A tutorial on Manifold Learning for real data

The Fields Institute Workshop on Manifold and Graph-based learning

Lectures 2-3 Notes

Marina Meilă

Department of Statistics
University of Washington

19-20 May, 2022

Outline

1. What is manifold learning good for?
2. Manifolds, Coordinate Charts and Smooth Embeddings
3. Non-linear dimension reduction algorithms
   - Local PCA
   - PCA, Kernel PCA, MDS recap
   - Principal Curves and Surfaces (PCS)
   - Embedding algorithms
   - Heuristic algorithms
4. Metric preserving manifold learning – Riemannian manifolds basics
   - Embedding algorithms introduce distortions
   - Metric Manifold Learning – Intuition
   - Estimating the Riemannian metric
5. Neighborhood radius and other choices
   - What graph? Radius-neighbors vs. k nearest-neighbors
   - What neighborhood radius/kernel bandwidth?

Non-linear dimension reduction: Three principles

Algorithm given $\mathcal{D} = \{\xi_1, \ldots \xi_n\}$ from $\mathcal{M} \subset \mathbb{R}^D$, map them by Algorithm f to $\{y_1, \ldots y_n\} \subset \mathbb{R}^m$
Assumption if points from $\mathcal{M}$, $n \to \infty$, f is embedding of $\mathcal{M}$
(f "recovers" $\mathcal{M}$ of arbitrary shape).

1. Local (weighted) PCA (lPCA)
2. Principal Curves and Surfaces (PCS)
3. Embedding algorithms (Diffusion Maps/Laplacian Eigenmaps, Isomap, LTSA, MVU, Hessian Eigenmaps,. . .)

4. [Other, heuristic] t-SNE, UMAP, LLE

What makes the problem hard?

- Intrinsic dimension d
  - must be estimated (we assume we know it) (Lecture 3)
  - sample complexity is exponential in d – **NONPARAMETRIC** (upcoming)
- non-uniform sampling
- **volume** of $\mathcal{M}$ (we assume volume finite; larger volume requires more samples)
- **injectivity radius/reach** of $\mathcal{M}$ (next page)
- curvature

- ESSENTIAL smoothness parameter: the **neighborhood radius** (Lecture 3)

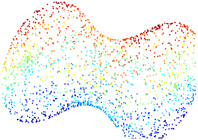
Neighborhood graphs

- All ML algorithms start with a **neighborhood graph** over the data points
    - neigh_i denotes the neighbors of ξ_i, and $k_i = |\text{neigh}_i|$.
    - $\Xi_i = [\xi_{i'}]_{i' \in \text{neigh}_i} \in \mathbb{R}^{D \times k_i}$ contains the coordinates of ξ_i's neighbors
- In the **radius-neighbor** graph, the neighbors of ξ_i are the points within distance r from ξ_i, i.e. in the ball $B_r(\xi_i)$.
- In the **k-nearest-neighbor (k-nn)** graph, they are the k nearest-neighbors of ξ_i.

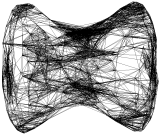
- k-nn graph has many computational advantages
    - constant degree k (or $k-1$)
    - connected for any $k > 1$
    - more software available

isometry

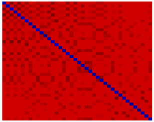
- but much more difficult to use for consistent estimation of manifolds (see later, and)



data $\xi_1, \ldots \xi_n \subset \mathbb{R}^D$



neighborhood graph



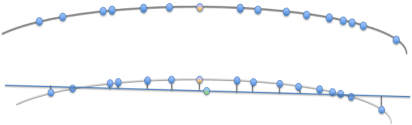
A (sparse) matrix of distances between neighbors

Local Principal Components Analysis (LPCA)

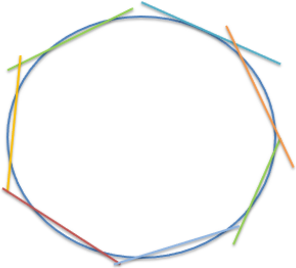
Idea Approximate $\mathcal{M}$ with tangent subspaces at a finite number of data points

1. Pick a point $\xi_i \in \mathcal{D}$
2. Find neigh_i, perform PCA on $\mathrm{neigh}_i \cup \{\xi_i\}$ and obtain (affine) subspace with basis $T_i \in \mathbb{R}^{D\times d}$
3. Represent $\xi_{i'} \in \mathrm{neigh}_i$ by $y_i = \mathrm{Proj}_{T_i}\, \xi_{i'}$

$$y_{i'} = T_i^T(\xi_{i'} - \xi_i) \quad \text{new coordinates of } \xi_{i'} \text{ in } \mathcal{T}_{\xi_i}\mathcal{M} \tag{1}$$



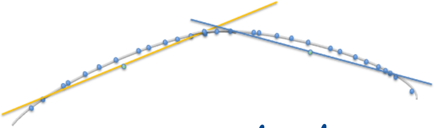
Repeat for a sample of $n' < n$ data points

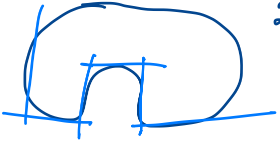


Local PCA

- For n, n' sufficiently large, $\mathcal{M}$ can be approximated with arbitrary accuracy

  So, are we done?
  Some issues with LPCA
- Point ξ_j may be represented in multiple T_i's (minor)
- New coordinates y_j are relative to local T_i
- Fine for local operations like regression
- Number of charts depends on extrinsic properties
- Cumbersome for larger scale operations like following a curve on $\mathcal{M}$
- Biased in noise





2 charts sufficient

>> 2 charts needed
for LPCA

Multi-dimensional scaling (MDS)

- (See notes for PCA, Kernel PCA, centering matrix H, MDS for details)
- Problem Given matrix of (squared) distances $D \in \mathbb{R}^{n \times n}$, find a set of n points in d dimensions $Y = d \times n$ so that

$$D_Y = [\|y_i - y_j\|^2]_{i,j} \approx D$$

- Useful when
  - original points are not vectors but we can compute distances (e.g string edit distances, philogenetic distances)
  - original points are in high dimensions
  - original distances are geodesic distances on a manifold $\mathcal{M}$

MDS Algorithm

1. Calculate $K = -\frac{1}{2} HDH^T$
2. Compute its d principal e-vectors/values: $K = V\Sigma^2 V^T$
3. $Y = \Sigma V^T$ are new coordinates

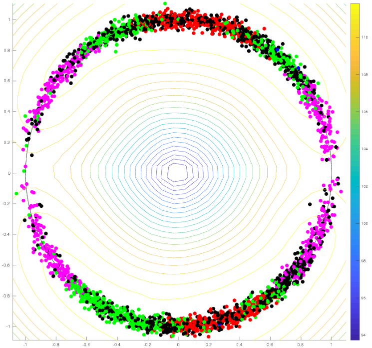
The **Centering Matrix** H

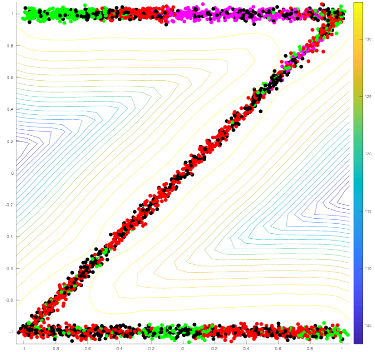
$$H = I - \frac{1}{n} 1_{n \times n}$$

Q: Could MDS be an embedding algorithm? What is different about MDS and upcoming algorithms?

Principal Curves and Surfaces (PCS)

??

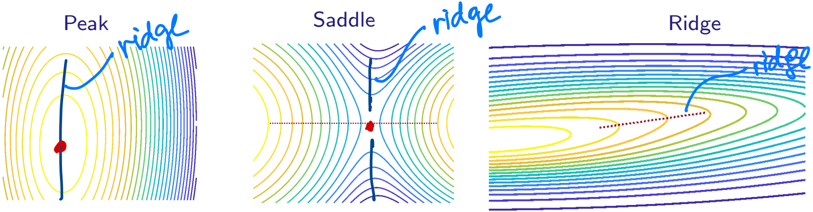




- Elegant algorithm , most useful for $d = 1$ (curves)
- Also works in noise **??**

- data in $\mathbb{R}^D$ near a curve (or set of curves)
- **Goal:** track the **ridge** of the data density (will be biased estimator of curve $\mathcal{M}$)

What is a density ridge



Peak	Saddle	Ridge
$\nabla p = 0$ $\nabla^2 p \prec 0$	$\nabla p = 0$ $\nabla^2 p$ has $\lambda_1 > 0$, $\lambda_{2:D} < 0$	$\nabla p = 0$ in span$\{v_{2:D}\}$ $\nabla^2 p$ has $\lambda_{2:D} < 0$, ($v_{1:D}$ e-vectors $\nabla^2 p$)

In other words, on a **ridge**

- $\nabla p \propto v_1$ direction of **least negative curvature (LNC)** of $\nabla^2 p$
- $\nabla p, v_1$ are tangent to the ridge

$p(\,)$ sampling density on $\mathbb{R}^D$
↳ estimated by KDE

Gradient and Hessian for Gaussian KDE

- Data $\xi_{1:n} \in \mathbb{R}^D$
- Let $p()$ be the kernel density estimator with some kernel width h.

$$p(\xi) \;=\; \frac{1}{nh^d}\sum_{i=1}^{n}\kappa(\frac{\xi-\xi_i}{h}) \;=\; \frac{1}{nh^d}\sum_{i=1}^{n}\exp\left(-\frac{(\xi-\xi_i)^2}{2h^2}\right)/\omega_d \tag{2}$$

- We prefer to work with $\ln p$ which has the same critical points/ridges as p
- $\nabla \ln p = \frac{1}{p}\nabla p = g$
- $\nabla^2 \ln p = -\frac{1}{p^2}\nabla p \nabla p^T + \frac{1}{p}\nabla^2 p = H$

Gradient and Hessian for Gaussian KDE

- Data $\xi_{1:n} \in \mathbb{R}^D$
- Let $p()$ be the kernel density estimator with some kernel width h.

$$p(\xi) = \frac{1}{nh^d}\sum_{i=1}^{n}\kappa(\frac{\xi-\xi_i}{h}) = \frac{1}{nh^d}\sum_{i=1}^{n}\exp\left(-\frac{(\xi-\xi_i)^2}{2h^2}\right)/\omega_d \qquad (2)$$

- We prefer to work with $\ln p$ which has the same critical points/ridges as p
- $\nabla \ln p = \frac{1}{p}\nabla p = g$
- $\nabla^2 \ln p = -\frac{1}{p^2}\nabla p \nabla p^T + \frac{1}{p}\nabla^2 p = H$

$$g(\xi) = -\frac{1}{h^2}[\xi - \sum_{i=1}^{n}\xi_i \underbrace{\exp\left(-\frac{(\xi-\xi_i)^2}{2h^2}\right)/\sum_{i=1}^{n}\exp\left(-\frac{(\xi-\xi_i)^2}{2h^2}\right)}_{w_i}] = -\frac{1}{h^2}[\underbrace{\xi - m(\xi)}_{\textbf{Mean-shift}}] \quad (3)$$

$\sum w_i = 1$

Gradient and Hessian for Gaussian KDE

- Data $\xi_{1:n} \in \mathbb{R}^D$
- Let $p()$ be the kernel density estimator with some kernel width h.

$$p(\xi) \;=\; \frac{1}{nh^d}\sum_{i=1}^{n}\kappa\left(\frac{\xi-\xi_i}{h}\right) \;=\; \frac{1}{nh^d}\sum_{i=1}^{n}\exp\left(-\frac{(\xi-\xi_i)^2}{2h^2}\right)/\omega_d \tag{2}$$

- We prefer to work with $\ln p$ which has the same critical points/ridges as p
- $\nabla \ln p = \frac{1}{p}\nabla p = g$
- $\nabla^2 \ln p = -\frac{1}{p^2}\nabla p \nabla p^T + \frac{1}{p}\nabla^2 p = H$

$$g(\xi) \;=\; -\frac{1}{h^2}[\xi - \sum_{i=1}^{n}\xi_i \underbrace{\exp\left(-\frac{(\xi-\xi_i)^2}{2h^2}\right)/\sum_{i=1}^{n}\exp\left(-\frac{(\xi-\xi_i)^2}{2h^2}\right)}_{w_i}] \;=\; -\frac{1}{h^2}[\underbrace{\xi - m(\xi)}_{\textbf{Mean}-\textbf{shift}}] \tag{3}$$

- $H(\xi) \;=\; \sum_{i=1}^{n} w_i u_i u_i^T - g(\xi)g(\xi)^T - \frac{1}{h^2}I$

$$u_i = \frac{\xi_i - \xi}{h^2}$$

SCMS Algorithm

SCMS = Subspace Constrained Mean Shift

Init any ξ^1 — Density estimated by p =data $\star$ Gaussian kernel of width h

for $k = 1, 2, \ldots$

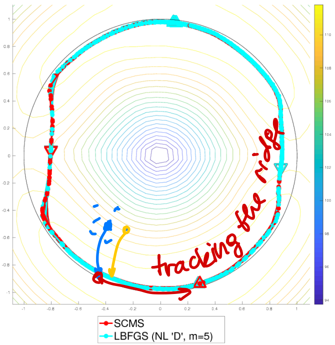
1. calculate $g^k \propto \nabla \ln p(\xi^k)$ — by Mean-Shift $\mathcal{O}(nD)$
2. $H^k = \nabla^2 \ln p(\xi^k)$ — $\mathcal{O}(nD^2)$
3. compute v_1 principal e-vector of H^k — $\mathcal{O}(D^2)$
4. $\xi^{k+1} \leftarrow \xi^k + \mathrm{Proj}_{v_1^\perp}\, g^k$ — $\mathcal{O}(D)$

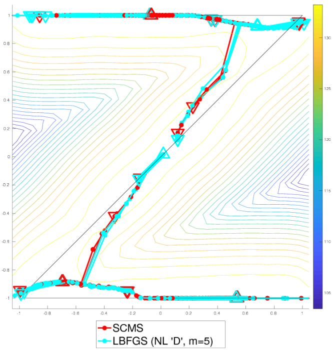
until convergence

- Algorithm SCMS finds 1 point on ridge; n restarts to cover all density

- Run time $\propto nD^2$/iteration
- Storage $\propto D^2$

Principal curves found by SCMS





LBFGS=accelerated, approximate SCMS – coming next!

Accelerating SCMS

- reduce dependency on n per iteration
  - ignore points far away from ξ
  - use approximate nearest neighbors (clustering, KD-trees,...)
- reduce number of SCMS runs: start only from $n' < n$ points
- reduce number iterations: **track ridge** instead of cold restarts
  - project ∇p on v_1 instead of $v_1^\perp$
  - tracking ends at critical point (peak or saddle)
- **reduce dependence on D**
  - approximate v_1 without computing whole H
  - $D^2 \leftarrow mD$ with $m \approx 5$

Non-linear dimension reduction algorithms summary

Paradigm	Input	Output	$f(\text{new } \xi)$	$f^{-1}(\text{new } p)$
local PCA	$\xi_{1:n} \in \mathbb{R}^D$	$y_{1:n} \in \mathbb{R}^d$ local maps (many)	✓	?
Principal Curves SCMS	$\xi_{1:n} \in \mathbb{R}^D$	$\xi'_{1:n} \in \mathbb{R}^D$ global map	✓ (if data kept)	N/A
Embedding Algorithm	$\xi_{1:n} \in \mathbb{R}^D$	$y_{1:n} \in \mathbb{R}^m$ global map or $\in \mathbb{R}^d$ local maps	ad-hoc or interpolation	ad-hoc or interpolation

e.g. kernel regression

—1—

Embedding algorithms

Diffusion Maps/Laplacian Eigenmaps, Isomap, LTSA, MVU, Hessian Eigenmaps,. . .

- Map $\mathcal{D}$ to $\mathbb{R}^m$ where $m \geq d$ (global coordinates)
- Can also map a local neighborhood $U \subseteq \mathcal{D}$ to $\mathbb{R}^d$ (local, intrinsic coordinates)

Input

- embedding dimension m
- neighborhood radius/kernel width ϵ
  - usually radius $r \approx 3 \times \epsilon$
- neighborhood graph
  $\{\text{neigh}_i, \ \Xi_i, \ \text{for } i = 1 : n\}$
  $A = [\|\xi_i - \xi_j\|]_{i,j=1}^n$ distance matrix, with $A_{ij} = \infty$ if $i \notin \text{neigh}_j$

The Isomap algorithm

Isomap Algorithm [Tennenbaum, deSilva & Langford 00]

Input A, dimension d

1. Find all shortest path distances in neighborhood graph $\approx$ geodesic distance
   if $A_{ij} = \infty$, then $A_{ij} \leftarrow$ graph distance between i, j
2. Construct matrix of squared distances

$$M = [(A_{ij})^2]$$

3. use Multi-Dimensional Scaling MDS(M, d) to obtain d dimensional coordinates Y for $\mathcal{D}$

- Works also for $m > d$

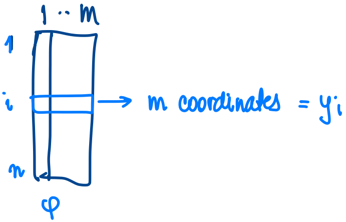
The Diffusion Maps (DM)/ Laplacian Eigenmaps (LE) Algorithm

Diffusion Maps Algorithm

Input distance matrix $A \in \mathbb{R}^{n \times n}$, bandwidth ϵ, embedding dimension m

1. Compute Laplacian $L \in \mathbb{R}^{n \times n}$
2. Compute eigenvectors of L for smallest $m+1$ eigenvalues $[\phi_0 \, \phi_1 \, \ldots \phi_m] \in \mathbb{R}^{n \times m}$
   - ϕ_0 is constant and not informative

The **embedding coordinates** of p_i are $(\phi_{i1}, \ldots \phi_{is})$



The (renormalized) Laplacian

Laplacian

Input distance matris $A \in \mathbb{R}^{n \times n}$, **bandwidth** ϵ

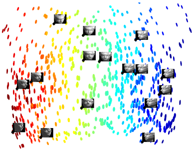
1. Compute **similarity matrix** $S_{ij} = \exp\left(-\frac{A_{ij}^2}{\epsilon^2}\right) = \kappa(A_{ij}/\epsilon)$
2. Normalize columns $d_j = \sum_{i=1}^n S_{ij}$, $\tilde{L}_{ij} = S_{ij}/d_j$
3. Normalize rows $d_i' = \sum_{j=1}^n \tilde{L}_{ij}$, $P_{ij} = \tilde{L}_{ij}/d_i'$
4. $L = \frac{1}{\epsilon^2}(I - P)$
5. Output L, d_i'/d_i

- Laplacian L central to understanding the manifold geometry
- $\lim_{n\to\infty} L = \Delta_{\mathcal{M}}$ [Coifman,Lafon 2006]
- Renormalization trick cancels effects of (non-uniform) sampling density [Coifman & Lafon 06]

Other Laplacians

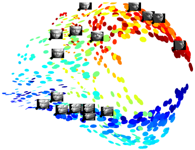
- $L^{un} = \mathrm{diag}\,\{d_{1:n}\} - \cancel{A}\,S$ — unnormalized Laplacian
- $L^{rw} = I - \mathrm{diag}\,\{d_{1:n}\}^{-1}\cancel{A}\,S$ — random walk Laplacian
- $L^{n} = I - \mathrm{diag}\,\{d_{1:n}\}^{-1/2}\cancel{A}\,S\,\mathrm{diag}\,\{d_{1:n}\}^{-1/2}$ — normalized Laplacian

Isomap vs. Diffusion Maps



Isomap

- Preserves geodesic distances
  - but only when $\mathcal{M}$ is flat and "data" convex
- Computes all-pairs shortest paths $\mathcal{O}(n^3)$
- Stores/processes dense matrix



DiffusionMap

- Distorts geodesic distances
- Computes only distances to nearest neighbors $\mathcal{O}(n^{1+\epsilon})$
- Stores/processes sparse matrix

ML Software

scikit-learn.org

mmp2.github.io/megaman

- t-SNE, UMAP visualization algorithms

Heuristic algorithms

- **Local Linear Embedding (LLE)**
- one of the first embedding algorithms
- later analysis showed that LLE has no limit when $n \to \infty$
- closest modern version is **Local Tangent Space Alignment (LTSA)**

- **t-Stochastic Neighbor Embedding (t-SNE)**

Input similarity matrix S, embedding dimension s

Init choose embedding points $y_{1:n} \in \mathbb{R}^s$ at random

1. $S_{ii} \leftarrow 0$, normalize rows $d_i = \sum_j S_{ij}$, $P_{ij} = S_{ij}/d_i$
2. symmetrize $P = \frac{1}{2n}(P + P^T)$ P is distribution over pairs of neighbors (i,j)
3. $\tilde{S}_{ij} = \tilde{\kappa}(\|y_i - y_j\|)$compute similarity in output space
   where $\tilde{\kappa}(z) = \frac{1}{1+z^2}$ the Cauchy (Student t with 1 degree of freedom)
4. Define distribution Q with $Q_{ij} \propto S_{ij}$
5. Change $y_{i:n}$ to decrease the **Kullbach-Leibler divergence** $KL(P||Q) = \sum_{i,j} P_{ij} \ln \frac{P_{ij}}{Q_{ij}}$ (by gradient descent) and repeat from step 3

- t-SNE is empirically useful for visualizing clusters
- t-SNE is proved to create artefacts

UMAP: Uniform Manifold Approximation and Projection [McInnes, Healy, Melville,2018]

Input k number nearest neighbors, d,

1. Find k-nearest neighbors
2. Construct (asymmetric) similarities w_{ij}, so that $\sum_j w_{ij} = \log_2 k$. $W = [w_{ij}]$.
3. Symmetrize $S = W + W^T - W.*W^T$ is similarity matrix.
4. Initialize embedding ϕ by LAPLACIANEIGENMAPS.
5. Optimize embedding.
   Iteratively for n_{iter} steps
   1. Sample an edge ij with probability $\propto \exp -d_{ij}$
   2. Move ϕ_i towards ϕ_j
   3. Sample a random j' uniformly
   4. Move ϕ_i away from $\phi_{j'}$

   Stochastic approximate **logistic regression** of $||\phi_i - \phi_j||$ on d_{ij}.

Output ϕ

Embedding algorithms summary

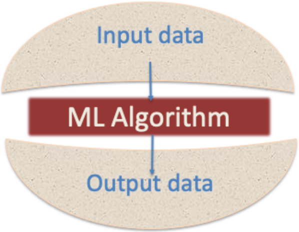
- Many different algorithms exist
- All start from neighborhood graph and distance matrix A
- Most use e-vectors of a tranformation of A (preserve the sparsity pattern)

- DiffusionMaps – can separate manifold shape from sampling density
- LTSA – "correct" at boundaries
- Isomap – best for flat manifolds with no holes, small data

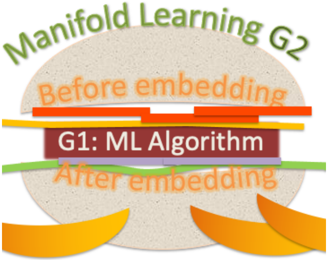
- Most embeddings sensitive to
  - choice of radius ϵ (within "correct" range)
  - sampling density p
  - neighborhoods K-nn vs. radius

  i.e. most embeddings introduce distortions

Manifold Learning as a sandwich



Manifold Learning as a sandwich



- what distance measure?
- what graph? [Maier,von Luxburg, Hein 2009]
- what kernel width ϵ? [Perrault-Joncas,M,McQueen NIPS17]
- what intrinsic dimension d? [Chen,Little,Maggioni,Rosasco] and variant by [Perrault-Joncas,M,McQueen NIPS17]
- what embedding dimension $\cancel{s}\ m \geq d$? [Chen,M,NeurIPS19]

ML Algorithm: DiffMaps, LTSA

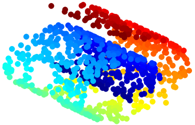
- Cluster [M,Shi 00],[M,Shi 01]... [M NeurIPS18]
- Estimate/correct distortion: Metric Learning and Riemannian Relaxation [McQueen, M, Perrault-Joncas NIPS16]
- Validate d, $\cancel{s}\ m$ [select eigenvectors] [Chen, M NeurIPS19]
- Topological Data Analysis (TDA)
- Meaning of coordinates [M,Koelle,Zhang, 2018,2022]

- Manifolds with vector fields [Perrault-Joncas, M, 2013, Chen, M, Kevrekidis 2021]
- Finding ridges and saddle points (in progress)

Outline

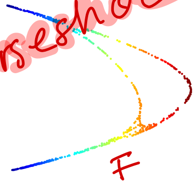
Embedding in 2 dimensions by different manifold learning algorithms

Original data
(Swiss Roll with hole)



Laplacian Eigenmaps (LE)

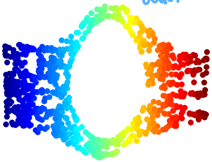
Horseshoe



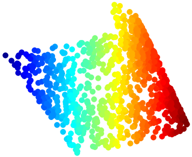
F

Isomap

distortion

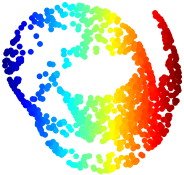


Hessian Eigenmaps (HE)



F

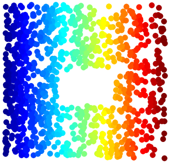
Local Linear Embedding (LLE)



[topology] failure

F

Local Tangent Space Alignment (LTSA)



affine distortion

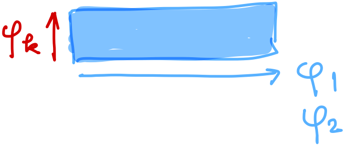
Failures vs. distortions

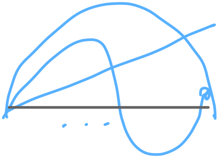
- Distortion vs failure
  - ϕ distorts if distances, angles, density not preserved, but ϕ smooth and invertible
  - If ϕ does not preserve topology (=preserve neighborhoods), then we call it a failure, for simplicity.
  - Examples: points ξ_i, ξ_j are not neighbors in $\mathcal{M}$ but are neighbors in $\phi(\mathcal{M})$, or viceversa (hence ϕ is not invertible, or not continuous)

- Most common modes of failure
  - distance matrix A does not capture topology (artificial "holes" or "bridges")
  - usually becasuse kernel width ϵ too small or too large
  - choice of e-vectors

Artefacts

- **Artefacts**=features of the embedding that do not exist in the data (clusters, holes, "arms", "horseshoes")
- What to beware of when you compute an embedding
  - algorithms that claim to choose ϵ automatically
  - confirming the embedding is "correct" by visualization: tends to over-smooth, i.e. ϵ over-estimated
  - K-nn (default in `sk-learn`!) instead of radius-neighbors: tends to create clusters
  - large variations in density: subsample data to make it more uniform
  - "horseshoes": choose other e-vectors (ϕ is almost singulare)
- Very popular heuristics (no guarantees/artefacts probable): LLE, t-SNE, UMAP, neural networks





- harmonics
- variations in $p()$

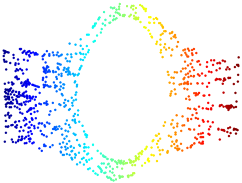
Preserving topology vs. preserving (intrinsic) geometry

- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$
- Mapping $\mathcal{M} \longrightarrow \phi(\mathcal{M})$ is diffeomorphism
  - preserves topology
  - often satisfied by embedding algorithms

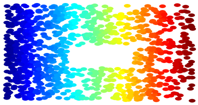
so far

- Mapping ϕ is **isometry**
  - preserves distances along curves in $\mathcal{M}$, angles, volumes
  - For most algorithms, in most cases, ϕ is not isometry

Preserves topology



Preserves topology + intrinsic geometry



Theoretical results in isometric embedding

Positive results

General theory

- Nash's Theorem: Isometric embedding is possible.
- Diffusion Maps embedding is isometric in the limit [Berard,Besson,Gallot 94],[Portegies:16]

Special cases

- Isomap [Bernstein, Langford, Tennenbaum 03] recovers flat manifolds isometrically
- LE/DM recover sphere, torus with equal radii (sampled uniformly)
    - Follows from consistency of Laplacian eigenvectors [Hein & al 07,Coifman & Lafon 06, Singer 06, Ting & al 10, Gine & Koltchinskii 06]

Negative results

- Obvious negative examples
- No affine recovery for normalized Laplacian algorithms [Goldberg&al 08]

Empirically, most algorithms

- preserve neighborhoods (=topology)
- distort distances along manifold (=geometry)
- distortions occur even in the simplest cases
- distortion persists when $n \to \infty$
- one cause of distortion is variations in sampling density p; [Coifman& Lafon 06] introduced Diffusion Maps (DM) to eliminate these

Metric Manifold Learning

Wanted

- eliminate distortions for any "well-behaved" $\mathcal{M}$
- and any any "well-behaved" embedding $\phi(\mathcal{M})$
- in a tractable and statistically grounded way

Metric Manifold Learning

Wanted

- eliminate distortions for any "well-behaved" $\mathcal{M}$
- and any any "well-behaved" embedding $\phi(\mathcal{M})$
- in a tractable and statistically grounded way

Idea

Given data $\mathcal{D} \subset \mathcal{M}$, some embedding $\phi(\mathcal{D})$ that preserves topology
(true in many cases)

- Estimate distortion of ϕ and correct it!

Metric Manifold Learning

Wanted

- eliminate distortions for any "well-behaved" $\mathcal{M}$
- and any any "well-behaved" embedding $\phi(\mathcal{M})$
- in a tractable and statistically grounded way

Idea

Given data $\mathcal{D} \subset \mathcal{M}$, some embedding $\phi(\mathcal{D})$ that preserves topology
(true in many cases)

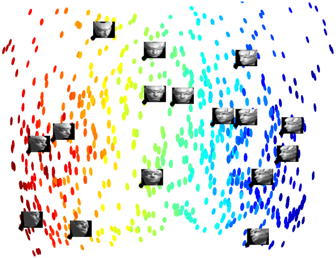
- Estimate distortion of ϕ and correct it!
- The correction is called the **pushforward Riemannian Metric** g
- The distortion is the **dual** pushforward Riemannian Metric h

$G_i \succeq 0$ rank d

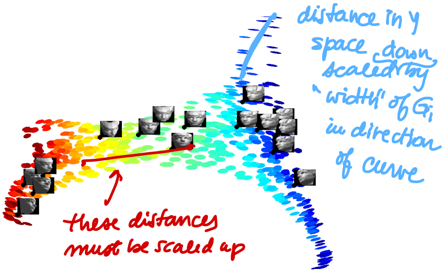
$H_i \succeq 0$

at point i

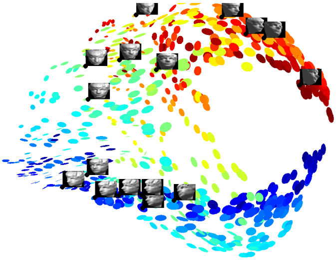
Corrections for 3 embeddings of the same data



Isomap



LTSA

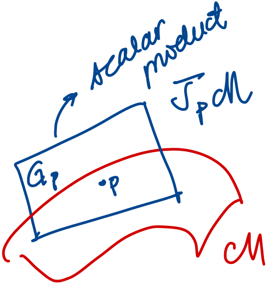


Laplacian Eigenmaps

What is a (Riemannian) metric?

- In Euclidean space $\mathbb{R}^d$, the **scalar product** $\langle u, v \rangle = u^T v$
- From the scalar product we derive norms $\|u\|^2 = \langle u, u \rangle$, distances $\|u - v\|$, angles $\cos(u, v) = \langle u, v \rangle / (\|u\| \|v\|)$.
- Any other scalar product on $\mathbb{R}^d$ is defined by $\langle u, v \rangle_G = u^T G v = (G^{1/2} u)^T (G^{1/2} v)$, with $G \succ 0$ defines the **metric**
- Note that whenever $G \succ 0$, $H = G^{-1} \succ 0$ also defines a metric

- On a manifold $\mathcal{M}$, at each $p \in \mathcal{M}$ we have a different G_p
- The function $g(p) = G_p$ is called the **Riemannian metric**



All (intrinsic) geometric quantities on $\mathcal{M}$ involve g

- Volume element on manifold

$$Vol(W) = \int_W \sqrt{\det(g)} dx^1 \dots dx^d .$$

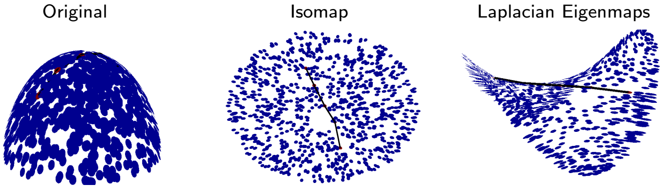
$G_i^{1/2}$ = Jacobian of embedding φ

- Length of curve γ

$$l(\gamma) = \int_a^b \sqrt{\sum_{ij} g_{ij} \frac{dx^i}{dt} \frac{dx^j}{dt}} dt,$$

- Under a change of parametrization, g changes in a way that leaves geometric quantities invariant

Calculating distances in the manifold $\mathcal{M}$



true distance $d = 1.57$

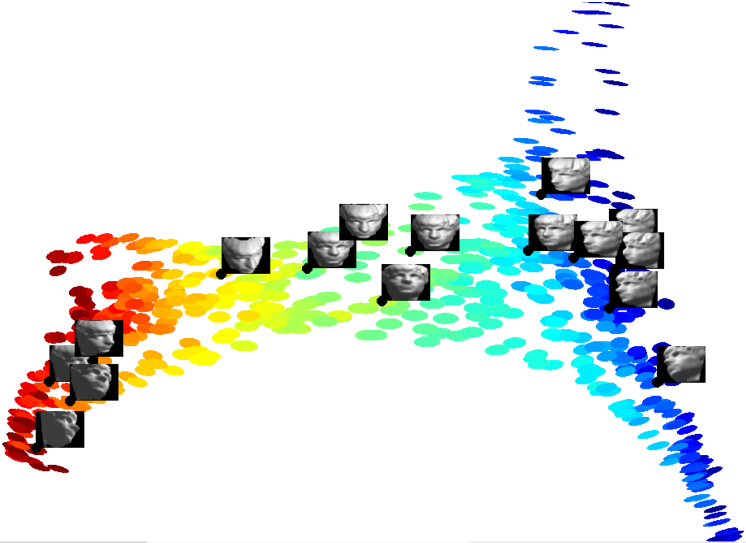
Embedding	$\|\|f(p) - f(p')\|\|$	Shortest Path	Metric $\hat{d}$	Rel. error
Original data	1.41	1.57	1.62	3.0%
Isomap $m = 2$	1.66	1.75	1.63	3.7%
LTSA $m = 2$	0.07	0.08	1.65	4.8%
LE $m = 2$	0.08	0.08	1.62	3.1%

curve $\gamma \approx (y_0, y_1, \ldots y_K)$ path in graph

$$\text{geodesic distance } \hat{d} = \sum_{k=0}^{K} \sqrt{(y_k - y_{k-1})^T \frac{G(y_k) + G(y_{k-1})}{2} (y_k - y_{k-1})}$$

G for Sculpture Faces

- $n = 698$ gray images of faces in $D = 64 \times 64$ dimensions
- head moves up/down and right/left



Problem: Estimate the g associated with ϕ

- Given:
  - data set $\mathcal{D} = \{p_1, \ldots p_n\}$ sampled from Riemannian manifold $(\mathcal{M}, g_0)$, $\mathcal{M} \subset \mathbb{R}^D$
  - embedding $\{ y_i = \phi(p_i),\ p_i \in \mathcal{D} \}$
    by e.g DiffusionMap, Isomap, LTSA, ...
- Estimate $G_i \in \mathbb{R}^{m \times m}$ the **pushforward Riemannian metric** at $p_i \in \mathcal{D}$
  in the embedding coordinates ϕ

- The embedding $\{y_{1:n}, G_{1:n}\}$ will preserve the geometry of the original data

Relation between g and Δ

- Δ = Laplace-Beltrami operator on $\mathcal{M}$
  - $\Delta = \mathrm{div} \cdot \mathrm{grad}$
  - on C^2, $\Delta f = \sum_j \frac{\partial^2 f}{\partial \xi_j^2}$
  - on weighted graph with similarity matrix S, and $t_p = \sum_{pp'} S_{pp'}$, $\Delta = \mathrm{diag}\{t_p\} - S$
- Δ = Laplace-Beltrami operator on $\mathcal{M}$
- G Riemannian metric (in coordinates)
- $H = G^{-1}$ matrix inverse

(Differential geometric fact)

$$\Delta f = \sqrt{\det(H)} \sum_l \frac{\partial}{\partial x^l} \left(\frac{1}{\sqrt{\det(H)}} \sum_k H_{lk} \frac{\partial}{\partial x^k} f \right),$$

- L the renormalized Laplacian estimates Δ (very well studied ✓)

Estimation of G^{-1}

Let Δ be the Laplace-Beltrami operator on $\mathcal{M}$, $H = G^{-1}$, and $k, l = 1, 2, \ldots d$.

$$\frac{1}{2}\Delta(\phi_k - \phi_k(p))\,(\phi_l - \phi_l(p))|_{\phi_k(p),\phi_l(p)} \;=\; H_{kl}(p)$$

Intuition:

- Δ applied to test functions $f = \phi_k^{\text{centered}}\phi_l^{\text{centered}}$
- this produces $H(p)$ in the given coordinates
- consistent estimation of Δ is well studied [Coifman&Lafon 06,Hein&al 07]

Metric Manifold Learning algorithm

Given dataset $\mathcal{D}$

1. Preprocessing (construct neighborhood graph, ...)
2. Find an embedding ϕ of $\mathcal{D}$ into $\mathbb{R}^m$
3. Estimate discretized Laplace-Beltrami operator L
4. Estimate H_p and $G_p = H_p^\dagger$ for all p
   1. For $i, j = 1 : m$,
      $H^{ij} = \frac{1}{2}\left[L(\phi_i * \phi_j) - \phi_i * (L\phi_j) - \phi_j * (L\phi_i)\right]$
      where $X * Y$ denotes elementwise product of two vectors $X, Y \in \mathbb{R}^N$
   2. For $p \in \mathcal{D}$, $H_p = [H_p^{ij}]_{ij}$
   3. For $p \in \mathcal{D}$, $(V, \Sigma) \leftarrow SVD(H_p, d)$ and $G_p = V\Sigma^{-1}V^T = H_p^\dagger$ (rank d (pseudo)inverse of H_p)

Output (ϕ_p, G_p) for all p

Computational cost

$n = |\mathcal{D}|$, $D =$ data dimension, $m =$ embedding dimension

1. Neighborhood graph $+$
2. Similarity matrix $\mathcal{O}(n^2 D)$ (or less)
3. Laplacian $\mathcal{O}(n^2)$
4. EMBEDDINGALG e.g. $\mathcal{O}(mn^2)$ (eigenvector calculations)
5. Embedding metric
   - $\mathcal{O}(nm^2)$ obtain g^{-1} or $h^\dagger$
   - $\mathcal{O}(nm^3)$ obtain g or h

- Steps 1–3 are part of many embedding algorithms
- Steps 3–5 independent of ambient dimension D
- Matrix inversion/pseudoinverse can be performed only when needed

Metric Manifold Learning summary

Why useful

- Measures local distortion induced by any embedding algorithm
  $G_i = I_d$ when no distortion at p_i
- Corrects distortion
  - Integrating with the local volume/length units based on G_i
  - Riemannian Relaxation [McQueen, M, Perrault-Joncas NIPS16]
- Algorithm independent geometry preserving method
- Outputs of different algorithms on the same data are comparable

Applications

- Estimation of neighborhood radius [Perrault-Joncas,M,McQueen NIPS17]
- Helps with estimation of intrinsic dimension d (variant of [Chen,Little,Maggioni,Rosasco])
- selecting eigencoordinates [Chen, M NeurIPS19]

Outline

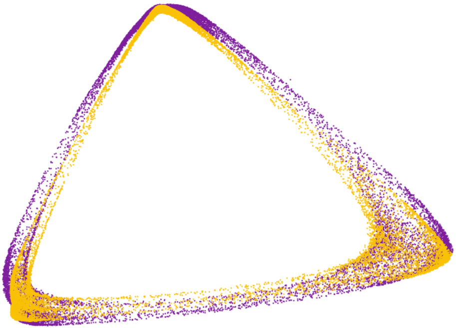
What graph? Radius-neighbors vs. k nearest-neighbors

- **k-nearest neighbors graph:** each node has degree k
- **radius neighbors graph:** p, p' neighbors iff $||p - p'|| \leq r$

- Does it matter?

What graph? Radius-neighbors vs. k nearest-neighbors

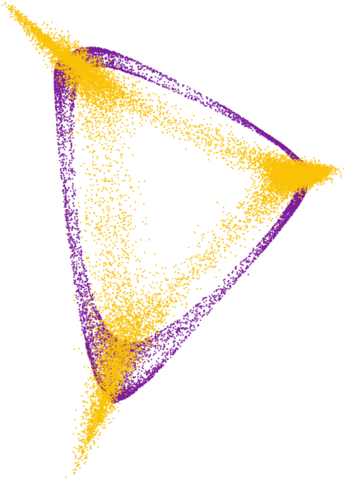
- **k-nearest neighbors graph:** each node has degree k
- **radius neighbors graph:** p, p' neighbors iff $||p - p'|| \leq r$ → L unbiased
- Does it matter?
- Yes, for estimating the Laplacian and distortion
  - Why? [Hein 07, Coifman 06, Ting 10, ...] k-nearest neighbor Laplacians do not converge to Laplace-Beltrami operator Δ
  - but to $\Delta + 2\nabla(\log p) \cdot \nabla$ (**bias** due to non-uniform sampling)



K-nearest neighbor
radius neighbor

configurations of ethanol $d = 2$

Effect of re-normalization



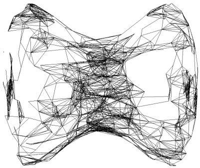
L^n simply normalized

L renormalized

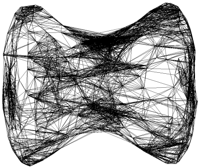
Choosing ϵ

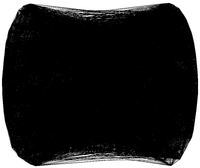
- Every manifold learning algorithm starts with a neighborhood graph
- Parameter ϵ
  - is neighborhood radius
  - and/or kernel banwidth

- recall $\kappa(p, p') = e^{-\frac{||p-p'||^2}{\epsilon^2}}$ if $||p - p'||^2 \leq c\epsilon$ and 0 otherwise ($c \in [1, 10]$)



ϵ too small





ϵ too large

Methods for choosing ϵ

- Theoretical (asymptotic) result $\sqrt{\epsilon} \propto n^{-\frac{1}{d+6}}$ [Singer06]

  In practice:
- Visual inspection? → tends to oversmooth
- Cross-validation ?
  - only if related to prediction task
- [Chen&Buja09] heuristic for k-nearest neighbor graph
  - unsupervised
  - depends on embedding method used
  - optimizes consistency of k-nn graph in data and embedding
  - k-nearest neighbor graph has different convergence properties than ϵ neighborhood
- Geometric Consistency heuristic [Perrault-Joncas&Meila17]
  - unsupervised
  - optimizes Laplacian, does not require embedding
  - computes "isometry" in 2 different ways and minimizes distortion between them

Geometric Consistency (GC): Idea

- Idea: choose ϵ so that geometry encoded by L_ϵ is closest to data geometry

- For given ϵ and data point p
  1. Project neighbors of p onto tangent subspace
     - local embedding around p
     - approximately isometric to original data
  2. Calculate Laplacian $L(\epsilon)$ at p and estimate distortion $\boxed{H_{\epsilon,p}}$
     - $H_{\epsilon,p}$ must be $\approx I_d$ identity matrix

dual
push-forward R.m

$\approx$

Id

The distortion measure

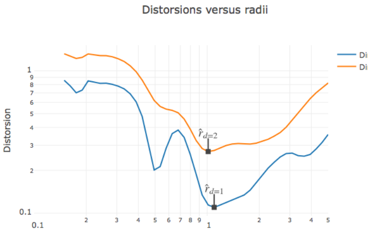
Input: data set $\mathcal{D}$, dimension $d' \leq d$, scale ϵ

1. Estimate Laplacian $L(\epsilon)$ and weights $w_i(\epsilon)$ with LAPLACIAN
2. Project data on tangent plane at p
   - For each p
   - Let $\text{neigh}_{p,\epsilon} = \{p' \in \mathcal{D},\ \|p' - p\| \leq c\epsilon\}$ where $c \in [1, 10]$
   - Calculate (weighted) local PCA wLPCA$(\text{neigh}_{p,\epsilon}, d')$ (with weights $w_i(\epsilon)$)
   - Calculate coordinates z_i in PCA space for points in $\text{neigh}_{p,\epsilon}$
3. Estimate $H_{\epsilon,p} \in \mathbb{R}^{d' \times d'}$ by RMETRIC
   - For each p
   - Use row p of $L(\epsilon)$
   - z_i's play the role of ϕ
4. Compute squared Loss over all p's $\mathrm{Loss}(\epsilon) = \sum_{p \in \mathcal{D}} ||H_{\epsilon,p} - I_d||_2^2$

Output $\mathrm{Loss}(\epsilon)$

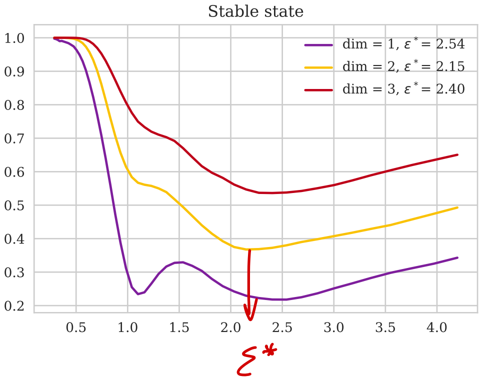
- Select $\epsilon^* = \mathrm{argmin}_\epsilon \mathrm{Loss}(\epsilon)$

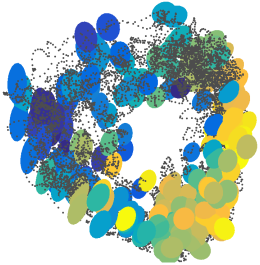
- $d' \leq d$ (more robust)
- minimize by 0-th order optimization (faster than grid search)



Example ϵ and distortion for aspirin

- Each point $=$ a configuration of the aspirin molecule
- Cloud of point in $D = 47$ dimensions embedded in $m = 3$ dimensions
- (only 1 cluster shown)

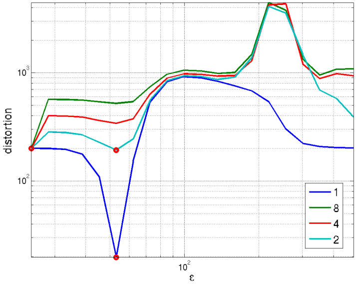




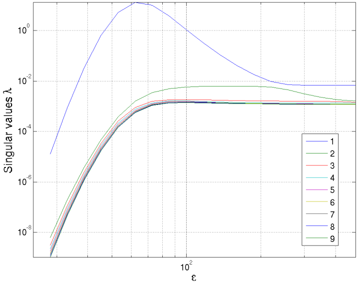
Bonus: Intrinsic Dimension Estimation in noise

- Geometric consistency + eigengap method of [Chen,Little,Maggioni,Rosasco,2011]
  1. do local PCA for a range of ϵ values
  2. choose appropriate radius ϵ (by Geometric consistency)
  3. dimension = largest eigengap between λ_k and λ_{k+1} at radius ϵ (proof by Chen&al)
     ("largest" = most frequent largest over a sample)

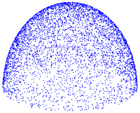
Loss(ϵ) vs. ϵ

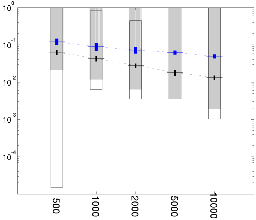


Singular values of LPCA vs. ϵ

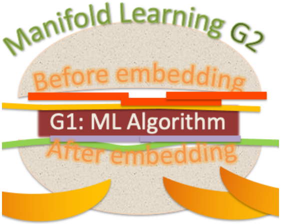


Example: Intrinsic Dimension Estimation results





Summary



- what distance measure?
- what graph? [Maier,von Luxburg, Hein 2009]
- what kernel width ϵ? [Perrault-Joncas,M,McQueen NIPS17]
- what intrinsic dimension d? [Chen,Little,Maggioni,Rosasco] and variant by [Perrault-Joncas,M,McQueen NIPS17]
- what embedding dimension $m \geq d$? [Chen,M,NeurIPS19]

ML Algorithm: DiffMaps, LTSA

- Cluster [M,Shi 00],[M,Shi 01]. . . [M NeurIPS18]
- Estimate/correct distortion: Metric Learning and Riemannian Relaxation [McQueen, M, Perrault-Joncas NIPS16]
- Validate d, m, [select eigenvectors] [Chen, M NeurIPS19]
- Topological Data Analysis (TDA)
- Meaning of coordinates [M,Koelle,Zhang, 2018,2022]

- Manifolds with vector fields [Perrault-Joncas, M, 2013, Chen, M, Kevrekidis 2021]
- Finding ridges and saddle points (in progress)

Thank You! Q?