Is Manifold Learning for Toy Data only?

Marina Meilă

University of Washington
mmp@stat.washington.edu

MMDS Workshop 2016

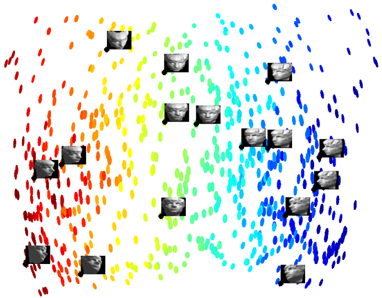
Outline

What is non-linear dimension reduction?

Metric Manifold Learning
- Estimating the Riemannian metric
- Riemannian Relaxation

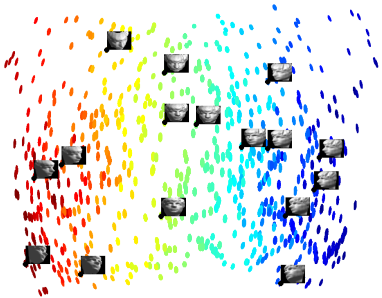
Scalable manifold learning
- megaman
- An application to scientific data

When to do (non-linear) dimension reduction



- high-dimensional data $p \in \mathbb{R}^D$, $D = 64 \times 64$
- can be described by a small number d of continuous parameters
- Usually, large sample size n

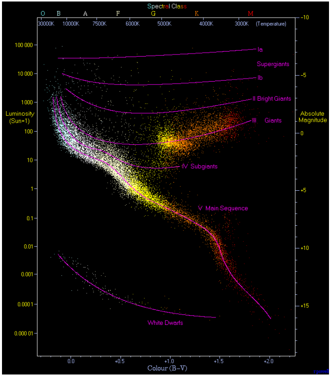
When to do (non-linear) dimension reduction



Why?

- To save space and computation
  - $n \times D$ data matrix $\rightarrow$ $n \times s$, $s \ll D$
- To understand the data better
  - preserve large scale features, suppress fine scale features
- To use it afterwards in (prediction) tasks

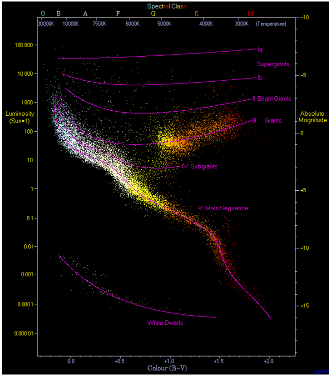
When to do (non-linear) dimension reduction



Why?

- To save space and computation
  - $n \times D$ data matrix $\rightarrow$ $n \times s$, $s \ll D$
- To understand the data better
  - preserve large scale features, suppress fine scale features
- To use it afterwards in (prediction) tasks

When to do (non-linear) dimension reduction

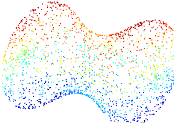


Why?

- To save space and computation
  - $n \times D$ data matrix $\rightarrow$ $n \times s$, $s \ll D$
- To understand the data better
  - preserve large scale features, suppress fine scale features
- To use it afterwards in (prediction) tasks

How? Brief intro to manifold learning algorithms

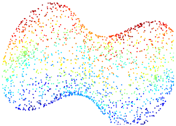
- **Input** Data $p_1, \dots p_n$, embedding dimension m, neighborhood scale parameter ϵ



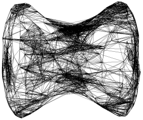
$p_1, \dots p_n \subset \mathbb{R}^D$

How? Brief intro to manifold learning algorithms

- **Input** Data $p_1, \ldots p_n$, embedding dimension m, neighborhood scale parameter ϵ
- Construct neighborhood graph p, p' neighbors iff $||p - p'||^2 \le \epsilon$

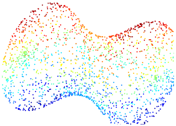


$p_1, \ldots p_n \subset \mathbb{R}^D$

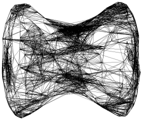


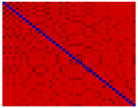
How? Brief intro to manifold learning algorithms

- **Input** Data $p_1, \ldots p_n$, embedding dimension m, neighborhood scale parameter ϵ
- Construct neighborhood graph p, p' neighbors iff $||p - p'||^2 \leq \epsilon$
- Construct a $n \times n$ matrix its leading eigenvectors are the coordinates $\phi(p_{1:n})$



$p_1, \ldots p_n \subset \mathbb{R}^D$





How? Brief intro to manifold learning algorithms

- **Input** Data $p_1, \dots p_n$, embedding dimension m, neighborhood scale parameter ϵ
- Construct neighborhood graph p, p' neighbors iff $||p - p'||^2 \leq \epsilon$
- Construct a $n \times n$ matrix its leading eigenvectors are the coordinates $\phi(p_{1:n})$

LAPLACIAN EIGENMAPS [Belkin & Niyogi 02]

- Construct similarity matrix

$$S = [S_{pp'}]_{p,p' \in \mathcal{D}} \quad \text{with} S_{pp'} = e^{-\frac{1}{\epsilon}||p-p'||^2} \quad \text{iff } p, p' \text{ neighbors}$$

- Construct Laplacian matrix $L = I - T^{-1}S$ with $T = \mathrm{diag}(S\mathbf{1})$
- Calculate $\psi^{1\dots m}$ = eigenvectors of L (smallest eigenvalues)
- coordinates of $p \in \mathcal{D}$ are $(\psi^1(p), \dots \psi^m(p))$

How? Brief intro to manifold learning algorithms

- **Input** Data $p_1, \ldots p_n$, embedding dimension m, neighborhood scale parameter ϵ
- Construct neighborhood graph p, p' neighbors iff $||p - p'||^2 \leq \epsilon$
- Construct a $n \times n$ matrix its leading eigenvectors are the coordinates $\phi(p_{1:n})$

Isomap [[Tennenbaum, deSilva & Langford 00]]

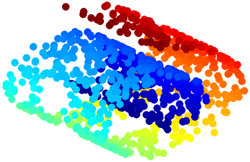
- Find all shortest paths in neighborhood graph, construct matrix of distances

$$M = [\text{distance}^2_{pp'}]$$

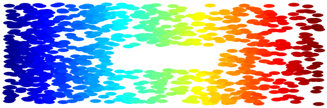
- use M and Multi-Dimensional Scaling (MDS) to obtain m dimensional coordinates for $p \in \mathcal{D}$

A toy example (the "Swiss Roll" with a hole)

points in $D \geq 3$ dimensions

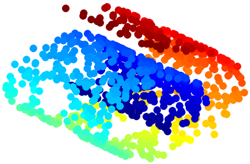


same points reparametrized in 2D



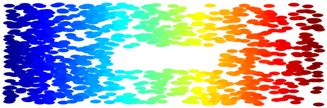
A toy example (the "Swiss Roll" with a hole)

points in $D \geq 3$ dimensions



Input

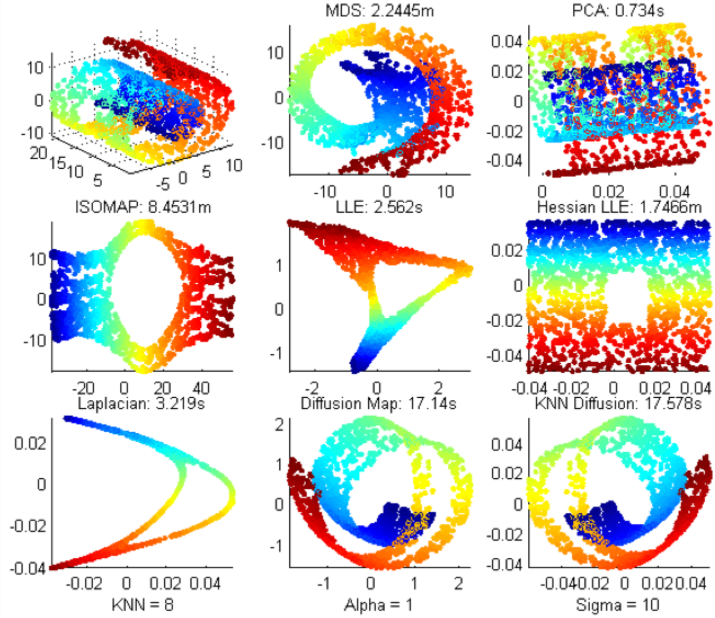
same points reparametrized in 2D



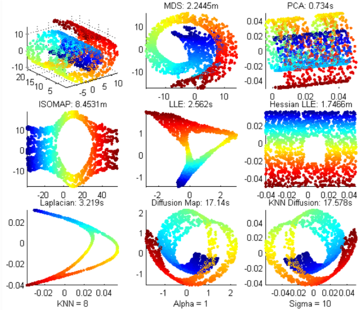
Desired output

Embedding in 2 dimensions by different manifold learning algorithms

Input



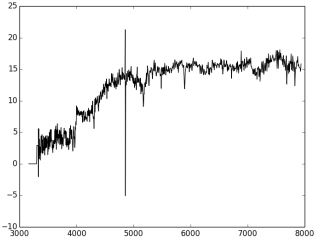
How to evaluate the results objectively?



- which of these embedding are "correct"?
- if several "correct", how do we reconcile them?
- if not "correct", what failed?

Algorithms Multidimensional Scaling (MDS), Principal Components (PCA), Isomap, Locally Linear Embedding (LLE), Hessian Eigenmaps (HE), Laplacian Eigenmaps (LE), Diffusion Maps (DM)

How to evaluate the results objectively?



Spectrum of a galaxy. Source SDSS, Jake VanderPlas

- which of these embedding are "correct"?
- if several "correct", how do we reconcile them?
- if not "correct", what failed?
- what if I have real data?

Outline

Outline

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

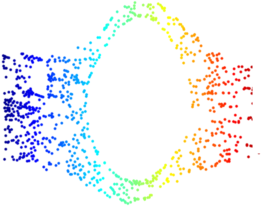
  often satisfied by embedding algorithms
- Mapping ϕ preserves
  - distances along curves in $\mathcal{M}$
  - angles between curves in $\mathcal{M}$
  - areas, volumes

Preserving topology vs. preserving (intrinsic) geometry

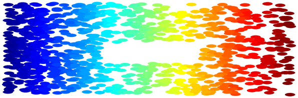
- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$
- Mapping $\mathcal{M} \longrightarrow \phi(\mathcal{M})$ is diffeomorphism
  preserves topology
  often satisfied by embedding algorithms
- Mapping ϕ preserves
  - distances along curves in $\mathcal{M}$
  - angles between curves in $\mathcal{M}$
  - areas, volumes

  ...i.e. ϕ is isometry
  For most algorithms, in most cases, ϕ is not isometry

Preserves topology



Preserves topology + intrinsic geometry



Previous known results in geometric recovery

Positive results

- Nash's Theorem: Isometric embedding is possible.
- algorithm based on Nash's theorem (isometric embedding for very low d) [Verma 11]
- Isomap recovers (only) flat manifolds isometrically
- Consistency results for Laplacian and eigenvectors
  - [[Hein & al 07,Coifman & Lafon 06, Ting & al 10, Gine & Koltchinskii 06]]
  - imply isometric recovery for LE, DM in special situations

Negative results

- obvious negative examples
- No affine recovery for normalized Laplacian algorithms [Goldberg&al 08]
- Sampling density distorts the geometry for LE [Coifman& Lafon 06]

Our approach: Metric Manifold Learning

[Perrault-Joncas,M 10]

Given

- mapping ϕ that preserves topology

true in many cases

Objective

- augment ϕ with geometric information g so that (ϕ, g) preserves the geometry

Dominique Perrault-Joncas

Our approach: Metric Manifold Learning

[Perrault-Joncas,M 10]

Given

- mapping ϕ that preserves topology

  true in many cases

Objective

- augment ϕ with geometric information g so that (ϕ, g) preserves the geometry

g is the Riemannian metric.

Dominique Perrault-Joncas

The Riemannian metric g

Mathematically

- $\mathcal{M}$ = (smooth) manifold
- p point on $\mathcal{M}$
- $T_p\mathcal{M}$ = **tangent subspace** at p
- g = Riemannian metric on $\mathcal{M}$

  g defines inner product on $T_p\mathcal{M}$

$$< v, w >= v^T g_p w \quad \text{for } v, w \in T_p\mathcal{M} \text{ and for } p \in \mathcal{M}$$

  - g is symmetric and positive definite tensor field
  - g also called first differential form
  - $(\mathcal{M}, g)$ is a Riemannian manifold

Computationally at each point $p \in \mathcal{M}$, g_p is a positive definite matrix of rank d

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

- Volume element on manifold

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

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

- Volume element on manifold

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

- Length of curve c

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

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

- Volume element on manifold

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

- Length of curve c

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

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

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

- Volume element on manifold

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

- Length of curve c

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

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

- Current algorithms: estimate $\mathcal{M}$
- This talk: estimate g along with $\mathcal{M}$

  (and in the same coordinates)

Problem formulation

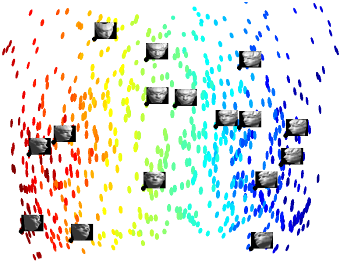
- Given:
  - data set $\mathcal{D} = \{p_1, \dots p_n\}$ sampled from manifold $\mathcal{M} \subset \mathbb{R}^D$
  - embedding $\{ x_i = \phi(p_i),\ p_i \in \mathcal{D} \}$
    by e.g LLE, Isomap, LE, ...
- Estimate $G_i \in \mathbb{R}^{m \times m}$ the (pushforward) Riemannian metric for $p_i \in \mathcal{D}$ in the embedding coordinates ϕ

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

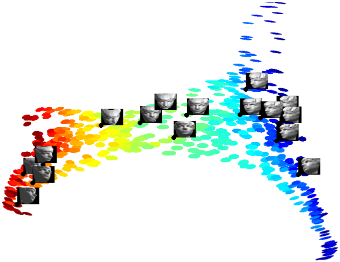
g for Sculpture Faces

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

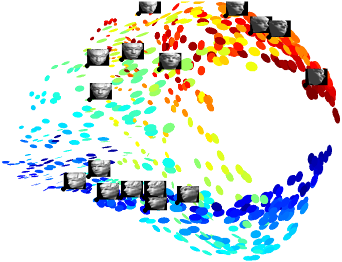
LTSA Algoritm



Isomap



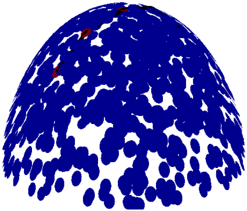
LTSA



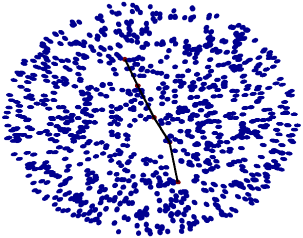
Laplacian Eigenmaps

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

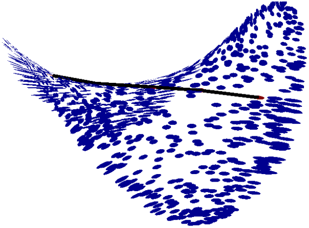
- Geodesic distance $=$ shortest path on $\mathcal{M}$
- should be invariant to coordinate changes



Original



Isomap



Laplacian Eigenmaps

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

true distance $d = 1.57$

Embedding	$\|\|f(p) - f(p')\|\|$	Shortest Path $d_{\mathcal{G}}$	Metric $\hat{d}$	Rel. error
Original data	1.41	1.57	1.62	3.0%
Isomap $s = 2$	1.66	1.75	1.63	3.7%
LTSA $s = 2$	0.07	0.08	1.65	4.8%
LE $s = 3$	0.08	0.08	1.62	3.1%

Relation between g and Δ

- Δ = Laplace-Beltrami operator on $\mathcal{M}$

Proposition 1 (Differential geometric fact)

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

where $h = g^{-1}$ (matrix inverse)

Relation between g and Δ

- Δ = Laplace-Beltrami operator on $\mathcal{M}$
  - $\Delta = \mathrm{div} \cdot \mathrm{grad}$
  - on $C^2(\mathbb{R}^d)$, $\Delta f = \sum_j \frac{\partial^2 f}{\partial x_j^2}$
  - on weighted graph with similarity matrix S, and $t_p = \sum_{pp'} S_{pp'}$, $\Delta = \mathrm{diag}\{ t_p \} - S$

Proposition 1 (Differential geometric fact)

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

where $h = g^{-1}$ (matrix inverse)

Estimation of g

Proposition 2 (Main Result 1)

Let Δ be the Laplace-Beltrami operator on $\mathcal{M}$. Then

$$h_{ij}(p) = \frac{1}{2}\Delta(\phi_i - \phi_i(p))(\phi_j - \phi_j(p))|_{\phi_i(p),\phi_j(p)}$$

where $h = g^{-1}$ (matrix inverse) and $i, j = 1, 2, \ldots m$ are embedding dimensions

Algorithm to Estimate Riemann metric g (Main Result 2)

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 $L \in \mathbb{R}^{n \times n}$
4. Estimate $H_p = G_p^{-1}$ and $G_p = H_p^{\dagger}$ for all $p \in \mathcal{D}$

Output (ϕ_p, G_p) for all p

Algorithm to Estimate Riemann metric g (Main Result 2)

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 $H_p = G_p^{-1}$ and $G_p = H_p^\dagger$ for all p

   4.1 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 $X, Y \in \mathbb{R}^N$

   4.2 For $p \in \mathcal{D}$, $H_p = [H_p^{ij}]_{ij}$ and $G_p = H_p^\dagger$

Output (ϕ_p, G_p) for all p

Metric Manifold Learning summary

Metric Manifold Learning = estimating (pushforward) Riemannian metric G_i along with embedding coordinates x_i 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 (coming next)
- Algorithm independent geometry preserving method
- Outputs of different algorithms on the same data are comparable

Outline

Riemannian Relaxation

Sometimes we can dispense with g

Idea

- If embedding is isometric, then push-forward metric is identity matrix I_d

Riemannian Relaxation

Sometimes we can dispense with g

Idea

- If embedding is isometric, then push-forward metric is identity matrix I_d

Idea, formalized

- Measure distortion by loss $= \sum_{i=1}^{n} ||G_i - I_d||^2$
  - where G_i is R. metric estimate at point i
  - I_d is identity matrix
- Iteratively change embedding $x_{1:n}$ to minimize loss

Riemannian Relaxation

Sometimes we can dispense with g

Idea

- If embedding is isometric, then push-forward metric is identity matrix I_d

Idea, formalized

- Measure distortion by loss $= \sum_{i=1}^{n} ||G_i - I_d||^2$
  - where G_i is R. metric estimate at point i
  - I_d is identity matrix
- Iteratively change embedding $x_{1:n}$ to minimize loss

More details

- loss is non-convex
- $||\ ||$ is derived from operator norm
- Extends to $s > d$ embeddings loss $= \sum_{i=1}^{n} ||G_i - U_i U_i^T||_\sigma^2$
- Extensions to principal curves and surfaces [Ozertem, Erdogmus 11], subsampling, non-uniform sampling densities

Riemannian Relaxation

Sometimes we can dispense with g

Idea

- If embedding is isometric, then push-forward metric is identity matrix I_d

Idea, formalized

- Measure distortion by loss $= \sum_{i=1}^{n} ||G_i - I_d||^2$
  - where G_i is R. metric estimate at point i
  - I_d is identity matrix
- Iteratively change embedding $x_{1:n}$ to minimize loss

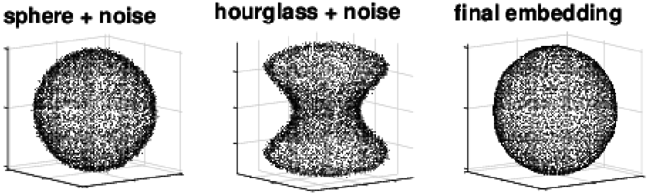
More details

- loss is non-convex
- $|| \ ||$ is derived from operator norm
- Extends to $s > d$ embeddings loss $= \sum_{i=1}^{n} ||G_i - U_i U_i^T||_\sigma^2$
- Extensions to principal curves and surfaces [Ozertem, Erdogmus 11], subsampling, non-uniform sampling densities

Implementation

- Initialization with e.g Laplacian Eigenmaps
- Projected gradient descent to (local) optimum

Riemannian Relaxation of a deformed sphere

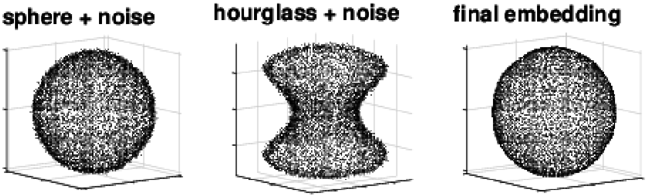


target

initialization

algorithm output

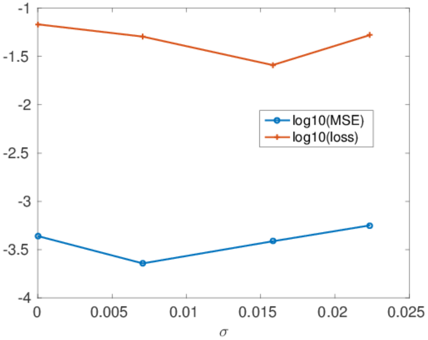
Riemannian Relaxation of a deformed sphere



target

initialization

algorithm output



Mean-squared error and loss vs. noise amplitude

Outline

Outline

Scaling: Statistical viewpoint

Rates of convergence as $n \longrightarrow \infty$

- Assume data sampled from manifold $\mathcal{M}$ with intrinsic dimension d,
  - $\mathcal{M}$, sampling distribution are "well behaved"
  - ϵ kernel bandwidth decreases slowly with n
- rate of Laplacian $n^{-\frac{1}{d+6}}$ [Singer 06], and of its eigenvectors $n^{-\frac{2}{(5d+6)(d+6)}}$ [Wang 15]
- minimax rate of manifold learning $n^{-\frac{2}{d+2}}$ [Genovese et al. 12]

Scaling: Statistical viewpoint

Rates of convergence as $n \longrightarrow \infty$

- Assume data sampled from manifold $\mathcal{M}$ with intrinsic dimension d,
  - $\mathcal{M}$, sampling distribution are "well behaved"
  - ϵ kernel bandwidth decreases slowly with n
- rate of Laplacian $n^{-\frac{1}{d+6}}$ [Singer 06], and of its eigenvectors $n^{-\frac{2}{(5d+6)(d+6)}}$ [Wang 15]
- minimax rate of manifold learning $n^{-\frac{2}{d+2}}$ [Genovese et al. 12]
- Estimating $\mathcal{M}$ and Δ accurately **requires big data**

Scaling: Computational viewpoint

LAPLACIAN EIGENMAPS revisited

1. Construct similarity matrix

   $$S = [S_{pp'}]_{p,p' \in \mathcal{D}} \text{ with} S_{pp'} = e^{-\frac{1}{\epsilon}||p-p'||^2}$$

   iff p, p' neighbors
2. Construct Laplacian matrix
   $L = I - T^{-1}S$ with $T = \mathrm{diag}(S\mathbf{1})$
3. Calculate $\psi^{1\dots m}$ = eigenvectors of L
   (smallest eigenvalues)
4. coordinates of $p \in \mathcal{D}$ are
   $(\psi^1(p), \dots \psi^m(p))$

Scaling: Computational viewpoint

LAPLACIAN EIGENMAPS revisited

1. Construct similarity matrix

   $$S = [S_{pp'}]_{p,p' \in \mathcal{D}} \text{ with} S_{pp'} = e^{-\frac{1}{\epsilon}||p-p'||^2}$$

   Nearest neighbor search in high dimensions

   iff p, p' neighbors
2. Construct Laplacian matrix
   $L = I - T^{-1}S$ with $T = \mathrm{diag}(S\mathbf{1})$
3. Calculate $\psi^{1\ldots m}$ = eigenvectors of L
   (smallest eigenvalues)
4. coordinates of $p \in \mathcal{D}$ are
   $(\psi^1(p), \ldots \psi^m(p))$

Scaling: Computational viewpoint

LAPLACIAN EIGENMAPS revisited

1. Construct similarity matrix

   $$S = [S_{pp'}]_{p,p' \in \mathcal{D}} \text{ with } S_{pp'} = e^{-\frac{1}{\epsilon}||p-p'||^2}$$

   iff p, p' neighbors
2. Construct Laplacian matrix
   $L = I - T^{-1}S$ with $T = \mathrm{diag}(S\mathbf{1})$
3. Calculate $\psi^{1\ldots m}$ = eigenvectors of L
   (smallest eigenvalues)
4. coordinates of $p \in \mathcal{D}$ are
   $(\psi^1(p), \ldots \psi^m(p))$

Nearest neighbor search in high dimensions

Sparse Matrix Vector multiplication

Scaling: Computational viewpoint

LAPLACIAN EIGENMAPS revisited

1. Construct similarity matrix

   $$S = [S_{pp'}]_{p,p' \in \mathcal{D}} \text{ with } S_{pp'} = e^{-\frac{1}{\epsilon}||p-p'||^2}$$

   iff p, p' neighbors
2. Construct Laplacian matrix
   $L = I - T^{-1}S$ with $T = \mathrm{diag}(S\mathbf{1})$
3. Calculate $\psi^{1\ldots m}$ = eigenvectors of L
   (smallest eigenvalues)
4. coordinates of $p \in \mathcal{D}$ are
   $(\psi^1(p), \ldots \psi^m(p))$

Nearest neighbor search in high dimensions

Sparse Matrix Vector multiplication

Principal eigenvectors

- of sparse, symmetric, (well conditioned) matrix

Manifold Learning with millions of points

https://www.github.com/megaman

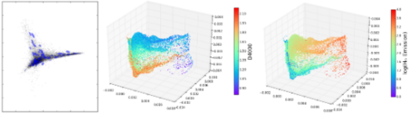
James McQueen	Jake VanderPlas	Jerry Zhang	Grace Telford
			

- Implemented in python, compatible with scikit-learn
- Designed for performance
  - sparse representation as default
  - incorporates state of the art FLANN package[1]
  - uses amp, lobpcg fast sparse eigensolver for SDP matrices
  - exposes/caches intermediate states (e.g. data set index, distances, Laplacian, eigenvectors)

[1] Fast Approximate Nearest Neighbor search

https://www.github.com/megaman

megaman: Manifold Learning for Millions of Points



build passing | pypi v0.1.1 | license BSD

`megaman` is a scalable manifold learning package implemented in python. It has a front-end API designed to be familiar to scikit-learn but harnesses the C++ Fast Library for Approximate Nearest Neighbors (FLANN) and the Sparse Symmetric Positive Definite (SSPD) solver Locally Optimal Block Precodition Gradient (LOBPCG) method to scale manifold learning algorithms to large data sets. On a personal computer megaman can embed 1 million data points with hundreds of dimensions in 10 minutes. megaman is designed for researchers and as such caches intermediary steps and indices to allow for fast re-computation with new parameters.

Package documentation can be found at http://mmp2.github.io/megaman/

You can also find our arXiv paper at http://arxiv.org/abs/1603.02763

Examples

- Tutorial Notebook

Installation with Conda

The easiest way to install `megaman` and its dependencies is with conda, the cross-platform package manager for the scientific Python ecosystem.

James McQueen

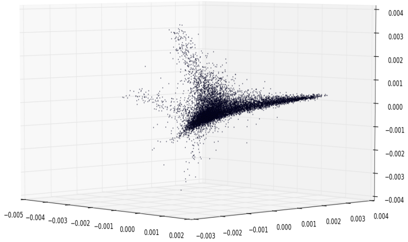
Jake VanderPlas

Jerry Zhang

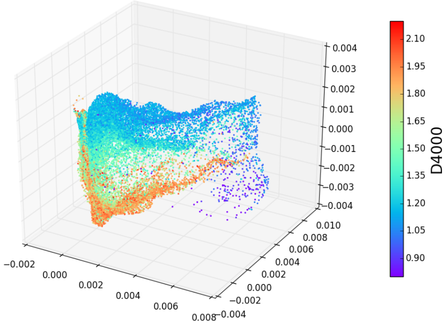
Grace Telford

Scalable Manifold Learning in python with megaman

https://www.github.com/megaman



English words and phrases taken from Google news (3,000,000 phrases originally represented in 300 dimensions by the Deep Neural Network word2vec [Mikolov et al])

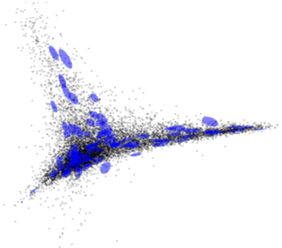


Main sample of galaxy spectra from the Sloan Digital Sky Survey (675,000 spectra originally in 3750 dimensions).

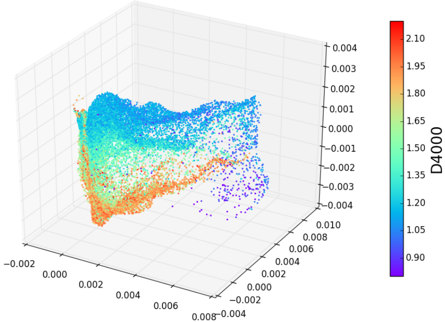
preprocessed by Jake VanderPlas, figure by Grace Telford

Scalable Manifold Learning in python with megaman

https://www.github.com/megaman



English words and phrases taken from Google news (3,000,000 phrases originally represented in 300 dimensions by the Deep Neural Network word2vec [Mikolov et al])



Main sample of galaxy spectra from the Sloan Digital Sky Survey (675,000 spectra originally in 3750 dimensions).

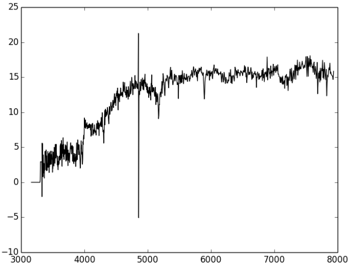
preprocessed by Jake VanderPlas, figure by Grace Telford

- Currently: on single core, embeds all data, all data in memory
- Near future: Nyström extension, lazy evaluations, multiple charts
- Next
  - `gigaman`?
  - scalable geometric/statistical tasks (search for optimal ϵ, Riemannian Relaxation, semi-supervised learning, clustering)

Outline

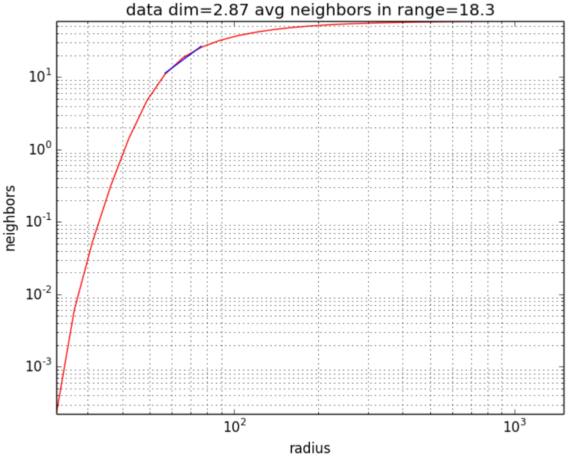
Manifold learning for SDSS Spectra of Galaxies (more in next talk!)

Main sample of galaxy spectra from the Sloan Digital Sky Survey (675,000 spectra originally in 3750 dimensions).

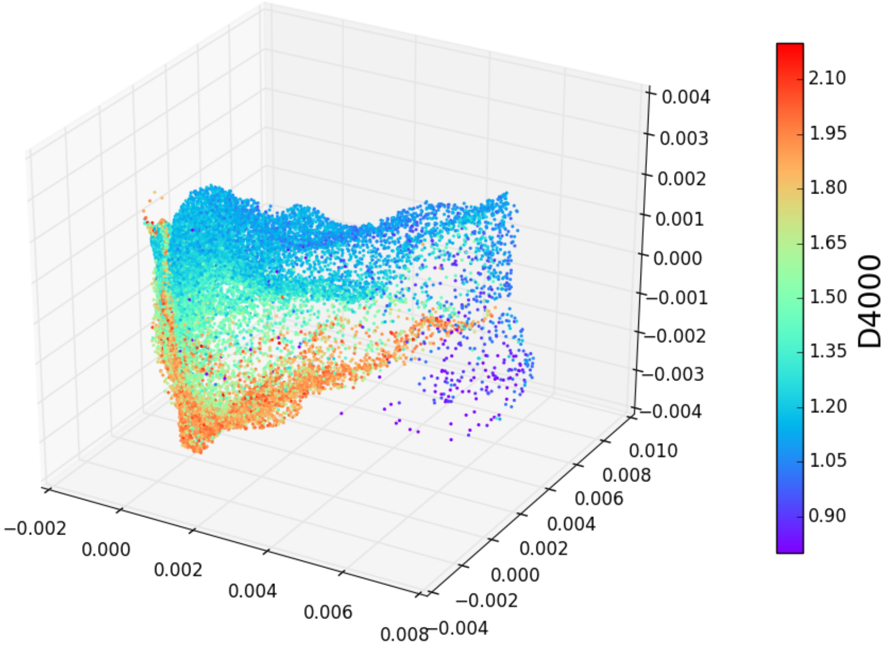


- data curated by Grace Telford,
- "noise removal" by Jake VanderPlas

Chosing the embedding dimension

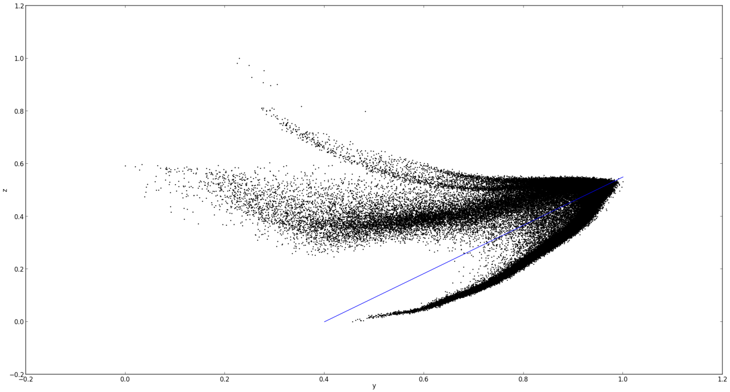


Embedding into 3 dimensions



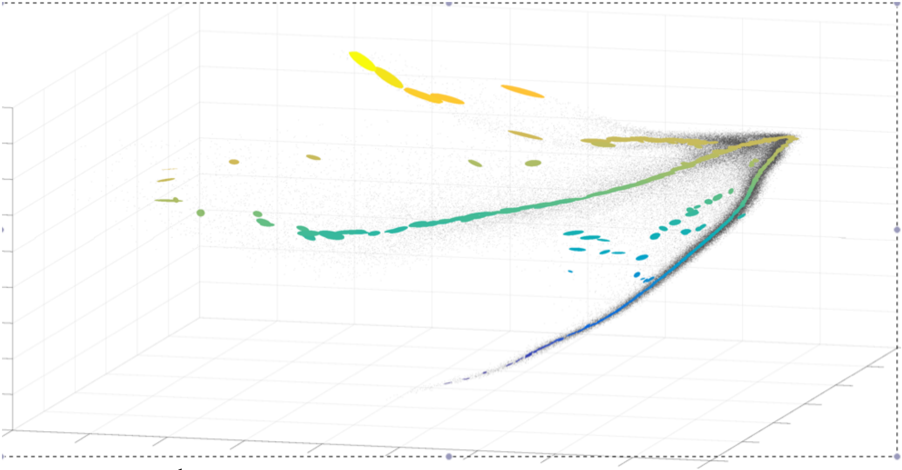
Same embedding...

- only high density regions
- another viewpoint



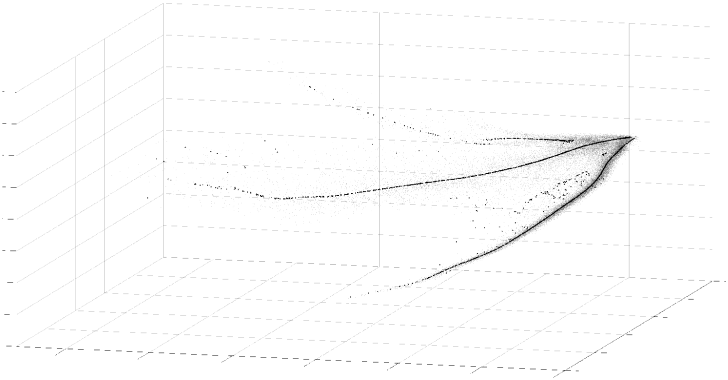
- how distorted is this embedding?

How distorted is this embedding?



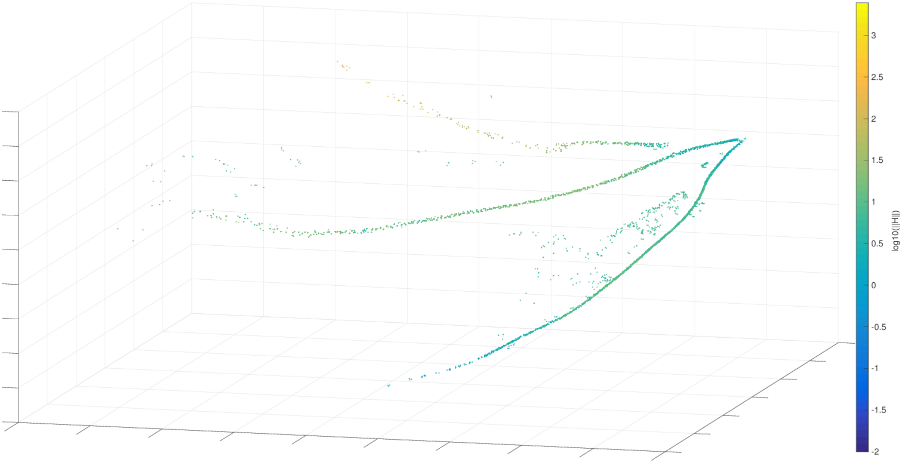
(ellipses represent G_i^{-1})

Riemannian Relaxation along principal curves



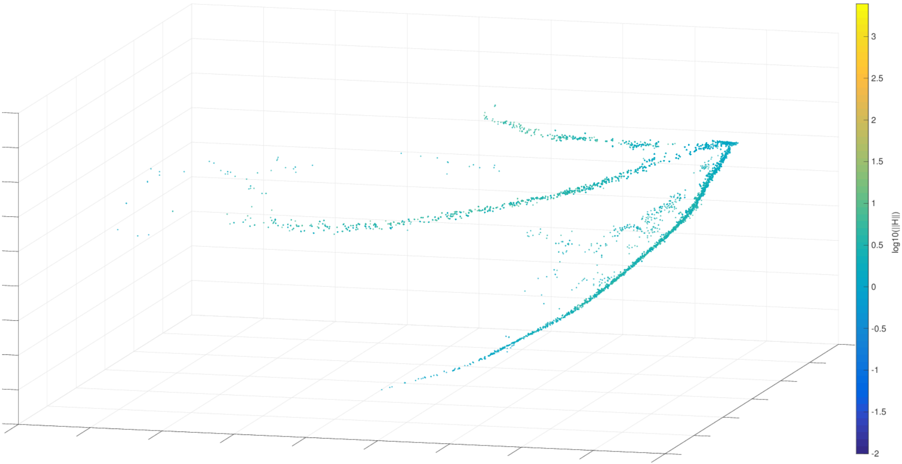
Find principal curves

Riemannian Relaxation along principal curves



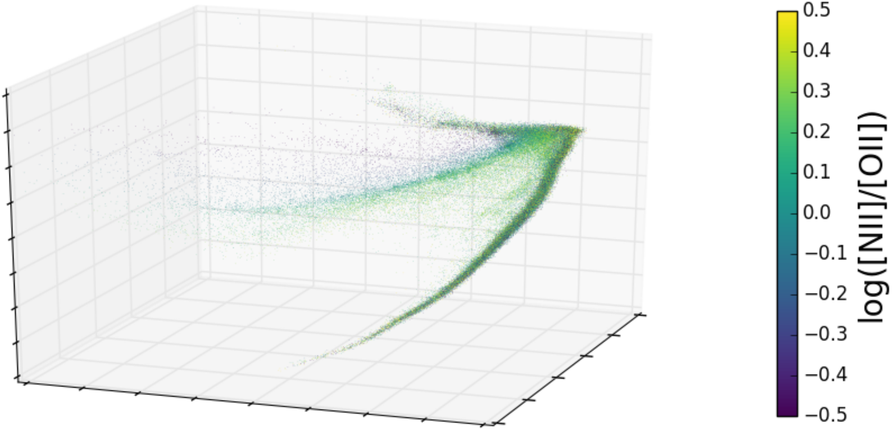
Points near principal curves, colored by $\log_{10}(G_i)$ (0 means no distortion)

Riemannian Relaxation along principal curves



Points near principal curves, colored by $\log_{10}(G_i)$, after Riemannian Relaxation
(0 means no distortion)

Riemannian Relaxation along principal curves



All data after Riemannian Relaxation

Manifold learning for sciences and engineering

Manifold learning is for toy data and toy problems

Manifold learning is for ~~toy data and toy problems~~

Manifold learning should be like PCA

- tractable
- “automatic”
- first step in data processing pipe-line

Manifold learning is for ~~toy data and toy problems~~

Manifold learning should be like PCA

- tractable
- "automatic"
- first step in data processing pipe-line

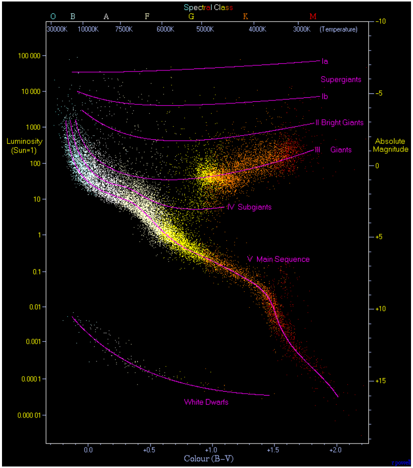
Metric Manifold learning

- Use any ML algorithm, estimate distortion by g
- and correct it (on demand)

`megaman`

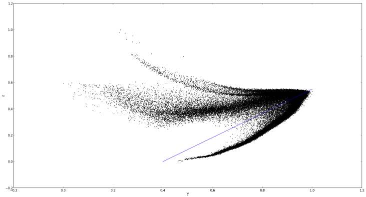
- tractable for millions of data
- (in progress) implementing quantitative validation procedure (topology preservation, choice of ϵ)
- future: port classification, regression, clustering to the manifold setting

Manifold Learning for engineering and the sciences



- scientific discovery by quantitative/statistical data analysis
- manifold learning as preprocessing for other tasks

Manifold Learning for engineering and the sciences



- scientific discovery by quantitative/statistical data analysis
- manifold learning as preprocessing for other tasks

Thank you