Mini-Course on Dimensionality Reduction and Manifold Learning
Part 2: Nonlinear Dimensionality Reduction

Keaton Hamm
University of Texas at Arlington

November 23, 2022

Manifold Hypothesis: Data lies on (or near) a manifold $\mathcal{M}$ embedded in $\mathbb{R}^m$. (Manifolds are topological spaces that are that locally homeomorphic to $\mathbb{R}^d$ for some d – this d is the same for the whole manifold and is it's dimension)

Manifold Hypothesis: Data lies on (or near) a manifold $\mathcal{M}$ embedded in $\mathbb{R}^m$. (Manifolds are topological spaces that are that locally homeomorphic to $\mathbb{R}^d$ for some d – this d is the same for the whole manifold and is it's dimension)

Often an implicit (though sometimes explicit) hypothesis for Machine Learning methods, e.g., Deep Neural Network classifiers

Manifold Hypothesis: Data lies on (or near) a manifold $\mathcal{M}$ embedded in $\mathbb{R}^m$. (Manifolds are topological spaces that are that locally homeomorphic to $\mathbb{R}^d$ for some d – this d is the same for the whole manifold and is it's dimension)

Often an implicit (though sometimes explicit) hypothesis for Machine Learning methods, e.g., Deep Neural Network classifiers

More generally: data can come from union of manifolds

Problem 3: Preserve distance structure / geometry of data while significantly reducing the dimension

Given: $\{x_i\}_{i=1}^N \subset \mathcal{M}$ d-dimensional smooth manifold embedded in $\mathbb{R}^D$

Find: $\phi : \mathbb{R}^D \to \mathbb{R}^m, \quad d \leq m \ll D$ such that

$$|\phi(x_i) - \phi(x_j)| \approx d_M(x_i, x_j).$$

ISOMAP

The ISOMAP algorithm [Tenenbaum, de Silva, Langford, 2000]

ISOMAP

The ISOMAP algorithm [Tenenbaum, de Silva, Langford, 2000]

Given: $X \subset \mathbb{R}^D$ that we suspect lies on a d–dimensional manifold $\mathcal{M}$ (note d is a parameter here)

The ISOMAP algorithm [Tenenbaum, de Silva, Langford, 2000]

Given: $X \subset \mathbb{R}^D$ that we suspect lies on a d–dimensional manifold $\mathcal{M}$ (note d is a parameter here)

Step 1: Estimate geodesic distances $D_{ij} \approx d_{\mathcal{M}}(x_i, x_j)$

ISOMAP

The ISOMAP algorithm [Tenenbaum, de Silva, Langford, 2000]

Given: $X \subset \mathbb{R}^D$ that we suspect lies on a d–dimensional manifold $\mathcal{M}$ (note d is a parameter here)

Step 1: Estimate geodesic distances $D_{ij} \approx d_{\mathcal{M}}(x_i, x_j)$

Step 2: Run MDS on D (i.e., $B = -\frac{1}{2}JD^{(2)}J$ and compute $B = V_d\Lambda_d V_d^T$) and set $y_i = (V_d\Lambda_d^{\frac{1}{2}})_{i:}$

ISOMAP

The ISOMAP algorithm [Tenenbaum, de Silva, Langford, 2000]

Given: $X \subset \mathbb{R}^D$ that we suspect lies on a d–dimensional manifold $\mathcal{M}$ (note d is a parameter here)

Step 1: Estimate geodesic distances $D_{ij} \approx d_{\mathcal{M}}(x_i, x_j)$

Step 2: Run MDS on D (i.e., $B = -\frac{1}{2}JD^{(2)}J$ and compute $B = V_d \Lambda_d V_d^T$) and set $y_i = (V_d \Lambda_d^{\frac{1}{2}})_{i:}$

The tricky part is Step 1

Estimating geodesics:

ISOMAP

Estimating geodesics:

Step 1: Make a graph $G = (X, E, w)$ (e.g., ε–neighborhood or k–NN)

ISOMAP

Estimating geodesics:

Step 1: Make a graph $G = (X, E, w)$ (e.g., ε–neighborhood or k–NN)

Step 2: Compute graph shortest path distances (sometimes called APSP, or All-Pairs Shortest Path in the theoretical CS literature)

ISOMAP

Estimating geodesics:

Step 1: Make a graph $G = (X, E, w)$ (e.g., ε–neighborhood or k–NN)

Step 2: Compute graph shortest path distances (sometimes called APSP, or All-Pairs Shortest Path in the theoretical CS literature)

Set $D_{ij} = d_G(x_i, x_j)$ – the expectation is that $d_G(x_i, x_j) \approx d_{\mathcal{M}}(x_i, x_j)$

Graphs From Data

Fundamental Question: Given data structure, and a task, how do we form a graph from the data to accomplish our task well?

Graphs From Data

Fundamental Question: Given data structure, and a task, how do we form a graph from the data to accomplish our task well?

Unfortunately, there are few principled ways to do this in general

Graphs From Data

Fundamental Question: Given data structure, and a task, how do we form a graph from the data to accomplish our task well?

Unfortunately, there are few principled ways to do this in general

Given data $\{x_i\}_{i=1}^n \subset \mathbb{R}^m$, set

$$V = \{x_1, \ldots, x_n\}$$

Graphs From Data

Fundamental Question: Given data structure, and a task, how do we form a graph from the data to accomplish our task well?

Unfortunately, there are few principled ways to do this in general

Given data $\{x_i\}_{i=1}^n \subset \mathbb{R}^m$, set

$$V = \{x_1, \ldots, x_n\}$$

Many ways to define edges

Graphs From Data

Euclidean Graph – Connect all vertices to each other (complete graph); weights given by

$$w_{ij} := w(x_i, x_j) := |x_i - x_j|$$

Graphs From Data

Euclidean Graph – Connect all vertices to each other (complete graph); weights given by

$$w_{ij} := w(x_i, x_j) := |x_i - x_j|$$

ε–neighborhood graph – At each vertex, place ball of radius $\varepsilon > 0$ and connect to vertices inside that ball

$$w_{ij}^{\varepsilon} := \begin{cases} |x_i - x_j|, & |x_i - x_j| \leq \varepsilon \\ \infty, & \text{otherwise.} \end{cases}$$

Graphs From Data

Euclidean Graph – Connect all vertices to each other (complete graph); weights given by

$$w_{ij} := w(x_i, x_j) := |x_i - x_j|$$

ε–neighborhood graph – At each vertex, place ball of radius $\varepsilon > 0$ and connect to vertices inside that ball

$$w_{ij}^{\varepsilon} := \begin{cases} |x_i - x_j|, & |x_i - x_j| \leq \varepsilon \\ \infty, & \text{otherwise.} \end{cases}$$

k–Nearest Neighbors (kNN) graph – For $i = 1, \dots, n$, let K_i be the set of k nearest neighbors of x_i (nearness determined by Euclidean distance).

Graphs From Data

Euclidean Graph – Connect all vertices to each other (complete graph); weights given by

$$w_{ij} := w(x_i, x_j) := |x_i - x_j|$$

ε–neighborhood graph – At each vertex, place ball of radius $\varepsilon > 0$ and connect to vertices inside that ball

$$w_{ij}^{\varepsilon} := \begin{cases} |x_i - x_j|, & |x_i - x_j| \leq \varepsilon \\ \infty, & \text{otherwise.} \end{cases}$$

k–Nearest Neighbors (kNN) graph – For $i = 1, \ldots, n$, let K_i be the set of k nearest neighbors of x_i (nearness determined by Euclidean distance).
Connect $x_i \sim x_j$ with an edge of weight $|x_i - x_j|$ if $x_j \in K_i$.

Graphs From Data

Mutual kNN graph – $x_i \sim x_j$ iff $x_i \in K_j$ and $x_j \in K_i$

Graphs From Data

Mutual kNN graph – $x_i \sim x_j$ iff $x_i \in K_j$ and $x_j \in K_i$

Gaussian weighted graph – Fully connected with weights

$$w_{ij}^{\sigma} := e^{-\frac{|x_i - x_j|^2}{\sigma^2}}$$

Graphs From Data

Mutual kNN graph – $x_i \sim x_j$ iff $x_i \in K_j$ and $x_j \in K_i$

Gaussian weighted graph – Fully connected with weights

$$w_{ij}^{\sigma} := e^{-\frac{|x_i - x_j|^2}{\sigma^2}}$$

Note: σ, ε, and k are parameters that must be chosen. Poor choices can lead to bad results

ISOMAP Redux:

Step 1: Form a graph from the given data

Step 2: Compute APSP and set $D_{ij} = d_G(x_i, x_j)$

Step 3: Run MDS on D and set $y_i = (V_d \Lambda_d^{\frac{1}{2}})_{i:}$

ISOMAP Redux:

Step 1: Form a graph from the given data

Step 2: Compute APSP and set $D_{ij} = d_G(x_i, x_j)$

Step 3: Run MDS on D and set $y_i = (V_d \Lambda_d^{\frac{1}{2}})_{i:}$

Parameters: d – embedding dimension, and ε or k (for neighborhood graph)

ISOMAP

ISOMAP Redux:

Step 1: Form a graph from the given data

Step 2: Compute APSP and set $D_{ij} = d_G(x_i, x_j)$

Step 3: Run MDS on D and set $y_i = (V_d \Lambda_d^{\frac{1}{2}})_{i:}$

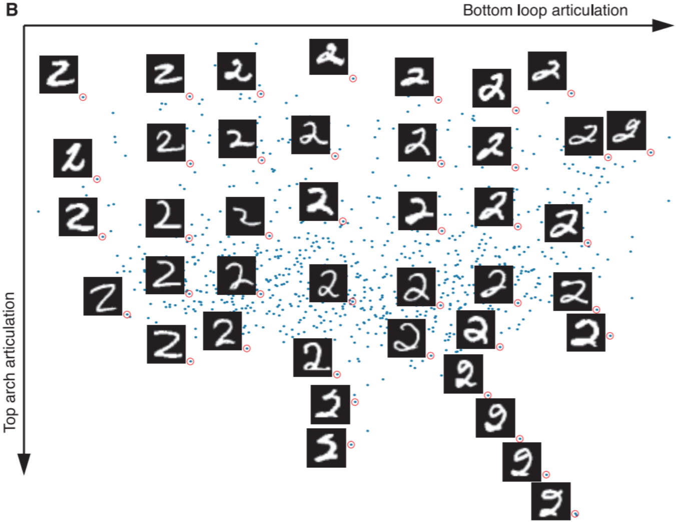
Parameters: d – embedding dimension, and ε or k (for neighborhood graph)

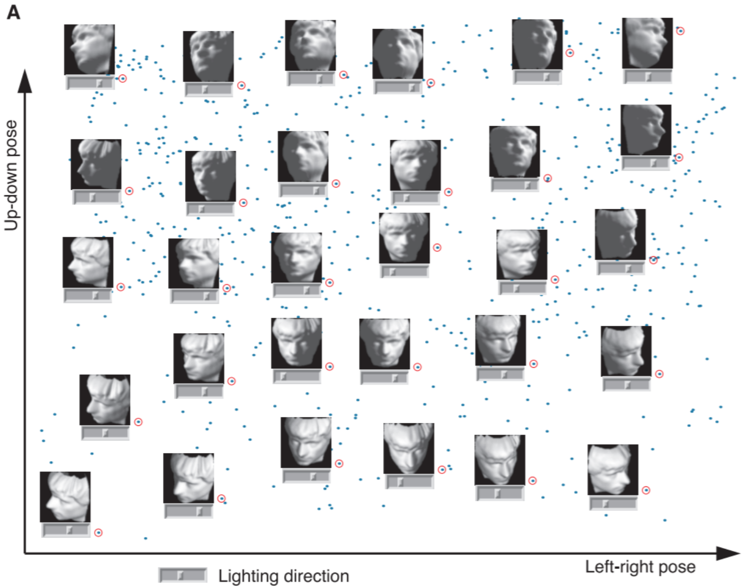
Also note: we need a large sampling of the manifold to ensure that $d_G \approx d_{\mathcal{M}}$

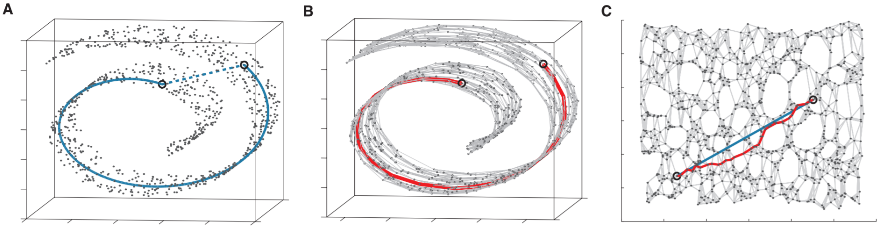
Measuring success is difficult, so ISOMAP is often used as exploratory analysis

ISOMAP

Measuring success is difficult, so ISOMAP is often used as exploratory analysis




A
Up-down pose
Left-right pose
Lighting direction


A
B
C

Exact Embeddings

If one knows $d_{\mathcal{M}}(x_i, x_j)$ and $\mathcal{M}$ is isometric up to a constant to $\Omega \subset \mathbb{R}^d$, then ISOMAP using the manifold distances is MDS on these, and hence the embedding satisfies $|y_i - y_j| = c d_{\mathcal{M}}(x_i, x_j)$.

Exact Embeddings

If one knows $d_{\mathcal{M}}(x_i, x_j)$ and $\mathcal{M}$ is isometric up to a constant to $\Omega \subset \mathbb{R}^d$, then ISOMAP using the manifold distances is MDS on these, and hence the embedding satisfies $|y_i - y_j| = c d_{\mathcal{M}}(x_i, x_j)$.

In general we need to understand

- How well d_{G^ε} approximates $d_{\mathcal{M}}$
- Perturbations of MDS embeddings

Geodesic Approximation

Theorem (Bernstein et al. '00, Arias-Castro and Le Gouic. '17)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a smooth, compact manifold with reach r. Suppose $X = \{x_i\} \subset \mathcal{M}$ is a δ–sampling of $\mathcal{M}$. If G^ε is an ε–neighborhood graph over X and $\varepsilon < r$, then

$$\left(1 - c_0\left(\frac{\varepsilon}{r}\right)^2\right) d_{\mathcal{M}}(x_i, x_j) \leq d_{G^\varepsilon}(x_i, x_j) \leq \left(1 + c_0\left(\frac{\delta}{\varepsilon}\right)^2\right) d_{\mathcal{M}}(x_i, x_j)$$

- δ–sampling: for every $x \in \mathcal{M}$, $d_{\mathcal{M}}(x, x_i) \leq \delta$ for some x_i
- reach: sup of $t \geq 0$ such that every point at distance t away from $\mathcal{M}$ has a unique closest point in $\mathcal{M}$

MDS Perturbation Bound

Theorem (Arias-Castro, Javanmard, Pelletier, '20)

Let $y_1, \dots, y_N \in \mathbb{R}^d$ be centered, span $\mathbb{R}^d$, and set $\Delta_{ij} = |y_i - y_j|^2$. Let $\{\Lambda_{ij}\}_{i,j=1}^N$ be arbitrary real numbers.

If $\|Y^\dagger\| \|\Lambda - \Delta\|_F^{\frac{1}{2}} \leq \frac{1}{\sqrt{2}}$, then MDS with input $\{\Lambda_{i,j}\}$ and embedding dimension d returns a point set $z_1, \dots, z_N \in \mathbb{R}^d$ satisfying

$$\min_{Q \in \mathcal{O}(d)} \|Z - YQ\|_F \leq (1 + \sqrt{2}) \|Y^\dagger\| \|\Lambda - \Delta\|_F.$$

ISOMAP Perturbation Bound

Theorem (Arias-Castro et al. '20)

Let $\mathcal{M}$ be as before and isometric to a convex subset of $\mathbb{R}^d$. Let $\xi = c_0 \max\left\{\left(\frac{\varepsilon}{r}\right)^2, \left(\frac{\delta}{\varepsilon}\right)^2\right\}$. Let $\{y_i\} \subset \mathbb{R}^d$ be an exact centered embedding of $\{x_i\} \subset \mathcal{M}$. If $\xi \leq \frac{1}{24}(\|Y\|\|Y^\dagger\|)^{-2}$, then ISOMAP returns points $\{z_i\} \subset \mathbb{R}^d$ such that

$$\min_{Q \in \mathcal{O}(d)} \|Z - YQ\|_F \leq 36\sqrt{d}\|Y\|^3\|Y^\dagger\|^2\xi.$$

Note: For fixed ε, small ξ corresponds to small reach and small δ (denser sampling) so that graph geodesics more closely approximate manifold geodesics.

Corollary

Let $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$ be arbitrary. Suppose $\mathcal{M} \subset \mathbb{R}^D$ is isometric to $\Omega \subset \mathbb{R}^d$, and $\{\hat{x}_i\}_{i=1}^N \subset \mathcal{M}$ and $\{y_i\} \subset \Omega$ are such that $|y_i - y_j| = |\hat{x}_i - \hat{x}_j|$.
Let $\Delta_{ij} := d_{\mathcal{M}}(\hat{x}_i, \hat{x}_j)^2$, $\Gamma_{ij} := |x_i - x_j|^2$, and $\Lambda_{ij} := \lambda_{ij}^2$ for some $\lambda_{ij} \in \mathbb{R}$.
Let z_i be the points given by MDS with embedding dimension d from Λ.
If $|\Gamma_{ij} - \Delta_{ij}| \leq \tau_1$ and $|\Delta_{ij} - \Lambda_{ij}| \leq \tau_2$, and if

$$\|Y^\dagger\|\sqrt{N}\,(\tau_1 + \tau_2)^{\frac{1}{2}} \leq \frac{1}{\sqrt{2}},$$

then $\{z_i\}_{i=1}^N \subset \mathbb{R}^d$ satisfies

$$\min_{Q \in \mathcal{O}(d)} \|Z - YQ\|_F \leq (1 + \sqrt{2})\|Y^\dagger\|N\,(\tau_1 + \tau_2)\,.$$

- τ_1 – how far away $\{x_i\}$ is away from the manifold samples
- τ_2 – how well geodesics are estimated

More Nonlinear Dimensionality Reduction Methods!

Now we will take a look at some similar methods:

- Local Linear Embedding (LLE) [Roweis and Saul ’00]
- Laplacian Eigenmaps [Belkin and Niyogi, ’03]
- Diffusion Maps [Coifman and Lafon, ’06]

Parameters: k – number of neighbors, d –embedding dimension

LLE

Parameters: k – number of neighbors, d –embedding dimension

Step 1: For all $x \in X$, compute the k nearest neighbors of X, $N_x^k \subset X$

LLE

Parameters: k – number of neighbors, d –embedding dimension

Step 1: For all $x \in X$, compute the k nearest neighbors of X, $N_x^k \subset X$

Step 2: For all $x_i \in X$, compute weights to best linearly reconstruct x from points in $N_{x_i}^k$:

$$\min_{w_{ij}} \|x - \sum_{j:x_j \in N_{x_i}^k} w_{ij} x_j\|_2^2$$

LLE

Parameters: k – number of neighbors, d –embedding dimension

Step 1: For all $x \in X$, compute the k nearest neighbors of X, $N_x^k \subset X$

Step 2: For all $x_i \in X$, compute weights to best linearly reconstruct x from points in $N_{x_i}^k$:

$$\min_{w_{ij}} \|x - \sum_{j:x_j \in N_{x_i}^k} w_{ij} x_j\|_2^2$$

Step 3: Compute embedding coordinates Y as follows: compute the SVD of $(I - W)^T(I - W) = V\Sigma V^T$, let V'_{N-d-1} contain columns $V_{:,N-d-1}, \dots, V_{:,N-1}$ (so we ignore the eigenvalue corresponding to $\lambda_N = 0$), and let $Y_i := V'_{i,N-d-1}$

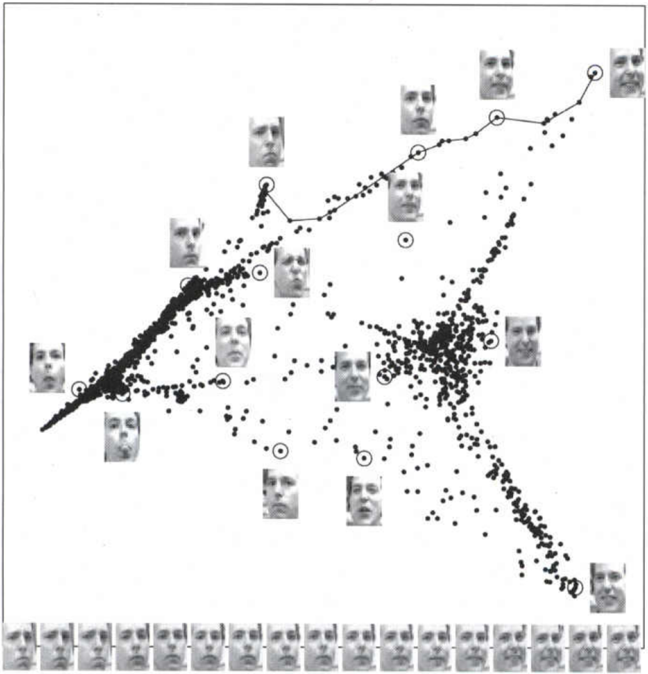
LLE [Roweis and Saul, ’00]

LLE is a local method – it reconstructs points via their nearest neighbors, then uses the graph structure of the weight matrix to find the embedding. This preserves high-dimensional neighborhoods in the embedding

LLE [Roweis and Saul, '00]

LLE is a local method – it reconstructs points via their nearest neighbors, then uses the graph structure of the weight matrix to find the embedding. This preserves high-dimensional neighborhoods in the embedding

LLE is more computationally efficient than ISOMAP (only deal with small neighborhoods of each point rather than estimating global geodesics)



Interlude – Graph Laplacians

Recall given $G = (V, E, w)$, D – degree matrix, $W = \{w_{ij}\}$ – weight matrix

Interlude – Graph Laplacians

Recall given $G = (V, E, w)$, D – degree matrix, $W = \{w_{ij}\}$ – weight matrix

The Unnormalized Graph Laplacian of G is

$$L = D - W$$

Interlude – Graph Laplacians

Recall given $G = (V, E, w)$, D – degree matrix, $W = \{w_{ij}\}$ – weight matrix

The Unnormalized Graph Laplacian of G is

$$L = D - W$$

The Symmetric, Normalized Graph Laplacian of G is

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

Interlude – Graph Laplacians

Recall given $G = (V, E, w)$, D – degree matrix, $W = \{w_{ij}\}$ – weight matrix

The Unnormalized Graph Laplacian of G is

$$L = D - W$$

The Symmetric, Normalized Graph Laplacian of G is

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

The Random Walk Graph Laplacian of G is

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

Properties of Graph Laplacians

Theorem

The following hold:

- $\forall\, x \in \mathbb{R}^n$,

$$\langle L_{\text{sym}} x, x\rangle = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}}\right)^2$$

- *L_{sym} and L_{rw} are SPSD*
- *(λ, u) is an eigenpair of L_{rw} iff $(\lambda, D^{\frac{1}{2}}u)$ is an eigenpair of L_{sym}*
- *$(0, \mathbb{1})$ is an eigenpair of L_{rw}. Hence $(0, D^{\frac{1}{2}}\mathbb{1})$ is an eigenpair of L_{sym}.*

Laplacian Eigenmaps [Belkin and Nyogi, ’03]

Parameters: ε/k for neibhborhood graph, d – embedding dimension

Laplacian Eigenmaps [Belkin and Nyogi, ’03]

Parameters: ε/k for neibhborhood graph, d – embedding dimension
Step 1: Create ε–neighborhood or k–NN graph over X, but weight edges as

$$w_{ij} = e^{-\frac{|x_i - x_j|_2^2}{\sigma^2}}, \quad (i,j) \in E.$$

Laplacian Eigenmaps [Belkin and Nyogi, '03]

Parameters: ε/k for neibhborhood graph, d – embedding dimension
Step 1: Create ε–neighborhood or k–NN graph over X, but weight edges as

$$w_{ij} = e^{-\frac{|x_i - x_j|_2^2}{\sigma^2}}, \quad (i,j) \in E.$$

Step 2: For each connected component of G, solve the generalized eigenvalue problem

$$Lx = \lambda Dx, \quad (L = D - W)$$

Laplacian Eigenmaps [Belkin and Nyogi, '03]

Step 3: Embedded points are

$$y_i = V'_{i,N-d-1}$$

(as in LLE)

Diffusion Maps [Coifman and Lafon, '06]

Start with a graph as before. Consider a random walk on the graph, with transition probabilities

$$\mathbb{P}[X(t+1) = j \mid X(t) = i] = \frac{w_{ij}}{d_i}$$

Diffusion Maps [Coifman and Lafon, ’06]

Start with a graph as before. Consider a random walk on the graph, with transition probabilities

$$\mathbb{P}[X(t+1) = j \mid X(t) = i] = \frac{w_{ij}}{d_i}$$

$$M_{ij} := \frac{w_{ij}}{d_i} \Rightarrow M = D^{-1}W$$

Diffusion Maps [Coifman and Lafon, ’06]

Start with a graph as before. Consider a random walk on the graph, with transition probabilities

$$\mathbb{P}[X(t+1) = j \mid X(t) = i] = \frac{w_{ij}}{d_i}$$

$$M_{ij} := \frac{w_{ij}}{d_i} \Rightarrow M = D^{-1}W$$

Now suppose we start a walk at node i and look at where we get at time t

Diffusion Maps [Coifman and Lafon, '06]

Start with a graph as before. Consider a random walk on the graph, with transition probabilities

$$\mathbb{P}[X(t+1) = j \mid X(t) = i] = \frac{w_{ij}}{d_i}$$

$$M_{ij} := \frac{w_{ij}}{d_i} \Rightarrow M = D^{-1}W$$

Now suppose we start a walk at node i and look at where we get at time t

$$\mathbb{P}[X(t) = j \mid X(0) = i] = (M^t)_{ij}$$

Diffusion Maps [Coifman and Lafon, '06]

Start with a graph as before. Consider a random walk on the graph, with transition probabilities

$$\mathbb{P}[X(t+1) = j \mid X(t) = i] = \frac{w_{ij}}{d_i}$$

$$M_{ij} := \frac{w_{ij}}{d_i} \Rightarrow M = D^{-1}W$$

Now suppose we start a walk at node i and look at where we get at time t

$$\mathbb{P}[X(t) = j \mid X(0) = i] = (M^t)_{ij}$$

Thus the “probability cloud" of points with their probabilities of the random walker at time t is the row $M^t_{i:}$

Diffusion Maps [Coifman and Lafon, '06]

Note: we could very well represent the graph by $M^t_{i:}$, but this would have embedding dimension $n = |V|$, which isn't good. So let's keep working.

Diffusion Maps [Coifman and Lafon, '06]

Note: we could very well represent the graph by $M^t_{i:}$, but this would have embedding dimension $n = |V|$, which isn't good. So let's keep working.

$$M_s := D^{\frac{1}{2}} M D^{-\frac{1}{2}} = V \Lambda V^T$$

Diffusion Maps [Coifman and Lafon, '06]

Note: we could very well represent the graph by $M^t_{i:}$, but this would have embedding dimension $n = |V|$, which isn't good. So let's keep working.

$$M_s := D^{\frac{1}{2}} M D^{-\frac{1}{2}} = V \Lambda V^T$$

Note:

$$M = D^{-\frac{1}{2}} M_s D^{\frac{1}{2}} = D^{-\frac{1}{2}} V \Lambda V^T D^{\frac{1}{2}} = \left(D^{-\frac{1}{2}} V\right) \Lambda \left(D^{\frac{1}{2}} V\right)^T =: \Phi \Lambda \Psi^T.$$

Diffusion Maps [Coifman and Lafon, ’06]

Note: we could very well represent the graph by $M^t_{i:}$, but this would have embedding dimension $n = |V|$, which isn’t good. So let’s keep working.

$$M_s := D^{\frac{1}{2}} M D^{-\frac{1}{2}} = V \Lambda V^T$$

Note:

$$M = D^{-\frac{1}{2}} M_s D^{\frac{1}{2}} = D^{-\frac{1}{2}} V \Lambda V^T D^{\frac{1}{2}} = \left(D^{-\frac{1}{2}} V\right) \Lambda \left(D^{\frac{1}{2}} V\right)^T =: \Phi \Lambda \Psi^T.$$

Note: Φ, Ψ form a biorthogonal system – i.e., $\Psi^T \Phi = \Phi^T \Psi = I$, equivalently $\phi_i^T \psi_j = \delta_{ij}$

Diffusion Maps [Coifman and Lafon, '06]

Note: we could very well represent the graph by $M^t_{i:}$, but this would have embedding dimension $n = |V|$, which isn't good. So let's keep working.

$$M_s := D^{\frac{1}{2}} M D^{-\frac{1}{2}} = V \Lambda V^T$$

Note:

$$M = D^{-\frac{1}{2}} M_s D^{\frac{1}{2}} = D^{-\frac{1}{2}} V \Lambda V^T D^{\frac{1}{2}} = \left(D^{-\frac{1}{2}} V\right) \Lambda \left(D^{\frac{1}{2}} V\right)^T =: \Phi \Lambda \Psi^T.$$

Note: Φ, Ψ form a biorthogonal system – i.e., $\Psi^T \Phi = \Phi^T \Psi = I$, equivalently $\phi_i^T \psi_j = \delta_{ij}$
Also,

$$M\phi_k = \lambda_k \phi_k, \qquad \psi_k^T M = \lambda_k \phi_k^T$$

Diffusion Maps [Coifman and Lafon, '06]

From $M = \Phi \Lambda \Psi^T$,

$$M = \sum_{i=1}^{n} \lambda_i \phi_i \psi_i^T$$

Diffusion Maps [Coifman and Lafon, '06]

From $M = \Phi\Lambda\Psi^T$,

$$M = \sum_{i=1}^{n} \lambda_i \phi_i \psi_i^T$$

Thus

$$M^t = \sum_{i=1}^{n} \lambda_i^t \phi_i \psi_i^T$$

Diffusion Maps [Coifman and Lafon, '06]

From $M = \Phi\Lambda\Psi^T$,

$$M = \sum_{i=1}^{n} \lambda_i \phi_i \psi_i^T$$

Thus

$$M^t = \sum_{i=1}^{n} \lambda_i^t \phi_i \psi_i^T$$

Back to our suggestion before:

$$M_{k:}^t = \sum_{i=1}^{n} \lambda_i^t \phi_i(k) \psi_i^T$$

Diffusion Maps [Coifman and Lafon, ’06]

From $M = \Phi\Lambda\Psi^T$,

$$M = \sum_{i=1}^{n} \lambda_i \phi_i \psi_i^T$$

Thus

$$M^t = \sum_{i=1}^{n} \lambda_i^t \phi_i \psi_i^T$$

Back to our suggestion before:

$$M^t_{k:} = \sum_{i=1}^{n} \lambda_i^t \phi_i(k) \psi_i^T$$

So we can represent node v_i in terms of the basis Ψ, and put

$$v_i \mapsto \begin{bmatrix} \lambda_1^t \phi_1(i) \\ \vdots \\ \lambda_n^t \phi_n(i) \end{bmatrix}$$

Diffusion Maps [Coifman and Lafon, '06]

Note that $M\mathbb{1} = \mathbb{1}$, and $\phi_1 = \mathbb{1}$ with $\lambda_1 = 1$ by previous analysis (note that $M = D^{-1}W = I - L_{\mathrm{rw}}$)

Diffusion Maps [Coifman and Lafon, '06]

Note that $M\mathbb{1} = \mathbb{1}$, and $\phi_1 = \mathbb{1}$ with $\lambda_1 = 1$ by previous analysis (note that $M = D^{-1}W = I - L_{\text{rw}}$)

Thus ϕ_n doesn't tell us any information, so we define the diffusion map $\phi_t : V \to \mathbb{R}^{n-1}$ via

$$v_i \mapsto \begin{bmatrix} \lambda_2^t \phi_1(i) \\ \vdots \\ \lambda_n^t \phi_n(i) \end{bmatrix}$$

Diffusion Maps [Coifman and Lafon, '06]

Note that $M\mathbb{1} = \mathbb{1}$, and $\phi_1 = \mathbb{1}$ with $\lambda_1 = 1$ by previous analysis (note that $M = D^{-1}W = I - L_{\text{rw}}$)

Thus ϕ_n doesn't tell us any information, so we define the diffusion map $\phi_t : V \to \mathbb{R}^{n-1}$ via

$$v_i \mapsto \begin{bmatrix} \lambda_2^t \phi_1(i) \\ \vdots \\ \lambda_n^t \phi_n(i) \end{bmatrix}$$

Similar to other methods, the truncated diffusion map is $\phi_t^{(d)} : V \to \mathbb{R}^d$ via

$$\phi_t^{(d)}(v_i) = \begin{bmatrix} \lambda_2^t \phi_2(i) \\ \vdots \\ \lambda_{d+1}^t \phi_{d+1}(i) \end{bmatrix} = (\Lambda'_{d+1})^t (\Phi'_{d+1})_{i:}$$

Diffusion Maps [Coifman and Lafon, '06]

Useful Property: Diffusion maps give a measure of distance between probability clouds after time t for walkers starting at different nodes:

Diffusion Maps [Coifman and Lafon, '06]

Useful Property: Diffusion maps give a measure of distance between probability clouds after time t for walkers starting at different nodes:

Theorem

For any v_i, v_j

$$\|\phi_t(v_i) - \phi_t(v_j)\|_2^2 = \sum_{k=1}^{n} \frac{1}{d_k} \left(\mathbb{P}[X(t) = k \mid X(0) = i] - \mathbb{P}[X(t) = k \mid X(0) = j]\right)^2$$

Diffusion Maps [Coifman and Lafon, '06]

Algorithm
Step 1: Form graph (ε–neighborhood or k–NN)
Step 2: $M = \Phi\Lambda\Psi^T$
Step 3: Diffusion map: $\phi_t : V \to \mathbb{R}^d$ as above

Parameters: ε/k, t

Comparison

Question: So how are Diffusion Maps and Laplacian Eigenmaps different?

Comparison

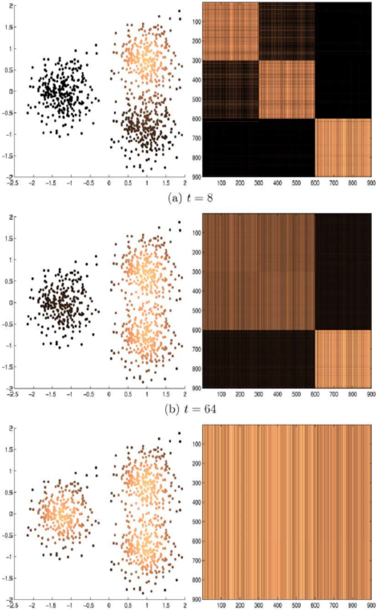
Question: So how are Diffusion Maps and Laplacian Eigenmaps different?

- DM uses L_{rw} and its eigenvectors, wheras LE uses L and its eigenvectors.

Comparison

Question: So how are Diffusion Maps and Laplacian Eigenmaps different?

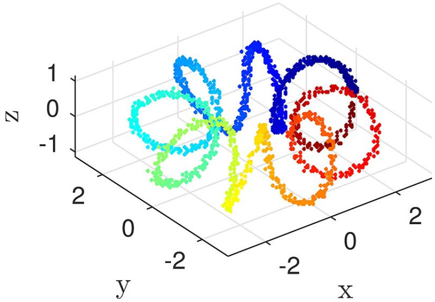
- DM uses L_{rw} and its eigenvectors, wheras LE uses L and its eigenvectors.
- DM uses scaling by powers of λ_i which represents a random walk diffusing over the graph (note: $|\lambda_i| \leq 1$ for all eigenvalues of M, so diffusion maps don't blow up)

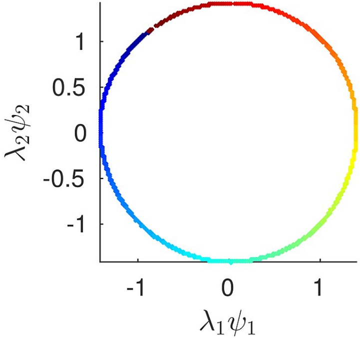


(a) $t = 8$

(b) $t = 64$

(c) $t = 1024$


z
1
0
-1
y
2
0
-2
x
-2
0
2


$\lambda_2\psi_2$
1
0.5
0
-0.5
-1
$\lambda_1\psi_1$
-1
0
1

Part III: Functional Manifold Learning

Image Manifold Learning Pipeline

$$\mathscr{F} \xrightarrow{\mathcal{H}} \mathbb{R}^D \xrightarrow{\phi} \mathbb{R}^d \xrightarrow{\mathcal{D}} \Lambda$$

Image Manifold Learning Pipeline

$$\mathscr{F} \xrightarrow{\mathcal{H}} \mathbb{R}^D \xrightarrow{\phi} \mathbb{R}^d \xrightarrow{\mathcal{D}} \Lambda$$

- $\mathscr{F}$ = image space
- Λ = decision/label space
- $\mathcal{H} : \mathscr{F} \to \mathbb{R}^D$ = imaging/discretization operator
- $\phi : \mathbb{R}^D \to \mathbb{R}^d$ = dimensionality reduction operator
- $\mathcal{D} : \mathbb{R}^d \to \Lambda$ = decision operator

Often one thinks of the manifold hypothesis as images are in $\mathcal{M} \subset \mathbb{R}^D$.

Issues:

Often one thinks of the manifold hypothesis as images are in $\mathcal{M} \subset \mathbb{R}^D$.

Issues:

- Ignores $\mathscr{F}$
- Ignores discretization process / imaging operation, which can vary greatly
- Treats preprocessing as a black box

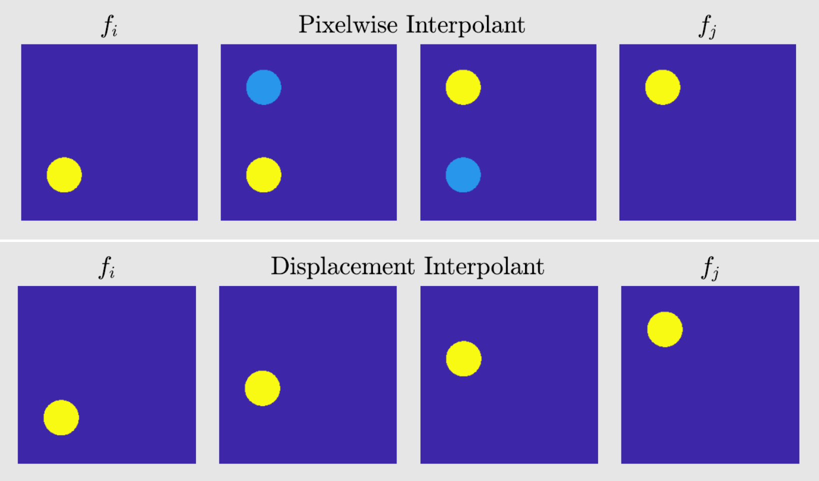
Often one thinks of the manifold hypothesis as images are in $\mathcal{M} \subset \mathbb{R}^D$.

Issues:

- Ignores $\mathscr{F}$
- Ignores discretization process / imaging operation, which can vary greatly
- Treats preprocessing as a black box

Now we will take a Functional Manifold Hypothesis: $\mathcal{M} \subset \mathscr{F}$

What's in a distance?



What's in a distance?

If we treat images as Euclidean, pixelwise (ℓ_2) distances can be meaningless

What function space should we consider as $\mathscr{F}$?

Case study: $\mathscr{F} = L_2(\mathbb{R}^m)$

[Donoho, Grimes '05]

$\mathcal{M} \subset L_2(\mathbb{R}^m)$

Question: Given two samples from $\mathscr{F}$, how can we estimate the geodesic distance between them?

Case study: $\mathscr{F} = L_2(\mathbb{R}^m)$

[Donoho, Grimes '05]

$\mathcal{M} \subset L_2(\mathbb{R}^m)$

Question: Given two samples from $\mathscr{F}$, how can we estimate the geodesic distance between them?

Case study: $\mathscr{F} = L_2(\mathbb{R}^m)$

[Donoho, Grimes '05]

$\mathcal{M} \subset L_2(\mathbb{R}^m)$

Question: Given two samples from $\mathscr{F}$, how can we estimate the geodesic distance between them?

Option 1: Use the *induced intrinsic metric* on $\mathscr{F}$ induced by the ambient L_2 norm

Case study: $\mathscr{F} = L_2(\mathbb{R}^m)$

[Donoho, Grimes '05]

$\mathcal{M} \subset L_2(\mathbb{R}^m)$

Question: Given two samples from $\mathscr{F}$, how can we estimate the geodesic distance between them?

Option 1: Use the *induced intrinsic metric* on $\mathscr{F}$ induced by the ambient L_2 norm

$\Gamma(f_i, f_j)$ = set of all continuous paths $\gamma : [0,1] \to L_2$ such that $\gamma(0) = f_i, \gamma(1) = f_j$

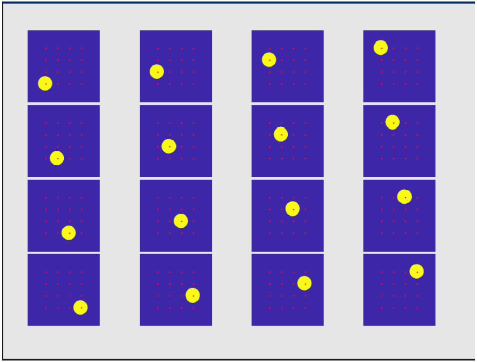
$$d_{\mathscr{F},L_2}(f_i, f_j) := \inf_{\gamma \in \Gamma(f_i,f_j)} L(\gamma) = \inf_{\gamma \in \Gamma(f_i,f_j)} \sup_{t_0,\ldots,t_m} \sum_{k=1}^{m} \|\gamma(t_{k-1}) - \gamma(t_k)\|_{L_2}$$

Case study: $\mathscr{F} = L_2(\mathbb{R}^m)$

- Geodesics based on $\|\cdot\|_{L_2}$ blow up unexpectedly (translates of an indicator of a ball)
- One workaround is to mollify functions with Gaussians of decreasing width and normalize by a reference trajectory

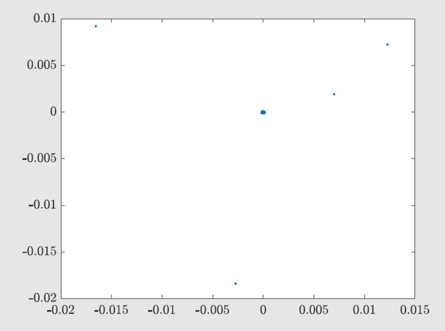
ISOMAP on L_2

Consider $\mathscr{F}^{\text{transl}} := \{f_0(\cdot - t), t \in \alpha\mathbb{Z}^2\} \subset L_2(\mathbb{R}^2)$ with $f_0 = \mathbb{1}_D$



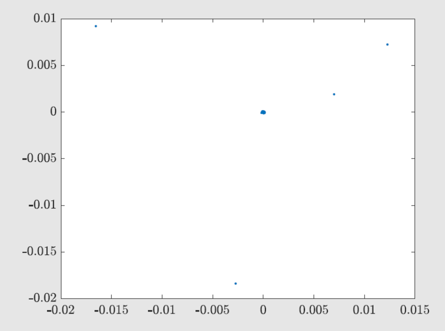
ISOMAP on L_2

Consider $\mathscr{F}^{\text{transl}} := \{f_0(\cdot - t), t \in \alpha\mathbb{Z}^2\} \subset L_2(\mathbb{R}^2)$ with $f_0 = \mathbb{1}_D$



ISOMAP on L_2

Consider $\mathscr{F}^{\text{transl}} := \{f_0(\cdot - t), t \in \alpha\mathbb{Z}^2\} \subset L_2(\mathbb{R}^2)$ with $f_0 = \mathbb{1}_D$



Problem: pairwise distances are essentially constant

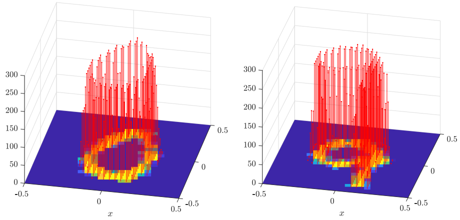
ISOMAP on L_2

Theorem (Donoho, Grimes '05)

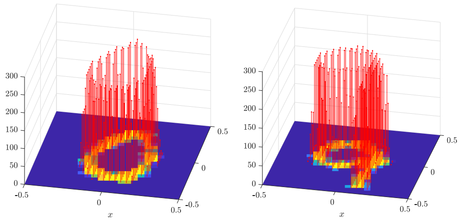
$(\mathscr{F}^{\text{transl}}, d_{\mathscr{F},L_2})$ *is isometric to* $\Omega \subset \mathbb{R}^d$ *if and only if* f_0 *is differentiable.*

Option 2: View images as non-negative L_1 functions with compact support.

Option 2: View images as non-negative L_1 functions with compact support.



Option 2: View images as non-negative L_1 functions with compact support.



These can be naturally embedded into the space of probability measures as follows

Map $(\mathscr{F}, d_{\mathscr{F}}) \subset (L_2, \|\cdot\|_{L_2})$ into $(\widetilde{\mathscr{F}}, W_2) \subset (\mathcal{P}(\mathbb{R}^2), W_2)$

$$f \mapsto \frac{f}{\|f\|_{L_1}}$$

Wasserstein Metric (A Fly-by Overview)

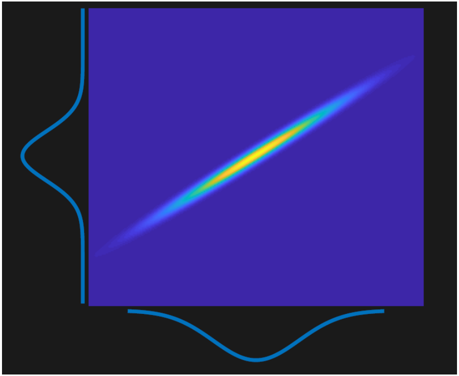
Main Idea: Optimal Transport. What is the optimal transport plan to map one probability distribution to another? (Monge, 1781)

Wasserstein Metric (A Fly-by Overview)

Main Idea: Optimal Transport. What is the optimal transport plan to map one probability distribution to another? (Monge, 1781)

Given $\mu, \nu \in \mathcal{P}(\mathbb{R}^2)$, denote the space of couplings

$$\Pi(\mu,\nu) := \{\pi \in \mathcal{P}(\mathbb{R}^4) : \pi(A \times \mathbb{R}^2) = \mu(A), \pi(\mathbb{R}^2 \times A) = \nu(A), A \in \mathbb{R}^2\}$$



Wasserstein Metric (A Fly-by Overview)

The 2–Wasserstein metric is defined by

$$W_2^2(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^{2d}} |x-y|^2 d\pi$$

Wasserstein Metric (A Fly-by Overview)

The 2–Wasserstein metric is defined by

$$W_2^2(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^{2d}} |x-y|^2 d\pi$$

(e.g., Villani's book) 1) $(\mathcal{P}(\mathbb{R}^d), W_2)$ is a length space, 2) the optimal coupling π^* is equivalent to finding a transport map (change of variables) such that

$$f_j(T(x))|J_T(x)| = f_i(x)$$

Wasserstein Metric (A Fly-by Overview)

The 2–Wasserstein metric is defined by

$$W_2^2(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^{2d}} |x-y|^2 d\pi$$

(e.g., Villani's book) 1) $(\mathcal{P}(\mathbb{R}^d), W_2)$ is a length space, 2) the optimal coupling π^* is equivalent to finding a transport map (change of variables) such that

$$f_j(T(x))|J_T(x)| = f_i(x)$$

Induces the *displacement interpolant*

$$T_t(x) := (1-t)x + tT(x)$$

The 2–Wasserstein metric is defined by

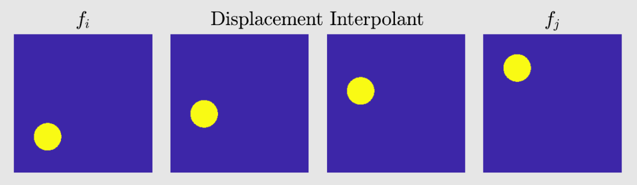
$$W_2^2(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^{2d}} |x-y|^2 d\pi$$

(e.g., Villani's book) 1) $(\mathcal{P}(\mathbb{R}^d), W_2)$ is a length space, 2) the optimal coupling π^* is equivalent to finding a transport map (change of variables) such that

$$f_j(T(x))|J_T(x)| = f_i(x)$$

Induces the *displacement interpolant*

$$T_t(x) := (1-t)x + tT(x)$$



Wasserstein Metric (A Fly-by Overview)

The 2–Wasserstein metric is defined by

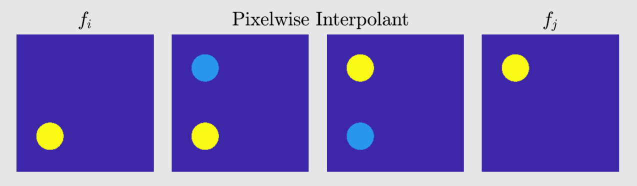
$$W_2^2(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^{2d}} |x-y|^2 d\pi$$

(e.g., Villani's book) 1) $(\mathcal{P}(\mathbb{R}^d), W_2)$ is a length space, 2) the optimal coupling π^* is equivalent to finding a transport map (change of variables) such that

$$f_j(T(x))|J_T(x)| = f_i(x)$$

Induces the *displacement interpolant*

$$T_t(x) := (1-t)x + tT(x)$$



Functional Wassmap[1]

[1] [H–Henscheid–Kang, '22]

Functional Wassmap[1]

Given $\{\mu_i\}_{i=1}^N \subset W_2(\mathbb{R}^m)$

- Compute $D = (W_2(\mu_i, \mu_j)^2)_{i,j=1}^N$
- APSP of neighborhood graph
- MDS

[1] [H–Henscheid–Kang, '22]

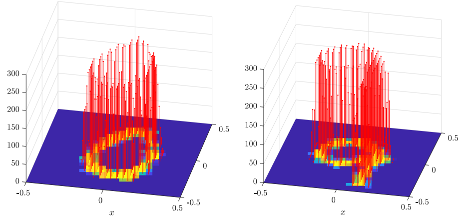
Discrete Wassmap

Given $\{x_i\}_{i=1}^{N} \subset \mathbb{R}^D$

Discrete Wassmap

Given $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$

- Measure Formation



- Functional Wassmap

Case studies

Manifolds generated by transformations of a fixed measure

$\Theta \subset \mathbb{R}^d$ some parameter set generating maps $T_\theta : \mathbb{R}^m \to \mathbb{R}^m$

$$\mathcal{M}(\mu_0, \Theta) := \{ T_{\theta\#}\mu_0 : \theta \in \Theta \}$$

Manifolds generated by transformations of a fixed measure

$\Theta \subset \mathbb{R}^d$ some parameter set generating maps $T_\theta : \mathbb{R}^m \to \mathbb{R}^m$

$$\mathcal{M}(\mu_0, \Theta) := \{ T_{\theta\#}\mu_0 : \theta \in \Theta \}$$

$$T_{\#}\mu(A) = \mu(T^{-1}(A))$$

Case studies

Manifolds generated by transformations of a fixed measure

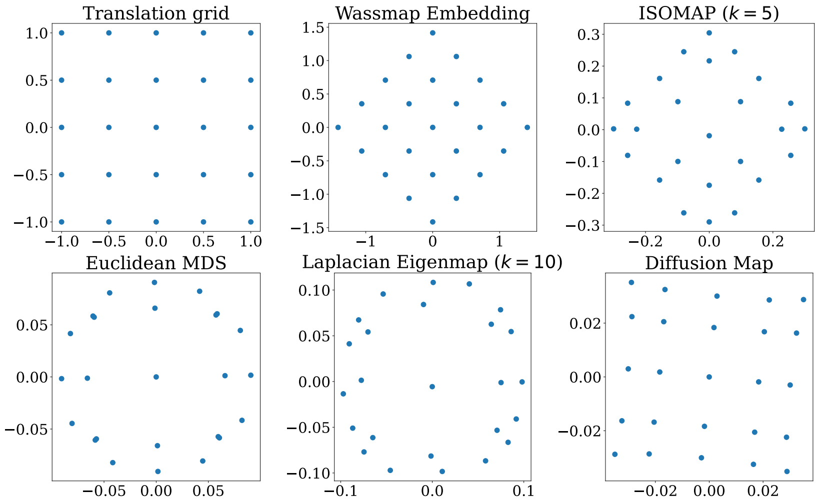
$\Theta \subset \mathbb{R}^d$ some parameter set generating maps $T_\theta : \mathbb{R}^m \to \mathbb{R}^m$

$$\mathcal{M}(\mu_0, \Theta) := \{T_{\theta\#}\mu_0 : \theta \in \Theta\}$$

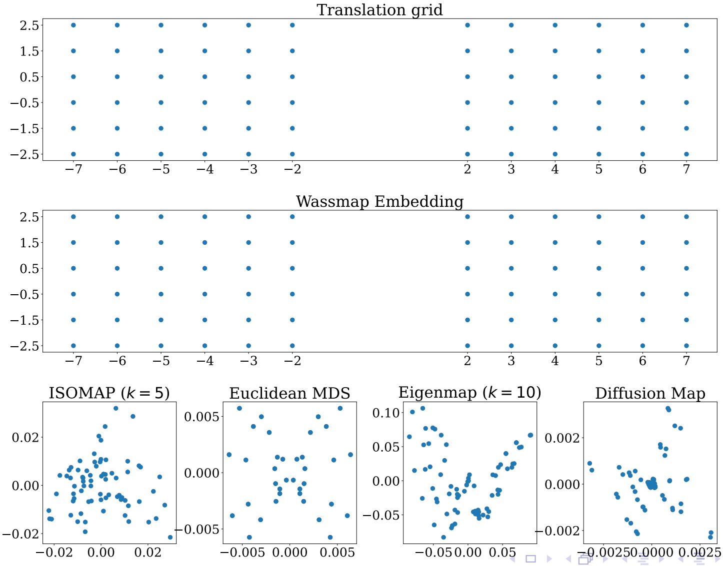
$$T_\# \mu(A) = \mu(T^{-1}(A))$$

- Translation: $\{\mu_0(\cdot - \theta)\}$
- Dilation: $\{\det(D_\theta)\mu_0(D_\theta \cdot)\}$ $\qquad D_\theta = \mathrm{diag}(\frac{1}{\vartheta_1}, \dots, \frac{1}{\vartheta_m})$
- Rotation: $\{\mu_0(R_\theta \cdot) : R_\theta \in SO(m)\}$

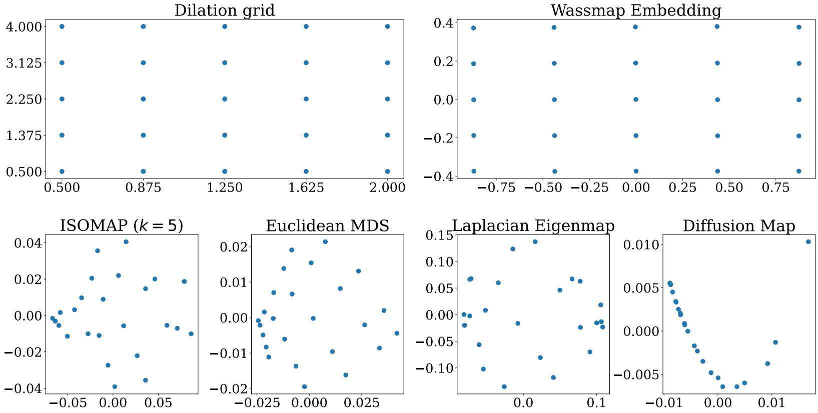
Translation manifold – $\mu_0 = \frac{1}{\pi}\mathbb{1}_D(x)dx$



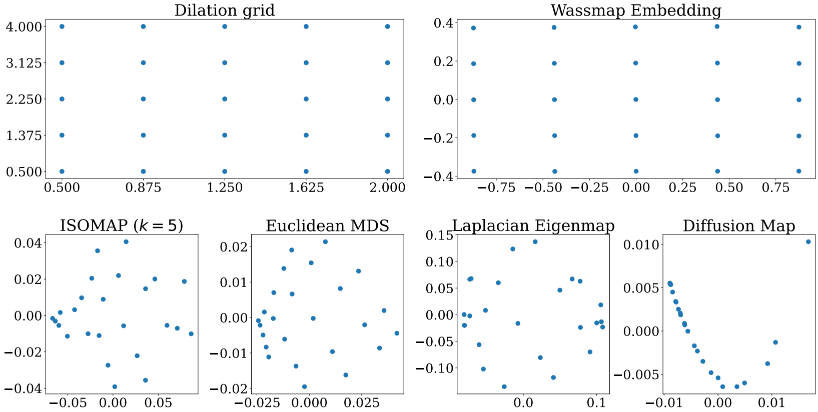
Translation manifold – $\mu_0 = \frac{1}{\pi}\mathbb{1}_D(x)dx$



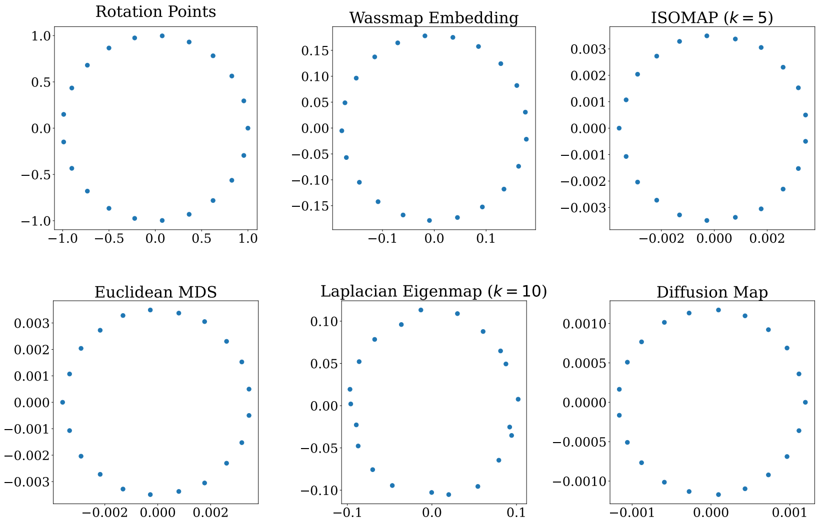
Dilation manifold – $\mu_0 = \frac{1}{\pi}\mathbb{1}_D(x)dx$



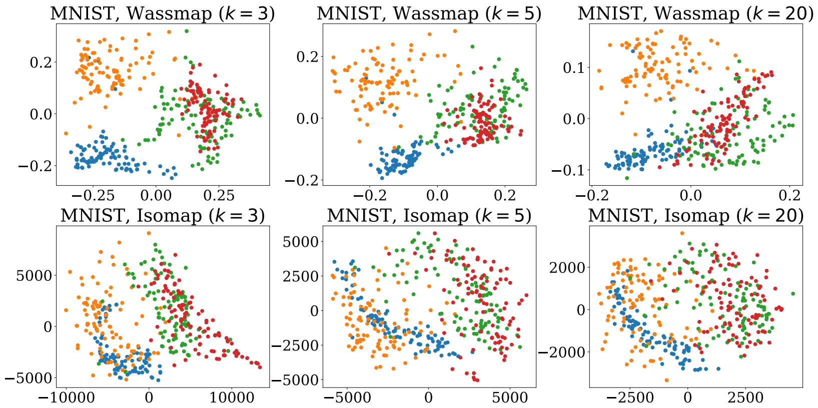
Dilation manifold – $\mu_0 = \frac{1}{\pi}\mathbb{1}_D(x)dx$



Rotation manifold – μ_0 indicator of origin centered ellipse



MNIST



blue: 1, orange: 2, green: 7, red: 9

Theory

Given $\{\theta_i\}_{i=1}^N \subset \mathbb{R}^d$ and observations $\{\mu_{\theta_i}\}_{i=1}^N \subset W_2(\mathbb{R}^m)$.

Theory

Given $\{\theta_i\}_{i=1}^N \subset \mathbb{R}^d$ and observations $\{\mu_{\theta_i}\}_{i=1}^N \subset W_2(\mathbb{R}^m)$.

Theorem[H-Henscheid-Kang,'22] For arbitrary $\mu_0 \in W_2$ (resp. discrete μ_0) Functional (resp. Discrete) Wassmap recovers up to rigid transformation

Theory

Given $\{\theta_i\}_{i=1}^N \subset \mathbb{R}^d$ and observations $\{\mu_{\theta_i}\}_{i=1}^N \subset W_2(\mathbb{R}^m)$.

Theorem[H-Henscheid-Kang,'22] For arbitrary $\mu_0 \in W_2$ (resp. discrete μ_0) Functional (resp. Discrete) Wassmap recovers up to rigid transformation

Translations $\quad \{\theta_i\}$

Theory

Given $\{\theta_i\}_{i=1}^N \subset \mathbb{R}^d$ and observations $\{\mu_{\theta_i}\}_{i=1}^N \subset W_2(\mathbb{R}^m)$.

Theorem[H-Henscheid-Kang,'22] For arbitrary $\mu_0 \in W_2$ (resp. discrete μ_0) Functional (resp. Discrete) Wassmap recovers up to rigid transformation

Translations	$\{\theta_i\}$
Dilations	$\{S\theta_i\}$

Theory

Given $\{\theta_i\}_{i=1}^N \subset \mathbb{R}^d$ and observations $\{\mu_{\theta_i}\}_{i=1}^N \subset W_2(\mathbb{R}^m)$.

Theorem[H-Henscheid-Kang,'22] For arbitrary $\mu_0 \in W_2$ (resp. discrete μ_0) Functional (resp. Discrete) Wassmap recovers up to rigid transformation

$$\begin{array}{cc} \text{Translations} & \{\theta_i\} \\ \text{Dilations} & \{S\theta_i\} \end{array}$$

$$S = \operatorname{diag}\left(M_2^{\frac{1}{2}}(P_1\mu_0), \cdots, M_2^{\frac{1}{2}}(P_m\mu_0)\right)$$

$$M_2(P_i\mu_0) := \int_{\mathbb{R}^m} |x_i|^2 d\mu_0(x)$$

Theory

Given $\{\theta_i\}_{i=1}^N \subset \mathbb{R}^d$ and observations $\{\mu_{\theta_i}\}_{i=1}^N \subset W_2(\mathbb{R}^m)$.

Theorem[H-Henscheid-Kang,'22] For arbitrary $\mu_0 \in W_2$ (resp. discrete μ_0) Functional (resp. Discrete) Wassmap recovers up to rigid transformation

$$\begin{array}{cc} \text{Translations} & \{\theta_i\} \\ \text{Dilations} & \{S\theta_i\} \end{array}$$

$$S = \operatorname{diag}\left(M_2^{\frac{1}{2}}(P_1\mu_0), \cdots, M_2^{\frac{1}{2}}(P_m\mu_0)\right)$$

$$M_2(P_i\mu_0) := \int_{\mathbb{R}^m} |x_i|^2 d\mu_0(x)$$

Remark: No proof currently for rotations (Brenier's Theorem)

Key Ingredients

$$W_2(\mu_0(\cdot - t), \mu_0(\cdot - s)) = |t - s|$$

Key Ingredients

$$W_2(\mu_0(\cdot - t), \mu_0(\cdot - s)) = |t - s|$$

$$W_2(\det(D_\theta)\mu(D_\theta \cdot), \det(D_{\theta'})\mu_0(D_{\theta'} \cdot))^2 = \sum_{i=1}^{m} |\vartheta_i - \vartheta_i'|^2 \int_{\mathbb{R}^m} |x_i|^2 d\mu_0$$

$$= |S\theta - S\theta'|^2$$

Key Ingredients

$$W_2(\mu_0(\cdot - t), \mu_0(\cdot - s)) = |t - s|$$

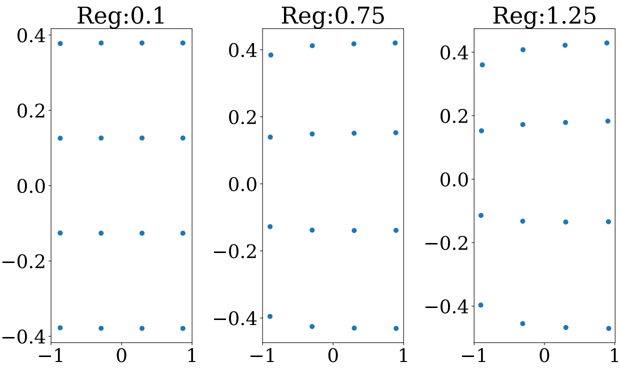
$$W_2(\det(D_\theta)\mu(D_\theta \cdot), \det(D_{\theta'})\mu_0(D_{\theta'} \cdot))^2 = \sum_{i=1}^{m} |\vartheta_i - \vartheta_i'|^2 \int_{\mathbb{R}^m} |x_i|^2 d\mu_0$$

$$= |S\theta - S\theta'|^2$$

Theorem: If $W_2(\mu_\theta, \mu_{\theta'}) = f(\theta, \theta')$ for absolutely continuous μ_0, and T_θ are uniformly Lipschitz, then the same holds for arbitrary μ_0.

On Computation

Using fast W_2 approximations



On Computation

Naïvely requires $O(N^2)$ Wasserstein computations

On Computation

Naïvely requires $O(N^2)$ Wasserstein computations

- In many cases can use LOT to reduce to $O(N)$ computations [Moosmüller, Cloninger '20], [Khurana et al. '22]

On Computation

Naïvely requires $O(N^2)$ Wasserstein computations

- In many cases can use LOT to reduce to $O(N)$ computations [Moosmüller, Cloninger '20], [Khurana et al. '22]
- Sliced Wasserstein distance

On Computation

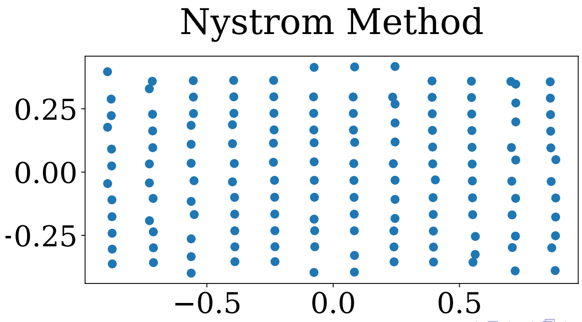
Naïvely requires $O(N^2)$ Wasserstein computations

- In many cases can use LOT to reduce to $O(N)$ computations [Moosmüller, Cloninger '20], [Khurana et al. '22]
- Sliced Wasserstein distance
- Can use Nyström method to reduce to $O(N \log N)$ computations

On Computation

Naïvely requires $O(N^2)$ Wasserstein computations

- In many cases can use LOT to reduce to $O(N)$ computations [Moosmüller, Cloninger '20], [Khurana et al. '22]
- Sliced Wasserstein distance
- Can use Nyström method to reduce to $O(N \log N)$ computations



Theorem (Cloninger–H–Khurana–Moosmüller, '22+)

Let $\{\mu_i\}_{i=1}^N \subset W_2(\mathbb{R}^n)$. Suppose $\mathcal{W} \subset W_2(\mathbb{R}^n)$ is a subset of Wasserstein space that is isometric to a subset of Euclidean space $\Omega \subset \mathbb{R}^d$, and $\{\nu_i\}_{i=1}^N \subset \mathcal{W}$ and $\{y_i\} \subset \Omega$ are such that $|y_i - y_j| = W_2(\nu_i, \nu_j)$. Let $\Delta_{ij} := W_2(\nu_i, \nu_j)^2$, $\Gamma_{ij} := W_2(\mu_i, \mu_j)^2$, and $\Lambda_{ij} := \lambda_{ij}^2$ for some $\lambda_{ij} \in \mathbb{R}$. Let z_i be the output of MDS on Λ.

If $|W_2(\mu_i, \mu_j)^2 - W_2(\nu_i, \nu_j)^2| \leq \tau_1$ and $|W_2(\mu_i, \mu_j)^2 - \lambda_{ij}^2| \leq \tau_2$ for some τ_1 and τ_2, and if

$$\|Y^\dagger\| \sqrt{N} (\tau_1 + \tau_2)^{\frac{1}{2}} \leq \frac{1}{\sqrt{2}}, \tag{1}$$

then $\{z_i\}_{i=1}^N \subset \mathbb{R}^d$ satisfies

$$\min_{Q \in \mathcal{O}(d)} \|Z - YQ\|_F \leq (1 + \sqrt{2}) \|Y^\dagger\| N (\tau_1 + \tau_2).$$

- τ_1 – how far the data is away from a Euclidean manifold in W_2
- τ_2 – how well the W_2 distances are estimated (can be done via entropic regularization or linear optimal transport, e.g., Akram Aldroubi's talk)

Thanks!