Dimension Reduction from the User's Perspective

Understanding the configuration spaces of molecules with Manifold Learning

Marina Meilă

mmp@stat.washington.edu

Cecilia Clementi, Stefan Chmiela, Pilar Cossio, Roberto Covino, Gabor Csany, Lars Dingeldein, Luke Evans, Chris Fu, Hugh Hillhouse, Oles Isayev, Risi Kondor, John Maddocks, Cosmin Marinica, Reinhard Maurer, Klaus-Robert Muller, Frank Noe, Feliks Nuske, Jim Pfaendtner, Christian Ratsch, Dima Shlyakhtenko, Christof Schutte, Gabriel Stoltz, Tom Swinburne, Alexandre Tkatchenko, A. Vasquez-Mayagoitia

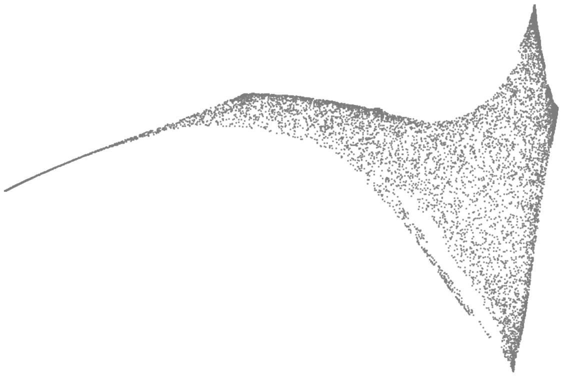
Dimensionality reduction techniques for molecular dynamics

ICMS Bayes Centre, Edinburgh

June 2, 2025

Can you recognize this shape?

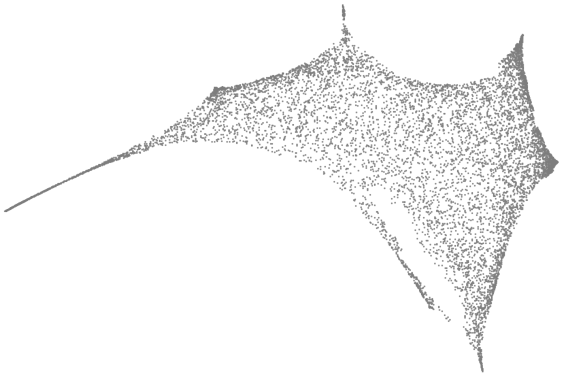
Embedding with LLE



Embeddings by Shuzhen Zhang https://github.com/mmp2/manifold-learning-examples

Can you recognize this shape?

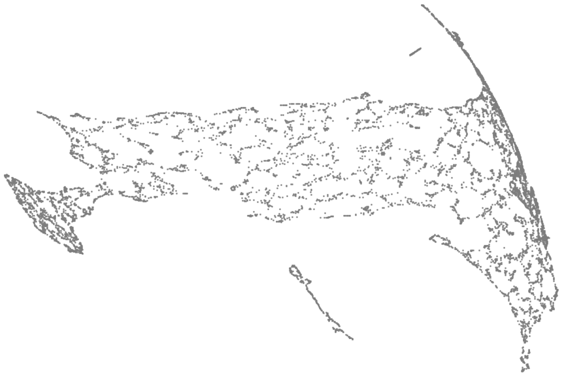
Embedding with Laplacian Eigenmaps



Embeddings by Shuzhen Zhang https://github.com/mmp2/manifold-learning-examples

Can you recognize this shape?

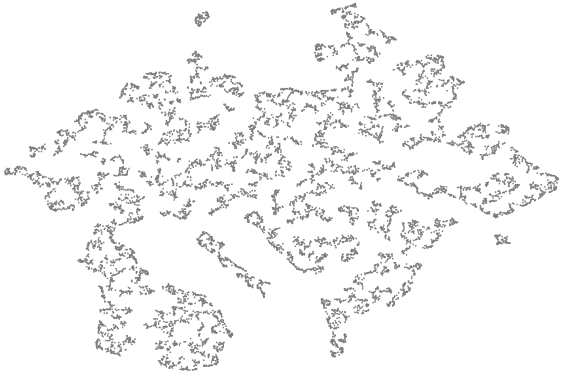
Embedding with UMAP



Embeddings by Shuzhen Zhang https://github.com/mmp2/manifold-learning-examples

Can you recognize this shape?

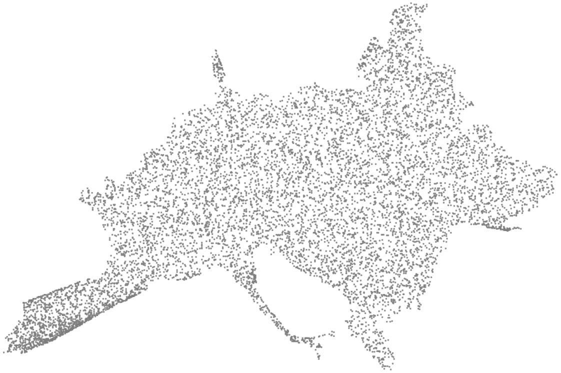
Embedding with t-SNE



Embeddings by Shuzhen Zhang https://github.com/mmp2/manifold-learning-examples

Can you recognize this shape?

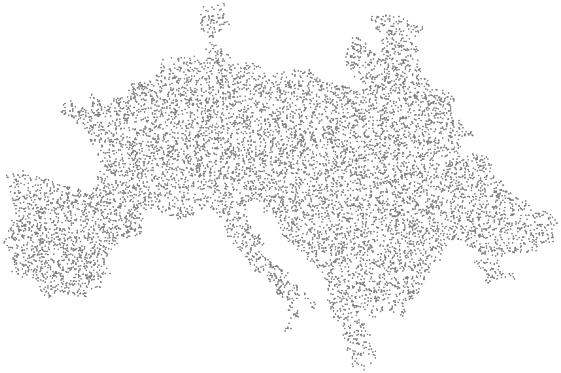
Embedding with Isomap



Embeddings by Shuzhen Zhang https://github.com/mmp2/manifold-learning-examples

Can you recognize this shape?

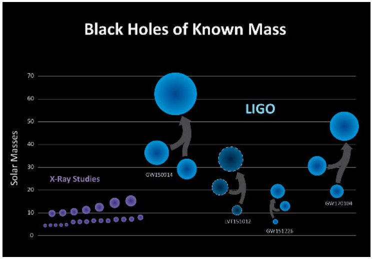
Embedding with LTSA



Embeddings by Shuzhen Zhang https://github.com/mmp2/manifold-learning-examples

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

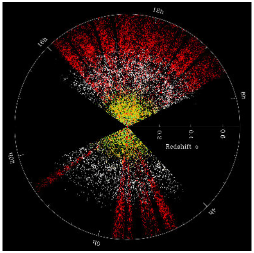
- Supervised, Reinforcement Learning
  - clear error measure
  - Cross-validation, bootstrap ✓
- Unsupervised learning:
  - =finding **structure** of data: clustering, dimension reduction, causal, . . .
  - formulating "error measure" is part of the problem
  - Cross-validation, bootstrap **X**
- Big 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



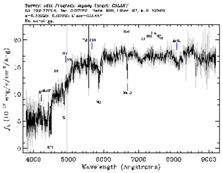
Outline

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

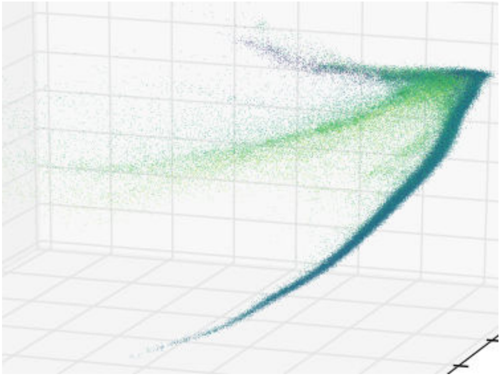
www.sdss.org



www.sdss.org

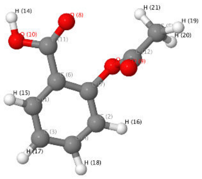


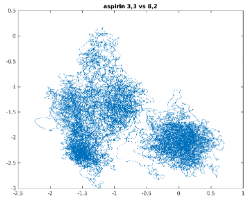
- Preprocessed by Jacob VanderPlas and Grace Telford
- $n = 675,000$ spectra $\times\ D = 3750$ dimensions


embedding by James McQueen

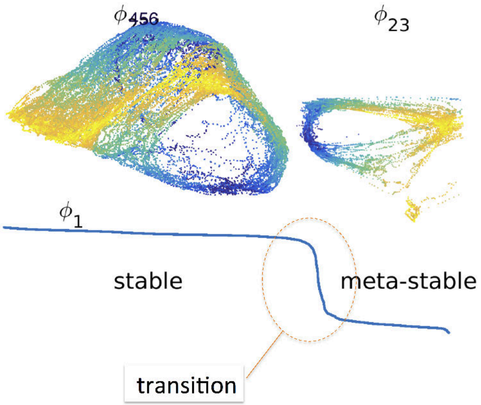
Molecular configurations

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

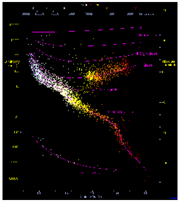
- big, high-dimensional data
- data, physics supports manifold models
- understanding & prediction equally important

Challenges for ML algorithms

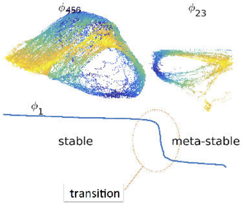
- scalable megaman ML package [McQueen et al JMLR 2015]
- find “something new, trustworthy, reproducible, interpretable”
- remove algorithmic artefacts
- data-driven parameter selection (replace grad student)
- validation on mathematical/statistical grounds as much as possible (replace experimental validation)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

When to do (non-linear) dimension reduction

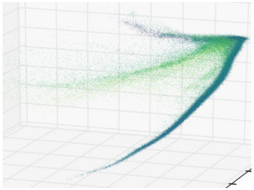
HR diagram



aspirin MD simulation



SDSS galaxy spectra



- high-dimensional
- can be described by a small number d of continuous parameters
- Usually, large sample size n

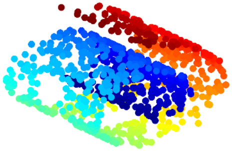
Why?

- 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

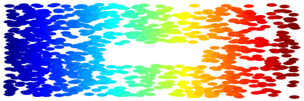
Outline

A toy example (the “Swiss Roll” with a hole)

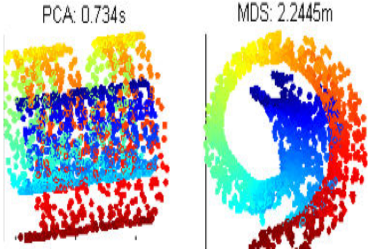
Input
points in $D \geq 3$ dimensions



Desired output
same points reparametrized in 2D

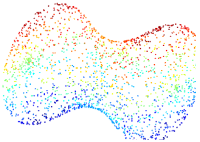


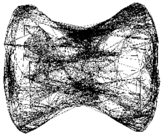
Linear dimension reduction fails

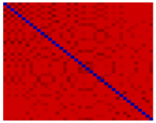


Neighborhood graphs

- 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 $(\phi_{i1}, \ldots \phi_{im})$

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 $d_j = \sum_{i=1}^n S_{ij}, \quad \tilde{L}_{ij} = S_{ij}/d_j$
3. Normalize rows $d_i' = \sum_{j=1}^n \tilde{L}_{ij}, \quad P_{ij} = \tilde{L}_{ij}/d_i'$
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 2006]

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)

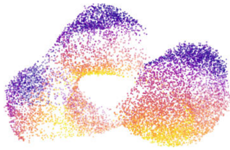
```
Input  similarity matrix $S$, embedding dimension $s$
Init   choose embedding points $y_{1:n} \in \mathbb{R}^s$ at random
  1  $S_{ii} \leftarrow 0$, normalize rows $d_i = \sum_j S_{ij}$, $P_{ij} = S_{ij}/d_i$
  2  symmetrize $P = \frac{1}{2n}(P + P^T)$ $P$ is distribution over pairs of neighbors $(i, j)$
  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
```

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

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

UMAP Algorithm

```
**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. Similarity matrix $S = W + W^T - W.*W^T$
4. Initialize embedding $\phi$ by LaplacianEigenmaps.
5. Optimize embedding.
   Iteratively for $n_{iter}$ steps
      1. Sample an edge $ij$ with probability $\propto \exp -d_{ij}$
      2. Move $\phi_i$ towards $\phi_j$
      3. Sample a random $j'$ uniformly
      4. Move $\phi_i$ away from $\phi_{j'}$
**Output** $\phi$
```



- t-SNE with appropriate choice of parameters can emulate UMAP [Böhm et al., 2022]

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

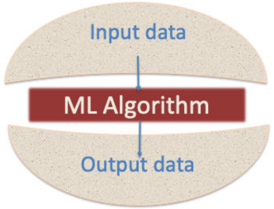
+ no collapsed embeddings

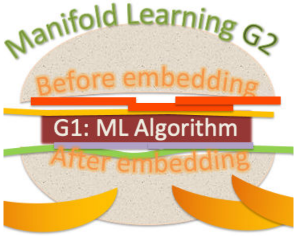
- $s = d$ intrinsic dimension

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

Outline

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]

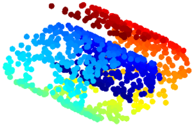
ML Algorithm: DiffMaps, LTSA, **t-SNE**

- 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] last 30min

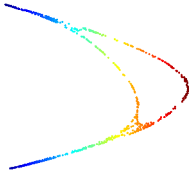
. . .

Distortions, failures, irreproducibility

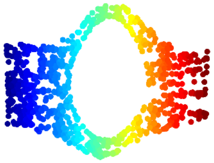
Original data
(Swiss Roll with hole)



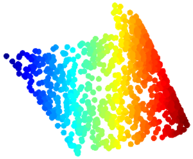
DiffMaps



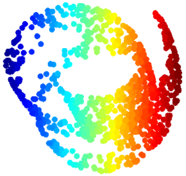
Isomap



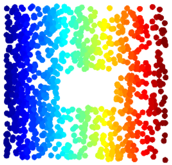
Hessian Eigenmaps (HE)



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

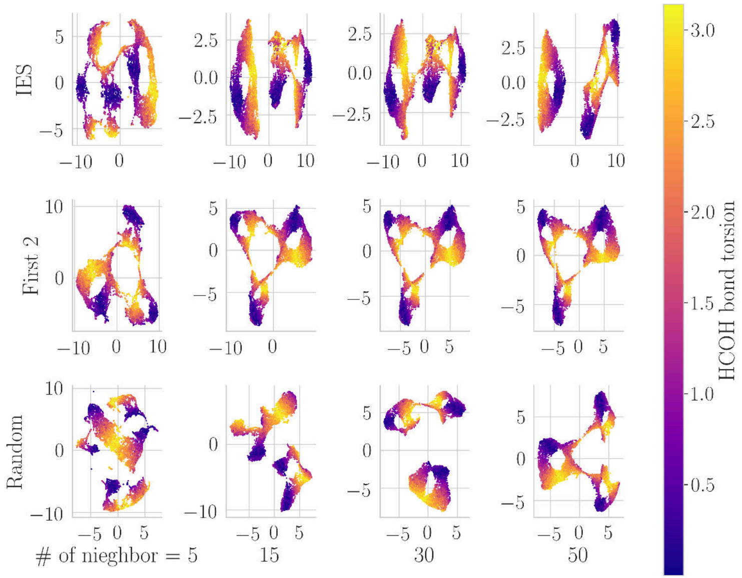
Distortions vs. Failures

- 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)

Most common modes of failure

1. distance matrix A does not capture topology (artificial "holes" or "bridges")
- usually because neighborhood too small or too large
2. attraction/repulsion "imbalances"
3. for e-vector based algorithms: choice of e-vectors
- or too few e-vectors (s too small)

UMAP for ethanol



Initialization by [Chen,M NeurIPS19]

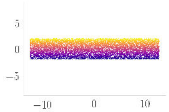
Laplacian Eigenmaps (default)

Random

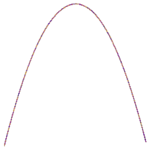
Artefacts

- 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
  1. algorithms that claim to choose ϵ automatically
  2. confirming the embedding is "correct" by visualization
     - tends to over-smooth, i.e. ϵ over-estimated
  3. K-nn (default in `sk-learn`!) instead of radius-neighbors: tends to create clusters
  4. large variations in density: stretching the low densities, contracting the high
     - subsample data to make it more uniform
  5. "horseshoes" ($\Rightarrow \phi$ almost singular):
     - select the e-vectors (coming next)
- Popular heuristics: LLE, t-SNE, UMAP, neural networks more prone to artefacts

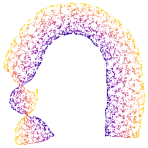
Horseshoes and the Repeated Eigenvector Problem. "Beware of the cosine"



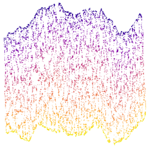
DiffMaps ϕ_1, ϕ_2



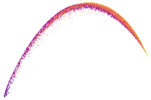
UMAP ϕ_1, ϕ_2



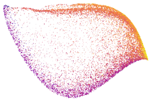
[Chen, M 2019] + UMAP



SDSS Galaxy Spectra
DiffMaps ϕ_1, ϕ_2



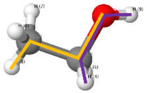
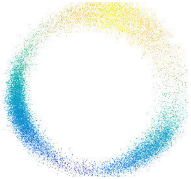
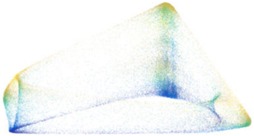
DiffMaps ϕ_1, ϕ_3



Outline

Metric Manifold Learning

- Most embeddings are distorted

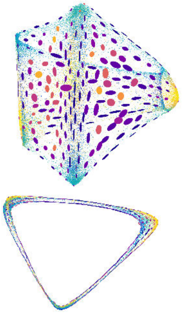
- Distortion depends on algorithm, parameters, data density, . . .
- How to fix?
- Isometric embedding eliminates distortion (hard)
- Metric learning estimate distortions (easy) and corrects geometric calculations (easy)

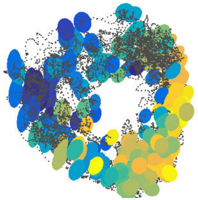
Estimating local distortion. The (push-forward) Riemannian metric

Idea at each point p

- along with embedding coordinates $(\phi_1(p), \dots \phi_s(p))$
- 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 $\phi(p)$

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 $d = 1.57$

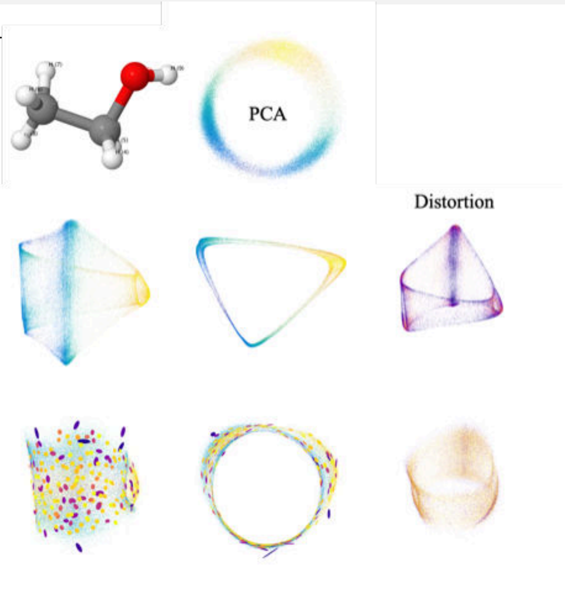
Embedding	Line segment	Shortest Path $d_{\mathcal{G}}$	Metric $\hat{d}$	Rel. err
Orig. data	1.41	1.57	1.62	3.0%
Isomap $s = 2$	1.66	1.75	1.63	3.7%
LTSA $s = 2$	0.07	0.08	1.65	4.8%
LE $s = 2$	0.08	0.08	1.62	3.1%

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.

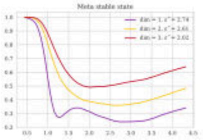


Manifold learning: beyond the embedding algorithm

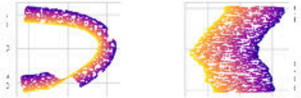
Correct algorithm distortion

Estimate Riemannian metric

Optimize neighborhood size [NIPS 2016]

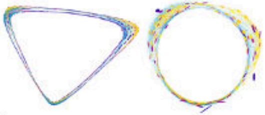


Choose independent e-vectors/Remove “horseshoes” [NeurIPS 2019]

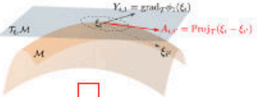


Distances, angles, areas preserved

Riemannian relaxation [NIPS 2015]

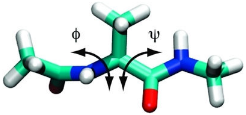


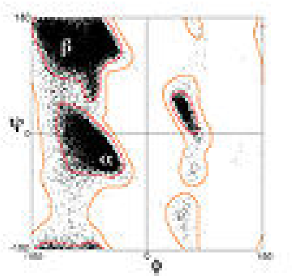
Vector fields preserved


$Y_{i,1} = \text{grad}_T \phi_1(\xi_i)$
$\mathcal{T}_{\xi_i}\mathcal{M}$
ξ_i
$A_{i,i'} = \text{Proj}_T(\xi_i - \xi_{i'})$
$\mathcal{M}$
$\xi_{i'}$

Outline

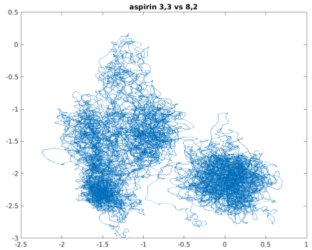
Data: from MD simulations

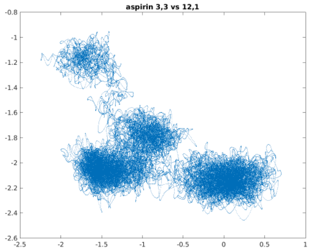




MD Data from [Chmiela et al. 2017], embeddings by Yu-Chia Chen

- 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!)

- 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

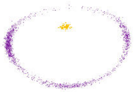
– mapping may be discontinous

– symmetry invariant kernels change topology

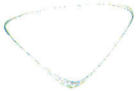
- Unsolved questions: preserving topology (or not?), dependence on distance (or kernel), symmetries

Procrustes

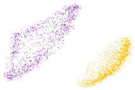
ethanol, PCA



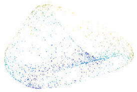
ethanol, DiffMaps



malonaldehyde, PCA

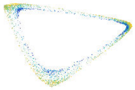


malonaldehyde, DiffMaps

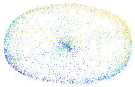


BondLen

ethanol, DiffMaps

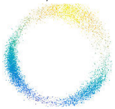


malonaldehyde, DiffMaps

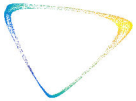


Angles

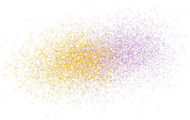
ethanol, PCA



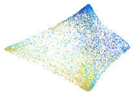
ethanol, DiffMaps



malonaldehyde, PCA



malonaldehyde, DiffMaps



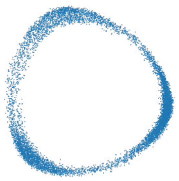
What graph? Radius-neighbors vs. k nearest-neighbors

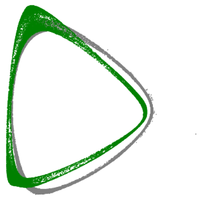
- k-nearest neighbors graph: each node has degree k
- radius neighbors graph: p, p' neighbors iff $||p - p'|| \leq r$

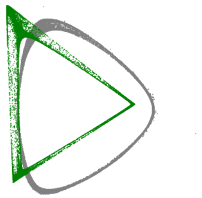
- Does it matter?

- 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 $\frac{1}{p^{2/d}}\Delta + (1 - 2/d)\nabla(\log p) \cdot \nabla$ (bias due to non-uniform sampling)
  - (normalized Laplacian converges to $\Delta + 2\nabla(\log p) \cdot \nabla$)
- Renormalization of Laplacian – counters the variable density effects

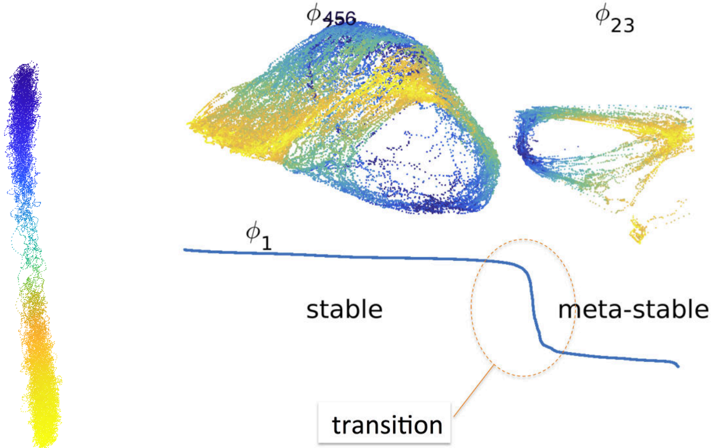
configurations of ethanol $d = 2$


original


K-nearest neighbor


no renormalization, renormalized

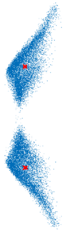
Clustering or/and Embedding

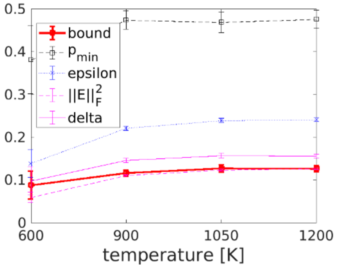


- In general, K eigenvalues ≈ 0 indicate K meta-stable states (K not too large)
- Simple normalization promotes clusters

Clustering with guarantees

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



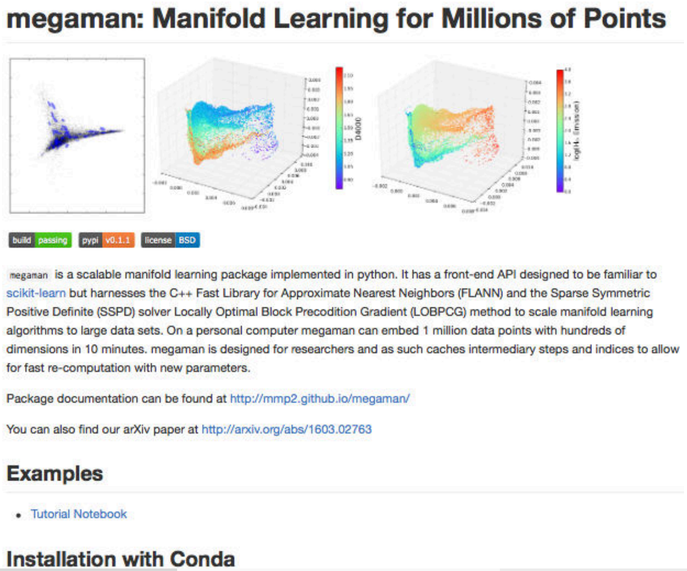


Data by Jim Pfaendtner and Chris Fu

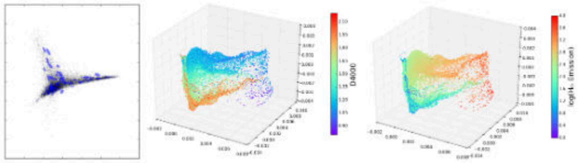
[M NeurIPS18] "How to tell when a clustering is (approximately) correct using convex relaxations"

Manifold 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

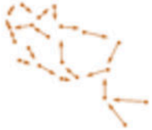
Examples

- Tutorial Notebook

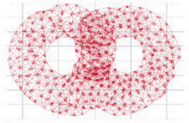
Installation with Conda

Learning with flows and vector fields

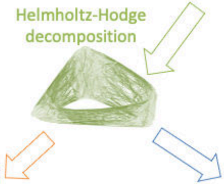
Directed graph embedding
Manifold + vector field [NIPS 2011]



1-Laplacian estimation
[Arxiv:2103.07626]

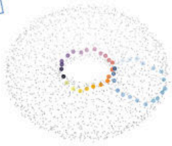


Helmholtz-Hodge
decomposition



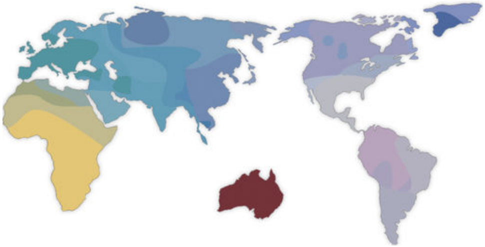
Smoothed vector fields

Independent loops
[Arxiv:2107.10970]
[NeurIPS 2021]

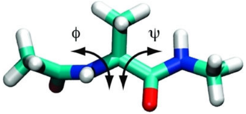


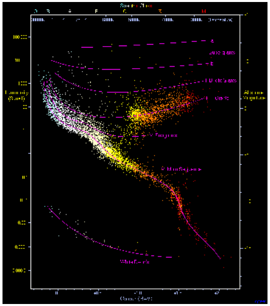
Outline

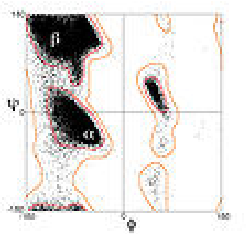
Coordinates with scientific meaning



[Cavalli-Sforza, Menozzi, Piazza, *"The history and geography of human genes"*, 1996]

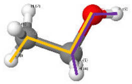




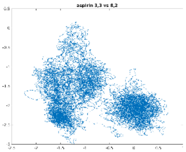


Motivation – understanding data from a Molecular Dynamics simulation

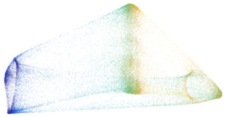
ethanol



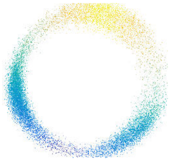
original data



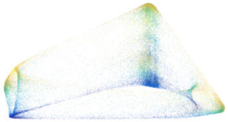
after ML
torsion 1



preprocessed



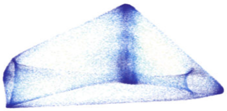
torsion 2



- 2 rotation angles (torsions) describe this manifold
- Can we discover these features automatically? Can we select these angles from a larger set of features with physical meaning?

Explaining a manifold with domain specific coordinates

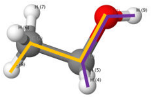
data driven coordinates (e.g. DiffMaps)



$\phi_{\xi_1}, \phi_{\xi_2}, \ldots \phi_{\xi_n}$

$+$

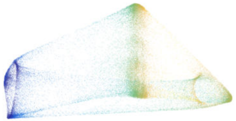
scientific language (torsions)



$\mathcal{F} = \{f_1, f_2 \ldots f_p\}$

$=$

interpretable coordinates



subset $f_{j_1}, \ldots f_{j_d} \subset \mathcal{F}$

with ManifoldLasso

Solution by sparse regression in function space

Wanted: Change of variable

$$\underset{\text{data driven coordinates}}{\phi} = \overset{\downarrow}{h} \circ \underset{\text{selected functions from } \mathcal{G} \text{ (collective coordinates)}}{f_S}$$

Challenges

- sparse, non-linear regression problem
- coordinates ϕ depend on data, algorithm parameters
- hence, h cannot take parametric form
- we cannot choose a basis for h
- cannot assume ϕ_k depends on single f_j
- cannot assume ϕ isometric

Idea: Chain Rule

$$D\phi = DhDf_S$$

- sparse linear regression problem
- $Y_i = \mathbf{X}_i\beta_i$ for every data i
  - $Y_i = \text{grad}\, \phi(\xi_i)$,
  - $\mathbf{X}_i = \text{grad}\, f_{1:p}(\xi)$
  - $\beta_{ij} = \frac{\partial h}{\partial f_j}(\xi_i)$
- Constraint: subset S is same for all i

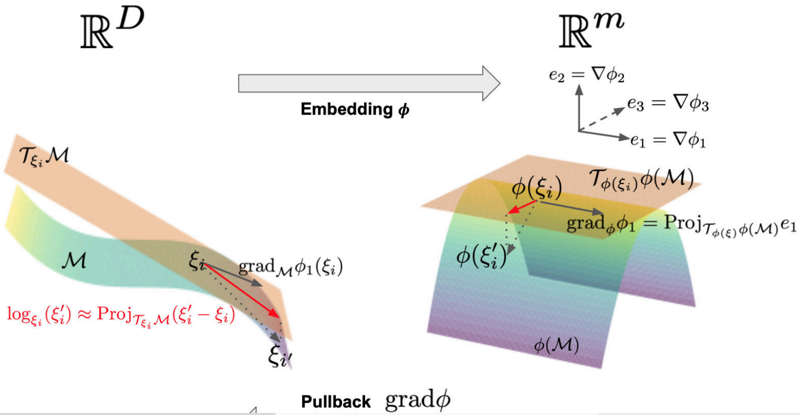
Solution by Group Lasso

- optimize

$$\min J_\lambda(\beta) = \frac{1}{2}\sum^{n} ||Y_i - \mathbf{X}_i\boldsymbol{\beta}_i||_2^2 + \lambda \sum ||\beta_j||, \quad (\textsc{ManifoldLasso})$$

Gradients in manifold setting

- gradients $\nabla \to$ manifold gradients grad in tangents subspace to $\mathcal{M}$
- grad f_j is in $\mathcal{T}_{\xi_i}\mathcal{M}$ (ambient space $\mathbb{R}^D$)
    - ∇f_j known analytically
- grad ϕ_k is in $\mathcal{T}_{\phi(\xi_i)}\phi(\mathcal{M})$ (embedding space $\mathbb{R}^m$)
    1. must estimate tangent subspace $\mathcal{T}_{\phi(\xi_i)}\phi(\mathcal{M})$
    2. must estimate grad $\phi_k(\phi(\xi_i))$ in tangent subspace $\mathcal{T}_{\phi(\xi_i)}\mathcal{M}$
    3. must pull-back grad $\phi_k(\phi(\xi_i))$ to $\mathcal{T}_{\xi_i}\mathcal{M}$



$\mathbb{R}^D$
$\mathbb{R}^m$
Embedding ϕ
$e_2 = \nabla\phi_2$
$e_3 = \nabla\phi_3$
$e_1 = \nabla\phi_1$
$\mathcal{T}_{\xi_i}\mathcal{M}$
$\mathcal{M}$
ξ_i
$\mathrm{grad}_{\mathcal{M}}\phi_1(\xi_i)$
$\log_{\xi_i}(\xi_i') \approx \mathrm{Proj}_{\mathcal{T}_{\xi_i}\mathcal{M}}(\xi_i' - \xi_i)$
$\xi_{i'}$
$\phi(\xi_i)$
$\mathcal{T}_{\phi(\xi_i)}\phi(\mathcal{M})$
$\mathrm{grad}_\phi\phi_1 = \mathrm{Proj}_{\mathcal{T}_{\phi(\xi)}\phi(\mathcal{M})}e_1$
$\phi(\xi_i')$
$\phi(\mathcal{M})$
Pullback $\mathrm{grad}\phi$

ManifoldLasso Algorithm

Given Data $\xi_{1:n}$, intrinsic dimension d, embedding $\phi(\xi_{1:n})$

dictionary $\mathcal{F} = \{f_{1:p}\}$

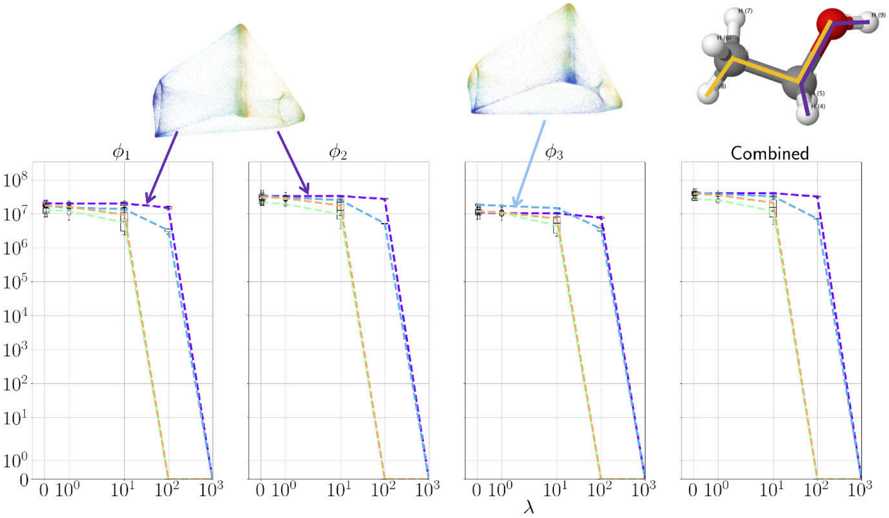
1. Estimate tangent subspace at ξ_i by (weighted) PCA
2. Project dictionary functions gradients ∇f_j on tangent subspace, obtain $\mathbf{X}_{1:n} \in \mathbb{R}^{d\times p}$
3. Estimate gradients of $\phi_{1:k}$, obtain $Y_{1:n} \in \mathbb{R}^{d\times m}$

   by pull-back from embedding space ϕ
4. Solve $\textsc{GroupLasso}(Y_{1:n}, \mathbf{X}_{1:n}, d)$, obtain support S

$$\min_\beta J_\lambda(\beta) \;=\; \frac{1}{2}\sum_{i=1}^{n} ||Y_i - \mathbf{X}_i\boldsymbol{\beta}_i||_2^2 + \lambda \sum_j ||\beta_j||, \quad (\textsc{ManifoldLasso})$$

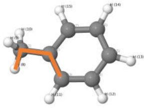
Output S

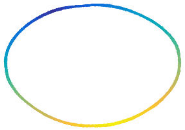
Ethanol MD simulation

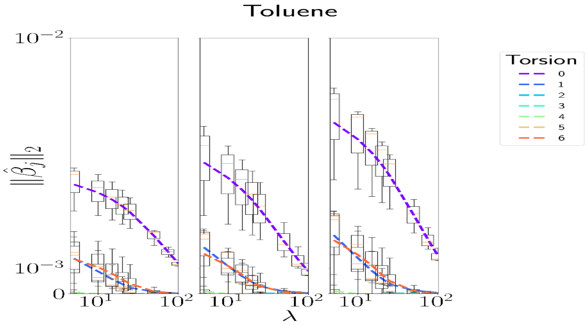


regularization paths $\beta_{1:p}$ vs λ

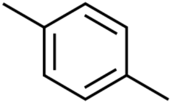
Toluene MD simulation

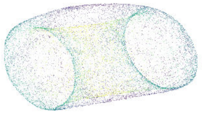


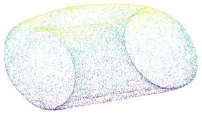


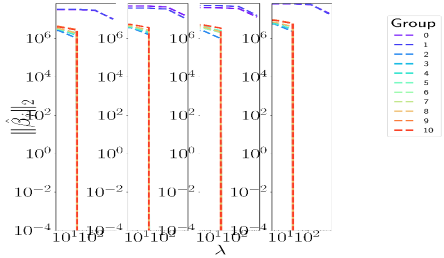


Para-xilene MD simulation









Theory

[Koelle et al., arXiv:1811.11891, JMLR 2022, AISTATS 2024]

- When is S unique? / When can $\mathcal{M}$ be uniquely parametrized by $\mathcal{F}$?
  Functional independence conditions on dictionary $\mathcal{F}$ and subset $f_{j_1,\ldots j_s}$
- Basic result

  $f_S = h \circ f_{S'}$ on U iff

$$\operatorname{rank}\begin{pmatrix} Df_S \\ Df_{S'} \end{pmatrix} = \operatorname{rank} Df_{S'} \quad \text{on } U$$

- When can Group Lasso recover S ?
  (Simple) Incoherence Conditions

$$\mu = \max_{i=1:n, j\in S, j'\notin S} \frac{|\mathbf{X}_{ji}^T \mathbf{X}_{j'i}|}{\|\mathbf{X}_{ji}\|\|\mathbf{X}_{j'i}\|} \quad \nu = \frac{1}{\min_{i=1:n} \|\mathbf{X}_{iS}^T \mathbf{X}_{iS}\|_2} \quad nd\sigma^2 = \sum_{i,k} \epsilon_{ik}^2$$

<u>Theorem</u> If, $\|\mathbf{X}_{1:p}\| = 1$, $\mu\nu\sqrt{d} + \frac{\sigma\sqrt{nd}}{\lambda} < 1$ then $\beta_j = 0$ for $j \notin S$.

Recovery for ManifoldLasso

Theorem 7 (Support recovery) *Assume that equation (30) holds, and that* $\sum_{i=1}^n ||x_{ij}||^2 = \gamma_j^2$ *for all* $j = 1 : p$. *Let* $\gamma_{\max} = \max_{j \notin S} \gamma_j$, $\kappa_S = \max_{i=1:n} \frac{\max_{j \in S} \|x_{ij}\|}{\min_{j \in S} \|x_{ij}\|}$. *Denote by* $\bar{\beta}$ *the solution of (31) for some* $\lambda > 0$. *If* $1 - (s-1)\mu > 0$ *and*

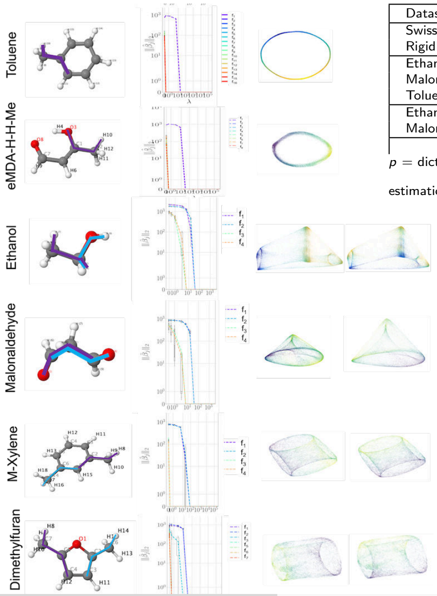
$$\gamma_{\max}\left(\frac{\mu}{1-(s-1)\mu}\frac{\kappa_S}{\min_{i=1}^n \min_{j' \in S} \|x_{ij'}\|} + \frac{\sigma\sqrt{d}}{\lambda\sqrt{n}}\right) \leq 1 \tag{37}$$

then $\bar{\beta}_{ij} = 0$ *for* $j \notin S$ *and all* $i = 1, \ldots n$.

Corollary 8 *Assume that equation* (31) *and condition (37) hold. Let* $\kappa = \frac{\mu}{1-(s-1)\mu}\frac{\kappa_S}{\min_{i=1}^n \min_{j' \in S} \|x_{ij'}\|}$ *and* $\gamma_S = \|\bar{\bar{X}}_S\|$. *Denote by* $\hat{\beta}$ *the solution to problem* (31) *for some* $\lambda > 0$. *If (1)* $\lambda = c\frac{\gamma_{\max}\sigma\sqrt{d}}{1-\kappa\gamma\max}$, $c > 1$, *and (2)* $||\beta_j^*|| > \sigma\sqrt{d}(\gamma_{\max} + \gamma_S) + \lambda(1+\sqrt{s})$ *for all* $j \in S$, *then the support* S *is recovered exactly and*

$$||\hat{\beta}_j - \beta_j^*|| < \sigma\sqrt{d}(\gamma_{\max} + \gamma_S) + \lambda(1+\sqrt{s}) = \sigma\sqrt{d}\gamma_{\max}\left[1 + \gamma_S/\gamma_{\max} + c\frac{1+\sqrt{s}}{1-\kappa\gamma_{\max}}\right] \quad \textit{for all } j \in S.$$

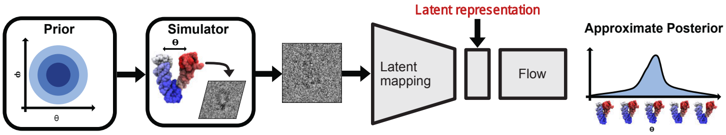
Experiments



Dataset	n	N_a	D	d	ϵ_N	m	n'	p
SwissRoll	10K	NA	51	2	.18	2	100	51
RigidEth	10K	9	50	2	3.5	3	100	12
Ethanol	50K	9	50	2	3.5	3	100	12
Malonald	50K	9	50	2	3.5	3	100	12
Toluene	50K	16	50	1	1.9	2	100	30
Ethanol	50K	9	50	2	3.5	3	100	756
Malonald	50K	9	50	2	3.5	3	100	756
	ϕ						MLasso	$\lvert\mathcal{G}\rvert$

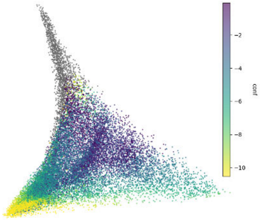
p = dictionary size, m = embedding dimension, n = sample size for manifold estimation, n' = sample size for ManifoldLasso

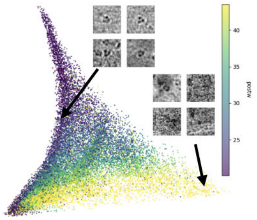
Understanding latent space representation of cryoEM images



- Estimating conformation of Hemagluttinin molecules from cryoEM images
- Neural network trained on simulated images [Dingeldein et. al. biorXiv:2024]
- Unsupervised study of hidden layer representation: **low dimensional!**

conformation θ SNR



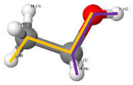


with Luke Evans, Vlad Murad, Lars Dingeldein, Pilar Cossio, Roberto Covino [NeurIPS 2024 MLSB Workshop, arXiv:2504.11249]

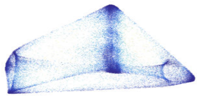
Summary of ManifoldLasso

- non-linear sparse regression in function spaces $\Rightarrow$ linear sparse regression (Group Lasso)
- ManifoldLasso= coordinate change from data driven coordinates $\phi_{1:m}$ to collective coordinates $\mathcal{F} = \{f_{1:p}\}$

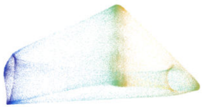
scientific language | data driven coordinates | interpretable coordinates



$+$



$=$

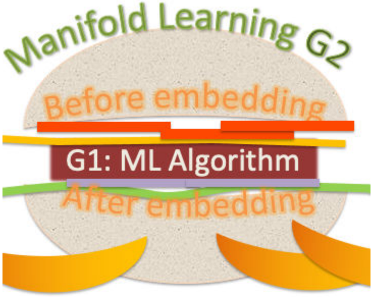


- explains large scale structure with domain-relevant functions
- transmissible knowledge, compare embeddings from different experiments
- non-linear, non-parametric, basis-free, not symbolic regression [Brunton et al. 2016, Rudy et al. 2019] [Udrescu, Tegmark 2020]
- No manifold necessary immediate extensions to Principal Components, autoencoders (low dimensional!), sparse functional regression

Applications

- set of f's that covary (e.g. small protein folding), level sets (in progress)
- simultaneous explanation of multiple systems
- dynamical systems (future)

Manifold learning for MD simulations



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

Embedding = Algorithm + user choices

- Similarity function (for MD)
- neighborhood scale (or k nearest neighbors)

Manifold Learning for MD simulations

- 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
  - ....
  - $\boxed{\text{your problem here}}$

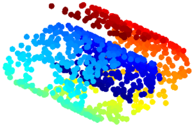
Hanyu Zhang, Samson Koelle, Vlad Murad, Yu-Chia Chen, Weicheng Wu
Ioannis Kevrekidis (JHU)

Alexandre Tkatchenko (Luxembourg), Stefan Chmiela (TU Berlin)
Pilar Cossio (Flatiron), Luke Evans (Flatiron)
Lars Dingeldein (Frankfurt), Roberto Covino (Frankfurt)

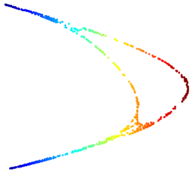
Thank you

Embedding in 2 dimensions by different manifold learning algorithms

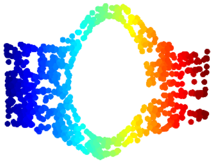
Original data
(Swiss Roll with hole)



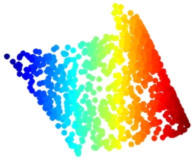
Laplacian Eigenmaps (LE)



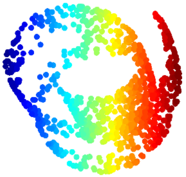
Isomap



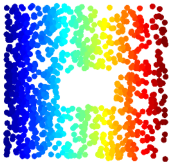
Hessian Eigenmaps (HE)



Local Linear Embedding
(LLE)



Local Tangent Space
Alignment (LTSA)

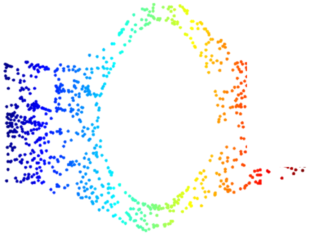


Preserving topology vs. preserving (intrinsic) geometry

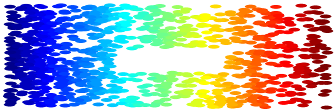
- Algorithm maps data $p \in \mathbb{R}^D \longrightarrow \phi(p) = x \in \mathbb{R}^m$

- 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.

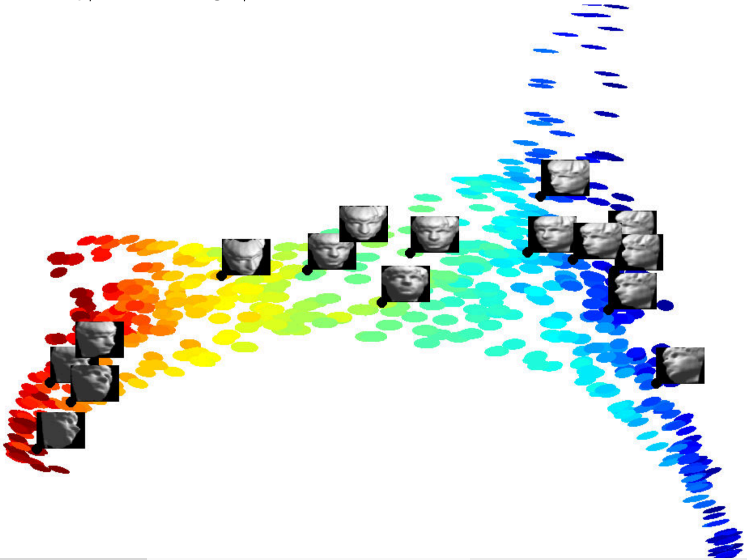
Dominique Perrault-Joncas

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

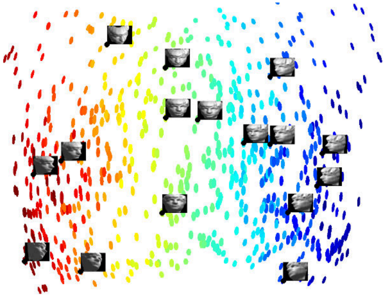
G for Sculpture Faces

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

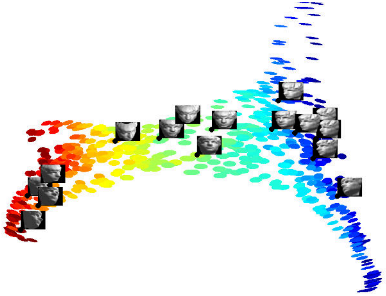


G for Sculpture Faces

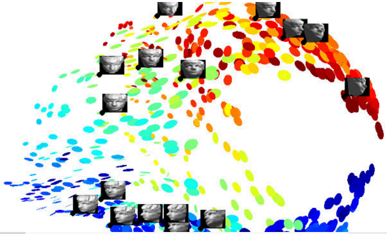
[Tenenbaum et al 2006]



Isomap



LTSA



Relation between g and Δ

- $\Delta =$ 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 $k, l = 1, 2, \ldots d$.

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

Intuition:

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