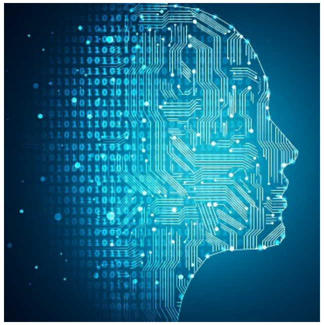
Machine Learning

Out of Sample Extension (OOSE)

Kevin Moon (kevin.moon@usu.edu)

STAT 6655



PCA and OOSE

- Suppose we have data $D = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \in \mathbb{R}^d$ with covariance matrix C
  - Assume the data are mean-centered
- Apply PCA to this data
  - Let $U_m = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_m]$ with $\boldsymbol{u}_i$ the ith eigenvector of C
  - PCA representation of $\boldsymbol{x}_j$ is $\boldsymbol{\theta}_j = U_m^T \boldsymbol{x}_j \in \mathbb{R}^m$
- How can we apply this embedding to a new point $\boldsymbol{x}$?
  - Project the point onto the space spanned by U_m
  - $\boldsymbol{\theta} = U_m^T \boldsymbol{x}$
  - Referred to as out of sample extension (OOSE)

Manifold Learning and OOSE

- Many manifold learning methods are based on an eigendecomposition of a kernel matrix
  - E.g. MDS (classical), Laplacian eigenmaps, diffusion maps, Isomap, spectral clustering
- These methods do not learn a general embedding function that can be applied to new points
- Nonlinear, so a linear projection is not appropriate
- Other approaches are required for out of sample extension (OOSE)

Nystrom Extension

- Two things needed: 1) eigendecomposition of a kernel matrix and 2) kernel similarities between training points and the new point
- The new representation consists of a linear combination of the kernel similarities
- Let’s go over this in detail…

Nystrom Extension: Framework

Consider methods that have the following steps given a dataset $D = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$

1. Construct a similarity matrix M with $M_{ij} = K_D(\boldsymbol{x}_i, \boldsymbol{x}_j)$
2. (Optional) Transform M to obtain a normalized matrix $\widetilde{M}$. Equivalent to generating $\widetilde{M}$ from $\widetilde{K}_D$
3. Compute the m largest positive eigenvalues λ_k and eigenvectors $\boldsymbol{v}_k$ of $\widetilde{M}$
4. Embed each example $\boldsymbol{x}_i$ as a vector $\boldsymbol{y}_i$ with y_{ik} as the ith element of $\boldsymbol{v}_k$
   - For MDS or Isomap, the embedding is $\boldsymbol{e}_i$ with $e_{ik} = \sqrt{\lambda_k} y_{ik}$

Framework: MDS (classical)

- Let $S_i = \sum_j M_{ij}$
- Classical MDS uses an eigendecomposition of the double-centered distance or affinity matrix
  - Double-centering converts distances to dot products

$$\widetilde{M}_{ij} = -\frac{1}{2}\left(M_{ij} - \frac{1}{n}S_i - \frac{1}{n}S_j + \frac{1}{n^2}\sum_k S_k\right)$$

- Embedding of $\boldsymbol{x}_i$ is $e_{ik} = \sqrt{\lambda_k} v_{ki}, k = 1, \ldots, m$

Framework: Spectral Clustering

- Let $S_i = \sum_j M_{ij}$
- Several normalization steps proposed
- One approach:

$$\widetilde{M}_{ij} = \frac{M_{ij}}{\sqrt{S_i S_j}}$$

- For m clusters, apply k-means to the first m eigenvectors of $\widetilde{M}_{ij}$
- Similar ideas for Laplacian eigenmaps

Framework: Isomap and Diffusion Maps

Isomap

- Compute pairwise geodesic distances along the k-NN graph to get M
- Obtain $\widetilde{M}$ via double centering as in MDS

Diffusion Maps

- $\widetilde{M} =$ the powered diffusion operator

From Eigenvectors to Eigenfunctions

- Heaviest computation in most of these methods is the eigendecomposition
- Nystrom method initially invented to speed up computation
  - Eigendecompose a subset of the data and then extend
  - Can use it for general OOSE
- Intuitively, we need to add a new column to $\widetilde{M}$ through a kernel function $\widetilde{K}_D$ that depends on the training data
- We'll use RKHS theory to do the extension

From Eigenvectors to Eigenfunctions

- Consider a Hilbert space $\mathcal{H}_p$ of functions with inner product and probability density p:
$$\langle f, g \rangle_p = \int f(\boldsymbol{x}) g(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$
- Given a kernel function, consider this linear operator:
$$\left(K_p f\right)(\boldsymbol{x}) = \int K(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y}) p(\boldsymbol{y}) d\boldsymbol{y}$$
- In practice, p is unknown $\Rightarrow$ approximate it with empirical distribution $\hat{p}$
  - New Hilbert space $\mathcal{H}_{\hat{p}}$
- The eigenfunction associated with the linear operator above gives the OOSE (next slide)

Nystrom Extension

Let $\widetilde{K}(\boldsymbol{a}, \boldsymbol{b})$ be a kernel function with a symmetric matrix $\widetilde{M}_{ij} = \widetilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ on a dataset $D = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$. Let $(\boldsymbol{v}_k, \lambda_k)$ solve the eigenvector equation $\widetilde{M}\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$. Let (f_k, λ'_k) solve the eigenfunction equation $(K_{\hat{p}} f_k)(\boldsymbol{x}) = \lambda'_k f_k(\boldsymbol{x})$ for any $\boldsymbol{x}$ with $\hat{p}$ the empirical distribution over D. Let $e_k(\boldsymbol{x}) = y_k(\boldsymbol{x})\sqrt{\lambda_k}$ or $y_k(\boldsymbol{x})$ denote the OOSE for $\boldsymbol{x}$. Then:

$$\lambda'_k = \frac{1}{n}\lambda_k, \qquad f_k(\boldsymbol{x}) = \frac{\sqrt{n}}{\lambda_k}\sum_{i=1}^{n} v_{ki}\widetilde{K}(\boldsymbol{x}, \boldsymbol{x}_i)$$

$$f_k(\boldsymbol{x}_i) = \sqrt{n}\boldsymbol{v}_{ki}, \qquad y_k(\boldsymbol{x}) = \frac{f_k(\boldsymbol{x})}{\sqrt{n}}$$

Nystrom Extension

$$\widetilde{M}\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$$

$$f_k(\boldsymbol{x}) = \frac{\sqrt{n}}{\lambda_k}\sum_{i=1}^{n} v_{ki}\widetilde{K}(\boldsymbol{x}, \boldsymbol{x}_i), \qquad y_k(\boldsymbol{x}) = \frac{f_k(\boldsymbol{x})}{\sqrt{n}}$$

Procedure

1. Compute $\widetilde{M}$ for training data
2. Solve the eigenvector equations $\widetilde{M}\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$
3. For each new point $\boldsymbol{x}$ and for each k, compute $f_k(\boldsymbol{x})$
4. Create new embedding $y_k(\boldsymbol{x})$ for $k = 1, \ldots, m$

- **Justification**: In the limit of more data, the eigenvector $\boldsymbol{v}_k$ converges to the eigenfunction f_k

Extending MDS

- Only need to define $\widetilde{K}$ as we defined $\widetilde{M}$ previously
- MDS

$$\widetilde{K}(\boldsymbol{a}, \boldsymbol{b}) = -\frac{1}{2}\left(d^2(\boldsymbol{a}, \boldsymbol{b}) - \sum_{i=1}^{n} d^2(\boldsymbol{x}_i, \boldsymbol{a}) - \sum_{i=1}^{n} d^2(\boldsymbol{x}_i, \boldsymbol{b}) + \sum_{i,j=1}^{n} d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)$$

Extending Laplacian Eigenmaps

- An initial kernel K is used (e.g. Gaussian)
- Normalized kernel:

$$\widetilde{K}(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{n} \frac{K(\boldsymbol{a}, \boldsymbol{b})}{\sqrt{\sum_{i=1}^{n} K(\boldsymbol{a}, \boldsymbol{x}_i) \sum_{j=1}^{n} K\left(\boldsymbol{b}, \boldsymbol{x}_j\right)}}$$

Extending Isomap and Diffusion Maps

<u>Isomap</u>

- Compute the geodesic distances for the new points using only the training points
- Use the double centered kernel in MDS

<u>Diffusion Maps</u>

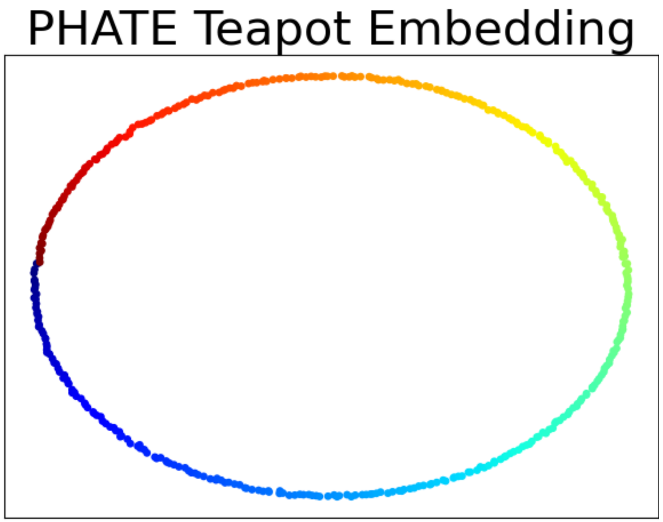
- More complicated
- See https://arxiv.org/abs/1802.08762 for one approach
- The compressed diffusion approach in PHATE also works
- Can try to use a different kernel for the extension than the one used for eigendecomposition
  - Requires more parameter tuning

Nystrom Summary

- Other variations on Nystrom exist (e.g. geometric harmonics)
- Nystrom methods work decently
- **Weakness 1**: nonparametric
  - Requires storing the training points
- **Weakness 2**: performance degrades pretty quickly in areas with sparse training data
- **Weakness 3**: selecting the extension kernel isn't always straightforward
  - Heavy tuning may be needed
- GRAE (a neural network approach) tends to do better at both and does not require careful selection of the kernel function
  - Although neural network tuning is required

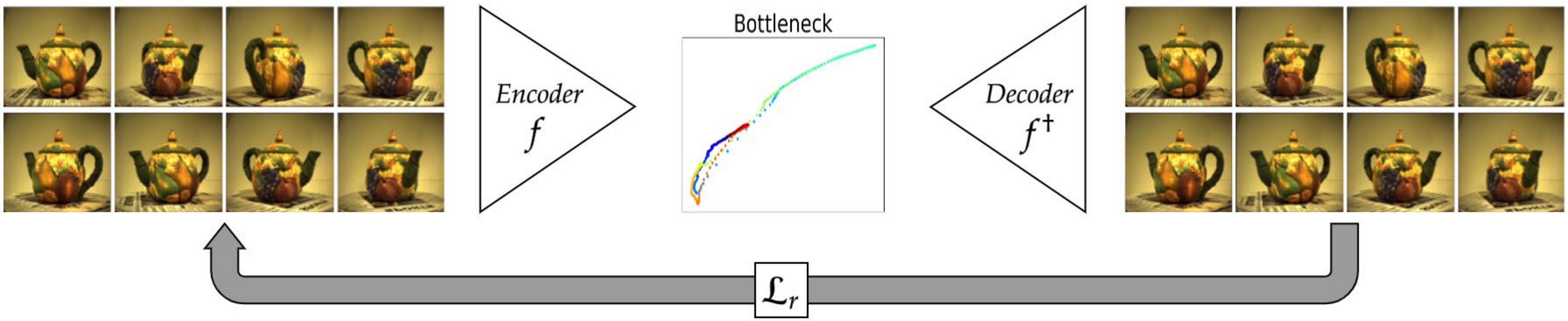
Geometry Regularized Autoencoders (GRAE)

- We combined kernel methods with AEs to obtain the advantages of both
  - Called it GRAE (Duque et al, 2022)
- **Example**: Rotated teapot dataset (400 images of a teapot from different angles)
- PHATE applied to the data:



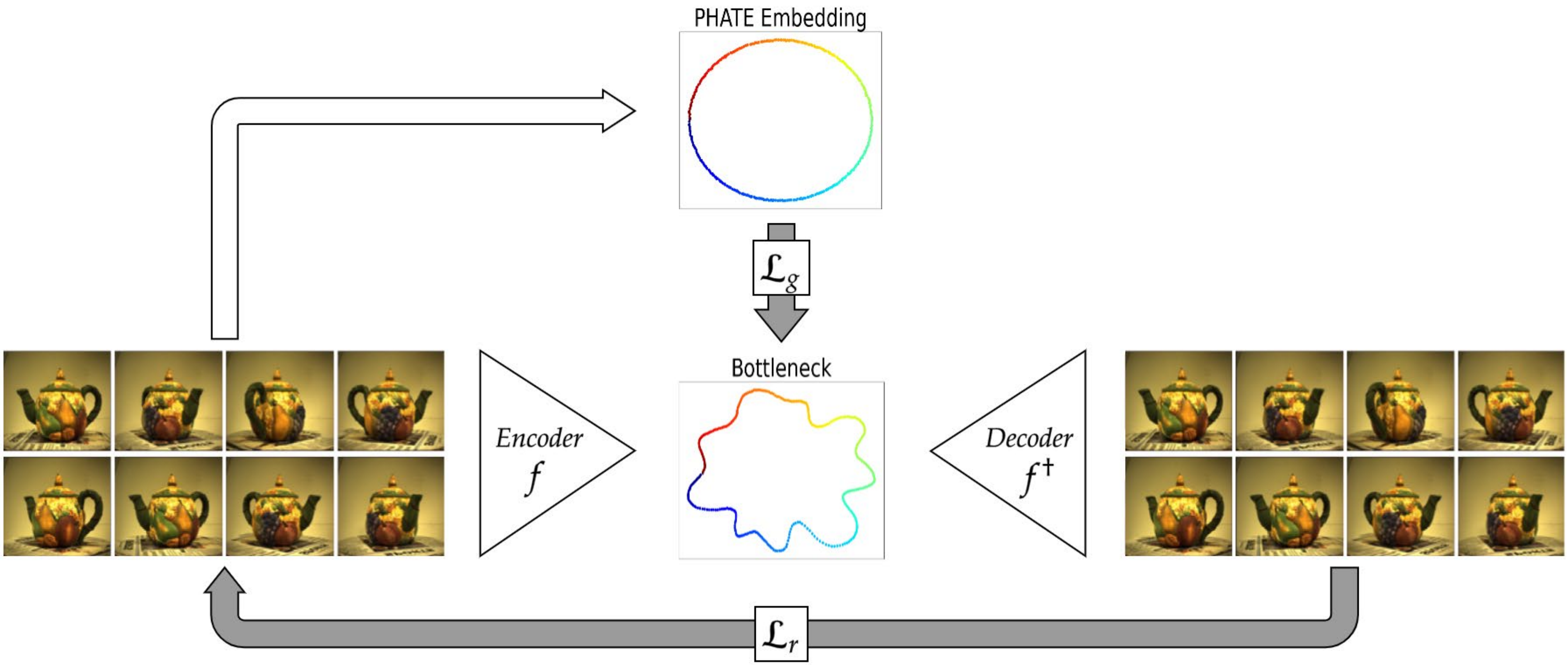
Geometry Regularized Autoencoders (GRAE)

- We combined kernel methods with AEs to obtain the advantages of both
  - Called it GRAE (Duque et al, 2022)
- **Example**: Rotated teapot dataset (400 images of a teapot from different angles)
- Standard AE applied to the data:



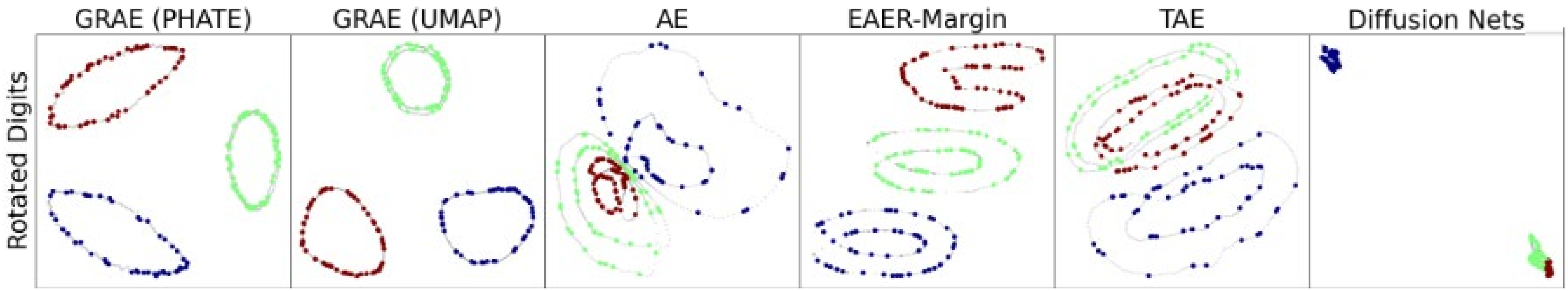
Geometry Regularized Autoencoders (GRAE)

- GRAE applied to the data with PHATE regularizing the bottleneck



GRAE Results – Rotated Digits

- Three MNIST digits with a full rotation
- GRAE preserves the rotation manifold and separates the 3 digits
- Other methods fail



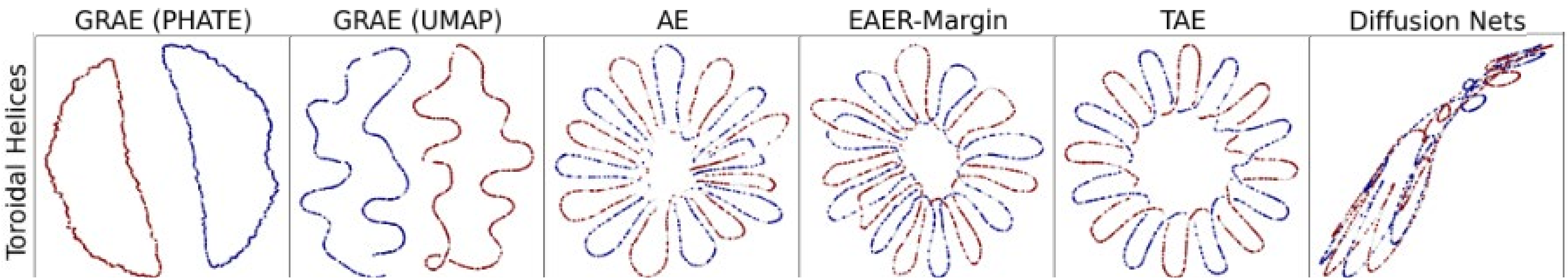
GRAE Results – Rotated Digits

- Three MNIST digits with a full rotation
- MSE – reconstruction error
- R^2 - linear regression predicting ground truth factors with the latent representation as input
- Acc. – classification accuracy with logistic regression on latent representation

Dataset	Model	Metrics		
		MSE	R^2	*Acc.*
Rotated Digits	GRAE (PHATE)	0.0002 (1)	0.6726 (2)	1.0000 (1)
	GRAE (UMAP)	0.0004 (2)	0.7042 (1)	1.0000 (1)
	AE	0.0021 (3)	0.1860 (5)	0.5309 (5)
	EAER-Margin	0.0045 (5)	0.3058 (3)	0.9222 (4)
	TAE	0.0034 (4)	0.2402 (4)	0.4679 (6)
	Diffusion Nets	0.0626 (6)	0.0442 (6)	0.9352 (3)

GRAE Results – Toroidal Helices

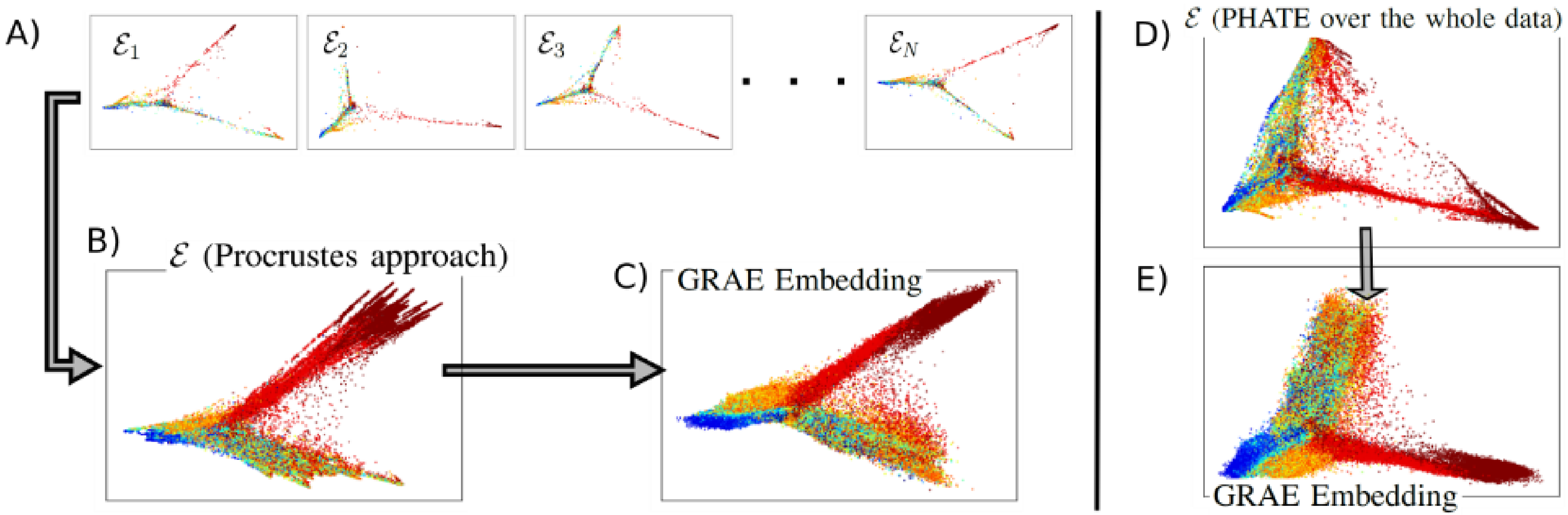
- A set of nonintersecting helices on the surface of a torus
- Again, the GRAE embeddings better represent the topology



Dataset	Model	Metrics		
		MSE	R^2	*Acc.*
Toroidal Helices	GRAE (PHATE)	0.0002 (1)	0.9845 (2)	0.9777 (2)
	GRAE (UMAP)	0.0002 (1)	0.7469 (5)	0.9998 (1)
	AE	0.0013 (3)	0.8159 (4)	0.5083 (4)
	EAER-Margin	0.0031 (5)	0.9199 (3)	0.5178 (3)
	TAE	0.0023 (4)	0.9984 (1)	0.5029 (5)
	Diffusion Nets	2.7309 (6)	0.1856 (6)	0.4918 (6)

GRAE on big data

- Applied GRAE to iPSC data (Zunder et al, 2015)
  - 220,000+ cells w/ 40+ markers
- Created multiple PHATE embeddings aligned with Procrustes
- Landmark PHATE: 3894 seconds
- GRAE: 850 seconds



Further reading

- Bengio et al, “Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering,” *NeurIPS*, 2003
- Geometric harmonics: https://www.sciencedirect.com/science/article/pii/S1063520306000522
- GRAE: https://doi.org/10.1109/TPAMI.2022.3222104