Neural FIM for learning Fisher information metrics from point cloud data

Oluwadamilola Fasina [*1] **Guillaume Huguet** [*2 3] **Alexander Tong** [2 4] **Yanlei Zhang** [2 3] **Guy Wolf** [2 3]
Maximilian Nickel [5] **Ian Adelstein** [†6] **Smita Krishnaswamy** [†7 8 1]

Abstract

Although data diffusion embeddings are ubiquitous in unsupervised learning and have proven to be a viable technique for uncovering the underlying intrinsic geometry of data, diffusion embeddings are inherently limited due to their discrete nature. To this end, we propose *neural FIM*, a method for computing the Fisher information metric (FIM) from point cloud data - allowing for a continuous manifold model for the data. Neural FIM creates an extensible metric space from discrete point cloud data such that information from the metric can inform us of manifold characteristics such as volume and geodesics. We demonstrate Neural FIM's utility in selecting parameters for the PHATE visualization method as well as its ability to obtain information pertaining to local volume illuminating branching points and cluster centers embeddings of a toy dataset and two single-cell datasets of IPSC reprogramming and PBMCs (immune cells).

1. Introduction

An important goal of unsupervised learning is understanding the underlying shape or geometry of data (Bronstein et al., 2017; Cheng et al., 2019; De Domenico, 2017; Tsitsulin et al., 2019). A key paradigm here is the manifold assumption which hypothesizes that high dimensional data, particularly from scientific domains, lies on a lower dimensional smoothly varying manifold (see Huguet et al. (2022); He et al. (2014); Bhaskar et al. (2022); Lin & Zha (2008)). Prior methods for learning data manifolds use data affinity kernels that compute a pairwise distance matrix from a data set, then pass the distances through a kernel function (such as a Gaussian kernel function) to convert distances to affinities (Belkin & Niyogi, 2003; Bunte et al., 2012; Mika et al., 1998). Eigenvectors of such an affinity matrix give the data a manifold-intrinsic coordinate representation. This method, while successful in some respects, has a key disadvantage that it is implicitly biased by the particular sampling of the data that is given, together with its irregularities. Further, there is usually no straightforward way of extending such manifold coordinate representations to unseen points.

Here we propose a neural-network based method of directly learning a Riemannian metric for data called the neural FIM. Loosely speaking, a Riemannian metric is an infinitesimal generator of manifold-intrinsic length and volume, based on an inner product structure on the tangent space of every point. Typically, a Riemannian metric cannot be learned from discrete data as there is no continuous model of the manifold. Although Bengio et al. (2003); Schoeneman et al. (2017); Law & Jain (2006); Dadkhahi et al. (2017) developed methods for extending coordinate manifold representations to unseen points, none of these methods admit continuous manifold models and involve expensive computations. Neural FIM is able to learn a continuous manifold model by using a neural network to embed data points into a latent space and creates a continuous implicit model of the data from which we can compute the metric.

The specific Riemannian metric we aim to learn the data manifold is the Fisher Information Metric (FIM). This type of metric is defined on *statistical manifolds*, manifolds where each datapoint is a probability distribution (Lauritzen, 1987; Lafferty et al., 2005; Noguchi, 1992). We obtain such a pointwise probability distribution on point cloud data by way of a data diffusion operator, as first defined in the seminal work on diffusion maps (Coifman & Lafon, 2006). After an affinity kernel is computed, it is row normalized to a stochastic matrix. This normalized matrix is treated as a Markovian operator which defines a random walk or a diffusion on the data. We associate each data point to the

[*]Equal contribution , [†]Co-senior authors, [1]Applied Math Program, Yale University, New Haven, CT, USA. [2]Mila - Quebec AI Institute, Montreal, QC, Canada. [3]Department of Mathematics and Statistics, Université de Montréal, Montreal, QC, Canada. [4]Department of Computer Science and Operations Research, Université de Montréal, Montreal, QC, Canada. [5]FAIR, Meta AI [6]Department of Math, Yale University, New Haven, CT, USA. [7]Department of Computer Science, Yale University, New Haven, CT, USA. [8]Department of Genetics, Yale University, New Haven, CT, USA.. Correspondence to: Smita Krishnaswamy <smita.krishnaswamy@yale.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

transition probability distribution given by its row of the stochastic matrix, and thus realize the point cloud data as a statistical manifold.

By utilizing a mathematical connection between the differential form of Jensen-Shannon Distance (JSD) (the square-root of the standard Jenson-Shannon (JS) divergence) and neural FIM, we derive a method of training the neural FIM on the basis of distances between PHATE embeddings that use JSD. An advantage of this approach is that—similar to PHATE—the embedding is globally contextualized due to the information-theoretic distances that are computed. We can then use the FIM to compute geometric quantities on the statistical manifold such as length and volume. We show how the geodesic or Fisher-Rao distance between pairs of points can be computed using an auxiliary neural Ordinary Differential Equation (ODE) network (Chen et al., 2018). This distance can be used for novel embeddings and downstream tasks. The magnitude of the volume element captures local distinguishability and can be used to reveal branching points in hierarchical data or decision boundaries in classification problems.

We showcase our results on three types of tasks. First, we show how to use the FIM to explore the space of parameters for the PHATE embedding method. Here, the statistical manifold is created from the diffusion operator resulting from various embedding parameters (on the same dataset). In specific, we explore selection of the time-of-diffusion and bandwidth variables. The second task involves computing the FIM of 3 different datasets: a toy tree dataset, an IPSC reprogramming mass cytometry dataset (Zunder et al., 2015) and a pbmc single cell RNA-sequencing dataset (10x Genomics, 2019). These statistical manifolds correspond to transition probability distributions of each datapoint within the dataset. Both the neural FIM embeddings and information from the FIM including volume and trace are shown for each of the three datasets. We see that the volume highlights freedoms of movements with branchpoints having higher volume. Finally, we utilize the neural ODE network to compute geodesic paths within the embedding between points. First we show this on data sampled from a sphere, and then on the IPSC dataset. Remarkably, in the IPSC dataset the geodesic follows the path of reprogramming of a cell from it starting state.[1]

The key contributions of this work include:

- Conceptually connecting data diffusion embeddings with statistical manifolds in order to derive a manifold model of the data, complete with a continuously-defined Fisher Information Metric tensor.
- Proposing the neural FIM method for extensible FIM computations (i.e., extensible to unseen data) trained by using Jensen-Shannon divergence between data diffusion probabilities.
- Proposing a neural-ODE based method for computing geodesic paths and distances based on the FIM.
- Showcasing the use of neural FIM in selecting parameters, visualizing single cell data, and locally extracting information about the volume, trace, and eigenspectrum of the metric to understand the underlying manifold geometry of the data.

[1]Code is available at: `https://github.com/guillaumehu/phate_fim`

2. Background

A useful assumption in manifold learning is that data measured in a high-dimensional ambient space originates from an intrinsic low-dimensional manifold. The manifold assumption asserts that if $\mathcal{M}^d$ is a hidden d dimensional manifold, it is observable by a collection of $n \gg d$ non-linear functions $f_1, \ldots, f_n : \mathcal{M}^d \to \mathbb{R}$ which enable its immersion in a high dimensional ambient space as $F(\mathcal{M}^d) = \{\mathbf{f}(z) = (f_1(z), \ldots, f_n(z))^T : z \in \mathcal{M}^d\} \subseteq \mathbb{R}^n$ from which data is collected. Conversely, given data $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^n$ of high dimensional observations, manifold learning methods assume the data originates from a sampling $Z = \{z_i\}_{i=1}^N \subset \mathcal{M}^d$ of the underlying manifold via $x_i = \mathbf{f}(z_i)$, and aim to learn a low dimensional intrinsic representation that approximates the manifold geometry of $\mathcal{M}^d$.

2.1. Data Diffusion

A popular class of methods for manifold learning uses a data diffusion operator, which models data based on transition or random walk probabilities through the data. Methods that use a data diffusion operator include diffusion maps (Coifman & Lafon, 2006), PHATE (Moon et al., 2019), tSNE (van der Maaten & Hinton, 2008), and diffusion pseudotime (Haghverdi et al., 2016). One can learn the manifold geometry with data diffusion by first computing local similarities defined via a kernel $\mathcal{K}(x, y)$, $x, y \in F(\mathcal{M}^d)$. We note that a popular choice for a kernel is the Gaussian kernel $\mathcal{G}(x, y) = \exp(-\|x - y\|^2/\sigma)$, where $\sigma > 0$ is interpreted as a user-configurable scale parameter. However, this choice encodes sampling density information together with local geometric information.

To construct a diffusion geometry that is robust to sampling density variations, we use an anisotropic kernel $\mathcal{K}(x, y) = \frac{\mathcal{G}(x,y)}{\|\mathcal{G}(x,\cdot)\|_1^\alpha \|\mathcal{G}(y,\cdot)\|_1^\alpha}$, where $0 \leq \alpha \leq 1$ controls the separation of geometry from density, with $\alpha = 0$ yielding the classic Gaussian kernel, and $\alpha = 1$ completely removing density and essentially providing uniform sampling of the manifold. Finally, we row-normalize $\mathcal{K}$ to define transition

probabilities $p(x, y) = \mathcal{K}(x, y)/\|\mathcal{K}(x, \cdot)\|_1$ and define an $N \times N$ diffusion matrix $\mathbf{P}_{ij} = p(x_i, x_j)$ that describes a Markovian diffusion over the the data.

2.2. PHATE

There are several dimensionality reduction methods that render data into 2-D visuals, such as PCA, tSNE (van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2018). However, these methods fail to preserve the global manifold structure of the data and are not robust to noise. PCA cannot denoise in non-linear dimensions, and tSNE/UMAP effectively only constrain for near neighbor preservation—losing global structure. This motivated the development of a method of dimensionality reduction that retains manifold structure and denoises data (Moon et al., 2019).

PHATE also builds upon the diffusion-based manifold learning framework from Coifman & Lafon (2006), and involves the creation of a diffused Markov transition matrix from data, $\mathbf{P}$. PHATE collects all of the information in the diffusion operator into two dimensions such that global and local distances are retained. To achieve this, PHATE considers the ith row of $\mathbf{P}^t$ as the representation of the ith datapoint in terms of its t-step diffusion probabilities to *all* other datapoints. PHATE then preserves a novel distance between two datapoints, based on this representation called *potential distance (pdist)*. Potential distance is an M-divergence between the distribution in row i, $\mathbf{P}^t_{i,\cdot}$ and the distribution in row j, $\mathbf{P}^t_{j,\cdot}$. These are indeed distributions as $\mathbf{P}^t$ is Markovian:

$$pdist(i, j) = \sqrt{\sum_k (\log(P^t(i, k)) - \log(P^t(j, k))^2} \quad (1)$$

The $\log$ scaling inherent in potential distance effectively acts as a damping factor which makes faraway points similarly equal to nearby points in terms of diffusion probability. This gives PHATE the ability to maintain global context. The paper also allows for other types of symmetric divergences such as the JS divergence, which we use in our work to train neural networks.

These potential distances are embedded with metric MDS as a final step to derive a data visualization. Moon et al. (2019) have shown that PHATE outperforms tSNE (van der Maaten & Hinton, 2008), UMAP (McInnes et al., 2018), force-directed layout, and 12 other methods on the preservation of manifold affinity, and adjusted rand index on clustered datasets, in a total of 1200 comparisons on synthetic and real datasets.

2.3. Information Geometry and Fisher Information

Information geometry (Amari, 2016; Nielsen, 2020; Arwini & Dodson, 2008; Li & Rubio, 2022; Lin et al., 2021) combines statistics and differential geometry to study the geometric structure of statistical manifolds. A statistical manifold is a Riemannian manifold (M, g) where every point in the space $p \in M$ is a probability distribution. The Fisher Information Metric (FIM) is the standard Riemannian metric on statistical manifolds and measures the distinguishability between points on the manifold (probability distributions).

In the Riemannian setting one endows a smooth manifold M^n with geometry by defining at each point $p \in M^n$ an inner product $g_p(\cdot, \cdot) : T_pM \times T_pM \to \mathbb{R}$, where T_pM represents the tangent space of M at p. The collection of inner products g defines a Riemannian metric on the manifold M.

On a statistical manifold M^n parameterized by the coordinates $\theta = (\theta_1, \ldots, \theta_n)$ we have that points are distributions $p(x, \theta) \in M$ over some common probability space (or data set) X. A Riemannian metric one can compute on a statistical manifold is the Fisher Information Metric (FIM):

$$I_{ij}(\theta) = \int_X \frac{\partial \log p(x, \theta)}{\partial \theta_i} \frac{\partial \log p(x, \theta)}{\partial \theta_j} p(x, \theta) dx \quad (2)$$

One can use this metric to compute geometric quantities on the statistical manifold such as length or volume. Generically, for a parameterized curve on a Riemannian manifold $c\colon [a, b] \to (M, g)$ its length is given by

$$L(c) = \int_a^b |\dot{c}(t)|\, dt = \int_a^b \sqrt{g_{c(t)}(\dot{c}(t), \dot{c}(t))}\, dt.$$

Volume of the manifold is given by

$$V(M) = \int_M \sqrt{|det(g)|}\, d\theta.$$

These geometric quantities can be used to provide insight into the original data space X. We note that access to a Riemannian metric theoretically provides access to its associated Riemann curvature tensor. However the Riemann curvature tensor is defined in via the metric's unique Levi-Civita connection, an object that we do not explore here.

Example: We consider a family of distributions $\mathcal{F}_\Theta$ parameterized by a parameter space Θ. For the Gaussian family the parameter space is $\Theta = \{(\mu, \sigma) : \mu \in \mathbb{R}, \sigma \in \mathbb{R}^+\}$, and the family is defined by $\mathcal{F}_\Theta = \{\mathcal{N}(\mu, \sigma) : (\mu, \sigma) \in \Theta\}$. This parameterization turns the Gaussian family into a 2-dimensional statistical manifold, with FIM

$$I(\mu, \sigma) = \begin{bmatrix} 1/\sigma^2 & 0 \\ 0 & 2/\sigma^2 \end{bmatrix}$$

We see that the Gaussian family admits a hyperbolic geometry, where the distance between distributions with fixed differences in means increases as their variance decreases. This example illuminates the main interpretation of the FIM and its associated (Fisher-Rao) distance: the more distinguishable are two distributions (say in terms of inference) the larger their Fisher-Rao distance. The FIM locates Gaussians with small variance at greater distances in the statistical manifold.

The FIM has a connection to other information theoretic quantities, in particular to Kullback-Liebler (KL) and Jensen-Shannon (JS) divergences, as shown in Crooks (2007). The FIM is an infinitesimal version of the KL-divergence. Key results are included below for reference.

Theorem 2.1. *(from Crooks, 2007) The infinitesimal Jensen-Shannon divergence,* $dJS = JS(p, p + dp) = \frac{1}{8}\sum_i \frac{(dp_i)^2}{p_i}$ *is equal to the FIM,* $\frac{dc^i}{dt} I_{ij}(c) \frac{dc^j}{dt} = \sum_x \frac{1}{p(x)}[\frac{dp(x)}{dt}]^2$.

One extends this infinitesimal result to more global objects by integrating over parameterized paths.

Corollary 2.2. *(from Crooks, 2007) The length (with respect to the FIM) of a parameterized path* $c(t)$ *equals the total Jensen-Shannon divergence over the curve:*

$$\int_a^b \sqrt{\frac{\partial c^i}{\partial t} I_{i,j} \frac{\partial c^j}{\partial t}}\, dt = \sqrt{8} \int_a^b d\sqrt{JS} \tag{3}$$

where $d\sqrt{JS}$ *is the infinitesimal change in the Jensen–Shannon divergence along* $c(t)$.

3. Methods

3.1. Neural FIM

To approximate a continuous FIM from a finite set of distributions, we approximate the family of distributions via a neural network, whose Jacobian we use to evaluate the FIM in a continuous manner. The basic framework, shown in Figure 1, consists of two parts. The first part (Figure 1A) is a neural network trained to match Jensen-Shannon Distances (explained below) from which a Jacobian is extracted for FIM computation. The second part (Figure 1B) is a neural ODE network that computes geodesic paths, i.e., shortest length paths on data manifolds.

We consider the dataset to be a point cloud $X \in \mathcal{X}$ of size n from a sigma finite distribution q. The first step is to translate such a point cloud into a family of distributions. To do so, we construct an affinity graph from the point cloud, and its diffusion operator $\boldsymbol{P}_n^t$. Each row of the diffusion operator is a distribution (probability mass function) that describes the transition probabilities of a random walk on the data; it thus defines a map $x \mapsto \boldsymbol{P}_n(x, \cdot)$, where $\boldsymbol{P}_n(x, \cdot)$ is the row corresponding to the observation x. The construction of this map is summarized in Algorithm 1 (see Appendix). Further, we assume that $x \mapsto \boldsymbol{P}_n(x, \cdot)$ is differentiable.

We first consider training a neural network $\phi : \mathbb{R}^d \to \mathcal{P}(Z)$, where $\mathcal{P}(Z)$ is the space of probability mass functions on a latent space $Z \subset \mathbb{R}^n$, minimizing the loss function

$$\begin{aligned} L(\phi) = \mathbb{E}_{x\sim p_{\text{data}}} \|\phi(x) - \boldsymbol{P}_n(x, \cdot)\|_2 \\ + \|\nabla_x \phi(x) - \nabla_x \boldsymbol{P}_n(x, \cdot)\|_2 \end{aligned} \tag{4}$$

This renders the data set into a statistical manifold consisting of n points with each point defining a probability distribution in $\mathbb{R}^n$ where n is the dimensionality of the last layer. We can then utilize the Jacobian of this embedding with respect to the inputs to obtain the partial derivatives required for the computation of an FIM at any $x \in \mathbb{R}^d$ via

$$I_\phi(x)_{i,j} = \sum_k \boldsymbol{J}_\phi(x)_{k,i} \boldsymbol{J}_\phi(x)_{k,j} \phi(x)_k \tag{5}$$

where $\boldsymbol{J}_\phi$ is the Jacobian matrix of ϕ with respect to the input variables. Notably, $\boldsymbol{J}_\phi$ is *not* the Jacobian used for the training of the neural network, i.e., the Jacobian with respect to parameters such as weights and biases of the neural network. However, this training method requires that the dimensionality of the last layer would be very high, and that we train to match data derivatives which hard to obtain. Thus we do not use this loss in practice.

We instead offer a much more efficient alternative: we reduce the data to an arbitrarily low m dimensional latent space Z by training the neural network to match the Jensen-Shannon divergence between rows of the distribution, which would be similar to PHATE (Moon et al., 2019) distance using this alternative divergence:

$$\begin{aligned} &JS(\boldsymbol{P}_n^t(i, \cdot), \boldsymbol{P}_n^t(j, \cdot)) := \\ &\tfrac{1}{2} KL((\boldsymbol{P}_n^t(i, \cdot)||M) + KL((\boldsymbol{P}_n^t(j, \cdot)||M), \end{aligned}$$

where $M := (1/2)(\boldsymbol{P}_n^t(i, \cdot) + \boldsymbol{P}_n^t(j, \cdot))$.

Since we match only distances here, there is no restriction on the dimensionality of the output space Z. We achieve this by using this alternative loss function:

$$\begin{aligned} L_{JS}(\phi) := \mathbb{E}_{x\sim p_{\text{data}}} \Big\| \sqrt{JS(\phi(x), \phi(y))} \\ - \sqrt{JS(p(x), p(y))} \Big\|_2^2 \end{aligned} \tag{6}$$

Below, we show that in either case our neural network Jacobians can be used to compute the FIM of datapoints within the manifold of the data.

Proposition 3.1. *Assume for any* $x \in X$*, we have uniform convergence of* $\lim_{n\to\infty} \boldsymbol{P}_n(x, \cdot) = P(x, \cdot)$ *where* $\boldsymbol{P}_n$ *and* P *are Markov operators with compact support. If* $L(\phi_n) \to 0$ *uniformly, then we have* $\lim_{n\to\infty} \nabla_x \phi_n(x) = \nabla_x P(x, \cdot)$.

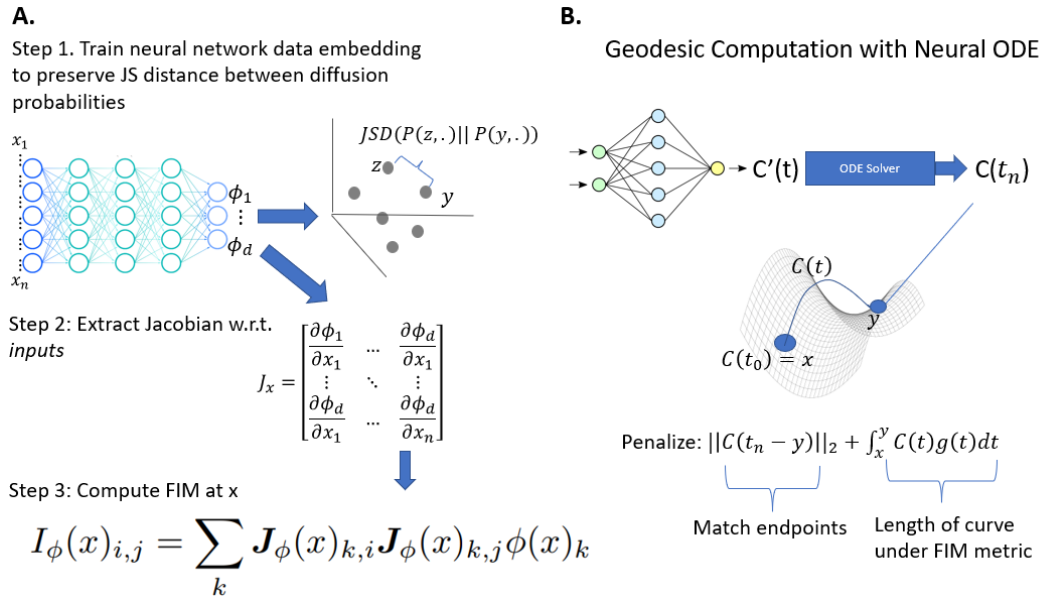


Figure 1. Schematic of neural-FIM which is used to generate a continuous FIM embedding (A) and schematic of neural-ODE used to find geodesics with FIM (B).

Proof. Since $\boldsymbol{P}_n$ is continuous and has compact support, by the universal approximation theorem (Cybenko, 1989), $\boldsymbol{P}_n$ can be approximated by a feed forward neural network with a finite number of neurons. By the definition of loss function in Equation 4, for a fixed n, converging uniformly to 0 implies $\phi_n(x)$ converging uniformly to $\boldsymbol{P}_n(x,\cdot)$. Because the convergence of derivatives is also uniform, we can interchange the limit and the derivative, obtaining

$$\begin{aligned}\lim_{n\to\infty} \nabla_x \phi_n(x)_j &= \nabla_x \lim_{n\to\infty} \phi_n(x)_j \\ &= \nabla_x \lim_{n\to\infty} \boldsymbol{P}_n(x, x_j) \\ &= \nabla_x P(x, x_j).\end{aligned}$$

□

The above proposition shows that a neural network trained to match diffusion probabilities as in Equation 4 will have its derivatives as described in in Equation 5. Thus, the neural network can be used to compute the partial derivatives needed to compute the FIM. Directly enforcing the loss is simple, however it requires the network output size to scale linearly with the dataset. To allow for FIM to be continuous, we require that our neural-FIM embeddings to be continuously differentiable.

If instead of using the loss function from Equation 4 we use the alternative loss function from Equation 6, we still converge to the FIM as below.

Proposition 3.2. *Assume that p has compact support. As $|X| \to \infty$, if $L_{JS}(\phi)$ converges to 0, then $I_\phi(x) = I_p(x)$ for all $x \in X$.*

Proof. Since L_{JS} converges to 0, for an infinitesimal Δx,

$$JS(\phi(x), \phi(x + \Delta x)) = JS(p(x), p(x + \Delta x)).$$

For any C^1 path c between x and $x + \Delta x$, we can apply Theorem 2.1 twice, yielding

$$\begin{aligned}8\frac{dc}{dt} I_\phi(c) \frac{dc}{dt} &= JS(\phi(x), \phi(x + \Delta x)) \\ &= JS(p(x), p(x + \Delta x)) \\ &= 8\frac{dc}{dt} I_p(c) \frac{dc}{dt},\end{aligned}$$

which implies $I_\phi(x) = I_p(x)$. □

Fixing the aforementioned embeddings to embeddings generated using PHATE, we exploit the fact that

$$\begin{aligned}\|\text{PHATE}_{JSD}(x) - \text{PHATE}_{JSD}(y)\|_2 \\ = JSD(P(x,\cdot), P(y,\cdot)),\end{aligned} \tag{7}$$

meaning we can use the loss

$$\begin{aligned}L(\phi) = \mathbb{E}_{x\sim p_{\text{data}}} \Big\| \sqrt{JS(\phi(x), \phi(y))} \\ - \|\text{PHATE}_{JSD}(x) - \text{PHATE}_{JSD}(y)\|_2 \Big\|_2^2,\end{aligned} \tag{8}$$

which motivates the use of PHATE with a Jensen-Shannon divergence MDS step. Overall, this analysis also connects the dimensionality reduction method PHATE with neural FIM in that the former is essentially a discrete version of the latter.

3.2. Geodesic optimization with Neural ODEs

Using the Neural FIM we can compute various Riemannian quantities to describe and understand our dataset. The most important quantity is the geodesic (manifold-intrinsic) distance between datapoints using the FIM. For the FIM the length of the geodesic is also known as the Fisher-Rao distance. In order to compute this we use a neural ODE that optimizes over all paths between datapoints in order to minimize path length.

Given a Riemannian manifold $(\mathcal{M}, g)$ the length of a C^1-curve $\gamma\colon [a,b] \to \mathcal{M}$ is $L(\gamma) = \int_a^b \sqrt{(\frac{d\gamma}{dt})^T \cdot g_{\gamma(t)} \cdot \frac{d\gamma}{dt}}\, dt$. For two distributions p_{θ_1} and p_{θ_2}, a path from p_{θ_1} to p_{θ_2} can be obtained by parameterizing a function $\frac{d\gamma}{dt} = f_\theta(t, \gamma)$ in parameter space so that

$$\hat{\theta}_2 = \gamma(b) = \theta_1 + \int_a^b f_\theta(t, \gamma(t))\, dt$$

Among all the paths parameterized by f_θ, penalizing the length of the curve and the prediction loss $||\hat{\theta}_2 - \theta_2||_2$ gives the geodesic path, i.e.,

$$\arg\min_\theta \ \lambda \|\hat{\theta}_2 - \theta_2\|_2^2 + \int_a^b \sqrt{f_\theta^T \cdot g_{\gamma(t)} \cdot f_\theta}\, dt. \qquad (9)$$

Thus this network queries the neural FIM network in order to optimize path length based on the FIM.

4. Empirical Results

In this section, we provide empirical results of our method. First, we provide a practical use case of the FIM, in selecting parameters for PHATE (Moon et al., 2019). Specifically, PHATE and other diffusion-based methods use two prominent parameters, one that describes the bandwidth of the Gaussian (or related) kernel and another that describes the time of diffusion. We show how the FIM can be used to explore the parameter space by rendering this as a statistical manifold. Second, we apply neural FIM to single cell and toy datasets to showcase information about the data revealed by neural FIM including local volume, trace and eigenspectrum of the metric tensor at individual datapoints. Finally we demonstrate applications of our neural ODE by computing geodesics, which involves optimizing over path lengths of curves computed by the FIM.

4.1. Parameter selection for diffusion potentials with FIM

As described in Sec. 2.1, data diffusion is a powerful framework for exploring the manifold-intrinsic structure of a dataset based on exploring the data through an a Gaussian affinity matrix K which is then normalized Markovian random walk process P and powered to a diffusion time scale t to mimic different steps of random walks. In PHATE, these parameters have significant effects on the visualization (See Figure 2). Here we create a 2-dimensional statistical manifold consisting of parameters σ corresponding to the bandwidth of the Gaussian Kernel, and the diffusion time scale, t. Though we only discuss the diffusion potential matrix rendered by PHATE, the reader should note that using the lens of the FIM for various transformations of point cloud data to notions of similarity or distance could be of interest to practitioners.

We consider the FIM described in Equation 5 with respect to the bandwidth and diffusion time scale parameters, $I_\Psi(\sigma, t)$, where $\Psi : \mathbb{R}^N \to \mathbb{R}^{N\times N}$ is used to construct the diffusion potential matrix. To achieve this, we first generate point cloud data using the tree dataset available in the PHATE package. We then subsampled points randomly and generate the diffusion potential matrix $\mathbf{P}^t_{ij}$ using techniques from Sec. 2.1. To understand how $\theta = (t, \sigma)$ affect the construction of the diffusion potential matrix, we compute the volume of the $[2 \times 2]$ FIM using $V(M) = \int_M \sqrt{|det(I_\Psi(\sigma, t))|}\, d\theta$ for each combination of $\theta_k = (t_k, \sigma_m)$ for a finite range of values $t = [1, 15]$, $m = [50, 150]$. In Figure 2A, we show four different PHATE embeddings of the data corresponding to different parameter selections. We generally see that more details of branches are available in the embedding at lower values of t, and that higher values of bandwidth in this range retain more of the geometric structure. Thus, differences in these parameters have marked effects on the embedding.

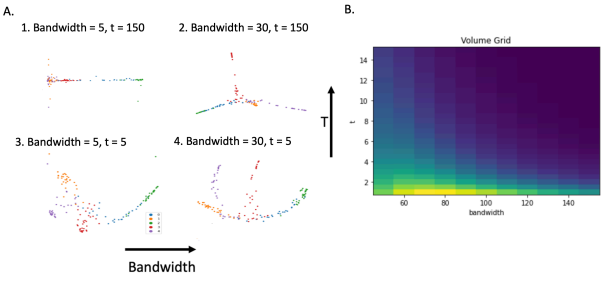


Figure 2. A. PHATE embeddings of the same artificial tree data with different bandwidth σ and t parameters. B. Volume of FIM with respect to t and σ.

In Figure 2B we show a heat map of the FIM volume for different parameters of $\theta_k = (t_k, \sigma_k)$. From inspection, one can see certain combinations of t and σ yield the highest volume. In particular, brighter regions of the volume grid reveal the combinations of t and σ that undergo the most change from the point cloud space to the diffusion potential space while the converse is true for darker regions. This aligns with our intuition about how t and σ affect the diffusion potential construction of the graph: since the diffusion time scale and the bandwidth both incorporate trade-offs between

local and global structure depending on their magnitude, we expect there to be a finite range in the parameter space where this trade-off is optimal. In this case, we discover that this range is between 60 and 90 for σ and between 0 and 4 for t. We also notice that the influence of the bandwidth depends on t; for smaller, t a change of the bandwidth results in a larger change on the diffusion probabilities.

4.2. Neural FIM Embeddings

Here, we deploy neural FIM on three datasets: 1) a toy tree dataset generated similarly to the one above, 2) a single cell mass cytometry dataset of induced pluripotent stem cell (IPSC) reprogramming (Zunder et al., 2015) containing 220450 cells and 33 features, 3) a single cell RNA-sequencing dataset measuring peripheral blood mononucleocyte cells (i.e., immune cells) from a healthy donor (publicly available on the 10x website) (10x Genomics, 2019) containing 2638 cells and 1838 features. For each dataset, we compute the FIM g for each dataset, and we explore the point-wise trace $tr(g)$ and volume $\sqrt{|det(g)|}$ to understand manifold-intrinsic structure and geometric properties of our datasets. The embeddings for each dataset are generated by applying PHATE with a JSD between rows of the diffusion potential matrix in the ambient space (see 11. for comparison between PHATE and PHATE-JSD).

To obtain the continuous FIM for each dataset, we first compute the diffusion operator matrix $\mathbf{P}(x, \cdot) \in R^{n \times n}$ for each batch of point cloud data. We then embed each point in the batch $x_i \in R^d$ using the encoding network, neural-FIM $\phi : R^d \to R^m$ where d is the original dimension of the point cloud data and m is the last dimension of the encoder. Next, train the neural FIM using the loss defined in Equation 6.

We can then compute the Jacobian $\mathbf{J}(x_i) \in R^{m \times d}$ for each point of the network output $x_i \in R^m$ with respect to the input coordinates. An FIM $I_\phi(x_i) \in R^{n \times n}$ can then be computed (using Equation 5) for any point input to neural-FIM, thus yielding a continuous FIM for the manifold. Crucially, this allows one to compute information-theoretic and geometric quantities such as divergences (infinitesimally), volume, length, and relatedly—geodesics.

4.2.1. Toy Data

The artificial tree dataset we use for this was randomly generated using a built-in function in the PHATE package (Moon et al., 2019) which allows one to generate random trees by specifying the number of branches and the number of dimensions.

To validate the FIM computation, we color the embedding of the tree with the volume (Figure 3A) and trace (Figure 3B) which are now accessible with the continuous FIM. Intuitively, the magnitude of the volume and trace are high for regions of the data where there are several directions of progression available for datapoints corresponding to each of the branches. In such areas, the metric tensor has several high eigenvalues. Conversely, the volume and trace will be low in regions of the manifold where there is a single direction of progression such as along individual branches. This relationship can be seen in Figures 3A and 3B—the region of the manifold where the branching occurs (in the center) contains the highest magnitude of volume and trace while the converse holds for sparse areas of the manifold. This variational coloring of trace and volume we observe empirically is a good sanity check that the neural-FIM network is computing what we expect and motivates us to move to real-world examples.

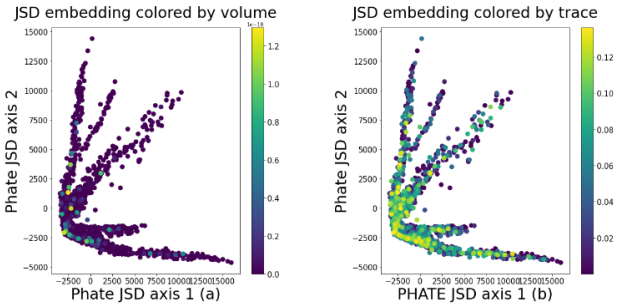


Figure 3. PHATE embedding of tree data colored by the volume (a) and trace (B) of the FIM.

4.2.2. Single Cell Data

In Figures 4A and 4B the trace and the volume are colored on a 2D visualization of an embedding for peripheral blood mononucleic cell (PBMC) dataset. This dataset consists of three major classes of immune cells: T cells, B cells and Monocytes. In each cluster the center of the cluster has highest volume with boundaries having lower volume. Interestingly boundaries that are at the edges of the data (pointing away from other clusters) have even lower volume. Again, the volume seems to indicate potential choices for traveling along the manifold. Hence, the FIM can be used as a tool for illuminating regions of the manifold that retain the most informative components of the manifold.
The same analysis was carried out for Induced Pluripotent Stem Cell (IPSC) data (Zunder et al., 2015) measured using a different single cell technology: mass cytometry, which measures protein abundances. In this dataset, fibroblasts are being reprogrammed into pluripotent stem cells—a process that reverses natural differentiation (potentially for therapeutic purposes). The neural FIM embedding correctly shows a 'Y' shape corresponding to the two branches described in (Zunder et al., 2015). One branch is successfully reprogramming and the other corresponds to failed reprogramming. We again embedded points using neural-FIM on the IPSC data and color by volume and trace in Figures 5A and 5B, respectively. Here, the trace colored along the manifold

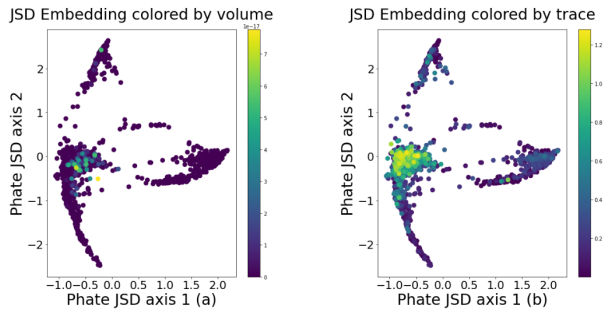


Figure 4. PHATE embedding of pbmc data colored by the volume (a) and trace (b) of the FIM.

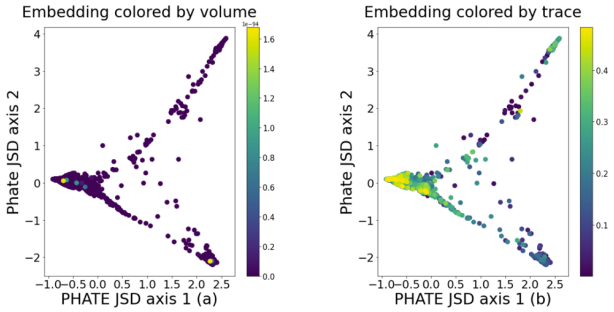


Figure 5. PHATE embedding of IPSC data colored by the volume (a) and trace (b) of the FIM.

reaches its highest magnitude where there are more axes of potential change and its lowest magnitude along the edges and tails where the datapoints do not have many directions to go. We also show the eigenspectrum of the FIM for all embeddings in Figures 6A, 6B, and 6C which can have utility for discerning the number and index of relevant axes of information during the neural-FIM mapping. Using the same line of reasoning, we can look at the decay of the FIM eigenspectra for points located on different regions (e.g. sparse vs. dense) as well as the eigenvectors to extract insightful information about point cloud data.

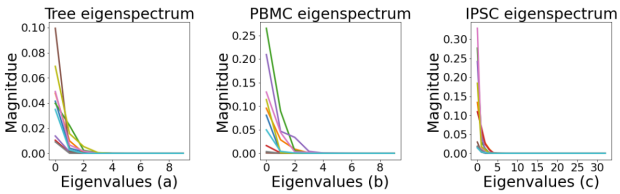


Figure 6. NeuralFIM eigenspectrums for different datasets: (a) Tree (b) PBMC (c) IPSC.

4.3. Learning Geodesics using Neural ODEs

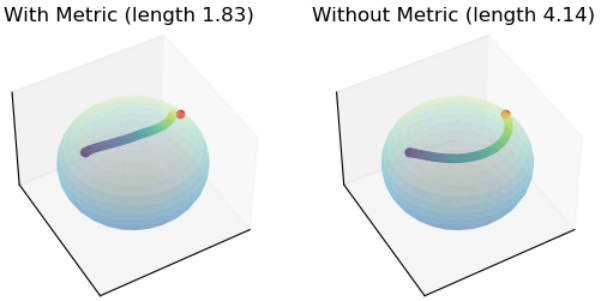
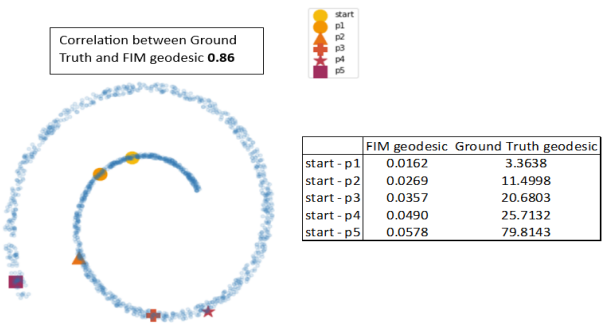


Figure 7. Learned paths from a Neural ODE, the path with the spherical metric learn a curve closer to the geodesic.

We test the training objective defined in Equation 9, to learn the geodesic on a sphere of radius one. In spherical coordinates $(\theta, \psi) \in [0, \pi] \times [0, 2\pi]$ we know the metric $ds^2 = d\theta^2 + \sin^2\theta d\psi^2$, and can thus compute the length of any path on the sphere. We train a neural ODE with three layers of width 64 and SeLU activation function between each layer. We approximate the integration with the Runge-Kutta solver of order four. In Figure 7 we learn the path between two points above the equator $(\pi/4, 0)$ to $(\pi/4, \pi)$; the geodesic passes closer to the north pole. We see that the path of the neural ODE trained to minimize the length indeed finds the right geodesic, while the one trained only to minimize the MSE with the final time point learns a longer path (θ appears to be constant along the path).



	FIM geodesic	Ground Truth geodesic
start - p1	0.0162	3.3638
start - p2	0.0269	11.4998
start - p3	0.0357	20.6803
start - p4	0.0490	25.7132
start - p5	0.0578	79.8143

Figure 8. Ground Truth and NeuralFIM geodesics between a fixed point and 5 randomly selected points on swiss roll dataset

In Figure 8 we validate our method of computing geodesics with the learned FIM's against ground truth geodesics on a swiss roll. For a quick sanity check, one can see that increasing the path length from the start to end point on the swiss roll corresponds to an increase in geodesic magnitude for the FIM and ground truth geodesic. Additionally, we see the FIM geodesic is strongly correlated with the ground truth geodesic.

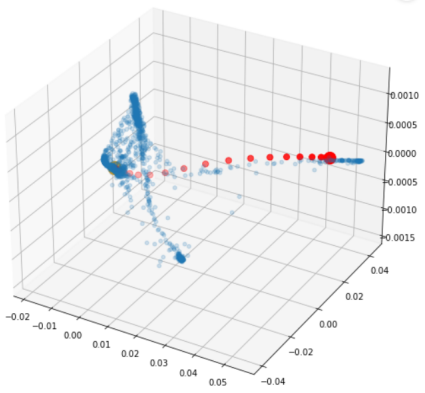


Figure 9. Learned neural ODE geodesic from the IPSC data connecting an initial cell to its final reprogrammed destination. Notably, the trajectory goes along the manifold.

Figure 9 shows a geodesic path that goes from a fibroblast cell (at the intersection of the branches) towards a reprogramming endpoint. We see that with only the endpoints specificed the neural ODE is able to find the path of reprogramming, i.e., as the geodesic path along the manifold. Generally, this could be highly useful for finding differentiation and progression paths from single cell data.

5. Conclusion

Here we presented neural FIM, a novel method for learning a Riemannian metric from high dimensional point cloud data. We utilize FIM as a metric for data points represented by data diffusion probability distributions. Such distributions are computed via a Markovian diffusion operator which is used in diffusion maps, PHATE, diffusion pseudotime and other popular data science techniques. Neural FIM then allows us to compute underlying manifold information such as volume, and geodesic distances in this space in a way that is extensible to new datapoints. To compute geodesics in data space we introduce an auxiliary neural ODE network that minimizes length computed using the FIM on learned curve between two datapoints. We showcase neural FIM on PHATE parameter selection, and in finding the underlying manifold of toy data as well as single cell data.

6. Acknowledgements

This research was enabled in part by compute resources provided by Mila (mila.quebec) and Yale. It was partially funded and supported by ESP *Mérite* [G.H.], CIFAR AI Chair [G.W.], NSERC Discovery grant 03267 [G.W.], NIH grants (1F30AI157270-01, R01HD100035, R01GM130847, R01GM135929) [G.W.,S.K.], NSF Career grant 2047856 [S.K.], the Chan-Zuckerberg Initiative grants CZF2019-182702 and CZF2019-002440 [S.K.], the Sloan Fellowship FG-2021-15883 [S.K.], and the Novo Nordisk grant GR112933 [S.K.]. The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies. The funders had no role in study design, data collection & analysis, decision to publish, or preparation of the manuscript.

References

10x Genomics. Pbmcs from c57bl/6 mice (v1, 150x150), single cell immune profiling dataset by cell ranger 3.1.0, 2019.

Amari, S.-i. *Information geometry and its applications*, volume 194. Springer, 2016.

Arwini, K. A. and Dodson, C. T. *Information geometry*. Springer, 2008.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Bengio, Y., Paiement, J.-f., Vincent, P., Delalleau, O., Roux, N., and Ouimet, M. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Advances in neural information processing systems*, 16, 2003.

Bhaskar, D., MacDonald, K., Fasina, O., Thomas, D., Rieck, B., Adelstein, I., and Krishnaswamy, S. Diffusion curvature for estimating local curvature in high dimensional data. *arXiv preprint arXiv:2206.03977*, 2022.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Bunte, K., Haase, S., Biehl, M., and Villmann, T. Stochastic neighbor embedding (sne) for dimension reduction and visualization using arbitrary divergences. *Neurocomputing*, 90:23–45, 2012.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Cheng, X., Rachh, M., and Steinerberger, S. On the diffusion geometry of graph laplacians and applications. *Applied and Computational Harmonic Analysis*, 46(3): 674–688, 2019.

Coifman, R. R. and Lafon, S. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006. ISSN 10635203. doi: 10.1016/j.acha.2006.04.006.

Cover, T. M. and Thomas, J. A. Elements of information theory second edition solutions to problems. *Internet Access*, pp. 19–20, 2006.

Crooks, G. E. Measuring thermodynamic length. *Physical Review Letters*, 99(10):100602, 2007.

Cybenko, G. V. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

Dadkhahi, H., Duarte, M. F., and Marlin, B. M. Out-of-sample extension for dimensionality reduction of noisy time series. *IEEE Transactions on Image Processing*, 26 (11):5435–5446, 2017.

De Domenico, M. Diffusion geometry unravels the emergence of functional clusters in collective phenomena. *Physical review letters*, 118(16):168301, 2017.

Haghverdi, L., Büttner, M., Wolf, F. A., Buettner, F., and Theis, F. J. Diffusion pseudotime robustly reconstructs lineage branching. *Nature methods*, 13(10):845–848, 2016.

He, J., Ding, L., Jiang, L., Li, Z., and Hu, Q. Intrinsic dimensionality estimation based on manifold assumption. *Journal of Visual Communication and Image Representation*, 25(5):740–747, 2014.

Huguet, G., Magruder, D. S., Tong, A., Fasina, O., Kuchroo, M., Wolf, G., and Krishnaswamy, S. Manifold interpolating optimal-transport flows for trajectory inference. In *NeurIPS*, 2022.

Lafferty, J., Lebanon, G., and Jaakkola, T. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6(1), 2005.

Lauritzen, S. L. Statistical manifolds. *Differential geometry in statistical inference*, 10:163–216, 1987.

Law, M. H. and Jain, A. K. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE transactions on pattern analysis and machine intelligence*, 28(3): 377–391, 2006.

Li, W. and Rubio, F. J. On a prior based on the wasserstein information matrix. *Statistics & Probability Letters*, 190: 109645, 2022.

Lin, A. T., Li, W., Osher, S., and Montúfar, G. Wasserstein proximal of gans. In *Geometric Science of Information: 5th International Conference, GSI 2021, Paris, France, July 21–23, 2021, Proceedings*, pp. 524–533. Springer, 2021.

Lin, T. and Zha, H. Riemannian manifold learning. *IEEE transactions on pattern analysis and machine intelligence*, 30(5):796–809, 2008.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

Mika, S., Schölkopf, B., Smola, A., Müller, K.-R., Scholz, M., and Rätsch, G. Kernel pca and de-noising in feature spaces. *Advances in neural information processing systems*, 11, 1998.

Moon, K. R., van Dijk, D., Wang, Z., Gigante, S., Burkhardt, D. B., Chen, W. S., Yim, K., van den Elzen, A., Hirn, M. J., Coifman, R. R., Ivanova, N. B., Wolf, G., and Krishnaswamy, S. Visualizing structure and transitions in high-dimensional biological data. *Nat Biotechnol*, 37 (12):1482–1492, 2019.

Nielsen, F. An elementary introduction to information geometry. *Entropy*, 22(10):1100, 2020.

Noguchi, M. Geometry of statistical manifolds. *Differential Geometry and its Applications*, 2(3):197–222, 1992.

Schoeneman, F., Mahapatra, S., Chandola, V., Napp, N., and Zola, J. Error metrics for learning reliable manifolds from streaming data. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 750–758. SIAM, 2017.

Tsitsulin, A., Munkhoeva, M., Mottin, D., Karras, P., Bronstein, A., Oseledets, I., and Müller, E. The shape of data: Intrinsic distance for data distributions. *arXiv preprint arXiv:1905.11141*, 2019.

van der Maaten, L. and Hinton, G. E. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

Zunder, E. R., Lujan, E., Goltsev, Y., Wernig, M., and Nolan, G. P. A continuous molecular roadmap to ipsc reprogramming through progression analysis of single-cell mass cytometry. *Cell stem cell*, 16(3):323–337, 2015.

A. Sensitivity Analysis of neuralFIM hyperparameters

Here, we complete a sensitivity analysis of a selection of neuralFIM hyperparameters on the tree dataset. We perturb the k-nearest neighbors (kNN), the noise level, and the Encoder Dimensions to understand whether our is robust with respect to the aforementioned hyperparameters. See Table 1 and Figure 10 for empirical results.

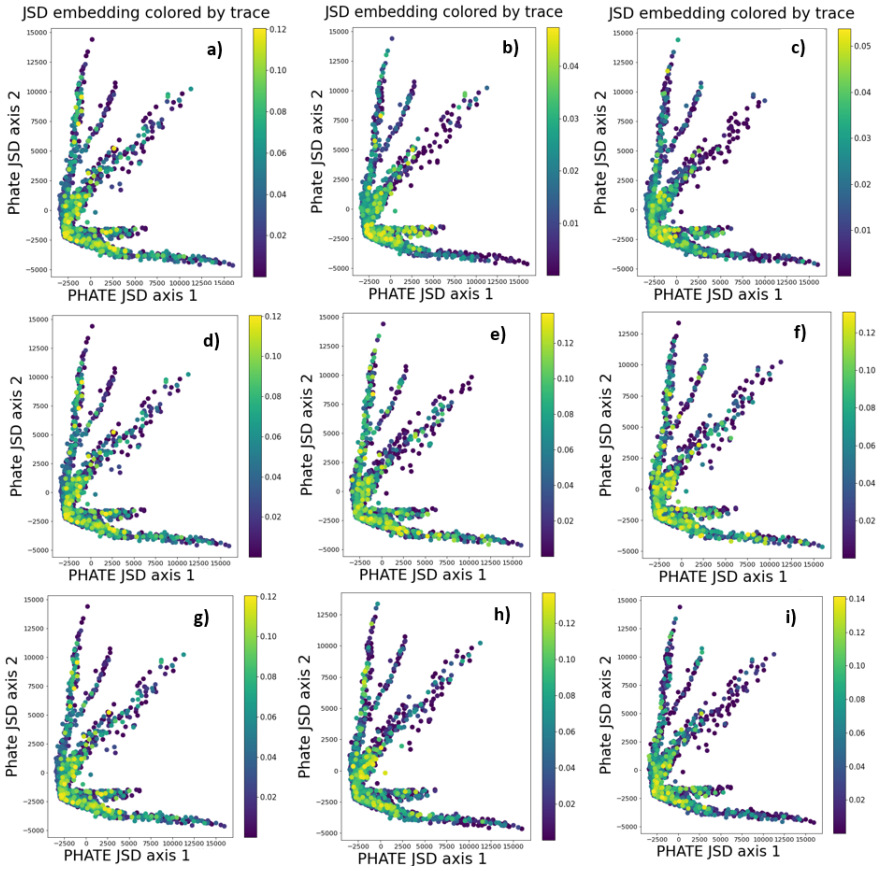


Figure 10. Visualization of PHATE-JSD embedding of tree dataset colored by FIM trace computed with NeuralFIM. We vary the kNN for the PHATE kernel, the noise level added to the input dataset and the dimensions of the autoencoder. One can readily see that neuralFIM is robust with respect to the perturbed hyperparameters. In row 1 we perturb the kNN across figures a-c [knn=5,10,15], respectively, in row 2 (d-f) we perturb the noise level added to the data input to the autoencoder [noise level = 0.0005,0.0010,0.0015], and in row 3 (g-i) we perturb the encoder dimension(ED) [ED=100,100,50; ED=100,80,30; ED=100,70,20].

Table 1. Sensitivity Analysis: Correlation of FIM trace between encoder noise, embedding dimension, and KNN for tree dataset

KNN				**Noise Level**				**Embedding Dimension**			
	5	*10*	*15*		*0.0005*	*0.0010*	*0.0015*		*20*	*30*	*50*
5	1	0.9972	0.9989	*0.0005*	1	0.9981	0.9990	*20*	1	0.9971	0.9944
10	0.9972	1	0.9985	*0.0010*	0.9981	1	0.9994	*30*	0.9971	1	0.9985
15	0.9989	0.9985	1	*0.0015*	0.9990	0.9994	1	*50*	0.9944	0.9985	1

A.1 Algorithm

Below we describe the probability distribution constructed for FIM computation.

Algorithm 1 Phate Fisher Information Distribution

Input: $N \times d$ dataset X, matrix diffusion time t
Returns: $N \times N$ diffusion potential matrix, $\boldsymbol{U}$
$\boldsymbol{D}_{ij} \leftarrow \|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2$
$\boldsymbol{A} \leftarrow \text{kernel}(\boldsymbol{D})$
$\boldsymbol{Q} \leftarrow \text{Diag}(\boldsymbol{A}\mathbf{1})$
$\boldsymbol{K} \leftarrow \boldsymbol{Q}^{-1}\boldsymbol{A}\boldsymbol{Q}^{-1}$
$\boldsymbol{P} \leftarrow \text{RowNormalize}(\boldsymbol{K})$
/ Here* log *is applied elementwise, power is matrix power.* */
return $\boldsymbol{U} \leftarrow \log(\boldsymbol{P}^t)$

A.2 Connection between FIM and KL divergence

In (Cover & Thomas, 2006) the Fisher Information Metric is derived as the second derivative of KL-divergence. To derive this the authors consider two probability distributions $P(x)$ and $P(y)$ that are infinitesimally close to one another. $P(y) = P(x) + \sum_j \Delta x_j \frac{\partial P}{\partial x_j}$ where Δx_j is an infinitesimally small change of x in the j direction.

Since KL-divergences are 0 when two distributions are equal to one another, they use a second order Taylor expansion of the KL-divergence as given by:

$g_x(y) = KL(P(x)||P(y) = \frac{1}{2}\sum_{i,j} \Delta x_i \Delta x_j g_{ij}(x)$

A.3 JS Distance PHATE

Typically the PHATE dimensionality reduction method works along the following steps:

- Compute diffusion operator P from data using an alpha-decay kernel given in (Moon et al., 2019).
- Compute potential distances which are M-divergences between rows of the diffusion operator $P(x, \cdot)$ as given by $pdist(i, j) = \sqrt{\sum_k (\log(P^t(i, k) - P^t(j, k))}$ between points i and j.
- Use the potential distance matrix as input to metric MDS to reduce to two dimensions.

To train the neural FIM, we instead replace the M-divergence in PHATE with JS distance given by:

$$JS(\boldsymbol{P}_n^t(i,\cdot), \boldsymbol{P}_n^t(j,\cdot)) :=$$
$$\frac{1}{2}KL((\boldsymbol{P}_n^t(i,\cdot)||M) + KL((\boldsymbol{P}_n^t(j,\cdot)||M),$$

where $M := (1/2)(\boldsymbol{P}_n^t(i,\cdot) + \boldsymbol{P}_n^t(j,\cdot))$.

Then we use the same metric MDS steps to reduce to an arbitrary k, though not typically 2 dimensions. We note that this preserves manifold structure as well as or better than the originally proposed M-divergences. In Figure 11 we show embeddings of artificially generated tree-structured data with original PHATE on the left and neural FIM trained with JSD-PHATE on the right. The embeddings look similar with the JSD embedding looking even more denoised than the original PHATE embedding.

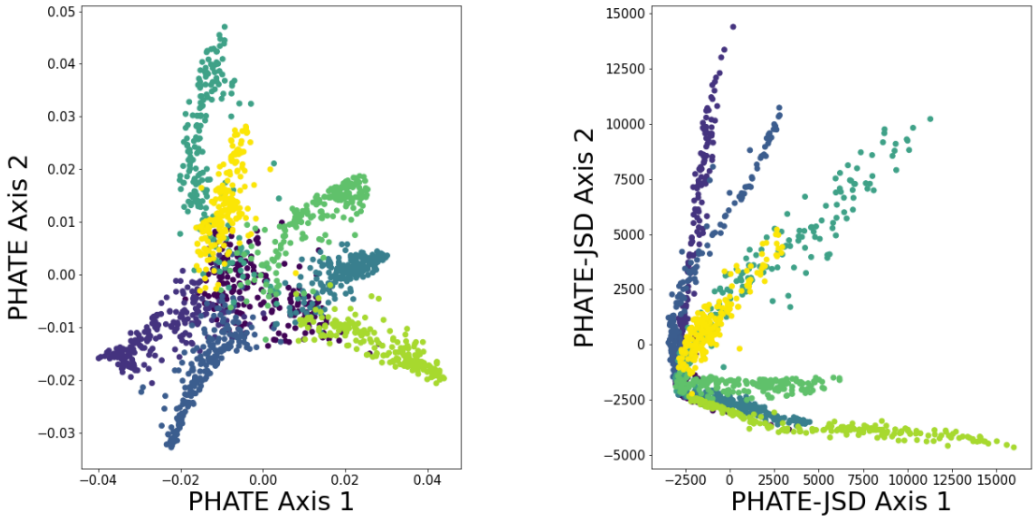


Figure 11. Embeddings of artificially generated tree data with PHATE and neural FIM trained with JSD-PHATE

A.4 Experimental details

Each dataset was run with the same neural network parameters: Encoding Layers = [100,100,50] (for k=50); [100,80,30] (for k=30); [100,70,20] (for k=20) where k = latent dimensions, 150 epochs, ReLu activation between encoding layers, and using the AdamW optimizer with learning rate = 1e-4. For the neuralODE, we use 3 hidden layers [64,64,64] and use Runge-Kutta for the ODE solver. For the experimental results, we use 20 time steps between start and end point and train for 250 epochs again using the AdamW optimizer with learning rate = 1e-4.