

When poll is active, respond at **PollEv.com/yaleml**

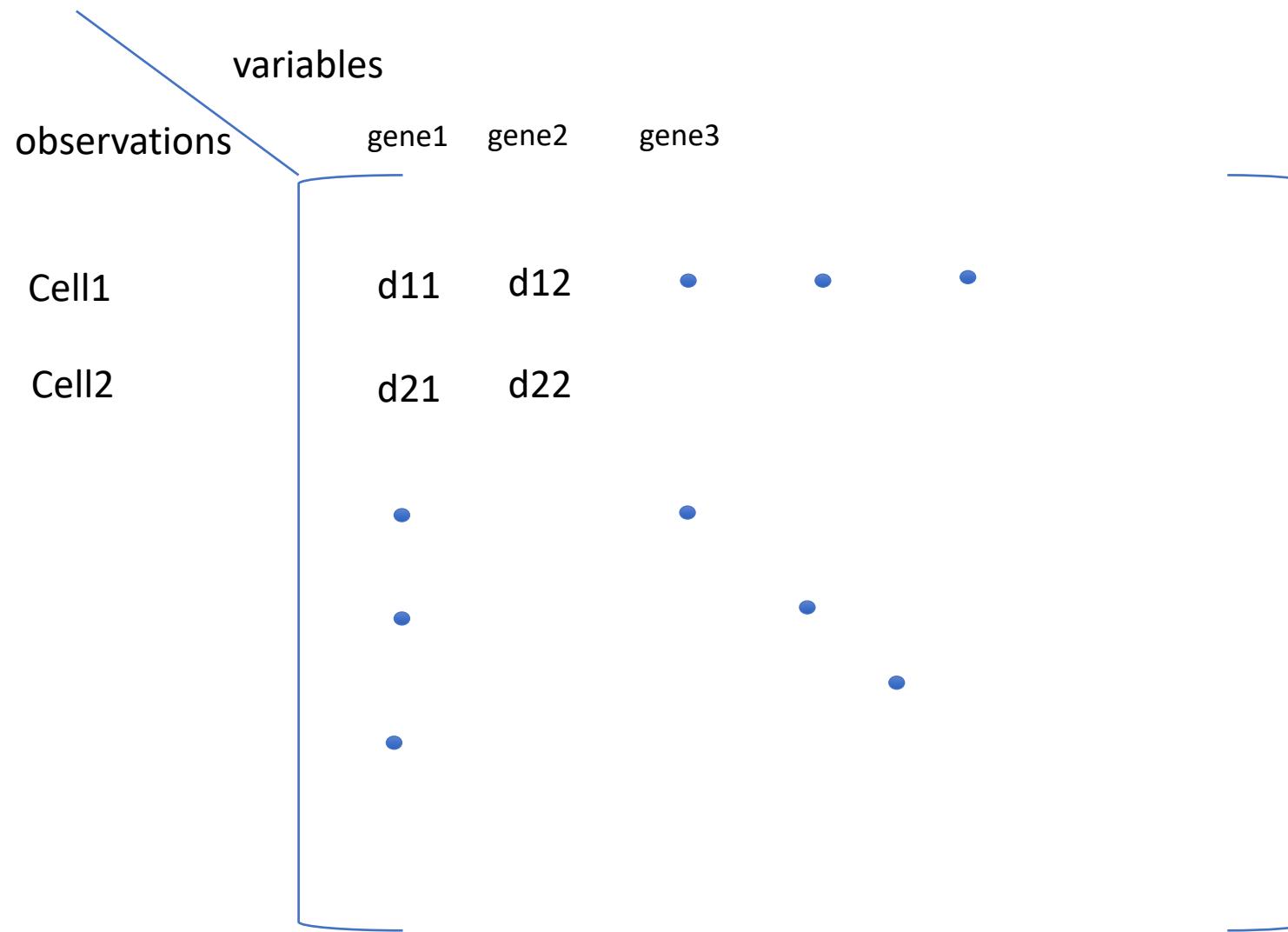
Text **YALEML** to **22333** once to join

What's the last great TV show or movie you watched?

Day 2: Visualization, Dimensionality Reduction and Manifold Learning

Thinking about high dimensional data

Our Data is a High-dimensional Matrix



Features are dimensions

vector coordinate space

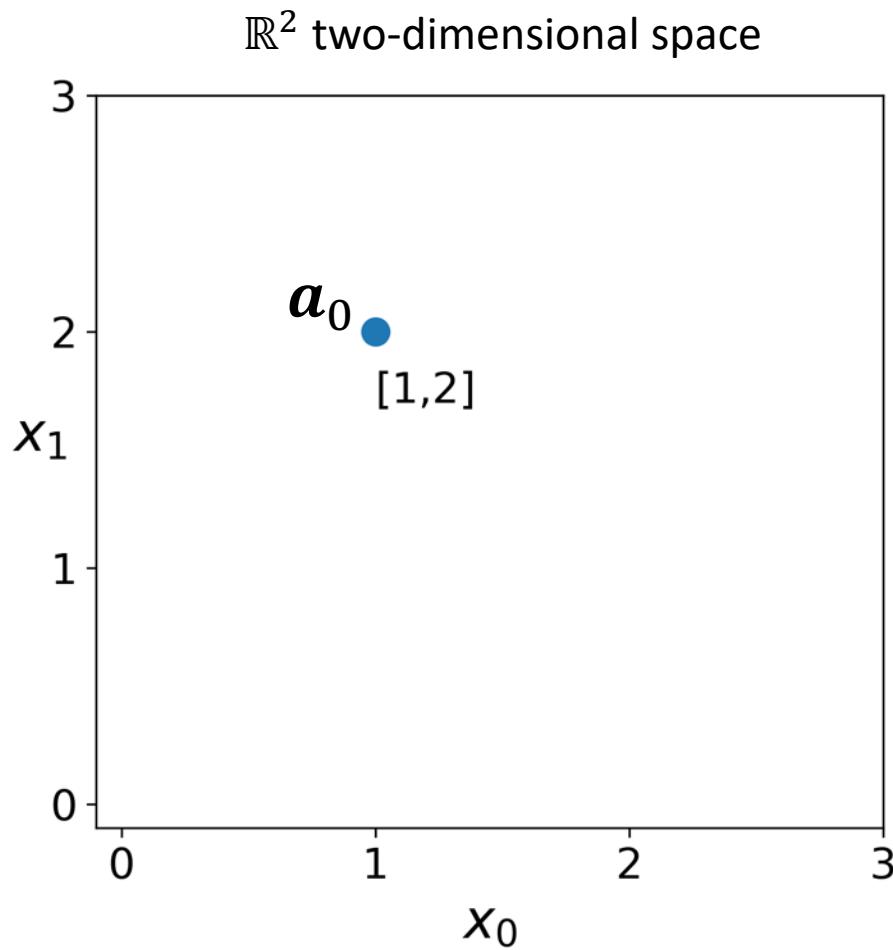
$a_0 \in \mathbb{R}^2$

$a_0 = [1, 2]$

$x_0 \quad x_1$

features

$a_{[:,0]}$



Features are dimensions

$$\mathbf{a} \in \mathbb{R}^2$$

$$\mathbf{a} = [1, 2]$$

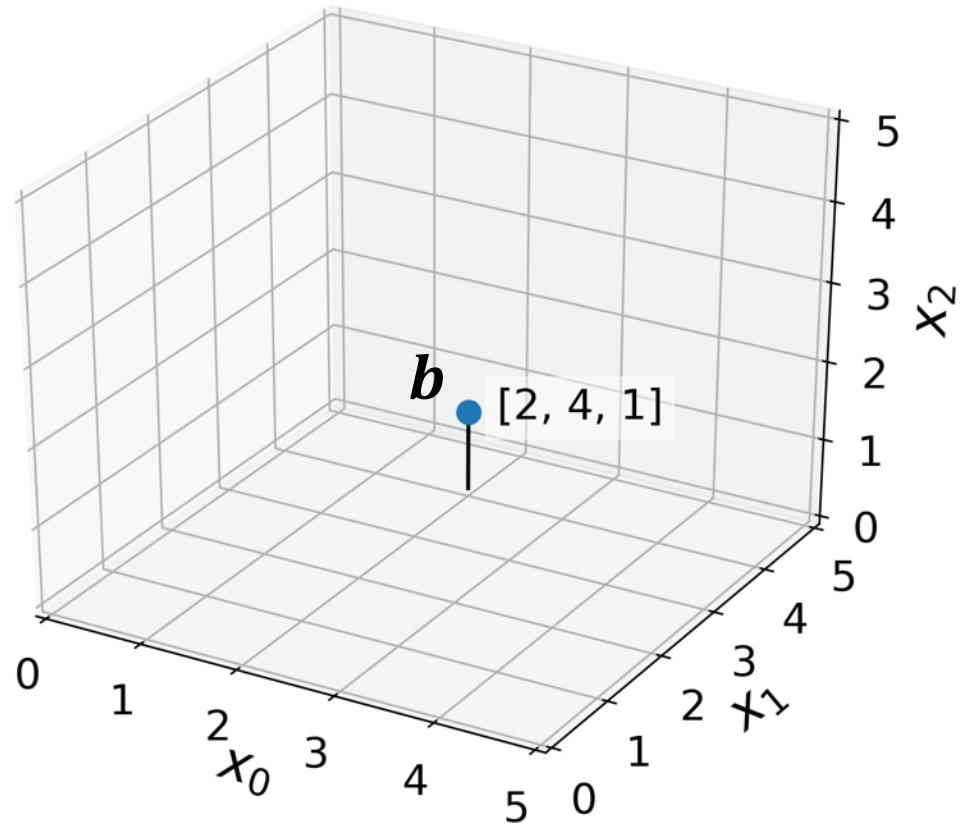
x_0 x_1

$$\mathbf{b} \in \mathbb{R}^3$$

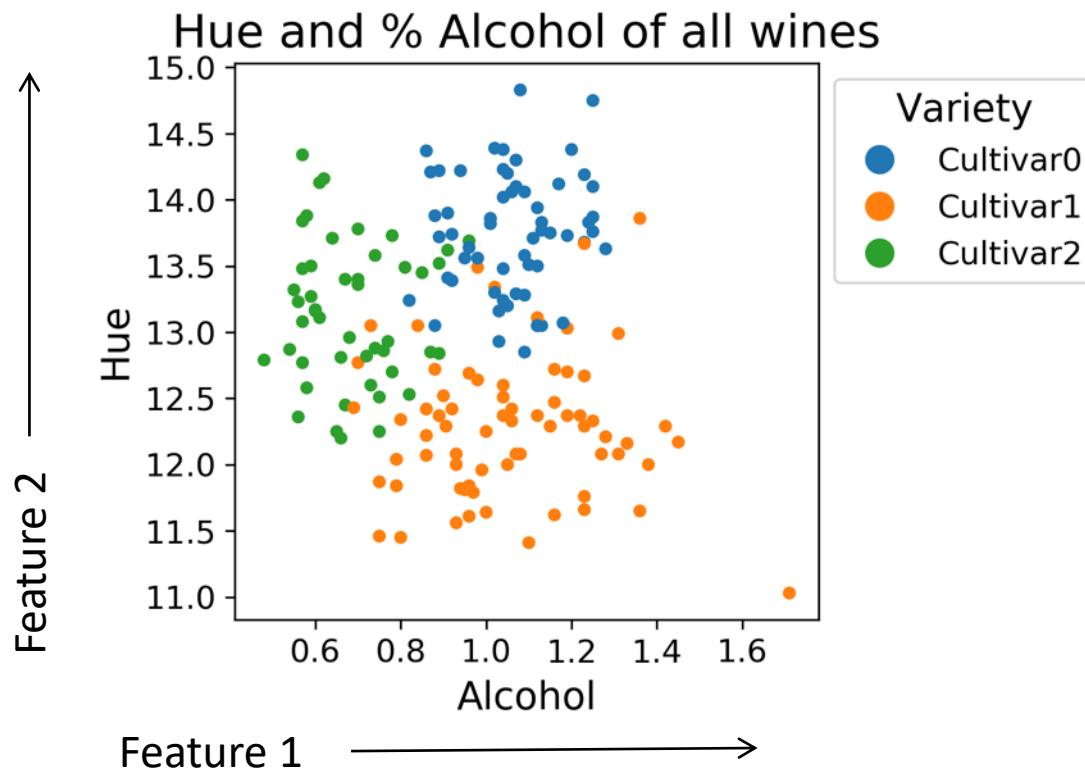
$$\mathbf{b} = [2, 4, 1]$$

x_0 x_1 x_2

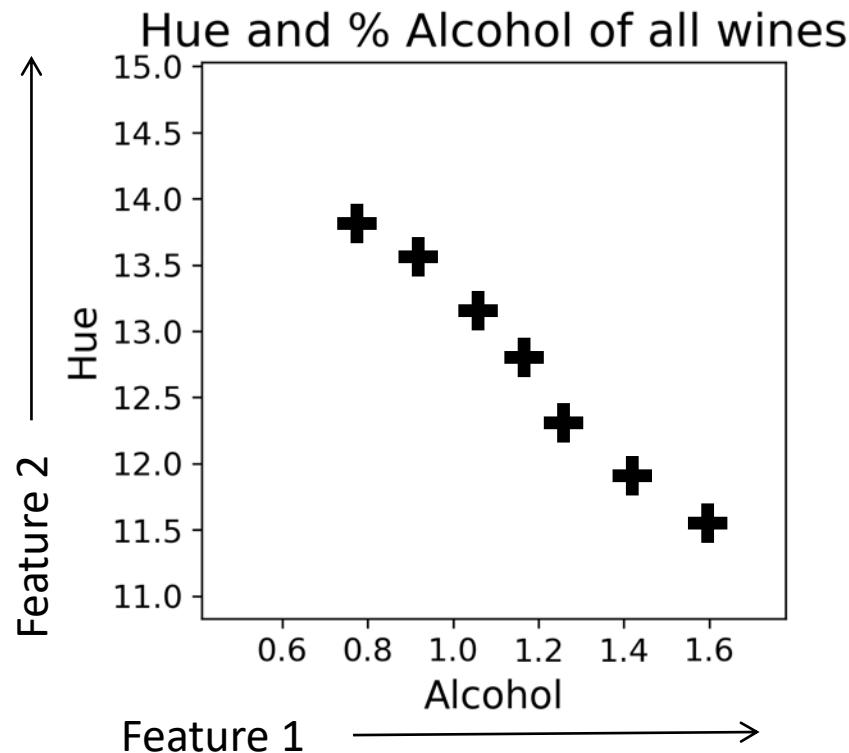
\mathbb{R}^3 three-dimensional space



Features reveal structure

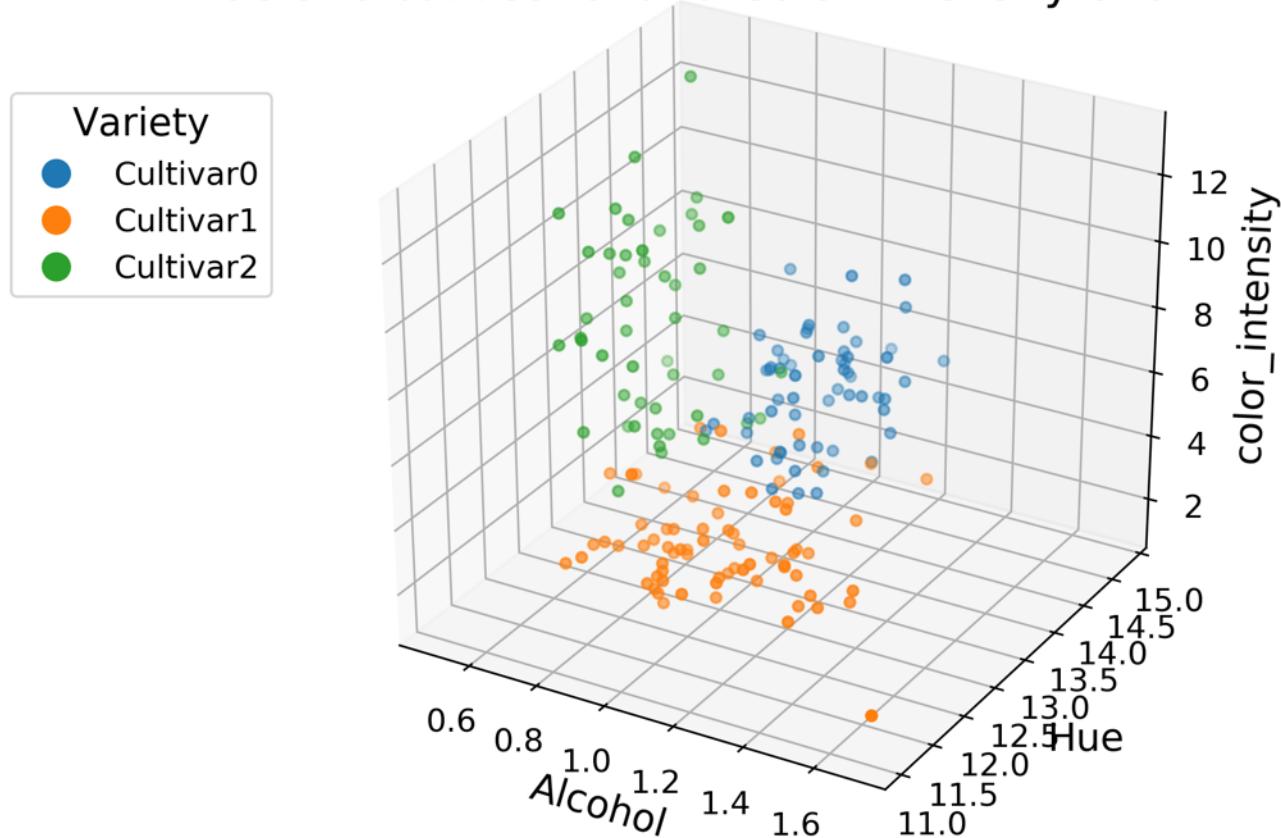


Many types of structure can exist in data

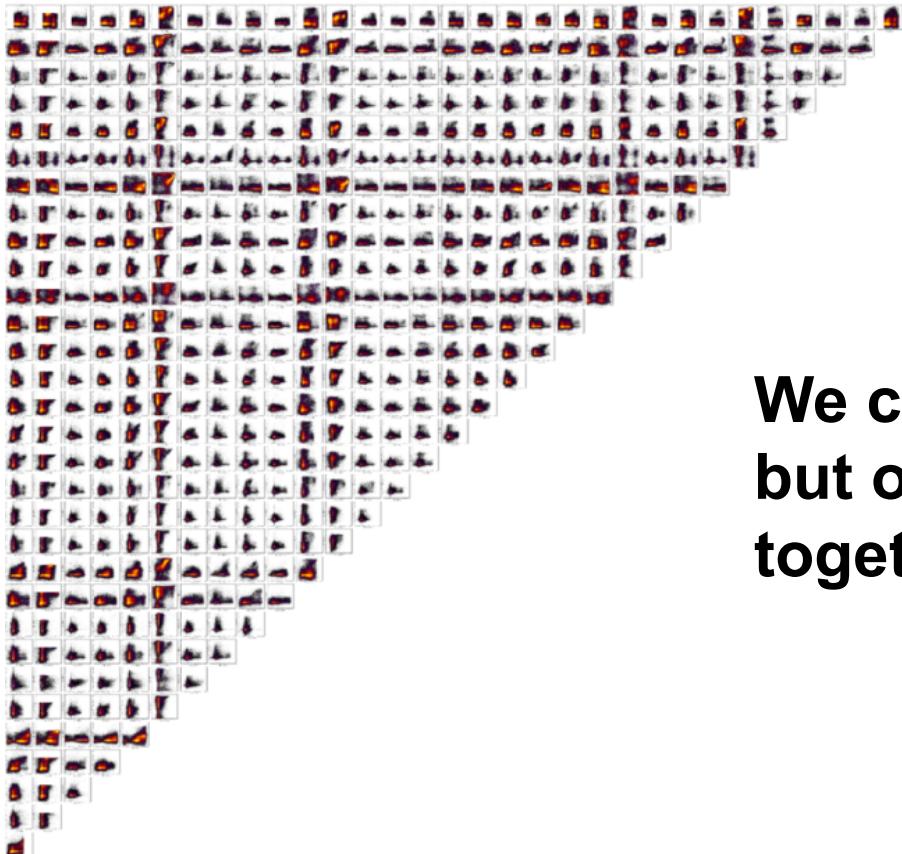


More Dimensions = More Accurate Structure

Hue and % Alcohol and Color Intensity of all wines



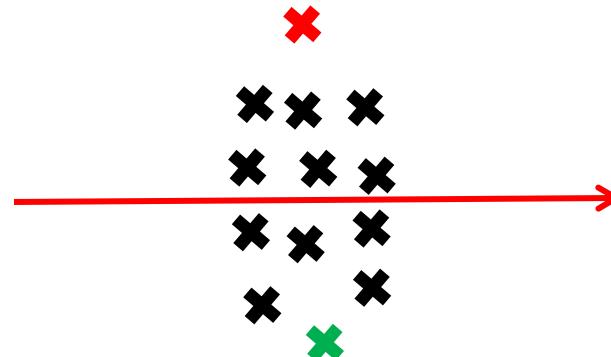
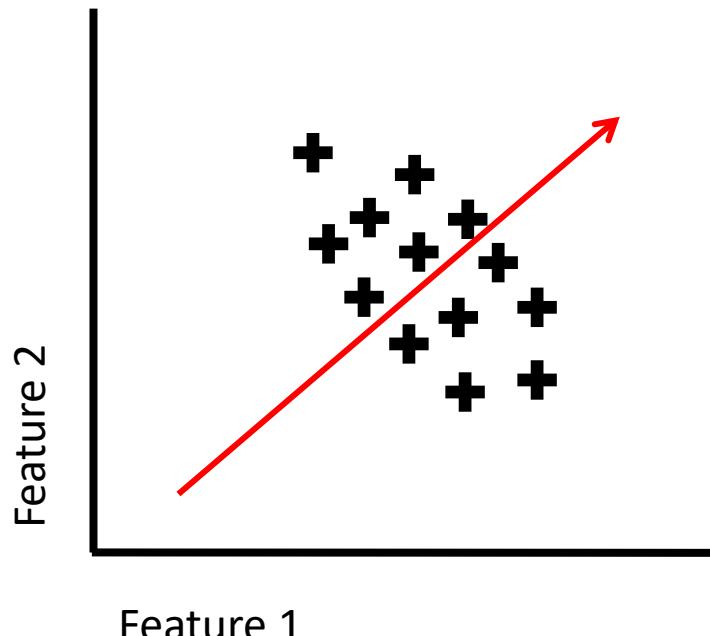
Problem: We can only see in 3D (not 20K D)



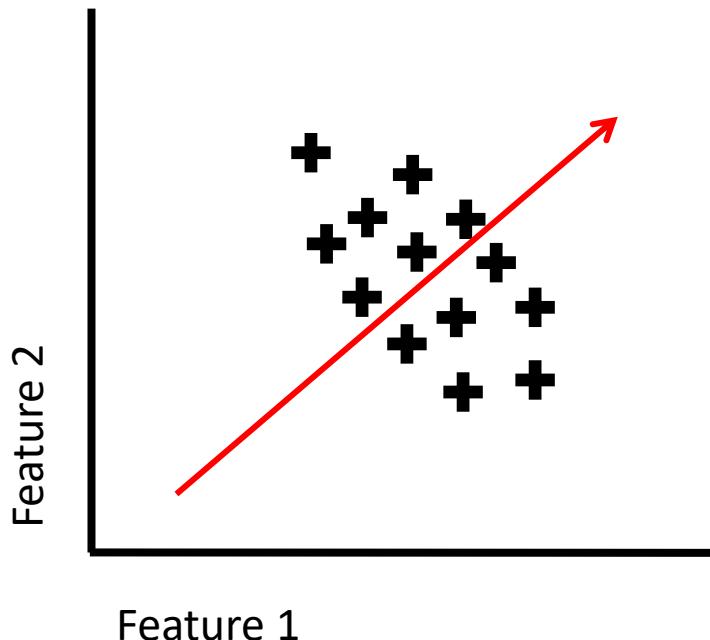
**We could try all 2D pairs,
but our brains can't put this
together**

Solution: Dimensionality Reduction

Thought Exercise: Reduce this to 1D



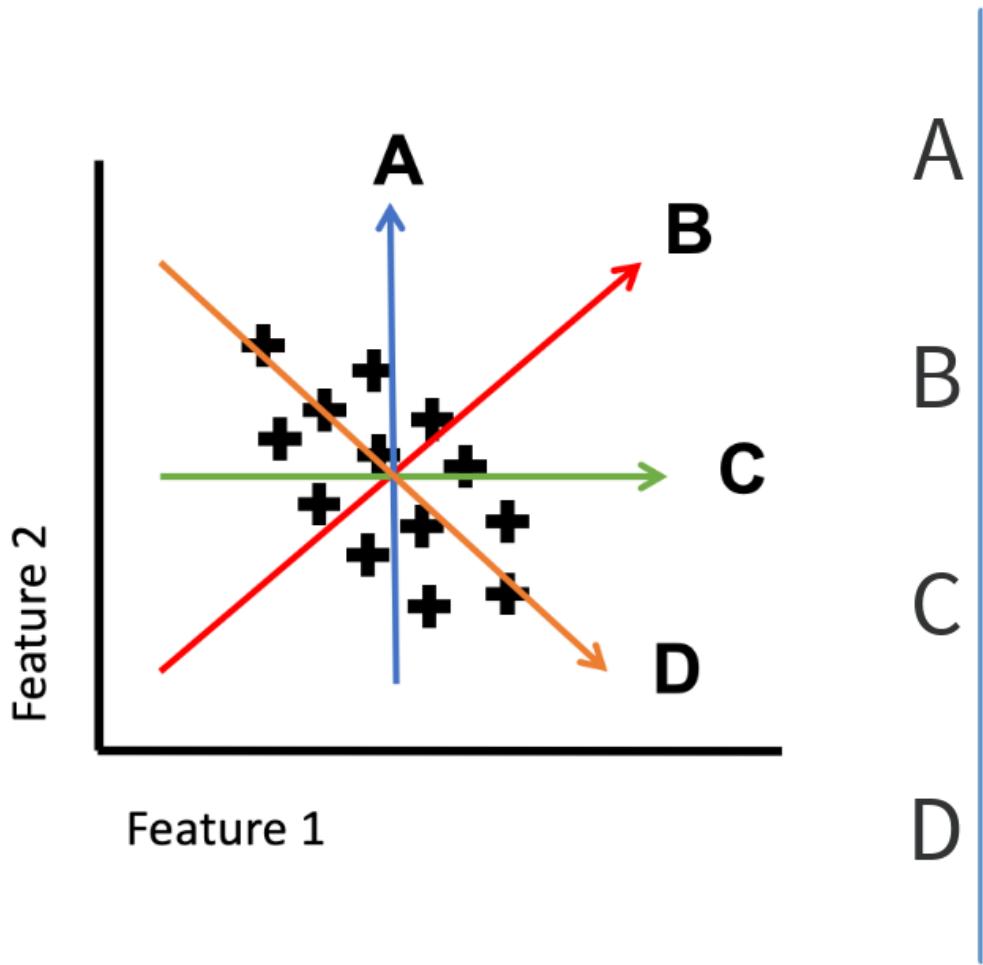
Thought Exercise: Reduce this to 1D



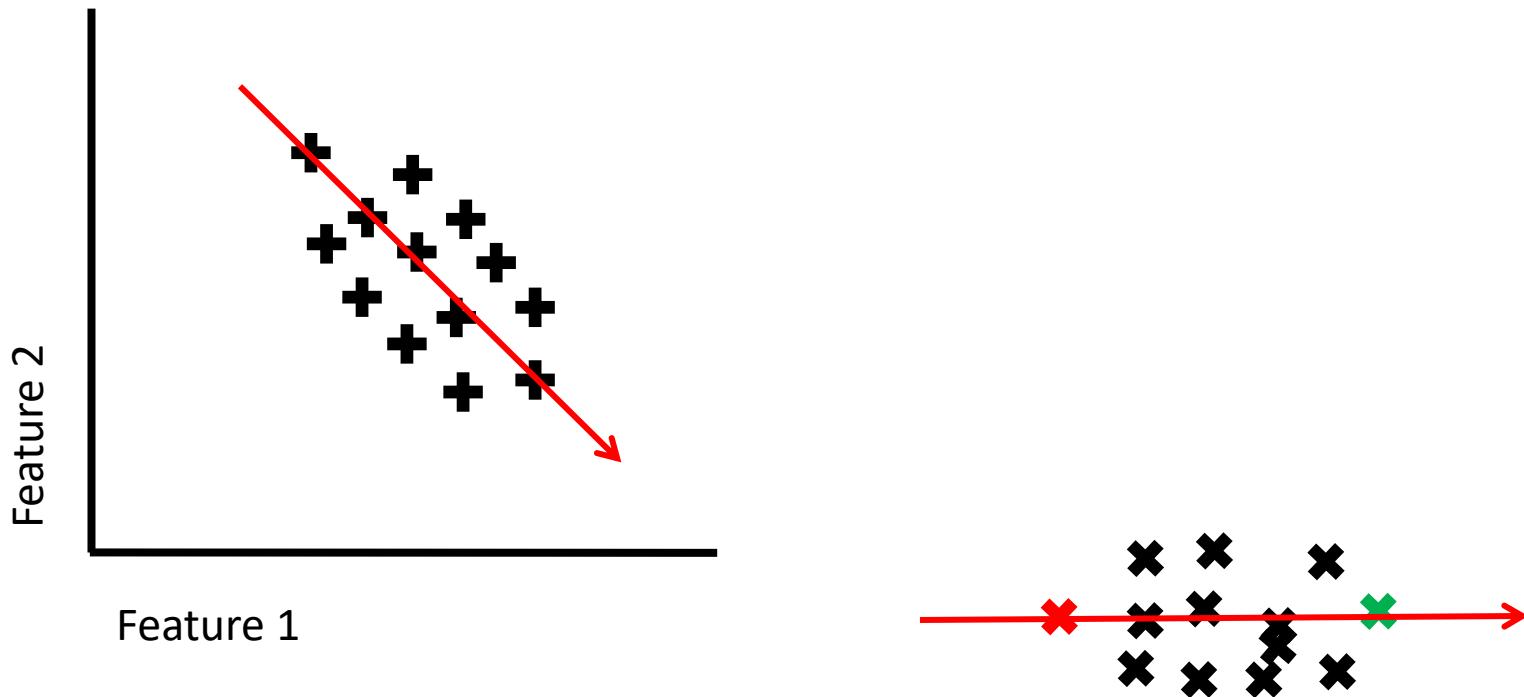
**Most distant points on
top of each other !**



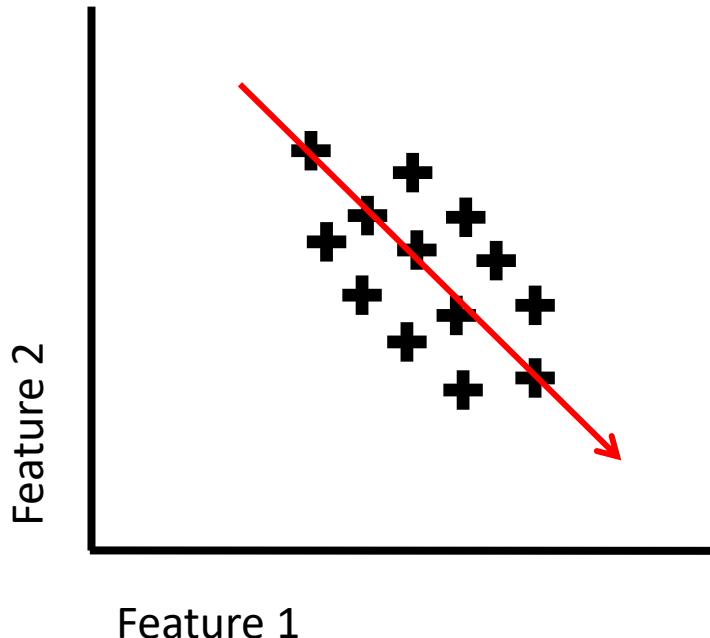
Which line do you think would be the best line to consider the data onto?



Thought Exercise: Reduce this to 1D



Thought Exercise: Reduce this to 1D

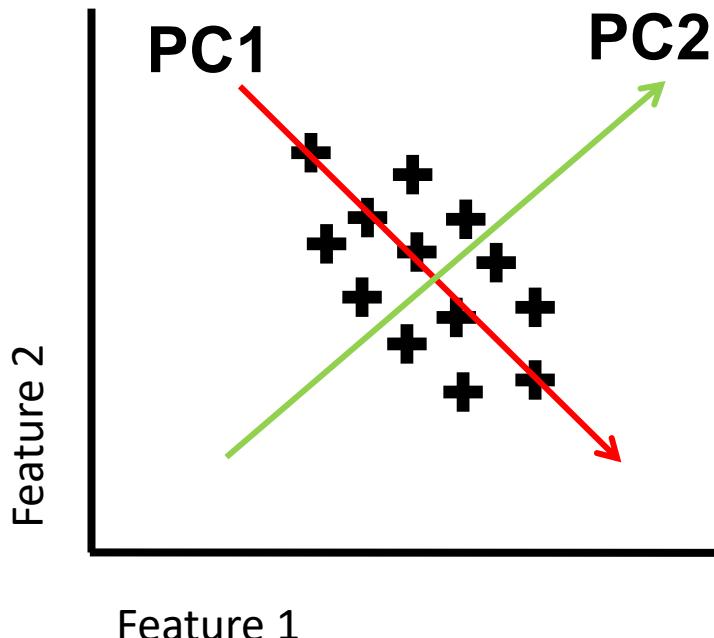


What is special about this line?

It explains most of the variance in the data



Principle Components Analysis



PCA linear directions
in the data that explain
the most variance

PC1 explains the most

PC2 explains the next most
and is orthogonal to PC1

How do we find these directions?

Covariance Matrix

$$\mathbf{X} = (X_1, X_2, \dots, X_n)^T$$

$$K_{X_i X_j} = \text{cov}[X_i, X_j] = E[(X_i - E[X_i])(X_j - E[X_j])]$$

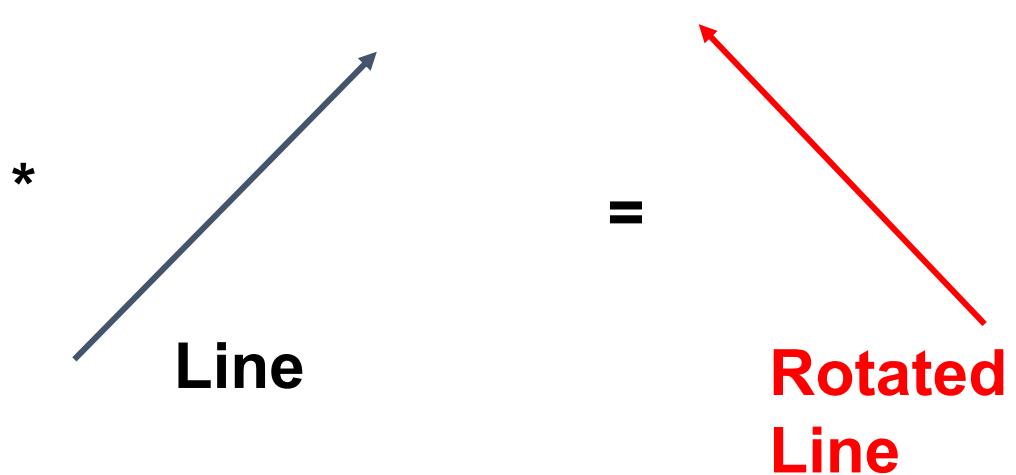
$$K_{\mathbf{X}\mathbf{X}} = \begin{bmatrix} E[(X_1 - E[X_1])(X_1 - E[X_1])] & E[(X_1 - E[X_1])(X_2 - E[X_2])] & \cdots & E[(X_1 - E[X_1])(X_n - E[X_n])] \\ E[(X_2 - E[X_2])(X_1 - E[X_1])] & E[(X_2 - E[X_2])(X_2 - E[X_2])] & \cdots & E[(X_2 - E[X_2])(X_n - E[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n - E[X_n])(X_1 - E[X_1])] & E[(X_n - E[X_n])(X_2 - E[X_2])] & \cdots & E[(X_n - E[X_n])(X_n - E[X_n])] \end{bmatrix}$$

Matrices as Transformations

- Matrices only store data, but can also represent ***transformations***
- Specifically they are linear transformations
- They transform lines by changing their length and angle from origin

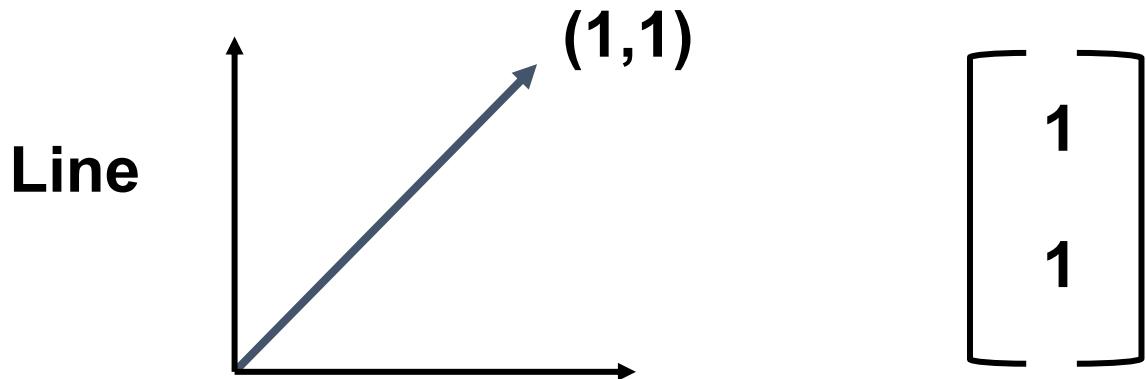
$$\begin{bmatrix} \cos(90) & -\sin(90) \\ \sin(90) & \cos(90) \end{bmatrix}$$

Rotation matrix



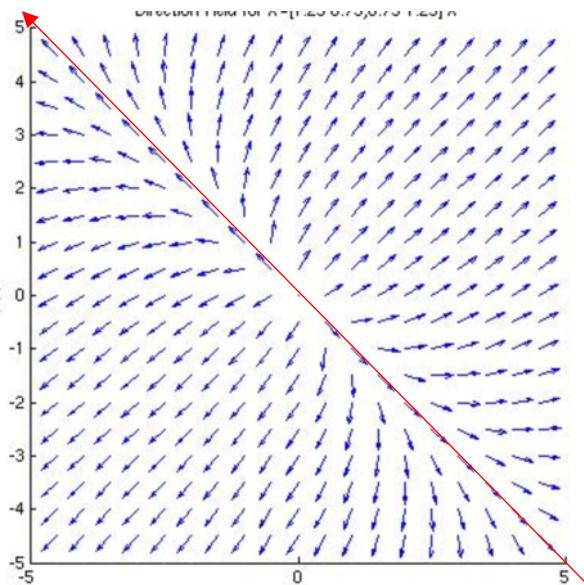
Matrix–Vector Notation

- Lines can be described as vectors from the origin



$$\begin{bmatrix} \cos(90) & -\sin(90) \\ \sin(90) & \cos(90) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Eigenvectors



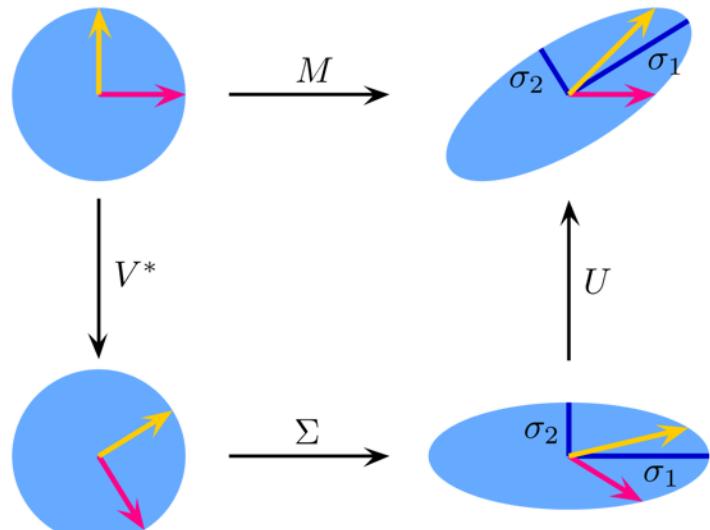
Rotation matrices rotate lines in only one direction

Other matrices can move different lines in different directions

- Such linear transformations have fixed points called *Eigenvectors*
- In other words, the transformation only **stretches** the vector and does not rotate it.

Eigendecomposition

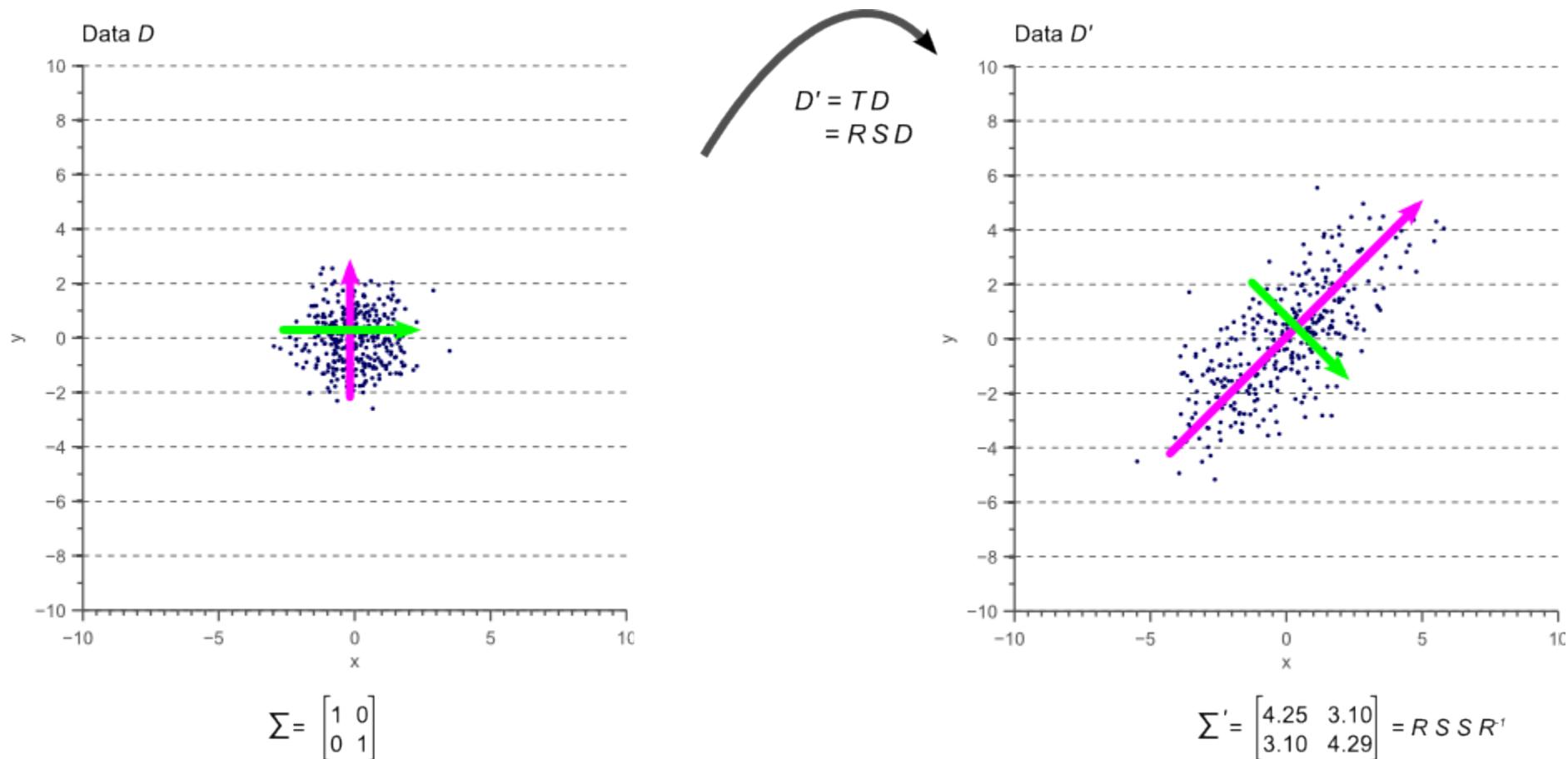
- Eigenvectors can be used to decompose a linear transformation into a rotation + stretch + anti-rotation



Non-square matrices have similar OP called **Singular value decomposition**

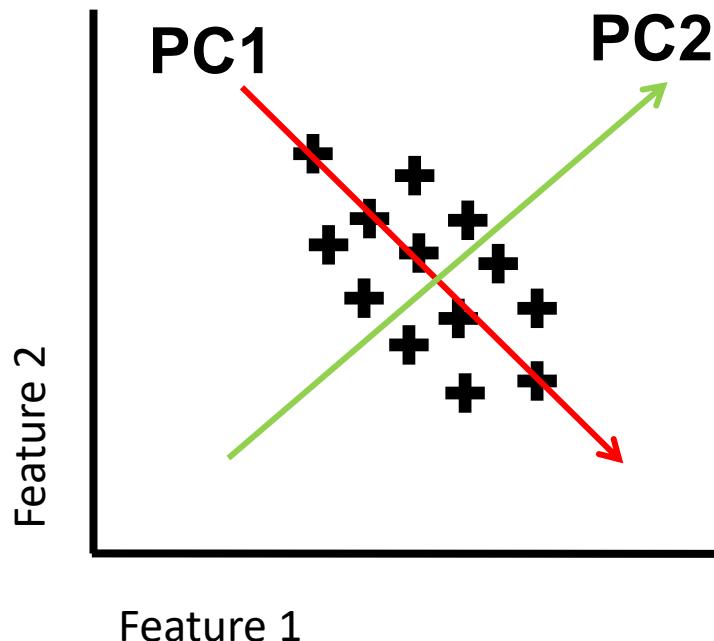
$$M = U \cdot \Sigma \cdot V^*$$

Eigenvectors of Covariance



Matrix creates correlation structure of data on unit blob

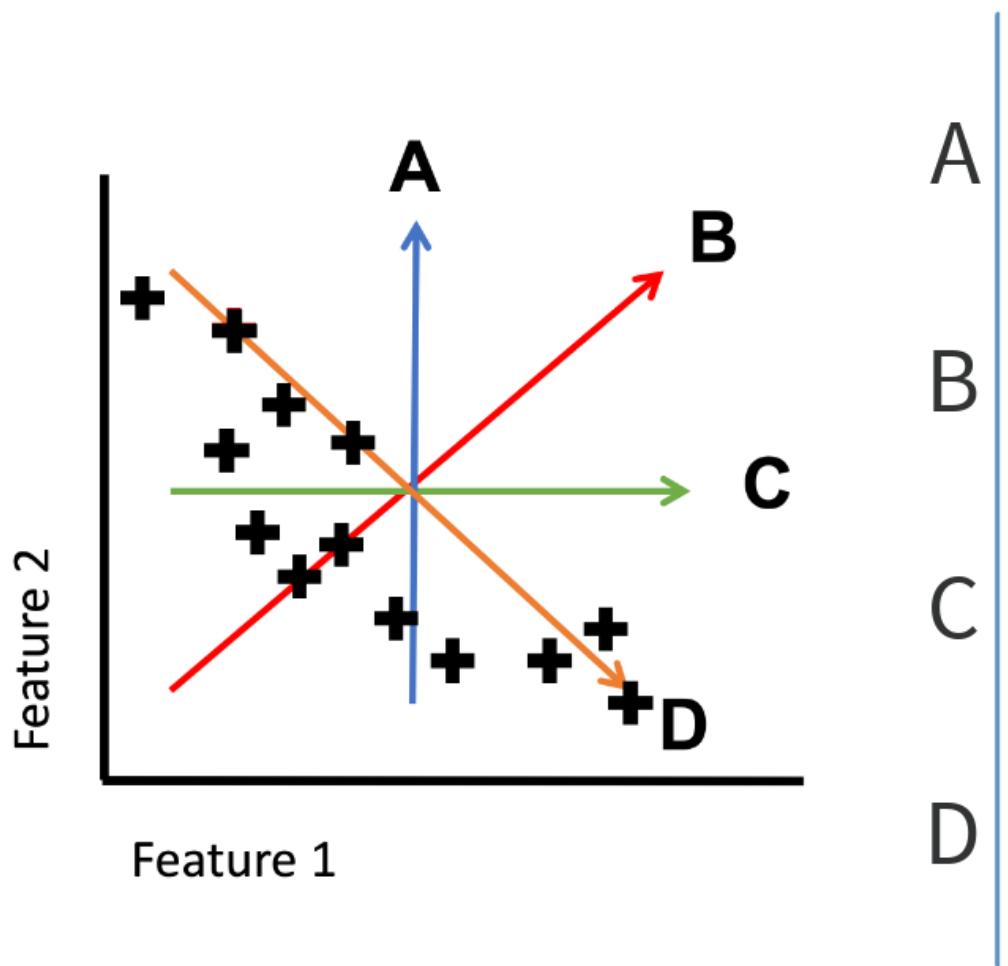
Principle Components Analysis



PC1 is first eigenvector (with
Highest eigenvalue)

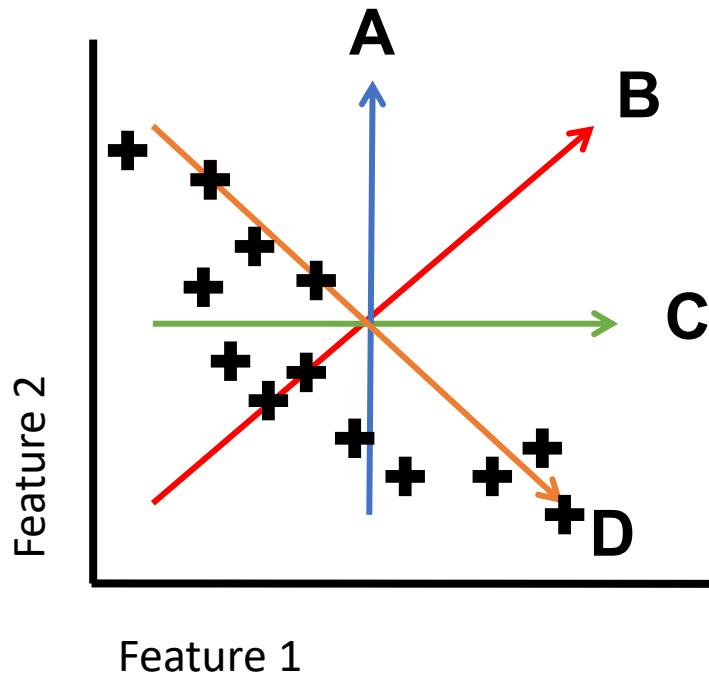
PC2 is next eigenvector

Which line would be PC1 for the following data?

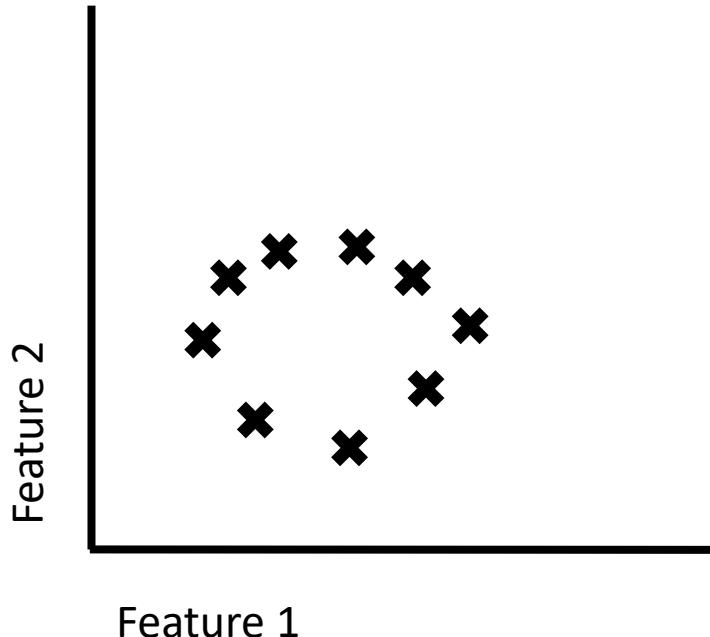


Quick quiz

Which line would be PC1 for the following data?



Non-linear structure



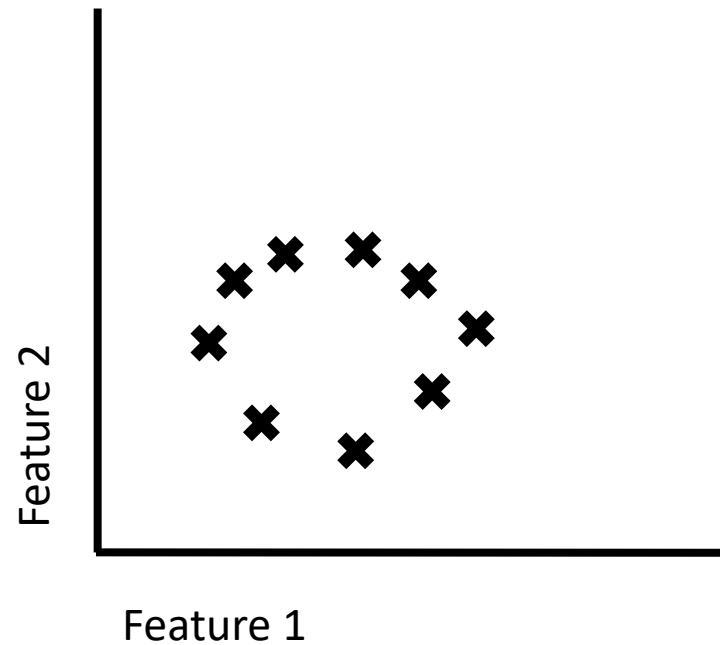
What if the data looked like this?

What line would be good
to project to here?

Non-linear dimensionality reduction

Non-linear dimensions

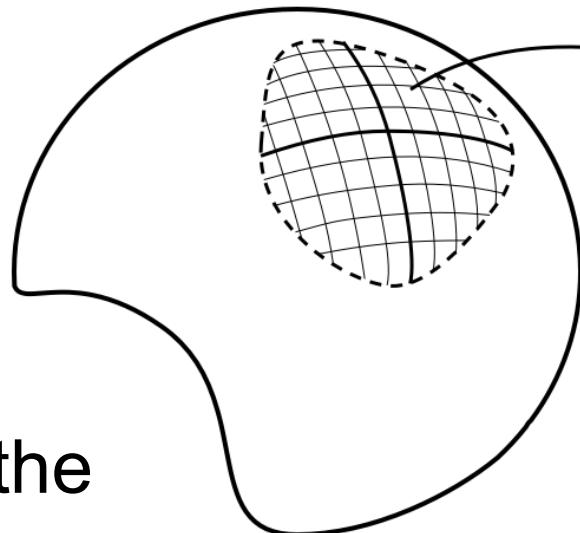
- To find non-linear or “coiled” dimensions in a dataset we have to understand, and describe the **shape** of the data
- The data can live in lower dimensions
- This can be abstractly called the data “manifold”



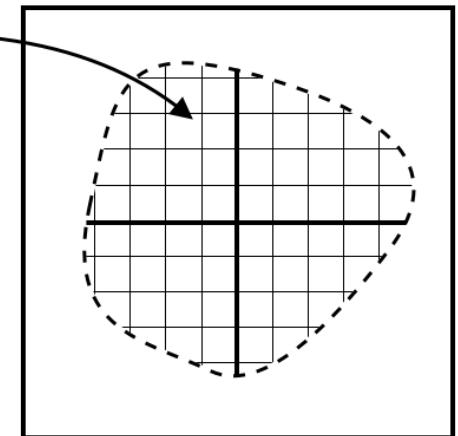
What is a manifold?

- Locally smooth
- Locally Euclidean
- Generally, lower dimensional than the ambient space (i.e. a subspace)

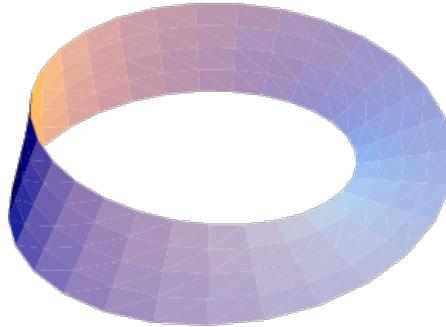
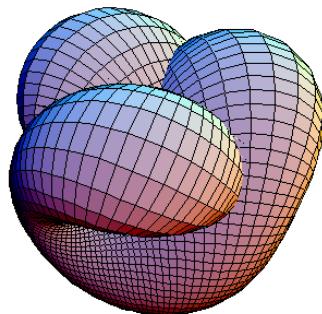
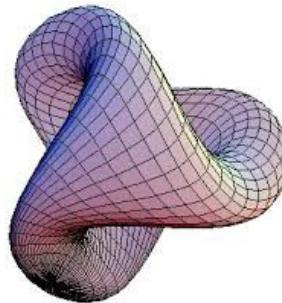
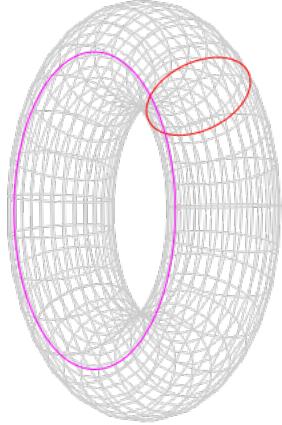
Surface in \mathbb{R}^3



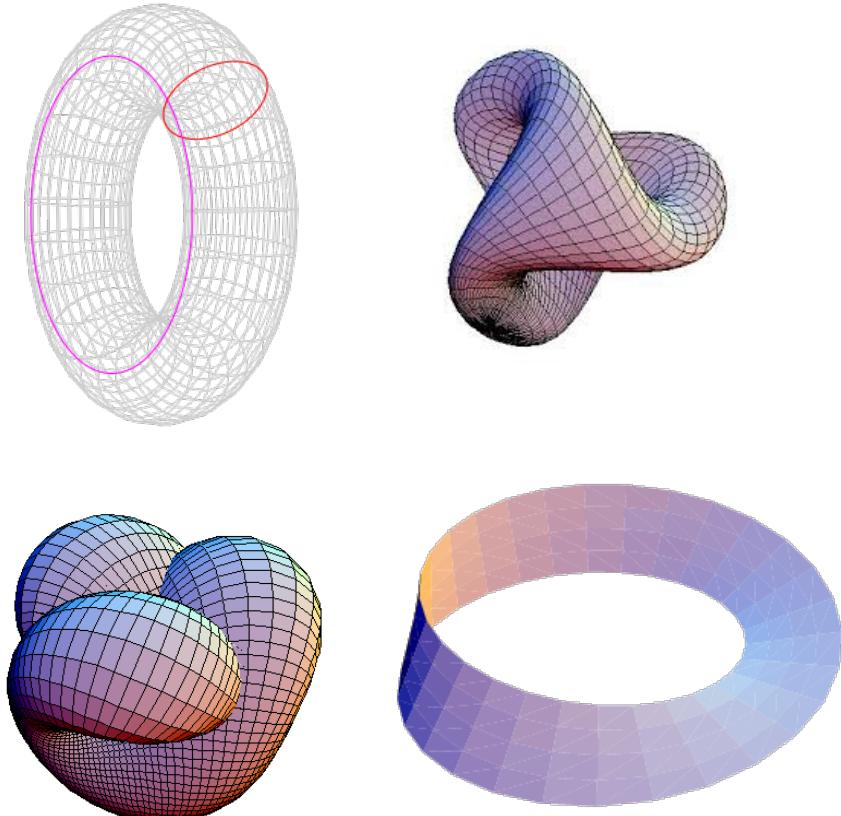
Local view in \mathbb{R}^2



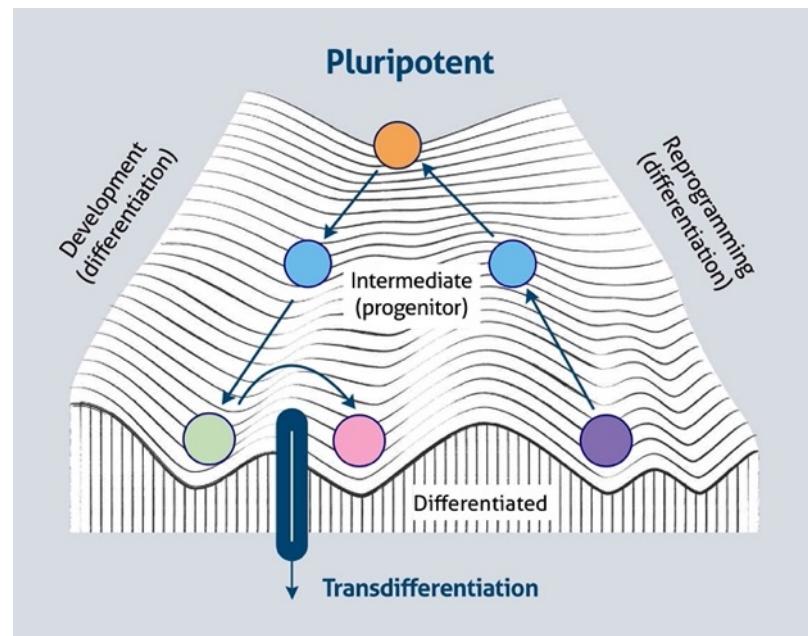
Examples of manifolds



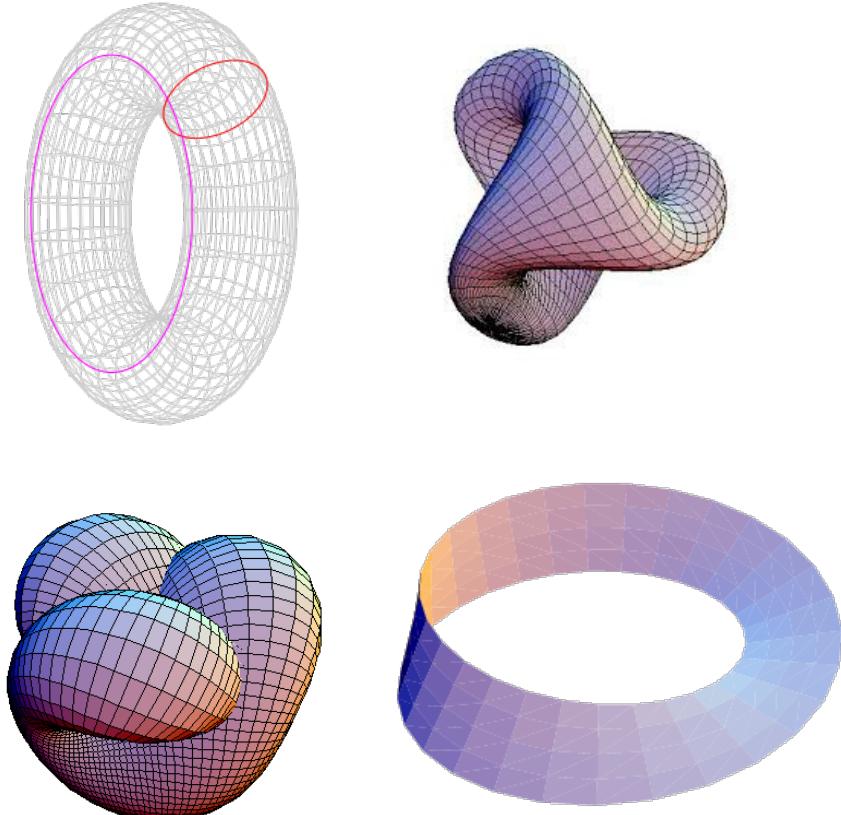
Examples of manifolds



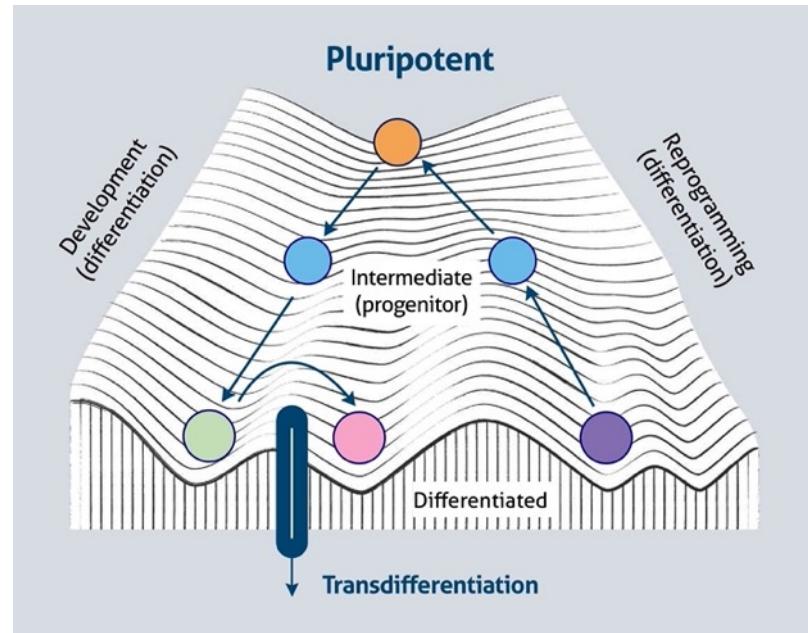
Waddington's landscape



Examples of manifolds

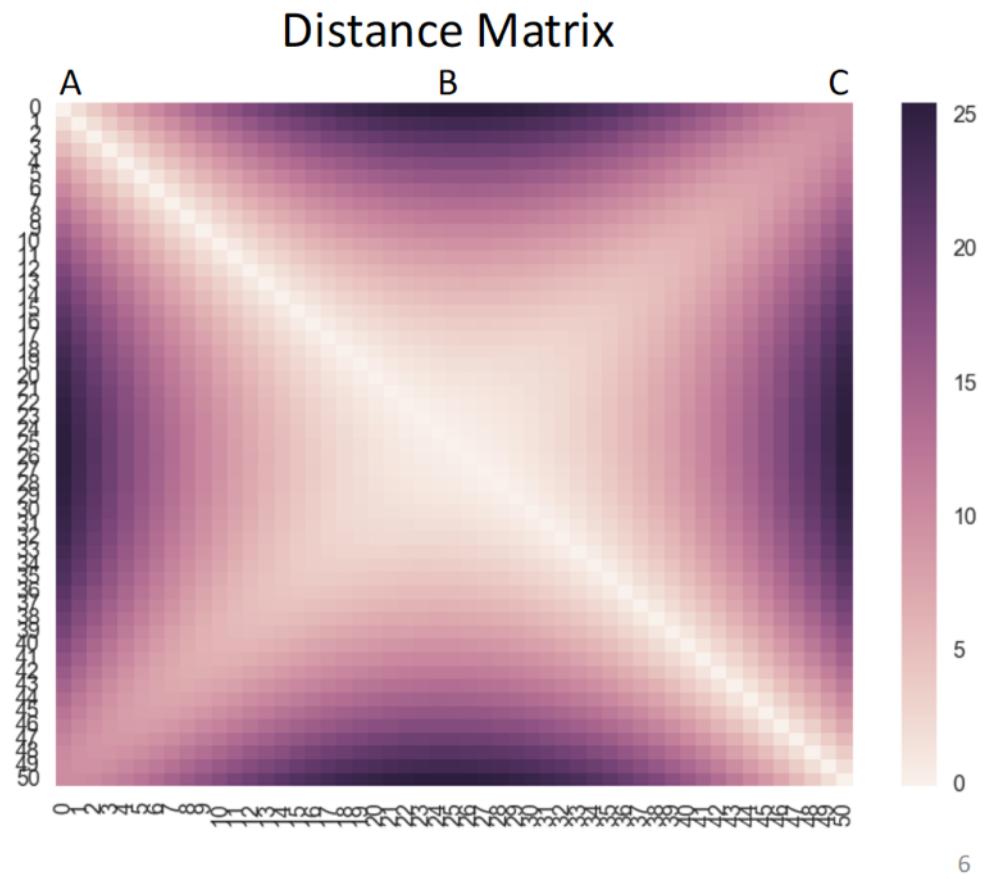
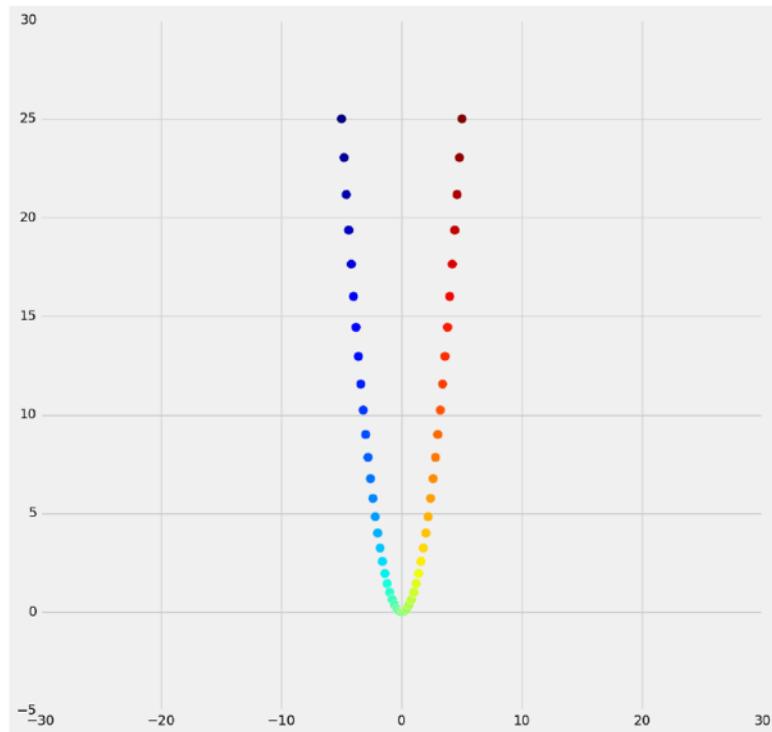


Waddington's landscape

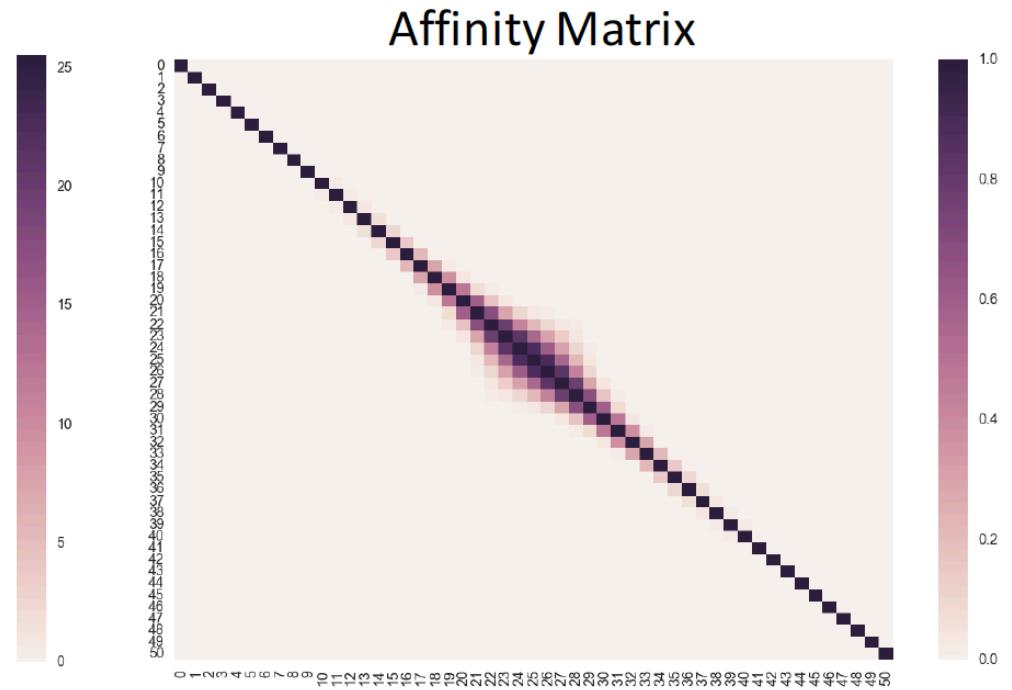
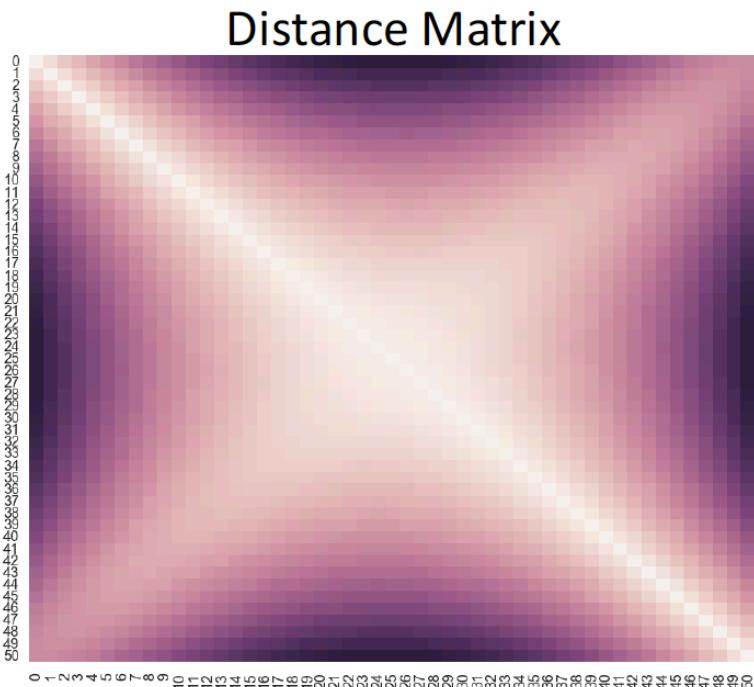
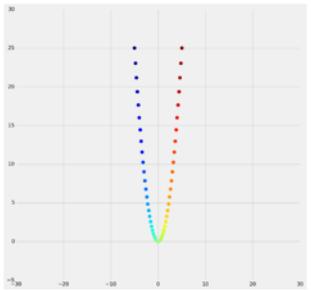


- Smooth transitions between states
- Small changes in gene expression are linear

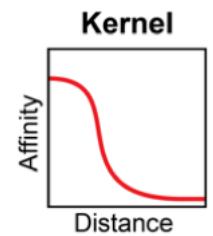
Kernel functions measure “similarity” or “affinity” on a graph



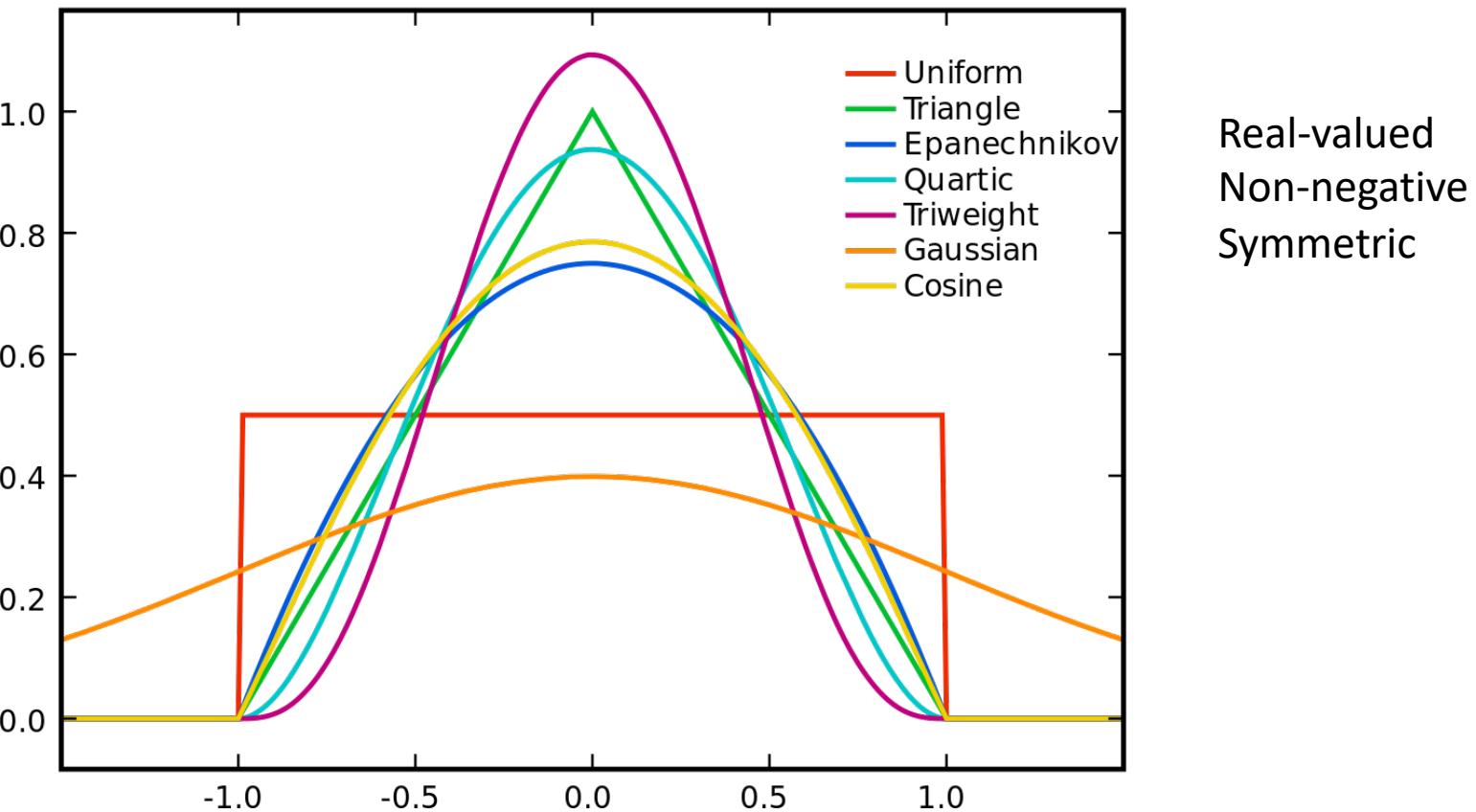
Affinity is the inverse of distance + locality



$$Affinity_{i,j} = s_{i,j} = \exp\left(-\frac{dist(x_i, x_j)^2}{2\sigma^2}\right)$$

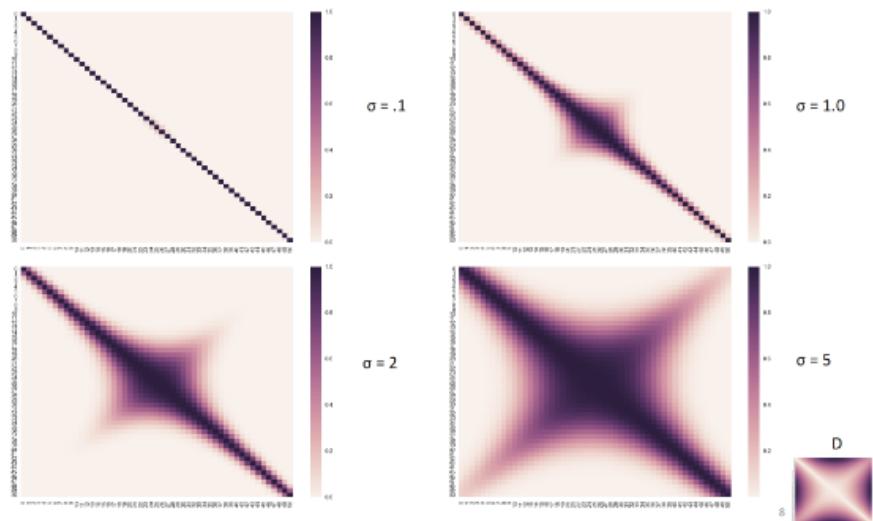


Distance to Affinity via Kernels



Affinities correlations in this hidden hypothetical space

Which value of sigma produces the graph with the least global connectivity



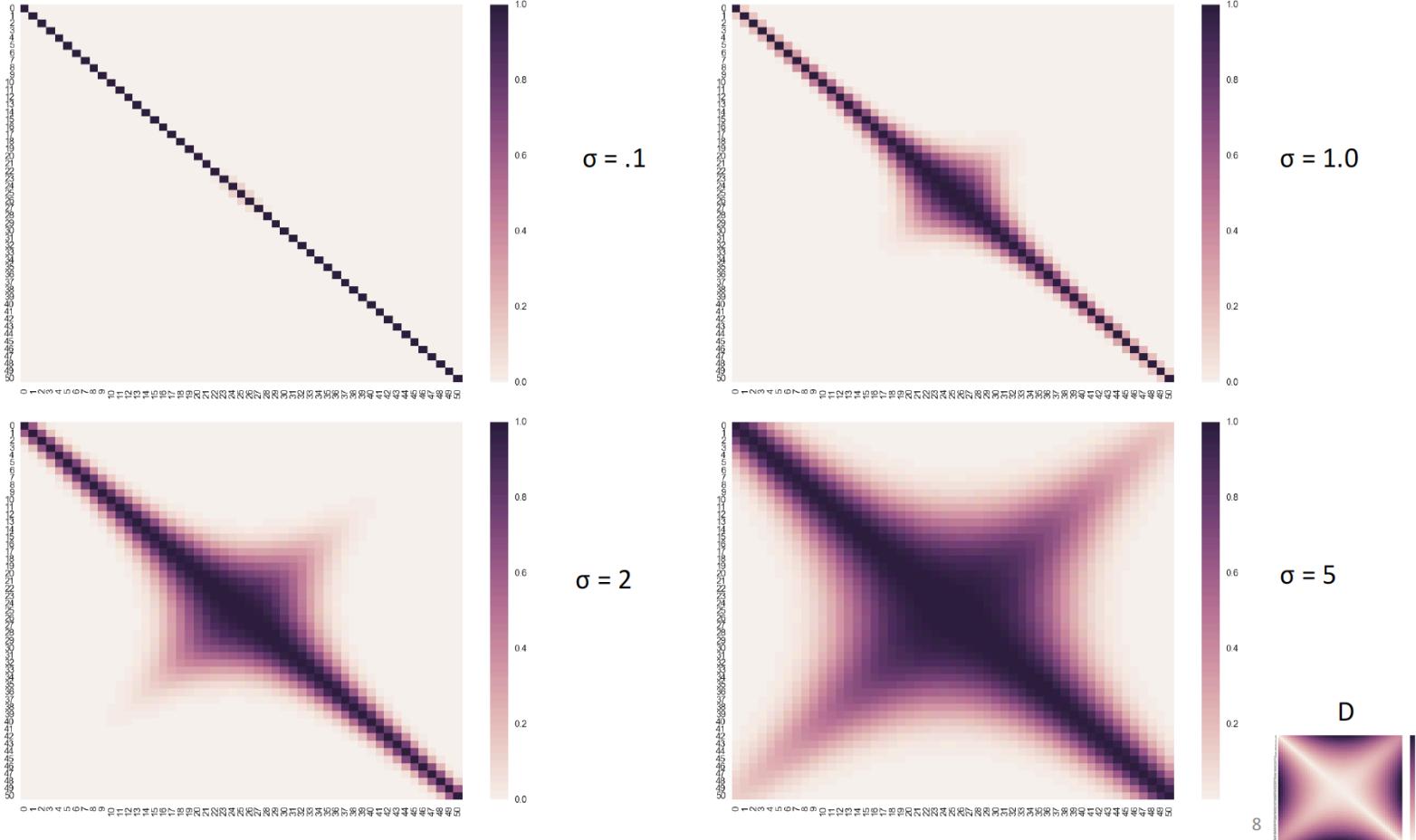
$\sigma = 0.1$

$\sigma = 1$

$\sigma = 2$

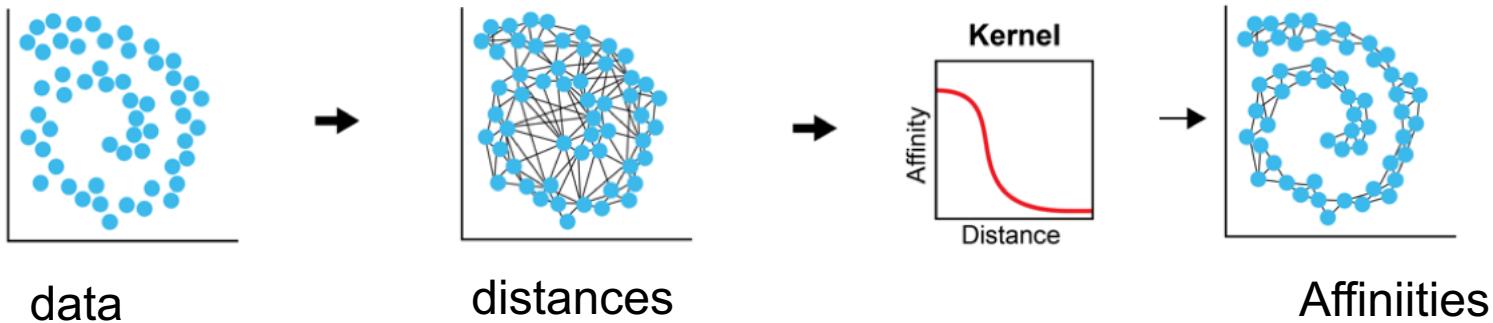
$\sigma = 5$

Effect of Changing Kernel Width



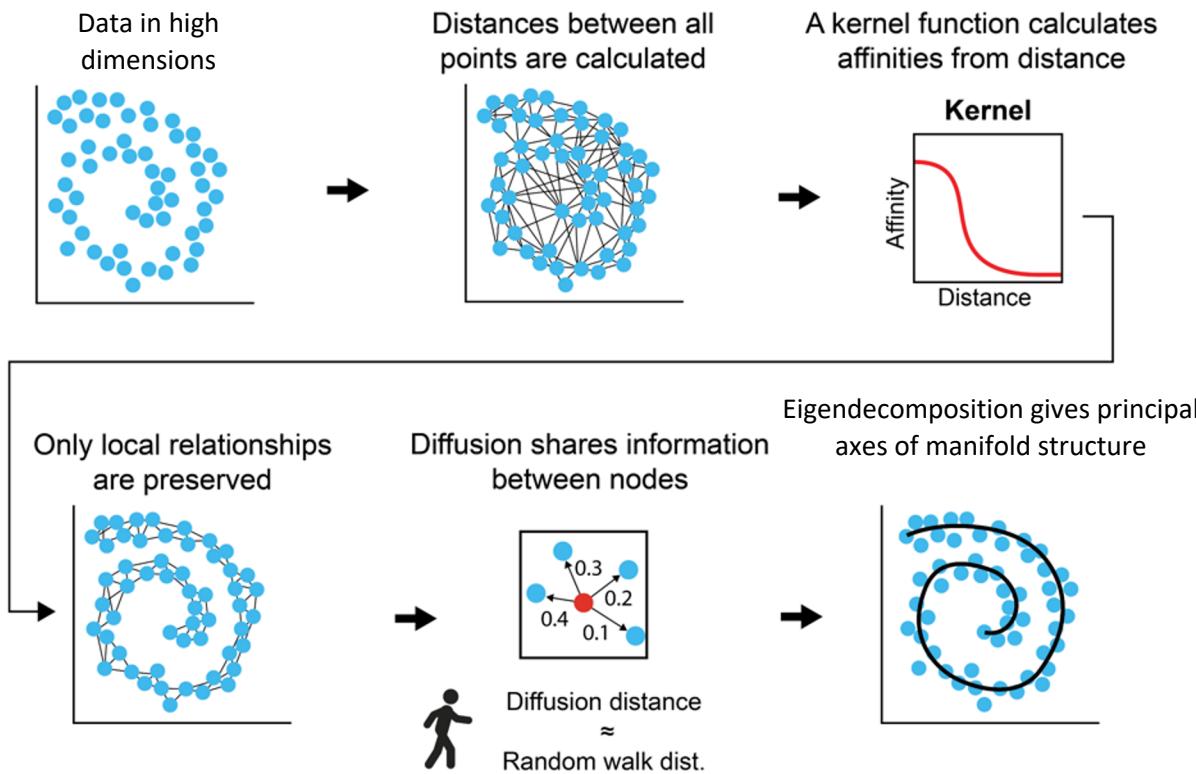
$$s_{i,j} = \exp\left(-\frac{\text{dist}(x_i, x_j)^2}{2\sigma^2}\right)$$

Kernel PCA



- Use eigenvectors of an **affinity matrix** instead of covariance matrix
- The family of methods called kernel PCA:
- Specific variants include:
 - Laplacian Eigenmaps
 - Diffusion Maps

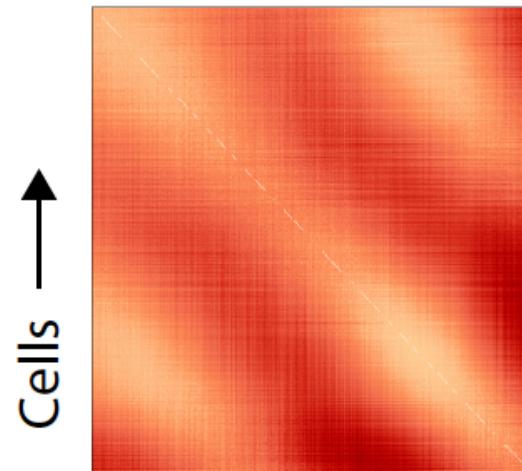
Diffusion Maps



Distance Matrix



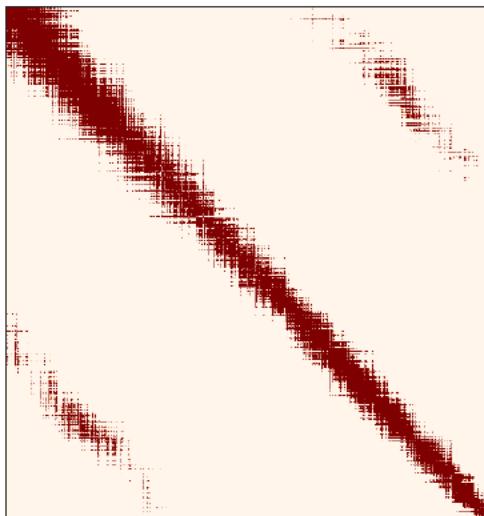
Distance Matrix



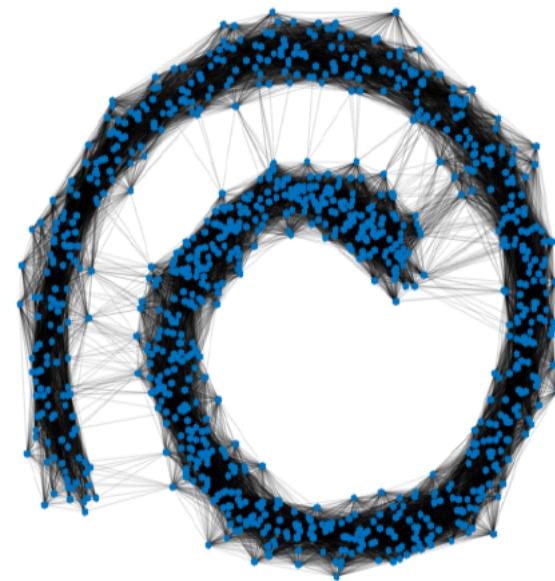
Entries are distances

$$D(x_i, x_j) = \sqrt{||x_i - x_j||}$$

Affinity Matrix



(Gaussian Kernel)



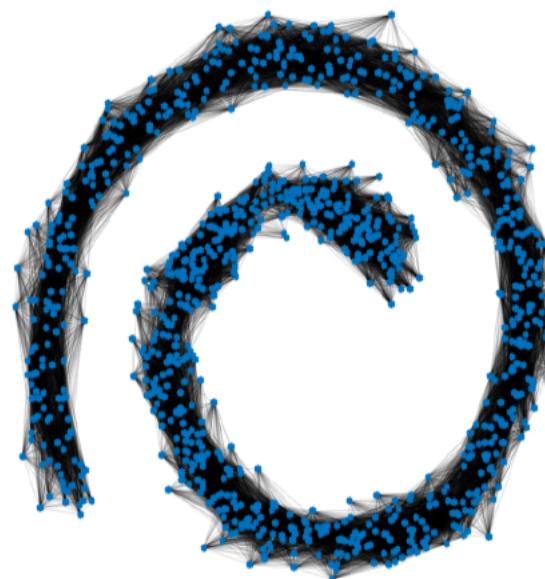
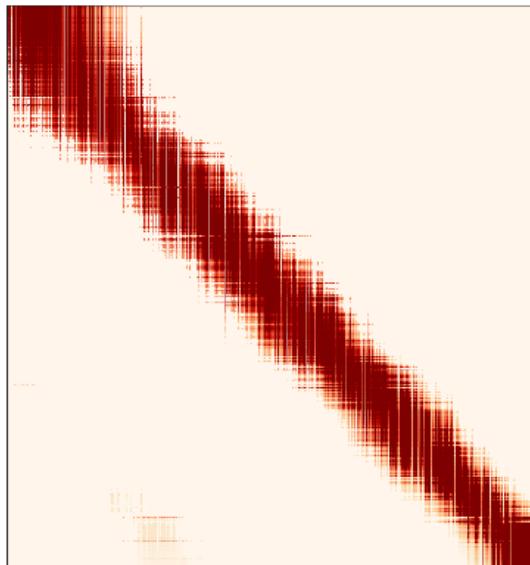
Graph Representation

Entries are affinities

$$A(x_i, x_j) = \exp\left(-\frac{D(x_i, x_j)^2}{\sigma}\right)$$

Powered Markov Matrix

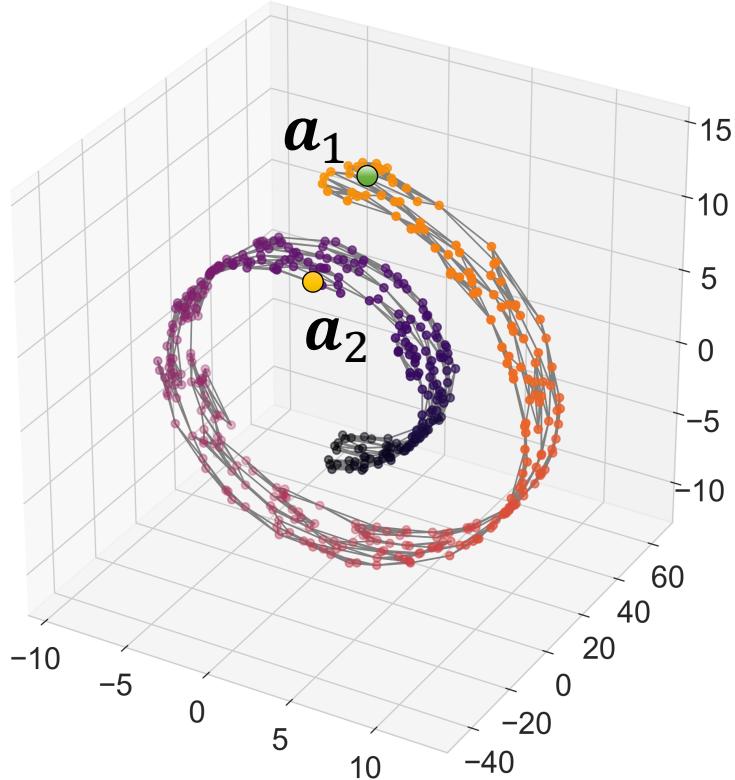
Powered Markov Matrix



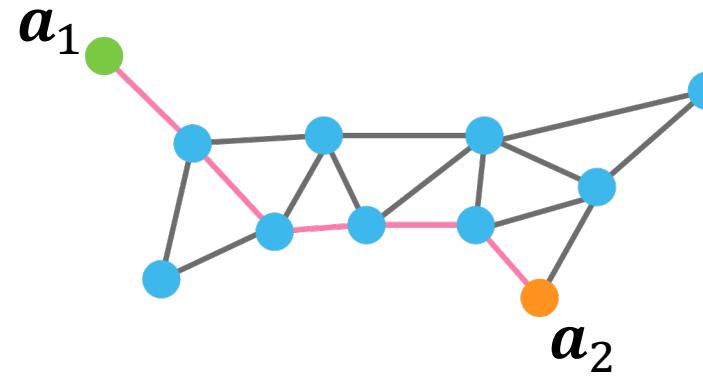
Entries are row normalized affinities

$$M(x_i, x_j) = \frac{A(x_i, x_j)}{\sum_j A(x_i, x_j)}$$

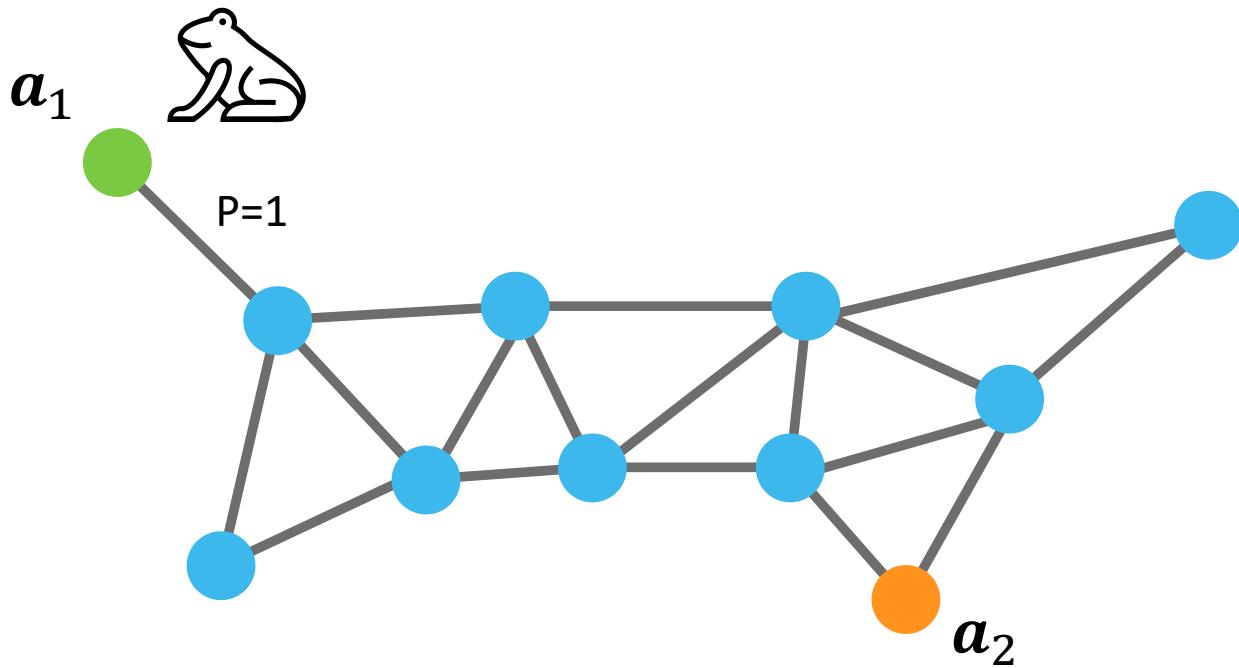
Powering or “walks” on data graphs reveal dominant data manifold directions



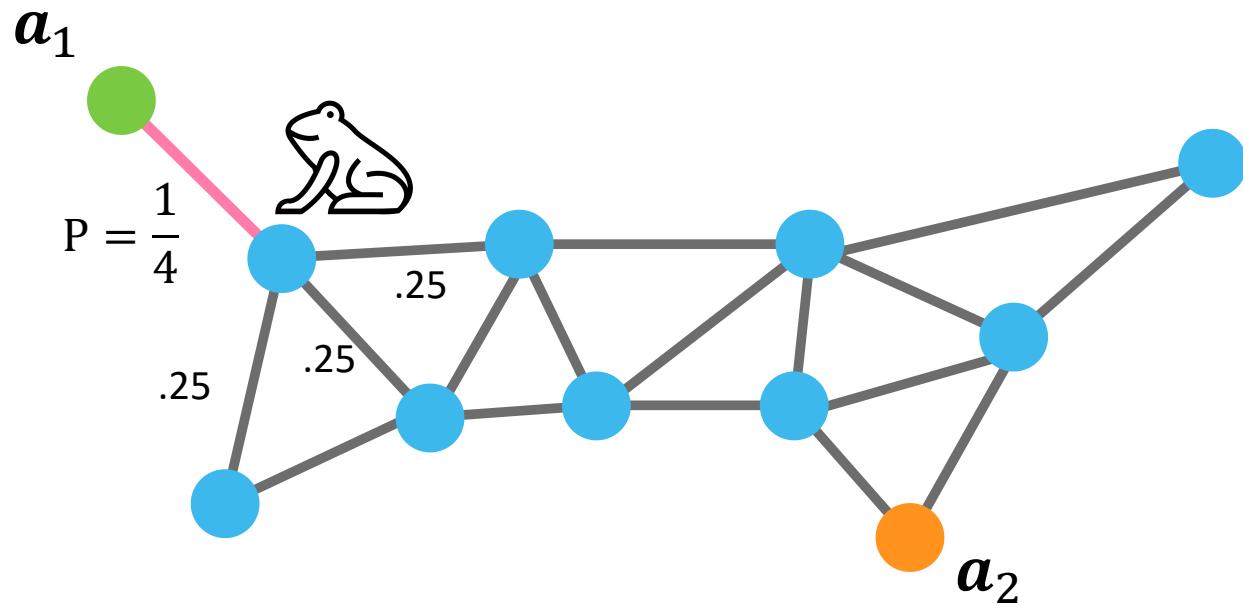
Shortest path between points



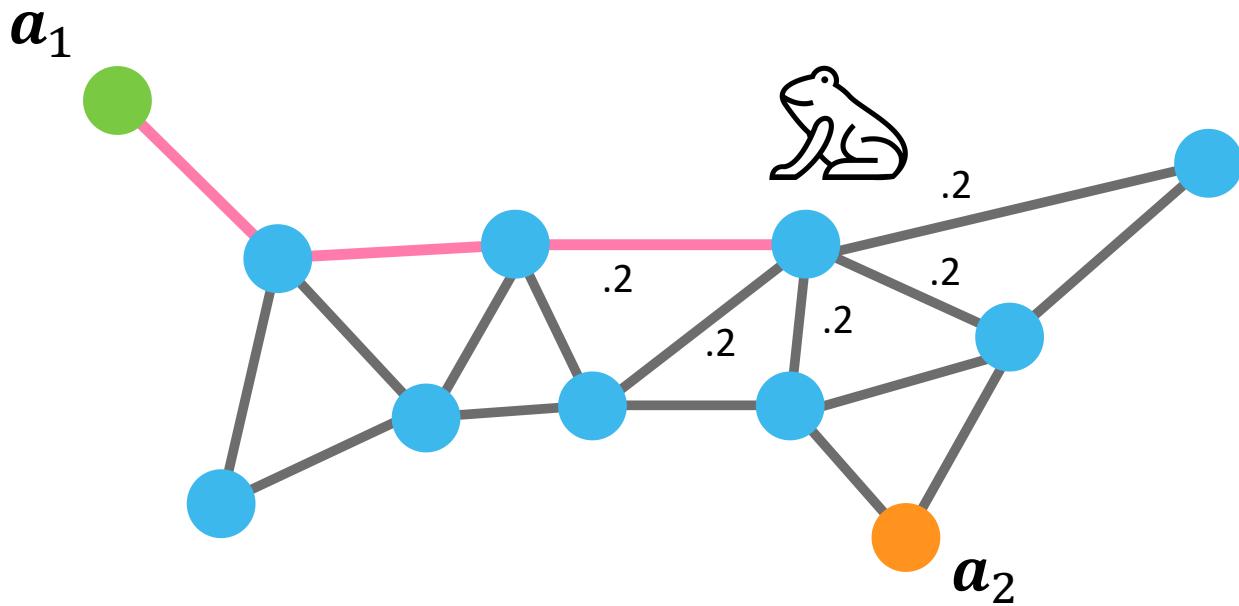
Paths or “walks” on graphs reveal dominant data manifold directions



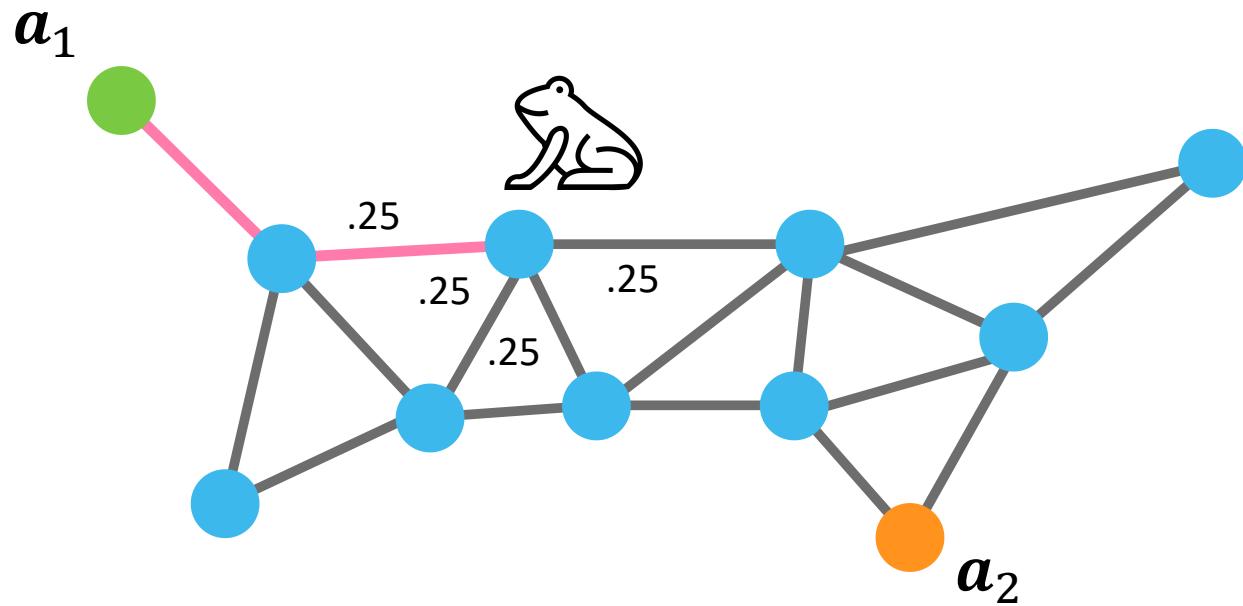
Paths or “walks” on graphs reveal dominant data manifold directions



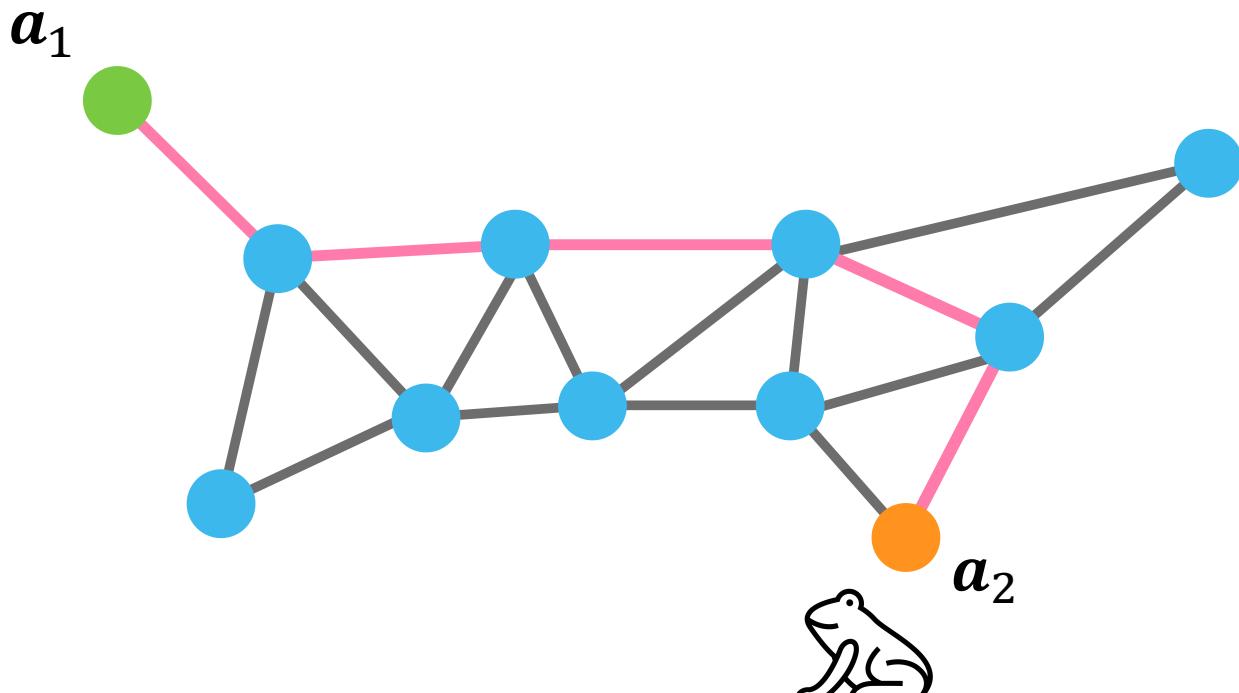
Paths or “walks” on graphs reveal dominant data manifold directions



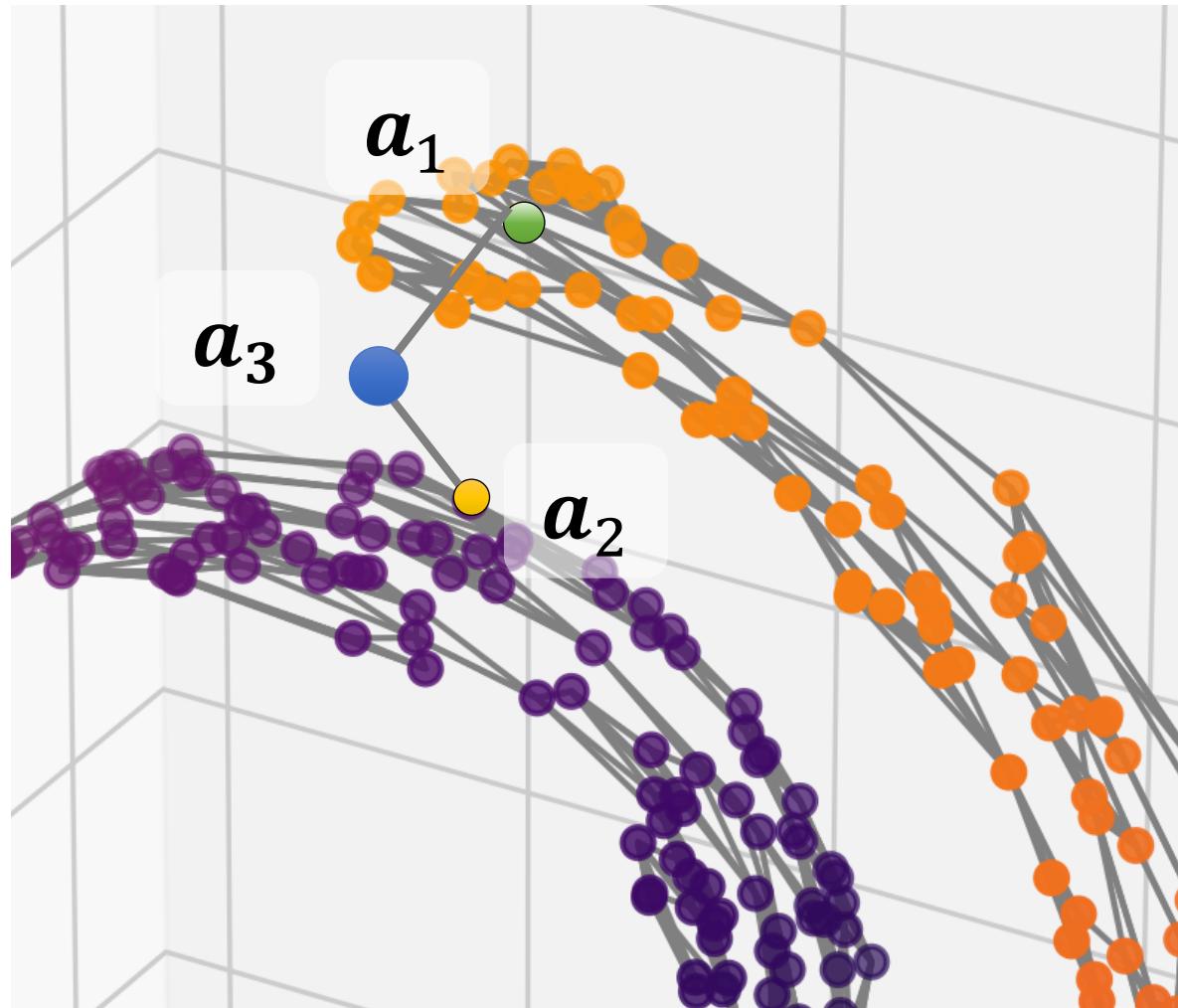
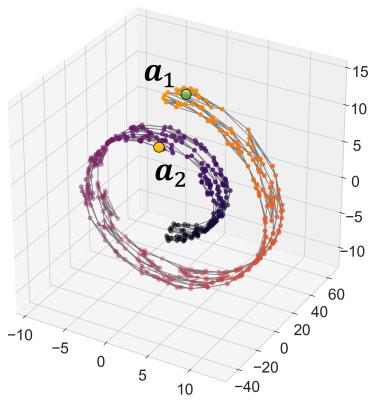
Paths or “walks” on graphs reveal dominant data manifold directions

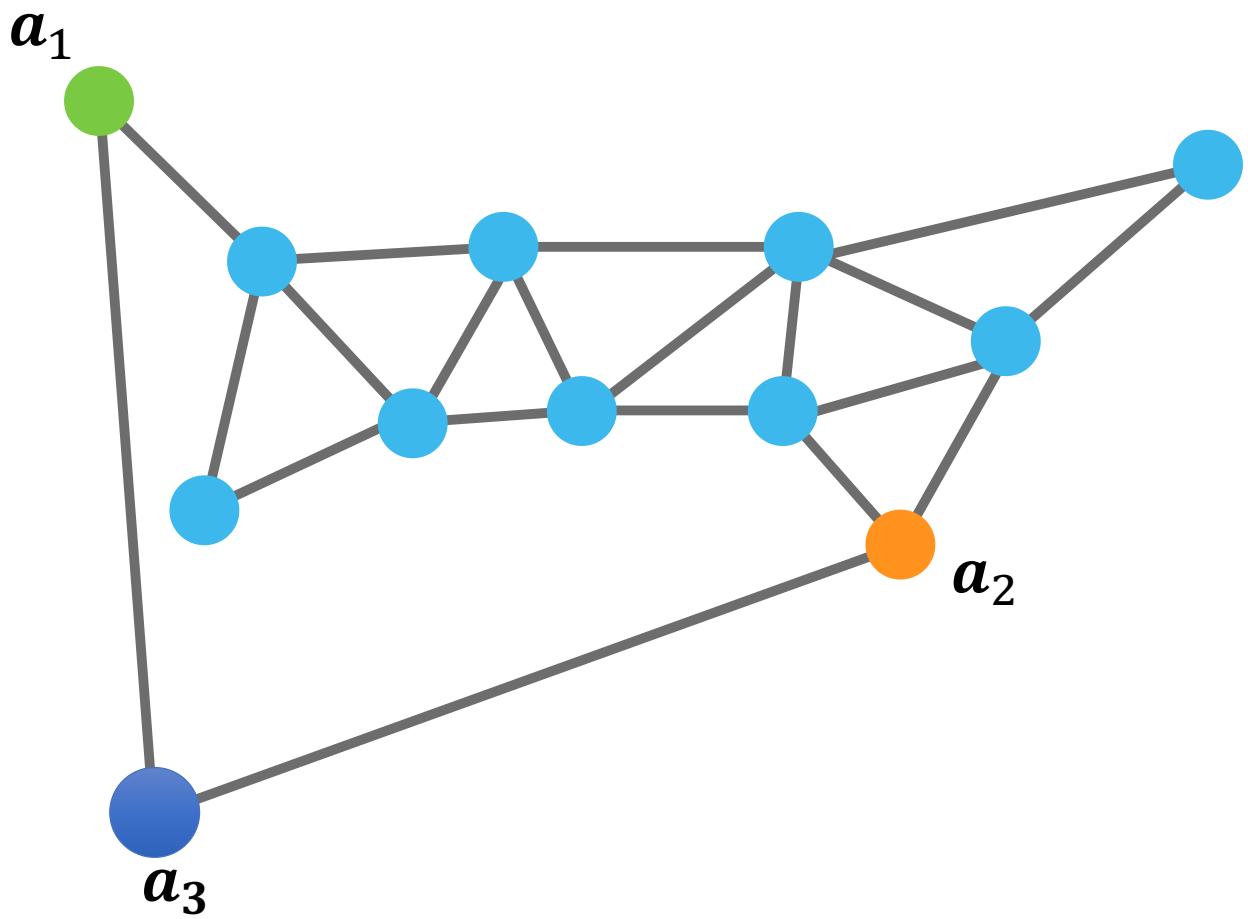
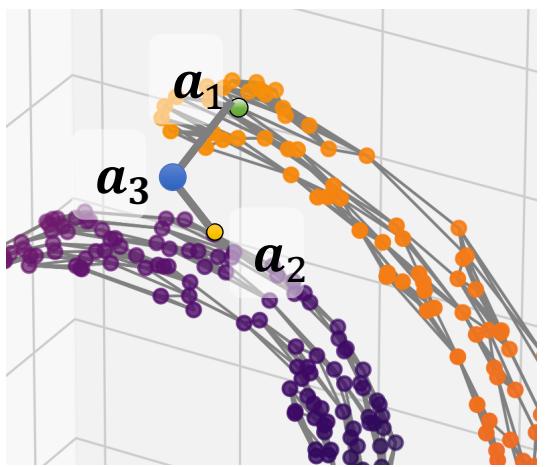


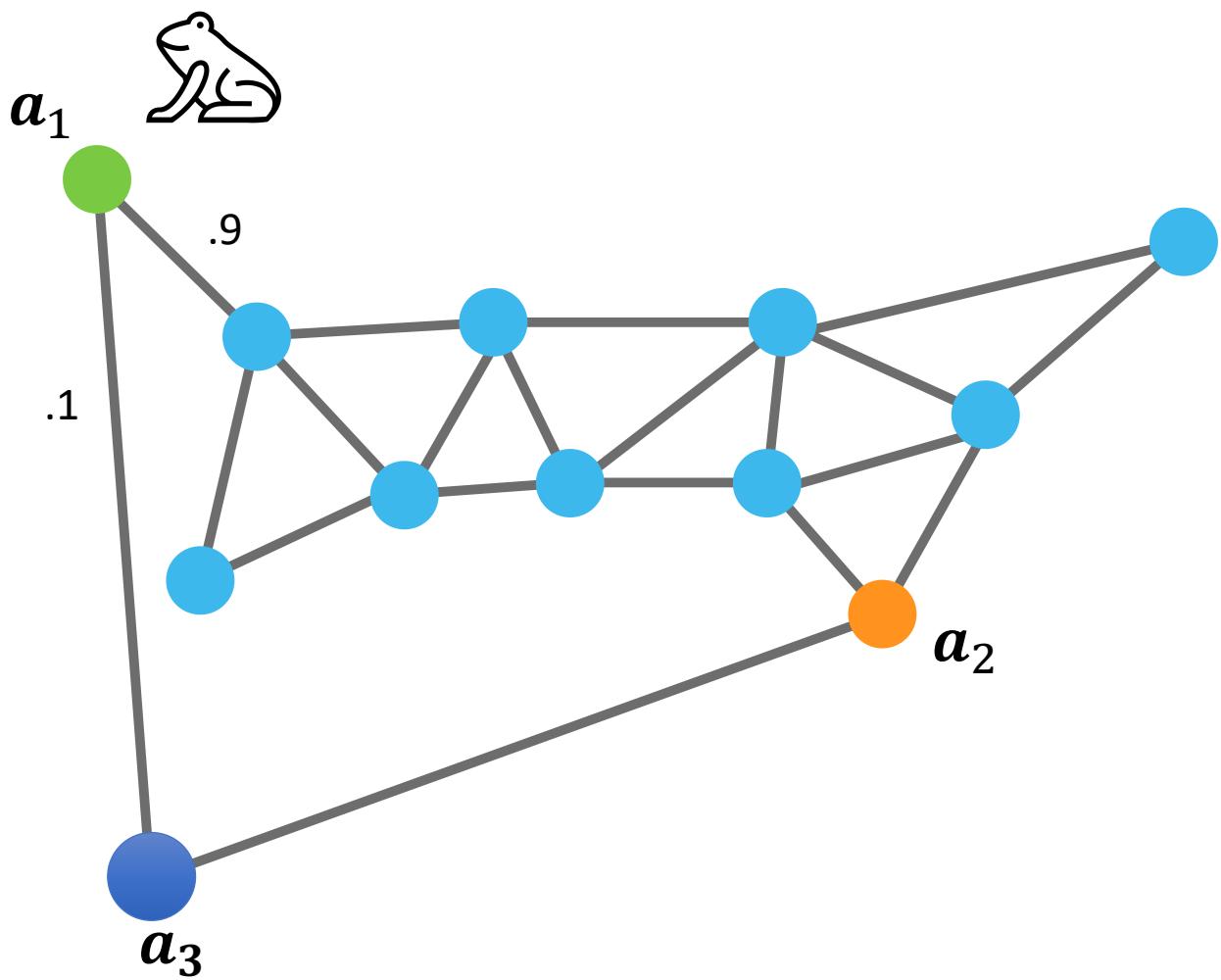
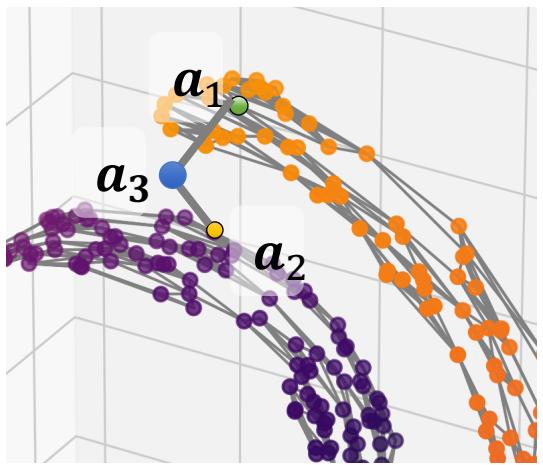
Paths or “walks” on graphs reveal dominant data manifold directions



Steps = 5

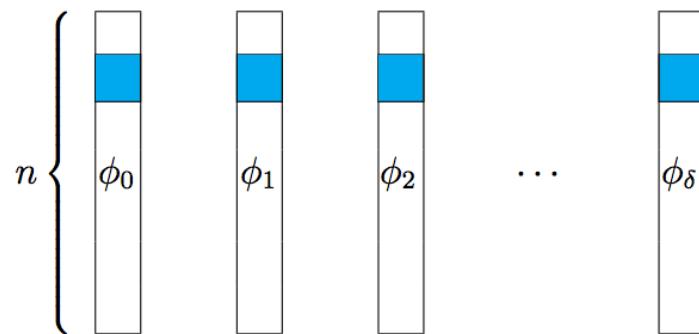






Eigendecomposition of Diffusion Operator = DMs

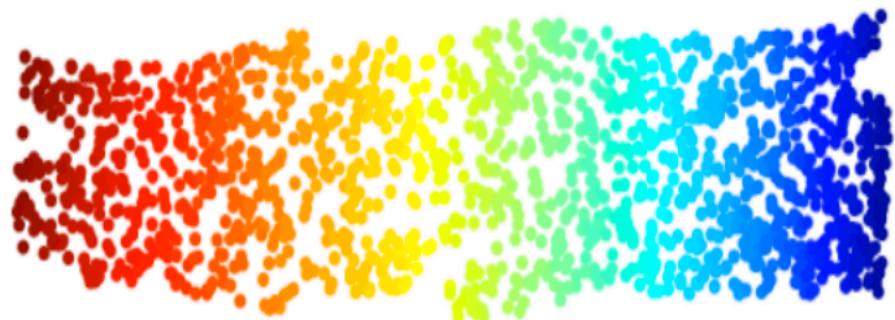
$$1 = \boxed{\lambda_0} \geq \boxed{\lambda_1} \geq \boxed{\lambda_2} \geq \dots \geq \boxed{\lambda_\delta} > 0$$



$$x \mapsto \Phi(x) \triangleq [\lambda_0\phi_0(x), \lambda_1\phi_1(x), \lambda_2\phi_2(x), \dots, \lambda_\delta\phi_\delta(x)]^T$$

[Coifman, Lafon, ACHA 2006]

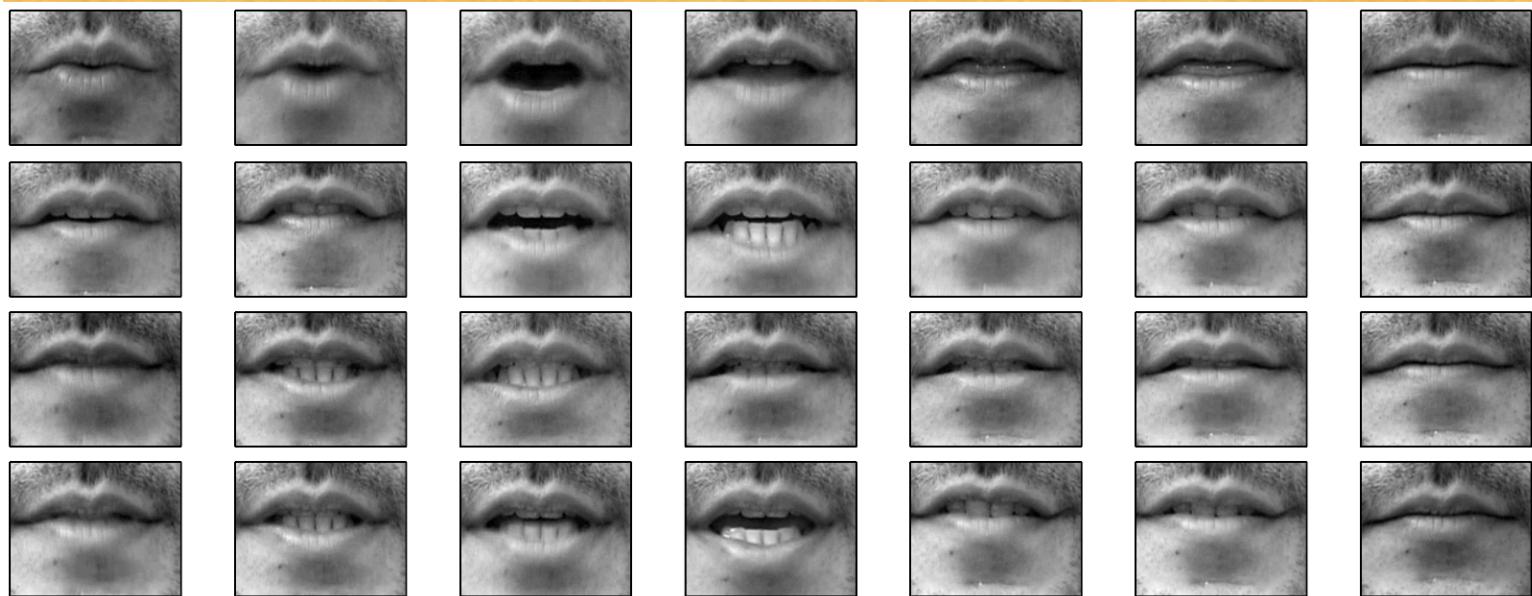
Uncoiling the space



First non-trivial eigenvector DM1 tracks the most prominent non-linear path

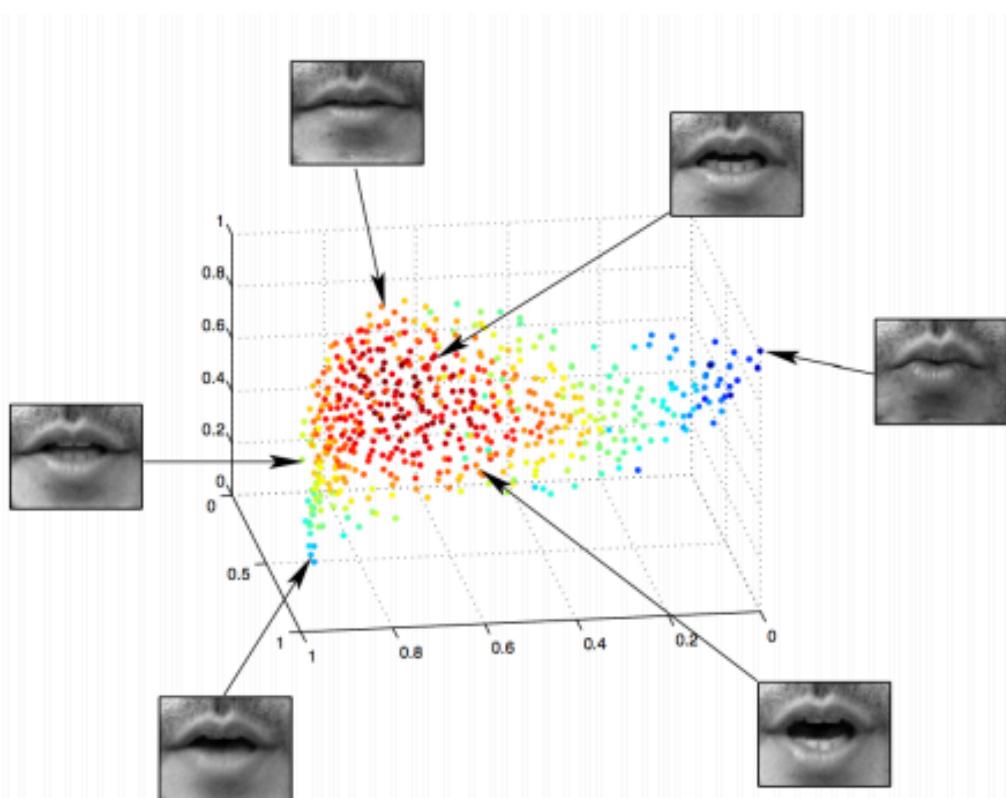
Lips Dataset: Spoken Digits

ONE



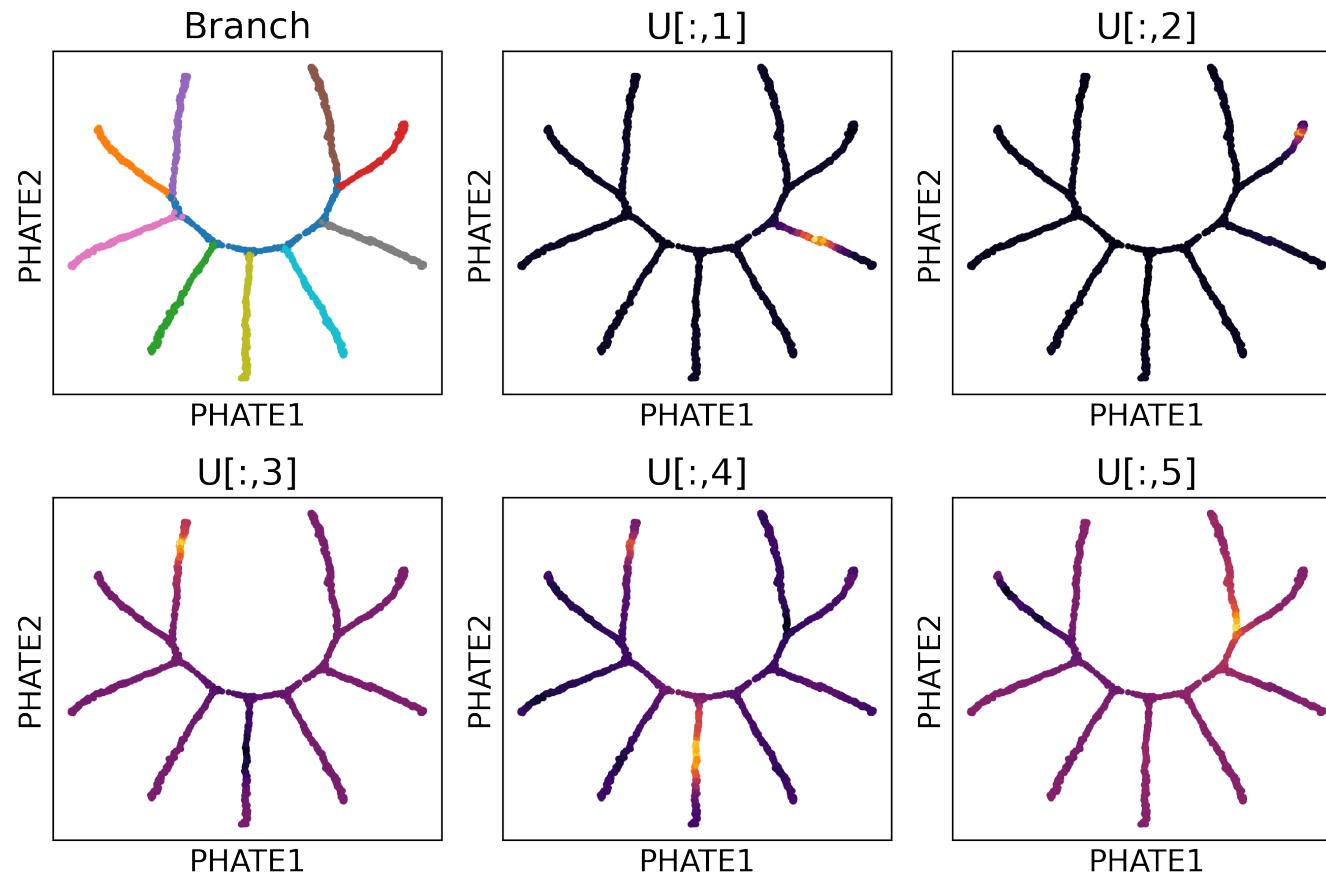
³S. Lafon, Y. Keller, and R. R. Coifman. Data Fusion and Multi-Cue Data Matching by Diffusion Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pages 1784–1797, 2006.

Lips Manifold

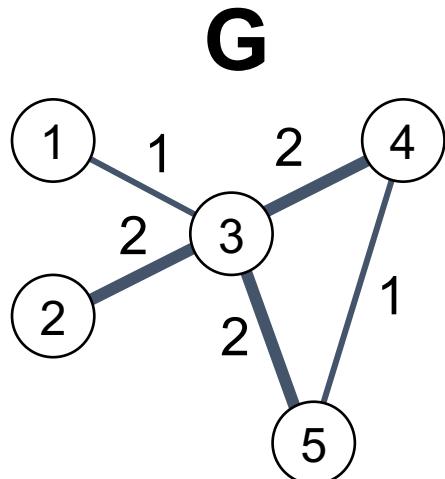


Opening of mouth
portion of teeth
visible

Diffusion components orthogonal



The graph Laplacian is a matrix representation of a graph



A adjacency matrix

$$\mathbf{A} = \begin{vmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 2 & 2 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 & 0 \end{vmatrix}$$

D degree matrix

$$\mathbf{D} = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{vmatrix}$$

Laplacian matrix

$$\mathcal{L} = \mathbf{D} - \mathbf{A} = \begin{vmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ -1 & -2 & 7 & -2 & -2 \\ 0 & 0 & -2 & 3 & -1 \\ 0 & 0 & -2 & -1 & 3 \end{vmatrix}$$

Laplacian Eigenmaps

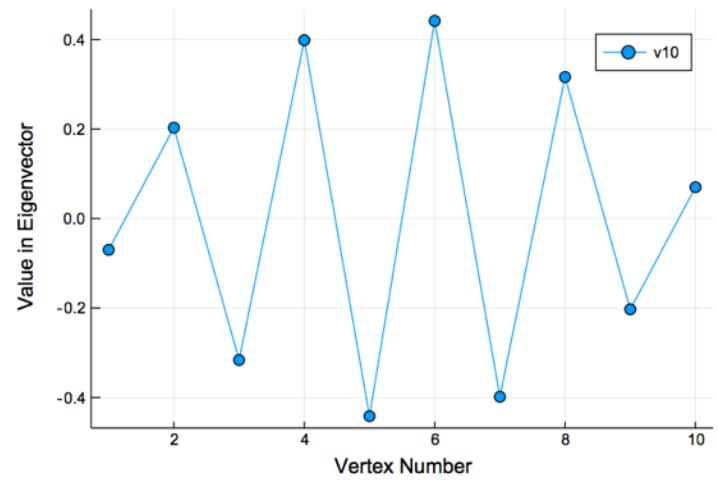
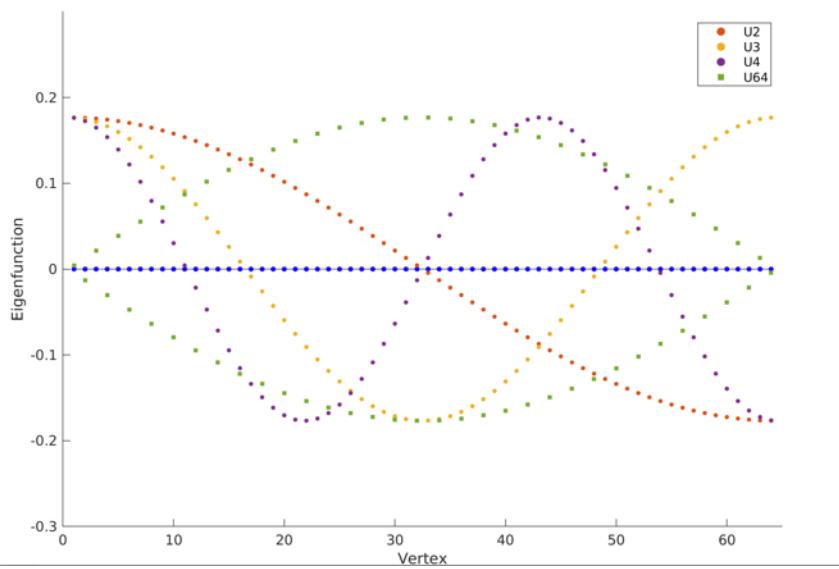
$$\mathcal{L} = \mathbf{D} - \mathbf{A} = \begin{vmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ -1 & -2 & 7 & -2 & -2 \\ 0 & 0 & -2 & 3 & -1 \\ 0 & 0 & -2 & -1 & 3 \end{vmatrix}$$

Eigenvectors of Laplacian matrix form Laplacian Eigenmaps

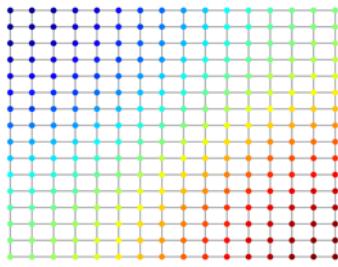
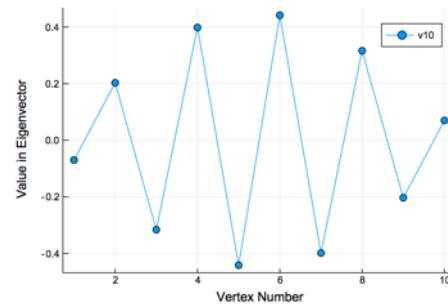
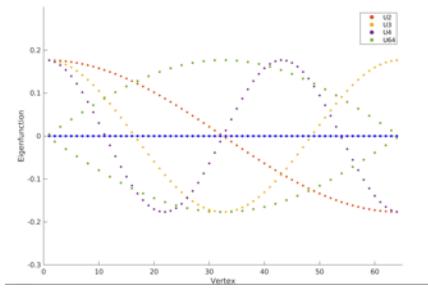
Graph Laplacian

- A difference operator based on the graph adjacency matrix A .
- (unnormalized Laplacian) $L = D - A$
 - Degree matrix: $D_{ii} = \sum_j A_{ij}$, 0 everywhere else
- (normalized Laplacian) $L = I - D^{-1/2}AD^{-1/2}$
 - I is identity
 - Measures how similar a point is to its neighbors
- (random walk Laplacian) $L = I - D^{-1}A = I - M$
 - Related to Markovian matrix

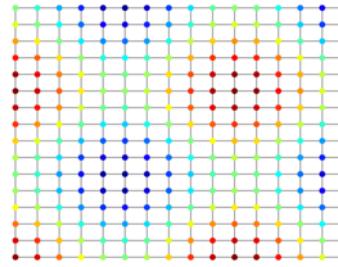
Eigenvectors form a Graph Spectrum



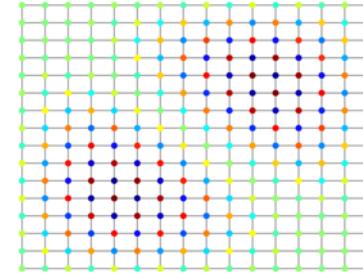
Eigenvectors are frequency harmonics



2nd Eigenvector



10th Eigenvector

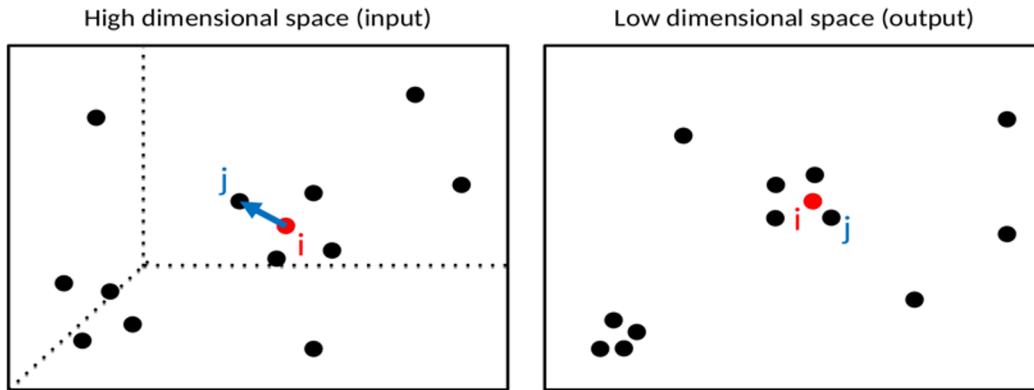


2nd to last eigenvector

For visualization

- These methods preserve information from an affinity matrix in 2-3 dimensions exactly
- There are as many eigenvectors as datapoints, so taking only the top 2 misses information
 - **tSNE/UMAP:** greedy method for preserving near neighbors in 2D
 - **PHATE:** creates a new distance matrix from a distance between diffusion probabilities of points and squeezes that variation into 2D

tSNE



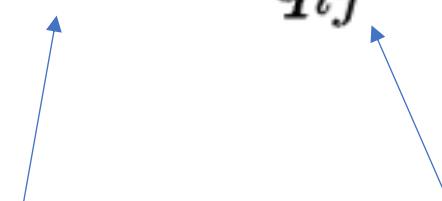
Goal: Maintain the local neighborhood of each cell

Steps:

1. Place points randomly in a 2D plane
2. Move points to locally optimal location via gradient descent

$$\frac{\delta C}{\delta y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

tSNE Penalty

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$


Neighbors in high dimensions Neighbors in low dimensions

What happens when the p_{ij} is SMALL?

No effective penalty for placement of points that are not close

Combination of attraction and repulsion

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Neighbors in high dimensions

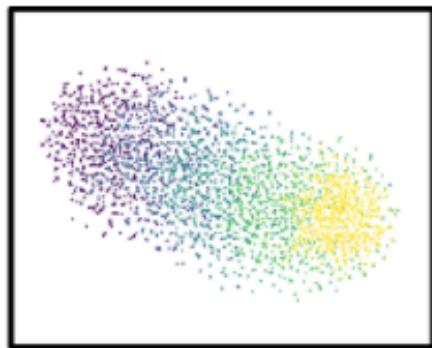
High probabilities to some neighbors have to be counteracted by some low probabilities to other points (sum has to be 1)

To fit close neighbors together have to repel far points. This is what leads to a “explosion”-like embedding

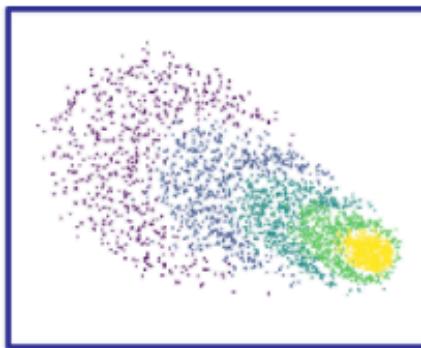
UMAP

- Very similar to tSNE (but its NOT published!)
- What about the “uniform manifold approximation?”
 - This is actually saying that UMAP approximates a uniform manifold from your data
 - In other words tSNE/UMAP plots have even density throughout
- Why uniform density?
 - They use a Markov matrix where the bandwidth is set to the Kth nearest neighbor, so each point is placed near its K-nearest neighbors no matter how near or far they are
 - **Advantage** of this is that you can see how many points are in each cluster
 - **Disadvantage** is you don’t get an idea of the real (even local) density

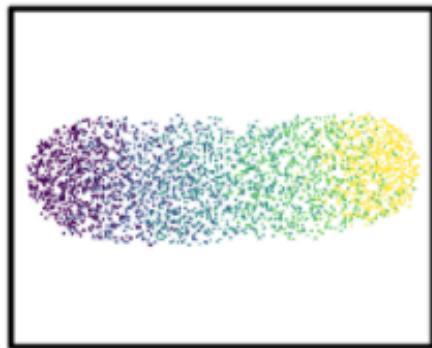
Local density inducing variant



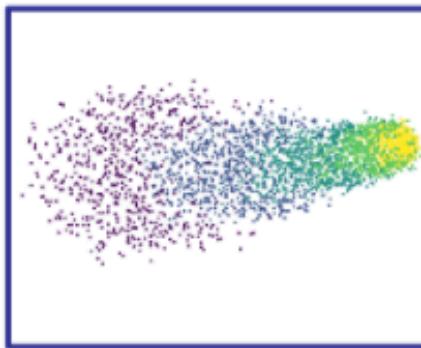
t-SNE



den-SNE



UMAP

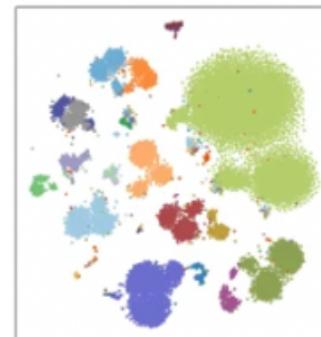


densMAP

<http://cb.csail.mit.edu/cb/densvis/>

Differences from tSNE

- Initialized with a Laplacian eigenmap, stronger start to a greedy method
- Noise-contrastive estimation in the implementation
 - UMAP uses an un-normalized matrix (probabilities don't sum to 1)
 - Instead some points are sampled and if they are far away they have to be repelled
 - All nearest neighbors have to be put close
 - This decreases the repulsion
- Less “explosive looking”



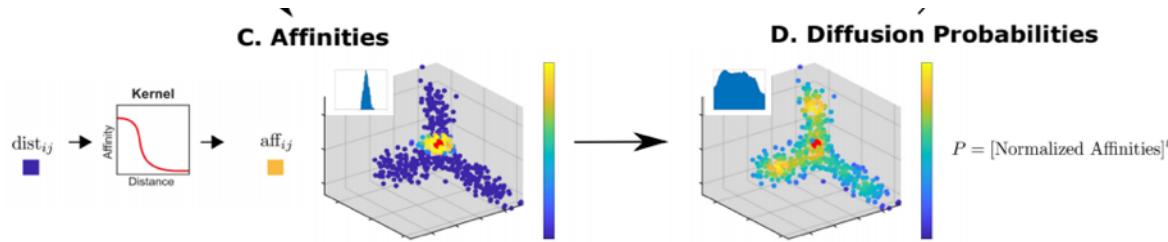
tSNE



UMAP

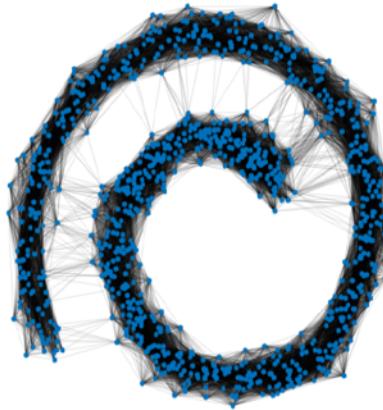
PHATE:

- Goal: to preserve manifold structure
- Step 1. Each cell is represented as a t-step diffusion probability to other cells

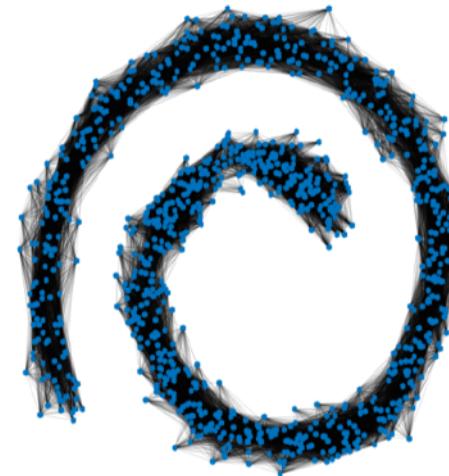


- Diffusion induces global connectivity and denoises paths
- PHATE uses an adaptive Markov matrix too! But using it for diffusion is very different from using it to match neighbors directly

Diffusion in PHATE denoises



**Connectivity
without diffusion**



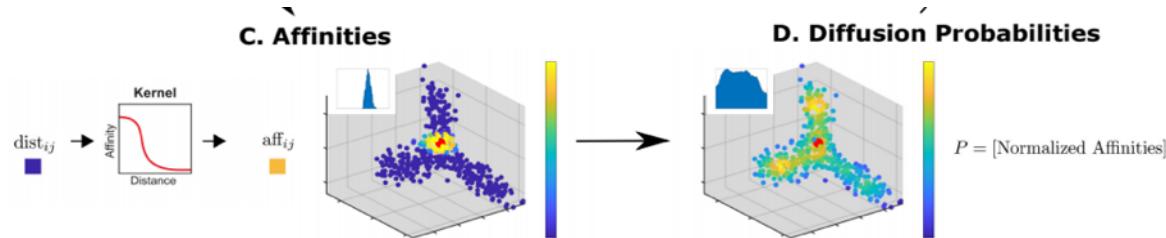
**Connectivity after
 t -steps of diffusion**

Adaptive Alpha-decay kernel

- Adaptive kernel in PHATE separate geometry from density
- If there are density variations in the initial Markov matrix creation then paths tend to drift towards dense regions and it DISTORTS geometry
- Adaptive kernel (bandwidth set to Kth nearest neighbor) reveals geometry

PHATE:

- Goal: to preserve manifold structure
- Step 1. Each cell is represented as a t-step diffusion probability to other cells



- Step 2: Re-represent each point by its diffusion probabilities to all other points

$$C_1 = [p_{11} \ p_{12} \ \dots \ p_{1n}]$$

$$C_2 = [p_{21} \ p_{22} \ \dots \ p_{2n}]$$

...

$$C_n = [p_{n1} \ p_{n2} \ \dots \ p_{nn}]$$

- Step 3. Define a new distance between points with this new representation

$$C_1 = [p_{11} \ p_{12} \ \dots \ p_{1n}]$$

.....

$$C_i = [p_{i1} \ p_{i2} \ \dots \ p_{in}]$$

...

$$C_j = [p_{j1} \ p_{j2} \ \dots \ p_{jn}]$$

...

$$C_n = [p_{n1} \ p_{n2} \ \dots \ p_{nn}]$$

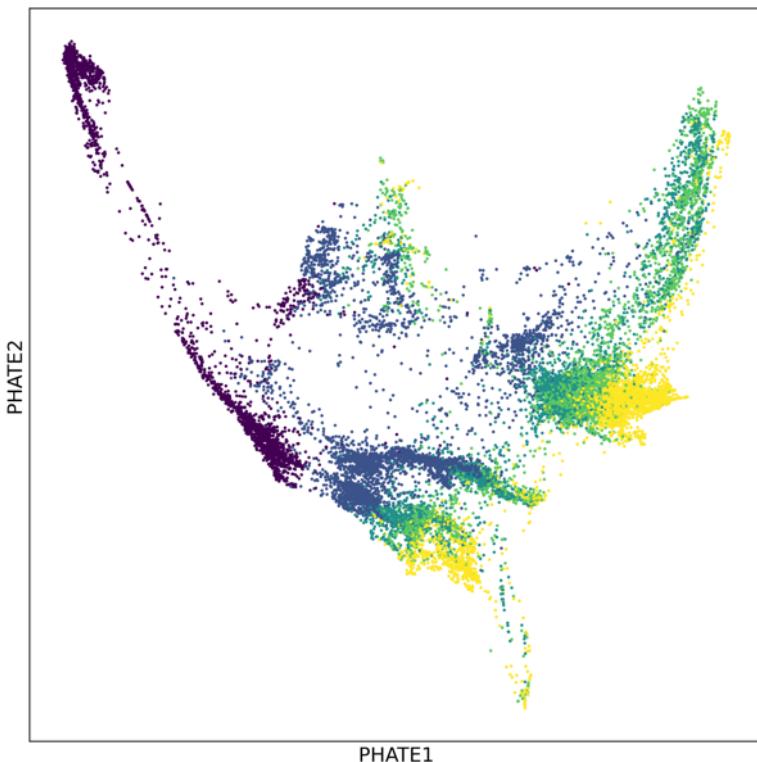
$$\text{dist}_{ij} = \sqrt{\| \log P_i - \log P_j \|^2}$$



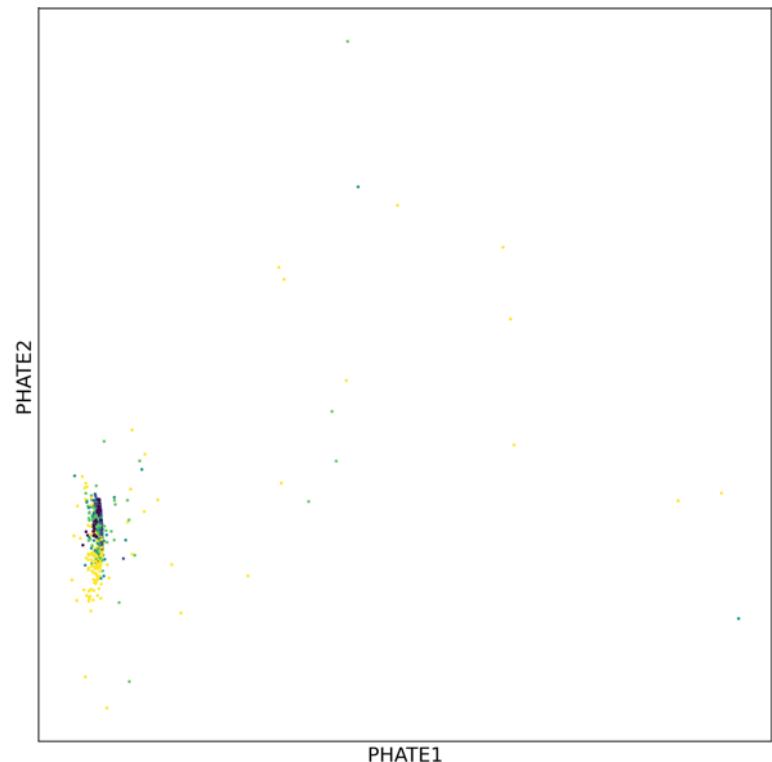
- Step 4: Embed with multidimensional scaling
- This reorganizes information in a diffusion map into 2D

Adaptive vs fixed bandwidth kernel

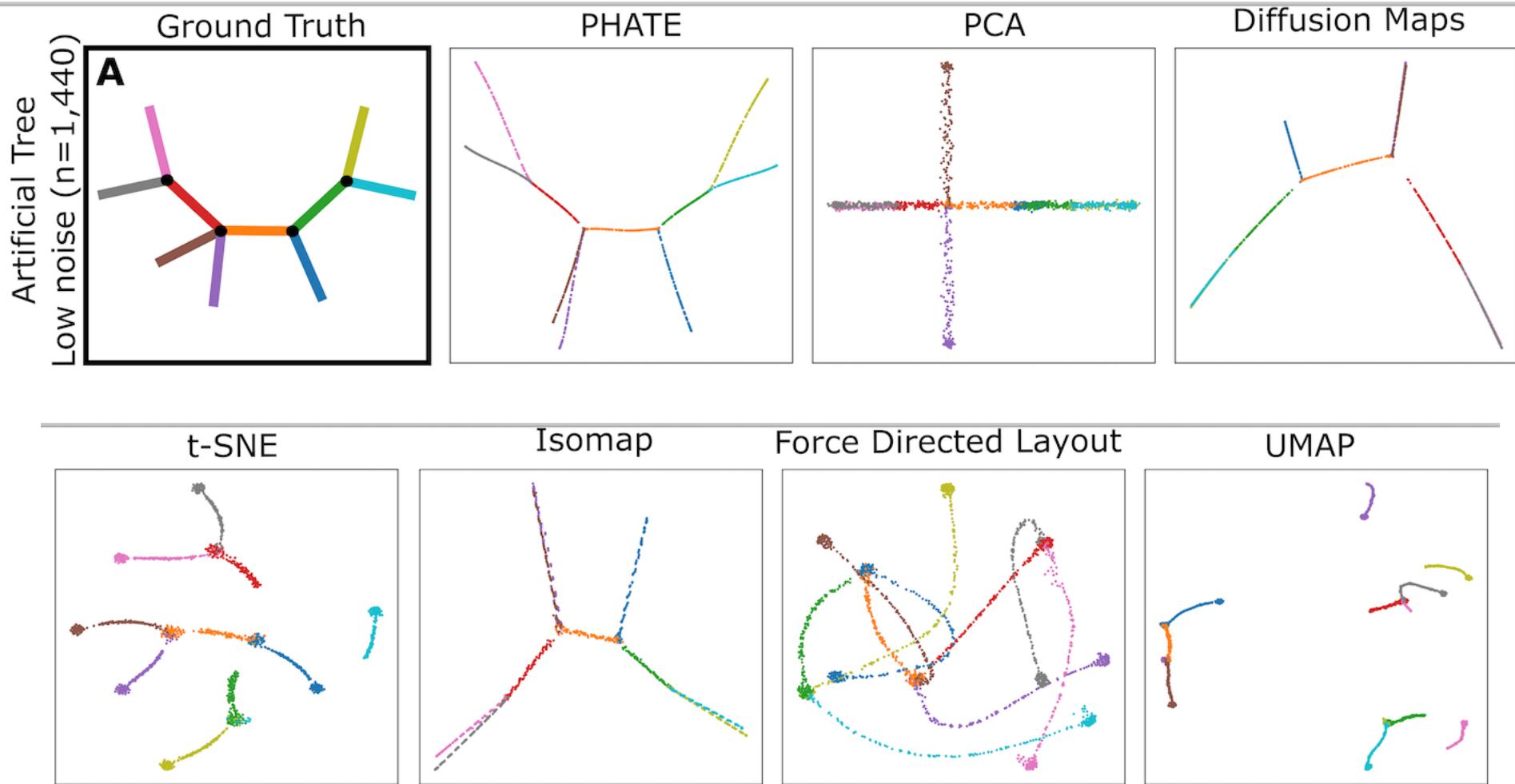
Adaptive



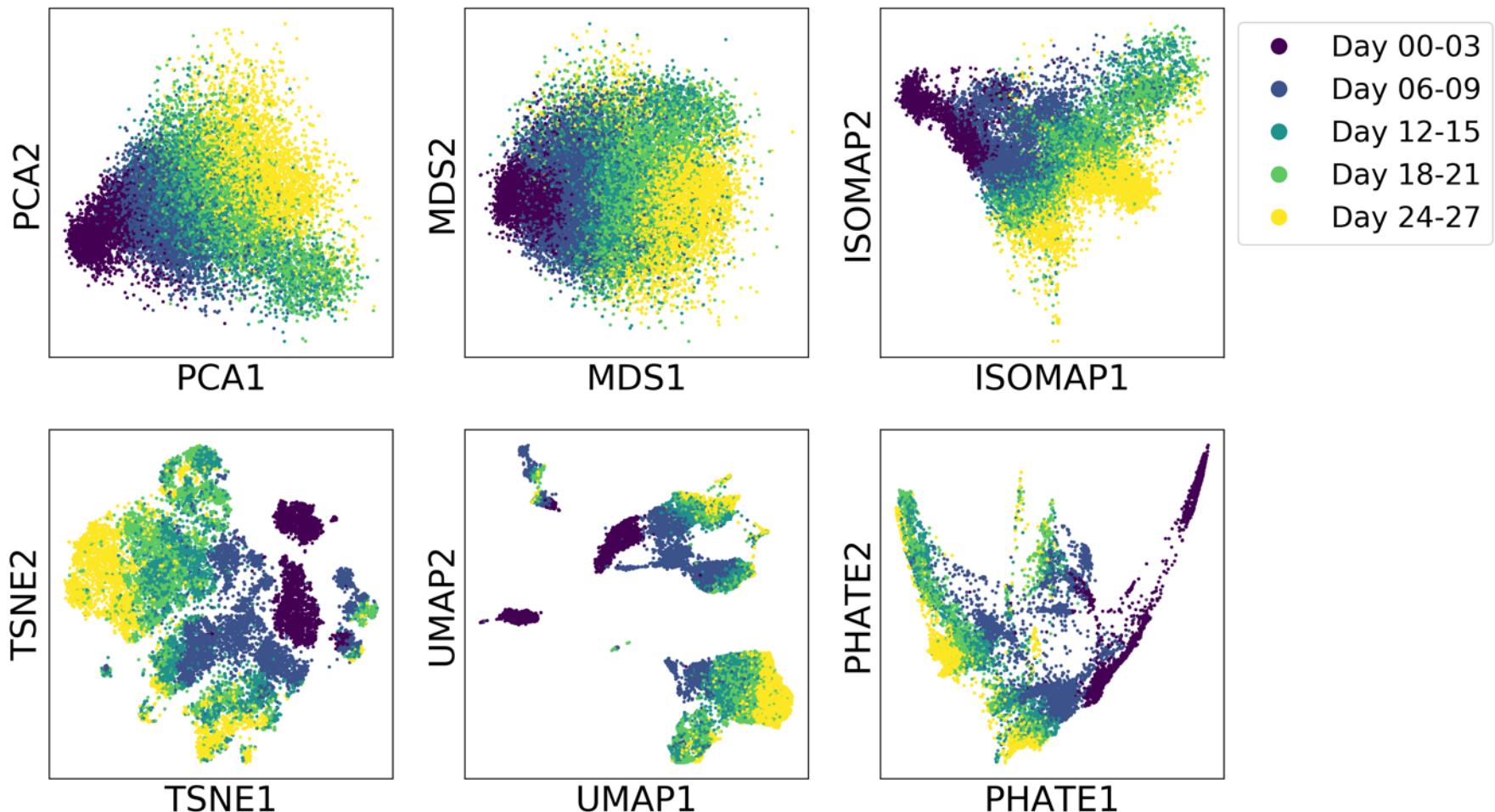
Fixed

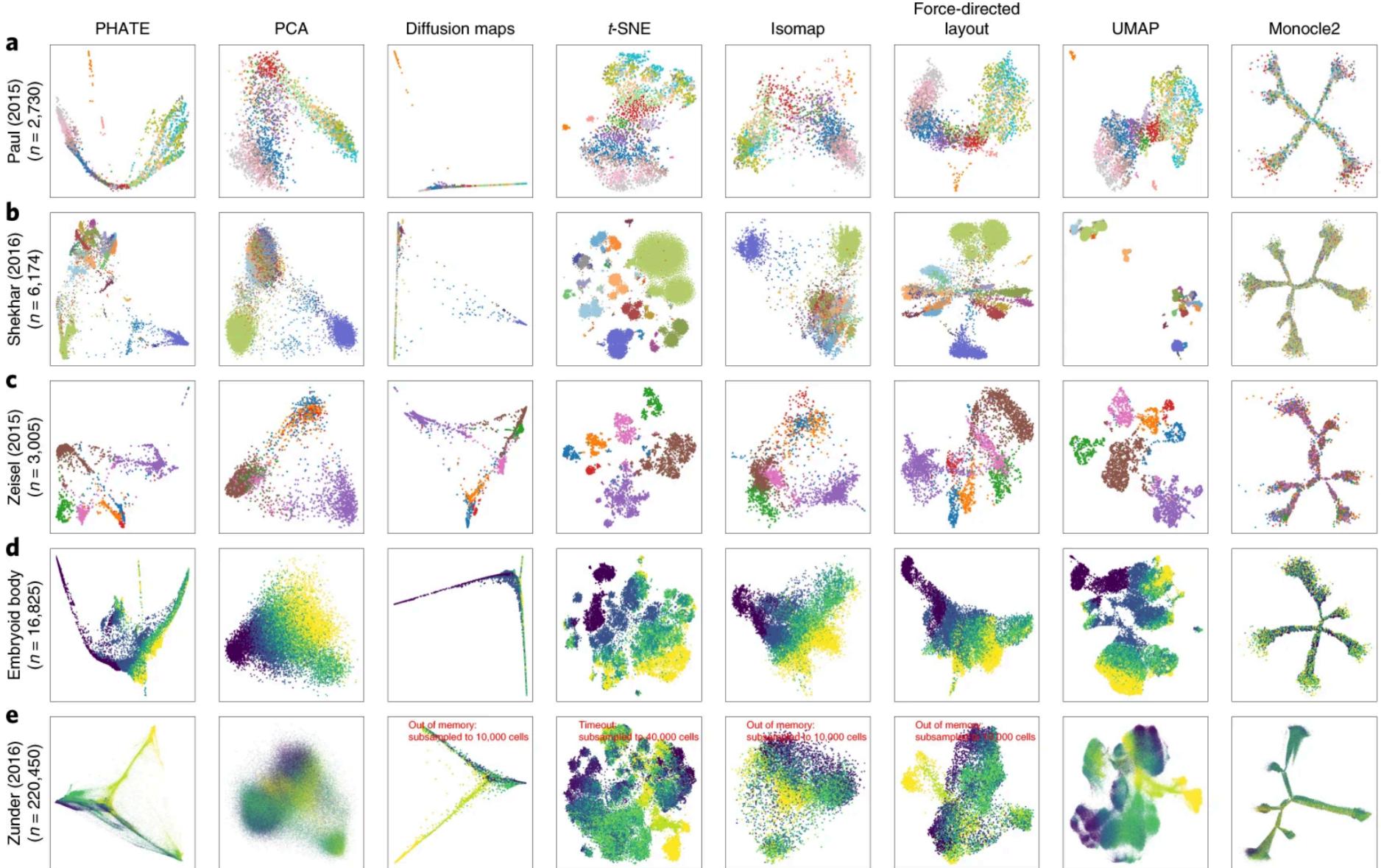


Comparing Artificial Case

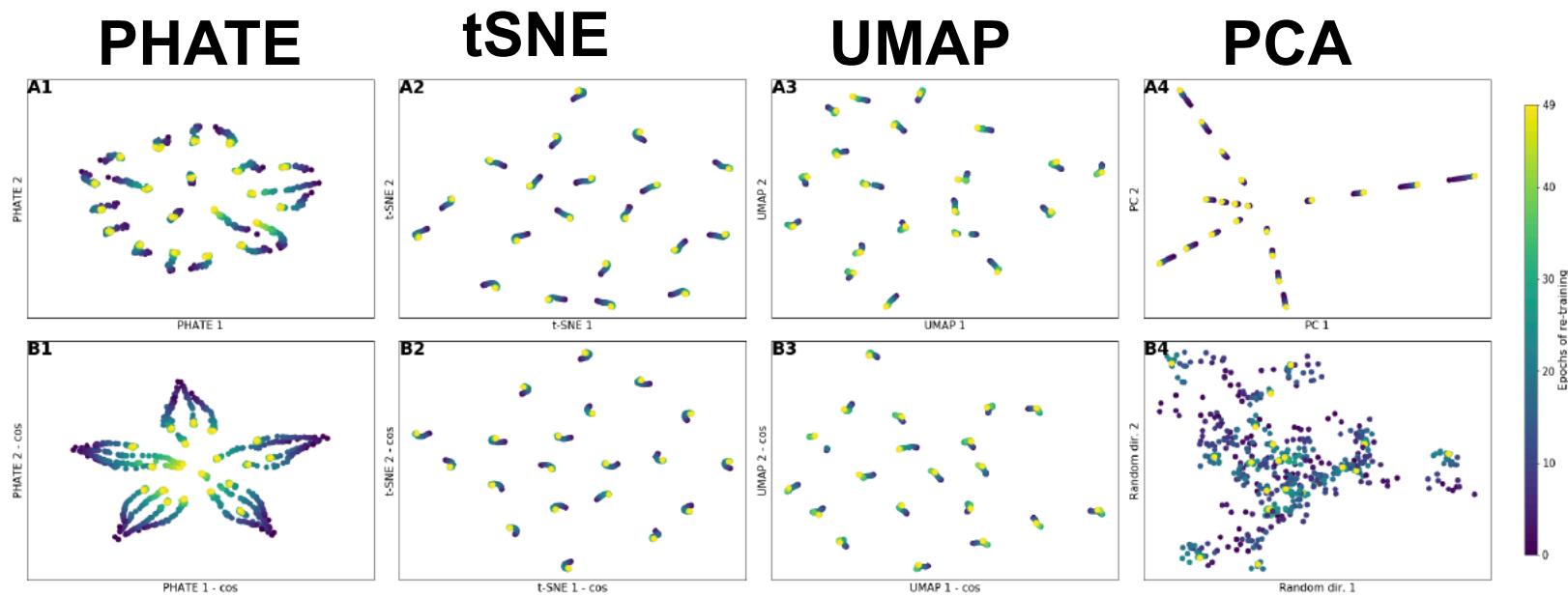


Comparing visualization algorithms on single-cell data



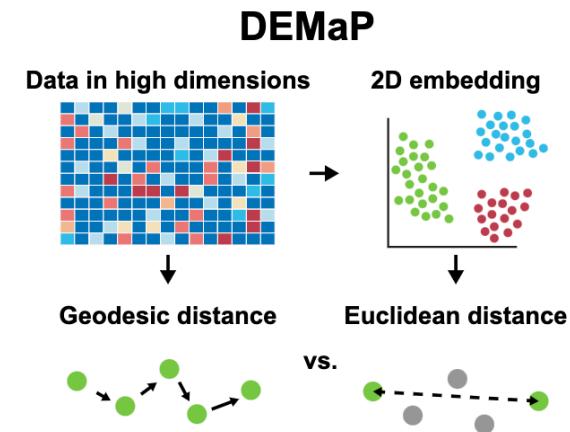
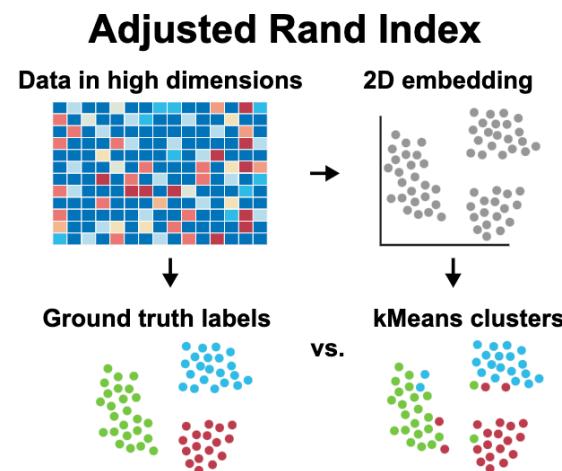
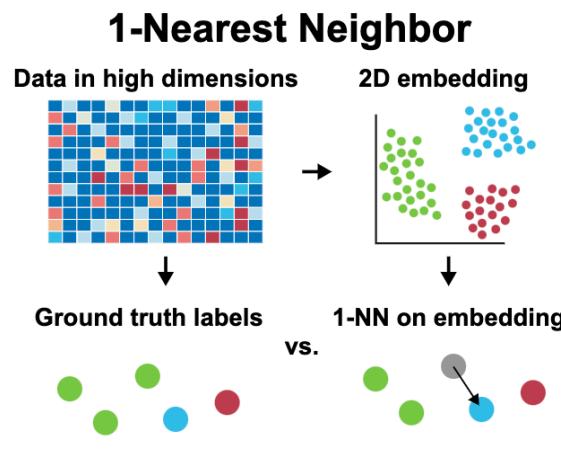


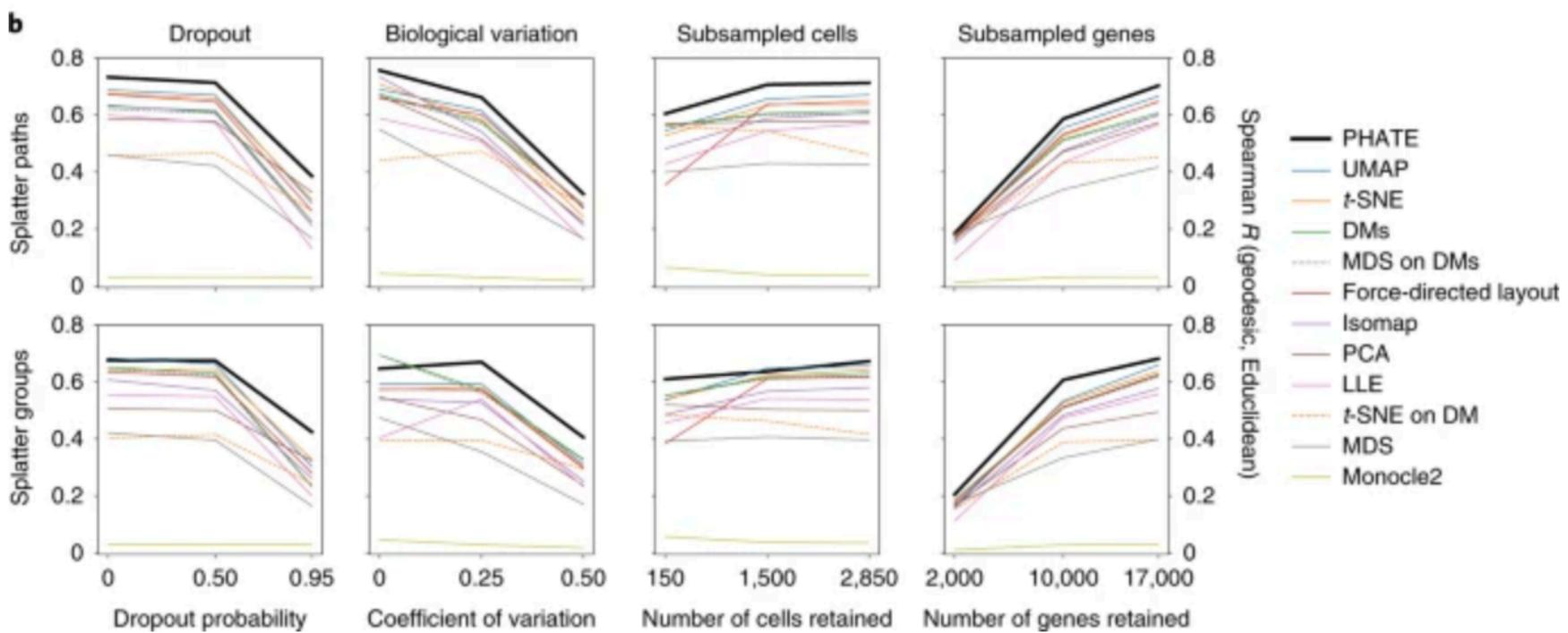
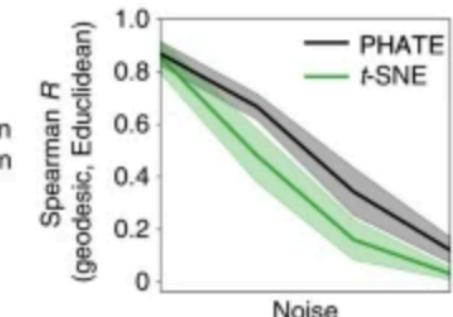
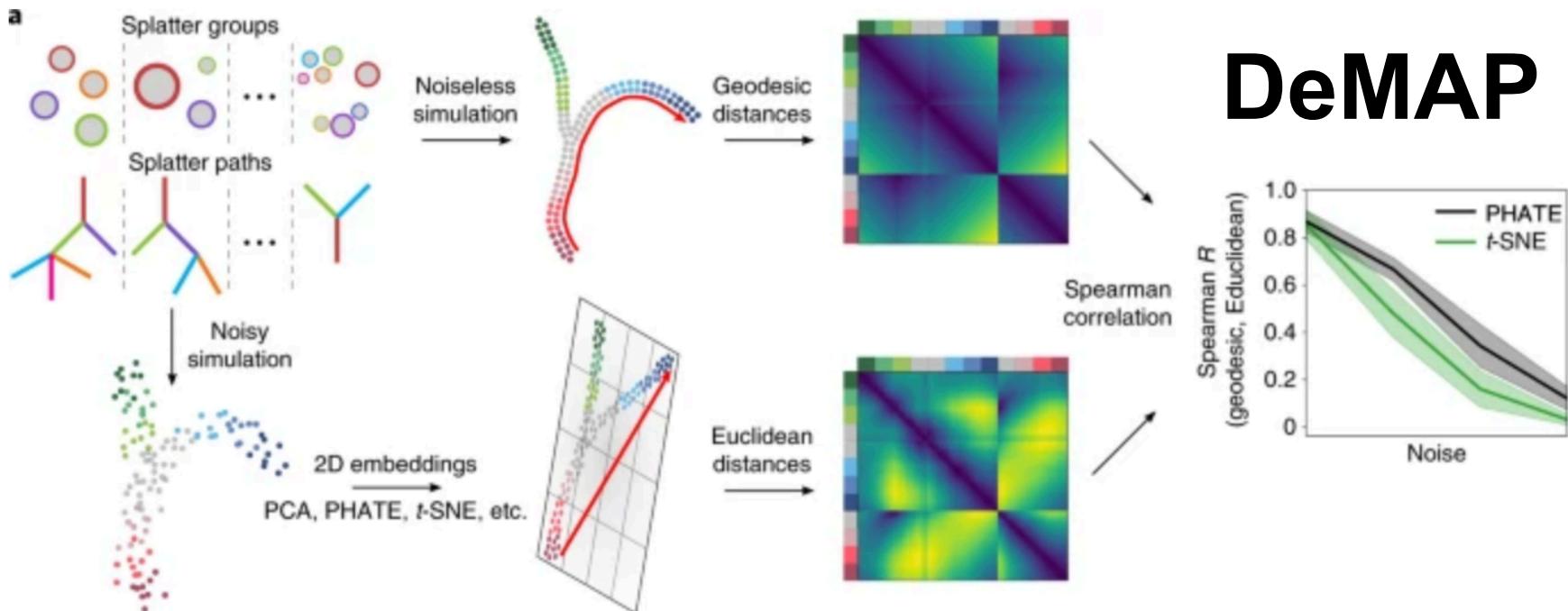
Data agnostic: Neural network loss landscapes

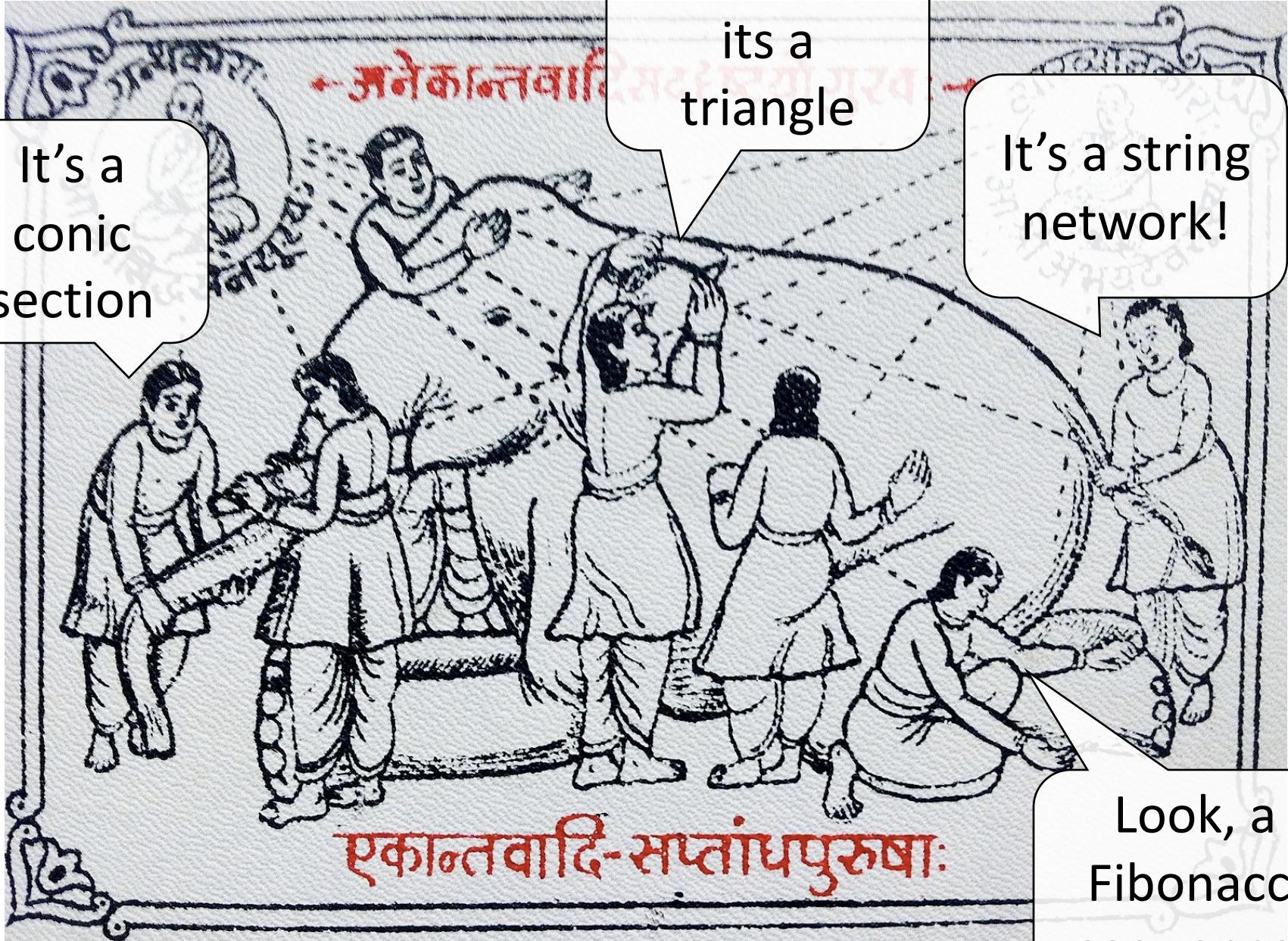


[Horoi et al. 2020]

Quantitative Comparisons







Conclusions

- Dimensionality reduction and visualization helps to identify structure in data
- PCA uses eigenvectors to identify linear combinations of features that explain maximum variance
- Manifold learning identifies non-linear patterns and structure
- Graphs can be used to approximate manifolds and are helpful for modelling single cell data
- Laplacian Eigenmaps and diffusion maps visualize data using dominant non-linear directions in data manifold
- PHATE summarizes manifold information in 2D
- UMAP, tSNE are non-linear methods focused on preserving local neighborhoods in 2D

What questions do you have about today's lecture material?

Top