Manifold Learning for Scientists

Marina Meilă

University of Washington
mmp@stat.washington.edu

MLP2019 Tutorials

Outline

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?

Supervised vs. Unsupervised Learning

Supervised Learning

Prediction

Data pairs (x_i, y_i) from unknown distribution P_{XY}

Goal learn to predict output y from input x

in probabilistic terms, learn $P_{Y|X}$

Clearly defined objective: minimize cost of error in predicting y

Unsupervised Learning

Description / Analysis / Exploration / Modeling

Data observations x_i from unknown distribution P_X

Goal learn [something about] P_X

- Defining the objective is part of the problem
- Sometimes query or goal not known at the time of learning
- Objective can be defined in multiple ways

Example: Graphical models

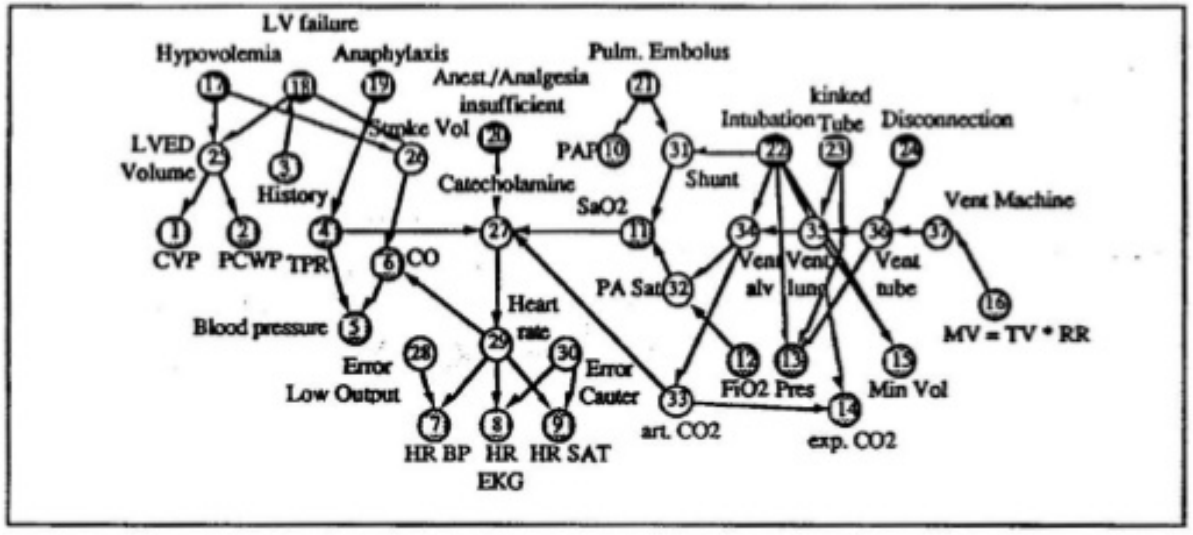
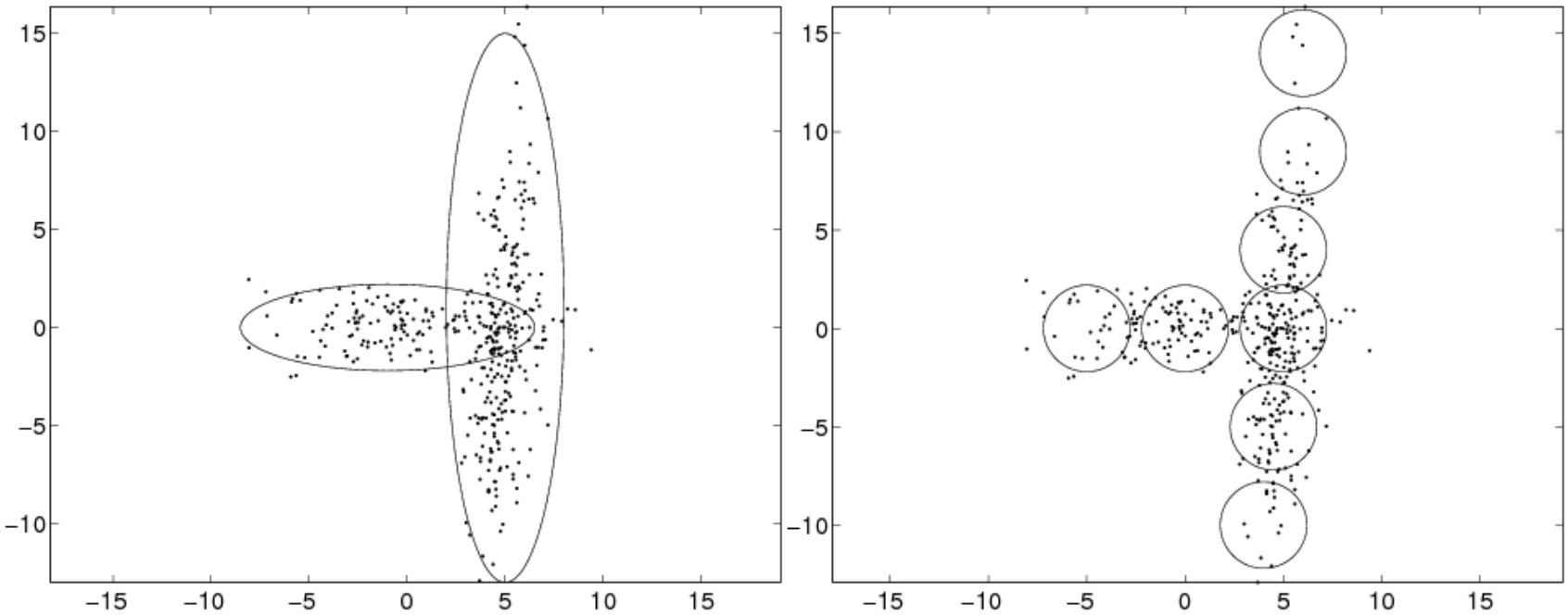


Fig. 1 The ALARM network representing causal relationships is shown with diagnostic (●), intermediate (○) and measurement (◎) nodes. CO: cardiac output, CVP: central venous pressure, LVED volume: left ventricular end-diastolic volume, LV failure: left ventricular failure, MV: minute ventilation, PA Sat: pulmonary artery oxygen saturation, PAP: pulmonary artery pressure, PCWP: pulmonary capillary wedge pressure, Pres: breathing pressure, RR: respiratory rate, TPR: total peripheral resistance, TV: tidal volume

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine, pages 247-256. Springer-Verlag, 1989

- Graphical model describes joint probabilistic dependence of all variables
- After it is learned, we can ask a variety of questions
  - What is the probability distribution of Pres is high if Kinked Tube is True?
  - What is the most probable failure if PA Sat is observed (and out of limits)?

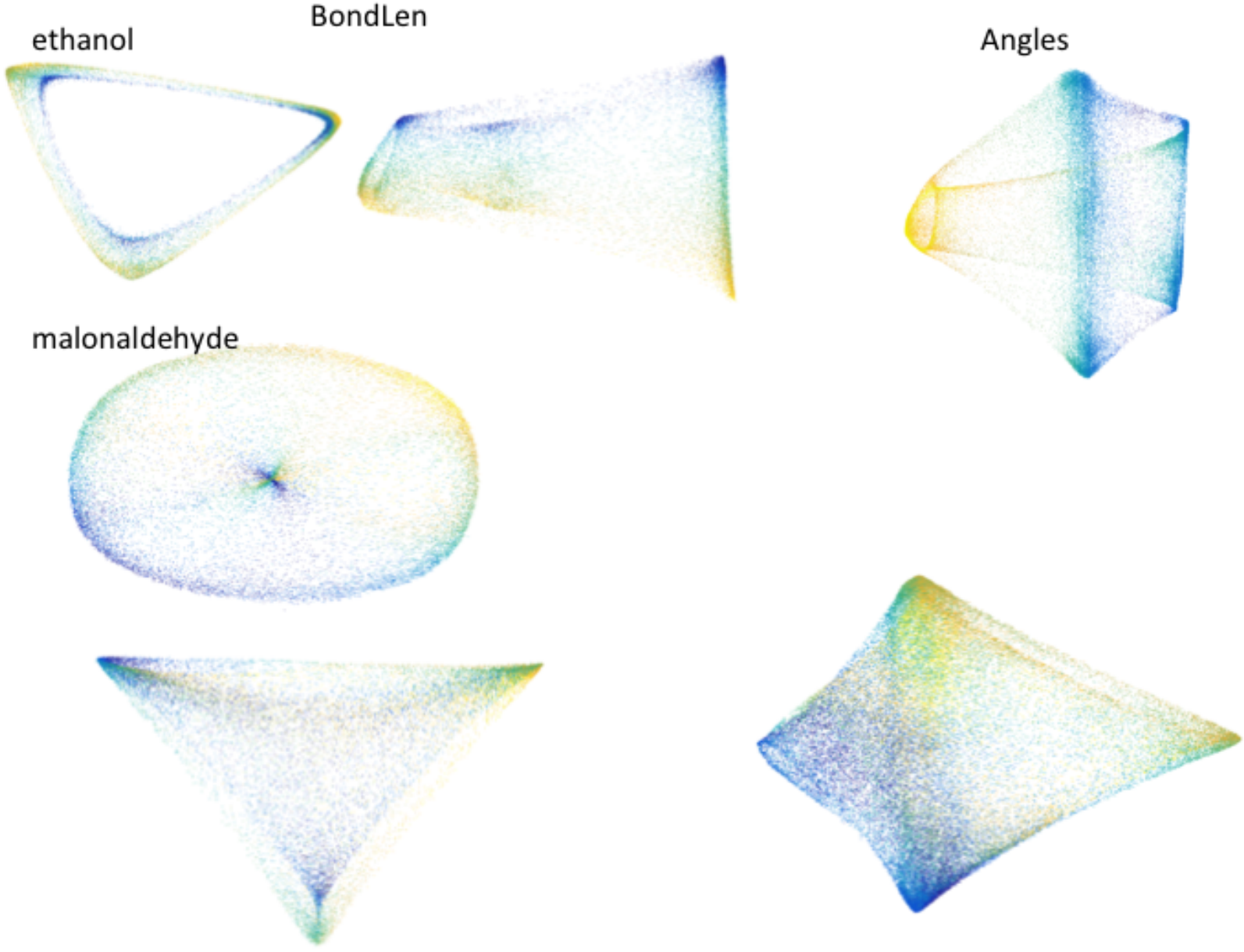
Example: Clustering



- These data support multiple definitions for clusters)can be grouped in several ways)

Example: Dimension reduction

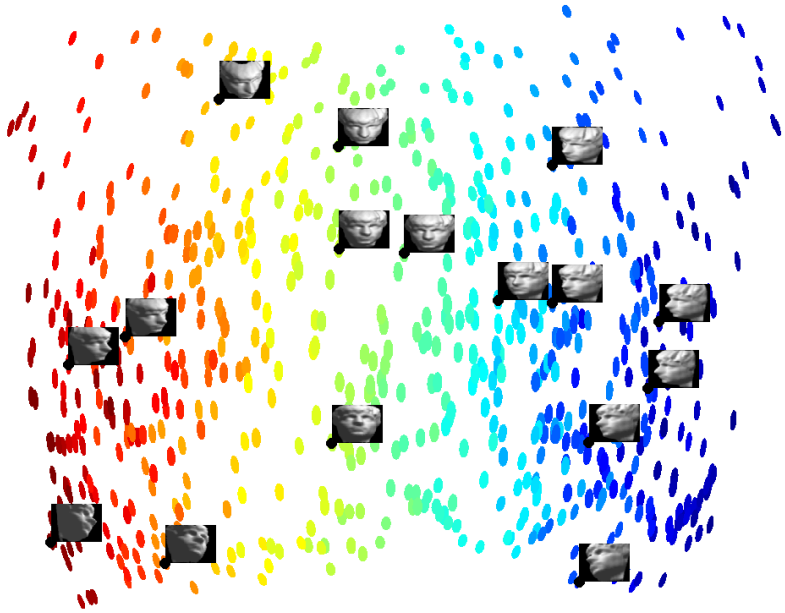
Manifolds from MD simulations for small molecules



[1] [Chmiela et al.16]

Outline

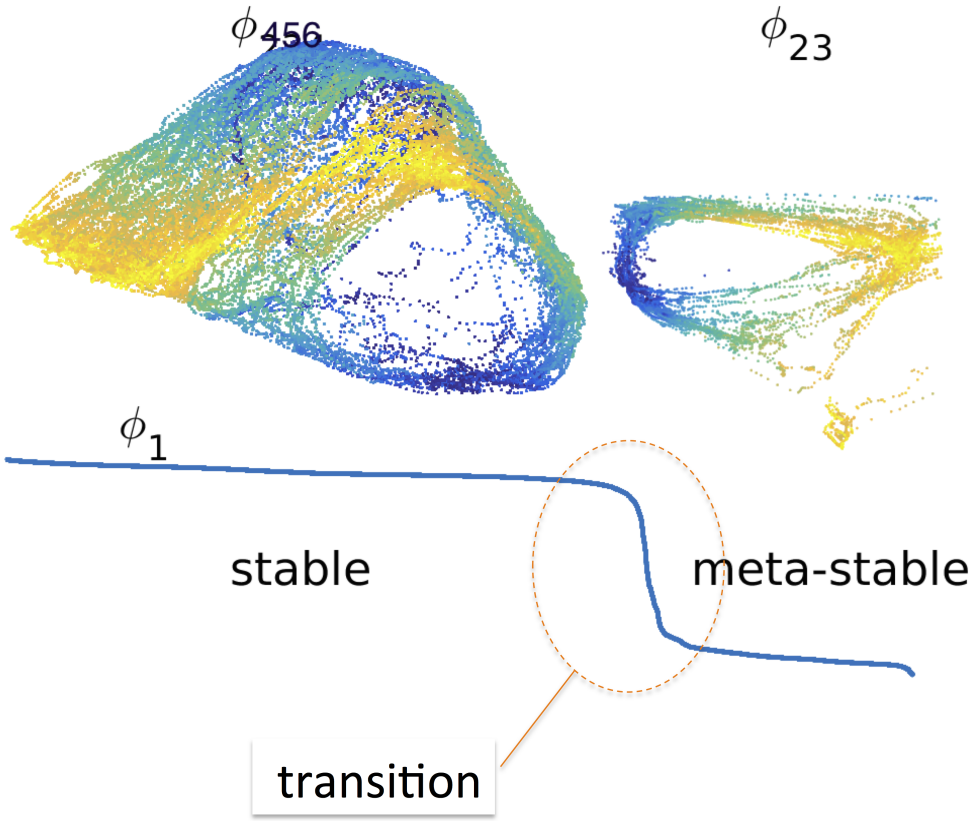
When to do (non-linear) dimension reduction



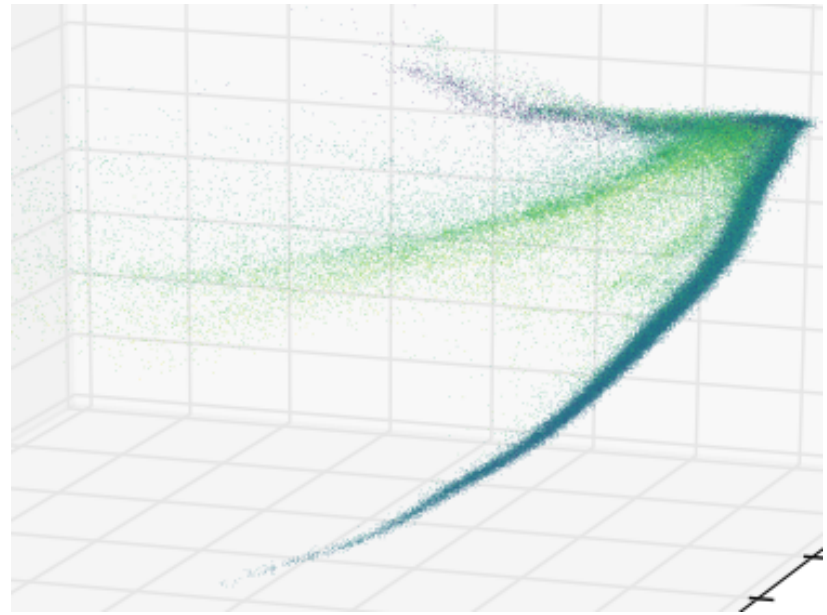
- high-dimensional data $p \in \mathbb{R}^D$, $D = 64 \times 64$
- can be described by a small number d of continuous parameters (order parameters)
- Usually, large sample size n

When to do (non-linear) dimension reduction

aspirin MD simulation



SDSS galaxy spectra

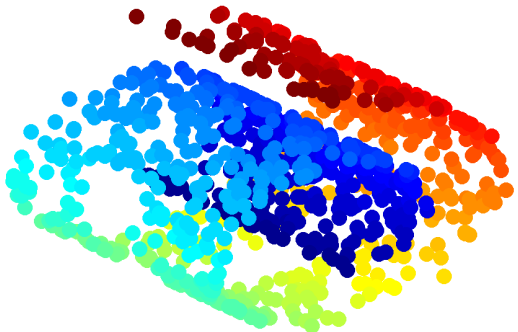


Why?

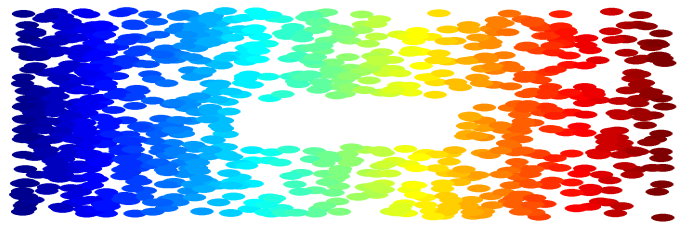
- To save space and computation
  - $n \times D$ data matrix $\rightarrow$ $n \times s$, $s \ll D$
- To use it afterwards in (prediction) tasks
- To understand the data better
  - preserve large scale features (slow modes), suppress fine scale features

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

points in $D \geq 3$ dimensions

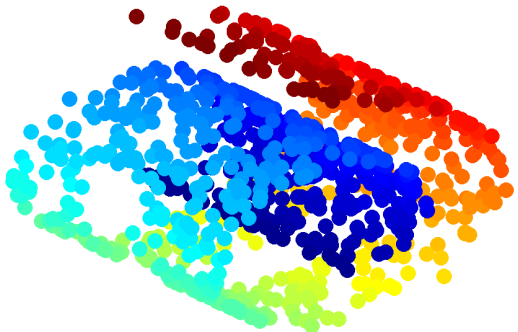


same points reparametrized in 2D



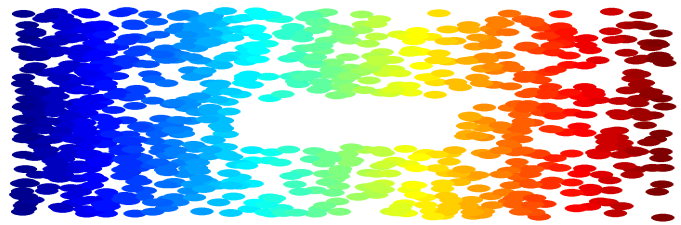
A toy example (the "Swiss Roll" with a hole)

points in $D \geq 3$ dimensions



Input

same points reparametrized in 2D



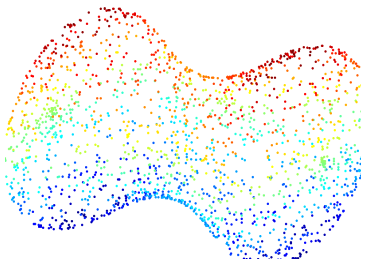
Desired output

Outline

Brief intro to manifold learning algorithms

ALL ML Algorithms

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

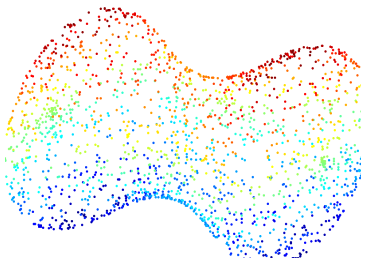


$p_1, \ldots p_n \subset \mathbb{R}^D$

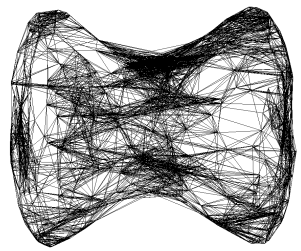
Brief intro to manifold learning algorithms

ALL ML Algorithms

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



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



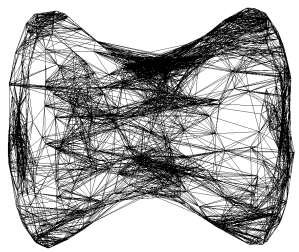
Brief intro to manifold learning algorithms

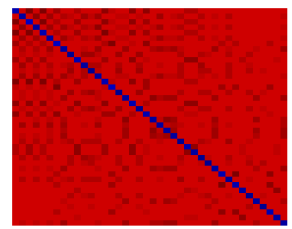
ALL ML Algorithms

- **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 $n \times n$ sparse distance matrix

$$D = [||p - p'||_{p,p' \text{neighbors}}]$$

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





Brief intro to manifold learning algorithms

ALL ML Algorithms

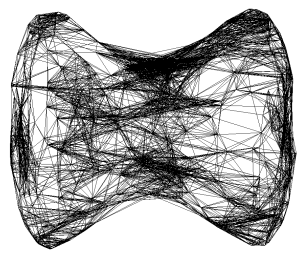
- **Input** Data $p_1, \ldots p_n$, embedding dimension m, neighborhood scale parameter ϵ
- Construct neighborhood graph p, p' neighbors iff $||p - p'||^2 \leq \epsilon$
- Construct a $n \times n$ sparse distance matrix

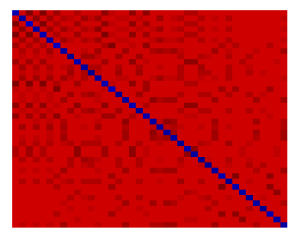
$$D = [||p - p'||_{p,p' \text{neighbors}}]$$

- Optional: construct similarity matrix (or kernel matrix), .e.g

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

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





Isomap ML algorithm

ISOMAP [Tennenbaum, deSilva & Langford 00]

1. Find all shortest path distances in neighborhood graph
2. Construct matrix of distances

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

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

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

Input k number nearest neighbors, d,

1. Find k-nearest neighbors
2. Construct (asymmetric) similarities w_{ij}, so that $\sum_j w_{ij} = \log_2 k$. $W = [w_{ij}]$.
3. Symmetrize $S = W + W^T - W .* W^T$ is similarity matrix.
4. Initialize embedding ϕ by LaplacianEigenmaps.
5. Optimize embedding.

   Iteratively for n_{iter} steps

   5.1 Sample an edge ij with probability $\propto \exp -d_{ij}$
   5.2 Move ϕ_i towards ϕ_j
   5.3 Sample a random j' uniformly
   5.4 Move ϕ_i away from $\phi_{j'}$
   Stochastic approximate logistic regression of $||\phi_i - \phi_j||$ on d_{ij}.

Output ϕ

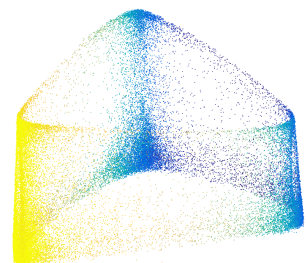
Non-linear embedding by Diffusion Maps (DM)

Diffusion Maps Algorithm

Input coordinates $U \in \mathbb{R}^{n \times D}$, bandwidth h, embedding dimension s

1. Compute Laplacian $L \in \mathbb{R}^{n \times n}$
2. Compute eigenvectors of L for smallest $s+1$ eigenvalues $[\phi_0\, \phi_1\, \ldots \phi_s] \in \mathbb{R}^{n \times s}$
   - These are the slow modes of the system
   - ϕ_0 is constant and not informative

The embedding coordinates of $U_{i:}$ are $(\phi_{i1}, \ldots \phi_{is})$



- Embedding dimension s = number of eigenvectors
- Intrinsic dimension $d \leq s$ effective number of collective variables

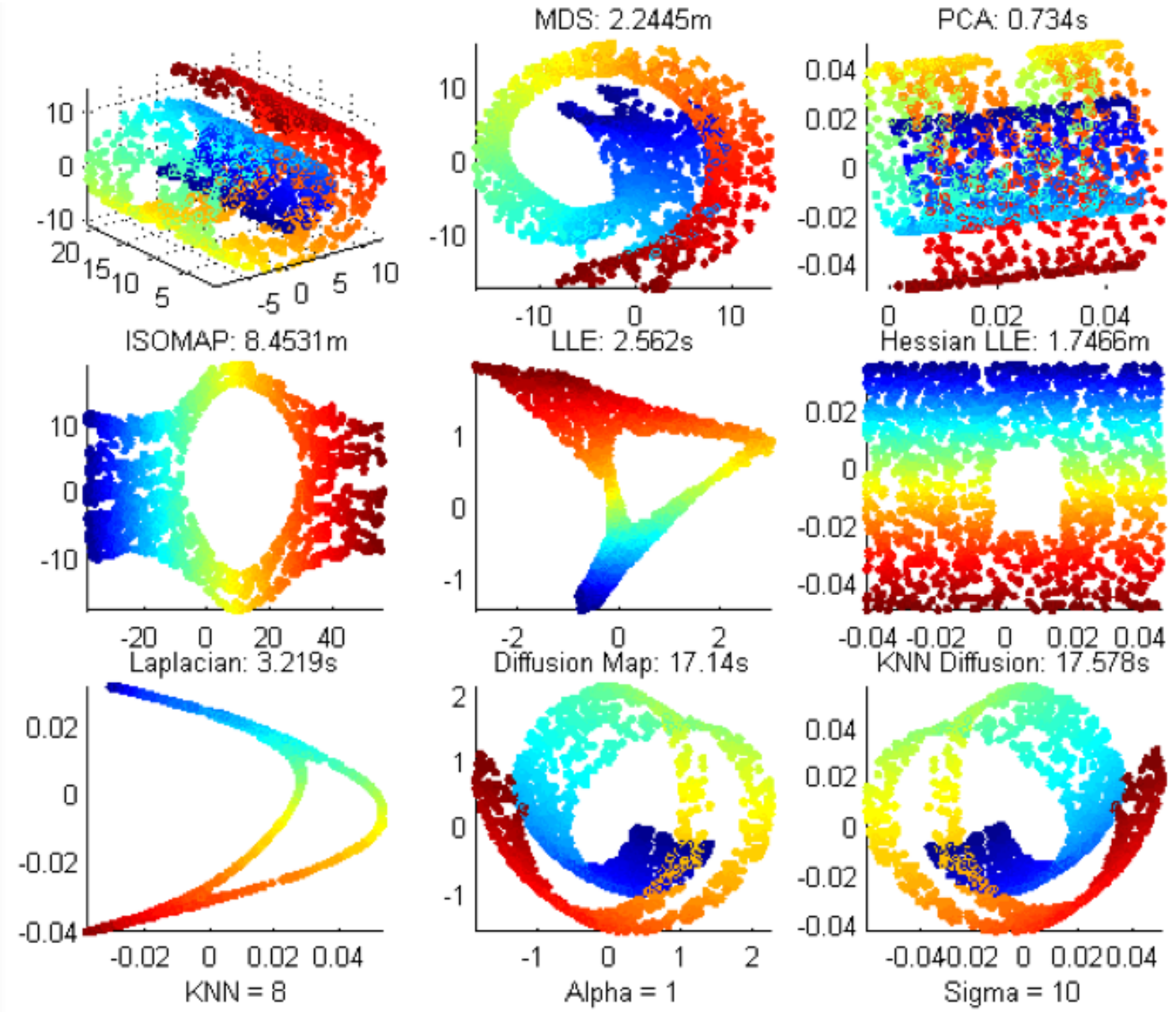
Laplacian

Input coordinates $U \in \mathbb{R}^{n \times D}$, kernel width h

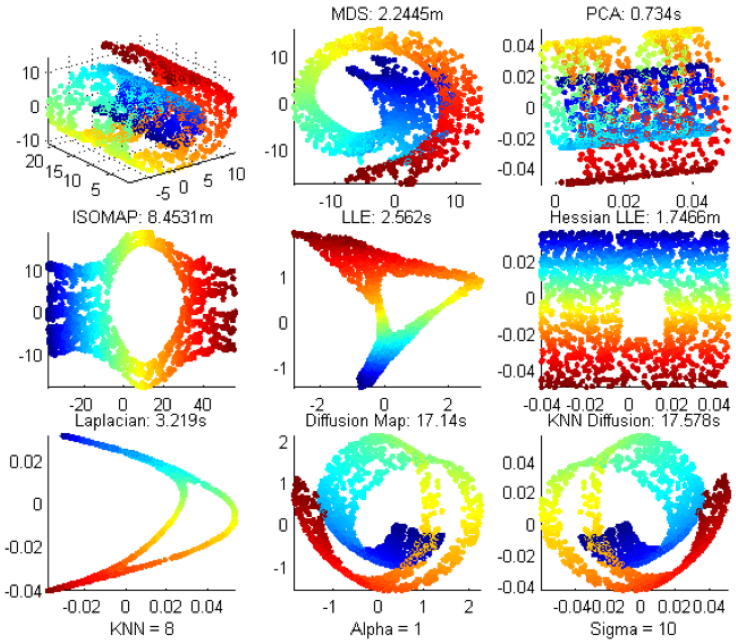
1. Compute kernel matrix $S_{ij} = \exp\left[-\frac{||U_{i:} - U_{j:}||^2}{h^2}\right]$
2. First normalization $d_i = \sum_{j=1}^{n} S_{ij}$, $\tilde{L}_{ij} = L_{ij}/\sqrt{d_i d_j}$
3. Second normalization $d'_i = \sum_{j=1}^{n} \tilde{L}_{ij}$, $L_{ij} = \tilde{L}_{ij}/d'_i$ removes the biases due to sampling density
4. Output L, d'_{in}

- Laplacian L central to understanding the manifold geometry
- S_{ij} can be replaced by other kernels (e.g. Coulomb matrix, SLATM kernel, SOAP kernel. (Or U can be replaced with another high-dimensional representation.)
- h represents the scale of the "slow" dynamics; see later how to choose it

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?
- what if I have real data?

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

Outline

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

In coordinates 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)

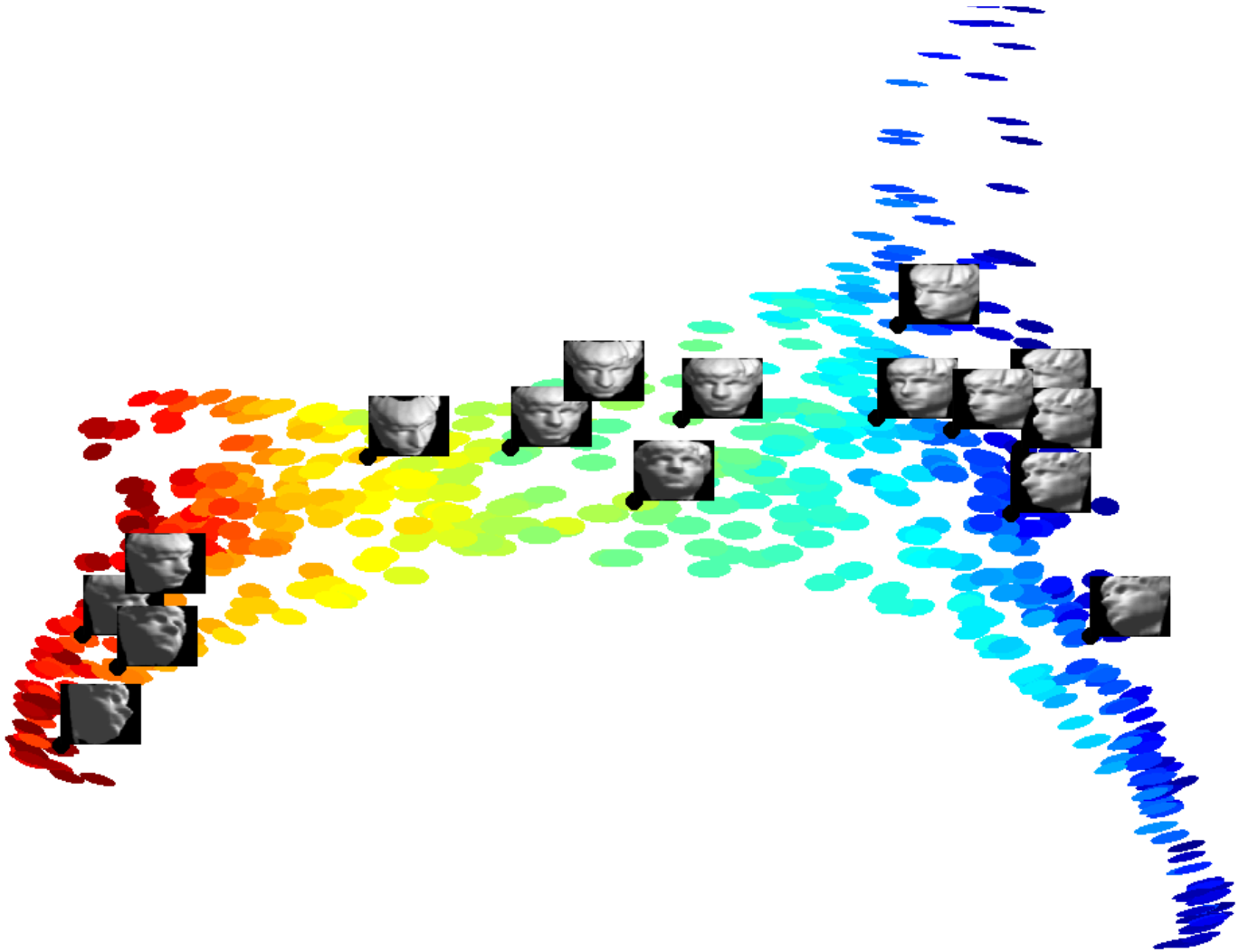
Problem formulation

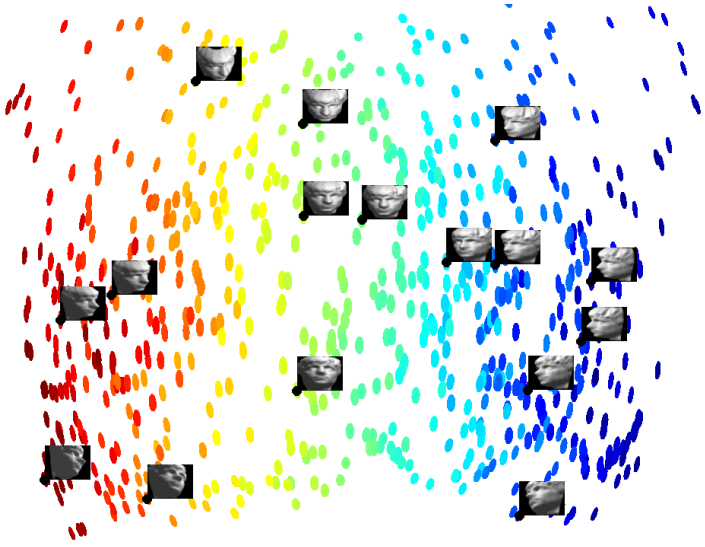
- Given:
  - data set $\mathcal{D} = \{p_1, \dots p_n\}$ sampled from manifold $\mathcal{M} \subset \mathbb{R}^D$
  - embedding $\{ x_i = \phi(p_i),\ p_i \in \mathcal{D} \}$
    by e.g LLE, Isomap, LE, ...
- Estimate $G_i \in \mathbb{R}^{m \times m}$ the (pushforward) Riemannian metric for $p_i \in \mathcal{D}$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

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

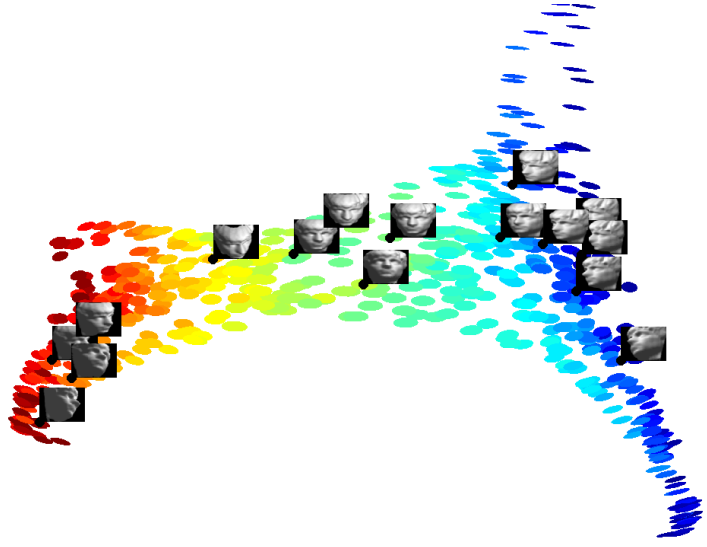
g for Sculpture Faces

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

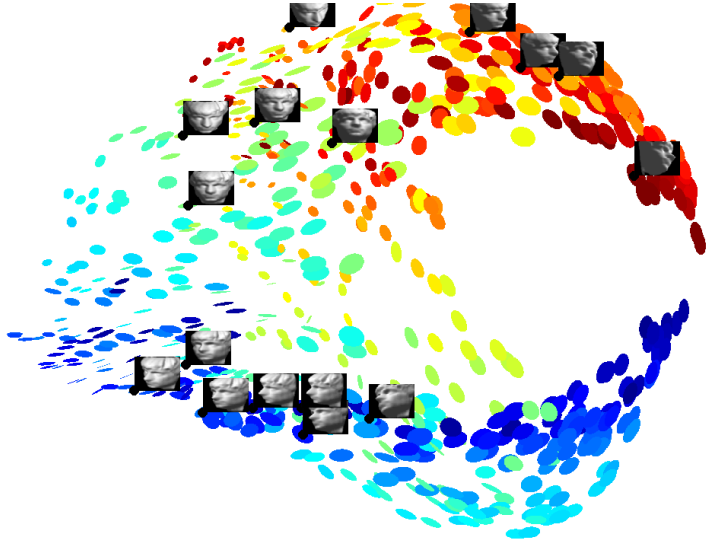




Isomap



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)

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

Estimation of g

Proposition

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

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

where $h = g^{-1}$ (matrix inverse) and $k, l = 1, 2, \ldots m$ 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 well studied [Coifman&Lafon 06,Hein&al 07]

Algorithm to Estimate Riemann metric g

Given dataset $\mathcal{D}$

1. Preprocessing (construct neighborhood graph, ...)
2. Find an embedding ϕ of $\mathcal{D}$ into $\mathbb{R}^m$
3. Estimate discretized Laplace-Beltrami operator L
4. Estimate H_p and $G_p = H_p^\dagger$ for all p
   - 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

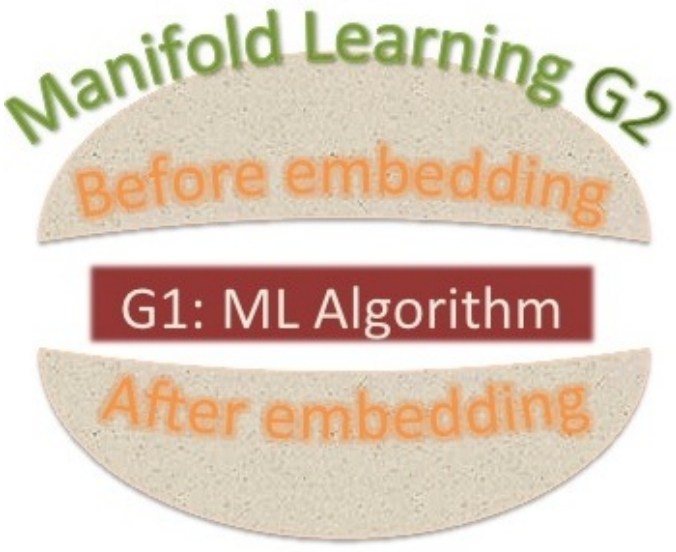
Why useful

- 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

- Estimating distortion
- Correcting distortion
  - Integrating with the local volume/length units based on G_i
  - Riemannian Relaxation [McQueen, M, Perrault-Joncas NIPS16]
- Estimation of neighborhood radius [Perrault-Joncas,M,McQueen NIPS17] and of intrinsic dimension d (variant of [Chen,Little,Maggioni,Rosasco])
- Accelerating Topological Data Analysis, selecting eigencoordinates,. . . (in progress)

Outline

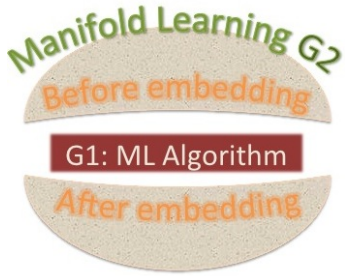


Manifold Learning G2

- what distance measure?
- what graph?
- what bandwidth h? (what kernel)
- what intrinsic dimension d?
- what embedding dimension $s \geq d$?

almost any ML Algorithm

- Cluster
- Estimate distortion
- Validate d, s, [select eigenvectors]
- Topological Data Analysis (TDA)
- Meaning of coordinates



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

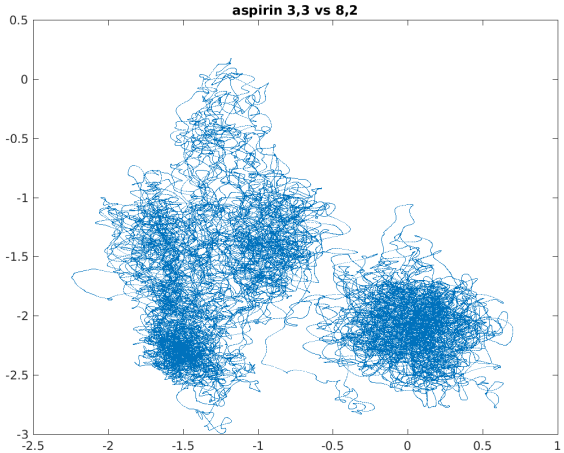
ML Algorithm: DiffMaps, LTSA

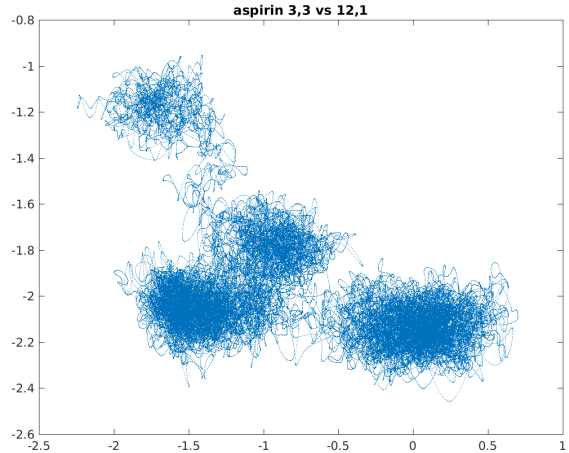
- Cluster [M,Shi 00],[M,Shi 01]. . . [M NeurIPS18]
- Estimate/correct distortion: Metric Learning and Riemannian Relaxation [McQueen, M, Perrault-Joncas NIPS16]
- Validate d, s, [select eigenvectors] [Chen, M NeurIPS19]
- Topological Data Analysis (TDA)
- Meaning of coordinates [M,Koelle,Zhang, 2018]

- Manifolds with vector fields
- Finding ridges and saddle points (in progress)

Data: from MD simulations

- Example: $n = 210,000$ configurations of aspirin simulated at $T = 500k$.
- $N = 21$ atoms, $21\times$ 3 - 6 $= 57$ degrees of freedom





- Data $R_{1:n} \in \mathbb{R}^{3\times N}$

What distance?

Procrus Align all configurations by a rigid transformation

- $R_i \leftarrow T_i(R_i)$
- distance$(i, i') = ||R_i - R_{i'}||$

BondLen For each R_i, compute all pairwise distances $b_i = (||R_{iA} - R_{iB}||$ for A, B atoms)

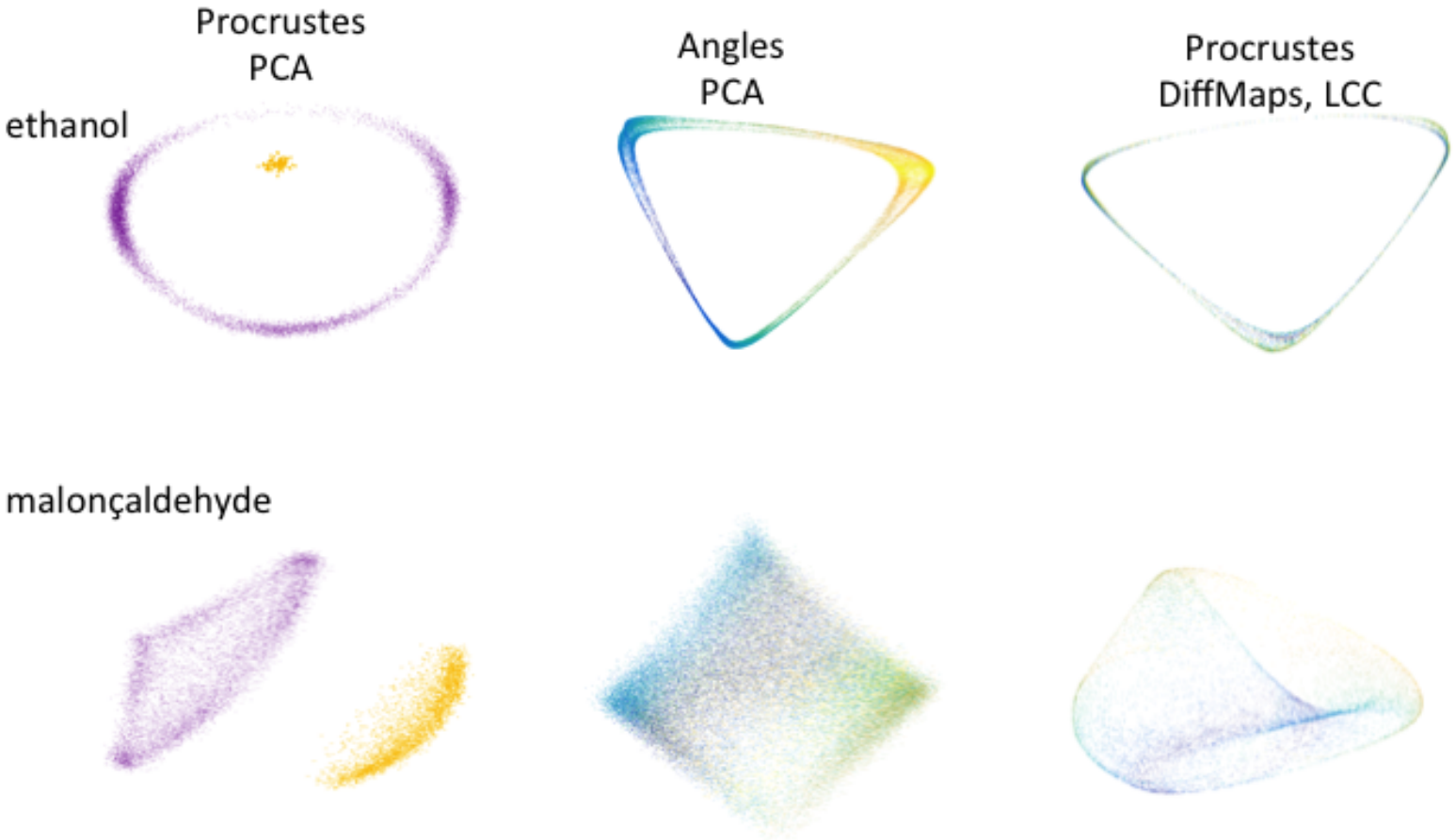
- (discard pairs A, B that are always too far to interact)
- rescale each distance to have geometric mean 1
- distance$(i, i') = ||b_i - b_{i'}||$

Angles For A, B, C atoms, compute 2 angles of triangle ABC

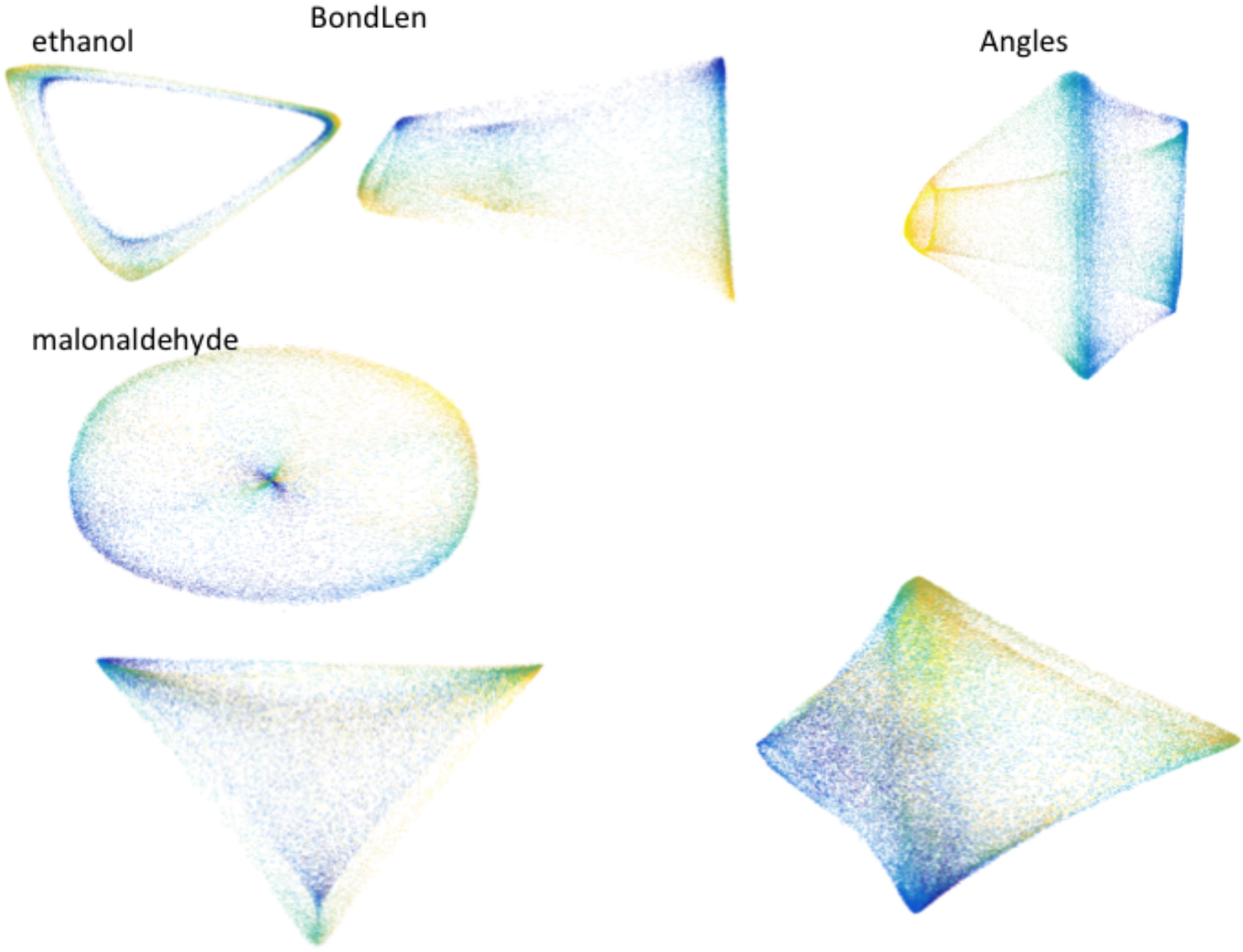
- (discard "edges" A, B that are always long)
- $a_i = (\text{angles}(ABC)\text{ for all}A, B, C)$
- distance$(i, i') = ||a_i - a_{i'}||$
- loses the scale information! (OK when cell consists of a single molecule)

- Follow up with PCA (0.9999 variance explained) to remove residual linear dependencies
- Kernels (Coulomb Matrix, SLATM, SOAP) instead of distances?

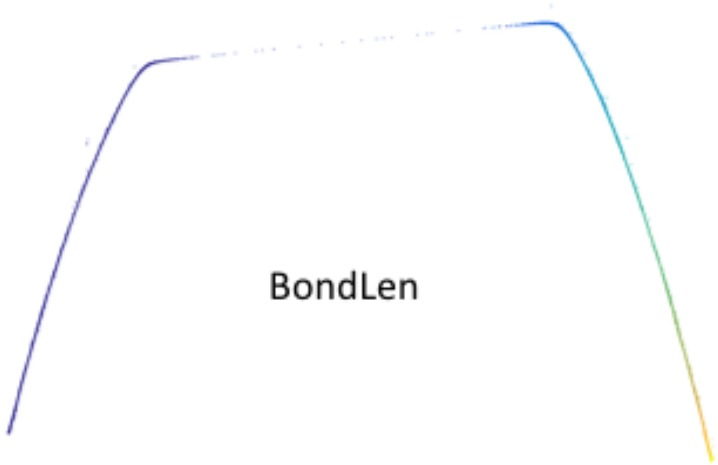
Procrustes creates artificial disconnects

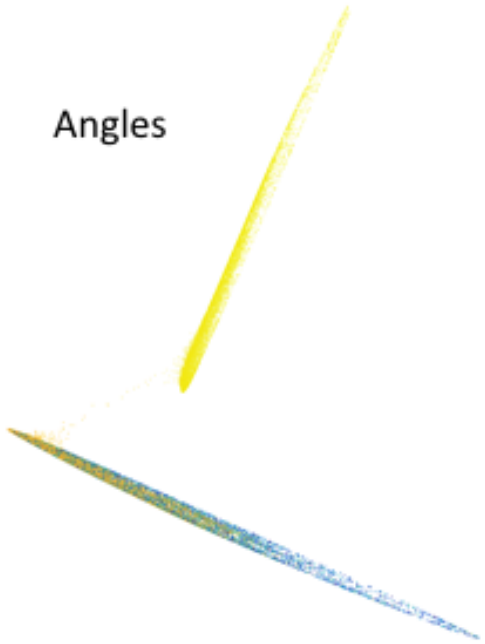


BondLen vs Angles – no bonds broken



BondLen vs Angles: $CH_3Cl + Cl^- \leftrightarrow CH_3Cl + Cl^-$





Angles, 1 cluster

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

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

- Does it matter?

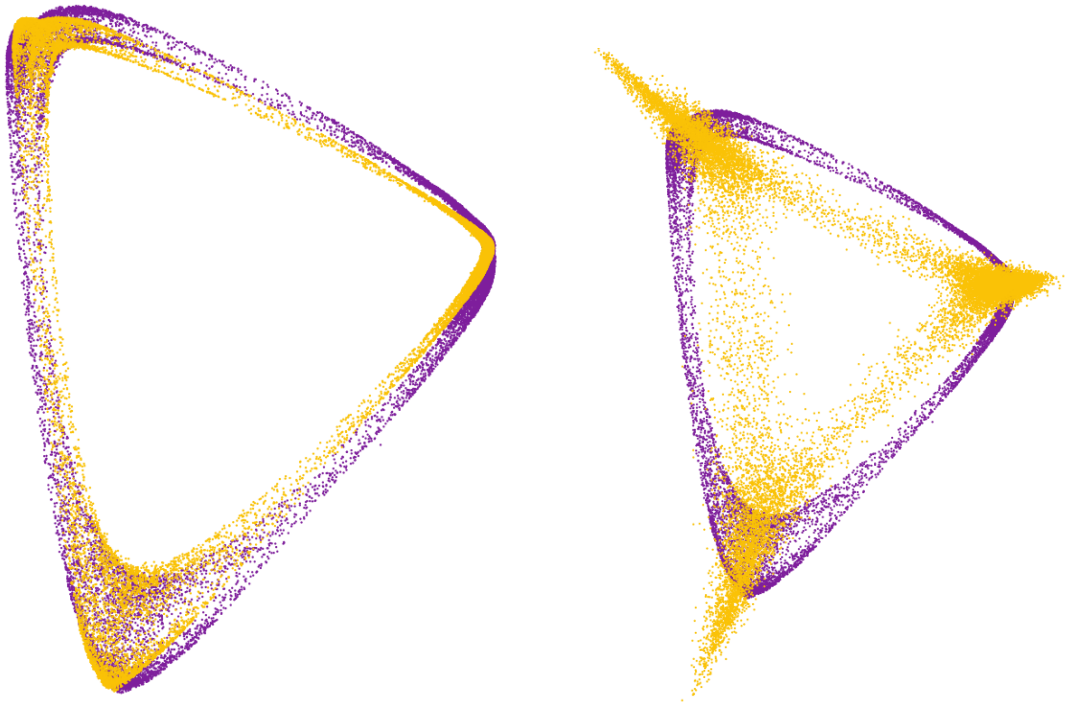
What graph? Radius-neighbors vs. k nearest-neighbors

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

- Does it matter?

- Yes, for estimating the Laplacian and distortion
  - Why? [Hein 07, Coifman 06, Ting 10, . . .] k-nearest neighbor Laplacians do not converge to Laplace-Beltrami operator Δ
  - but to $\Delta + 2\nabla(\log p) \cdot \nabla$ (bias due to non-uniform sampling)
- Renormalization of Laplacian also necessary

configurations of ethanol $d = 2$



K-nearest neighbor without renormalization

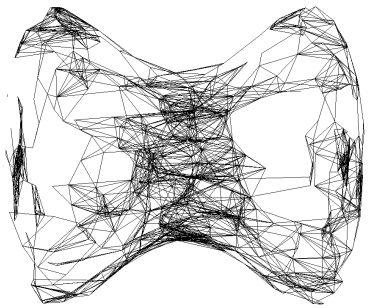
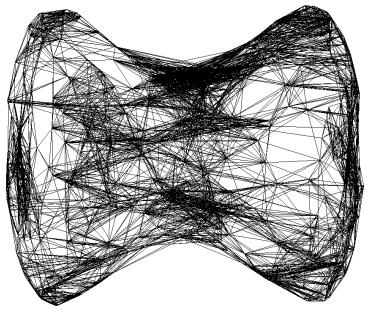
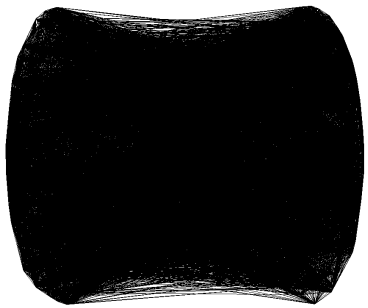
Self-consistent method of chosing ϵ

- 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?

Our idea

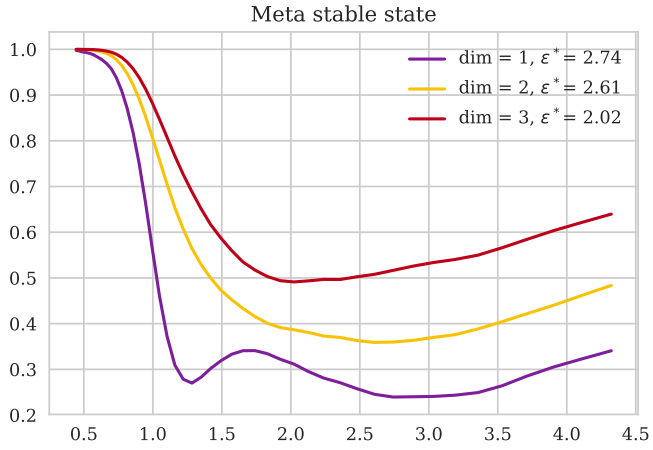
For given ϵ and data point p

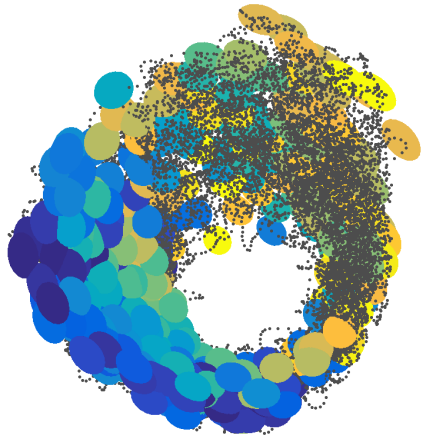
- Project neighbors of p onto tangent subspace
  - this "embedding" is approximately isometric to original data
- Calculate Laplacian $L(\epsilon)$) and estimate distortion $H_{\epsilon,p}$ at p
  - $H_{\epsilon,p}$ must be $\approx I_d$ identity matrix

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

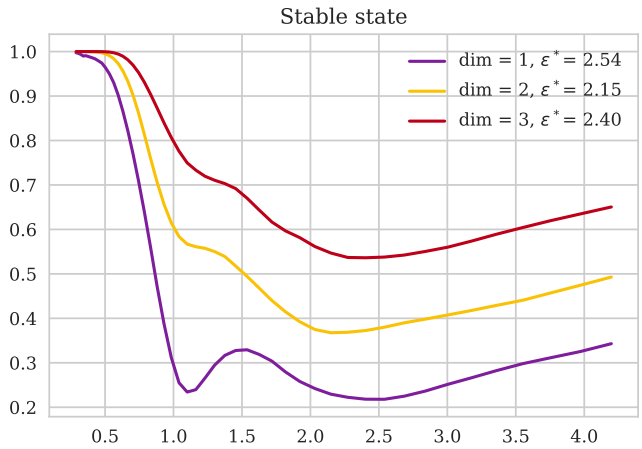
ϵ and distortion for aspirin

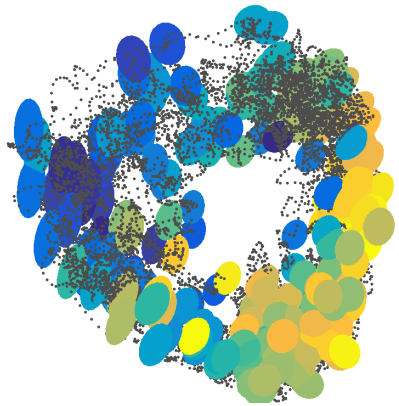
meta-stable cluster





stable cluster





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

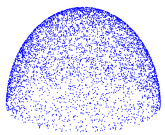
- Method of [Chen,Little,Maggioni,Rosasco,2011]
  - do local SVD for a range of neighborhood radii
  - choose a an appropriate radius ϵ
  - dimension = largest eigengap at radius ϵ
- used our self-concordant method to find ϵ

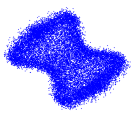
- Experiments: artificial 2D manifolds with noise

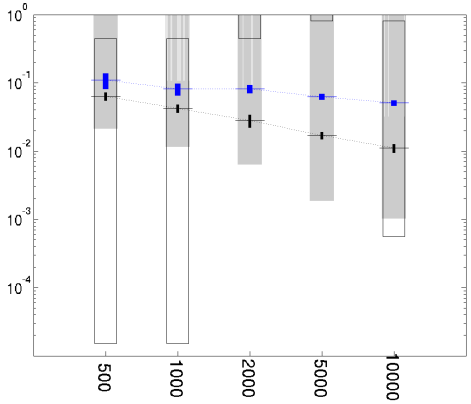
Intrinsic Dimension Estimation

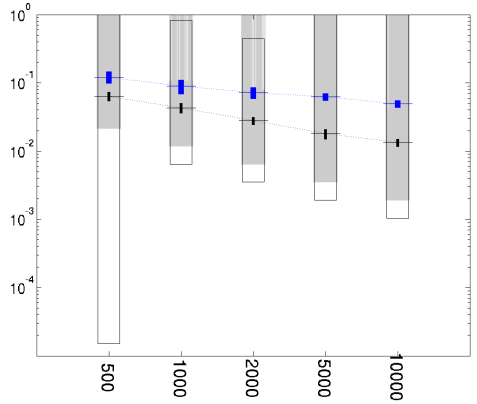
[Chen,Little,Maggioni,Rosasco,2011] and variation by

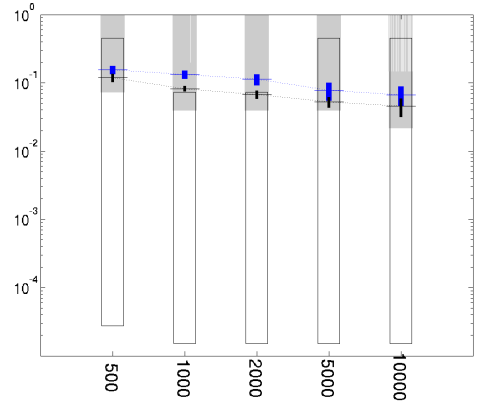
$\sigma = 0.001$ $\sigma = 0.01$ $\sigma = 0.1$

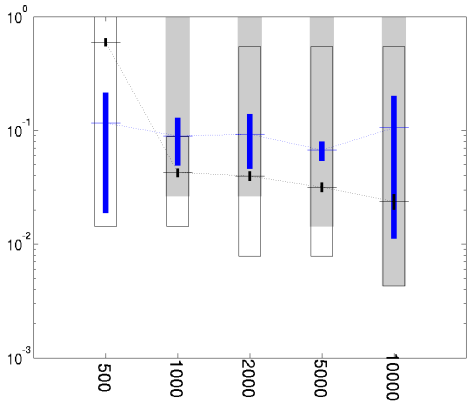


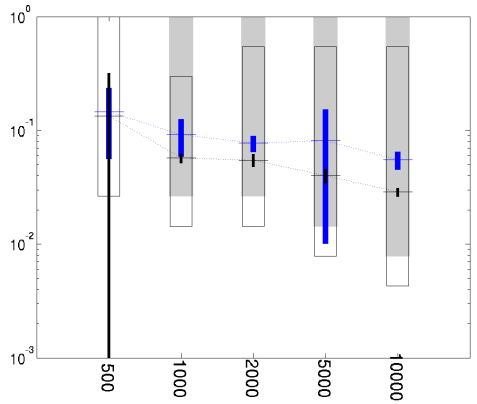


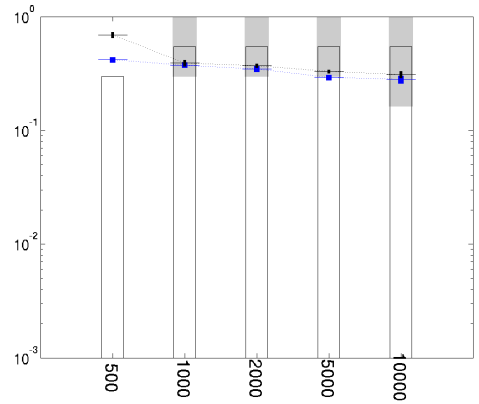




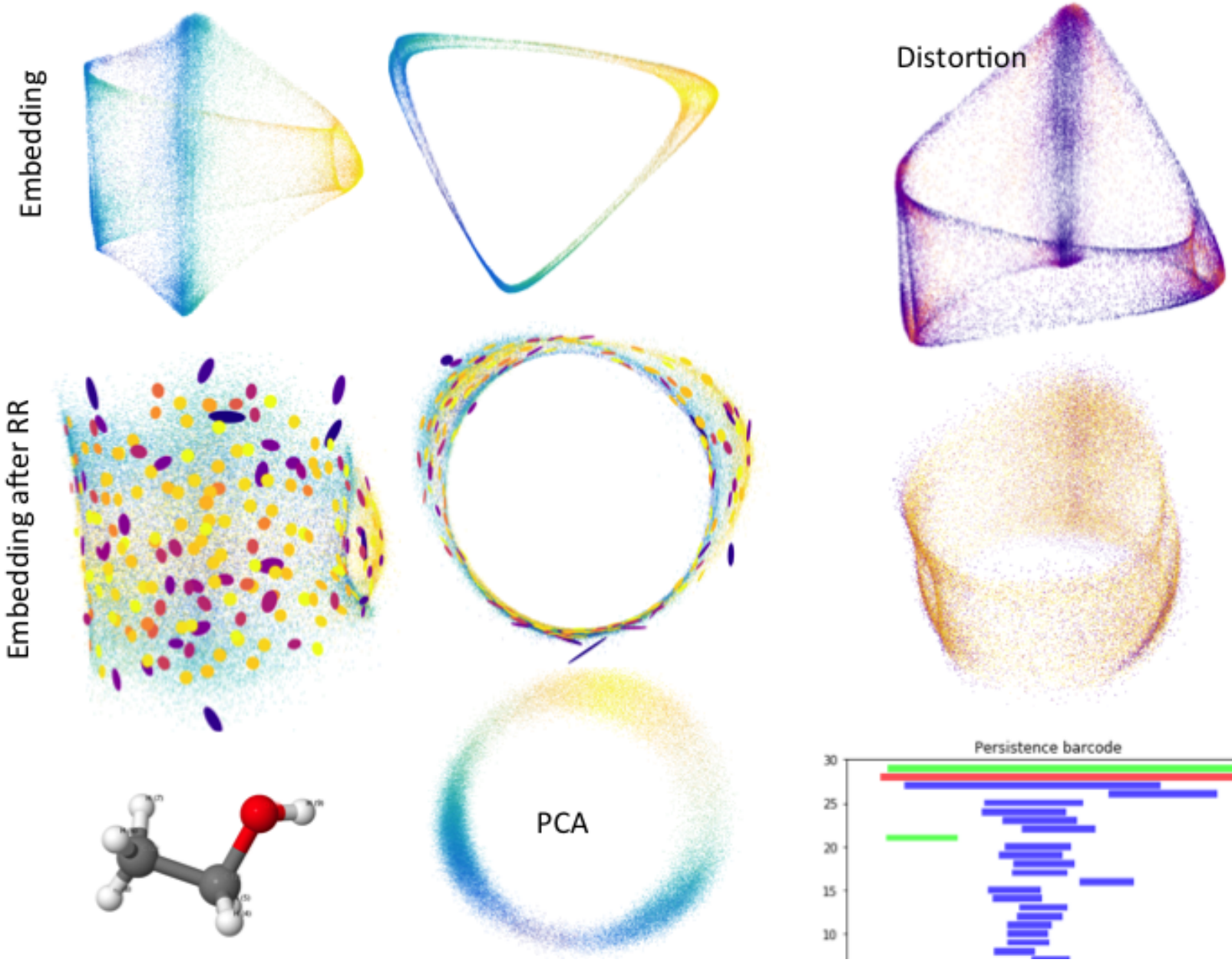






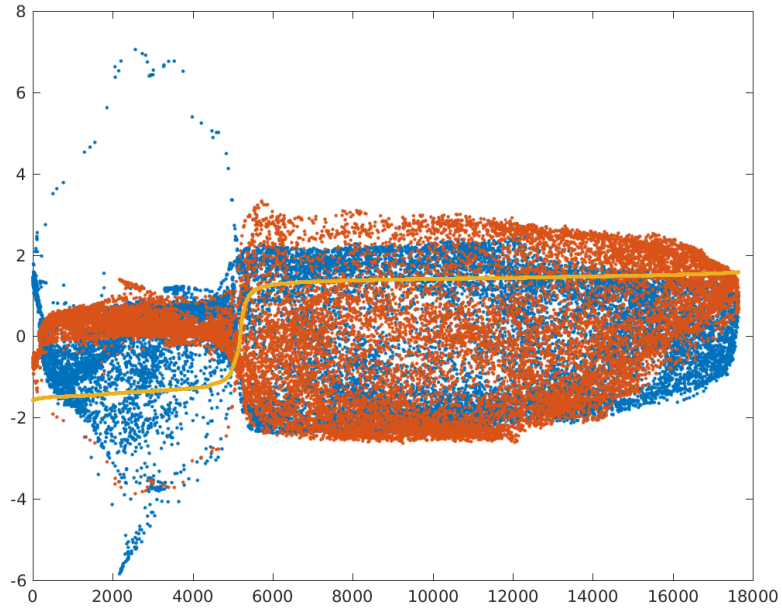


Correcting distortion: Riemannian Relaxation for Ethanol

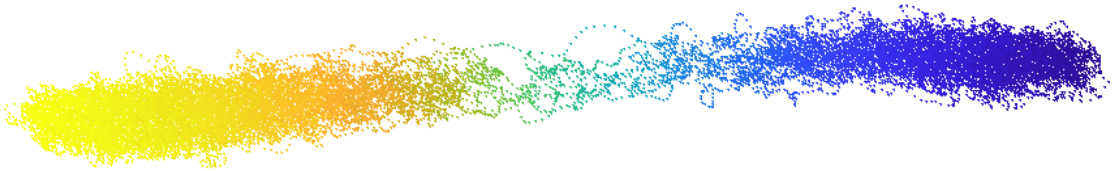


Clustering vs. Embedding

Aspirin, DiffMaps: coordinates 1,2,3 sorted by coordinate 1



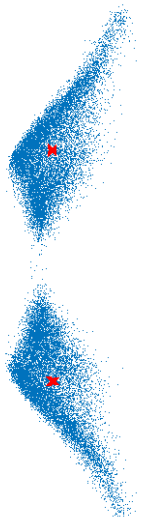
- Coordinate 1 (yellow)
  - indicates 2 clusters
  - "collective coordinate", separating the stable from the metastable state
- In general, K eigenvalues ≈ 0 indicate K meta-stable states
  - for K not too large

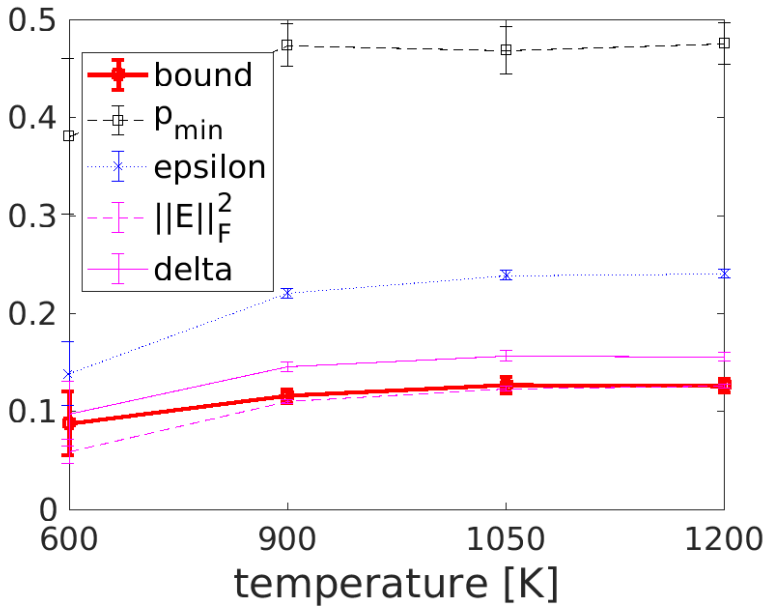


guaranteed err $< 6.5\%$ [M NeurIPS18] "How to tell when a clustering is (approximately) correct using convex relaxations"

Clustering with guarantees

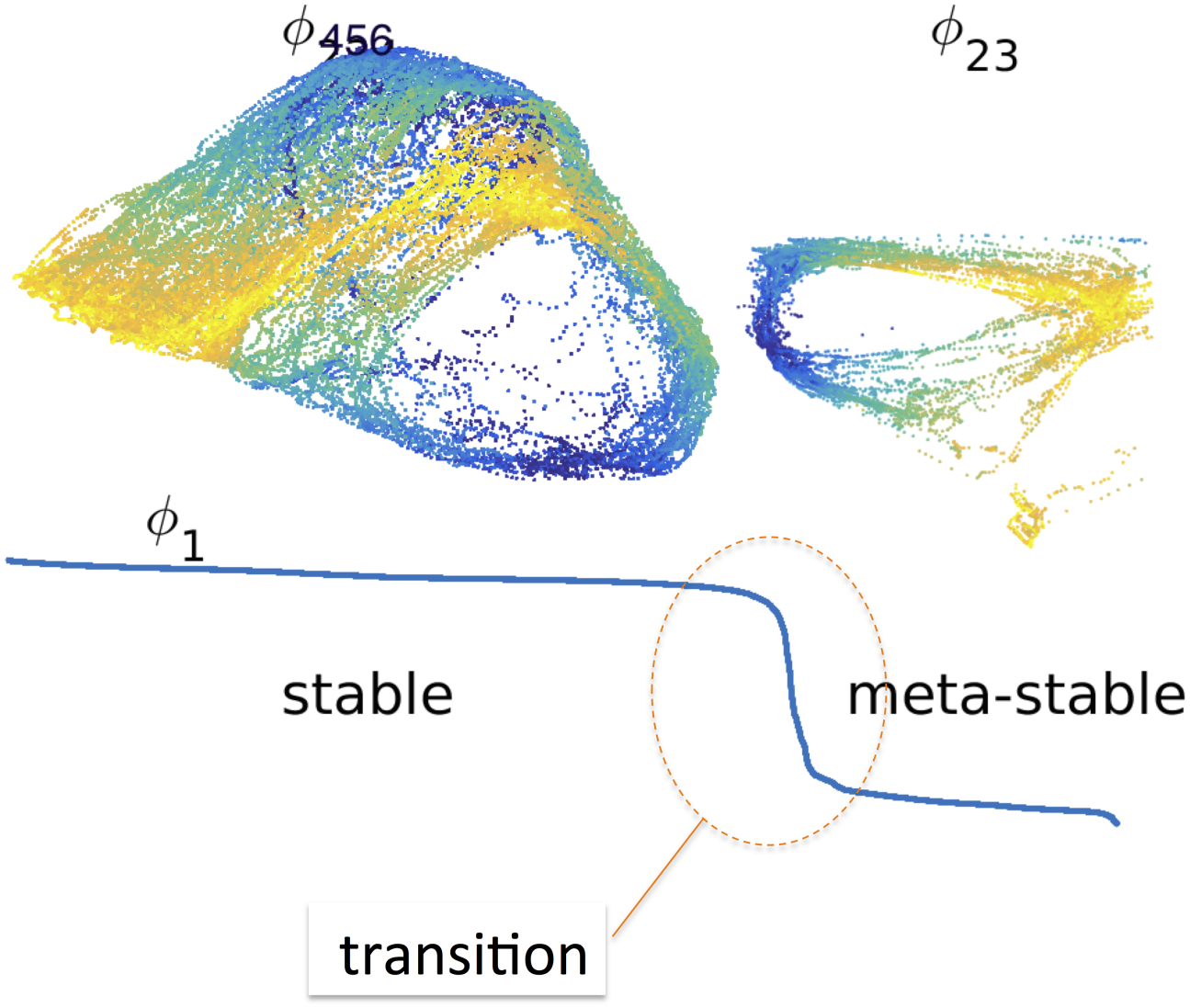
Molecular dynamics simulation of $CH_3Cl + Cl^- \leftrightarrow CH_3Cl + Cl^-$





Data by Jim Pfaendtner and Chris Fu

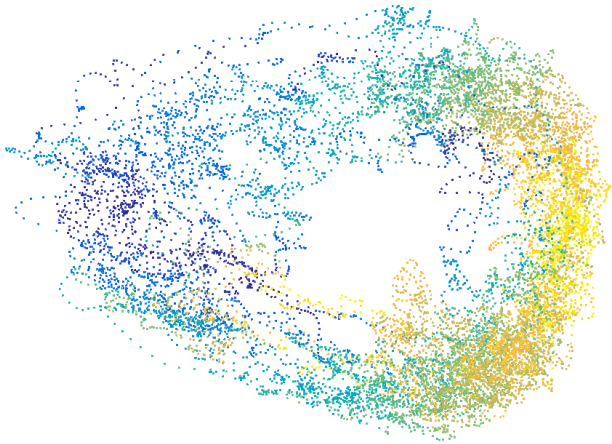
Embedding coordinates, after clustering



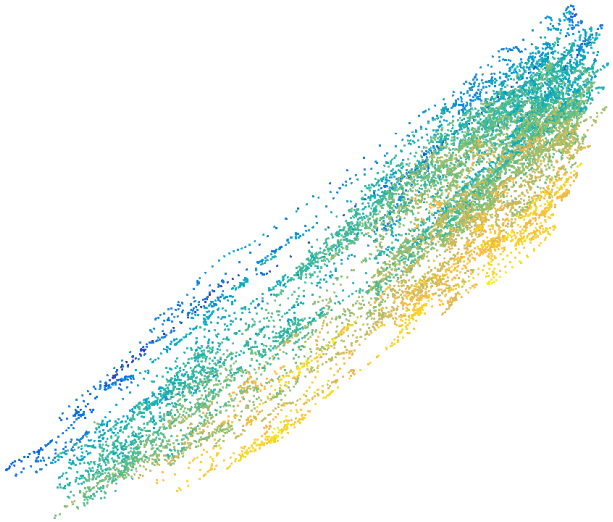
- after clustering, embed analyze separarately
  - each cluster
  - transition regions

Aspirin simulation data by cluster

Stable state

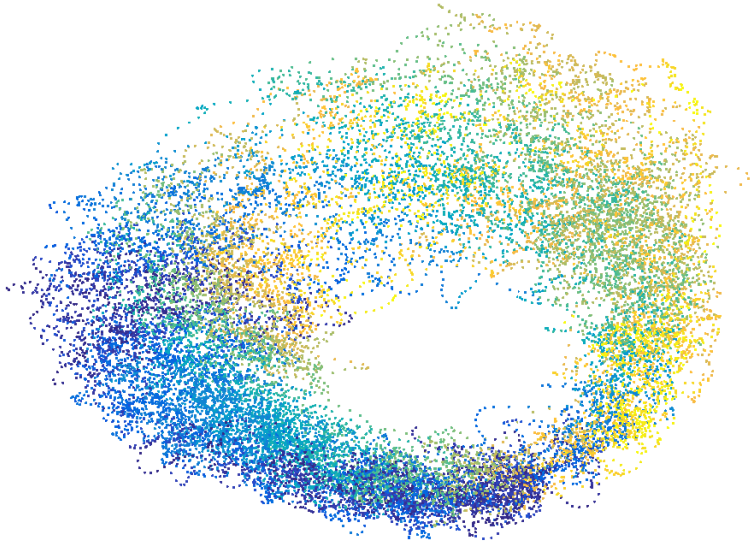


correlation with methil bond torsion

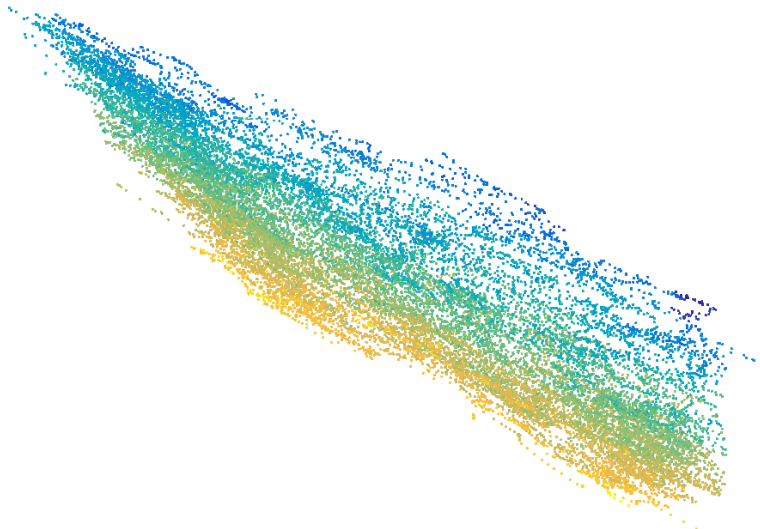


correlation with C — O bond torsion

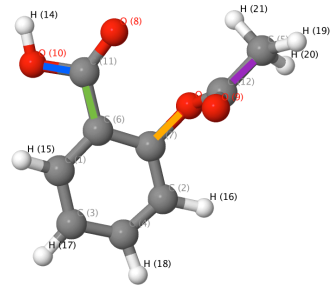
Metastable state



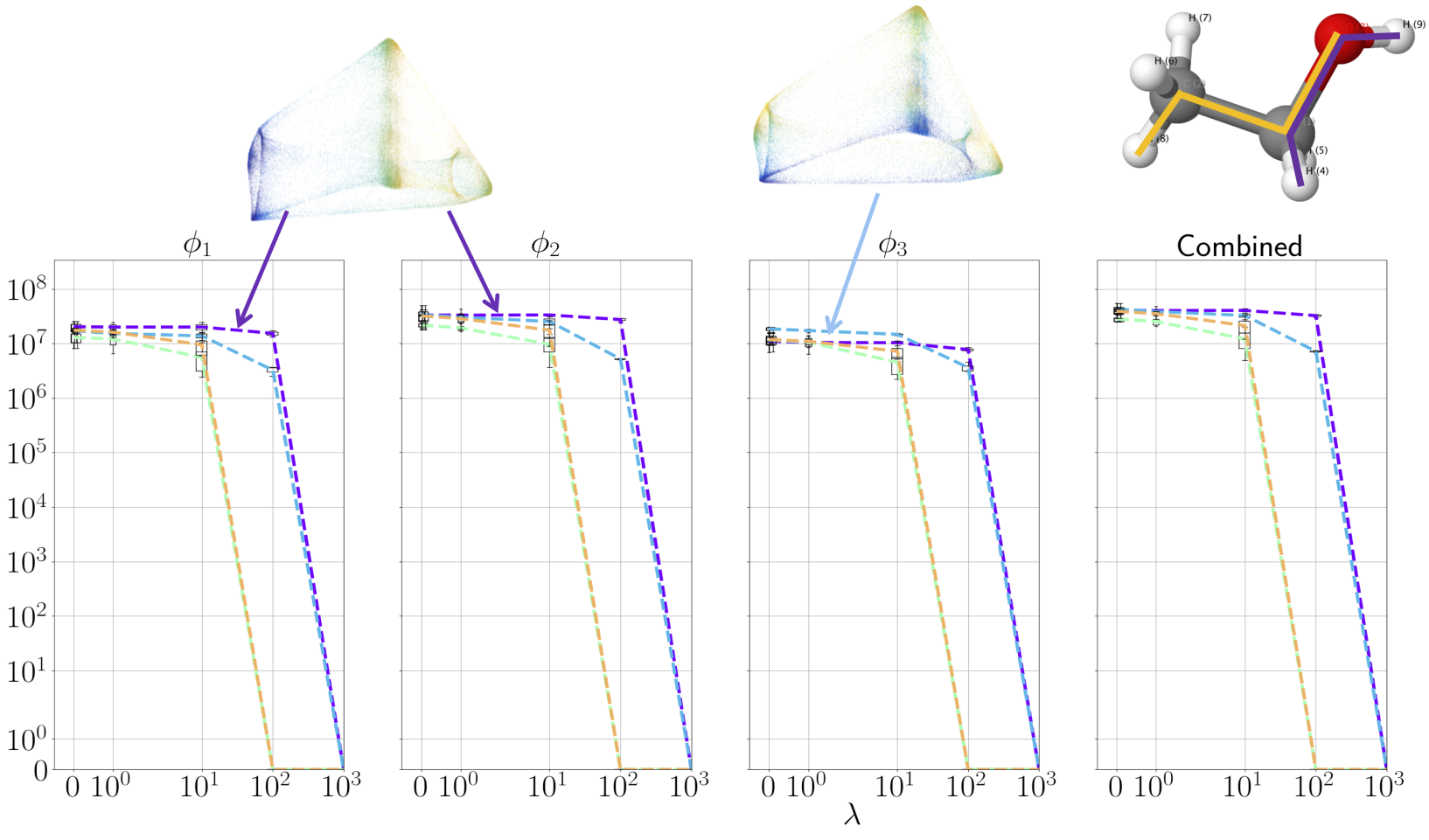
correlation with methil bond torsion (5–12)



correlation with C — O bond torsion (7–13)



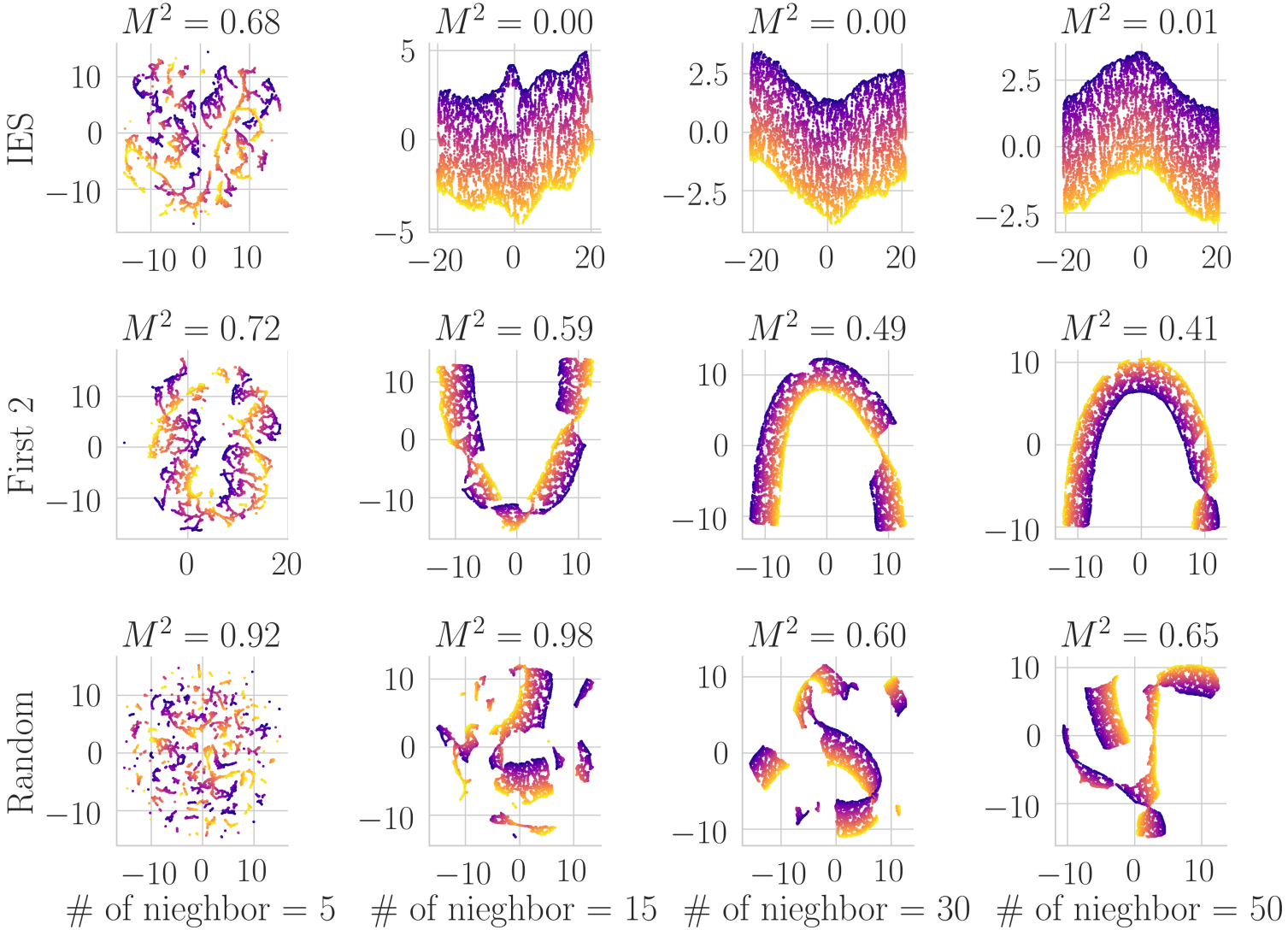
Manifold coordinates with physical meaning



Outline

UMAP on long strip

Rectangle, aspect ratio 6.28:1, sampled uniformly [Chen,M NeurIPS19]

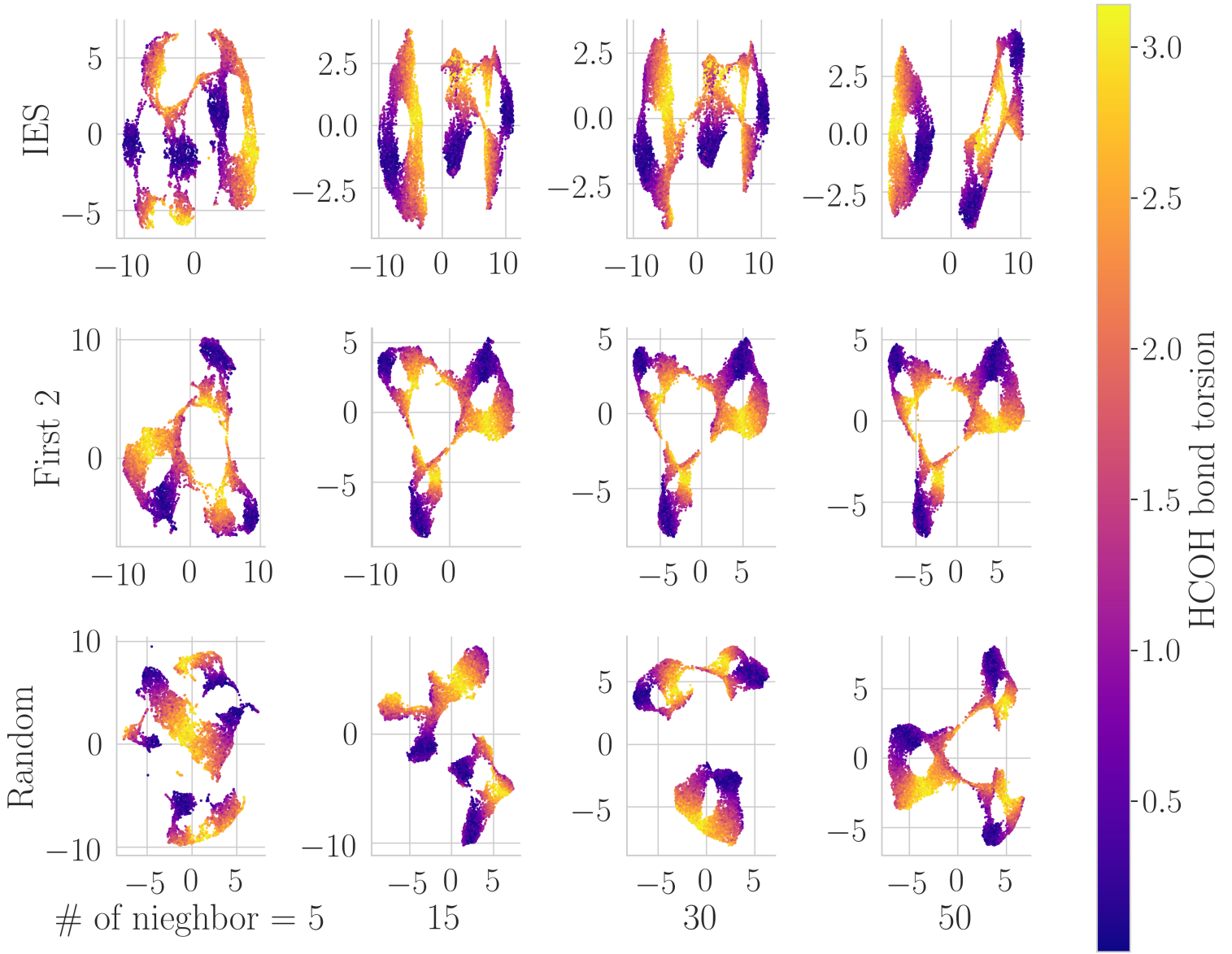


Initialization by [Chen,M NeurIPS19]

Laplacian Eigen-maps (default)

Random

UMAP for ethanol



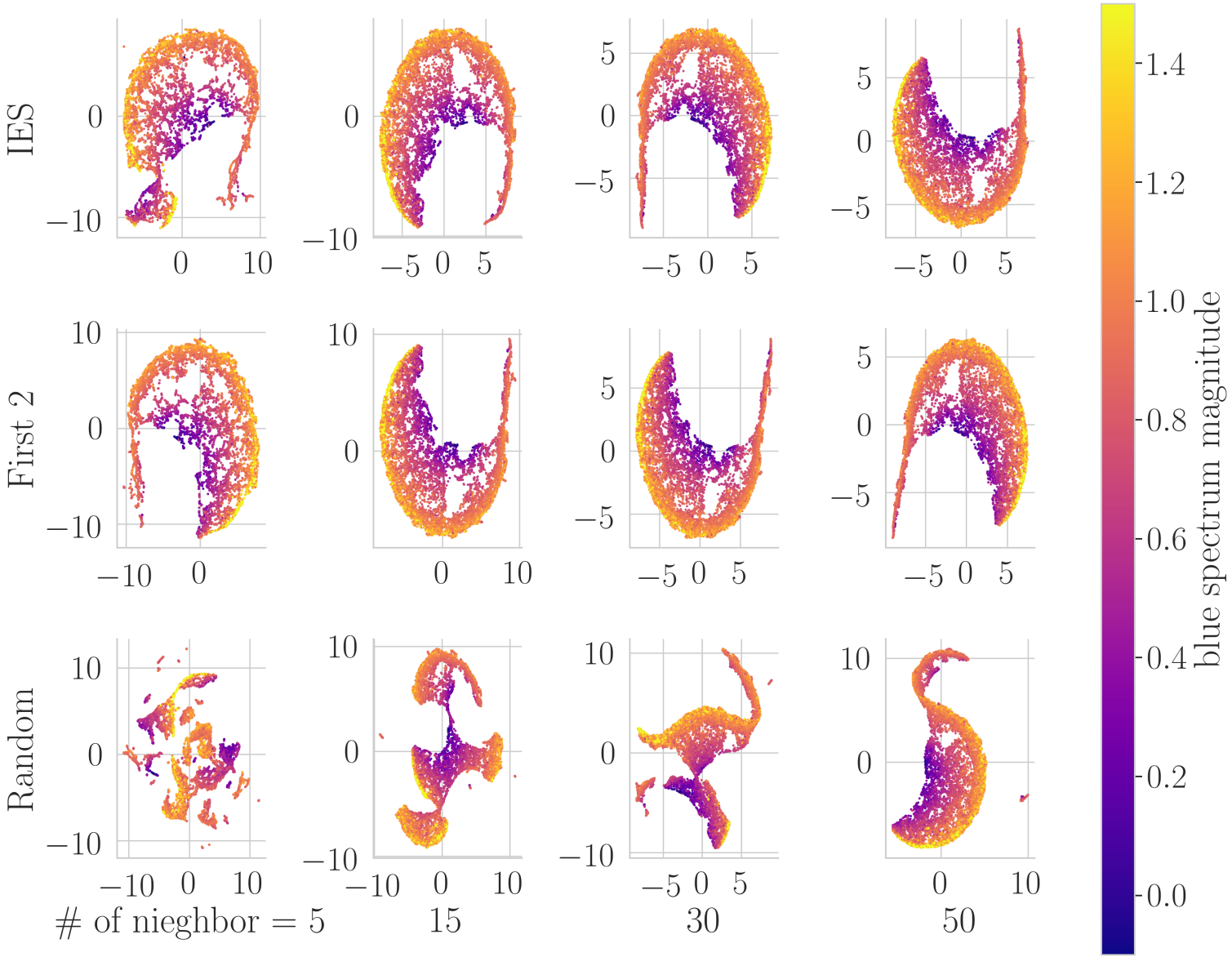
Initialization by [Chen,M NeurIPS19]

Laplacian Eigen-maps (default)

Random

UMAP for SDSS

Galaxy spectra in $D = 3,750$ dimensions from SDSS Curated by Grace Telford and Jacob VanderPlas

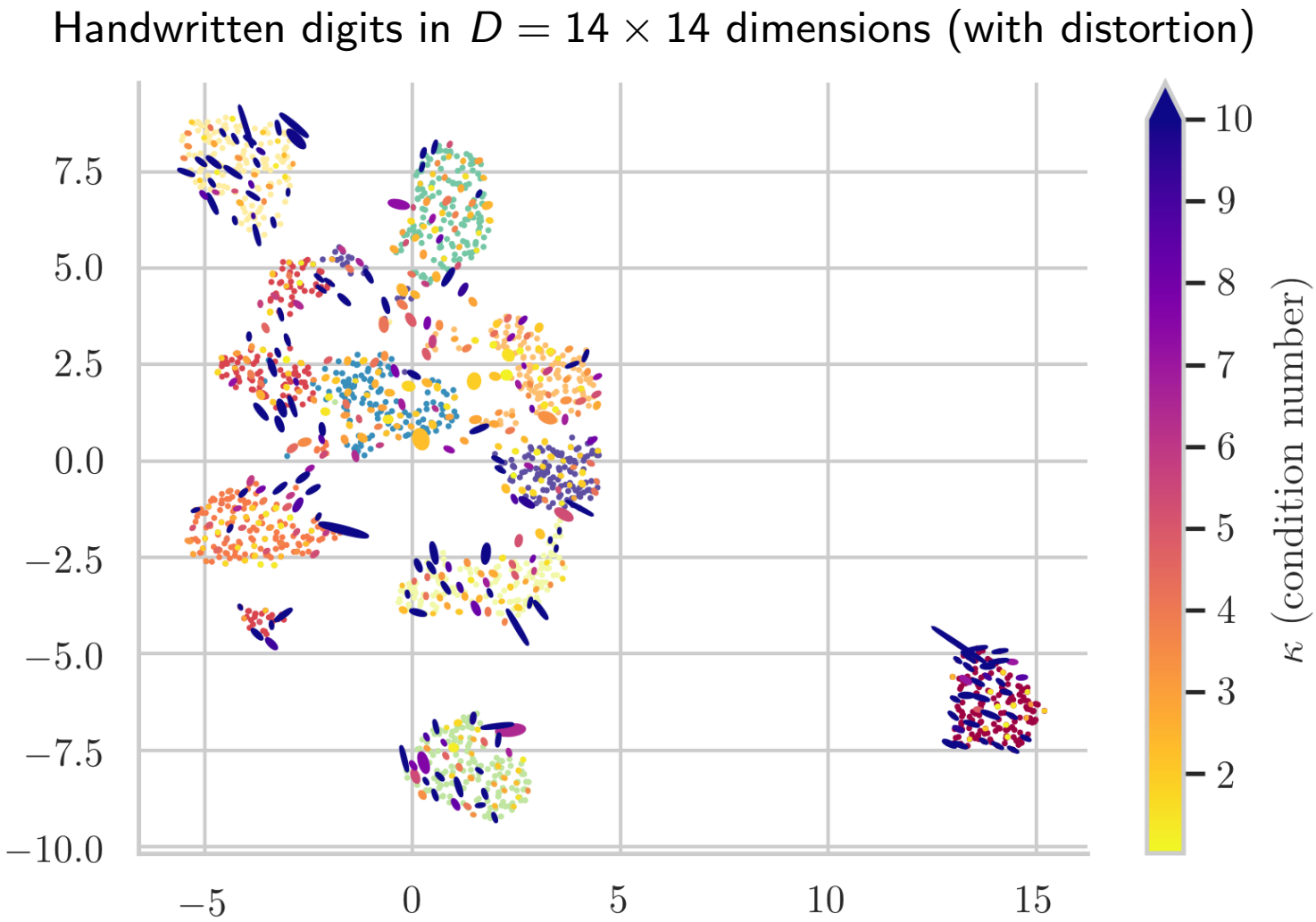


Initialization by [Chen,M NeurIPS19]

Laplacian Eigen-maps (default)

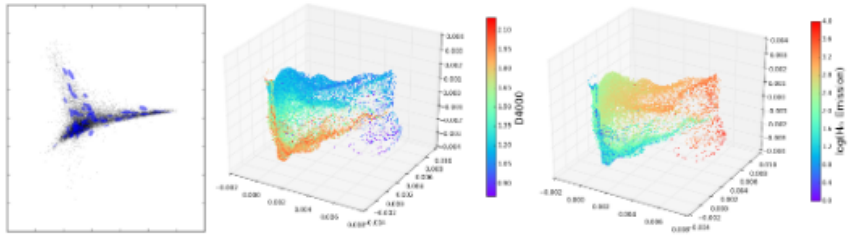
Random

UMAP on MNIST data



https://www.github.com/mmp2/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

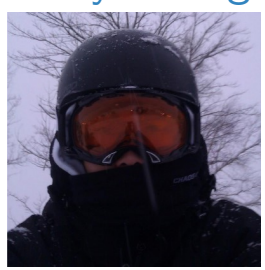
Installation with Conda

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

James McQueen

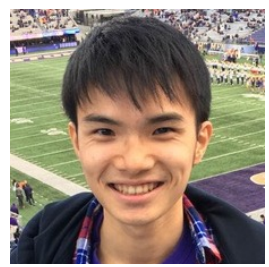
Jake VanderPlas

Jerry Zhang



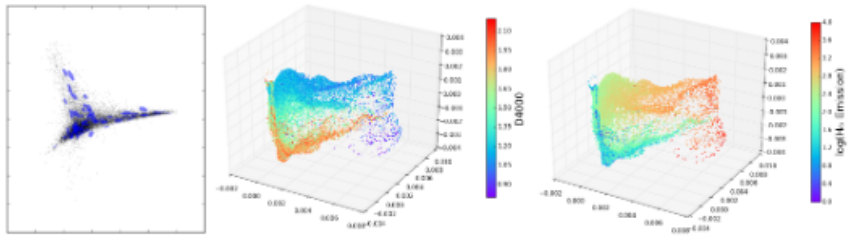
Grace Telford

Yu-chia Chen



https://www.github.com/mmp2/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.

- Recently deployed on Theta HPC architecture at the Argonne National Computing Facility for the analysis of molecular crystals
- Becoming a python platform for scalable unsupervised learning
  - Contributions are invited!

Manifold learning for sciences and engineering

Manifold learning should be like PCA

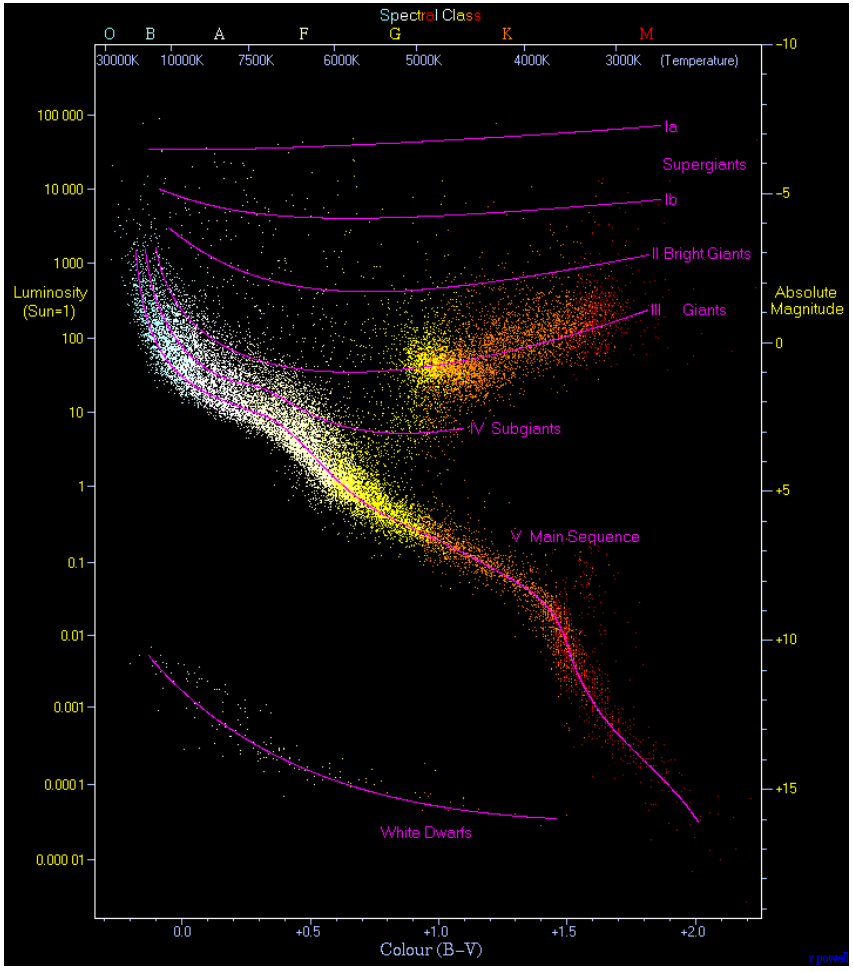
- tractable/scalable
- "automatic" – minimal burden on human
- first step in data processing pipe-line

  should not introduce artefacts

More than PCA

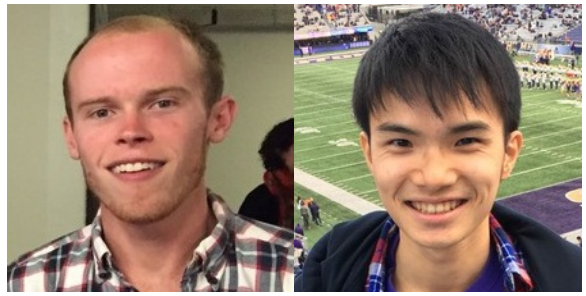
- estimate richer geometric/topological information
- dimension
- borders, stratification
- clusters
- Morse complex
- meaning of coordinates/continuous parametrization

Manifold Learning for engineering and the sciences



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

Sam Koelle, Yu-Chia Chen, Hanyu Zhang, Alon Milchgrub



Dominique-Perrault Joncas (Google), James McQueen (Amazon)

Jacob VanderPlas (Google), Grace Telford (UW Astronomy)
Jim Pfaendtner (UW), Chris Fu (UW)
A. Tkatchenko (Luxembourg), S. Chmiela (TU Berlin), A. Vasquez-Mayagoitia (ALCF)

Thank you

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

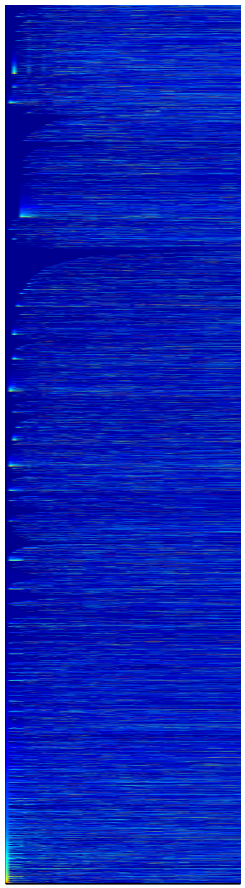
person 1:	19	19	3	3	3	3	3	3	3	3	3	3	3	10	10
person 2:	3	3	3	3	3	3	3	1	3	1	3	1	3	1	35
person 3:	152	5	71	1	1	1	71	36	36	5	4	5	5	4	4
person 4:	3	3	3	3	9	3	3	8	2	8	5	5	8	239	239

(Data curated by Marc Scott, NYU)

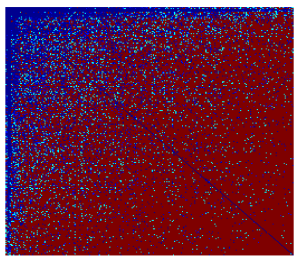
- Problem how to embed a directed graph with directed edges?

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 graph with directed edges?

Generative model

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

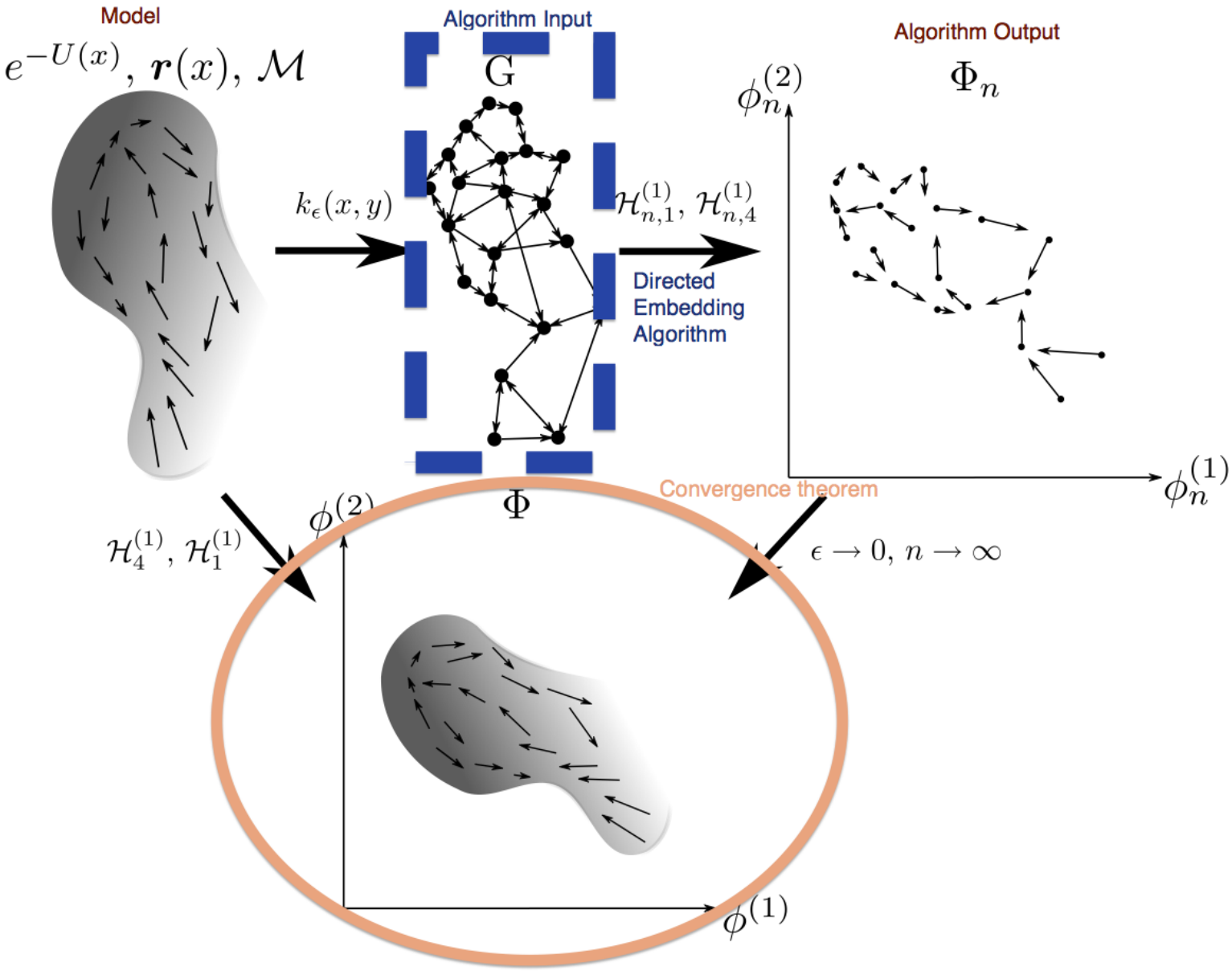
Assume Nodes of G sampled from manifold $\mathcal{M}$ of dimension d

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

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

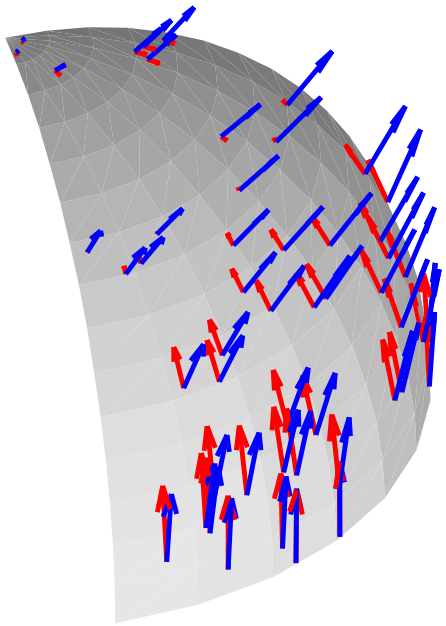
Wanted Estimates of manifold $\mathcal{M}$, density $p = e^{-U}$, vector field $\vec{\mathbf{r}}$

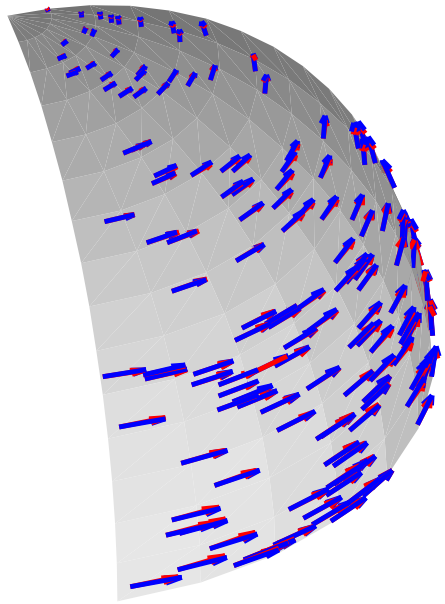
Generative model, data, and output

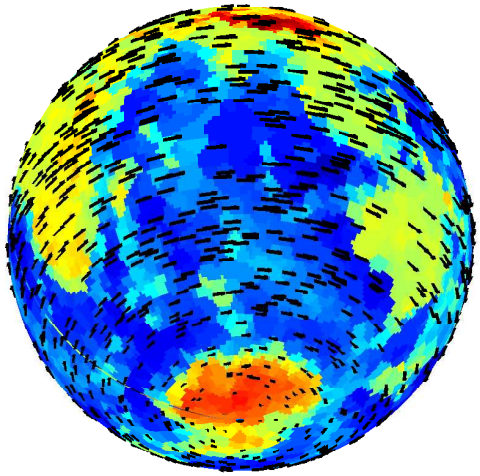


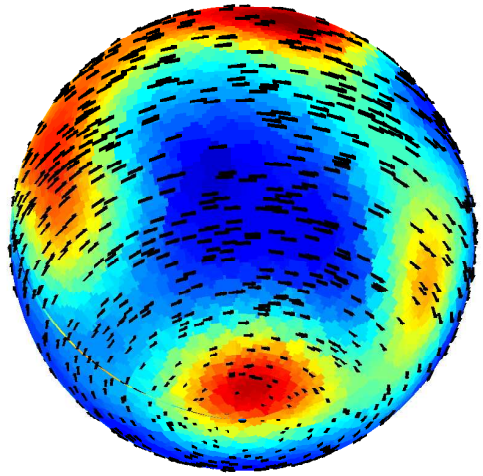
Examples – toy data

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

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

and $\vec{\mathbf{r}}(x,y) = \vec{\mathbf{r}}(y,x)$ vector field (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 $$\lim_{\epsilon \to 0, n \to \infty} \frac{(T_\epsilon - I)}{\epsilon}$$

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 (diffusion maps)

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

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

$\alpha = 1$

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

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

$\alpha = 1$

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

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

Isolating the Vector Field $\mathbf{r}$

Coordinate free

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

Coordinate Representation of $\mathbf{r}$

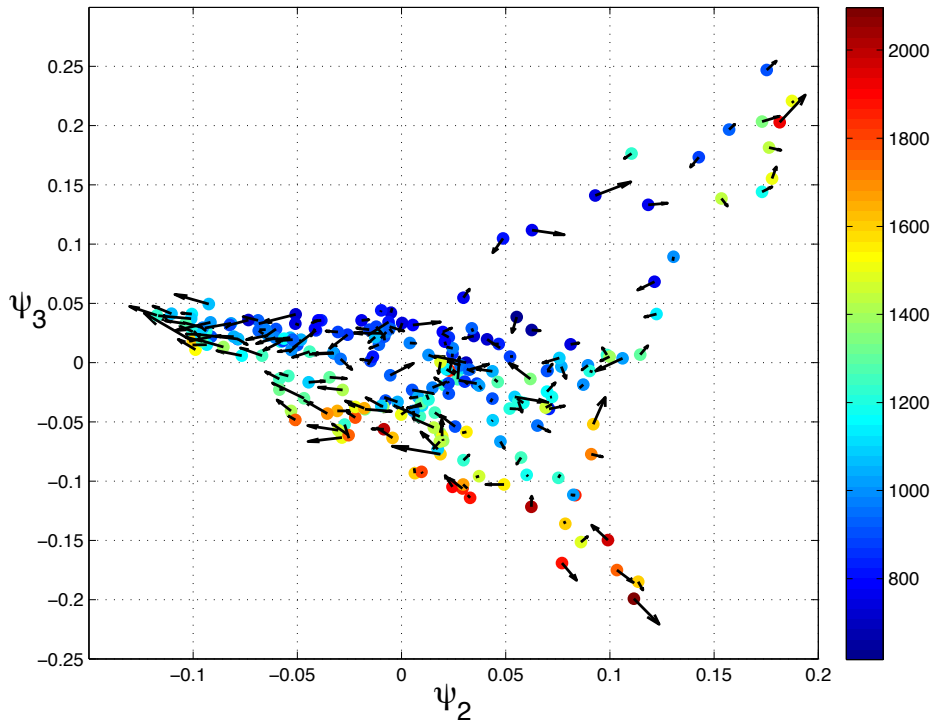
- Let Φ be an diffeomorphic embedding of $\mathcal{M}$
- Then
$$\vec{\mathbf{r}}_{||} = \vec{\mathbf{r}} \cdot \nabla \phi \,,$$
- and component of $\vec{\mathbf{r}}$ along coordinate ϕ_k is $\mathbf{r} \cdot \nabla \phi_k$
- Note that only $\vec{\mathbf{r}}_{||}$ is recovered

Directed Embedding Algorithm

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

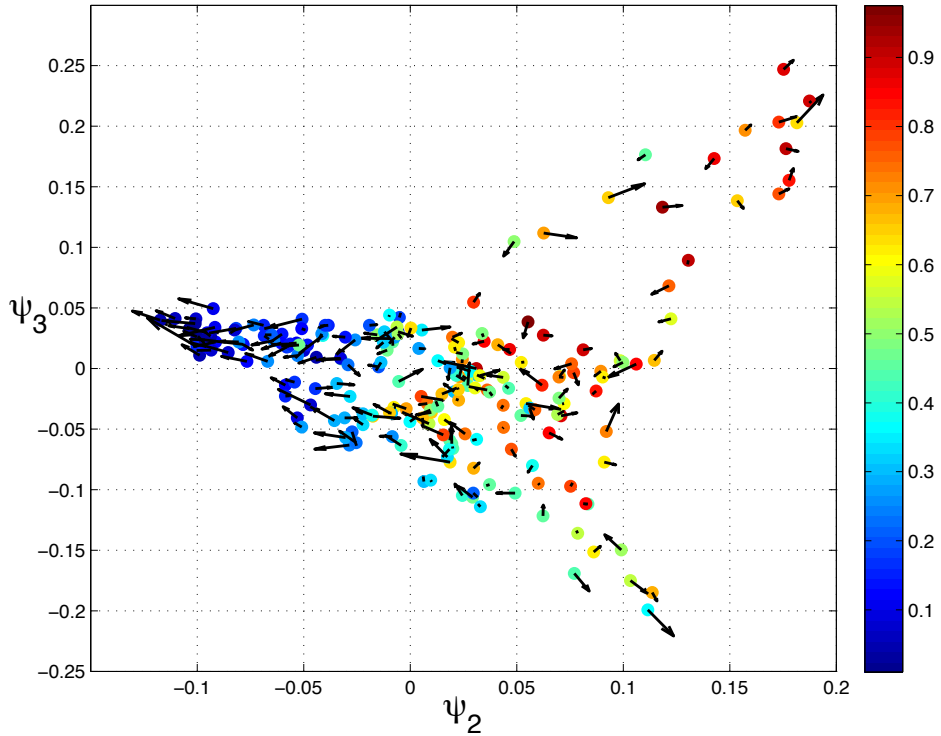
1. $S \leftarrow (W + W^T)/2$ *Estimate the Diffusion Maps embedding*
2. $q_i \leftarrow \sum_{j=1}^{n} S_{i,j}, \; Q = diag(q)$
3. $V \leftarrow Q^{-1} S Q^{-1}$
4. $q_i^{(1)} \leftarrow \sum_{j=1}^{n} V_{i,j}, \; Q^{(1)} = diag(q^{(1)})$
5. $H_{ss,n}^{(1)} \leftarrow {Q^{(1)}}^{-1} V$
6. Compute ϕ 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* $\vec{\mathbf{r}}$
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.

NLSY Data: creating a job landscape



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

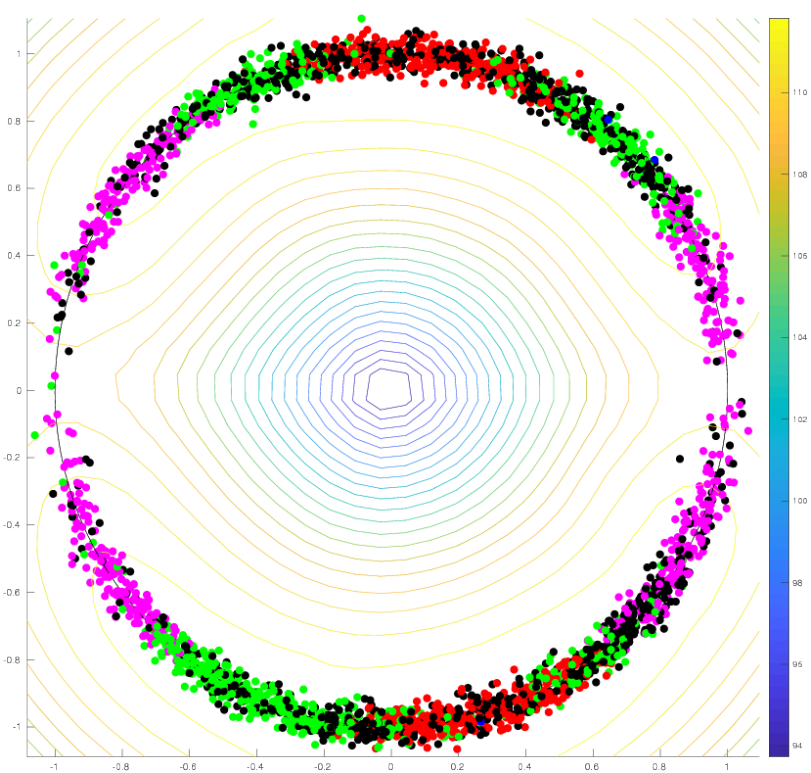
NLSY Data: creating a job landscape

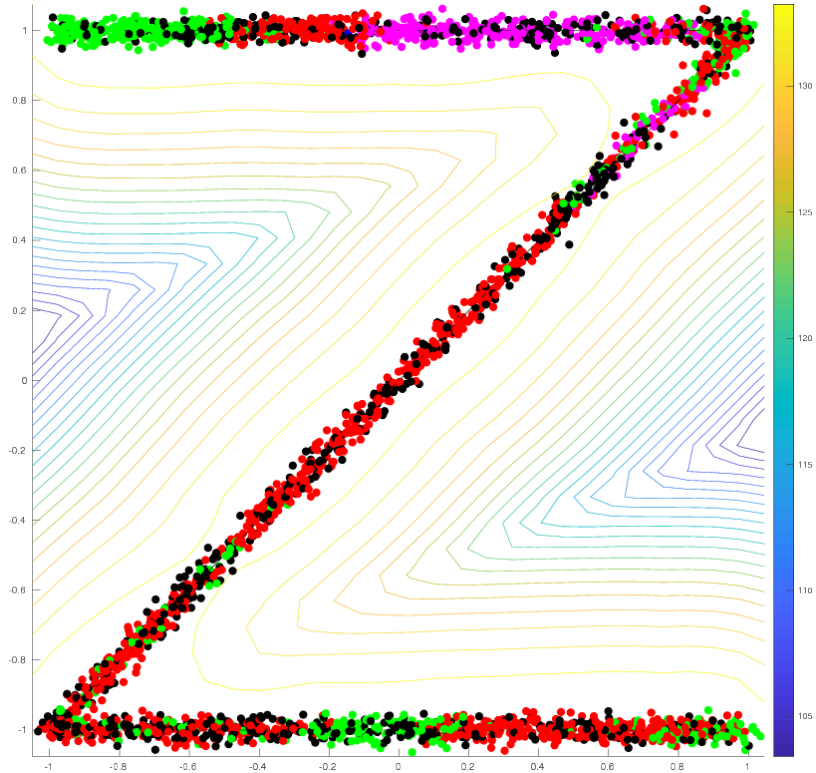


Embedding of the job market along with field $\mathbf{r} - 2\nabla U$ over the first two non-constant eigenvectors. Color = gender proportion in each job (with male = 0, female = 1).

Outline

Finding filaments in high dimensions

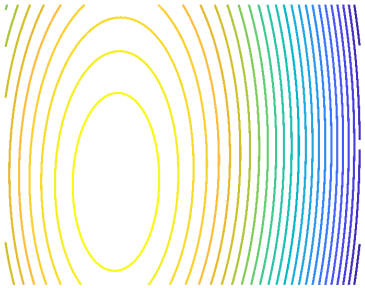




- data in $\mathbb{R}^D$ near a curve (or set of curves)
- wanted: track the ridge of the data density

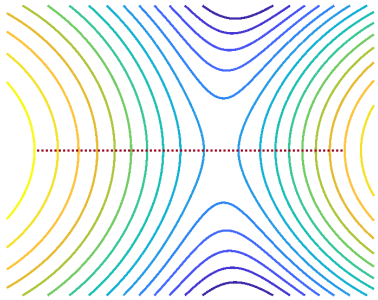
Mathematically,

Peak



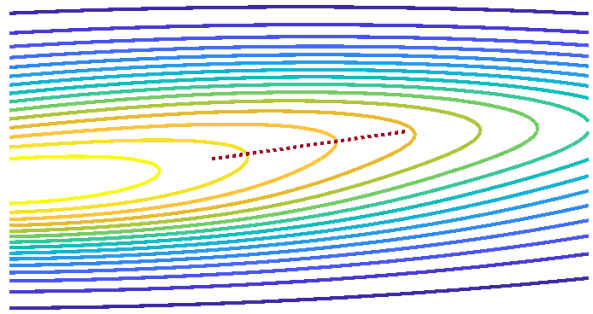
$\nabla p = 0$
$\nabla^2 p \prec 0$

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

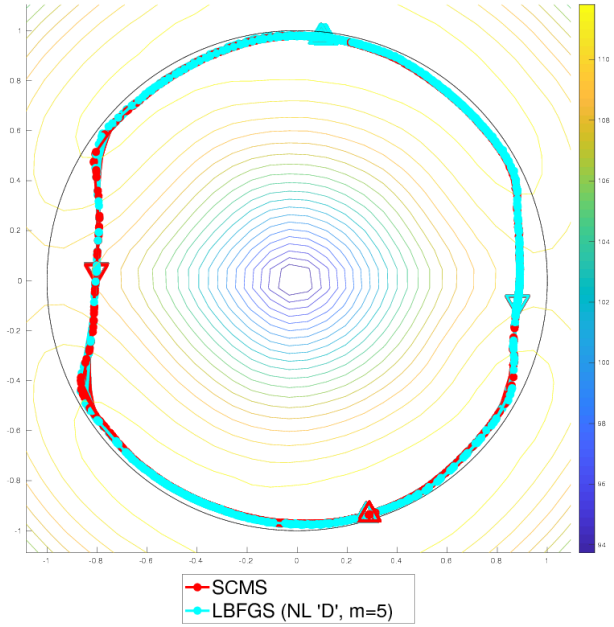
- Second Idea
  1. store $\{x^{k-l} - x^{k-l-1}\}_{l=1:m} \cup \{g^{k-l} - g^{k-l-1}\}_{l=1:m} = V$
  2. minimize $v^T H v$ s.t. $v \in \text{span } V$ where H is exact Hessian

- 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

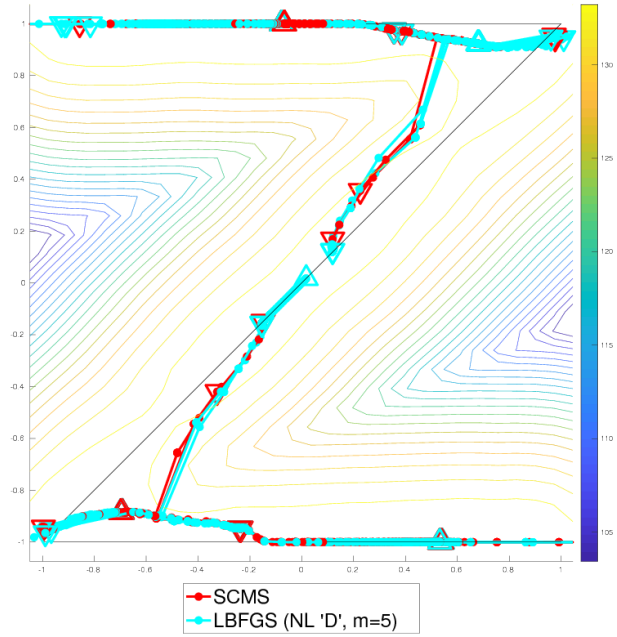
- Run time $\propto n' Dm + m^2$ / iteration (instead of nD^2)
- Storage $\propto 2mD$

- Much more accurate

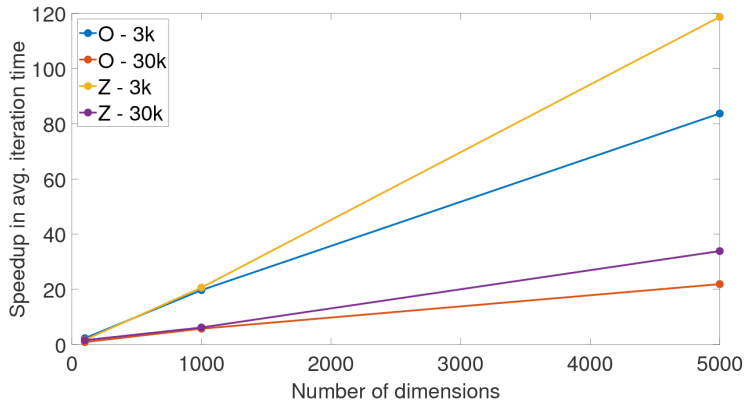
Principal curves SCMS vs. L-SCMS

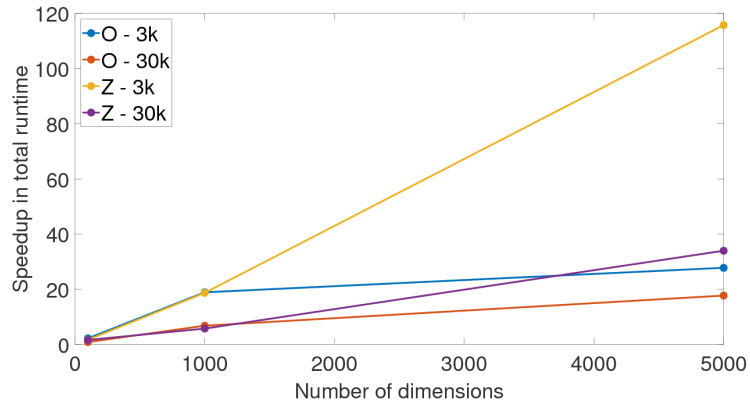


Speedup per iteration



Total speedup w.r.t. SCMS





For large n neighbor search dominates