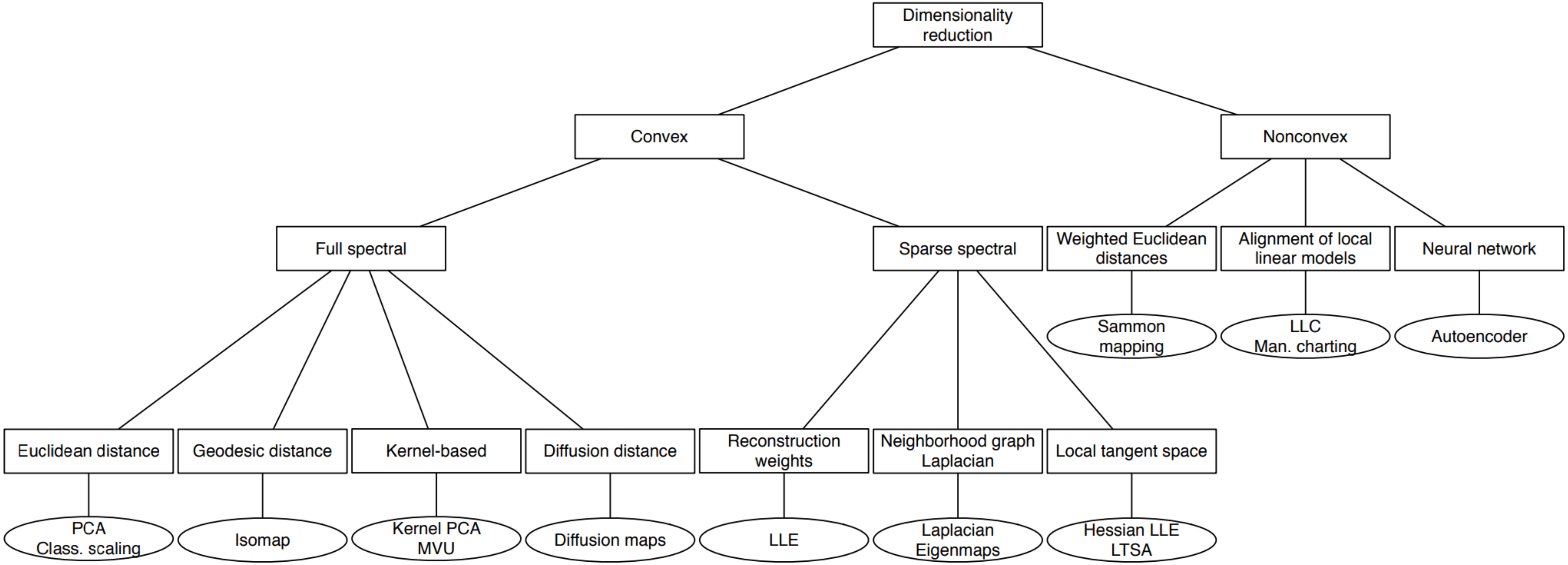
NONLINEAR DIMENSIONALITY REDUCTION

Christian Bueno
University of California, Santa Barbara

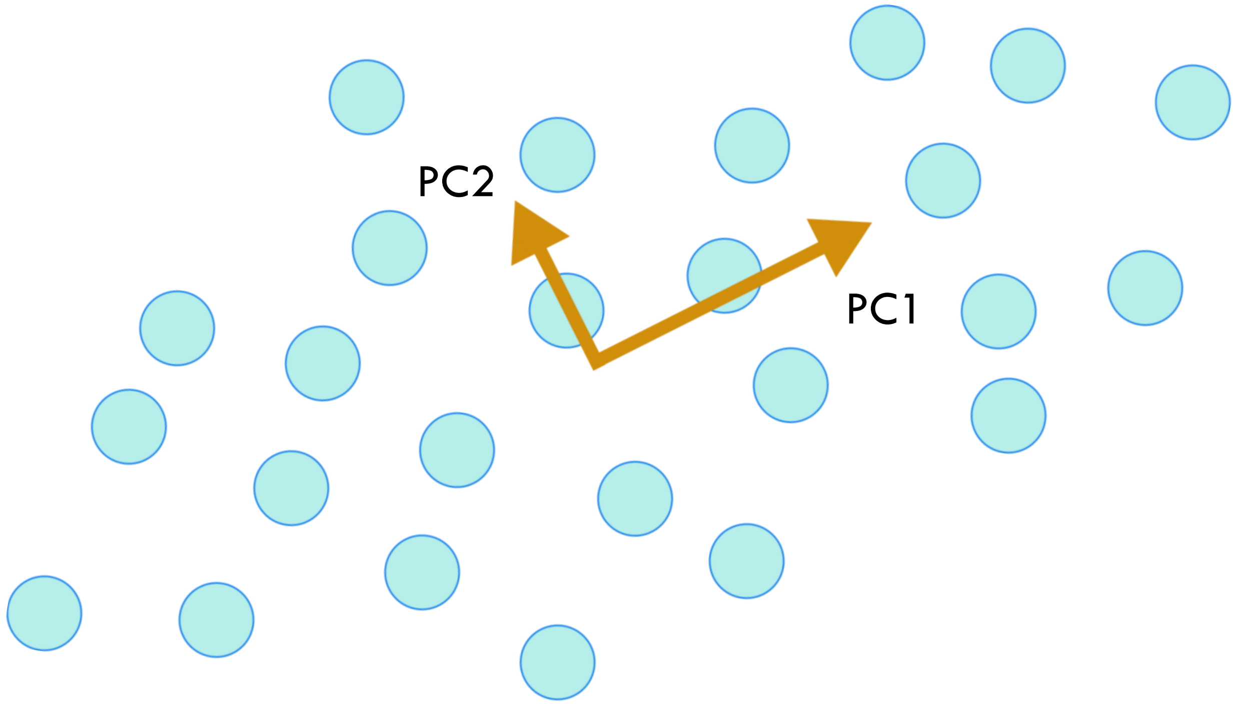
MIT Center for Brain, Minds + Machines Tutorial (09/17/20)

NLDR Zoo



van der Maaten, Postma, van der Herik (2009). Dimensionality Reduction: A Comparative Review

Principal Component Analysis (PCA)



Introduced by Pearson (1901) and named/rediscovered by Hotelling (1930)

Idea: Finds linear subspace that preserves the most variation in the data.

PCA via Covariance Eigendecomposition

Assuming centered data $\{x_1, \dots, x_n\} \subseteq \mathbb{R}^p$ the $p \times p$ sample covariance matrix is

$$C = \frac{1}{n-1} X^T X = \frac{1}{n-1} \sum_{i=1}^{n} x_i x_i^T = V \Lambda V^T \approx V_d \Lambda_d V_d^T$$

C is symmetric ($C^T = C$) and positive semi-definite ($u^T C u \geq 0$ for all $u \in \mathbb{R}^p$).

Has orthonormal basis (ONB) of eigenvectors $v_1, v_2, \dots, v_p$ with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_q \geq \lambda_{q+1} = \cdots = \lambda_p = 0$.

v_i is the i-th **principal axis/direction** and projection of x onto v_i is the i-th **principal component** of x. The variance of dataset in direction v_i is λ_i.

To reduce to d dimensions, keep PC-1 up to PC-d (where $d \leq q$).

$$X = \begin{pmatrix} — & x_1^T & — \\ — & x_2^T & — \\ & \vdots & \\ — & x_n^T & — \end{pmatrix} \qquad n \times p$$

$$\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_q) \qquad q \times q$$

$$V = \begin{pmatrix} | & & | \\ v_1 & \cdots & v_q \\ | & & | \end{pmatrix} \qquad p \times q$$

PCA via SVD is Faster and Better

$$X = \underbrace{U\Sigma}_{\text{Principal Components}} V^T$$

PCA Projection

$$C = \frac{X^T X}{n-1} = V \Lambda V^T \approx V_d \Lambda_d V_d^T$$

$$X = U \Sigma V^T$$

$x \in \mathbb{R}^p$ (Data space) $\mapsto \mathbb{R}^d$ (PC space)

$$V_d^T x = \begin{pmatrix} v_1^T x \\ | \\ v_d^T x \end{pmatrix}$$

Out-of-Sample Extension (OoSE)

$y \in \mathbb{R}^d$ (PC space) $\mapsto \mathbb{R}^p$ (Data space)

$$V_d \, y = y_1 v_1 + \ldots y_d v_d$$

"Preimage"

To map all data X down to $\mathbb{R}^d$

$$X V_d = \begin{pmatrix} — & x_1^T V_d & — \\ & \vdots & \\ — & x_n^T V_d & — \end{pmatrix}$$

Using SVD we can see that

$$XV = U \Sigma V^T V = U \Sigma$$

and so the projection by SVD is

$$X V_d = U_d \Sigma_d$$

PCA via Gram Matrix Eigendecomposition

The Gram matrix of $\{x_1, \ldots, x_n\}$ is the $n \times n$ matrix of dot products.

$$K_{ij} = \langle x_i, x_j \rangle = x_i^T x_j \qquad\qquad K = XX^T$$

K is sym. pos. semi-definite (SPSD) $\implies$ has ONB of eigvecs $a_1, \ldots, a_n$ & eigvals $\rho_1 \geq \cdots \geq \rho_n \geq 0$.

Claim: If $Ka = \rho a$ and $\rho \neq 0$ define $v = \frac{1}{\sqrt{\rho}} X^T a$. Then $||v|| = ||a||$ and $Cv = \frac{\rho}{n-1} v$.

Proof:

$$\begin{aligned} \|v\|^2 &= \left(\tfrac{1}{\sqrt{\rho}} X^T a\right)^T \tfrac{1}{\sqrt{\rho}} X^T a \\ &= \tfrac{1}{\rho} a^T X X^T a \\ &= \tfrac{1}{\rho} a^T K a \\ &= a^T a = \|a\|^2 \end{aligned}$$

$$\begin{aligned} Cv &= \left(\tfrac{1}{n-1} X^T X\right)\left(\tfrac{1}{\sqrt{\rho}} X^T a\right) \\ &= \tfrac{1}{n-1} X^T X X^T a \, \tfrac{1}{\sqrt{\rho}} \\ &= \tfrac{1}{n-1} X^T K a \, \tfrac{1}{\sqrt{\rho}} \\ &= \tfrac{\rho}{n-1} X^T a \, \tfrac{1}{\sqrt{\rho}} = \tfrac{\rho}{n-1} \, v \end{aligned}$$

Bottom Line: Eigenpairs of K can be used to build principal directions of C… also K has q positive eigvals like C.

PCA via Gram Matrix Eigendecomposition

So $Ka_i = \rho_i a_i$ and $v_i = \frac{1}{\sqrt{\rho_i}} X^T a_i$ for $i = 1, \ldots, q$. In matrix form that is

$$KA = AR \qquad\qquad V = X^T \begin{pmatrix} | & & | \\ a_1 & \cdots & a_q \\ | & & | \end{pmatrix} \begin{pmatrix} \rho_1^{-1/2} & & 0 \\ & \ddots & \\ 0 & & \rho_q^{-1/2} \end{pmatrix} = X^T A R^{-1/2}$$

Bonus-1: By above we can directly get principal components.

$$\begin{matrix}\text{Projected} \\ \text{data}\end{matrix} = XV = XX^T A R^{-1/2} = KAR^{-1/2} = ARR^{-1/2} = AR^{1/2}$$

Bonus-2: Eigendecomposition shows rows of $AR^{1/2}$ yield the same Gram matrix.

$$K = XX^T = ARA^T = AR^{1/2}R^{1/2}A^T = (AR^{1/2})(AR^{1/2})^T$$

Torgerson (1958) and Gower (1966)

→ Classical Multi-Dimensional Scaling (MDS)

Question: Given only the Euclidean distances between points $d_{ij} = ||x_i - x_j||$… Can we recover the configuration of points (up to rigid motion)?

Answer: Yes! Observe that

$$d_{ij}^2 = \|x_i - x_j\|^2 = \langle x_i - x_j, x_i - x_j \rangle = \|x_i\|^2 + \|x_j\|^2 - 2\langle x_i, x_j \rangle$$

Removing mean $\mu = \frac{1}{n}\sum_i x_i$ from the data can be achieved by "double-centering"

$$K_{ij} := \langle x_i - \mu, x_j - \mu \rangle = -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{n}\sum_j d_{ij}^2 - \frac{1}{n}\sum_i d_{ij}^2 + \frac{1}{n^2}\sum_{i,j} d_{ij}^2 \right)$$

Amazingly we have the mean-centered Gram matrix K solely in terms d_{ij}! If we use the i-th rows of $AR^{1/2}$ as coordinates for x_i, i.e. letting $AR^{1/2}$ be the $n \times q$ data matrix, then we win.

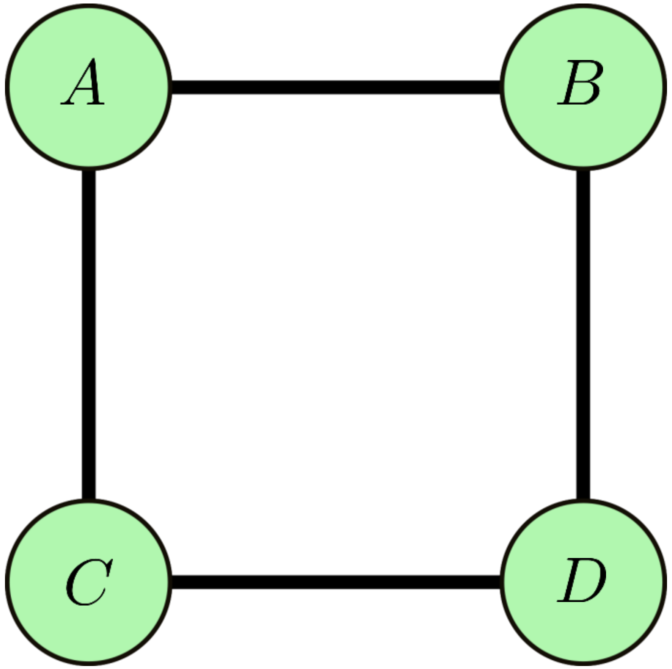
$$x_i = \left(\sqrt{\rho}_1 a_{1i}, \ldots, \sqrt{\rho}_q a_{qi} \right)^T$$

Torgerson (1958) and Gower (1966)

Classical Multi-Dimensional Scaling (MDS)

Not all distances come from Euclidean space!
Perfect Euclidean representation is sometimes impossible

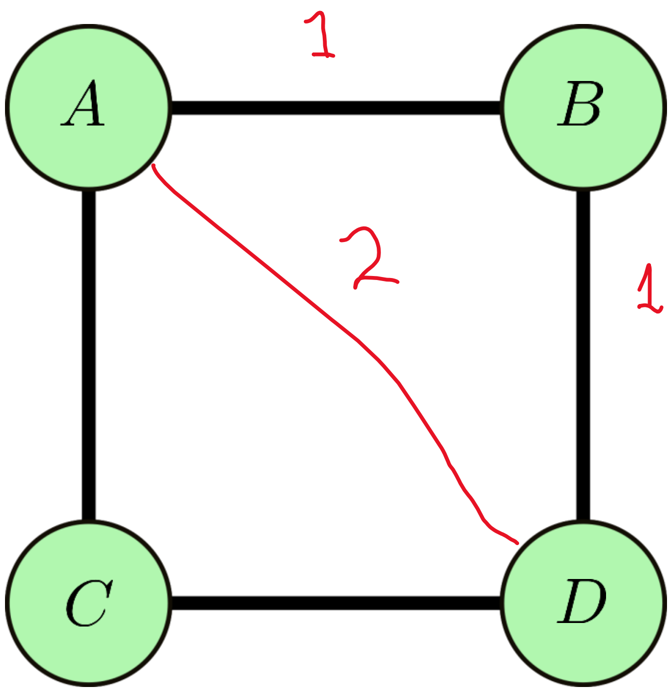
Example: Here each edge is distance 1 and d(A,D)=d(B,C)=2.



Classical Multi-Dimensional Scaling (MDS)

Not all distances come from Euclidean space!
Perfect Euclidean representation is sometimes impossible

Example: Here each edge is distance 1 and d(A,D)=d(B,C)=2.

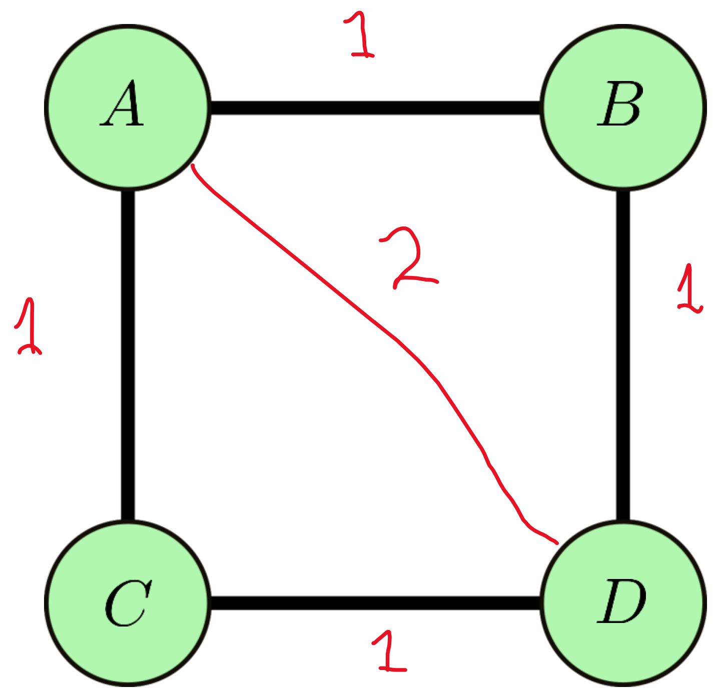


$\Rightarrow B$ is the midpoint of AD

Classical Multi-Dimensional Scaling (MDS)

Not all distances come from Euclidean space!
Perfect Euclidean representation is sometimes impossible

Example: Here each edge is distance 1 and d(A,D)=d(B,C)=2.



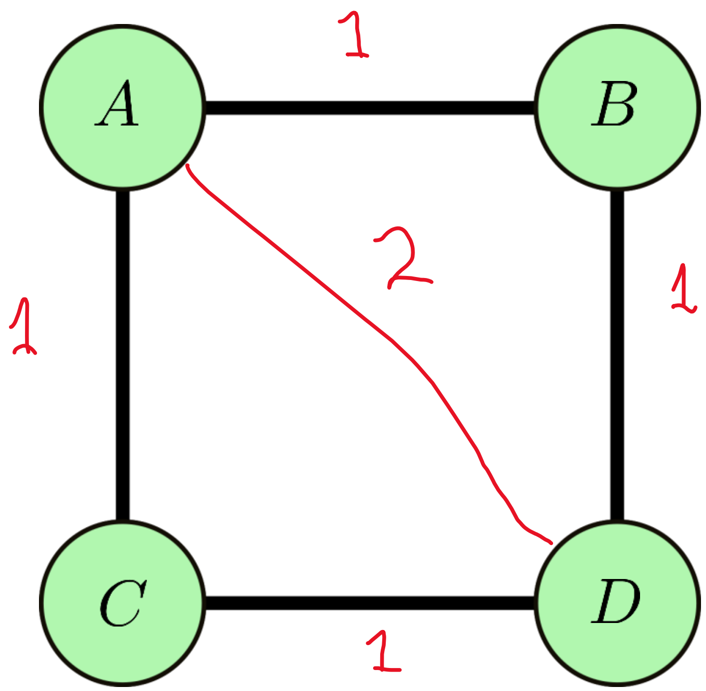
$\Rightarrow$ B is the midpoint of AD

$\Rightarrow$ C is also the midpoint of AD

Classical Multi-Dimensional Scaling (MDS)

Not all distances come from Euclidean space!
Perfect Euclidean representation is sometimes impossible

Example: Here each edge is distance 1 and d(A,D)=d(B,C)=2.



$\Rightarrow B$ is the midpoint of AD

$\Rightarrow C$ is also the midpoint of AD

$\Rightarrow B = C$ even though $d(B, C) = 2$

Contradiction!

Classical Multi-Dimensional Scaling (MDS)

Bonus: The algorithm is still useful even if d_{ij} is non-Euclidean b/c we don't need dot products to know K.

$$K_{ij} = -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{n}\sum_j d_{ij}^2 - \frac{1}{n}\sum_i d_{ij}^2 + \frac{1}{n^2}\sum_{i,j} d_{ij}^2\right)$$

Algorithm: Write $K = ARA^T \approx A_q R_q A_q^T$ where we keep the q largest eigvals>0. Return $A_q R_q^{1/2}$ as data matrix.

Theorem: The d_{ij} can be perfectly realized as Euclidean points if and only if K is SPSD.

Theorem: This algorithm minimizes "*Strain*" which is given by

$$Strain(x_1, \ldots, x_n) = \sum_{i,j=1}^{n} (K_{ij} - \langle x_i, x_j \rangle)^2$$

de Leeuw (1977)

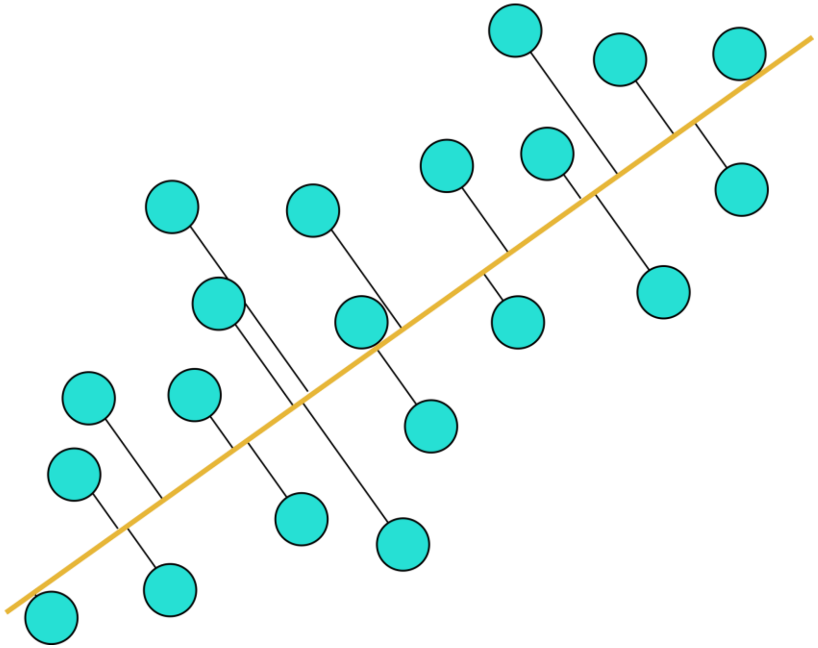
Stress Minimizing MDS

Other approaches directly minimize the "Stress" which is given by

$$Stress(x_1, \ldots, x_n) = \sum_{i,j=1}^{n} (d_{ij} - \|x_i - x_j\|)^2$$

These approaches need to be iterative and there is no closed form solution

PCA via Reconstruction Error

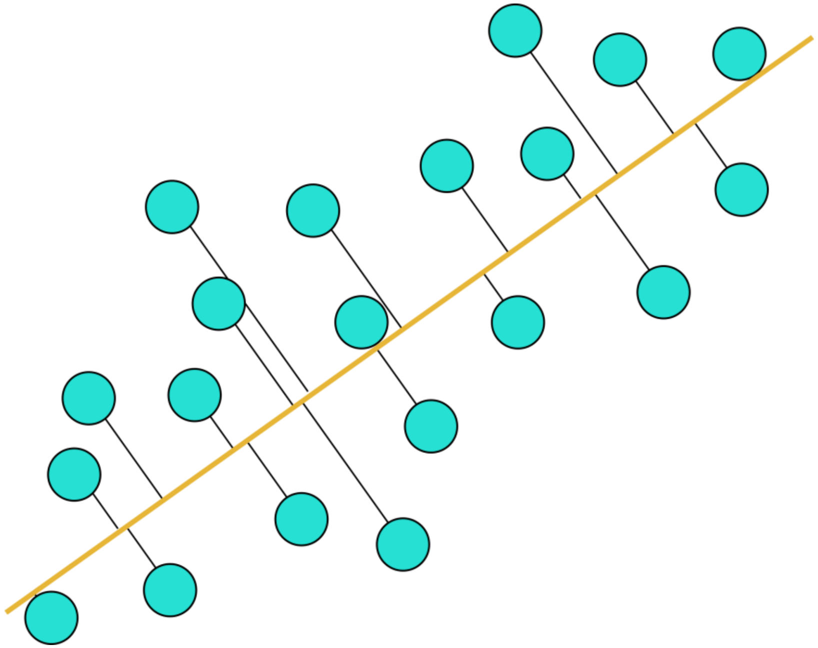


Minimize Reconstruction Error subject to constraint

$$\underset{V}{\text{minimize}} \qquad \sum_i \|x_i - V_d V_d^T x_i\|^2$$

$$\text{Subject to} \qquad V_d^T V_d = I$$

PCA via Reconstruction Error

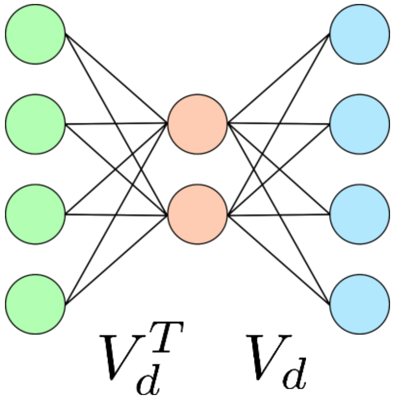


Minimize Reconstruction Error subject to constraint

$$\underset{V}{\text{minimize}} \quad \sum_i \|x_i - V_d V_d^T x_i\|^2$$

$$\text{Subject to} \quad V_d^T V_d = I$$

Linear Autoencoder



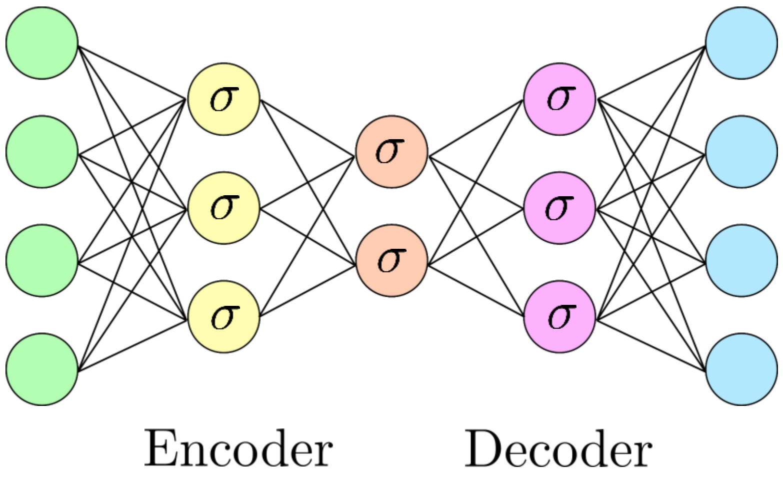
Rumelhart, Hinton, Williams (1986)

→ Deep Autoencoders

Introduce nonlinearities + Minimize reconstruction error.

Pros:
Far more general than PCA. Can be trained online. Out-of-Sample Extension & Preimage are immediate.

Cons:
Lots of hyperparameters (architecture, activations, training, etc). Changing latent dimension means retraining.

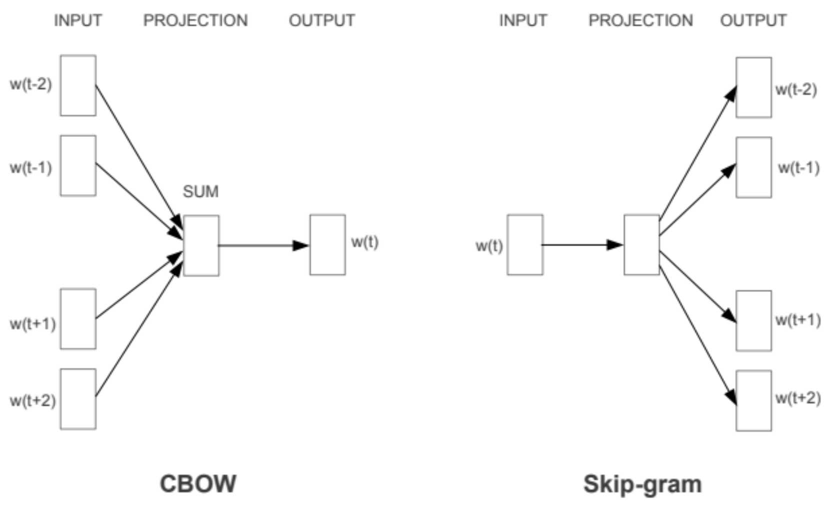


Deep Hidden Representations

Many supervised DNNs discover good hidden representations.

Take the hidden representation of deep networks as embedding.

Examples such as Word2Vec show the wide utility of such approaches.



Mikolov et al (2013)

Nonlinear Dimensionality Enlargement?

$$\Phi : \mathbb{R}^p \to \mathbb{R}^?$$

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle$$

Apply PCA/MDS upstairs

The Kernel Trick

Mercer Condition:
$k: \Omega \times \Omega \to R$ is continuous, symmetric and the matrix $K_{ij} = k(x_i, x_j)$ is SPSD for every choice of $\{x_1, \dots, x_n\}$ in Ω.

Exists Feature Map:

$$\Phi : \mathbb{R}^p \to H$$

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle$$

Schölkopf, Smola, Müller (1998)

Kernel PCA (KPCA)

To do PCA, same argument for Gram matrix works here

Claim: If $Ka = \rho a$ and $\rho \neq 0$ define $v = \frac{1}{\sqrt{\rho}} X^T a$. Then $||v|| = ||a||$ and $Cv = \frac{\rho}{n-1} v$.

(Here $X^T a = a_1 \Phi(x_1) + \cdots + a_n \Phi(x_n)$)

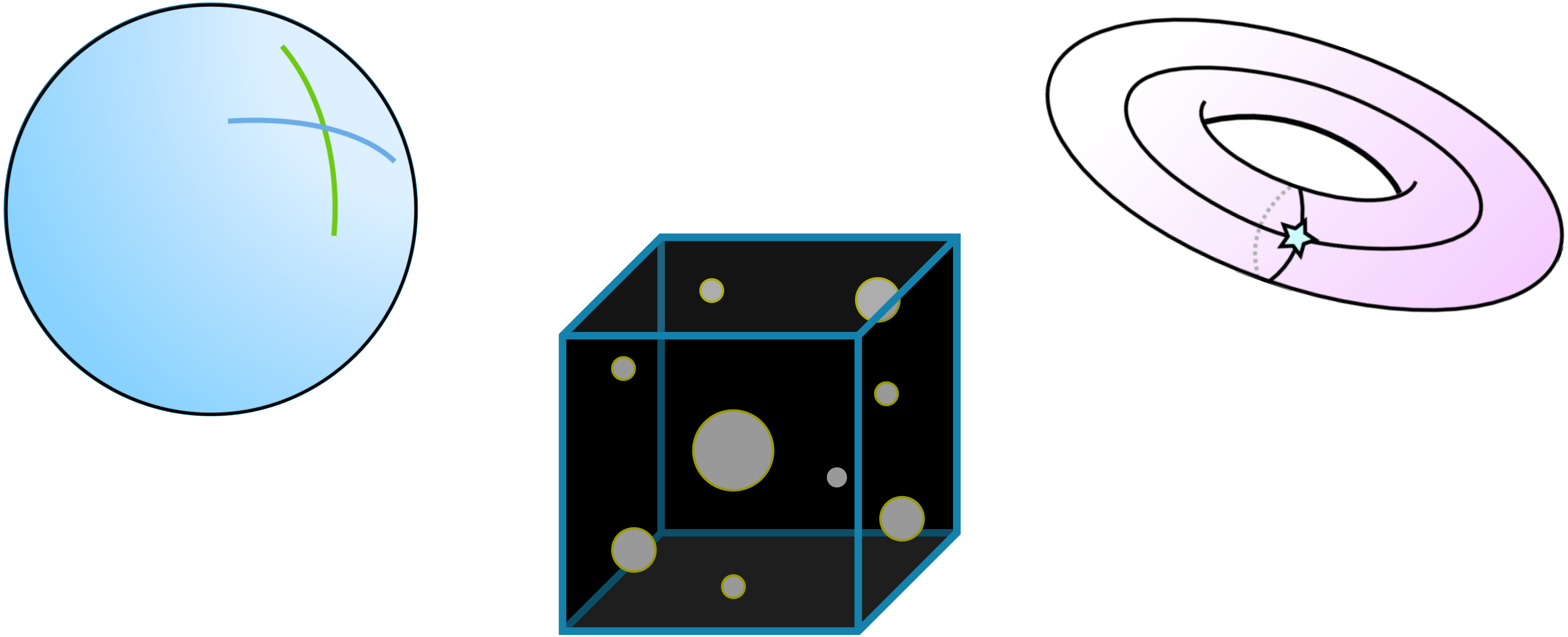
But we must do double centering first

$$K' = K - 1_n K - K 1_n + 1_n K 1_n$$

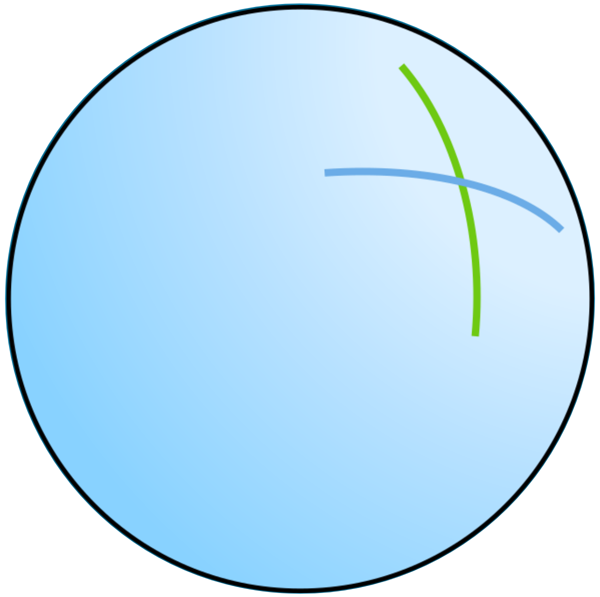
Then correctly projected data is given by $AR^{1/2}$

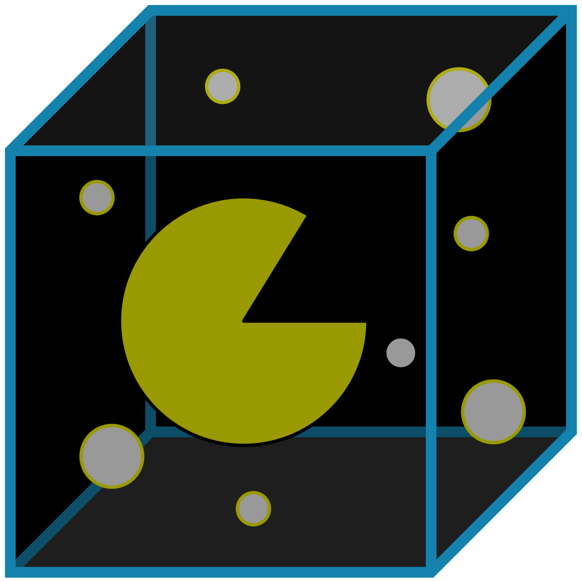
TOPOLOGY AND RIEMANNIAN GEOMETRY

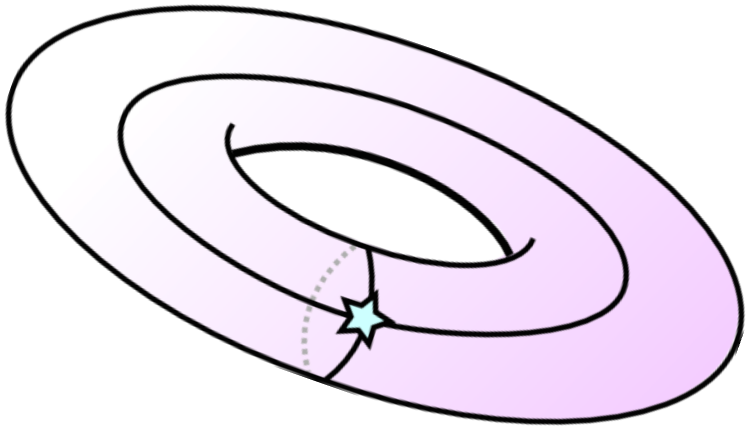
Intrinsic vs Extrinsic Dimension



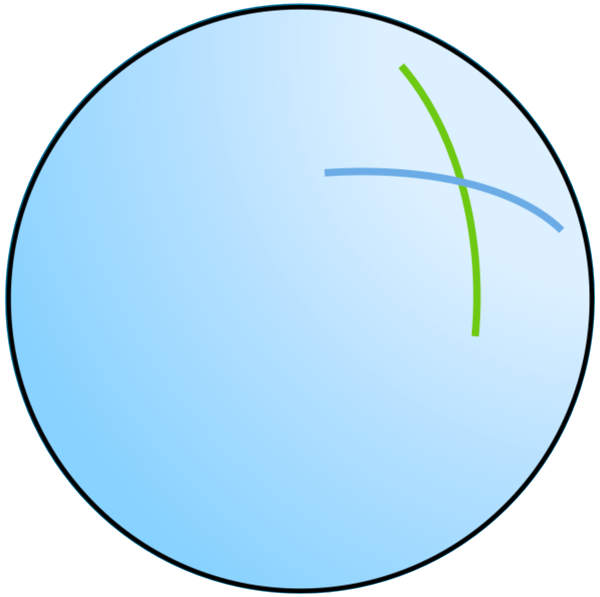
Intrinsic vs Extrinsic Dimension

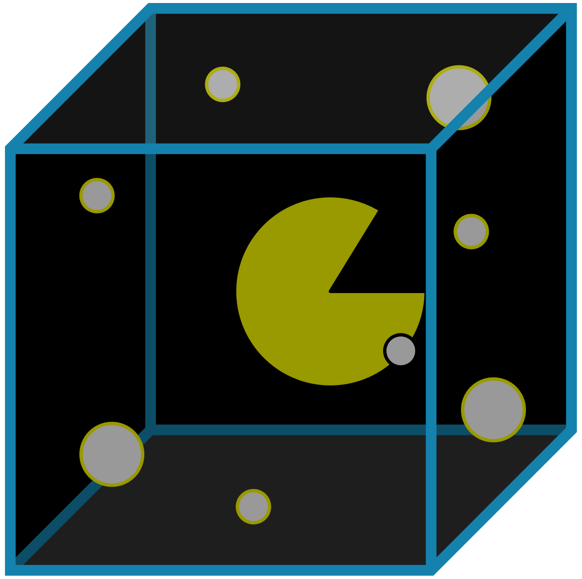


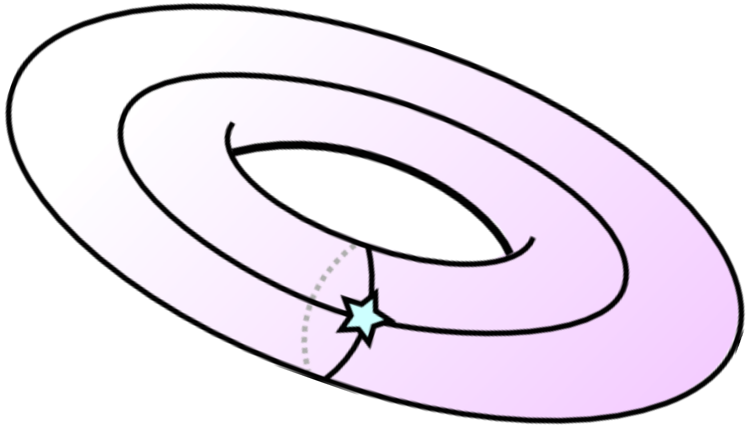




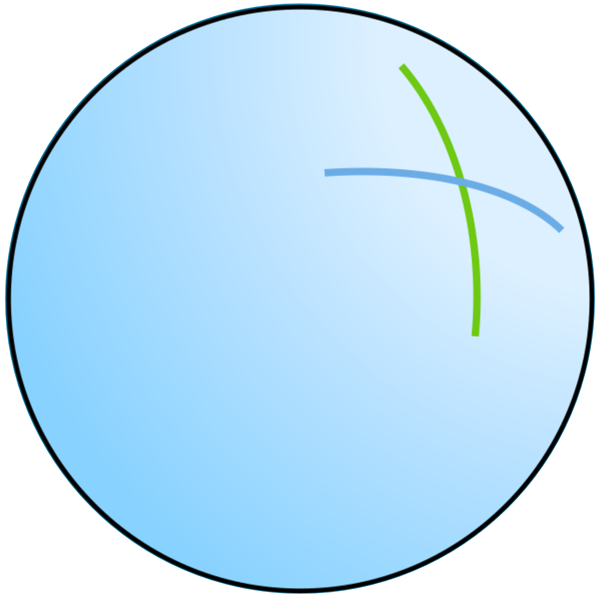
Intrinsic vs Extrinsic Dimension

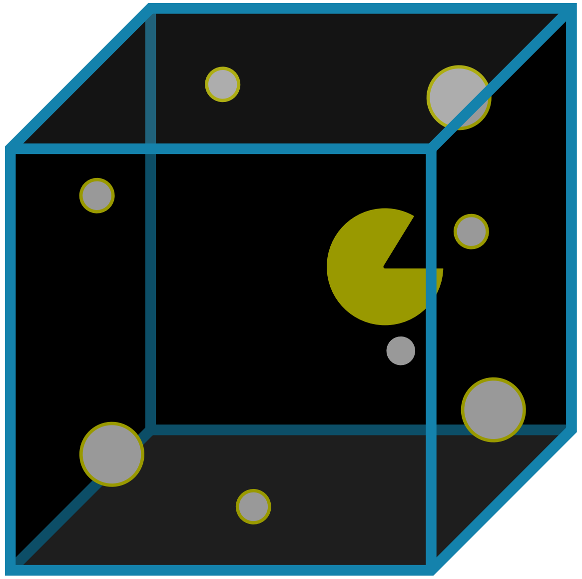


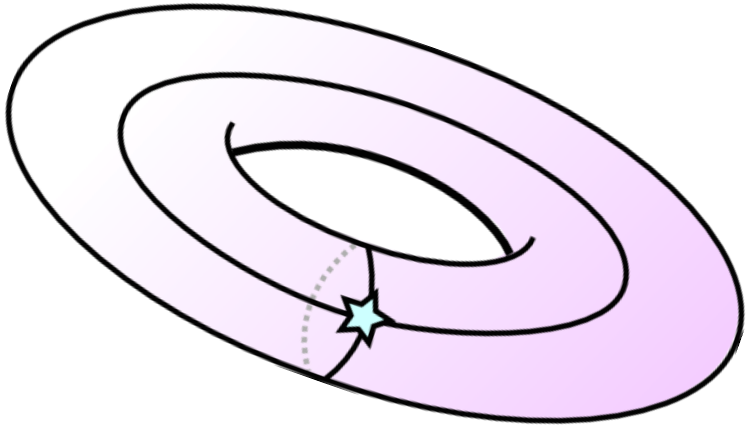




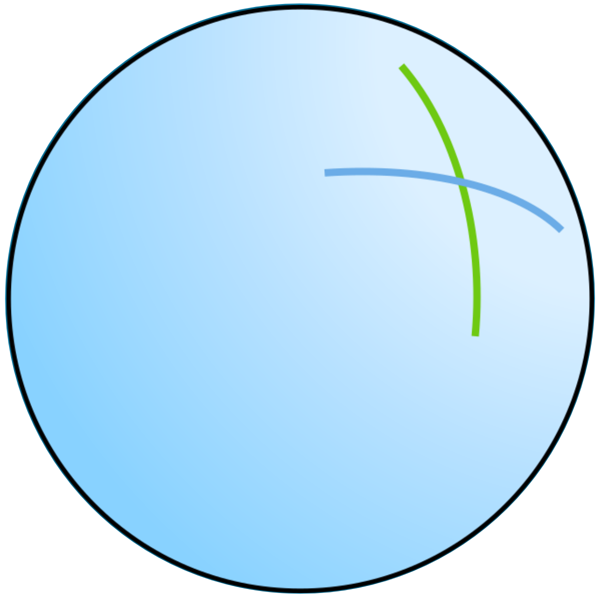
Intrinsic vs Extrinsic Dimension

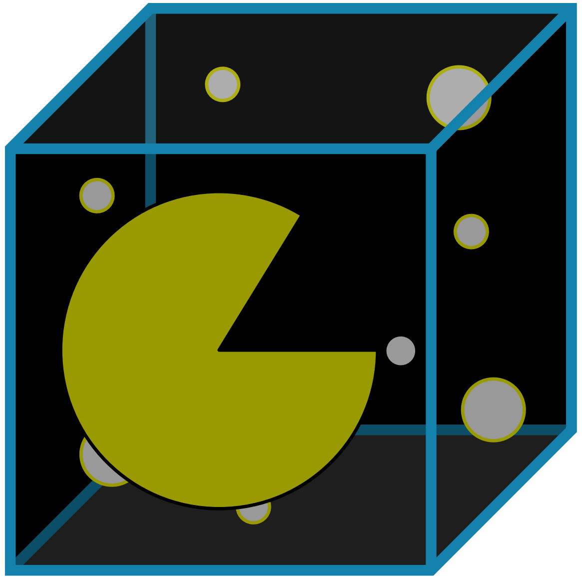


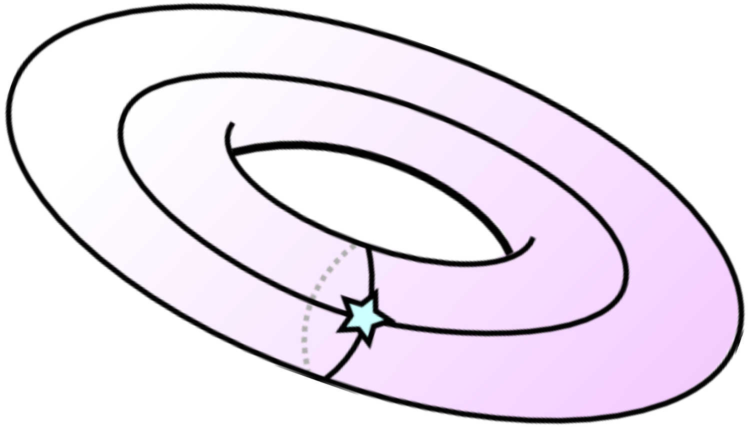




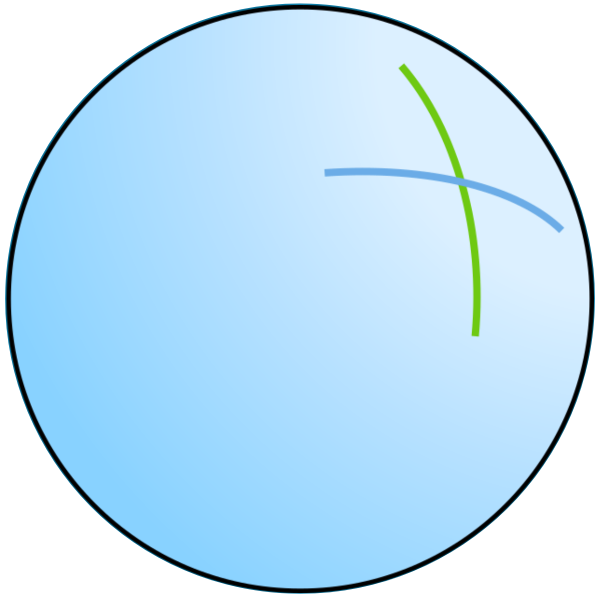
Intrinsic vs Extrinsic Dimension

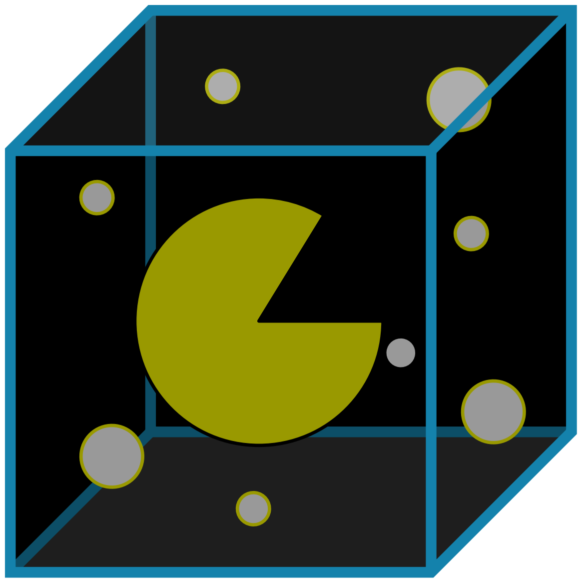


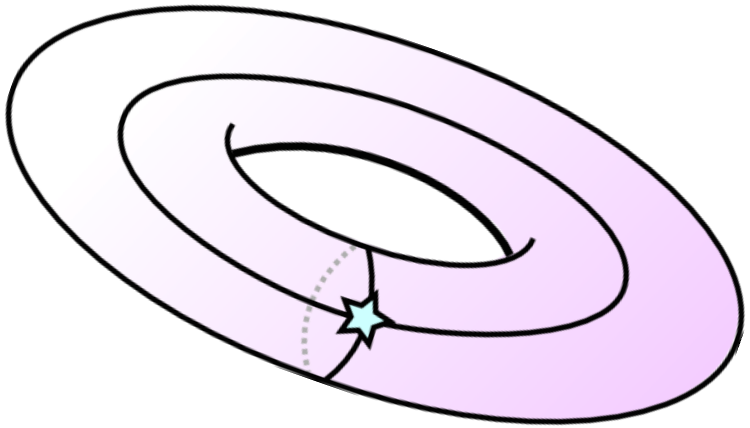




Intrinsic vs Extrinsic Dimension

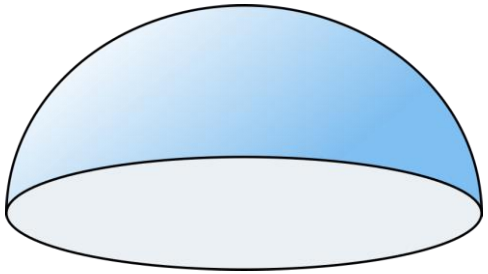


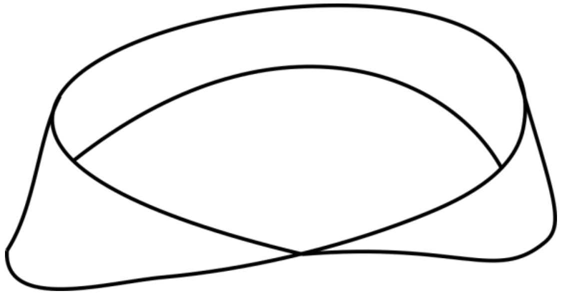




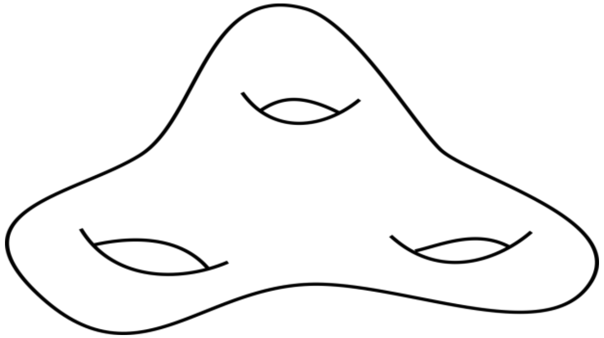
Compact manifolds

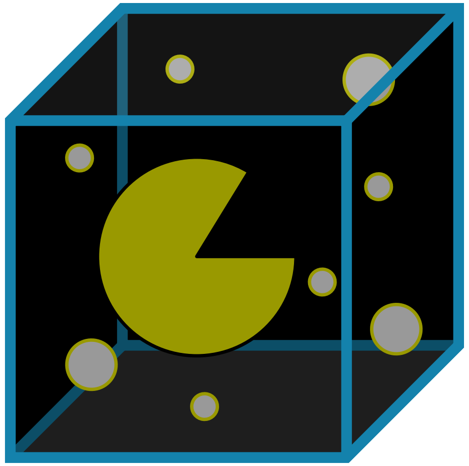
Manifold with Boundary





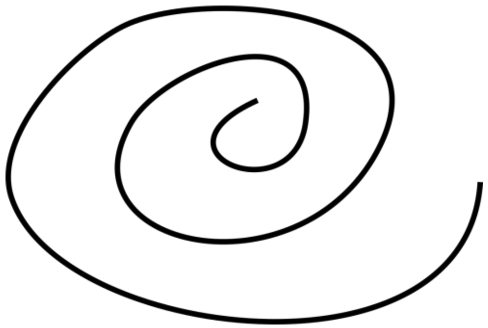
Closed manifolds



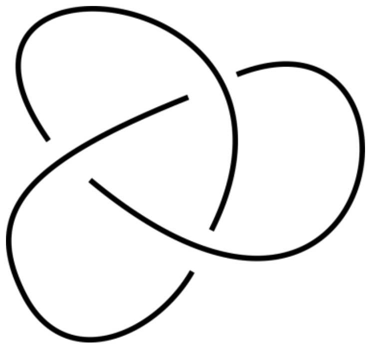


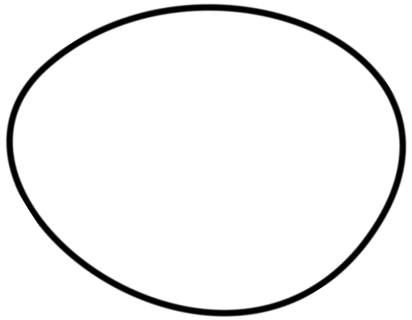
1D Compact Manifolds

Curve Segments

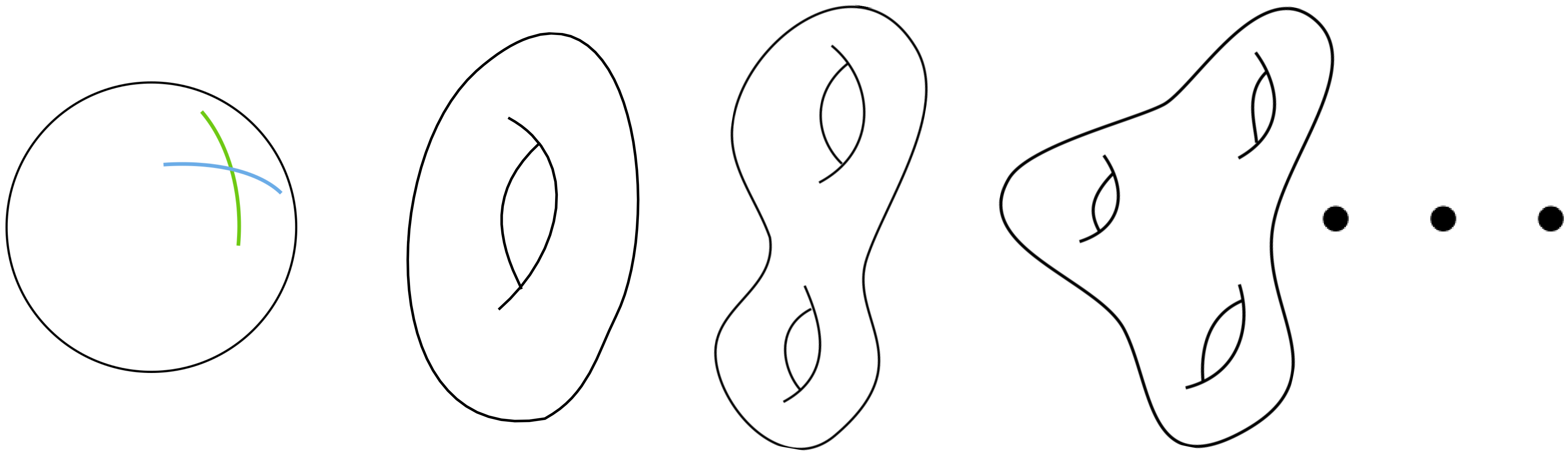


Closed Loops





2D Closed Orientable Manifolds



Riemannian Metrics and Geodesics

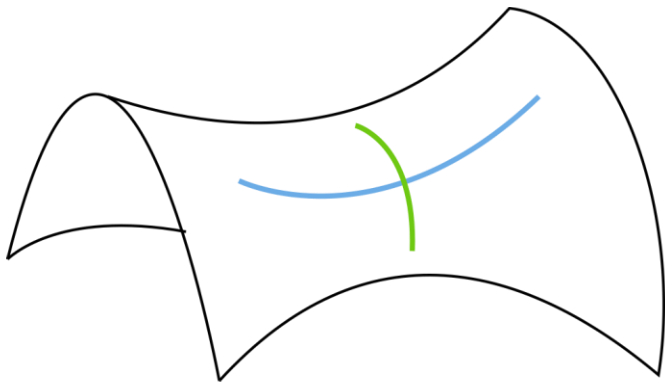
Riemannian metric $g_{ij} = g\left(\partial_i, \partial_j\right)$ gives local notion of scale and angle

This yields a notion of arclength of for smooth curves.

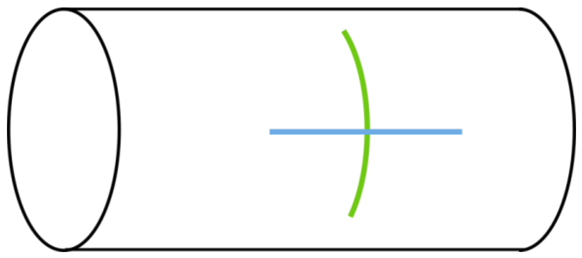
Geodesic distance from p to q is arclength of shortest path from p to q on manifold.

Not very interesting on 1D manifolds.

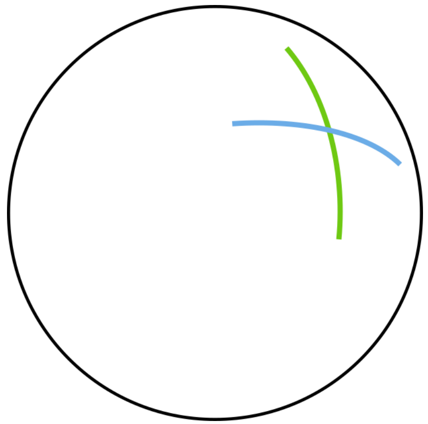
Intrinsic Curvature



Negative Curvature



Zero Curvature



Positive Curvature

MANIFOLD LEARNING

Tenenbaum, de Silva, Langford (2000)

Isomap

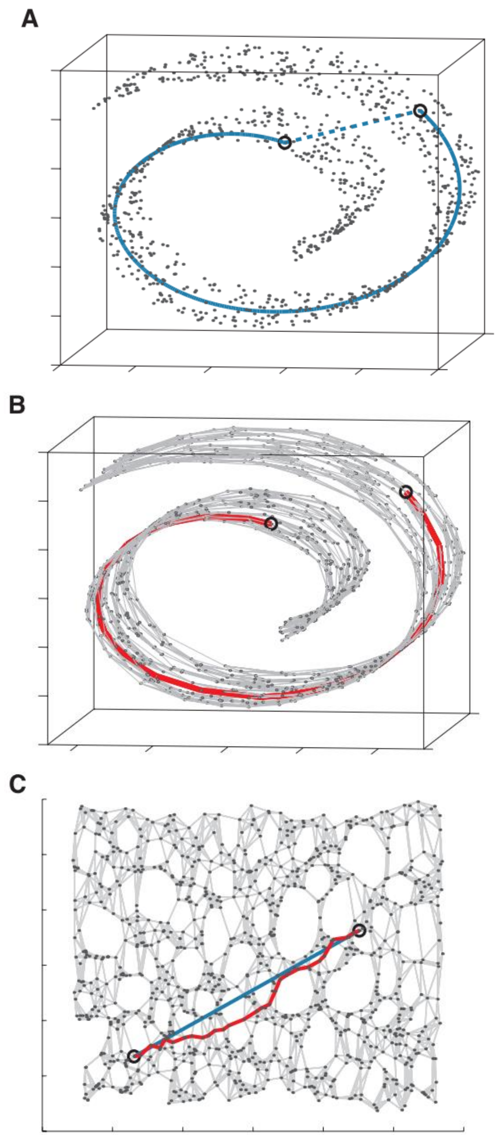
Replace point cloud with a **neighborhood graph.**

Trust local distances.

Geodisic distance ↔ Shortest graph path

Create matrix G of all pairwise geodesic distances

Find low-dimensionsal embedding by applying cMDS on G.



Tenenbaum, de Silva, Langford (2000)

Spectral Graph Theory Perspective:

Treat data as weighted graph (vertices are data points)

- Close or similar data points → High weights $w_{ij} \geq 0$

Idea: Embed vertices into $\mathbb{R}^d$ in way that minimizes spring-like potential

$$\underset{\substack{\rho \\ R^T R = I \\ R^T \mathbf{1} = 0}}{\arg\min} \; \frac{1}{2} \sum_{i,j} w_{ij} \left\| \rho(v_i) - \rho(v_j) \right\|^2$$

$$R = \begin{pmatrix} - & \rho(v_1)^T & - \\ - & \rho(v_2)^T & - \\ & \vdots & \\ - & \rho(v_n)^T & - \end{pmatrix} = \begin{pmatrix} | & | & & | \\ x^1 & x^2 & \dots & x^d \\ | & | & & | \end{pmatrix}$$

Can be solved by finding eigenvectors of $L = D - W$

Graph
Laplacian
Matrix

Laplace-Beltrami Operator (a.k.a The Laplacian)

Riemannian Metric:

$$g_{ij} = g\left(\partial_i, \partial_j\right)$$

Coordinate-Free:

$$\Delta_g f = \mathrm{div}_g \nabla_g f$$

In Coordinates:

$$\Delta_g f = \frac{1}{\sqrt{|\det(g)|}} \sum_{ij} \partial_i \left(\sqrt{|\det(g)|}\, g^{ij}\, \partial_j\, f \right)$$

Examples:

$$\Delta_{\mathbb{R}^2} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}, \qquad \Delta_{\mathbb{S}^2} = \frac{1}{\sin\theta} \frac{\partial}{\partial\theta}\left(\sin\theta \frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta} \frac{\partial^2}{\partial\varphi^2}$$

Spectral Geometry

Neumann Eigenvalues/Eigenfunctions

For compact M

$$-\Delta_M \varphi_i = \lambda_i \varphi_i, \qquad \text{in } \operatorname{int}(M)$$
$$\frac{\partial \varphi_i}{\partial \hat{n}} = 0, \qquad \text{in } \partial M$$

$$0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \to \infty$$

Weyl's Law (1911) for closed M says eigenvalues determine volume.

John Milnor (1964) found non-isometric 16-dimensional torii with same eigenvalues.

Spectral Embedding

Let M be closed and t>0

$$\psi_t : M \to \ell^2$$

$$x \mapsto \left\{ e^{-\lambda_j t} \varphi_j(x) \right\}_{j=1}^{\infty}$$

P. Bérard, et. al. (1994) showed that continuously embeds M into ℓ^2.

Jones, Maggioni, Schul (2013) showed one can do something similar to obtain local charts.

Laplacian Eigenmaps

Dataset: $D = \{x_i\}_{i=1}^{m} \in \mathbb{R}^N$

Construct Kernel Weight Matrix:

$$K_{ij} = \exp\left(-\frac{\left|x_i - x_j\right|^2}{\epsilon}\right)$$

Degree Normalize:

$$P_{ij} = \frac{K_{ij}}{d_i} \quad \text{where} \quad d_i = \sum_j K_{ij}$$

Get Eigenvalues:

$$1 \geq \lambda_0 \geq \cdots \geq \lambda_{m-1} \geq 0$$

Get Eigenvectors:

$$w_k = (w_{1k}, w_{2k}, \dots, w_{mk})^T$$

Laplacian Eigenmaps:

$$\Psi\colon \mathrm{D} \to \mathbb{R}^\wedge d$$

$$\Psi(x_i) = (w_{i1}, w_{i2}, \dots, w_{id})^T$$

$$d \in \{1,2 \dots, m-1\}$$

Scale Parameter ϵ

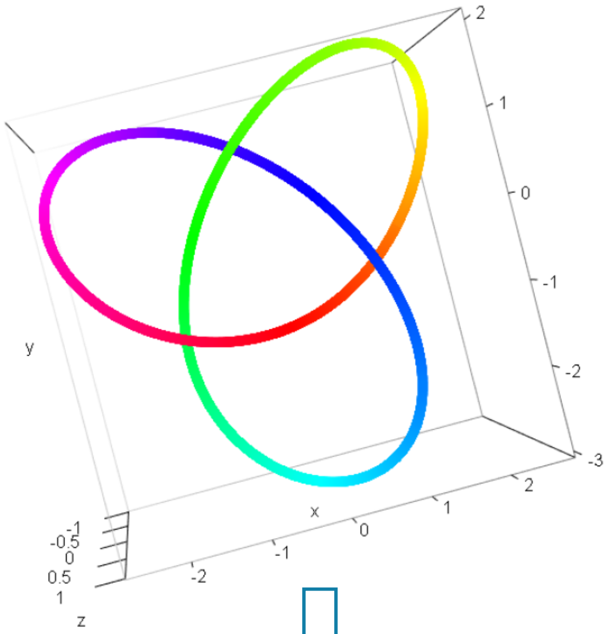
Parameter ϵ controls influence of neighbors:

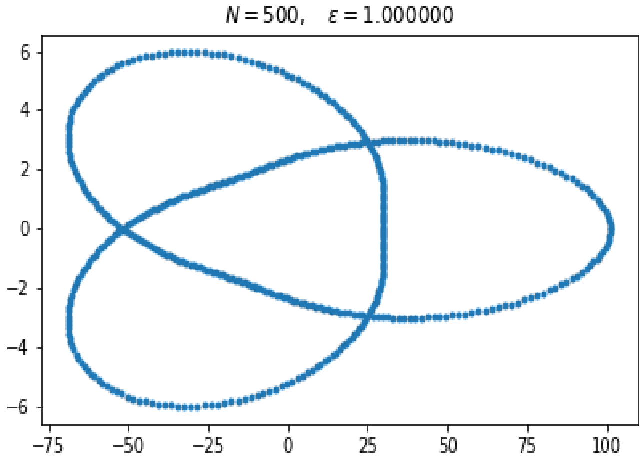
- Larger ϵ → Coarser scale (Global features)
- Smaller ϵ → Finer scale (Local features)

$$K_{ij} = \exp\left(-\frac{\left|x_i - x_j\right|^2}{\epsilon}\right)$$

Trefoil 2D image as ϵ goes to zero:

- Tangled → Untangled
- Sampling density matters
- Unstable as ϵ gets too small (not shown)





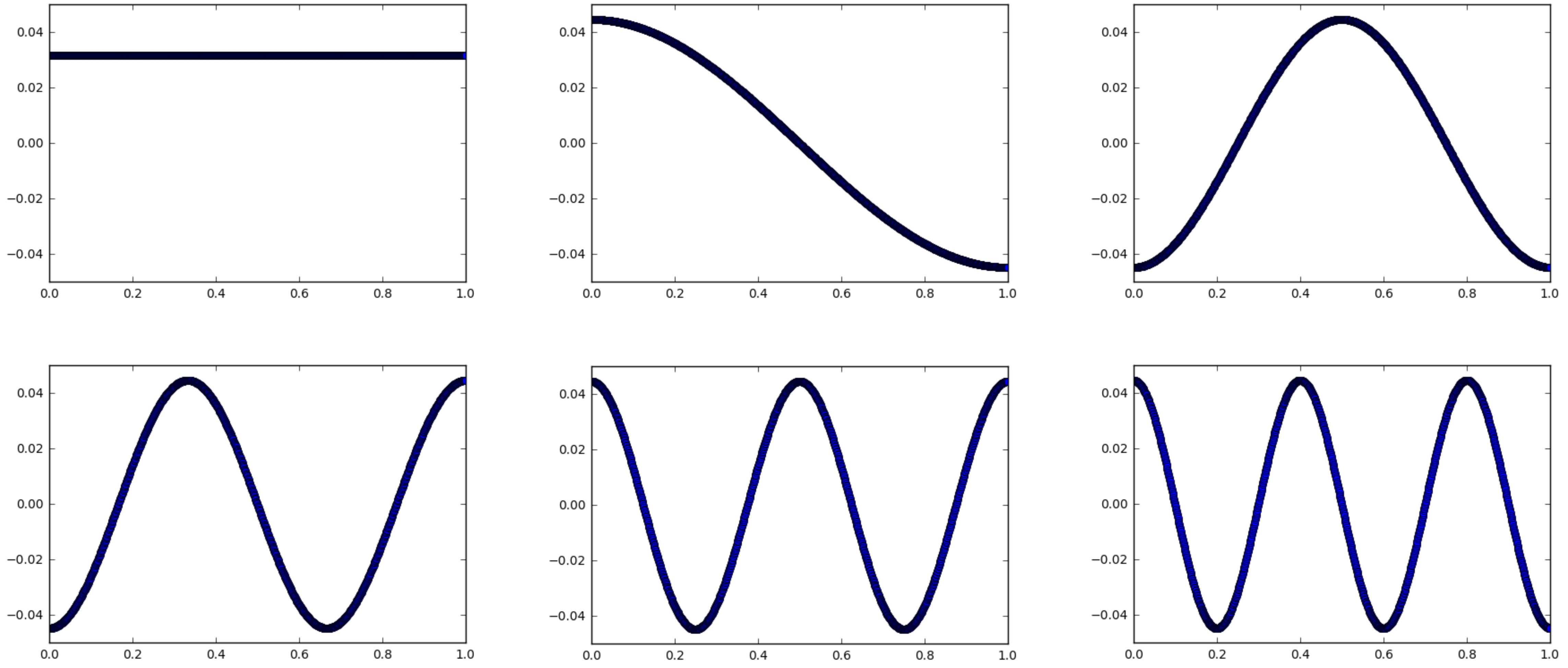
Connection to Laplace-Beltrami Operator

If M is sampled uniformly $\rightarrow$ Eigenvectors in Laplacian Eigenmaps correspond to Neumann eigenfunctions of Δ_M… Namely $w_{ik} \approx \varphi_k(x_i)$.

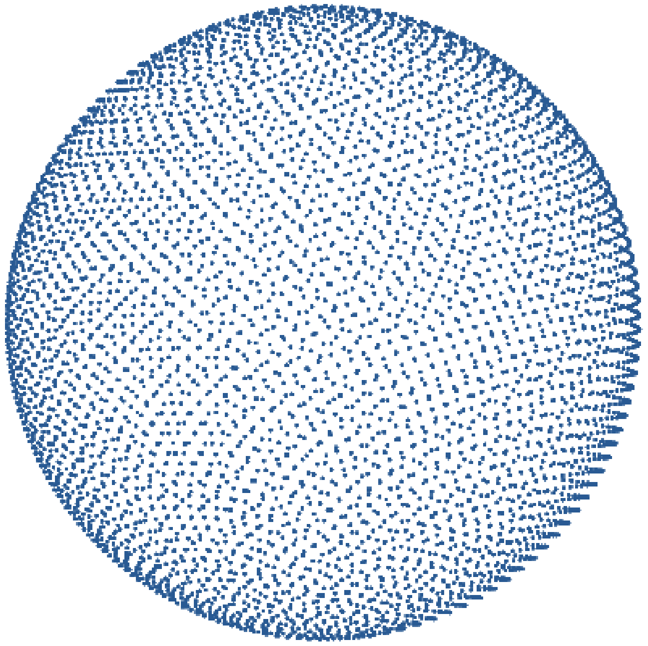
Eigenvectors are highly sensitive to sampling distribution.

Laplacian Eigenmaps $\approx$ discrete truncation of the continuous spectral embedding

EXAMPLE: Unit Interval [0,1]

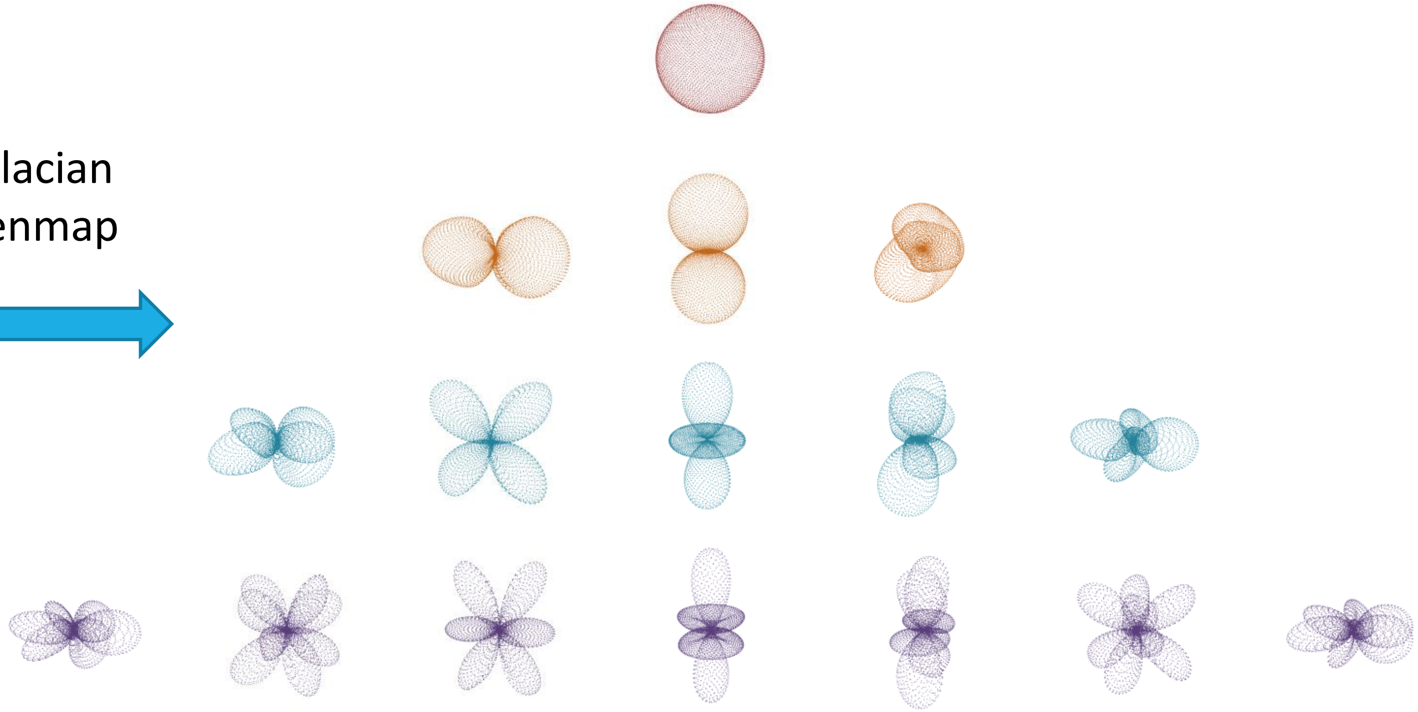


EXAMPLE: Unit Sphere



Sampling of the Sphere

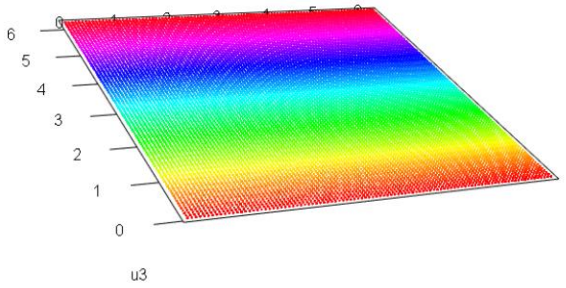
Laplacian
Eigenmap

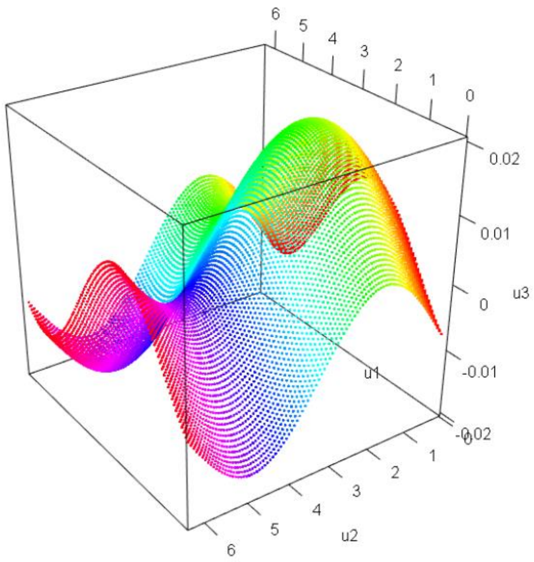


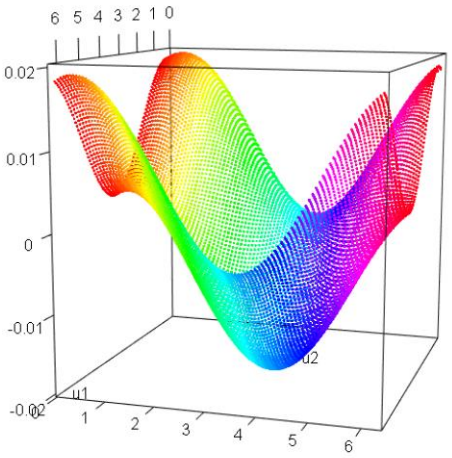
Laplacian Eigenmap Components ≈ Spherical Harmonics

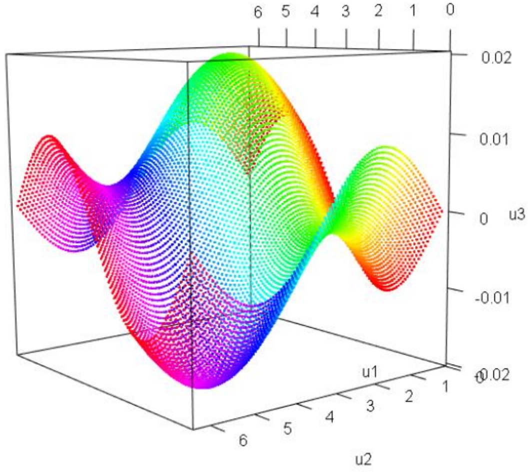
EXAMPLE: Flat Torus

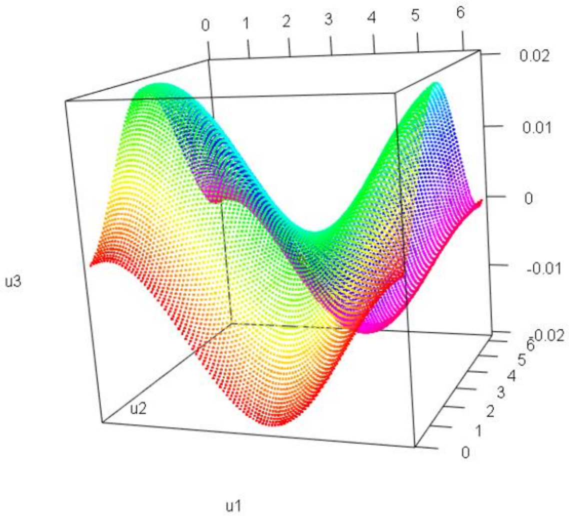
$$f(\theta, \varphi) = (\cos\theta\,, \sin\theta, \cos\varphi\,, \sin\varphi) \in \mathbb{R}^4$$

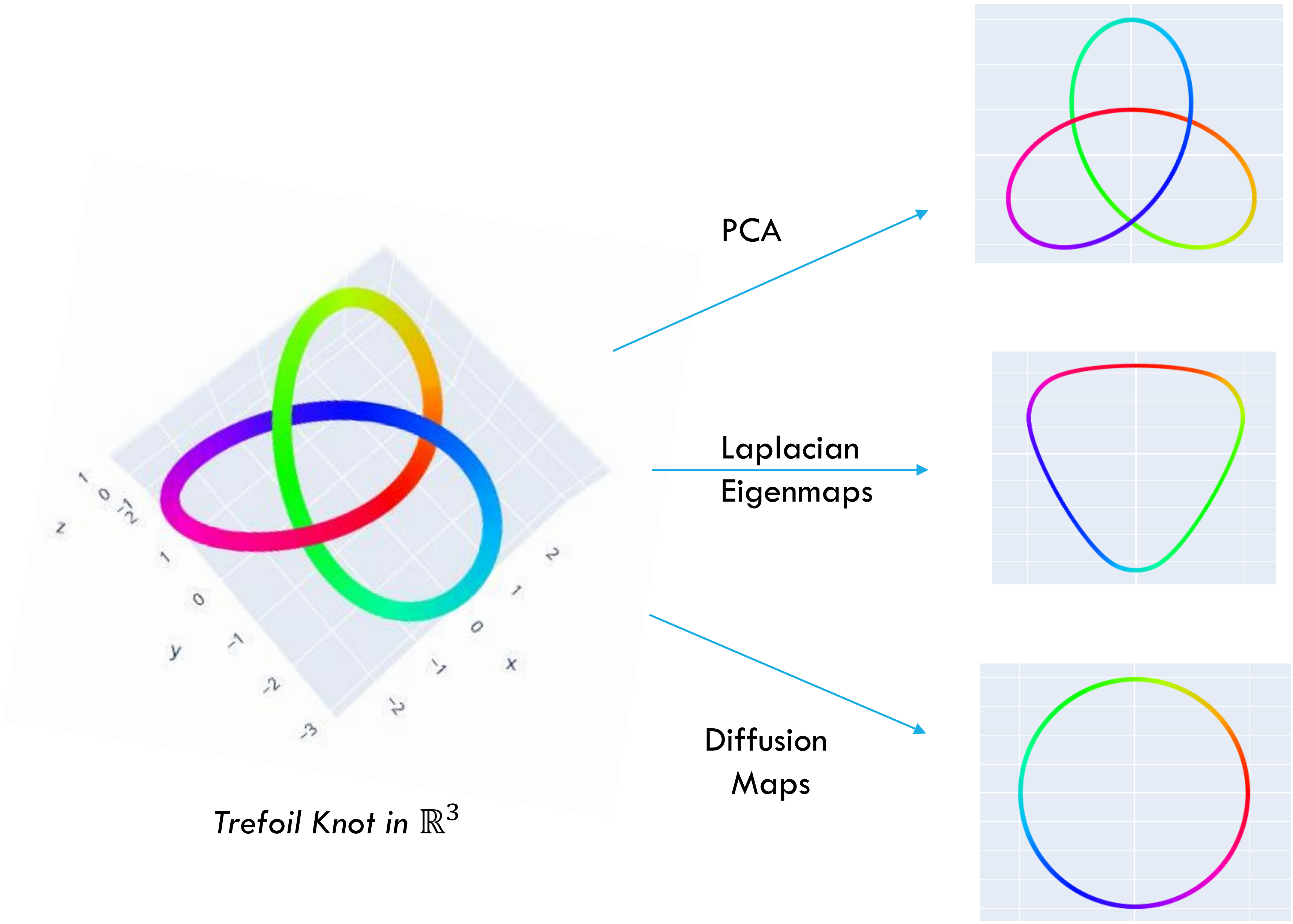






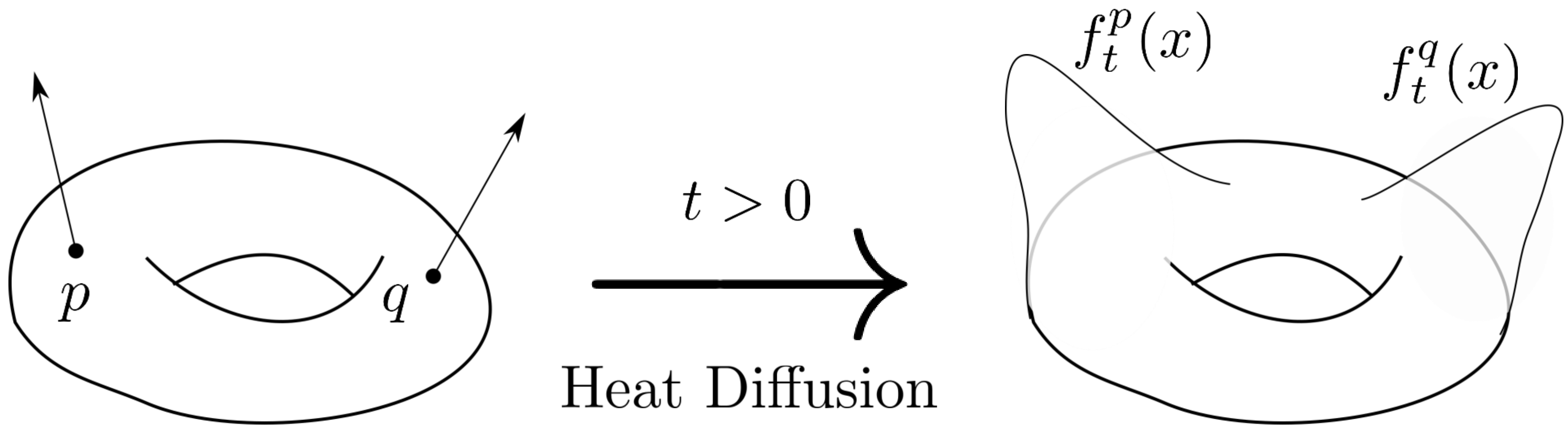






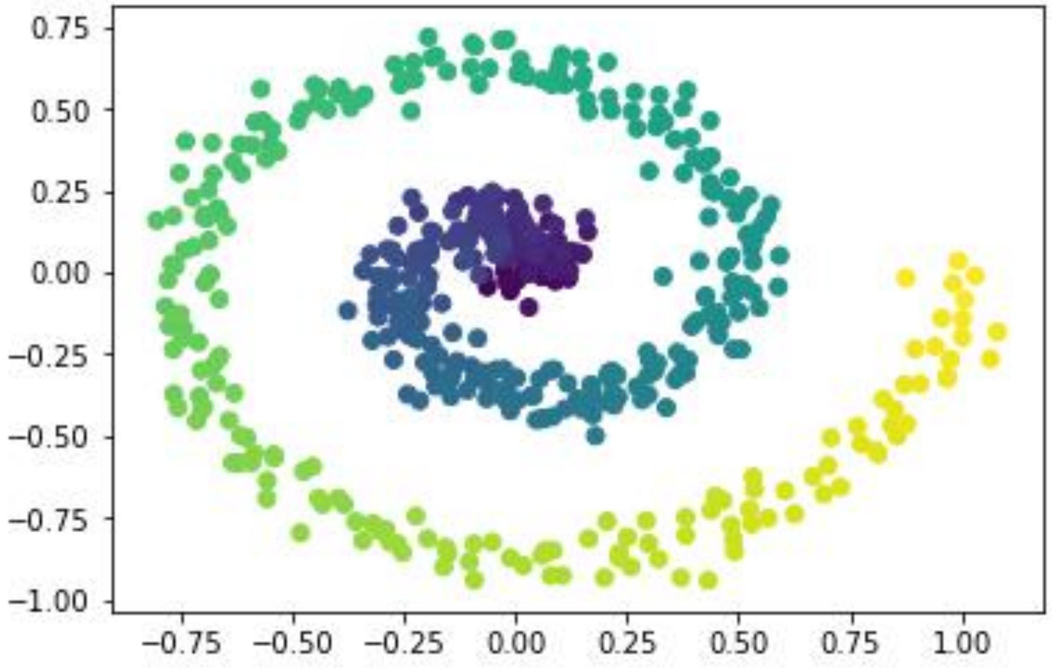
PCA
Laplacian
Eigenmaps
Diffusion
Maps
Trefoil Knot in $\mathbb{R}^3$

DIFFUSION DISTANCE

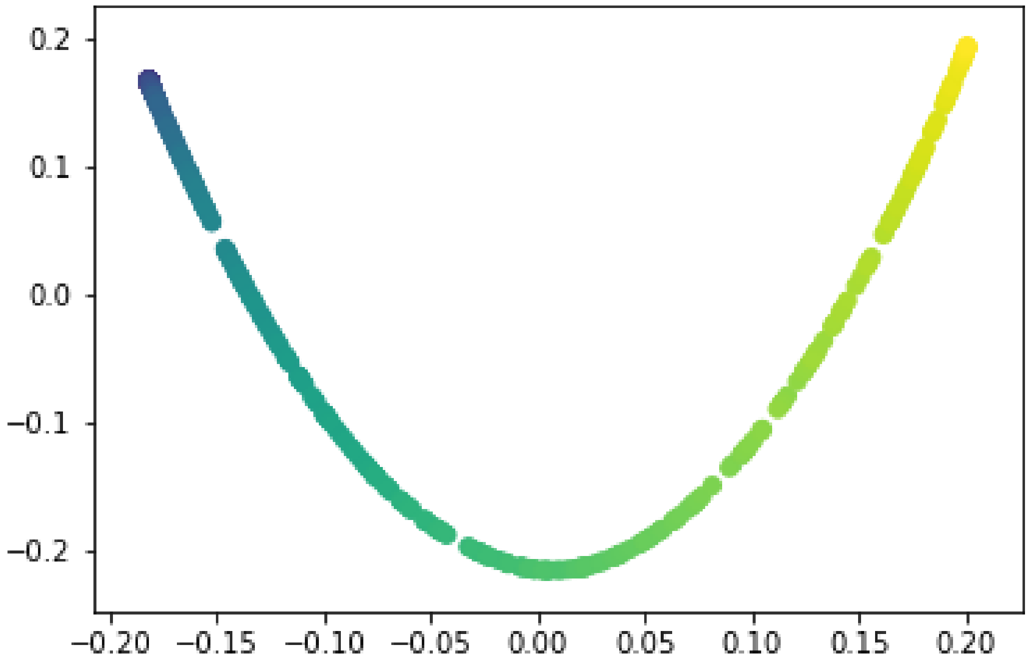


$$D_t^2(p,q) = \|f_t^p - f_t^q\|_2^2 = \int_M |f_t^p - f_t^q|^2 \mathrm{d}x$$

Diffusion Maps - Nonlinear Denoising



A Noisy Spiral



After Diffusion Maps

Recall Laplacian Eigenmaps

Dataset: $D = \{x_i\}_{i=1}^{m} \in \mathbb{R}^N$

Construct Kernel Weight Matrix:

$$K_{ij} = \exp\left(-\frac{\left|x_i - x_j\right|^2}{\epsilon}\right) \qquad d_i = \sum_j K_{ij}$$

Degree Normalize:

$$P_{ij} = \frac{K_{ij}}{d_i}$$

Get Eigenvalues:

$$1 \geq \lambda_0 \geq \cdots \geq \lambda_{m-1} \geq 0$$

Get Eigenvectors:

$$w_k = (w_{1k}, w_{2k}, \dots, w_{mk})^T$$

Laplacian Eigenmaps:

$$\Psi\colon \mathrm{D} \to \mathbb{R}\text{^}d$$

$$\Psi(x_i) = (w_{i1}, w_{i2}, \dots, w_{id})^T$$

$$d \in \{1,2 \dots, m-1\}$$

Diffusion Maps Construction:

Data set:

$$D = \{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^N$$

Parameters:

$$0 \leq \alpha \leq 1, \epsilon > 0$$

Diffusion Maps Construction:

Data set:

$$D = \{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^N$$

Parameters:

$$0 \leq \alpha \leq 1, \epsilon > 0$$

Kernels, Operators and Densities:

Isotropic Kernel

$$K_\epsilon(x, y) = \exp\left(-\frac{\|x - y\|^2}{\epsilon}\right)$$

Kernel Density Estimate

$$q_\epsilon(x) = \sum_j K_\epsilon(x, x_j)$$

Anisotropic Kernel

$$S_{\epsilon,\alpha}(x, y) = \frac{K_\epsilon(x, y)}{q_\epsilon(x)^\alpha q_\epsilon(y)^\alpha}$$

Anisotropic Kernel Density Estimate

$$d_{\epsilon,\alpha}(x) = \sum_j S_{\epsilon,\alpha}(x, x_j)$$

Diffusion Maps Kernel

$$J_{\epsilon,\alpha}(x, y) = \frac{S_{\epsilon,\alpha}(x, y)}{\sqrt{d_{\epsilon,\alpha}(x) d_{\epsilon,\alpha}(y)}}$$

Random Walk Kernel

$$P_{\epsilon,\alpha}(x, y) = \frac{S_\epsilon(x, y)}{d_{\epsilon,\alpha}(x)}$$

Diffusion Maps Construction:

Data set:

$$D = \{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^N$$

Parameters:

$$0 \leq \alpha \leq 1, \epsilon > 0$$

Kernels, Operators and Densities:

Isotropic Kernel

$$K_\epsilon(x,y) = \exp\left(-\frac{\|x-y\|^2}{\epsilon}\right)$$

Kernel Density Estimate

$$q_\epsilon(x) = \sum_j K_\epsilon(x, x_j)$$

Anisotropic Kernel

$$S_{\epsilon,\alpha}(x,y) = \frac{K_\epsilon(x,y)}{q_\epsilon(x)^\alpha q_\epsilon(y)^\alpha}$$

Anisotropic Kernel Density Estimate

$$d_{\epsilon,\alpha}(x) = \sum_j S_{\epsilon,\alpha}(x, x_j)$$

Diffusion Maps Kernel

$$J_{\epsilon,\alpha}(x,y) = \frac{S_{\epsilon,\alpha}(x,y)}{\sqrt{d_{\epsilon,\alpha}(x) d_{\epsilon,\alpha}(y)}}$$

Random Walk Kernel

$$P_{\epsilon,\alpha}(x,y) = \frac{S_\epsilon(x,y)}{d_{\epsilon,\alpha}(x)}$$

Matrices:

$$J_{\epsilon,\alpha}[i,j] = J_{\epsilon,\alpha}(x_i, x_j)$$

$$P_{\epsilon,\alpha}[i,j] = P_{\epsilon,\alpha}(x_i, x_j)$$

Diffusion Maps Construction:

Data set:

$D = \{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^N$

Parameters:

$0 \leq \alpha \leq 1, \epsilon > 0$

Need:

Spectral decomposition of $P_{\epsilon,\alpha}[i,j]$

Kernels, Operators and Densities:

Isotropic Kernel

$$K_\epsilon(x,y) = \exp\left(-\frac{\|x-y\|^2}{\epsilon}\right)$$

Kernel Density Estimate

$$q_\epsilon(x) = \sum_j K_\epsilon(x, x_j)$$

Anisotropic Kernel

$$S_{\epsilon,\alpha}(x,y) = \frac{K_\epsilon(x,y)}{q_\epsilon(x)^\alpha q_\epsilon(y)^\alpha}$$

Anisotropic Kernel Density Estimate

$$d_{\epsilon,\alpha}(x) = \sum_j S_{\epsilon,\alpha}(x, x_j)$$

Diffusion Maps Kernel

$$J_{\epsilon,\alpha}(x,y) = \frac{S_{\epsilon,\alpha}(x,y)}{\sqrt{d_{\epsilon,\alpha}(x) d_{\epsilon,\alpha}(y)}}$$

Random Walk Kernel

$$P_{\epsilon,\alpha}(x,y) = \frac{S_\epsilon(x,y)}{d_{\epsilon,\alpha}(x)}$$

Matrices:

$$J_{\epsilon,\alpha}[i,j] = J_{\epsilon,\alpha}(x_i, x_j)$$

$$P_{\epsilon,\alpha}[i,j] = P_{\epsilon,\alpha}(x_i, x_j)$$

Diffusion Maps Construction:

Data set:

$$D = \{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^N$$

Parameters:

$$0 \leq \alpha \leq 1, \epsilon > 0$$

Need:

Spectral decomposition of $P_{\epsilon,\alpha}[i,j]$

Eigenvalues:

$$1 \geq \lambda_0 \geq \cdots \geq \lambda_{m-1} \geq 0$$

Eigenvectors:

$$w_k = (w_{1k}, \dots, w_{mk})^T$$

Kernels, Operators and Densities:

Isotropic Kernel

$$K_\epsilon(x,y) = \exp\left(-\frac{\|x-y\|^2}{\epsilon}\right)$$

Kernel Density Estimate

$$q_\epsilon(x) = \sum_j K_\epsilon(x, x_j)$$

Anisotropic Kernel

$$S_{\epsilon,\alpha}(x,y) = \frac{K_\epsilon(x,y)}{q_\epsilon(x)^\alpha q_\epsilon(y)^\alpha}$$

Anisotropic Kernel Density Estimate

$$d_{\epsilon,\alpha}(x) = \sum_j S_{\epsilon,\alpha}(x, x_j)$$

Diffusion Maps Kernel

$$J_{\epsilon,\alpha}(x,y) = \frac{S_{\epsilon,\alpha}(x,y)}{\sqrt{d_{\epsilon,\alpha}(x) d_{\epsilon,\alpha}(y)}}$$

Random Walk Kernel

$$P_{\epsilon,\alpha}(x,y) = \frac{S_\epsilon(x,y)}{d_{\epsilon,\alpha}(x)}$$

Matrices:

$$J_{\epsilon,\alpha}[i,j] = J_{\epsilon,\alpha}(x_i, x_j) \qquad P_{\epsilon,\alpha}[i,j] = P_{\epsilon,\alpha}(x_i, x_j)$$

Diffusion Maps Construction:

Data set:

$$D = \{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^N$$

Parameters:

$$0 \leq \alpha \leq 1, \epsilon > 0$$

Kernels, Operators and Densities:

Isotropic Kernel

$$K_\epsilon(x,y) = \exp\left(-\frac{\|x-y\|^2}{\epsilon}\right)$$

Kernel Density Estimate

$$q_\epsilon(x) = \sum_j K_\epsilon(x, x_j)$$

Anisotropic Kernel

$$S_{\epsilon,\alpha}(x,y) = \frac{K_\epsilon(x,y)}{q_\epsilon(x)^\alpha q_\epsilon(y)^\alpha}$$

Anisotropic Kernel Density Estimate

$$d_{\epsilon,\alpha}(x) = \sum_j S_{\epsilon,\alpha}(x, x_j)$$

Diffusion Maps Kernel

$$J_{\epsilon,\alpha}(x,y) = \frac{S_{\epsilon,\alpha}(x,y)}{\sqrt{d_{\epsilon,\alpha}(x) d_{\epsilon,\alpha}(y)}}$$

Random Walk Kernel

$$P_{\epsilon,\alpha}(x,y) = \frac{S_\epsilon(x,y)}{d_{\epsilon,\alpha}(x)}$$

Matrices:

$$J_{\epsilon,\alpha}[i,j] = J_{\epsilon,\alpha}(x_i, x_j)$$

$$P_{\epsilon,\alpha}[i,j] = P_{\epsilon,\alpha}(x_i, x_j)$$

Need:

Spectral decomposition of $P_{\epsilon,\alpha}[i,j]$

Eigenvalues:

$$1 \geq \lambda_0 \geq \cdots \geq \lambda_{m-1} \geq 0$$

Eigenvectors:

$$w_k = (w_{1k}, \ldots, w_{mk})^T$$

Diffusion Maps:

$$\Psi : D \to \mathbb{R}^d$$

$$\Psi(x_i) = (\lambda_1^t w_{i1}, \ldots, \lambda_d^t w_{id})^T$$

$$d \in \{1, \ldots, m-1\}, \quad t \geq 0$$

Significance of Parameter α

Sample with density q on M:

$(\alpha = 0)$ $\rightarrow$ Laplacian Eigenmaps

$(\alpha = 1/2)$ $\rightarrow$ Intermediate

(Fokker-Planck operator)

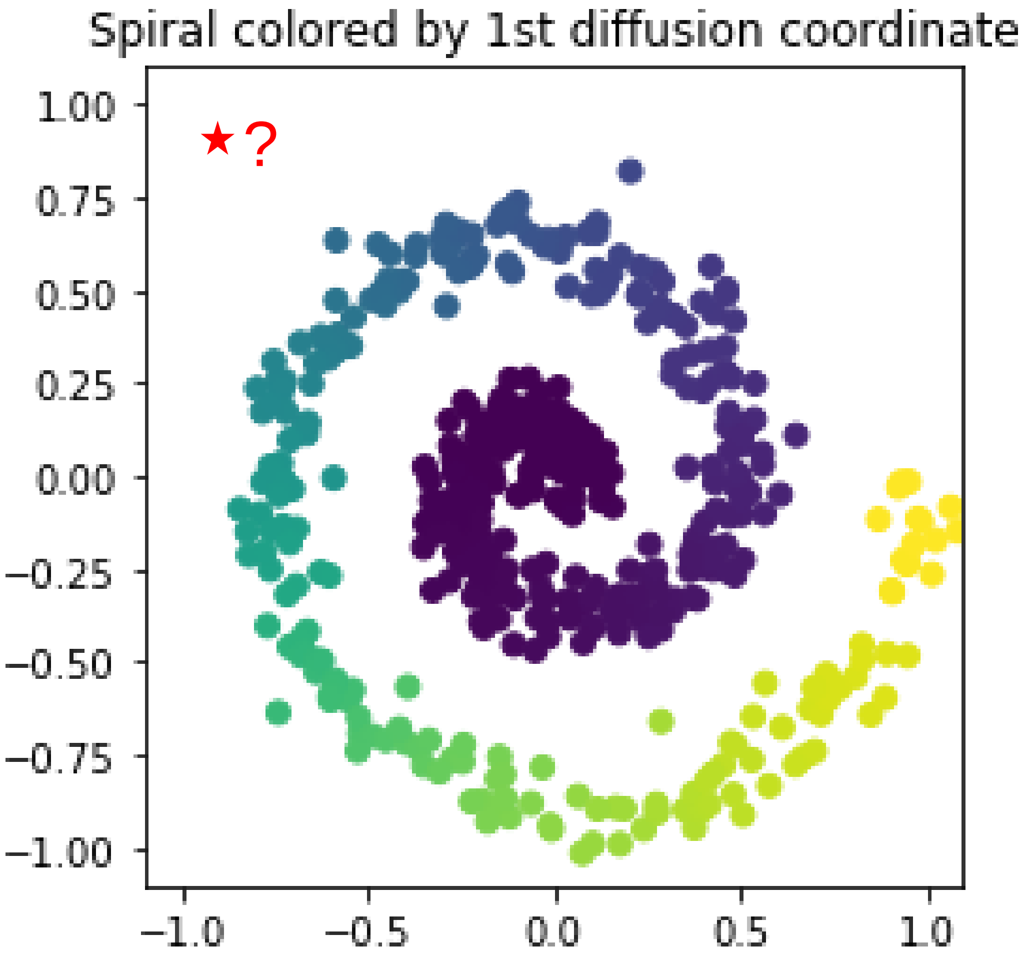
$(\alpha = 1)$ $\rightarrow$ Independent of sampling

$$L_{\epsilon,\alpha} = \frac{I - P_{\epsilon,\alpha}}{\epsilon}$$

$$\lim_{\epsilon \to 0} L_{\epsilon,\alpha} f = \frac{\Delta_M(f q^{1-\alpha})}{q^{1-\alpha}} + \frac{\Delta_M(q^{1-\alpha})}{q^{1-\alpha}} f$$

Large data limit

Out-of-Sample-Extension (OOSE)



Nyström Extension

$J_{\epsilon,\alpha}$ is a data-dependent Mercer kernel → Diffusion maps related KPCA

$$K_\epsilon(x,y) = \exp\left(-\frac{\|x-y\|^2}{\epsilon}\right) \qquad q_\epsilon(x) = \sum_j K_\epsilon(x, x_j)$$

$$S_{\epsilon,\alpha}(x,y) = \frac{K_\epsilon(x,y)}{q_\epsilon(x)^\alpha q_\epsilon(y)^\alpha} \qquad d_{\epsilon,\alpha}(x) = \sum_j S_{\epsilon,\alpha}(x, x_j)$$

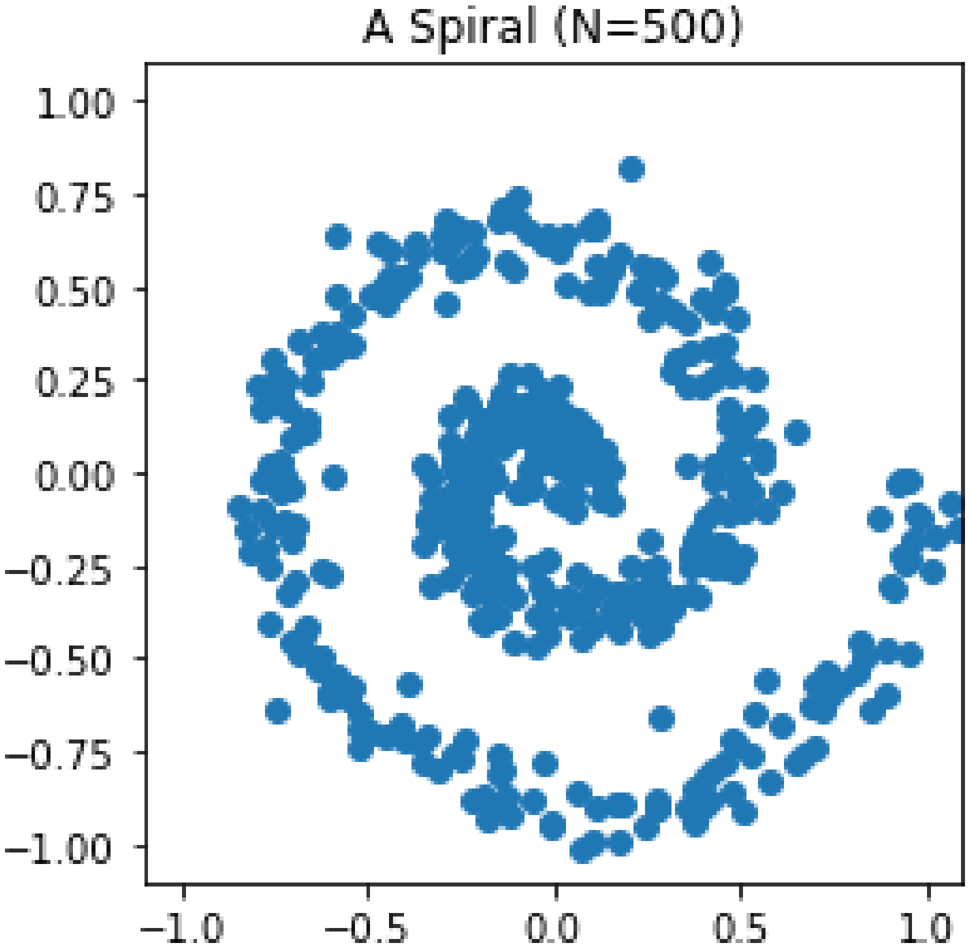
$$J_{\epsilon,\alpha}(x,y) = \frac{S_{\epsilon,\alpha}(x,y)}{\sqrt{d_{\epsilon,\alpha}(x) d_{\epsilon,\alpha}(y)}} \qquad P_{\epsilon,\alpha}(x,y) = \frac{S_\epsilon(x,y)}{d_{\epsilon,\alpha}(x)}$$

Some algebra + KPCA theory yields associated smooth extension formula

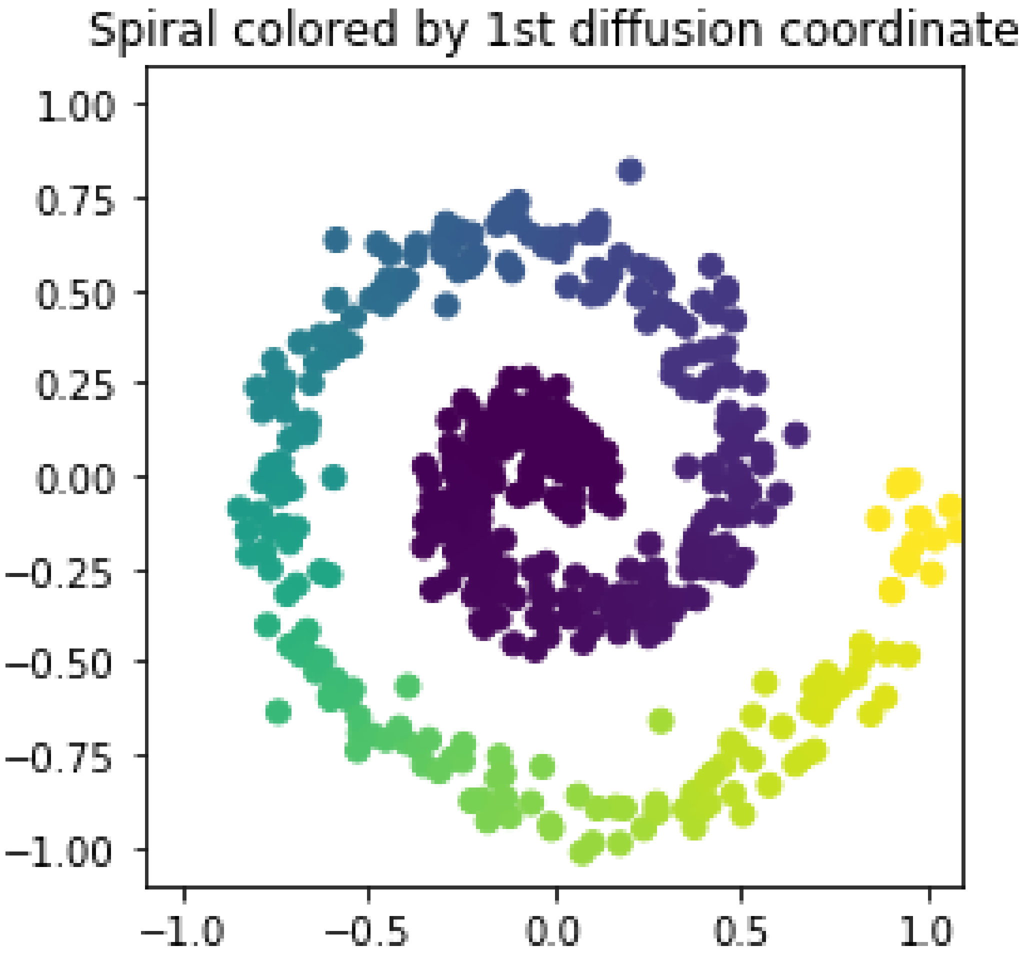
Agreement on data

$$f_k(x) = \lambda^{t-1} \sum_{j=1}^{m} P_{\epsilon,\alpha}(x, x_j) w_{jk} \qquad f_k(x_i) = \lambda^{t-1} \sum_{j=1}^{m} P_{\epsilon,\alpha}[i,j] w_{jk} = \lambda_k^{t-1} \lambda_k w_{ik} = \Psi(x_i)_k$$

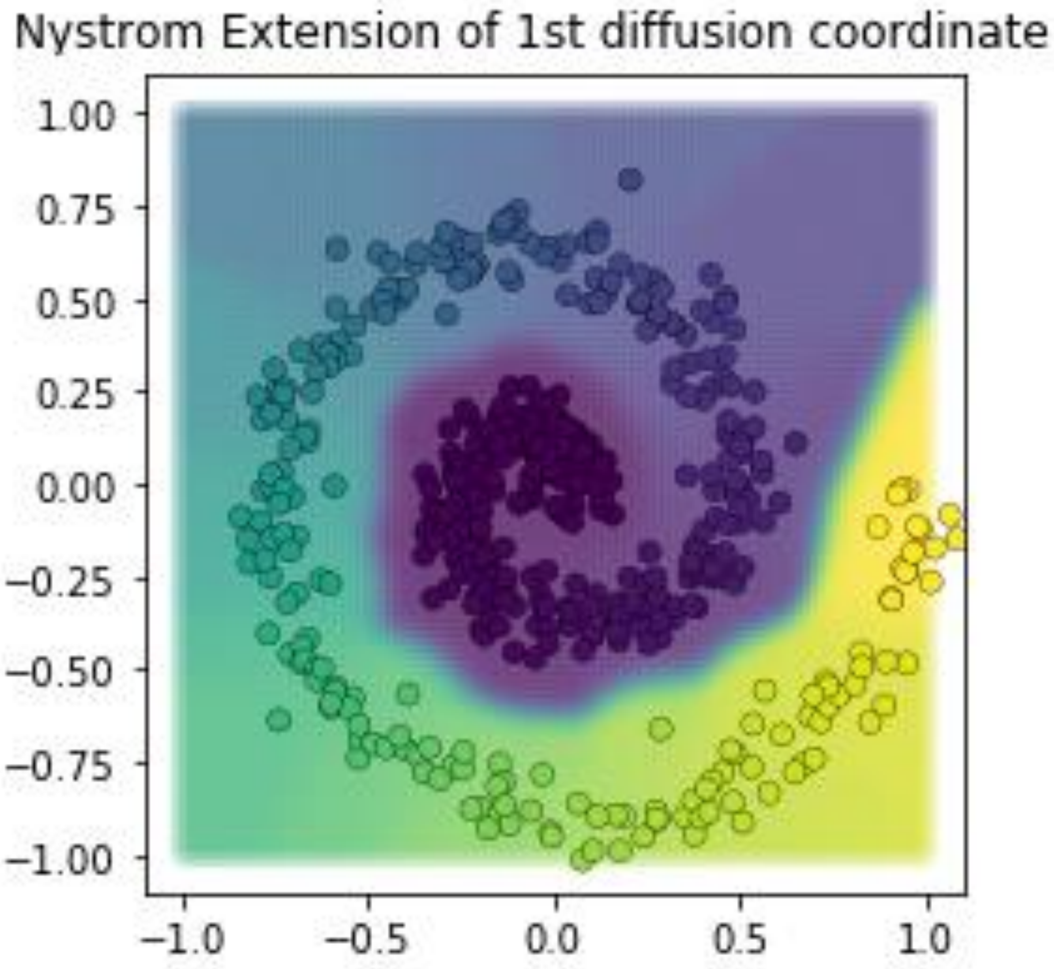
Nyström Extension



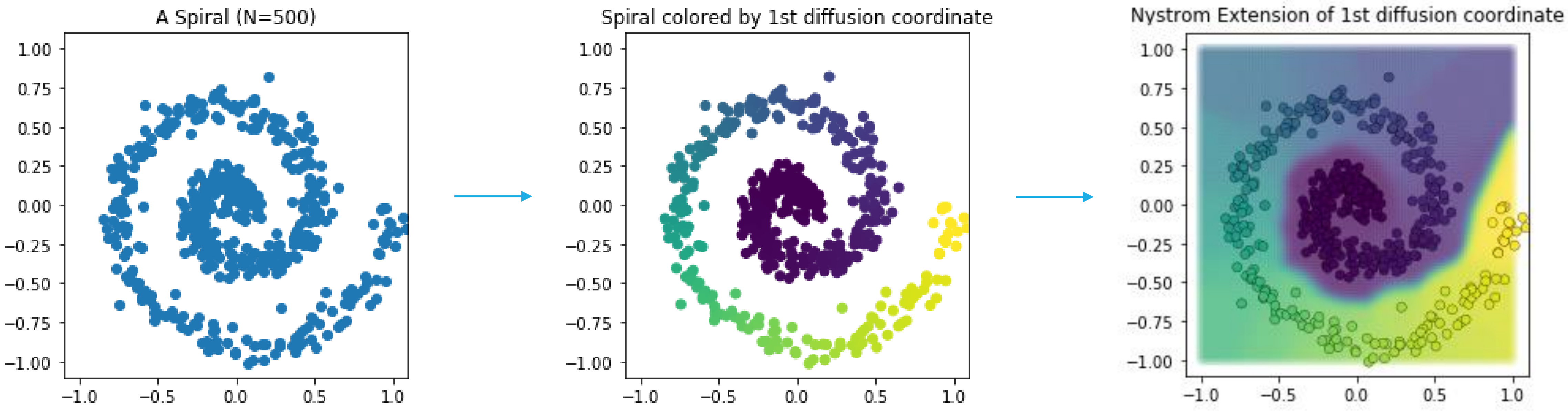
Nyström Extension

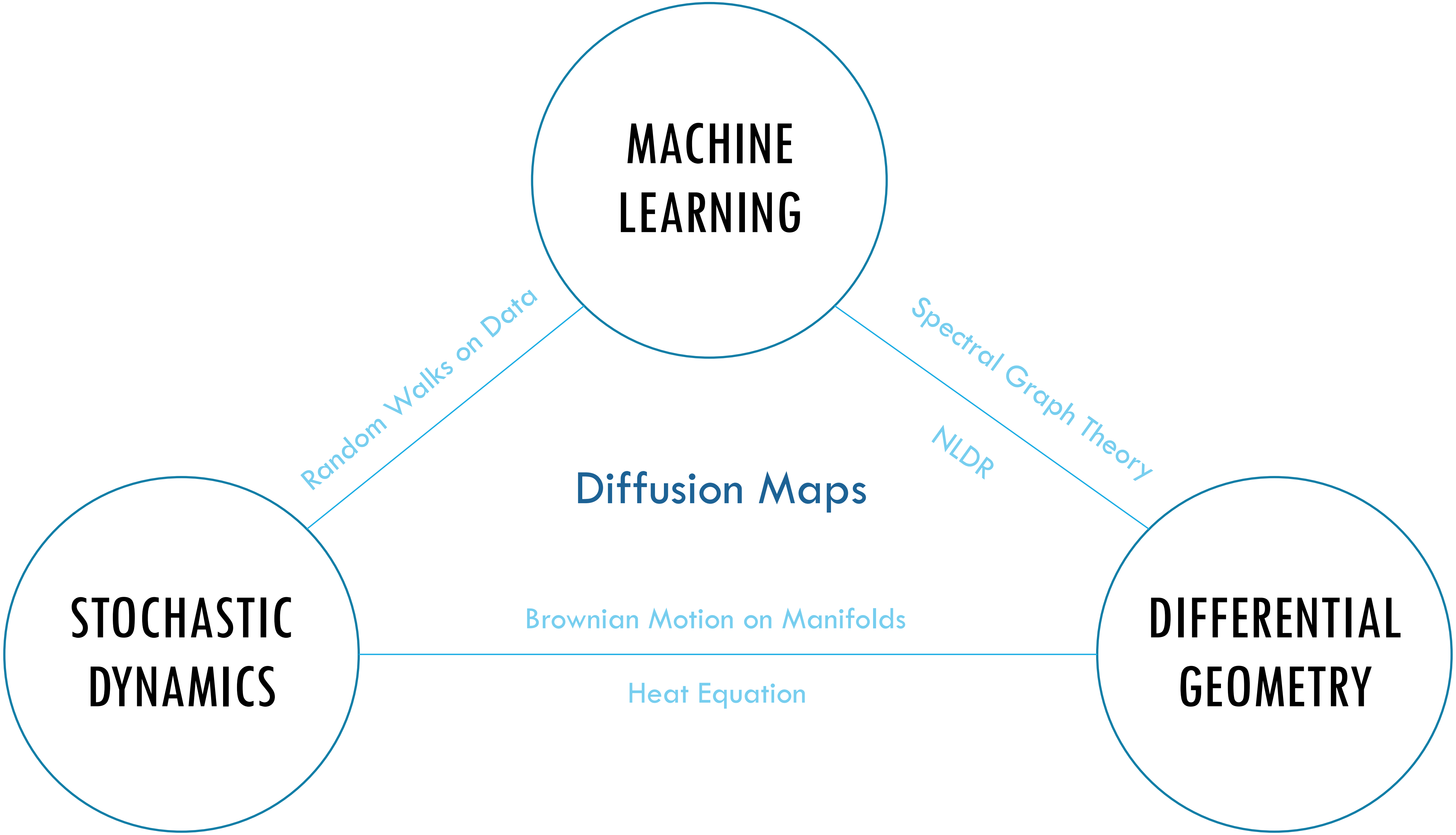


Nyström Extension



Nyström Extension




MACHINE LEARNING
Random Walks on Data
Spectral Graph Theory
NLDR
Diffusion Maps
STOCHASTIC DYNAMICS
DIFFERENTIAL GEOMETRY
Brownian Motion on Manifolds
Heat Equation

STOCHASTIC DYNAMICAL SYSTEMS

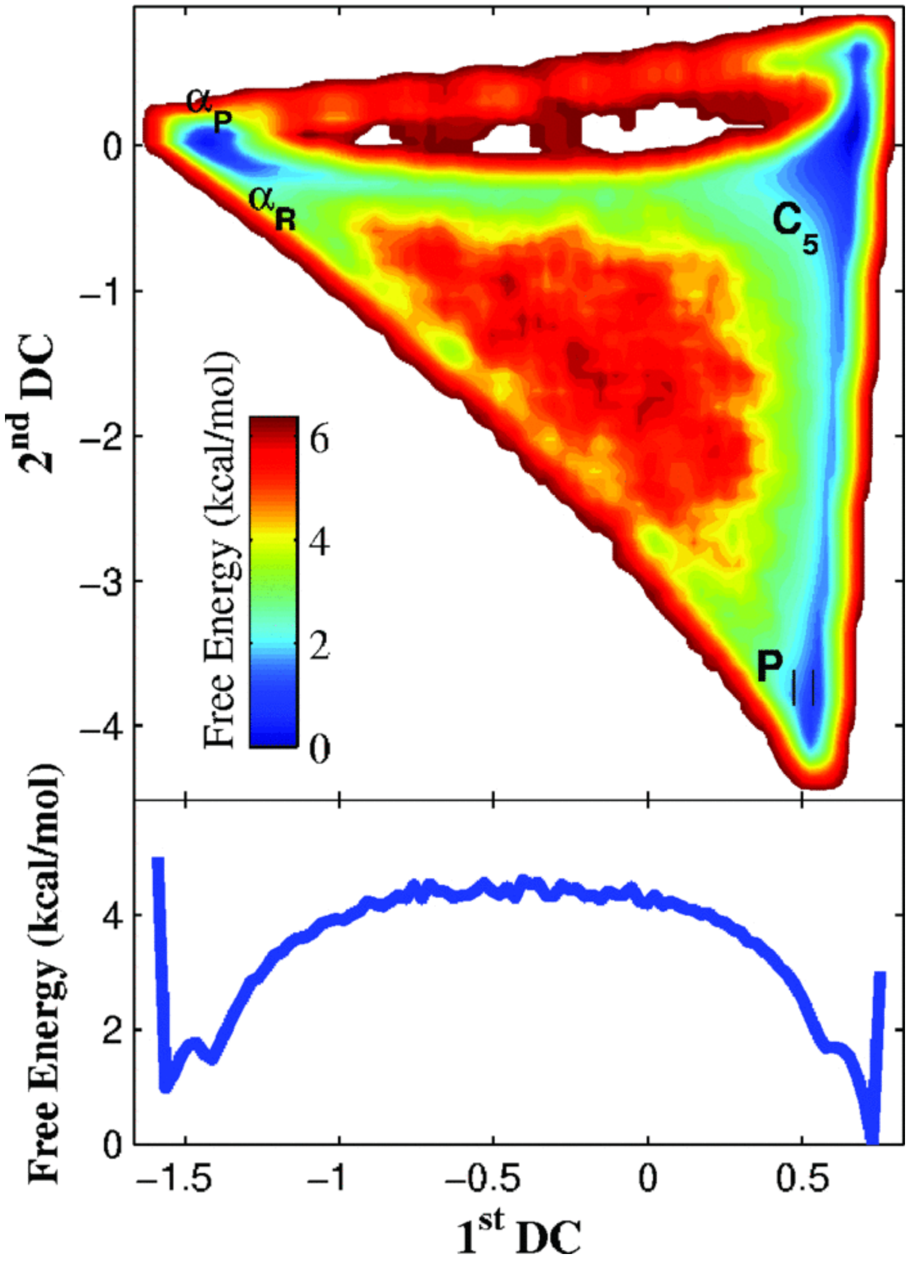
Ongoing work with
Paul J. Atzberger

Rohrdanz, Zheng, Maggioni, Clementi (2011)

NLDR of simulations

In some (stochastic) dynamical systems, the overall behavior doesn't deviate far from a manifold.

Applying Diffusion Maps to molecular dynamics can reveal reaction coordinates (right)



Itô diffusion

Integral Form:

$$X_t = X_0 + \int_0^t a(s, X_s)ds + \int_0^t b(s, X_s)dW_t$$

Ordinary Integral

Itô Integral

Differential Notation:

$$dX_t = a(t, X_t)dt + b(t, X_t)dW_t$$

Drift Term

Diffusion Term

Itô Lemma (Stochastic Chain Rule):

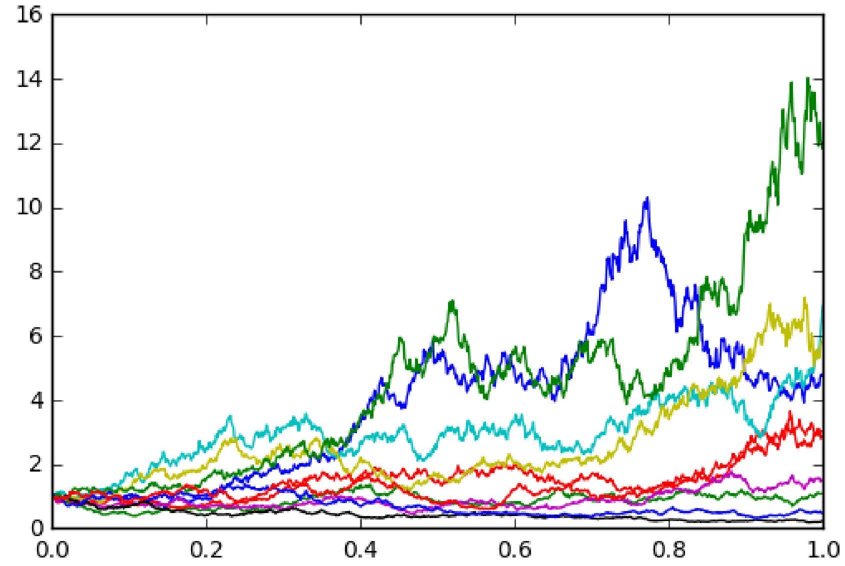
Traditional Chain Rule

Ito's Correction

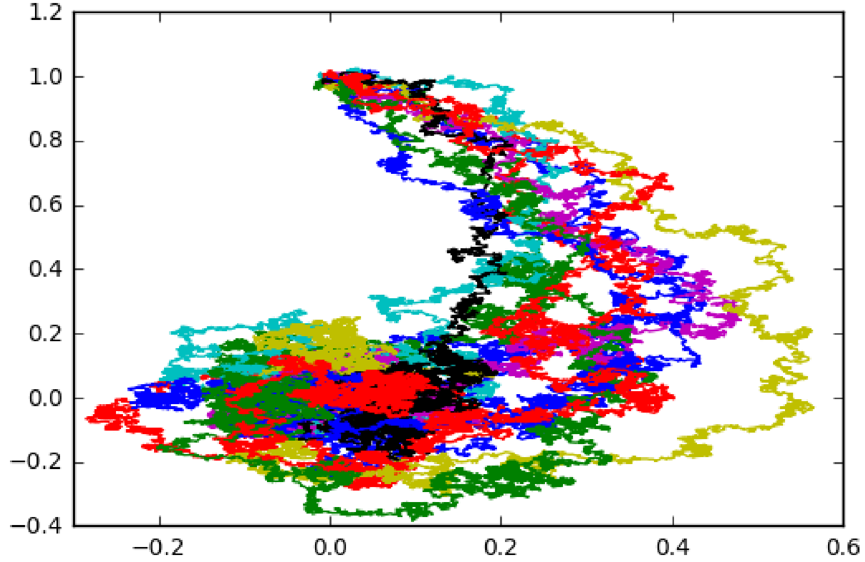
$$d[f(t, X_t)] = \boxed{\frac{\partial f}{\partial t}(t, X_t)dt + \frac{\partial f}{\partial x}(t, X_t)dX_t} + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(t, X_t)dX_t \cdot dX_t$$

$dt \cdot dt = dt \cdot dW_t = 0$ and $dW_t \cdot dW_t = dt$

Geometric Brownian Motion (1D)



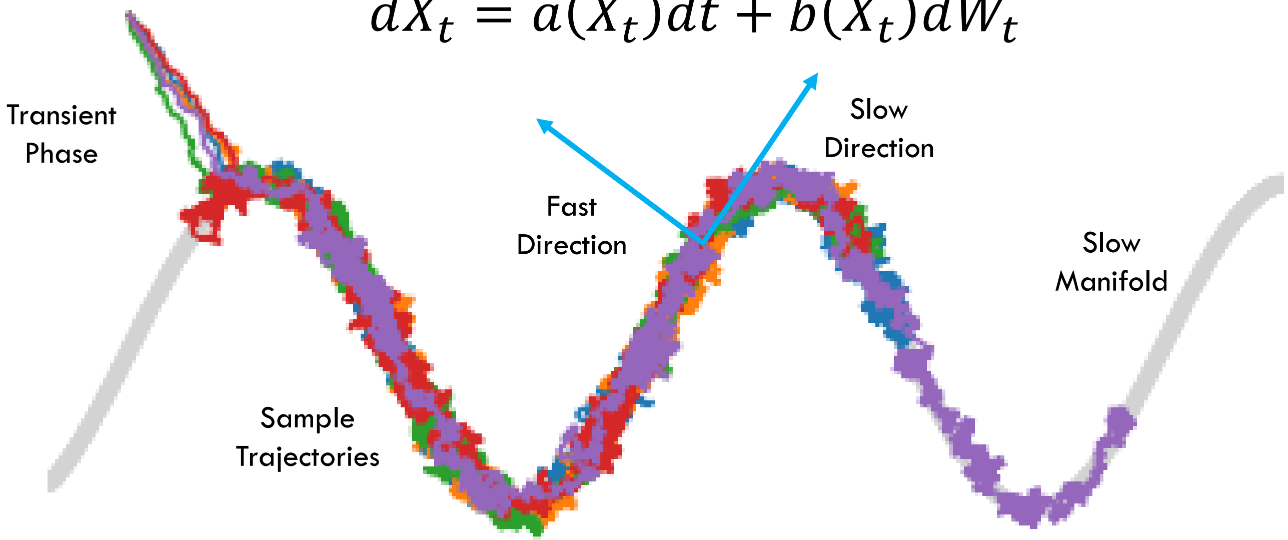
Langevin Dynamics (2D)



Itô Diffusion Assumption

$X_t \in \mathbb{R}^N$ **High** dimensional SDE:

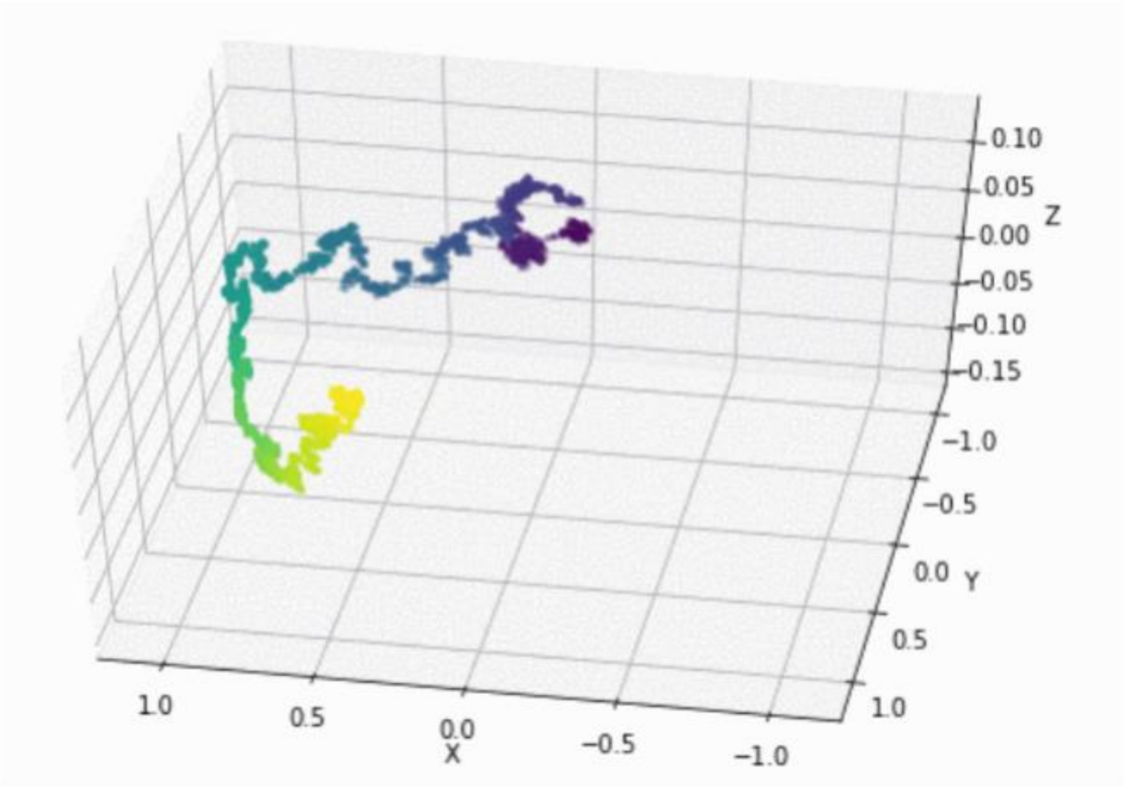
$$dX_t = a(X_t)dt + b(X_t)dW_t$$

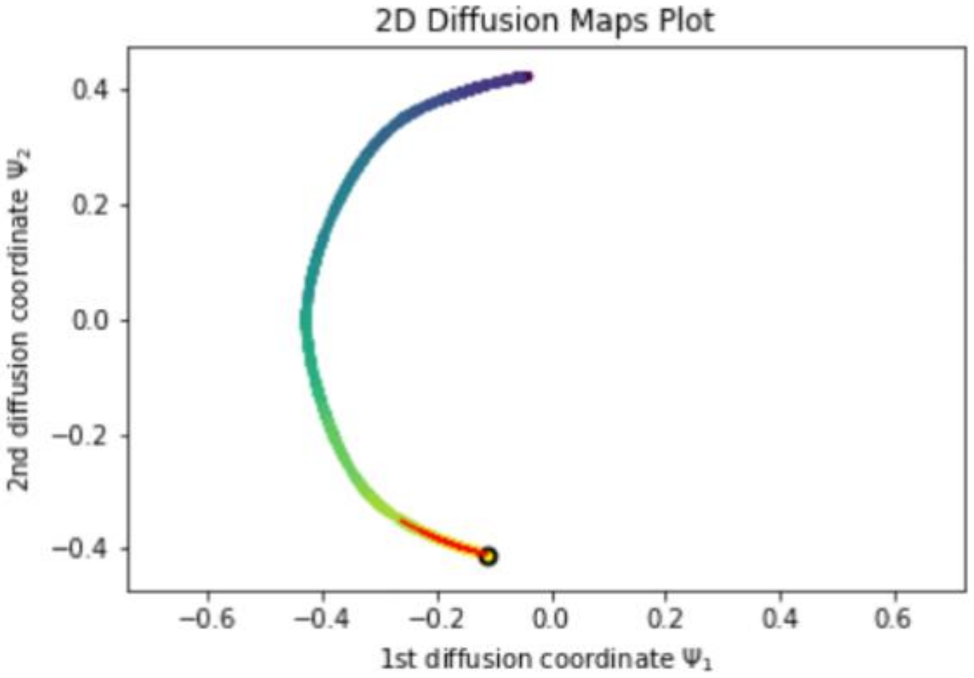


$Z_t = \Psi(X_t) \in \mathbb{R}^n$ ($n \ll N$) **Low** dimensional SDE:

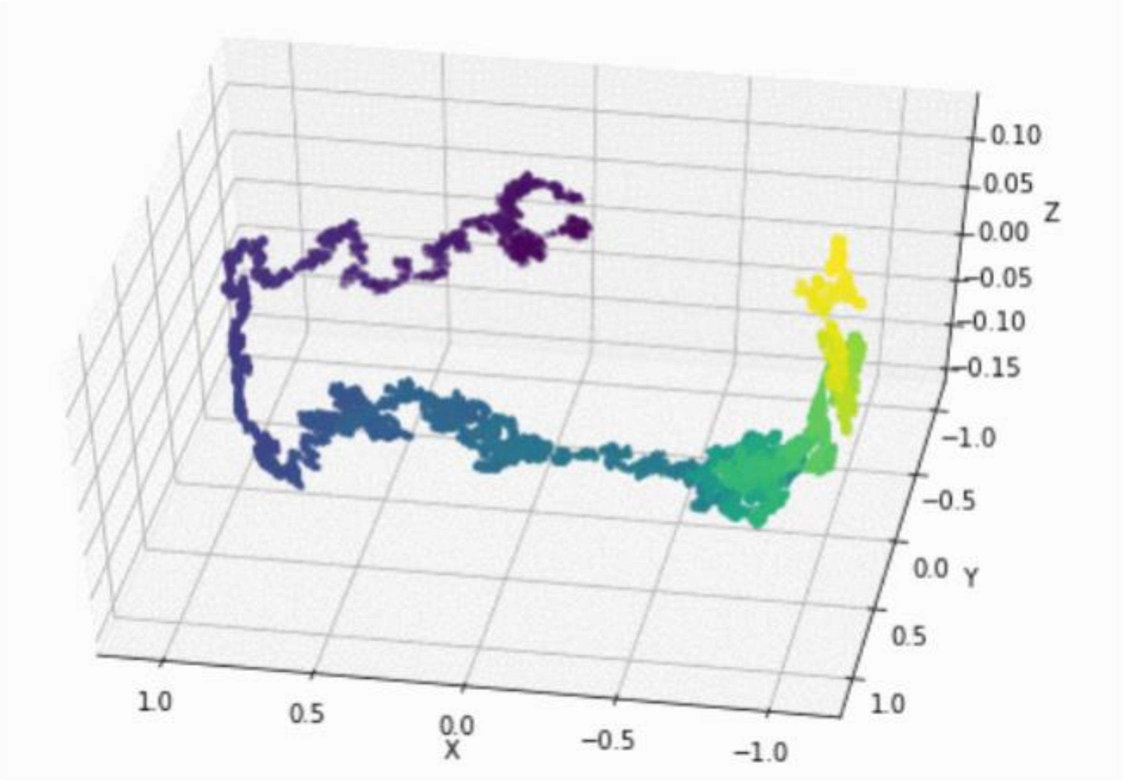
$$dZ_t = \tilde{a}(Z_t)dt + \tilde{b}(Z_t)dW_t$$

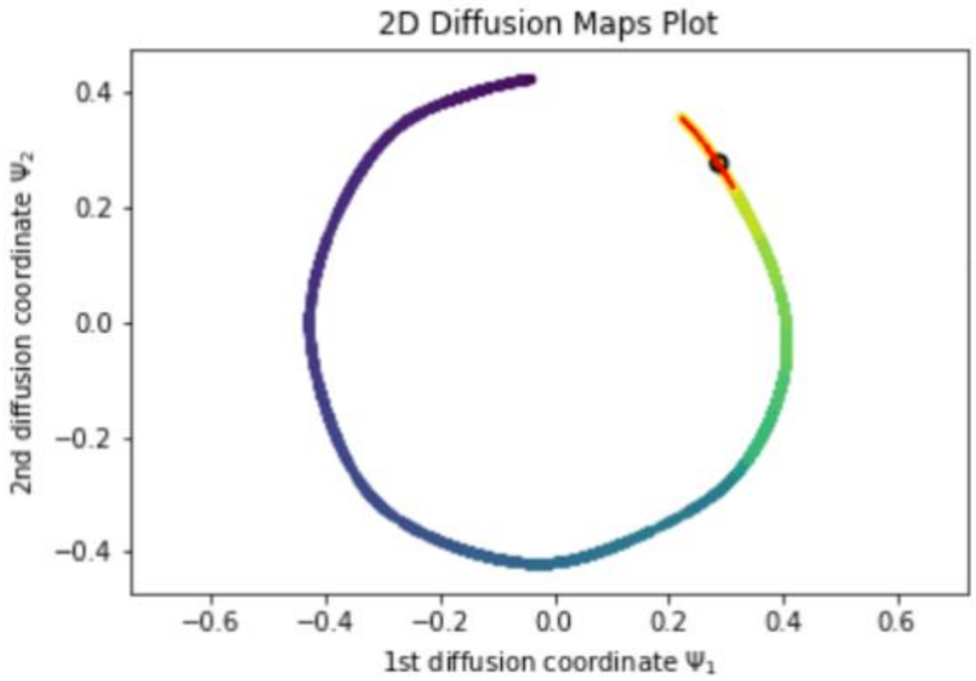
Simulate SDE



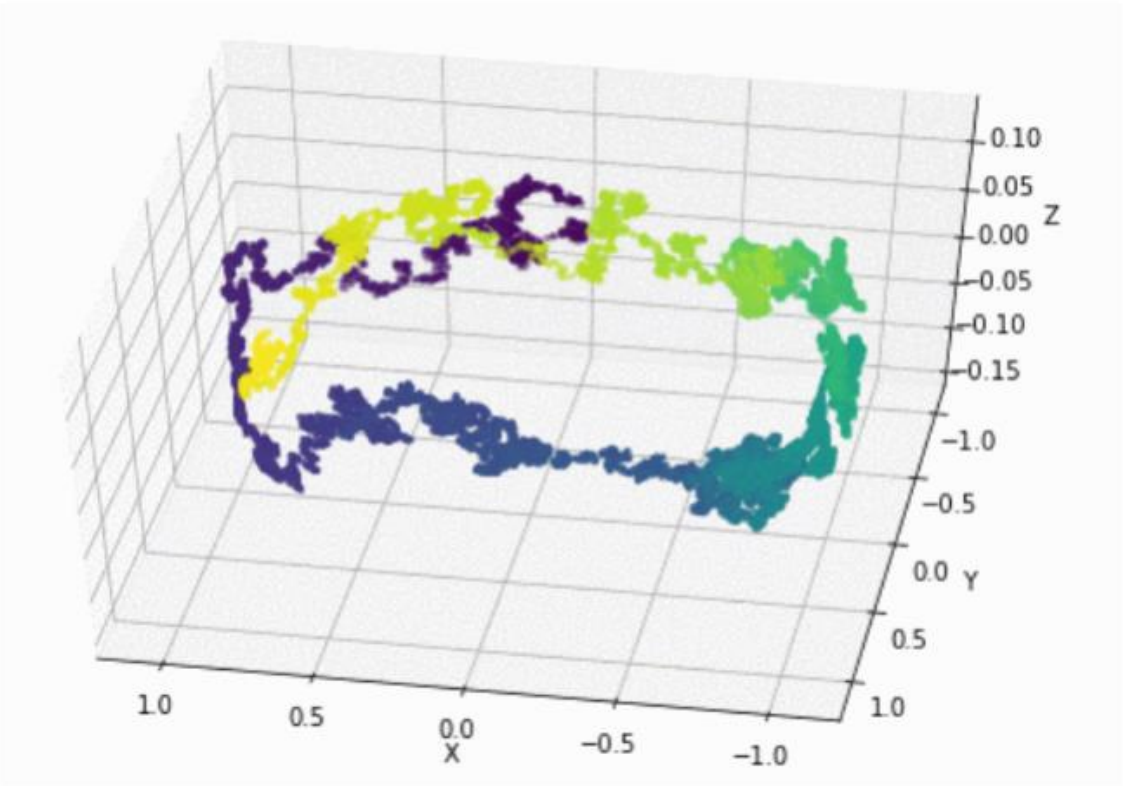


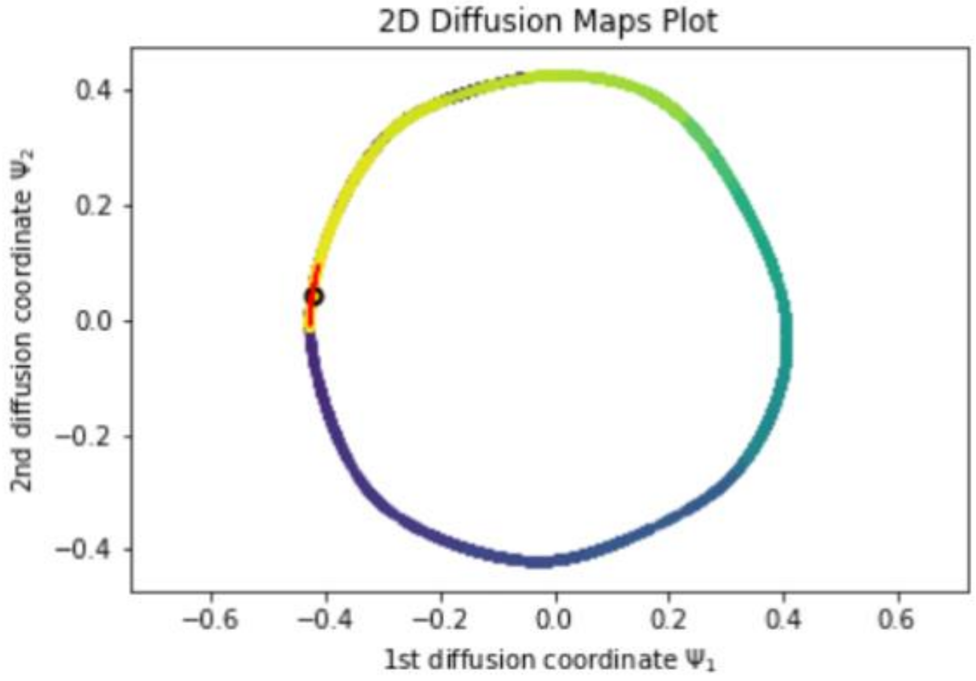
Simulate SDE



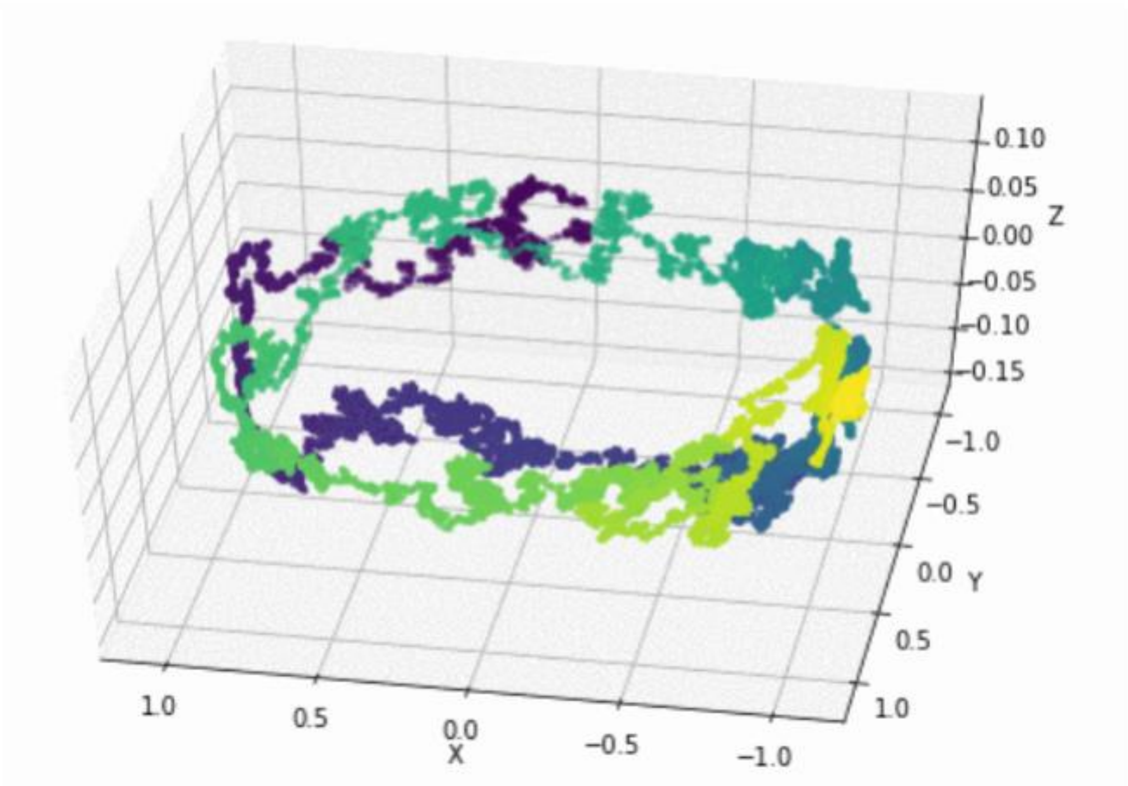


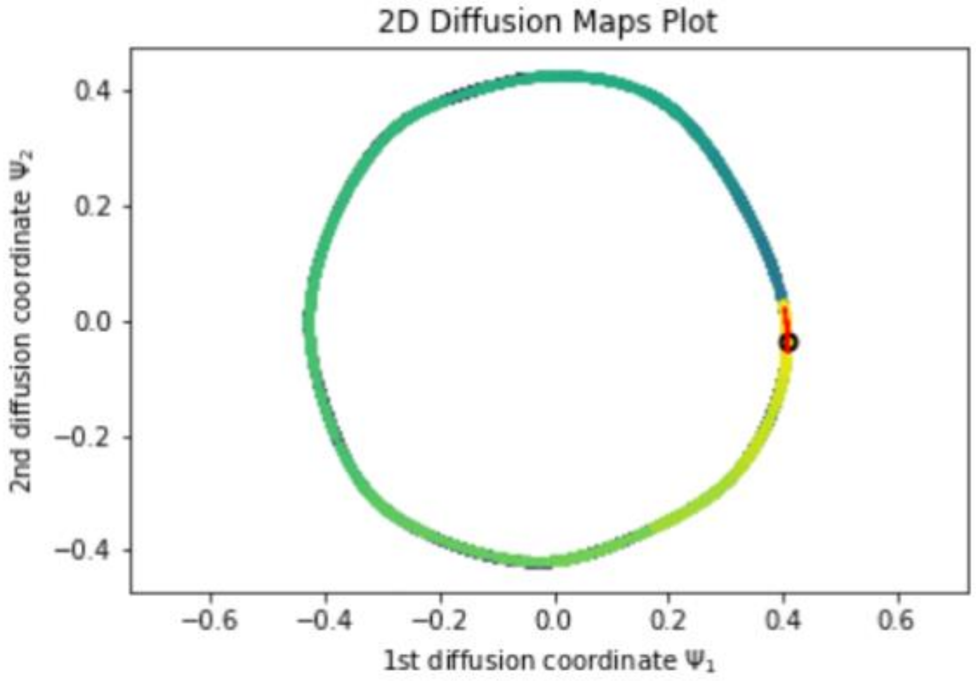
Simulate SDE



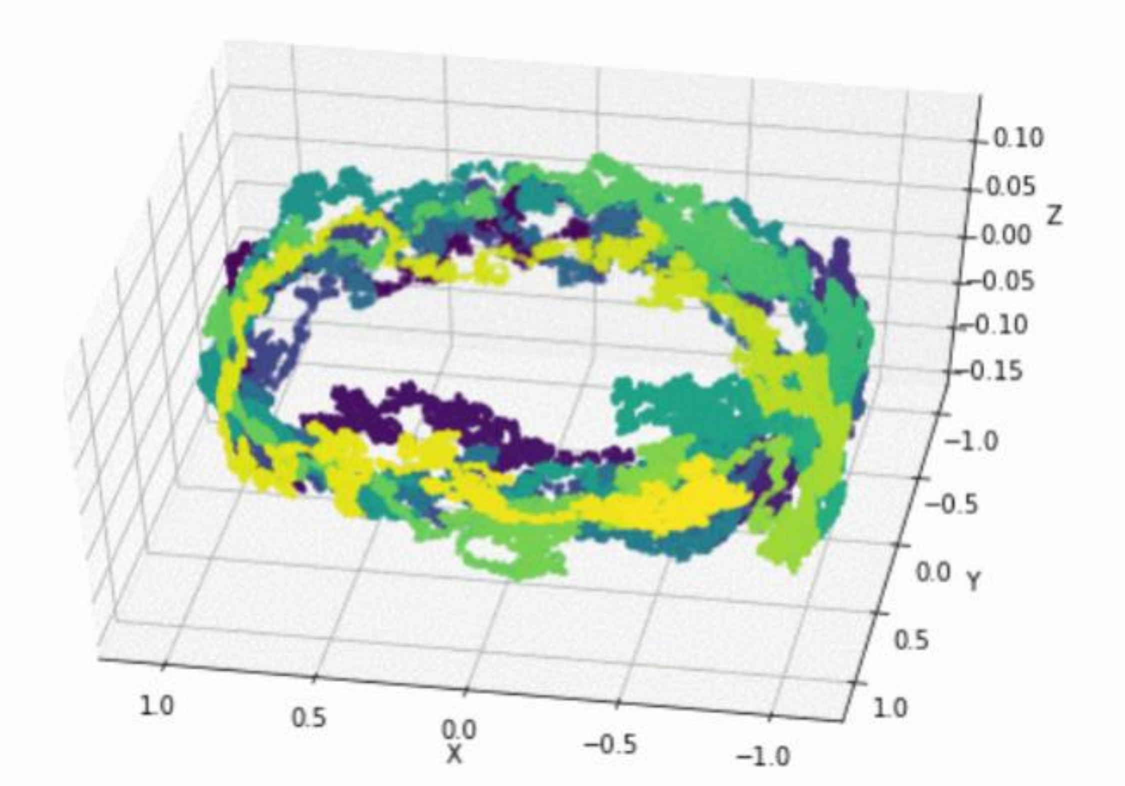


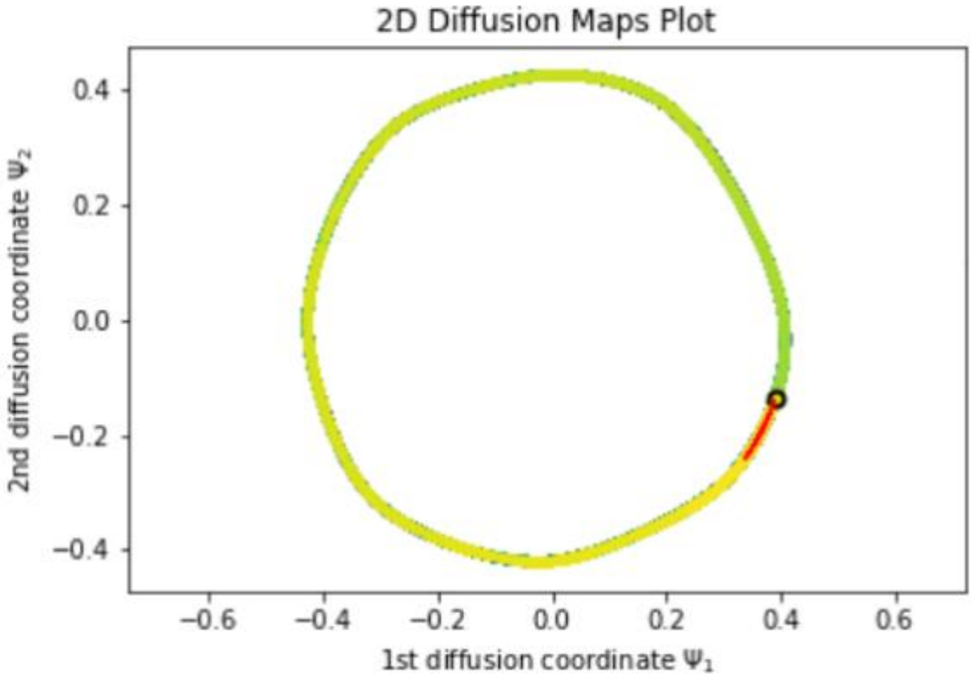
Simulate SDE



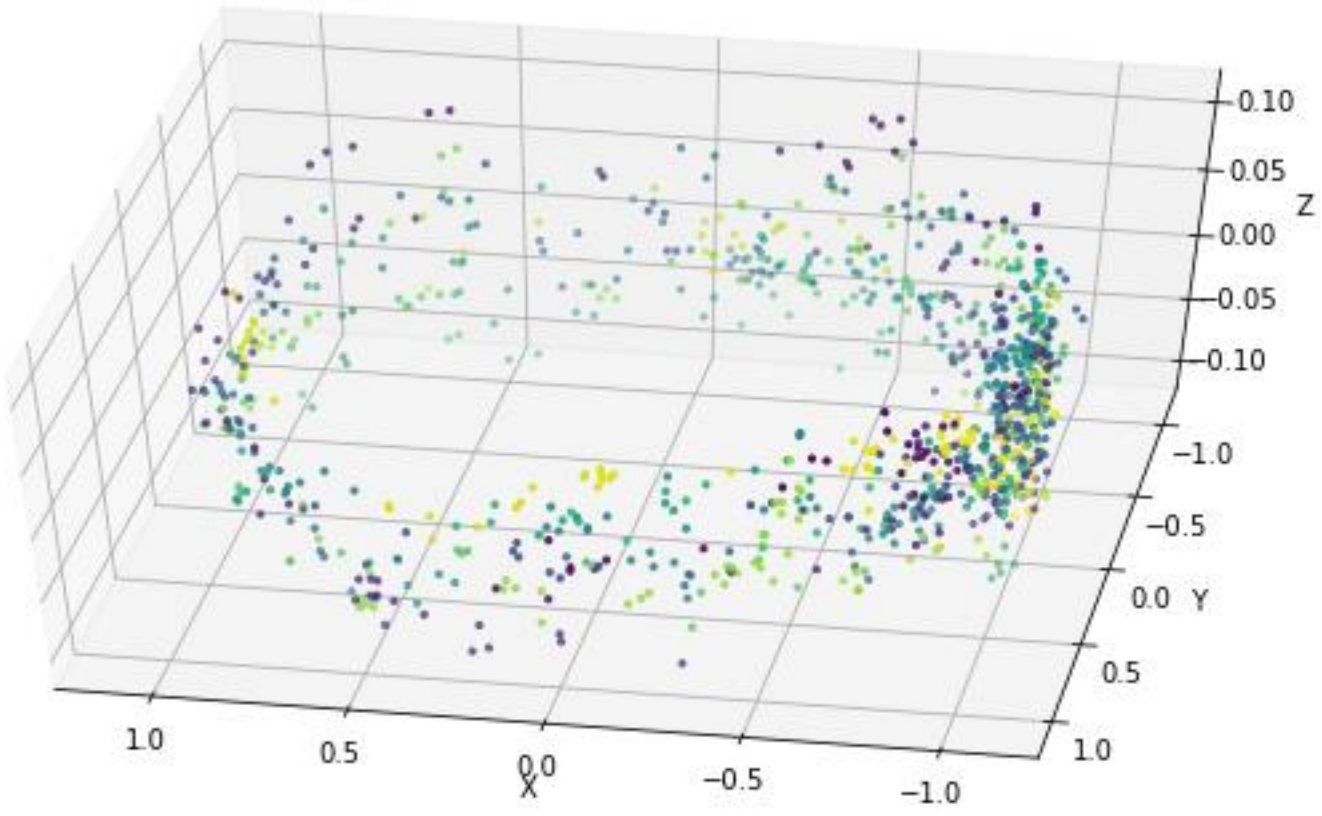


Simulate SDE

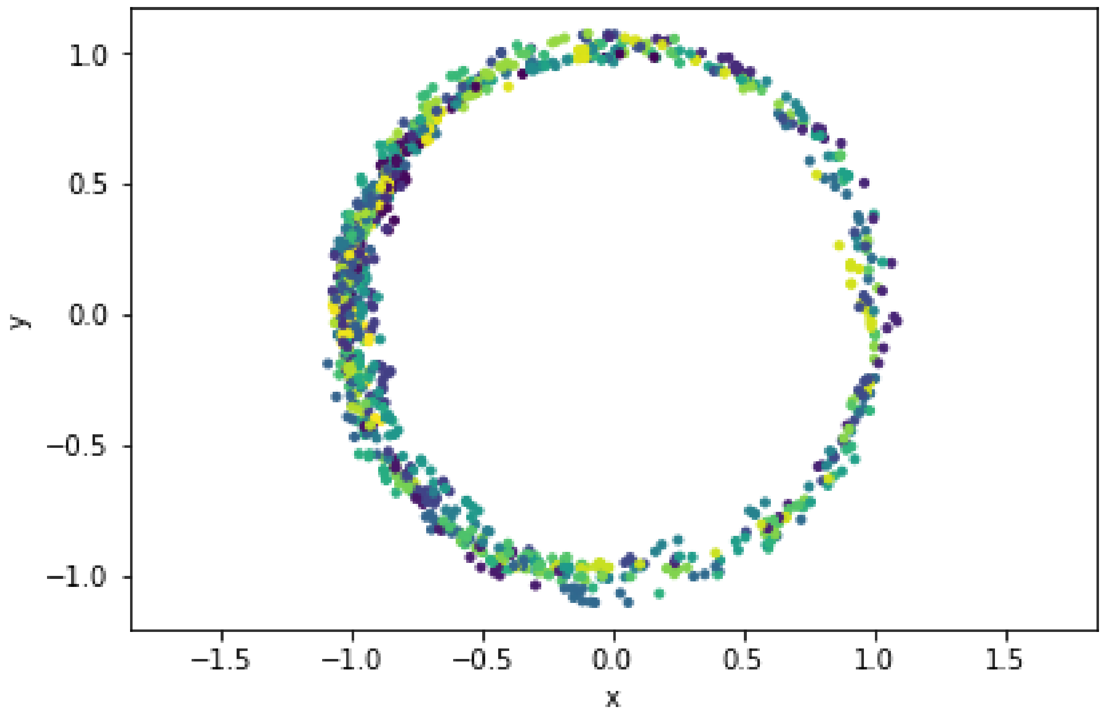




Downsample (Optional)

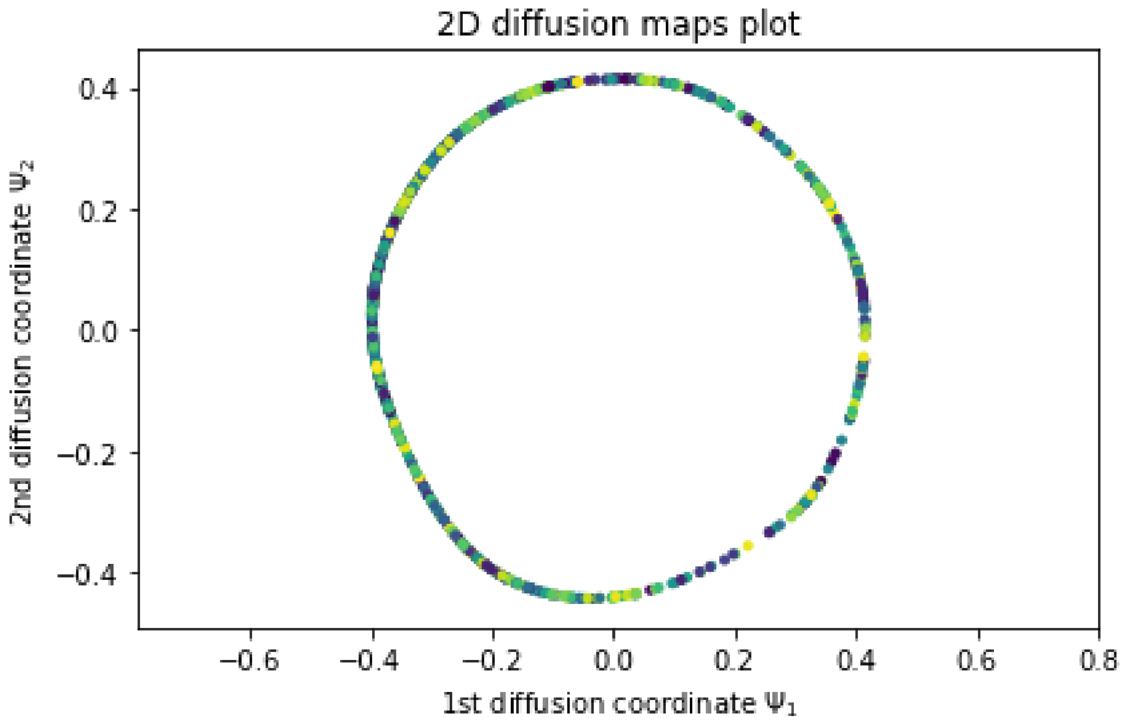


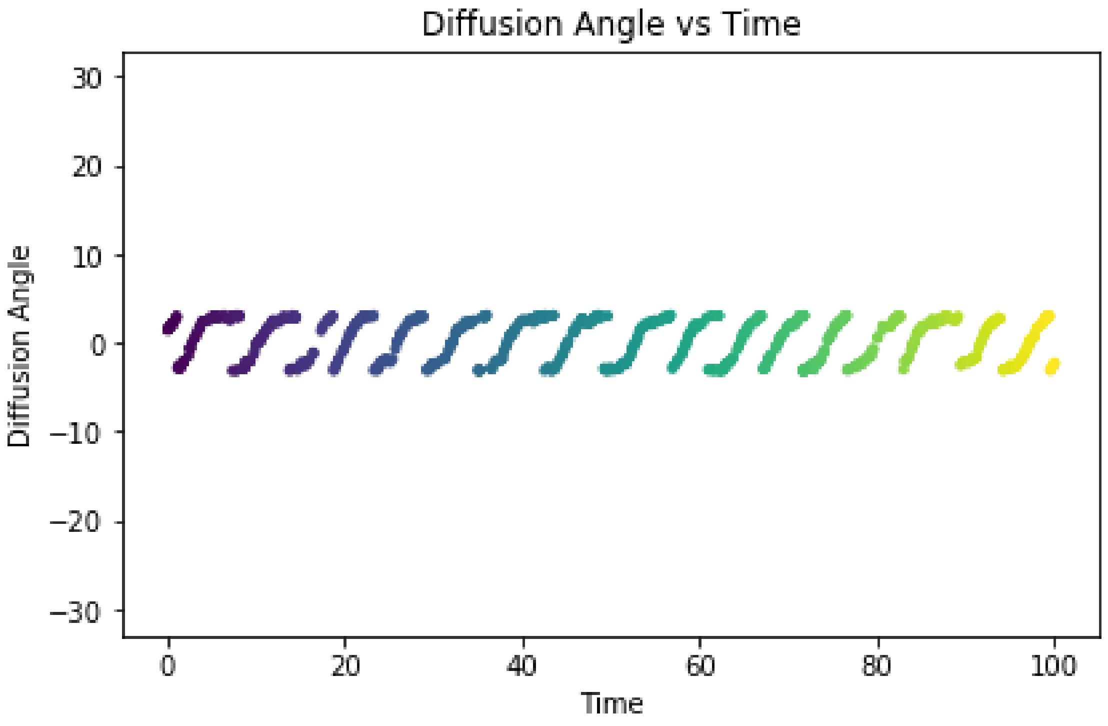
XYZ Plot



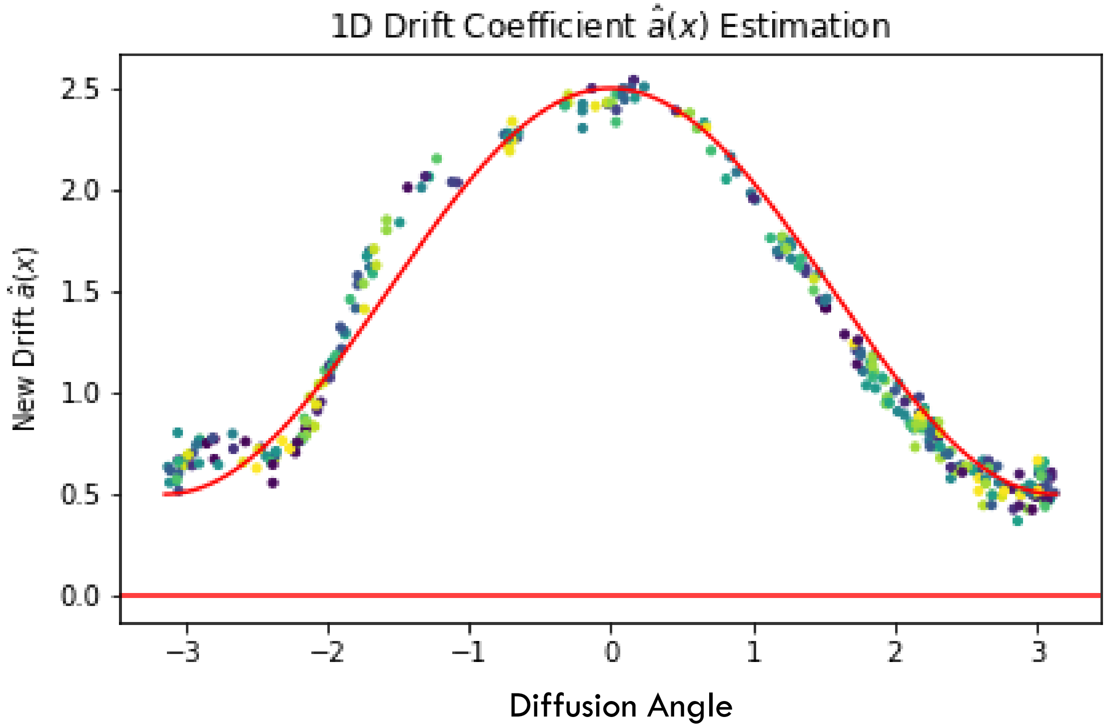
XY Orthogonal Projection

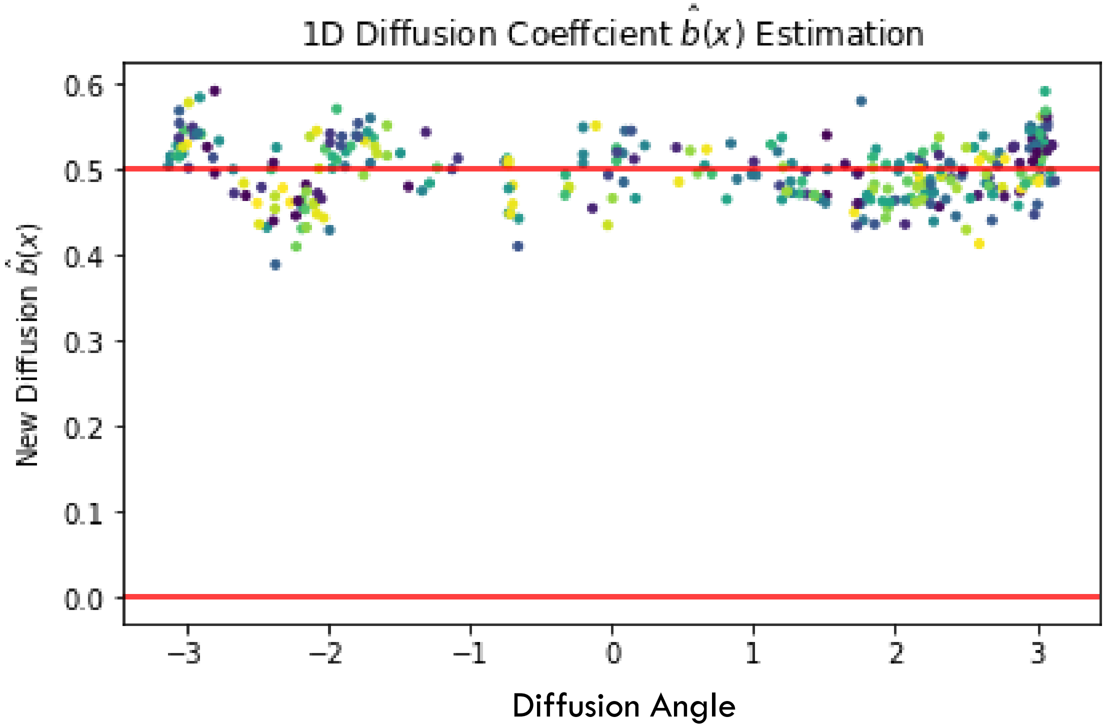
Apply Diffusion Maps





Estimate Drift and Diffusion Coefficients





REFERENCES

- M. Kac, “Can One Hear the Shape of a Drum?” (1966).
- D. E. Rumelhart, G. E. Hinton, R. J. Williams, “Learning representations by back-propagating errors.” (1986).
- B. Schölkopf, A. Smola, K. R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem.” (1998).
- M. do Carmo, “Riemmanian Geometry.” (1992).
- P. Bérard, G. Besson, S. Gallot, “Embedding Riemannian Manifolds by their Heat Kernel.” (1994)
- F. R. Chung, “Lectures on spectral graph theory.” CBMS Lectures, Fresno, (1996).
- I. T. Jolliffe, “Principal Component Analysis.” 2nd Edition (2002).
- B. Øksendal, “Stochastic Differential Equations: An Introduction with Applications.” (2003).
- M. Belkin, P. Niyogi, “Laplacian Eigenmaps for Dimensionality Reduction and Data Representation.” (2003).
- Y. Bengio, J. F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, M. Ouimet, “Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering.” (2004)
- R. Coifman, S. Lafon, “Diffusion Maps.” (2006).
- M. A. A. Cox, and T. F. Cox., "Multidimensional scaling." (2008).
- P.W. Jones, M. Maggioni, R. Schul, “Manifold Parametrizations by eigenfunctions of the Laplacian and heat kernels.” (2008).
- R. Coifman, I.G. Kevrekidis, S. Lafon, M. Maggioni, B. Nadler, “Diffusion Maps, Reduction Coordinates, and Low Dimensional Representation of Stochastic Systems.” (2008).
- L. Maaten, E. Postma, J. Herik, “Dimensionality Reduction: A Comparative Review.” (2009)
- M. A. Rohrdanz, W. Zheng, M. Maggioni, C. Clementi, “Determination of reaction coordinates via locally scaled diffusion map.” (2011).
- T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient estimation of word representations in vector space.” (2013).
- H. Strange, R. Zwiggelaar, “Open Problems in Spectral Dimensionality Reduction.” (2014).

THANK YOU FOR YOUR TIME!