Diffusion-based Approaches to Visualization and Exploration of High-dimensional Data

Scott Anthony Gigante
Yale University Graduate School of Arts and Sciences, scottgigante@gmail.com

Follow this and additional works at: https://elischolar.library.yale.edu/gsas_dissertations

Recommended Citation
Gigante, Scott Anthony, "Diffusion-based Approaches to Visualization and Exploration of High-dimensional Data" (2021). Yale Graduate School of Arts and Sciences Dissertations. 162. https://elischolar.library.yale.edu/gsas_dissertations/162

This Dissertation is brought to you for free and open access by EliScholar – A Digital Platform for Scholarly Publishing at Yale. It has been accepted for inclusion in Yale Graduate School of Arts and Sciences Dissertations by an authorized administrator of EliScholar – A Digital Platform for Scholarly Publishing at Yale. For more information, please contact elischolar@yale.edu.
In recent years, modern technologies have enabled the collection of exponentially larger quantities of data in the biomedical domain and elsewhere. In particular, the advent of single-cell genomics has allowed for the collection of datasets containing hundreds of thousands of cells measured in tens of thousands of dimensions. This rapid expansion of common datasets beyond the possibility of manual annotation brings forth the need for large-scale exploratory data analysis. In this thesis, we will explore the problem of dimensionality reduction for visualization of high-dimensional datasets. Visualization of high-dimensional data is an essential task in exploratory data analysis, as the low-dimensional visualization of the data is used to understand, interrogate and present the results of many other analyses applied to the data. However, the repertoire of existing algorithms used for this task suffer from various algorithmic flaws leading to sub-optimal visualizations, including the trade-off between representing both local and global structure; the inherent sacrifices that must be made to reduce a dataset of intrinsic dimension greater than three to a form which can be interpreted by the human eye; and the computational complexity of the computations as the datasets increase in scale. Here, we use the framework provided by diffusion maps to present a new dimensionality reduction algorithm called PHATE, which seeks to address all three of these issues. In order to make the PHATE algorithm scalable, we present an approximation of the diffusion map through discrete partitions of the data called Compression-based Fast Diffusion Maps. Further, we use the insights gained from visualizing single-cell genomics data to present a manifold alignment algorithm called Harmonic Alignment, which allows for the correction of systemic differences between experiments, or the fusion of datasets collected from the same biological system using different assays. And finally, we present an extension of PHATE to longitudinal data, and demonstrate its utility for the purpose of machine learning interpretability by visualizing the hidden units of a neural network in training. While many open problems remain, the presentation of the methods herein chart a path towards a more systematic understanding of how we visualize high-dimensional data for exploratory data analysis.
Diffusion-based Approaches to Visualization and Exploration of High-dimensional Data

A Dissertation
Presented to the Faculty of the Graduate School
Of
Yale University
In Candidacy for the Degree of
Doctor of Philosophy
By
Scott Gigante

Dissertation Directors: Smita Krishnaswamy & Ronald Coifman

June 2021
# Contents

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>iv</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>2 Background</strong></td>
<td>7</td>
</tr>
<tr>
<td>2.1 Diffusion Maps</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Dimensionality reduction for visualization</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Single cell genomics</td>
<td>13</td>
</tr>
<tr>
<td><strong>3 Compressed diffusion</strong></td>
<td>19</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Problem setup</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Inverse-density MGC kernel</td>
<td>22</td>
</tr>
<tr>
<td>3.4 Compressed kernel and diffusion transitions</td>
<td>22</td>
</tr>
<tr>
<td>3.5 Compression-based fast diffusion maps</td>
<td>23</td>
</tr>
<tr>
<td>3.6 Empirical Results</td>
<td>27</td>
</tr>
<tr>
<td><strong>Appendices</strong></td>
<td>29</td>
</tr>
<tr>
<td>3.A Proof of Theorem 3.3.1</td>
<td>29</td>
</tr>
<tr>
<td>3.B Proof of Theorem 3.4.1</td>
<td>30</td>
</tr>
<tr>
<td>3.C Proof of Proposition 3.5.2</td>
<td>32</td>
</tr>
<tr>
<td>3.D Proof of Proposition 3.5.3</td>
<td>33</td>
</tr>
<tr>
<td><strong>4 Visualizing structure in high-dimensional data</strong></td>
<td>36</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Results</td>
<td>40</td>
</tr>
<tr>
<td>4.3 Discussion</td>
<td>67</td>
</tr>
</tbody>
</table>
## CONTENTS

4.4 Methods ................................................. 69

### Appendices

4.A Mathematical Principles of PHATE ........................................... 93
4.B Extended Comparison of PHATE to Other Methods ......................... 101
4.C PHATE for Data Exploration on Various Data Types ..................... 112
4.D Detailed EB Data Results ................................................ 118
4.E Supplementary videos .................................................... 120

5 Visualizing the PHATE of Neural Networks .................................. 136

5.1 Introduction ..................................................... 137
5.2 Background ....................................................... 139
5.3 Multiscale PHATE ............................................... 140
5.4 Results ......................................................... 142
5.5 Conclusion ....................................................... 147

### Appendices

5.A Multislice graph construction ............................................. 151
5.B Selection of representative subset $Y$ ..................................... 152
5.C Parameters for visualization methods comparison ....................... 153
5.D Continual Learning .................................................. 153
5.E Generalization ...................................................... 159
5.F M-PHATE parameters ............................................... 161
5.G Computing infrastructure .............................................. 161
5.H Validation data selection .............................................. 161

6 Harmonic Alignment .................................................. 167

6.1 Introduction ..................................................... 168
6.2 Problem setup ..................................................... 170
6.3 Preliminaries and background ........................................... 171
6.4 Harmonic alignment ................................................ 174
6.5 Numerical results .................................................. 177

### Appendices

6.A Algorithm ....................................................... 183
6.B Proof of Lemma 6.4.1 .................................................. 189
6.C Comparison to Wang and Mahadevan ................................. 192

7 Future Work ...................................................................... 197
  7.1 Hierarchical interactive visualization ................................. 197
  7.2 Perception-driven design ............................................... 199
  7.3 Selecting the right tool for the job .................................... 200

Bibliography ....................................................................... 203
Acknowledgements

It takes a village.

When I applied to PhD programs in the winter of 2016, while working as a research assistant under the tutelage of Chris Woodruff at the Walter & Eliza Hall Institute of Medical Research (WEHI) in Melbourne, I had no idea what I was signing up for. My work at WEHI had inspired me to pursue a career in academia, and a PhD seemed like the logical next step. I knew not what I wanted to study, nor what I wanted to achieve; simply that a PhD was the assumed next step in my educational journey.

Four years later, I have learned many things. I have learned a great deal about manifold learning and single-cell genomics, of course — but I have also learned a lot about myself, and what is important in life. I have learned that my greatest achievements will not be those listed on my résumé, but those shared with my friends and family. And it is in that spirit that I dedicate this thesis to those friends and family who helped me complete what is otherwise a long and solitary degree program.

To Mum, Sarah & Katelyn, who are always there for me: we will see each other soon. To Annalisa, who has taught me more than perhaps anyone else. To my New Haven crew: Kayleigh, Nina, Mila, Lex, and the production team at Jabroni Studios, who taught me the meaning of community. To Nick, who truly understood the struggle. To Maggie and Nathán, who gave me hope and something to fight for. To Laura, Misha, Natalie, and all of my friends in the Yale Camerata: we haven’t been able to sing together in over a year, but I hope that will change some time soon. To my labmates, especially my fellow graduate students Dan, Jay, Matt, Alex, Manik, Egbert, and Aarthi: thanks for keeping it fun. To all the members of Yale CBB, Yale Forward, the Gruber Fellows Steering Committee, the Endowment Justice Coalition, the Single Cell Open Problems team, and everyone who contributed to open source software with me. To Lisa, for keeping us on on track, no matter how much trouble we gave you. Thank you all.
Chapter 1

Introduction

Recent advances in data generation, from technology development to data collection and storage improvements, have led to an explosion in size and complexity of real-world datasets, both in the biomedical domain and elsewhere. Specifically, the advent of high-throughput DNA sequencing combined with the development of single-cell sequencing technologies (Tang et al., 2009; Kolodziejczyk et al., 2015) have led to an orders-of-magnitude increase in the quantity of genomic and epigenomic data generated for the characterization of gene expression (Angerer et al., 2017), chromatin accessibility (Buenrostro et al., 2015; Cusanovich et al., 2015), DNA methylation (Smallwood et al., 2014), and other genomic variables at the cellular level. The corresponding increase in dataset size is considerable: best-practices in bulk RNA sequencing recommend 6–12 replicates (Schurch et al., 2016), leading to experiments with tens of data points; a single single-cell assay can contain tens or hundreds of thousands of cells, with some experiments generating millions of cells (10x, 2017). This increase in dataset size has lead to an increased need for the application of techniques from data mining, machine learning and unsupervised data analysis to extract salient information from newly generated datasets (Kahn, 2011; Berger et al., 2013). In particular, a great deal of work has been done in single-cell transcriptomics to better visualize high-dimensional point clouds for the purpose of assisting researchers in the tasks of cell type identification, trajectory inference, and experimental effect quantification (Amir et al., 2013; Becht et al., 2019; Haghverdi et al., 2015; Burkhardt et al., 2021).

The use of dimensionality reduction for visualization as a first step in exploratory analysis of high-dimensional data is almost ubiquitous, especially in single-cell genomics (Luecken and Theis, 2019; Bacher and Kendzierski, 2016; Hwang et al., 2018). Current best practices involve visualization to better understand: batch effects, or the systematic difference in structure between assays or sets of assays (Haghverdi et al., 2018); cluster structure, or the number and type of clusters present in the data (Luecken and Theis, 2019);
trajectory structure, or the number and type of progressions and branchings present in the data [Saelens et al. 2019]; and cell type characterization [Bach et al. 2017; Shekhar et al. 2016], just to name a few. The widespread application of dimensionality reduction for visualization to single-cell data has led to a proliferation of techniques for visualizing such data, beginning with the first application of t-SNE [Maaten and Hinton 2008] to single-cell transcriptomics by Amir et al. (2013) through to more recent applications of diffusion maps [Angerer et al. 2016] and UMAP [Becht et al. 2019].

While the decision-making process for selecting the right dimensionality reduction algorithm (or indeed any kind of algorithm) for single-cell data is complex, the high-dimensional, non-linear and low signal-to-noise ratio of such datasets makes these data particularly suitable to the application of manifold learning [Moon et al. 2018]. Manifold learning is a field of unsupervised machine learning in which we assume data are drawn from a low-dimensional manifold embedded in high-dimensional space, and in reconstructing this low-dimensional manifold, we can better estimate distances between data points for downstream analysis. This assumption allows for a powerful simplification of the data analysis problem, as the high dimensionality of single-cell genomic data, often in the tens of thousands, presents computational and theoretical issues associated with the “curse of dimensionality” [Bellman 1957; Köppen 2000], as well as a strong mathematical foundation for algorithmic design [Coifman and Lafon 2006]. However, the choice between commonly used algorithms for visualizing single-cell data, manifold learning or otherwise, lack both theoretical and empirical justification. In particular, commonly used algorithms such as t-SNE [Maaten and Hinton 2008] and UMAP [McInnes et al. 2018] are known to perform suboptimally in the presence of trajectory data [Wattenberg et al. 2016; Kobak and Linderman 2021], which is common in single-cell genomic data [Moon et al. 2018].

Diffusion maps provide a mathematical framework for approximating low-dimensional manifolds from noisy, high-dimensional data [Coifman and Lafon 2006; Coifman et al. 2005]. The underlying principle of diffusion maps is that we can approximate the fundamental geometry of a manifold using properties of a graph built from noisy samples from said manifold. Specifically, we consider a diffusion process in which the transition probabilities from one data point to another are defined by a Gaussian kernel, decaying proportionately to the Euclidean distance between the data points. The eigenvectors of the matrix defining these pairwise transition probabilities give a basis for the data which, in the limit, approximates the eigenfunctions of the underlying manifold [Coifman and Hirn 2014; Coifman and Lafon 2006]. Furthermore, it has been shown that diffusion maps provide a useful basis on which to construct many different types of analyses of single-cell data, including visualization [Angerer et al. 2016], trajectory inference [Haghverdi et al. 2016], data denoising [Van Dijk et al. 2018], experimental effect quantification [Burkhardt et al. 2021], and more.

In this thesis, we explore applications of manifold learning using diffusion maps with the explicit goal
of visualizing high-dimensional data, especially data from single-cell genomics. Specifically, we develop a highly scalable algorithm to produce a fast approximation of the diffusion map using a two-step diffusion process between points and discrete partitions of the data we call Compression-based Fast Diffusion Maps; from this basis, we develop a dimensionality reduction algorithm called PHATE, which leverages diffusion maps to provide a low-dimensional visualization which preserves both local and global structure; we apply our learnings from the application of manifold learning to single-cell genomics to develop an algorithm we call Harmonic Alignment, which combines the insight that datasets with similar underlying biology but displaying substantial systemic differences (i.e., batch effects) will nonetheless have a similar low-dimensional manifold with recent advances in graph signal processing \cite{Shuman2013} to align multiple manifolds with shared biological characteristics; and finally, we extend the visualization to longitudinal and hierarchical data using a novel multislice graph construction, and apply this construction to the interpretation of the learning process of deep neural networks.

Bibliography

Our 1.3 million single cell dataset is ready to download, February 2017. URL \url{https://community.10xgenomics.com/t5/10x-Blog/Our-1-3-million-single-cell-dataset-is-ready-to-download/ba-p/276}.


Dmitry Kobak and George C Linderman. Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, pages 1–2, 2021. doi:10.1038/s41587-020-00809-z


Chapter 2

Background

2.1 Diffusion Maps

Diffusion maps provides a fundamental tool for estimating manifold distances from high-dimensional data. Specifically, given a dataset \( X = \{x_1, \ldots, x_n\} \) drawn from a Riemannian manifold \( M \) and a positive semidefinite function \( k(x,y) \), we can construct the Markov matrix

\[ M = D^{-1} A \]

where \( A \) is the \( n \times n \) affinity matrix with entries \( A(i,j) = k(x_i,x_j) \) and \( D \) is a diagonal degree matrix with entries \( D(i,i) = d(i) = \sum_j k(x_i,x_j) \). In the original construction of Nadler et al. (2005), we define \( k \) to be a Gaussian kernel, for which the Markov matrix approximates the diffusion of heat over the manifold. We hence compute the affinities between points as

\[ k(x_i,x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{\sigma^2} \right) \]

where \( \sigma \) is a constant defining the kernel bandwidth. Then, we can define the diffusion distances

\[ d^t(x_i,x_j)^2 = \sum_k (\phi_k(i) - \phi_k(j))^2 \exp(-2\lambda_k t) \]

where \( \phi_1,\ldots,\phi_n \) and \( \lambda_1,\ldots,\lambda_n \) are the eigenvectors and eigenvalues of \( M \) respectively, and \( t \) is a parameter denoting diffusion time, or the length of random walk over the manifold with transition probabilities defined by \( M \). Coifman and Lafon (2006) show that \( d^t \) converges to distances on \( M \) in the limit as the number of points sampled approaches infinity and the kernel bandwidth approaches zero. Further, we can
then define a new representation of the data called the diffusion map

\[ \Phi^t(x_i) = \left[ \lambda_1^t \phi_1(i) \cdots \lambda_n^t \phi_n(i) \right] \]

which gives a representation of the originally sampled data points such that Euclidean distances in the diffusion map are equal to diffusion distances. Further, using the assumption that the underlying manifold is low-dimensional, we can truncate the diffusion map to the first \( r \) eigenvectors, where \( r \) is set for example by a threshold below which \( \lambda_k \) nears zero and its contribution to the diffusion distance is thus negligible.

Diffusion maps and related techniques have been used for many applications in single-cell biology, including visualization [Angerer et al. (2016)], trajectory inference [Haghverdi et al. (2016)], and denoising [Van Dijk et al. (2018)]. The success of diffusion-based methods when specifically applied to single-cell methods can be attributed to the curse of dimensionality, which states that, as the number of dimensions grows, the Euclidean distance between random pairs of points approaches a constant; that is, Euclidean distances become largely meaningless [Marimont and Shapiro (1979)]. Since single-cell methods are typically applied to a dataset with at least as many dimensions as there are genes, these datasets are typically very high-dimensional, ranging in the tens of thousands of dimensions, which necessitates approaches like diffusion maps to address this so-called “curse”.

### 2.2 Dimensionality reduction for visualization

Dimensionality reduction refers to a broad category of algorithms designed to represent a set of points \( x \in \mathbb{R}^m \) in a lower dimensional space \( \mathbb{R}^d \) such that some property of the set is retained across the two representations. Applications of such algorithms can broadly be categorized into two groups: applications which generate the lower-dimensional embedding as a preprocessing step before applying some other algorithm, and applications which produce the low-dimensional embedding for the purpose of visualization [Sedlmair et al. (2012)].

Here, we focus on the subset of dimensionality reduction algorithms used explicitly for visualization. Many such algorithms exist, some of which were designed specifically for this application (e.g. MDS [Cox and Cox (2008)], ISOMAP [Tenenbaum et al. (2000)], t-SNE [Maaten and Hinton (2008)] and force-directed layouts [Fruchterman and Reingold (1991)]). Each of these algorithms seek to minimize some objective function \( f : \mathbb{R}^m, \mathbb{R}^d \to \mathbb{R} \) which defines the imperfection between the low-dimensional representation and the ambient data, such that the algorithm output gives

\[ \hat{Y} = \arg \min_Y f(X, Y) \]
where \( X \) is the matrix of input data points. Additionally, there exist some algorithms frequently used for visualization which were not designed for this task, but are sometimes used for visualization by truncating to two of three dimensions (e.g., PCA [Abdi and Williams 2010], diffusion maps [Coifman and Lafon 2006]). Here, the algorithm defines an objective function \( f : \mathbb{R}^m, \mathbb{R}^k \rightarrow \mathbb{R} \) which defines the imperfection between some representation of the data (it need not be low-dimensional) and the ambient data, such that the algorithm output gives

\[
\hat{W} = \arg \min_W f(X, W)
\]

and then the visualization \( \hat{Y} \) is constructed by simply truncating \( \hat{W} \) to two of three dimensions.

Within dimensionality reduction algorithms used for visualization, one can categorize the commonly used algorithms into three groups: algorithms which emphasize global structure (PCA, MDS, diffusion maps); algorithms which emphasize local structure (t-SNE, UMAP), and algorithms which attempt to balance both (ISOMAP, force-directed layout). It is clear that only emphasizing one aspect of a dataset’s structure will lead to an imperfect representation of the data; however, single it is impossible to represent high-rank data in low dimensions without sacrificing some aspect of the data structure, even those algorithms which attempt to balance both local and global structure will produce an imperfect representation of the data. Selecting the right algorithm, then, depends on the user’s goal.

Sedlmair et al. (2012) characterize the use of dimensionality reduction for visualization in the literature as primarily fitting into the following use cases: grouping together like points; confirming likeness of pre-labeled groups; annotating trajectories; and confirming pre-annotated trajectories. Etemadpour et al. (2015) provide an alternative grouping of tasks, defining pattern identification (e.g., counting clusters or outliers), behavior comparison (e.g., comparing point densities), member disambiguation (e.g., estimating the number of points in a group), and relation seeking (e.g., finding points most similar to another point). Since sacrifices must be made in the reduction of dimensionality, the optimal choice of algorithm must therefore depend on the task at hand. Below, we briefly describe the algorithms of the most commonly used dimensionality reduction algorithms, their strengths, and their weaknesses.

### 2.2.1 PCA

PCA [Abdi and Williams 2010] is a dimensionality reduction algorithm which finds an orthogonal basis \( \{u_1, u_2, \ldots, u_k\} \) which maximizes the variance of each successive basis vector \( u_i \), such that

\[
u_1 = \sum_{i=0}^{m} a_i x_i \text{ where } \{a_1, \ldots, a_m\} = \arg \max (\text{Var} u_1), a_i \in \mathbb{R}
\]
and

\[ u_{j+1} = \sum_{i=0}^{m} a_i x_i \text{ where } \{a_1, \ldots, a_m\} = \arg \max_{u \perp u_j} \{ \text{Var} \ u_{j+1} \}, a_i \in \mathbb{R} \]

where \( x_i \) refers to the \( i \)th canonical basis vector of a dataset \( X \) with \( n \) samples and \( m \) features, and \( k = \text{rank } X \).

In practice, PCA is generally computed as an eigendecomposition problem. It is trivial to show that the solution to the above maximization is equal to the eigendecomposition of the covariance matrix; for computational reasons, one may simplify this to the singular value decomposition of the mean-centered data matrix \( Y \) where

\[ Y_{ij} = x_{ij} - \frac{1}{n} \sum_{k=0}^{n} x_{kj} \]

such that the matrix solution

\[ \text{PCA}(X) = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \end{bmatrix} \]

can be computed by

\[ \text{PCA}(X) = U \Sigma \text{ where } Y = U \Sigma V^T \]

We note here two important features of PCA as it pertains to dimensionality reduction for visualization. First, PCA is a linear method; that is, every principle component is a linear combination of the canonical basis vectors, or features in the ambient space. This means that complex, nonlinear features, as are often found in real-world data, cannot be represented in a single visual dimension using PCA. Second, PCA produces as many components as the data has rank (generally \( \min(n, m) \) for real data) which requires the user to a) discard most of the dimensions produced (or indeed avoid computing them altogether by using a partial eigendecomposition) and b) examine more than just the two of three dimensions we are most comfortable with. A common approach to this issue is the examine pairwise plots of more than just the first two dimensions in what is known as a scatterplot matrix, though this is generally less effective than simply changing to a different dimensionality reduction method (Sedlmair et al., 2013).
2.2.2 MDS

MDS \cite{KruskalWish1978, CoxCox2008} is a class of dimensionality reduction algorithms which aim to minimize the discrepancy between pairwise distances computed in the ambient space and pairwise distances computed in the low-dimensional space. The most general form of MDS can be written as follows:

$$\text{MDS}(X) = \arg \min_{Y} f(\text{dist}(X), \text{dist}(Y))$$

with

$$\text{dist}(X) = \{d(x_i, x_j) : x_i, x_j \in X, i \neq j\}$$

where $x_i$ and $y_i$ are the $i$th samples of $X$ and $Y$ respectively, $d$ is a distance function, and $f$ is a loss function. One of the most commonly used forms of MDS used explicitly for data visualization is metric MDS, in which the distance function $d$ is defined as Euclidean distance, and the loss function $f$ is the residual sum of squares (known here as the “stress”)

$$f(\text{dist}(X), \text{dist}(Y)) = \sqrt{\sum_{i \neq j} (d(x_i, x_j) - d(y_i, y_j))^2}$$

Further modifications can be made to MDS to allow for non-Euclidean distances and non-monotonic relationships between $\text{dist}(X)$ and $\text{dist}(Y)$, the latter of which is known as non-metric MDS.

MDS can be thought of as the first dimensionality reduction algorithm designed explicitly for visualization. However, its reliance on all pairwise distances leads to an overwhelming weakness in high-dimensional data: the curse of dimensionality \cite{Bellman1957}. Firstly, these pairwise distances can be overwhelmingly time-consuming to compute; secondly, long-range Euclidean distances become increasingly meaningless as the number of dimensions increases \cite{Koppen2000}. Therefore, since the full pairwise distance matrix is dominated by long-range distances, the placement of a point to its nearest neighbors has relatively little influence over the final coordinate embedding, producing highly noisy and sometimes unintuitive visualizations.

2.2.3 ISOMAP

ISOMAP \cite{Tenenbaumetal2000} seeks to address the problems stated above by incorporating nearest-neighbor graph structure into the dimensionality reduction algorithm. Like MDS, ISOMAP seeks to match distances in low dimensions to distances in high dimensions; however, instead of using Euclidean distances,
ISOMAP using geodesic distances calculated on a nearest neighbor graph. Define the graph

\[ G = (V, E), \quad V = \{ x_i : x_i \in X \}, \quad E = \{ (x_i, x_j, d_{ij}) : x_i, x_j \in X, i \neq j \} \]

where

\[ d_{ij} = \begin{cases} \| x_i - x_j \|_2^2 & \text{if } x_i \in N(x_j) \text{ or } x_j \in N(x_i) \\ \infty & \text{otherwise.} \end{cases} \]

where \( N(x) \) denotes the nearest neighbors of \( x \). We can then compute geodesic (or “shortest path”) distances on \( G \), after which MDS can be applied as normal.

This move from Euclidean to geodesic distances removes the aforementioned long-range Euclidean distances from the computation, and furthermore, enhances the speed of the algorithm through the use of nearest neighbor search algorithms, which need not compute all pairwise distances (Nene and Nayar, 1997; Kushilevitz et al., 2000; Malkov and Yashunin, 2018). Furthermore, Tenenbaum et al. (2000) show that in the noiseless case, these geodesic distances converge exactly to manifold distances, giving an optimal embedding of the data; however, this noiseless assumption is a strong one, and Balasubramanian and Schwartz (2002) showed shortly thereafter that ISOMAP fails to produce an optimal embedding in the case of noisy data.

### 2.2.4 t-SNE

t-SNE (Maaten and Hinton, 2008) is by far the most popular dimensionality reduction algorithm used for visualization over the last 15 years. t-SNE builds off an earlier algorithm called Stochastic Neighbor Embedding (SNE) proposed by Hinton and Roweis (2002) in which points are positioned in low-dimensional space based exclusively on the distances to their near neighbors. Specifically, t-SNE computes an asymmetric affinity of each point to its nearest neighbors by an adaptive-bandwidth Gaussian kernel

\[ k(x_i, x_j) = \exp \left( -\frac{\| x_i - x_j \|^2}{2\sigma^2} \right) \]

giving rise to a Markov matrix \( P \) with transition probabilities proportional to the affinities

\[ p_{ji} = \frac{k(x_i, x_j)}{\sum_{k \neq i} k(x_i, x_k)} \]

where the bandwidth \( \sigma \) is tuned per-point via binary search in order to achieve a desired “perplexity” (or effective nearest neighbors)
CHAPTER 2. BACKGROUND

Perp(x_i) = 2^{H(P_i)} where \( H(P_i) = - \sum_j p_{ji} \log_2 p_{ji} \).

These affinities are then symmetrized as

\[
p_{ij} = p_{ji} = \frac{p_{ji} + p_{ij}}{2}
\]

and compared to low-dimensional affinities defined by the Student’s t-distribution

\[
q_{ij} = \frac{(1 + \|y_i - y_j\|_2^2)^{-1}}{\sum_{k \neq i} (1 + \|y_k - y_i\|_2^2)^{-1}}
\]

via the KL divergence

\[
KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}.
\]

This optimization, which focuses exclusively on short-range connections, provides a visualization which is much less susceptible to noise and places points close to their nearest neighbors in the low-dimensional space. However, it suffers from a lack of global structure, since the global structure is not encoded anywhere in the algorithm. Specifically, t-SNE suffers from a problem known as “shattering” in which groups of points that are similar can be separated by random initialization, after which the position of these clusters relative to one another is totally random (Wattenberg et al., 2016).

A more recent modification to the t-SNE algorithm, known as UMAP (McInnes et al., 2018), has grown in popularity in recent years (Becht et al., 2019). However, the apparent improvement of UMAP over t-SNE, obfuscated in part by its grounding in topological data analysis, can be explained entirely by an improved initialization via Laplacian Eigenmaps (Belkin and Niyogi, 2002) and a computationally optimized implementation (Kobak and Linderman, 2021).

2.3 Single cell genomics

Single-cell genomics (sometimes single-cell -omics) is used to describe a wide array of cellular assays which capture biological information at the single-cell level. The most prominent of these is single-cell transcriptomics, or single-cell RNA sequencing, which is the extension of bulk RNA sequencing (Ozsolak and Milos, 2011; Stark et al., 2019) to the single-cell level. Briefly, RNA sequencing works by extracting RNA molecules from the nuclear and cytoplasm of a cell, reverse transcribing the RNA to DNA, and sequencing these
molecules to count the number of transcripts of each gene present in the sample. Single-cell sequencing extends this assay to the cellular level through the addition of unique DNA tags added to each reverse-transcribed RNA transcript, frequently through the use of microfluidics (Klein et al., 2015; Macosko et al., 2015; Zheng et al., 2017).

In recent years, the ability to use DNA sequencing to characterize individual cells in a sample has grown rapidly, both in terms of the quantity of cells able to be sequenced (Svensson et al., 2018) and in terms of the characteristics capable of being sequenced, including DNA sequence, gene expression, DNA modifications, chromatin accessibility, chromosome conformation, and more (Zhu et al., 2020). The rapid increase in data volume has led to many computational challenges in scalability (Lähnemann et al., 2020), data normalization (Vallejos et al., 2017), data integration (Stuart and Satija, 2019) and other related stages of the single-cell analysis pipeline.

Many analytical tools for single-cell analysis use the manifold assumption as a fundamental simplifying assumption to drive algorithm development (Moon et al., 2018). Put simply, we assume that data are drawn non-uniformly from a collection of low-dimensional manifolds (representing the cellular state space) embedded in high-dimensional space (the space of measured cellular characteristics) in which short Euclidean distances in the high-dimensional space correspond to short manifold distances (Lin and Zha, 2008). This assumption can be easily understood by establishing the contrapositive; if the cellular state space did not exist in a lower dimension than the space of measured cellular characteristics, then every possible combination of characteristics would be valid. This is clearly not true when we consider the interdependence of gene expression, chromatin conformation and other cellular characteristics commonly measured in such assays.

By making this assumption, we can now address many of the computational and theoretical issues raised by the “curse of dimensionality”; specifically, 1) that the application of any algorithm which scales in the number of dimensions becomes computationally intractable; and 2) that in high dimensions, the Euclidean distances between all points converge to the same value, making comparisons between points nigh impossible. The manifold assumption, which allows us to effectively reduce the dimensionality of our data, will guide the analysis of all high-dimensional data, including single-cell data, presented in this work.

**Bibliography**


Philipp Angerer, Laleh Haghverdi, Maren Büttner, Fabian J Theis, Carsten Marr, and Florian Buet-


Dmitry Kobak and George C Linderman. Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, pages 1–2, 2021, doi:10.1038/s41587-020-00809-z


Chapter 3

Compressed diffusion


In its simplest form, the diffusion maps algorithm has cubic complexity, due either to the powering of the diffusion operator or the computation of the full eigendecomposition thereof. Partial eigendecomposition can speed this up to \( \mathcal{O}(n^2k) \) where \( k \) is the number of diffusion components computed. However, the diffusion operator itself is also a useful tool in analysis of the manifold (as we will show in Chapter 4), and for this reason, it is of interest to find a similar operator of smaller size that captures this structure without sacrificing accuracy.

In the following work, we define a compressed affinity matrix and corresponding diffusion operator which retains the structure of the diffusion operator; further, the diffusion map computed on this compressed operator can be interpolated to compute a close approximation to the full diffusion map. By doing so, we create a matrix whose size depends only on the complexity of the manifold, rather than the number of points sampled thereof; this will be useful in downstream analysis, particularly in the computation of PHATE (Moon et al., 2019), which using the powered diffusion operator as the basis of a low-dimensional representation of the data.

*: Equal contribution. †: Equal contribution.
Contributions

As a co-first author of this work, my contributions were primarily twofold: I co-developed the proofs with Jay Stanley, and I wrote the code to compute and summarize the empirical experiments comparing our Compression-based Fast Diffusion Maps with alternative methods.

3.1 Introduction

Manifold learning approaches are often used for modeling and uncovering intrinsic low dimensional structure in high dimensional data (e.g., in genomics Moon et al. (2018)). Diffusion maps Coifman and Lafon (2006), in particular, are a popular method that capture data manifolds with random walks that propagate through nonlinear pathways in the data. Transition probabilities of a Markovian diffusion process define an intrinsic diffusion distance metric that is amenable to low dimensional embedding. Indeed, by arranging transition probabilities in a row-stochastic diffusion operator, and taking its leading eigenvalues and eigenvectors, one can derive a small set of coordinates where diffusion distances are approximated as Euclidean distances, and intrinsic manifold structures are revealed.

While the embedding provided by diffusion maps is useful for data analysis, it is also challenging to apply to modern “big data” settings due to scalability issues. As is often the case with kernel methods, diffusion maps require the computation of pointwise transition probabilities, which is computationally quadratic in the size of the data, and an eigendecomposition which is cubic if all eigenvectors are computed. Here, we show that one can reduce the computational cost significantly by only considering transition probabilities between data regions, which can be efficiently computed from a compressed affinity kernel over a fixed-size partition of the data. Our construction is based on coarse-graining a measure-based kernel Bermanis et al. (2013, 2016) with an inverse-density measure, which was recently introduced in Lindenbaum et al. (2018) for geometry-based data generation and uniform resampling of data manifolds. Here we extend the uses of this kernel to provide efficient implementation and application of diffusion maps by first embedding data regions, and then interpolating the pointwise embedding from their diffusion coordinates.

The main contributions of this work are as follows. On the theoretical side, we further establish the relations between the original diffusion framework from Coifman and Lafon (2006) and the construction in Lindenbaum et al. (2018), both at a pointwise and compressed (i.e., data-region) level. On the practical side, we suggest a novel partitioning method, the results of which indicate significant speedups in the computation of the diffusion embedding, which outperform other approaches, and enables the application of diffusion-based manifold learning well beyond the data sizes traditionally used with kernel methods.
CHAPTER 3. COMPRESSED DIFFUSION

3.2 Problem setup

3.2.1 Preliminaries

Let $\mathcal{M} \subseteq \mathbb{R}^m$ be a compact $d$ dimensional manifold immersed in the ambient space $\mathbb{R}^m$, where $d \ll m$, which represents the intrinsic geometry of data sampled from it. For simplicity, the integration notation $\int \cdot dy$ in this paper will refer to the Lebesgue integral $\int_{\mathcal{M}} \cdot dy$ over the manifold, instead of the whole space $\mathbb{R}^n$. Further, while (for simplicity) such integrals are written without a specific measure one can equivalently, w.l.o.g., replace $dx$ with an appropriate measure representing data sampling distribution over $\mathcal{M}$. Let

$$g(x, y) \equiv \exp\left(-\frac{\|x - y\|^2}{\varepsilon}\right), \quad x, y \in \mathcal{M}, \quad \varepsilon > 0,$$

define the Gaussian kernel used in Coifman and Lafon (2006) to capture local neighborhoods from data sampled from $\mathcal{M}$. Following Coifman and Lafon (2006) and related work, we define the Gaussian degree $q(x) = \|g(x, \cdot)\|_1 = \int g(x, y)dy$ and assume it provides a suitable approximation of the distribution (or local density) of data over the manifold $\mathcal{M}$. Finally, given a measure $\mu$ over the manifold, an MGC kernel is defined as

$$k_\mu(x, y) = \int g(x, r)g(y, r)d\mu(r).$$

Note that while we use a Gaussian kernel for the remainder of this work, the definitions and theorems to follow do not depend on the choice of $g$, so long as it is a kernel function.

The original MGC construction in Bermanis et al. (2016) considered measures that represent data distribution, and used the constructed MGC kernel to define diffusion maps (see Sec. 3.2.2 and Coifman and Lafon (2006)) with them. Recently, it was shown in Lindenbaum et al. (2018) that an MGC kernel constructed with inverse-density measure allows separation of data geometry and distribution, which in turn allows uniform data generation over the data manifold, with applications in alleviating sampling biases in data analysis (e.g., imbalanced classification). Here, we further explore the inverse-density MGC kernel and its properties. In particular, we show this kernel enables compression of its resulting diffusion geometry into data regions, instead of data points, to efficiently capture the intrinsic manifold geometry of analyzed data.

3.2.2 Diffusion Maps

Diffusion maps utilize a set of local affinities to define a Markovian diffusion process over analyzed data, which captures the intrinsic data geometry via diffusion distances. The original construction defines transition probabilities between data points based on Gaussian affinities as $p(x, y) = g(x, y)/q(x)$, where it can be verified that $\int p(x, y)dy = 1$. This construction can also be generalized to other affinity kernels, such as the MGC kernel $k_\mu(x, y)$ from Bermanis et al. (2016) or variation in Lindenbaum et al. (2018). Under mild conditions on the affinity kernel, the resulting transition probability operator has a discrete decaying spectrum of eigenvalues $1 = \lambda_0 \geq |\lambda_1| \geq |\lambda_2| \geq \ldots$ which are...
used together with their corresponding eigenfunctions $\phi_0, \phi_1, \phi_2, \ldots$ (with $\phi_0$ being constant) to achieve the diffusion map of the data. Each data point $x \in M$ is embedded by this diffusion map to the diffusion coordinates $\Phi_t(x) = (\lambda_1^t \phi_1(x), \ldots, \lambda_\delta^t \phi_\delta(x))$, where the exact value of $\delta$ depends on the spectrum of the transition probabilities operator $Pf(x) = \int p(x,y)f(y)dy$, whose kernel is $p(x,y)$.

### 3.3 Inverse-density MGC kernel

Our construction here is based on an inverse-density MGC kernel, first introduced in [Lindenbaum et al. (2018)](#), which is defined as follows:

**Definition 3.3.1.** The inverse-density MGC (ID-MGC) kernel is defined as $k(x,y) = \int g(x,r)g(r,y)q(r)dr$, $x,y \in M$, with the associated integral operator $Kf(x) = \int k(x,y)f(y)dy$.

This kernel corresponds to $k_\mu(x,y)$ with $d\mu(x) = q^{-1}(x)dx$, where in practice $q(x)$ accounts (up to normalization) for data density over $M$. Therefore, the connectivity captured by this kernel normalizes density variations by enhancing relations in sparse regions compared to dense ones. Indeed, in [Lindenbaum et al. (2018)](#) this property was used to adjust the distribution of data and provide uniform resampling of data manifolds. We note that since we consider $M$ as representing the intrinsic geometry of collected (or observed) data, we expect a certain amount of data points to exist in each local region, and thus $q^{-1}(x)$ can be expected to have a finite upper bound over $M$.

Several useful properties of the ID-MGC kernel and its relation to the Gaussian-based diffusion operator in Sec. 3.2.2 are summarized in the following theorem (proof of this proposition is found in Appendix 3.A):

**Theorem 3.3.1.** Let the kernel $k(\cdot, \cdot)$ and operator $K$ be defined as in Def. 3.3.1. Then, this kernel (and operator) can be related to the Gaussian kernel $G$ and diffusion operator $P$ via

1. the operator norm: $\|K\| \leq \|G\|$;
2. the kernel degrees: $\|k(x, \cdot)\|_1 = \|g(x, \cdot)\|_1$, $x \in M$; and
3. two-step diffusion: $P^2f(x) = \int \frac{k(x,y)}{\|k(x, \cdot)\|_1} f(y)dy$.

### 3.4 Compressed kernel and diffusion transitions

Let $S_1, \ldots, S_n \subset M$ be a partition of the manifold into measurable subsets, such that $M = \bigcup_{j=1}^n S_j$, and $S_i \cap S_j = \emptyset$ for every $i \neq j$. We define a compressed kernel over such a partition as follows:
Definition 3.4.1. Let \( k(x, y) \), \( x, y \in \mathcal{M} \) be defined as in Def. \[3.3.1\] The compressed kernel over partition \( S = \{ S_j \}_{j=1}^n \) of \( \mathcal{M} \) is given by

\[
k_S(S, T) = \int_S \int_T k(x, y) dx dy, \quad S, T \in \mathcal{S},
\]

with the corresponding \( n \times n \) kernel matrix given by

\[
[K_S]_{ij} = k(S_i, S_j), \quad i, j = 1, \ldots, n.
\]

Similar to the original diffusion map construction in Sec. \[3.2.2\], we normalize the compressed kernel to get diffusion probabilities \( p_S(S, T) = k(S, T)/\|k(S, \cdot)\|_1 \), organized in an \( n \times n \) row-stochastic matrix \( P_S \). This matrix captures diffusion transition probabilities between data regions. The relation between the compressed construction and the original diffusion framework is summarized in the following theorem (proof of this proposition is found in Appendix \[3.B\]):

Theorem 3.4.1. The compressed construction here can be related to the pointwise one in Sec. \[3.2.2\] via

1. operator norm (in matrix & operator form): \( \|K_S\| \leq O(\|G\|) \) with a constant that only depends on the finite volume of \( \mathcal{M} \);
2. kernel degrees: \( \|k_S(S, \cdot)\|_1 = \int_S \|g(x, \cdot)\|_1 dx, \quad S \in S \); and
3. diffusion probabilities:

\[
P_S(S, T) = \int \Pr[S \overset{1 \text{ step}}{\sim} r] \Pr[r \overset{1 \text{ step}}{\sim} T] dr
\]

where

\[
\Pr[r \overset{1 \text{ step}}{\sim} T] = \int_T p(r, y) dy \quad \text{and} \quad \Pr[S \overset{1 \text{ step}}{\sim} r] = \int_S p(x, r) \Pr[x \mid S] dx
\]

with prior \( \Pr[x \mid S] = \|g(x, \cdot)\|_1 / \|k_S(S, \cdot)\|_1 \).

3.5 Compression-based fast diffusion maps

The compressed construction of \( K_S \) and \( P_S \) gives rise to a natural approximation of diffusion maps. Given a partitioning \( S \), we define a compressed diffusion map \( \Phi^S_t : S \to \mathbb{R}^d \) analogous to the pointwise one in \[3.2.2\] using eigenvectors of \( P_S \) (with corresponding eigenvalues) in lieu of the eigenfunctions of \( P \). This diffusion map provides an embedding of data regions, rather than points, which represents a coarse version of the diffusion geometry over \( \mathcal{M} \). Then, using the region-to-point transition probabilities \( \Pr[S \overset{1 \text{ step}}{\sim} x] \)
from Theorem 3.4.1, we approximate the pointwise diffusion map as \( \tilde{\Phi}_t(x) = \sum_{j=1}^{n} \Phi_t(S_j) \Pr[S_j \rightsquigarrow x] \).

Results in Sec. 3.6 indicate that with the proper choice of partitions in \( S \), the compressed diffusion map \( \tilde{\Phi}_t \) provides a good approximation of the original pointwise one from Coifman and Lafon (2006), while also providing significant scalability and performance advantages over both direct computation and other alternative approximations.

It now remains to derive a strategy for efficiently partitioning the data manifold such that compression over \( S \) will suitably capture (albeit at a coarser level) the pointwise diffusion geometry defined by \( P \). To this end, it is convenient to consider diffusion affinities and distances Coifman and Lafon (2006); Bermanis et al. (2013, 2016), rather than transition probabilities. These are defined as follows:

**Definition 3.5.1.** The diffusion affinity Coifman and Lafon (2006) kernel is the symmetric conjugate of \( P \), which is defined spectrally as

\[
Af(x) = \sum_{j=1}^{\infty} \lambda_j \langle \psi_j, f \rangle \psi_j(x)
\]

with \( \psi_j(x) = q^{j/2}(x) \phi_j(x) \).

**Definition 3.5.2.** The diffusion distance Coifman and Lafon (2006) between two points \( x \) and \( y \) sampled from \( M \) is defined via the kernel function \( a^t(\cdot, \cdot) \) of \( A^t \), and equivalently via its eigendecomposition as

\[
[D^t(x, y)]^2 = D_2^t(x, y) = \left\| a^t(x, \cdot) - a^t(y, \cdot) \right\|^2_2 = \sum_{j=0}^{\infty} \lambda_j^2 \left[ \psi_j(x) - \psi_j(y) \right]^2
\]

We note that in the described compressed diffusion scheme, we replace pointwise affinities with affinities between data points and partitions, rather than just between partitions. Under proper partitioning of the data, we expect (or assume) such substitution would retain intrinsic structure of the data, as captured by the diffusion geometry defined by \( P \) or \( A \). The following proposition examines the difference between pointwise and coarse-grained diffusion affinity information, and relates this difference to locality of partitions with respect to the diffusion distance metric.

**Proposition 3.5.1.** Let \( S \in S \) and \( \xi > 0 \). If the diffusion affinities satisfy

\[
\sup_{x,\xi \in S_j} \left| a(x, z) - \mathbb{E}_{y \in S_j}[a(y, z)] \right| < \xi,
\]

then the diffusion distances \( D^t(x, y) \), \( x, y \in S \), are bounded from above by \( O(\xi) \), with a constant that only depends on \( \text{vol}M \).

**Proof.** For any \( x, y \in S_j \) and \( z \in M \), due to the triangle inequality, we have

\[
|a(x, z) - a(y, z)| \leq |a(x, z) - \mathbb{E}_{u \in S_j}[a(u, z)]| + |a(y, z) - \mathbb{E}_{u \in S_j}[a(u, z)]| \leq 2\varepsilon.
\]
Figure 3.1: Example of approximations of diffusion maps on the Swiss Roll. Landmarks or centroids are shown in green. Mean squared error (MSE) of each approximation is shown in parentheses. Color shows MSE for each point.
Therefore, \[\int |a(x, z) - a(y, z)|^2 dz \leq 4\varepsilon^2 \text{vol} M,\] which together with Definition 3.5.2 yields the result in the proposition.

Indeed, this result indicates that to faithfully capture the diffusion geometry by given partitioning \(S\), the partitions should be local in the diffusion geometry, since the diffusion geometry in turn preserves the local geometry of the manifold [Coifman and Lafon (2006)]. This understanding motivates our formulation of a partitioning strategy, as explained below.

Here we turn to practical settings of data analysis applications. Therefore, in the following we consider finite settings, particularly for some dataset \(X \subset M, |X| < \infty\), sampled from the manifold, with \(P\) and \(A\) being \(|X| \times |X|\) matrices constructed from it. Note also that in the this setting, the eigenfunctions of \(P\) or \(A\) become eigenvectors (i.e., in \(\mathbb{R}^{|X|}\)), and the RHS in Definition 3.5.2 is written analogously with the sum going over \(|X|\) eigenpairs.

In order to define a partition on \(X\), we first choose a set of partition centroids \(L\). Noting the success of coherence-based sampling strategies for signal compression and recovery [Puy et al. (2018)], we propose a random sampling without replacement biased by coherence, where we estimate the coherence as follows.

**Proposition 3.5.2.** Let \(\mu_{\ell}(x) = \sum_{j=0}^{\ell} |\psi_j(x)|^2\) be the order-\(\ell\) coherence of \(x \in X\) as in [Puy et al. (2018)]. Then, the \(t\)-step diffusion coherence \(\rho_t(x) = \|a^t(x, \cdot)\|_2^2\) is an estimate of \(\mu_{\ell}(x)\) with error

\[\|\mu_{\ell}(x) - \rho_t(x)\|_2^2 \propto \ell - \sum_{j=0}^{|X|-1} \lambda_j^2 t\]

for every \(x \in X\).

Proof of this proposition can be seen in Appendix 3.C.

According to Proposition 3.5.1, we should assign \(x \in X\) to the partition associated with centroid \(y \in L\) such that \(D_2^t(x, y)\) is minimized. However, this distance is in practice biased by the diffusion coherence \(\rho_t(y)\). We therefore define the *angular diffusion distance*, which we will show maximizes transition probabilities between points and the assigned centroids.

**Definition 3.5.3.** The angular diffusion distance between \(x, y \in M\) is defined using the eigendecomposition of the diffusion affinity kernel from Def. 3.5.1 as

\[D_\ell^t(x, y) = \arccos \left( \frac{\sum_{j=0}^{\ell} \lambda_j^{2t} \psi_j(x) \psi_j(y)}{\sqrt{\rho_t(x) \rho_t(y)}} \right).\]

**Proposition 3.5.3.** For \(x \in M\), the following dualities hold:
CHAPTER 3. COMPRESSED DIFFUSION

Figure 3.2: Quantitative evaluations. Left: Runtime scaling for 150 partitions over an increasing number of points. Middle: Approximation error on 32 diffusion components over an increasing number of partitions. Right: Approximation error compared to runtime over $2^{14}$ points.

1. $\arg\min_{y \in \mathcal{M}} D^c_t(x,y) = \arg\max_{y \in \mathcal{M}} \frac{a^{2t}(x,y)}{\rho_t(x) \rho_t(y)}$;

2. $\arg\min_{y \in \mathcal{M}} \left\| \frac{a^t(x, \cdot)}{\rho_t(x)} - \frac{a^t(y, \cdot)}{\rho_t(y)} \right\|_2^2 = \arg\min_{y \in \mathcal{M}} D^c_t(x,y)$.

Proof of this proposition can be seen in Appendix 3.D. Propositions 3.5.2 and 3.5.3 provide a convenient method to select a set of partitions that optimize proposition 3.5.1. To do this, we assign points $x \in X$ to partitions $S_i \in S$ corresponding to centroids $y_i \in L$ according to

$$S_i = \{ x \in X : D^c_t(x,y_i) = \arg\min_{y \in L} D^c_t(x,y) \}.$$

3.6 Empirical Results

3.6.1 Swiss roll

Here we compare three techniques for approximation of diffusion maps: linear interpolated diffusion on cluster centroids, volume-weighted Nystrom extension Long and Ferguson (2019), and our proposed compression-based fast diffusion map (CFDM).

The Swiss roll is a canonical test dataset for diffusion maps, and manifold learning in general. Figure 3.1 shows the diffusion map of the Swiss roll and each approximation, with the partition diffusion coordinates plotted in green. To quantify the reconstruction, we calculated the sum of squared error (SSE) across 32 diffusion components of 200 Swiss rolls, allowing for sign flips and reordering of components. CFDM produces a lower approximation error and a smaller approximate kernel for less computational cost than competing methods (Fig. 3.2). For $n \geq 10^4$, CFDM offers an approximately 10x speed-up over exact diffusion maps, and this advantage grows slightly as $n$ increases.
CHAPTER 3. COMPRESSED DIFFUSION

3.6.2 Mass cytometry of induced pluripotent stem cells

Mass cytometry is the measurement of protein abundance in individual cells via mass spectrometry. We used CFDM to approximate the diffusion map of 20,000 single cells in an induced pluripotent stem cell (iPSC) system [Zander et al., 2015] with 500 partitions. Figure 3.3 shows the exact and approximated diffusion maps. CFDM produces a visually equivalent embedding in a fraction of the time, accurately revealing the differentiation of skin cells into both iPSCs and a failed reprogramming state. Such an approximation will be beneficial for constructing diffusion maps on large graphs, such as social networks, as well as for downstream applications, where algorithms of high computational complexity are run on the diffusion map.

Acknowledgments

This work was partially funded by: the Gruber Foundation [S.G.]; IVADO (l’institut de valorisation des données) [G.W.]; Chan-Zuckerberg Initiative grants 182702 & CZF2019-002440 [S.K.]; and NIH grants R01GM135929 & R01GM130847 [G.W., S.K.].
Appendix

3.A Proof of Theorem 3.3.1

We prove the theorem in three parts, via the following lemmas. We start with the kernel degrees:

Lemma 3.A.1. \( \|k(x, \cdot)\|_1 = \|g(x, \cdot)\|_1 \)

Proof. Direct computation yields

\[
\|k(x, \cdot)\|_1 = \int k(x, y) dy = \int \int \frac{g(x, r)}{\|g(x, \cdot)\|_1} g(r, y) dr dy
\]

\[
= \int \frac{g(x, r)}{\|g(x, \cdot)\|_1} \left( \int g(r, y) dy \right) dr
\]

\[
= \int g(x, r) dr
\]

\[
= \|g(x, \cdot)\|_1.
\]

Next, we prove the relation to diffusion transition probabilities:

Lemma 3.A.2. \( P^2 f(x) = \frac{1}{\|k(x, \cdot)\|_1} \int k(x, y) f(y) dy \)
CHAPTER 3. COMPRESSED DIFFUSION

Proof. Direct computation yields

\[ P^2 f(x) = \int p(x, r) \int p(r, y) f(y) dy dr \]

\[ = \frac{1}{\| g(x, \cdot) \|_1} \int \int \frac{g(x, r)}{\| g(r, \cdot) \|_1} g(r, y) f(y) dy dr \]

\[ = \frac{1}{\| g(x, \cdot) \|_1} \int \int \frac{g(x, r)}{\| g(r, \cdot) \|_1} g(r, y) dr f(y) dy \]

\[ = \frac{1}{\| k(x, \cdot) \|_1} \int k(x, y) f(y) dy. \]

Finally, we consider the operator norm:

Lemma 3.A.3. \( \| K \| \leq \| G \| \)

Proof. Let \( Q f(x) = q(x) f(x) \); thus, we can verify \( G = Q P \) and by combining the previous two lemmas we get

\[ \| K \| = \| Q P^2 \| = \| G P \| \leq \| G \| \| P \| = \| G \| \]

, since \( P \) is row-stochastic with \( \| P \| = 1 \).

3.B Proof of Theorem 3.4.1

We prove the theorem in three parts, via the following lemmas. We start with the operator norm:

Lemma 3.B.1. \( \| K_S \| \leq [\text{vol } \mathcal{M}]^{3/2} \| G \| \)

Proof. Let \( v \in \mathbb{R}^n \), and define the simple function \( f_v = \sum_{j=1}^{n} [v]_j \chi_{S_j} \), where \( \chi_{S_j} \) is the characteristic (i.e., indicator) function of \( S_j \). Notice that since \( \mathcal{M} \) is compact with finite volume, then

\[ \| f \|_2 = \left[ \sum_{j=1}^{n} [v]_j^2 \text{vol } S_j \right]^{1/2} \leq [\text{vol } \mathcal{M}]^{1/2} \| v \|_2. \]

Further, for \( i = 1, \ldots, n \),

\[ [K_S v]_i = \sum_{j=1}^{n} \left[ \int_{S_i} \int_{S_j} k(x, y) dy dx \right] [v]_j = \int_{S_i} \int_{S_j} k(x, y) f_v(y) dy dx = \int_{S_i} K f(x) dx. \]

Therefore,
where the last inequality is due to Hölder inequality. Finally, by combining these results with Theorem 3.3.1 we get

\[ \|K_S v\|_2^2 \leq \|\text{vol }\mathcal{M}\| K f \|_2^2 \|G\|_2 \|v\|_2 \]

for every \( v \in \mathbb{R}^n \), which yields the result in the lemma.

Next, we consider the diffusion degrees:

**Lemma 3.B.2.** \( \|k_S(S, \cdot)\|_1 = \int_S \|g(x, \cdot)\|_1 dx \) for all \( S \in \mathcal{S} \)

**Proof.** Direct computation yields

\[ \|k_S(S, \cdot)\|_1 = \sum_j \|k_S(S, S_j)\|_1 = \int_S \left[ \sum_j \int_{S_j} k(x, y) dy \right] dx = \int_S \int k(x, y) dy dx = \int_S \|k(x, \cdot)\|_1 dx, \]

and together with Theorem 3.3.1 we get the result in the lemma.

Finally, we prove the relation to diffusion transition probabilities:

**Lemma 3.B.3.**

\[ P_S(S, T) = \int \Pr[S \overset{\text{\(1\) step}}{\rightsquigarrow} r] \Pr[r \overset{\text{\(1\) step}}{\rightsquigarrow} T] dr \]

where

\[ \Pr[r \overset{\text{\(1\) step}}{\rightsquigarrow} T] = \int_T p(r, y) dy \text{ and } \Pr[S \overset{\text{\(1\) step}}{\rightsquigarrow} r] = \int_S p(x, r) \Pr[x \mid S] dx \]

with prior

\[ \Pr[x \mid S] = \frac{\|g(x, \cdot)\|_1}{\|k_S(S, \cdot)\|_1} \]
Proof. Direct computation yields:

\[ P_S(S, T) = \frac{k_S(S, T)}{\|k_S(S, \cdot)\|_1} = \frac{1}{\|k_S(S, \cdot)\|_1} \int_S \int_T k(x, y) dy\, dx \]

\[ = \frac{1}{\|k_S(S, \cdot)\|_1} \int_S \int_T g(x, r) \frac{g(r, y)}{\|g(r, \cdot)\|_1} dy\, dr \]

\[ = \int_S g(x, r) \frac{g(r, y)}{\|g(r, \cdot)\|_1} dy\, dr \]

\[ = \sum_{j=0}^{X-1} \lambda_j^t \]

3.C Proof of Proposition 3.5.2

Let \( \Psi \) be a matrix whose columns are the eigenvectors \( \psi_j \), \( j = 0, \ldots, |X| - 1 \), of the \( A \) (in finite settings), and \( \Lambda \) be a diagonal matrix that consists of the corresponding eigenvalues \( 1 = \lambda_0 \geq \cdots \lambda_{|X|-1} \). Since \( A \) is symmetric, it yields an orthonormal set of eigenvectors, thus \( \Psi \) is an orthogonal matrix and further, we can write \( A^t = \Psi \Lambda^t \Psi^T \) for any \( t \geq 0 \). Let \( I_\ell \) be an \( N \times N \) diagonal matrix in which

\[
[I_\ell]_{ii} = \begin{cases} 
1 & i \leq \ell \\
0 & i > \ell.
\end{cases}
\]

Finally, since we are considering finite settings, we can enumerate that dataset as \( X = \{x_1, \ldots, x_N\} \), and thus the order-\( \ell \) coherence defined in Prop. 3.5.2 as \( \mu_\ell(x_i) = [\Psi I_\ell \Psi^T]_{ii} \) for \( i = 1, \ldots, N \). Similarly, we can also write

\[
\rho_\ell(x_i) = \|a_\ell(x_i, \cdot)\|^2 = \langle a_\ell(x_i, \cdot), a_\ell(x_i, \cdot) \rangle = [A^2 T]_{ii}, \quad i = 1, \ldots, N.
\]

With this matrix formulation, we now show the stated result \( |\mu_\ell(x) - \rho_\ell(x)| \propto \left| \ell - \sum_{j=0}^{X-1} \lambda_j^2 \right| \) from
the proposition by direction computation, as:

\[
\left| \sum_{i=1}^{N} \mu_i(x_i) - \rho_i(x_i) \right| = \left| \sum_{i=1}^{N} [\Psi_f \Psi^T]_{ii} - [A^{2t}]_{ii} \right| \\
= \left| \text{trace} \left( \Psi_f^N \Psi^T - A^{2t} \right) \right| = \left| \text{trace} \left( \Psi \left( I_f^N - \Lambda^{2t} \right) \Psi^T \right) \right| \\
= \left| \text{trace} \left( I_f^N - \Lambda^{2t} \right) \right| = \ell - \sum_{j=0}^{|X|-1} \lambda_{2t}^j \tag{3.1}
\]

where (3.1) is due to the eigendecomposition of \( A^{2t} \) and (3.2) is due to the orthogonality of \( \Psi \).

\[\square\]

3.D Proof of Proposition 3.5.3

We note that as shown in Coifman and Lafon (2006), the spectral theorem yields

\[
\sum_{j=0}^{\infty} \lambda_j^{2t} \psi_j(x_i) \psi_j(x_j) = a^{2t}(x, y) = \langle a^t(x, \cdot), a^t(y, \cdot) \rangle
\]

for every \( t \geq 0 \). Furthermore, since by definition \( \rho_t(x) = \|a^t(x, \cdot)\|_2^2 \), we have

\[
\sum_{j=0}^{\infty} \lambda_j^{2t} \psi_j(x_i) \psi_j(x_j) \sqrt{\rho_t(x) \rho_t(y)} = \left\{ \frac{a^t(x, \cdot)}{\|a^t(x, \cdot)\|_2}, \frac{a^t(y, \cdot)}{\|a^t(y, \cdot)\|_2} \right\} \in [0, 1], \quad x, y \in X,
\]

as an inner product of normalized (i.e., unit-norm) non-negative functions. Therefore, since \( \arccos \) is monotonically decreasing in the interval \([0, 1]\), we get the first duality. Namely, for each \( x \in \mathcal{M}, t \geq 0 \),

\[
\arg \min_{y \in \mathcal{M}} D_c^t(x, y) = \arg \max_{y \in \mathcal{M}} \frac{\sum_{j=0}^{\infty} \lambda_j^{2t} \psi_j(x) \psi_j(y)}{\sqrt{\rho_t(x) \rho_t(y)}} = \arg \max_{y \in \mathcal{M}} \frac{a^{2t}(x, y)}{\sqrt{\rho_t(x) \rho_t(y)}}.
\]

To show the second duality, we first write

\[
\left\| \frac{a^t(x, \cdot)}{\sqrt{\rho_t(x)}} - \frac{a^t(y, \cdot)}{\sqrt{\rho_t(y)}} \right\|_2^2 = \left\| \frac{a^t(x, \cdot)}{\rho_t(x)} \right\|_2^2 + \left\| \frac{a^t(y, \cdot)}{\rho_t(y)} \right\|_2^2 - 2 \left\langle \frac{a^t(x, \cdot)}{\sqrt{\rho_t(x)}}, \frac{a^t(y, \cdot)}{\sqrt{\rho_t(y)}} \right\rangle = 2 - 2 \frac{a^{2t}(x, y)}{\sqrt{\rho_t(x) \rho_t(y)}}.
\]

Therefore, together with the first duality, we now get

\[
\arg \min_{y \in \mathcal{M}} \left\| \frac{a^t(x, \cdot)}{\sqrt{\rho_t(x)}} - \frac{a^t(y, \cdot)}{\sqrt{\rho_t(y)}} \right\|_2^2 = \arg \min_{y \in \mathcal{M}} 2 - 2 \frac{a^{2t}(x, y)}{\sqrt{\rho_t(x) \rho_t(y)}} = \arg \max_{y \in \mathcal{M}} \frac{a^{2t}(x, y)}{\sqrt{\rho_t(x) \rho_t(y)}} = \arg \min_{y \in \mathcal{M}} D_c^t(x, y),
\]

which completes the proof. \[\square\]
Bibliography


Scott Gigante, Jay S Stanley, Ngan Vu, David van Dijk, Kevin R Moon, Guy Wolf, and Smita Krishnaswamy. Compressed diffusion. In *Sampling Theory and Applications*, pages 1–4. IEEE, July 2019, doi:10.1109/SampTA45681.2019.9030994 © 2019 IEEE. Reprinted with permission. In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Yale University’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.


Kevin R Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B Burkhardt, William S Chen, Kristina Yim, Antonia van den Elzen, Matthew J Hirn, Ronald R Coifman, Natalia B Ivanova, Guy Wolf, and


Chapter 4

Visualizing structure and transitions in high-dimensional biological data


The need for visualization of high-dimensional data is clear, especially in the realm of single-cell genomics. Human brains are capable of visualizing at most four dimensions (three dimensions, plus color) while the datasets at hand can number in the tens of thousands of dimensions. Thus, in order to understand the structure of a dataset, it is necessary to reduce the dimensionality of a dataset for the purpose of visualization. This visualization is then frequently used for displaying clusters (Luecken and Theis 2019), trajectories (Saelens et al. 2019), gene expression (Van Dijk et al. 2018), and other relevant metadata.

The methods most commonly used for this task are PCA (Abdi and Williams 2010) and t-SNE (Maaten and Hinton 2008). These methods are no doubt useful, but lack some crucial qualities essential for a good visualization. On the one hand, PCA is a linear method, and cannot capture the entire structure of the dataset in just two or three dimensions. It therefore presents the most salient global structure in the first few dimensions, giving the viewer a useful, if simplistic, representation of the data. On the other hand, t-SNE is designed specifically for visualization in two dimensions, but lacks any reference to this salient global structure; its algorithm optimizes for preservation of local structure only, with the placement of groups of

*: Equal contribution. †: Equal contribution.
highly similar points relative to one another left essentially to chance.

In order to address these problems, we develop PHATE, a dimensionality reduction algorithm designed specifically for visualization in two or three dimensions based on the diffusion framework. Here, we show that PHATE better preserves manifold distances of noisy data when presented in two dimensions, and can be used to discover novel biological insights in single-cell transcriptomic data. Specifically, we introduce De-noised Embedding Manifold Preservation (DEMaP), a new metric to quantify the fidelity of a visualization to manifold distances, in which we measure the correlation between manifold distances computed on known ground truth data and Euclidean distances computed on dimensionality reduced noisy data. PHATE performs best on simulations of both trajectories and clusters when quantifying performance with DEMaP, and also performs best when quantifying performance on clustered data with the Adjusted Rand Index (Hubert and Arabie 1985) applied to k-means clustering on the embedding.

Contributions

As a co-first author of this work, my contributions were threefold: I performed and analyzed the qualitative comparisons section, in which I compared PHATE to 11 other algorithms on 20 datasets, including both real and simulated data; I designed and wrote DEMaP and used it to perform and analyze the quantitative comparisons section, in which I quantified PHATE’s performance relative to the 11 algorithms shown in the qualitative comparison section; and I evaluated PHATE’s performance on large datasets, including benchmarking runtime over datasets of increasing size, and evaluating performance on a dataset of 1.3M cells; and additionally, I provided assistance with writing and re-evaluation of the original experiments in response to reviewer comments, and implementation of the software in Python.

4.1 Introduction

High dimensional, high-throughput data are accumulating at a staggering rate, especially of biological systems measured using single-cell transcriptomics and other genomic and epigenetic assays. Because humans are visual learners, it is important that these datasets are presented to researchers in intuitive ways to understand both the overall shape and the fine granular structure of the data. This is especially important in biological systems, where structure exists at many different scales and a faithful visualization can lead to hypothesis generation.

There are many dimensionality reduction methods for visualization (Maaten and Hinton 2008; Amir et al. 2013; Linderman et al. 2019; Tenenbaum et al. 2000; McInnes et al. 2018; Roweis and Saul 2000; Cox and Cox 2008; De Silva and Tenenbaum 2004; van Unen et al. 2017; Chen and Buja 2009; Moon and...
of which the most commonly used are PCA (Moon and Stirling, 2000) and t-SNE (Maaten and Hinton, 2008; Amir et al., 2013; Linderman et al., 2019). However, these methods are suboptimal for exploring high-dimensional biological data. First, they tend to be sensitive to noise. Biomedical data is generally very noisy, and methods like PCA and Isomap (Tenenbaum et al., 2000) fail to explicitly remove this noise for visualization, rendering fine grained local structure impossible to recognize. Second, nonlinear visualization methods such as t-SNE often scramble the global structure in data. Third, many dimensionality reduction methods (e.g., PCA and diffusion maps) fail to optimize for two-dimensional visualization as they are not specifically designed for visualization.

Furthermore, common implementations of dimensionality reduction methods often lack computational scalability. The volume of biomedical data being generated is growing at a scale that far outpaces Moore’s Law. State-of-the-art methods such as MDS and t-SNE were originally presented (e.g., in Cox and Cox (2008); Maaten and Hinton (2008)) as proofs-of-concept with somewhat naïve implementations that do not scale well to datasets with hundreds of thousands, let alone millions, of data points due to speed or memory constraints. Although some heuristic improvements may be made (see, for example, De Silva and Tenenbaum (2004); Linderman et al. (2019)), most available packages still follow the original implementation and thus cannot run on big data, which severely limits the usability of these methods in the medium to long term.

Finally, we note that some methods try to alleviate visualization challenges by directly imposing a fixed geometry or intrinsic structure on the data. However, methods that impose a structure on the data generally have no way of alerting the user whether the structural assumption is correct. For example, any data will be transformed to fit a tree with Monocle2 (Qiu et al., 2017) or clusters with t-SNE. While such methods are useful for data that fit their prior assumptions, they can generate misleading results otherwise, and are often ill suited for hypothesis generation or data exploration.

To address the above concerns, we have designed a dimensionality reduction method for visualization named Potential of Heat-diffusion for Affinity-based Transition Embedding (PHATE). PHATE generates a low-dimensional embedding specific for visualization which provides an accurate, denoised representation of both local and global structure of a dataset in the required number of dimensions without imposing any strong assumptions on the structure of the data, and is highly scalable both in memory and runtime. To achieve this, we combine ideas from manifold learning, information geometry, and data-driven diffusion geometry and integrate them with current state-of-the-art methods. The result is that high-dimensional and nonlinear structures, such as clusters, nonlinear progressions, and branches, become apparent in two or three dimensions and can be extracted for further analysis (Figure 4.1.1A).

We develop a new metric called ‘Denoised Embedding Manifold Preservation’ (DEMaP) to quantify the ability of an embedding to preserve denoised manifold distances, we show that PHATE consistently
Figure 4.1.1:
outperforms 11 other methods on synthetically generated data with known ground truth. We also use PHATE to visualize several biological and non-biological real world datasets, showing PHATE’s capacity to visualize datasets with many different underlying structures including trajectories, clusters, disconnected and intersecting manifolds, and more (Figure 4.1.1). To demonstrate the ability of PHATE to reveal new biological insights, we apply PHATE to a newly generated single-cell RNA-sequencing dataset of human embryonic stem cells grown as embryoid bodies over a period of 27 days to observe differentiation into diverse cell lineages. PHATE successfully captures all known branches of development within this system as well as differentiation pathways that have—to the best of our knowledge—not been described before, and enables the isolation of rare populations based on surface markers, which we validate experimentally.

4.2 Results

Visualizing complex, high-dimensional data in a way that is both easy to understand and faithful to the data is a difficult task. Such a visualization method needs to preserve local and global structure in the high-dimensional data, denoise the data so that the underlying structure is clearly visible, and preserve as much information as possible in low (2-3) dimensions. Additionally, a visualization method should be robust in the sense that the revealed structure of the data is insensitive to user configurations of the algorithm and scalable to the large sizes of modern data.

Popular dimensionality reduction methods are deficient in one or more of these attributes. For example, t-SNE (Maaten and Hinton, 2008) focuses on preserving local structure, often at the expense of the global structure (Figure 4.1.1B-C), while PCA focuses on preserving global structure at the expense of the local structure (Figure 4.1.1B-C). Although PCA is often used for denoising as a preprocessing step, both PCA and t-SNE provide noisy visualizations when the data is noisy, which can obscure the structure of the data (Figure 4.1.1B-C). In contrast, diffusion maps (Coifman and Lafon, 2006) effectively denoises data and learns the local and global structure. However, diffusion maps typically encodes this information in higher dimensions (Haghverdi et al., 2016), which is not amenable to visualization, and can introduce distortions in the visualization under certain conditions (see Figures 4.4.1 and 4.4.2A).

PHATE is designed to overcome these weaknesses and provide a visualization that preserves the local and global structure of the data, denoises the data, and presents as much information as possible into low dimensions. There are three major steps in the PHATE algorithm.

1. **Encode local data information via local similarities (Figure 4.2.1A-C).** For some data types, such as Hi-C chromatin conformation maps (Darrow et al., 2016), the local relationships are encoded directly in the measurements. However, for most data types, the local similarities must be learned. We assume
that component-wise, the data are well-modeled as lying on a manifold. Effectively this means that local relationships between data points, even noisy, are meaningful with respect to the overall structure of the data as they can be chained together to learn global relationships along the manifold. We apply a kernel function we developed (called the \(\alpha\)-decay kernel) to Euclidean distances to accurately encode the local structure of the data even when the data is not uniformly sampled along the underlying manifold structure.

2. **Encode global relationships in data using the potential distance (Figure 4.2.1D).** Diffusing through data is a concept that was popularized in the derivation of Diffusion Maps (DM) \cite{CoifmanLafon2006}. Diffusion is performed by first transforming the local similarities into probabilities that measure the probability of transitioning from one data point to another in a single step of a random walk and then powering this operator to \(t\) steps to give \(t\)-step walk probabilities. Thus both the local and global manifold distances are represented in the newly-calculated multi-step transition probabilities, referred to as the diffusion probabilities. For example, two points that have multiple potential, short paths that connect them will have a higher diffusion probability than two points that either have only long paths or relatively few paths connecting them. By considering all possible random walks, the diffusion process also denoises the data by downweighting spurious paths created by noise. However, directly embedding the diffusion probabilities into 2 or 3 dimensions via eigenvalue decomposition results in either a loss of information (Figure 4.4.1) or an unstable embedding (Figures 4.4.2A and 4.4.1D, respectively). In PHATE we interpret the diffusion probability of each point to all other points as the "global context of the datapoint," and derive an information-theoretic potential distance between each pair of cells that compares the entire global context. Potential distance is computed as a divergence between the associated diffusion probability distributions of the two cells to all other cells. Thus the relationship of each cell to both near neighbors and distant points is accounted for in this distance. Notably, many divergences use a sublinear transformation of probability distributions (such as a logscale transformation) which prevents nearest neighbors from dominating the distance.

3. **Embed potential distance information into low dimensions for visualization (Figure 4.2.1E-F).** The information in the potential distances are then squeezed into low dimensions for visualization via metric MDS, which creates an embedding by matching the distances in the low-dimensional space to the input distances. Unlike PCA, this ensures that all variability is squeezed into the two dimensions for a maximally informative embedding.

These steps are outlined in Table 4.3.1. All of these steps are necessary to create a good visualization that preserves local and global structure in the high-dimensional data, denoises the data, and presents as much information as possible in low dimensions. Further details on all of the steps of PHATE are included in Online Methods, Table 4.4.1 and 4.4.1. PHATE is also robust to the choice of parameters (Online Methods
CHAPTER 4. VISUALIZING STRUCTURE IN HIGH-DIMENSIONAL DATA

Figure 4.2.1: Steps of the PHATE algorithm. (A) Data. (B) Euclidean distances. Data points are colored by their Euclidean distance to the highlighted point. (C) Markov-normalized affinity matrix. Distances are transformed to local affinities via a kernel function and then normalized to a probability distribution. Data points are colored by the probability of transitioning from the highlighted point in a single step random walk. (D) Diffusion probabilities. The normalized affinities are diffused to denoise the data and learn long-range relationships between points. Data points are colored by the probability of transitioning from the highlighted point in a \( t \) step random walk. (E) Informational distance. An informational distance (e.g. the potential distance) that measures the dissimilarity between the diffused probabilities is computed. The informational distance is better suited for computing differences between probabilities than the Euclidean distance. See the text for a discussion. (F) The final PHATE embedding. The informational distances are embedded into low dimensions using MDS. Note that distances or affinities can be directly input to the appropriate step in cases of connectivity data. Therefore, the Euclidean distance or our constructed affinities can be replaced with distances or affinities that best describe the data. For example, in Figure 4.2.11D we replace our affinity matrix with the Facebook connectivity matrix.
and Figure 4.2.2) and produces the same results every time it is run, regardless of random seed (Figure 4.2.3).

In addition to the exact computation of PHATE, we developed an efficient and scalable version of PHATE that produces near-identical results. In this version, PHATE uses landmark subsampling, sparse matrices, and randomized matrix decompositions. For more details on the scalability of PHATE see Online Methods, Table 4.4.2 and Figure 4.2.4, which shows the fast runtime of PHATE on datasets of different sizes, including a dataset of 1.3 million cells (2.5 hours) and a network of 1.8 million nodes (12 minutes).

4.2.1 Extracting Information from PHATE

PHATE embeddings contain a large amount of information on the structure of the data, namely, local transitions, progressions, branches or splits in progressions, and end states of progression. Here we present new methods that provide suggested end points, branch points, and branches based on the information from higher dimensional PHATE embeddings. These may not always correspond to real decision points, but provide an annotation to aid the user in interpreting the PHATE visual.

- Branch Point Identification with Local Intrinsic Dimensionality. In biological data, branch points often encapsulate switch-like decisions where cells sharply veer towards one of a small number of fates (see Figure 4.2.5A for an example). Identifying branch points is of critical importance for analyzing such decisions. We make a key observation that most points in PHATE plots of biological data lie on low-dimensional progressions with some noise as demonstrated in Figure 4.2.6Aii. Since branch points lie at the intersections of such progressions, they have higher local intrinsic dimensionality and can thus be identified by estimating the local intrinsic dimension. Figure 4.2.6Aii shows that points of intersection in the artificial tree data indeed have higher local intrinsic dimensionality than points on branches.

- Endpoint Identification with Diffusion extrema. We identify endpoints in the PHATE embedding as those that are least central and most distinct by computing the eigenvector centrality and the distinctness of a cellular state relative to the general data by considering the minima and maxima of diffusion eigenvectors (see Figure 4.2.6Ai). After identifying branch points and endpoints, the remaining points are assigned to branches between two branch points or between a branch point and endpoint using an approach based on the branch point detection method in Haghverdi et al. (2016) that compares the correlation and anticorrelation of neighborhood distances. Figure 4.2.6Aiii gives a visual demonstration of this approach and details are given in Online Methods. Figure 4.2.6B shows the results of our approach to identifying branch points, endpoints, and branches on an artificial tree dataset, a scRNA-seq dataset of bone marrow (Paul et al., 2015), and an iPSC CyTOF dataset.
Figure 4.2.2: (Continued on the following page.)
CHAPTER 4. VISUALIZING STRUCTURE IN HIGH-DIMENSIONAL DATA

Figure 4.2.2: Visual and quantitative demonstrations of the robustness of PHATE to subsampling and the choice of parameters. (A) The PHATE visualization for the iPSC mass cytometry dataset from Zunder et al. (Cell Stem Cell, vol. 16, no. 3, pp. 323-337, 2015) with varying number of subsample sizes \( N \). The main branches present for \( N = 10000 \) are also visible for the other values of \( N \), demonstrating that the PHATE embedding is robust to the size of the subsample. (B) The PHATE visualization of the same iPSC CyTOF dataset with varying scale parameter \( t \) with \( n = 50000 \) cells. The embeddings for all \( t \) preserve the branching structure and the visualizations are very similar to each other, demonstrating that the embedding is robust to the choice of \( t \). (C) Heatmap of the Spearman correlation coefficient between geodesic distances of the ground truth data and the Euclidean distances of the PHATE visualization applied to the simulated paths dataset using Splatter (Zappia et al., Genome Biology, vol. 18, no. 1, p. 174, 2017). The results are presented using different values for \( k \), \( t \), and \( \alpha \). The value of \( t \) selected using the kneepoint method in this case is 8. The number of simulated cells is \( n = 3000 \). (D) Heatmap of the Spearman correlation coefficient between geodesic distances of the ground truth data and the Euclidean distances of the PHATE visualization applied to the simulated groups dataset using Splatter. The results are presented using different values for \( k \), \( t \), and \( \alpha \). For both the groups and paths datasets, the results are very stable for \( \alpha \geq 10 \). The value of \( t \) selected using the kneepoint method in this case is 8. The number of simulated cells is \( n = 3000 \).

(Zunder et al., 2015). Our procedure identifies the branches on the artificial tree perfectly and defines biologically meaningful branches on the other two datasets which we will use for data exploration.

4.2.2 Comparison of PHATE to Other Methods

Here we compare PHATE to multiple dimensionality reduction methods. We provide quantitative comparisons on simulated data where the ground truth is known, and provide a qualitative comparison using both simulated and real biological data.

Quantitative Comparisons. Quantifying the accuracy of a dimensionality reduction for visualization is an open problem in machine learning (Lui et al., 2018; Tsai, 2012; Bertini et al., 2011) as it is generally impossible to greatly reduce the dimensionality of a dataset without loss of information. To quantify the quality of a visualization, we need a metric that judges whether a method preserves the information that is necessary for visual understanding. Prior work has focused on preserving pairwise distances or local neighborhoods (McInnes et al., 2018; Van Der Maaten et al., 2009; Vankadara and von Luxburg, 2018). However, these quantifications are not strictly desirable. For example, classical MDS is analytically the optimal solution to pairwise distance preservation in \( n \) dimensions (Cox and Cox, 2008). However, MDS, as is visible in Figures 4.B.2 and 4.B.1, often does not produce clear or insightful visualizations for complex, nonlinear data. On the other hand, preserving local neighborhoods is the basis of the objective function for t-SNE (Maaten and Hinton, 2008), which fails to incorporate global structure and is hence insufficient for our purposes (Figure 4.B.1).

Prior work has also emphasized the utility of geodesic distances in computing both dimensionality reductions (Tenenbaum et al., 2000) and associated metrics (Tsai, 2012). Similar computations have been used to compare the output of trajectory inference algorithms (Saelens et al., 2019). However, this metric
Figure 4.2.3: Visual and quantitative demonstrations of the reproducibility of PHATE compared to PCA, tSNE, and UMAP. Reproducibility was computed on 4 different datasets (4 columns) that were generated using Splatter. The different runs had different random seeds and \( n = 2000 \) cells. (A) Boxplots show RMSE computed between 10 runs of each method. RMSE was computed between each unique pair of runs (thus 45 in total) after aligning the pair of embeddings with Procrustes. Thus, RMSE here quantifies how much embeddings change between runs, with lower RMSE signifying greater reproducibility. In the boxplots, the box limits indicate the lower and upper quartile values with a line at the median while the whiskers show the range of the data. (B) For each method (rows) and each dataset (columns) two example runs are shown (orange and blue points) to visually demonstrate the reproducibility. In line with the RMSE boxplots, PHATE and PCA show almost perfectly overlapping embeddings while tSNE and UMAP show significant variability between runs.
Figure 4.2.4: Scalability tests of PHATE. (A) Scalable PHATE embedding of iPSC CyTOF data ($n = 220450$ cells) from Zunder et al. (Cell Stem Cell, vol. 16, no. 3, pp. 323-337, 2015) with a subset of the landmarks shown in red (200 out of 2000). (B) Robustness of PHATE to the number of landmarks chosen. PHATE on the EB data ($n = 16825$ cells) computed using increasing numbers of landmarks (X-axis) was compared to exact PHATE, i.e. without landmarks. Comparison was done using Procrustes analysis (optimal linear transformation) and the sum of squared error (SSE, Y-axis) is shown. To ensure a stable embedding that accurately approximates exact PHATE we choose 2000 landmarks as default. The inset shows the histogram of pairwise distances in the visualization computed using fast PHATE (2000 landmarks) on the EB data vs. the pairwise distances from exact PHATE. The correspondence and the Pearson correlation coefficient are very high. (C) PHATE and t-SNE embeddings of a mouse brain cell dataset from 10X genomics with a large number of cells ($n = 1,300,774$ cells). The PHATE embedding was calculated with 2000 landmarks and completed in three hours. A subset (10 of 60) of the clusters provided by 10X are shown in color, the rest in gray. t-SNE shatters the cluster structure, while PHATE retains clusters as contiguous groups of cells. (D) Runtime of PHATE, t-SNE and UMAP on increasingly large subsamples of the EB data. Runtime was averaged across four runs. (E) Runtime of 12 visualization methods shown in Figures 4.B.1 and 4.B.2 across 19 datasets and corresponding line of best fit for each method. Where a method ran out of memory or took longer than one hour, the runtime is not shown and linear fits are cut off accordingly.
Figure 4.2.5: (Continued on the following page.)
Figure 4.2.5: Annotated PHATE visualizations of CyTOF iPSC data \( (n = 50000) \) cells from Zunder et al. \((\textit{Cell Stem Cell}, \text{vol. 16, no. 3, pp. 323-337, 2015}) \) and branch expression analysis. (A) The primary branch point between the two major branches (reprogrammed and refractory) of the data is highlighted. (B) The PHATE visualization colored by Lin28 (a marker associated with the transition to pluripotency \((\text{Polo et al., Cell, vol. 151, no. 7, pp. 1617-1632, 2012}) \) ) and Ccasp3 (associated with cell apoptosis). Lin28 expression is limited to the reprogrammed branch while Ccasp3 is primarily expressed in the refractory branch, indicating that the failure to reprogram may initiate apoptosis in these cells. (C) Analysis of branches on the PHATE embedding for the same iPSC CyTOF data, (D) bone marrow scRNA-seq dataset \( (n = 2730) \) cells from Paul et al. \((\textit{Cell}, \text{vol. 163, no. 7, pp. 1663-1677, 2015}) \), and (E) newly generated embryoid body scRNA-seq data \( (n = 16825) \) cells. (Left) The PHATE visualization with identified branches. (Middle) Expression level for each cell ordered by branch and ordering within the branch. Cell ordering is calculated using Wanderlust \((\text{Bendall et al., Cell, vol. 157, no. 3, pp. 714-725, 2014}) \) starting on the left-most point of each branch. Expression levels are z-scored for each gene. A colorbar is given below the expression matrices that identifies each branch and (in the case of the bone marrow scRNA-seq data) cell type. (Right) DREMI scores \((\text{Krishnaswamy et al., Science, vol. 346, no. 6213, p. 1250689, 2014}) \) between gene expression levels and cell order within each branch. MAGIC \((\text{van Dijk et al., Cell, vol. 174, no. 3, pp. 716-729, 2018}) \) is applied first in (D) and (E) to impute missing values using the same kernel used for PHATE and smaller \( t \). For branch analysis of the bone marrow data in (D), we used 3 PHATE dimensions to obtain clearer branch separation.

is insufficient for our use for two reasons: 1. unlike in trajectory inference, the raw data is noisy, and we wish to quantify the ability of a visualization method to denoise the data; and 2. geodesic distances on low-dimensional visualizations fail to capture the inherent meaning of curvature. Since visualizations do not suffer from the curse of dimensionality, we are able instead to use Euclidean distances, which capture the difference between straight and curved lines which are also meaningful to the human eye.

Hence, to quantitatively compare PHATE to other visualization methods, we formulated the Denoised Embedding Manifold Preservation (DEMaP) metric. DEMaP is designed to encapsulate the desirable properties of a dimensionality reduction method that is intended for visualization. These include: 1. the preservation of relationships in the data such that cells close together on the manifold are close together in the embedded space and cells that are far apart on the manifold are far apart in the embedding, including disconnected manifolds (e.g. clusters) which should be as well separated as possible; and 2. denoising, such that the low-dimensional embedding accurately represents the ground truth data and is as invariant as possible to biological and technical noise. DEMaP encapsulates each of these properties by comparing the geodesic distances on the noiseless data to the Euclidean distances of the embedding extracted from noisy data. An overview of DEMaP is presented in Figure 4.2.7A. See Online Methods for details.

To compare the performance of PHATE to 12 dimensionality reduction methods, we simulated scRNA-seq data from Splatter \((\text{Zappia et al., 2017}) \). Splatter uses a parametric model to generate data with various structures, such as branches or clusters. This simulated data provides a ground truth reference to which we can add various types of noise. We then use this noisy data as input for each dimensionality reduction algorithm, and quantify the degree to which each representation preserves local and global structures and
Figure 4.2.6:
Table 4.2.1: The DEMaP score calculated on simulated Splatter data with various levels of dropout. The best performing method for each test is shown in bold. Results show the mean correlation coefficient ± standard deviation over 20 trials. Methods are ordered by mean correlation across all tests. Simulation is performed with 3000 cells.

<table>
<thead>
<tr>
<th>Splatter paths</th>
<th>dropout = 0.0</th>
<th>dropout = 0.5</th>
<th>dropout = 0.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>0.73 ± 0.11</td>
<td>0.71 ± 0.11</td>
<td>0.39 ± 0.13</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.69 ± 0.12</td>
<td>0.67 ± 0.13</td>
<td>0.29 ± 0.13</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.68 ± 0.12</td>
<td>0.66 ± 0.12</td>
<td>0.30 ± 0.09</td>
</tr>
<tr>
<td>DM</td>
<td>0.63 ± 0.11</td>
<td>0.62 ± 0.12</td>
<td>0.23 ± 0.16</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>0.62 ± 0.12</td>
<td>0.61 ± 0.12</td>
<td>0.23 ± 0.16</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.67 ± 0.11</td>
<td>0.65 ± 0.13</td>
<td>0.26 ± 0.14</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.63 ± 0.13</td>
<td>0.61 ± 0.14</td>
<td>0.21 ± 0.14</td>
</tr>
<tr>
<td>PCA</td>
<td>0.59 ± 0.12</td>
<td>0.58 ± 0.12</td>
<td>0.33 ± 0.11</td>
</tr>
<tr>
<td>LLE</td>
<td>0.60 ± 0.15</td>
<td>0.57 ± 0.15</td>
<td>0.13 ± 0.16</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.46 ± 0.10</td>
<td>0.47 ± 0.09</td>
<td>0.26 ± 0.13</td>
</tr>
<tr>
<td>MDS</td>
<td>0.46 ± 0.09</td>
<td>0.42 ± 0.10</td>
<td>0.17 ± 0.04</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.03 ± 0.01</td>
<td>0.03 ± 0.01</td>
<td>0.03 ± 0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Splatter groups</th>
<th>dropout = 0.0</th>
<th>dropout = 0.5</th>
<th>dropout = 0.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>0.68 ± 0.12</td>
<td>0.67 ± 0.11</td>
<td>0.42 ± 0.15</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.68 ± 0.12</td>
<td>0.66 ± 0.11</td>
<td>0.30 ± 0.11</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.64 ± 0.12</td>
<td>0.65 ± 0.11</td>
<td>0.33 ± 0.11</td>
</tr>
<tr>
<td>DM</td>
<td>0.65 ± 0.14</td>
<td>0.63 ± 0.14</td>
<td>0.23 ± 0.20</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>0.65 ± 0.14</td>
<td>0.62 ± 0.14</td>
<td>0.23 ± 0.19</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.63 ± 0.12</td>
<td>0.62 ± 0.12</td>
<td>0.28 ± 0.11</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.61 ± 0.14</td>
<td>0.57 ± 0.15</td>
<td>0.26 ± 0.14</td>
</tr>
<tr>
<td>PCA</td>
<td>0.51 ± 0.18</td>
<td>0.50 ± 0.17</td>
<td>0.32 ± 0.16</td>
</tr>
<tr>
<td>LLE</td>
<td>0.56 ± 0.20</td>
<td>0.55 ± 0.18</td>
<td>0.20 ± 0.17</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.40 ± 0.08</td>
<td>0.42 ± 0.09</td>
<td>0.24 ± 0.14</td>
</tr>
<tr>
<td>MDS</td>
<td>0.42 ± 0.09</td>
<td>0.40 ± 0.09</td>
<td>0.16 ± 0.06</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.03 ± 0.02</td>
<td>0.03 ± 0.02</td>
<td>0.03 ± 0.01</td>
</tr>
</tbody>
</table>

denoises the data using DEMaP. To generate a diverse set of ground truth references, we simulated 50 datasets containing clusters and 50 datasets containing branches. See Online Methods for simulation details.

For each method, we used the default parameters and calculated DEMaP on each simulated dataset using different noise settings. The results are presented in Figure 4.2.7B and Tables 4.2.1 to 4.2.4. We found that PHATE had the highest DEMaP score in 22/24 comparisons and was the top-performing method overall. UMAP was the second best performing method overall but had the highest DEMaP score in only two of the comparisons, one of which is equal with PHATE. We ran further tests on cluster data using the adjusted Rand Index (Rand 1971) and found that on average PHATE preserves local cluster structure as well or better than t-SNE, UMAP, and PCA. The results are presented in Figure 4.2.8. From all of these results, we conclude that PHATE captures the true structure of high dimensional data more accurately than existing visualization methods.

**Qualitative Comparisons.** In addition to the quantitative comparison, we can visually compare the
Figure 4.2.7: PHATE most accurately represents manifold distances in a 2D embedding. (A) Schematic description of performance comparison procedure. For each method and each type of corruption, Euclidean distances in the 2D embedding are compared to geodesic distances in an equivalent noiseless simulation by Spearman correlation. (B) Performance of 12 different methods across varying levels of corruption by dropout, decreased signal-to-noise ratio (BCV), randomly subsampled cells (subsample) and randomly subsampled genes (n_genes). Mean correlation of 20 runs for each configuration is shown. For further details see Tables 4.2.1 to 4.2.4.
Figure 4.2.8: PHATE preserves separations and cluster structure in addition to continuum structure. To quantify the ability of PHATE to preserve cluster structure, we generated 30 random datasets with cluster structure using the Splatter package (Zappia et al., Genome Biology, vol. 18, no. 1, p. 174, 2017). Each dataset has \( n = 2000 \) cells and between 7 and 14 clusters. We then computed the Adjusted Rand Index (Rand, Journal of the American Statistical Association, vol. 66, no. 336, pp. 846-850, 1971) (ARI, y-axis) between the ground truth clusters and clusters obtained by running k-means clustering on the embeddings. An ARI of 1 means perfect recovery of the clusters. We performed this analysis on Splatter data with increasing amounts of noise added during generation. For each noise level we compare clustering on the raw data, on 2-dimensional PCA, 2D t-SNE, 2D UMAP, and 2D PHATE. On average, PHATE preserves local cluster structure as well or better than the other methods. In the boxplots, the box limits indicate the lower and upper quartile values with a line at the median while the whiskers show the range of the data.

Table 4.2.2: The DEMaP score calculated on simulated Splatter data with various amounts of noise. The best performing method for each test is shown in bold. Results show the mean correlation coefficient \( \pm \) standard deviation over 20 trials. Methods are ordered by mean correlation across all tests. Simulation is performed with 3000 cells.

<table>
<thead>
<tr>
<th>Splatter paths</th>
<th>BCV = 0.0</th>
<th>BCV = 0.25</th>
<th>BCV = 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>\textbf{0.76 \pm 0.09}</td>
<td>\textbf{0.66 \pm 0.13}</td>
<td>\textbf{0.32 \pm 0.19}</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.69 \pm 0.09</td>
<td>0.62 \pm 0.14</td>
<td>0.27 \pm 0.16</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.71 \pm 0.08</td>
<td>0.58 \pm 0.14</td>
<td>0.24 \pm 0.14</td>
</tr>
<tr>
<td>DM</td>
<td>0.67 \pm 0.10</td>
<td>0.58 \pm 0.14</td>
<td>0.29 \pm 0.18</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>0.67 \pm 0.10</td>
<td>0.57 \pm 0.14</td>
<td>0.29 \pm 0.18</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.66 \pm 0.12</td>
<td>0.60 \pm 0.13</td>
<td>0.28 \pm 0.16</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.73 \pm 0.09</td>
<td>0.54 \pm 0.15</td>
<td>0.21 \pm 0.15</td>
</tr>
<tr>
<td>PCA</td>
<td>0.67 \pm 0.09</td>
<td>0.51 \pm 0.14</td>
<td>0.22 \pm 0.15</td>
</tr>
<tr>
<td>LLE</td>
<td>0.59 \pm 0.12</td>
<td>0.51 \pm 0.17</td>
<td>0.16 \pm 0.17</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.44 \pm 0.11</td>
<td>0.47 \pm 0.09</td>
<td>0.28 \pm 0.17</td>
</tr>
<tr>
<td>MDS</td>
<td>0.55 \pm 0.07</td>
<td>0.36 \pm 0.10</td>
<td>0.17 \pm 0.06</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.04 \pm 0.01</td>
<td>0.03 \pm 0.01</td>
<td>0.02 \pm 0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Splatter groups</th>
<th>BCV = 0.0</th>
<th>BCV = 0.25</th>
<th>BCV = 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>0.65 \pm 0.13</td>
<td>\textbf{0.67 \pm 0.12}</td>
<td>\textbf{0.41 \pm 0.15}</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.59 \pm 0.14</td>
<td>0.59 \pm 0.11</td>
<td>0.31 \pm 0.10</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.57 \pm 0.14</td>
<td>0.58 \pm 0.11</td>
<td>0.30 \pm 0.09</td>
</tr>
<tr>
<td>DM</td>
<td>\textbf{0.69 \pm 0.13}</td>
<td>0.57 \pm 0.16</td>
<td>0.33 \pm 0.17</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>\textbf{0.69 \pm 0.13}</td>
<td>0.56 \pm 0.16</td>
<td>0.33 \pm 0.17</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.58 \pm 0.12</td>
<td>0.57 \pm 0.12</td>
<td>0.30 \pm 0.11</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.54 \pm 0.20</td>
<td>0.53 \pm 0.16</td>
<td>0.25 \pm 0.14</td>
</tr>
<tr>
<td>PCA</td>
<td>0.55 \pm 0.17</td>
<td>0.47 \pm 0.18</td>
<td>0.24 \pm 0.13</td>
</tr>
<tr>
<td>LLE</td>
<td>0.40 \pm 0.24</td>
<td>0.54 \pm 0.18</td>
<td>0.23 \pm 0.17</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.40 \pm 0.10</td>
<td>0.39 \pm 0.11</td>
<td>0.30 \pm 0.13</td>
</tr>
<tr>
<td>MDS</td>
<td>0.47 \pm 0.09</td>
<td>0.36 \pm 0.10</td>
<td>0.17 \pm 0.07</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.05 \pm 0.03</td>
<td>0.03 \pm 0.01</td>
<td>0.02 \pm 0.01</td>
</tr>
</tbody>
</table>
Table 4.2.3: The DEMaP score calculated on simulated Splatter data with various proportions of cells retained. The best performing method for each test is shown in bold. Results show the mean correlation coefficient ± standard deviation over 20 trials. Methods are ordered by mean correlation across all tests. Simulation is performed with 3000 cells.

<table>
<thead>
<tr>
<th>Splatter paths</th>
<th>n_cells = 2850</th>
<th>n_cells = 1500</th>
<th>n_cells = 150</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>0.71 ± 0.11</td>
<td>0.71 ± 0.12</td>
<td>0.60 ± 0.12</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.67 ± 0.13</td>
<td>0.66 ± 0.13</td>
<td>0.55 ± 0.13</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.65 ± 0.14</td>
<td>0.63 ± 0.14</td>
<td>0.53 ± 0.14</td>
</tr>
<tr>
<td>DM</td>
<td>0.62 ± 0.12</td>
<td>0.61 ± 0.12</td>
<td>0.56 ± 0.12</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>0.61 ± 0.12</td>
<td>0.60 ± 0.12</td>
<td>0.56 ± 0.12</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.64 ± 0.13</td>
<td>0.64 ± 0.13</td>
<td>0.36 ± 0.10</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.61 ± 0.14</td>
<td>0.59 ± 0.16</td>
<td>0.48 ± 0.13</td>
</tr>
<tr>
<td>PCA</td>
<td>0.58 ± 0.12</td>
<td>0.58 ± 0.12</td>
<td>0.57 ± 0.11</td>
</tr>
<tr>
<td>LLE</td>
<td>0.57 ± 0.15</td>
<td>0.54 ± 0.17</td>
<td>0.43 ± 0.18</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.46 ± 0.08</td>
<td>0.54 ± 0.10</td>
<td>0.56 ± 0.12</td>
</tr>
<tr>
<td>MDS</td>
<td>0.43 ± 0.10</td>
<td>0.43 ± 0.09</td>
<td>0.40 ± 0.10</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.04 ± 0.01</td>
<td>0.04 ± 0.02</td>
<td>0.07 ± 0.03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Splatter groups</th>
<th>n_cells = 2850</th>
<th>n_cells = 1500</th>
<th>n_cells = 150</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>0.67 ± 0.12</td>
<td>0.64 ± 0.13</td>
<td>0.61 ± 0.16</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.66 ± 0.12</td>
<td>0.65 ± 0.12</td>
<td>0.54 ± 0.15</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.64 ± 0.10</td>
<td>0.62 ± 0.12</td>
<td>0.54 ± 0.13</td>
</tr>
<tr>
<td>DM</td>
<td>0.63 ± 0.14</td>
<td>0.62 ± 0.15</td>
<td>0.55 ± 0.17</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>0.62 ± 0.14</td>
<td>0.61 ± 0.15</td>
<td>0.55 ± 0.18</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.62 ± 0.13</td>
<td>0.61 ± 0.11</td>
<td>0.38 ± 0.10</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.58 ± 0.14</td>
<td>0.57 ± 0.15</td>
<td>0.49 ± 0.18</td>
</tr>
<tr>
<td>PCA</td>
<td>0.50 ± 0.17</td>
<td>0.50 ± 0.18</td>
<td>0.52 ± 0.17</td>
</tr>
<tr>
<td>LLE</td>
<td>0.54 ± 0.17</td>
<td>0.54 ± 0.18</td>
<td>0.46 ± 0.20</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.42 ± 0.12</td>
<td>0.46 ± 0.11</td>
<td>0.48 ± 0.17</td>
</tr>
<tr>
<td>MDS</td>
<td>0.40 ± 0.10</td>
<td>0.41 ± 0.09</td>
<td>0.39 ± 0.09</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.03 ± 0.02</td>
<td>0.04 ± 0.02</td>
<td>0.06 ± 0.04</td>
</tr>
</tbody>
</table>
Table 4.2.4: The DEMaP score calculated on simulated Splatter data with various numbers of genes retained. The best performing method for each test is shown in bold. Results show the mean correlation coefficient ± standard deviation over 20 trials. Methods are ordered by mean correlation across all tests. Simulation is performed with 3000 cells.

<table>
<thead>
<tr>
<th>Splatter paths</th>
<th>n_genes = 17000</th>
<th>n_genes = 10000</th>
<th>n_genes = 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>0.70 ± 0.12</td>
<td>0.59 ± 0.16</td>
<td>0.18 ± 0.11</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.67 ± 0.13</td>
<td>0.56 ± 0.16</td>
<td>0.16 ± 0.10</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.65 ± 0.13</td>
<td>0.53 ± 0.15</td>
<td>0.16 ± 0.09</td>
</tr>
<tr>
<td>DM</td>
<td>0.61 ± 0.13</td>
<td>0.51 ± 0.16</td>
<td>0.18 ± 0.11</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>0.60 ± 0.13</td>
<td>0.51 ± 0.16</td>
<td>0.18 ± 0.11</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.65 ± 0.14</td>
<td>0.53 ± 0.15</td>
<td>0.17 ± 0.10</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.60 ± 0.14</td>
<td>0.47 ± 0.17</td>
<td>0.15 ± 0.10</td>
</tr>
<tr>
<td>PCA</td>
<td>0.57 ± 0.12</td>
<td>0.47 ± 0.15</td>
<td>0.17 ± 0.09</td>
</tr>
<tr>
<td>LLE</td>
<td>0.57 ± 0.16</td>
<td>0.43 ± 0.19</td>
<td>0.09 ± 0.12</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.45 ± 0.09</td>
<td>0.43 ± 0.11</td>
<td>0.17 ± 0.10</td>
</tr>
<tr>
<td>MDS</td>
<td>0.42 ± 0.09</td>
<td>0.34 ± 0.11</td>
<td>0.18 ± 0.06</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.03 ± 0.01</td>
<td>0.03 ± 0.01</td>
<td>0.01 ± 0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Splatter groups</th>
<th>n_genes = 17000</th>
<th>n_genes = 10000</th>
<th>n_genes = 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHATE</td>
<td>0.68 ± 0.12</td>
<td>0.61 ± 0.14</td>
<td>0.21 ± 0.15</td>
</tr>
<tr>
<td>UMAP</td>
<td>0.66 ± 0.11</td>
<td>0.53 ± 0.13</td>
<td>0.16 ± 0.12</td>
</tr>
<tr>
<td>TSNE</td>
<td>0.64 ± 0.11</td>
<td>0.53 ± 0.14</td>
<td>0.17 ± 0.10</td>
</tr>
<tr>
<td>DM</td>
<td>0.63 ± 0.15</td>
<td>0.51 ± 0.18</td>
<td>0.19 ± 0.15</td>
</tr>
<tr>
<td>MDS on DM</td>
<td>0.62 ± 0.15</td>
<td>0.51 ± 0.19</td>
<td>0.19 ± 0.15</td>
</tr>
<tr>
<td>Force Directed Layout</td>
<td>0.62 ± 0.11</td>
<td>0.51 ± 0.13</td>
<td>0.18 ± 0.12</td>
</tr>
<tr>
<td>Isomap</td>
<td>0.58 ± 0.15</td>
<td>0.48 ± 0.16</td>
<td>0.16 ± 0.13</td>
</tr>
<tr>
<td>PCA</td>
<td>0.50 ± 0.17</td>
<td>0.44 ± 0.17</td>
<td>0.17 ± 0.11</td>
</tr>
<tr>
<td>LLE</td>
<td>0.56 ± 0.15</td>
<td>0.47 ± 0.19</td>
<td>0.11 ± 0.15</td>
</tr>
<tr>
<td>TSNE on DM</td>
<td>0.40 ± 0.12</td>
<td>0.39 ± 0.12</td>
<td>0.15 ± 0.09</td>
</tr>
<tr>
<td>MDS</td>
<td>0.40 ± 0.09</td>
<td>0.33 ± 0.10</td>
<td>0.17 ± 0.07</td>
</tr>
<tr>
<td>Monocle2</td>
<td>0.03 ± 0.01</td>
<td>0.03 ± 0.01</td>
<td>0.01 ± 0.01</td>
</tr>
</tbody>
</table>
embeddings provided by different methods. Figure 4.2.9 shows a comparison of the PHATE visualization to seven other methods on five single-cell datasets with known trajectory (Fig. 4.2.9 A,D,E) and cluster (Fig. 4.2.9 B-C) structures. We see that PHATE provides a clean and relatively denoised visualization of the data that highlights both the local and global structure: local clusters or branches are visually connected to each other in a global structure in each of the PHATE visualizations. Many of these branches are consistent with cell types or clusters validated by the authors (Zunder et al., 2015; Shekhar et al., 2016; Zeisel et al., 2015; Paul et al., 2015) and are also present in other visualizations such as force-directed layout and t-SNE, suggesting that the structures in the PHATE embedding reflect true structure in the dataset. However, force-directed layout tends to give a noisier visualization with fewer clear branches. Additionally, t-SNE (Van Der Maaten et al., 2009) tends to shatter trajectories into clusters, creating the false impression that the data contain natural clusters. We characterize each of these visualizations in detail in 4.B.

We obtained similar results by comparing PHATE to eleven methods on nine non-biological datasets, including four artificial datasets where the ground truth is known (Figure 4.B.1). Expanded comparisons on single-cell data, including additional datasets and visualization methods, are also included in Figure 4.B.2. See 4.B for a full discussion of each method in all of these comparisons.

### 4.2.3 Data Exploration with PHATE

PHATE can reveal the underlying structure of the data for a variety of datatypes. 4.C discusses PHATE applied to multiple different datasets, including SNP data, microbiome data, Facebook network data, Hi-C chromatin conformation data, and facial images (Figures 4.2.10 and 4.2.11). In this section, however, we show the insights gained through the PHATE visualization of this structure for single-cell data. See Online Methods for details on preprocessing steps.

We show that the identifiable trajectories in the PHATE visualization have biological meaning that can be discerned from the gene expression patterns and the mutual information between gene expression and the ordering of cells along the trajectories. We analyze the mouse bone marrow scRNA-seq (Paul et al., 2015) and iPSC CyTOF (Zunder et al., 2015) datasets described previously. Our analysis of the iPSC CyTOF data is presented here while the analysis of the mouse bone marrow data is presented in 4.C. For both of these datasets, we used our new methods for detecting branches and branch points. We then ordered the cells within each trajectory using Wanderlust (Bendall et al., 2014) applied to higher-dimensional PHATE coordinates. We note that ordering could also be based on other pseudotime ordering software such as those in Setty et al. (2016); Haghverdi et al. (2016); Liiv (2010); Hahsler et al. (2008); Wolf et al. (2019). To estimate the strength of the relationship between gene expression and cell ordering along branches, we estimated the DREMI score.
Figure 4.2.9: Comparison of PHATE to other visualization methods on biological datasets. Columns represent different visualization methods, rows different datasets.
Figure 4.2.10: (Continued on the following page.)
Figure 4.2.10: PHATE reveals structure in a variety of high-dimensional datasets. (A) A 3D PHATE visualization of the Frey Faces dataset \((n = 1965\) images) used in Roweis and Saul (Science, vol. 290, no. 5500, pp. 2323-2326, 2000). Points are colored by time within the video. Multiple branches corresponding to different poses are clearly visible. (B) PCA and PHATE embeddings of microbiome data from the American Gut project \((n = 9660\) human samples), colored by body site, and branches annotated by their dominant genera or phyla. (C) The PHATE embedding of the same data from the American Gut project colored by 2 genera (bacteroides and prevotella) and a phylum (actinobacteria) of bacteria. (D) The PHATE embedding of only the fecal samples from the American Gut project \((n = 8596\) colored by various genera (bacteroides and prevotella) and phyla (f Firmicutes, Verrucomicrobia, and Proteobacteria) of bacteria. Each PHATE branch is associated with one of these bacteria groups. (E) PCA and PHATE embeddings of SNP data from the Human Origins dataset \((n = 2345\) present-day humans) showing genotyped present-day humans from 203 populations (Patterson et al., Genetics, vol. 192, no. 3, pp. 1065-1093, 2012) with the population legend in (F).

(a weighted mutual information that eliminates biases to reveal shape-agnostic relationships between two variables (Krishnaswamy et al., 2014)) between gene expression and the Wanderlust-based ordering within each branch. Genes with a high DREMI score within a branch are changing along the branch. We also use PHATE to analyze the transcriptional heterogeneity in rod bipolar cells to demonstrate PHATE’s ability to preserve cluster structure (see 4.C and Figure 4.C.1A).

Figure 4.2.5C shows the mass cytometry dataset from Zunder et al. (2015) that shows cellular reprogramming with Oct4 GFP from mouse embryonic fibroblasts (MEFs) to induced pluripotent stem cells (iPSCs) at the single-cell resolution. The protein markers measure pluripotency, differentiation, cell-cycle and signaling status. The cellular embedding (with combined timepoints) by PHATE shows a unified embedding that contains five main branches, further segmented in our visualization, each corresponding to biology identified in Zunder et al. (2015). Branch 2 contains early reprogramming intermediates with the correct set of reprogramming factors Sox2+ /Oct4+ /Klf4+ /Nanog+ and with relatively low CD73 at the beginning of the branch. Branch 2 splits into two additional branches. Branches 4 and 6 (Figure 4.2.5) show the successful reprogramming to ESC-like lineages expressing markers such as Nanog, Oct4, Lin28 and Ssea1, and Epcam that are associated with transition to pluripotency (Polo et al., 2012). Branch 5 shows a lineage that is refractory to reprogramming, does not express pluripotency markers, and is referred to as “mesoderm-like” in Zunder et al. (2015).

The other branches are similarly analyzed in 4.C. In addition, the data features can be reweighted to obtain specific “views” of the data (see 4.C and Figure 4.2.12).

4.2.4 PHATE Analysis of Human ESC Differentiation Data

To test the ability of PHATE to provide novel insights in a complex biological system, we generated and analyzed scRNA-seq data from human embryonic stem cells (hESCs) differentiating as embryoid bodies (EB) (Martin and Evans 1975), a system which has never before been extensively analyzed at the single-
CHAPTER 4. VISUALIZING STRUCTURE IN HIGH-DIMENSIONAL DATA

Figure 4.2.11: (Continued on the following page.)
Figure 4.2.11: PHATE reveals structure in a variety of connectivity datasets. (A) 3D PHATE visualization of human Hi-C data (Darrow et al., Proceedings of the National Academy of Sciences, p. 201609643, 2016) using all 23 chromosomes at 50 kb resolution ($n = 56702$ locations on the chromosomes), colored by chromosome. Each point corresponds to a genomic fragment. (B) PHATE visualizations of the same human Hi-C data in A for chromosome 1 at 10 kb resolution colored by chromosome location ($n = 22128$ chromosome locations). (C) 2D PHATE visualization of the same human Hi-C data for chromosome 1 at 10 kb resolution, colored by selected chromatin modification markers from ChIP-seq data ($n = 22128$ chromosome locations). (D) Force-directed layout and PHATE visualizations of Facebook network data with data points colored by their degree (number of connections). The subnetworks are taken from the friend networks of selected individuals within the entire network. In all cases, PHATE reveals more structure. For the entire network, $n = 3927$ nodes. For subnetworks 1 and 2, $n = 1034$ and 532 nodes, respectively.

cell level. EB differentiation is thought to recapitulate key aspects of early embryogenesis and has been successfully used as the first step in differentiation protocols for certain types of neurons, astrocytes and oligodendrocytes (Bibel et al. 2007; Kang et al. 2007; Zhao et al. 2006; Liour et al. 2006), hematopoietic, endothelial and muscle cells (Nakano et al. 1996; Nishikawa et al. 1998; Wiles and Keller 1991; Potocnik et al. 1994; Tsai et al. 2000; Fairchild et al. 2000; Yamashita et al. 2000; Maltsev et al. 1993; Rohwedel et al. 1994), hepatocytes and pancreatic cells (Kania et al. 2004; Schroeder et al. 2006), as well as germ cells (Geijsen et al. 2004; Kehler et al. 2005). However, the developmental trajectories through which these early lineage precursors emerge from hESCs as well as their cellular and molecular identities remain largely unknown, particularly in human models.

We measured approximately 31,000 cells, equally distributed over a 27-day differentiation time course (Figure 4.2.14A and Online Methods). Samples were collected at 3-day intervals and pooled for measurement on the 10x Chromium platform. The PHATE embedding of the EB data revealed a highly ordered and clean cellular structure dominated by continuous progressions (Figures 4.1.1C and 4.2.13A), unlike other methods such as PCA or t-SNE (Figure 4.2.2). Exploratory analysis of this system using PHATE uncovered a comprehensive map of four major germ layers with both known and novel differentiation intermediates that were not captured with other visualization methods.

4.2.5 A Comprehensive Lineage Map of Embryoid Bodies from PHATE

Importantly, PHATE retained global structure and organization of the data as is evidenced by the retention of a strong time trend in the embedding, although sample time was not included in creating the embedding. Further, PHATE revealed greater phenotypic diversity at later time points as seen by the larger space encompassed by the embedding at days 18 to 27 (Figure 4.1.1C).

This phenotypic heterogeneity was further analyzed by both an automated analysis (see 4.D Figure 4.2.13A, and Tables 4.2.5 and 4.2.6) and by manual examination of the embedding in conjunction with the established literature on germ layer development (Figure 4.2.14B). For the manual analyses, we used 80
Figure 4.2.12: PHATE using reweighted distances to highlight specific biological processes or “views” of the data. (A) PHATE embedding of the CyTOF iPSC data (n = 220450) from Zunder et al. (Cell Stem Cell, vol. 16, no. 3, pp. 323-337, 2015) using (i) unweighted distances, (ii) distances after upweighting cell cycle markers, (iii) distances after upweighting stem cell markers, (iv) distances after upweighting mitotic markers. (B) PHATE embedding of the same dataset colored by different markers (columns). From top to bottom: (i) PHATE cell cycle “view”, (ii) PHATE stem cell “view” (iii) PHATE mitotic “view”.
Intrinsic dimension of PHATE embedding identifies branches

- **i. 30 clusters**
- **ii. Intrinsic dimension**
- **iii. 10 sub-branches**

Lineage map of EB differentiation revealed by PHATE

- Lateral Plate ME
- Epicardial Precursors
- Hemangioblast
- Smooth Muscle Precursors
- Cardiac Precursors
- Posterior EN
- Neuronal subtypes

EMD scores across PHATE clusters

- Neural crest cells (sub-branch iii)
- Cardiac progenitors (sub-branch vii)

Expression of isolated NCCs and CPs matches PHATE branch predictions

Sorted Neural Crest Cells
Sorted Cardiac Progenitors

Figure 4.2.13: (Continued on the following page.)
Figure 4.2.13: PHATE analysis of embryoid body scRNA-seq data with \( n = 16,285 \) cells. (A) i) The PHATE visualization colored by clusters. Clustering is done on a ten dimensional PHATE embedding. The number of cells in each cluster is given in Table 4.2.6. ii) The PHATE visualization colored by estimated local intrinsic dimensionality with selected branch points highlighted. iii) Branches and sub-branches chosen from contiguous clusters for analysis. (B) Lineage tree of the EB system determined from the PHATE analysis showing embryonic stem cells (ESC), the primitive streak (PS), mesoderm (ME), endoderm (EN), neuroectoderm (NE), neural crest (NC), neural progenitors (NP), and others. Red font indicates novel cell precursors. See supplemental videos 1, 2, and 3 for 3D PHATE visualizations of each stage in the tree. (C) PHATE embedding overlaid with each of the populations in the lineage tree. Other abbreviations include lateral plate ME (LP ME), hemangioblast (H), cardiac (C), epicardial precursors (EP), smooth muscle precursors (SMP), cardiac precursors (CP), and neuronal subtypes (NS). (D) Heatmap showing the EMD score between the cluster distribution and the background distribution for each gene. Relevant genes for identifying the main lineages were manually identified. Genes are organized according to their maximum EMD score. The number of cells in each cluster is given in Table 4.2.6. (E) The EMD scores of the top scoring surface markers in the targeted sub-branches (sub-branches iii and vii). (F) Scatter plots of the bulk transcription factor expression vs. the mean single-cell transcription factor expression in sub-branches iii (left, \( n = 2,537 \) cells) and vii (right, \( n = 1,314 \) cells). The Spearman correlation coefficients are calculated for \( n = 1,213 \) transcription factors.

markers from the literature to identify populations along the PHATE map which gave rise to a detailed germ layer specification map (Figure 4.2.13B, Videos 1, 2, and 3). These populations are shown on the PHATE visualization in Figure 4.2.13C. In the lineage tree, the dots are the populations and the arrows represent transitions between the populations. Our map shows in detail how hESCs give rise to germ layer derivatives via a continuum of defined intermediate states.

4.2.6 Novel Transitional Populations in Embryoid Bodies

The comprehensive nature of the lineage map generated from the PHATE embedding allowed us to identify novel transitional populations that have not yet been characterized. Three novel pre-cursor states were identified in both manual and automated analyses: a bi-potent NC and NP pre-cursor, a novel EN precursor, and a novel cardiac precursor.

Within the ectodermal lineage, differentiation begins with the induction of pre-NE state characterized by downregulation of \( POU5F1 \) and induction of \( OTX2 \). This state is resolved into two precursors, NE-1 (\( GBX2^+ZIC2/5^+ \)) and NE-2 (\( GBX2^+OLIG2^+HOXD1^+ \)). While NE-1 neuroectoderm appeared to develop along the canonical NE specification route and expressed a set of well established anterior NE markers (\( ZIC2/5, PAX6, GLI3, SIX3/6 \)), the NE-2 neuroectoderm gave rise to a bi-potent \( HOXA2^+HOXB1^+ \) precursor that subsequently separated into the NC branch and neural progenitor (NP) branch. Given its potential to generate both NE and NC cell types, the \( HOXA2^+HOXB1^+ \) precursor could represent the equivalent of the neural plate border cells that have been defined in model organisms (Betancur et al., 2010; Barembaum and Bronner-Fraser, 2005).

Within the EN branch, the canonical \( EOMES^+FOXA2^+SOX17^+ \) EN precursor was clustered together
Figure 4.2.14: (Continued on the following page.)
Figure 4.2.14: Further analysis of the EB scRNA-seq data. (A) Inverted images of hESCs and EBs at each timepoint of data collection. Structures of different densities are clearly visible late in the time course (D15-D27) indicating the formation of distinct cell types. The experiments were repeated independently \( n = 3 \) times. (B) The PHATE embedding of the EB data \( (n = 16825 \text{ cells}) \) colored by expression levels of selected markers. (C) Heatmap showing gene expression level in each cell in four of the branches starting with ESC. The number of cells in each branch is \( n = 2294, \, 9507, \, 5543, \) and \( 4938 \) for the EN, ME, NE, and NC branches, respectively. Cell ordering is determined using Wanderlust (Bendall et al., *Cell*, vol. 157, no. 3, pp. 714-725, 2014). Genes were selected either manually or by high DREMI scores (Krishnaswamy et al., *Science*, vol. 346, no. 6213, p. 1250689, 2014) between gene expression and cell ordering. (D) The PHATE embedding of the EB data \( (n = 16825 \text{ cells}) \) colored by CD49d expression level from the scRNA-seq data (top) and by Spearman correlation between the scRNA-seq transcription factor expression and the CD49d-sorted bulk RNA-seq transcription factor expression per cell (bottom, \( n = 1213 \text{ transcription factors} \)). (E) Same as D, with CD142 and CD82. The Spearman correlation coefficient is highest in branch vii, which is the branch with the highest CD142 and CD82 expression. Bottom right: Scatter plot of single cell expression levels \( (n = 16825 \text{ cells}) \) between CD82 and CD142. Color corresponds to the Spearman correlation between the scRNA-seq expression and the CD142+CD82+ sorted bulk RNA-seq expression \( (n = 15111 \text{ genes}) \). The branch with highest correlation corresponds to cells that are positive in both CD142 and CD82. (F) Scatter plots showing the gating procedure for FACS sorting cell populations of sub-branch iii (CD49d and CD63) and sub-branch vii (CD82 and CD142). The experiments were repeated independently \( n = 3 \) times.

<table>
<thead>
<tr>
<th>Branch</th>
<th>Sub-branches</th>
<th>Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC</td>
<td>i, ii</td>
<td>1-6</td>
</tr>
<tr>
<td>Neural Crest</td>
<td>iii</td>
<td>13-15</td>
</tr>
<tr>
<td>Neuroectoderm</td>
<td>iv</td>
<td>7-12</td>
</tr>
<tr>
<td>Endoderm</td>
<td>v</td>
<td>16-20</td>
</tr>
<tr>
<td>Mesoderm</td>
<td>vi-vii</td>
<td>20-30</td>
</tr>
</tbody>
</table>

Table 4.2.5: Branches of the EB data identified for analysis and the corresponding clusters (Figure 4.2.13Ai) and sub-branches (Figure 4.2.13Aiii) that comprise them.

with the novel \( \text{EOMES-FOXA2-GATA3+SATB1+KLF8+} \) precursor, which further differentiated into cells expressing posterior EN markers \( \text{NKX2-1, CDX2, ASCL2, and KLF5} \). Finally, a novel \( \text{T+GATA4+CER1+PROX1+} \) cardiac precursor cell was identified within the ME lineage that gave rise to \( \text{TNNT2+} \) cells via a \( \text{GATA6+HAND1+} \) differentiation intermediate.

A more detailed analysis of the novel and canonical cell types derived from the PHATE embedding is given in 4.D.

### 4.2.7 Experimental Validation of PHATE-Identified Lineages

We next used the ability of PHATE to extract data on specific regions within the visualization to define a set of surface markers for the isolation and molecular characterization of specific cell populations within the EB differentiation process.

We focused on two specific regions that correspond to the NC branch (sub-branch iii, Figure 4.2.13Aiii) and cardiac precursor sub-branch within the ME branch (sub-branch vii, Figure 4.2.13Aiii). Differential expression analysis identified a set of candidate markers for each region (Figures 4.2.13D-E). We focused on
Table 4.2.6: The number of cells in each cluster of the EB data presented in Figure 4.2.13Ai.

<table>
<thead>
<tr>
<th>Cluster</th>
<th># of cells</th>
<th>Cluster</th>
<th># of cells</th>
<th>Cluster</th>
<th># of cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>499</td>
<td>11</td>
<td>432</td>
<td>21</td>
<td>619</td>
</tr>
<tr>
<td>2</td>
<td>301</td>
<td>12</td>
<td>77</td>
<td>22</td>
<td>84</td>
</tr>
<tr>
<td>3</td>
<td>109</td>
<td>13</td>
<td>560</td>
<td>23</td>
<td>2187</td>
</tr>
<tr>
<td>4</td>
<td>213</td>
<td>14</td>
<td>1047</td>
<td>24</td>
<td>1042</td>
</tr>
<tr>
<td>5</td>
<td>630</td>
<td>15</td>
<td>630</td>
<td>25</td>
<td>838</td>
</tr>
<tr>
<td>6</td>
<td>649</td>
<td>16</td>
<td>258</td>
<td>26</td>
<td>1149</td>
</tr>
<tr>
<td>7</td>
<td>828</td>
<td>17</td>
<td>213</td>
<td>27</td>
<td>340</td>
</tr>
<tr>
<td>8</td>
<td>811</td>
<td>18</td>
<td>388</td>
<td>28</td>
<td>455</td>
</tr>
<tr>
<td>9</td>
<td>562</td>
<td>19</td>
<td>131</td>
<td>29</td>
<td>524</td>
</tr>
<tr>
<td>10</td>
<td>432</td>
<td>20</td>
<td>182</td>
<td>30</td>
<td>355</td>
</tr>
</tbody>
</table>

markers with a high Earth Mover’s Distance (EMD) [Treleaven and Frazzoli, 2013] score in the targeted sub-branch, and low EMD scores in all other sub-branches (see Online Methods for more details on the EMD). Based on these analyses and the availability of antibodies, CD49D/ITGA4 was chosen for the neural crest (the highest scoring surface marker for sub-branch iii) while CD142/F3 and CD82 were chosen for cardiac precursors (among the top 6% of surface markers and the top 3% of all genes by EMD). We FACS-purified CD49d+CD63- and CD82+CD142+ and performed bulk RNA-sequencing (Figure 4.2.14F) on these sorted populations.

To verify that we isolated the correct regions, we calculated the Spearman correlation between the gene expression pattern of each cell and the bulk RNA-seq data from the CD49d+CD63- sorted cells (Figures 4.2.13F and 4.2.14D). The correlation coefficient was the highest in the neural crest branch (branch iii), which corresponds to the highest expression of CD49d. Similar results were obtained for the cardiac precursor cells (Figures 4.2.13F and 4.2.14E). Taken together, our analyses show that PHATE has the potential to greatly accelerate the pace of biological discovery by suggesting hypotheses in the form of finely grained populations and identifying markers with which to isolate populations. These populations can be probed further using alternative measurements such as epigenetic or protein expression assays.

4.3 Discussion

With large amounts of high-dimensional high-throughput biological data being generated in many types of biological systems, there is a growing need for interpretable visualizations that can represent structures in data without strong prior assumptions. However, most existing methods are highly deficient at retaining structures of interest in biology. These include clusters, trajectories or progressions of various dimensionality, hybrids of the two, as well as local and global nonlinear relations in data. Furthermore, existing methods
have trouble contending with the sizes of modern datasets and the high degree of noise inherent to biological datasets. PHATE provides a unique solution to these problems by creating a diffusion-based informational geometry from the data, and by preserving a divergence metric between datapoints that is sensitive to near and far manifold-intrinsic distances in the dataspace. Additionally, PHATE is able to offer clean and denoised visualizations because the information geometry created in PHATE is based on data diffusion dynamics which are robust to noise. Thus, PHATE reveals intricate local as well as global structure in a denoised way.

We applied PHATE to a wide variety of datasets, including single-cell CyTOF and RNA-seq data, as well as Gut Microbiome and SNP data, where the datapoints are subjects rather than cells. We also tested PHATE on network data, such as Hi-C and Facebook networks. In each case, PHATE was able to reveal structures of visual interest to humans that other methods entirely miss. Moreover, we have implemented PHATE in a scalable way that enables it to process millions of datapoints in a matter of hours. Hence, PHATE can efficiently handle the datasets that are now being produced using single-cell RNA sequencing technologies.

To showcase the ability of PHATE to explore data generated in new systems, we applied PHATE to our newly generated human EB differentiation dataset consisting of roughly 31,000 cells sampled over a differentiation time course. We found that PHATE successfully resolves cellular heterogeneity and correctly maps all germ layer lineages and branches based on scRNA-seq data alone, without any additional assumptions on the data. Through detailed sub-population and gene expression analysis along these branches we identified both canonical and novel differentiation intermediates. The insights obtained with PHATE in this system will be a valuable resource for researchers working on early human development, human ES cells, and their regenerative medicine applications.

We expect numerous biological, but also non-biological, data types to benefit from PHATE, including applications in high-throughput genomics, phenotyping, and many other fields. As such, we believe that PHATE will revolutionize biomedical data exploration by offering a new way of visualizing, exploring and extracting information from large scale high-dimensional data.

**Algorithm 4.3.1** function $\text{PHATE}(X,K) = Y$

Compute the PHATE visualization from data.

**Input:**
- Data set $X$
- Kernel parameters $K$

**Output:**
- PHATE visualization $Y$

1. Compute the pairwise distances from $X$.
2. Transform the distances to affinities using parameters $K$ to encode local information.
3. Learn global relationships via the diffusion process.
4. Encode the learned relationships using the potential distance.
5. Embed the potential distance information into low dimensions for visualization.
4.4 Methods

Here we present an expanded explanation of our computational methods, experimental methods, and data processing steps. For the computational details, we first provide a detailed overview of the PHATE algorithm followed by a robustness analysis of PHATE with respect to the parameters and the number of datapoints. We then provide details on the scalable version of PHATE, identifying branch points and branches, and the EMD score analysis.

The embedding provided by PHATE is designed for visualizing global and local structure in the data in exploratory settings with the following properties in mind: 1) The visualization should capture the relevant structure in low (2-3) dimensions. 2) The visualization should preserve and emphasize global and local structure including transitions and clusters. 3) The visualization is denoised to enable data exploration. 4) The visualization is robust in the sense that the revealed structure is insensitive to user configurations.

The mathematical steps of PHATE are provided in Table 4.4.1. We now provide further details about each of the steps in the PHATE algorithm and we explain how these steps ensure that PHATE meets the four properties described above. For even further mathematical details of the algorithm, see 4.4.1.

Algorithm 4.4.1 The steps in the PHATE algorithm.

| Input:  | Data matrix \( X \)  |
| Neighbor size \( k \)  |
| Locality scale \( \alpha \)  |
| Embedding dimension \( m \)  |
| Output: | PHATE embedding \( Y_m \)  |

1: \( D \leftarrow \text{compute pairwise distance matrix from } X \)
2: Compute the \( k \)-nearest neighbor distance \( \varepsilon_k(x) \) for each column \( x \) of \( X \)
3: \( K_{k,\alpha} \leftarrow \text{compute local affinity matrix from } D \text{ and } \varepsilon_k \) (see Eq. 4.3)
4: \( P \leftarrow \text{normalize } K_{k,\alpha} \text{ to form a Markov transition matrix (diffusion operator; see Eq. 4.2)} \)
5: \( t \leftarrow \text{compute time scale via Von Neumann Entropy (see Eq. 4.5)} \)
6: Diffuse \( P \) for \( t \) time steps to obtain \( P^t \)
7: Compute potential representations: \( U_t \leftarrow -\log(P^t) \)
8: \( \Upsilon^t \leftarrow \text{compute potential distance matrix from } U_t \) (see Eq. 4.6)
9: \( Y_{class} \leftarrow \text{apply classical MDS to } \Upsilon^t \)
10: \( Y_m \leftarrow \text{apply metric MDS to } \Upsilon^t \text{ with } Y_{class} \text{ as an initialization} \)

4.4.1 Distance Preservation

Consider the common approach of linearly embedding the raw data matrix itself, e.g., with PCA, to preserve the global structure of the data. PCA finds the directions of the data that capture the largest global variance. However, in most cases local transitions are noisy and global transitions are nonlinear. Therefore, linear notions such as global variance maximization are insufficient to capture latent patterns in the data,
and they typically result in a noisy visualization (Figure 4.B.1 Column 2). To provide reliable *structure preservation* that emphasizes transitions in the data, we need to consider the *intrinsic* structure of the data. This implies and motivates preserving distances between data points (e.g., cells) that consider gradual changes between them along these nonlinear transitions (Figure 4.2.1A-B).

### 4.4.2 Local Affinities and the Diffusion Operator

A standard choice of a distance metric is the Euclidean distance. However, global Euclidean distances are not reflective of transitions in the data, especially in biological datasets that have nonlinear and noisy structures. For instance, cells sampled from a developmental system, such as hematopoiesis or embryonic stem cell differentiation, show gradual changes where adjacent cells are only slightly different from each other. But these changes quickly aggregate into nonlinear transitions in marker expression along each developmental path. Therefore, we transform the global Euclidean distances into local affinities that quantify the similarities between nearby (in the Euclidean space) data points (Figure 4.2.1C).

A common approach to transforming global (e.g. Euclidean) distances to local similarities is to apply a kernel function to all pairs of points. A popular kernel function is the Gaussian kernel $k_\varepsilon(x, y) = \exp(-\|x - y\|^2 / \varepsilon)$ that quantifies the similarity between the two points $x$ and $y$ based on their Euclidean distance. The bandwidth $\varepsilon$ determines the radius (or spread) of neighborhoods captured by this kernel. Let $\mathcal{X} \subset \mathbb{R}^d$ be a dataset with $N$ points sampled i.i.d. from a probability distribution $p : \mathbb{R}^d \to [0, \infty)$ (with $\int p(x) dx = 1$) that is essentially supported on a low dimensional manifold $\mathcal{M}^m \subset \mathbb{R}^d$, where $m$ is the dimension of $\mathcal{M}$ and $m \ll d$. A kernel matrix that includes all pairwise measures of local affinity is constructed by computing the kernel function between all pairs of points in $\mathcal{X}$.

Embedding local affinities directly can result in a loss of global structure as is evident in t-SNE (Figures 4.1.1 4.2.9 4.B.2 and 4.B.1) or kernel PCA embeddings. For example, t-SNE only preserves data clusters, but not transitions between clusters, since it does not enforce any preservation of global structure. In contrast, a faithful structure-preserving embedding (and visualization) needs to go beyond local affinities (or distances), and also consider global relations between parts of the data. To accomplish this, PHATE is based on constructing a diffusion geometry to learn and represent the shape of the data (Coifman and Lafon 2006, Nadler et al. 2005, 2006). This construction is based on computing local similarities between data points, and then *walking or diffusing* through the data using a Markovian random-walk diffusion process to infer more global relations (Figure 4.2.1D).

The initial probabilities in this random walk are calculated by normalizing the row-sums of the kernel
matrix. In the case of the Gaussian kernel described above, we obtain the following:

\[ \nu_\varepsilon(x) = \|k_\varepsilon(x, \cdot)\|_1 = \sum_{z \in \mathcal{X}} k_\varepsilon(x, z) \]  

resulting in a \( N \times N \) row-stochastic matrix

\[ \left[ P_\varepsilon \right]_{(x,y)} = \frac{k_\varepsilon(x, y)}{\nu_\varepsilon(x)}, \quad x, y \in \mathcal{X}. \]

The matrix \( P_\varepsilon \) is a Markov transition matrix where the probability of moving from \( x \) to \( y \) in a single time step is given by \( \Pr[x \to y] = \left[ P_\varepsilon \right]_{(x,y)} \): This matrix is also referred to as the diffusion operator.

### 4.4.3 The \( \alpha \)-decaying Kernel and Adaptive Bandwidth

When applying the diffusion map framework to data, the choice of the kernel \( K \) and bandwidth \( \varepsilon \) plays a key role in the results. In particular, choosing the bandwidth corresponds to a tradeoff between encoding global and local information in the probability matrix \( P_\varepsilon \). If the bandwidth is small, then single-step transitions in the random walk using \( P_\varepsilon \) are largely confined to the nearest neighbors of each data point. In biological data, trajectories between major cell types may be relatively sparsely sampled. Thus, if the bandwidth is too small, then the neighbors of points in sparsely sampled regions may be excluded entirely and the trajectory structure in the probability matrix \( P_\varepsilon \) will not be encoded. Conversely, if the bandwidth is too large, then the resulting probability matrix \( P_\varepsilon \) loses local information as \( \left[ P_\varepsilon \right]_{(x, \cdot)} \) becomes more uniform for all \( x \in \mathcal{X} \), which may result in an inability to resolve different trajectories. Here, we use an adaptive bandwidth that changes with each point to be equal to its \( k \)th nearest neighbor distance, along with an \( \alpha \)-decaying kernel that controls the rate of decay of the kernel.

The original heuristic proposed in [Coifman and Lafon (2006)] suggests setting \( \varepsilon \) to be the smallest distance that still keeps the diffusion process connected. In other words, it is chosen to be the maximal 1-nearest neighbor distance in the dataset. While this approach is useful in some cases, it is greatly affected by outliers and sparse data regions. Furthermore, it relies on a single manifold with constant dimension as the underlying data geometry, which may not be the case when the data is sampled from specific trajectories rather than uniformly from a manifold. Indeed, the intrinsic dimensionality in such cases differs between mid-branch points that mostly capture one-dimensional trajectory geometry, and branching points that capture multiple trajectories crossing each other.

This issue can be mitigated by using a locally adaptive bandwidth that varies based on the local density of the data. A common method for choosing a locally adaptive bandwidth is to use the \( k \)-nearest neighbor
(NN) distance of each point as the bandwidth. A point \( x \) that is within a densely sampled region will have a small \( k \)-NN distance. Thus, local information in these regions is still preserved. In contrast, if \( x \) is on a sparsely sampled trajectory, the \( k \)-NN distance will be greater and will encode the trajectory structure. We denote the \( k \)-NN distance of \( x \) as \( \varepsilon_k(x) \) and the corresponding diffusion operator as \( P_k \).

A weakness of using locally adaptive bandwidths alongside kernels with exponential tails (e.g., the Gaussian kernel) is that the tails become heavier (i.e., decay more slowly) as the bandwidth increases. Thus for a point \( x \) in a sparsely sampled region where the \( k \)-NN distance is large, \([P_k(x,\cdot)]\) may be close to a fully-supported uniform distribution due to the heavy tails, resulting in a high affinity with many points that are far away. This can be mitigated by using the following kernel

\[
K_{k,\alpha}(x, y) = \frac{1}{2} \exp\left(-\left(\frac{\|x - y\|_2}{\varepsilon_k(x)}\right)^\alpha\right) + \frac{1}{2} \exp\left(-\left(\frac{\|x - y\|_2}{\varepsilon_k(y)}\right)^\alpha\right),
\]

which we call the \( \alpha \)-decaying kernel. The exponent \( \alpha \) controls the rate of decay of the tails in the kernel \( K_{k,\alpha} \). Increasing \( \alpha \) increases the decay rate while decreasing \( \alpha \) decreases the decay rate. Since \( \alpha = 2 \) for the Gaussian kernel, choosing \( \alpha > 2 \) will result in lighter tails in the kernel \( K_{k,\alpha} \) compared to the Gaussian kernel. We denote the resulting diffusion operator as \( P_{k,\alpha} \). This is similar to common utilizations of Butterworth filters in signal processing applications [Butterworth, 1930]. See Figure 4.4.2B for a visualization of the effect of different values of \( \alpha \) on this kernel function.

Our use of a locally adaptive bandwidth and the kernel \( K_{k,\alpha} \) requires the choice of two tuning parameters: \( k \) and \( \alpha \). \( k \) should be chosen sufficiently small to preserve local information, i.e., to ensure that \([P_{k,\alpha}(x,\cdot)]\) is not a fully-supported uniform distribution. However, \( k \) should also be chosen sufficiently large to ensure that the underlying graph represented by \( P_{k,\alpha} \) is sufficiently connected, i.e., the probability that we can walk from one point to another within the same trajectory in a finite number of steps is nonzero.

The parameter \( \alpha \) should also be chosen with \( k \). \( \alpha \) should be chosen sufficiently large so that the tails of the kernel \( K_{k,\alpha} \) are not too heavy, especially in sparse regions of the data. However, if \( k \) is small when \( \alpha \) is large, then the underlying graph represented by \( P_{k,\alpha} \) may be too sparsely connected, making it difficult to learn long range connections. Thus we recommend that \( \alpha \) be fixed at a large number (e.g. \( \alpha \geq 10 \)) and then \( k \) can be chosen sufficiently large to ensure that points are locally connected. In practice, we find that choosing \( k \) to be around 5 and \( \alpha \) to be about 10 works well for all the data sets presented in this work. However, the PHATE embedding is robust to the choice of these parameters as discussed later in the Online Methods.

In addition to progression or trajectory structures, the recommendations provided in this section work well for visualizing data that naturally separate into distinct clusters. In particular, the \( \alpha \)-decay kernel
ensures that relationships are preserved between distinct clusters that are relatively close to each other.

### 4.4.4 Propagating Affinities via Diffusion

Here we discuss diffusion, i.e., raising the diffusion operator to its $t$-th power as shown in Table 4.4.1 (Figure 4.2.1D). To simplify the discussion we use the notation $P$ for the diffusion operator, whether defined with a fixed-bandwidth Gaussian kernel or our adaptive kernel. This matrix is referred to as the diffusion operator, since it defines a Markovian diffusion process that essentially only allows single-step transitions within local data neighborhoods whose sizes depend on the kernel parameters ($\varepsilon$ or $k$ and $\alpha$). In particular, let $x \in \mathcal{X}$ and let $\delta_x$ be a Dirac at $x$, i.e., a row vector of length $N$ with a one at the entry corresponding to $x$ and zeros everywhere else. The $t$-step distribution of $x$ is the row in $P_t^\varepsilon$ corresponding to $x$:

$$p^t_x \triangleq \delta_x P^t = [P^t]_{(x,:)}.$$  

(4.4)

These distributions capture multi-scale (where $t$ serves as the scale) local neighborhoods of data points, where locality is considered via random walks that propagate over the intrinsic manifold geometry of the data. This provides a global and robust intrinsic data distance that preserving the overall structure of the data. In addition to learning the global structure, powering the diffusion operator has the effect of low-pass filtering the data such that the main pathways in it are emphasized and small noise dimensions are diminished, thus achieving the denoising objective of our method as well.

### 4.4.5 Choosing the Diffusion Time Scale $t$ with Von Neumann Entropy

The diffusion time scale $t$ is an important parameter that affects the embedding. The parameter $t$ determines the number of steps taken in a random walk. A larger $t$ corresponds to more steps compared to a smaller $t$. Thus, $t$ provides a tradeoff between encoding local and global information in the embedding. The diffusion process can also be viewed as a low-pass filter where local noise is smoothed out based on more global structures. The parameter $t$ determines the level of smoothing. If $t$ is chosen to be too small, then the embedding may be too noisy. On the other hand, if $t$ is chosen to be too large, then some of the signal may be smoothed away.

We formulate a new algorithm for choosing the timescale $t$. Our algorithm quantifies the information in the powered diffusion operator with various values of $t$. This is accomplished by computing the spectral or Von Neumann Entropy (VNE) [Neumann 1932, Anand et al. 2011] of the powered diffusion operator. The amount of variability explained by each dimension is equal to its eigenvalue in the eigendecomposition of the related (non-Markov) affinity matrix that is conjugate to the Markov diffusion operator. The VNE is
calculated by computing the Shannon entropy on the normalized eigenvalues of this matrix. Due to noise in the data, this value is artificially high for low values of $t$, and rapidly decreases as one powers the matrix. Thus, we choose values that are around the "knee" of this decrease.

More formally, to choose $t$, we first note that its impact on the diffusion geometry can be determined by considering the eigenvalues of the diffusion operator, as the corresponding eigenvectors are not impacted by the time scale. To facilitate spectral considerations and for computational ease, we use a symmetric conjugate

$$[A(x,y)] = \sqrt{\nu(x)}[P(x,y)]/\sqrt{\nu(y)}$$

of the diffusion operator $P$ with the row-sums $\nu$. This symmetric matrix is often called the diffusion affinity matrix. The VNE of this diffusion affinity is used to quantify the amount of variability. It can be verified that the eigenvalues of $A^t$ are the same as those of $P^t$, and furthermore these eigenvalues are given by the powers $\{\lambda_i^t\}_{i=1}^{N-1}$ of the spectrum of $P$. Let $\eta(t)$ be a probability distribution defined by normalizing these (nonnegative) eigenvalues as $[\eta(t)]_i = \lambda_i^t / \sum_{j=0}^{N-1} \lambda_j^t$. Then, the VNE $H(t)$ of $A^t$ (and equivalently of $P^t$) is given by the entropy of $\eta(t)$, i.e.,

$$H(t) = -\sum_{i=1}^{N} [\eta(t)]_i \log[\eta(t)]_i,$$

(4.5)

where we use the convention of $0 \log(0) \equiv 0$. The VNE $H(t)$ is dominated by the relatively large eigenvalues, while eigenvalues that are relatively small contribute little. Therefore, it provides a measure of the number of the relatively significant eigenvalues.

The VNE generally decreases as $t$ increases. As mentioned previously, the initial decrease is primarily due to a denoising of the data as less significant eigenvalues (likely corresponding to noise) decrease rapidly to zero. The more significant eigenvalues (likely corresponding to signal) decrease much more slowly. Thus the overall rate of decrease in $H(t)$ is high initially as the data is denoised but then low for larger values of $t$ as the signal is smoothed. As $t \to \infty$, eventually all but the first eigenvalue decrease to zero and so $H(t) \to 0$.

To choose $t$, we plot $H(t)$ as a function of $t$ as in the first plot of Figure 4.4.2C. Choosing $t$ from among the values where $H(t)$ is decreasing rapidly generally results in noisy visualizations and embeddings (second plot in Figure 4.4.2C). Very large values of $t$ result in a visualization where some of the branches or trajectories are combined together and some of the signal is lost (fourth plot in Figure 4.4.2C). Good PHATE visualizations can be obtained by choosing $t$ from among the values where the decrease in $H(t)$ is relatively slow, i.e. the set of values around the “knee” in the plot of $H(t)$ (third plot in Figure 4.4.2C and the PHATE visualizations in Figure 4.1.1). This is the set of values for which much of the noise in the data
CHAPTER 4. VISUALIZING STRUCTURE IN HIGH-DIMENSIONAL DATA

has been smoothed away, and most of the signal is still intact. The PHATE visualization is fairly robust to
the choice of \( t \) in this range, as discussed later in this section.

In the code, we include an automatic method for selecting \( t \) based on a knee point detection algorithm
that finds the knee by fitting two lines to the VNE curve \([Kaplan\ 2012]\). This algorithm calculates the error
between the VNE plot and two lines fitted to the data. The first line has endpoints at the first VNE value
and the suggested knee point. The second line has endpoints at the suggested knee point and the last VNE
value. The suggested knee point with the minimum error is selected.

### 4.4.6 Potential Distances

To resolve instabilities in diffusion distances and embed the global structure captured by the diffusion ge-
ometry in low (2 or 3) dimensions, we use a novel diffusion-based informational distance, which we call
potential distance (Figure 4.2.1E). It is calculated by computing the distance between log-transformed tran-
sition probabilities from the powered diffusion operator. The key insight in formulating the potential distance
is that an informational distance between probability distributions is more sensitive to global relationships
(between far-away points) and more stable at boundaries of manifolds than straight point-wise comparisons
of probabilities (i.e., diffusion distances). This is because the diffusion distance is sensitive to differences
between the main modes of the diffused probabilities and is largely insensitive to differences in the tails. In
contrast, the potential distance, or more generally informational distances, use a submodular function (such
as a log) to render distances sensitive to differences in both the main modes and the tails. This gives PHATE
the ability to preserve both local and manifold-intrinsic global distances in a way that is optimized for vi-
sualization. The resulting metric space also quantifies differences between energy potentials that dominate
“heat” propagation along diffusion pathways (i.e., based on the heat-equation diffusion model) between data
points, instead of simply considering transition probabilities along them.

The potential distance is inspired by information theory and stochastic dynamics, both fields where
probability distributions are compared for different purposes. First, in information theory literature, inform-
ation divergences are used to measure discrepancies between probability distributions in the information
space rather than the probability space, as they are more sensitive to differences between the tails of the
distributions as described above. Second, when analyzing dynamical systems of moving particles, it is not
the point-wise difference between absolute particle counts that is used to compare states, but rather the
ratio between these counts. Indeed, in the latter case the *Boltzmann Distribution Law* directly relates these
ratios to differences in the energy of a state in the system. Therefore, similar to the information theory case,
dynamical states are differentiated in energy terms, rather than probability terms. We employ the same
reasoning in our case by defining our potential distance using localized diffusion energy potentials, rather than diffusion transition probabilities.

To go from the probability space to the energy (or information) space, we log transform the probabilities in the powered diffusion operator and consider an $L^2$ distance between these localized energy potentials in the data as our intrinsic data distance, which forms an M-divergence between the diffusion probability distributions \cite{Salicru1985,Salicru1995}. Mathematically, if $U^t_x = -\log(p^t_x)$ for $x \in X$, then the t-step potential distance is defined as

$$V^t(x, y) = ||U^t_x - U^t_y||_2, x, y \in X.$$  \hspace{1cm} (4.6)

To give a more intuitive view, consider two points $x$ and $y$ that are on different sides of a line of points $W = \{w_1, w_2, \ldots, w_n\}$ (See Figure 4.2.1E), suppose that there is a small set of distant points $Z = \{z_1, z_2, \ldots, z_n\}$ that are on the same side of $W$ as $y$ but opposite side as $x$ such that they are twice as far from $x$ as from $y$. The representation of each point $x$ is as its t-step diffusion probability to all other points. So to compute the potential distance between $x$ and $y$ we compare these probabilities. What is the right type of distance to measure the distinction between these two probability distributions? One solution has been the diffusion distance which is simply the Euclidean distance between these probability distributions. However, in the example mentioned above the diffusion distance would be dominated by larger probabilities and the probabilities to the $Z$ points would not affect the distance from $x$ to $y$ perhaps making them seem close. But instead, we take a divergence between the probabilities from $x$ and $y$ by first log-scale transforming the probabilities and then taking their Euclidean distance, which makes the distance sensitive to fold-change. Thus, if a probability of 0.01 from $x$ to a point $z_i$ is changed to 0.02 from $y$ then this has the same effect as if the probabilities had been 0.1 and 0.2. Thus, PHATE is sensitive to small differences in probability distribution corresponding to differences in long-range global structure, which allows PHATE to preserve global manifold relationships using this potential distance.

We note that the potential distance is a particular case of a wider family of diffusion-based informational distances that view the diffusion geometry as a statistical manifold in information geometry. See \textsection 4.A for details on this family of distances.

### 4.4.7 Embedding the Potential Distances in Low Dimensions

A popular approach for embedding diffusion geometries is to use the eigendecomposition of the diffusion operator to build a diffusion map of the data. However, this approach tends to isolate progression trajectories into numerous diffusion coordinates (i.e., eigenvectors of the diffusion operator; see Figure 4.4.1). In fact,
this specific property was used in Haghverdi et al. (2016) as a heuristic for ordering cells along specific developmental tracks. Therefore, while diffusion maps preserve global structure and denoise the data, their higher intrinsic dimensionality is not amenable for visualization. Instead, we squeeze the variability into low dimensions using metric multidimensional scaling (MDS), a distance embedding method (Figure 4.2.1F).

There are multiple approaches to MDS. Classical MDS (CMDS) Cox and Cox (2008) takes a distance matrix as input and embeds the data into a lower-dimensional space as follows. The squared potential distance matrix is double centered:

$$B = -\frac{1}{2} J \mathcal{W}^{(2)} J,$$

where $\mathcal{W}^{(2)}$ is the squared potential distance matrix (i.e. each entry is squared) and $J = I - \frac{1}{N} 1 1^T$ with $1$ a vector of ones with length $N$. The CMDS coordinates are then obtained by an eigendecomposition of the matrix $B$. This is equivalent to minimizing the following “strain” function:

$$\text{Strain}(\hat{x}_1, \ldots, \hat{x}_N) = \sqrt{\sum_{i,j} (B_{ij} - \langle \hat{x}_i, \hat{x}_j \rangle)^2 / \sum_{i,j} B_{ij}^2},$$

over embedded $m$-dimensional coordinates $\hat{x}_i \in \mathbb{R}^m$ of data points in $X$. We apply CMDS to the potential distances of the data to obtain an initial configuration of the data in low dimension $m$.

While classical MDS is computationally efficient relative to other MDS approaches, it assumes that the input distances directly correspond to low-dimensional Euclidean distances, which is overly restrictive in our setting. Metric MDS relaxes this assumption by only requiring the input distances to be a distance metric. Metric MDS then embeds the data into lower dimensions by minimizing the following “stress” function:

$$\text{Stress}(\hat{x}_1, \ldots, \hat{x}_N) = \sqrt{\sum_{i,j} (\mathcal{W}^{(2)}_{x_i,x_j} - \| \hat{x}_i - \hat{x}_j \|)^2 / \sum_{i,j} \mathcal{W}^{(2)}_{x_i,x_j}^2}.$$

over embedded $m$-dimensional coordinates $\hat{x}_i \in \mathbb{R}^m$ of data points in $X$.

If the stress of the embedded points is zero, then the input data is faithfully represented in the MDS embedding. The stress may be nonzero due to noise or if the embedded dimension $m$ is too small to represent the data without distortion. Thus, by choosing the number of MDS dimensions to be $m = 2$ (or $m = 3$) for visualization purposes, we may trade off distortion in exchange for readily visualizable coordinates. However, some distortion of the distances/dissimilarities is tolerable in many of our applications since precise dissimilarities between points on two different trajectories are not important as long as the trajectories are visually distinguishable. By using metric MDS, we find an embedding of the data with the desired dimension for visualization and the minimum amount of distortion as measured by the stress. When analyzing the
PHATE coordinates (e.g. for clustering or branch detection), we use metric MDS with $m$ chosen to explain most of the variance in the data as determined by the eigenvalues of the diffusion operator (as is done for von Neumann entropy). In this case, minimal distortion is introduced into the analysis.

A naïve approach towards obtaining a truly low dimensional embedding of diffusion geometries is to directly apply metric MDS, from the diffusion map space to a two dimensional space. However, as seen in Figures 4.B.1 (Column 5) and 4.B.2, direct embedding of these distances produces distorted visualizations. Embedding the potential distances (defined in Def. 4.A.1) is more stable at boundary conditions near end points compared to diffusion maps, even in the case of simple curves that contain no branching points. Figure 4.4.2A shows a half circle embedding with diffusion distances versus distances between log-scaled diffusion. We see that points are compressed towards the boundaries of the figure in the former. Additionally, this figure demonstrates that in the case of a full circle (i.e., with no end points or boundary conditions), our potential embedding (PHATE) yields the same representation as diffusion maps.

PHATE achieves an embedding that satisfies all four properties delineated previously: PHATE preserves and emphasizes the global and local structure of the data via:

1. a localized affinity that is chained via diffusion to form global affinities through the intrinsic geometry of the data;

2. denoises the data by low-pass filtering through diffusion;
Figure 4.4.2:
3. provides a distance that accounts for local and global relationships in the data and has robust boundary conditions for purposes of visualization; and

4. captures the data in low dimensions, using MDS, for visualization.

We have shown by demonstration in Figures 4.B.1 and 4.B.2 that all of the steps of PHATE, including the potential transform and MDS, are necessary, as diffusion maps, t-SNE on diffusion maps, and MDS on diffusion maps fail to provide an adequate visualization in several benchmark test cases with known ground truth (even when using the same customized $\alpha$-decaying kernel we developed for PHATE). We have also shown that PHATE is robust to the choice of parameters.

### 4.4.8 Robustness Analysis of PHATE

Here we show that the PHATE embedding is robust to subsampling and the choice of parameters. We demonstrate this both qualitatively and quantitatively. For the quantitative demonstrations, we simulated scRNA-seq data using the Splatter package (Zappia et al., 2017) as in Section 4.2.2. We first calculated the geodesic pairwise distances for the noiseless data. Then for each setting, we calculated the pairwise Euclidean distances in the 2-dimensional embedding. We then compared the geodesic distances with the embedded distances via the Spearman correlation coefficient to compute DEMaP. We used both the paths and groups options of the Splatter package. Simulation details are discussed later in Online Methods.

Tables 4.2.1 to 4.2.4 show that PHATE is robust to subsampling on the Splatter datasets. For the paths dataset, the average Spearman correlation is the same when 95% and 50% of the data points are retained. For the groups dataset, the correlation drops slightly when going from 95% retention to 50% retention. Additionally, the correlation coefficient is still quite high (and better than all other methods) when only 5% of the data points are retained. Thus, quantitatively, PHATE is robust to subsampling.

We also demonstrate this visually. We ran PHATE on the iPSC mass cytometry dataset from Zunder et al. (2015) with varying subsample sizes $N$. Figure 4.2.2A shows the PHATE embedding for $N = 1000, 2500, 5000, 10000$. Note that the primary branches or trajectories that are visible when $N = 50000$ (Figure 4.2.5C) are still visible for all subsamples. Thus, PHATE is robust to the subsampling size. Similar results can be obtained on other datasets.

We also show that the PHATE embedding is robust to the choice of $t$, $k$, and $\alpha$. Figure 4.2.2B shows the PHATE embedding on the iPSC mass cytometry dataset from Zunder et al. (2015) with varying scale parameter $t$. This figure shows that the embeddings for $50 \leq t \leq 200$ are nearly identical. Thus, PHATE is very visually robust to the scale parameter $t$. Similar results can be obtained on other datasets and with the $k$ and $\alpha$ parameters.
The embedding is also quantitatively robust to the parameter choices. Figure 4.2.2C-D shows heatmaps of the Spearman correlation coefficient between geodesic distances of the ground truth data and the Euclidean distances of the PHATE visualization applied to the simulated Splatter datasets for different values of \(k\), \(t\), and \(\alpha\). For \(\alpha \geq 10\), the correlation coefficients are very similar for all values of \(k\), \(t\), and \(\alpha\). This demonstrates that PHATE is robust to the choices of these parameters.

### 4.4.9 Scalability of PHATE

The native form of PHATE is limited in scalability due to the computationally intensive steps of computing potential distances between all pairs of points, computing metric MDS, and storing in memory the diffused operator. Thus, we describe here, and in Table 4.4.2, an alternative way to compute a PHATE embedding that is highly scalable and provides a good approximation of the native PHATE described previously. The scalable version of PHATE uses a slight difference in computing \(t\)-step diffusion probabilities between points. It requires that every other step that the diffusion takes goes through one of a small number of “landmarks.” Each landmark is selected to be a central point that is representative of a portion of the manifold, selected by spectrally clustering manifold dimensions.

**Algorithm 4.4.2** The steps in the scalable PHATE algorithm.

<table>
<thead>
<tr>
<th>Input:</th>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data matrix</td>
<td>PHATE embedding</td>
</tr>
<tr>
<td>Neighborhood size (k)</td>
<td>Points (Y_{\text{points}})</td>
</tr>
<tr>
<td>Locality scale</td>
<td></td>
</tr>
<tr>
<td>Embedding dimension (m)</td>
<td></td>
</tr>
<tr>
<td>Number of landmarks (M)</td>
<td></td>
</tr>
</tbody>
</table>

1. \(K_{k,\alpha} \leftarrow \) compute sparse \(\alpha\)-decaying kernel with radius-based nearest neighbor search
2. \(P \leftarrow \) normalize \(K_{k,\alpha}\) to form a Markov transition matrix (diffusion operator; see Eq. 4.2)
3. \(C_1, \ldots, C_M \leftarrow \) compute landmarks clusters by applying spectral clustering to \(P\)
4. \(P_{NM} \leftarrow \) compute transition probabilities from points to landmarks (see Eq. 4.10)
5. \(P_{MN} \leftarrow \) compute transition probabilities from landmarks to points (see Eq. 4.11)
6. \(P_{MM} \leftarrow P_{MN}P_{NM}\)
7. \(Y_{\text{landmarks}} \leftarrow \) compute \(m\) dimensional embedding as in Table 4.4.1 using \(P_{MM}\) instead of \(P\)
8. \(Y_{\text{points}} \leftarrow \) compute final embedding as \(P_{NM}Y_{\text{landmarks}}\)

First, we construct the \(\alpha\)-decaying kernel on the entire dataset. This can be calculated efficiently and stored as a sparse matrix by using radius-based nearest neighbor searches and thresholding (i.e., setting to zero) connections between points below a specified value (e.g., 0.0001), as we regard them numerically insignificant for the constructed diffusion process. The resulting affinity matrix \(K_{k,\alpha}\) will be sparse as long as \(\alpha\) is sufficiently large (e.g., \(\alpha \geq 10\)) to enforce sharp decay of the captured local affinities. The full diffusion operator \(P\) is constructed from \(K_{k,\alpha}\) by normalizing by row-sums as described previously.

However, powering the sparse diffusion operator would result in a dense matrix, causing memory issues.
To avoid this, we instead perform diffusion between points via a series of $M$ landmarks where $M < N$. We select the landmarks by first applying PCA to the diffusion operator and then using $k$-means clustering on the principal components to partition the data into $M$ clusters. This is a variation on spectral clustering.

We then calculate the probability of transitioning in a single step from the $i$-th point in $\mathcal{X}$ to any point in the $j$-th cluster for all pairs of points and clusters. Mathematically, we can write this as

$$P_{NM}(i, j) = \sum_{\xi \in C_j} P(i, \xi)$$

(4.10)

where $C_j$ is the set of points in the $j$th cluster. Thus, we can view each cluster as being represented by a landmark and the $(i, j)$-th entry in $P_{NM}$ gives the probability of transitioning from the $i$th point in $\mathcal{X}$ to the $j$-th landmark in a single step. Similarly, we construct the matrix $P_{MN}$ where the $(j, i)$-th entry contains the probability of transitioning from the $j$-th landmark to the $i$-th point in $\mathcal{X}$. In this case, we cannot simply sum the transition probabilities $P(\xi, i)$, $\xi \in C_j$, since we also have to consider the prior probability $Q(j, \xi)$ of the $\xi$-th point (with $\xi \in C_j$) being the source of a transition from a cluster $C_j$. For this purpose we use the prior proposed in Wolf et al. (2012), and write

$$P_{MN}(j, i) = \sum_{\xi \in C_j} Q(j, \xi) P(\xi, i)$$

(4.11)

with $Q(j, \xi) = \sum_i K_{k, \alpha}(\xi, i) / \sum_{\zeta \in C_j} \sum_i K_{k, \alpha}(\zeta, i)$.

We use the two constructed transition matrices to compute $P_{MM} = P_{MN}P_{NM}$, which provides the probability of transitioning from landmark to landmark in a random walk by walking through the full point space. Diffusion is then performed by powering the matrix $P_{MM}$. This can be written as

$$P_{MM}^t = P_{MN}P_{NM}P_{MN}P_{NM} \ldots P_{MN}P_{NM}.$$  

(4.12)

From this expression, we see that powering the matrix $P_{MM}$ is equivalent to taking a random walk between landmarks by walking from landmarks to points and then back to landmarks $t$ times.

We then embed the landmarks into the PHATE space by calculating the potential distances between landmarks and applying metric MDS to the potential distances. Denote the resulting embedding as $Y_{\text{landmarks}}$. We then perform an out of sample extension to all points from the landmarks by multiplying the point to landmark transition matrix $P_{NM}$ by $Y_{\text{landmarks}}$ to get

$$Y_{\text{points}} = P_{NM}Y_{\text{landmarks}}.$$  

(4.13)
CHAPTER 4. VISUALIZING STRUCTURE IN HIGH-DIMENSIONAL DATA

Since $M$ is chosen to be vastly less than $N$, the memory requirements and computational demands of the powering the diffusion operator and embedding the potential distances are much lower.

The described steps are summarized in Table 4.4.2. In Figure 4.2.4A-E we show that this constrained diffusion preserves distances between datapoints in the final PHATE embedding, with the scalable version giving near-identical results to the exact computation of PHATE. Further, in Figure 4.2.4B we show that the embedding achieved by this approach is robust to the number of landmarks chosen.

We note that if the only computational bottleneck were in computing MDS, scalable versions of MDS could be used (De Silva and Tenenbaum, 2004; Platt, 2005; Yang et al., 2006). However, since storing the entries of the powered diffusion operator in memory is also an issue, we employ the use of landmarks earlier in the process. It has also been shown that "compressing" the process of diffusion through landmarks in the fashion described here performs better than simply applying Nystrom extension (which includes landmark MDS (Platt, 2005)) to diffusion maps (Gigante et al., 2019).

The fast version of PHATE was used in Figures 4.2.9, 4.2.10, 4.2.11, 4.4.2D, 4.2.4A-E, 4.2.12 and 4.4.1. All other plots were generated using the exact version of PHATE.

To demonstrate the scalability of PHATE for data exploration on large datasets, we applied PHATE to the 1.3 million mouse brain cell dataset from 10x (10x, 2017). Figure 4.2.4C shows a comparison of PHATE to t-SNE, colored by 10 of the 60 clusters provided by 10x. We see that PHATE retains cluster coherence while t-SNE shatters some of the cluster structure.

4.4.10 Branch Identification

Here we describe the methods we developed for identifying branches in a PHATE visualization and selecting representative branch- and endpoints.

We use the estimated local intrinsic dimensionality to identify branch points. We can regard intrinsic dimensionality in terms of degrees of freedom in the progression modeled by PHATE. If there is only one fate possible for a cell (i.e. a cell lies on a branch as in Figure 4.2.6Aii) then there are only two directions of transition between data points—forward or backward—and the local intrinsic dimension is low. If on the other hand, there are multiple fates possible, then there are at least three directions of transition possible—a single direction backwards and at least two forward. This cannot be captured by a one dimensional curve and will require a higher dimensional structure such as a plane, as shown in Figure 4.2.6Aii. Thus, we can use the concept of local intrinsic dimensionality for identifying branch points.

We used the local intrinsic dimension estimation method derived in Costa and Hero III (2006); Carter et al. (2010) to provide suggested branch points. This method uses the relationship between the radius and
volume of a \( d \)-dimensional ball. The volume increases exponentially with the dimensionality of the data. So as the radius increases by \( \delta \), the volume increases by \( \delta^d \) where \( d \) is the dimensionality of the data. Thus the intrinsic dimension can be estimated via the growth rate of a \( k \)-nn ball with radius equal to the \( k \)-nn distance of a point. The procedure is as follows. Let \( Z_n = \{z_1, \ldots, z_n\} \) be a set of independent and identically distributed random vectors with values in a compact subset of \( \mathbb{R}^d \). Let \( N_{k,j} \) be the \( k \) nearest neighbors of \( z_j \); i.e. \( N_{k,j} = \{z \in Z_n \setminus \{z_j\} : |z - z_j| \leq \epsilon_k(z_j)\} \). The \( k \)-nn graph is formed by assigning edges between a point in \( Z_n \) and its \( k \)-nearest neighbors. The power-weighted total edge length of the \( k \)-nn graph is related to the intrinsic dimension of the data and is defined as

\[
L_{\gamma,k}(Z_n) = \sum_{i=1}^{n} \sum_{z \in N_{k,i}} ||z - z_i||^\gamma,
\]

where \( \gamma > 0 \) is a power weighting constant. Let \( m \) be the global intrinsic dimension of all the data points in \( Z_n \). It can be shown that for large \( n \),

\[
L_{\gamma,k}(Z_n) = n^{\beta(m)} c + \epsilon_n,
\]

where \( \beta(m) = (m - \gamma)/m \), \( \epsilon_n \) is an error term that decreases to 0 as \( n \to \infty \), and \( c \) is a constant with respect to \( \beta(m) \) (Costa and Hero III, 2006). A global intrinsic dimension estimator \( \hat{m} \) can be defined based on this relationship using non-linear least squares regression over different values of \( n \) (Costa and Hero III, 2006, Carter et al., 2010).

A local estimator of intrinsic dimension \( \tilde{m}(i) \) at a point \( z_i \) can be defined by running the above procedure in a smaller neighborhood about \( z_i \). This approach is demonstrated in Figure 4.2.6A, where a \( k \)-nn graph is grown locally at each point in the data. However, this estimator can have high variance within a neighborhood. To reduce this variance, majority voting within a neighborhood of \( z_i \) can be performed:

\[
\hat{m}(i) = \arg \max_\ell \sum_{z_j \in N_{k,i}} I(\tilde{m}(j) = \ell),
\]

where \( I(\cdot) \) is the indicator function (Carter et al., 2010).

We note that other local intrinsic dimension estimation methods could be used such as the maximum likelihood estimator in (Levina and Bickel, 2005).

We also identify endpoints in the PHATE embedding. These points can correspond to the beginning or end-states of differentiation processes. For example, Figure 4.2.5A shows the PHATE visualization of the iPSC CyTOF dataset from Zunder et al. (2015) with highlighted endpoints, or end-states, of the repro-
grammed and refractory branches. While many major endpoints can be identified by inspecting the PHATE visualization, we provide a method for identifying other endpoints or end-states that may be present in the higher dimensional PHATE embedding. We identify these states using data point centrality and distinctness as described below.

First, we compute the centrality of a data point by quantifying the impact of its removal on the connectivity of the graph representation of the data (as defined using the local affinity matrix $K_{k,\alpha}$). Removing a point that is on a one dimensional progression pathway, either branching point or not, breaks the graph into multiple parts and reduces the overall connectivity. However, removing an endpoint does not result in any breaks in the graph. Therefore we expect endpoints to have low centrality, as estimated using the eigenvector centrality measure of $K_{k,\alpha}$.

Second, we quantify the distinctness of a cellular state relative to the general data. We expect the beginning or end-states of differentiation processes to have the most distinctive cellular profiles. As shown in Cheng et al. (2018) we quantify this distinctness by considering the minima and the maxima of diffusion eigenvectors (see Figure 4.2.6Ai). Thus we identify endpoints in the embedding as those that are most distinct and least central.

After identifying branch points and endpoints, the remaining points can be assigned to branches between two branch points or between a branch point and endpoint. Due to the smoothly-varying nature of centrality and local intrinsic dimension, the previously described procedures identify regions of points as branch points or endpoints rather than individual points. However, it can be useful to reduce these regions to representative points for analysis such as branch detection and cell ordering. To do this, we reduce these regions to representative points using a “shake and bake” procedure similar to that in David and Averbuch (2012). This approach groups collections of branch points or endpoints together into representative points based on their proximity.

Let $\mathcal{V}_n = \{v_1, \ldots, v_n\}$ be the set of branch points and endpoints in the high-dimensional PHATE coordinates that we wish to reduce. We create a Voronoi partitioning of these points as follows. We first permute the order of $\mathcal{V}_n$, which we denote as $\mathcal{V}'_n = \{v'_1, \ldots, v'_n\}$. We then take the first point $v'_1$ and find all the points in $\mathcal{V}'_n$ that are within a distance of $h$, where $h$ is a scale parameter provided by the user. These points (including $v'_1$) are assigned to the first component of the partition and removed from the set $\mathcal{V}'_n$. This process is then repeated until all points in $\mathcal{V}_n$ are assigned to the partition. To ensure that each point is assigned to the nearest component of the partition (as measured by proximity to the centroid), we next calculate the distance of each point to all centroids of the partition, and reassign the point to the component with the nearest centroid. This reassignment process is repeated until a stable partition is achieved. This completes the process of constructing the Voronoi partition.
The Voronoi partition constructed from this process may be sensitive to the ordering of the points in \( V_n \). To reduce this sensitivity, we repeat this process multiple times (e.g., 40-100) to create multiple Voronoi partitions. We then construct a distance between points by estimating the probability that two points are not in the same component from this partitioning process. This provides a notion of distance that is robust to noise, random permutations, and the scale parameter \( h \). We then partition the data again using the above procedure except we use these probability-based distances. The representative points are then selected from the resulting centroids of this final partition.

A representative point is labeled an endpoint if the corresponding collection of points contains one or more endpoints as identified using centrality and distinctness. Otherwise, the representative point is labeled a branch point.

After representative points have been selected, the remaining points can be assigned to corresponding branches. We use an approach based on the branch point detection method in Haghverdi et al. (2016) that compares the correlation and anticorrelation of neighborhood distances. However, we use higher dimensional PHATE coordinates instead of the diffusion maps coordinates. Figure 4.2.6Aiii gives a visual demonstration of this approach. Here we consider two reference cells \( X \) and \( Y \). We wish to determine if cells \( Q_1 \) and \( Q_2 \) belong to the branch between \( X \) and \( Y \) or not. Consider \( Q_1 \) first which does belong to this branch. If we move from \( Q_1 \) towards \( X \), we also move farther away from \( Y \). Thus the distances to \( X \) and \( Y \) of a neighborhood of points around \( Q_1 \) (which will be located on the branch) are negatively correlated with each other. Now consider \( Q_2 \) which does not belong to the branch between \( X \) and \( Y \). In this case, if we move from \( Q_2 \) towards \( Y \), we also move closer to \( X \). Thus the distances to \( X \) and \( Y \) of a neighborhood of points around \( Q_2 \) are positively correlated with each other. In practice, these distance-based correlations are computed for each possible branch and the point is assigned to the branch with the largest anticorrelation (i.e. the most negative correlation coefficient).

4.4.11 EMD Score Analysis

The EMD is measure of dissimilarity between two probability distributions that is particularly popular in computer vision (Rubner et al., 1998). The EMD was chosen to perform differential expression analysis in the EB scRNA-seq data due to its stability in estimation compared to other divergence measures. Intuitively, if each distribution is viewed as a pile of dirt, the EMD can be thought of as the minimum cost of converting one pile of dirt into the other. If the distributions are identical, then the cost is zero. When comparing univariate distributions (as we do as we only consider a single gene at a time), the EMD simplifies to the \( L^1 \) distance between the cumulative distribution functions (Treleaven and Frazzoli, 2013). That is, if \( P \)
and $Q$ are the cumulative distributions of densities $p$ and $q$, respectively, then the EMD between $p$ and $q$ is $\int |P(x) - Q(x)|\,dx$. While the EMD is nonnegative, we assign a sign to the EMD score based on the difference between the medians of the distributions.

4.4.12 Biological Methods

The processes for generating the EB data and for preprocessing the biological data are described here.

4.4.13 Generation of Human Embryoid Body Data

Low passage H1 hESCs were maintained on Matrigel-coated dishes in DMEM/F12-N2B27 media supplemented with FGF2. For EB formation, cells were treated with Dispase, dissociated into small clumps and plated in non-adherent plates in media supplemented with 20% FBS, which was prescreened for EB differentiation. Samples were collected during 3-day intervals during a 27 day-long differentiation timecourse. An undifferentiated hESC sample was also included (Figure 4.2.14A). Induction of key germ layer markers in these EB cultures was validated by qPCR (data not shown). For single cell analyses, EB cultures were dissociated, FACS sorted to remove doublets and dead cells and processed on a 10x genomics instrument to generate cDNA libraries, which were then sequenced. Small scale sequencing determined that we have successfully collected data on 31,161 cells distributed throughout the timecourse. After preprocessing the data as described below, we are left with 16,825 cell measurements for data analysis. See also the Life Sciences Reporting Summary for further details.

4.4.14 Data Preprocessing

Here we discuss methods we used to preprocess the various datasets.

Data Subsampling: The full PHATE implementation scales well for sample sizes up to approximately $N = 50000$. For $N$ much larger than 50000, computational complexity can become an issue due to the multiple matrix operations required. All of the scRNAseq datasets considered in this paper have $N < 50000$. Thus, we used the full data and did not subsample these datasets. However, the mass cytometry datasets have much larger sample sizes. To aid in branch analysis, we randomly subsampled these datasets for analysis in Section 4.2.3 using uniform subsampling. For the comparison figures (Figures 4.2.9, 4.B.1, and 4.B.2), scalable PHATE was used and subsampling was not performed except as indicated in the figures. The PHATE embedding is robust to the number of samples chosen, which we demonstrated in Section 4.4.8.

Mass Cytometry Data Preprocessing: We process the mass cytometry datasets according to Bendall et al. (2011).
Single-Cell RNA-Sequencing Data Preprocessing: This data was processed from raw reads to molecule counts using the Cell Ranger pipeline \cite{zheng2017}. Additionally, to minimize the effects of experimental artifacts on our analysis, we preprocess the scRNA-seq data. We first filter out dead cells by removing cells that have high expression levels in mitochondrial DNA. In the case of the EB data which had a wide variation in library size, we then remove cells that are either below the 20th percentile or above the 80th percentile in library size. scRNA-seq data have large cell-to-cell variations in the number of observed molecules in each cell or library size. Some cells are highly sampled with many transcripts, while other cells are sampled with fewer. This variation is often caused by technical variations due to enzymatic steps including lysis efficiency, mRNA capture efficiency, and the efficiency of multiple amplification rounds \cite{grun2014}. Removing cells with extreme library size values helps to correct for these technical variations. We then drop genes that are only expressed in a few cells and then perform library size normalization. Normalization is accomplished by dividing the expression level of each gene in a cell by the library size of the corresponding cell.

After normalizing by the library size, we take the square root transform of the data and then perform PCA to improve the robustness and reliability of the constructed affinity matrix $K_{k,\alpha}$. We choose the number of principal components to retain approximately 70% of the variance in the data which results in 20-50 principal components.

Gut Microbiome Data Preprocessing: We use the cleaned L6 American Gut data and remove samples that are near duplicates of other samples. We then preprocess the data using a similar approach for scRNA-seq data. We first perform “library size” normalization to account for technical variations in different samples. We then log transform the data and then use PCA to reduce the data to 30 dimensions.

Applying PHATE to this data reveals several outlier samples that are very far from the rest of the data. We remove these samples and then reapply PHATE to the log-transformed data to obtain the results that are shown in Figure 4.1.1D.

ChIP-seq Processing for Hi-C Visualization: We used narrow peak bed files and took the average peak intensity for each bin at a 10 kb resolution. For visualization, we smoothed the average peak intensity values based on location using a 25 bin moving average.

4.4.15 DEMaP

To quantitatively compare each dimensionality reduction tool, we wish to calculate the degree to which each method preserves the underlying structure of the reference dataset and removes noise. Since single-cell RNA-sequencing and other biological types of data are highly noisy, visual renderings of the data that can offer denoised embeddings that reveal the underlying structure of the data are desirable. Therefore, the goal of
our accuracy metric is to quantify the correspondence between distances in the low-dimensional embedding and manifold distances in the ground truth reference.

To define a quantitative notion of manifold distance we use geodesic distances. Geodesic distances are shortest path distances on a nearest-neighbor graph of the data weighted by the Euclidean distances between connected points \cite{Tenenbaum2000}. In cases where points are sampled noiselessly from a manifold, such as in our ground truth reference, geodesic distances converge exactly to distances along the manifold of the data \cite{Tenenbaum2000,Balasubramanian2002}. Thus we reason that if geodesic distances between points on the noiseless manifold are preserved by an embedding computed on the noisy data then the data is sufficiently denoised and the manifold structure is also preserved.

We take this approach to formulate our ground-truth manifold distance as a quantification of the degree to which each dimensionality reduction method preserves the pairwise geodesic distances of the noiseless data after low-dimensional embedding of the corresponding noisy data. Since the low dimensional embedding is often a result of a non-linear dimensionality reduction, curves and major paths in the data are “straightened” such that Euclidean distances in the embedding space correspond to manifold distance in the high dimensional space \cite{Cox2008}. Thus we quantify the preservation of manifold distances as the correlation between geodesic distance in the noiseless reference dataset and Euclidean distances in the embedding space as a measure of structure preservation which we call *Denoised Embedding Manifold Preservation (DEMaP)*. An overview of DEMaP is presented in Figure 4.2.7A.

### 4.4.16 Construction of the Artificial Tree Test Case

The artificial tree data shown in Figure 4.1.1B is constructed as follows. The first branch consists of 100 linearly spaced points that progress in the first four dimensions. All other dimensions are set to zero. The 100 points in the second branch are constant in the first four dimensions with a constant value equal to the endpoint of the first branch. The next four dimensions then progress linearly in this branch while all other dimensions are set to zero. The third branch is constructed similarly except the progression occurs in dimensions 9-12 instead of dimensions 5-8. All remaining branches are constructed similarly with some variation in the length of the branches. We then add 40 points at each endpoint and branch point and add zero mean Gaussian noise with a standard deviation of 7. This construction models a system where progression along a branch corresponds to an increase in gene expression in several genes. Prior to adding noise, we also constructed a small gap between the first branch point and the orange branch that splits into a blue and purple branch (see the top set of branches in the left part of Figure 4.1.1B). This simulates gaps that are often present in measured biological data. We also added additional noise dimensions, bringing the
total dimensionality of the data to 60.

### 4.4.17 Splatter Simulation Details

Splatter is a scRNA-seq simulation package that uses a parametric model to generate data with various structures, such as branches or clusters \(\text{Zappia et al. (2017)}\). We use Splatter to simulate multiple ground truth datasets for multiple experiments. To select parameters for the simulation, we fit the Splatter simulation to the EB data, and then modified the resulting dataset from both the Splatter "paths" and the Splatter "groups" simulations as described in Section 4.2.2. Note that we do not make use of Splatter’s built-in dropout function, since it uses a zero-inflated model; multiple studies have shown that an undersampling (binomial) model is more appropriate \(\text{Van Dijk et al. (2018)}\) \(\text{Vieth et al. (2017)}\) \(\text{Brennecke et al. (2013)}\) \(\text{Hwang et al. (2018)}\) \(\text{Kim et al. (2015)}\). Each simulation is performed with 3000 simulated cells. The mean correlation coefficient and standard deviations are calculated from 20 trials.

To generate a diverse set of ground truth references, we simulated 50 datasets containing clusters and 50 datasets containing branches. In each of these simulated datasets, the number and size of the clusters of branches as well as the global position of the clusters or branches with respect to each other is random. Furthermore, the local relationships between individual cells on these structures is random. Finally, the changes in gene expression within clusters or along branches is random. The output of this simulation is the ground truth reference.

Next, we add biological and technical noise to the reference data. First, to simulate stochastic gene expression we use Splatter’s Biological Coefficient of Variation (BCV) parameter, which controls the level of gene expression in each cell following an inverse gamma distribution. Second, to simulate the inefficient capture of mRNA in single cells, we undersample from the true counts using the default BCV. Third, to demonstrate robustness to varying of total genes measured, we randomly remove genes from the data matrix. Finally, to demonstrate robustness to the number of cells captured, we randomly remove cells from each dataset. We vary each of these parameters, including by default some degree of biological variation and mRNA undersampling to each simulation.

The default parameters used in the simulation are the following:

- \text{batchCells=3000}
- \text{nGenes=17580}
- \text{mean.shape=6.6}
- \text{mean.rate=0.45}
- \text{lib.loc=9.1}
- \text{lib.scale=0.33}
- \text{out.prob=0.016}
- \text{out.facLoc=5.4}
CHAPTER 4. VISUALIZING STRUCTURE IN HIGH-DIMENSIONAL DATA

- out.facScale=0.90
- bcv.common=0.18
- bcv.df=21.6
- de.prob=0.2

We also set dropout.type="none", with a post-hoc binomial dropout of 50%. For the groups simulation we drew the number of groups \( n \) from a Poisson distribution with rate \( \lambda = 10 \), and then drew the \( \text{group.prob} \) parameter from a Dirichlet distribution with \( n \) categories and a uniform concentration \( \alpha_1 = \cdots = \alpha_n = 1 \).

For the paths simulation, we set \( \text{group.prob} \) as above, and additionally set the \( i \)th entry in the parameter \( \text{path.from} \) as a random integer between 0 and \( i - 1 \), drew the parameter \( \text{path.nonlinearProb} \) from a uniform distribution on the interval \((0,1)\), and drew the parameter \( \text{path.skew} \) from a beta distribution with shape \( \alpha = 10, \beta = 10 \). Note that here the library size was doubled from the fit value, since the EB data itself suffers from dropout. To reduce the number of genes for the \( n\_\text{genes} \) simulation, we randomly removed genes post-hoc in order to avoid changing the state of the random number generator in building the simulation.

For the ground truth simulations, we set \( \text{bcv.common} \) to 0, did not perform binomial dropout, and did not remove genes or cells. For the \( \text{BCV} \) simulation, we performed 50% post-hoc binomial dropout, did not remove genes or cells, and set \( \text{bcv.common} \) to 0, 0.25, and 0.5. For the \( \text{dropout} \) simulation, we set \( \text{bcv.common} \) to 0.18, did not remove genes or cells, and performed 0%, 50%, and 95% post-hoc binomial dropout. For the \( \text{subsample} \) simulation, we set \( \text{bcv.common} \) to 0.18, performed 50% post-hoc binomial dropout, did not remove genes, and subsampled rows of the matrix to retain 95%, 50%, and 5% of the total cells. For the \( n\_\text{genes} \) simulation, we set \( \text{bcv.common} \) to 0.18, performed 50% post-hoc binomial dropout, did not remove cells, and subsampled columns of the matrix to retain 17000, 10000, and 2000 genes.

4.4.18 PHATE Experimental Details

For all of the quantitative comparisons, we have used the default parameter settings for the PHATE plots. For the majority of the qualitative comparisons in Figures 4.2.9, 4.B.2, and 4.B.1 we also used the default parameter settings for all methods. Exceptions to this are the artificial tree (Figure 4.B.1A), the intersecting circles (Figure 4.B.1D), and the MNIST dataset (Figure 4.B.1I). In these cases, the PHATE parameters have been tuned to give a clearer separation of the branches. However, in general, the default PHATE settings give good results on most datasets, especially those that are complex, high-dimensional, and noisy as demonstrated empirically in Section 4.4.8 The default settings are also used in Figures 4.4.2D, 4.2.12, and 4.C.1 For all other PHATE plots, the parameters were tuned slightly to better highlight the structure of the data.
4.4.19 Data Availability

The embryoid body scRNA-seq and bulk RNA-seq datasets generated and analyzed during the current study are available in the Mendeley Data repository at:

http://dx.doi.org/10.17632/v6n743h5ng.1

Figure 4.2.14A contains images of the raw single cells while Figure 4.2.14F contains scatter plots showing the gating procedure for FACS sorting cell populations for the bulk RNA-seq data.

4.4.20 Code Availability

Python, R, and Matlab implementations of PHATE are available on GitHub, for academic use:

https://github.com/KrishnaswamyLab/PHATE

Acknowledgements

This research was supported in part by: the Gruber Foundation /S.G./; the Eunice Kennedy Shriver National Institute of Child Health & Human Development of the NIH (Award Number: F31HD097958) /D.B./; the Alfred P. Sloan Fellowship (grant FG-2016-6607), the DARPA Young Faculty Award (grant D16AP00117), NSF grants 1620216, 1912906, and the NSF CAREER award (grant 1845856) /M.H./; NIH grant 1R01HG008383-01A1 /R.R.C./; NIH grant R01GM107092 /N.B.I./; IVADO (l’institut de valorisation des données) /G.W./; the Chan-Zuckerberg Initiative (grant ID: 182702), NIH grant R01GM130847, and the State of Connecticut (grant 16-RMB-YALE-07) /S.K./.
Appendix

4.A Mathematical Principles of PHATE

Here we provide details on the mathematical principles that underlie the diffusion process, the potential distances, alternative information distances, and nonmetric MDs. We first briefly discuss the challenges of visualizing high-dimensional data.

Visualizing High-Dimensional Data

A common way of visualizing high dimensional data is by using dimensionality reduction methods, which map high dimensional data points to low dimensional coordinates (two or three dimensions in this case) by minimizing some notion of distortion. For example, PCA aims to preserve variance, and thus its notion of distortion is derived from variance lost by linear projection (Moon and Stirling, 2000). Similarly, diffusion maps preserve diffusion affinities as inner products (Coifman and Lafon, 2006; Coifman et al., 2005), and Isomap preserves geodesic distances as Euclidean distances (Tenenbaum et al., 2000). The minimized notion of distortion in these methods can be derived directly from the appropriate inner products or distances and related to their algorithmic steps. Finally, the popular t-SNE method (Maaten and Hinton, 2008) aims to directly preserve neighborhood structure, and uses the KL divergence between high dimensional Gaussian neighborhoods and low dimensional neighborhoods (captured via Student t-distributions) as its distortion.

The effectiveness of a given dimensionality reduction method in a particular application should be considered by ensuring its distortion notion, or resulting embedding, faithfully preserve structures and patterns of interest. Here, we focus on clustering and progression (e.g., trajectory) structures, due to their established significance and importance in biological data. It should be noted that many, if not most, of the common dimensionality reduction methods were not originally designed for visualization purposes, but rather for alleviating the curse of dimensionality (Bellman, 1957; Bellman and Dreyfus, 1962). While they can be used for visualization by selecting the dimension of the embedding appropriately, there is no guarantee that the resulting visualization will reliably express clusters, trajectories, or other patterns of interest.
The embedding provided by PHATE is specifically designed to enable visualization of global and local structure in exploratory settings with the following properties in mind:

**Visualization:** To enable visualization, PHATE captures variance in low (2-3) dimensions.

**Manifold-structure preserving:** To provide an interpretable view of dynamics (e.g., pathways or progressions) in the data, PHATE preserves and emphasizes global nonlinear transitions in the data, in addition to local transitions.

**Denoising:** To enable unsupervised data exploration, PHATE denoises data such that structure within the data is immediately identifiable.

**Robust:** PHATE produces a robust embedding in the sense that the revealed boundaries and the intersections of progressions within the data are insensitive to user configurations of the algorithm.

PHATE fulfills these criteria with an abstract geometric model based primarily on two properties that we typically observe in high-throughput data (biomedical and otherwise). First, transitions between data points tend to be incremental and gradual. There may be many such patches of incremental change but nevertheless these gradual transitions are usually prevalent. Second, there are a limited number of intrinsic directions (or pathways) along which datapoints progress. Therefore, the dynamics captured by collected data are inherently more like a set of rivers, rather than a cloud (expanding outwards in all directions).

Data with such properties can thus be modeled geometrically by a collection of smoothly varying data patches defined by local neighborhoods \cite{Moon2018}. This collection essentially fits the manifold learning paradigm, which relies on a mathematical manifold model for the geometry of a progression track, together with analysis tools for characterizing it. Furthermore, data manifolds often have a low intrinsic dimension, even if curvature and noise forces them to span a high dimensional ambient volume in the collected feature space. Finally, progression tracks form trajectories, with a limited number of “branching points”, where progression splits into several directions. Therefore, in this case the underlying data geometry can implicitly be regarded as a collection of intrinsically low-dimensional manifolds (i.e., curves, surfaces) that cross each other in branching points. By exploiting this low-dimensional structure, we avoid the effects of the curse of dimensionality.

It has been shown in several works \cite{Berard1994,Jones2008} that manifold geometries are closely related to heat diffusion, which is modeled by the heat equation – a differential equation defined in terms of the Laplace-Beltrami operators. Indeed, meta-stable solutions of the heat equation over a manifold capture its intrinsic properties, while providing embeddings, affinities, and distance metrics that capture intrinsic manifold relations. It has further been shown that these can be robustly discretized for empirical
observations that correlate with hidden (or latent) manifold models, e.g., by considering diffusion maps embedding of the data [Coifman and Lafon 2006; Nadler et al. 2005, 2006]. The embedding obtained by PHATE extends these results by considering this diffusion geometry as a statistical manifold of diffusion distributions and using tools of information geometry (namely, $\alpha$-representations) to capture its metric structure and embed it in visualizable (i.e., two or three) dimensions. Further, as we discuss later in this note, the information distance metric we use also relates to Boltzmann energy potentials of the diffusion process, and therefore it combines together both the dynamical systems and information geometry aspects of data-driven diffusion geometries. In particular, for the case of transition structures, this approach enables the consideration of the underlying data geometry consisting of multiple low-dimensional manifolds (such as trajectory curves) that cross each other, while alleviating boundary-condition instabilities to maintain low dimensionality of the embedded space that is better-suited for visualization.

We note that the trajectory structure is not artificially generated in our case, but rather it is expected to be dominant (albeit latent or hidden) in the data. Therefore, the PHATE visualization will only show trajectory structures when data fits such a geometry; otherwise, other (e.g., cluster) patterns will be expressed in the PHATE visualization.

We now provide some additional mathematical details on the diffusion process, the potential distances, alternative information distances, and nonmetric MDS.

The Diffusion Process

To simplify the discussion we use the notation $P$ for the diffusion operator, whether defined with a fixed-bandwidth Gaussian kernel or our adaptive kernel. Let $x \in \mathcal{X}$ and let $\delta_x$ be a Dirac at $x$, i.e., a row vector of length $N$ with a one at the entry corresponding to $x$ and zeros everywhere else. Recall that the $t$-step distribution of $x$ is the row in $P_t^\varepsilon$ corresponding to $x$:

\[
p_t^x \triangleq \delta_x P_t^\varepsilon = [P_t^\varepsilon]_{(x,\cdot)}.
\]

(4.17)

For appropriate choices of kernel parameters, the diffusion process defined by $P$ is ergodic and it thus has a unique stationary distribution $p^\infty$ that is independent of the initial conditions of the process. Thus $p_x^\infty = p^\infty$ for all $x \in \mathcal{X}$. The stationary distribution $p^\infty$ is the left eigenvector of $P$ with eigenvalue $\lambda_0 = 1$ and can be written explicitly as $\nu / \|\nu\|_1$ with the row-sums from Eq. 4.1 (possibly adapted to use $K_{k,\alpha}$ from Eq. 4.3). It can be shown Nadler et al. (2006) that for fixed-bandwidth Gaussian-kernel diffusion, $p^\infty$ converges asymptotically to the original distribution $p$ of the data as $N \to \infty$ and $\varepsilon \to 0$.

The representation provided by the diffusion distributions $p_t^x$, $x \in \mathcal{X}$, defines a diffusion geometry with
the diffusion distance

\[
D^t(x, y) \triangleq \|p^t_x - p^t_y\|_{\ell_2(1/p^{\infty})} = \left( \sum_{z \in X} \frac{(p^t_x(z) - p^t_y(z))^2}{p^{\infty}(z)} \right)^{1/2},
\]  

(4.18)

which is given by a weighted $\ell_2$ distance between the diffusion distributions originating from the data points $x$ and $y$. This distance incorporates a comparison between intrinsic manifold regions of the two data points as well as the concentration of data between them, i.e., the difference between the mass distributions.

The diffusion distance at all time scales can be approximated by the Euclidean distance in the diffusion map embedding, which is defined as follows. If the diffusion process is connected, the eigenvalues of $P$ can be indexed as

\[1 = \lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{N-1} \geq 0.\]

Let $\psi_i$ and $\phi_i$ be the corresponding $i$th left and right eigenvectors of $P$, respectively. The diffusion map embedding is defined as

\[
\Phi^t(x) = (\lambda_1^t \phi_1(x), \lambda_2^t \phi_2(x), \ldots, \lambda_{N-1}^t \phi_{N-1}(x)).
\]  

(4.19)

The time scale $t$ only impacts the scaling of the embedded coordinates via the powers of the eigenvalues. It can then be shown that $D^t(x, y) = \|\Phi^t(x) - \Phi^t(y)\|_2$.

**Potential Distances**

To provide further mathematical context to the potential distance used in PHATE, we relate it here to the heat propagation dynamics that govern the diffusion geometry and the diffusion process we use to build it. This heat diffusion process can be analyzed by considering two possible scenarios for the origin of the dataset $X$ and its distribution $p$, as described in Nadler et al. (2005, 2006). In the first scenario, the data generation process is modeled as an instantiation of a dynamical system that has reached an equilibrium state independent of the initial conditions. Mathematically, let $U(x)$ be a potential and $w(x)$ be a $d$-dimensional Brownian motion process. The data distribution is the steady state solution of the stochastic differential equation (SDE) $\dot{x} = -\nabla U(x) + \sqrt{2} \dot{w}$, where $\dot{x}$ denotes differentiation of $x$ with respect to time. The time steps of the system are dominated by the forward and backward Fokker-Planck equations. This steady state solution is given by

\[
p(x) = \exp(-U(x)),
\]

up to normalization in the $L^1$ norm to form a proper probability distribution.

The distribution of the data in this case is dominated by the potential $U$ that models the underlying structure of the data. As an example, if the data is uniformly distributed on or around a manifold, then this potential is minimal on the manifold itself and increases rapidly when deviating from the manifold.
underlying potential also incorporates data densities that are not uniform. For example, data clusters are represented as local wells or pits in the underlying potential, while progression trajectories and transitions between clusters are represented as rivers or branches in the potential. See Nadler et al. (2005, 2006) for more details.

In the second scenario, the data generation process is not modeled as a dynamical system. Instead, we consider the data in this case as generated by drawing \( N \) i.i.d. samples from the probability distribution \( p(x) \). We then artificially define the underlying potential of the data as

\[
U(x) = -\log(p(x)).
\]

The potential \( U \) can be used in this scenario since its properties and its relation to the structure of the data are not directly related to the notion of time. Furthermore, in both scenarios, the diffusion-based analysis introduces the notion of diffusion time to reveal intrinsic data geometry. Finally, as shown in Nadler et al. (2005, 2006), in both scenarios the Markov process that defines the diffusion geometry converges asymptotically to a diffusion process governed by Fokker-Planck equations with a potential \( 2U(x) \), whether the original potential is defined naturally or artificially.

Using the same relationship between a potential \( U \) and an equilibrium distribution \( p \), we can define a diffusion potential from the stationary distribution \( p^\infty \) as \( U^\infty = -\log(p^\infty) \). This potential corresponds to data generation using the random walk process defined by \( P_\varepsilon \) with \( t \to \infty \) with random initial conditions. Similarly, if we consider a data generation process using this random walk process with \( t \)-steps and a fixed initial condition \( \delta_x \), then the generated data is distributed according to \( p^t_\varepsilon \) and the corresponding \( t \)-step potential representation of \( x \) is \( U^t_\varepsilon, x = -\log(p^t_\varepsilon) \).

Given the potential representations \( U^t_\varepsilon, x \in \mathcal{X} \) of the data in \( \mathcal{X} \), we define the following potential distance metric as an alternative to the distribution-based diffusion distance:

Definition 4.A.1. The \( t \)-step potential distance is defined as \( \mathcal{V}^t(x, y) \triangleq \|U^t_x - U^t_y\|_2 \), \( x, y \in \mathcal{X} \).

The following proposition shows a relation between the two metrics by expressing the potential distance in embedded diffusion map coordinates\(^1\) for fixed-bandwidth Gaussian-based diffusion (i.e., generated by \( P_\varepsilon \) from Eq. 4.2):

Proposition 4.A.1. Given a diffusion process defined by a fixed-bandwidth Gaussian kernel, the potential distance from Def 4.A.1 can be written as \( \mathcal{V}^t(x, y) = \left( \sum_{z \in \mathcal{X}} \log^2 \left( \frac{1 + \phi^t/2(x, \phi^t/2(y))}{1 + \phi^t/2(y, \phi^t/2(y))} \right) \right)^{1/2} \)

\(^1\)Recall the diffusion distance is simply the Euclidean distance in these coordinates
Proof. According to the spectral theorem, the entries of $P^t_\varepsilon$ can be written as

$$[P^t_\varepsilon](x,y) = \psi_0(y) + \sum_{i=1}^{n-1} \lambda_i^t \phi_i(x) \psi_i(y)$$

since powers of the operator $P_\varepsilon$ only affect the eigenvalues, which are taken to the same power, and since the trivial eigenvalue $\lambda_0$ is one and the corresponding right eigenvector $\phi_0$ only consists of ones. Furthermore, it can be verified that the left and right eigenvectors of $P_\varepsilon$ are related by $\psi_i(y) = \phi_i(y) \phi_0(y)$, thus, combined with Eqs. 4.4 and 4.19 we get

$$p^t_{\varepsilon,x}(y) = \psi_0(y) \left( 1 + \sum_{i=1}^{n-1} \lambda_i^t \phi_i(x) \phi_i(y) \right) = \psi_0(y) \left( 1 + \left\{ \Phi^{t/2}_{\varepsilon}(x), \Phi^{t/2}_{\varepsilon}(y) \right\} \right).$$

By applying the logarithm to both ends of this equation we express the entries of the potential representation $U^t_{\varepsilon,x}$ as

$$U^t_{\varepsilon,x}(y) = -\log(1 + \left\{ \Phi^{t/2}_{\varepsilon}(x), \Phi^{t/2}_{\varepsilon}(y) \right\}) - \log(\psi_0(y)),$$

and thus for any $j = 1, \ldots, N$,

$$\left( U^t_{\varepsilon,x}(x_j) - U^t_{\varepsilon,y}(x_j) \right)^2 = \left[ \log(1 + \left\{ \Phi^{t/2}_{\varepsilon}(x), \Phi^{t/2}_{\varepsilon}(x) \right\}) \right]$$

$$- \left[ \log(1 + \left\{ \Phi^{t/2}_{\varepsilon}(y), \Phi^{t/2}_{\varepsilon}(x) \right\}) \right]^2$$

$$= \log^2 \left( \frac{1 + \left\{ \Phi^{t/2}_{\varepsilon}(x), \Phi^{t/2}_{\varepsilon}(x) \right\}}{1 + \left\{ \Phi^{t/2}_{\varepsilon}(y), \Phi^{t/2}_{\varepsilon}(x) \right\}} \right),$$

which yields the result in the proposition.

Alternative Informational Distances

The potential distance is a particular case of a wider family of diffusion-based informational distances that view the diffusion geometry as a statistical manifold in information geometry. We offer a generalization of these to a family of distances, where the diffusion distance (Eq. 4.18) is at one extreme of this family and the potential distance (Def. 4.A.1) is at the other. Diffusion distances directly compare probability distributions pointwise with no damping of high probabilities. Thus changes in the tails of distributions do not contribute much to the distance. The potential distance is damped using the log function to the point where fold changes, even in low probabilities, have impact on the distance. Other distances, including the Hellinger distance [Hellinger 1909], are in between these two. We introduce a parameter $\gamma$ as a knob to control the level of damping, with the diffusion distance having $\gamma = -1$, the potential distance with $\gamma = 1$. 

and the Hellinger distance at $\gamma = 0$.

Mathematically, let the diffusion distributions $p^t_x$ and $p^t_y$ represent data points $x, y \in \mathcal{X}$. Given an intermediate data point $z \in \mathcal{X}$, and a meta-parameter $-1 \leq \gamma \leq 1$, let

$$
\Delta_{(x,y)}^{(\gamma)}(z) = - \int_{p^t_x(z)} p^t_y(z) \exp \frac{\gamma}{2} \, du = \begin{cases}
  \frac{1}{\gamma} \left[ \log \frac{p^t_x(z)}{p^t_y(z)} \right] & \gamma = -1 \\
  \frac{1}{\gamma} \left[ \left( \frac{p^t_x(z)}{p^t_y(z)} \right)^{\frac{1}{\gamma}} - \left( \frac{p^t_y(z)}{p^t_x(z)} \right)^{\frac{1}{\gamma}} \right] & \gamma = +1 \\
  \left[ \frac{2}{1-\gamma} \left( \frac{p^t_x(z)}{p^t_y(z)} \right)^{\frac{1}{\gamma}} - \left( \frac{p^t_y(z)}{p^t_x(z)} \right)^{\frac{1}{\gamma}} \right] & \text{otherwise}
\end{cases}
$$

(4.20)

quantify the difference between the transition probabilities $p^t_x(z), p^t_y(z)$ from $x, y$ to $z$. Then, diffusion distances and potential distances are given by $L^2$ norms of $\Delta_{(x,y)}^{(-1)}$ and $\Delta_{(x,y)}^{(+1)}$ correspondingly (albeit with a different measure over $z$, since diffusion distances are defined over $L^2(\frac{1}{p^t_x})$ in Eq. 4.18). Therefore, these distances can be regarded as two extremes of a general family of distances over the diffusion geometry. Moreover, as we show in Prop 4.A.2, diffusion dissimilarities of the form $\left\| \Delta_{(x,y)}^{(\gamma)} \right\|_2$ combine both this diffusion-geometry notion of a data-driven distance metric, and an information-theory notion of divergence between diffusion distributions, which is provided in Def. 4.A.2 for completeness.

**Definition 4.A.2 (Divergence [Information Theory]).** Let $S$ be a space of probability distributions with common support. A divergence on $S$ is a function $D(\cdot || \cdot) : S \times S \to \mathbb{R}$ s.t.

1. $D(p || q) \geq 0$ for all $p, q \in S$, and
2. $D(p || q) = 0$ if and only if $p = q$ (Eguchi, 1985).

Some specific classes of divergences include:

1. M-divergence [Salicrú and Pons, 1985; Salicrú et al., 1995]: Let $p$ and $q$ be probability mass functions and let $g$ be a differentiable function with continuous second derivative. Then the M-divergence between $p$ and $q$ is $M_g(p, q) = \sum_i (g(p_i) - g(q_i))^2$.

2. $f$-divergence [Csiszár, 1964; Ali and Silvey, 1966]: Let $f$ be a convex function s.t. $f(1) = 0$ and let $p$ and $q$ be probability mass functions. The $f$-divergence between $p$ and $q$ is $D_f(p || q) = \sum_i q_i f \left( \frac{p_i}{q_i} \right)$.

3. Bregman divergence [Bregman, 1967]: Let $\Omega$ be a closed convex set and let $F : \Omega \to \mathbb{R}$ be a strictly convex and continuously differentiable function. The Bregman divergence between the points $p, q \in \Omega$ is $D_F(p, q) = F(p) - F(q) - \langle \nabla F(q), p - q \rangle$, where $\nabla F(q)$ is the gradient of $F$ evaluated at $q$.

Unlike (formal) distance metrics, a divergence is not required to be symmetric nor satisfy the triangle inequality. Examples of $f$-divergences include the Kullback-Leibler divergence [Kullback and Leibler, 1951].
and the Hellinger distance [Hellinger 1909]. An example of a Bregman divergence is the squared Euclidean distance where $F(x) = x^2$. Note that these different types of divergences are not mutually exclusive as is evident in the following proposition.

**Proposition 4.A.2.** The squared norm $\left\| \Delta^{(\gamma)}_{(x,y)} \right\|^2$ forms an $M$-divergence between the diffusion probability distributions $p^t_x$ and $p^t_y$ for all $\gamma \in [-1, 1]$. Furthermore, $\left\| \Delta^{(\gamma)}_{(x,y)} \right\|^2$ forms an $f$-divergence for $\gamma = 0$ and a Bregman divergence for $\gamma = -1$.

*Proof.* For $\gamma \in (-1, 1)$, it follows that $\left\| \Delta^{(\gamma)}_{(x,y)} \right\|^2$ is an $M$-divergence from Definition 4.A.2 with $g(x) = \frac{2}{1-\gamma} x^{\frac{1-\gamma}{\gamma}}$. Similarly, $g(x) = x$ and $g(x) = \log x$ yield this result for $\gamma = -1, 1$, respectively. For $\gamma = 0$, we obtain $\left\| \Delta^{(0)}_{(x,y)} \right\|^2 = 4 \sum_z \left[ \sqrt{p^t_x(z)} - \sqrt{p^t_y(z)} \right]^2$, which is indeed an $f$-divergence as it is proportional to the Hellinger divergence. Finally, since $\gamma = -1$ yields a squared Euclidean distance between the distributions it is indeed a Bregman divergence.

The family of distances (or divergences) formed by $\left\| \Delta^{(\gamma)}_{(x,y)} \right\|^2$ is also directly related to $\alpha$-representations used in information geometries [Amari 2016] when defining statistical manifolds over probability distributions. Indeed, the $\alpha$-representation of a distribution $p$ is defined as

$$\ell^{(\alpha)}(p) = \begin{cases} 
\frac{2}{1-\alpha} p^{\frac{1-\alpha}{\alpha}} & \alpha \neq 1 \\
\log p & \alpha = 1,
\end{cases}$$

where $\ell^{(-1)}$ and $\ell^{(+1)}$ give rise to the popular mixed family ($m$-family) and exponential family ($e$-family) in information geometry [Amari 2016; Amari and Nagaoka 2000], correspondingly. We also note at this point that the third popular $\alpha$-family in information geometry is the 0-family, which gives a Fisher geometry as its Riemannian metric as given by Fisher information Amari and Nagaoka (2000). Interestingly, this family corresponds to setting $\gamma = 0$ in our case, which yields a distance $\left\| \Delta^{(0)}_{(x,y)} \right\|^2 \propto \|\sqrt{p^t_x} - \sqrt{p^t_y}\|^2$ that is proportional to the Hellinger distance between diffusion distributions.

Since, as we discussed here, $\Delta^{(\gamma)}_{(x,y)}$ encodes differences between information geometry $\ell^{(\gamma)}$ representations of diffusion distributions, we refer to the distances $\left\| \Delta^{(\gamma)}_{(x,y)} \right\|^2$ (from Prop. 4.A.2) as diffusion-based informational distances. In general, this family of informational distances creates an exciting connection between diffusion geometries and information geometries for exploring emergent structures in data exploration. In particular, in PHATE we focus on the potential distance as an $e$-family distance Amari and Nagaoka (2000); Amari (2016) that combines both the Boltzman distribution law approach and the information geometry approach towards capturing a stable metric structure of the diffusion geometry, and use it for the purpose of visualizing progression by embedding this metric structure in low dimensions.
CHAPTER 4. VISUALIZING STRUCTURE IN HIGH-DIMENSIONAL DATA

Figure 4.4.2 shows PHATE visualizations of the retinal bipolar data from Shekhar et al. (2016) using different values of \( \gamma \). This figure also indicates the impact of \( \gamma \) on the global structure captured by these distances, where indeed the global structure of the potential distance (\( \gamma = 1 \)) is more similar (as compared to other \( \gamma \) values) to the structure captured by PCA, which is known to preserve global structure. Another way to see this is that the structure is unraveled less than when using diffusion distances.

Nonmetric MDS

In some cases, it may be advantageous to relax our assumptions further on the input distances. In this case, non-metric MDS may be used. In contrast with metric MDS, non-metric MDS does not require the input distances to be an actual distance or metric. Non-metric MDS minimizes the differences between a monotonic transformation of the input dissimilarities and the distances in the embedded space. Mathematically, non-metric MDS minimizes the following stress function:

\[
\text{Stress}^\prime(\hat{x}_1, \ldots, \hat{x}_N) = \sqrt{\sum_{i,j} \left( f\left( \frac{d^\prime(x_i, x_j)}{\|\hat{x}_i - \hat{x}_j\|} \right) - \|\hat{x}_i - \hat{x}_j\| \right)^2 / \sum_{i,j} \|\hat{x}_i - \hat{x}_j\|^2},
\]

where \( f \) is a monotonic transformation of the input dissimilarities.

In our experience, the resulting visualizations from metric MDS and non-metric MDS are nearly identical for most datasets. Furthermore, metric MDS is computationally faster than non-metric MDS. Thus, we recommend metric MDS for most problems.

4.B Extended Comparison of PHATE to Other Methods

Here we provide an extended qualitative comparison of PHATE to methods of dimensionality reduction, graph rendering, and visualization on multiple datasets. These comparisons are found in Figures 4.B.1 and 4.B.2. Details on parameter selection are included in Section 4.4.18.

Figure 4.B.1 shows comparisons on non-biological datasets. These include (in the order shown)

1. Artificial tree data with 7 branches in 60 dimensions;
2. Simulated Splatter path data Zappia et al. (2017);
3. Simulated Splatter groups data Zappia et al. (2017);
4. Three intersecting curves in 3 dimensions;
5. Video of a rotating teapot Weinberger et al. (2004);
6. Swiss roll data used in Tenenbaum et al. (2000);
7. Frey faces video Roweis and Saul (2000);
8. Columbia Object Image Library (COIL-20), 20 videos of rotating objects Nene et al. (1996);
9. MNIST, 70,000 images of handwritten digits from 0 to 9 LeCun et al. (1998).

Figure 4.B.2 shows comparisons on biological datasets including (in the order shown):
1. Developing mouse bone marrow cells, enriched for the myeloid and erythroid lineages, which were measured with the MARS-seq single cell RNA-sequencing technology Paul et al. (2015);
2. Single cell RNA-sequencing of epithelial cells from mouse small intestine and organoids Haber et al. (2017);
3. Mass cytometry data measuring T cell development into CD8+ and CD4+ T cells in mouse thymus Setty et al. (2016);
4. New embryoid body data from a 27-day timecourse;
5. Mass cytometry data showing iPSC reprogramming of mouse embryonic fibroblasts Zunder et al. (2015);
6. Single cell RNA-sequencing of mouse retinal bipolar cells Shekhar et al. (2016);
7. Single cell RNA-sequencing of mouse cortical cells from the somatosensory cortex and hippocampal CA1 region Zeisel et al. (2015).

Some of these biological datasets represent differentiating processes within the body, and hence visualizing progression is key to understanding the structure of these datasets. Other artificial datasets show a combination of clusters and trajectories, while the artificial datasets give a plausible range of manifold structures that could be found in biological data.

PHATE is primarily a dimensionality reduction method that takes high dimensional raw data and embeds it, via a metric preserving embedding, into low dimensions that naturally show trajectory structure. Thus, we focus our comparisons of PHATE to existing dimensionality reduction methods such as PCA, t-SNE, and diffusion maps.

We note that several methods exist that focus on finding pseudotime orderings of cells, such as Wanderlust Bendall et al. (2014), Wishbone Setty et al. (2016), and diffusion pseudotime Haghverdi et al. (2016). Wanderlust can find single non-branching progressions. Wishbone recognizes a single branch, while diffusion pseudotimes provides potentially multiple branches. However, pseudotime approaches do not naturally provide a dimensionality reduction method to visualize such structure. Therefore, the resulting cell orderings
Figure 4.B.1: Comparison of PHATE to various methods on multiple artificial and non-biological datasets. Note that methods with strong structural assumptions on the data, such as t-SNE (clusters) and Monocle2 (tree) are expected to fail on the subset of datasets which do not fit their assumptions. See 4.B for discussion. See the figure for the respective sample sizes for each dataset.
Figure 4.B.2: (Continued on the following page.)
Figure 4.B.2: Comparison of PHATE to various methods on multiple biological datasets. Note that methods with strong structural assumptions on the data, such as t-SNE (clusters) and Monocle2 (tree) are expected to fail on the subset of datasets which do not fit their assumptions. See 4.B for discussion.

can be difficult to interpret and verify, especially in the context of the entire data set. Since PHATE reveals the entire branching structure in low dimensions, it can be beneficial to visualize, verify and interpret the results of any pseudotime algorithm displayed on PHATE. These pseudotime methods can also be used alongside PHATE to order parts of the branching progressions. Indeed, we use Wanderlust on the EB data in Section 4.2.4 to extract ordering from the branches identified from PHATE. A comprehensive overview and comparison of state of the art pseudotime and trajectory inference methods for single-cell RNA sequencing is given in Saelens et al. (2019).

A few pseudotime methods can create a visualization based on finding and rendering a differentiation tree structure, including SPADE (Qiu et al., 2011; Anchang et al., 2016) and Monocle2 (Qiu et al., 2017). However, SPADE is not well-suited for analyzing scRNA-seq data without prior knowledge (see Anchang et al. (2016) and Section 4.B). Thus we only compare to Monocle2.

Note that we do not include any metadata, such as sample time or clusters, in any of the analyses. Therefore, we are focusing on the performance of these methods in an unsupervised setting. Instead, we use this metadata as a tool for comparing the results of the various methods.

Comparison of PHATE to Dimensionality Reduction Methods

Figures 4.B.1 and 4.B.2 compare the PHATE visualization to the dimensionality reduction methods of principal components analysis (PCA), Diffusion Maps (DM), Metric Multidimensional Scaling (MDS), MDS performed on Diffusion Maps, t-SNE, t-SNE performed on Diffusion Maps, Local Linear Embedding (LLE), Isomap, Force Directed Layout, UMAP and Monocle2 on nine non-biological datasets and seven biological datasets, including the five biological datasets shown on a subset of these methods in Figure 4.2.9. The datasets show a combination of intersecting and distinct manifolds, clusters, and branching trajectories, examining a range of both real biological manifolds and plausible challenging structures for visualization. The PHATE visualization most consistently distinguishes branches, trajectories and clusters in order to give a denoised representation of the underlying structure of the data. We focus on each method individually. A discussion of desirable attributes of some of these methods for the purposes of dimensionality reduction can also be found in Van Der Maaten et al. (2009).

Comparison of PHATE to PCA: PCA is a popular method of data analysis that uses eigendecomposition of the covariance matrix to learn axes within the high-dimensional data that account for the largest
amount of variance within the data Moon and Stirling (2000). However, PCA assumes a linear structure on the data, the visualization amounts to projecting the data onto a slicing plane, which creates a noisy visualization. Also, since biological data are rarely linear, PCA is unable to optimally reduce non-linear noise along the manifold and reveal progression structure in low dimensions. This is evident in Figure 4.B.1A where we compare PCA to PHATE on artificial tree data. This data contains seven distinct branches uniformly sampled in 60 dimensions. See Section 4.4.16 for details. PCA does capture some of the global structure in this relatively low-noise data. However, many branches are not visible in the first two PCA dimensions and the trajectories in the PCA visualization are noisy compared to the PHATE visualization, in which all seven branches are easily identifiable.

For the other datasets in Figures 4.B.1 and 4.B.2, PCA captures some of the overall global structure of the datasets. For example, the PCA dimensions in Figure 4.B.2D encode the overall time progression of the noisy EB scRNA-seq data. Thus, PCA captures some of the global structure. However, PCA presents mostly a cloud of cells in this case and any finer branching structure is not visible. This contrasts with PHATE which shows multiple branches and trajectories. Similar results are obtained from the mouse bone marrow scRNA-seq and iPSC CyTOF datasets in Figures 4.B.2A and E respectively, demonstrating that PCA is unable to accurately visualize the global and local structure of the data simultaneously.

Comparison of PHATE to t-SNE: t-SNE (t-distributed stochastic neighbor embedding) Maaten and Hinton (2008) is a visualization method that emphasizes local neighborhood structure within data. Recently, t-SNE has become popular for revealing cluster structure or separations in single cell data Amir et al. (2013). However, due to its emphasis on preserving local neighborhoods, t-SNE tends to shatter trajectories into clusters as seen in the artificial tree and intersecting curves in Figures 4.B.1A and D as well as the EB data and the iPSC data in Figures 4.B.2D and E. In all of these cases, the data naturally have a strong trajectory structure either by design (the artificial trees and intersecting curves) or due to the developmental nature of the data (the EB and iPSC datasets). Thus t-SNE creates the false impression that the data contain natural clusters, which could lead to incorrect analysis.

Furthermore, the adaptive kernel used in t-SNE for calculating neighborhood probabilities tends to spread out neighbors such that dense clusters occupy proportionally more space in the visualization compared to sparse clusters (Wattenberg et al., 2016). Thus, the relative location of data points within the t-SNE embedding often does not accurately reflect the relationships between them. This is clearly visible in the t-SNE plot in Figure 4.B.1A where the shattered branches are located far away from where they originated in the main structure in the artificial tree data. Similarly, t-SNE creates clusters in the EB and iPSC data in Figures 4.B.2D and E which split the time samples into different components. Since the relative position
of clusters in t-SNE is generally meaningless the overall progression of the data is destroyed.

Even in the case where the data are more naturally separated into clusters, t-SNE can destroy the global information about the relative relationships between clusters due to this weakness. In contrast, PHATE separates clusters that are sufficiently separated from each other (see Figure 4.B.1C) while maintaining the relative relationships of clusters based on the relative positions of the clusters in the PHATE embedding. In other words, PHATE preserves both the global and local structure while t-SNE only preserves local structure.

One proposed solution to the failure of t-SNE to retain global structure is to use a random walk to learn the global structure, and then apply t-SNE to the resulting kernel [Maaten and Hinton (2008)]. One approach to do this is to apply t-SNE to the resulting kernel [Maaten and Hinton (2008)]. One approach to do this is to apply t-SNE to DM. However, our experiments show that this fails to capture the global structure of the data. In the artificial tree data (Figure 4.B.1A) the intersecting curves (Figure 4.B.1D), as well as the new EB data (Figure 4.B.2D) t-SNE on DM shatters trajectories. Furthermore, in the retinal bipolar dataset (Figure 4.B.2F) t-SNE on DM shatters clusters and creates misleading trajectory-like structures. Hence, performing t-SNE on diffusion maps suffers from the same shortcomings as t-SNE, with additional distortion from the denoising aspect of diffusion maps, as t-SNE tends to shatter trajectories less frequently on noisier data (see Figures 4.B.1A and B). Due to the nature of the t-SNE penalty function, global distances encoded in the diffusion distances are ignored, and the resulting embedding is a denoised equivalent of t-SNE, which is more prone to shattering trajectories than t-SNE and lacks the global structure benefits from Diffusion Maps.

Comparison of PHATE to Diffusion Maps: Diffusion maps effectively encode continuous relationships between cells. However, different trajectories are often encoded in different dimensions (i.e., since they represent different meta-stable states of the diffusion process) as seen in Figure 4.4.1 which is unsuitable for visualization. In contrast, PHATE effectively encodes trajectories in lower dimensions for visualization. This is also seen clearly in Figures 4.B.1A and D in the comparison of PHATE to diffusion maps on the artificial tree and intersecting curves data (for each data set, the same kernel and diffusion scale \( t \) is used for both diffusion maps and PHATE). In this case, the diffusion maps visualization is denoised and the global structure is visible. However, multiple branches are not visible in the low-dimensional visualization of diffusion maps. In fact, approximately six diffusion maps coordinates are needed to separate all ten branches (see Figure 4.4.1). In contrast, all of the ten branches of the artificial tree data are clearly visible in the PHATE visualization. Similarly, multiple branches that are visible in the PHATE visualization are not visible in the diffusion maps visualization for the bone marrow and EB scRNA-seq datasets (Figures 4.B.2A and D, respectively). Additionally, the diffusion maps instabilities mentioned previously appear to cause very noisy data (e.g. the scRNA-seq data in Figures 4.B.2A, B, and D) to contract too much into thin trajectories,
which can distort some of the underlying progression structure. In summary, while diffusion maps works well for nonlinear dimensionality reduction, it is not well-suited for visualizing data with multiple trajectories due to its instabilities and its propensity to encode different trajectories in different dimensions.

A logical question is whether applying MDS on diffusion distances would be sufficient for encoding the high dimensional spatial information from diffusion maps in low dimensions for visualization. However, in Figures 4.B.1D and H on the intersecting curves and COIL20 we show that MDS on DM suffers equivalently from instability at boundary conditions and intersections of manifolds, producing a totally structureless embedding. Additionally, MDS on DM collapses trajectories into thin trajectories as shown on the mouse bone marrow scRNAseq, EB data, and iPSC CyTOF data (Figures 4.B.2A, D, and E respectively), masking the intricate structure visible in PHATE. Thus, performing MDS on diffusion maps without applying the informational potential transformation of PHATE is insufficient for high quality visualization.

Comparison of PHATE to MDS: Multidimensional scaling (MDS) (Kruskal and Wish, 1978) aims to preserve or approximate the metric structure of the data by optimizing a stress loss between original (Euclidean) distances and embedded ones. While MDS ostensibly preserves both local and global distances, it does not rely on or infer any particular intrinsic structure from the data. Therefore, it cannot separate meaningful relations in complex high dimensional data from superfluous ones. In particular, this causes MDS to be strongly affected by noise, as shown in Splatter simulations in Figures 4.B.1B and C where the embeddings show no structure at all. Further, this causes problems in visualizing noisy biological datasets, such as the new EB scRNAseq data (Figure 4.B.2D) where local trajectories are lost, and the iPSC CyTOF data (Figure 4.B.2E) where the branching structure is entirely masked by noise. MDS also fails to separate clusters in noisy data, as shown in biological datasets from Zeisel et al. (2015) and Shekhar et al. (2016) (Figures 4.B.2F and G) as well as in MNIST (Figure 4.B.1I).

Comparison of PHATE to Isomap: Isomap (Tenenbaum et al., 2000) embeds the intrinsic metric structure of the data by applying MDS to geodesic distances, which are obtained by constructing a k-nearest neighbor graph over the data, and then applying all-pairs shortest path search (e.g., Dijkstra (Dijkstra, 1959) or Floyd (Floyd, 1962)) to compute distances. Like other manifold learning methods, Isomap works under the assumption that the data is sampled from an underlying manifold, and thus the geodesic distances approximate intrinsic manifold distances and the coordinate assigned by MDS should provide a global intrinsic coordinate system. However, this assumption is mainly valid when the manifold itself is convex, with no holes, and the data is sampled uniformly from it with only small amount of noise away from the manifold. Indeed, the main weaknesses of Isomap is its topological instability when such assumptions are not satisfied.
in practice (Van Der Maaten et al., 2009; Balasubramanian and Schwartz, 2002; Lee and Verleysen, 2005), as we also show here. These instabilities render Isomap susceptible to spurious connections created in noisy datasets, as shown by the failure to separate branches on the artificial tree and Splatter paths (Figure 4.B.1A and B). Further, Isomap is incapable of embedding clusters, such as the Splatter groups (Figure 4.B.1D), the mouse retinal bipolar scRNAseq (Figure 4.B.2F), and MNIST (Figure 4.B.1I), in which many clusters are merged together since geodesic distances do not consider the data distribution and do not quantify relations between disconnected clusters. Additionally, Isomap is also unstable to intersecting manifolds, as shown by the Intersecting Curves dataset, where Isomap fails to distinguish between two of the curves and shows no intersections (Figure 4.B.1D), and does not clearly display branching points, as shown on the iPSC data where Isomap only separates points by timepoint (Figure 4.B.2E).

Comparison of PHATE to Locally Linear Embedding (LLE): LLE Roweis and Saul (2000) is a manifold learning algorithm that is similar to Isomap in that it assumes the data is sampled from a single smooth manifold, approximated by a $k$-NN graph, and tries to use its geometric properties to embed the data. However, unlike Isomap, it only considers local information - namely, it uses the low dimensional coordinate neighborhoods, which are (independently) linearly related to local manifold patches, and attempts to tile these local coordinates into a consistent global embedding in low dimensions. This is done by first optimizing weights that allow the approximation of each data point as a linear combination of its neighbors, and then optimizing low dimensional coordinates that preserve the same linear relations encoded by these weights.

While LLE is less susceptible to shortcut connections, it heavily relies on a smooth well-connected manifold structure to provide global connections through the data. Therefore, LLE is ill-suited for embedding separate clusters, as shown by its failure to separate the distinct digits in the MNIST dataset (Figure 4.B.1I). Further, due to its reliance on local interpolation for encoding relations between each point and its neighbors, it strongly affected by boundary conditions that make such interpolation unstable, as shown by its complete failure to embed the Gaussian Mixture Model (Figure 4.B.1C). Additionally, LLE does not handle intersecting manifolds and high curvatures, due its implicit assumption that local data patches correspond to manifold coordinate neighborhoods, which can be linearly approximated by a tangent space of the same intrinsic dimension of the manifold. For example, on the Intersecting Curves dataset, while PHATE shows a clean intersection between the curves, LLE treats one of the intersections as a noisy subcluster, and the other intersection is not shown, with the orange curve shown simply as a continuation of the blue curve (Figure 4.B.1D). Finally, we note that some additional weaknesses of LLE for visualization of even simple synthetic biomedical data were reported previously, e.g., in Van Der Maaten et al. (2009); Lim et al. (2003).
Comparison of PHATE to UMAP  
Like t-SNE, upon which the UMAP algorithm is modeled, UMAP encourages formation of clusters even when cluster structures do not exist. As such, UMAP shatters trajectories, as shown in the artificial trees and intersecting circles (Figures 4.B.1A and D respectively) as well as in the new EB data (Figure 4.B.2D). Additionally, although UMAP’s algorithm is designed to respect global relationships between the clusters that it forms, this does not always hold between distant clusters; take for example the artificial tree, in which UMAP places the shattered cyan branch at the opposite end of the plot to the green and yellow branches to which it is connected, and does the same for the purple branch, which should be connected to be the brown, orange and red branches, but is placed far from all of these (Figure 4.B.1A).

Comparison of PHATE to Graph-Rendering Methods

Graph-rendering methods differ fundamentally from dimensionality reduction methods in that they do not produce a reduced dimension representation of the data and instead focus only on providing a specific rendering of the data. However, these renderings are often limited by structural assumptions on the data (e.g., a tree) that may be inaccurate. Figures 4.B.1 and 4.B.2 compare the PHATE visualization to graph-rendering methods including Force Directed Layout (FDL) and Monocle2.

Comparison of PHATE to Force-Directed Methods:  
FDL algorithms attempt to draw a weighted graph in a two-dimensional space so that all edge weights are approximately preserved as distances and relatively few edges cross each other. To this end, such methods solve a $n$-body problem modeled by attractive and repulsive forces that resemble physical systems, such as elastic (e.g., Hooke’s law), electric (e.g., Coulomb’s law), or nuclear forces. Typically, repulsive forces are used to separate and spread out all pairs of nodes while attractive ones are used to keep neighboring nodes on the graph close to each other in the embedding. These attractive forces are scaled based on the strength of the connections within the graph. We specifically apply the Spring Layout method within the NetworkX Python package [Hagberg et al. 2008] to the graph defined by the $\alpha$-decaying kernel produced for PHATE. This method uses the Fruchterman-Reingold algorithm [Fruchterman and Reingold 1991], which is motivated by the aesthetics of rendering a planar graph, and models the attractive and repulsive forces based on a combination of notions of elastic attraction, nuclear repulsion, and global stability criteria (e.g., ideal uniform distance and decaying temperature of the system).

FDL algorithms are generally computationally expensive, making it difficult to scale them to larger datasets (Figure 4.2.4E). FDL algorithms also do not denoise the connections between data points, but rather assume that attractive and repulsive forces will eventually balance each other. Thus, they suffer from...
the same sensitivity to graph construction as Isomap, where spurious connections in noisy data can dominate
the resulting force-dynamics and strongly affect the embedding (see the artificial tree and intersecting curves
in Figures 4.B.1A and D respectively). Additionally, exceedingly weak forces between distant clusters, which
are clearly not well-connected in the graph, lead to a failure to retain long-range global distances, as shown
in Figure 4.B.2F, in which rod bipolar cells and cone bipolar cells do not appear substantially distinct. These
shortcomings can lead to the loss of information, such as a lack of branching structure in the iPSC CyTOF
data, or a merging of cell types in the neuronal scRNAseq from Zeisel et al. (2015) (Figures 4.B.2E and G
respectively.)

Comparison of PHATE to Tree-Rendering Methods (SPADE and Monocle2):

SPADE Qiu et al. (2011); Anchang et al. (2016) and Monocle2 Qiu et al. (2017) are popular methods that fit
the data to a predetermined structure such as a tree. These methods first attempt to do data reduction by
clustering the data. Clustering methods tend to make less restrictive assumptions on the structure of the data
compared to PCA. However, clustering methods assume that the underlying data can be partitioned into
discrete separate regions. In reality, biological data are often continuous, and the apparent cluster structure
given by clustering methods is only a result of non-uniform density and finite sampling of the continuous
underlying state space. Additionally, the results from these methods will be incorrect if the underlying data
does not lie on a tree. In contrast, PHATE does not make any assumptions on the data and instead learns
the underlying structure.

SPADE fits a minimal spanning tree to the clusters and was originally designed for mass cytometry
data Qiu et al. (2011). In Anchang et al. Anchang et al. (2016), the authors applied SPADE to scRNA-
seq data by selecting relevant genes to perform dimensionality reduction. This makes it difficult to do data
exploration as gene selection must be performed first. In contrast, PHATE does not require any gene selection
procedure although PHATE can be used to analyze specific genes of interest by including only the relevant
genes. SPADE has several other limitations according to Anchang et al. Anchang et al. (2016). First, the
SPADE results can be sensitive to the number of clusters which must be specified by the user. Second,
down-sampling is required to visualize large datasets. SPADE is very sensitive to random down-sampling
and will produce very different trees even when only down-sampling to 99% Anchang et al. (2016). Thus,
the random nature of the SPADE results make it difficult to discover the right structure of the data with
SPADE. Given these limitations, we do not make a direct comparison between PHATE and SPADE.

Monocle2 also fits a tree to cell clusters using the DDRTree algorithm Mao et al. (2015) 2016 as a
default. We compare Monocle2 to PHATE in Figures 4.B.2 and 4.B.1. For the artificial tree in Figure 4.B.1A,
Monocle2 successfully identifies the branches of the tree. However, for the bone marrow scRNA-seq data in
Figure 4.B.2A, Monocle2 fails to detect several branches that are visible in the PHATE visualization. At the same time, Monocle2 shows several branches that are not detected by PHATE. The number and location of these branches vary from run to run on the same data with the same settings. Thus it is difficult to determine if the branches shown by Monocle2 are spurious or not.

Similar results are obtained when applying Monocle2 to the EB scRNA-seq data where the number and location of the branches within the Monocle2 visualization differ drastically from run to run. Thus it is difficult to determine the underlying structure of this data using Monocle2. In contrast, for the same set of parameters, PHATE produces the same results with each run while preserving the relative relationships between different branches directly in the visualization based on their proximity.

Furthermore, since Monocle2 enforces a tree structure on the data, the application of Monocle2 to data without a tree structure produces unintelligible results, giving tree-shaped visualizations for clustered (Figs. 4.B.1C, H, I and 4.B.2F), circular (Fig. 4.B.1E) and planar (Fig. 4.B.1F data.

### 4.C PHATE for Data Exploration on Various Data Types

PHATE can be used for exploratory data analysis on a variety of data types. Here we present some examples of PHATE applied to other datasets including non-single-cell data such as microbiome data, SNP data, facial images, Hi-C chromatin conformation maps, and Facebook network data.

#### PHATE on High-dimensional High-throughput Data

As a general dimensionality reduction method, PHATE is applicable to many datatypes. Here we show that PHATE reveals and preserves global transitional structure in mouse bone marrow scRNA-seq data, microbiome data, human SNP data, and (non-biological) image data.

#### Extended Analysis of the iPSC Mass Cytometry Data

Here we discuss the remaining branches discovered when applying PHATE to the iPSC mass cytometry data from Zunder et al. (2015). Branch 3 represents an intermediate, partially reprogrammed state also containing Oct4+/Klf4+/CD73+ but is not yet expressing pluripotency markers like Nanog or Lin28. However, the PHATE embedding indicates that Epcam, which is known to promote reprogramming generally (Huang et al., 2011), increases along this branch. This is evidenced by the high DREMI score between Epcam and the cell ordering within the branch (Figure 4.2.5C). This branch joins into branch 4 at a later stage, showing perhaps an alternative path or timing of reprogramming. Finally, branch 1 shows a lineage that has failed to reprogram, perhaps due to the wrong stoichiometry of the reprogramming factors (Takahashi and Yamanaka, 2006). Of note, this lineage
contains low Klf4 which is an essential reprogramming factor.

Additionally, the PHATE embedding shows a decrease in p53 expression in precursor branches (2 and 3) indicating that these cells are released from cell cycle arrest induced by initial reprogramming factor over expression \cite{Hong2009}. However, along the refractory branch (branch 5) we see an increase in cleaved-caspase3, potentially indicating that the failure to reprogram correctly initiates apoptosis in these cells \cite{Zunder2015}.

By default PHATE produces a single low dimensional embedding of a dataset. However, we can obtain variants of this embedding by reweighting the features before computing distances. Such reweightings correspond to specific "views" of the data. For example, in a biological context, we can upweight genes that are involved in a specific process to have PHATE prominently reflect this process. To demonstrate this reweighting scheme, we computed three alternative PHATE embeddings of the iPSC data, by upweighting either cell cycle markers, stem cell markers, or mitotic markers (Figure 4.2.12). PHATE, after upweighting cell cycle markers, gives an embedding with a circular structure (Figure 4.2.12Ai) that reflects the cyclical nature of the cell cycle. In addition to the circular structure, the embedding shows a small protrusion, with high expression of Ccasp3, suggesting that these cells are apoptotic. Upweighting stem cell markers gives an embedding with a 1-dimensional progression. Expression analysis reveals that stem cell markers such as Sox2 are high at one end of the progression and low on the other end. Moreover, the progression is correlated with time (measurement day), further supporting the idea that the progression that PHATE reveals marks the extent to which the cells are stem-like, with early timepoints being less stem-like. Finally, after upweighting mitotic markers, PHATE shows a different 1-dimensional progression. Here, the progression appears to be correlated with mitotic state, as can be seen by the expression of several mitosis-related genes (Figure 4.2.12Aiii), such as pAKT, that are high only in one end of the embedding. Thus, PHATE computed after reweighting the genes can be used to obtain a process specific embedding to gain insight into predefined biological processes.

**Bone Marrow scRNA-seq Data Reveals New Structure**  Figure 4.2.5D shows the color-coded 3D PHATE embedding and gene expression matrix for scRNA-seq data from mouse bone marrow. This data is enriched for myeloid and erythroid lineages and was organized into clusters in \cite{Paul2015}, as shown in Figure 4.2.9A. Here, we show that PHATE reveals a continuous progression structure instead of cluster structure and illustrates the connections between clusters. The PHATE embedding shows a continuous progression from progenitor cell types in the center to erythroid lineages towards the right and myeloid lineages towards the left. The expression matrix shows increasing expression of erythroid markers in the rightmost branches (branches 4, 5, and 6) such as hemoglobin subunits Hba-a2 and Hbb-b1 as well as heme
synthesis pathway enzyme \(Cpox\) as the lineage progresses to the right. Towards the left in branches 1 and 2, we see an enrichment for myeloid markers, including \(CD14\) and \(Elane\), which are primarily monocyte and neutrophil markers, respectively.

In addition, PHATE splits the erythrocytes into three branches not distinguished by the authors of [Paul et al. (2015)]. These branches show differential expression of several genes. Branch 6 is more highly expressed in \(Gata1\) and \(Gfi1B\), both of which are involved in erythrocyte maturation. Branch 4 is also more highly expressed in \(Zfpm1\), which is involved in erythroid and megakaryocytic cell differentiation. Additionally, branches 4 and 5 are more highly expressed in \(Car2\), which is associated with the release of oxygen. Given these differential expression levels, it is likely that the different branches correspond to erythrocytes at different levels of maturity and in different states ([Yang et al. 2008; Crispino 2005; Fujiwara et al. 1996; Pevny et al. 1991; Fiolka et al. 2006; Van der Meer et al. 2010; Yang et al. 2007]). In addition, the branches at the right have high mutual information with \(CD235a\), which is an erythroid marker that progressively increases in those lineages.

PHATE in 3 dimensions more clearly reveals a separation of the myeloid lineages to the left into two branches. \(Cd14\) and \(Sfpi1\) are both more highly expressed at the beginning of branch 2 than in branch 1, suggesting that branch 2 is associated with monocytes while branch 1 is associated with neutrophils.

We note that due to the lack of common myeloid progenitors in this sample, one may expect to see a gap between the monocytes and megakaryocyte lineage, since PHATE does not artificially connect separable data clusters (see Figure 4.B.1C). However, we note that all but one of the 12 embeddings of this data in Figure 4.B.2A also lack a gap between these trajectories, suggesting there is a genuine similarity between cells in the monocyte and megakaryocyte lineages, whether due to biological similarities or a technical artifact of sequencing (e.g., contamination.)

**PHATE reveals transcriptional heterogeneity in Rod Bipolar Cells**  
Figure 4.C.1Ai shows PHATE on scRNA-seq data of mouse retinal bipolar neurons from [Shekhar et al. (2016)]. Cells were collected from an adult mouse and sorted for transgenic retinal bipolar markers. PHATE visualizes cluster structure while preserving relationships between clusters. The embedding is colored by the clusters described in the original study, which seeks to transcriptionally characterize all subtypes of bipolar cells. In the original characterization, rods bipolar cells (the largest cluster of cells) are shown as a single homogeneous cell type. However, the PHATE embedding in Figure 4.C.1Ai reveals a bifurcating trajectory within this cluster. We zoomed in on this bifurcating trajectory, by embedding just those cells, in order to determine their sub-structure.

This embedding of just rod bipolar cells (Figure 4.C.1Aii) reveals four distinct sub-clusters (found by
A B

Figure 4.C.1:

k-means clustering) of rod bipolar cells. We characterize the transcriptional profile of these sub-clusters in Figure 4.C.1B, showing all genes used for cell type assignment in Shekhar et al. (2016). Our results show a trajectory between rod bipolar cell types that is consistent with previous work by Tsukamoto and Omi (2017), in which cell types RB1 and RB2 are shown to be a continuum of variants of a single type. Further, we show distinct differences between these in known marker genes distinguishing RB1 and RB2 Trnp1, Rho and Pde6b (Kim et al., 2008), indicating that the four clusters we observe may be further subtypes of these two cell types.

PHATE on Facial Images

To demonstrate that PHATE can also be used to learn and visualize the underlying structure of nonbiological data, we applied PHATE to the Frey Face dataset used in Roweis and Saul (2000). This dataset consists of nearly 2000 video frames of a single subject’s face in various poses. Figure 4.2.10A shows a 3D visualization of this dataset using PHATE, colored by time. Multiple branches are clearly visible in the visualization and each branch corresponds to the progression of a different pose. A short video highlighting two of these branches is found in Video 4. The video highlights the continuous nature of the data as points along branches correspond to transitions from pose to pose.

PHATE on Microbiome Data Reveals Archetypal Structure

Recently there have been many studies of bacterial species abundance in the human intestinal tract, saliva, vagina and other membranes as measured by sequencing of the 16S ribosomal-RNA-encoding gene (16s sequencing) or by whole genome shotgun sequencing (metagenomics). However, most analysis of microbiome data has been limited to clustering and PCA. Here we use PHATE to analyze microbiome data from the American Gut Project.

First we note that PCA applied to 9660 fecal, oral, and skin samples (Figure 4.2.10B left) results in an undifferentiated cloud with two density centers corresponding to fecal samples on the right and oral/skin samples on the left. In contrast, PHATE shows branching structures with 4 branches emanating from a point of origin for fecal sample, and additional structures on the right that differentiates between skin samples,
which form their own progression, and oral samples, which again result in several branches. Figure 4.2.10C shows the PHATE embedding colored by two genera (Bacteroides and Prevotella) and a phylum (actinobacteria) of bacteria on the same 9660 samples. These figures show that the Bacteroides genus of bacteria is almost exclusively found in the fecal samples. The Prevotella genus of bacteria is found in certain stool and oral samples while the Actinobacteria phylum is primarily found in the oral and skin samples. This is consistent with the work in Silverman et al. (2017) which showed that different genera and phyla of bacteria are prevalent in the different body sites.

Upon “zooming in” to the 8596 fecal samples in Figure 4.2.10D, we see 4 major branches, instead of the three enterotypes reported in previous literature (Arumugam et al., 2011), with highly expressed Firmicutes, Prevotella, Bacteroides and Verrucomicrobia respectively. Furthermore, the Firmicutes/Bacteroides branches seem to form a smooth continuum with samples falling into various parts of a triangular simplex shape typically seen in archetypal analysis (Hart et al., 2015; Shoval et al., 2012). This shows that individuals can exist as mixed phenotypes between archetypal bacterial states as well as in a continuum with more or less prevalence for each of these states.

PHATE on SNP Data Reveals Geographic Structure To demonstrate PHATE on population data, we examined a dataset containing 2345 present-day humans from 203 populations genotyped at 594,924 autosomal SNPs with the Human Origins array (Patterson et al., 2012). In Figure 4.2.10E, we see that as compared to PCA, the PHATE embedding shows clear population structures, such as the near eastern Jewish populations near the bottom (Iranian and Iraqi Jews, Jordanians), with further branches showing progression within the same population, such as the Jordanian population show as orange diamonds (see Figure 4.2.10F for population labels.) Further, PHATE shows a global structure that mimics geography, with European populations generally towards the top and Near Eastern populations towards the bottom. Thus PHATE shows that the occurrence and structure of these SNPs follows a progression based on geography and population divergence. PCA tends to crowd populations together into two linear branches, without clearly distinguishing between population groups or showing population divergence.

PHATE on Connectivity Data

Thus far we have used PHATE to embed high-dimensional data. That is, we have mapped the original feature space of the data to the PHATE dimensions. However, the PHATE algorithm also allows for embedding any data that exists in either an inner product space or a metric space. In other words, we can apply PHATE to data that is naturally described by distances or affinities instead of features. For example, network data (i.e. graphs) can be embedded using PHATE simply by skipping the initial steps of the algorithm that go from
the original feature space to an affinity matrix. We can thus replace the affinity matrix with the network (as represented by an affinity matrix) and proceed with the rest of the PHATE algorithm.

There are abundant examples of natively networked data in biology, such as chromatin conformation contact maps (Hi-C), gene/protein interaction networks, and neural connectivity data such as fMRI. Outside of biology, network data is also prevalent. A common example is social network data that contain information of friend-communities (e.g. Facebook) or interest-communities (e.g. Twitter). We show that PHATE provides a visualization of network data that emphasizes major structure (i.e. pathways) in the network, better than typical graph layout methods.

**PHATE on Hi-C Data Reveals Spatial Chromatin Structure** We use PHATE to visualize human Hi-C data from Darrow et al. (2016) by using the Hi-C contact map as the affinity matrix in the PHATE algorithm. The contact map gives the frequency with which genomic locations are observed in spatial proximity. Hi-C contact maps are typically visualized using the matrix directly – often using the 45 degree counterclockwise rotated upper triangle part of the matrix Darrow et al. (2016). While this depiction can show chromosomal domains, it is not a reconstruction of the actual spatial structure of the chromatin. In contrast, the PHATE embedding reconstructs the relative positions of the genomic locations both locally and globally in such a way that the embedding represents an actual projection of the spatial structure within and between chromosomes. As a result, we get an intuitive visual of the Hi-C contact map that not only shows the topological domains present in the data but also how they are connected to one another. Figure 4.2.11A shows a 3D PHATE embedding of the chromatin, colored by chromosome. We see that the embedding resembles the fractal globule structure proposed in Lieberman-Aiden et al. (2009), with the total chromatin organized in a spherical shape and individual chromosomes mostly connected within themselves.

PHATE can also effectively visualize a single chromosome. Figure 4.2.11B shows PHATE just on chromosome 1 contact map at 10 kilobase (kb) resolution. Each point corresponds to a genomic fragment and is colored by its location within the genome. In this visualization, multiple “folds” are clearly visible. For rotating videos of the 3D visualizations, see Videos 5 and 6.

To validate that the PHATE embedding of Hi-C data is meaningful we color the embedding by the ChIP-seq signals of several chromatin modification markers. Figure 4.2.11C shows a 2D PHATE embedding of chromosome 1 colored by various methylation and acetylation markers (ChIP-seq from Consortium (2012), dataset ENCSR977QPF). Histone methylation and acetylation play an important role in global gene regulation via control of chromatin organization. They are often used in combination with Hi-C data to investigate the open or closed structure of chromatin Darrow et al. (2016), where the ChIP-seq signal of various histone modification markers correlates with so called topologically associated domains or TADs. Figure 4.2.11C
shows an organized methylation (H3K27me3 and H3K4me2) and acetylation (H3K9ac and H3K27ac) signal on PHATE, with clusters of similar intensity of the marker, suggesting that the PHATE embedding has biological meaning with respect to the spatial organization of the chromatin.

These results indicate that PHATE provides a visualization of Hi-C data that captures more spatial information compared to directly looking at the contact map.

**PHATE on Facebook Data Reveals Super Connectors** We visualize Facebook network data using PHATE. The data consist of networks of friends. This network graph is directly converted into a 0-1 affinity matrix and fed to PHATE. Figure 4.2.11D compares the PHATE visualization of the network to a force-directed layout, a common method for visualizing network data. In both plots, edges (friends) are shown and each node/person is colored by degree (the number of friends a person has). The PHATE embedding clearly shows multiple branching structures in the network, that are not visible in the force-directed layout which shows a T shape.

Several subnetworks and important nodes (super-connectors) between subnetworks are visible in the PHATE embedding that are not visible in the force-directed layout visualization. For example, in the top-left corner of the PHATE visualization of the entire network, a single node connects the top-left group of people to the remainder of the network. Several other important nodes can be identified that bridge the gap between the left section of the network and the center region. Thus PHATE can be used for visualizing network data and identifying important features of the network.

We also show that PHATE can be used to find more structure within subnetworks as identified by the friend networks of selected individuals (referred to as ego nodes in Leskovec and Mcauley (2012)) in Figure 4.2.11D. Again, we find that PHATE finds more structure in subnetworks.

Therefore, we see that PHATE can be used to visualize any type of data, high-dimensional or featureless. Further, we see that PHATE will emphasize continuous transitional structure, while maintaining separation between clusters in these datasets.

**4.D Detailed EB Data Results**

Here we provide a more detailed analysis of the EB data based on the PHATE visualization. We examined markers of cell proliferation as they are known to be correlated with the time trend. Undifferentiated hESCs have very fast cell cycles with prominent S and M phases and very short G1 phase, while differentiating cells have longer G1 phase and cycle slower (White and Dalton, 2005). Indeed, we found that S-phase cyclins \(CCNE1\) and \(CCNE2\) had the highest level of expression in the earlier time points while G1-specific CDK
inhibitors **CDKN1A**, **CDKN1B** and **CDKN1C** were induced at later time points as shown in Figure 4.2.14B.

To examine localized lineages and branching structures, we visualized the expression of known germ layer markers on PHATE (as in Figure 4.2.14B), resulting in the germ layer specification chart shown in Figure 4.2.13B. Note that in the remaining description we refer to the branches based on the labels shown in Figure 4.2.13Aiii. These branches are identified as follows. First, we performed clustering of the data using the $k$-means algorithm on ten-dimensional PHATE (Figure 4.2.13Ai) to identify differentiation intermediates within each germ layer branch. This can be viewed as a variant on spectral clustering using PHATE dimensions instead of the eigenvectors of a Laplacian matrix [Von Luxburg, 2007; Ng et al., 2002]. We note that clustering could also be applied directly to the potential distances and other clustering approaches can be used including hierarchical clustering, Louvain clustering [Blondel et al., 2008; Levine et al., 2015], and others [Galluccio et al., 2013; Xu and Tian, 2015]. For each cluster, we performed differential expression analysis by comparing, for each gene, its expression distribution within the cluster to the expression distribution in all other cells (i.e., the “background” expression distribution) using the earth mover’s distance or EMD (Figure 4.2.13D), a measure of dissimilarity between probability distributions [Rubner et al., 1998] (see Section 4.4.11 in the supplement for details). We confirmed that the genes expressed in several differentiation intermediates, which we identified by manual inspection, can also be identified based on the EMD scores of the corresponding cluster. For example, **SOX10** has a high EMD score in clusters 14 and 15, which corresponds with neural crest cells. Similarly, **EOMES** has a high EMD score in clusters 4, 17, and 20, which corresponds with the primitive streak cells. We also observed several canonical differentiation intermediates that were previously defined based on studies in mice. These include anterior NE and NC branches in the ectodermal lineage, anterior EN, cardiac and lateral plate ME, and hemangioblasts in the mesendoderm lineage.

The advantage of clustering over manual population extraction is that the resolution of the clustering can be chosen by setting $k$ (the number of clusters), and very finely grained populations can be extracted and characterized. Here, we characterized thirty clusters, at finer resolution than the manual analysis above. However, a larger $k$ can be used for even finer exploration of the data. These clusters can then be characterized on the basis of gene expression for FACS sorting and further experimentation as we show in the next section.

ESC-specific transcripts **POU5F1**, **NANOG**, and **DPPA2/4** were highly expressed in cells located at the left-most part of sub-branch i, indicating that this is the starting point of the data (Figure 4.2.14B). As cells traveled along sub-branch i into sub-branch ii, the epiblast marker **OTX2** was upregulated and then downregulated, followed by a sharp increase in markers associated with primitive streak (**CER1**, **MIXL1**, **EOMES**, and **T**), indicating that mesoderm (ME) and endoderm (EN) differentiation begins in this region.
These markers continued to be highly expressed in cells located within the sub-region of branch v, which also exhibits high expression of the anterior EN markers FOXA2 and SOX17, while the posterior EN transcripts CDX2, NKX2-1, and KLF5 were expressed in the adjacent region of the same branch. ME transcripts NKX2-5, TNNT2, TAL1, TBX5, and TBX18 were specifically expressed in different sub-branches vi-x.

Further along sub-branch ii, past the ME/EN initiation region, the neuroectoderm (NE) markers PAX6, ZBTB16, SOX1, SOX2, NEUROG1, and DCX were expressed throughout branch iv, while branch iii was positive for the neural crest (NC) markers PAX3, FOXD3, SOX9, and SOX10. Figures 4.2.5E and 4.2.14C further show how the expression of these and other selected genes changes at different points within the branches. From these analyses, it is clear that PHATE successfully resolved all known germ layer branches based solely on the scRNA-seq data without any prior assumption about the structure of the data.

Furthermore, the PHATE embedding suggested three novel differentiation trajectories. Within the ectodermal lineage, a distinct $\text{GBX2}^+\text{ZIC2/5-HOXD1}^+\text{GLI3}^+$ NE progenitor gave rise to a bi-potent $\text{HOXA2}^+\text{HOXB1}^+$ precursor that separated into the NC branch and neural progenitor (NP) branch. The latter, in turn, split into at least five specialized neuronal subtypes. Within the EN branch, the canonical $\text{EOMES}^+\text{FOXA2}^+\text{SOX17}^+$ EN precursor was clustered together with the novel $\text{EOMES-FOXA2-GATA3}^+\text{SATB1}^+\text{KLF8}^+$ precursor, which further differentiated into cells expressing posterior EN markers NKX2-1, CDX2, ASCL2, and KLF5. Importantly, both the canonical and non-canonical EN cells expressed ARID3A, which can thus be defined as a pan-EN marker in humans. Moreover, we have identified a novel $\text{T}^+\text{GATA4}^+\text{CER1}^+\text{PROX1}^+$ cardiac precursor cell type within the ME lineage that gives rise to a $\text{TNNT2}^+$ cell via a $\text{GATA6-HAND1}^+$ differentiation intermediate.

4.E Supplementary videos

Video 1: Rotating video of 3D PHATE visualizations of the endoderm branch of the EB scRNA-seq data colored by the geometric mean of selected genes at each stage of the lineage specification tree in Figure 4.2.13B.

Video 2: Rotating video of 3D PHATE visualizations of the mesoderm branch of the EB scRNA-seq data colored by the geometric mean of selected genes at each stage of the lineage specification tree in Figure 4.2.13B.

Video 3: Rotating video of 3D PHATE visualizations of the neuroectoderm branches of the EB scRNA-seq data colored by the geometric mean of selected genes at each stage of the lineage specification tree in Figure 4.2.13B.
Video 4: Video showing the PHATE visualization (left) for the Frey Face dataset used in Roweis and Saul (Science, vol. 290, no. 5500, pp. 2323-2326, 2000) (right). PHATE reveals multiple branches in the data that correspond to different poses. Two of the branches are highlighted in this video. The corresponding point in the PHATE visualization is highlighted as the video progresses.

Video 5: Rotating video of 3D PHATE visualization of chromosome 1 in the Hi-C data from Darrow et al. (Proceedings of the National Academy of Sciences, p. 201609643, 2016) at 10 kb resolution. Multiple folds are clearly visible in the visualization.

Video 6: Rotating video of 3D PHATE visualization of all chromosomes in the Hi-C data from Darrow et al. (Proceedings of the National Academy of Sciences, p. 201609643, 2016) at 50 kb resolution. The embedding resembles the fractal globule structure proposed in Lieberman-Aiden et al. (Science, vol. 326, no. 5950, pp. 289-293, 2009).

Bibliography


Scott Gigante, Jay S Stanley, Ngan Vu, David van Dijk, Kevin R Moon, Guy Wolf, and Smita Krishnaswamy. Compressed diffusion. In *Sampling Theory and Applications*, pages 1–4. IEEE, July 2019. doi:10.1109/SampTA45681.2019.9030994 © 2019 IEEE. Reprinted with permission. In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Yale University’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications_](http://www.ieee.org/publications_).
standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.


Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to

John Aldo Lee and Michel Verleysen. Nonlinear dimensionality reduction of data manifolds with essential


Jacob H Levine, Erin F Simonds, Sean C Bendall, Kara L Davis, D Amir El-ad, Michelle D Tadmor,
Oren Litvin, Harris G Fienberg, Astraea Jager, Eli R Zunder, Rachel Finck, Amanda L. Gedman,
Ina Radtke, James R. Downing, Dana Pe’er, and Garry P. Nolan. Data-driven phenotypic dissec-
ton of AML reveals progenitor-like cells that correlate with prognosis. Cell, 162(1):184–197, 2015,
doi:10.1016/j.cell.2015.05.047.

Erez Lieberman-Aiden, Nynke L. van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes
Telling, Ido Amit, Bryan R. Lajoie, Peter J. Sabo, Michael O. Dorschner, Richard Sandstrom, Bradley
Bernstein, M. A. Bender, Mark Groudine, Andreas Guirke, John Stamatoyannopoulos, Leonid A. Mirny,
Eric S. Lander, and Job Dekker. Comprehensive mapping of long-range interactions reveals folding prin-

Innar Liiv. Seriation and matrix reordering methods: An historical overview. Statistical Analysis and Data

Ik Soo Lim, Pablo de Heras Ciechomski, Sofiane Sarni, and Daniel Thalmann. Planar arrangement of high-
dimensional biomedical data sets by ISOMAP coordinates. In 16th IEEE Symposium on Computer-Based

George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast
interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. Nature Methods, page 1,


Dimensionality reduction for data visualization need not be restricted to the domain of single-cell genomics; indeed, the techniques discussed in the preceding chapters can largely be applied to any form of high-dimensional data, so long as the data are drawn from a collection of low-dimensional manifolds and embedded in high-dimensional space.

One application area suffering a dearth of visualization techniques for data exploration and interpretation is deep learning. Deep learning describes the field of machine learning in which deep neural networks are trained to solve a variety of problems across classification, regression, data generation, and more (LeCun et al., 2015; Creswell et al., 2018). Despite the fact that deep learning attains state-of-the-art performance in many areas, our understanding of how and why these models perform as well as they do is lacking (Olah et al., 2018).

Many works seek to address the gap in our understanding of neural networks and their performance on various tasks, by analyzing the gradients of the objective function (Goodfellow and Vinyals, 2015; Li et al., 2018), the relationships between data points in the latent space defined by the hidden layers of the network (Hristov et al., 2018; Lucieri et al., 2020), or the hypothetical inputs which most strongly activate a specific hidden unit (Olah et al., 2017). However, all of these analyses rely on a pre-existing trained network.
On the other hand, the analysis of networks throughout the course of training is typically limited to examining
the decrease of the loss function over the course of training, or examining the network outputs (Salimans

Here, we seek to understand the evolution of the hidden layers of a neural network over time. To do
so, we extend PHATE to longitudinal data through a construction we call Multislice PHATE (M-PHATE).
M-PHATE allows us to understand the relationships between hidden units of a neural network not only at
the conclusion of network training, but also during training itself; this affords us greater understanding of
why certain networks outperform others on specific tasks, and gives insight into the mechanisms by which
this performance is achieved.

Contributions

As the sole first author of this work, I wrote the algorithm and performed all experiments. Analysis was
primarily completed in collaboration with Adam Charles and Gal Mishne.

5.1 Introduction

Despite their massive increase in popularity in recent years, deep networks are still regarded as opaque
and difficult to interpret or analyze. Understanding how and why certain neural networks perform better
than others remains an art. The design of neural networks and their training: choice of architectures,
regularization, activation functions, and hyperparameters, while informed by theory and prior work, is
often driven by intuition and tuned manually (Shahriari et al. 2016). The combination of these intuition-
driven selections and long training times even on high-performance hardware (e.g., 3 weeks on 8 GPUs for
the popular ResNet-200 network for image classification), means that the combinatorial task of testing all
possible choices is impossible, and must be guided by more principled evaluations and explorations.

A natural and widely used measure of evaluation for the difference between network architectures and
optimizers is the validation loss. In some situations, the validation loss lacks a clearly defined global meaning,
i.e., when the loss function itself is learned, and other evaluations are required (Salimans et al. 2016; Lucic
et al. 2018). While such scores are useful for ranking models on the basis of performance, they crucially
do not explain why one model outperforms another. To provide additional insight, visualization tools have
been employed, for example to analyze the “loss landscape” of a network. Specifically, these visualizations
depict how architectural choices modify the smoothness of local minima (Goodfellow and Vinyals 2015; Li
et al. 2018) — a quality assumed to be related to generalization abilities.
Local minima smoothness, however, is only one possible correlate of performance. Another internal quality that can be quantified is the hidden representations of inputs provided by the hidden unit activations. The multi-layered hidden representations of data are, in effect, the single most important feature distinguishing neural networks from classical machine learning techniques in generalization (LeCun et al., 2015; Bengio et al., 2005; Bengio, 2009; Montúfar and Morton, 2015; Montúfar et al., 2014). We can view the changes in representation by stochastic gradient descent as a dynamical system evolving from its random initialization to a converged low-energy state. Observing the progression of this dynamical system gives more insight into the learning process than simply observing it at a single point in time (e.g., after convergence.) In this paper, we contribute a novel method of inspecting a neural network’s learning: we visualize the evolution of the network’s hidden representation during training to isolate key qualities predictive of improved network performance.

Analyzing extremely high-dimensional objects such as deep neural networks requires methods that can reduce these large structures into more manageable representations that are efficient to manipulate and visualize. Dimensionality reduction is a class of machine learning techniques which aim to reduce the number of variables under consideration in high-dimensional data while maintaining the structure of a dataset. There exist a wide array of dimensionality reduction techniques designed specifically for visualization, which aim specifically to capture the structure of a dataset in two or three dimensions for the purposes of human interpretation, e.g., MDS (Cox and Cox, 2008), t-SNE (Maaten and Hinton, 2008), and Isomap (Tenenbaum et al., 2000). In this paper, we employ PHATE (Moon et al., 2019), a kernel-based dimensionality reduction method designed specifically for visualization which uses multidimensional scaling (MDS) (Cox and Cox, 2008) to effectively embed the diffusion geometry (Coifman and Lafon, 2006) of a dataset in two or three dimensions.

In order to visualize the evolution of the network’s hidden representation, we take advantage of the longitudinal nature of the data; we have in effect many observations of an evolving dynamical system, which lends itself well to building a graph from the data connecting observations across different points in time. We construct a weighted multislice graph (where a “slice” refers to the network state at a fixed point in time) by creating connections between hidden representations obtained from a single unit across multiple epochs, and from multiple units within the same epoch. A pairwise affinity kernel on this graph reflects the similarity between hidden units and their evolution over time. This kernel is then dimensionality reduced with PHATE and visualized in two dimensions.

The main contributions of this paper are as follows. We present Multislice PHATE (M-PHATE), which combines a novel multislice kernel construction with the PHATE visualization (Moon et al., 2019). Our kernel captures the dynamics of an evolving graph structure, that when visualized, gives unique intuition about
CHAPTER 5. VISUALIZING THE PHATE OF NEURAL NETWORKS

the evolution of a neural network over the course of training and re-training. We compare M-PHATE to other
dimensionality reduction techniques, showing that the combined construction of the multislice kernel and the
use of PHATE provide significant improvements to visualization. In two vignettes, we demonstrate the use
M-PHATE on established training tasks and learning methods in continual learning, and in regularization
techniques commonly used to improve generalization performance. These examples draw insight into the
reasons certain methods and architectures outperform others, and demonstrate how visualizing the hidden
units of a network with M-PHATE provides additional information to a deep learning practitioner over
classical metrics such as validation loss and accuracy, all without the need to access validation data.

5.2 Background

Diffusion maps (DMs) (Coifman and Lafon, 2006) is an important nonlinear dimensionality reduction method
that has been used to extract complex relationships between high-dimensional data (He et al., 2009; Farbman
We briefly review the two approaches.

Given a high-dimensional dataset \( \{ x_i \} \), DMs operate on a pairwise similarity matrix \( W \) (e.g., computed
via a Gaussian kernel \( W(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\epsilon) \)). and return an embedding of the data in a low-
dimensional Euclidean space. To compute this embedding, the rows of \( W \) are normalized by \( P = D^{-1} W \),
where \( D_{ii} = \sum_j W_{ij} \). The resulting matrix \( P \) can be interpreted as the transition matrix of a Markov chain
over the dataset and powers of the matrix, \( P^t \), represents running the Markov chain forward \( t \) steps. The
matrix \( P \) thus has a complete sequence of bi-orthogonal left and right eigenvectors \( \phi_i, \psi_i \), respectively, and
a corresponding sequence of eigenvalues \( 1 = \lambda_0 \geq |\lambda_1| \geq |\lambda_2| \geq \ldots \). Due to the fast spectrum decay of
\( \{ \lambda_i \} \), we can obtain a low-dimensional representation of the data using only the top \( \ell \) eigenvectors. Diffusion
maps, defined as \( \Psi_t(x) = (\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), \ldots, \lambda_\ell^t \psi_\ell(x)) \), embeds the data points into a Euclidean space
\( \mathbb{R}^\ell \) where the Euclidean distance approximates the diffusion distance:

\[
D_t^2(x_i, x_j) = \sum_{x_k} (p_t(x_i, x_k) - p_t(x_j, x_k))^2 \phi_0(x_k) \approx \| \Psi_t(x_i) - \Psi_t(x_j) \|_2^2
\]

Note that \( \psi_0 \) is neglected because it is a constant vector.

To enable successful data visualization, a method must reduce the dimensionality to two or three dimen-
sions; diffusion maps, however, reduces only to the intrinsic dimensionality of the data, which may be much
higher. Thus, to calculate a 2D or 3D representation of the data, PHATE applies MDS (Cox and Cox, 2008)
to the informational distance between rows $i$ and $j$ of the diffusion kernel $P^t$ defined as

$$\Phi_t(i, j) = \| \log P^t(i) - \log P^t(j) \|_2$$

where $t$ is selected automatically as the knee point of the Von Neumann Entropy of the diffusion operator. For further details, see Moon et al. (2019).

### 5.2.1 Related work

We consider the evolving state of a neural network’s hidden units as a dynamical system which can be represented as a multislice graph on which we construct a pairwise affinity kernel. Such a kernel considers both similarities between hidden units in the same epoch or time-slice (denoted intraslice similarities) and similarities of a hidden unit to itself across different time-slices (denoted interslice similarities). The concept of constructing a graph for data changing over time is motivated by prior work both in harmonic analysis (Coifman and Hirn, 2014; Lindenbaum et al., 2020; Lederman and Talmon, 2018; Marshall and Hirn, 2018; Banisch and Kolta, 2017) and network science (Mucha et al., 2010). For example, Coifman and Hirn (2014) suggest an algorithm for jointly analyzing DMs built over data points that are changing over time by aligning the separately constructed DMs, while Mucha et al. (2010) suggest an algorithm for community detection in multislice networks by connecting each node in one network slice to itself in other slices, with identical fixed weights for all intraslice connections. In both cases, such techniques are designed to detect changes in intraslice dynamics over time, yet interslice dynamics are not incorporated into the model.

### 5.3 Multiscale PHATE

#### 5.3.1 Preliminaries

Let $F$ be a neural network with a total of $m$ hidden units applied to $d$-dimensional input data. Let $F_i : \mathbb{R}^d \rightarrow \mathbb{R}$ be the activation of the $i$th hidden unit of $F$, and $F^{(r)}$ be the representation of the network after being trained for $r \in \{1, \ldots, n\}$ epochs on training data $X$ sampled from a dataset $\mathcal{X}$.

A natural feature space for the hidden units of $F$ is the activations of the units with respect to the input data. Let $Y \subset \mathcal{X}$ be a representative sample of $p \ll |X|$ points. (In this paper, we use points not used in training; however, this is not necessary. Further discussion of this is given in Section 5.B.) Let $Y_k$ be the $k$th sample in $Y$. We use the hidden unit activations $F(Y)$ to compute a shared feature space of dimension $p$ for the hidden units. We can then calculate similarities between units from all layers. Note that one may
instead consider the hidden units’ learned parameters (e.g. weight matrices and bias terms); however, these are not suitable for our purposes as they are not necessarily the same shape between hidden layers, and additionally the parameters may contain information not relevant to the data (for example, in dimensions of $\mathcal{X}$ containing no relevant information.)

We denote the time trace $T$ of the network as a $n \times m \times p$ tensor containing the activations at each epoch $\tau$ of each hidden unit $F_i$ with respect to each sample $Y_k \in Y$. We note that in practice, the major driver of variation in $T$ is the bias term contributing a fixed value to the activation of each hidden unit. Further, we note that the absolute values of the differences in activation of a hidden unit are not strictly meaningful, since any differences in activation can simply be magnified by a larger kernel weight in the following layer. Therefore, to calculate more meaningful similarities, we first $z$-score the activations of each hidden unit at each epoch $\tau$

$$T(\tau, i, k) = \frac{F_i^{(\tau)}(Y_k) - \frac{1}{p} \sum_{k} F_i^{(\tau)}(Y_k)}{\sqrt{\text{Var}_\tau F_i^{(\tau)}(Y_k)}}.$$

### 5.3.2 Multislice Kernel

The time trace gives us a natural substrate from which to construct a visualization of the network’s evolution. We construct a kernel over $T$ utilizing our prior knowledge of the temporal aspect of $T$ to capture its dynamics. Let $K$ be a $nm \times nm$ kernel matrix between all hidden units at all epochs (the $(\tau m + j)$th row or column of $K$ refers to $j$-th unit at epoch $\tau$). We henceforth refer to the $(\tau m + j)$th row of $K$ as $K((\tau, j), :)$ and the $(\tau m + j)$th column of $K$ as $K(:, (\tau, j))$.

To capture both the evolution of a hidden unit throughout training as well as its community structure with respect to other hidden units, we construct a multislice kernel matrix which reflects both affinities between hidden units $i$ and $j$ in the same epoch $\tau$, or intraslice affinities

$$K^{(\tau)}_{\text{intraslice}}(i, j) = \exp\left(-\|T(\tau, i) - T(\tau, j)\|_2^2 / \sigma_{(\tau, i)}^2\right)$$

as well as affinities between a hidden unit $i$ and itself at different epochs, or interslice affinities

$$K^{(i)}_{\text{interslice}}(\tau, \upsilon) = \exp\left(-\|T(\tau, i) - T(\upsilon, i)\|_2^2 / \epsilon^2\right)$$

where $\sigma_{(\tau, i)}$ is the intraslice bandwidth for unit $i$ at epoch $\tau$, $\epsilon$ is the fixed intraslice bandwidth, and $\alpha$ is the adaptive bandwidth decay parameter.

In order to maintain connectivity while increasing robustness to parameter selection for the intraslice
affinities $K_{\text{intraslice}}^{(\tau)}$, we use an adaptive-bandwidth Gaussian kernel (termed the alpha-decay kernel (Moon et al., 2019)), with bandwidth $\sigma_{(\tau,i)}$ set to be the distance of unit $i$ at epoch $\tau$ to its $k$th nearest neighbor across units at that epoch: $\sigma_{(\tau,i)} = d_k(T(\tau,i), T(\tau,:))$, where $d_k(x,X)$ denotes the $L_2$ distance from $x$ to its $k$th nearest neighbor in $X$. Note that the use of the adaptive bandwidth means that the kernel is not symmetric and will require symmetrization. In order to allow the kernel to represent changing dynamics of units over the course of learning, we use a fixed-bandwidth Gaussian kernel in the interslice affinities $K_{\text{interslice}}^{(i)}$, where $\epsilon$ is the average across all epochs and all units of the distance of unit $i$ at epoch $\tau$ to its $\kappa$th nearest neighbor among the set consisting of the same unit $i$ at all other epochs $\epsilon = \frac{1}{nm} \sum_{\tau=1}^{n} \sum_{i=1}^{m} d_\kappa(T(\tau,i), T(:,i))$.

Finally, the multislice kernel matrix contains one row and column for each unit at each epoch, such that the intraslice affinities form a block diagonal matrix and the interslice affinities form off-diagonal blocks composed of diagonal matrices (see Figures 5.A.1 and 5.A.2 for a diagram):

$$K((\tau,i),(\nu,j)) = \begin{cases} K_{\text{intraslice}}^{(\tau)}(i,j), & \text{if } \tau = \nu; \\ K_{\text{intraslice}}^{(i)}(\tau,\nu), & \text{if } i = j; \\ 0, & \text{otherwise}. \end{cases}$$

We symmetrize this kernel as $K' = \frac{1}{2}(K + K^T)$, and row normalize it to obtain $P = D^{-1}K$, which represents a random walk over all units across all epochs, where propagating from $(\tau,i)$ to $(\nu,j)$ is conditional on the transition probabilities between epochs $\tau$ and $\nu$. PHATE (Moon et al., 2019) is applied to $P$ to visualize the time trace $T$ in two or three dimensions.

5.4 Results

5.4.1 Example visualization

To demonstrate our visualization, we train a feedforward neural network with 3 layers of 64 hidden units to classify digits in MNIST (LeCun et al., 1998). The visualization is built on the time trace $T$ evaluated on the network over a single round of training that lasted 300 epochs and reached 96% validation accuracy.

We visualize the network using M-PHATE (Fig. 5.4.1) colored by epoch, hidden layer and the digit for which examples of that digit most strongly activate the hidden unit. The embedding is clearly organized longitudinally by epoch, with larger jumps between early epochs and gradually smaller steps as the network converges. Additionally, increased structure emerges in the latter epochs as the network learns meaningful representations of the digits, and groups of neurons activating on the same digits begin to co-localize. Neurons
CHAPTER 5. VISUALIZING THE PHATE OF NEURAL NETWORKS

Figure 5.4.1: Visualization of a simple 3-layer MLP trained on MNIST with M-PHATE. Visualization is colored by epoch (left), hidden layer (center), and most active digit for each unit (right).

of different layers frequently co-localize, showing that our visualization allows meaningful comparison of hidden units in different hidden layers.

5.4.2 Comparison to other visualization methods

To evaluate the quality of the M-PHATE visualization, we compare to three established visualization methods: diffusion maps, t-SNE and ISOMAP. We also compare our multislice kernel to the standard formalism of these visualization techniques, by computing pairwise distances or affinities between all units at all time points without taking into account the multislice nature of the data.

Figure 5.4.2: Comparison of standard application of visualization algorithms. Each point represents a hidden unit at a given epoch during training and is colored by the epoch.

Figure 5.4.2 shows the standard and multislice visualizations for all four dimensionality reduction tech-
Table 5.4.1: Neighborhood preservation of visualization methods applied to a FFNN classifying MNIST.

<table>
<thead>
<tr>
<th></th>
<th>Multislice</th>
<th></th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PHATE</td>
<td>DM</td>
<td>Isomap</td>
</tr>
<tr>
<td>Intraslice, (k = 10)</td>
<td>0.26</td>
<td>0.19</td>
<td>0.11</td>
</tr>
<tr>
<td>Interslice, (k = 10)</td>
<td>0.95</td>
<td>0.58</td>
<td>0.79</td>
</tr>
<tr>
<td>Intraslice, (k = 40)</td>
<td>\textbf{0.45}</td>
<td>0.36</td>
<td>0.25</td>
</tr>
<tr>
<td>Interslice, (k = 40)</td>
<td>0.93</td>
<td>0.75</td>
<td>0.78</td>
</tr>
<tr>
<td>Loss Correlation</td>
<td>\textbf{0.81}</td>
<td>0.61</td>
<td>0.61</td>
</tr>
</tbody>
</table>

Techniques of the network in Section 5.4.1 For implementation details, see Section 5.C Only the Multislice PHATE visualization reveals any meaningful evolution of the neural network over time. To quantify the quality of the visualization, we compare both interslice and intraslice neighborhoods in the embedding to the equivalent neighborhoods in the original data. Specifically, for a visualization \(V\) we define the intraslice neighborhood preservation of a point \(V(t, i) \in V\) as

\[
\frac{1}{|k|} \left| \mathcal{N}_V^k(V(t, i)) \cap \mathcal{N}_T^k(T(t, i)) \right|
\]

and the interslice neighborhood preservation of \(V(t, i)\) as

\[
\frac{1}{|k|} \left| \mathcal{N}_V^k(V(t, i)) \cap \mathcal{N}_T^k(T(t, i)) \right|
\]

where \(\mathcal{N}_X^k(x)\) denotes the \(k\) nearest neighbors of \(x\) in \(X\). We also calculate the Spearman correlation of the rate of change of each hidden unit with the rate of change of the validation loss to quantify the fidelity of the visualization to the diminishing rate of convergence towards the end of training.

M-PHATE achieves the best neighborhood preservation on all measures except the interslice neighborhood preservation, in which it performs on-par with standard t-SNE. Additionally, the multislice kernel construction outperforms the corresponding standard kernel construction for all methods and all measures, except again in the case of t-SNE for interslice neighborhood preservation. M-PHATE also has the highest correlation with change in loss, making it the most faithful display of network convergence.

5.4.3 Continual learning

An ongoing challenge in artificial intelligence is in making a single model perform well on many tasks independently. The capacity to succeed at dynamically changing tasks is often considered a hallmark of genuine intelligence, and is thus crucial to develop in artificial intelligence (Parisi et al., 2019). Continual learning is one attempt at achieving this goal sequentially training a single network on different tasks with the aim of instilling the network with new abilities as data becomes available.
To assess networks designed for continual learning tasks, a set of training baselines have been proposed. Hsuetal. (2018) define three types of continual learning scenarios for classification: incremental task learning, in which a separate binary output layer is used for each task; incremental domain learning, in which a single binary output layer performs all tasks; and incremental class learning, in which a single 10-unit output layer is used, with each pair of output units used for just a single task. Further details are given in Section 5.D.

We implemented a 2-layer MLP with 400 units in each hidden layer to perform incremental, domain and class learning tasks using three described baselines: standard training with Adagrad (Duchi et al., 2011) and Adam (Kingma and Ba, 2015), and an experience replay training scheme called Naive Rehearsal (Hsu et al., 2018) in which a small set of training examples from each task are retained and replayed to the network during subsequent tasks. Each network was trained for 4 epochs before switching to the next task. Overall, we find that validation performance is fairly consistent with results reported in Hsu et al. (2018), with Naive Rehearsal performing best, followed by Adagrad and Adam. Class learning was the most challenging, followed by domain learning and task learning.

Figure 5.4.3 shows M-PHATE visualizations of learning in networks trained in each of three baselines, with network slices taken every 50 batches rather than every epoch for increased resolution. Notably, we observe a stark difference in how structure is preserved over training between networks, which is predictive of task performance. The highest-performing networks all tend to preserve representational structure across changing tasks. On the other hand, networks trained with Adam — the worst performing combinations — tend to have a structural “collapse”, or rapid change in connectivity, as the tasks switch, consistent with the rapid change (and eventual increase) in validation loss.

Further, the frequency of neighborhood changes for hidden units throughout training (appearing as a crossing of unit trajectories in the visualization) corresponds to an increase in validation loss; this is due to a change in function of the hidden units, corrupting the intended use of such units for earlier tasks. We quantify this effect by calculating the Adjusted Rand Index (ARI, Santos and Embrechts, 2009) on cluster assignments computed on the subset of the visualization corresponding to the hidden units pre- and post-task switch, and find that the average ARI is strongly negatively correlated with the network’s final validation loss averaged over all tasks ($\rho = 0.94$). Results are similar for the same experiment run in CIFAR10 ($\rho = 0.86$, see Section 5.D).

Looking for such signatures, including rapid changes in hidden unit structure and crossing of unit trajectories, can thus be used to understand the efficiency of continual learning architectures.
CHAPTER 5. VISUALIZING THE PHATE OF NEURAL NETWORKS

Table 5.4.2: Adjusted Rand Index of cluster assignments computed on the subset of the PHATE visualization corresponding to the hidden units pre- and post-task switch. ARI is averaged across all four task switches, 6 different choices of clustering parameter (between 3–8 clusters) and 20 random seeds. Loss refers to average validation loss averaged over all tasks after completion of training.

<table>
<thead>
<tr>
<th>Task</th>
<th>Domain</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val. Loss</td>
<td>0.047</td>
<td>0.104</td>
</tr>
<tr>
<td>ARI</td>
<td>0.741</td>
<td>0.772</td>
</tr>
</tbody>
</table>

Table 5.4.3: Summed variance per epoch of the PHATE visualization is associated with the difference between a network that is memorizing and a network that is generalizing. Memorization error refers to the difference between train loss and validation loss.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Activity</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout</td>
<td>L1</td>
<td>L2</td>
</tr>
<tr>
<td>Memorization</td>
<td>-0.09</td>
<td>0.02</td>
</tr>
<tr>
<td>Variance</td>
<td>382</td>
<td>141</td>
</tr>
</tbody>
</table>

5.4.4 Generalization

Despite being massively overparametrized, neural networks frequently exhibit astounding generalization performance [Zhang et al. 2017; Allen-Zhu et al. 2018]. Recent work has showed that, despite having the capacity to memorize, neural networks tend to learn abstract, generalizable features rather than memorizing each example, and that this behaviour is qualitatively different in gradient descent compared to memorization [Arpit et al. 2017].

In order to demonstrate the difference between networks that learn to generalize and networks that learn to memorize, we train a 3-layer MLP with 128 hidden units in each layer to classify MNIST with: no regularization; L1/L2 weight regularization; L1/L2 activity regularization; and dropout. Additionally, we train the same network to classify MNIST with random labels, as well as to classify images with randomly valued pixels, such networks being examples of pure memorization. Each network was trained for 300 epochs, and the discrepancy between train and validation loss reported.

We note that in Figure 5.4.4, the networks with the poorest generalization (i.e. those with greatest divergence between train and validation loss), especially Activity L1 and Activity L2, display less heterogeneity in the visualization. To quantify this, we calculate the sum of the variance for all time slices of each embedding and regress this against the memorization error of each network, defined as the discrepancy between train and test loss after 300 epochs (Table 5.4.3), achieving a Spearman correlation of $\rho = -0.98$. Results are similar for the same experiment run in CIFAR10 ($\rho = -0.97$, see Section 5.E).

To understand this phenomenon, we consider the random labels network. In order to memorize random labels, the neural network must hone in on minute differences between images of the same true class in
order to classify them differently. Since most images won’t satisfy such specific criteria most nodes will not respond to any given image, leading to low activation heterogeneity and high similarities between hidden units. The M-PHATE visualization clearly exposes this intuition visually, depicting very little difference between these hidden units. Similar intuition can be drawn from the random pixels network, in which the difference between images is purely random. We hypothesize that applying $L_1$ or $L_2$ regularization over the activations has a qualitatively similar effect; reducing the variability in activations and effectively over-emphasizing small differences in the hidden representation. This behavior effectively mimics the effects of memorization.

On the other hand, we consider the dropout network, which displays the greatest heterogeneity. Initial intuition evoked the idea that dropout emulates an ensemble method within a single network; by randomly removing units from the network during training, the network learns to combine the output of many sub-networks, each of which is capable of correctly classifying the input [Srivastava et al. 2014]. M-PHATE visualization of training with dropout recommends a more mechanistic version of this intuition: dropped-out nodes are protected from receiving the exact same gradient signals and diverge to a more expressive representation. The resulting heterogeneity in the network reduces the reliance on small differences between training examples and heightens the network’s capacity to generalize. This intuition falls in line with other theoretical explorations, such as viewing dropout as a form of Bayesian regularization [Gal and Ghahramani 2016] or stochastic gradient descent [Baldi and Sadowski 2013] and reinforces our understanding of why dropout induces generalization.

We note that while this experiment uses validation data as input to M-PHATE, we have repeated this experiment in Section 5.B and show equivalent results. In doing so, we provide a mechanism to understand the generalization performance of a network without requiring access to validation data.

5.5 Conclusion

Here we have introduced a novel approach to examining the process of learning in deep neural networks through a visualization algorithm we call M-PHATE. M-PHATE takes advantage of the dynamic nature of the hidden unit activations over the course of training to provide an interpretable visualization otherwise unattainable with standard visualizations. We demonstrate M-PHATE with two vignettes in continual learning and generalization, drawing conclusions that are not apparent without such a visualization, and providing insight into the performance of networks without necessarily requiring access to validation data. In doing so, we demonstrate the utility of such a visualization to the deep learning practitioner.
Acknowledgments

This work was partially supported by the Gruber Foundation [S.G.]; the Chan-Zuckerberg Initiative (grant ID: 182702) and the National Institute of General Medical Sciences of the National Institutes of Health (grant ID: R01GM130847) [S.K.]; and the National Institute of Neurological Disorders and Stroke of the National Institutes of Health (grant ID: R01EB026936) [G.M.].
Figure 5.4.3: Visualization of a 2 layer MLP trained on Split MNIST for five-task continual learning of binary classification. Training loss and accuracy are reported on the current task. Validation loss and accuracy is reported on a test set consisting of an even number of samples from all tasks. Only 100 neurons are shown for clarity. Full plots are available in Section 5.D.
Figure 5.4.4: Visualization of a 3-layer MLP trained to classify MNIST with different regularizations or manipulations applied to affect generalization performance.
Appendix

5.A Multislice graph construction

Figure 5.A.1: Example schematic of the multislice graph used in M-PHATE. The intra- and interslice kernels represent the similarities between the graph nodes at different time-points, providing PHATE with a time-aware distance to visualize the data with.
In Section 5.3, we describe a multislice affinity kernel \( K \) built from an *intraslice* kernel, which connects hidden units in the same epoch, and an *interslice* kernel, which connects each hidden unit to itself at different epochs. We further clarify the intuition behind such an affinity kernel in two schematics.

Figure 5.A.1 displays a graph of 10 hidden units in a dynamically changing graph structure over the course of four time slices. Each hidden unit’s local neighborhood within its own time slice (its intraslice affinities) changes as the system evolves, with connectivity shown as black lines. Additionally, each hidden unit is connected to itself across different epochs, with strength of these interslice connections (shown as dotted lines) also dependent on similarities (rather than simply a fixed-weight connection).

Figure 5.A.2 displays the top left corner of an example of a multislice affinity kernel. The full multislice kernel \( \langle K(\tau,i), (\tau,j) \rangle \), left) is composed on the intraslice kernels placed down the block diagonal \( K_{\text{intraslice}}^{(i,j)}, \ldots, K_{\text{intraslice}}^{(\tau,i)} \), middle) and the interslice kernels forming the diagonals of each off-diagonal block \( K_{\text{interslice}}^{(\tau,i)}, \ldots, K_{\text{interslice}}^{(j,i)} \), right).

Figure 5.A.2: Example schematic of the multislice kernel used in M-PHATE. This kernel is a sum of intaslice and interslice affinities.

### 5.B Selection of representative subset \( Y \)

In Section 5.3, we state that the representative subset \( Y \) is taken from points not used in training. However, there is no reason why this should be the case. To demonstrate that M-PHATE can be used successfully without accessing data external to the training set, we show in Figure 5.B.1 a repetition of the generalization experiment, using only training data to build the visualization. Using the same quantification of variance and memorization as in Section 5.4.4, we obtain an equally strong correlation (Spearman’s \( \rho = -0.95 \), Table 5.D.1). Further, we note that the visualizations are qualitatively very similar to those obtained using training data, indicating that M-PHATE can be used to understand the generalization performance of a
network without having access to an external validation set.

Figure 5.B.1: Visualization of a 3-layer MLP trained to classify MNIST with different regularizations or manipulations applied to affect generalization performance, where the visualization is built using only training data.

5.C Parameters for visualization methods comparison

In Section 5.4.2, we compare M-PHATE to Diffusion Maps, t-SNE and Isomap in both a standard and multiscale context. Since t-SNE and Isomap require distance matrices, not affinity matrices, we convert the multislice kernel to geodesic distances by computing the shortest-path over the graph with the distance $D = -\log K'$. For standard application of Isomap and t-SNE, we use the default parameters in sklearn (Pedregosa et al., 2011). Since diffusion maps can be applied to any symmetric non-negative affinity kernel and does not have a reference implementation, we apply diffusion maps to the adaptive bandwidth kernel built in PHATE.

5.D Continual Learning

5.D.1 Continual Learning Schemes

Hsu et al. (2018) describe three schemes of continual learning commonly used in the literature.

Incremental task learning describes the process of learning shared hidden units for separated output layers.
Table 5.D.1: Summed variance per epoch of the PHATE visualization is associated with the difference between a network that is memorizing and a network that is generalizing, where the visualization is built using only training data. Memorization error refers to the difference between train loss and validation loss.

<table>
<thead>
<tr>
<th></th>
<th>Dropout</th>
<th>Kernel L1</th>
<th>Kernel L2</th>
<th>Activity L1</th>
<th>Activity L2</th>
<th>Random Labels</th>
<th>Random Pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memorization</td>
<td>-0.09</td>
<td>0.02</td>
<td>0.04</td>
<td>0.05</td>
<td>0.10</td>
<td>0.13</td>
<td>0.53</td>
</tr>
<tr>
<td>Variance</td>
<td>59</td>
<td>77</td>
<td>35</td>
<td>28</td>
<td>0.66</td>
<td>0.34</td>
<td>0.37</td>
</tr>
</tbody>
</table>

for each task; the output units for task $i$ are therefore protected from gradient signals during the training of task $j \neq i$. This is akin to the standard model of transfer learning, in which all but the final layer of a network are copied for a new task, with a fresh output layer attached for the new task.

Incremental domain learning describes the process of learning an entirely shared network which learns to perform all tasks separately, but with the same units; in this case the output units for task $i$ are the same units that are used in task $j$ and must learn to correctly classify training examples from separate tasks as though they were the same class.

Incremental class learning describes the process of learning an entirely shared network which learns to perform all tasks at once, with no knowledge of which task is currently being performed. The network contains separate output units for each task, but must select which output units to use, in contrast to incremental task learning in which the task is specified. This is by far the most difficult setting, since in training any one task, the optimal solution is to never predict the output classes of any other task; this strongly encourages catastrophic forgetting.

Figure 5.D.1 demonstrates these three architectures on Split MNIST.

5.D.2 Network Parameters

The networks in Section 5.4.3 are trained as follows. Input data is scaled from 0 to 1. All networks consist of a MLP with 2 layers of 400 units with ReLU activation, and a softmax classification output layer. All networks are trained with a batch size of 128, split to batches of 64 new data and 64 rehearsal data in the case of Naive Rehearsal. For the Adam optimizer, we use a learning rate of $1e^{-5}$. For the Adagrad optimizer, we use a learning rate of $1e^{-4}$. For Naive Rehearsal, we use the Adam optimizer. All networks are built and trained in Keras using a Tensorflow backend.

5.D.3 MNIST Full Results

Figure 5.4.3 shows the visualizations of the continual learning networks for a subset of 100 hidden units from each layer of the MLP with 2 layers of 400 units. Figures 5.D.2 and 5.D.3 show the full embedding of layers
CHAPTER 5. VISUALIZING THE PHATE OF NEURAL NETWORKS

Figure 5.D.1: Architectures for incremental learning scenarios. Reproduced with permission from Hsu et al. (2018).

1 and 2 respectively. In all cases, the visualizations are computed on all hidden units and subsampled for plotting purposes only.

We note the striking difference between layer 1 and layer 2 in all visualizations. In each case, there is a strong vertical pattern in layer 2, indicating that layer 2 is undergoing very large changes in hidden representation such that successive time-slices of the network are largely disconnected from one another. This can be most clearly seen in ADAM Incremental Class learning, in which the network appears to entirely forget the learned representations in layer 2, which is corroborated by the validation loss, which resets to the same point after each task. In comparison, the Naive Rehearsal visualizations remain connected at each task switch, which is consistent with the improved capacity of the network to retain the performance achieved on previous tasks.

5.D.4 CIFAR10 Results

To show that the results shown above generalize beyond one specific dataset, we repeated the same experiment with CIFAR10. Since this continual learning task is substantially more difficult with CIFAR10 than with MNIST, we doubled the number of epochs per task to 8. The layer-wise M-PHATE embeddings of the learning process are shown in Figures 5.D.4 and 5.D.5. As with MNIST, we see that the simpler tasks retain
Figure 5.D.2: Visualization of layer 1 of a 2 layer MLP trained on Split MNIST for five-task continual learning of binary classification. Accuracy is reported on a test set consisting of an even number of samples from all tasks.
Figure 5.D.3: Visualization of layer 2 of a 2 layer MLP trained on Split MNIST for five-task continual learning of binary classification. Accuracy is reported on a test set consisting of an even number of samples from all tasks.
more structure in the visualization at the point of task switch. Additionally, the vertical patterning observed in layer 2 is seen here once again, indicating that this is a feature of the network, rather than the task.

Once again, we quantify the effect of structural collapse in the visualization by calculating the Adjusted Rand Index (ARI) on cluster assignments computed on the subset of the visualization corresponding to the hidden units pre- and post-task switch, and find that the average ARI is strongly negatively correlated with the network’s final validation loss averaged over all tasks ($\rho = 0.86$, Table 5.5.E.1), as it was with the MNIST experiment.

Figure 5.D.4: Visualization of layer 1 of a 2 layer MLP trained on Split CIFAR10 for five-task continual learning of binary classification. Training loss and accuracy are reported on the current task. Validation loss and accuracy is reported on a test set consisting of an even number of samples from all tasks.
Figure 5.D.5: Visualization of layer 2 of a 2 layer MLP trained on Split CIFAR10 for five-task continual learning of binary classification. Training loss and accuracy are reported on the current task. Validation loss and accuracy is reported on a test set consisting of an even number of samples from all tasks.

5.E Generalization

5.E.1 Network Parameters

The networks in Section 5.4.4 are trained as follows. Input data is scaled from 0 to 1. All networks consist of a MLP with 3 layers of 128 units with Leaky ReLU activation with $\alpha = 0.1$, and a softmax classification output layer. All networks are trained with a batch size of 256 with the Adam optimizer and a learning rate of $1e^{-5}$. All regularizations are applied with a weight of $1e^{-4}$. Dropout is applied with $p = 0.5$. For the random labels network, we randomly permute the output labels of the training data, leaving the validation data intact. For the random pixels network, we randomly assign all pixel values from a standard normal distribution. All networks are built and trained in Keras (Chollet et al., 2015) using a Tensorflow (Abadi...
Table 5.D.2: Adjusted Rand Index of cluster assignments computed on the subset of the PHATE visualization corresponding to the hidden units pre- and post-task switch on networks trained on Split CIFAR10 for 8 epochs on each task. ARI is averaged across all four task switches, 6 different choices of clustering parameter (between 3–8 clusters) and 20 random seeds. Loss refers to average validation loss averaged over all tasks after completion of training.

<table>
<thead>
<tr>
<th>Task Domain</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val. Loss</td>
<td>0.483</td>
</tr>
<tr>
<td>ARI</td>
<td>0.478</td>
</tr>
</tbody>
</table>

5.E.2 CIFAR10 Results

To show that the results shown above generalize beyond one specific dataset, we repeated the same experiment with CIFAR10 (Figure 5.E.1). Once again, there is a strong association between entropy of the visualization and the final difference between training loss and validation loss ($\rho = -0.97$, Table 5.E.1), indicating that this representation of network generalization performance is a feature of the network, rather than the specific dataset used.

Figure 5.E.1: Visualization of a 3-layer MLP trained to classify CIFAR10 with different regularizations or manipulations applied to affect generalization performance.
Table 5.E.1: Summed variance per epoch of the PHATE visualization is associated with the difference between a network that is memorizing and a network that is generalizing when trained on CIFAR10. Memorization error refers to the difference between train loss and validation loss.

<table>
<thead>
<tr>
<th></th>
<th>Dropout</th>
<th>L1</th>
<th>L2</th>
<th>Vanilla</th>
<th>L1</th>
<th>L2</th>
<th>Labels</th>
<th>Pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memorization</td>
<td>-0.12</td>
<td>0.12</td>
<td>0.15</td>
<td>0.15</td>
<td>0.38</td>
<td>0.46</td>
<td>0.23</td>
<td>8.71</td>
</tr>
<tr>
<td>Variance</td>
<td>74</td>
<td>61</td>
<td>42</td>
<td>28</td>
<td>0.45</td>
<td>0.18</td>
<td>0.41</td>
<td>0.02</td>
</tr>
</tbody>
</table>

5.F M-PHATE parameters

All multislice graphs are built with \( k = 2, \alpha = 5 \) and \( \kappa = 25 \). We apply PHATE on the multislice affinity matrix with PHATE parameters \( \gamma = 0 \) and \( n_{\text{landmark}} = 3000 \), and use the automatically selected parameter of \( t \) provided by the PHATE algorithm.

5.G Computing infrastructure

All computation was done on a single 36-core workstation running Arch Linux with a NVIDIA TITAN X graphics card and 512GB of RAM.

5.H Validation data selection

Training and validation data were separated into pre-defined groups as given in Keras (Chollet et al., 2015).

Bibliography


François Chollet et al. Keras. https://keras.io 2015.


Chapter 6

Harmonic Alignment


Due to the complex nature of the biological experiments often performed in single-cell genomics, datasets are often composed of the outcomes of multiple biological assays. These assays may consist of multiple runs of the same assay on different samples, or runs of multiple different assays. In the former case, the phenomenon of systematic bias between runs is known as a batch effect; this is typically regarded as unwanted technical variation and as such is frequently removed; however, the removal of these effects can often also lead to the accidental removal of the true biological variation between samples, i.e. the very target of the study. In the latter case, multiple assays can be used to collect information from different parts of the cell, for example, single-cell RNA sequencing measures changes in gene expression, while single-cell ATAC sequencing measures changes in chromatin accessibility. In order to analyze datasets of multiple modes, it is first necessary to align the modalities such that the features of one modality can be compared to the features of the other.

Both of these analysis goals can fit neatly into the category of manifold alignment. Initially, we have two datasets forming samples from a shared manifold $\mathcal{M}$, but drawn with different maps $f_i : \mathcal{M} \to \mathbb{R}^{d_i}$. Should we wish to visualize the manifold $\mathcal{M}$ using data from multiple datasets sampled in this way, we must first model the manifolds separately, and then align them before further analysis can take place. In this chapter, we show how the diffusion operator used in the preceding work can be used as a basis for manifold alignment, specifically in the case where we assume that an unknown subset of the data features correspond

*: Equal contribution. †: Equal contribution.
between the two manifolds in question. The fundamental assumptions driving this work are twofold; first, that the manifolds estimated from the data, as visualized using PHATE ([Moon et al. 2019](#)), appear to be similar to one another up to an isometry when the datasets are drawn from systems with similar biological characteristics; and second, that we can use the relationships between the graph eigenvectors and the data features to characterize this isometry, and hence to build a map between the datasets.

**Contributions**

As a co-first author of this work, my primary contribution was the conception and evaluation of the multimodal data fusion experiment, in addition to assistance with writing and re-evaluation of the original experiments in response to reviewer comments, and implementation of the software in Python.

### 6.1 Introduction

High dimensional data have become increasingly common in many fields of science and technology, and with them the need for robust representations of intrinsic structure and geometry in data, typically inferred via manifold learning methods. Furthermore, modern data collection technologies often produce multisample data, which contain multiple datasets (or data batches) that aim to capture the same phenomena but originate from different equipment, different calibration, or different experimental environments. These introduce new challenges in manifold learning, as naïve treatment in such cases produces data geometry that largely separates different data batches into separate manifolds. Therefore, special processing is required to align and integrate the data manifolds in such cases in order to allow for the study and exploration of relations between and across multiple datasets.

As a particular application field, we focus here on single cell data analysis, which has gained importance with the advent of new sequencing technologies, such as scRNA-seq and scATAC-seq. While numerous works have shown that manifold learning approaches are particularly effective on such data ([Moon et al. 2018](#)), a common challenge in their analysis is a set of technical artifacts termed *batch effects* (caused by data collection from separate experimental runs) that tend to dominate downstream analysis, unless explicitly corrected. For instance, it is often the case that naïve data manifold construction groups such data into clusters that correspond to measurement time or equipment used, rather than by meaningful biological variations. Under such circumstances, it is necessary that batch artifacts be eliminated while actual biological differences between the samples be retained. Further, it can also be the case that different variables of data are measured on the same biological system. For example, cells from the same tissue can be measured with transcriptomic and proteomic technologies. However, the cells themselves are destroyed
in each measurement, even though they are sampled from the same underlying cellular manifold. Therefore, there is no correspondence that can be established between the two sets of measurements directly.

Recent manifold learning methods focusing on multisample data often treat each dataset as a different “view” of the same system or latent manifold, and construct a multiview geometry (e.g., based on the popular diffusion maps framework from Coifman and Lafon (2006)) to represent them (e.g., Ham et al. (2005); Wang and Mahadevan (2008); Coifman and Hirn (2014); Tuia and Camps-Valls (2016); Lederman and Talmon (2018); Boumal et al. (2018)). Importantly, these methods often require at least partial, if not full, bijection between views (e.g., both sets of measurements conducted on the same cells), which is often impossible to obtain in experimental scenarios where data is collected asynchronously or independently. In particular, as mentioned above, genomic and proteomic data (especially at the single-cell resolution) often originate on destructive collection technologies, and thus data point (i.e., cell) correspondence becomes an impractical (if not impossible) assumption to impose on their analysis. Other works attempt to directly match data points, either in the ambient space Haghverdi et al. (2018); Amodio and Krishnaswamy (2018) or by local data geometry Wang and Mahadevan (2009). These approaches can be very sensitive to differences in sampling density rather than data geometry, as discussed in §6.2-6.3. Furthermore, a complete matching is often not feasible as certain datasets may contain distinct local phenomena (e.g., rare subpopulation only captured in one dataset but not present in the other).

In this paper, we formulate the processing of multisample data with no data point correspondence in terms of manifold alignment, and present an approach towards such alignment by bridging the geometric-harmonic framework provided by diffusion geometry Coifman and Lafon (2006) (§6.3.1) together with data feature filtering enabled by graph signal processing Shuman et al. (2013) (§6.3.2). Our alignment approach relies on correspondence between underlying features quantified by data collection or measurement systems, phrased here as feature correspondence, which is often more realistic that data point correspondence. Indeed, related systems often observe similar “entities” (e.g., cells, patients) and aim to capture related properties in them. As explained in §6.2 and §6.3.2 we treat measured data features as manifold signals (i.e., over the data manifold) and relate them to intrinsic coordinates of a diffusion geometry Coifman and Lafon (2006) of each dataset, which also serve as intrinsic data harmonics. Then, as explained in §6.4 we leverage feature correspondence to capture pairwise relations between the intrinsic diffusion coordinates of the separate data manifolds (i.e., of each dataset). Finally, we use these relations to compute an isometric transformation that aligns the data manifolds on top of each other without distorting their internal structure.

We demonstrate the results of our method in §6.5 on artificial manifolds and single-cell biological data for both batch effect removal and multimodal data fusion. In each case, our method successfully aligns data manifolds such that they have appropriate neighbors both within and across the two datasets. Further,
we show an application of our approach in transfer learning by applying a k-NN classifier to one unlabeled dataset based on labels provided by another dataset (with batch effects between them), and compare the classification accuracy before and after alignment. Finally, comparisons with recently developed methods such as the MNN-based method from Haghverdi et al. (2018) show significant improvements in performance and denoising by our harmonic alignment methods.

6.2 Problem setup

Let \( X = \{x_1, \ldots, x_M\} \) and \( Y = \{y_1, \ldots, y_N\} \) be two finite datasets that aim to measure the same phenomena. For simplicity, we assume that both datasets have the same number of features, i.e., \( X, Y \subseteq \mathbb{R}^n \) for some sufficiently high dimension \( n \). We consider here a setting where these datasets are collected via different instruments or environment, but are expected to capture some equivalent information which can be used to align the two datasets. To leverage a manifold learning approach in such settings, we consider the common latent geometry of the data as an unknown manifold \( \mathcal{M} \), which is mapped to the two feature spaces via functions \( f, g : \mathcal{M} \rightarrow \mathbb{R}^n \) that represent the two data spaces. Namely, each data point \( x \in X \) is considered as a result \( x = f(z) = (f_1(z), \ldots, f_n(z)) \in \mathbb{R}^n \) for some \( z \in \mathcal{M} \) and similarly each \( y \in Y \) as a result of \( y = g(z) = (g_1(z), \ldots, g_n(z)) \in \mathbb{R}^n \). Therefore, we aim to provide a common data representation of both datasets, which captures the geometry of \( \mathcal{M} \) while allowing data fusion of \( X, Y \) and integrated processing or analysis of their data features.

We note that while we clearly do not have access to the points \( z \in \mathcal{M} \) on the underlying manifold, we do have access to a finite sampling of the feature functions \( f_s, g_s : \mathcal{M} \rightarrow \mathbb{R} \), \( s = 1, \ldots, n \), by considering \( X, Y \) as points-by-features matrices (i.e., rewriting them, by slight abuse of notation, as \( X \in \mathbb{R}^{M \times n} \) and \( Y \in \mathbb{R}^{N \times n} \), rather than finite subsets of \( \mathbb{R}^n \)) and taking their corresponding columns. Further, previous multiview manifold learning methods typically consider aligned datasets, i.e., assuming that all feature functions are sampled over the same manifold points. Instead, here we remove this assumption, thus allowing independently sampled datasets, and replace it with a feature correspondence assumption. Namely, we assume the feature functions \( f_s, g_s \) (for given \( 1 \leq s \leq n \)) aim to capture similar structures in the data and should therefore share some common information, although each may also contain sensor-specific or dataset-specific bias. While this is an informal notion, it fits well with many experimental data collection settings.
6.3 Preliminaries and background

6.3.1 Diffusion maps

To learn a manifold geometry from collected data we use the diffusion maps (DM) construction (Coifman and Lafon, 2006), which we briefly describe here for one of the data spaces \( X \), but is equivalently constructed on \( Y \). This construction starts by considering local similarities, which we quantify via an anisotropic kernel

\[
\mathcal{K}(x_i, x_j) = \frac{G(x_i, x_j)}{\|G(x_i, \cdot)\|_1 \|G(x_j, \cdot)\|_1},
\]

where \( G(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2/\sigma\right) \) is the Gaussian kernel with neighborhood radius \( \sigma > 0 \). As shown in Coifman and Lafon (2006), this kernel provides neighborhood construction that is robust to sampling density variations and enables separation of data geometry from its distribution. Next, the kernel \( \mathcal{K} \) is normalized to define transition probabilities

\[
p(x_i, y_j) = \mathcal{K}(x_i, x_j)/\|\mathcal{K}(x_i, \cdot)\|_1
\]

that define a Markovian diffusion process over the data. Finally, a DM is defined by organizing these probabilities in a row stochastic matrix \( P \) (typically referred to as the diffusion operator) as

\[
P_{ij} = p(x_i, x_j),
\]

and using its eigenvalues \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \) and (corresponding) eigenvectors \( \{\phi_j\}_{j=1}^N \) to map each \( x_i \in X \) to diffusion coordinates

\[
\Phi_t(x_i) = [\lambda_1^t \phi_1(x_i), \ldots, \lambda_N^t \phi_N(x_i)]^T.
\]

The parameter \( t \) in this construction represents a diffusion time or the number of transitions considered in the diffusion process. To simplify notations, we also use \( \Phi_t = \{\Phi_t(x_i) : x_i \in X\} \) to denote the DM of the entire dataset \( X \). We note that in general, as \( t \) increases, most of the eigenvalue weights \( \lambda_j^t, j = 1, \ldots, N \), become numerically negligible, and thus truncated DM coordinates (i.e., using only non-negligible weights) can be used for dimensionality reduction purposes, as discussed in Coifman and Lafon (2006).

6.3.2 Graph Fourier transform

A classic result in spectral graph theory (see, e.g., Brooks et al. (1994)) shows that the discrete Fourier basis (i.e., pure harmonics, such as sines and cosines, organized by their frequencies) can be derived as Laplacian eigenvectors of the ring graphs. This result was recently used in graph signal processing (Shuman et al., 2013) to define a graph Fourier transform (GFT) by treating eigenvectors of the graph Laplacian as generalized Fourier harmonics (i.e., intrinsic sines and cosines over a graph). Further, as discussed in Coifman and Lafon (2006; Nadler et al. (2006), diffusion coordinates are closely related to these Laplacian eigenvectors, and can essentially serve as geometric harmonics over data manifolds.

In our case, we regard the kernel \( \mathcal{K} \) from §6.3.1 as a weighted adjacency matrix of a graph whose vertices
are the data point in $X$. Then, the resulting normalized graph Laplacian is given by $L = I - D^{1/2}PD^{-1/2}$, where $D$ is a diagonal matrix with $D_{ii} = ||\mathcal{K}(x_i, \cdot)||_1$. Therefore, the eigenvectors of $L$ can be written as $\psi_j = D^{1/2}\phi_j$ with corresponding eigenvalues $\omega_j = 1 - \lambda_j$. The resulting GFT of a signal (or function) $f$ over $X$ can thus be written as $\hat{f}[j] = \langle f, \psi_j \rangle = \langle f, D^{1/2}\phi_j \rangle$. We note that here we treat either $\omega_j$ or $\lambda_j$ as providing a “frequency” organization of their corresponding eigenvectors $\psi_j$ or $\phi_j$ (treated as intrinsic harmonics). In the latter case, eigenvectors with higher eigenvalues correspond to lower frequencies on the data manifold, and vice versa. As noted before, the same construction of GFT here and DM in §6.3.1 can be equivalently constructed for $Y$ as well. This frequency-based organization of diffusion coordinates derived from $X$ and $Y$, and their treatment as geometric harmonics, will be leveraged in §6.4 to provide an isometric alignment between the intrinsic data manifolds represented by the DMs of the two datasets by also leveraging the (partial) feature correspondence assumption from §6.2.

6.3.3 Related work on manifold alignment

Algorithms for semi-supervised and unsupervised manifold alignment exist in classical statistics [Gower (1975); Thompson (1984)], deep learning [Zhu et al. (2017); Kim et al. (2017); Amodio and Krishnaswamy (2018)] and manifold learning [Haghverdi et al. (2018); Wang and Mahadevan (2008; 2009)]. As mentioned in §6.1 much work has been done on finding common manifolds between data based on known (partial) bijection between data points [Coifman and Hirn (2014); Tuia and Camps-Valls (2016); Lederman and Talmon (2018); Boumal et al. (2018)]. In a sense, these methods can be regarded as nonlinear successors of the classic canonical correlation analysis (CCA) [Thompson (1984)], in the same way as many manifold learning methods can be regarded as generalizing PCA. Indeed, similar to PCA, the CCA method finds a common linear projection, but on directions that maximize covariance or correlation (typically estimated empirically via known pointwise correspondence) between datasets rather than just variance within one of them. However, in this work we mainly focus on settings where no data point correspondence is available, and therefore we focus our discussion in this section on related work that operate in such settings.

One of the earliest attempts at manifold alignment (in particular, with no point correspondence), was presented in [Wang and Mahadevan (2009)], which proposes a linear method based on embedding a joint graph built over both datasets to preserve local structure in both manifolds. This method provides a mapping from both original features spaces to a new feature space defined by the joint graph, which is shared by both datasets with no assumption of feature correspondence. More recently, in biomedical data analysis, mutual nearest neighbors (MNN) batch correction [Haghverdi et al. (2018)] focuses on families of manifold deformations that are often encountered in biomedical data. There, locally linear manifold alignment is
provided by calculating a correction vector for each point in the data, as defined by the distances from the point to all points for which it is a mutual $k$-nearest neighbor. This correction vector is then smoothed by taking a weighted average over a Gaussian kernel.

Beyond manifold learning settings, deep learning methods have been proposed to provide alignment and transfer learning between datasets. For example, cycle GANs [Zhu et al. (2017)] are a class of deep neural network in which a generative adversarial network (GAN) is used to learn a nonlinear mapping from one domain to another, and then a second GAN is used to map back to the original domain. These networks are then optimized to (approximately) satisfy cycle consistency constraints such that the result of applying the full cycle to a data point reproduces the original point. MAGAN [Amadio and Krishnaswamy (2018)] is a particular cycle GAN that adds a supervised partial feature correspondence to enforce alignment of two data manifolds over the mapping provided by the trained network. However, this correspondence can be disturbed by noise or sparsity in the data.

Additionally, a similar problem exists in isometric shape matching, albeit limited to low dimensional data (i.e., shapes in at most three dimensions). For example, the method in [Ovsjanikov et al. (2012)] takes shapes with a known Laplace-Beltrami operator and aligns them using a representation of the corresponding eigenfunctions. This work was extended further in [Pokrass et al. (2016)] to settings where ambient functions are defined intrinsically by the shape in order to learn region-region correspondences from an unknown bijection. Recent work [Vestner et al. (2017)] has relaxed the requirement for shapes to be isometric and the need for prior knowledge of the Laplace-Beltrami operator, instead estimating the manifold with a kernel density estimate over the shape boundary. However, the application of these methods is limited to shapes, rather than a regime of point clouds as seen in high-dimensional data analysis.

In contrast, in this work we consider more general settings of aligning intrinsic data manifolds in arbitrary dimensions, while being robust to noise, data collection artifacts, and density variations. We provide a nonlinear method for aligning two datasets using their diffusion maps [Coifman and Lafon (2006)] under the assumption of a partial feature correspondence. Unlike MAGAN, we do not need to know in advance which features should correspond, and our results show that even with correspondence as low as 15% we achieve good alignment between data. Further, unlike shape matching methods, we are not limited to datasets describing the boundary of a shape or dominated by density distribution. Our formulation allows us to obtain more information from datasets with partial feature correspondence than methods that assume no correspondence, but without the burden of determining in advance which or how many features correspond. To evaluate our method, in §6.5 we focus on comparison with MAGAN, as a leading representative of deep learning approaches, and MNN, as a leading representative of manifold learning approaches. We note that to the best of our knowledge, the method in [Wang and Mahadevan (2009)] is not provided with standard
implementation, and our attempts at implementing the algorithm have significantly under performed other methods. For completeness, partial comparison to this method is demonstrated in Appendix 6.C.

### 6.4 Harmonic alignment

Given datasets $X, Y$, as described in §6.2, we aim to construct a unified DM over both of them, which represents the global intrinsic structure of their common manifold $\mathcal{M}$ while still retaining local differences between the datasets (e.g., due to distributional differences or local patterns only available in one of the dataset). As mentioned before, global shifts and batch effects often make direct construction of such DM (or even the construction of local neighborhood kernels) over the union of both datasets unreliable and impractical. Instead, we propose here to first construct two separate DMs $\Phi(X), \Phi(Y)$ (based on eigenpairs $(\lambda_i^{(X)}, \phi_i^{(X)}), i = 1, \ldots, M,$ and $(\lambda_j^{(Y)}, \phi_j^{(Y)}), j = 1, \ldots, N,$ correspondingly), which capture the intrinsic geometry of each dataset. We then align their coordinates via an orthogonal transformation that preserves the rigid structure of the data in each DM, which is computed by orthogonalizing a correlation matrix computed between the diffusion coordinates of the two DMs.

However, since the diffusion coordinates are associated with intrinsic notions of frequency on data manifolds (as explained in §6.3.2), there is no need to compute the correlation between every pair $\phi_i^{(X)}, \phi_j^{(Y)}$, $i = 1, \ldots, M$, $j = 1, \ldots, N$. Indeed, leveraging the interpretation of such coordinate functions as intrinsic diffusion harmonics, we can determine that they should not be aligned between the geometry of $X$ and $Y$ if their corresponding frequencies (i.e., captured via the eigenvalues $\lambda_i^{(X)}, \lambda_j^{(Y)}$) are sufficiently far from each other. Therefore, in §6.4.1 we describe the construction of a bandlimited correlation matrix, and then use it in §6.4.2 to align the two DMs. Since our alignment method is based on the treatment of DM coordinates as manifold harmonics, we call this method harmonic alignment.
6.4.1 Bandlimited correlation

In order to partition the diffusion harmonics into local frequency bands, we consider the following window
functions, which are inspired by the itersine filter bank construction Perraudin et al. (2018):

\[
    w_\xi(\lambda) = \begin{cases} 
    \sin\left(\frac{\pi}{2} \cos^2\left(\frac{\pi}{2}(\ell \lambda - \xi)\right)\right) & \frac{\xi-1}{\ell} \leq \lambda \leq \frac{\xi+1}{\ell} \\
    0 & \text{otherwise},
    \end{cases}
\]

where \( \xi = 0, \ldots, \ell \) and \( \ell \) considered as a meta-parameter of the construction. We note that experimental
evidence indicate that fine tuning \( \ell \) does not significantly affect alignment quality. Each window \( w_\xi(\cdot) \) is
supported on an interval of length \( 2\ell \) around \( \xi/\ell \), while decaying smoothly from \( w_\xi(\xi/\ell) = 1 \)
to zero. Two
consecutive windows (i.e., \( w_\xi(\cdot), w_{\xi+1}(\cdot) \)) share an overlap of half their support; otherwise (i.e., \( w_\xi(\cdot), w_\xi(\cdot) \)
with \( |\xi - \xi'| \geq 2 \)) they have disjoint supports. Finally, we recall the spectra (i.e., eigenvalues) of \( P^{(X)}, P^{(Y)} \)
are contained in the interval \([0, 1]\), which is entirely covered by \( \ell + 1 \) window functions \( w_\xi(\cdot), \xi = 0, \ldots, \ell \),
as illustrated in Fig. 6.4.1(c). Notice that only half the support of \( w_0(\cdot) \) and \( w_\ell(\cdot) \) are shown in there, since
half of their support is below zero or above one, correspondingly.

Using the soft partition defined by \( w_\xi(\cdot), \xi = 0, \ldots, \ell \), we now define bandlimiting weights

\[
    w_{ij}^{(X,Y)} = \sum_{\xi=1}^{\ell} w_\xi(\lambda_i^{(X)}) w_\xi(\lambda_j^{(Y)}),
\]

for \( i = 1, \ldots, M \) and \( j = 1, \ldots, M \), between diffusion harmonics of \( X \) and \( Y \). As shown in the following
lemma, whose proof appears in Appendix 6.1, these weights enable us to quantitatively identify diffusion
harmonics that correspond similar frequencies and ignore relations between ones that have significantly
different ones.

**Lemma 6.4.1.** The bandlimiting weights from (6.2) satisfy the following properties: \( w_{ij}^{(X,Y)} \) is continuous
and differentiable in \( \lambda_i^{(X)} \) and \( \lambda_j^{(Y)} \); if \( \lambda_i^{(X)} = \lambda_j^{(Y)} \) then \( w_{ij} = 1 \); if \( |\lambda_i^{(X)} - \lambda_j^{(Y)}| > \frac{2}{\ell} \) then \( w_{ij} = 0 \); and the
rate of change of \( w_{ij}^{(X,Y)} \) w.r.t. \( |\lambda_i^{(X)} - \lambda_j^{(Y)}| \) is bounded by \( O(\ell) \).

Next, we use the weights from (6.2) to construct a \( M \times N \) bandlimited correlation matrix \( C \) defined as

\[
    [C]_{ij} = w_{ij}^{(X,Y)} \text{corr}\left(\phi_i^{(X)}, \phi_j^{(Y)}\right)
\]

for \( i = 1, \ldots, M \) and \( j = 1, \ldots, N \), which only considers correlations between diffusion harmonics within
similar frequency bands.
Finally, for each $i, j$ with nonzero weight $w_{ij}^{(X,Y)}$, we now need to compute a correlation between the diffusion harmonics $\phi_i^{(X)}, \phi_j^{(Y)}$. If we had partial data point correspondence, as is assumed in many previous works (e.g., Coifman and Hirn [2014]; Lederman and Talmon [2018]), we could estimate such correlation directly from matching parts of the two datasets. However, in our case we do not assume any a priori matching between data points. Instead, we rely on the assumed feature correspondence and leverage the GFT from §6.3.2 to express the harmonics $\phi_i^{(X)}, \phi_j^{(Y)}$ in terms of the data features via their Fourier coefficients. Namely, we take the GFT of the data features $f_s, g_s, s = 1, \ldots, n$ (i.e., the “columns” of the points-by-features representation of $X, Y$ as $M \times n, N \times n$ data matrices, correspondingly), and use them to represent $\phi_i^{(X)}, \phi_j^{(Y)}$ by the $n$ dimensional vectors $\hat{x}_i = (\hat{f}_1[i], \ldots, \hat{f}_n[i])^T$ and $\hat{y}_j = (\hat{g}_1[j], \ldots, \hat{g}_n[j])^T$, correspondingly. Then, we compute a correlation between the harmonics $\phi_i^{(X)}, \phi_j^{(Y)}$ indirectly via a correlation between $\hat{x}_i, \hat{y}_j$. For simplicity, and by slight abuse of terminology, we use an inner product in lieu of the latter, to define

$$\text{corr}(\phi_i^{(X)}, \phi_j^{(Y)}) = \langle \hat{x}_i, \hat{y}_j \rangle.$$

Therefore, together with (6.3), our bandlimited correlation matrix is given by $[C]_{ij} = w_{ij}^{(X,Y)} \langle \hat{x}_i, \hat{y}_j \rangle = \sum_{\xi=1}^t w_{\xi}(\lambda_i^{(X)}) \hat{x}_i \cdot w_{\xi}(\lambda_j^{(Y)}) \hat{y}_j$.

### 6.4.2 Rigid alignment

Given the bandlimited correlation matrix $C$, we use its SVD given by $C = UV^T$ (e.g., as shown in Schönemann [1966]) that defines an isometric transformation between the diffusion maps of the two samples, which we refer to as harmonic alignment. Finally, we can now compute a unified diffusion map, which can be written in (block) matrix form as

$$\Phi_t^{(X,Y)} = \left[ \begin{array}{cc} \Phi_0^{(X)} & \Phi_0^{(X)} T \\ \Phi_0^{(Y)} T^T & \Phi_0^{(Y)} \end{array} \right] \left[ \begin{array}{cc} \Lambda^{(X)} & 0 \\ 0 & \Lambda^{(Y)} \end{array} \right]^t,$$

where $\Lambda^{(X)}, \Lambda^{(Y)}$ are diagonal matrices with the diffusion eigenvalues $\{\lambda_i^{(X)}\}_{i=1}^N, \{\lambda_j^{(Y)}\}_{j=1}^M$ (correspondingly) as their main diagonal, and $t$ is an integer diffusion time parameter as in §6.3.1. A summary of the described steps is presented in Appendix 6.A. While this construction is presented here in terms of two datasets for simplicity, it can naturally be generalized to multiple datasets by considering multiple blocks (rather than the two-by-two block structure in (6.4)), based on orthogonalizing pairwise bandlimited correlations between datasets. This generalization is discussed in detail in Appendix 6.A.2.

Finally, given aligned DMs in $\Phi_t^{(X,Y)}$, we can construct a new neighborhood kernel over their coordinates.
(i.e., in terms of a combined diffusion distance) and build a robust unified diffusion geometry over the entire
data in \( X \cup Y \) that is invariant to batch effects and also enables denoising of data collection artifacts
that depend on environment or technology rather than the underlying measured phenomena. This diffusion
geometry can naturally be incorporated in diffusion-based methods for several data processing tasks, such as
dimensionality reduction & visualization [Moon et al. (2019)], denoising & imputation [Van Dijk et al. (2018)],
latent variable inference [Lederman and Talmon (2018)], and data generation [Lindenbaum et al. (2018)]. In
particular, in §6.5.3 we demonstrate the application of harmonic alignment to batch effect removal and
multimodal data fusion with various single-cell genomic technologies.

6.5 Numerical results

6.5.1 Artificial feature corruption

To demonstrate the accuracy of harmonic alignment, we assess its ability to recover \( k \)-nearest neighborhoods
after random feature corruption, and compare it to MNN [Haghverdi et al. (2018)] and MAGAN [Amodio and
Krishnaswamy (2018)], which are leading manifold- and deep-learning methods respectively, as discussed in
§6.3.3. To this end, we drew two random samples \( X \) and \( Y \) of 1000 MNIST digit images, each of which is a
784-dimensional vector. For each trial, we generate a random orthogonal \( 784 \times 784 \) corruption matrix \( O_0 \). To
vary the amount of feature corruption, we produce partial corruption matrices \( O_p \) by randomly substituting
\( p \% \) of the columns in \( O_0 \) with columns of the identity matrix. Right multiplication of \( Y \) by these matrices
yields corrupted images with only \( p \% \) preserved pixels (Fig. 6.1(b), ‘Corrupted’).

To assess the alignment of the corrupted images \( YO_p \) to the uncorrupted images \( X \), we perform lazy
classification on digits (i.e., rows) in \( YO_p \) by using the labels of each aligned image’s \( k \) nearest neighbors in
\( X \). The results of this experiment, performed for \( p = \{0, 5, 10, \ldots, 95, 100\} \), are reported in Fig. 6.1(a). For
robustness, at each \( p \) we sampled three different non-overlapping pairs \( X, Y \), and for each pair we sampled
three random \( O_p \) matrices. It should be noted that while we report results in terms of mean classification
accuracy, we do not aim to provide an optimal classifier here. Our evaluation merely aims to provide a
quantitative assessment of neighborhood quality before and after alignment. We regard a lazy learner as
ideal for such evaluation since it directly exposes the quality of data neighborhoods, rather than obfuscate
it via a trained model. For comparison, results for harmonic alignment with a SVM classifier are shown in
§6.5.2 and Fig. 6.1(d).

In general, none of the methods recovers \( k \)-nearest neighborhoods under total corruption, showing 10%
accuracy for very small \( p \), essentially giving random chance accuracy. Note that this case clearly violates
our (partial) feature correspondence assumption. However, when using sufficiently many bandlimited filters, harmonic alignment quickly recovers over 80% accuracy and consistently outperforms both MNN and MA-
Figure 6.5.1: Recovery of k-nearest neighborhoods under feature corruption. Mean over 3 iterations is reported for each method. At each iteration, two sets $X$ and $Y$ of 1000 points were sampled from MNIST. $Y$ was then distorted by a $784 \times 784$ corruption matrix $O_p$ for various identity percentages $p$ (§6.5.1). Subsequently, a lazy classification scheme was used to classify points in $Y O_p$ using a 5-nearest neighbor vote from $X$. Results for harmonic alignment with $\ell \in \{2, 4, 8, 64\}$ (§6.4.1), mutual nearest neighbors (MNN), and classification without alignment are shown. (b) Reconstruction of digits with only 25% uncorrupted features. Left: Input digits. Left middle: 75% of the pixels in the input are corrupted. Right middle: Reconstruction without harmonic alignment. Right: Reconstruction after harmonic alignment. (c) Lazy classification accuracy relative to input size with unlabeled randomly corrupted digits with 35% preserved pixels. (d) Transfer learning performance. For each ratio, 1K uncorrupted, labeled digits were sampled from MNIST, and then 1K, 2K, 4K, and 8K (x-axis) unlabeled points were sampled and corrupted with 35% column identity.

GAN, except under under very high correspondence (i.e., when $O_p \approx I$). The method proposed by Wang and Mahadevan (2009) was excluded since it did not show improvement over unaligned classification, but is discussed in supplemental materials for completeness. We note that the performance of harmonic alignment is relatively invariant to the choice of $\ell$, with the exclusion of extremely high values. For the remainder of the experiments, we fix $\ell = 8$. All experiments use the default parameter $t = 1$.

Next, we examined the ability of harmonic alignment to reconstruct the corrupted data (Fig. 6.1(b)). We performed the same corruption procedure with $p = 25\%$ and selected one example of each MNIST digit. Ground truth from $Y$ and corrupted result $Y O_{25}$ are shown in Fig. 6.1(b). Then, reconstruction was performed by setting each pixel in a new image to the dominant class average of the 10 nearest neighbors from $X$. In the unaligned case, we see that most examples give smeared fives or ones; this is likely a random intersection formed by $X$ and $Y O_{25}$. On the other hand, reconstructions produced by harmonic alignment resemble the original input examples.

Finally, in Fig. 6.1(c) we consider the effect of data size on obtained alignment. To this end, we fix $p = 35\%$ and vary the size of the two aligned datasets. We compare harmonic alignment, MNN, and MAGAN on input sizes ranging from 200 to 1600 MNIST digits, while again using lazy classification accuracy to measure neighborhood preservation and quantify alignment quality. The results in Fig. 6.1(d) show that both MNN and MAGAN are not significantly affected by dataset size, and in particular do not improve with additional data. Harmonic alignment, on the other hand, not only outperforms them significantly – its alignment quality increases monotonically with input size.

6.5.2 Transfer learning

An interesting use of manifold alignment algorithms is transfer learning. In this setting, an algorithm is trained to perform well on a small (e.g., pilot) dataset, and the goal is to extend the algorithm to a new larger dataset (e.g., as more data is being collected) after alignment. In this experiment, we first randomly
selected 1,000 uncorrupted examples of MNIST digits, and constructed their DM to use as our training set. Next, we took 65%-corrupted unlabeled points (§6.5.1) in batches of 1,000, 2,000, 4,000, and 8,000, as a test set for classification using the labels from the uncorrupted examples. As shown in 6.1(d), with a 5-nearest neighbor lazy classifier, harmonic alignment consistently improves as the dataset gets larger, even with up to eight test samples for every one training sample. When the same experiment is performed with a linear SVM, harmonic alignment consistently outperforms other methods with performance being independent of test set size (or train-to-test ratio). This is due to the increased robustness and generalization capabilities of trained SVM. Further discussion of transfer learning is given in the supplementary materials. In addition to showing the use of manifold alignment in transfer learning, this example also demonstrates the robustness of our algorithm to imbalance between samples.

6.5.3 Biological data

Batch effect correction

![Image of biological data](image_url)

Figure 6.5.2: (a)-(c) Batch effect removal. 4K cells were subsampled from two single-cell mass cytometry immune profiles on blood samples of two patients infected with Dengue fever. Top: Both patients exhibit heightened IFNγ (x-axis), a pro-inflammatory cytokine associated with TNFα (y-axis) Bottom: IFNγ histograms for each batch. (a) Data before denoising. (b) Denoising of unaligned data enhances a technical effect between samples in IFNγ. (c) Harmonic alignment corrects the IFNγ shift. (d) Multimodal data fusion. Overlap of cell neighborhoods from joint gene expression and chromatin profiling of single cells. Harmonic alignment most accurately recovers the pointwise relationship between the manifolds.

To illustrate the need for robust manifold alignment in computational biology, we turn to a simple real-world example from Amodio et al. (2019) (Fig. 6.5.2). This dataset was collected by mass cytometry (CyTOF) of peripheral blood mononuclear cells from patients who contracted dengue fever Amodio et al. (2019).

The canonical response to dengue infection is upregulation of interferon gamma (IFNγ) Chakravarti and...
Kumaria (2006) During early immune response, IFN-γ works in tandem with acute phase cytokines such as tumor necrosis factor alpha (TNFα) to induce febrile response and inhibit viral replication Ohmori et al. (1997). We thus expect to see upregulation of these two cytokines together.

In Fig. 6.2(a) we show the relationship between IFN-γ and TNFα without denoising. Note that there is a substantial difference between the IFN-γ distributions of the two samples (Earth Mover’s Distance [EMD] = 2.699). In order to identify meaningful relationships in CyTOF data, it is common to denoise it first Moon et al. (2018). We used a graph low-pass filter proposed in Van Dijk et al. (2018) to denoise the cytokine data. The results of this denoising are shown in Fig. 6.2(b). This procedure introduced more technical artifacts by enhancing differences between batches, as seen by the increased EMD (3.127) between IFN-γ distributions of both patients. This is likely due to substantial connectivity differences between the two batch submanifold in combined data manifold.

Next, we performed harmonic alignment of the two patient profiles (Fig. 6.2(c)). Harmonic alignment corrected the difference between IFN-γ distributions and restored the canonical correlation of IFN-γ and TNFα (EMD=0.135). This example illustrates the utility of harmonic alignment for biological data, where it can be used for integrated analysis of data collected across different experiments, patients, and time points.

**Multimodal Data Fusion**

Since cells contain numerous types of components that are informative of their state (genes, proteins, epigenetics), modern experimental technologies are starting to measure of each of these components separately at the single cell level. Since most single-cell assays are destructive, it is challenging or impossible to obtain all desired measurements in the same cells. It is therefore desirable to perform each assay on a subset of cells from a single sample, and align these datasets *in silico* to obtain a pseudo-joint profile of the multiple data types.

To demonstrate the utility of harmonic alignment in this setting, we use a dataset obtained from Cao et al. (2018) of 11,296 cells from adult mouse kidney collected by a joint measurement technique named sci-CAR, which measures both gene expression (scRNA-seq) and chromatin accessibility (scATAC-seq) in the same cells simultaneously. The datasets are normalized separately as in Van Dijk et al. (2018), using a square root transformation for the scRNA-seq and a log transformation with a pseudocount of 1 for the scATAC-seq data, and finally the dimensionality of each dataset is reduced to 100 using truncated SVD. After randomly permuting the datasets to scramble the correspondence between them, we align the two manifolds in order to recover the known bijection between data modalities. Let \( f(i) \in F \) be the scRNA-seq measurement of cell \( i \), and \( g(i) \in G \) be the scATAC-seq measurement of cell \( i \). Fig. 6.2(d) shows the average percentage overlap of neighborhoods of \( f(i) \) in \( F \) with neighborhoods of \( g(i) \) in \( G \), before and after alignment with: MAGAN,
MNN and Harmonic Alignment. Harmonic Alignment most accurately recovers cell neighborhoods, thereby allowing the generation of *in silico* joint profiles across data types and obviating the need for expensive or infeasible *in vitro* joint profiling.

**Acknowledgments**

This work was partially funded by: the Gruber Foundation [S.G.]; IVADO (l’institut de valorisation des données) [G.W.]; Chan-Zuckerberg Initiative grants 182702 & CZF2019-002440 [S.K.]; and NIH grants R01GM135929 & R01GM130847 [G.W., S.K.].
Appendix

6.A Algorithm

6.A.1 Standard implementation

Consider the standard setting of harmonic alignment as described in Section 6.4. We provide here a detailed description of the Harmonic Alignment algorithm in pseudocode.

Let $X$ and $Y$ be collections of data points $\vec{x} \in \mathbb{R}^d$. Let the diffusion time $t$ and band count $\ell$ be positive integers. Let $\sigma^{(X)}, \sigma^{(Y)}$ be bandwidth functions (generally either positive constants or a function of the distance from a point to its $k$th nearest neighbor) and the anisotropy $q^{(X)}, q^{(Y)} \in [0, 1]$. Let the kernel parameters $K^{(i)} = \{\sigma^{(i)}, q^{(i)}\}$.

Then the aligned data is given by $\Phi_{t}^{(X,Y)} = \text{HarmonicAlignment} \left( X, Y, \{K^{(X)}, K^{(Y)}\}, t, \ell \right)$, where the first $\#|X|$ points in $\Phi_{t}^{(X,Y)}$ are the aligned coordinates of $X$ and the last $\#|Y|$ points in $\Phi_{t}^{(X,Y)}$ are the aligned coordinates of $Y$.

Algorithm 6.A.1 function HarmonicAlignment($X, Y, K, t, \ell$) = $\Phi_{t}^{(X,Y)}$

Align the diffusion maps of two datasets.

<table>
<thead>
<tr>
<th>Input:</th>
<th>Data sets ${X, Y}$</th>
<th>Kernel parameters $K = {K^{(X)}, K^{(Y)}}$</th>
<th>Alignment diffusion time $t$</th>
<th>Alignment band count $\ell$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Aligned diffusion map $\Phi_{t}^{(X,Y)}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1: for $Z=X,Y$ do
2: $G^{(Z)} \leftarrow \{W^{(Z)}, L^{(Z)}, D^{(Z)}\} \leftarrow \text{GaussKernelGraph} \left( Z, K^{(Z)} \right)$  \quad \triangleright Alg. 6.A.2
3: $\Psi^{(Z)}, \Lambda^{(Z)} \leftarrow \text{SVD} \left( I - L^{(Z)} \right)$  \quad \triangleright Graph Fourier basis
4: end for
5: $\Phi_{t}^{(X,Y)} \leftarrow \text{Align} \left( \{\Psi^{(X)}, \Lambda^{(X)}, D^{(X)}\}, \{\Psi^{(Y)}, \Lambda^{(Y)}, D^{(Y)}\}, t, \ell \right)$  \quad \triangleright Alg. 6.A.3
6: return $\Phi_{t}^{(X,Y)}$
Algorithm 6.A.2 algorithm

**Input:** Dataset $X = \{ \vec{x}_1, \ldots, \vec{x}_N : x \in \chi \} \subseteq \mathbb{R}^d$

Bandwidth function $\sigma : X \mapsto \mathbb{R}$

**Output:** Kernel matrix $W$

Degree matrix $D$

Normalized Laplacian $L$

1: for $i = 1, N$ do
2:   $D(i, j) \leftarrow 0$
3: for $j = 1, N$ do
4:   $W(i, j) \leftarrow \frac{1}{2} \left( \exp \left( -\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\epsilon(x_i)} \right) + \exp \left( -\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\epsilon(x_j)} \right) \right)$ ▷ Symmetric kernel
5:   $D(i, i) \leftarrow D(i, i) + W(i, j)$ ▷ Degrees
6: end for
7: end for
8: $L \leftarrow I - D^{-1/2}WD^{-1/2}$ ▷ Normalized graph Laplacian
9: return $\{W, D, L\}$

function GaussKernelGraph($X, \epsilon$) = $\{W, D, L\}$
Apply a Gaussian kernel to a dataset and compute the corresponding graph matrices.

Algorithm 6.A.3 function Align($G^{(X)}, G^{(Y)}, t, \ell$) = $\Phi_t^{(X,Y)}$
Compute and apply an alignment matrix to two diffusion maps.

**Input:** Laplacian eigensystems and degrees $G^{(X)}, G^{(Y)} ; G^{(Z)} = \{\psi^{(Z)}, \Lambda^{(Z)}, D^{(Z)}\}$

Alignment diffusion $t$

Alignment band count $\ell$

**Output:** Aligned diffusion map $\Phi_t^{(X,Y)}$

1: for $Z=X,Y$ do
2:   $\Psi^{(Z)} \leftarrow \psi_1^{(Z)} ; \Lambda^{(Z)} \leftarrow \Lambda_1^{(Z)}$
3: $\Phi_0^{(Z)} \leftarrow D^{(Z)1/2} \Psi^{(Z)}$
4: $\hat{Z} \leftarrow \Psi^{(Z)^T} Z$ ▷ Graph Fourier transform
5: end for
6: $w^{(X,Y)} \leftarrow \text{BandlimitingWeights}(\Lambda^{(X)}, \Lambda^{(Y)}, \ell)$ ▷ Alg. 6.A.4
7: for $i = 2, N_1$ do
8:   for $j = 2, N_2$ do
9:     $C(i - 1, j - 1) \leftarrow w_{ij}^{(X,Y)} \{\hat{X}(i - 1, :), \hat{Y}(j - 1, :)\}$ ▷ Bandlimited correlations
10: end for
11: end for
12: $U, S, V \leftarrow \text{SVD}(C)$ ▷ Orthogonalization Sec. 6.4.3
13: $T \leftarrow UV^T$
14: $\Phi_t^{(X,Y)} \leftarrow \begin{bmatrix} \Phi_0^{(X)} & \Phi_0^{(X)} \end{bmatrix} \begin{bmatrix} \Lambda^{(X)} & 0 \\ 0 & \Lambda^{(Y)} \end{bmatrix}^T$
15: return $\Phi_t^{(X,Y)}$
Algorithm 6.A.4 \text{function} \text{BandlimitingWeights}(\Lambda^{(X)}, \Lambda^{(Y)}, \ell) = w^{(X,Y)}

Compute the joint bandlimiting weights for two graphs.

\textbf{Input:} Normalized Laplacian eigenvalues $\Lambda = \left\{ \Lambda^{(X)}, \Lambda^{(Y)} : \Lambda^{(Z)} = \left\{ \lambda_j^{(Z)} \right\}_{j=2}^{\#[Z]} \right\}$

Alignment band count $\ell$

\textbf{Output:} Pairwise frequency weights $w^{(X,Y)} : \Lambda^{(X)} \times \Lambda^{(Y)} \mapsto [0, 1]$

1: $f \leftarrow f : \lambda, \ell, \xi \mapsto I(\xi - 1 \leq \lambda \ell \leq \xi + 1) \sin \left( \frac{\pi}{2} \cos^2 \left( \frac{\pi}{Y} (\ell \lambda - \xi) \right) \right)$ \hfill \text{▷ Iterative wavelet 6.4.1}

2: for $i = 2, N_1$ do
3: \hspace{1em} for $j = 2, N_2$ do
4: \hspace{2em} $w_{ij}^{(X,Y)} \leftarrow 0$
5: \hspace{2em} for $\xi = 1, \ell$ do
6: \hspace{3em} $w_{ij}^{(X,Y)} \leftarrow w_{ij}^{(X,Y)} + f(\lambda_i^{(X)}, \ell, \xi) f(\lambda_j^{(Y)}, \ell, \xi)$
7: \hspace{2em} end for
8: \hspace{1em} end for
9: end for
10: return $w^{(X,Y)}$

### 6.A.2 Multiple dataset alignment

While the previous construction was presented in terms of two datasets for simplicity, we can naturally generalize harmonic alignment to $n$ datasets by considering multiple blocks (rather than the two-by-two block structure in 6.4), based on orthogonalizing pairwise bandlimited correlations between datasets. We briefly elaborate this approach.

Consider $\mathcal{X} = \left\{ \mathcal{X}^{(i)} \subset \mathbb{R}^d : \# \mathcal{X}^{(i)} = N_i \right\}_{i=1}^n$. As in the case with $n = 2$ datasets, we apply a Gaussian kernel to each $\mathcal{X}^{(i)} \in \mathcal{X}$, which yields a Laplacian $\mathcal{L}^{(i)}$ and degree matrix $\mathcal{D}^{(i)}$. Diagonalization of $I - \mathcal{L}^{(i)}$ produces a Fourier basis $\left\{ \Psi^{(i)}, \Lambda^{(i)} \right\}$. As before, we consider the eigenspace $\Psi^{(i)} = \Psi^{(i)} \setminus \psi_1^{(i)}$, $\Lambda^{(i)} = \Lambda^{(i)} \setminus \lambda_1^{(i)}$, from which we compute a graph Fourier transform $\hat{X}^{(i)} = \Psi^{(i)T} X^{(i)}$ and diffusion map $\Phi^{(i)}_0 = D^{(i)1/2} \Psi^{(i)}$ for each dataset. This is the same initialization that one performs for the case when $n = 2$ as in Alg. 6.A.1 6.A.3

Next, for every pair $i \neq j \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ we generate bandlimiting weights (see Alg. 6.A.4)

$$w^{(i,j)} = \text{BandlimitingWeights}\left( \Lambda^{(i)}, \Lambda^{(j)}, \ell \right).$$

Then the correlation between each diffusion map pair is

$$C^{(i,j)}(h-1, k-1) = w^{(i,j)} \left( \lambda_h^{(i)}, \lambda_k^{(j)} \right) \left( \hat{X}^{(i)}(h-1, :) \right) \left( \hat{X}^{(j)}(k-1, :) \right) \text{ for } h=2, \ldots, N_i, k=2, \ldots, N_j.$$
Factoring $C^{(i,j)} = U^{(i,j)} S^{(i,j)} V^{(i,j)^T}$, we have the rigid alignment operator

$$T^{(i\rightarrow j)} = U^{(i,j)} V^{(i,j)^T}$$

and its adjoint $T^{(j\rightarrow i)} = (T^{(i\rightarrow j)})^T$.

Next, let $B^{(1,2,...,n)}(i,j)$ be an $(N_i - 1) \times (N_j - 1)$ matrix such that

$$B^{(1,2,...,n)}(i,j) = \begin{cases} \Phi_0^{(i)} & i = j \\ \Phi_0^{(i)} T^{(i\rightarrow j)} & j > i \\ \Phi_0^{(i)} T^{(j\rightarrow i)} & j < i. \end{cases}$$

Then the $i,j$ block $\Phi_t^{(1,2,...,n)}(i,j)$ is the $(N_i - 1) \times (N_j - 1)$ submatrix of the diffusion coordinates of $X^{(i)}$ aligned into the diffusion space of $X^{(j)}$,

$$\Phi_t^{(1,2,...,n)}(i,j) = B^{(1,2,...,n)}(i,j) \Lambda^{(j)}.$$ 

This matrix is

$$\Phi_t^{(1,2,...,n)} = \begin{bmatrix} \Phi_0^{(1)} & \Phi_0^{(1)} T^{(1\rightarrow 2)} & \cdots & \Phi_0^{(1)} T^{(1\rightarrow n)} \\ \Phi_0^{(2)} T^{(2\rightarrow 1)} & \Phi_0^{(2)} & \cdots & \Phi_0^{(2)} T^{(2\rightarrow n)} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_0^{(n)} T^{(n\rightarrow 1)} & \Phi_0^{(n)} T^{(n\rightarrow 2)} & \cdots & \Phi_0^{(n)} \Lambda^{(n)} \end{bmatrix}. \Lambda^{(1)} \Lambda^{(2)} \cdots \Lambda^{(n)}.$$

Alg. 6.A.5 summarizes this process. When $n = 2$, it simplifies to Alg. 6.A.1

### 6.A.3 Runtime analysis and implementation

Here we provide an informal analysis of algorithmic runtime and suggest a collection of possible improvements that could be made in order to increase the efficiency of our algorithm. We show that the algorithm with cubic sample complexity and linear feature complexity in its naïve implementation, and can be reduced to quadratic sample complexity with relatively straightforward modifications. Breaking the problem down via a divide-and-conquer approach could yield further improvements to give an algorithm with linear sample complexity. We note that in all experiments in this paper, the naïve approach was used.
Algorithm 6.A.5 function MultiAlignment($X, K, t, \ell$) = $\Phi_t^{(1, \ldots, n)}$

Align the diffusion maps of multiple datasets.

| Input: | Data sets $X = \{X^{(1)}, \ldots, X^{(n)}\}$ |
|        | Kernel parameters $K = \{K^{(1)}, \ldots, K^{(n)}\}$ |
|        | Alignment diffusion time $t$ |
|        | Alignment band count $\ell$ |

| Output: | Unified diffusion map $\Phi_t^{(1, \ldots, n)}$ |

1: for $i = 1, n$ do
2:     $G^{(i)} \leftarrow \{W^{(i)}, L^{(i)}, D^{(i)}\} \xrightarrow{\text{GaussKernelGraph}}$ GaussKernelGraph($X^{(i)}, K^{(i)}$) \hspace{1cm} $\triangleright$ Alg. 6.A.3
3:     $\Psi^{(i)}, \Lambda^{(i)} \leftarrow \text{SVD}(I - L^{(i)})$
4:     $\phi^{(i)} \leftarrow \Psi^{(i)} \setminus \psi_1^{(i)}; \Lambda^{(i)} \leftarrow \Lambda^{(i)} \setminus \lambda_1^{(i)}$
5:     $X^{(i)} \leftarrow \phi^{(i)^T} \Lambda^{(i)}$
6:     $\Phi_0^{(i)} \leftarrow D^{(i)^{1/2}} \phi^{(i)}$
7: end for
8: for $i = 1, n$ do
9:     $\Phi_t(i, i)^{(1, \ldots, n)} \leftarrow \Phi_0^{(i)} \Lambda_t^{(i)}$
10: for $j = i + 1, n$ do
11:     $w^{(i,j)} \leftarrow \text{BandlimitingWeights}(\Lambda^{(i)}, \Lambda^{(j)}, \ell)$ \hspace{1cm} $\triangleright$ Alg. 6.A.4
12:     for $\ell = 2, N_i$ do
13:         for $k = 2, N_j$ do
14:             $C^{(i,j)}(\ell - 1, k - 1) \leftarrow w^{(i,j)} (\lambda^{(i)}_\ell, \lambda^{(j)}_k) (\hat{X}^{(i)}(\ell - 1,:), \hat{X}^{(j)}(k - 1,:)) \triangleright$ Bandlimited correlation
15:         end for
16:     end for
17:     $U^{(i,j)}, S^{(i,j)}, V^{(i,j)} \leftarrow C^{(i,j)}$
18:     $T^{(i,j)} \leftarrow U^{(i,j)} V^{(i,j)^T}; T^{(j,i)} \leftarrow V^{(i,j)} U^{(i,j)^T}$
19:     $\Phi_t(i, j)^{(1, \ldots, n)} \leftarrow \Phi_0^{(i)} T^{(i,j)} \Lambda^{(j)}$
20:     $\Phi_t(j, i)^{(1, \ldots, n)} \leftarrow \Phi_0^{(i)} T^{(j,i)} \Lambda^{(i)}$
21: end for
22: return $\Phi_t^{(1, \ldots, n)}$

The runtime complexity of a naïve implementation of Alg. 6.A.1 is

\[
O\left(\begin{array}{l}
N_1^2 + N_2^2 + (N_1^2 + N_2^2) d + N_1 N_2 (d + N_1) + N_1 N_2 (N_1 + N_2)
\end{array}\right) \subset O\left(\begin{array}{l}
N_2^3 + N_1 N_2 d
\end{array}\right),
\]

where $N_1 < N_2$ are the size of two data sets to align and $d$ is the number of dimensions. The major costs of the proposed algorithm are partitioned according to their step in the algorithm (see underbraces).

Some simple observations about the rank of each system will pave the way to reducing alignment runtime.

Our primary tool will be randomized truncated SVD, e.g. Rokhlin et al. (2010); Halko et al. (2011), which computes the first $k$ singular vectors of an $m \times n$ system in $O(mn \log k)$.

First we reduce the size of the input data. Assuming that the features of $X^{(i)}$ are independent, the
simplest setting for dimensionality and rank reduction occurs when \( N_1 < d \) (recalling that \( N_1 < N_2 \)). It is clear by construction that the rank of the correlation matrix is at most

\[
    r_{\text{max}} = \min\{d, N_1\}.
\]

Thus, if the input data \( X^{(i)} \subset \mathbb{R}^d \) for \( d > r_{\text{max}} \), then the most efficient algorithm will use truncated SVD to only consider the first \( r_{\text{max}} \) principal components of each \( X^{(i)} \) as alignment features.

Applying rank reduction to only the input data introduces an additional \( \log r_{\text{max}} \) term to the embedding \( (a) \) through a truncated SVD

\[
    O \left( (N_1^3 + N_2^3) + (N_1^2 + N_2^2) \log r_{\text{max}} \right).
\]

However, the GFT \( (b) \) now runs in

\[
    O \left( (N_1^2 + N_2^2) r_{\text{max}} \right).
\]

Subsequently, correlation and orthogonalization \( (c) \) runs in

\[
    O \left( N_1 N_2 (r_{\text{max}} + N_1) \right),
\]

where the \( r_{\text{max}} \) term is due to the product of a \( N_1 \times r_{\text{max}} \) matrix with a \( r_{\text{max}} \times N_2 \) matrix and the second is the full SVD of the \( N_1 \times N_2 \) correlation matrix.

The same argument can be applied to reduce the number of diffusion coordinates such that \( r_{\text{max}} \) components are taken. The correlation \( (c) \) then collapses to two \( r_{\text{max}}^3 \) operations: one is the matrix product of two square \( r_{\text{max}} \times r_{\text{max}} \) matrices and the other is the full SVD of this product. This reduces the total complexity of harmonic alignment to

\[
    O \left( \begin{array}{c}
        (N_1^2 + N_2^2) 3 \log r_{\text{max}} + (N_1 + N_2) r_{\text{max}}^2 + 2r_{\text{max}}^3 + N_1 N_2 (2r_{\text{max}}) \\
        \text{(a) embedding} \\
        \text{(b) GFT} \\
        \text{(c) correlation} \\
        \text{(d) alignment} \\
        \text{& SVD}
    \end{array} \right) \subset O \left( N_2^2 \log r_{\text{max}} + N_1 N_2 r_{\text{max}} \right).
\]

It is often the case that one is only interested in \( k \ll r_{\text{max}} \) PCA components or diffusion components. For example, Donoho et al. (2013) proves an optimal singular value cutoff for denoising of \( m \times n \) data matrices. One could select a different rank for each PCA and diffusion maps operation; we will denote these as \( \{k_{PCA}^{(1)}, k_{PCA}^{(2)}\} \) and \( \{k_{DM}^{(1)}, k_{DM}^{(2)}\} \), simplifying to \( k_{PCA} = \max\{k_{PCA}^{(1)}, k_{PCA}^{(2)}\} \), \( k_{DM} = \max\{k_{DM}^{(1)}, k_{DM}^{(2)}\} \) and \( k = \max\{k_{PCA}, k_{DM}\} \).
The total complexity of harmonic alignment is now

\[
O \left( \left( N_1^2 + N_2^2 \right) 3 \log k_{PCA} + (N_1 + N_2)k_{PCA}^2 + 2k_{DM}^3 + 2N_1N_2k_{DM} \right) \subset O(N_2^2 \log k + N_1N_2k),
\]

Finally, we can use the multiple alignment algorithm presented in Alg. 6.A.5 to ‘chunk’ very large datasets to reduce runtime. The general scheme would be to break the input into many smaller datasets of \(N_c\) points. The result is a set of \(\sum_i \lceil N_i/N_c \rceil \approx N_1 + N_2 N_c\) smaller problems that run in

\[
O \left( \frac{N_1 + N_2}{N_c} \left( 6N_c \log k_{PCA} + 2N_c k_{PCA}^2 + 2k_{DM}^3 + 2N_c^2 k_{DM} \right) \right) \subset O(N_2 N_c k),
\]

Further analysis must be done to examine the effect on the accuracy of the output when one divides-and-conquers in this way. It is clear that in practice it is important to accordingly adjust \(k\) as the rank structure will vary depending on the sizes of the submatrices chosen to align.

### 6.B Proof of Lemma 6.4.1

The bandlimiting weights from (6.2) satisfy the following properties:

**Claim 6.B.1.** \(w_\xi(\lambda)\) is continuous in \(\lambda\).

**Proof.** As a piecewise function of continuous functions, it suffices to check that \(w_\xi(\lambda)\) is continuous at \(\lambda = \frac{\ell - 1}{\ell}\) and \(\lambda = \frac{\ell + 1}{\ell}\).

\[
\lim_{h \to 0^+} w_\xi \left( \frac{\ell + 1}{\ell} \mp h \right) = \lim_{h \to 0^+} \left( \sin \left( \frac{\ell}{\ell} \cos \left( \frac{\ell}{\ell} \left( \frac{\ell + 1}{\ell} \mp h \right) - \xi \right) \right) \right) = \sin \left( \frac{\ell}{\ell} \cos \left( \frac{\ell}{\ell} \left( \frac{\ell + 1}{\ell} \mp h \right) - \xi \right) \right) = \sin \left( \frac{\ell}{\ell} \cos \left( \pm \frac{\ell}{\ell} \right) \right) = 0 \lim_{h \to 0^-} w_\xi \left( \frac{\ell + 1}{\ell} \mp h \right).
\]

\(\square\)
Corollary 6.B.1. As a sum of continuous functions in $\lambda_i^{(X)}$ and $\lambda_j^{(Y)}$, $w_{ij}^{(X,Y)}$ is continuous in $\lambda_i^{(X)}$ and $\lambda_j^{(Y)}$.

Claim 6.B.2. $w_\xi(\lambda)$ is differentiable in $\lambda$.

Proof. Note first that

$$
\frac{dw_\xi}{d\lambda} = \begin{cases} 
-\frac{\pi^2}{Y} \cos\left(\frac{\pi}{Y} (\ell \lambda - \xi)\right) \cos\left(\frac{\pi}{Y} (\ell \lambda - \xi)\right) \sin\left(\frac{\pi}{Y} (\ell \lambda - \xi)\right) & \frac{\xi - 1}{\ell} \leq \lambda \leq \frac{\xi + 1}{\ell}; \\
0 & \text{otherwise}
\end{cases} 
$$

Then, as a piecewise function of continuous functions, it suffices to check that $\frac{dw_\xi}{d\lambda}$ is continuous at $\lambda = \frac{\xi - 1}{\ell}$ and $\lambda = \frac{\xi + 1}{\ell}$.

$$
\lim_{h \to 0^+} \frac{dw_\xi}{d\lambda} \left( \frac{\xi \pm 1}{\ell} + h \right) = -\frac{\pi^2}{4} \sin\left(\frac{\pi}{Y} \cos\left(\frac{\pi}{Y} \cos\left(\frac{\pi}{Y}\right)\right)\right) = 0 = \lim_{h \to 0^-} \frac{dw_\xi}{d\lambda} \left( \frac{\xi \pm 1}{\ell} - h \right). 
$$

\[ \Box \]

Corollary 6.B.2. As a sum of differentiable functions in $\lambda_i^{(X)}$ and $\lambda_j^{(Y)}$, $w_{ij}^{(X,Y)}$ is differentiable in $\lambda_i^{(X)}$ and $\lambda_j^{(Y)}$.

Claim 6.B.3. If $k \in \mathbb{Z}$ such that $0 \leq k \leq \ell$ and $\frac{k}{\ell} \leq \lambda \leq \frac{k + 1}{\ell}$ then $w_\xi(\lambda) = 0$ for all $\xi \notin \{k, k + 1\}$.

Proof. Assume $k \in \mathbb{Z}$ such that $0 \leq k \leq \ell$ and $\frac{k}{\ell} \leq \lambda \leq \frac{k + 1}{\ell}$. Let $\xi \in \mathbb{Z}$ such that $\xi \notin \{k, k + 1\}$.

For the case where $\xi > k + 1$, then $\xi \geq k + 2$ and so $\lambda \leq \frac{\xi - 1}{\ell}$.

For the case where $\xi < k$, then $\xi \leq k - 1$ and so $\lambda \geq \frac{\xi + 1}{\ell}$.

In each case, this implies that $w_\xi(\lambda) = 0$. \[ \Box \]

Claim 6.B.4. If $\lambda_i^{(X)} = \lambda_j^{(Y)}$ then $w_{ij} = 1$.

Proof. Assume $\lambda_i^{(X)} = \lambda_j^{(Y)} = \lambda$ and let $k \in \mathbb{Z}$ such that $\frac{k}{\ell} \leq \lambda \leq \frac{k + 1}{\ell}$.

Then
Claim 6.B.5. If \(|\lambda_i^{(X)} - \lambda_j^{(Y)}| \geq \frac{2}{\ell}\) then \(w_{ij}^{(X,Y)} = 0\).

**Proof.** Assume \(|\lambda_i^{(X)} - \lambda_j^{(Y)}| \geq \frac{2}{\ell}\) and let \(k_i^{(X)}, k_j^{(Y)} \in \mathbb{Z}\) such that \(\frac{k_i^{(X)}}{\ell} \leq \lambda_i^{(X)} \leq \frac{k_i^{(X)} + 1}{\ell}\) and \(\frac{k_j^{(Y)}}{\ell} \leq \lambda_j^{(Y)} \leq \frac{k_j^{(Y)} + 1}{\ell}\).

Assume without loss of generality that \(\lambda_i^{(X)} < \lambda_j^{(Y)}\). Then by Claim 6.B.3

\[
\lambda_i^{(X)} - \frac{2}{\ell} \leq \frac{k_j^{(Y)}}{\ell} + 1 - \frac{2}{\ell} = \frac{k_j^{(Y)} - 1}{\ell} \implies w_\xi(\lambda_i^{(X)}) = 0 \text{ for all } \xi \geq k_j^{(Y)}
\]

and since \(w_\xi(\lambda_j^{(Y)}) = 0\) for all \(\xi \notin \{k_j^{(Y)}, k_j^{(Y)} + 1\}\), then

\[
w_\xi(\lambda_i^{(X)})w_\xi(\lambda_j^{(Y)}) = 0 \text{ for all } \xi.
\]

**Claim 6.B.6.** The rate of change of \(w_{ij}^{(X,Y)}\) w.r.t. both \(\lambda_i^{(X)}\) and \(\lambda_j^{(Y)}\) is bounded by \(O(\ell)\).

**Proof.** From Equation 6.6

\[
\left| \frac{dw_\xi}{d\lambda} \right| \leq \frac{\pi^2 \ell}{4} \text{ for all } \lambda.
\]

Without loss of generality we consider only \(\frac{dw_{ij}^{(X,Y)}}{d\lambda_i^{(X,Y)}}\). Let \(k \in \mathbb{Z}\) such that \(\frac{k}{\ell} \leq \lambda_i^{(X)} \leq \frac{k+1}{\ell}\).
\[
\frac{dw_{ij}^{(X,Y)}}{d\lambda_i^{(X)}} = \sum_{\ell \geq 1} \frac{dw_{\xi}}{d\lambda_i^{(X)}} w_\xi(\lambda_j^{(Y)})
\]
\[
= \frac{dw_k}{d\lambda_i^{(X)}} w_k(\lambda_j^{(Y)}) + \frac{dw_{k+1}}{d\lambda_i^{(X)}} w_{k+1}(\lambda_j^{(Y)})
\]
\[
\leq \frac{dw_k}{d\lambda_i^{(X)}} + \frac{dw_{k+1}}{d\lambda_i^{(X)}}.
\]

So
\[
\left| \frac{dw_{ij}^{(X,Y)}}{d\lambda_i^{(X)}} \right| \leq \left| \frac{dw_k}{d\lambda_i^{(X)}} \right| + \left| \frac{dw_{k+1}}{d\lambda_i^{(X)}} \right| \leq \frac{\pi^2}{Y} \ell.
\]

\[\square\]

**Corollary 6.B.3.** The rate of change of \(w_{ij}^{(X,Y)}\) w.r.t. \(|\lambda_i^{(X)} - \lambda_j^{(Y)}|\) is bounded by \(O(\ell)\).

### 6.C Comparison to Wang and Mahadevan

Despite being a natural candidate for comparison to our method, unfortunately no standard implementation of the method proposed by [Wang and Mahadevan (2009)](http://proceedings.mlr.press/v80/amodio18a.html) is available. Our implementation of their method performed extremely poorly (worse than random) on the comparisons and is extremely computationally intensive. The method is therefore not shown in the main comparisons; however, for completeness, the results are shown in Figure 6.C.1.

### Bibliography


Figure 6.C.1: Recovery of k-neighborhoods under feature corruption. Mean over 3 iterations is reported for each method. (a) Lazy classification accuracy relative to input size with unlabeled randomly corrupted digits with 35% preserved pixels. (b) Transfer learning performance. For each ratio, 1K uncorrupted, labeled digits were sampled from MNIST, and then 1K, 2K, 4K, and 8K (x-axis) unlabeled points were sampled and corrupted with 35% column identity.


Chapter 7

Future Work

7.1 Hierarchical interactive visualization of large data with coarsened graph diffusion

The application of PHATE to large datasets (many millions of cells) is limited by both the capacity to apply
the algorithm to such datasets, and by the capacity of the human user to view and understand a plot of this
many points. Here, we propose Hierarchical PHATE (H-PHATE), an extension of PHATE to the “big data”
realm through an interactive, hierarchical visualization tool that will allow users to explore their data on
multiple scales, providing the ability to understand a large dataset holistically as well as to explore subsets
of the data in greater detail, all in real time.

H-PHATE consists of three main components. First, we will speed up the computation of PHATE’s
alpha-decay kernel through the use of an approximate nearest neighbors kernel (Arya et al., 1998)

\[ K_{ANN}(x, y) = \begin{cases} K_{k,\alpha}(x, y), & \text{if } d(x, y) < \varepsilon_{k_{\max}}(x) \\ 0, & \text{otherwise.} \end{cases} \]

combined with random sampling to fill in the substantial non-zero connections between points based on
the kernel’s long tail

\[ K_{random}(x, y) = \begin{cases} K_{k,\alpha}(x, y), & \text{if } d(x, y) > \varepsilon_{k_{\max}}(x) \text{ and } a \sim Uniform(0, 1) < \sigma(x) \\ 0, & \text{otherwise.} \end{cases} \]

to give an approximate kernel \( \hat{K}_{k,\alpha} = K_{ANN} + K_{random} \). Second, we will facilitate the visualization of
millions of points by presenting to the user only a Laplacian-consistent coarsening of the graph. This can be done using edge-based contraction from [Loukas (2019)], which has been shown to preserve a low-rank approximation of the eigenvalues and eigenvectors of the original graph while reducing the number of nodes by up to 70% at a computational cost that scales linearly with the number of edges. This coarsening provides a hierarchical representation \( \mathcal{H} = \{ \mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_\ell \} \) of the original graph \( \mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0) \) where there exists a surjective map \( c_i : \mathcal{V}_i \to \mathcal{V}_{i+1} \) and

\[
\frac{1}{2} |\mathcal{V}_i| \leq |\mathcal{V}_{i+1}| < |\mathcal{V}_i|.
\]

Through this graph coarsening, we can show the user a constant number of points on the screen at any one time, where the size of the point represents the size of the cluster it represents. This gives an approximate understanding of the dataset at a high level with relatively few points, as shown in Figure 7.1.1.

Finally, in order to maintain the level of resolution available in standard PHATE visualizations, we will provide an interactive visualization interface such that the user can “zoom in” on areas of interest in real time in order to alleviate the loss in resolution enforced by the coarsened graph, as in [van Unen et al. (2017)]. Each zoom action requires only the inversion of \( c_i \) on the graph subset of interest. By forcing the user to zoom in on no more than half of the graph at any one time, the total number of points to be displayed on the screen is roughly constant, so the computational cost of visualization at runtime is also constant. The
Table 7.2.1: Representative tasks for applications of dimensionality reduction for visualization.

<table>
<thead>
<tr>
<th>Task category</th>
<th>Task Description</th>
<th>Error metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster identification</td>
<td>Count clusters on screen</td>
<td>Absolute error</td>
</tr>
<tr>
<td>Cluster confirmation</td>
<td>Merge the most similar groups</td>
<td>Rank of merged groups</td>
</tr>
<tr>
<td>Trajectory identification</td>
<td>Identify ends of dominant traj.</td>
<td>Angle from true traj.</td>
</tr>
<tr>
<td>Trajectory confirmation</td>
<td>Identify quartiles of labeled traj.</td>
<td>Percentage error</td>
</tr>
<tr>
<td>Outlier detection</td>
<td>Label the outliers on screen</td>
<td>Sum of missed outlier score</td>
</tr>
<tr>
<td>Density estimation</td>
<td>Count points in a group</td>
<td>Absolute error</td>
</tr>
<tr>
<td>Similarity estimation</td>
<td>Rank groups by variance</td>
<td>Correlation of rankings</td>
</tr>
</tbody>
</table>

The largest drawback of DEMaP, the metric we designed to assess the performance of PHATE in Moon et al. (2019), was the behavioral assumption that a preservation of manifold distances in low-dimensional Euclidean space would lead to a superior analysis by the user. Here, we will explore the intersection between mathematical quantification of quality of visualizations and the ability of users to interact with the visualization to perform exploratory data analysis tasks relevant to single cell analysis.

To do this, we will design a web interface in which users perform a combination of cluster-based and trajectory-based tasks informed by the dimensionality reduction taxonomy described in Sedlmair et al. (2012); Etemadpour et al. (2015) (Table 7.2.1). We will assess the performance of both expert and naive dimensionality reduction users in order to a) find an automated metric that most closely mirrors human performance for a range of tasks, and b) design a new dimensionality reduction algorithm that attempts to optimize for this metric.

Specifically, we will produce a ranking of dimensionality reductions for each dataset according to human performance of each task, and compute the Spearman correlation with each of the following automated metrics of quality: DEMaP (Moon et al. 2019), 1-nearest neighbor (Maaten and Hinton 2008), and Adjusted Rand Index (Hubert and Arabie 1985), as shown in Figure 7.2.1.

Following this, we will attempt to design a metric which most closely mirrors human performance on one or more tasks, potentially by combining existing metrics, as is done in Icke and Rosenberg (2011). We will then use this designed metric as an optimization goal for our modified dimensionality reduction algorithm.
7.3 Selecting the right tool for the job: a comparison of visualization algorithms

Selecting an appropriate dimensionality reduction algorithm for the purposes of visualization is a daunting task. Dozens of algorithms exist, and there currently exists no simple way to compare the most common of these. Here, we will distill the information learned in the above work into a pedagogical summary of dimensionality reduction for machine learning. This summary, a preliminary version of which is found in Burkhardt et al. (2019), will present information in such a way that non-experts can explore the differences between algorithms and parameter selections in order to more optimally select tools for their own analyses.

First, we will present each algorithm as a series of steps, each presented visually as applied to a simple, representative dataset. This interaction will provide users with an understanding of how the algorithms work, and will establish the foundations for understanding the strengths, limitations, and unique characteristics of each algorithm.

Second, we will explore the challenges of parameter selection. Through an interactive visualization of the Swiss Roll dataset, we will allow the user to select common parameters of each algorithm and use this to guide the user to an understanding of a) the importance of parameter selection and b) the common pitfalls of visualizing data with poorly tuned parameters. An example of this parameter selection widget is shown in Figure 7.3.1.

Third, we will show a number of real-world datasets visualized with each algorithm, and use these datasets as representatives of the kinds of complex structures which are represented both well and poorly with each of the algorithms. By using a combination of image data (which is common in the machine learning literature), population genetics data (the structure of which relates to the Earth’s continental structure, and hence is easy to understand), single-cell transcriptomic data (which is one of the most common applications of t-SNE...
Figure 7.3.1: Interactive widget showing the influence of parameter selection on visualization. (left) 3D visualization of the Swiss Roll dataset. (right) t-SNE embedding of the Swiss Roll dataset with the selected parameters. (bottom) Sliders allowing modification of key parameters.

and UMAP), and network data (which displays a unique capability of some algorithms to visualize data that exists natively on graphs), we will give a survey of the kinds of data frequently visualized with these algorithms and ensure that all users have at least one example familiar to them.

Finally, we will explore the challenges of quantification by repeating the simulation experiment in Moon et al. (2019) in which we apply DEMaP, 1-nearest neighbor classification, and Adjusted Rand Index to a series of simulations from Zappia et al. (2017). By showing these simulations one by one alongside a running aggregate, users will understand that no one method outperforms all other methods all the time, and that the selection of the right method depends not only on the qualities of the method but also on the qualities of the data and the task at hand.

Bibliography


Daniel B Burkhardt, Scott Gigante, and Smita Krishnaswamy. Selecting the right tool for the job: a


Bibliography


El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall,


Daniel B Burkhardt, Jay S Stanley, Alexander Tong, Ana Luisa Perdigoto, Scott A Gigante, Kevan C Herold, Guy Wolf, Antonio J Giraldez, David van Dijk, and Smita Krishnaswamy. Quantifying the


François Chollet et al. Keras. https://keras.io 2015.


Scott Gigante, Jay S Stanley, Ngan Vu, David van Dijk, Kevin R Moon, Guy Wolf, and Smita Krishnaswamy. Compressed diffusion. In *Sampling Theory and Applications*, pages 1–4. IEEE, July 2019b, doi:10.1109/SampTA45681.2019.9030994 © 2019 IEEE. Reprinted with permission. In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Yale University’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications_standards/publications/rights/rights_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.


Dmitry Kobak and George C Linderman. Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, pages 1–2, 2021, doi:10.1038/s41587-020-00809-z.


