Geometric Manifold Learning and Collective Variables

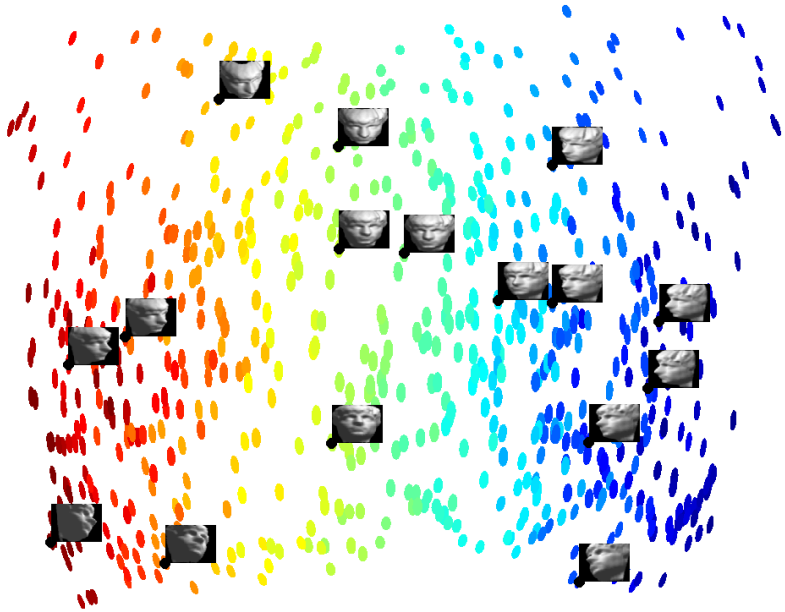
Marina Meilă

University of Washington
mmp@stat.washington.edu

MPS2016 5/12/2016

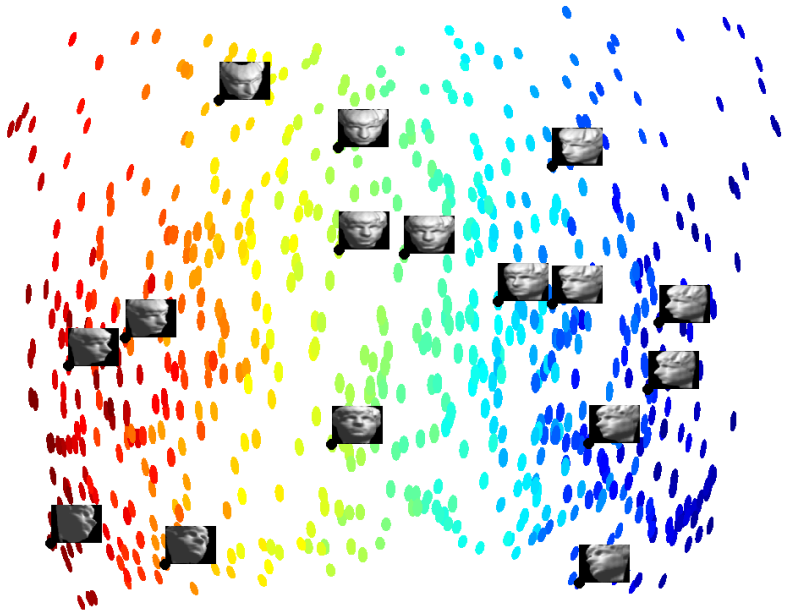
Outline

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

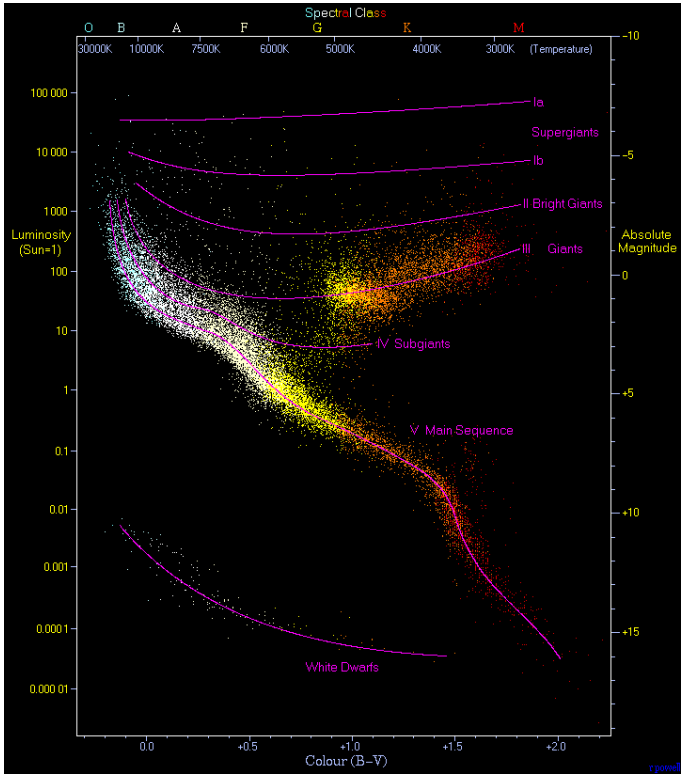
When to do (non-linear) dimension reduction



Why?

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

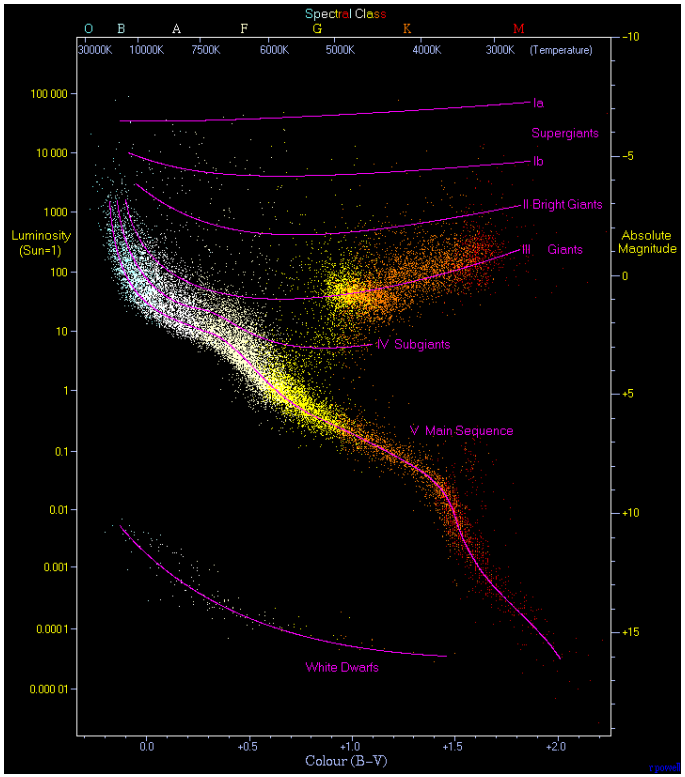
When to do (non-linear) dimension reduction



Why?

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

When to do (non-linear) dimension reduction

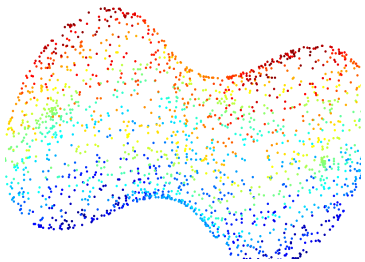


Why?

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

How? Brief intro to manifold learning algorithms

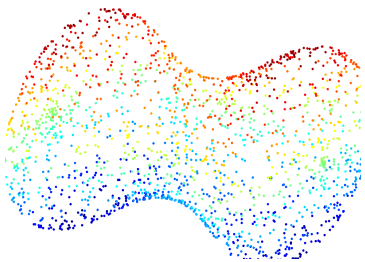
- **Input** Data $p_1, \ldots p_n$, embedding dimension m, neighborhood scale parameter ϵ



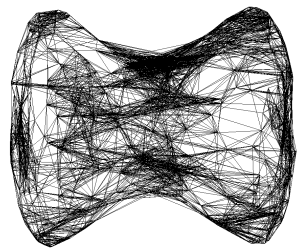
$p_1, \ldots p_n \subset \mathbb{R}^D$

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

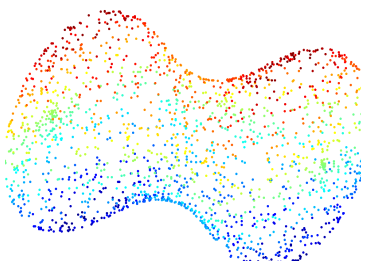


$p_1, \ldots p_n \subset \mathbb{R}^D$

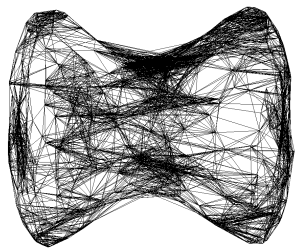


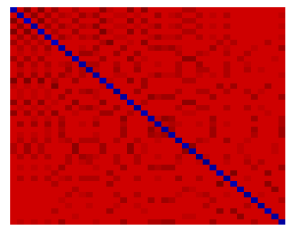
How? Brief intro to manifold learning algorithms

- **Input** Data $p_1, \ldots p_n$, 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 $p_1, \ldots p_n$, 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

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

- 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 $(\phi^1(p), \ldots \psi^m(p))$

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]

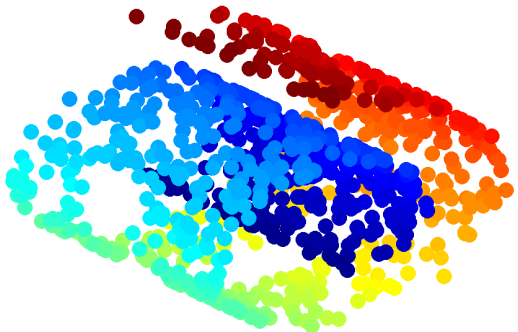
- Find all shortest paths in neighborhood graph, construct matrix of distances

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

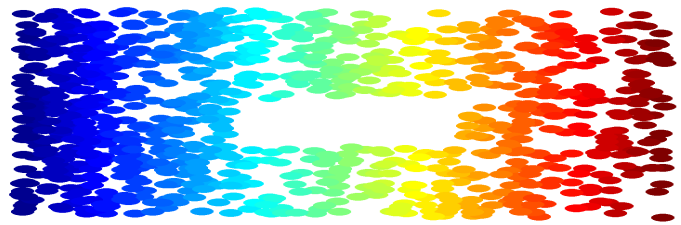
- use M and Multi-Dimensional Scaling (MDS) to obtain m dimensional coordinates for $p \in \mathcal{D}$

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

points in $D \geq 3$ dimensions

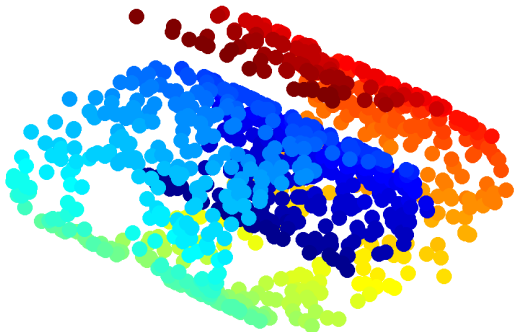


same points reparametrized in 2D



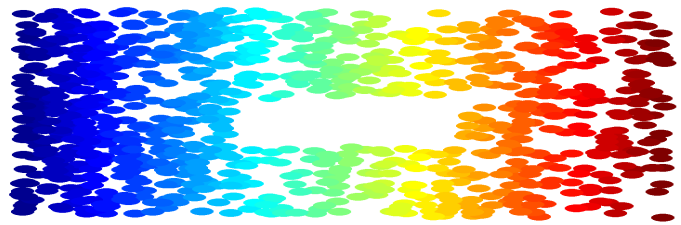
A toy example (the "Swiss Roll" with a hole)

points in $D \geq 3$ dimensions



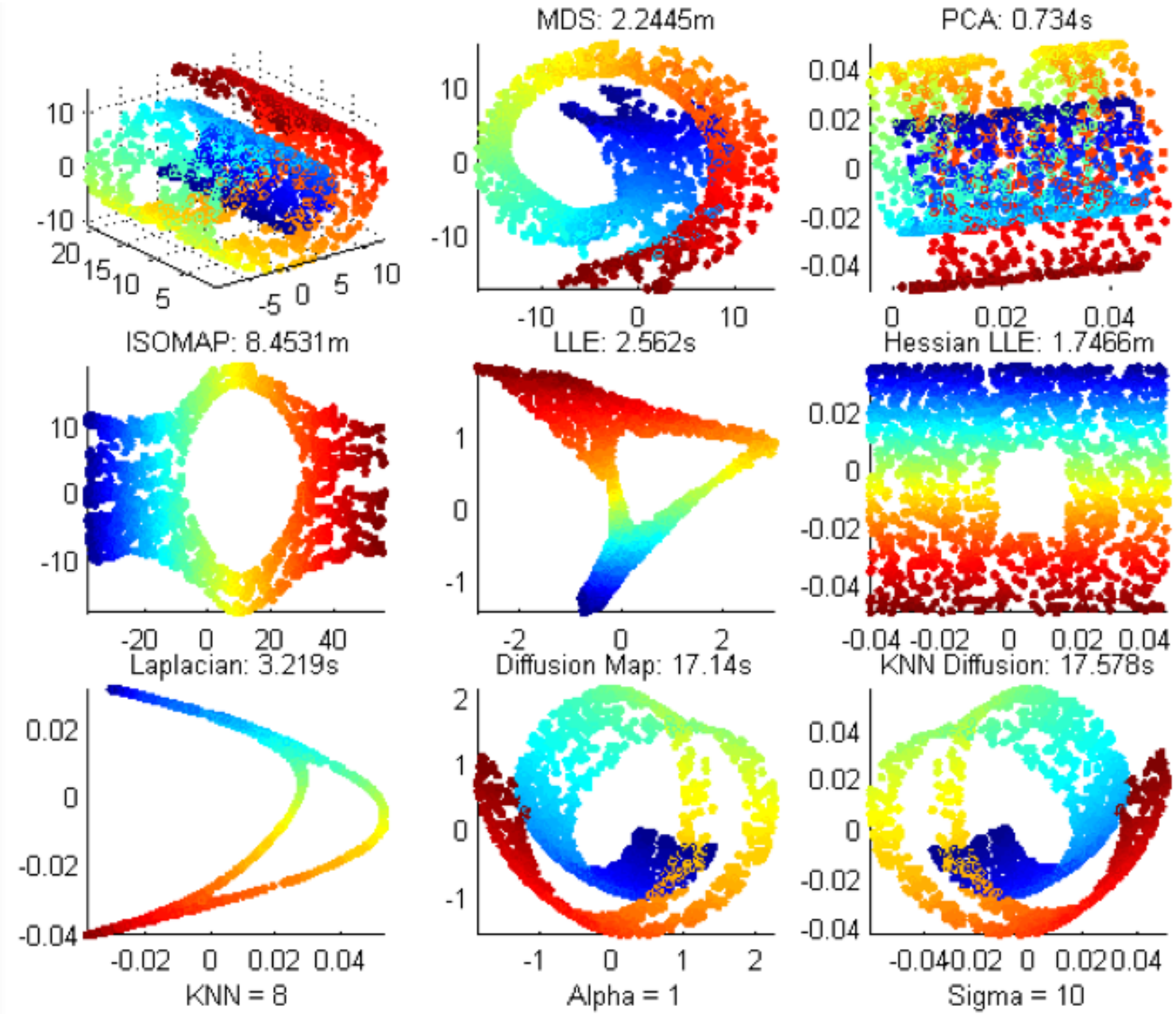
Input

same points reparametrized in 2D

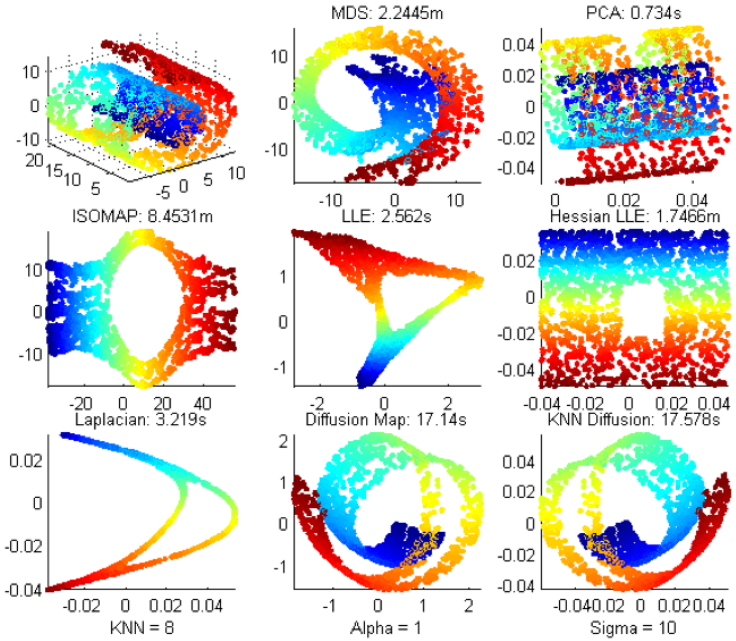


Desired output

Embedding in 2 dimensions by different manifold learning algorithms



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)

Outline

Outline

Preserving topology vs. preserving (intrinsic) geometry

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

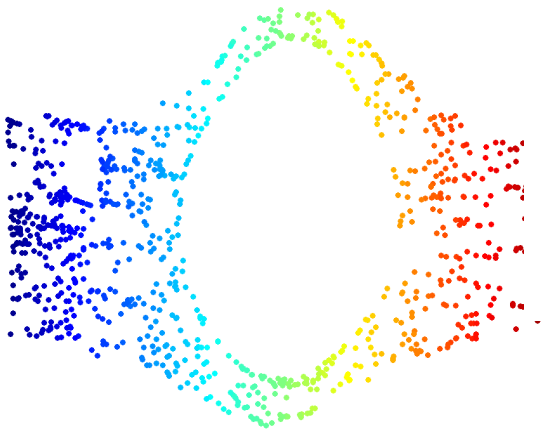
- Mapping $\mathcal{M} \longrightarrow \phi(\mathcal{M})$ is diffeomorphism
  - preserves topology
  - often satisfied by embedding algorithms
- Mapping ϕ preserves
  - distances along curves in $\mathcal{M}$
  - angles between curves in $\mathcal{M}$
  - areas, volumes

Preserving topology vs. preserving (intrinsic) geometry

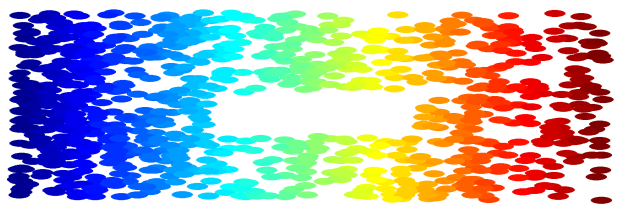
- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$

- 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



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.

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

distances, volumes, angles, ...

Dominique Perrault-Joncas

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

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

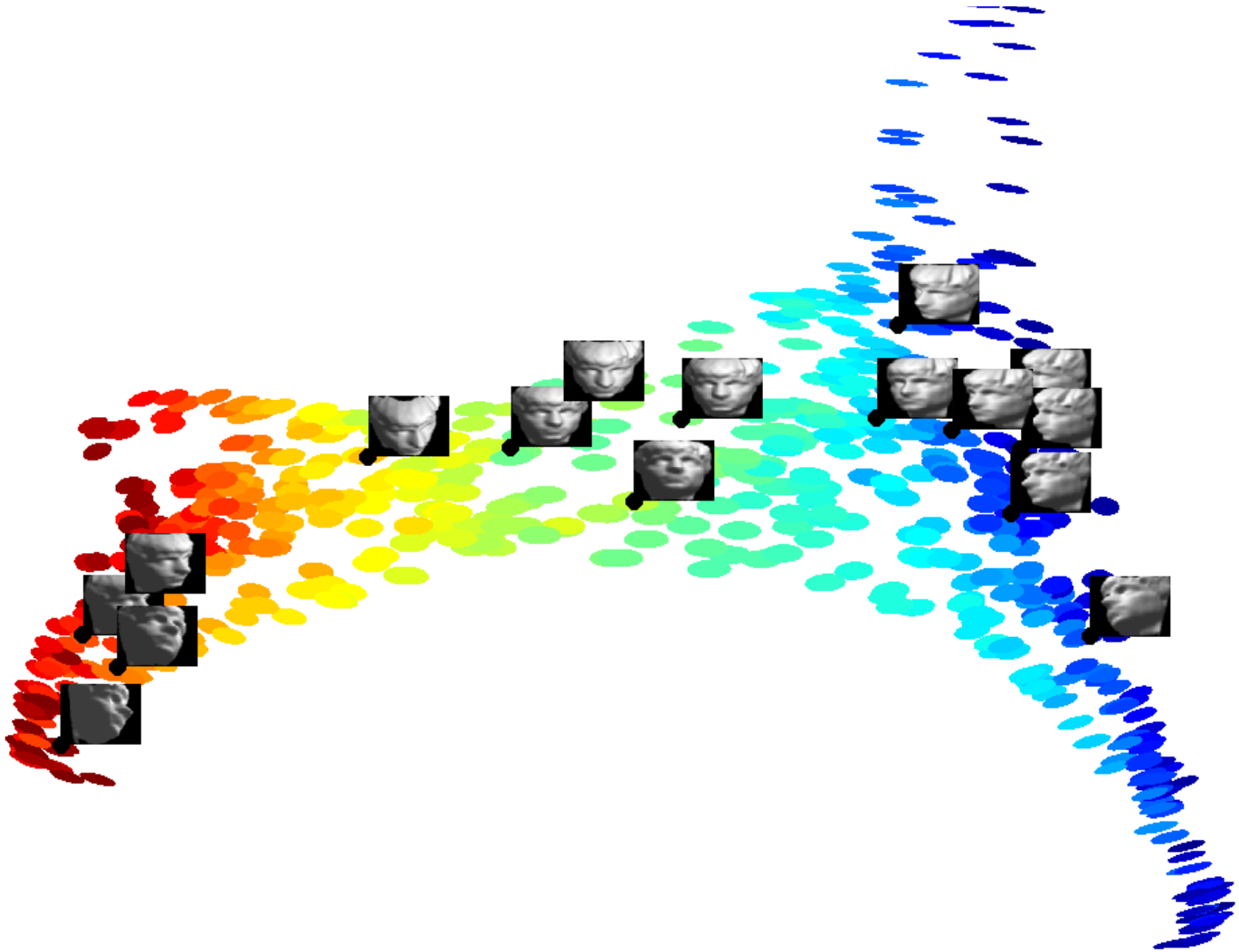
- Under a change of parametrization, g changes in a way that leaves geometric quantities invariant

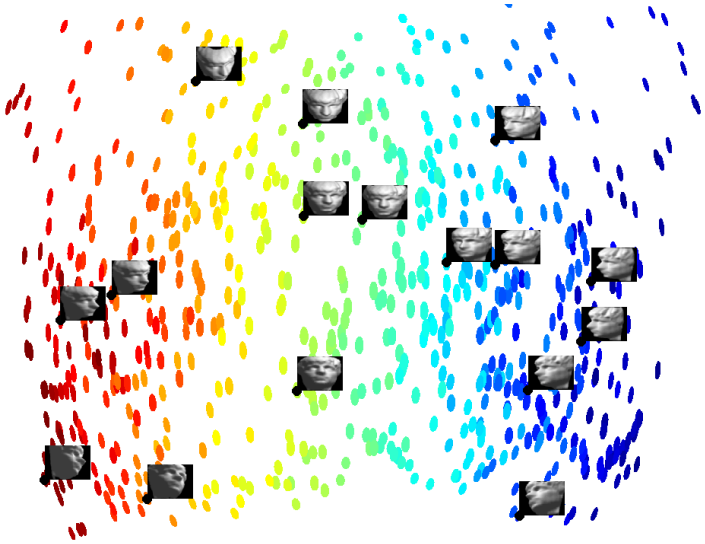
- Current algorithms: estimate $\mathcal{M}$
- This talk: estimate g along with $\mathcal{M}$

  (and in the same coordinates)

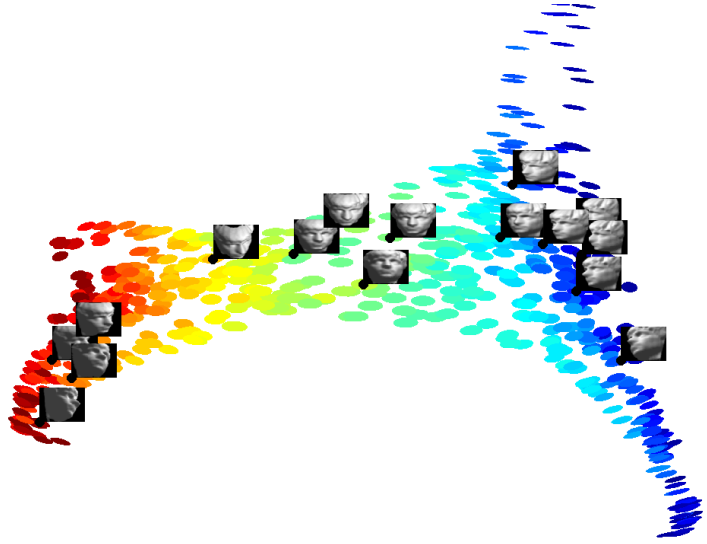
g for Sculpture Faces

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

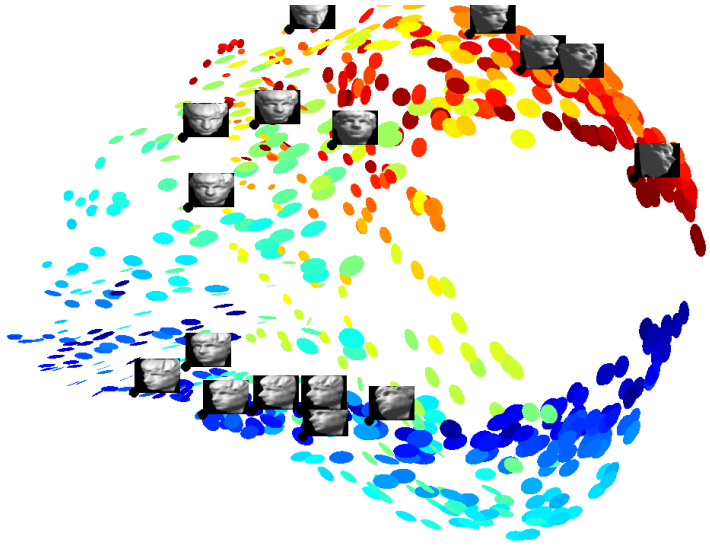




Isomap



LTSA



Laplacian Eigenmaps

Metric Manifold Learning summary

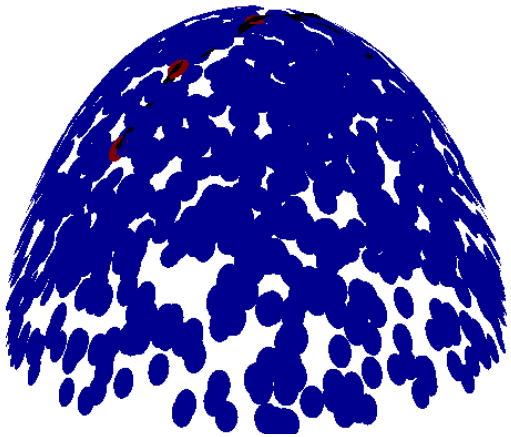
Metric Manifold Learning = estimating (pushforward) Riemannian metric G_i along with embedding coordinates x_i

Why useful

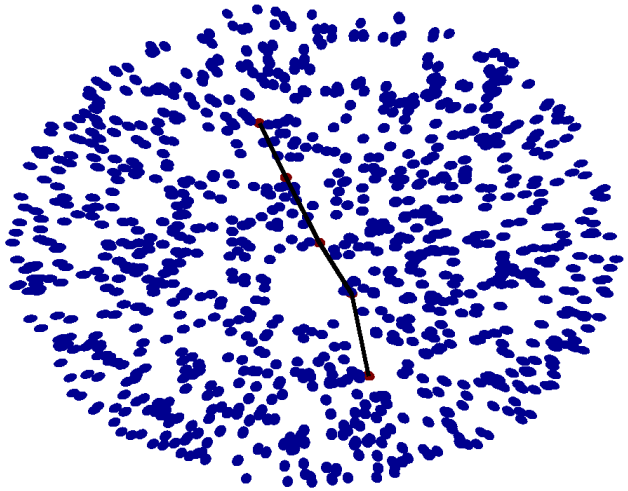
- Measures local distortion induced by any embedding algorithm
  $G_i = I_d$ when no distortion at p_i
- Corrects distortion
  - Integrating with the local volume/length units based on G_i
  - Riemannian Relaxation (coming next)
- Algorithm independent geometry preserving method
- Outputs of different algorithms on the same data are comparable
- Models built from compressed data are more interpretable

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

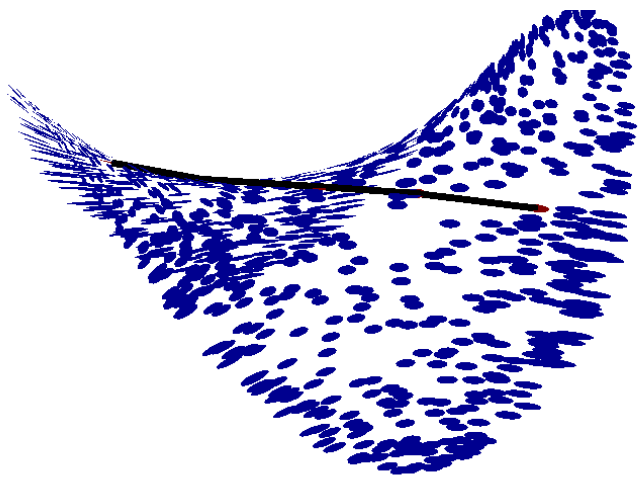
- Geodesic distance $=$ shortest path on $\mathcal{M}$
- should be invariant to coordinate changes



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%

Calculating Areas/Volumes in the manifold

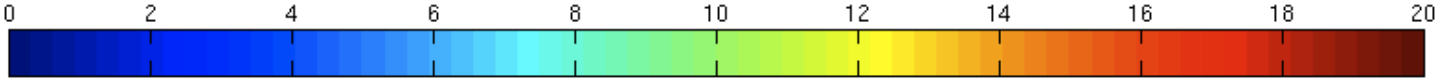
(Results for Hourglass data)

true area = 0.84

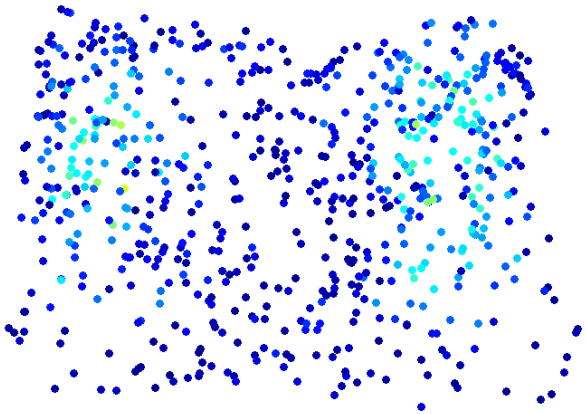
Embedding	Naive	Metric	Rel. err.
Original data	0.85 (0.03)	0.93 (0.03)	11.0%
Isomap	2.7	0.93 (0.03)	11.0%
LTSA	1e-03 (5e-5)	0.93 (0.03)	11.0%
LE	1e-05 (4e-4)	0.82 (0.03)	2.6%

Semisupervised learning with Gaussian Processes on Manifolds

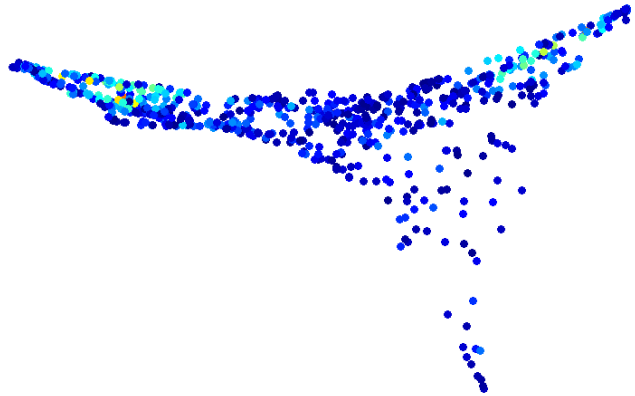
Sculpture Faces: Predicting Head Rotation



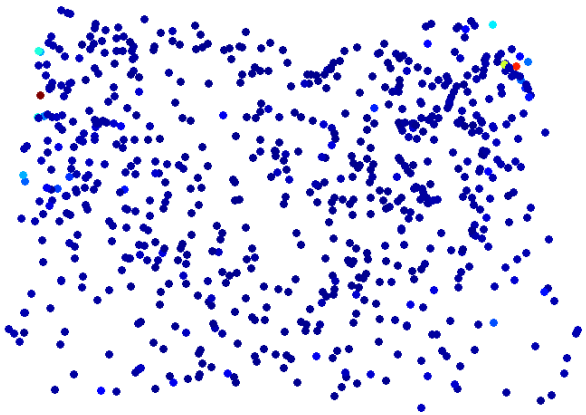
Absolute Error (AE) as percentage of range



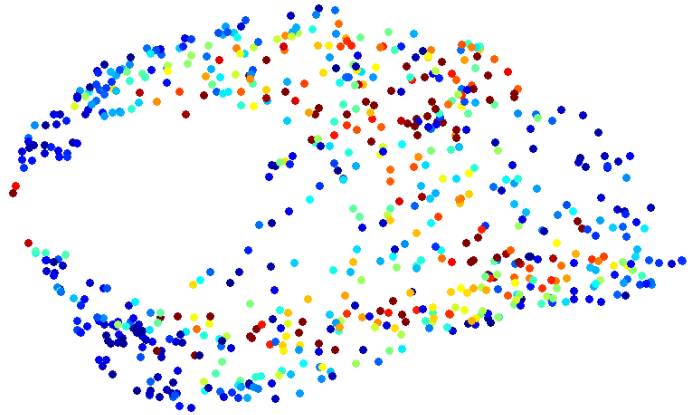
Isomap (Total AE = 3078)



LTSA (Total AE = 3020)



Metric (Total AE = 660)



Laplacian Eigenmaps (Total AE = 3078)

Outline

Riemannian Relaxation


James McQueen

Can we sometimes dispense with g?

Idea

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

Riemannian Relaxation


James McQueen

Can we sometimes 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


James McQueen

Can we sometimes 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


James McQueen

Can we sometimes 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

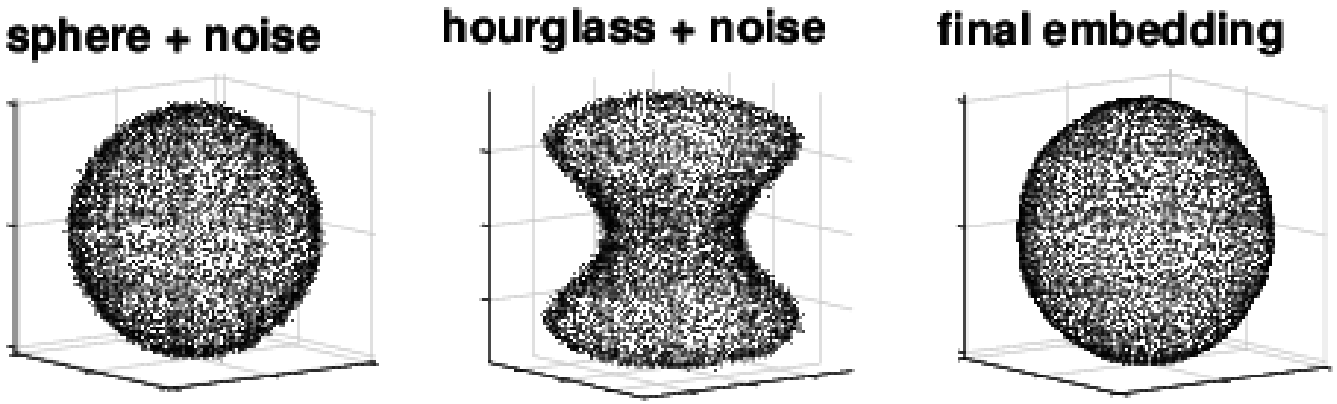
More details

- 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

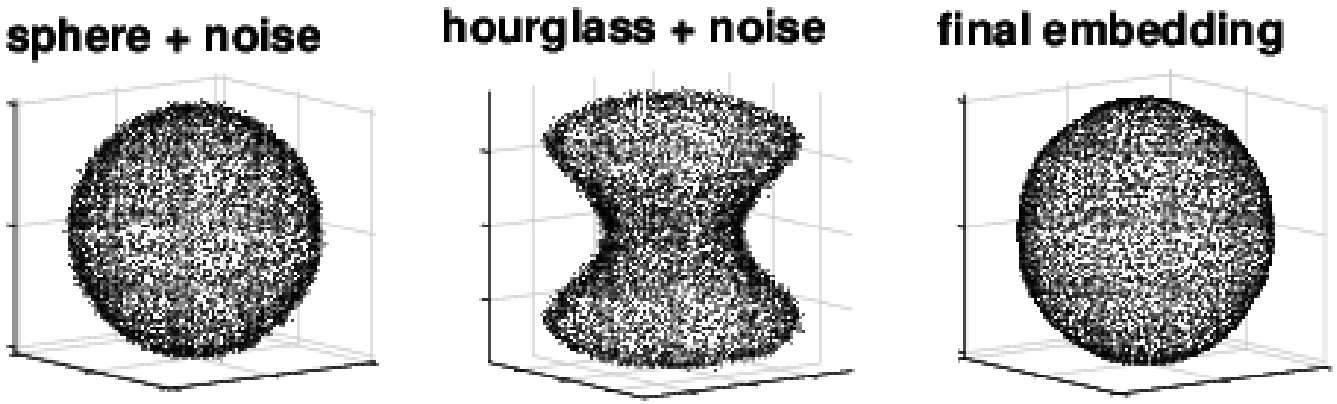


target

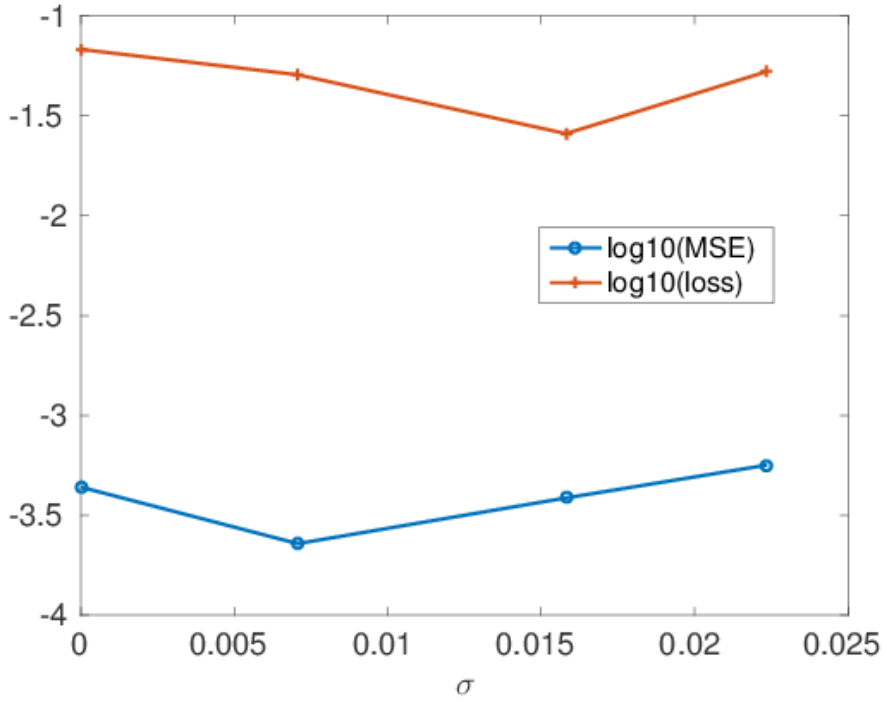
initialization

algorithm output

Riemannian Relaxation of a deformed sphere



target initialization algorithm output



Mean-squared error and loss vs. noise amplitude

Outline

Data as directed transitions

- Observations are
  - transitions between (discrete) states
  - or random walks on a graph
  - or weighted graph
- Transitions (weights) may not be symmetric ($W_{ij} \neq W_{ji}$)

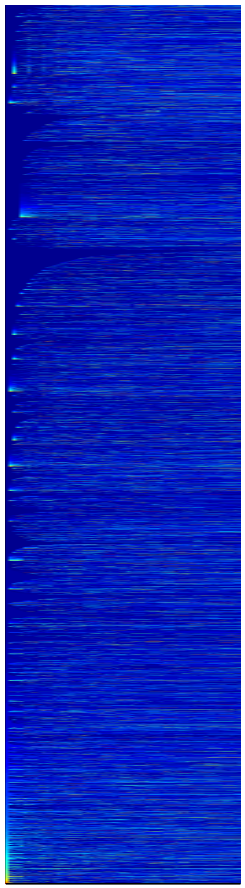
- Example: the National Longitudinal Survey of Youth (NLSY)
  - "carreer" sequences of length 16 years$\times$ 4 quarters (people aged 14–16 followed to age 30–32)
  - jobs (occupation, industry) represented by integer codes

person 1:	19	19	3	3	3	3	3	3	3	3	3	3	3	10	10
person 2:	3	3	3	3	3	3	3	1	3	1	3	1	3	1	35
person 3:	152	5	71	1	1	1	71	36	36	5	4	5	5	4	4
person 4:	3	3	3	3	9	3	3	8	2	8	5	5	8	239	239

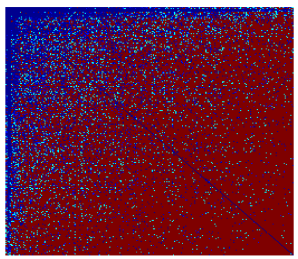
- Problem how to embed a directed graph with directed edges?

The NLSY data

data matrix
7,711 paths $\times 64$
quarters



$W \in \mathbb{R}^{356 \times 356}$, $W_{ij} \geq 0$, W asymmetric (blue is higher)



Problem how to embed a directed graph with directed edges?

Generative model

Observe a directed graph G, with n nodes, having directed weights $W = [W_{ij}]$ associated with its edges ($W_{ij} \neq W_{ji}$)

Assume Nodes of G sampled from manifold $\mathcal{M}$ of dimension d

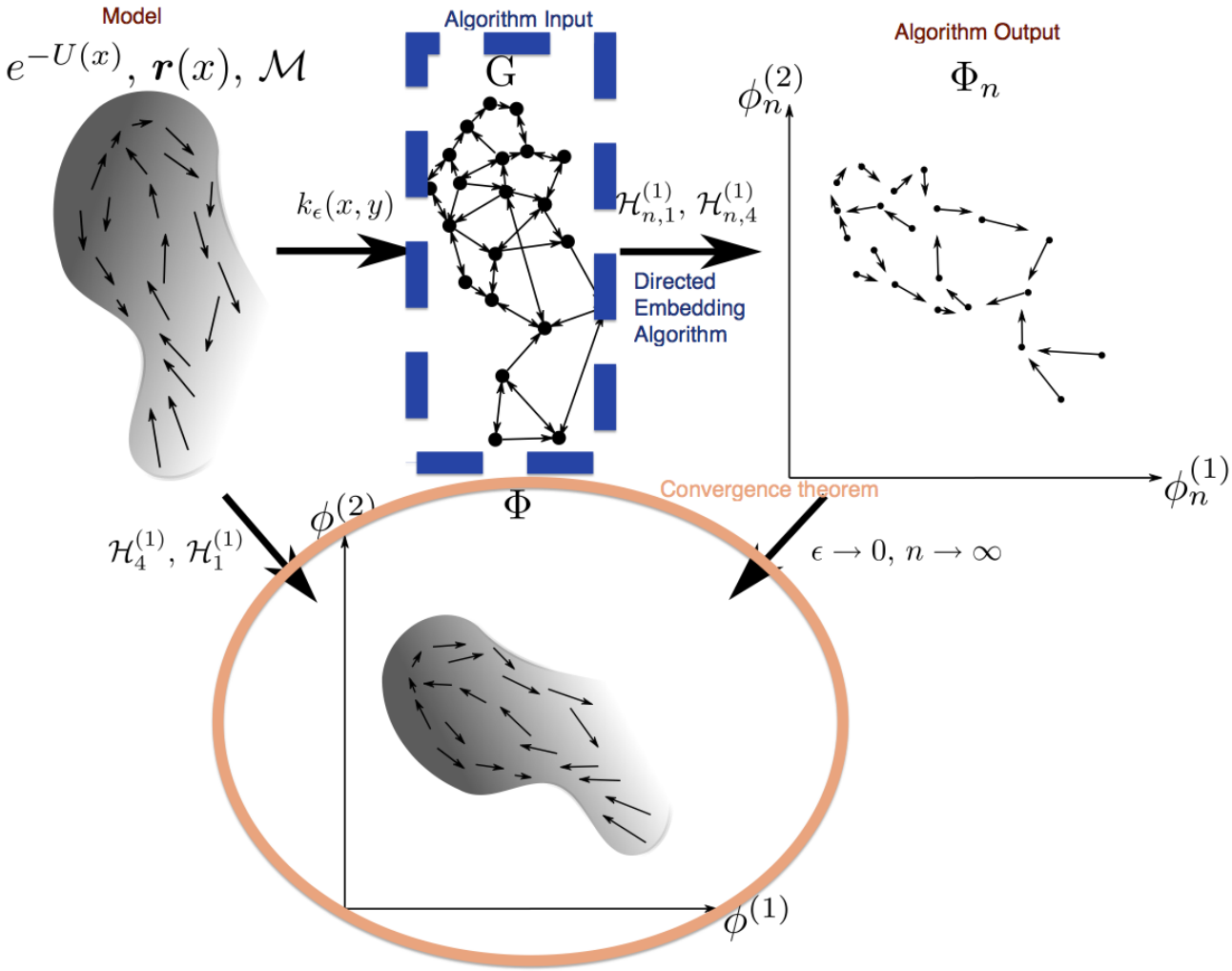
- according to some distribution $p(x) = e^{-U(x)}$
- edges weights W_{ij} are assigned by a directed similarity kernel $k_\epsilon(x_i, x_j)$ with

$$k_\epsilon(x, y) = \underbrace{h_\epsilon(x, y)}_{\text{symmetric}} + \underbrace{a_\epsilon(x, y)}_{\text{skew}-\text{symmetric}}$$

- $a_\epsilon(x_i, x_j)$ originates from a vector field $\mathbf{r}$ on $\mathcal{M}$

Wanted Estimates of manifold $\mathcal{M}$, density $p = e^{-U}$, vector field $\mathbf{r}$ from W

Generative model



Wanted Estimates of manifold $\mathcal{M}$, density $p = e^{-U}$, vector field $\mathbf{r}$ from W

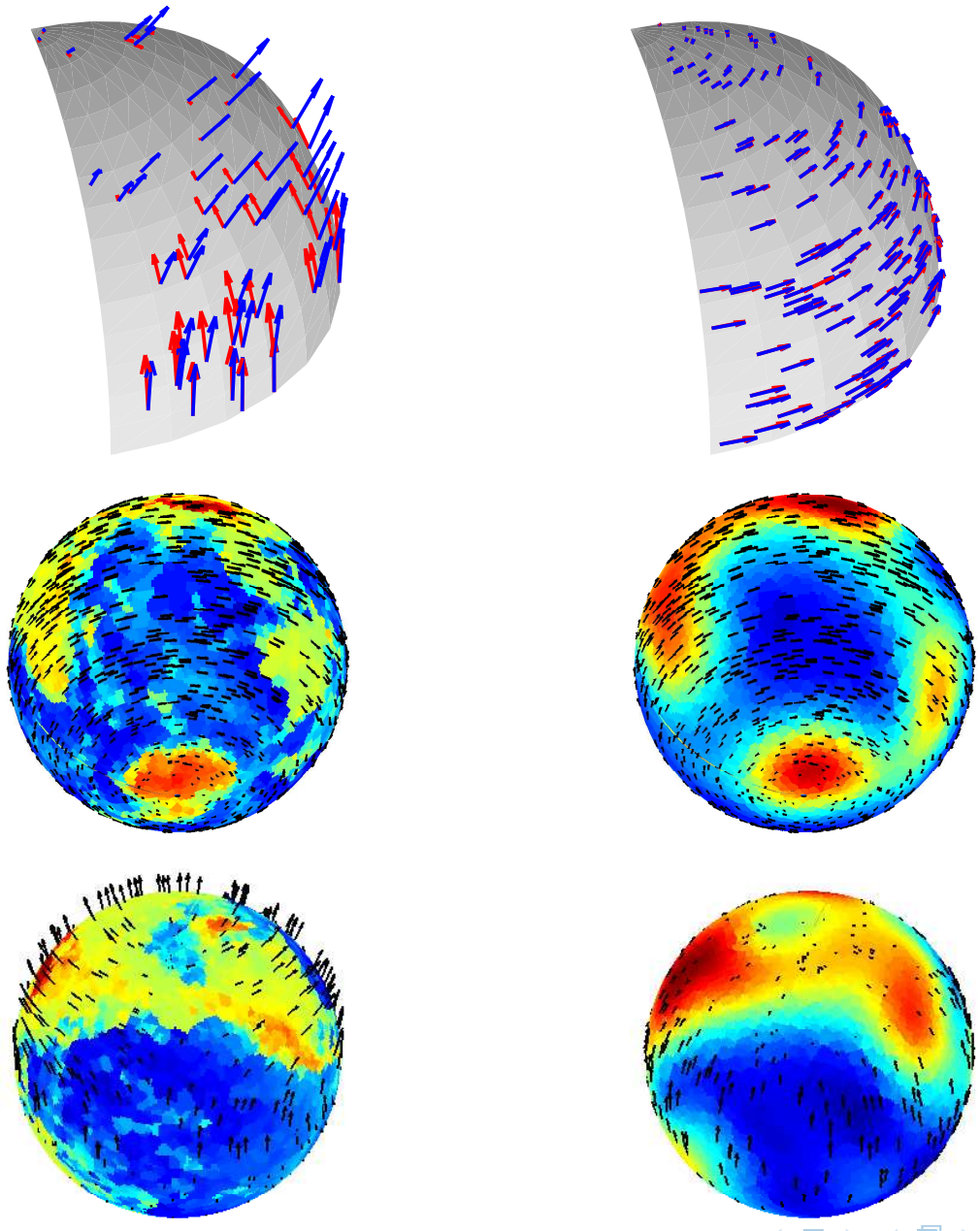
Directed Embedding Algorithm

Input: Affinity matrix $W_{i,j}$ and embedding dimension m, $(m \geq d)$

1. $S \leftarrow (W + W^T)/2$ *Estimate the Diffusion Maps embedding*
2. $q_i \leftarrow \sum_{j=1}^{n} S_{i,j}, \; Q = diag(q)$
3. $V \leftarrow Q^{-1} S Q^{-1}$
4. $q_i^{(1)} \leftarrow \sum_{j=1}^{n} V_{i,j}, \; Q^{(1)} = diag(q^{(1)})$
5. $H_{ss,n}^{(1)} \leftarrow {Q^{(1)}}^{-1} V$
6. Compute ϕ the $m+1$ largest right eigenvectors of $H_{ss,n}^{(1)}$ and discard ϕ_1.
7. Compute π left principal e-vector of $H_{ss,n}^{(1)}$. *Estimate the density*
8. $\pi \leftarrow \pi / \sum_{i=1}^{n} \pi_i$.
9. $p_i \leftarrow \sum_{j=1}^{n} W_{i,j}, \; P = diag(p)$ *Estimate the vector field* $\mathbf{r}$
10. $T \leftarrow P^{-1} W P^{-1}$
11. $p_i^{(1)} \leftarrow \sum_{j=1}^{n} T_{i,j}, \; P^{(1)} = diag(p^{(1)})$
12. $H_{aa,n}^{(1)} \leftarrow {P^{(1)}}^{-1} T$
13. $R \leftarrow (H_{aa,n}^{(1)} - H_{ss,n}^{(1)})\phi/2$. Columns 2 to $m+1$ of R are the vector field components in the direction of the corresponding coordinates of the embedding.

Examples – toy data

Input Output



Kernels, renormalized kernels, transport operators

The asymmetric transition kernel

$$k_\epsilon(x,y) = h_\epsilon(x,y) + a_\epsilon(x,y)$$

1. $h_\epsilon(x,y) = h_\epsilon(y,x)$ symmetric
   $h_\epsilon(x,y) = \epsilon^{-D} \exp(\frac{||x-y||^2}{\epsilon^2})$
   ϵ = kernel bandwidth
2. $a_\epsilon(x,y) = -a_\epsilon(y,x)$
   skew-symmetric

Set

$$a_\epsilon(x,y) = \frac{1}{2}(y-x)\cdot \mathbf{r}(x,y) h_\epsilon(x,y)$$

and $\mathbf{r}(x,y) = \mathbf{r}(y,x)$ vector field
This form is generic

Transport operators

$$p_\epsilon(x) = \int_{\mathcal{M}} k_\epsilon(x,y) p(y) dy$$

$$T_\epsilon[f](x) = \int_{\mathcal{M}} \frac{k_\epsilon(x,y)}{p_\epsilon(x)} f(y) p(y) dy$$

Renormalized kernels [Lafon,Coifman 06]

$$k_\epsilon^{(\alpha)}(x,y) = \frac{k_\epsilon(x,y)}{p_\epsilon^\alpha(x) p_\epsilon^\alpha(y)}, \text{ with } \alpha \in [0,1]$$

and $p_\epsilon^{(\alpha)}(x) = \int_{\mathcal{M}} k_\epsilon^{(\alpha)}(x,y) p(y) dy$.

Wanted $\lim_{\epsilon\to 0, n\to\infty} \frac{(T_\epsilon - I)}{\epsilon}$

This is the continuous limit of "diffusion maps"-type operators computed on graphs.

Limits of diffusion operators [Perrault-Joncas, M NIPS 11]

$$H_\epsilon = \lim_{\epsilon \to 0, n \to \infty} \frac{(T_\epsilon - I)f}{\epsilon}$$

Operators T_ϵ from α-renormalized, symmetric or asymmetric kernels

1. $H_{aa}^{(\alpha)}$: asymmetric $k_\epsilon^{(\alpha)}$ with assymmetric $p_\epsilon = \int_{\mathcal{M}} k_\epsilon(x,y)p(y)dy$
2. $H_{sa}^{(\alpha)}$: symmetric $h_\epsilon^{(\alpha)}$ with assymmetric p_ϵ
3. $H_{as}^{(\alpha)}$: asymmetric $k_\epsilon^{(\alpha)}$ with symmetric $q_\epsilon = \int_{\mathcal{M}} h_\epsilon(x,y)p(y)dy$
4. $H_{ss}^{(\alpha)}$: symmetric $h_\epsilon^{(\alpha)}$ with symmetric q_ϵ

$$\begin{aligned}
H_{aa}^{(\alpha)}[f] &= \Delta f - 2(1-\alpha)\nabla U \cdot \nabla f + \mathbf{r} \cdot \nabla f \\
H_{as}^{(\alpha)}[f] &= \Delta f - 2(1-\alpha)\nabla U \cdot \nabla f - cf + (\alpha - 1)(\mathbf{r} \cdot \nabla U)f - (\nabla \cdot \mathbf{r})f + \mathbf{r} \cdot \nabla f \\
H_{sa}^{(\alpha)}[f] &= \Delta f - 2(1-\alpha)\nabla U \cdot \nabla f + (c + \nabla \cdot r + (\alpha - 1)\mathbf{r} \cdot \nabla U)f \\
H_{ss}^{(\alpha)}[f] &= \Delta f - 2(1-\alpha)\nabla U \cdot \nabla f
\end{aligned}$$

Limits of diffusion operators [Perrault-Joncas, M NIPS 11]

$$H_\epsilon = \lim_{\epsilon \to 0, n \to \infty} \frac{(T_\epsilon - I)f}{\epsilon}$$

Operators T_ϵ from α-renormalized, symmetric or asymmetric kernels

1. $H_{aa}^{(\alpha)}$: asymmetric $k_\epsilon^{(\alpha)}$ with assymmetric $p_\epsilon = \int_{\mathcal{M}} k_\epsilon(x,y)p(y)dy$
2. $H_{sa}^{(\alpha)}$: symmetric $h_\epsilon^{(\alpha)}$ with assymmetric p_ϵ
3. $H_{as}^{(\alpha)}$: asymmetric $k_\epsilon^{(\alpha)}$ with symmetric $q_\epsilon = \int_{\mathcal{M}} h_\epsilon(x,y)p(y)dy$
4. $H_{ss}^{(\alpha)}$: symmetric $h_\epsilon^{(\alpha)}$ with symmetric q_ϵ

$\alpha = 1$

$$\begin{aligned}
H_{aa}^{(1)}[f] &= \Delta f - 2(1-1)\nabla U \cdot \nabla f + \mathbf{r} \cdot \nabla f \\
H_{as}^{(1)}[f] &= \Delta f - 2(1-1)\nabla U \cdot \nabla f - cf + (1-1)(\mathbf{r} \cdot \nabla U)f - (\nabla \cdot \mathbf{r})f + \mathbf{r} \cdot \nabla f \\
H_{sa}^{(1)}[f] &= \Delta f - 2(1-1)\nabla U \cdot \nabla f + (c + \nabla \cdot r + (1-1)\mathbf{r} \cdot \nabla U)f \\
H_{ss}^{(1)}[f] &= \Delta f - 2(1-1)\nabla U \cdot \nabla f
\end{aligned}$$

Limits of diffusion operators [Perrault-Joncas, M NIPS 11]

$$H_\epsilon = \lim_{\epsilon \to 0, n \to \infty} \frac{(T_\epsilon - I)f}{\epsilon}$$

Operators T_ϵ from α-renormalized, symmetric or asymmetric kernels

1. $H_{aa}^{(\alpha)}$: asymmetric $k_\epsilon^{(\alpha)}$ with assymmetric $p_\epsilon = \int_{\mathcal{M}} k_\epsilon(x,y)p(y)dy$
2. $H_{sa}^{(\alpha)}$: symmetric $h_\epsilon^{(\alpha)}$ with assymmetric p_ϵ
3. $H_{as}^{(\alpha)}$: asymmetric $k_\epsilon^{(\alpha)}$ with symmetric $q_\epsilon = \int_{\mathcal{M}} h_\epsilon(x,y)p(y)dy$
4. $H_{ss}^{(\alpha)}$: symmetric $h_\epsilon^{(\alpha)}$ with symmetric q_ϵ

$\alpha = 1$

$$H_{aa}^{(1)}[f] = \Delta f + \mathbf{r}\cdot\nabla f \quad \text{new operator, contains } \mathbf{r}$$

$$H_{ss}^{(1)}[f] = \Delta f \quad \text{Diffusion maps}$$

Isolating the Vector Field $\mathbf{r}$

Coordinate free

$$H_{aa}^{(\alpha)} - H_{ss}^{(\alpha)} = \mathbf{r} \cdot \nabla$$

Coordinate Representation of $\mathbf{r}$

- Let Φ be an diffeomorphic embedding of $\mathcal{M}$
- Then
$$\mathbf{r}_{||} = \mathbf{r} \cdot \nabla \phi \,,$$
- and component of $\mathbf{r}$ along coordinate ϕ_k is $\mathbf{r} \cdot \nabla \phi_k$
- Note that only $\mathbf{r}_{||}$ is recovered

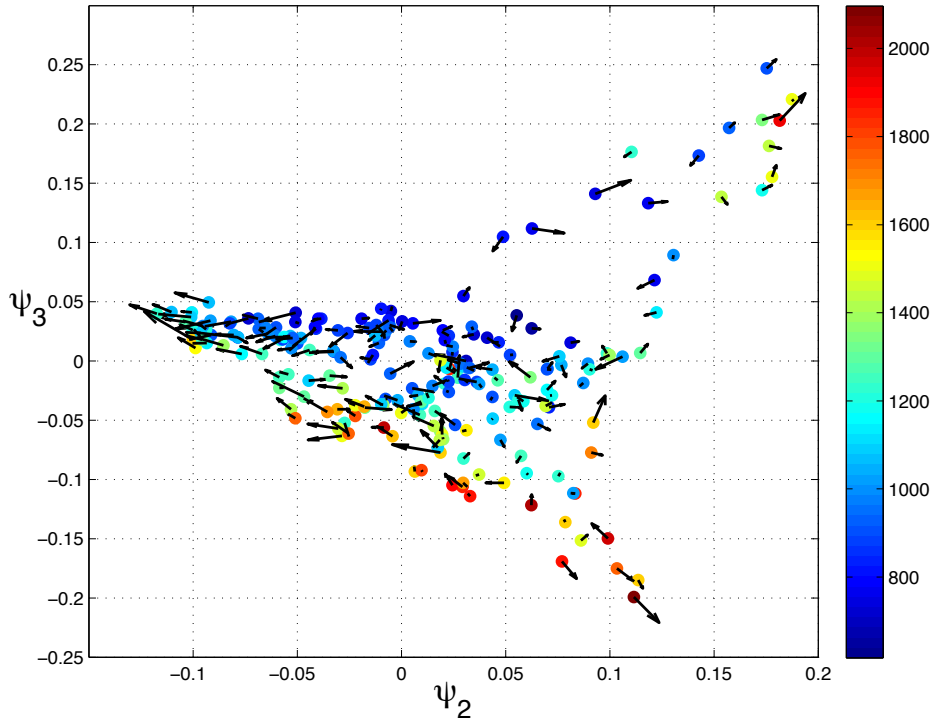
Real Data

- Source: National Longitudinal Survey of Youth (NLSY)
- Aim: obtain a representation of the job market as a diffusion process over a manifold.
- Data: Sample of 7,816 individual career sequences of length 64, listing the jobs a particular individual held every quarter between the ages of 20 and 36.
- Our graph G has 213 nodes - industry/occupation pairings
- Our observations consist of 7,816 walks between the 213 graph nodes.

Real Data

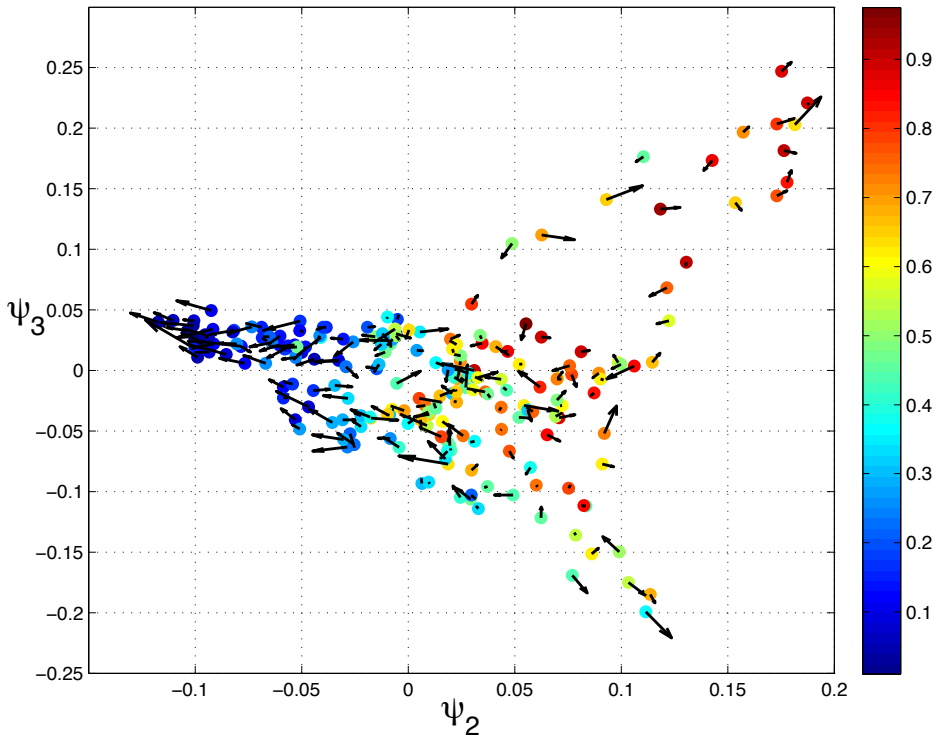
- Walks are converted to a directed graph with *affinity matrix* W.
- W_{ij}: number of times a transition from job i to job j was observed
- Normalizing each row i of W by its outdegree d_i gives $P = \text{diag}(d_i)^{-1} W$, the non-parametric MLE for the Markov chain over G for the progression of career sequences.
- This Markov chain has as limit operator $H_{aa}^{(0)}$.
- We want to estimate $\mathbf{r} \cdot \nabla - 2\nabla U \cdot \nabla$ where we can use $-2\nabla U \cdot \nabla = H_{ss}^{(0)} - H_{ss}^{(1)}$ to complement our algorithm.

Real Data



Embedding the job market along with field $\mathbf{r} - 2\nabla U$ over the first two non-constant eigenvectors. The color map corresponds the mean monthly wage of each job in dollars.

Real Data



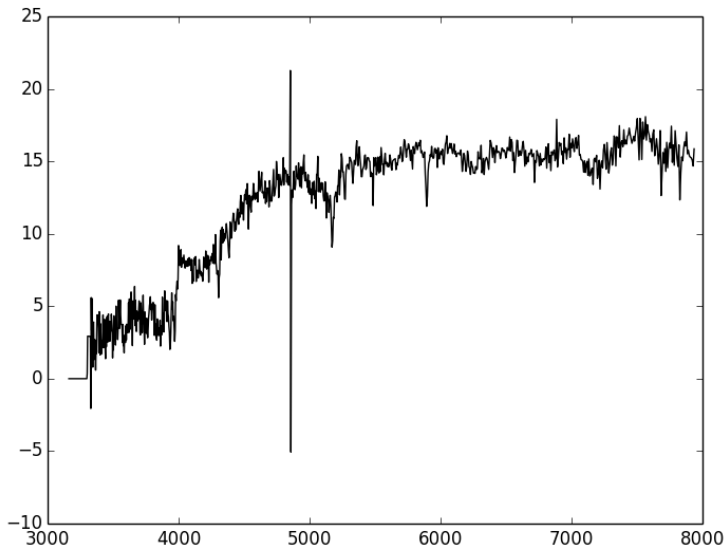
Embedding of the job market along with field $\mathbf{r} - 2\nabla U$ over the first two non-constant eigenvectors. The color map corresponds the gender proportion in each job (with male $= 0$ and female $= 1$).

Outline

Outline

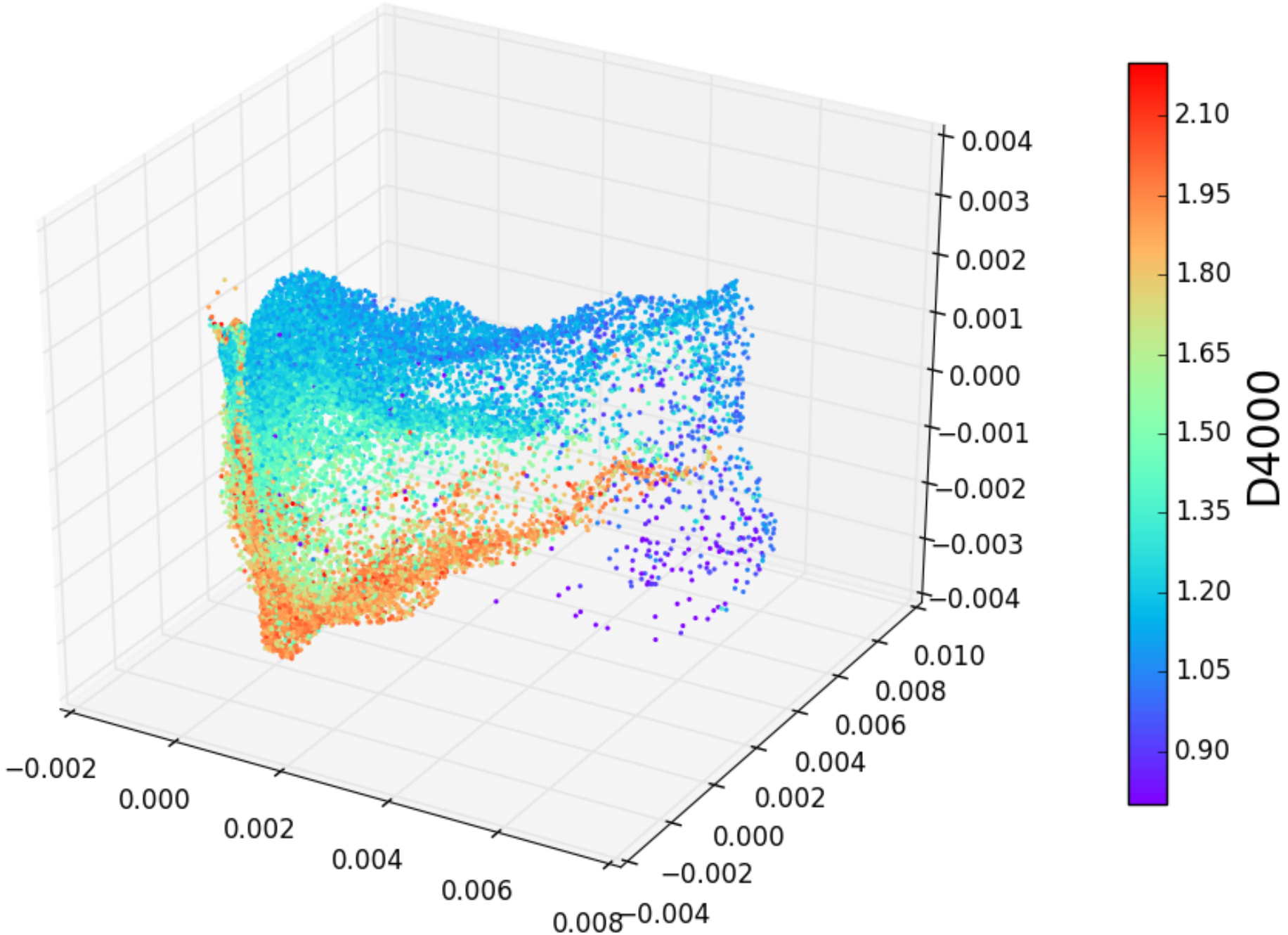
Manifold learning for SDSS Spectra of Galaxies

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



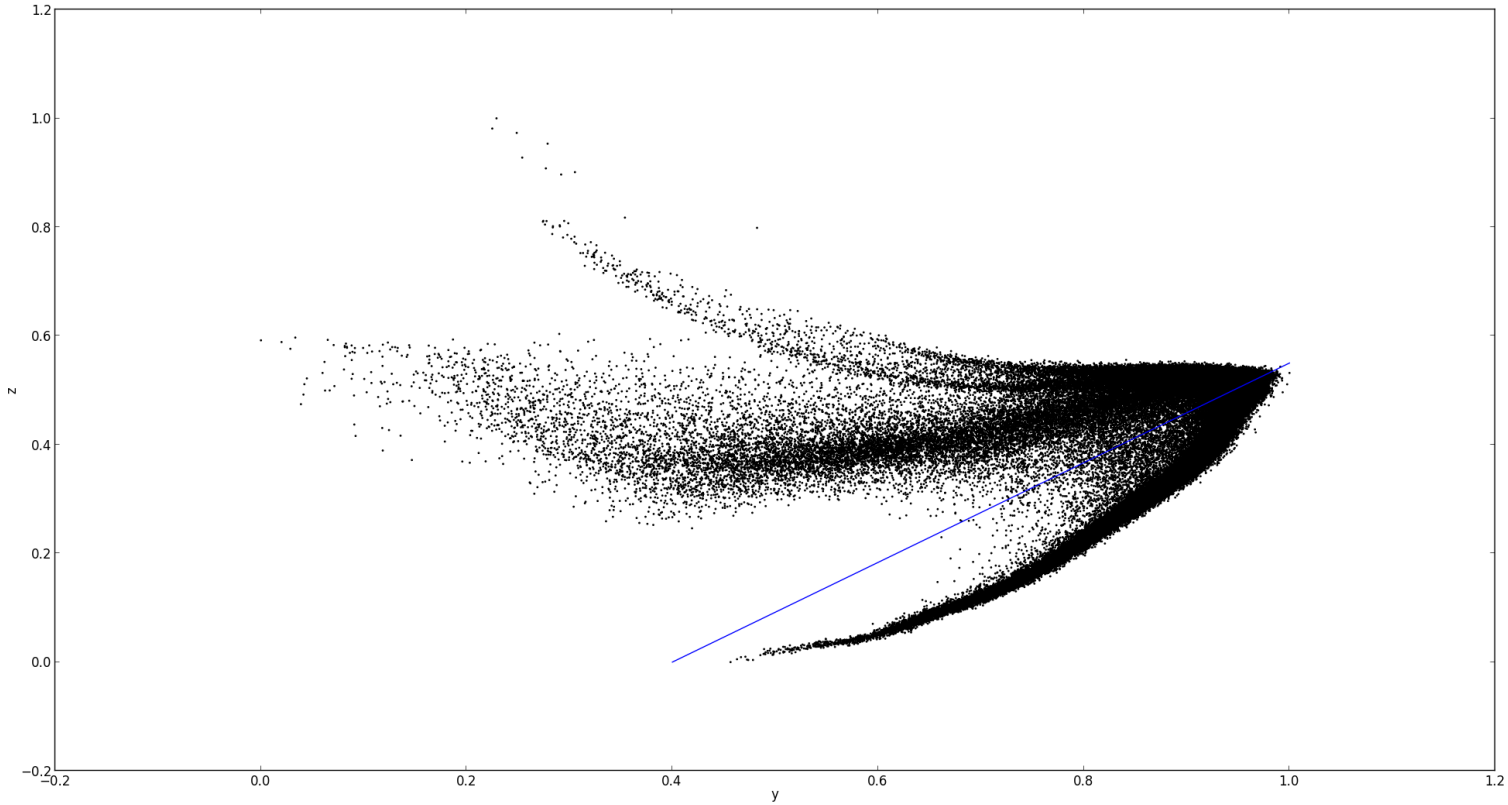
- data curated by Grace Telford,
- "noise removal" by Jake VanderPlas

Embedding into 3 dimensions



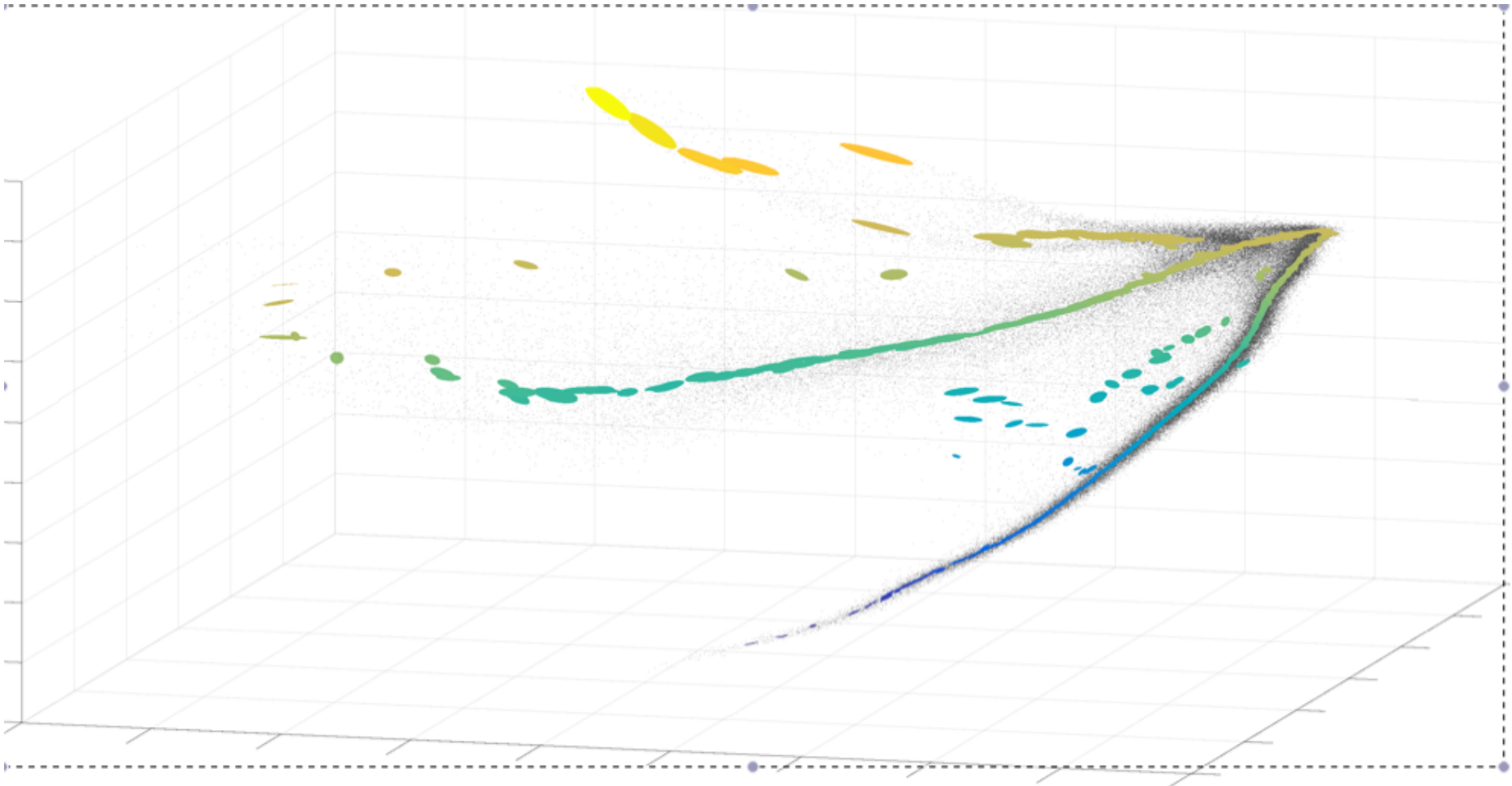
Same embedding. . .

- only high density regions
- another viewpoint



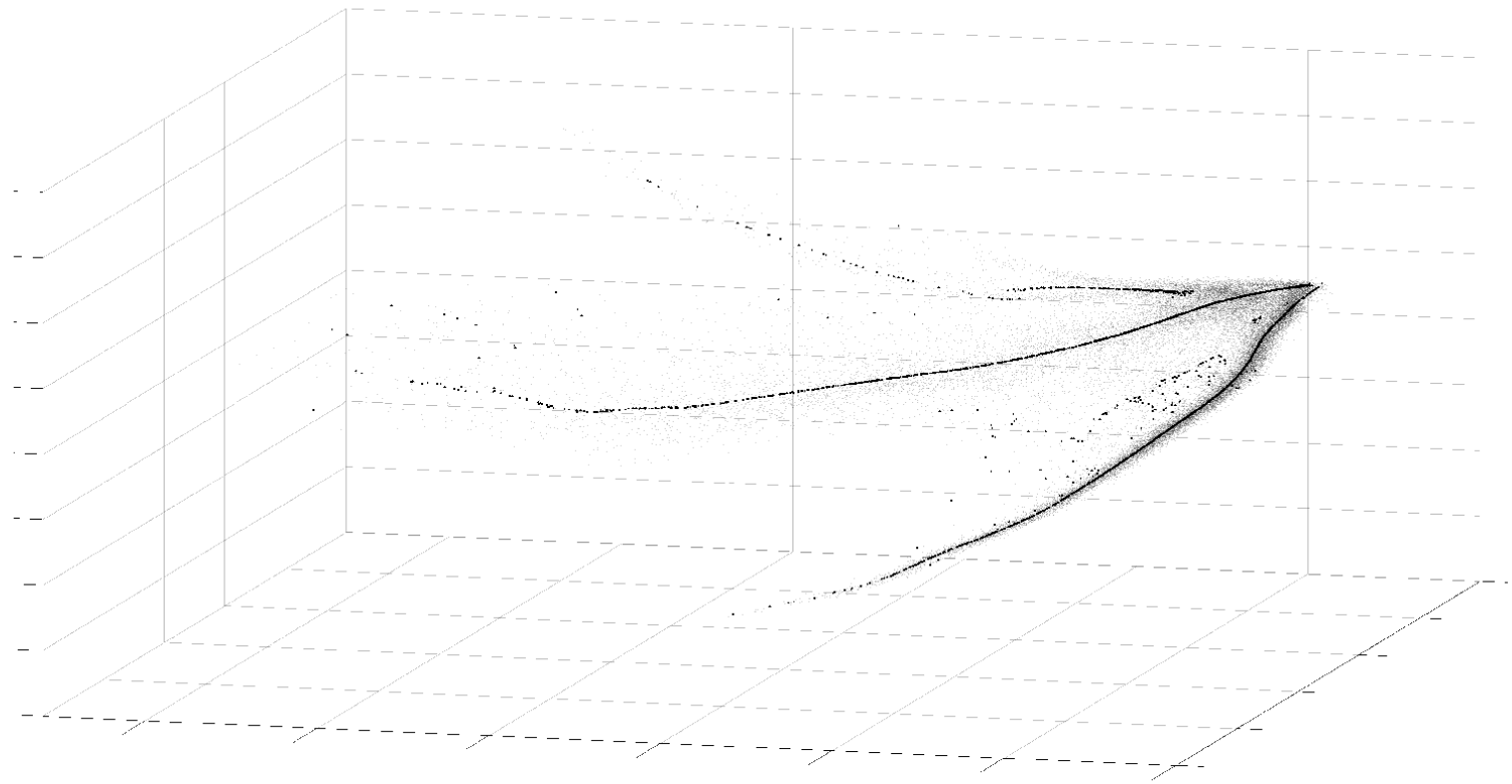
- how distorted is this embedding?

How distorted is this embedding?



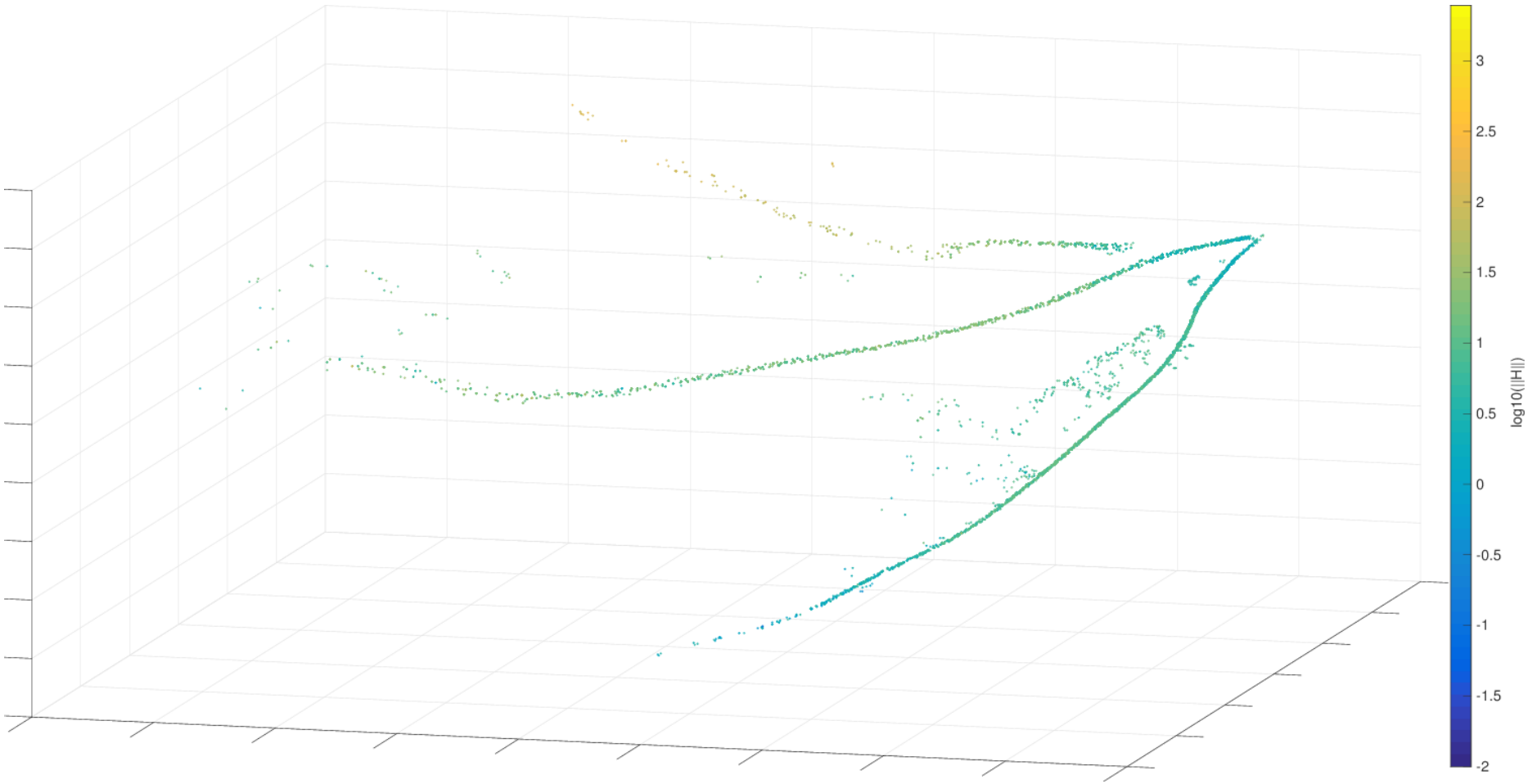
(ellipses represent G_i^{-1})

Riemannian Relaxation along principal curves



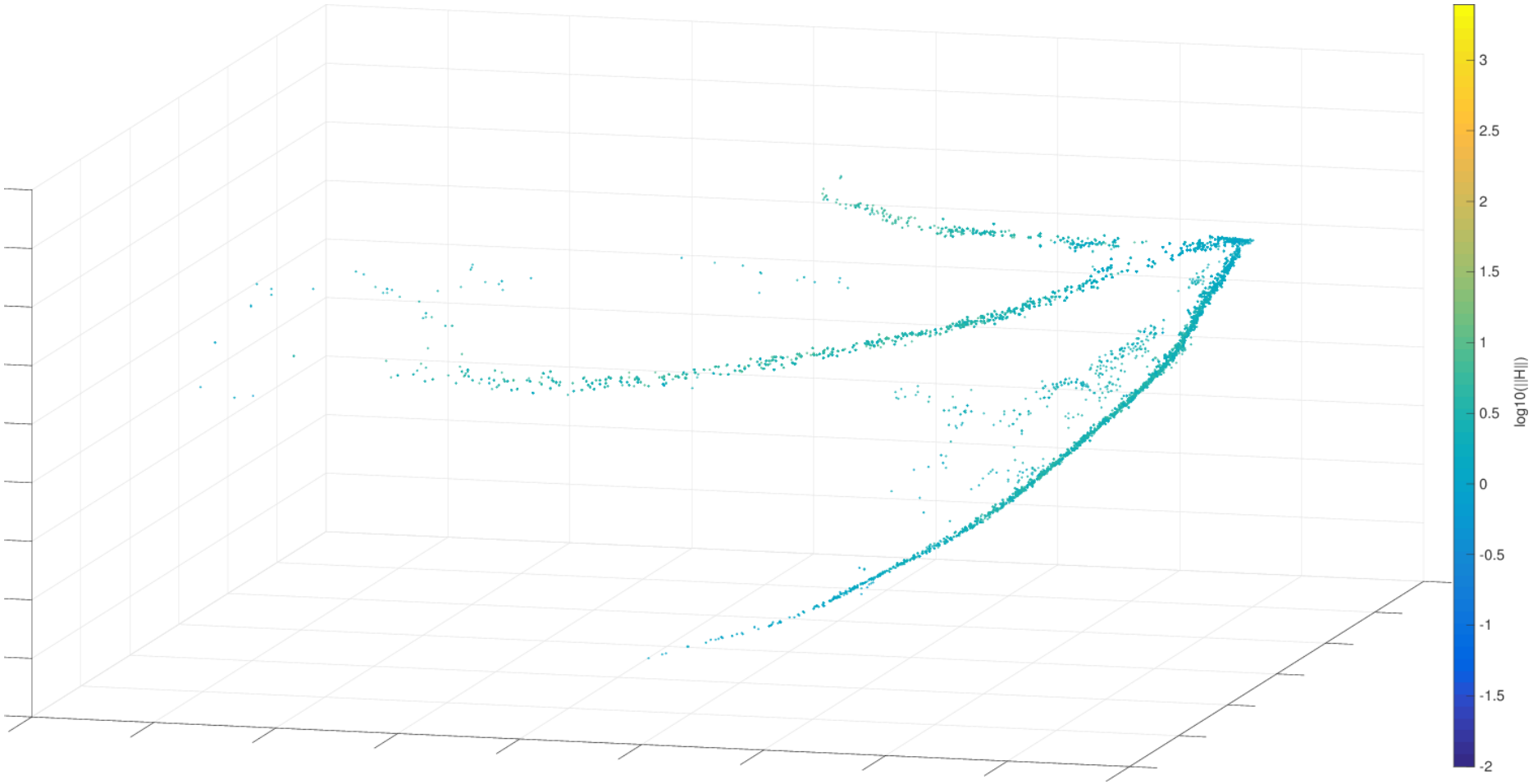
Find principal curves

Riemannian Relaxation along principal curves



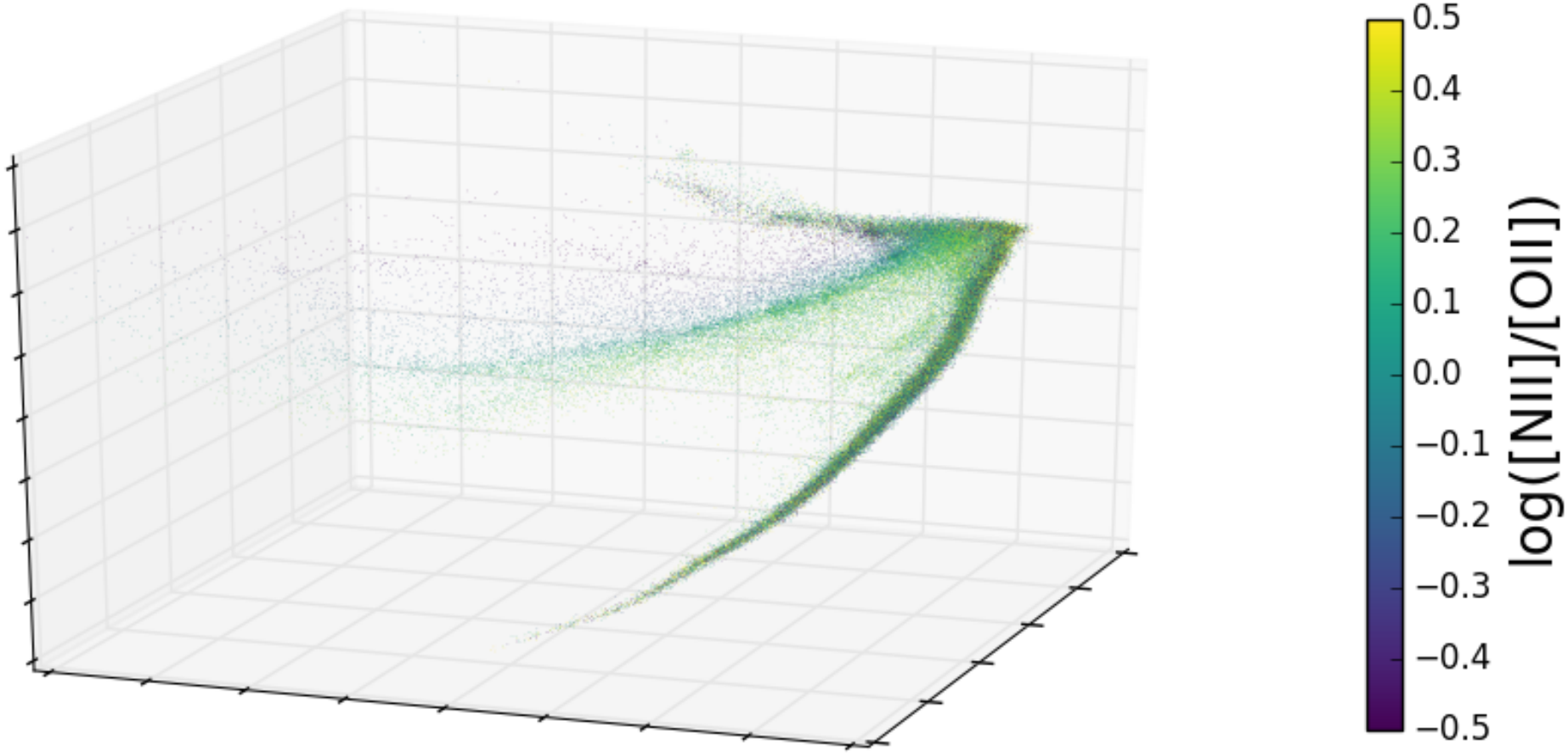
Points near principal curves, colored by $\log_{10}(G_i)$ (0 means no distortion)

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



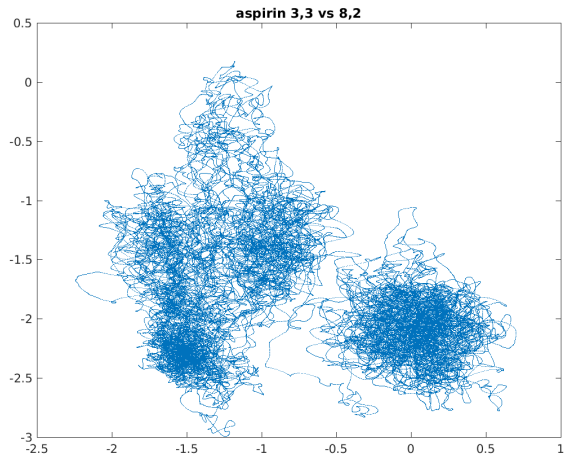
All data after Riemannian Relaxation

Outline

Data and preprocessing

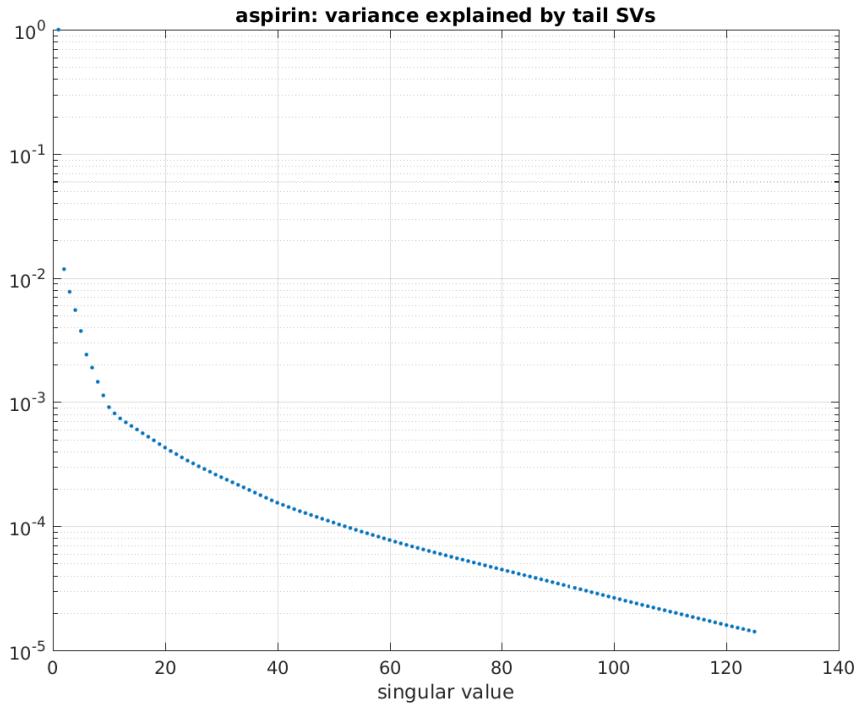
- Data

  simulations of Aspirin ($C_9H_8O_4$) molecular dynamics at $T = 500K$ by Stefan Chmiela and Alexandre Tkatchenko
  - atoms locations $R_{1:N} \in \mathbb{R}^{21\times 3}$
  - n =210,000 states
- Computed $M_0 = 624$ angles between selected atomic triplets
- Selected $D = 50$ input features by SVD



SVD

- SVD was used to eliminate all linear relations between the $M_0 = 624$ angles.



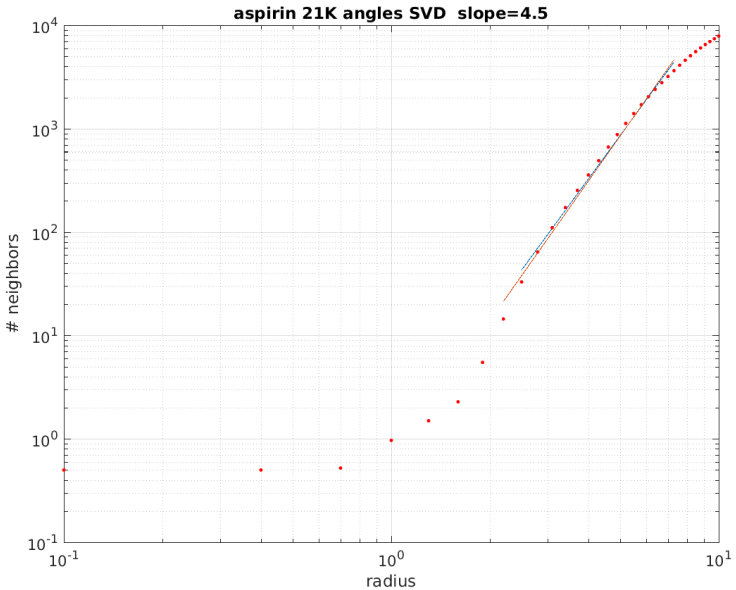
Note that the first component explains almost 99% of variance.

$D = 50$ principal components were kept. The residual variance is $< 10^{-4}$

- From here on, the data was subsampled by a factor of 12.

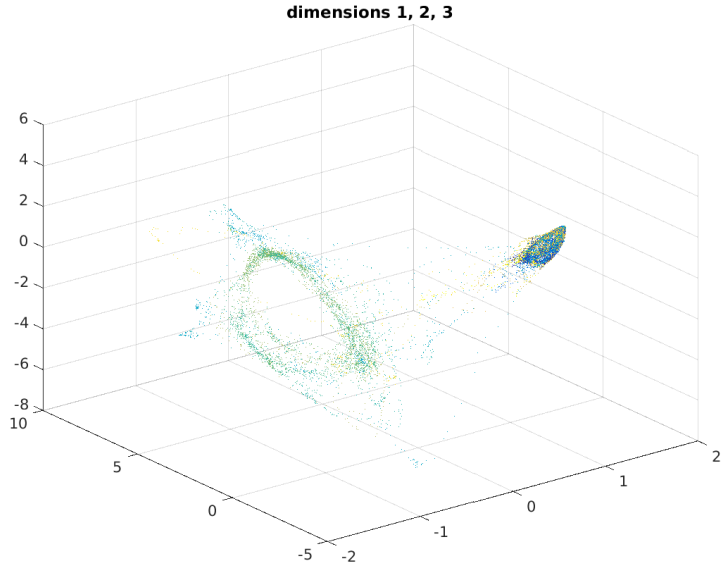
Choosing output dimensions and kernel radius

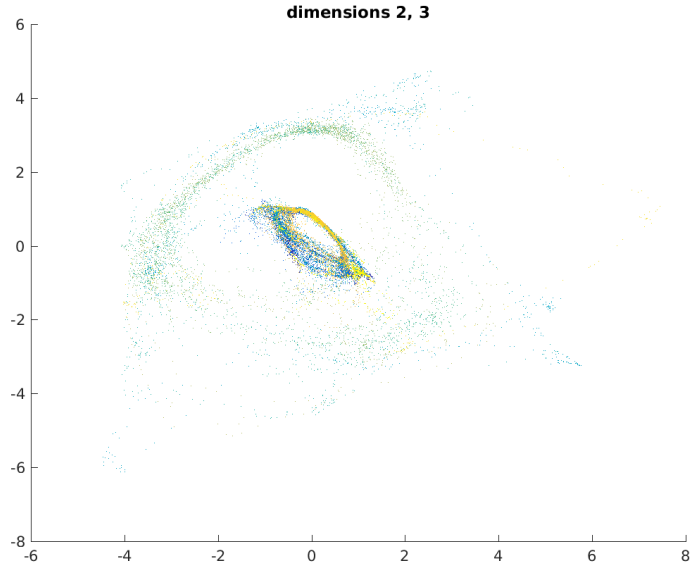
Data in $D = 50$ coordinates, $U\Sigma$



This plot suggests $d = 5$ therefore we embed into $m = 9$ dimensions.

Embedding 17K configurations

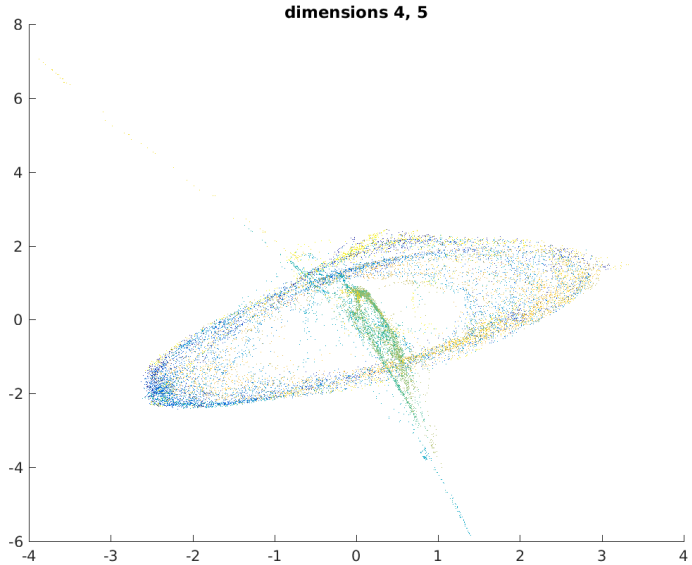


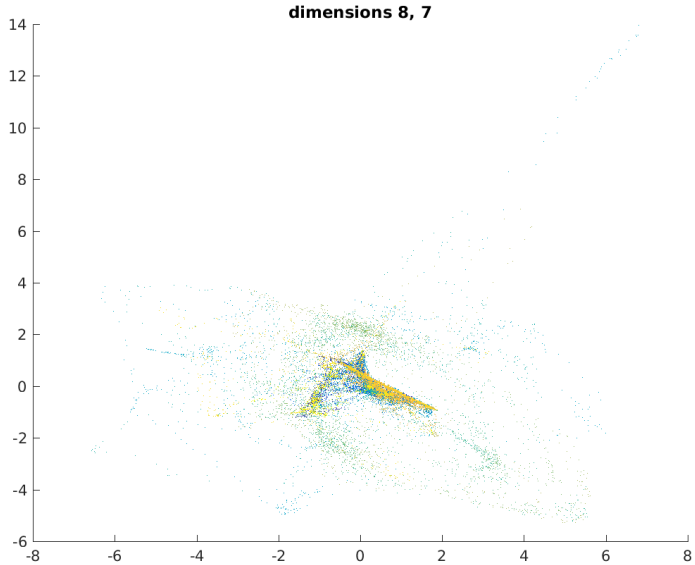


Points are colored sequentially

$m = 9$ embedding dimension, $n = 17,000$ (every 12th configuration), method Laplacian Eigenmaps/Diffusion Maps

Embedding for 17,000 configurations

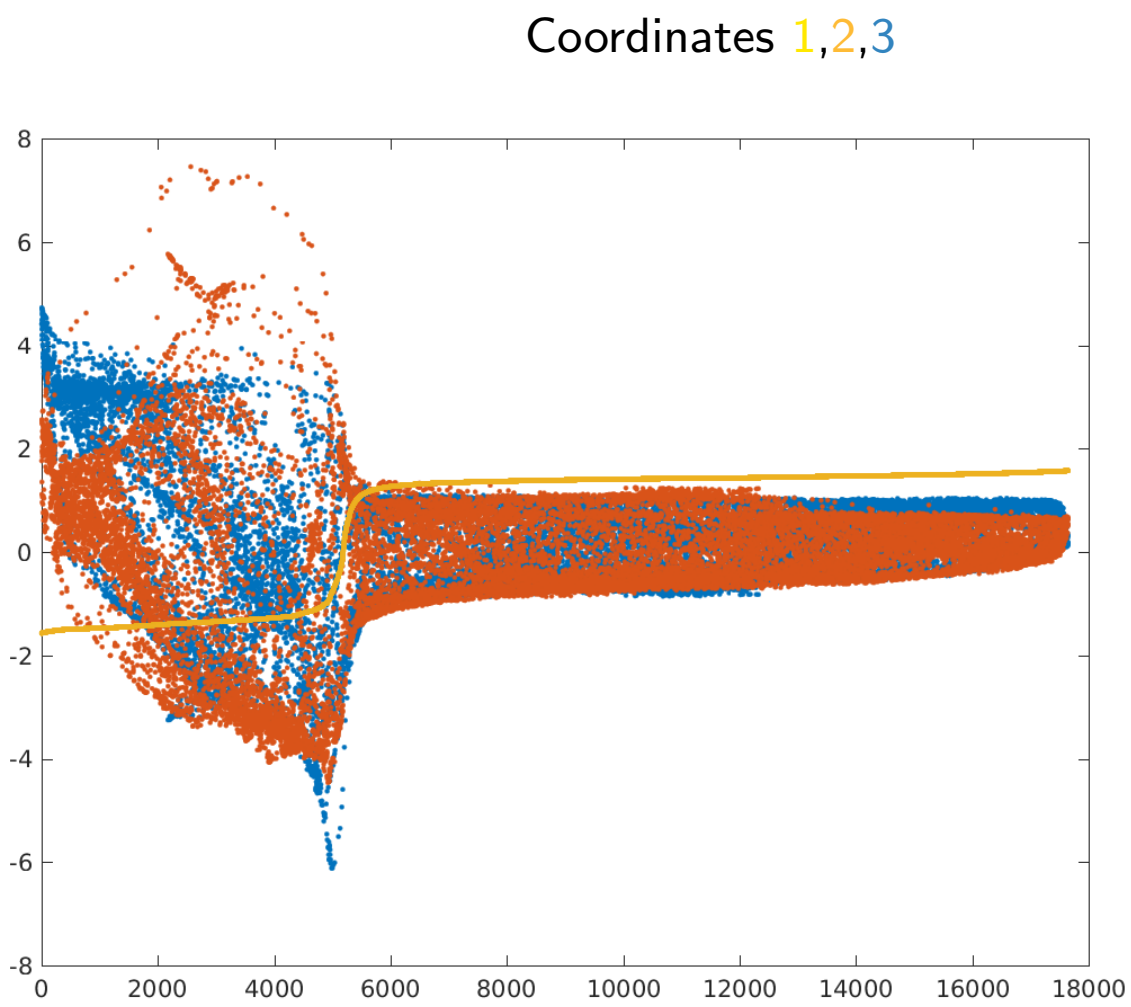




Ploints in the embedding are colored sequentially

$m = 9$ embedding dimension, $n = 17,000$ (every 12th configuration), method Laplacian Eigenmaps/Diffusion Maps

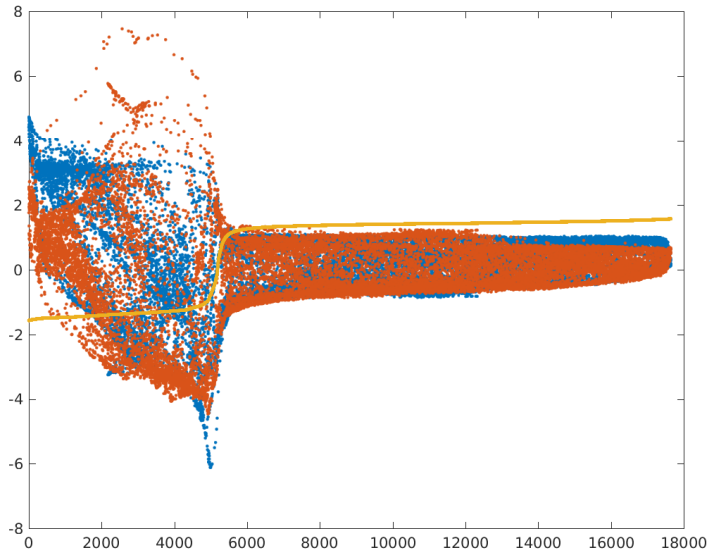
Embedding coordinates, sorted by coordinate 1



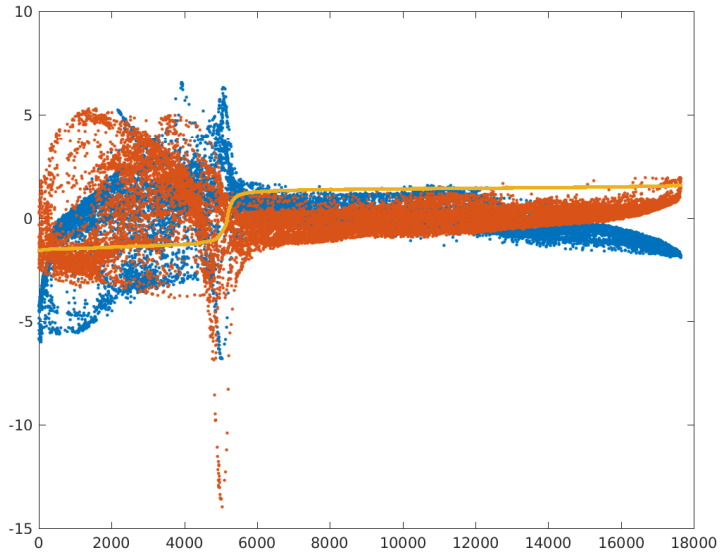
- Coordinate 1 shows 2 clusters in the data
- Coordinates 2, 3 describe metastable state (cluster 2)

Embedding coordinates, sorted by coordinate 1

Coordinates 1,2,3



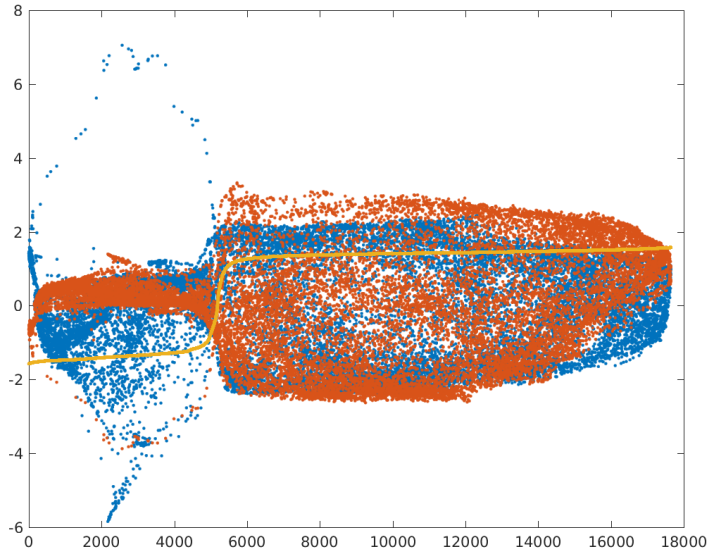
Coordinates 1,7,8



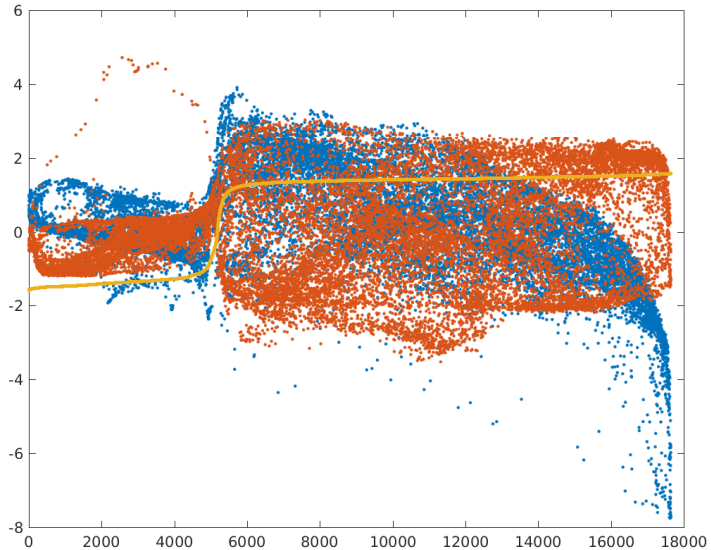
- Coordinate 1 shows 2 clusters in the data
- Coordinates 2, 3 describe metastable state (cluster 2)

Embedding coordinates, sorted by coordinate 1

Coordinates 1,4,5



Coordinates 1,6,9

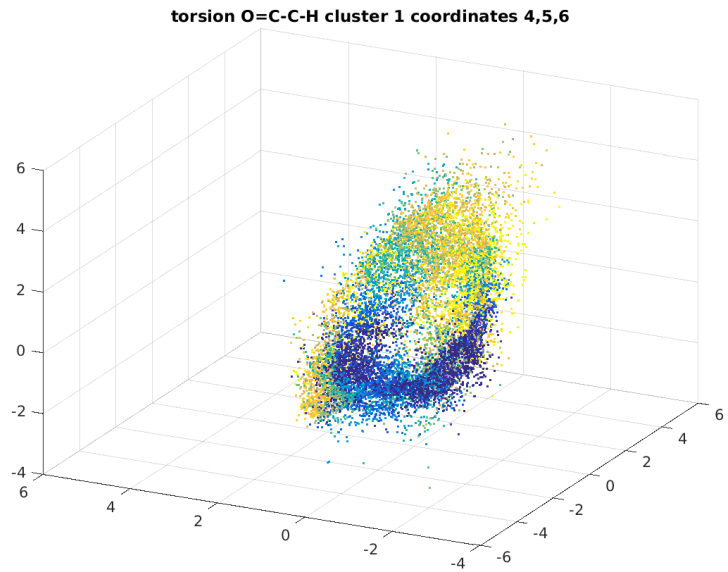


- Coordinate 1 shows 2 clusters in the data
- Coordinates 2, 3 describe metastable state (cluster 1)
- Coordinates 4,5,6,9 describe metastable state (cluster 2)

O=C–C–H torsion

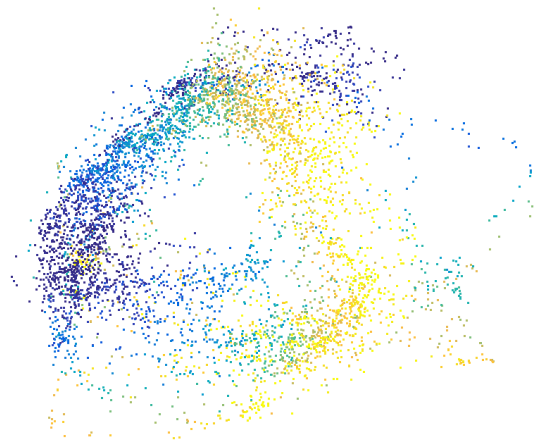
Stable state

coordinates 4, 5, 6



Meta-stable state

coordinates 2, 3



cosine of torsion τ

Manifold learning should be like PCA

- tractable
- "automatic" – quantitative measures of success/accuracy
- first step in data processing pipe-line

Manifold learning should be like PCA

- tractable
- "automatic" – quantitative measures of success/accuracy
- first step in data processing pipe-line

Metric Manifold learning

- Before embedding: choice of kernel width ϵ, choice of intrinsic dimension d
- After embedding: estimate distortion by H and correct it by Riemannian Relaxation
- Simultaneously with embedding: Gaussian process prediction, estimating vector fields (coordinate free)

Future

- input data not i.i.d, side information (e.g forces, potential), on-line
- connect with topological data analysis

Python package `megaman`

- tractable for millions of points, incorporates
- (in progress) quantitative validation (topology preservation, choice of ϵ, choice of d), Principal Curves and Surfaces
- future: extend classification, regression, clustering to the manifold setting

Dominique-Perrault Joncas, James McQueen (PhD Statistics)
Jacob VanderPlas, Grace Telford (UW Astronomy)
Oles Isayev, Alexandre Tkatchenko, Stefan Chmiela
Sadas Shankar, Ralf Banisch, Stefan Tautz, Klaus Mueller, Christian Ratsch

Thank you