Geometrically faithful non-linear dimension reduction

Is Manifold Learning for toy data only?

Marina Meila
Dominique Perrault-Joncas
James McQueen   Jacob VanderPlas   Jerry Zhang   Grace Telford

University of Washington
mmp@stat.washington.edu

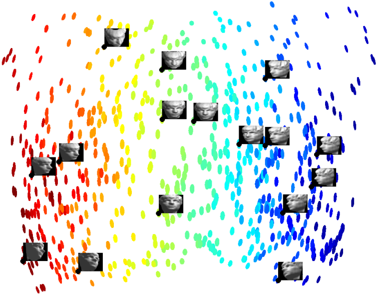
Statistics Symposium 4/22/2016

UCDAVIS
UNIVERSITY OF CALIFORNIA

Outline

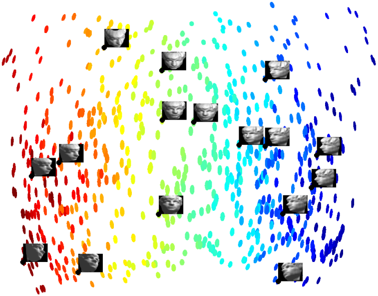
Outline

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

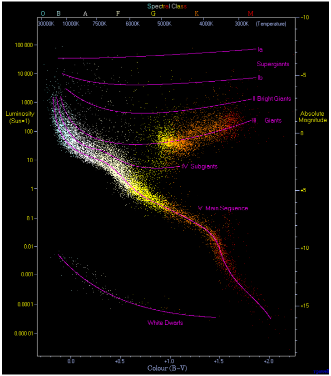
When to do (non-linear) dimension reduction



Why?

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

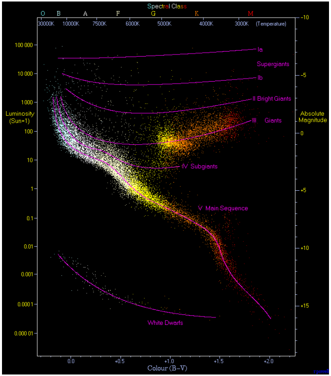
When to do (non-linear) dimension reduction



Why?

- To save space and computation
  - $n \times D$ data matrix $\rightarrow$ $n \times m$, $m \ll D$
- 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

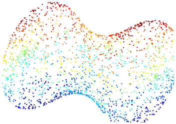


Why?

- To save space and computation
  - $n \times D$ data matrix $\to$ $n \times m$, $m \ll D$
- 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

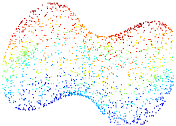
- **Input** Data $p_1, \ldots p_n$, embedding dimension m, neighborhood scale parameter ϵ



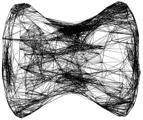
$p_1, \ldots p_n \subset \mathbb{R}^D$

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 $||p - p'||^2 \leq \epsilon$

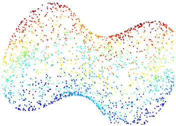


$p_1, \ldots p_n \subset \mathbb{R}^D$

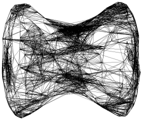


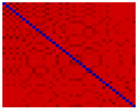
How? Brief intro to manifold learning algorithms

- **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 $\phi(p_{1:n})$



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





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 $||p - p'||^2 \leq \epsilon$
- Construct a $n \times n$ matrix its leading eigenvectors are the coordinates $\phi(p_{1:n})$

LAPLACIAN EIGENMAPS [Belkin & Nyogi 02]

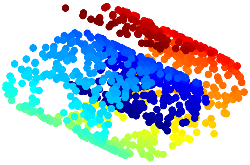
- Construct similarity matrix

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

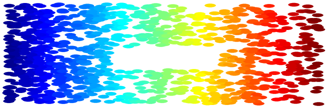
- Construct Laplacian matrix $L = I - T^{-1}S$ with $T = \mathrm{diag}(S\mathbf{1})$
- Calculate $\psi^{1\ldots m}$ = eigenvectors of L (smallest eigenvalues)
- coordinates of $p \in \mathcal{D}$ are $(\psi^1(p), \ldots \psi^m(p))$

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

points in $D \geq 3$ dimensions

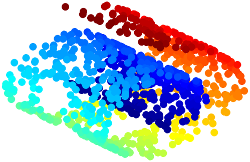


same points reparametrized in 2D



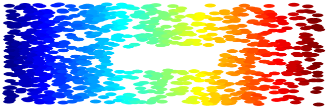
A toy example (the "Swiss Roll" with a hole)

points in $D \geq 3$ dimensions



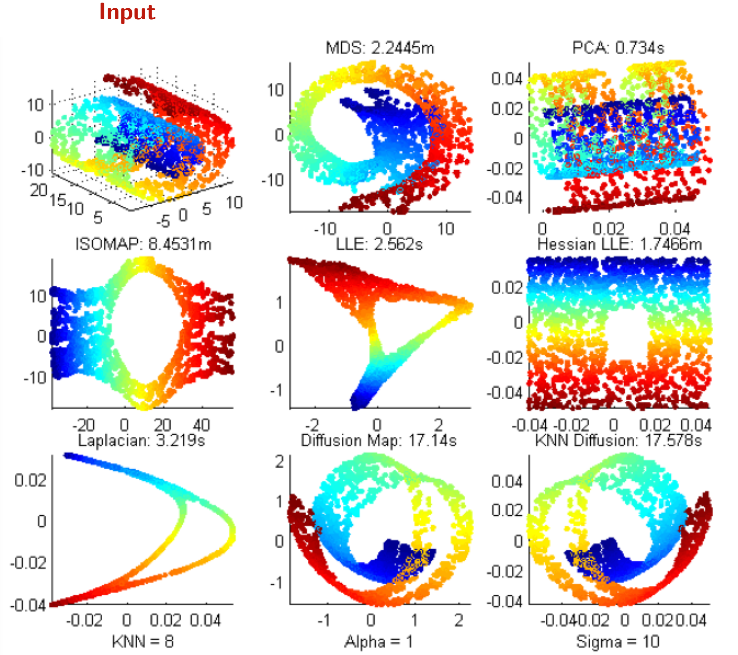
Input

same points reparametrized in 2D

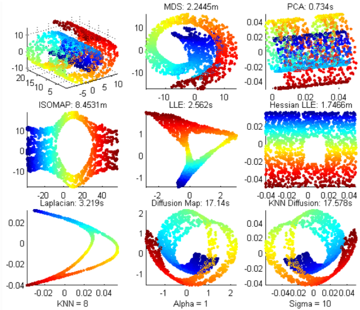


Desired output

Embedding in 2 dimensions by different manifold learning algorithms



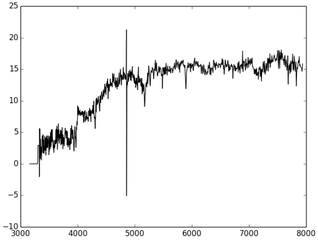
How to evaluate the results objectively?



- 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

Preserving topology vs. preserving (intrinsic) geometry

- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$

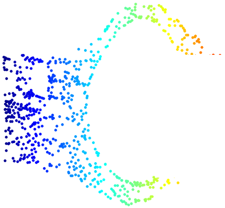
- Mapping $\mathcal{M} \longrightarrow \phi(\mathcal{M})$ is diffeomorphism
  preserves topology
  often satisfied by embedding algorithms
- Mapping ϕ preserves
  - distances along curves in $\mathcal{M}$
  - angles between curves in $\mathcal{M}$
  - areas, volumes

Preserving topology vs. preserving (intrinsic) geometry

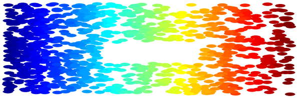
- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$
- Mapping $\mathcal{M} \longrightarrow \phi(\mathcal{M})$ is diffeomorphism
  preserves topology
  often satisfied by embedding algorithms
- Mapping ϕ preserves
  - distances along curves in $\mathcal{M}$
  - angles 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.
- 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
- Isomap recovers (only) flat manifolds isometrically
- algorithm based on Nash's theorem (isometric embedding for very low d) [Verma 2011]

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 [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 [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

- $\mathcal{M}$ = (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**

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

- Volume element on manifold

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

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 $\mathcal{D} = \{p_1, \ldots\ p_n\}$
    sampled from manifold $\mathcal{M} \subset \mathbb{R}^D$
  - embedding $\{\, \phi(p),\ p \in \mathcal{D}\,\}$
    by e.g LLE, Isomap, LE, ...
- Estimate $g_p \in \mathbb{R}^{m \times m}$ the Riemannian metric for $p \in \mathcal{D}$
  in the embedding coordinates ϕ
- The embedding (ϕ, g) will preserve the geometry of the original data manifold

Relation between g and Δ

- Δ = Laplace-Beltrami operator on $\mathcal{M}$
  - $\Delta = \mathrm{div} \cdot \mathrm{grad}$
  - on C^2, $\Delta f = \sum_j \frac{\partial^2 f}{\partial x_j^2}$
  - 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)

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

where $[g^{lk}] = g^{-1}$

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)

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

Consistency of the Riemannian metric estimator

Proposition

- If the embedding $\phi : \mathcal{M} \to \phi(\mathcal{M})$ is
  A diffeomorphic
  B consistent $\phi(\mathcal{D}_n) \overset{n\to\infty}{\longrightarrow} \phi(\mathcal{M})$
  C Laplacian consistent $L_n\phi(\mathcal{D}_n) \overset{n\to\infty}{\longrightarrow} \Delta\phi(\mathcal{M})$

  then the dual Riemannian metric estimator h is consistent

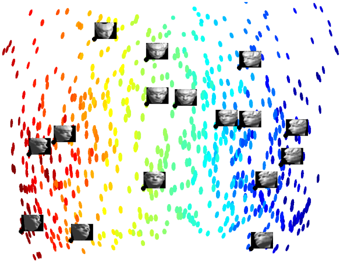
$$(\phi(\mathcal{D}_n), h_n) \overset{n\to\infty}{\longrightarrow} (\phi(\mathcal{M}), h)$$

- Laplacian Eigenmaps and Diffusion Map satisfy **A, B** if $\mathcal{M}$ compact

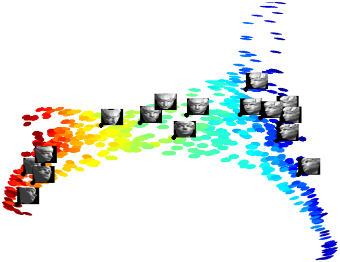
g for Sculpture Faces

- $n = 698$ with 64×64 gray images of faces
  - head moves up/down and right/left

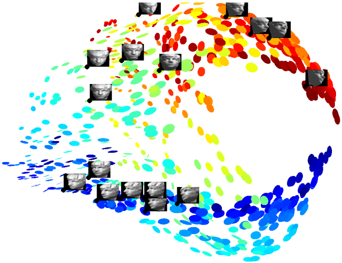
LTSA Algoritm



Isomap



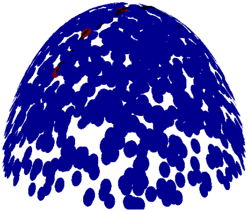
LTSA



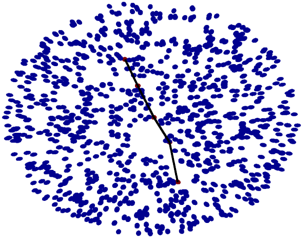
Laplacian Eigenmaps

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

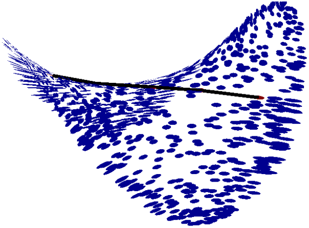
- Geodesic distance $=$ shortest path on $\mathcal{M}$
- should be invariant to coordinate changes



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 $m = 2$	1.66	1.75	1.63	3.7%
LTSA $m = 2$	0.07	0.08	1.65	4.8%
LE $m = 3$	0.08	0.08	1.62	3.1%

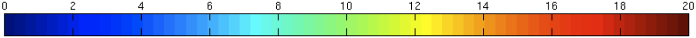
Calculating Areas/Volumes in the manifold

(Results for Hourglass data)

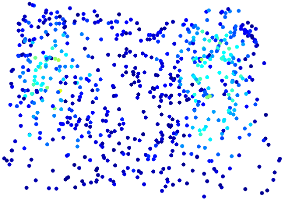
	true area = 0.84		
Embedding	Naive	Metric	Rel. err.
Original data	0.85 (0.03)	0.93 (0.03)	11.0%
Isomap	2.7	0.93 (0.03)	11.0%
LTSA	1e-03 (5e-5)	0.93 (0.03)	11.0%
LE	1e-05 (4e-4)	0.82 (0.03)	2.6%

Semisupervised learning with Gaussian Processes on Manifolds

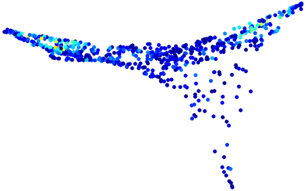
Sculpture Faces: Predicting Head Rotation



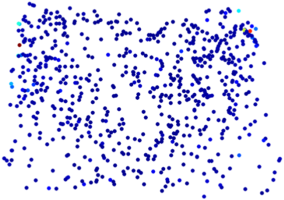
Absolute Error (AE) as percentage of range



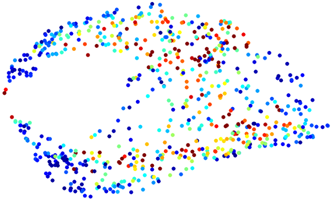
Isomap (Total AE = 3078)



LTSA (Total AE = 3020)



Metric (Total AE = 660)



Laplacian Eigenmaps (Total AE = 3078)

Self-consistent method of chosing ϵ

- Every manifold learning algorithm starts with a neighborhood graph
- Parameter $\sqrt{\epsilon}$
  - is neighborhood radius
  - and/or kernel banwidth

Self-consistent method of chosing ϵ

- Every manifold learning algorithm starts with a neighborhood graph
- Parameter $\sqrt{\epsilon}$
  - is neighborhood radius
  - and/or kernel banwidth

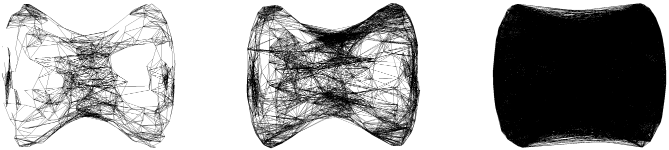
- For example, we use the kernel
  $K(p, p') = e^{-\frac{||p-p'||^2}{\epsilon}}$ if $||p - p'||^2 \leq \epsilon$ and 0 otherwise

Self-consistent method of chosing ϵ

- Every manifold learning algorithm starts with a neighborhood graph
- Parameter $\sqrt{\epsilon}$
  - is neighborhood radius
  - and/or kernel banwidth

- For example, we use the kernel
  $K(p,p') = e^{-\frac{||p-p'||^2}{\epsilon}}$ if $||p-p'||^2 \le \epsilon$ and 0 otherwise

- Problem: how to choose ϵ?



Existing work

- Theoretical (asymptotic) result $\sqrt{\epsilon} \propto n^{-\frac{1}{d+6}}$ [Singer06]
- Cross-validation
  - assumes a supervised task given
- heuristic for K-nearest neighbor graph [Chen&Buja09]
  - depends on embedding method used
  - K-nearest neighbor graph has different convergence properties than ϵ neighborhood
- Visual inspection

Our idea

- L_ϵ estimate of Laplace-Beltrami operator
  - contains the intrinsic geometry
  - ... and depends on ϵ by construction

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

Our idea

- L_ϵ estimate of Laplace-Beltrami operator
  - contains the intrinsic geometry
  - ... and depends on ϵ by construction
- Idea: choose ϵ so that geometry encoded by L_ϵ is closest to data geometry
- Idea, formalized
  - make $g_\epsilon(p)$ close to I_D the identity matrix, for each $p \in \mathcal{D}$
- This is a completely unsupervised method
  - L_ϵ is estimated independently of any embedding or task

Semisupervised learning benchmarks [Chapelle&al 08]

Multiclass classification problems

Classification error (%)

	Method		
Dataset	**CV**	[Chen&Buja]	**Ours**
Digit1	3.32	2.16	2.11
USPS	5.18	4.83	3.89
COIL	7.02	8.03	8.81
g241c	13.31	23.93	12.77
g241d	8.67	18.39	8.76
	superv.	fully unsupervised	

Results: Intrinsic Dimension Estimation

- Method of [Chen&al 11]
  - do local SVD for a range of neighborhood radii
  - choose a an appropriate radius ϵ
  - dimension = largest eigengap at radius ϵ
- used our self-concordant method to find ϵ
- Experiments: artificial 2D manifolds with noise

Consistency of the Riemannian metric estimator

Proposition

- If the embedding $\phi : \mathcal{M} \to \phi(\mathcal{M})$ is
  - **A** diffeomorphic
  - **B** consistent $\phi(\mathcal{D}_n) \stackrel{n\to\infty}{\longrightarrow} \phi(\mathcal{M})$
  - **C** Laplacian consistent $L_n\phi(\mathcal{D}_n) \stackrel{n\to\infty}{\longrightarrow} \Delta\phi(\mathcal{M})$

  then the dual Riemannian metric estimator h is consistent

$$(\phi(\mathcal{D}_n), h_n) \stackrel{n\to\infty}{\longrightarrow} (\phi(\mathcal{M}), h)$$

- Laplacian Eigenmaps and Diffusion Map satisfy **A, B** if $\mathcal{M}$ compact

Outline

Is Manifold Learning (ML) scalable?

- **Fact** ML is data intensive
  - large amounts of data needed to reach accurate estimation of a manifold (e.g at least 10^{3-4}) []
- **Rumor** “it is widely believed that ML is also computationally intensive”
  - in particular, it scales poorly with the sample size n

Is Manifold Learning (ML) scalable?

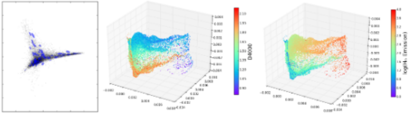
- **Fact** ML is data intensive
  - large amounts of data needed to reach accurate estimation of a manifold (e.g at least 10^{3-4}) []
- **Rumor** "it is widely believed that ML is also computationally intensive"
  - in particular, it scales poorly with the sample size n
- **My premise: ML is no more expensive that PCA**
  - i.e. non-linear dimension reduction is just as tractable as linear dimension reduction

Is Manifold Learning (ML) scalable?

- **Fact** ML is data intensive
  - large amounts of data needed to reach accurate estimation of a manifold (e.g at least 10^{3-4}) []
- **Rumor** "it is widely believed that ML is also computationally intensive"
  - in particular, it scales poorly with the sample size n
- **My premise: ML is no more expensive that PCA**
  - i.e. non-linear dimension reduction is just as tractable as linear dimension reduction
- **So?...**

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

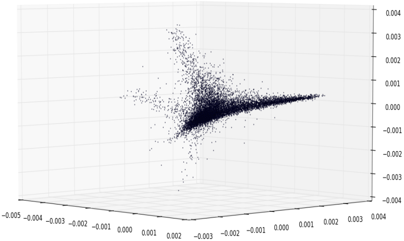
Jake VanderPlas

Jerry Zhang

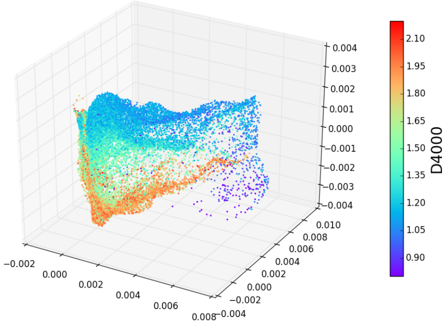
Grace Telford

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])

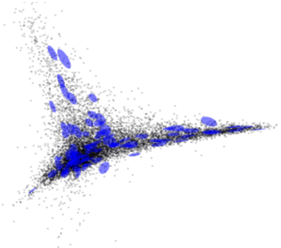


Main sample of galaxy spectra from the Sloan Digital Sky Survey (675,000 spectra originally in 3750 dimensions).

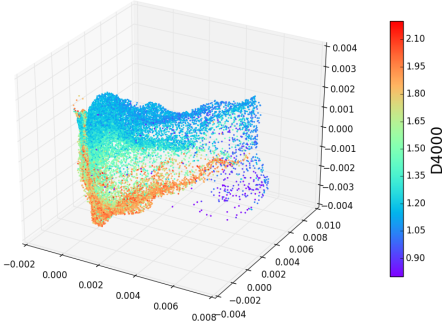
preprocessed by Jake VanderPlas, figure by Grace Telford

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])



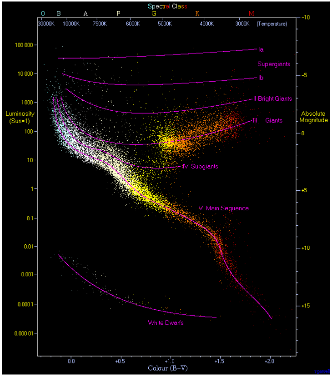
Main sample of galaxy spectra from the Sloan Digital Sky Survey (675,000 spectra originally in 3750 dimensions).

preprocessed by Jake VanderPlas, figure by Grace Telford

Significance

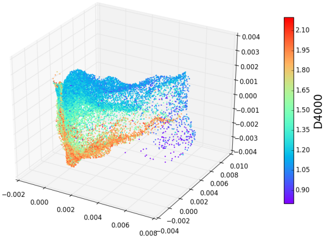
- Augmentation of manifold learning algorithms
  - For a given algorithm, all geometrical quantities are preserved simultaneously, by recovering g = geometry preserving embedding
  - We can obtain geometry preserving embeddings with any reasonable algorithm
- Unification of algorithms
  - Now, all "reasonable" algorithms/embeddings are asymptotically equivalent from the geometry point of view
  - We can focus on comparing algorithms based on other criteria speed, rate of convergence, numerical stability
  - g offers a way to compare the algorithms' outputs
    - Each algorithm has own ϕ
    - Hence outputs of different algorithms are incomparable
    - But (ϕ^A, g^A), (ϕ^B, g^B) should be comparable because they aim to represent intrinsic/geometric quantities

Manifold learning for sciences and engineering



- scientific discovery by quantitative/statistical analysis
- manifold learning as pre-processing for other tasks

Manifold learning for sciences and engineering



- scientific discovery by quantitative/statistical analysis
- manifold learning as pre-processing for other tasks

Thank you