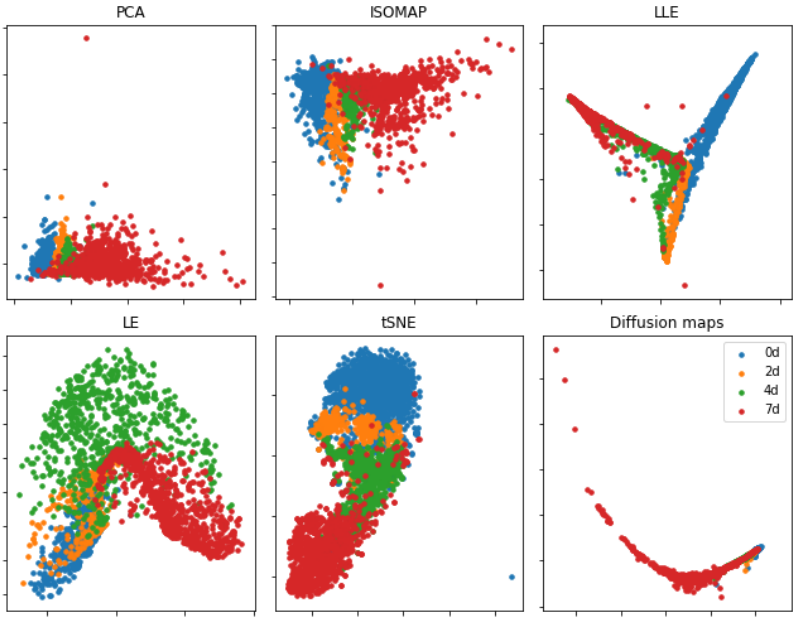


Figure A.5: Embedding of the *Klein* data.

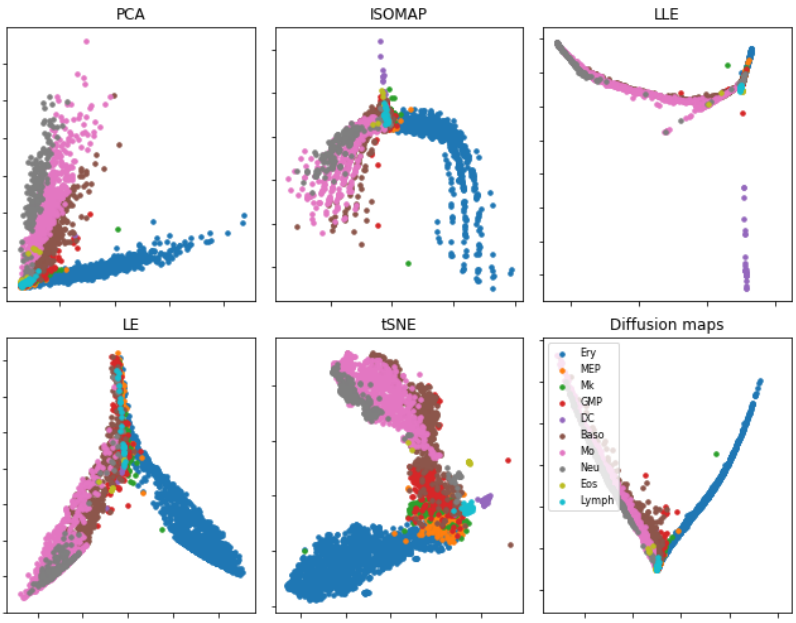


Figure A.6: Embedding of the *Paul* data. The diffusion maps embedding clearly shows the separation of two branches. However, the DC cell group is additionally highlighted in several plots, e.g. in ISOMAP, LLE and tSNE (cf. figure A.12).

A.2.2 Data Visualizations for σ_{LAF} and $\hat{\sigma}$

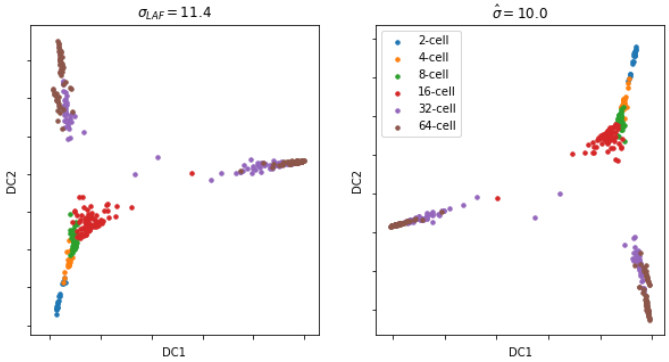


Figure A.7: Embedding of the *Guo* data, using Lafon's rule (left) and the selection technique by [HBT15] for the bandwidth σ.

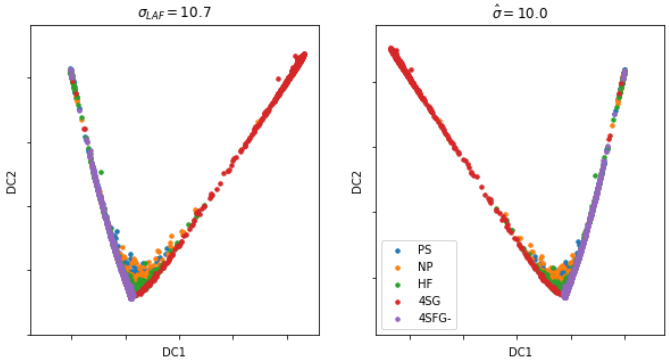


Figure A.8: Embedding of the *Goettgens* data, using Lafon's rule (left) and the selection technique by [HBT15] for the bandwidth σ.

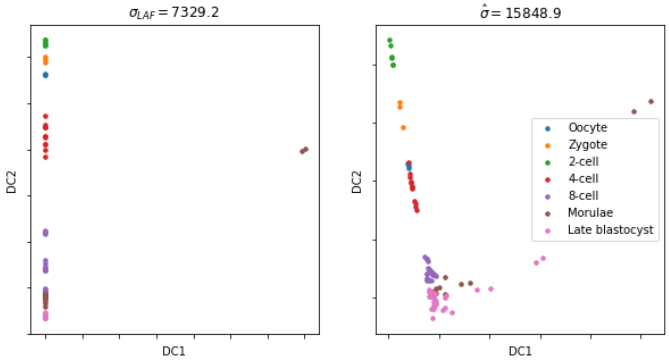


Figure A.9: Embedding of the *Yan* data, using Lafon's rule (left) and the selection technique by [HBT15] for the bandwidth σ.

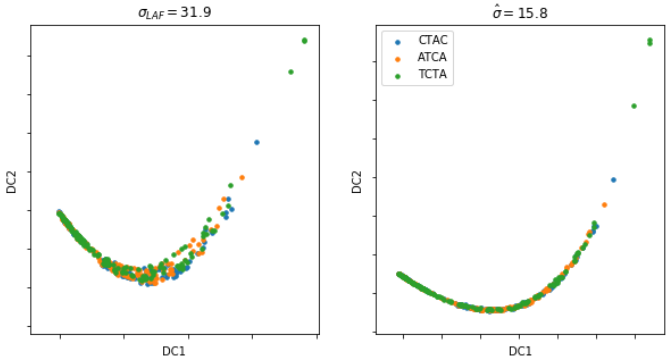


Figure A.10: Embedding of the *Mass* data, using Lafon's rule (left) and the selection technique by [HBT15] for the bandwidth σ.

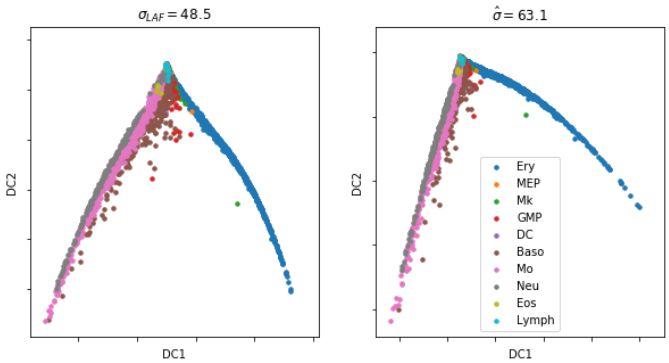


Figure A.11: Embedding of the *Paul* data, using Lafon's rule (left) and the selection technique by [HBT15] for the bandwidth σ.

A.2.3 Local Gaussian Kernels

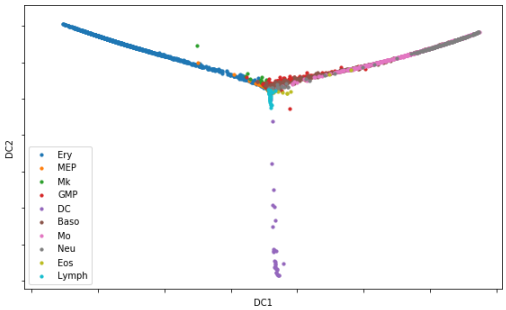


Figure A.12: Embedding of the *Paul* data, using the kernel $K_{k,a}$ with $k = 5$ and $a = 10$. In contrast to figure A.11, the local Gaussian kernel reveals the DC branch. This branch can be seen in the ISOMAP, LLE and tSNE embedding in figure A.6, as well.

A.3 Appendix to Chapter 4

In this section, the results of DPT analysis for further data sets for chapter 4 are shown.

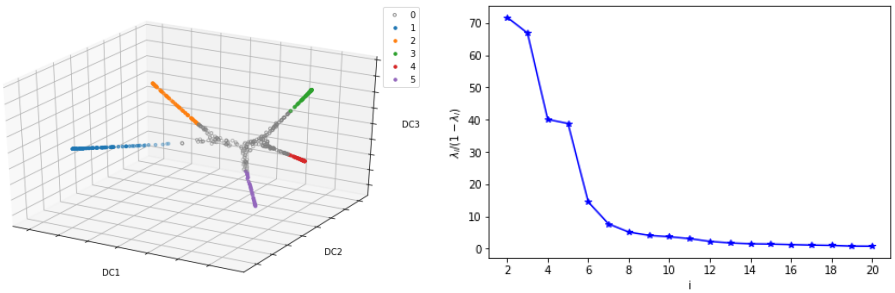


Figure A.13: DPT analysis for a toy data set with 5 branches. We used 10 components for computing the diffusion pseudotime distance. From the sixth value, the eigenvalues can be neglected. Here it is useful to choose the number of branches as the number of the last large enough eigenvalue.

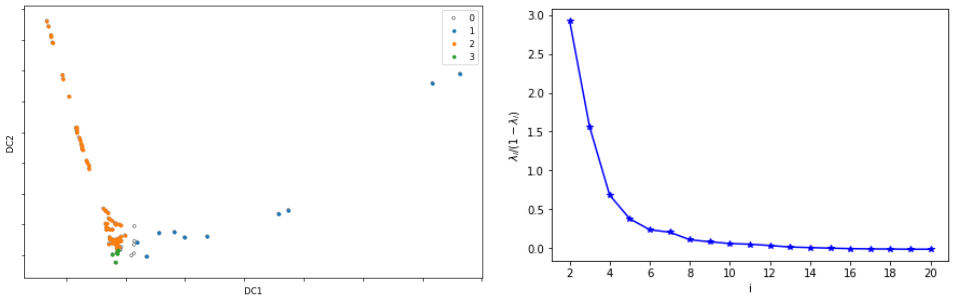


Figure A.14: DPT analysis for the *Yan* data. We used 10 components for computing the diffusion pseudotime distance with the classical Gaussian kernel, using $\hat{\sigma} = 15848.9$. We determined three branches. Unfortunately, there is no clear gap considering the eigenvalues.

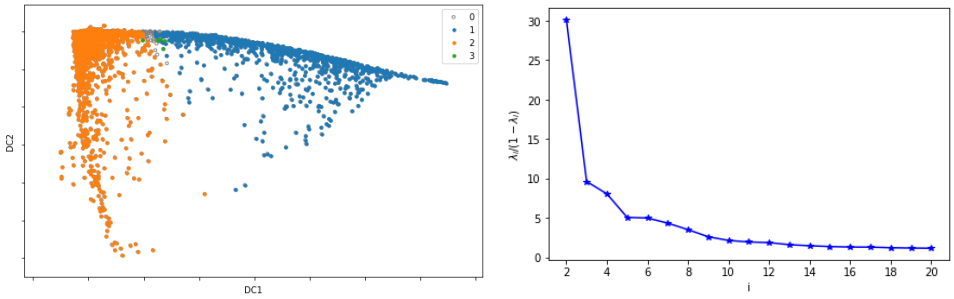


Figure A.15: DPT analysis for the *Klein* data. We used 10 components for computing the diffusion pseudotime distance, the symmetric matrix P_{sym} and kernel $K_{k,a}$ with $k = 5$ and $a = 10$. There is a gap between the third and the fourth eigenvalue. Thus, we chose three branches to determine.

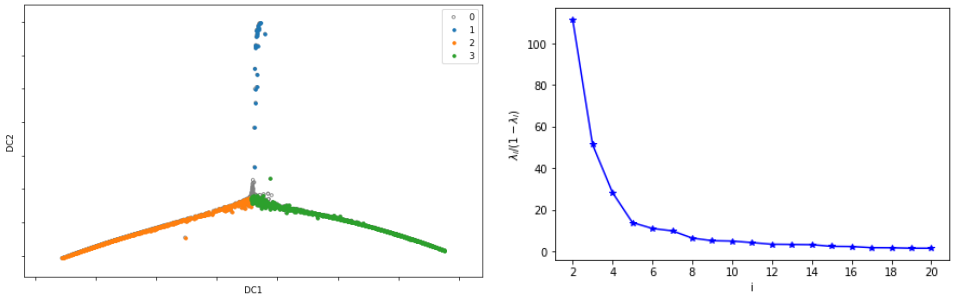


Figure A.16: DPT analysis for the *Paul* data. We used 10 components for computing the diffusion pseudotime distance and the kernel $K_{k,a}$ with $k = 5$ and $a = 10$. We determined three branches. There is no clear gap considering the eigenvalues, but we can neglect them from the fourth eigenvalue.

A.4 Appendix to Chapter 5

In this section, the results of DPT analysis for the three PCR data sets using the distance estimation algorithm 7 are demonstrated.

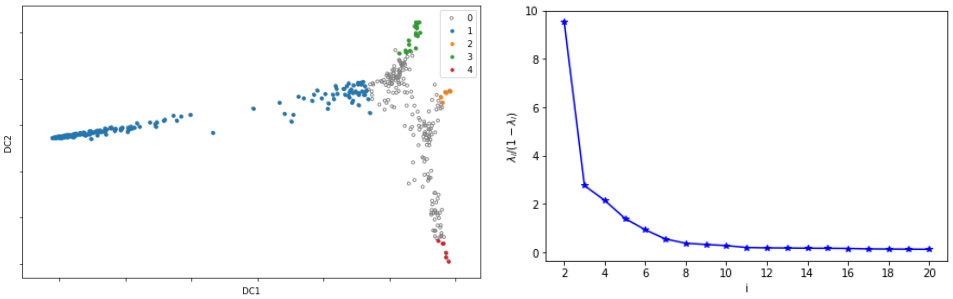


Figure A.17: DPT analysis for the *Guo* data using distance estimation in order to account for censored values. Unfortunately, there is no clear gap in the sequence of eigenvalues.

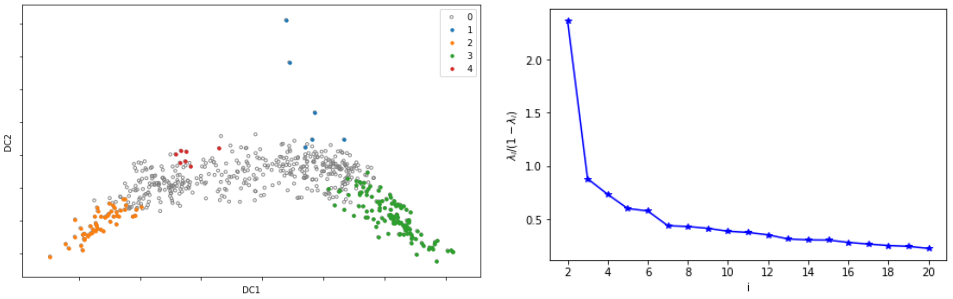


Figure A.18: DPT analysis for the *Moignard* data using distance estimation. We use the Gaussian kernel with bandwidth σ_{LAF}, P_{sym} and 5 components for computing the DPT distance. Although, there is a gap between the 5-th and 6-th eigenvalue, determining 5 branches instead of 4 does not lead to an appropriate result.

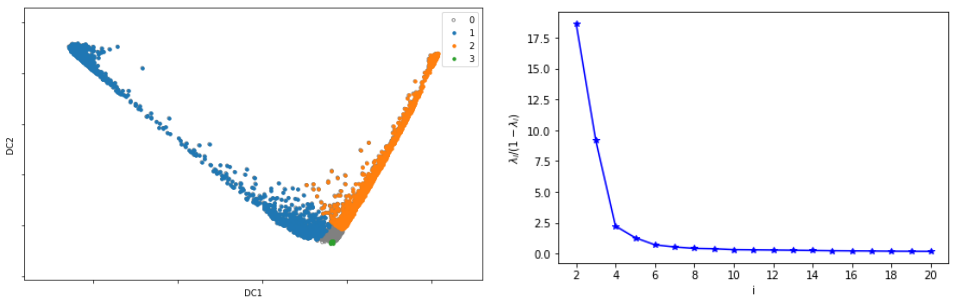


Figure A.19: DPT analysis for the *Goettgens* data using distance estimation in order to account for censored values.

References

[BK17] R. Banisch and P. Koltai. Understanding the geometry of transport: diffusion maps for lagrangian trajectory data unravel coherent sets. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(3):035804, 2017.

[BN01] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 14:586–691, 2001.

[CL06] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, 2006.

[EDVL13] E. Eirola, G. Doquire, M. Verleysen, and A. Lendasse. Distance estimation in numerical data sets with missing values. *Information Sciences*, 240:115–128, 2013.

[EKSX96] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[GHT+10] G. Guo, M. Huss, G. Q. Tong, C. Wang, L. L. Sun, N. D. Clarke, and P. Robson. Resolution of cell fate decisions revealed by single-cell gene expression analysis from zygote to blastocyst. *Developmental Cel*, 18:675–685, 2010.

[HBT15] L. Haghverdi, F. Buettner, and F.J. Theis. Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics*, 31:2989 – 2998, 2015.

[HBW+16] L. Haghverdi, M. Buettner, F.A. Wolf, F. Buettner, and F.J. Theis. Diffusion pseudotime robustly reconstructs lineage branching. *Nature Methods*, 13(10):845–848, 2016.

[Hot33] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2009.

[KMA+15] A. M. Klein, L. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. A. Weitz, and M. W. Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161:1187–1201, 2015.

[Laf04] S. Lafon. *Diffusion Maps and Geometric Harmonics*. Dissertation, Yale University, 2004.

[LV07] J.A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Information Science and Statistics. Springer, New York, 2007.

[MBF+16] E. Mass, I. Ballesteros, M. Farlik, F. Halbritter, P. GÃijnther, L. Crozet, C. E. Jacome-Galarza, K. Haendler, J. Klughammer, Y. Kobayashi, E. Gomez-Perdiguero, J. L. Schultze, M. Beyer, C. Bock, and F. Geissmann. Specification of tissue-resident macrophages during organogenesis. *Science*, 353, 2016.

[MMS+13] V. Moignard, I. C. Macaulay, G. Swiers, B. Buettner, J. Schuette, F. J. Calero-Nieto, S. Kinston, A. Joshi, R. Hannah, F. J. Theis, S. E. Jacobsen, M. F. de Bruijn, and B. Goettgens. Characterization of transcriptional networks in blood stem and progenitor cells using high-throughput single-cell gene expression analysis. *Nature Cell Biology*, 15(4):363–372, 2013.

[MvDW+17] K. R. Moon, D. van Dijk, Z. Wang, W. Chen, M. J. Hirn, R. R. Coifman, N. B. Ivanova, G. Wolf, and S. Krishnaswamy. PHATE: A dimensionality reduction method for visualizing trajectory structures in high-dimensional biological data. *bioRxiv*, 2017.

[MWH+15] V. Moignard, S. Woodhouse, L. Haghverdi, A. J. Lilly, Y. Tanaka, A. C. Wilkinson, F. Buettner, I. C. Macaulay, W. Jawaid, E. Diamanti, S. Nishikawa, N. Piterman, V. Kouskoff, F. J. Theis, J. Fisher, and B. Goettgens. Decoding the regulatory network of early blood development from single-cell gene expression measurements. *Nature Biotechnology*, 33(3):269–276, 2015.

[PAG+15] F. Paul, Y. Arkin, A. Giladi, D. A. Jaitin, E. Kenigsberg, H. Keren-Shaul, D. Winter, D. Lara-Asiaso, M. Gury, and A. et al. Weiner. Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell*, 163(7):1663–1677, 2015.

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559âĂŞ572, 1901.

[RCK+17] A. H. Rizvi, P. G. Camara, E. K. Kandror, T. J. Roberts, I. Schieren, T. Maniatis, and R. Rabadan. Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. *Nature biotechnology*, 35(6):551 – 560, 2017.

[RS00] S.T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[Sch13] A. Schwartz. Diffusion Maps und ihre Anwendung bei der Analyse von Automobildaten. Bachelor thesis, Institute for Numerical Simulation, University Bonn, 2013.

[TdSL00] J.B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[vdMH08] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[vL07] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[YYG+13] L. Yan, M. Yang, H. Guo, L. Yang, J. Wu, R. Li, P. Liu, Y. Lian, X. Zheng, J. Yan, J. Huang, M. Li, X. Wu, L. Wen, K. Lao, R. Li, J. Qiao, and F. Tang. Single-cell RNA-seq profiling of human preimplantation embryos and embryonic stem cells. *Nature Structural & Molecular Biology*, 20(9):1131–1139, 2013.