Manifold Learning in the Age of Big Data

Marina Meilă

University of Washington
mmp@stat.washington.edu

BDC2018 Workshop

Outline

Unsupervised Learning with Big Data

Manifold learning G2

- Basics of manifold learning algorithms
- Metric Manifold Learning
- Estimating the kernel bandwidth

Scalable manifold learning

- megaman

Finding filaments in high dimensions

Supervised, Unsupervised, Reinforcement Learning

- We are witnessing an AI/ML revolution
  - this is led by Supervised and Reinforcement Learning
  - i.e. Prediction and Acting
- Unsupervised learning (clustering analysis, dimension reduction, explanatory models)
  - is in a much more primitive state of development
  - it is harder conceptually: defining the objective is part of the problem
  - but everybody does it [in the sciences]
  - because exploration, explanation, understanding, uncovering the structure of the data are necessary
    in the language of the discipline
  - is the next big data challenge?

Unsupervised learning at scale and automatically validated

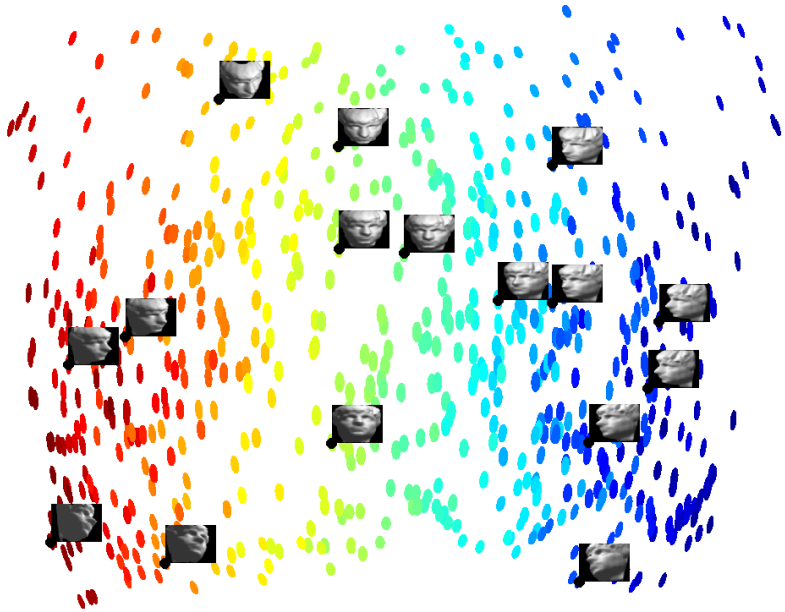
- Topics: Geometry and combinatorics
  - Non-linear dimension reduction
  - Topological data analysis
  - Graphs, rankings, clustering
- Mathematics/theory/theorems/models
  - validation/checking/guarantees
  - beyond discovering patterns
- Algorithms and computation

- Demands from practical problems stimulate good research

Outline

Outline

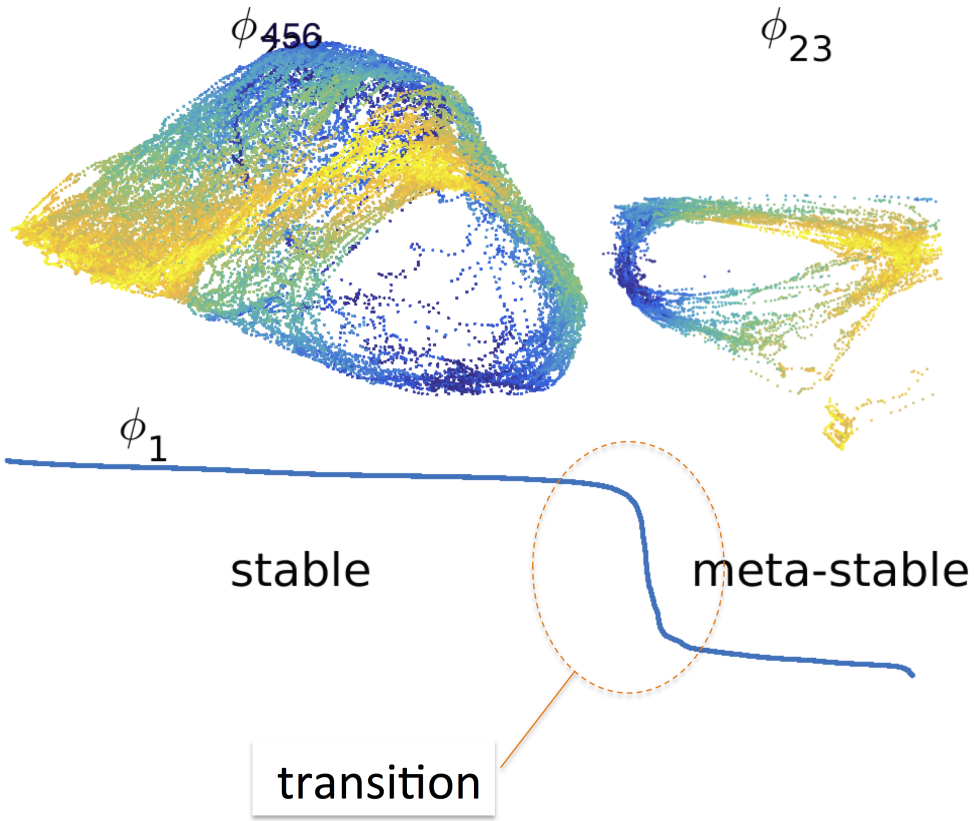
When to do (non-linear) dimension reduction



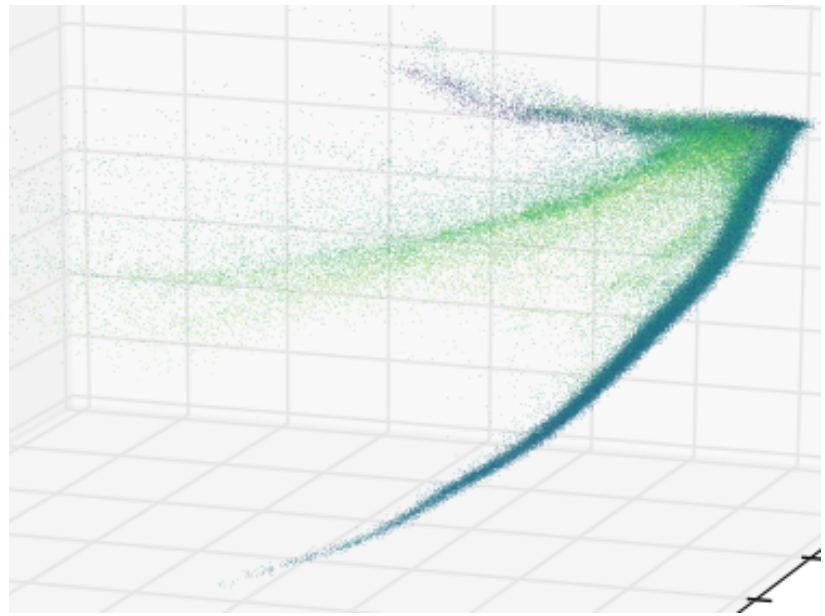
- high-dimensional data $p \in \mathbb{R}^D$, $D = 64 \times 64$
- can be described by a small number d of continuous parameters
- Usually, large sample size n

When to do (non-linear) dimension reduction

aspirin MD simulation



SDSS galaxy spectra

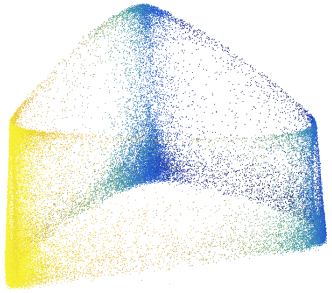


Why?

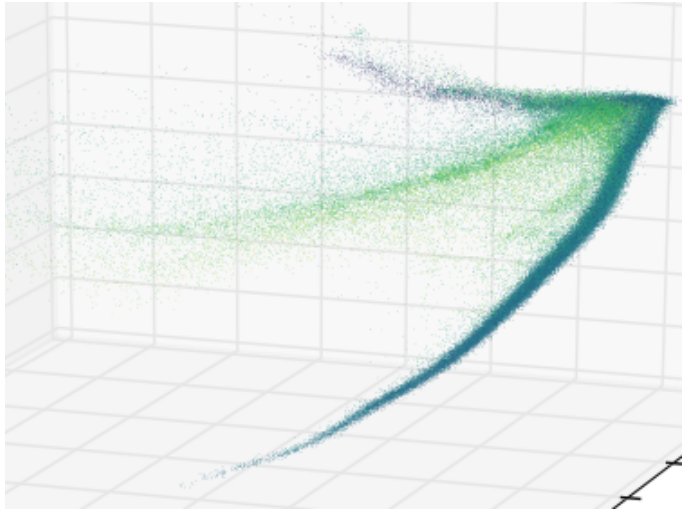
- To save space and computation
  - $n \times D$ data matrix $\rightarrow$ $n \times s$, $s \ll D$
- To use it afterwards in (prediction) tasks
- To understand the data better
  - preserve large scale features, suppress fine scale features

ML poses special computational challenges

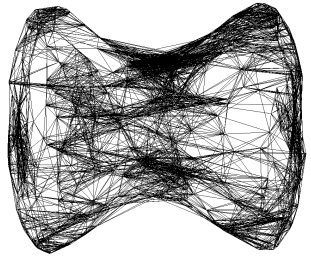
ethanol, $n = 50,000$, $D = 21$

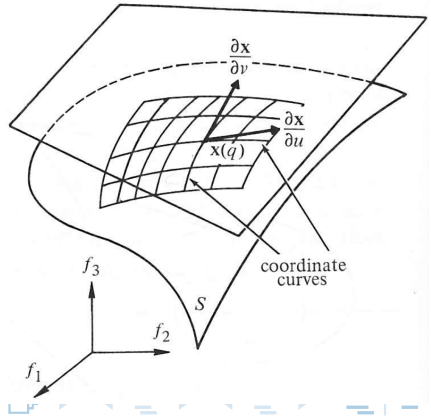


SDDS spectra, $n = 0.7M$, $D = 3750$



- structure of computation not regular
  - dictated by a random geometric graph
  - randomness: e.g. # neighbors of a data point
  - affects storage, parallelization, run time (by unknown condition numbers)
- intrinsic dimension d controls statistical and almost all numerical properties of ML algorithm
  - data dimension D arbitrarily large as long as d small
  - d must be guessed/estimated
- coordinate system is local (e.g protein folding)

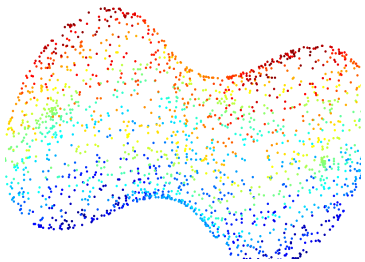




- data partitioning is open problem
- not easily parallelizable ?
- problem of finding neighbors efficiently

Brief intro to manifold learning algorithms

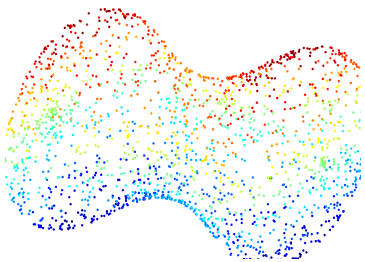
- **Input** Data $p_1, \ldots p_n$, embedding dimension m, neighborhood scale parameter ϵ



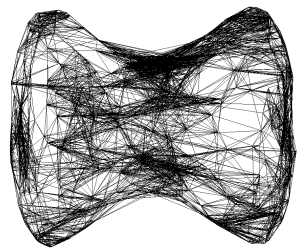
$p_1, \ldots p_n \subset \mathbb{R}^D$

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

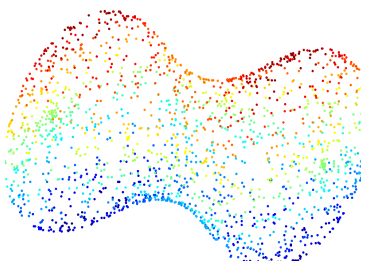


$p_1, \ldots p_n \subset \mathbb{R}^D$

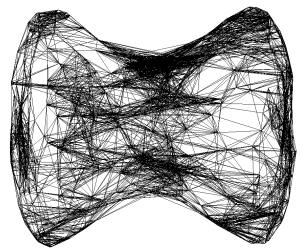


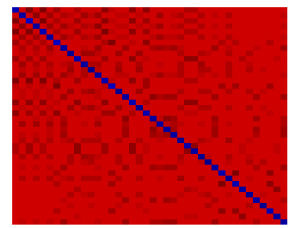
Brief intro to manifold learning algorithms

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



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





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/Diffusion Maps [Belkin,Niyogi 02,Nadler et al 05]

- Construct similarity matrix

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

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

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

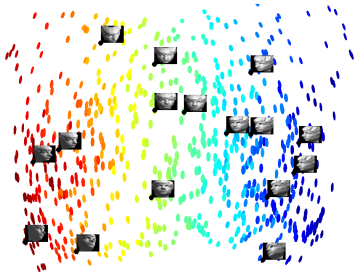
ISOMAP [Tennenbaum, deSilva & Langford 00]

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

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

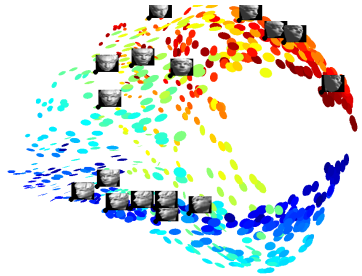
- use M and Multi-Dimensional Scaling (MDS) to obtain m dimensional coordinates for $p \in \mathcal{D}$

Isomap vs. Diffusion Maps



Isomap

- Preserves geodesic distances
  - but only sometimes []
- Computes all-pairs shortest paths $\mathcal{O}(n^3)$
- Stores/processes dense matrix

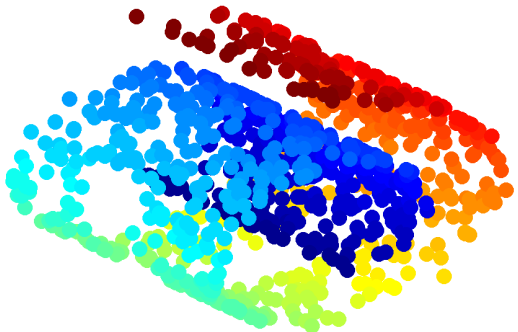


DiffusionMap

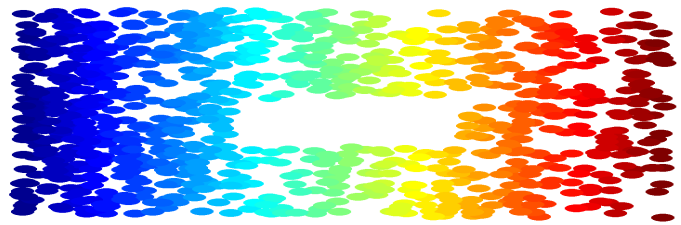
- Distorts geodesic distances
- Computes only distances to nearest neighbors $\mathcal{O}(n^{1+\epsilon})$
- Stores/processes sparse matrix

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

points in $D \geq 3$ dimensions

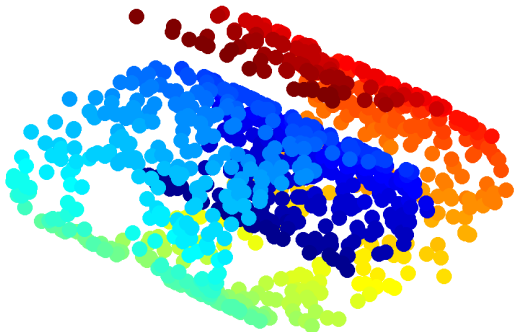


same points reparametrized in 2D



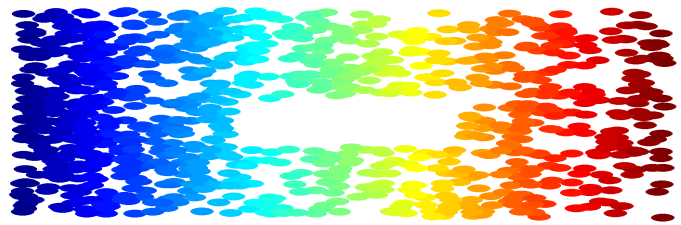
A toy example (the "Swiss Roll" with a hole)

points in $D \geq 3$ dimensions



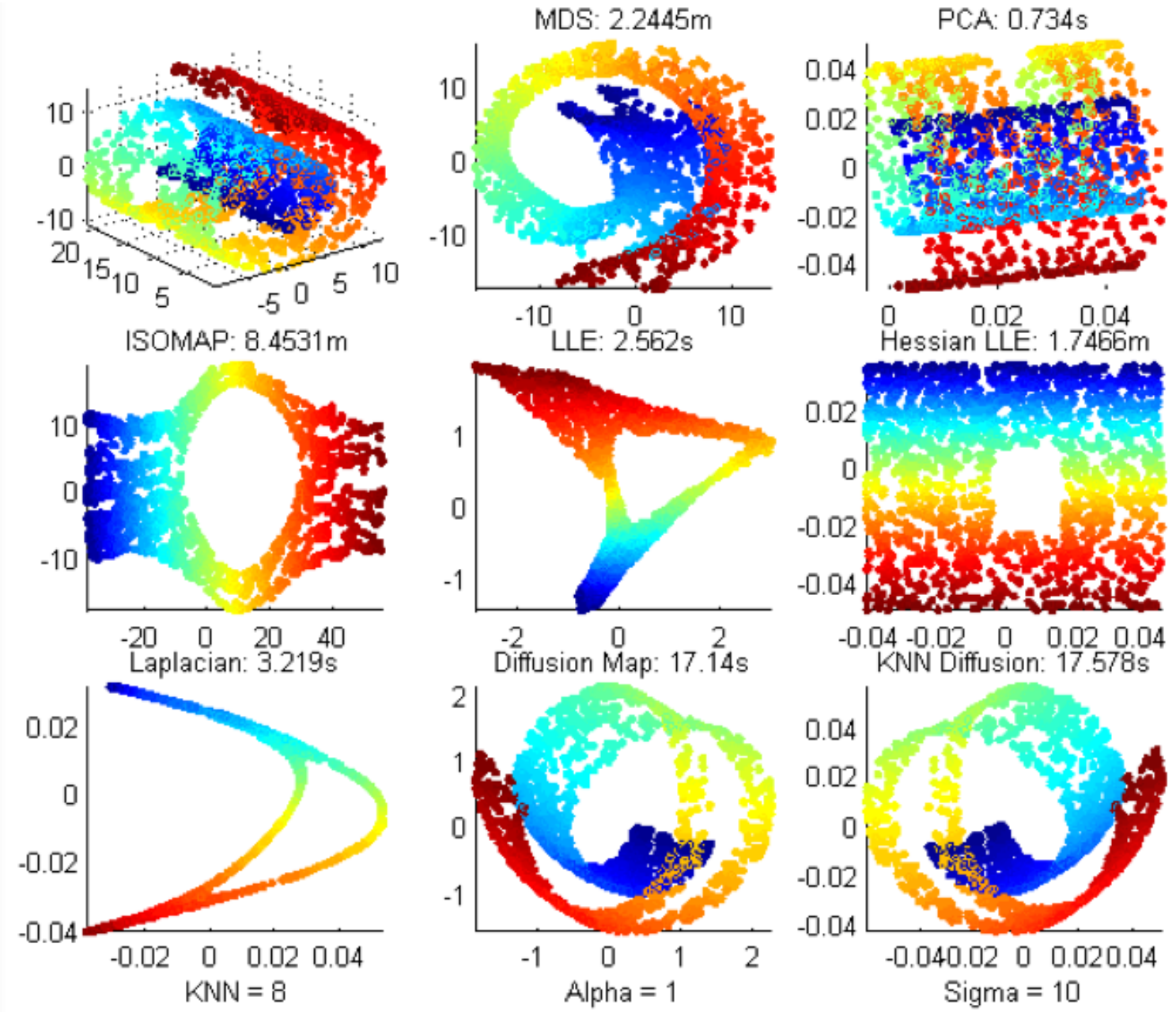
Input

same points reparametrized in 2D



Desired output

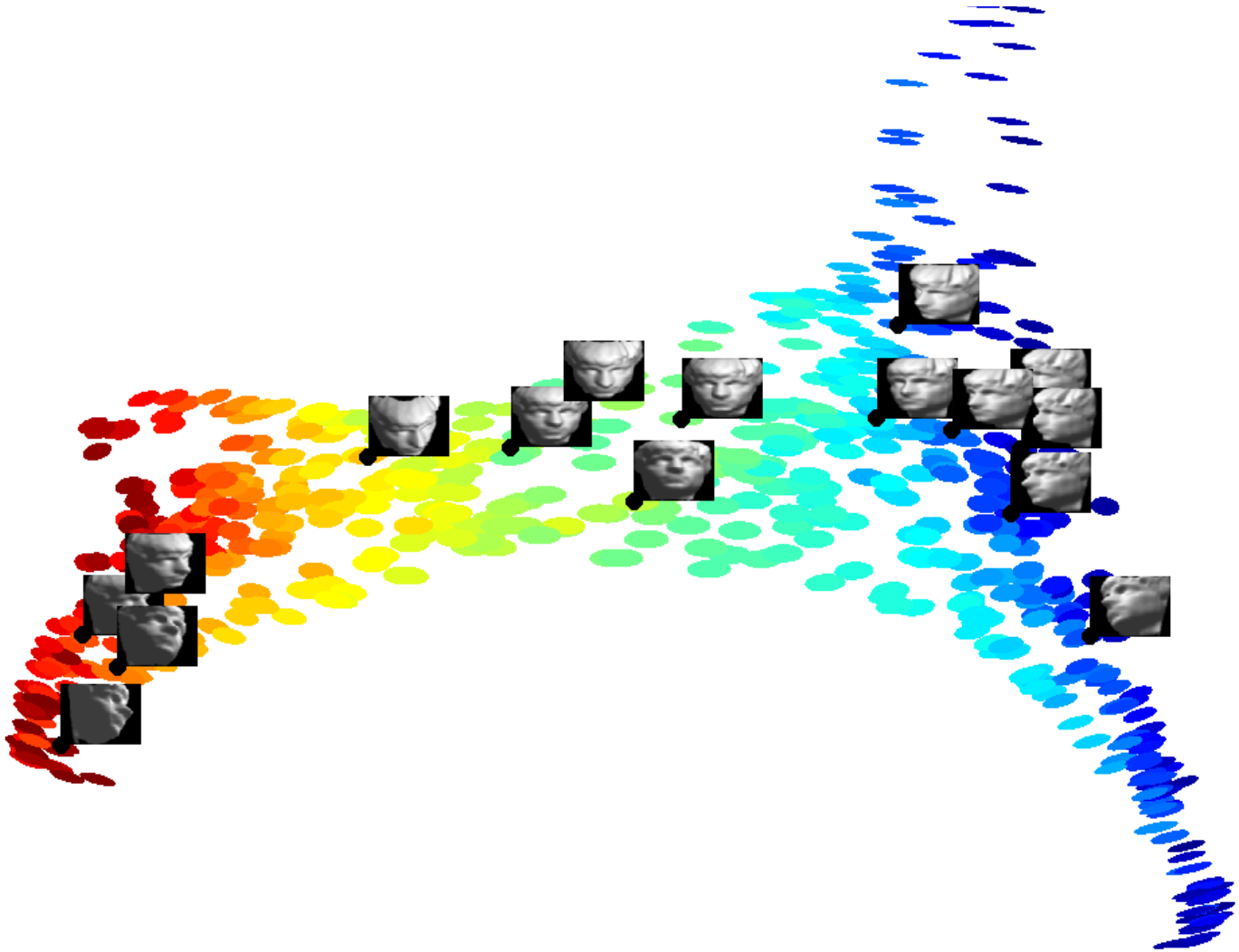
Embedding in 2 dimensions by different manifold learning algorithms

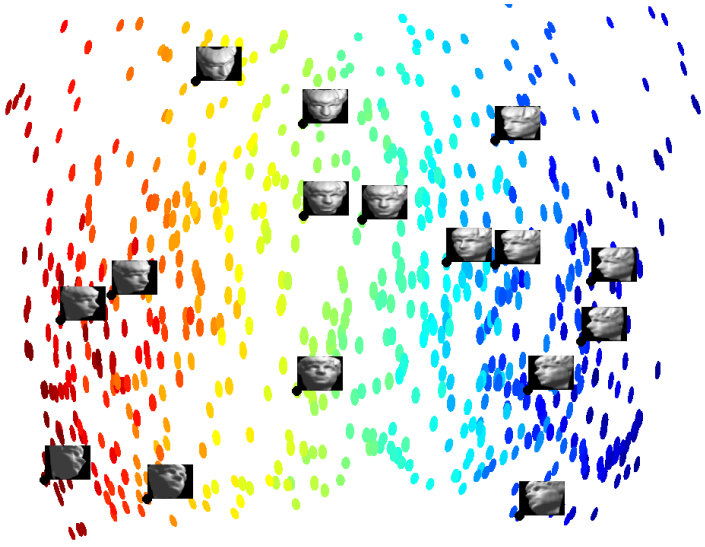


Outline

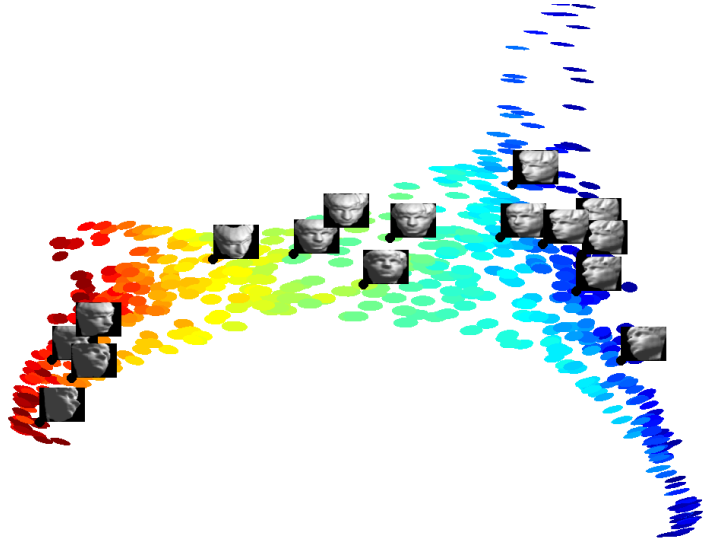
g for Sculpture Faces

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

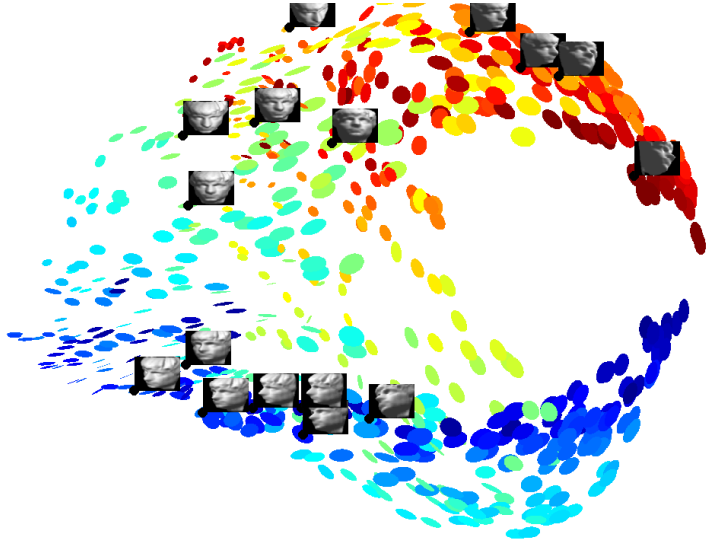




Isomap



LTSA

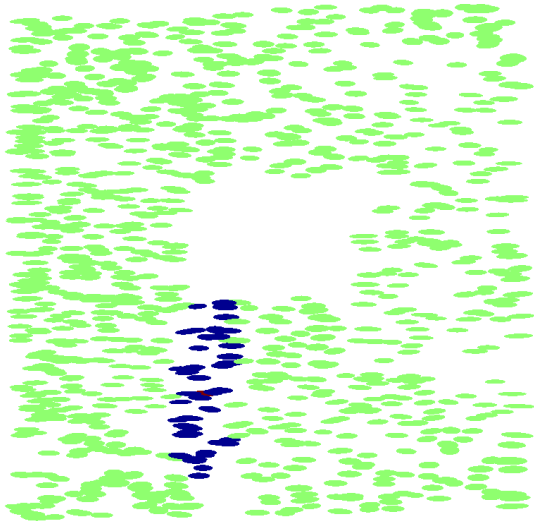


Laplacian Eigenmaps

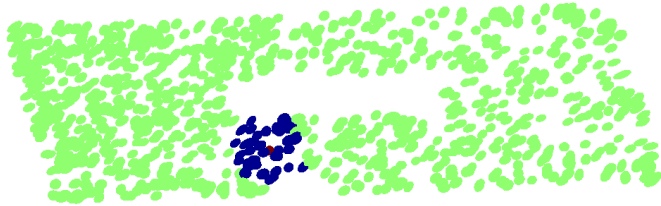
Metric ML unifies embedding algorithms

- Distortions can now be corrected
  - locally: Locally Normalized Visualization
  - globally: Riemannian Relaxation
  - implicitly: by integrating with the right lenght or volume element
- Hence, all embedding algorithms preserve manifold geometry

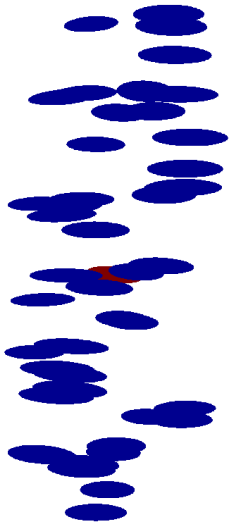
Locally Normalized Visualization

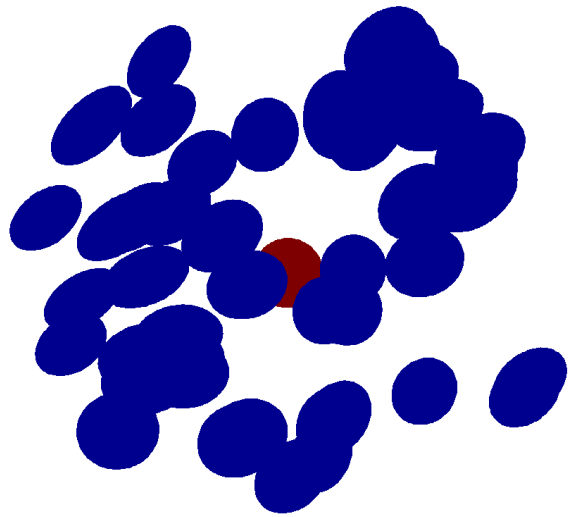


local neighborhood, unnormailzed

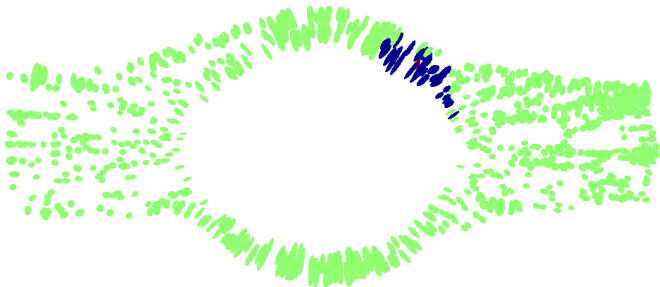


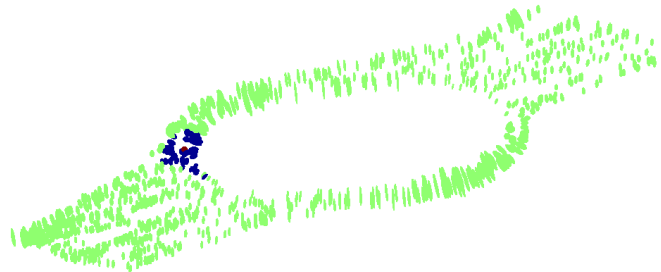
local neighborhood, Locally Normailzed





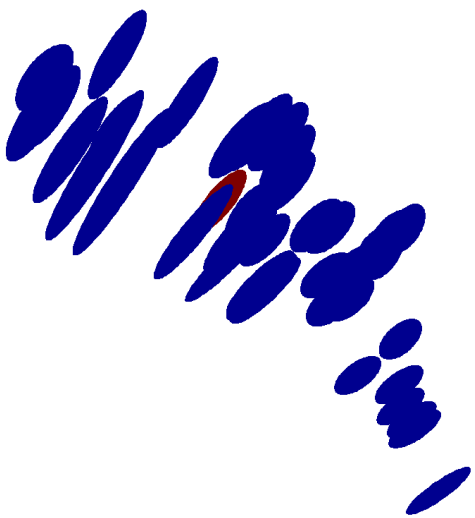
Locally Normalized Visualization

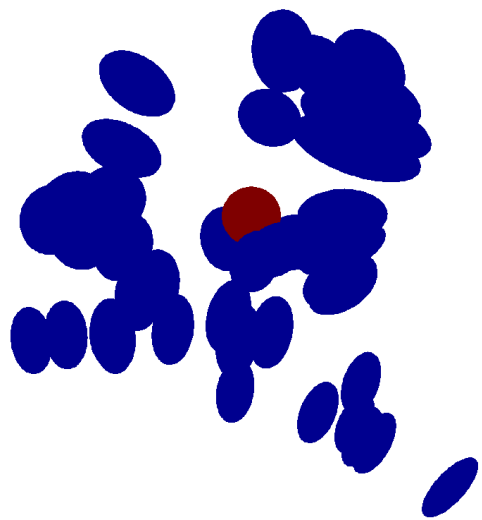




local neighborhood, unnormailzed

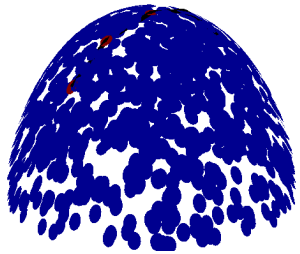
local neighborhood, Locally Normailzed(scaling 1.00)



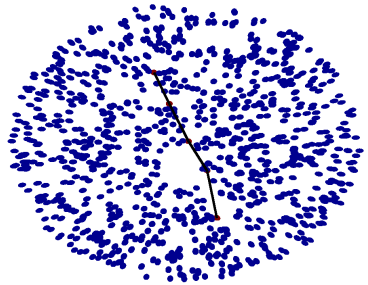


Calculating distances in the manifold $\mathcal{M}$

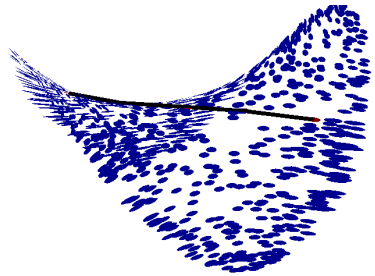
Original



Isomap



Laplacian Eigenmaps

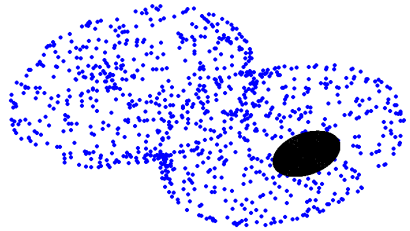


true distance $d = 1.57$

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 $s = 2$	0.08	0.08	1.62	3.1%

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

Calculating Areas/Volumes in the manifold



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%

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

Outline

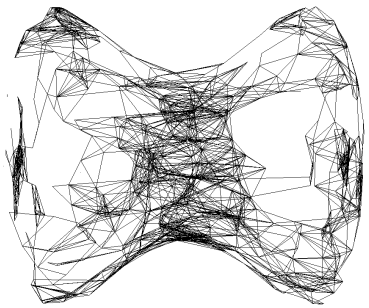
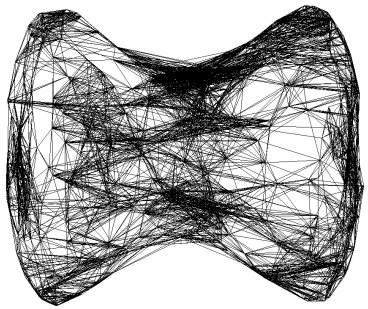
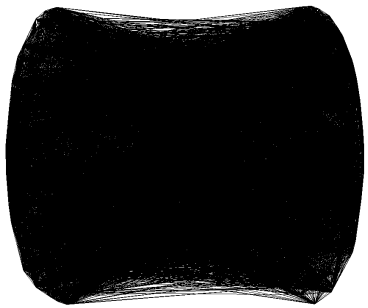
Self-consistent method of chosing ϵ

- 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

- 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

For given ϵ and data point p

- Project neighbors of p onto tangent subspace
  - this "embedding" is approximately isometric to original data
- Calculate Laplacian $L(\epsilon)$) and estimate distortion $H_{\epsilon,p}$ at p
  - $H_{\epsilon,p}$ must be $\approx I_d$ identity matrix

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

  ▸ Jupyter Notebook

- Completely unsupervised

The distortion measure

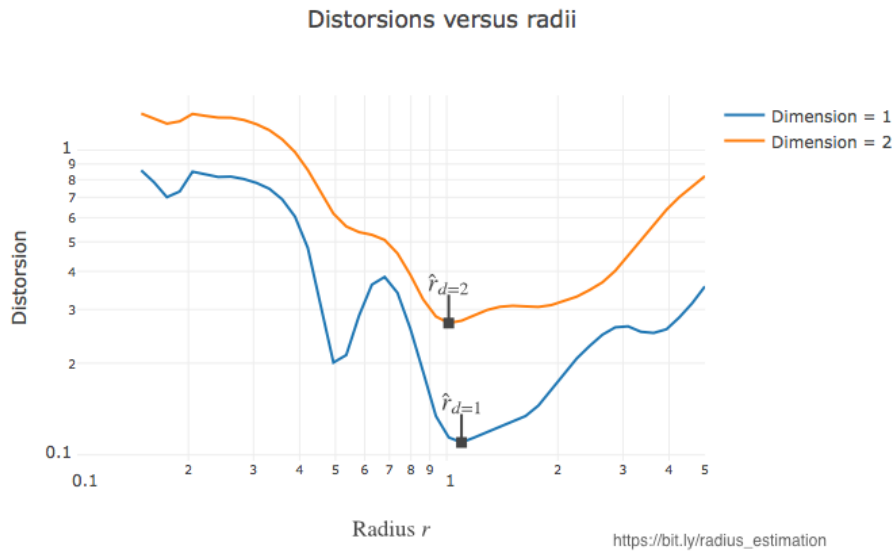
Input: data set $\mathcal{D}$, dimension $d' \leq d$, scale ϵ

1. Estimate Laplacian L_ϵ
2. For each p
   - Project data on tangent plane at p by local SVD with dimension d'
   - Estimate $H_{\epsilon,p} \in \mathbb{R}^{d' \times d'}$ for this projection
3. compute quadratic distortion $D(\epsilon) = \sum_{p \in \mathcal{D}} w_p ||H_{\epsilon,p} - I_d||_2^2$

Output $D(\epsilon)$

- by mapping into d' dimensions we curse of dimensionality, reduce vari
- $d' \leq d$
- h used instead of g – more robust

- Select $\epsilon^* = \operatorname{argmin}_\epsilon D(\epsilon)$
  - minimum found by 0-th order optimization



- Extension to embedding in $m > d$ dimensions

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	

Outline

Outline

Scaling: Statistical viewpoint

- Manifold estimation is non-parametric
  - model becomes more complex when more data available
  - provided ϵ kernel bandwidth decreases slowly with n
- Rates of convergence for manifold estimation as $n \longrightarrow \infty$
  - 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]
- Compare with rate of convergence for parametric estimation $n^{-\frac{1}{2}}$

- Hence,
  - for non-parametric models, accuracy improves very slowly with n
  - estimating $\mathcal{M}$ and g accurately requires big data

Scaling: Computational viewpoint

LaplacianEigenmap revisited

1. Construct similarity matrix

   $S = [S_{pp'}]_{p,p' \in \mathcal{D}}$ 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
   $(\psi^1(p), \ldots \psi^m(p))$

Scaling: Computational viewpoint

Laplacian Eigenmaps revisited

1. Construct similarity matrix

   $S = [S_{pp'}]_{p,p' \in \mathcal{D}}$ 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

ML with bandwidth [and dimension] estimation

Input ϵ_{max}, ϵ_{min}, n' subsample size, d' parameter

1. Find $3\epsilon_{max}$ neighborhoods
2. For ϵ in $[\epsilon_{min}, \epsilon_{max}]$
   - 2.1 Construct Laplacian $L(\epsilon)$ $\mathcal{O}(n' \times \#\text{ neighbors}^2 d')$
     - (graph given)
     - only n' rows needed
   - 2.2 Local PCA at n' points
   - 2.3 Calculate distortion $D(\epsilon)$ at n' points
   - 2.4 Optionally: Estimate dimension for this ϵ
3. Choose ϵ, recalculate L, [fix d]
4. Eigendecomposition SVD(L, m) with $m \leq 2d$

Manifold Learning and Clustering with millions of points

https://www.github.com/megaman

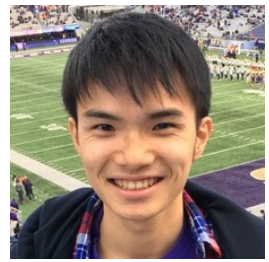
James McQueen

Jake VanderPlas

Jerry Zhang

Grace Telford

Yu-chia Chen

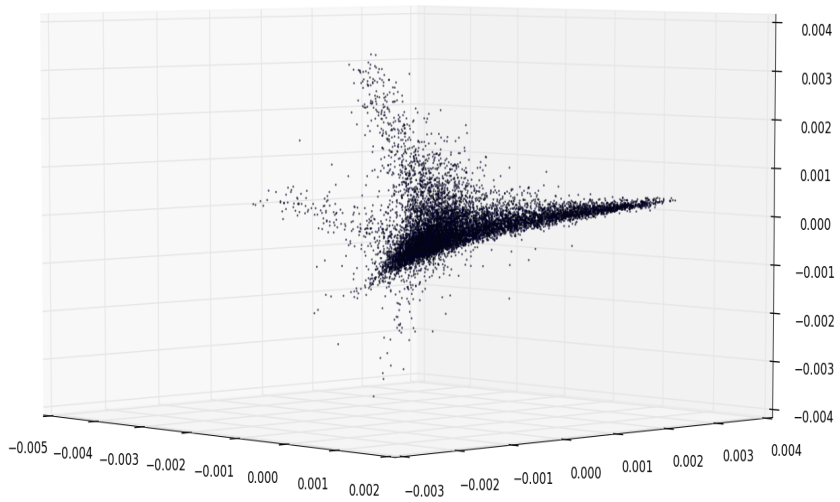


- Implemented in python, compatible with scikit-learn
- Statistical/methodological novelty
  - implements recent advances in the statistical understanding of manifold learning, e.g radius based neighborhoods [Hein 2007], consistent graph Laplacians [Coifman 2006], Metric learning
- Designed for performance
  - 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)
- Designed for extensions

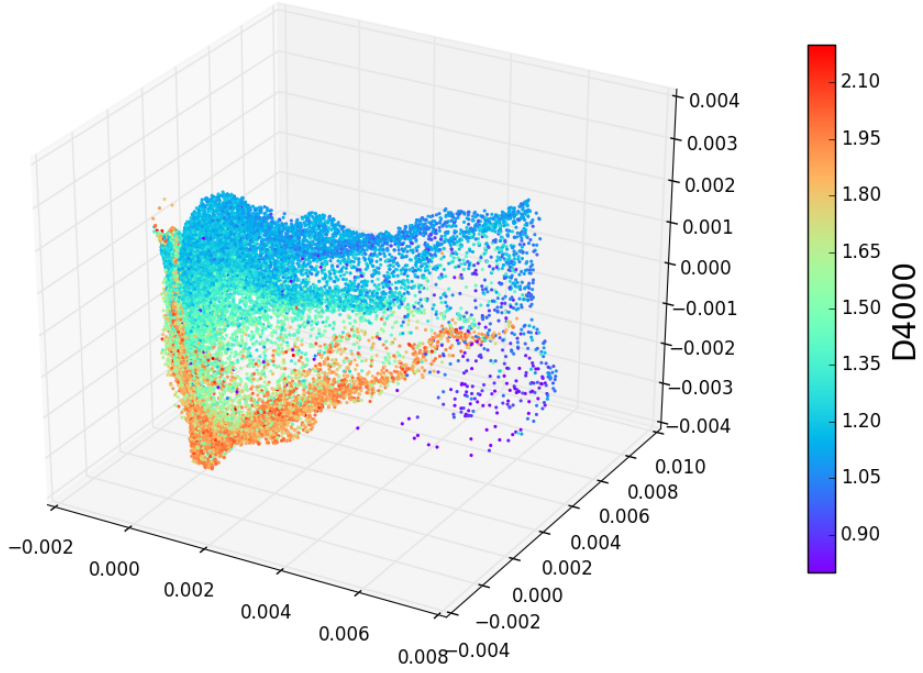
[1] Fast Approximate Nearest Neighbor search

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

Software overview

- Installation and dependencies ▸ Installation
- Examples/Tutorials ▸ Examples
- The `Geometry` class ▸ Geometry
- Embeddings ▸ Embeddings
- Other functionality
  - Metric manifold learning [Perrault-Joncas,M 2013], Riemannian Relaxation [McQueen,M,Perrault-Joncas NIPS2016], scale estimation [Perrault-Joncas,M,McQueen NIPS2017]
  - Clustering: k-means, spectral
  - Visualization tools

megaman on Theta

- Parallelism support (OpenMP)
  - Nearest neighbors (brute force)
  - Eigenproblem: SLEPc added
- Commonly used kernels for MD included and tested
  - SOAP, SLATM
- (in progress) Highly parallelizable random projection based graph Laplacian construction

"Polymorphic landscapes of molecular crystals

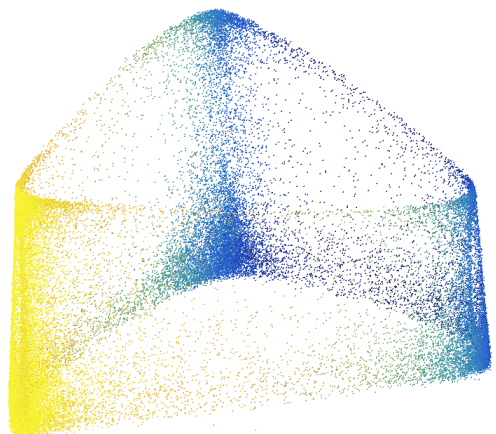
What next?

- Recent extensions:
  - Embedding calculated for subset of data, interpolated for the rest (Nyström extension), lazy evaluations of g
  - spectral and k-means clustering
  - Riemannian relaxation, scale estimation
- Near future (pilot implementations exist)
  - Gaussian Process regression on manifolds
  - Principal Curves and Surfaces [Ozertem 2011]
  - Integration with TDA, i.e. `GUDHI`
- Next
  - More scalable Laplacian construction?
  - Platform for scalable unsupervised learning and non-parametric statistics

Challenge: neighborhood graph

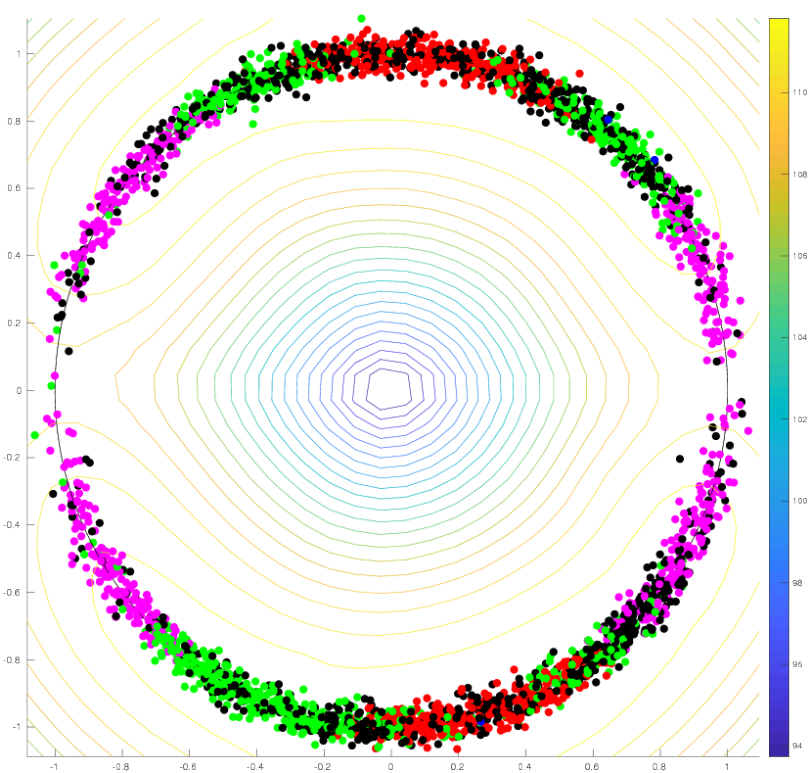
- We need radius-neighbors
  - i.e. [approximately] all neighbors of p_i within radius r
  - Why? [Hein 07, Coifman 06, Ting 10, . . .] k-nearest neighbor Laplacians do not converge to Laplace-Beltrami operator (but to $\Delta + 2\nabla(\log p) \cdot \nabla$)

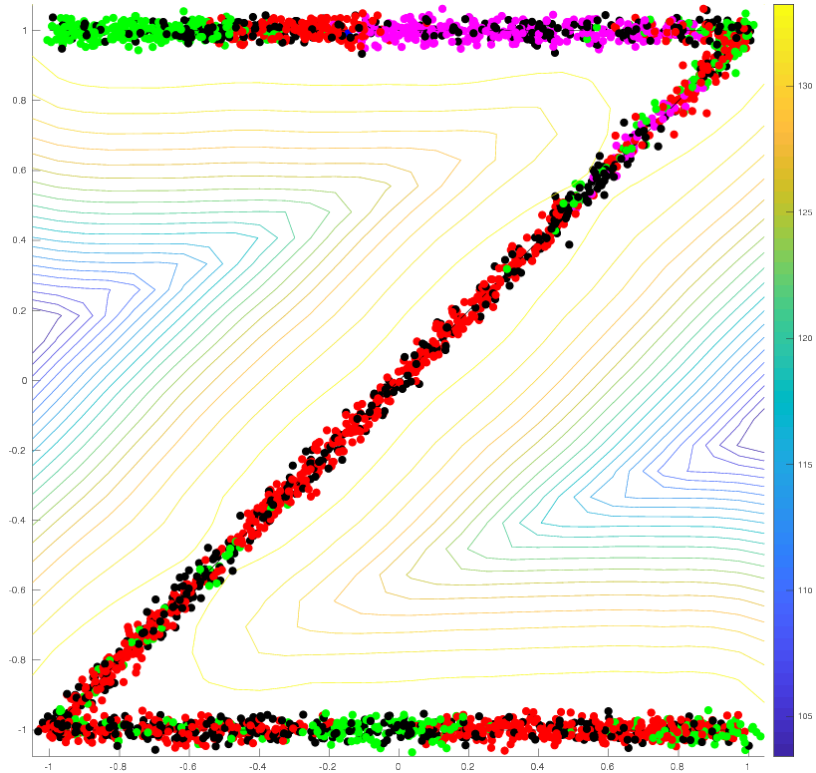
- How to construct them?
  - existing scalable software (FLANN) finds k-nearest neighbors
- Laplacian less tractable numerically
  - number of non-zeros in each row depends on data density



Outline

Finding Density Ridges

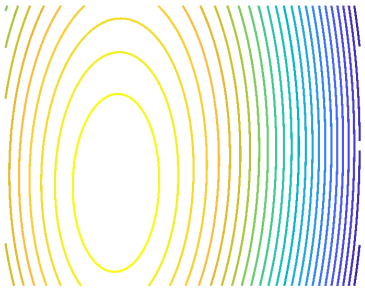




- data in $\mathbb{R}^D$ near a curve (or set of curves)
- wanted: track the ridge of the data density

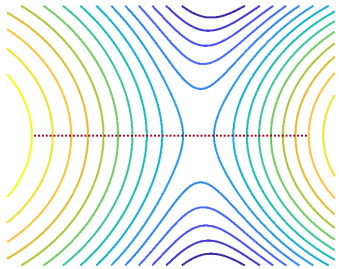
Mathematically,

Peak



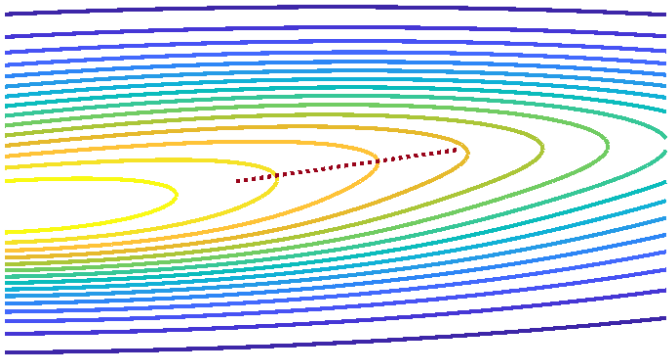
$\nabla p = 0$
$\nabla^2 p \prec 0$

Saddle



$\nabla p = 0$
$\nabla^2 p$ has $\lambda_1 > 0$,
$\lambda_{2:D} < 0$

Ridge



$\nabla p = 0$ in span$\{v_{2:D}\}$
v_j e-vector for λ_j, $j = 1 : D$
$\nabla^2 p$ has $\lambda_{2:D} < 0$

In other words, on a ridge

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

SCMS Algorithm

SCMS = Subspace Constrained Mean Shift

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

for $k = 1, 2, \ldots$

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

until convergence

- Algorithm SCMS finds 1 point on ridge; n restarts to cover all density
- Run time $\propto nD^2$/iteration
- Storage $\propto D^2$

Accelerating SCMS

- reduce dependency on n per iteration
  - index data (clustering, KD-trees, . . .)
  - we use FLANN [Muja,Lowe]
  - $n \leftarrow n'$ average number of neighbors
- 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
  - $D^2 \leftarrow mD$ with $m \approx 5$

(Approximate) SCMS step without computing Hessian

Recall SCMS = Subspace Constrained Mean Shift

- Given $g \propto \nabla p(x)$
- Wanted $\text{Proj}_{v_1^\perp} g = (I - v_1 v_1^T) g$
- Need v_1

  principal e-vector of $H = -\nabla^2(\ln p)$ for $\lambda_1 =$ smallest e-value of H

  without computing/storing H

(Approximate) SCMS step without Hessian: First idea

- Wanted
  v_1 principal e-vector of $H = -\nabla^2(\ln p)$ for $\lambda_1 =$ smallest e-value of H

- First Idea
  1. use LBFGSS to approximate H^{-1} by $\hat{H^{-1}}$ of rank $2m$ [Nocedal & Wright]
  2. $\hat{v}_1$ obtained by $2m \times 2m$ SVD + Gram-Schmidt

- Run time $\propto Dm + m^2$ / iteration (instead of nD^2)
- Storage $\propto 2mD$ for $\{x^{k-l} - x^{k-l-1}\}_{l=1:m}$, $\{g^{k-l} - g^{k-l-1}\}_{l=1:m}$

- Problem v_1 too inaccurate to detect stopping

(Approximate) SCMS step without Hessian: Second idea

- Wanted
  v_1 principal e-vector of $H = -\nabla^2(\ln p)$ for $\lambda_1 =$ smallest e-value of H

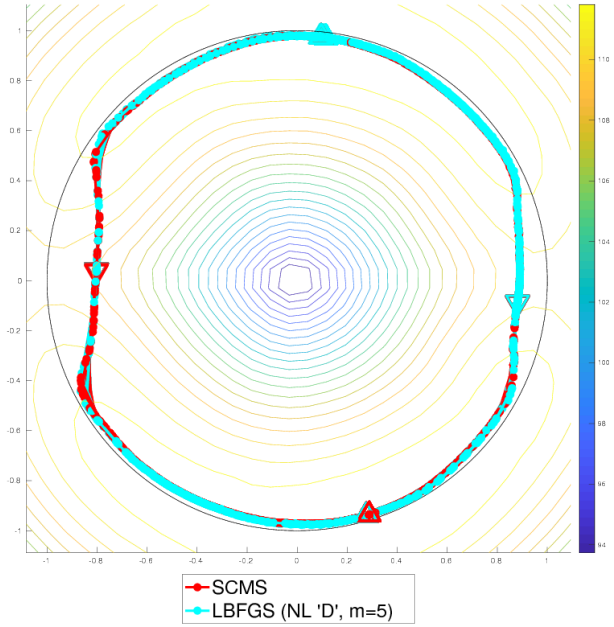
- Second Idea
  1. store $\{x^{k-l} - x^{k-l-1}\}_{l=1:m} \cup \{g^{k-l} - g^{k-l-1}\}_{l=1:m} = V$
  2. minimize $v^T H v$ s.t. $v \in \text{span } V$ where H is exact Hessian

- Possible because $H = \frac{1}{\sum c_i} \sum c_i u_i u_i^T - gg^T - \frac{1}{h^2} I$ with $c_{1:n}, u_{1:n}$ computed during Mean-Shift

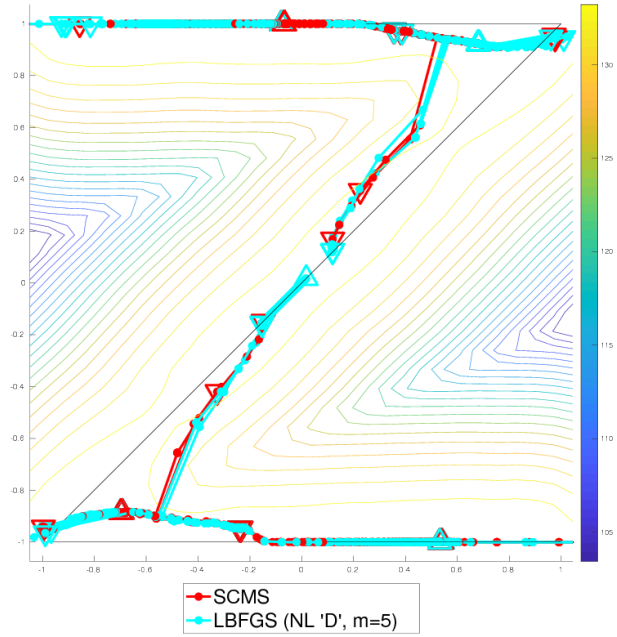
- Run time $\propto n' Dm + m^2$ / iteration (instead of nD^2)
- Storage $\propto 2mD$

- Much more accurate

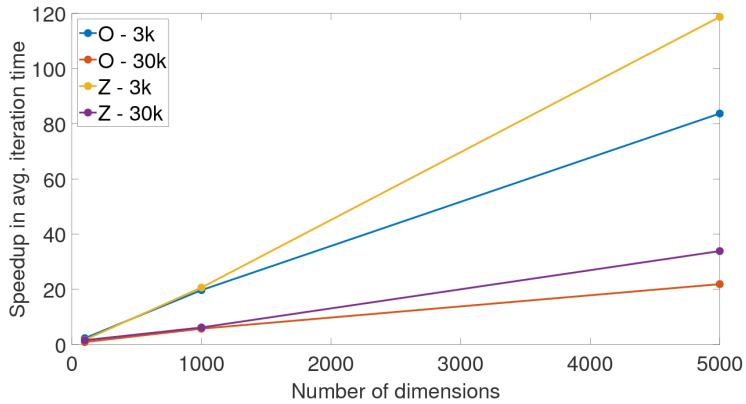
Principal curves SCMS vs. L-SCMS

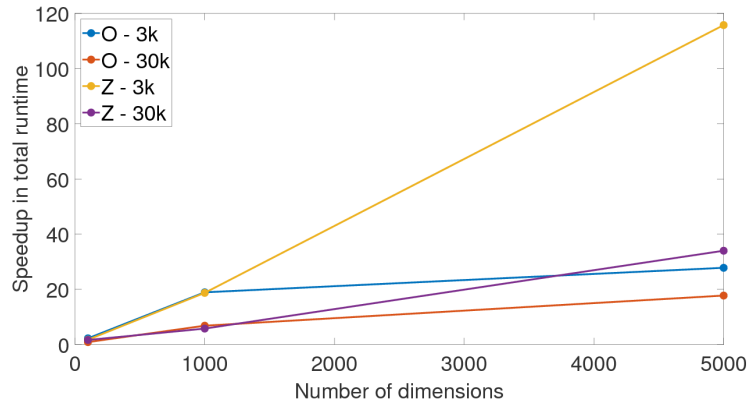


Speedup per iteration



Total speedup w.r.t. SCMS





For large n neighbor search dominates

Manifold learning for sciences and engineering

Manifold learning should be like PCA

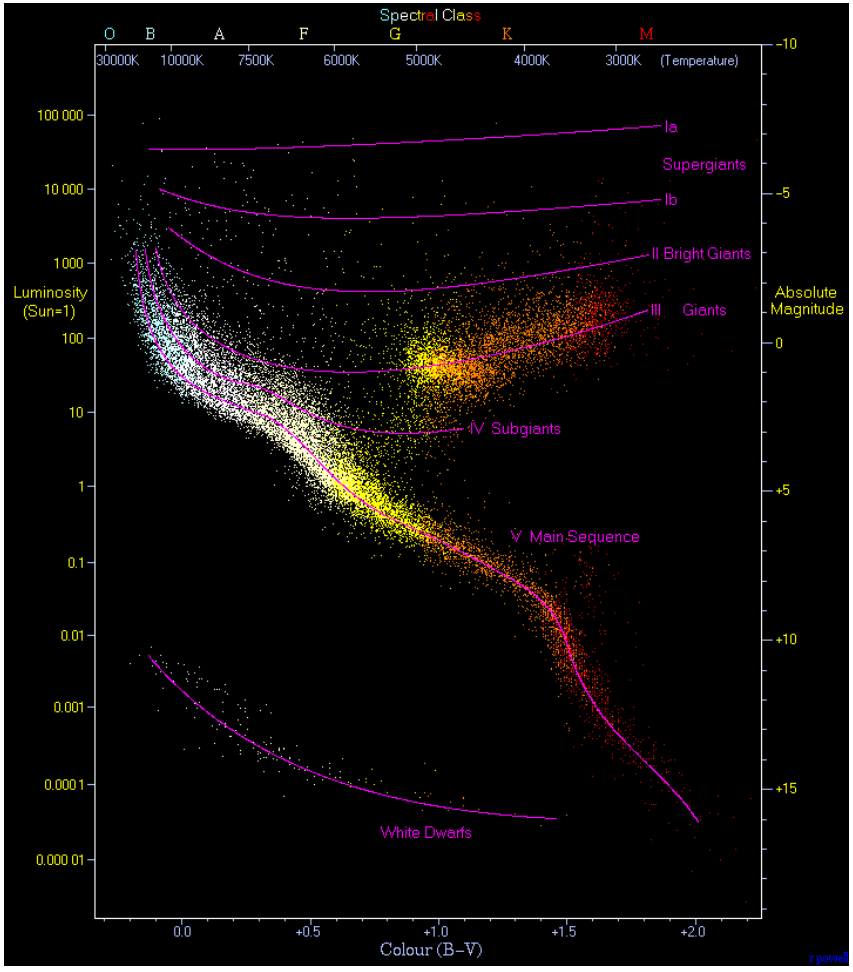
- tractable/scalable
- "automatic" – minimal burden on human
- first step in data processing pipe-line

  should not introduce artefacts

More than PCA

- estimate richer geometric/topological information
- dimension
- borders, stratification
- clusters
- Morse complex
- meaning of coordinates/continuous parametrization

Manifold Learning for engineering and the sciences



- "physical laws through machine learning"
- scientific discovery by quantitative/statistical data analysis
- manifold learning as preprocessing for other tasks

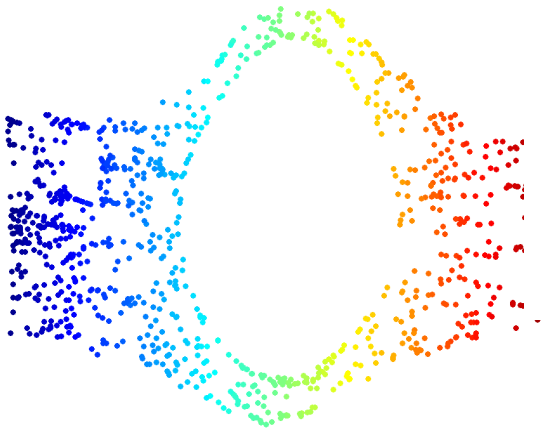
Thank you

Preserving topology vs. preserving (intrinsic) geometry

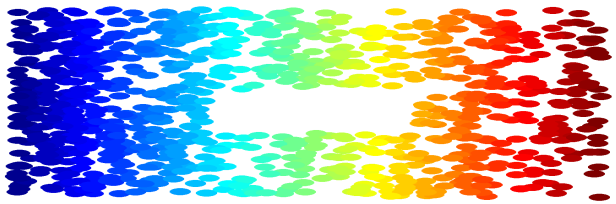
- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$

- 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.
- Diffusion Maps embedding is isometric in the limit [Besson 1994]
- algorithm based on Nash's theorem (isometric embedding for very low d) [Verma 11]
- Isomap [Tennenbaum,]recovers flat manifolds isometrically
- Consistency results for Laplacian and eigenvectors
  - [Hein & al 07,Coifman & Lafon 06, Singer 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

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 fundamental 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 (intrinsic) 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)

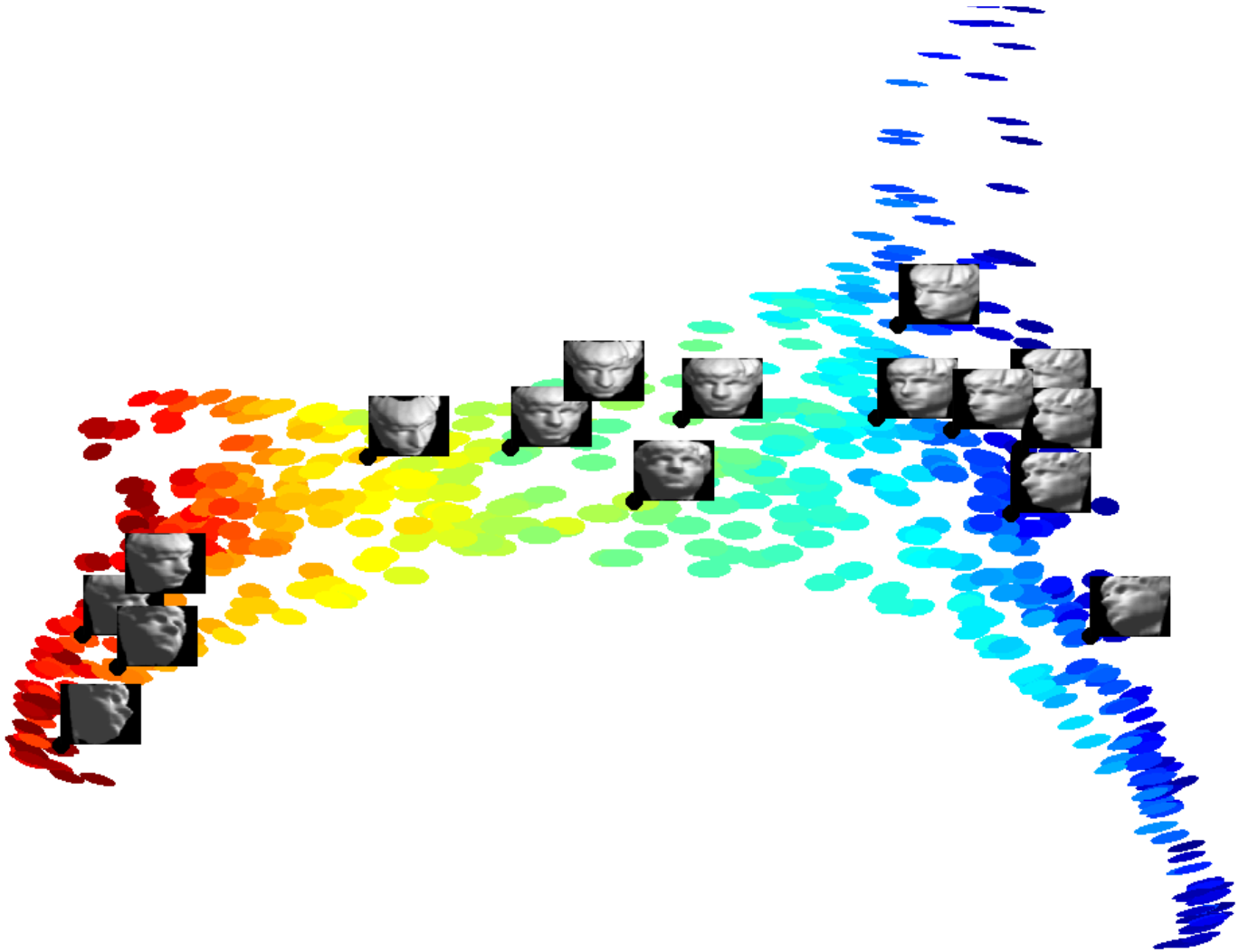
Problem formulation

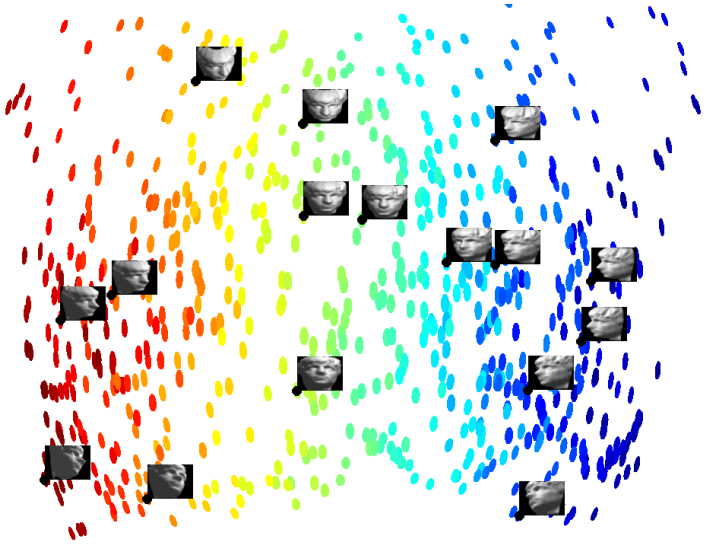
- Given:
  - data set $\mathcal{D} = \{p_1, \dots p_n\}$ sampled from manifold $\mathcal{M} \subset \mathbb{R}^D$
  - embedding $\{ x_i = \phi(p_i),\ p_i \in \mathcal{D} \}$
    by e.g LLE, Isomap, LE, ...
- Estimate $G_i \in \mathbb{R}^{m \times m}$ the (pushforward) Riemannian metric for $p_i \in \mathcal{D}$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

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

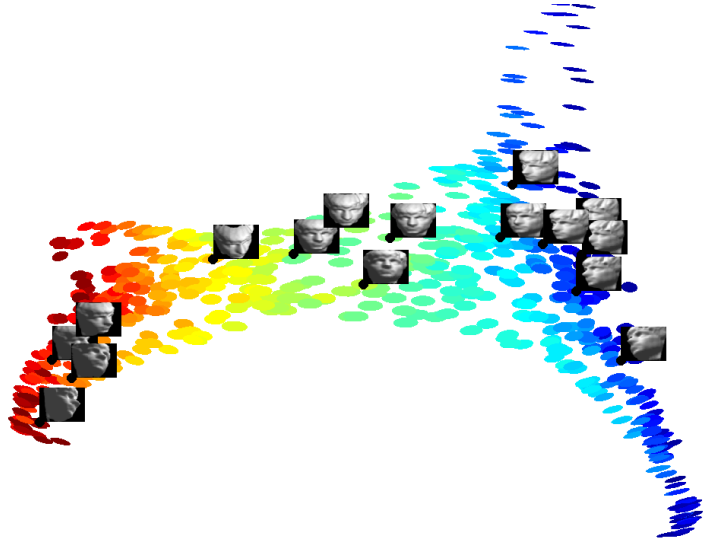
g for Sculpture Faces

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

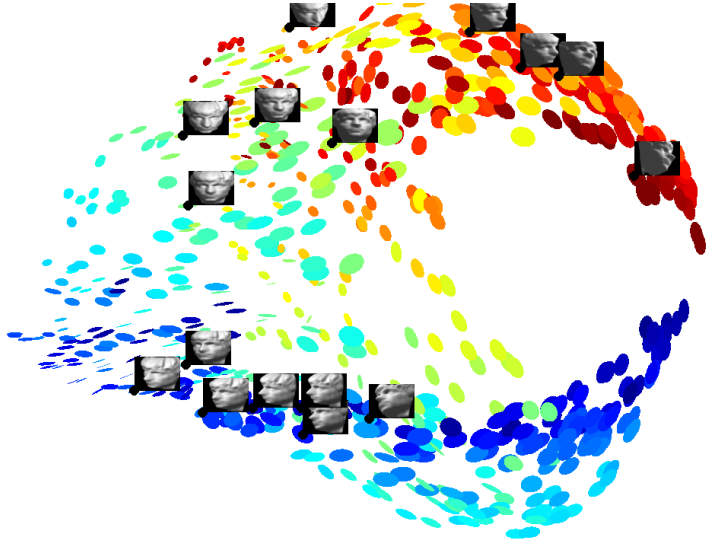




Isomap



LTSA



Laplacian Eigenmaps

Relation between g and Δ

- $\Delta =$ 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)

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

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

Intuition:

- at each point $p \in \mathcal{M}$, $g(p)$ is a $d \times d$ matrix
- apply Δ to embedding coordinate functions $\phi_1, \ldots \phi_m$
- this produces $g^{-1}(p)$ in the given coordinates
- our algorithm implements matrix version of this operator result
- consistent estimation of Δ is solved [Coifman&Lafon 06,Hein&al 07]

Algorithm to Estimate Riemann metric g (Main Result 2)

Given dataset $\mathcal{D}$

1. Preprocess (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

Algorithm MetricEmbedding

Input data $\mathcal{D}$, m embedding dimension, ϵ resolution

1. Construct neighborhood graph p, p' neighbors iff $||p - p'||^2 \leq \epsilon$
2. Construct similary matrix
   $S_{pp'} = e^{-\frac{1}{\epsilon}||p-p'||^2}$ iff p, p' neighbors, $S = [S_{pp'}]_{p,p' \in \mathcal{D}}$
3. Construct (renormalized) Laplacian matrix [Coifman & Lafon 06]
   3.1 $t_p = \sum_{p' \in \mathcal{D}} S_{pp'}$, $T = \operatorname{diag} t_p$, $p \in \mathcal{D}$
   3.2 $\tilde{S} = I - T^{-1} S T^{-1}$
   3.3 $\tilde{t}_p = \sum_{p' \in \mathcal{D}} \tilde{S}_{pp'}$, $\tilde{T} = \operatorname{diag} \tilde{t}_p, p \in \mathcal{D}$
   3.4 $P = \tilde{T}^{-1} \tilde{S}$.
4. Embedding $[\phi_p]_{p \in \mathcal{D}} = \text{GenericEmbedding}(\mathcal{D}, m)$
5. Estimate embedding metric H_p at each point

   denote $Z = X * Y$, $X, Y \in \mathbb{R}^N$ iff $Z_i = X_i Y_i$ for all i

   5.1 For $i, j = 1 : m$, $H^{ij} = \frac{1}{2}\left[P(\phi_i * \phi_j) - \phi_i * (P\phi_j) - \phi_j * (P\phi_i)\right]$ (column vector)

   5.2 For $p \in \mathcal{D}$, $\tilde{H}_p = [H_p^{ij}]_{ij}$ and $H_p = \tilde{H}_p^{\dagger}$

   Ouput $(\phi_p, H_p)_{p \in \mathcal{D}}$

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
- Algorithm independent geometry preserving method
- Outputs of different algorithms on the same data are comparable
- Models built from compressed data are more interpretable

Applications

- Correcting distortion
  - Integrating with the local volume/length units based on G_i
  - Riemannian Relaxation (coming next)
- Estimation of neighborhood radius [Perrault-Joncas,M,McQueen NIPS17]
- and of intrinsic dimension d (variant of [Chen,Little,Maggioni,Rosasco])
- Accelerating Topological Data Analysis (in progress)