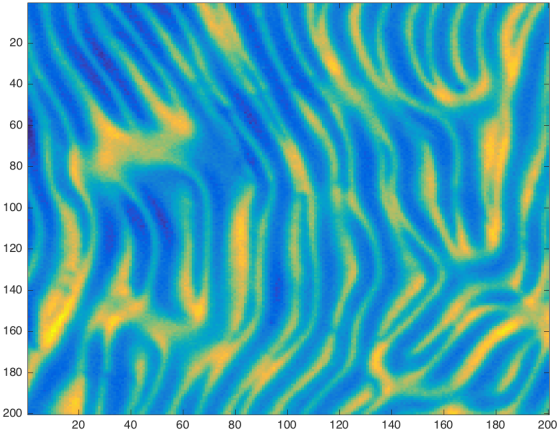
The Mathematics of Manifold Learning

Tyrus Berry
George Mason University

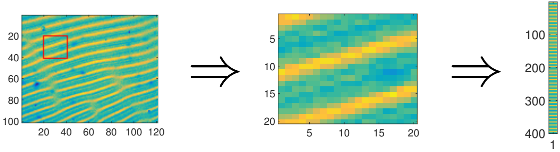
April 6, 2019

Motivating Example: Nematic Liquid Crystal



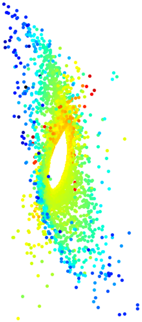
Video provided by Rob Cressman, Krasnow Institute, GMU

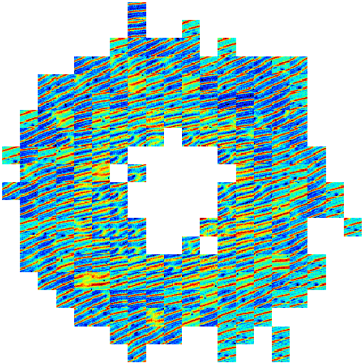
Finding Hidden Structure in Data



$\in \mathbb{R}^{400}$

The sub-image geometry:





OUTLINE

Lessons:

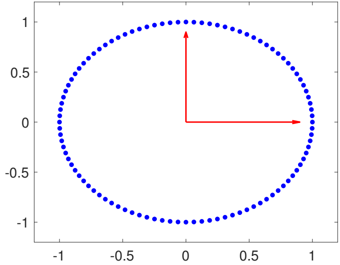
- **Dimensionality:** Intrinsic vs. Extrinsic
- **Nonlinearity:** Fourier Basis
- **Non-uniformity:** Respect the density

Challenges:

- Curse-of-dimensionality (intrinsic)
- Extrapolation

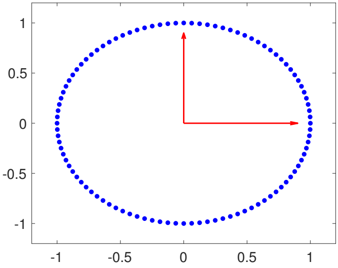
INTRINSIC VS. EXTRINSIC DIMENSION

100 points on a Circle



θ	x	y
0.0628	0.9980	0.0628
0.1257	0.9921	0.1253
0.1885	0.9823	0.1874
0.2513	0.9686	0.2487
0.3142	0.9511	0.3090
0.3770	0.9298	0.3681
0.4398	0.9048	0.4258
0.5027	0.8763	0.4818
⋮	⋮	
6.0319	0.9686	-0.2487
6.0947	0.9823	-0.1874
6.1575	0.9921	-0.1253
6.2204	0.9980	-0.0628
6.2832	1.0000	-0.0000

Intrinsic vs. Extrinsic Dimension



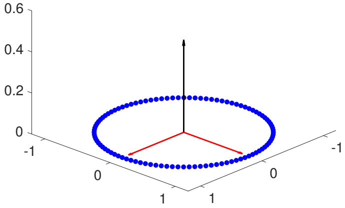
- Intrinsic Dimension $= 1$

$$\theta_i = 2\pi \frac{i}{100}$$

- Extrinsic Dimension $= 2$

$$(x_i, y_i) = (\cos(\theta_i), \sin(\theta_i))$$

Intrinsic vs. Extrinsic Dimension



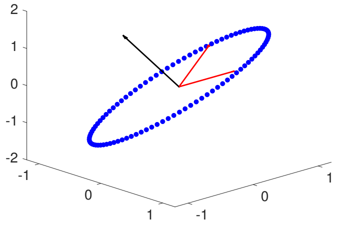
- Intrinsic Dimension $= 1$

$$\theta_i = 2\pi \frac{i}{100}$$

- Extrinsic Dimension $= 3$

$$(x_i, y_i, z_i) = (\cos(\theta_i), \sin(\theta_i), 0)$$

INTRINSIC VS. EXTRINSIC DIMENSION



- Intrinsic Dimension $= 1$

$$\theta_i = 2\pi \frac{i}{100}$$

- Extrinsic Dimension $= 3$

$$x_i = \cos(\theta_i)$$
$$y_i = \sin(\theta_i)$$
$$z_i = x_i + y_i$$

INTRINSIC VS. EXTRINSIC DIMENSION

- Intrinsic Dimension $= 1$

$$\theta_i = 2\pi \frac{i}{100}$$

- Extrinsic Dimension $= 2 + n$

$$\begin{aligned} x_i &= \cos(\theta_i) \\ y_i &= \sin(\theta_i) \\ z_i^1 &= a_1 x_i + b_1 y_i \\ &\vdots \\ z_i^n &= a_n x_i + b_n y_i \end{aligned} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ a_1 & a_2 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix} \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix} = A \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix}$$

A is a $(n+2) \times 2$ matrix

SOLUTION: LINEAR ALGEBRA!

- Hidden Data, $[\theta_1, \theta_2, \theta_3, ..., \theta_N]$
- Ideal Representation, $x_i = \cos(\theta_i), y_i = \sin(\theta_i)$

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_N \\ y_1 & y_2 & y_3 & \cdots & y_N \end{bmatrix}$$

- Given Data: $Y = AX$

$$Y = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_N \\ y_1 & y_2 & y_3 & \cdots & y_N \\ a_1x_1 + b_1y_1 & a_1x_2 + b_1y_2 & a_1x_3 + b_1y_3 & \cdots & a_1x_N + b_1y_N \\ a_2x_1 + b_2y_1 & a_2x_2 + b_2y_2 & a_2x_3 + b_2y_3 & \cdots & a_2x_N + b_2y_N \\ \vdots & \vdots & \vdots & & \vdots \\ a_nx_1 + b_ny_1 & a_nx_2 + b_ny_2 & a_nx_3 + b_ny_3 & \cdots & a_nx_N + b_ny_N \end{bmatrix}$$

- Rows of Y are linearly dependent!

Solution: Linear Algebra!

- Given data $Y = AX$ where both A and X are unknown
- Linear dependence means the rows, Y_i, are redundant:

$$\vec{c}^\top Y = c_1 Y_1 + c_2 Y_2 + \cdots + c_n Y_n = \vec{0}$$

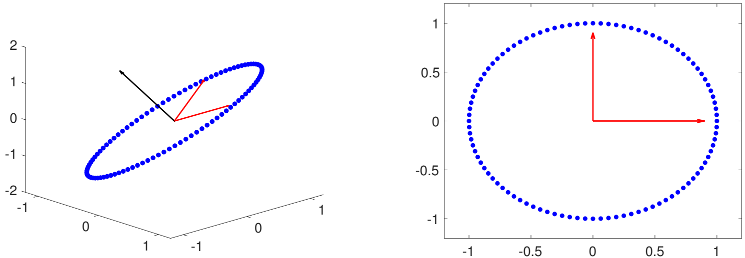
- There exists $\vec{c} = (c_1, ..., c_n) \neq 0$ such that $\vec{c}^\top Y = \vec{0}$
- $\vec{c}^\top Y = \vec{0}$ if and only if $\vec{c}^\top Y Y^\top \vec{c} = \vec{0}^\top \vec{0} = 0$
- So $\vec{c}$ is eigenvector of $YY^\top$ with eigenvalue zero

PRINCIPAL COMPONENT ANALYSIS (PCA)

- Compute the eigenvectors/values of $YY^\top = U\Lambda U^\top$
- Sort the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$
- Eigenvalue ≈ 0 represent linear redundancies
- **Principal Components:** Eigenvectors u_i with largest λ_i
- Choose $\vec{u}_1, \vec{u}_2, ..., \vec{u}_p$ corresponding to $\lambda_1, ..., \lambda_p$
- Form the projection matrix $P = [\vec{u}_1 \;\; \vec{u}_2 \;\; \cdots \;\; \vec{u}_p]$
- Remove redundancies: $\tilde{X} = PY$

Principal Component Analysis (PCA)

$$\mathbf{Y} \quad \Rightarrow \quad \tilde{\mathbf{X}} = \mathbf{PY}$$

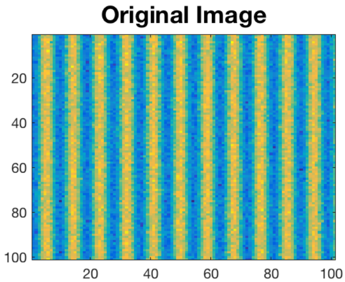


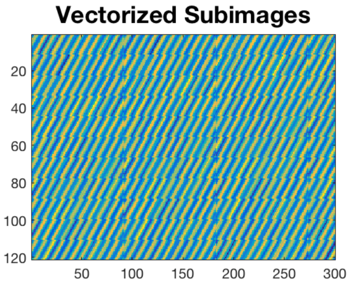
Principal Component Analysis (PCA)

- Matrix times *intrinsic* data $\Rightarrow$ *extrinsic* redundancy
- These *linear* redundancies are easy to remove
- PCA projects the data to remove redundancy
- Does this really happen?

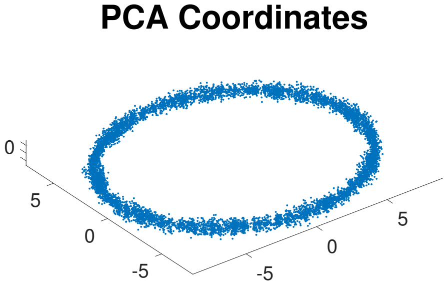
DOES THIS REALLY HAPPEN?

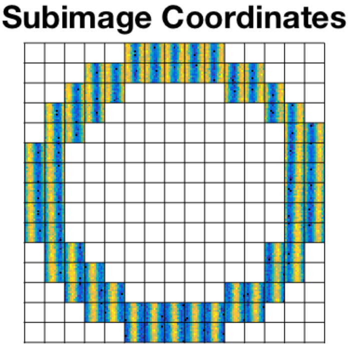
Consider 11×11 subimages from a pattern:



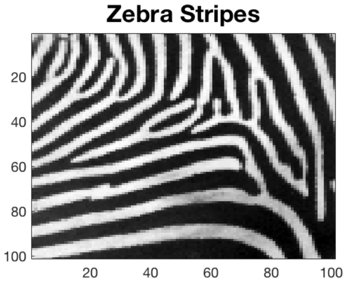


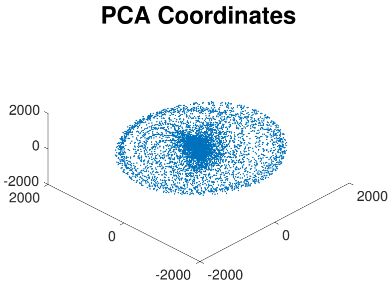
Does this really happen?

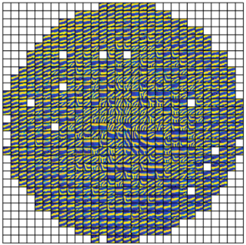




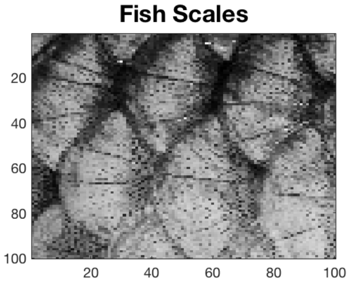
Does this really happen?

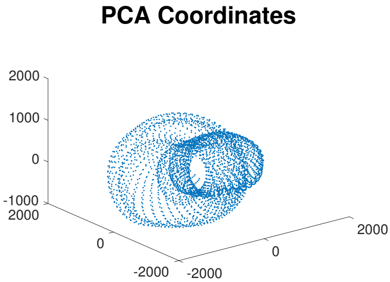


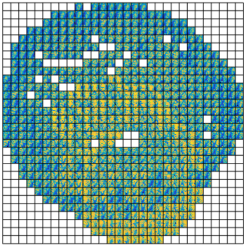




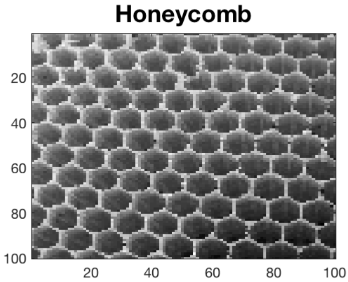
Does this really happen?

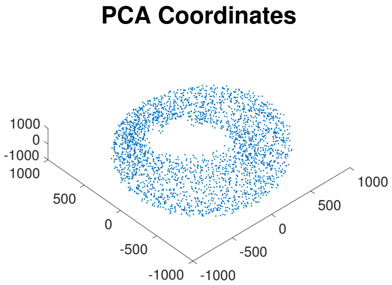


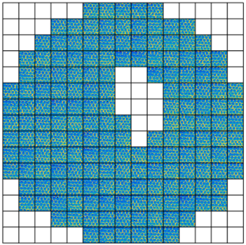




Does this really happen?







PRINCIPAL COMPONENT ANALYSIS (PCA)

- *Linear* redundancies are easy to remove

$$c_1 Y_1 + c_2 Y_2 + \cdots c_n Y_n = \vec{0}$$

Principal Component Analysis (PCA)

- *Linear* redundancies are easy to remove

$$c_1 Y_1 + c_2 Y_2 + \cdots c_n Y_n = \vec{0}$$

- PCA projects the data to remove redundancy

PRINCIPAL COMPONENT ANALYSIS (PCA)

- *Linear* redundancies are easy to remove

$$c_1 Y_1 + c_2 Y_2 + \cdots c_n Y_n = \vec{0}$$

- PCA projects the data to remove redundancy
- What about nonlinear redundancies?

$$F(Y_1, Y_2, ..., Y_n) = \vec{0}$$

PRINCIPAL COMPONENT ANALYSIS (PCA)

- *Linear* redundancies are easy to remove

$$c_1 Y_1 + c_2 Y_2 + \cdots c_n Y_n = \vec{0}$$

- PCA projects the data to remove redundancy
- What about nonlinear redundancies?

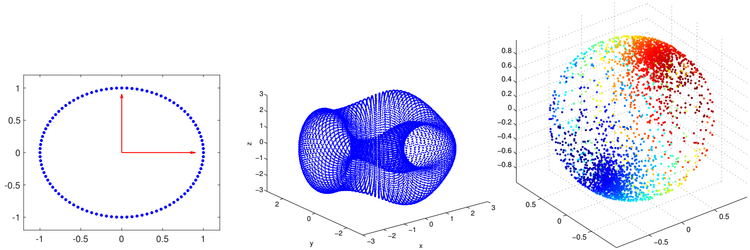
$$F(Y_1, Y_2, ..., Y_n) = \vec{0}$$

- Example, Circle: $Y_1 = \cos(\theta)$, $Y_2 = \sin(\theta)$

$$F(Y_1, Y_2) = Y_1^2 + Y_2^2 - 1 = \vec{0}$$

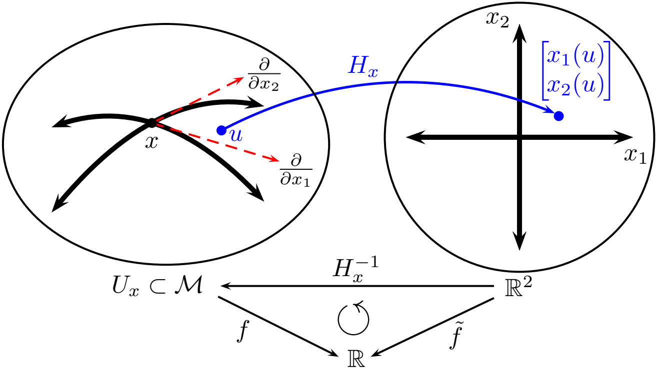
MANIFOLD LEARNING

A *manifold* $\mathcal{M}$ is a topological space that is *locally* Euclidean.



MANIFOLD LEARNING

Around each point $x \in \mathcal{M}$ we have an open neighborhood $U_x \subset \mathcal{M}$ and a homeomorphism $H_x : U_x \to \mathbb{R}^m$



MANIFOLD LEARNING

- When does a nonlinear redundancy define a manifold?

$$\mathcal{M} = \{\vec{y} \mid F(y_1, y_2, ..., y_n) = \vec{a}\} \subset \mathbb{R}^n$$

MANIFOLD LEARNING

- When does a nonlinear redundancy define a manifold?

$$\mathcal{M} = \{\vec{y} \mid F(y_1, y_2, ..., y_n) = \vec{a}\} \subset \mathbb{R}^n$$

- Need to be able to solve for the last $n - m$ variables:

$$\begin{aligned} \vec{a} &= F(y_1, ..., y_m, y_{m+1}, ..., y_n) \\ &= F(y_1, ..., y_m, G_1(y_1, ..., y_m), G_2(y_1, ..., y_m), ..., G_{n-m}(y_1, ..., y_m)) \end{aligned}$$

MANIFOLD LEARNING

- When does a nonlinear redundancy define a manifold?

$$\mathcal{M} = \{\vec{y} \mid F(y_1, y_2, ..., y_n) = \vec{a}\} \subset \mathbb{R}^n$$

- Need to be able to solve for the last $n - m$ variables:

$$\begin{aligned}\vec{a} &= F(y_1, ..., y_m, y_{m+1}, ..., y_n) \\ &= F(y_1, ..., y_m, G_1(y_1, ..., y_m), G_2(y_1, ..., y_m), ..., G_{n-m}(y_1, ..., y_m))\end{aligned}$$

- **Implicit Function Theorem:** If the Jacobian matrix $DF(\vec{y})$ is full rank then the functions $G_1, ...G_{n-m}$ exist near $\vec{y}$

MANIFOLD LEARNING

- When does a nonlinear redundancy define a manifold?

$$\mathcal{M} = \{\vec{y} \mid F(y_1, y_2, ..., y_n) = \vec{a}\} \subset \mathbb{R}^n$$

- Need to be able to solve for the last $n - m$ variables:

$$\begin{aligned}\vec{a} &= F(y_1, ..., y_m, y_{m+1}, ..., y_n) \\ &= F(y_1, ..., y_m, G_1(y_1, ..., y_m), G_2(y_1, ..., y_m), ..., G_{n-m}(y_1, ..., y_m))\end{aligned}$$

- **Implicit Function Theorem:** If the Jacobian matrix $DF(\vec{y})$ is full rank then the functions $G_1, ...G_{n-m}$ exist near $\vec{y}$
- **Sard's Theorem:** If F is smooth, then for *almost every* $\vec{a}$, the Jacobian $DF(\vec{y})$ is full rank for all $\vec{y} \in \mathcal{M}$

MANIFOLD LEARNING

- When does a nonlinear redundancy define a manifold?

$$\mathcal{M} = \{\vec{y} \mid F(y_1, y_2, ..., y_n) = \vec{a}\} \subset \mathbb{R}^n$$

Manifold Learning

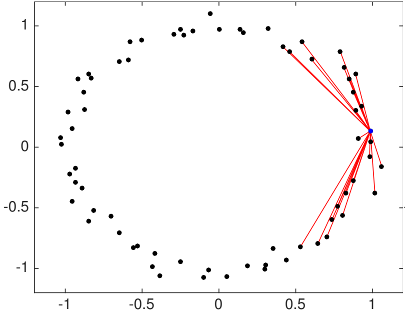
- When does a nonlinear redundancy define a manifold?

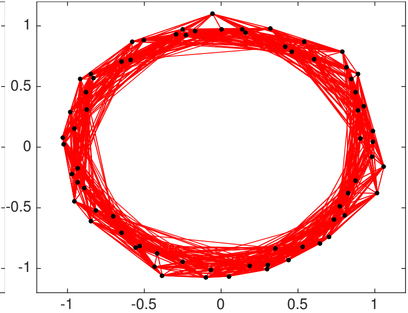
$$\mathcal{M} = \{\vec{y} \mid F(y_1, y_2, ..., y_n) = \vec{a}\} \subset \mathbb{R}^n$$

- When F is smooth, $\mathcal{M}$ is a manifold for almost every $\vec{a}$

MANIFOLD ⇒ GRAPH

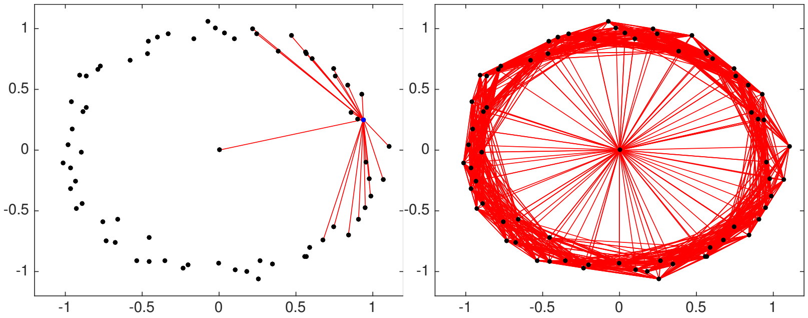
- Represent the nonlinear structure with a graph
- Locally Euclidean ⇒ Connect nearby points





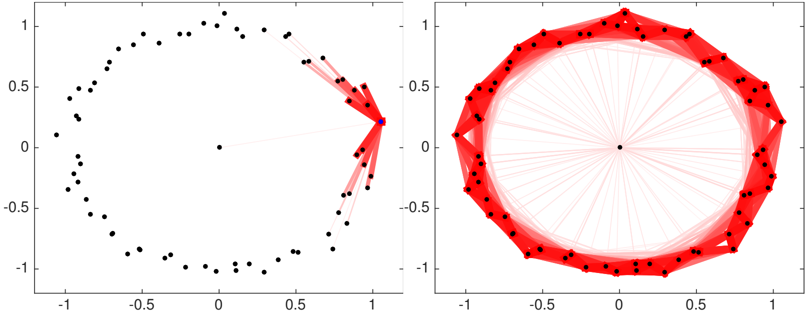
MANIFOLD ⇒ GRAPH

- **Problem:** Noise and outliers can lead to bridging



MANIFOLD $\Rightarrow$ GRAPH

- To prevent bridging, edges weighted: $K_\delta(x,y) = e^{-\frac{||x-y||^2}{4\delta^2}}$
- **Theorem:** Graph encodes all nonlinear information



WHAT IS MANIFOLD LEARNING?

- **Manifold learning $\Leftrightarrow$ Estimating Laplace Operator**
- Euclidean space:
  - Eigenfunctions of Δ are $e^{i\vec{\omega}\cdot\vec{x}}$
  - Basis for Fourier transform
- Unit circle:
  - Eigenfunctions of Δ are $e^{ik\theta}$
  - Basis for Fourier series
- **Theorem**: Eigenfunctions of Δ give the smoothest basis for square integrable functions on any manifold.

FINDING THE LAPLACIAN FROM DATA

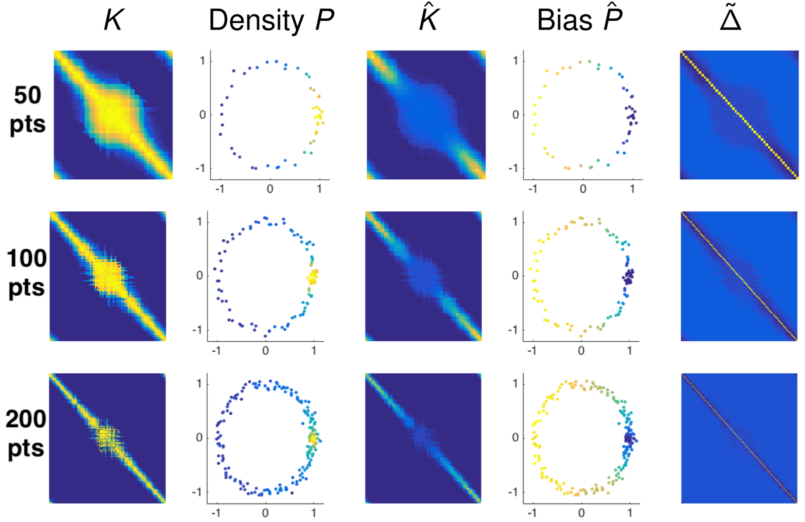
- We have converted our data set to a *weighted graph*
- Vertices $\Rightarrow$ Data points $\{x_1, x_2, ..., x_N\}$
- Edges $\Rightarrow$ Pairs of nearest neighbors
- Edge Weights $\Rightarrow K(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{4\epsilon}}$
- Represented as matrix $K_{ij} = K(x_i, x_j)$

DIFFUSION MAPS: THE KEY RESULT

1. Start with the matrix $K_{ij} = e^{-\frac{||x_i - x_j||^2}{4\epsilon}}$
2. Find the row sums $P_i = \sum_{j=1}^{N} K_{ij}$
3. Normalize the matrix $\hat{K}_{ij} = \frac{K_{ij}}{P_i P_j}$
4. Find the row sums again $\hat{P}_i = \sum_{j=1}^{N} \hat{K}_{ij}$
5. Markov Normalization $\tilde{K}_{ij} = \frac{\hat{K}_{ij}}{\hat{P}_i}$
6. Form the Laplacian matrix $\tilde{\Delta} = \frac{I - \tilde{K}}{\epsilon}$

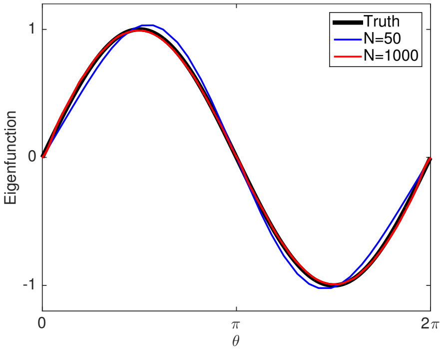
Theorem: As $N \to \infty$ and $\epsilon \to 0$ we have $\tilde{\Delta} \to \Delta$

DIFFUSION MAPS CONSTRUCTION



DIFFUSION MAPS CONSTRUCTION

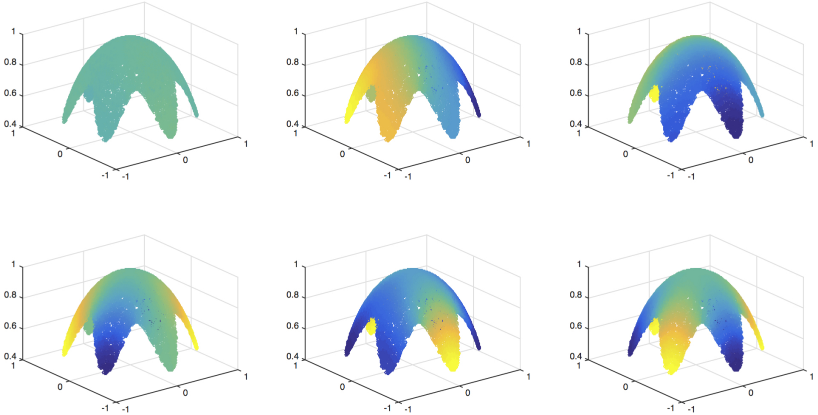
- $\tilde{\Delta}$ approximates the Laplacian Δ
- $\tilde{\Delta}$ encodes the geometry of the data
- Eigenvectors of $\tilde{\Delta}$ approximate eigenfunctions of Δ



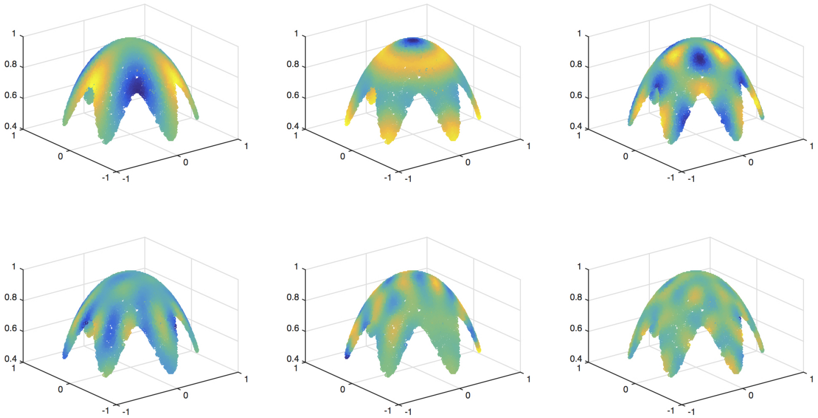
FOURIER BASIS ON MANIFOLDS

- Fourier functions $\sin(k\theta)$ are eigenfunctions of $\frac{d^2}{d\theta^2}$
- Eigenvectors of matrix $\tilde{\Delta}$ approximate eigenfunctions of Δ
- What is so great about these functions?
  - Smoothest possible functions on $\mathcal{M}$
  - $\varphi_0 =$ constant
  - φ_1 contains a single oscillation
  - φ_j is smoothest function orthogonal to previous

FOURIER BASIS ON MANIFOLDS



Fourier Basis on Manifolds



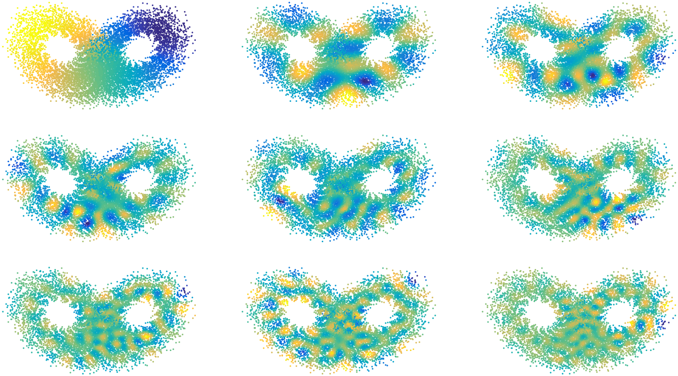
FORECASTING WITHOUT A MODEL

$$\begin{array}{ccc}
p(x,t) & \xrightarrow{\text{Nonparametric Forecast}} & p(x,t+\tau) \\
\Big\downarrow \langle p,\varphi_j \rangle & & \Big\uparrow \sum_j c_j \varphi_j p_{\mathrm{eq}} \\
\vec{c}(t) & \xrightarrow{A_{lj} \equiv \mathbb{E}[\langle \varphi_j, S\varphi_l \rangle_{p_{\mathrm{eq}}}]} & \vec{c}(t+\tau) = A\vec{c}(t).
\end{array}$$

- $\vec{c}(t)$ are the generalized Fourier coefficients of p
- Nonlinear dynamics become linear (matrix A) in this basis

MANIFOLD LEARNING ⇒ CUSTOM ‘FOURIER’ BASIS

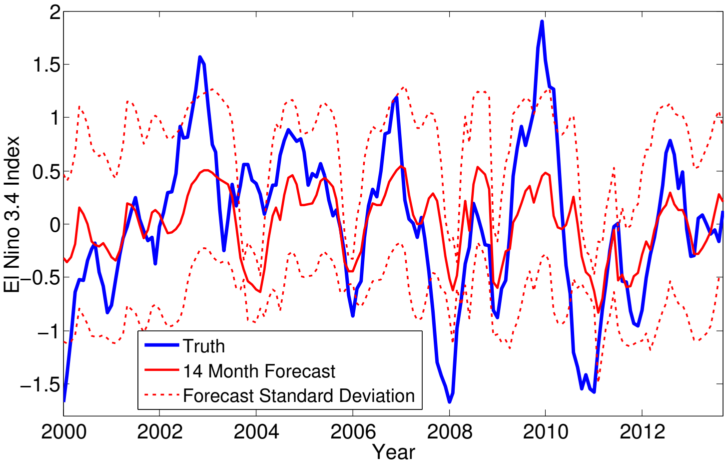
- **Optimal basis:** Minimum variance $A_{lj} \equiv \mathbb{E}[\langle \varphi_j, S\varphi_l \rangle_q]$



EXAMPLE: FORECASTING WITHOUT A MODEL

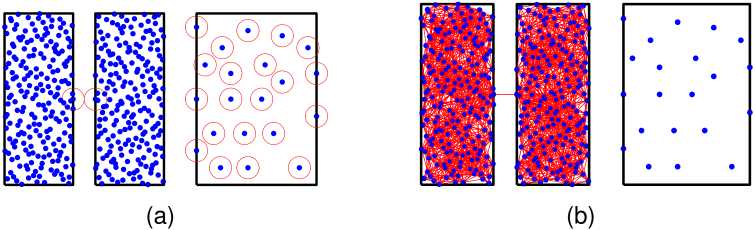
No Model Perfect Model

EXAMPLE: FORECASTING EL NINO



NONUNIFORM DENSITY: FIXED BALLS

Black outlines indicate true clusters:

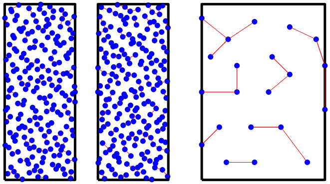


(a) Dense regions bridged before connecting sparse region

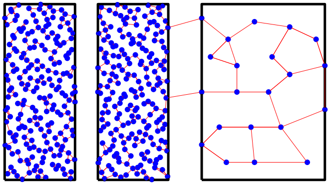
(b) Graph connecting all points with distance less than ϵ

$$||x - y|| < \epsilon$$

NONUNIFORM DENSITY: NEAREST NEIGHBORS (NN)



(c)

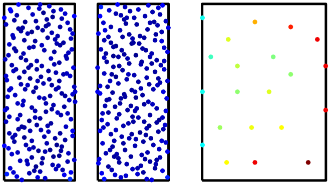


(d)

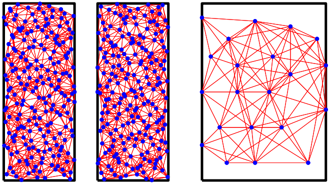
(c) Connect each point to its nearest neighbor (NN)

(d) Connect each point to its two nearest neighbors (2NN)

NONUNIFORM DENSITY: CKNN



(e)



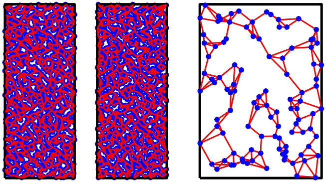
(f)

(e) Distance to 10-th nearest neighbor

(f) **Continuous k-Nearest Neighbors (CkNN)**

$$\frac{||x-y||}{\sqrt{||x-\mathrm{kNN}(x)||\cdot||y-\mathrm{kNN}(y)||}}<\delta$$

NONUNIFORM DENSITY: MORE DATA?

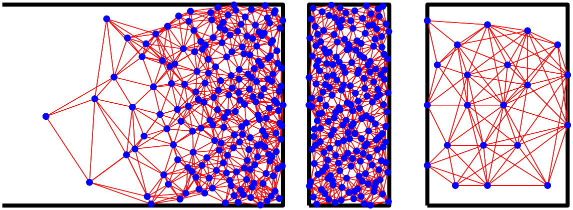


(g)

(g) Five times more data, 4 nearest neighbors works

Does nearest neighbors always work given sufficient data?

Nonuniform Density: Conclusion



(h)

(h) Real data has sparse tails: More data = bigger gaps!

Theorem: NN fails even with infinite data. CkNN succeeds.

Improved clustering using CkNN

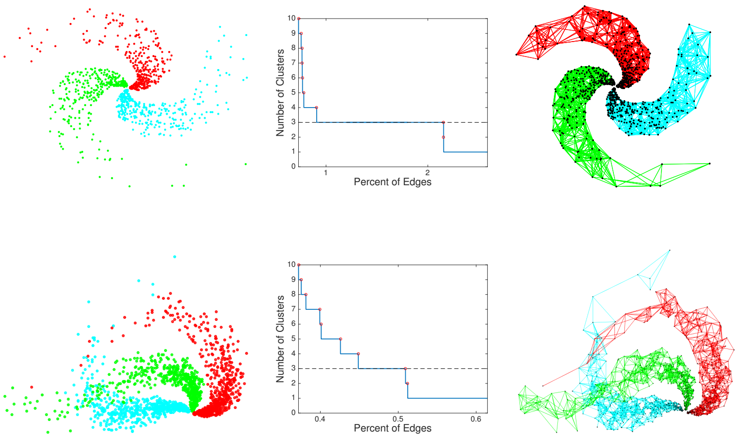
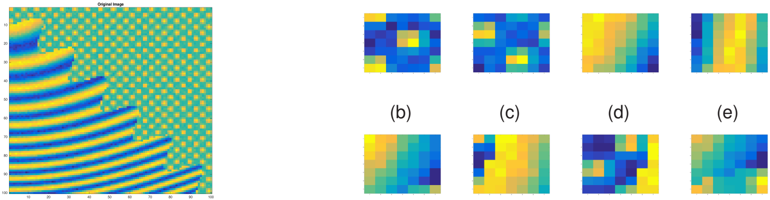


IMAGE SEGMENTATION

Original Image: Break into subimages



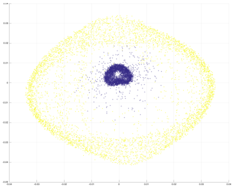
(a) (b) (c) (d) (e) (f) (g) (h) (i)

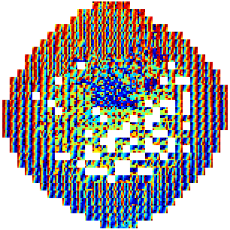
Images produced by Marilyn Vazquez.

IMAGE SEGMENTATION

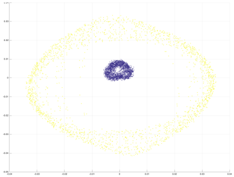
Clustering shown projected to two principal components

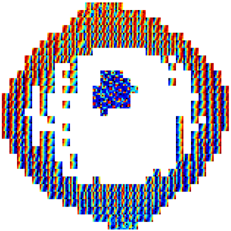
all points





with low density points removed

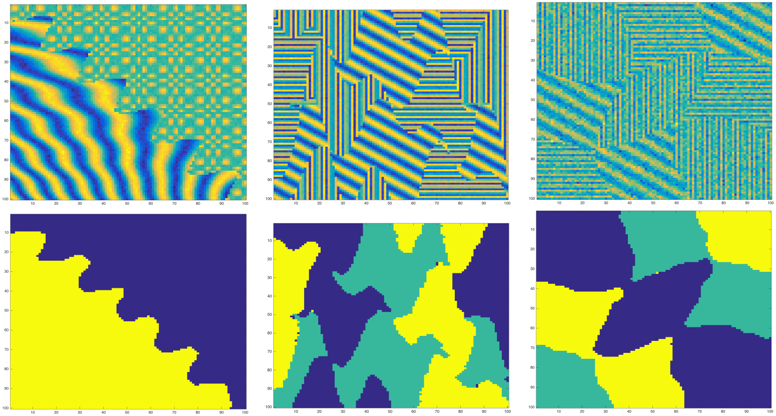




Images produced by Marilyn Vazquez.

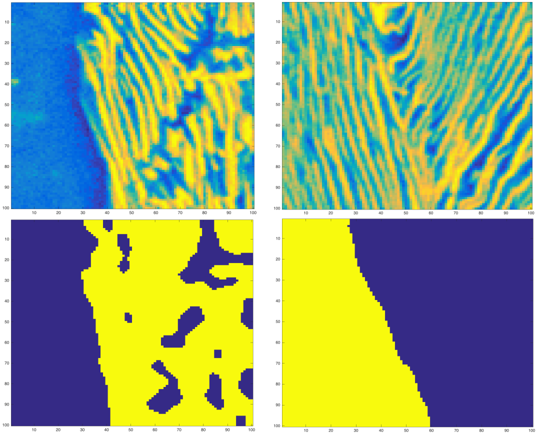
IMAGE SEGMENTATION

Results - synthetic images



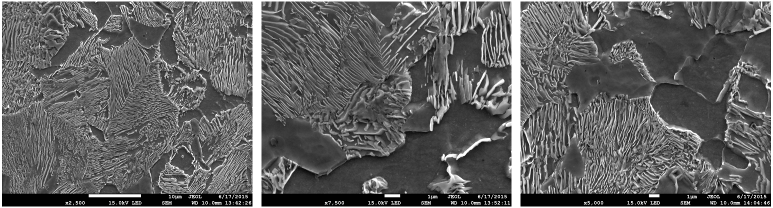
Images produced by Marilyn Vazquez.

IMAGE SEGMENTATION: REAL IMAGES

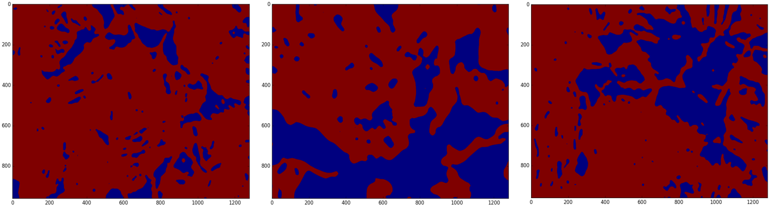


Images produced by Marilyn Vazquez.

Image segmentation: Real images



(g) (h) (i)



(j) (k) (l)

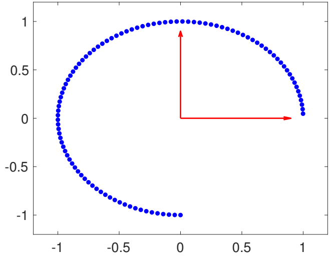
Original images by Mark R. Stoudt and Steve P. Mates. Analysis by Marilyn Vazquez.

CURSE-OF-(INTRINSIC)-DIMENSIONALITY

- **Try** to cut into independent components
- Otherwise math/stat says it is impossible
- Need more/better assumptions and/or questions
- **Better assumptions:** Smoothness
- **Better questions:** Feature of interest (supervised)

EXTRAPOLATION

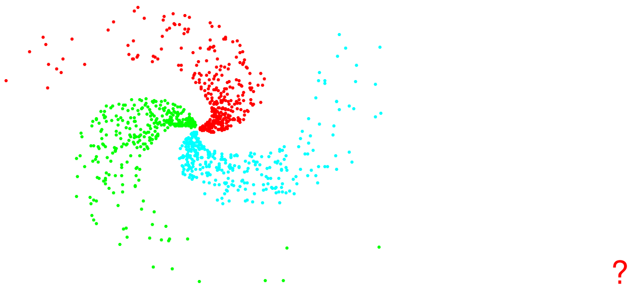
- Given only part of a structure recover the whole



- Need to exploit symmetry

Extrapolation

- Given only part of a structure recover the whole



- Need to exploit symmetry