Dimension Reduction from the User's Perspective

Understanding the configuration spaces of molecules with Manifold Learning

Marina Meilă

University of Washington
mmp@stat.washington.edu
Klaus-Robert Muller, Oles Isayev, Reinhard Maurer, Alexandre Tkatchenko, Stefan Chmiela, Feliks Nuske, Gabor Csany, Cecilia Clementi, Jim Pfaendtner, Chris Fu, Frank Noe, Risi Kondor, Dima Shlyakhtenko, Christof Schutte, Hugh Hillhouse

Machine Learning and Informatics for Chemistry and Materials
Workshop 3: Collective Variables and Coarse Grained Models
EARTH DAY April 22, 2024

Unsupervised learning for the sciences – how do we know machine learning is right?

Unsupervised learning for the sciences – how do we know machine learning is right?

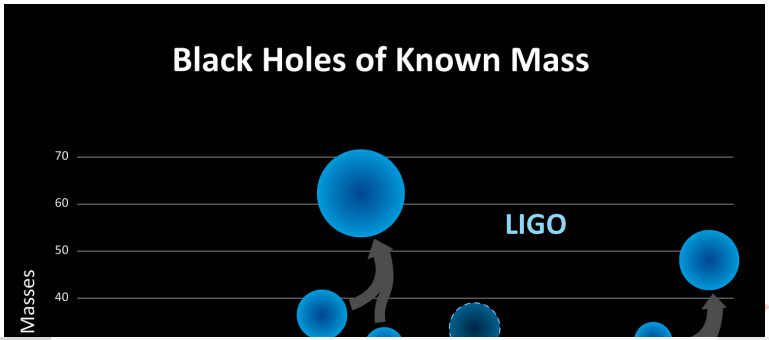
- Success of modern AI:
  - driven by predicting and acting
  - clear error measure
  - validation "easy" (e.g. speech recognition)
  - many local optima
- Unsupervised learning: clustering, dimension reduction
  - finding [geometric, causal] structure of data
  - formulating "error measure" is part of the problem
  - validation can be EXPENSIVE
  - uniqueness of solution matters

Unsupervised learning for the sciences – how do we know machine learning is right?

- Success of modern AI:
  - driven by predicting and acting
  - clear error measure
  - validation "easy" (e.g. speech recognition)
  - many local optima
- Unsupervised learning: clustering, dimension reduction
  - finding [geometric, causal] structure of data
  - formulating "error measure" is part of the problem
  - validation can be EXPENSIVE
  - uniqueness of solution matters
- Big scientific data
  - Allows us to ask more detailed questions (e.g "personalized medicine")
  - Big data contains more complex patterns
  - Machine Learning discovers patterns fast

Unsupervised learning for the sciences – how do we know machine learning is right?

- Success of modern AI:
  - driven by predicting and acting
  - clear error measure
  - validation "easy" (e.g. speech recognition)
  - many local optima
- Unsupervised learning: clustering, dimension reduction
  - finding [geometric, causal] structure of data
  - formulating "error measure" is part of the problem
  - validation can be EXPENSIVE
  - uniqueness of solution matters
- Big scientific data
  - Allows us to ask more detailed questions (e.g "personalized medicine")
  - Big data contains more complex patterns
  - Machine Learning discovers patterns fast
- Often Hypotheses are cheap, experiments are expensive
- Validation is the bottleneck



Drowning in hypotheses...

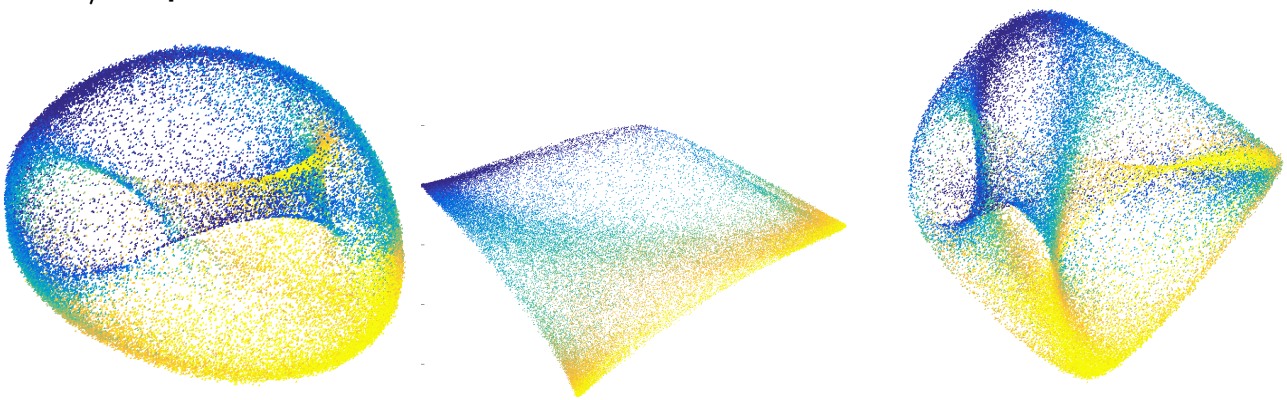
Validation is the bottleneck

- Validation by visualization
- is qualitative not quantitative

Drowning in hypotheses...

Validation is the bottleneck

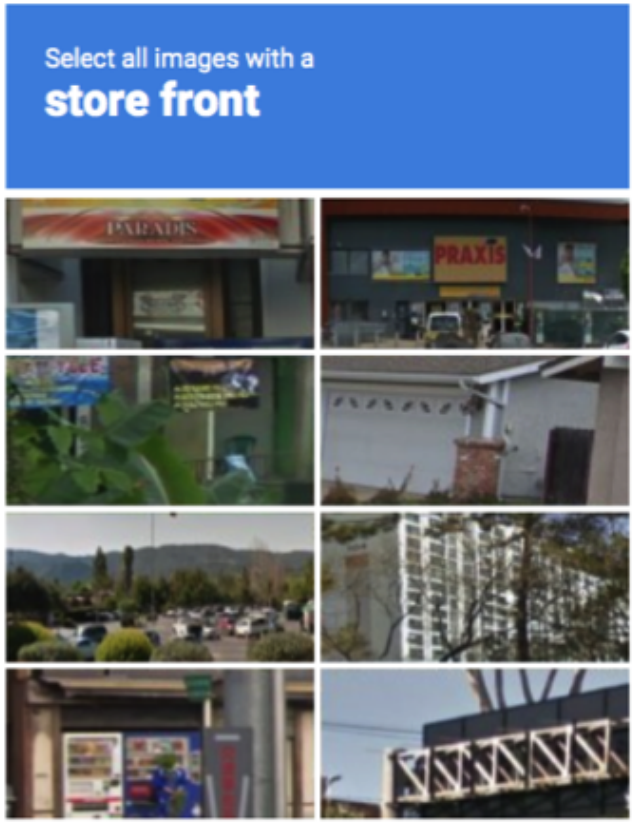
- Validation by visualization
- is qualitative not quantitative
- hard/impossible in dimension > 3

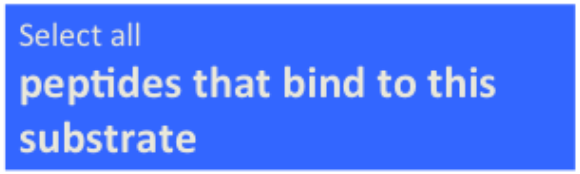


Drowning in hypotheses...

Validation is the bottleneck

- Validation by visualization
- is qualitative not quantitative
- hard/impossible in dimension > 3
- can't be crowdsourced





Drowning in hypotheses...

Validation is the bottleneck

- Validation by visualization
- is qualitative not quantitative
- hard/impossible in dimension > 3
- can't be crowdsourced
- discovering what is known?

1. When to do (non-linear) dimension reduction

2. The meat: Manifold learning algorithms
   - E-vector based embedding algorithms
   - Repulsion-based algorithms

3. The sandwich: distortions, artefacts, parameters

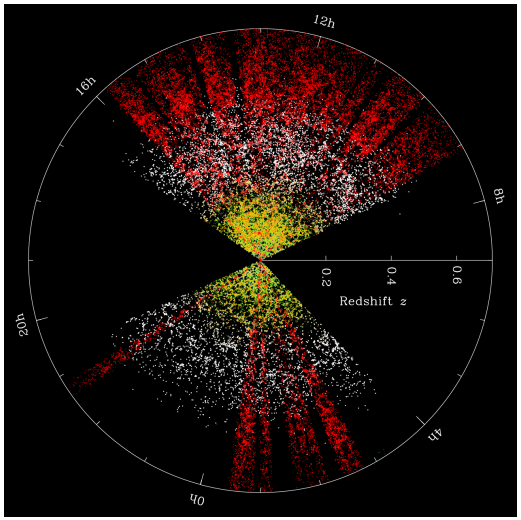
4. Metric Manifold Learning

5. Experiments with small molecules
   - What distance to use?
   - What graph? Radius-neighbors vs. k nearest-neighbors
   - Clustering vs. Embedding?
   - Manifold coordinates with physical meaning

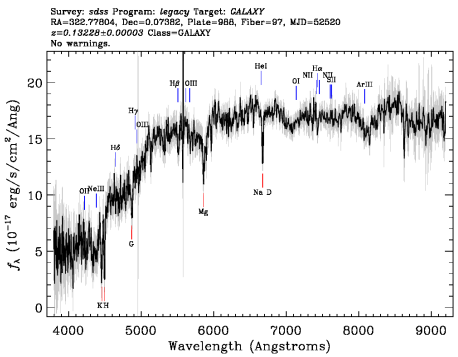
Outline

Spectra of galaxies measured by the Sloan Digital Sky Survey (SDSS)

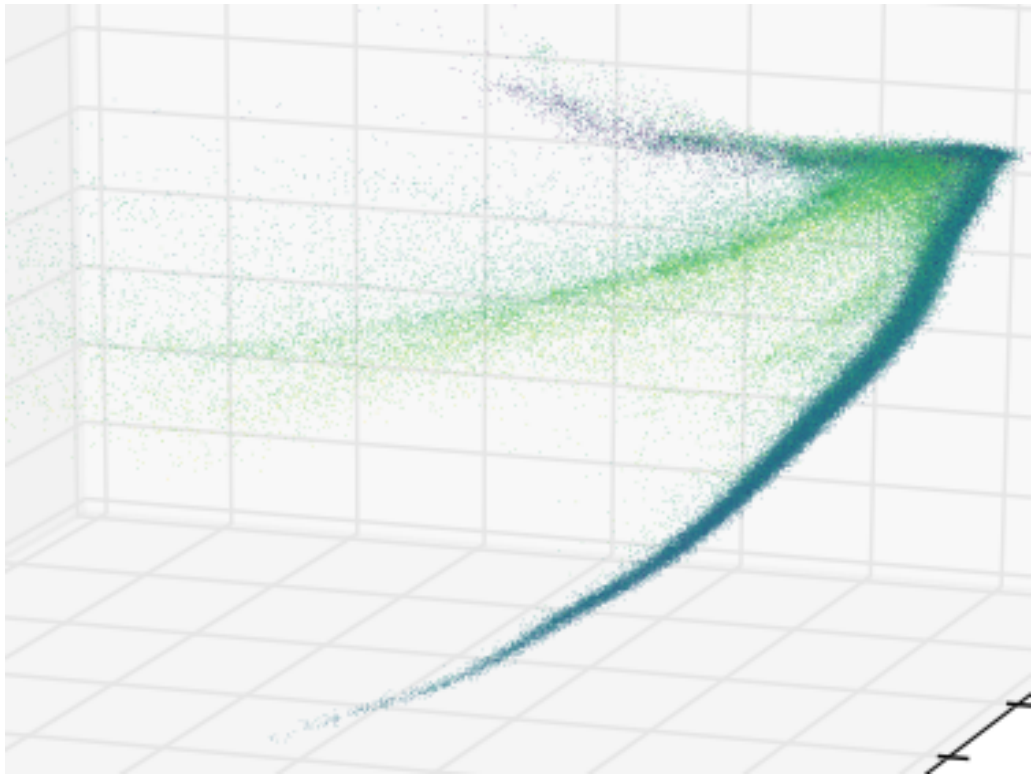
www.sdss.org



www.sdss.org



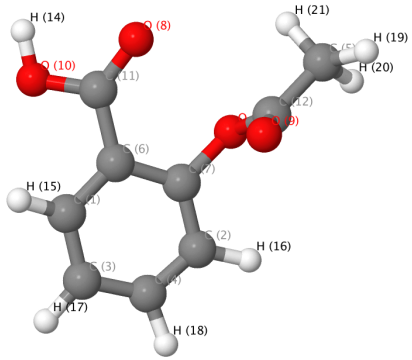
- Preprocessed by Jacob VanderPlas and Grace Telford
- $n = 675,000$ spectra $\times$ $D = 3750$ dimensions

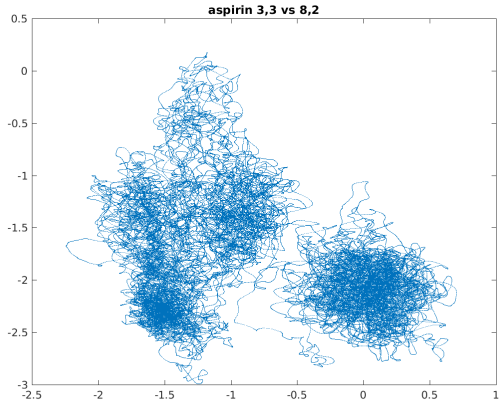


embedding by James McQueen

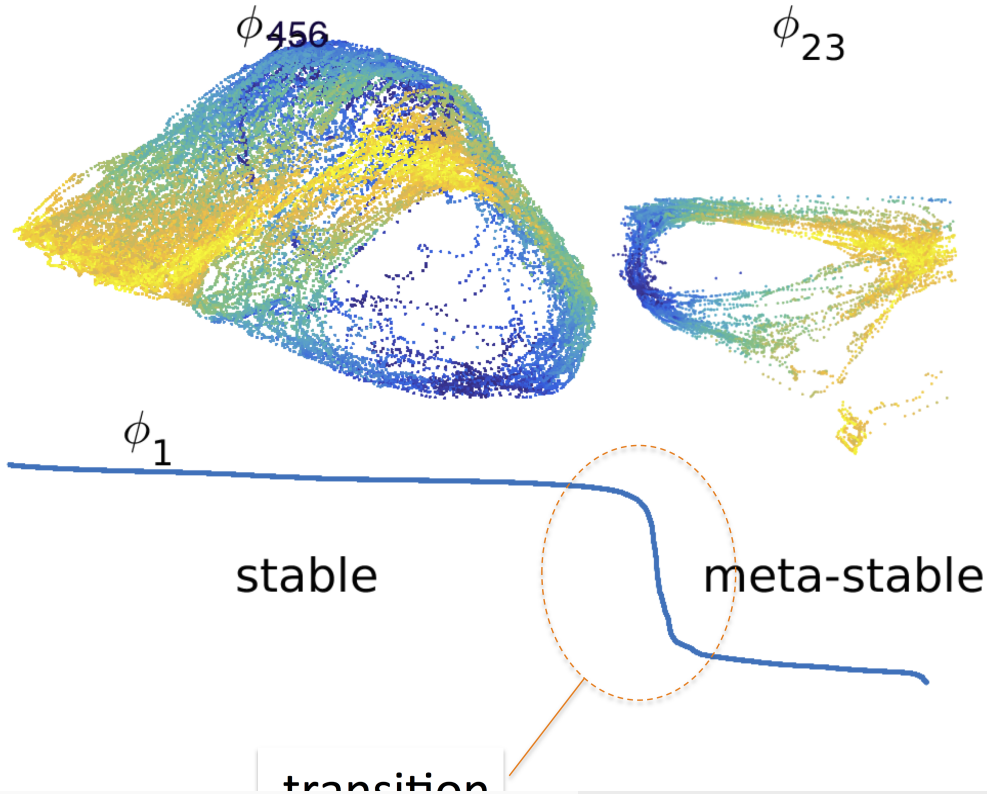
Molecular configurations

aspirin molecule





- Data from Molecular Dynamics (MD) simulations of small molecules by [Chmiela et al. 2016]
- $n \approx 200,000$ configurations $\times$ $D \sim 20 - 60$ dimensions



Manifold Learning (ML) for the physical sciences

- big, high-dimensional data
- data, physics supports manifold models
- understanding more important than prediction

- Simple assumption data i.i.d. from manifold $\mathcal{M}$ (no noise)

Manifold Learning (ML) for the physical sciences

- big, high-dimensional data
- data, physics supports manifold models
- understanding more important than prediction

- Simple assumption data i.i.d. from manifold $\mathcal{M}$ (no noise)

Challenges for ML algorithms

- scalable `megaman` ML package [McQueen et al JMLR 2015]
- find "something new, trustworthy, reproducible, interpretable"

Manifold Learning (ML) for the physical sciences

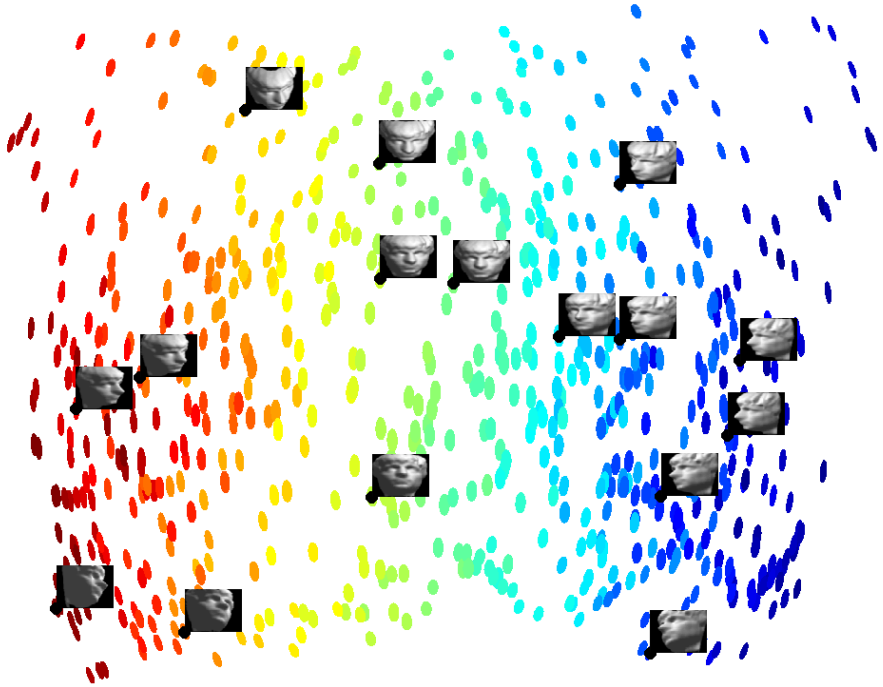
- big, high-dimensional data
- data, physics supports manifold models
- understanding more important than prediction

- Simple assumption data i.i.d. from manifold $\mathcal{M}$ (no noise)

Challenges for ML algorithms

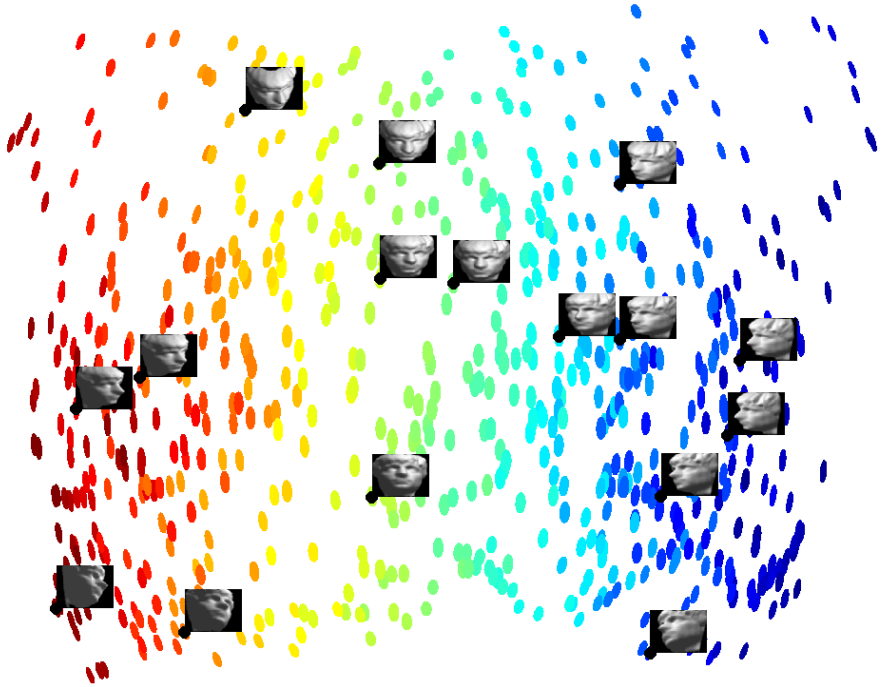
- scalable `megaman` ML package [McQueen et al JMLR 2015]
- find "something new, trustworthy, reproducible, interpretable"
- remove algorithmic artefacts
- data-driven parameter selection (replace grad student)
- validation on mathematical/statistical grounds as much as possible (replace experimental validation)
- use domain knowledge not domain expert

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



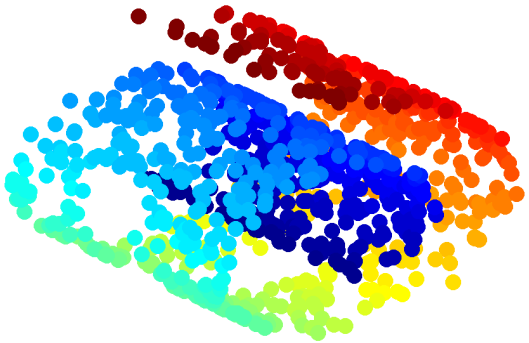
Why?

- 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

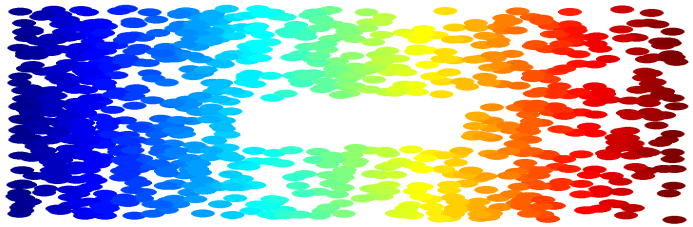
Outline

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

Input
points in $D \geq 3$ dimensions

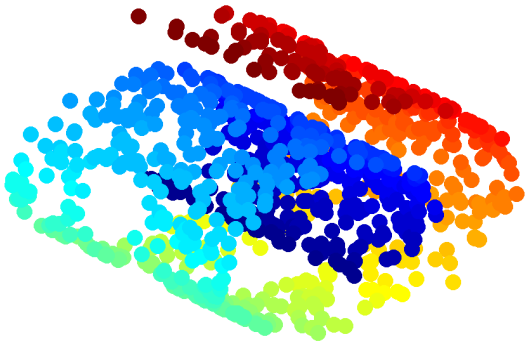


Desired output
same points reparametrized in 2D

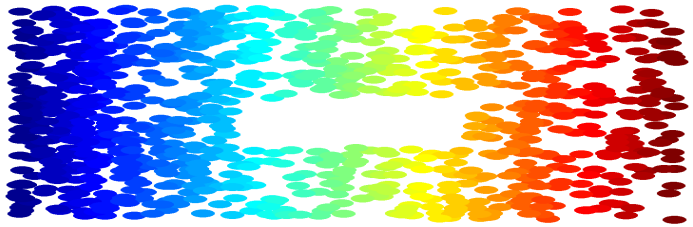


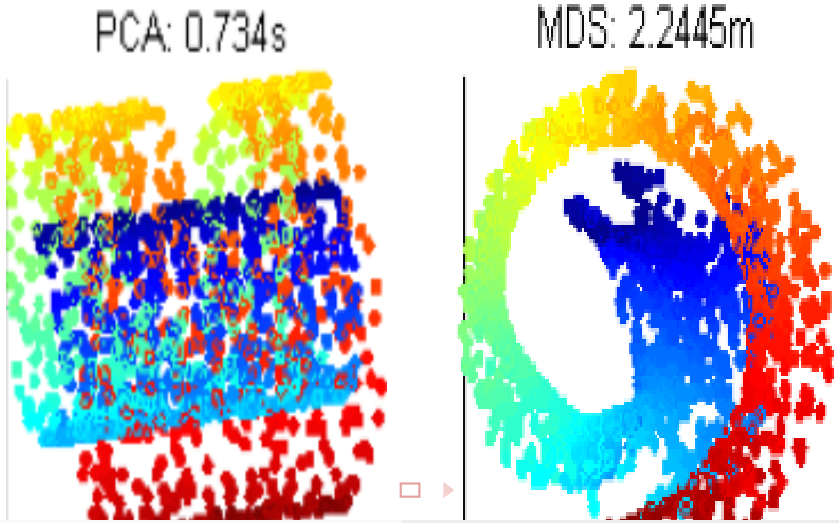
A toy example (the "Swiss Roll" with a hole)

Input
points in $D \geq 3$ dimensions



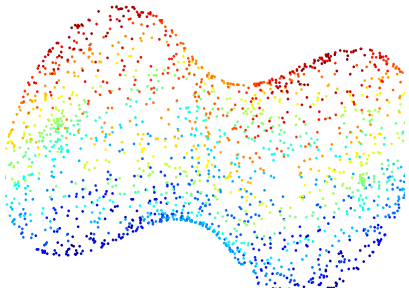
Desired output
same points reparametrized in 2D



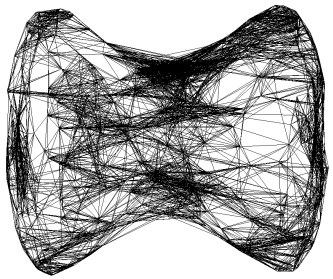


Neighborhood graphs

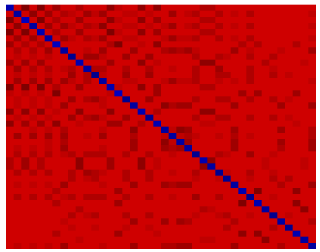
- All ML algorithms start with a neighborhood graph over the data points
  - neigh_i denotes the neighbors of p_i



data $\xi_1, \ldots \xi_n \subset \mathbb{R}^D$



neighborhood graph



A (sparse) matrix of distances between neighbors

The Isomap algorithm

Isomap Algorithm [Tennenbaum, deSilva & Langford 00]

Input A, dimension d

1. Find all shortest path distances in neighborhood graph

   if $A_{ij} = \infty$, then $A_{ij} \leftarrow$ graph distance between i, j
2. Construct matrix of squared distances

$$M = [(A_{ij})^2]$$

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

- Works also for $m > d$

The Diffusion Maps (DM)/ Laplacian Eigenmaps (LE) Algorithm

Diffusion Maps Algorithm

Input distance matrix $A \in \mathbb{R}^{n \times n}$, bandwidth ϵ, embedding dimension m

1. Compute Laplacian $L \in \mathbb{R}^{n \times n}$
2. Compute eigenvectors of L for smallest $m+1$ eigenvalues $[\phi_0\, \phi_1\, \ldots \phi_m] \in \mathbb{R}^{n \times m}$
   - ϕ_0 is constant and not informative

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

The (renormalized) Laplacian

Laplacian

Input distance matris $A \in \mathbb{R}^{n \times n}$, bandwidth ϵ

1. Compute similarity matrix $S_{ij} = \exp\left(-\frac{A_{ij}^2}{\epsilon^2}\right) = \kappa(A_{ij}/\epsilon)$
2. Normalize columns $d_j = \sum_{i=1}^n S_{ij}$, $\tilde{L}_{ij} = S_{ij}/d_j$
3. Normalize rows $d_i' = \sum_{j=1}^n \tilde{L}_{ij}$, $P_{ij} = \tilde{L}_{ij}/d_i'$
4. $L = \frac{1}{\epsilon^2}(I - P)$
5. Output L, d_i'/d_i

The (renormalized) Laplacian

Laplacian

Input distance matris $A \in \mathbb{R}^{n \times n}$, bandwidth ϵ

1. Compute similarity matrix $S_{ij} = \exp\left(-\frac{A_{ij}^2}{\epsilon^2}\right) = \kappa(A_{ij}/\epsilon)$
2. Normalize columns $d_j = \sum_{i=1}^n S_{ij}$, $\tilde{L}_{ij} = S_{ij}/d_j$
3. Normalize rows $d_i' = \sum_{j=1}^n \tilde{L}_{ij}$, $P_{ij} = \tilde{L}_{ij}/d_i'$
4. $L = \frac{1}{\epsilon^2}(I - P)$
5. Output L, d_i'/d_i

- Laplacian L central to understanding the manifold geometry
- $\lim_{n\to\infty} L = \Delta_{\mathcal{M}}$ [Coifman,Lafon 2006]
- Renormalization trick cancels effects of (non-uniform) sampling density [Coifman & Lafon 06]

The (renormalized) Laplacian

Laplacian

Input distance matris $A \in \mathbb{R}^{n \times n}$, bandwidth ϵ

1. Compute similarity matrix $S_{ij} = \exp\left(-\frac{A_{ij}^2}{\epsilon^2}\right) = \kappa(A_{ij}/\epsilon)$
2. Normalize columns $d_j = \sum_{i=1}^n S_{ij}$, $\tilde{L}_{ij} = S_{ij}/d_j$
3. Normalize rows $d'_i = \sum_{j=1}^n \tilde{L}_{ij}$, $P_{ij} = \tilde{L}_{ij}/d'_i$
4. $L = \frac{1}{\epsilon^2}(I - P)$
5. Output L, d'_i/d_i

- Laplacian L central to understanding the manifold geometry
- $\lim_{n\to\infty} L = \Delta_{\mathcal{M}}$ [Coifman,Lafon 2006]
- Renormalization trick cancels effects of (non-uniform) sampling density [Coifman & Lafon 06]

Other Laplacians

- $L^{un} = \text{diag}\{d_{1:n}\} - A$ unnormalized Laplacian
- $L^{rw} = I - \text{diag}\{d_{1:n}\}^{-1} A$ random walk Laplacian
- $L^{n} = I - \text{diag}\{d_{1:n}\}^{-1/2} A \,\text{diag}\{d_{1:n}\}^{-1/2}$ normalized Laplacian

Repulsion-based (heuristic) algorithms

t-Stochastic Neighbor Embedding (t-SNE)

Input similarity matrix S, embedding dimension s

Init choose embedding points $y_{1:n} \in \mathbb{R}^s$ at random

1. $S_{ii} \leftarrow 0$, normalize rows $d_i = \sum_j S_{ij}$, $P_{ij} = S_{ij}/d_i$
2. symmetrize $P = \frac{1}{2n}(P + P^T)$ P is distribution over pairs of neighbors (i, j)
3. $\tilde{S}_{ij} = \tilde{\kappa}(\|y_i - y_j\|)$compute similarity in output space
   where $\tilde{\kappa}(z) = \frac{1}{1+z^2}$ the Cauchy (Student t with 1 degree of freedom)
4. Define distribution Q with $Q_{ij} \propto S_{ij}$
5. Change $y_{i:n}$ to decrease the Kullbach-Leibler divergence $KL(P||Q) = \sum_{i,j} P_{ij} \ln \frac{P_{ij}}{Q_{ij}}$ (by gradient descent) and repeat from step 1

Repulsion-based (heuristic) algorithms

t-Stochastic Neighbor Embedding (t-SNE)

Input similarity matrix S, embedding dimension s

Init choose embedding points $y_{1:n} \in \mathbb{R}^s$ at random

1. $S_{ii} \leftarrow 0$, normalize rows $d_i = \sum_j S_{ij}$, $P_{ij} = S_{ij}/d_i$
2. symmetrize $P = \frac{1}{2n}(P + P^T)$ P is distribution over pairs of neighbors (i, j)
3. $\tilde{S}_{ij} = \tilde{\kappa}(\|y_i - y_j\|)$compute similarity in output space
   where $\tilde{\kappa}(z) = \frac{1}{1+z^2}$ the Cauchy (Student t with 1 degree of freedom)
4. Define distribution Q with $Q_{ij} \propto S_{ij}$
5. Change $y_{i:n}$ to decrease the Kullbach-Leibler divergence $KL(P||Q) = \sum_{i,j} P_{ij} \ln \frac{P_{ij}}{Q_{ij}}$ (by gradient descent) and repeat from step 1

- empirically useful for visualizing clusters (repulsion encourages cluster formation)
- non-deterministic, more parameters

UMAP: Uniform Manifold Approximation and Projection

UMAP [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

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

Output ϕ

UMAP: Uniform Manifold Approximation and Projection

UMAP [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

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

Output ϕ

Embedding algorithms summary

E-vector based

- DiffusionMaps
- Isomap
- LTSA [Zhang, Zha, 2004]
- ...

\+ well studied, params better understood

− collapsed embeddings, "horseshoes" possible

- require embedding dimension $s \geq d$

Repulsion based

- t-SNE
- UMAP
- ...

− heuristic, more parameters

\+ no collapsed embeddings

- $s = d$ intrinsic dimension

Embedding algorithms summary

E-vector based

- DiffusionMaps
- Isomap
- LTSA [Zhang, Zha, 2004]
- ...

\+ well studied, params better understood

− collapsed embeddings, "horseshoes" possible

- require embedding dimension $s \geq d$

Repulsion based

- t-SNE
- UMAP
- ...

− heuristic, more parameters

\+ no collapsed embeddings

- $s = d$ intrinsic dimension

- More importantly - sensitive to
  - neighborhoods scale ϵ and type K-nn vs. spherical
  - non-uniformity of point distribution
  - aspect ratio (E-vector based)

Embedding algorithms summary

E-vector based

- DiffusionMaps
- Isomap
- LTSA [Zhang, Zha, 2004]
- ...

\+ well studied, params better understood

− collapsed embeddings, "horseshoes" possible

- require embedding dimension $s \geq d$

Repulsion based

- t-SNE
- UMAP
- ...

− heuristic, more parameters

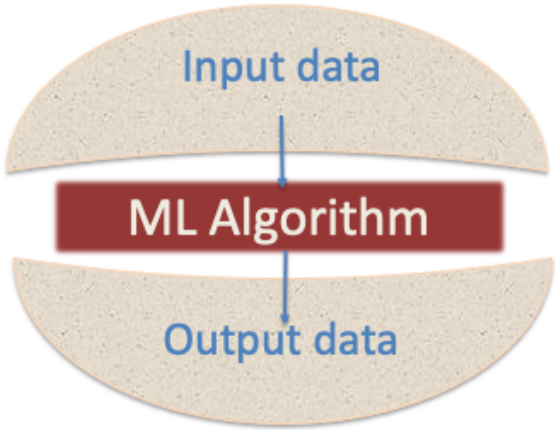
\+ no collapsed embeddings

- $s = d$ intrinsic dimension

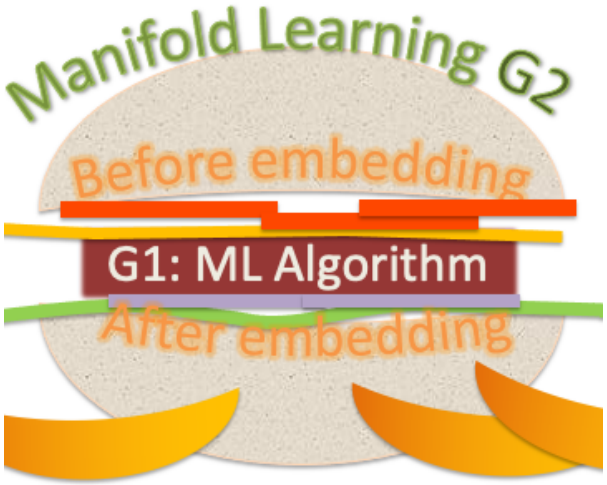
- More importantly - sensitive to
  - neighborhoods scale ϵ and type K-nn vs. spherical
  - non-uniformity of point distribution
  - aspect ratio (E-vector based)
- Almost always embedding algorithms distort shape of data

Outline

Manifold Learning as a sandwich



Manifold Learning as a sandwich



- what distance measure?
- what graph? [Maier,von Luxburg, Hein 2009]
- what kernel width ϵ? [Perrault-Joncas,M,McQueen NIPS17]
- what intrinsic dimension d? [Chen,Little,Maggioni,Rosasco]

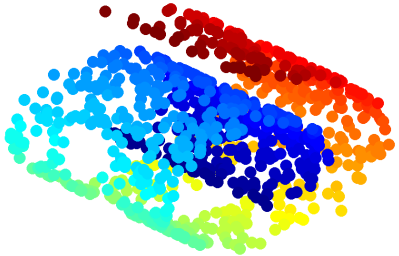
ML Algorithm: DiffMaps, LTSA, t-SNE

- Cluster [M,Shi 00],[M,Shi 01]. . . [M NeurIPS18]
- Estimate & correct distortion: Metric Learning, Riemannian Relaxation [McQueen, M, Perrault-Joncas NIPS16]
- Validate d, s, [select eigenvectors] [Chen, M NeurIPS19]
- Topological Data Analysis (TDA)
- Meaning of coordinates [Koelle,Zhang, M, Chen 2022] this Workshop

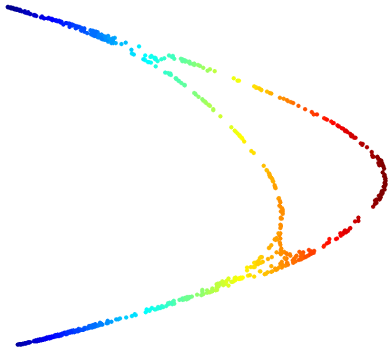
. . .

Distortions, failures, irreproducibility

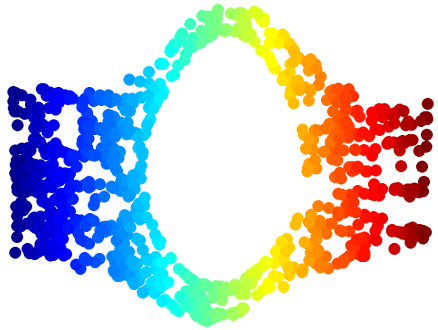
Original data
(Swiss Roll with hole)



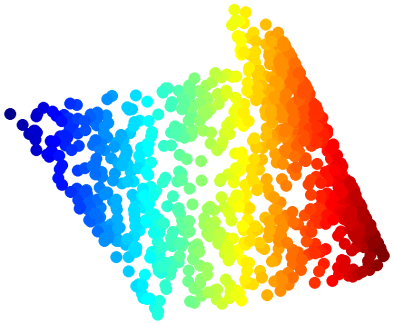
DiffMaps



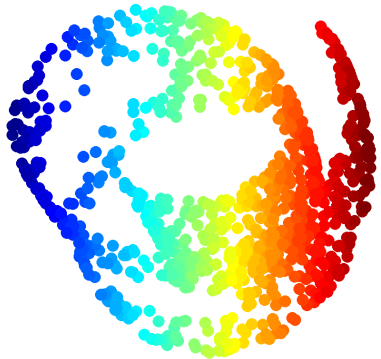
Isomap



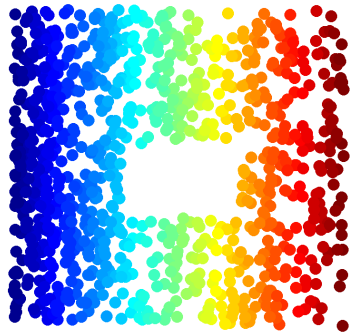
Hessian Eigenmaps (HE)



Local Linear Embedding
(LLE)

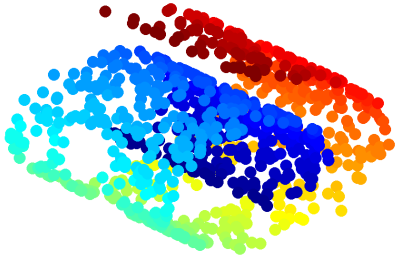


Local Tangent Space
Alignment (LTSA)

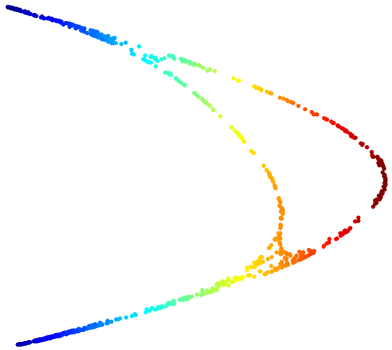


Distortions, failures, irreproducibility

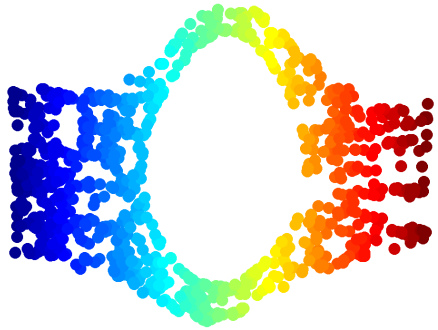
Original data
(Swiss Roll with hole)



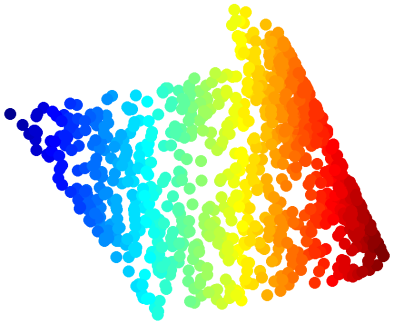
DiffMaps



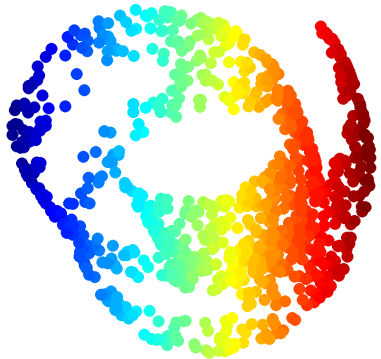
Isomap



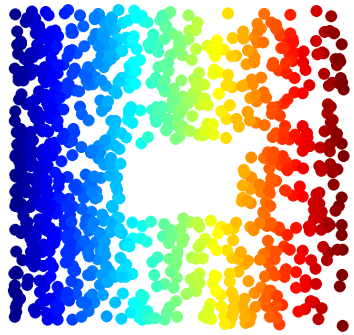
Hessian Eigenmaps (HE)



Local Linear Embedding
(LLE)



Local Tangent Space
Alignment (LTSA)



- fail to preserve neighborhoods LLE, DiffMaps, HE – must diagnose
- distortion Isomap (non-linear), LTSA (linear) – must recognize equivalence, correct

Distortions vs. Failures

- ϕ **distorts** if distances, angles, density not preserved, but ϕ smooth and invertible
- ϕ **fails** if it does not preserve topology (=preserve neighborhoods)
  - discontinuous deformation
  - non-invertible ϕ
  - breaks/collapses neighborhoods
  - collapse in dimension (local or global)

Distortions vs. Failures

- ϕ **distorts** if distances, angles, density not preserved, but ϕ smooth and invertible
- ϕ **fails** if it does not preserve topology (=preserve neighborhoods)
  - discontinuous deformation
  - non-invertible ϕ
  - breaks/collapses neighborhoods
  - collapse in dimension (local or global)

Most common modes of failure

- distance matrix A does not capture topology (artificial "holes" or "bridges")
- usually becasuse neighborhood too small or too large
- for e-vector based algorithms: choice of e-vectors

Artefacts

- Artefacts=features of the embedding that do not exist in the data (clusters, holes, "arms", "horseshoes")

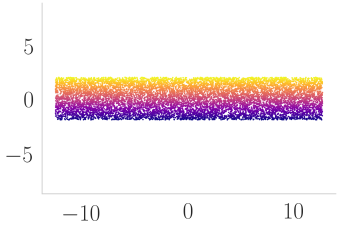
Artefacts

- Artefacts=features of the embedding that do not exist in the data (clusters, holes, "arms", "horseshoes")

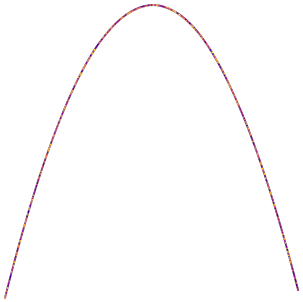
- What to beware of when you compute an embedding
  - algorithms that claim to choose ϵ automatically
  - confirming the embedding is "correct" by visualization: tends to over-smooth, i.e. ϵ over-estimated
  - K-nn (default in `sk-learn`!) instead of radius-neighbors: tends to create clusters
  - large variations in density: subsample data to make it more uniform
  - "horseshoes": choose other e-vectors (ϕ is almost singulare)

- Very popular heuristics: LLE, t-SNE, UMAP, neural networks more prone to artefacts

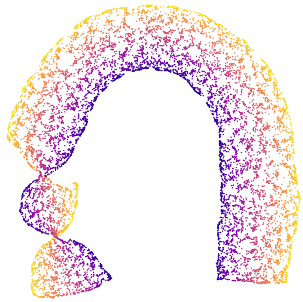
Horseshoes and the Repeated Eigenvector Problem



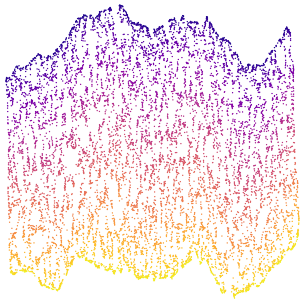
DiffMaps ϕ_1, ϕ_2



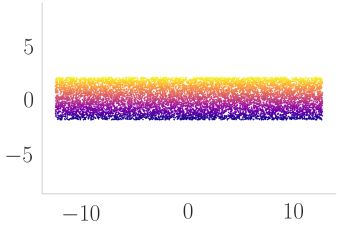
UMAP ϕ_1, ϕ_2



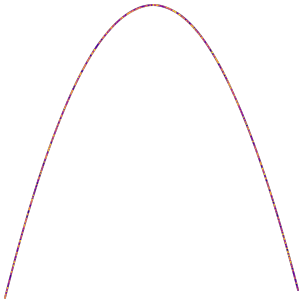
[Chen, M 2019] + UMAP

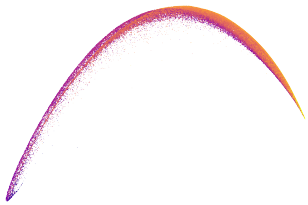


Horseshoes and the Repeated Eigenvector Problem

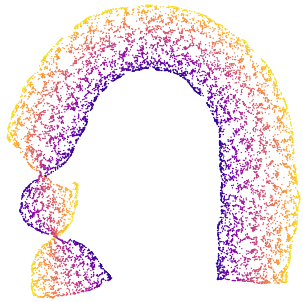


DiffMaps ϕ_1, ϕ_2



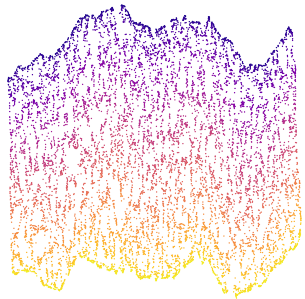


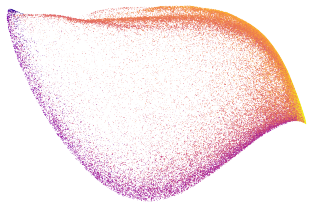
UMAP ϕ_1, ϕ_2



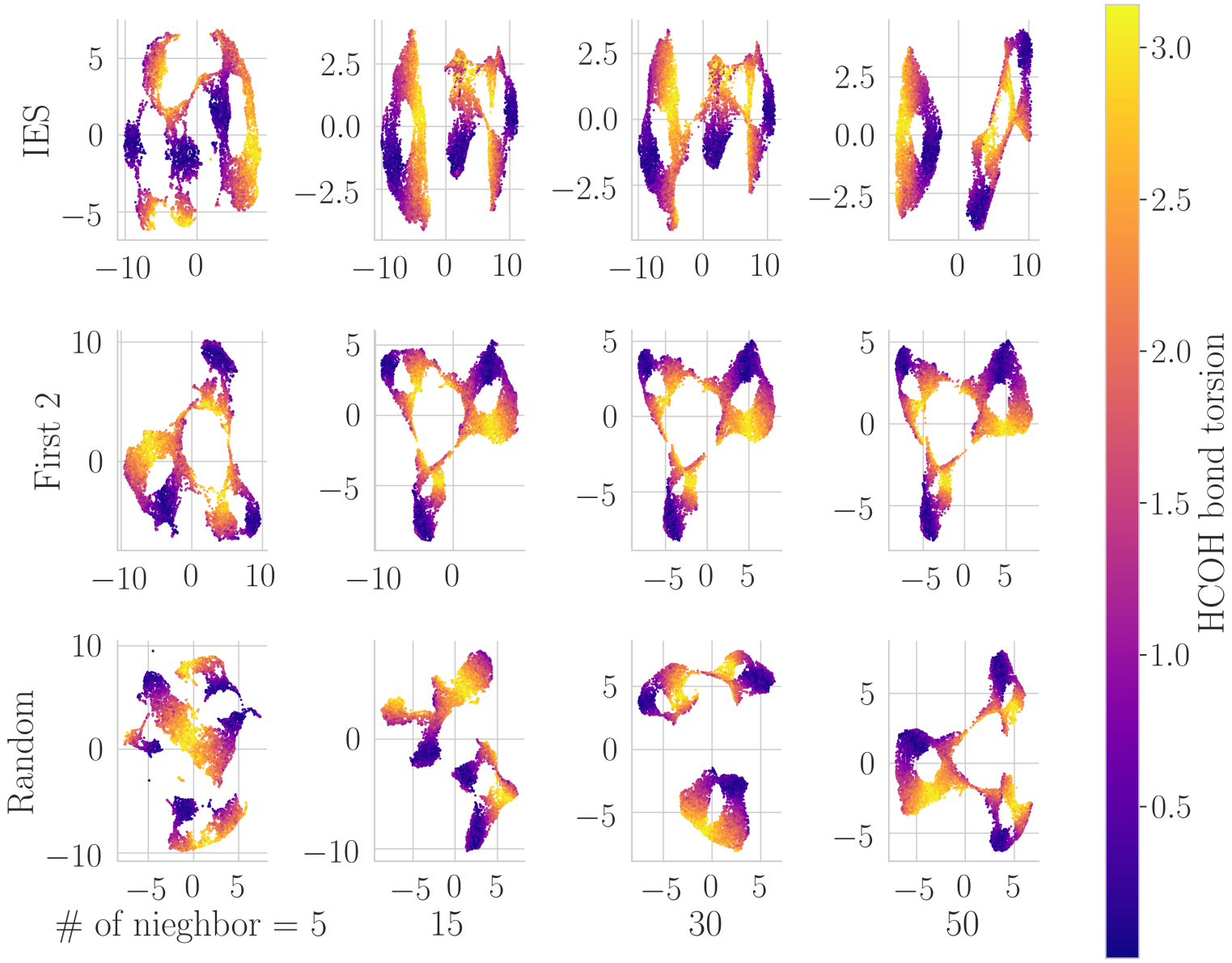
SDSS Galaxy Spectra

[Chen, M 2019] + UMAP





UMAP for ethanol

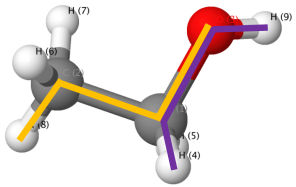


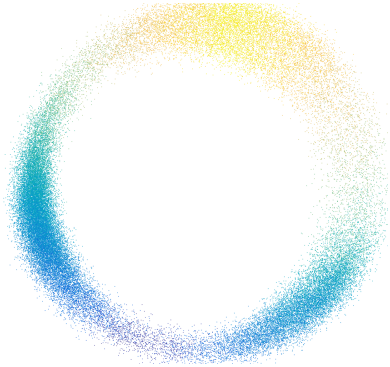
Initialization

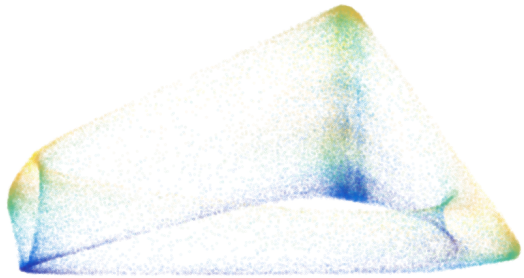
Outline

Metric Manifold Learning

- Most embeddings are distorted

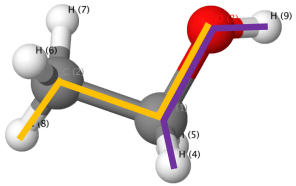


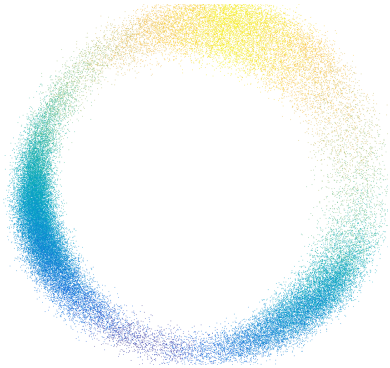


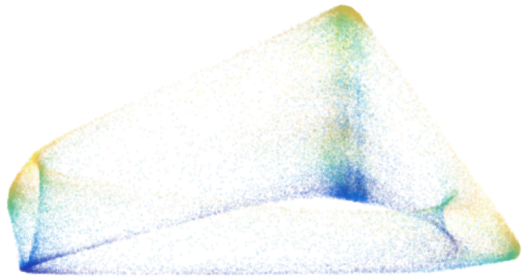


Metric Manifold Learning

- Most embeddings are distorted



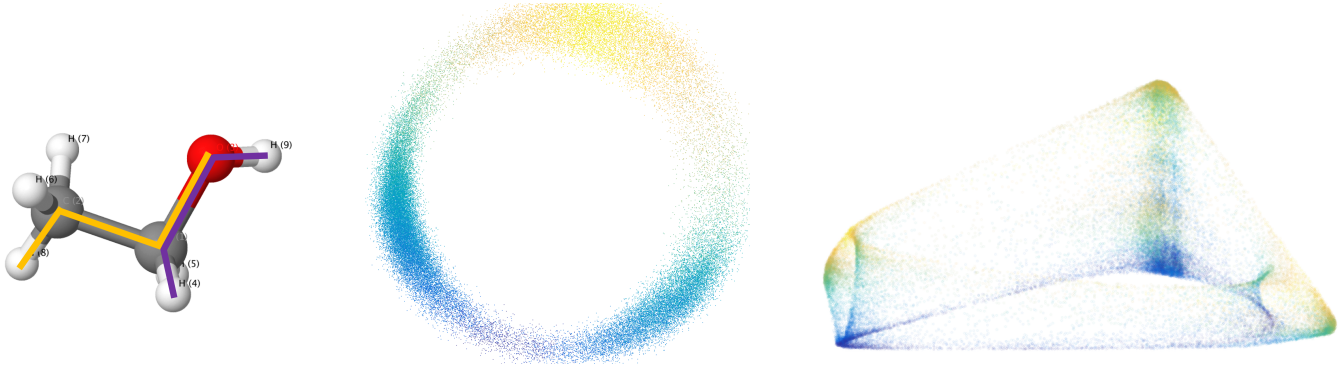




- Distortion depends on algorithm, parameters, data density, . . .

Metric Manifold Learning

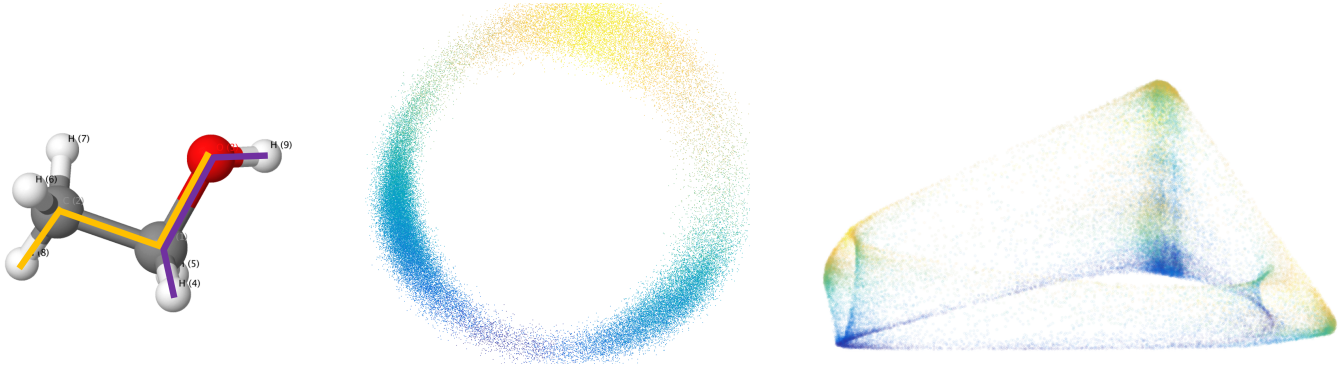
- Most embeddings are distorted



- Distortion depends on algorithm, parameters, data density, . . .
- How to fix?
- Isometric embedding eliminates distortion (hard)
- Metric learning estimate distortions (easy)

Metric Manifold Learning

- Most embeddings are distorted



- Distortion depends on algorithm, parameters, data density, ...

- How to fix?

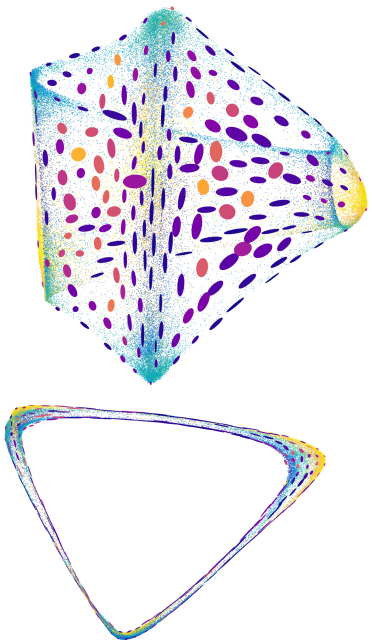
- Isometric embedding eliminates distortion (hard)
- Metric learning estimate distortions (easy) and corrects geometric calculations (easy)

Estimating local distortion. The (push-forward) Riemannian metric

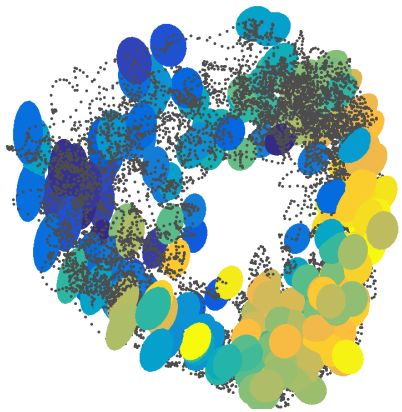
Idea at each point p

- along with embedding coordinates $(\phi_1(p), \ldots \phi_s(p))$
- estimate $H(p)$ $s \times s$ positive definite matrix

- $H(p)$ measures distortion at $\phi(p)$
  can be estimated from the LAPLACIAN
- $G(p) = H^{\dagger}(p)$ is push-forward Riemannian metric



ethanol



aspirin

Metric Manifold Learning summary

= estimate local distortion H at each embedded point $\phi(p)$

Metric Manifold Learning summary

= estimate local distortion H at each embedded point $\phi(p)$

Why useful

- Algorithm independent geometry preserving method

Metric Manifold Learning summary

= estimate local distortion H at each embedded point $\phi(p)$

Why useful

- Algorithm independent geometry preserving method
- Outputs of different algorithms on the same data are comparable

Metric Manifold Learning summary

= estimate local distortion H at each embedded point $\phi(p)$

Why useful

- Algorithm independent geometry preserving method
- Outputs of different algorithms on the same data are comparable

Correcting distortion

- Integrating with the local length element:
  $l(\text{curve}) = \int_a^b \sqrt{\sum_{ij} G_{ij} \frac{dx^i}{dt} \frac{dx^j}{dt}}\, dt,$
- Riemannian Relaxation

true distance $d = 1.57$

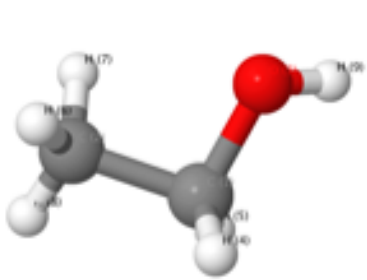
Embedding	Line segment	Shortest Path $d_{\mathcal{G}}$	Metric $\hat{d}$	Rel. err
Orig. data	1.41	1.57	1.62	3.0%
Isomap $s = 2$	1.66	1.75	1.63	3.7%
LTSA $s = 2$	0.07	0.08	1.65	4.8%
LE $s = 2$	0.08	0.08	1.62	3.1%

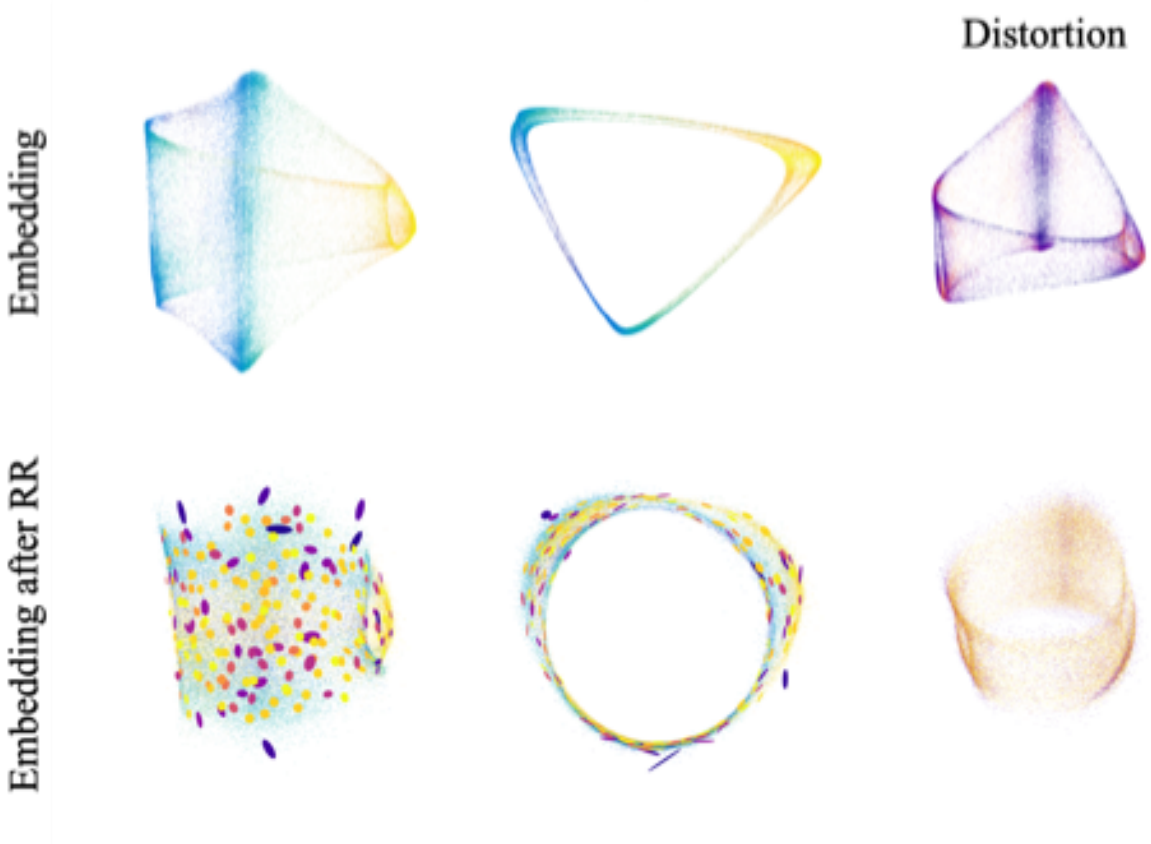
Applications

- Estimation of neighborhood scale [Perrault-Joncas,M,McQueen NIPS17]
- Gaussian Processes on manifolds and semi-supervised learning
- Vector pull-back/push-forward between tangent spaces [Koelle, Zhang, M, Chen 18]
- Fixing the "horseshoe"/collapsed dimension problem [Chen, M, 19]

Riemannian Relaxation for Ethanol molecular configurations

https://sites.stat.washington.



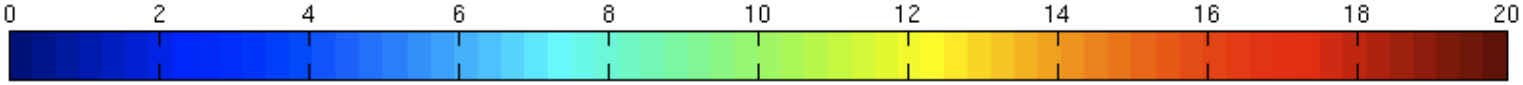


An Application: Gaussian Processes on Manifolds

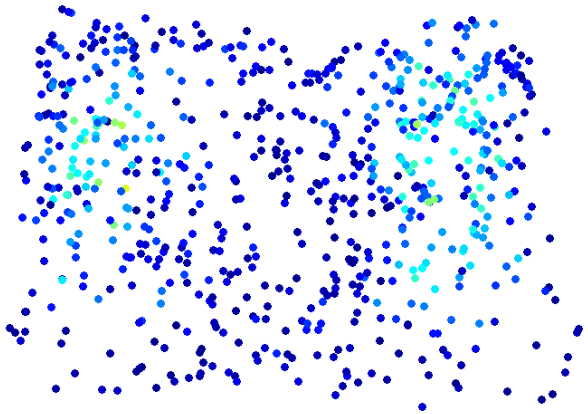
- Gaussian Processes (GP) can be extended to manifolds via SPDE's (Lindberg, Rue, and Lindstrom, 2011)
- Let $\left(\kappa^2 - \Delta_{\mathbb{R}^d}\right)^{\alpha/2} u(x) = \xi$ with ξ Gaussian with noise, then $u(x)$ is a Matérn GP
- Subsituting $\Delta_{\mathcal{M}}$ allows us to define a Matérn GP on $\mathcal{M}$
- Semi-supervised learning: unlabelled points can be learned by Kriging using the Covariance matrix Σ of $u(x)$

Semisupervised learning with Gaussian Processes on Manifolds

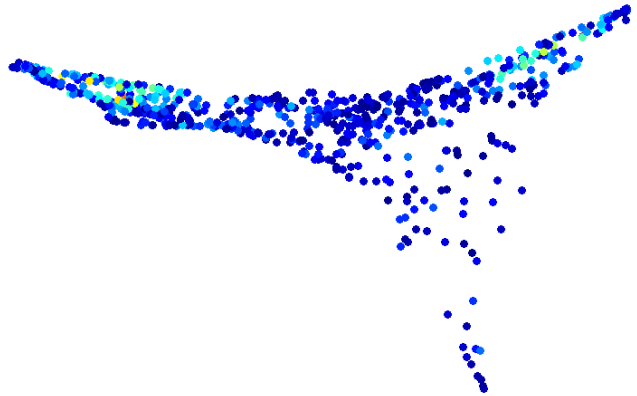
Sculpture Faces: Predicting Head Rotation



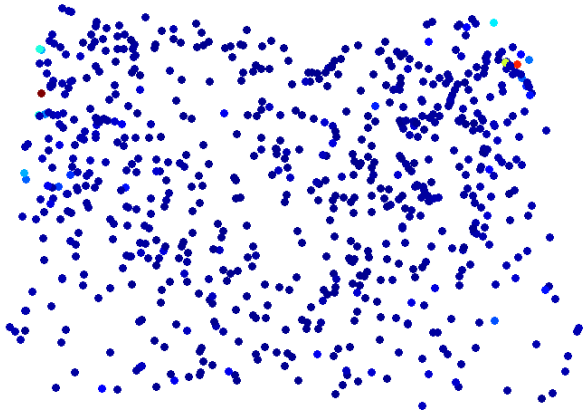
Absolute Error (AE) as percentage of range



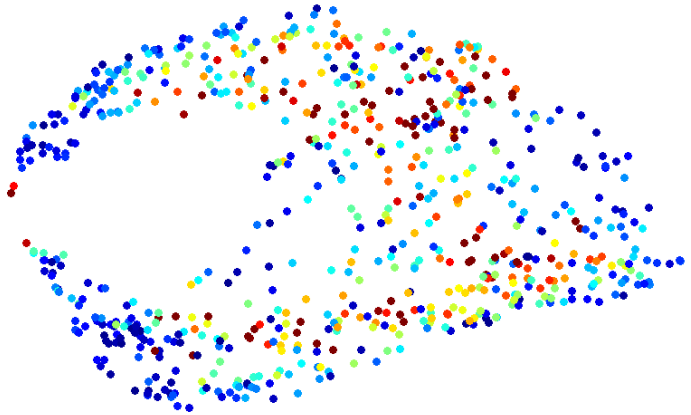
Isomap (Total AE = 3078)



LTSA (Total AE = 3020)



Metric (Total AE = 660)

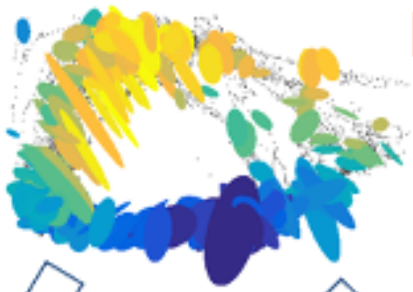


Laplacian Eigenmaps (Total AE = 3078)

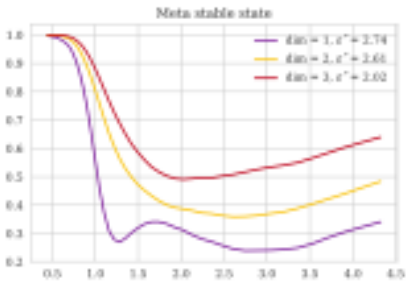
Manifold learning: beyond the embedding algorithm

Correct algorithm distortion

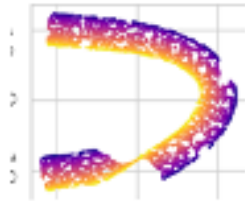
Estimate Riemannian metric

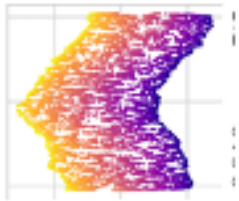


Optimize neighborhood size [NIPS 2016]



Choose independent e-vectors [NeurIPS 2019]

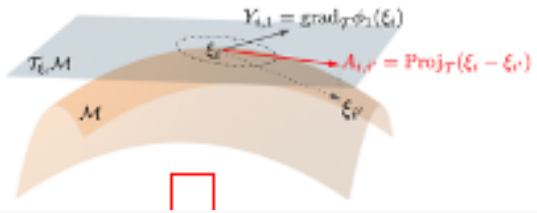




Distances, angles, areas preserved

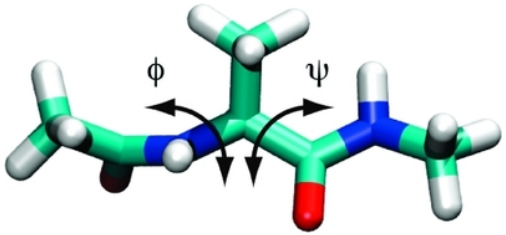
Riemannian relaxation [NIPS 2015]

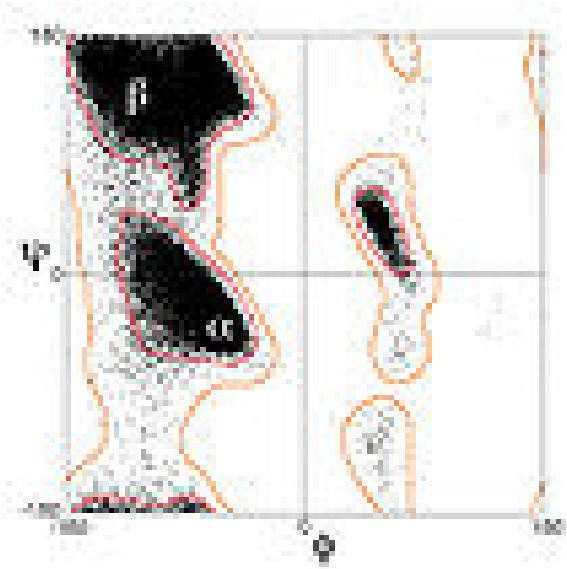
Vector fields preserved



Outline

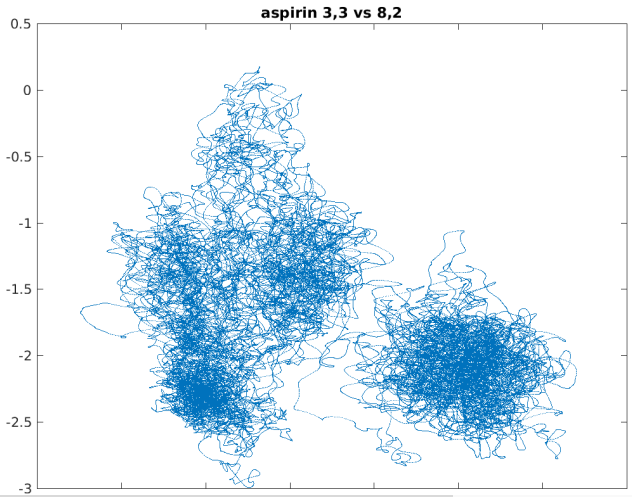
Data: from MD simulations

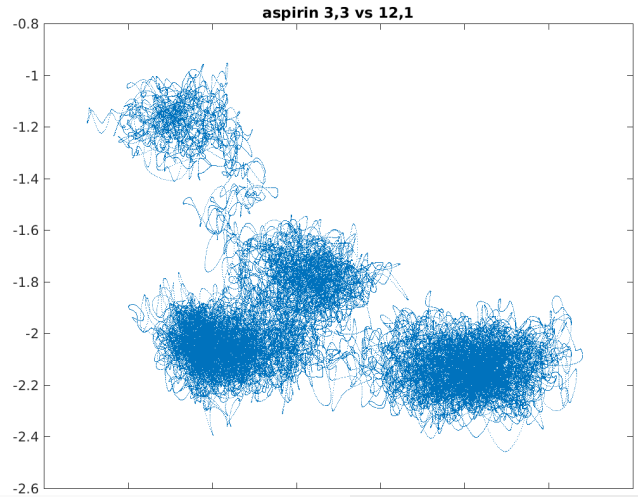




MD Data from [Chmiela et al. 2017]

- Ex. $n = 10^4 - -10^5$ configurations of Ethanol, Malonaldehide, Aspirin





What distance?

Procrus Align all configurations by a rigid transformation

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

Angles For A, B, C atoms, compute 2 angles of triangle ABC (loses the scale information!)

What distance?

Procrus Align all configurations by a rigid transformation

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

Angles For A, B, C atoms, compute 2 angles of triangle ABC (loses the scale information!)

- Follow up with PCA to remove residual linear dependencies

What distance?

Procrus Align all configurations by a rigid transformation

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

Angles For A, B, C atoms, compute 2 angles of triangle ABC (loses the scale information!)

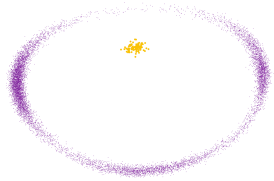
- Follow up with PCA to remove residual linear dependencies

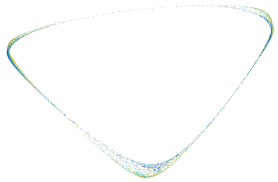
Specialized representations and kernels instead of distances?

+ Coulomb Matrix, SLATM, SOAP, MACE, ... are natural feature spaces
+ take into account atomic species
− mapping may be discontinous
− symmetry invariant kernels change topology

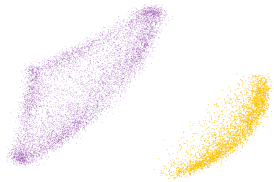
- Unsolved questions: preserving topology (or not?), dependence on distance (or kernel), symmetries

Procrustes ethanol, PCA

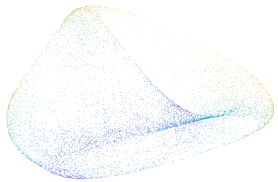




malonaldehyde, PCA

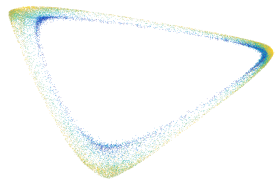


malonaldehyde, DiffMaps

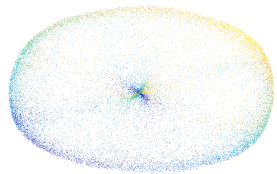


BondLen

ethanol, DiffMaps

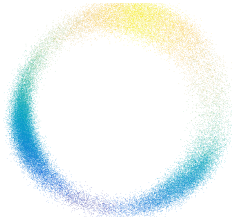


malonaldehyde, DiffMaps

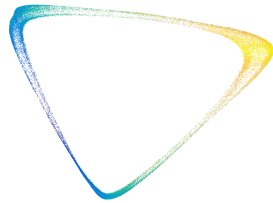


Angles

ethanol, PCA

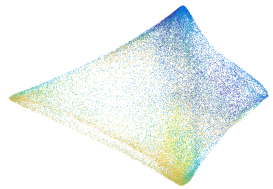


ethanol, DiffMaps



malonaldehyde, PCA

malonaldehyde, DiffMaps



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?

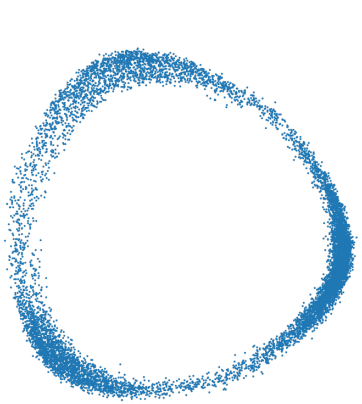
What graph? Radius-neighbors vs. k nearest-neighbors

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

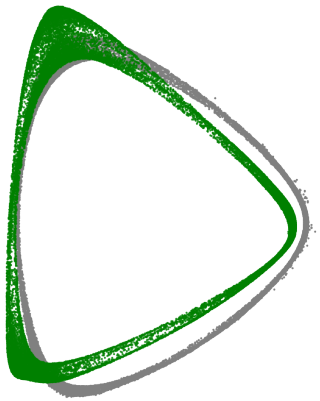
- Does it matter?

- 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 – counters the variable density effects

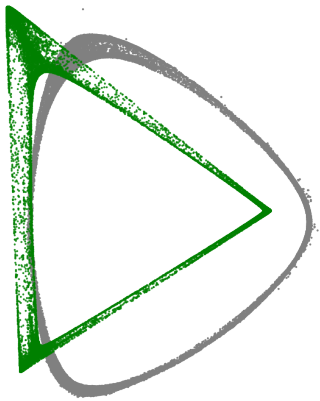
configurations of ethanol $d = 2$



original

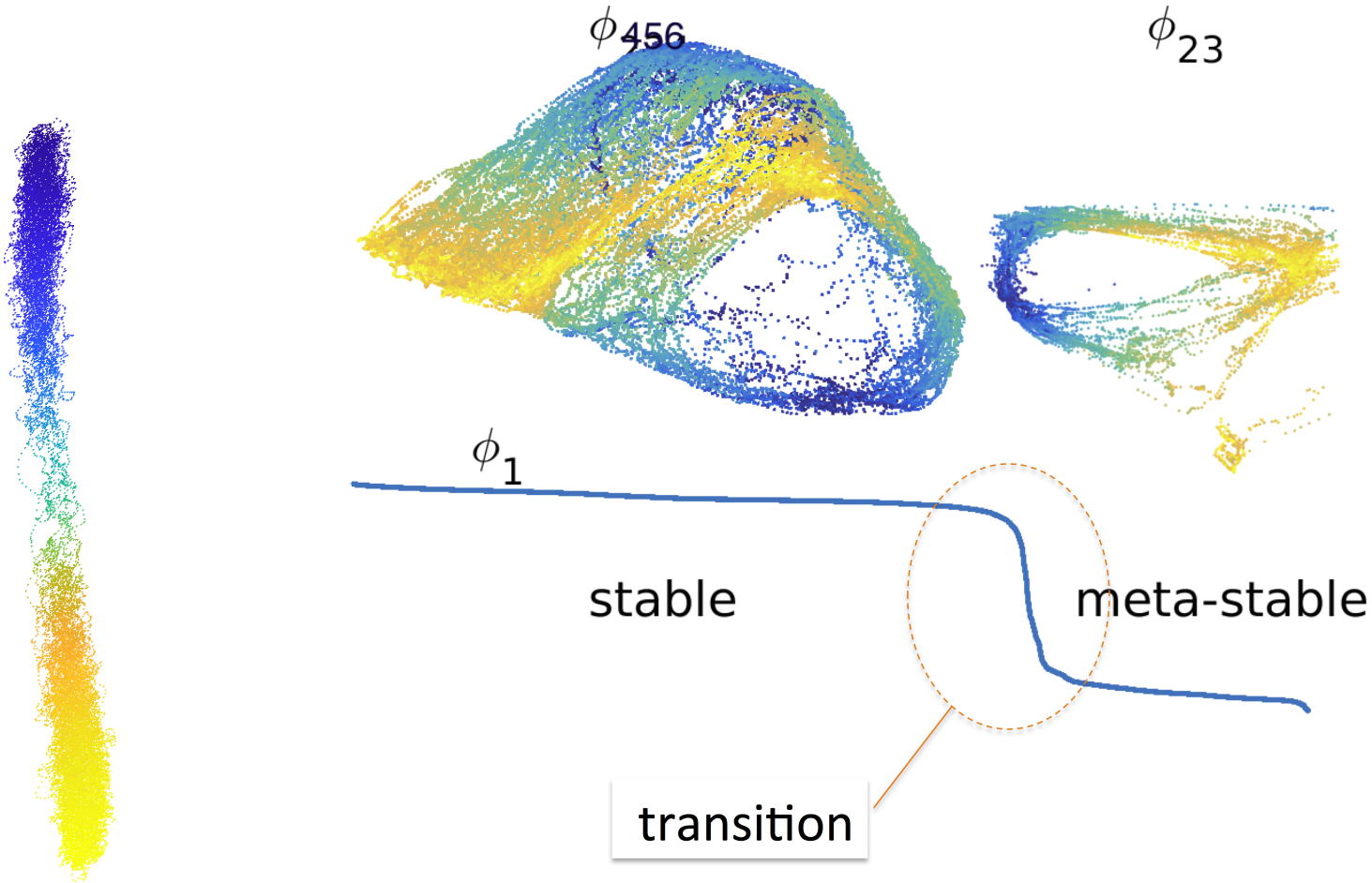


K-nearest neighbor



no renormalization, renormalized

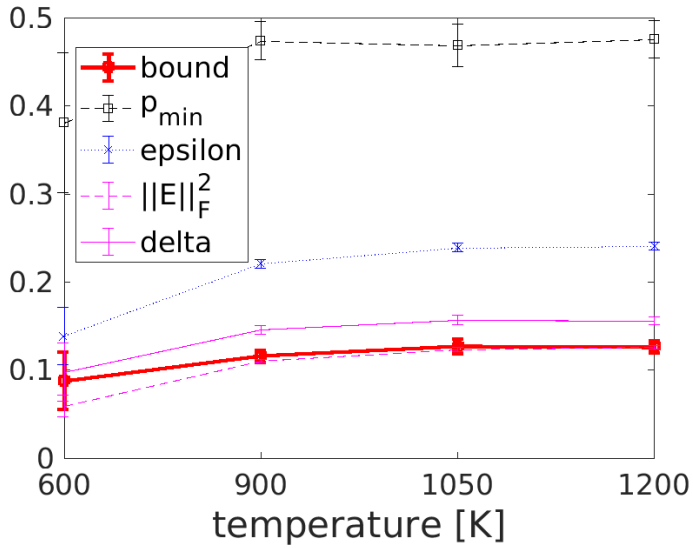
Clustering or/and Embedding



- In general, K eigenvalues ≈ 0 indicate K meta-stable states (K not too large)
- Simple normalization promotes clusters

Clustering with guarantees

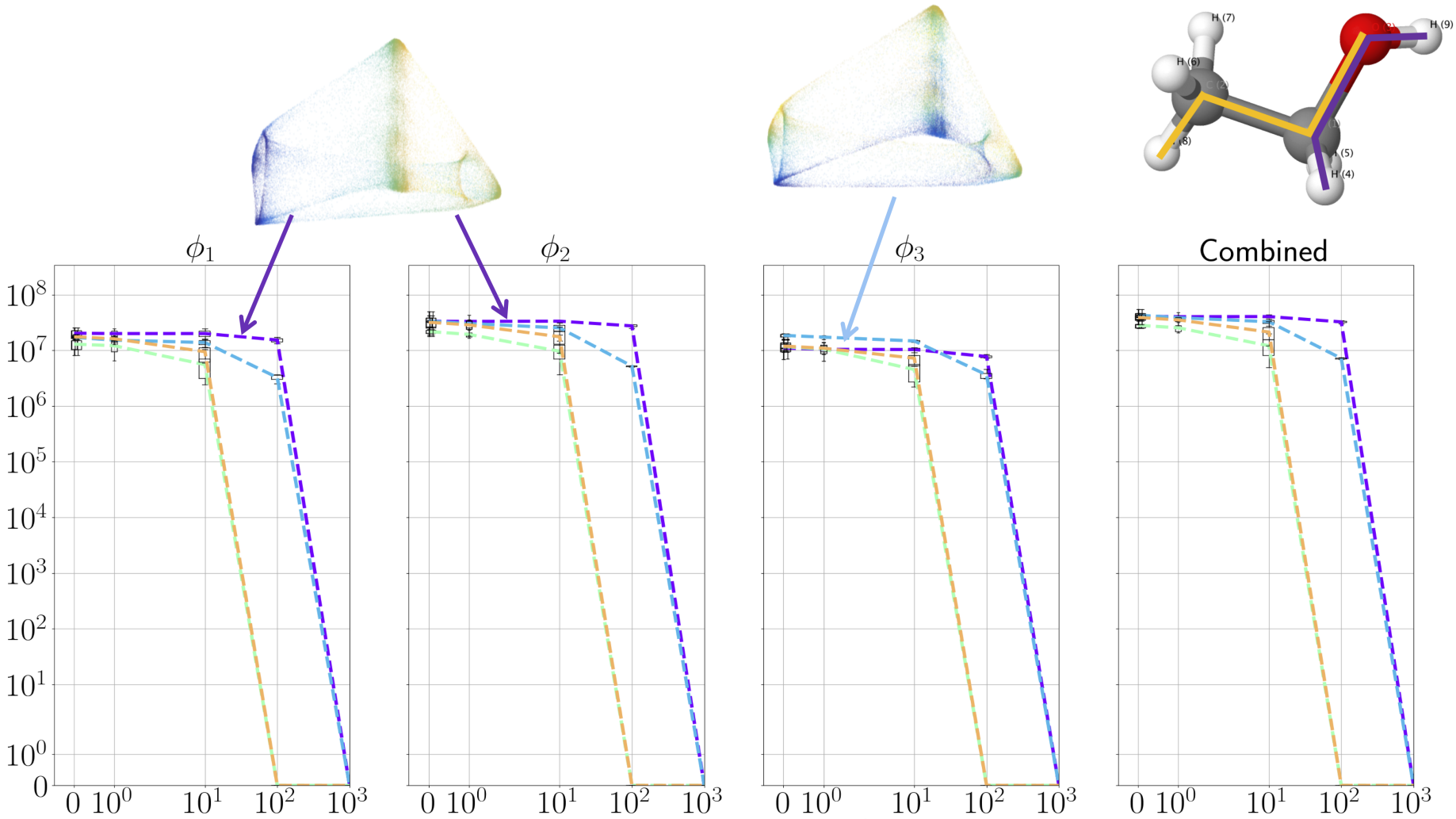
Molecular dynamics simulation of $CH_3Cl + Cl^- \leftrightarrow CH_3Cl + Cl^-$



Data by Jim Pfaendtner and Chris Fu

[M NeurIPS18] "How to tell when a clustering is (approximately) correct using convex relaxations"

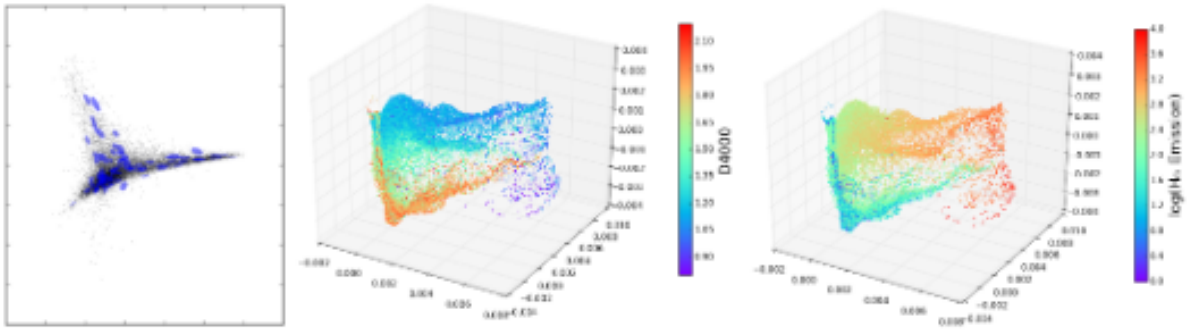
Manifold coordinates with physical meaning



Manifold Learning with millions of points

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

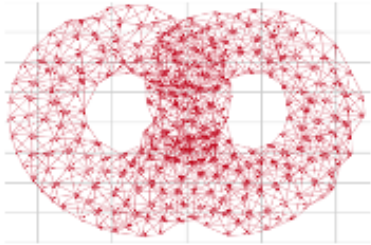
Examples

- Tutorial Notebook

Learning with flows and vector fields

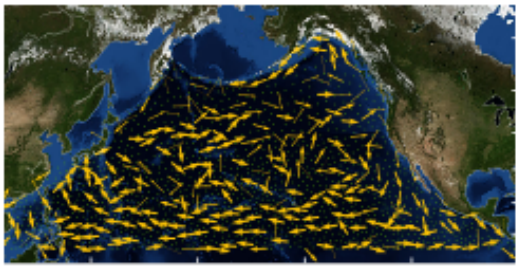
Directed graph embedding
Manifold + vector field [NIPS 2011]

1-Laplacian estimation
[Arxiv:2103.07626]

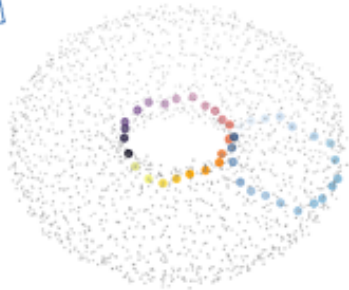


Helmholtz-Hodge
decomposition

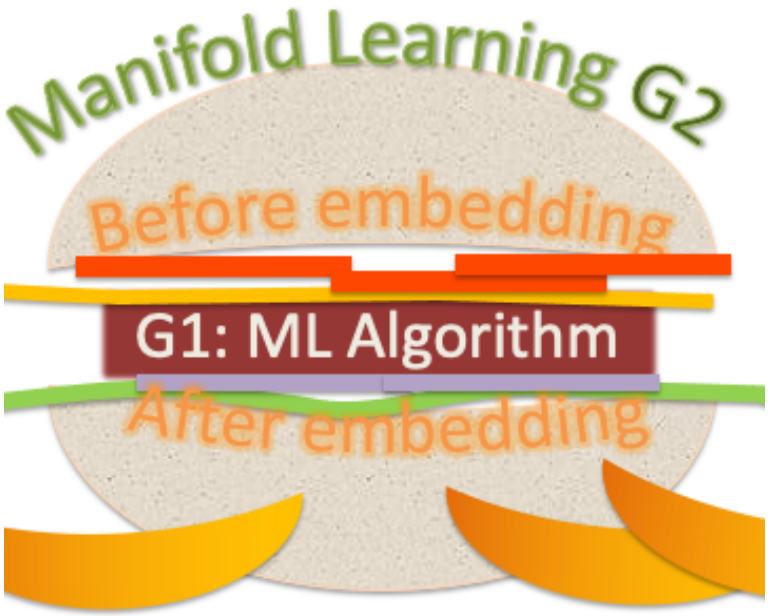
Smoothed vector fields



Independent loops
[Arxiv:2107.10970]
[NeurIPS 2021]



Manifold learning for materials

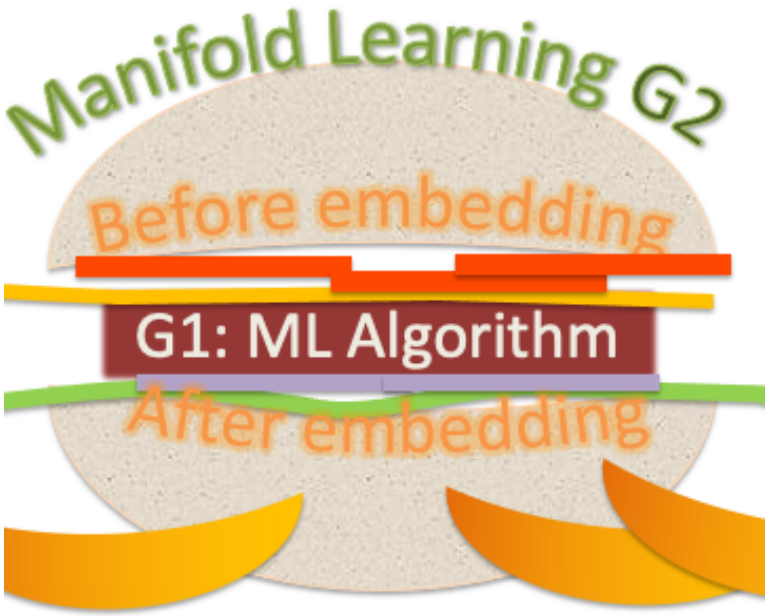


Manifold learning should be like PCA

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

  should not introduce artefacts

Manifold learning for materials



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

- estimates richer geometric/topological information
- adapts to data shape and dimension
- borders, stratification
- clusters
- Morse complex
- meaning of coordinates/continuous parametrization

Manifold Learning for materials and molecules

- Off-line
  - to understand the large scale shape of data
  - estimate slow manifold, interpret it
- On-line
  - Collective coordinates to enhance sampling
  - Estimate entire manifold or a patch

Manifold Learning for materials and molecules

- Off-line
  - to understand the large scale shape of data
  - estimate slow manifold, interpret it
- On-line
  - Collective coordinates to enhance sampling
  - Estimate entire manifold or a patch
- Open
  - What is "best" distance / kernel ?
  - How to know when two kernels are equivalent?
  - Symmetry and topology + Laplacian eigenfunctions
  - Use E, forces, other physical information to constrain manifold
  - End-to-end segmentation (meta-stable basins, transitions)
  - Collapsed "embedding" for visualization? (à la t-SNE)
  - Combine data-driven and a-priori collective coordinates
  - .....
  - your problem here

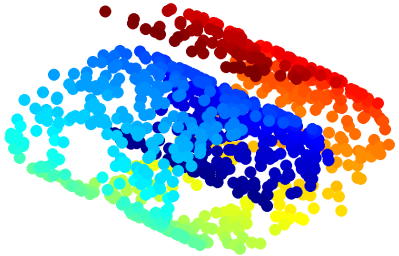
Hanyu Zhang, Samson Koelle, Yu-Chia Chen, Weicheng Wu
Ioannis Kevrekidis (JHU)

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

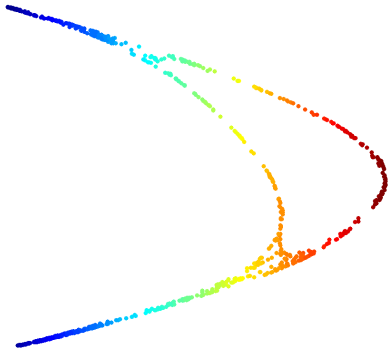
Thank you

Embedding in 2 dimensions by different manifold learning algorithms

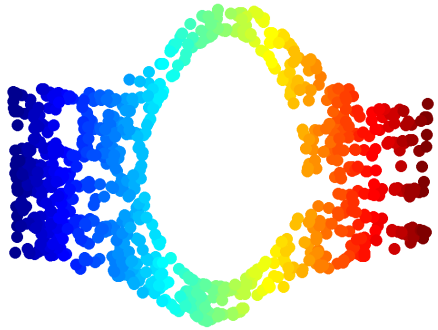
Original data
(Swiss Roll with hole)



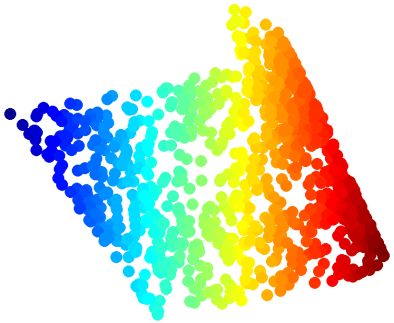
Laplacian Eigenmaps (LE)



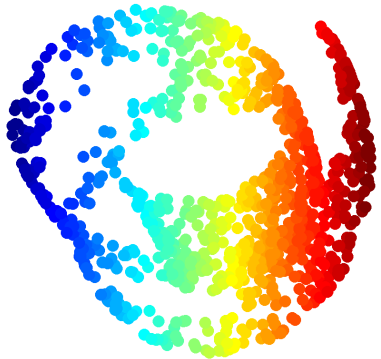
Isomap



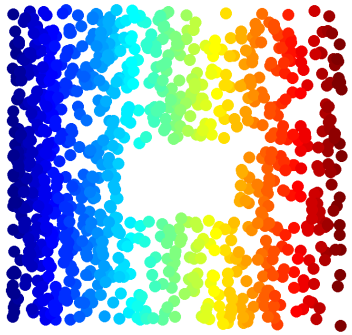
Hessian Eigenmaps (HE)



Local Linear Embedding
(LLE)



Local Tangent Space
Alignment (LTSA)

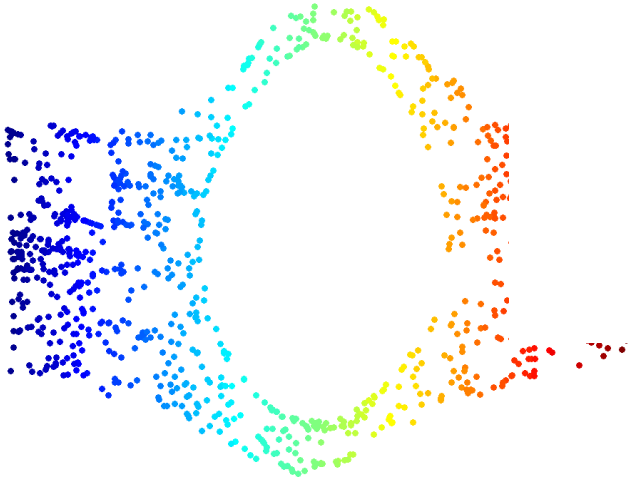


Preserving topology vs. preserving (intrinsic) geometry

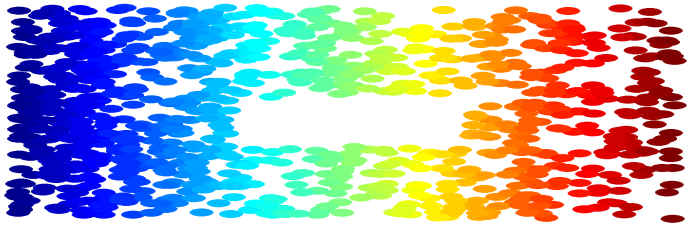
- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$

- Mapping $\mathcal{M} \longrightarrow \phi(\mathcal{M})$ is diffeomorphism
  preserves topology
  often satisfied by embedding algorithms
- Mapping ϕ is isometry
  - preserves distances along curves in $\mathcal{M}$, angles, volumes
    For most algorithms, in most cases, ϕ is not isometry

Preserves topology



Preserves topology + intrinsic geometry



Previous known results in isometric recovery

Positive results

- Nash's Theorem: Isometric embedding is possible.
- Diffusion Maps embedding is isometric in the limit [Berard,Besson,Gallot 94]
- algorithm based on Nash's theorem (isometric embedding for very low d) [Verma 11]
- Isomap [Tennenbaum,]recovers flat manifolds isometrically
- Consistency results for Laplacian and eigenvectors
  - [Hein & al 07,Coifman & Lafon 06, Singer 06, Ting & al 10, Gine & Koltchinskii 06]
  - imply isometric recovery for LE, DM in special situations

Negative results

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

Our approach: Metric Manifold Learning

[Perrault-Joncas,M 10]

Given

- mapping ϕ that preserves topology

  true in many cases

Objective

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

g is the Riemannian metric.

Dominique Perrault-Joncas

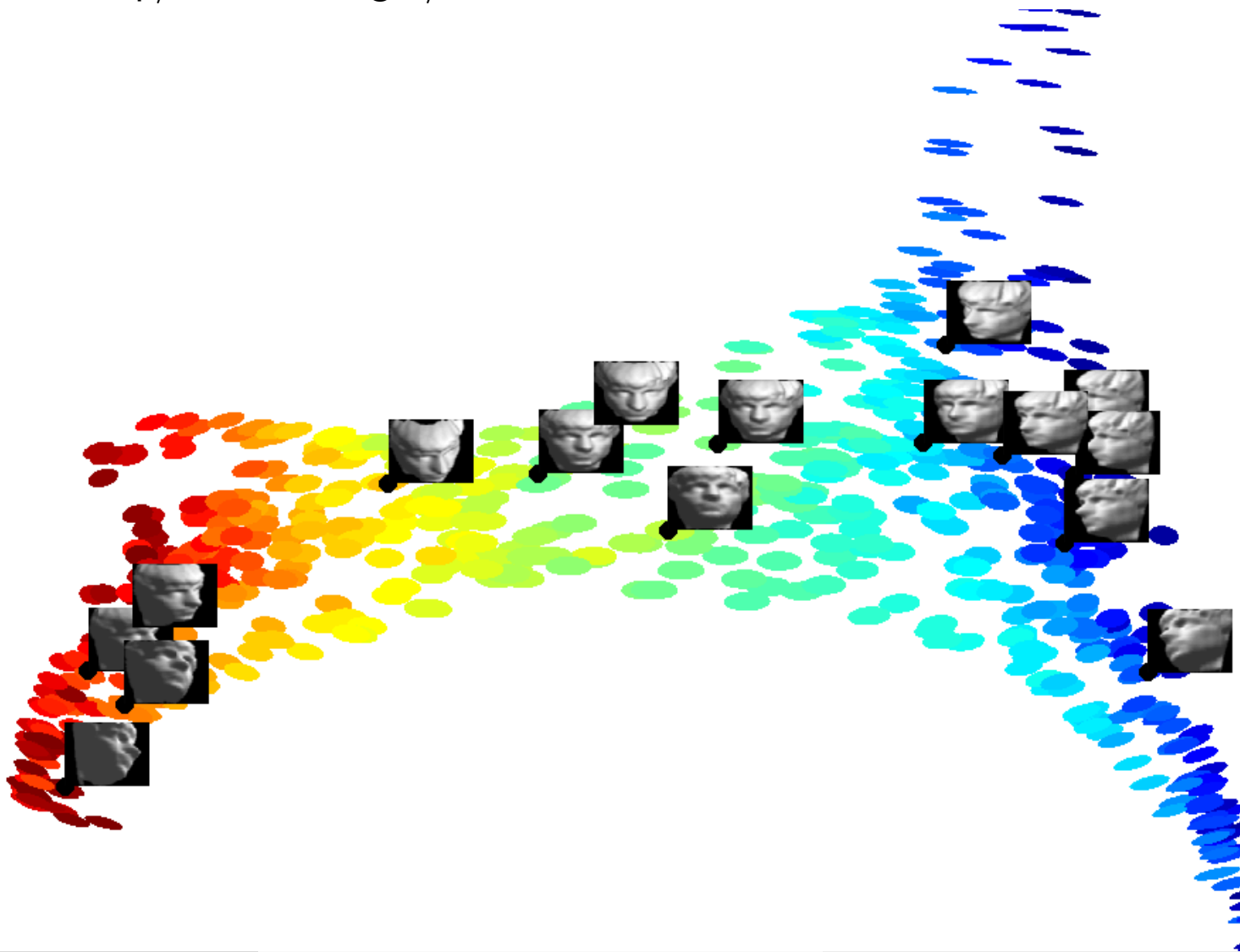
Problem formulation

- Given:
  - data set $\mathcal{D} = \{p_1, \dots p_n\}$ sampled from Riemannian manifold $(\mathcal{M}, g_0)$, $\mathcal{M} \subset \mathbb{R}^D$
  - embedding $\{ x_i = \phi(p_i),\ p_i \in \mathcal{D} \}$
    by e.g DiffusionMap, Isomap, LTSA, ...
- 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

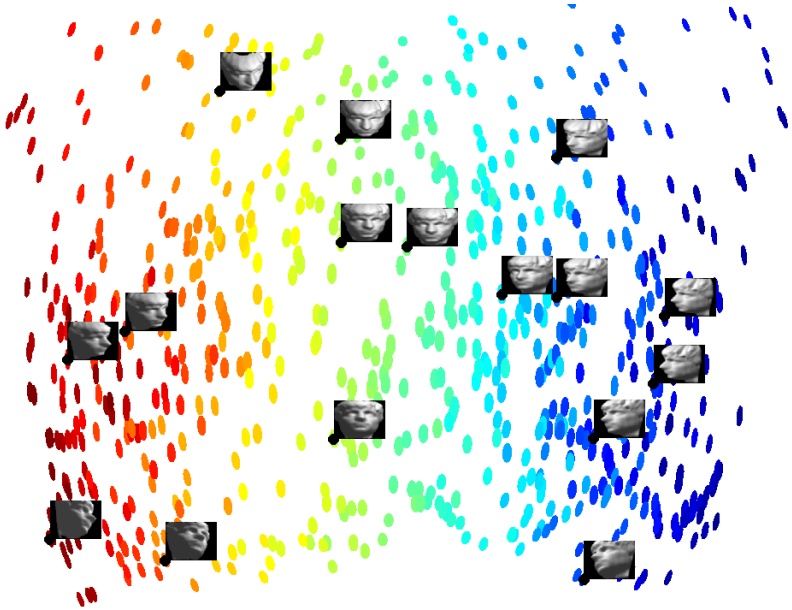
G for Sculpture Faces

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

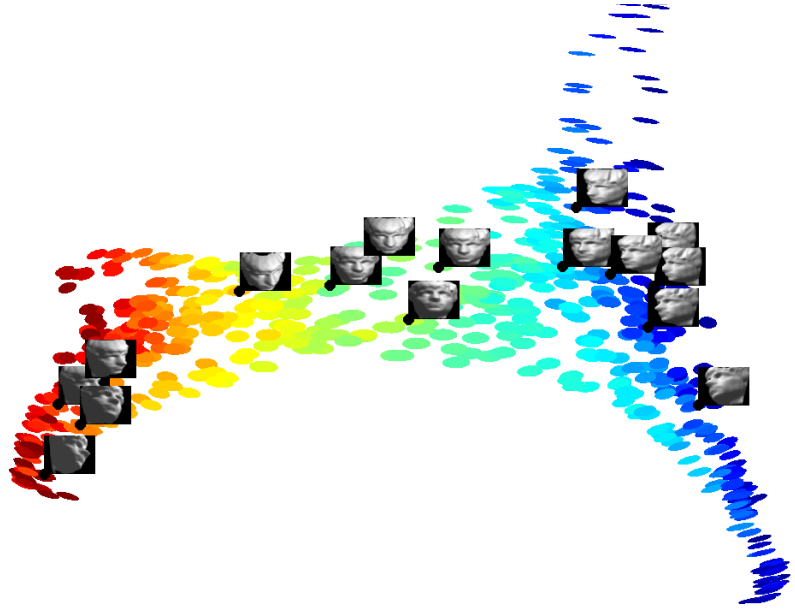


G for Sculpture Faces

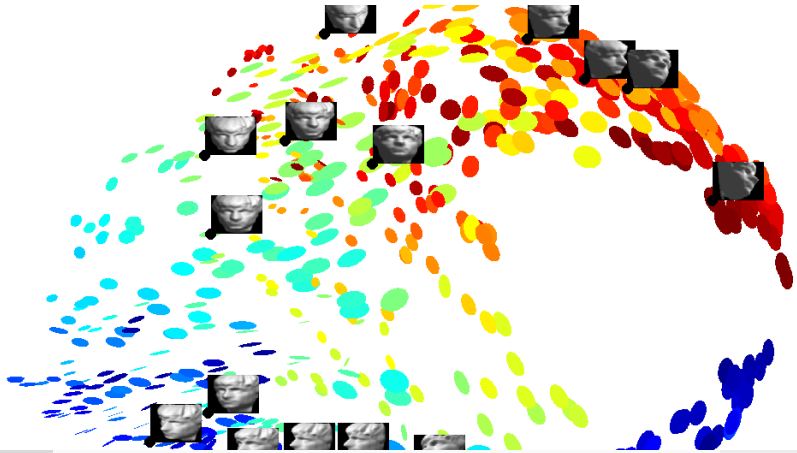
[Tenenbaum et al 2006]



Isomap



LTSA



Relation between g and Δ

- Δ = Laplace-Beltrami operator on $\mathcal{M}$
- G Riemannian metric (in coordinates)
- $H = G^{-1}$ matrix inverse

(Differential geometric fact)

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

Estimation of G^{-1}

Let Δ be the Laplace-Beltrami operator on $\mathcal{M}$, $H = G^{-1}$, and $k, l = 1, 2, \ldots d$.

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

Estimation of G^{-1}

Let Δ be the Laplace-Beltrami operator on $\mathcal{M}$, $H = G^{-1}$, and $k, l = 1, 2, \ldots d$.

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

Intuition:

- Δ applied to test functions $f = \phi_k^{\text{centered}} \phi_l^{\text{centered}}$
- 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]

Metric Manifold Learning algorithm

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