The Krishnaswamy Laboratory

Yale Genetics and Yale SEAS present

Machine Learning for Single Cell Analysis

Online - May 20-29, 2020

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?

Recap from Day 1

Two kinds of machine learning

Supervised learning

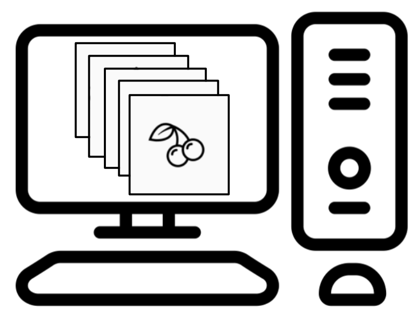
- Have a bunch of labelled data, want to label new data
- Learn a function $f(X) \rightarrow Y$ where all values of Y are known for some samples of X

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it
- Learn an embedding $f(X) \rightarrow Y, X \in \mathbb{R}^n, Y \in \mathbb{R}^m, n \gg m$
- Lower dimensional, easier to interpret (e.g. as clusters)

Supervised Learning Model

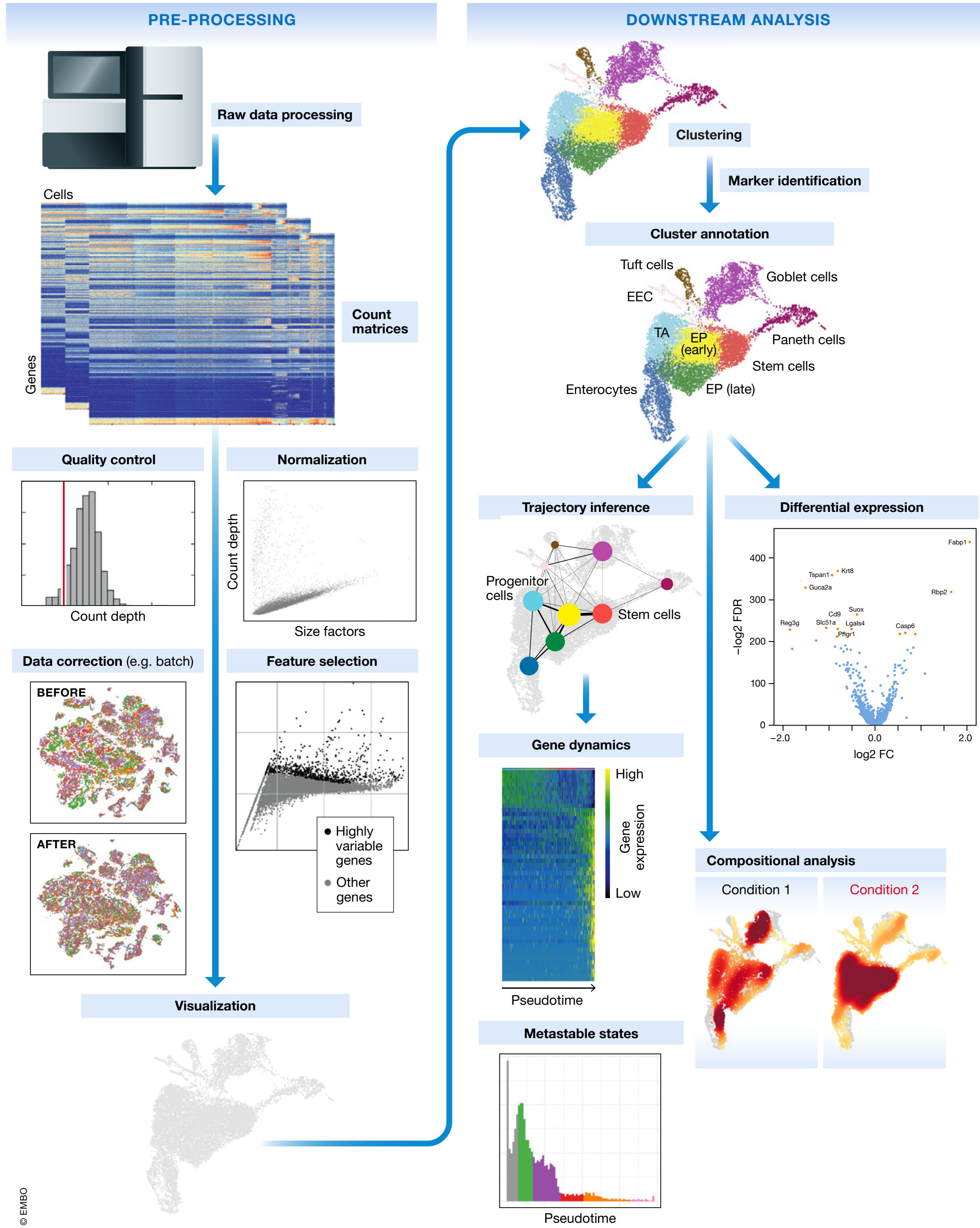
Unsupervised Learning Model

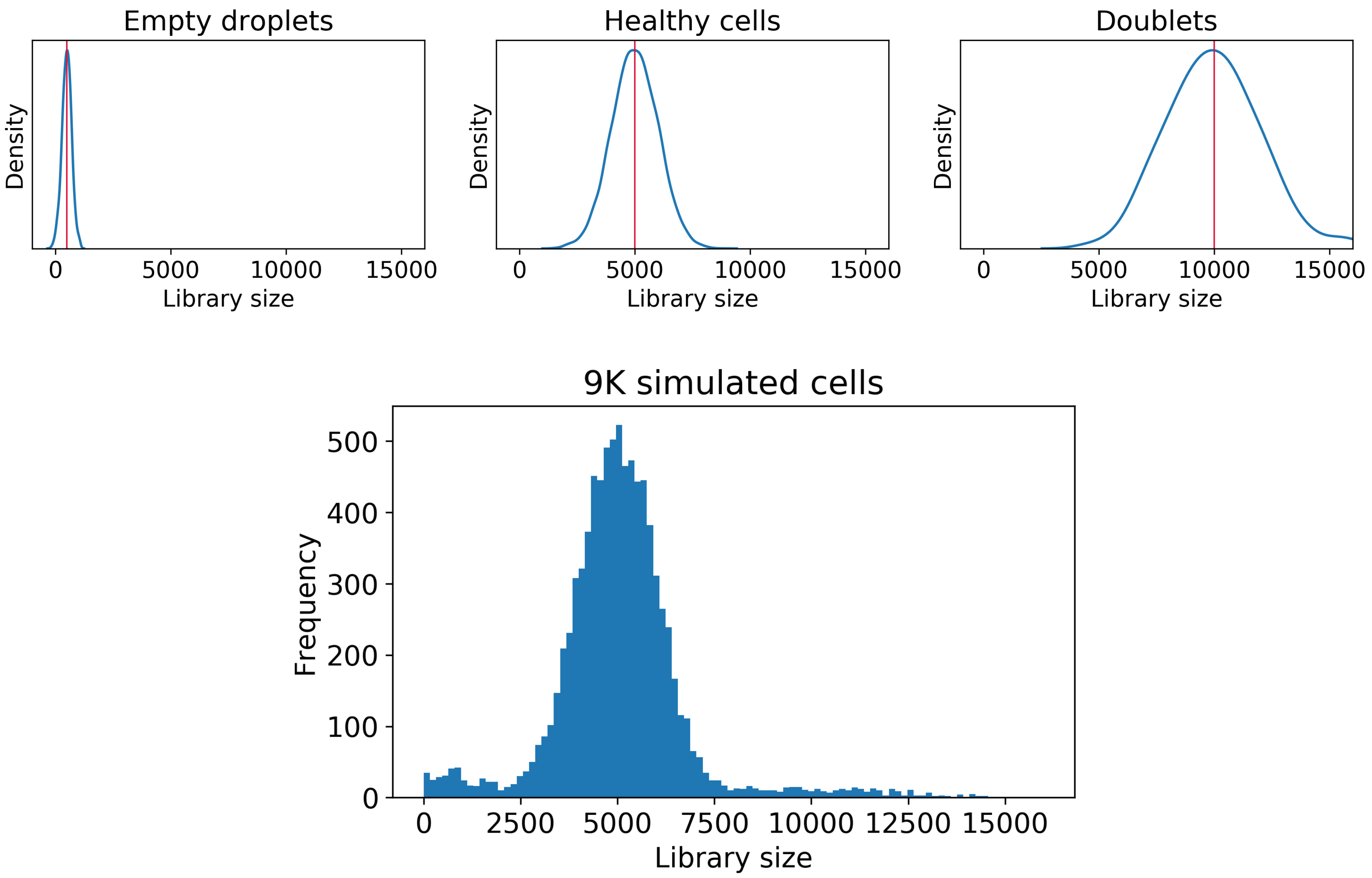


Noun project

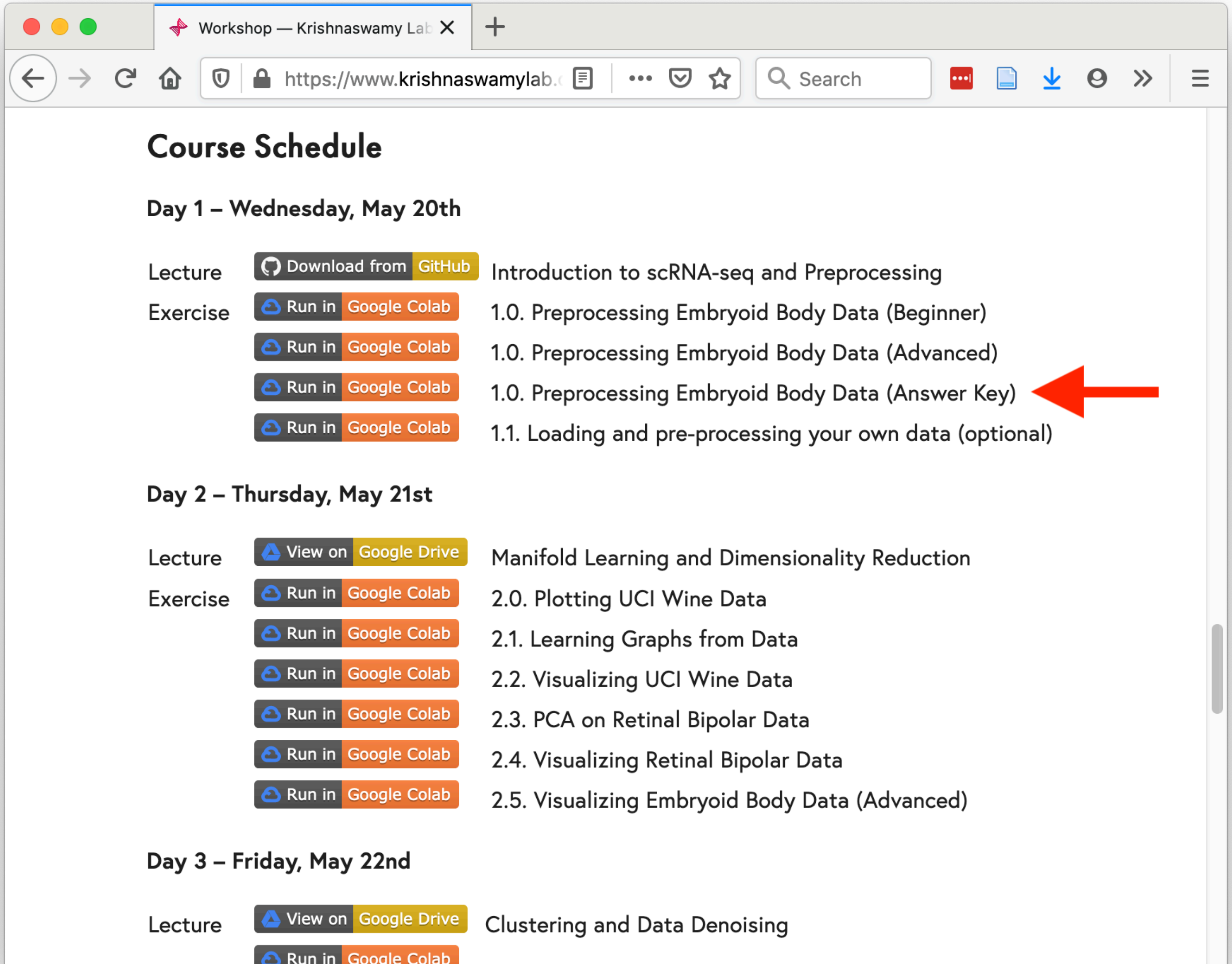
Standard Single-Cell RNA-seq Workflow

1. Sequencing and read mapping
2. Quality control and filtering
3. Normalization
4. Data Correction
5. Dimensionality reduction and visualization
6. Downstream analysis
    1. Clustering
    2. Trajectory inference
    3. Differential expression
7. Comparison of multiple conditions

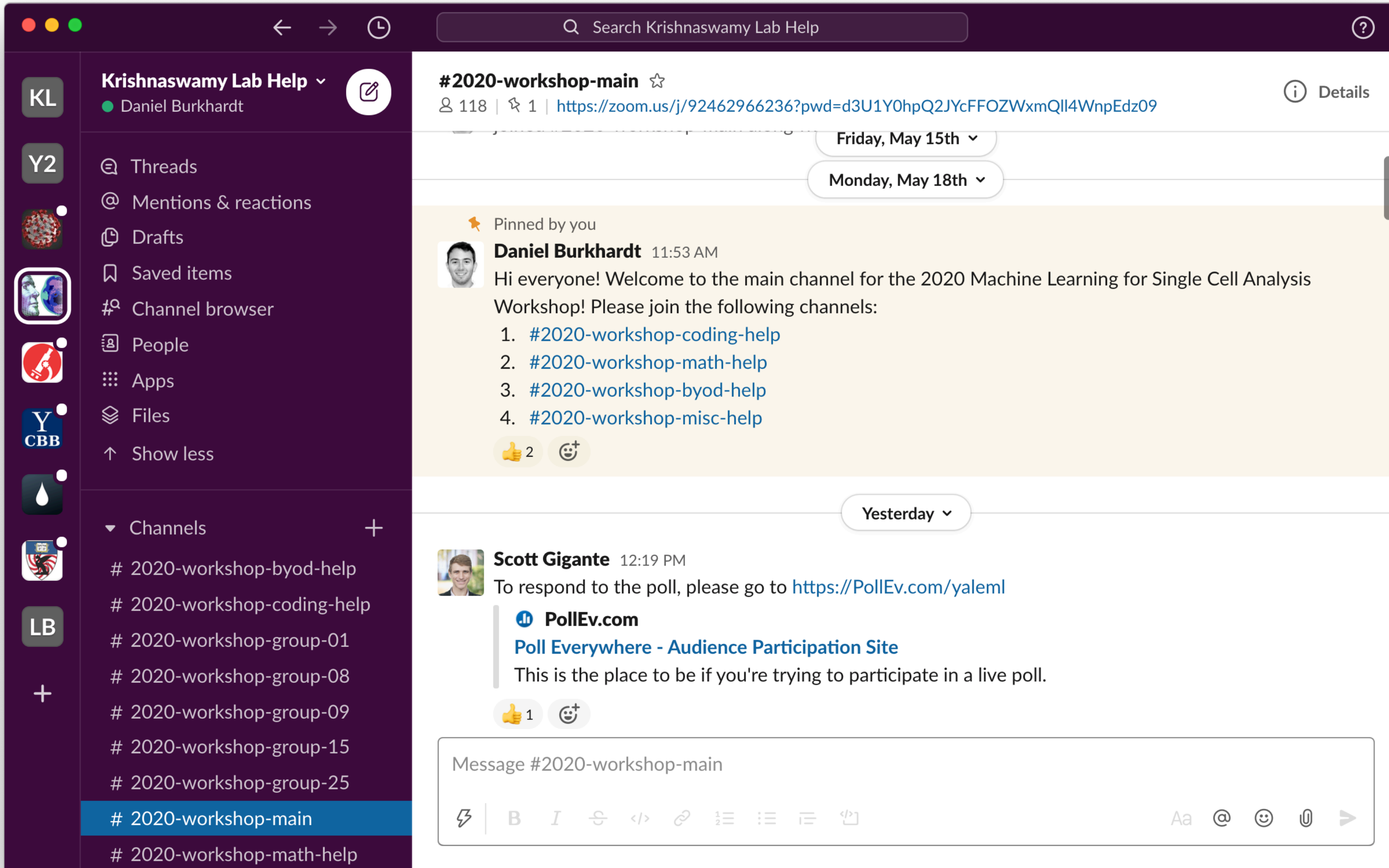




Empty droplets
Healthy cells
Doublets
Density
Library size
0
5000
10000
15000
9K simulated cells
Frequency
0
100
200
300
400
500
0
2500
5000
7500
10000
12500
15000
Library size



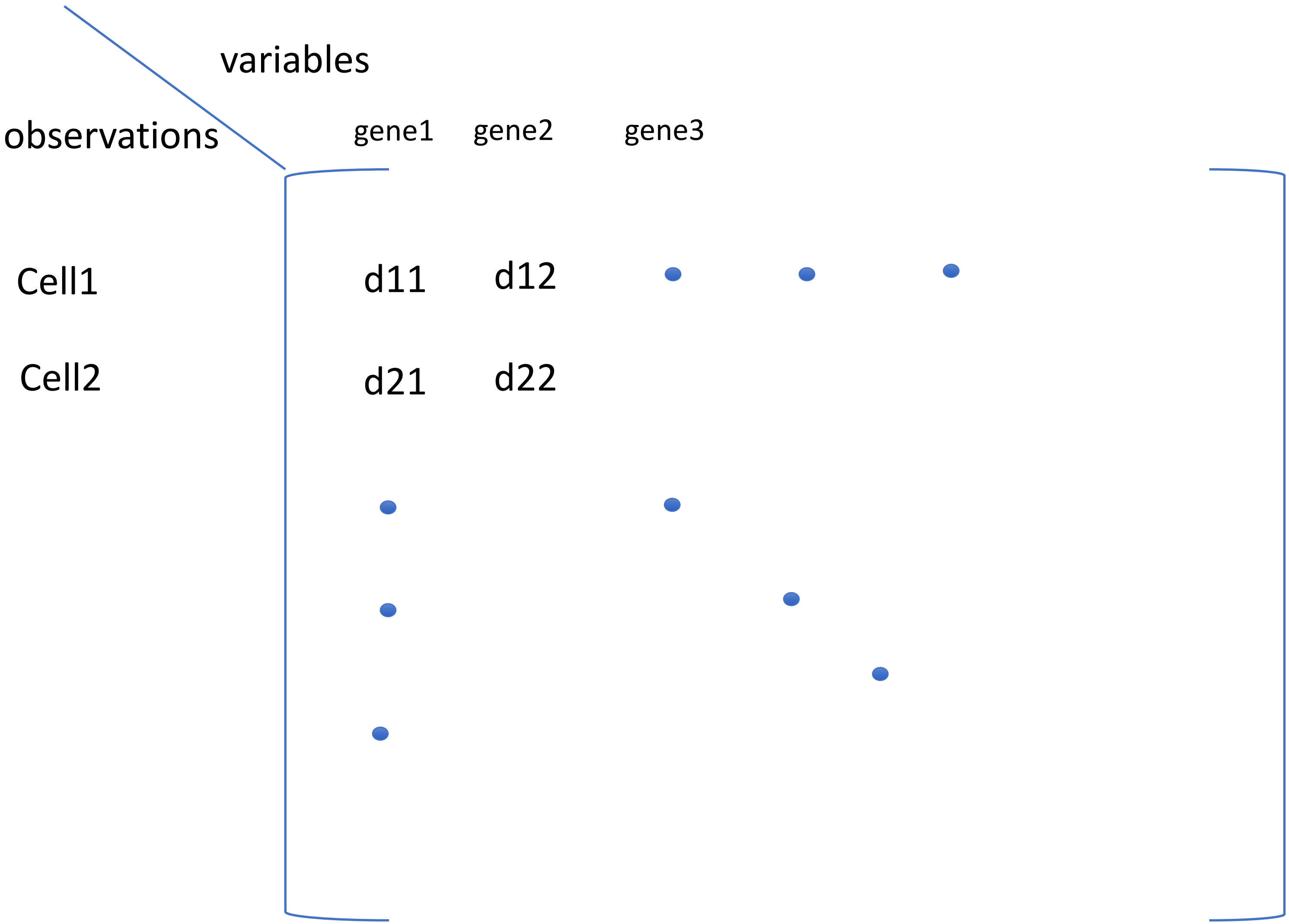
https://www.krishnaswamylab.org/workshop


Search Krishnaswamy Lab Help
KL
Y2
Y
CBB
LB
Krishnaswamy Lab Help
Daniel Burkhardt
Threads
Mentions & reactions
Drafts
Saved items
Channel browser
People
Apps
Files
Show less
Channels
2020-workshop-byod-help
2020-workshop-coding-help
2020-workshop-group-01
2020-workshop-group-08
2020-workshop-group-09
2020-workshop-group-15
2020-workshop-group-25
2020-workshop-main
2020-workshop-math-help
#2020-workshop-main
118 | 1 | https://zoom.us/j/92462966236?pwd=d3U1Y0hpQ2JYcFFOZWxmQll4WnpEdz09
Details
Friday, May 15th
Monday, May 18th
Pinned by you
Daniel Burkhardt 11:53 AM
Hi everyone! Welcome to the main channel for the 2020 Machine Learning for Single Cell Analysis Workshop! Please join the following channels:
1. #2020-workshop-coding-help
2. #2020-workshop-math-help
3. #2020-workshop-byod-help
4. #2020-workshop-misc-help
2
Yesterday
Scott Gigante 12:19 PM
To respond to the poll, please go to https://PollEv.com/yaleml
PollEv.com
Poll Everywhere - Audience Participation Site
This is the place to be if you're trying to participate in a live poll.
1
Message #2020-workshop-main
Aa

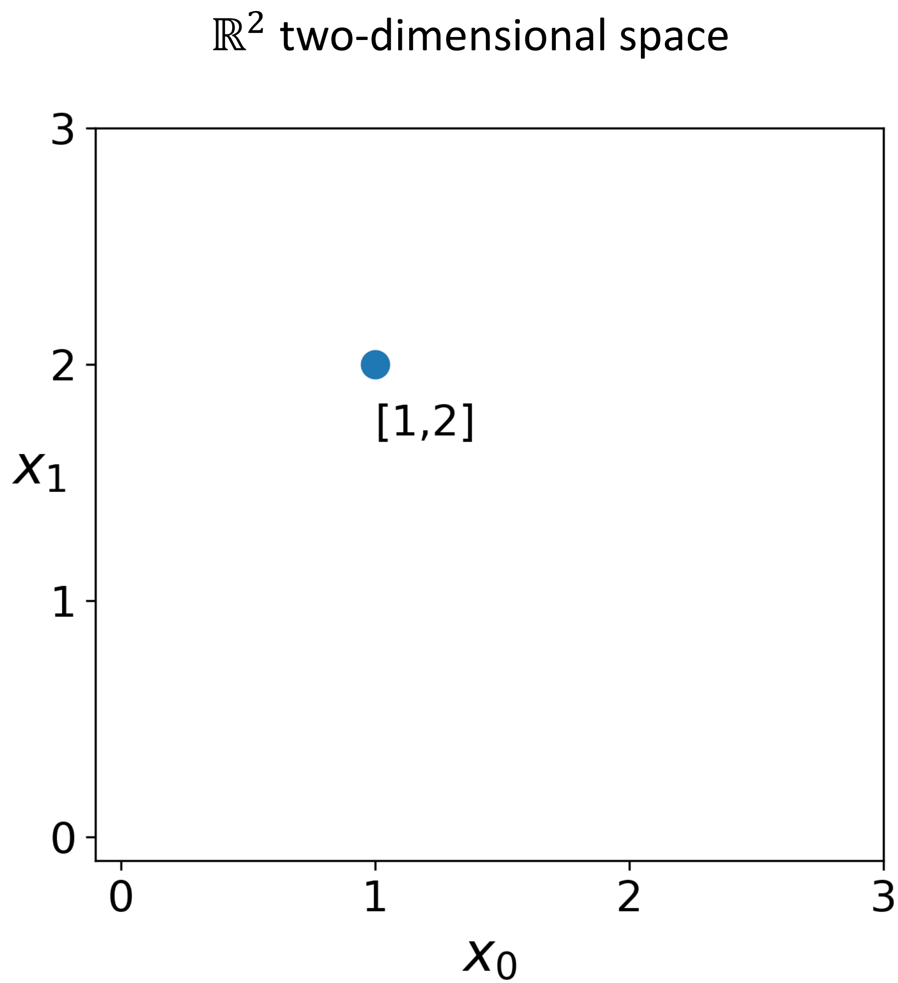
Day 2: Visualization, Dimensionality Reduction and Manifold Learning

Thinking about high dimensional data

Our Data is a High-dimensional Matrix



What does “dimensionality” mean?



Features are dimensions

vector coordinate space

$\boldsymbol{a}_0 \in \mathbb{R}^2$

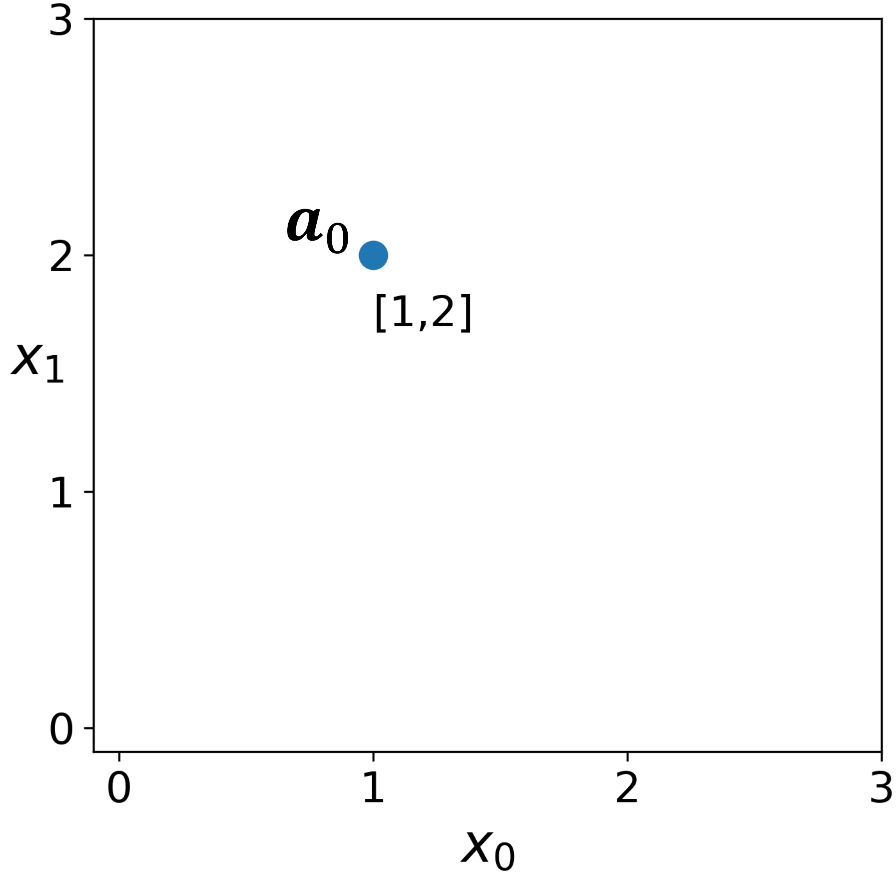
$\boldsymbol{a}_0 = [1,2]$

x_0 x_1

features

$\boldsymbol{a}_{[:,0]}$

$\mathbb{R}^2$ two-dimensional space



Features are dimensions

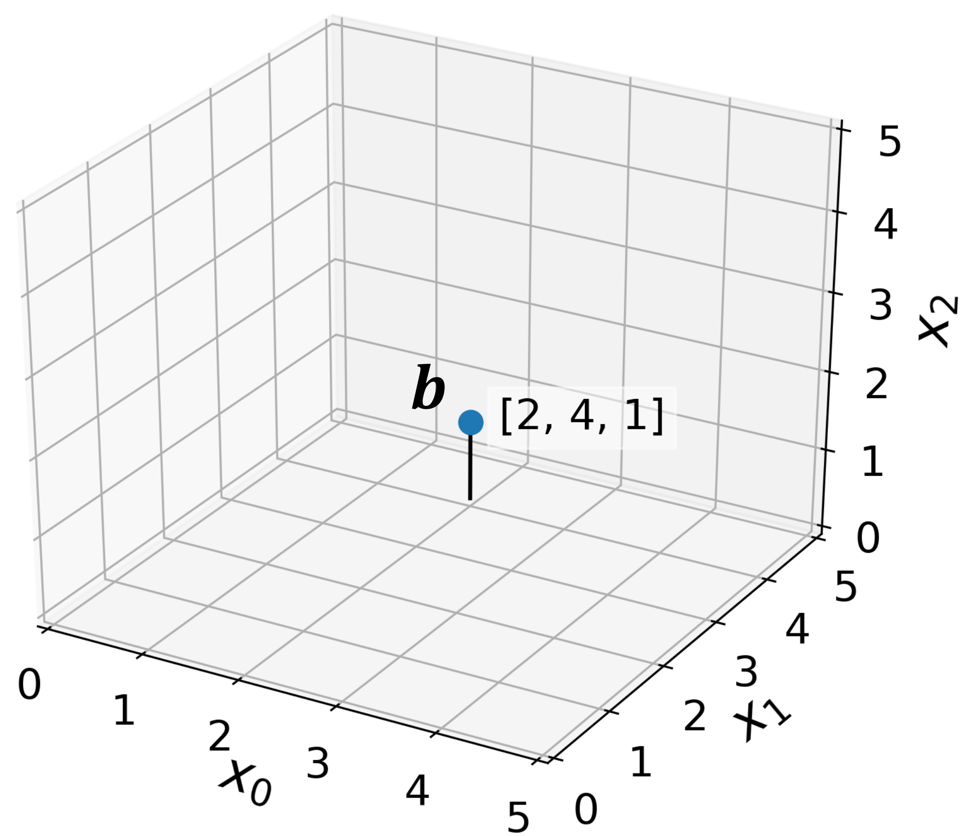
$\boldsymbol{a} \in \mathbb{R}^2$

$\boldsymbol{a} = [\underset{x_0}{1}, \underset{x_1}{2}]$

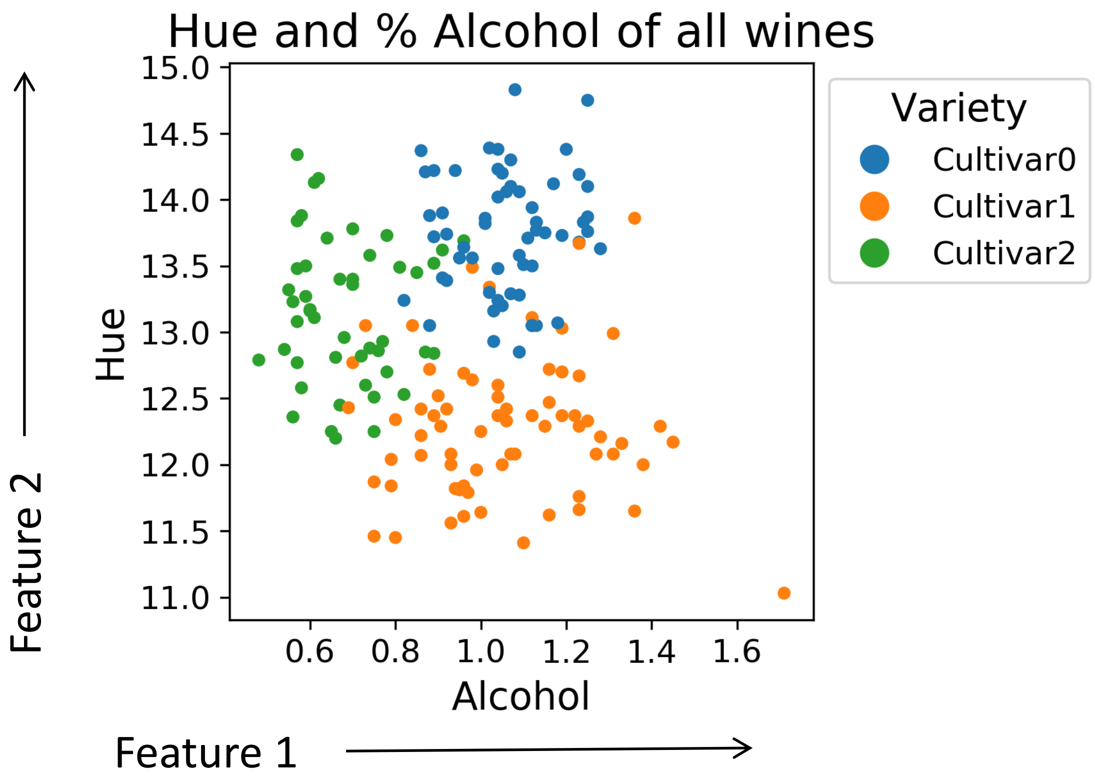
$\boldsymbol{b} \in \mathbb{R}^3$

$\boldsymbol{b} = [\underset{x_0}{2}, \underset{x_1}{4}, \underset{x_2}{1}]$

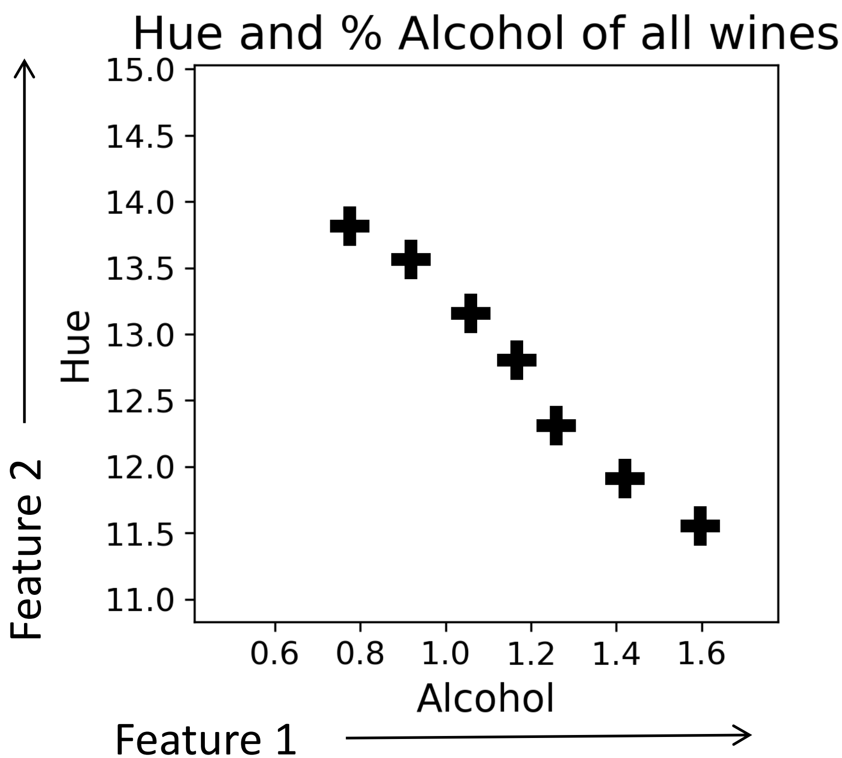
$\mathbb{R}^3$ three-dimensional space



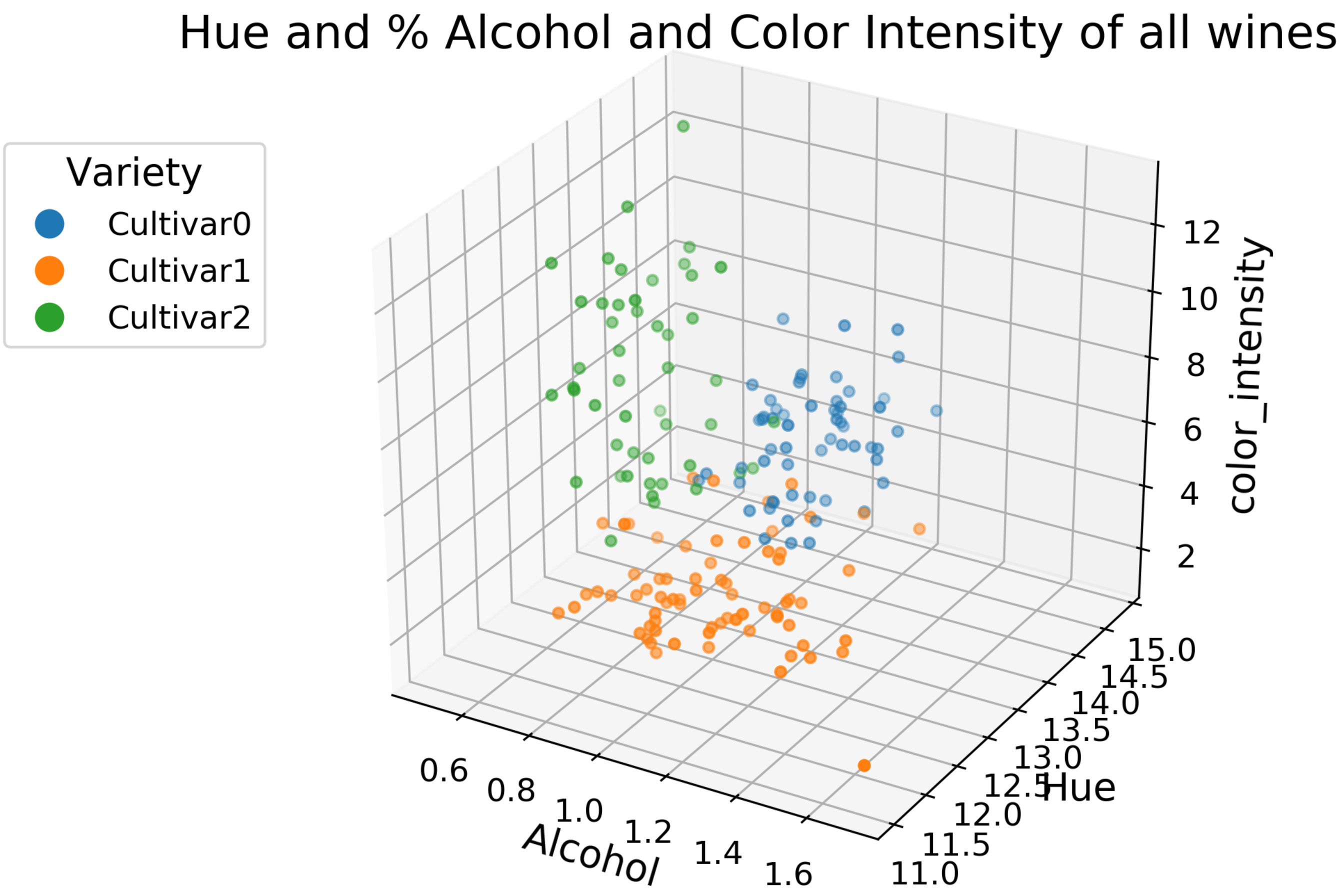
Features reveal structure



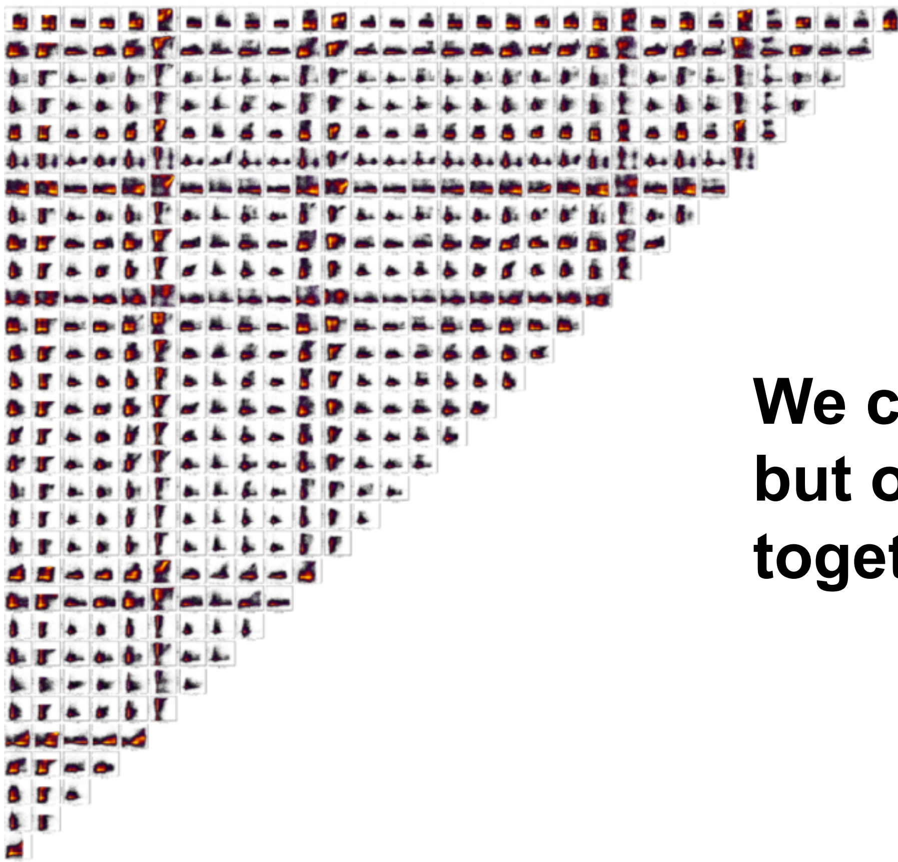
Many types of structure can exist in data



More Dimensions = More Accurate Structure



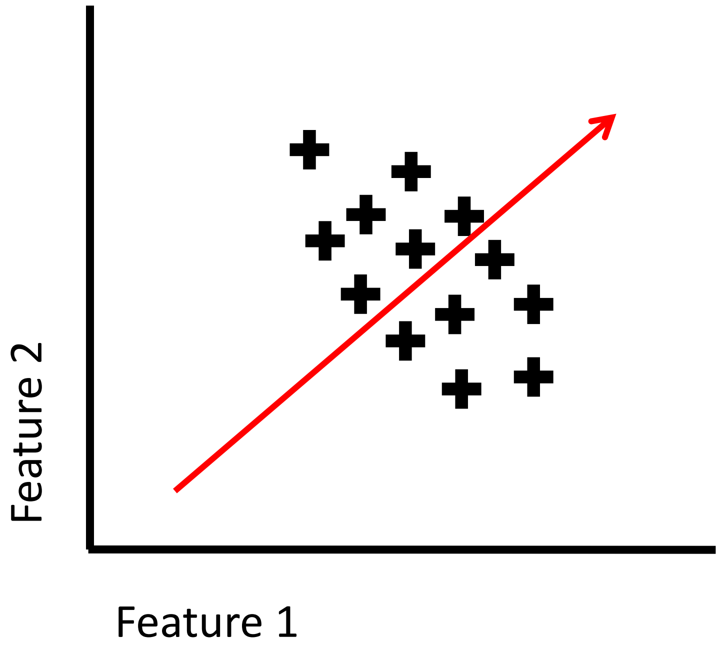
Problem: We can only see in 3D (not 20K D)

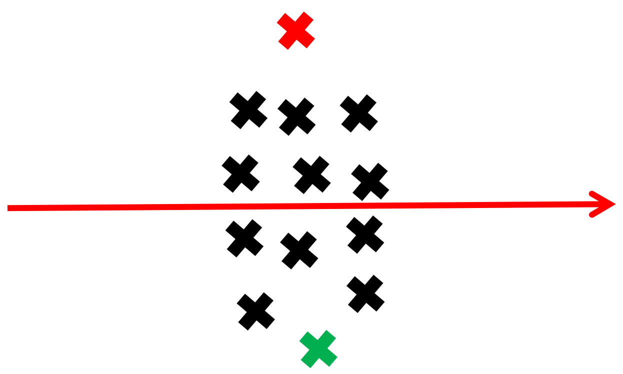


We could try all 2D pairs, but our brains can't put this together

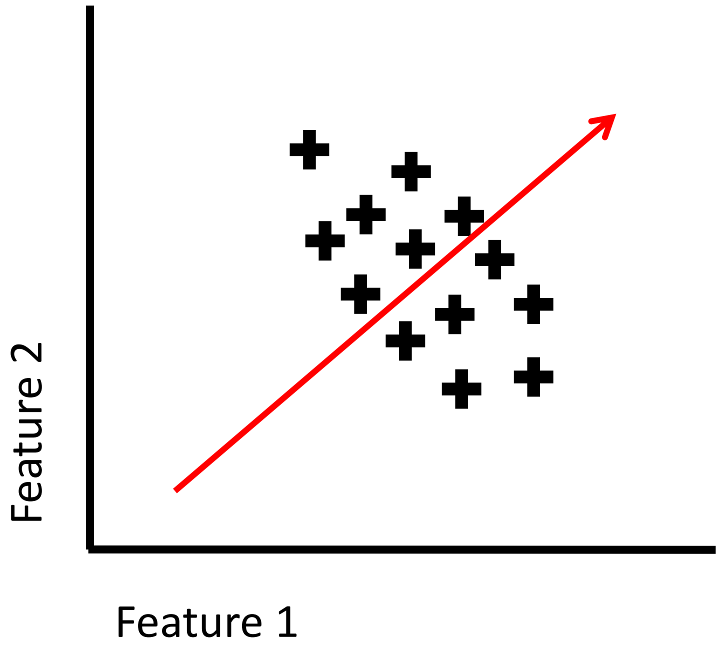
Solution: Dimensionality Reduction

Thought Exercise: Reduce this to 1D



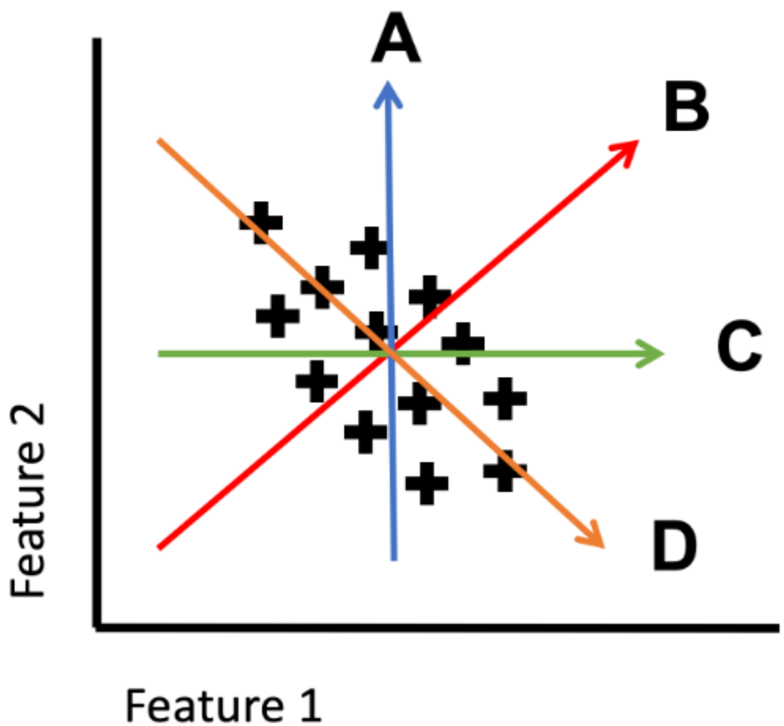


Thought Exercise: Reduce this to 1D



Most distant points on top of each other !

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



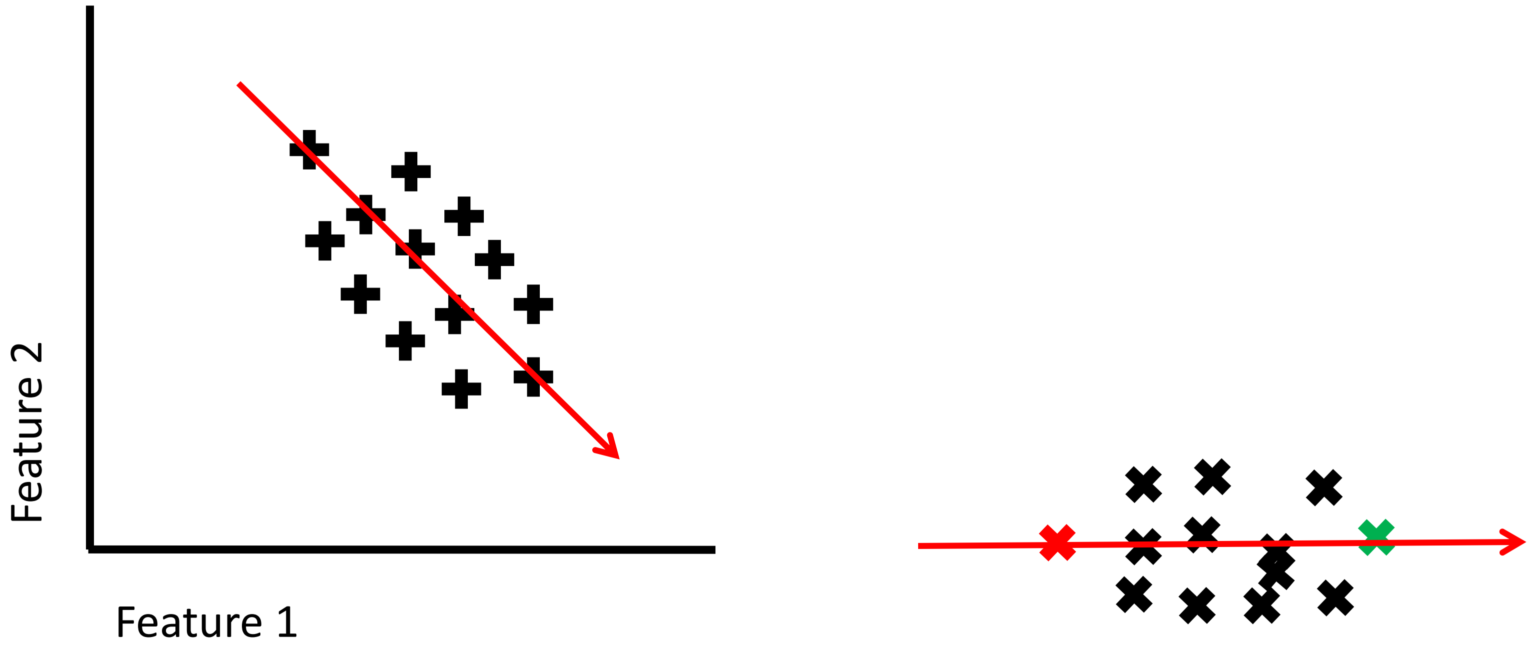
A

B

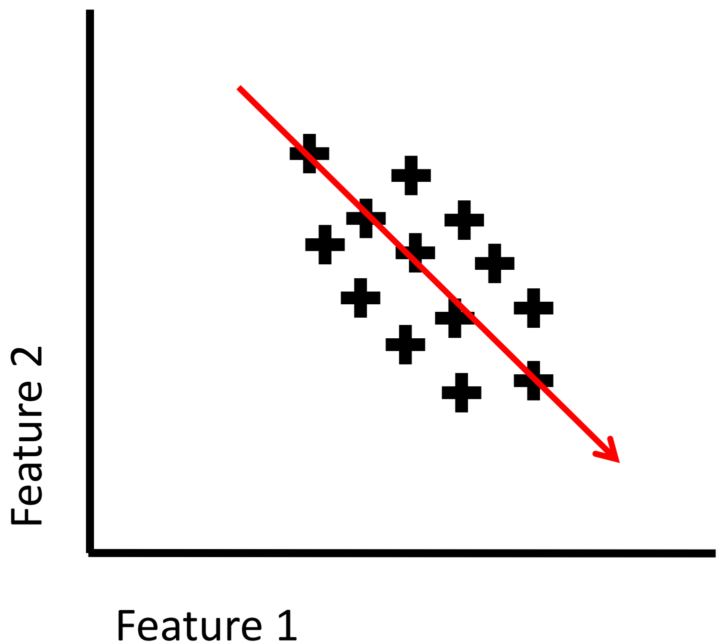
C

D

Thought Exercise: Reduce this to 1D



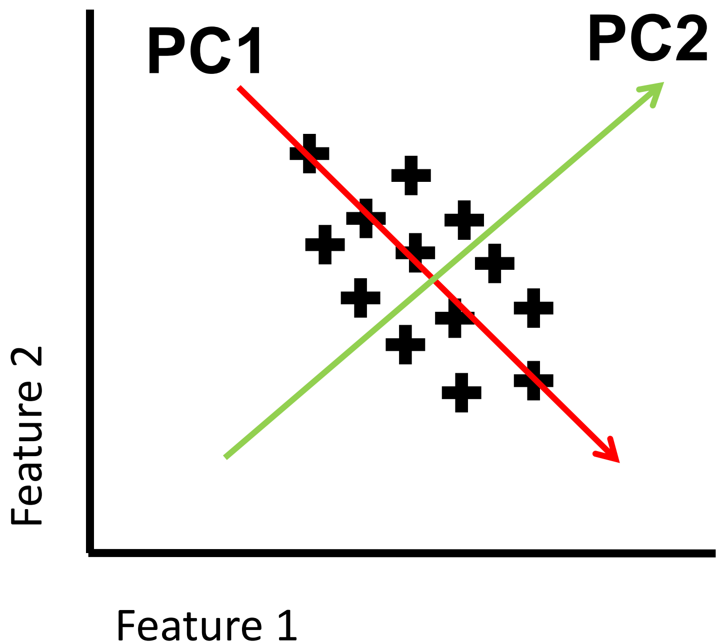
Thought Exercise: Reduce this to 1D



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

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

$$\mathrm{K}_{X_i X_j} = \operatorname{cov}[X_i, X_j] = \mathrm{E}[(X_i - \mathrm{E}[X_i])(X_j - \mathrm{E}[X_j])]$$

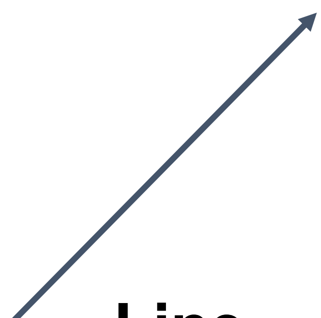
$$\mathrm{K}_{\mathbf{X}\mathbf{X}} = \begin{bmatrix} \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_n - \mathrm{E}[X_n])] \\ \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_1 - \mathrm{E}[X_1])]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

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

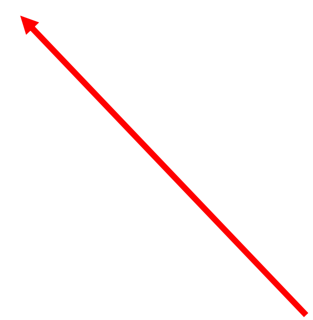
$$\begin{bmatrix} \cos(90) & -\sin(90) \\ \sin(90) & \cos(90) \end{bmatrix} \quad * $$

Rotation matrix



Line

=

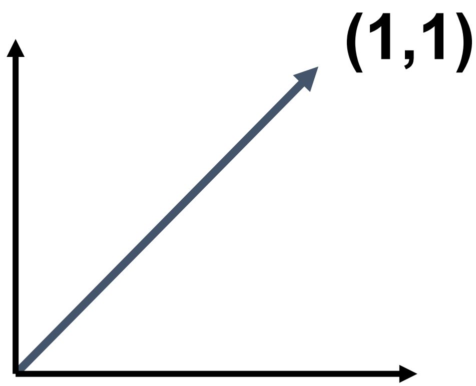


Rotated Line

Matrix–Vector Notation

- Lines can be described as vectors from the origin

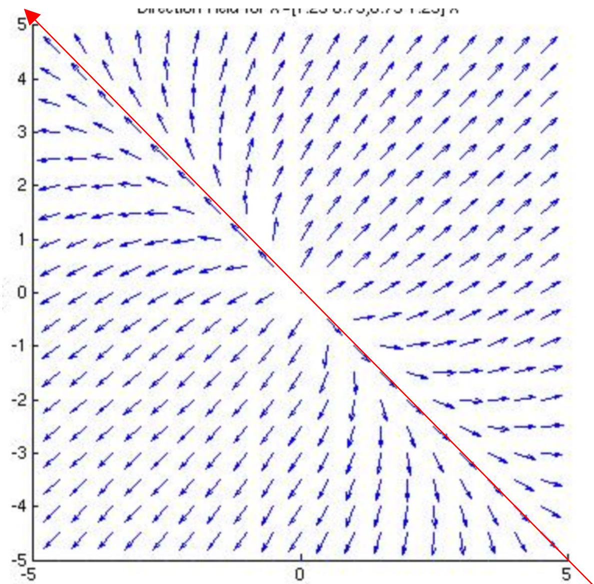
Line



$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

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

Eigenvectors



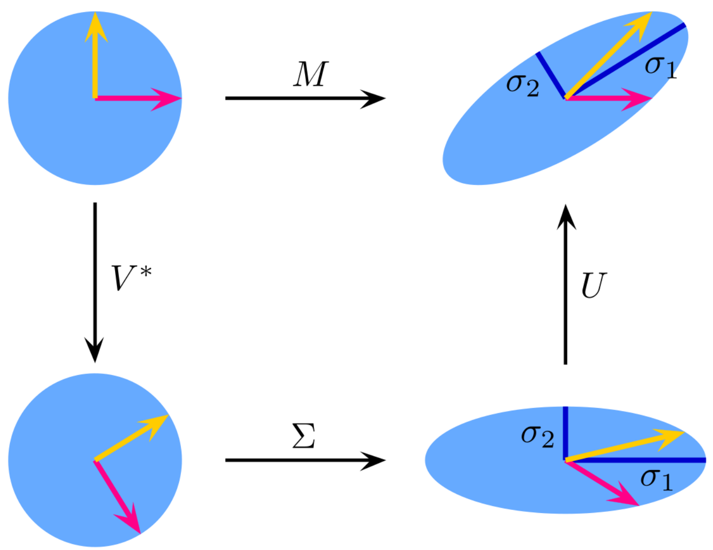
Rotation matrices rotate lines in only one direction

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

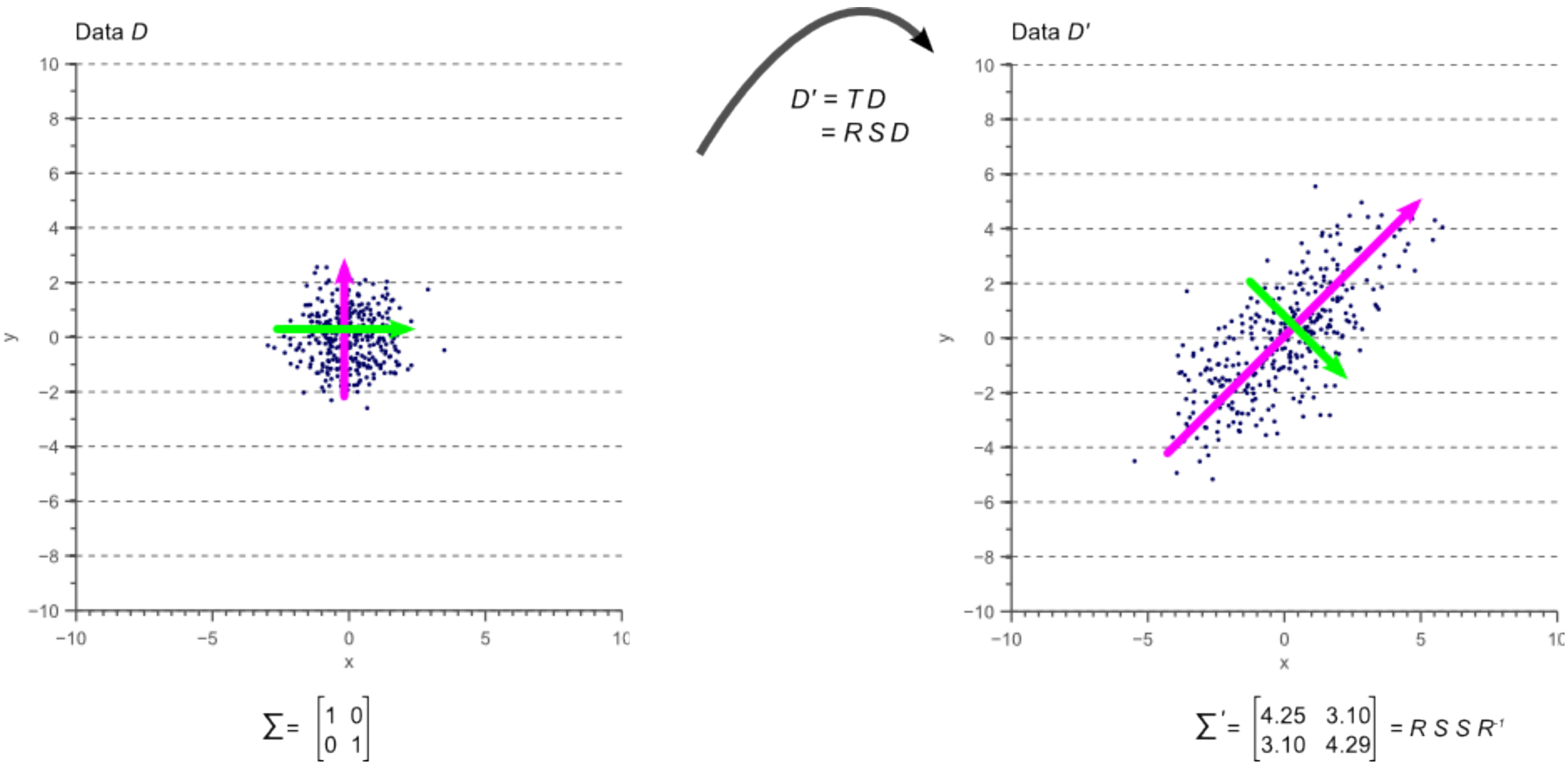
- Eigenvectors can be used to decompose a linear transformation into a rotation + stretch + anti-rotation



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

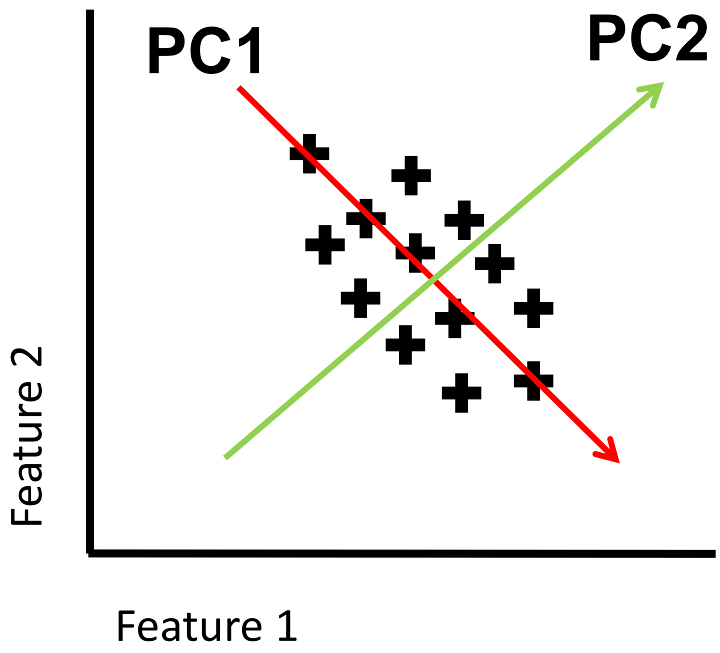
Non-square matrices have similar OP called **Singular value decomposition**

Eigenvectors of Covariance



Matrix creates correlation structure of data on unit blob

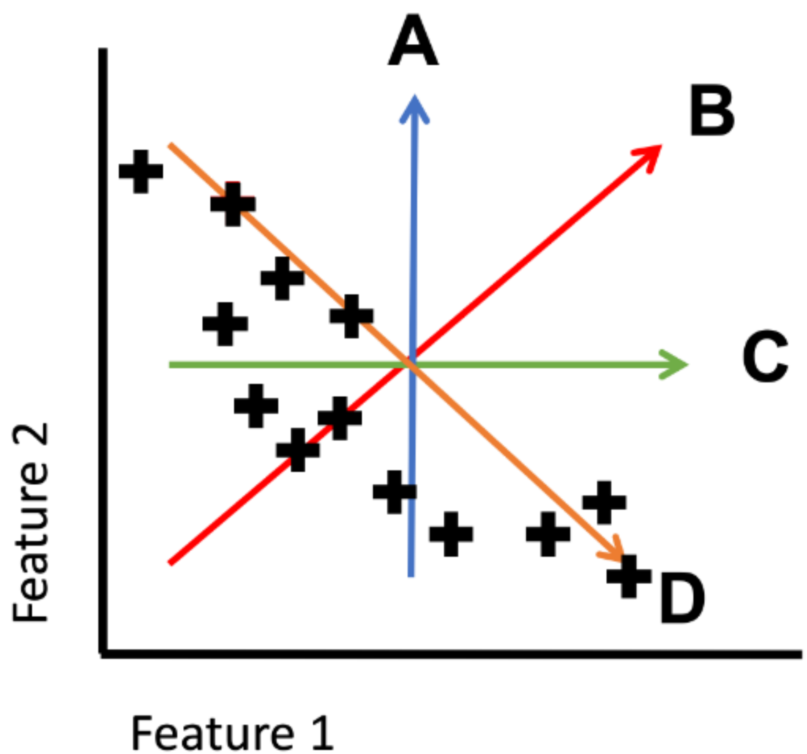
Principle Components Analysis



PC1 is first eigenvector (with Highest eigenvalue)

PC2 is next eigenvector

Which line would be PC1 for the following data?



A

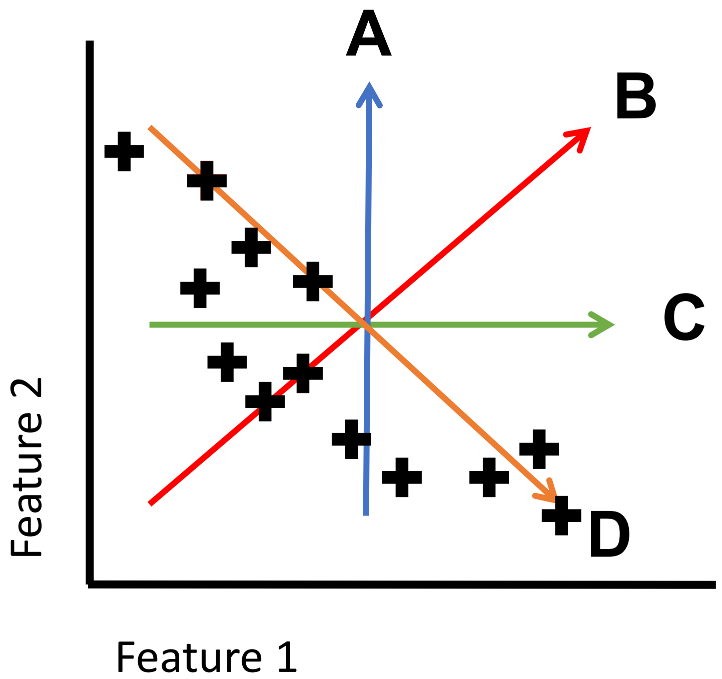
B

C

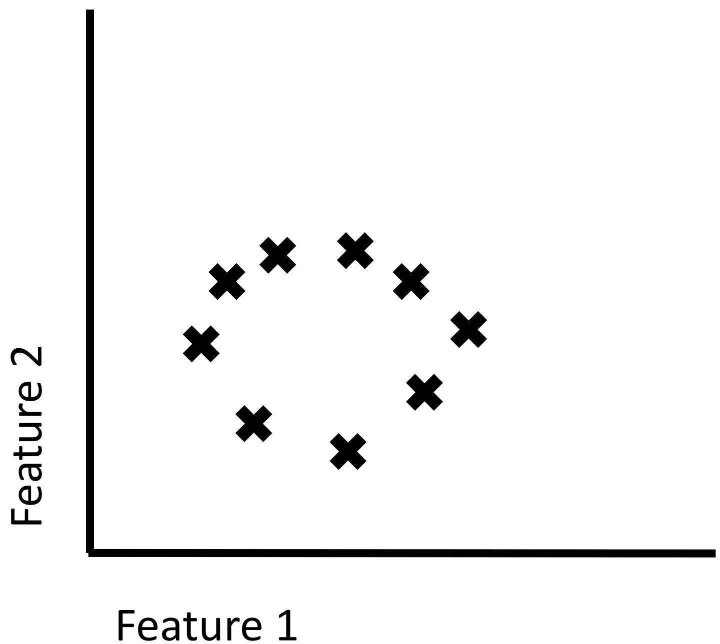
D

Quick quiz

Which line would be PC1 for the following data?



Non-linear structure



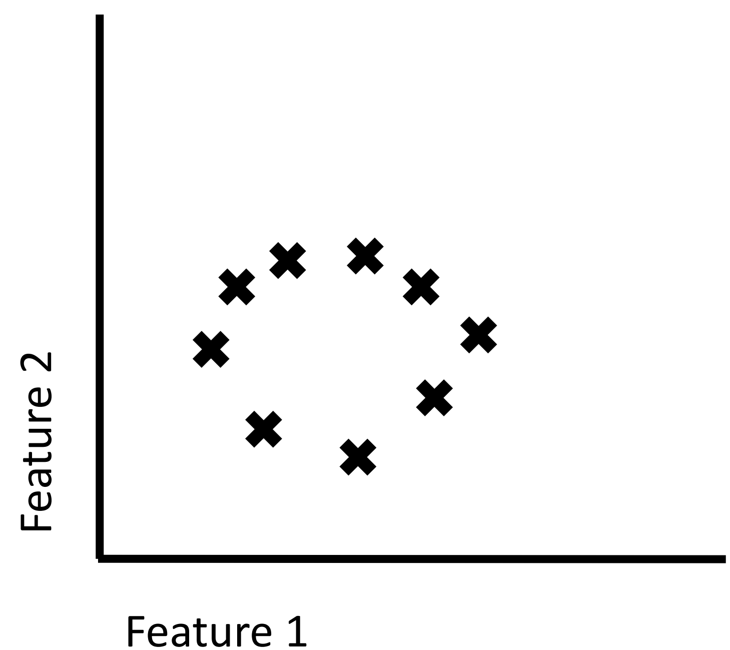
What if the data looked like this?

What line would be good
to project to here?

Non-linear dimensionality reduction

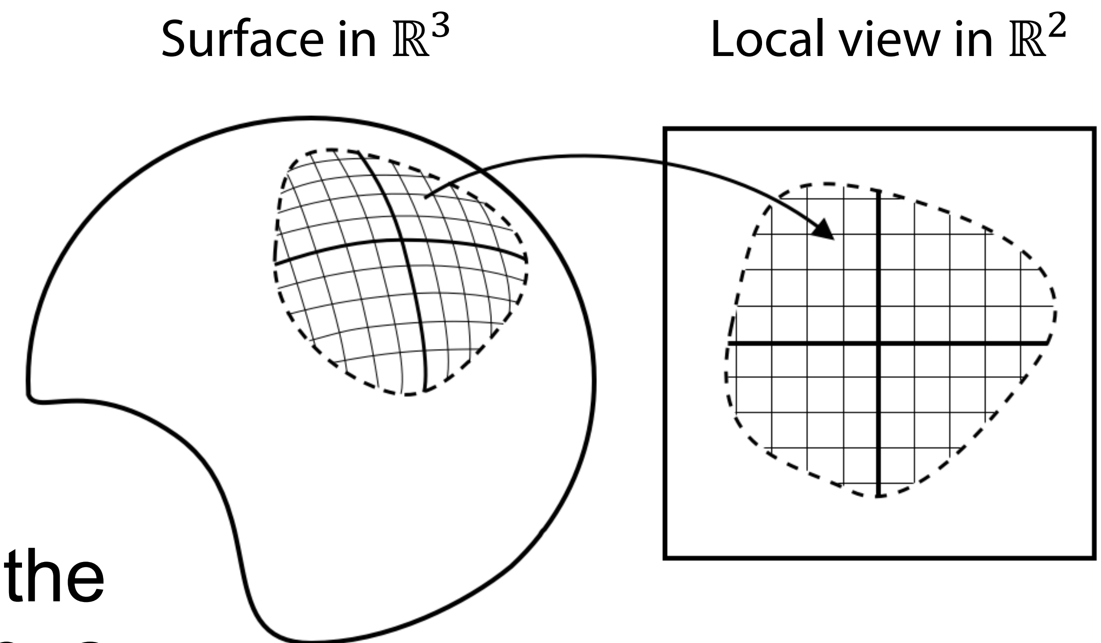
Non-linear dimensions

- 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"

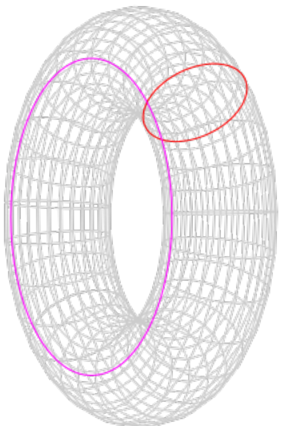


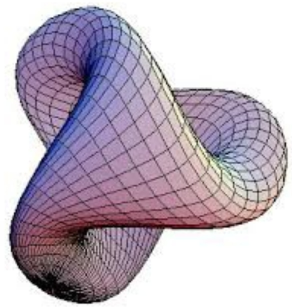
What is a manifold?

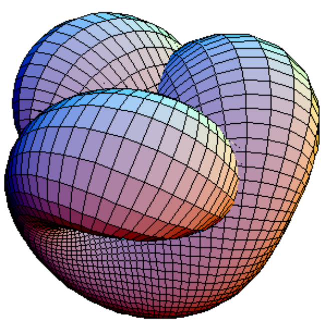
- Locally smooth
- Locally Euclidean
- Generally, lower dimensional than the ambient space (i.e. a subspace)

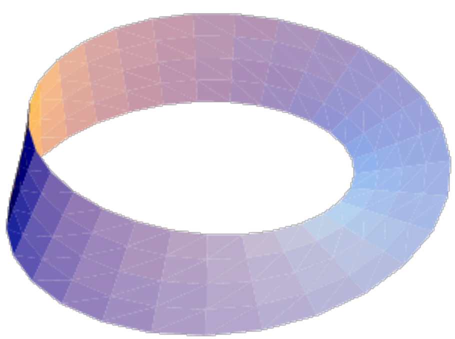


Examples of manifolds



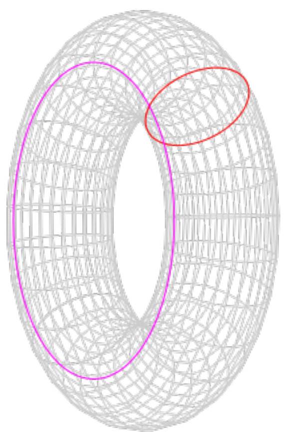


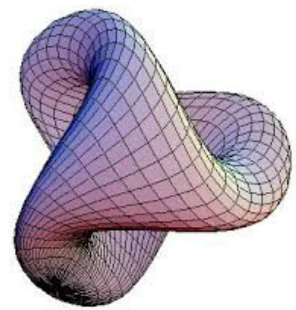


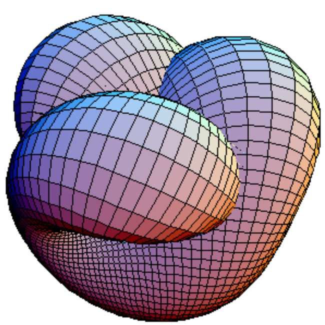


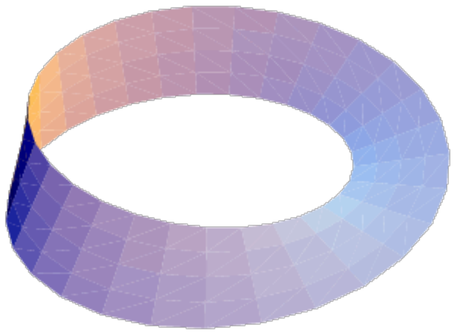
Wolfram

Examples of manifolds

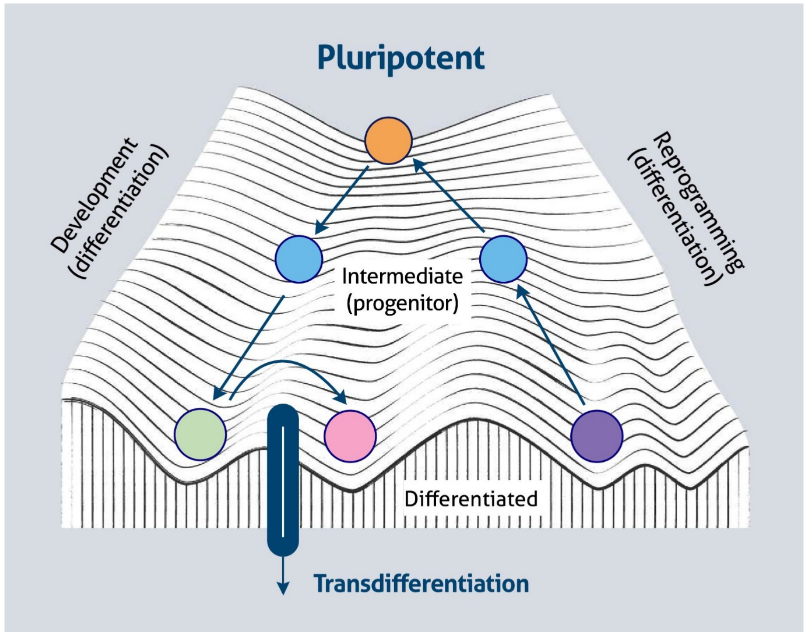








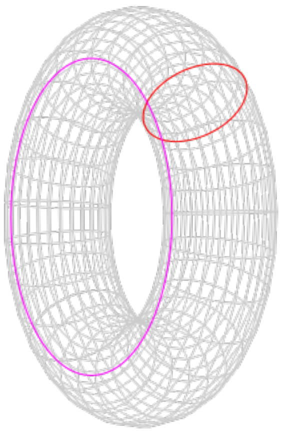
Waddington’s landscape

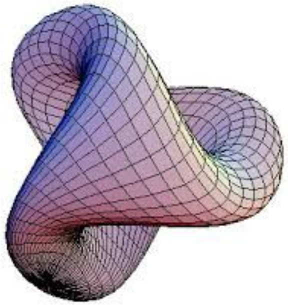


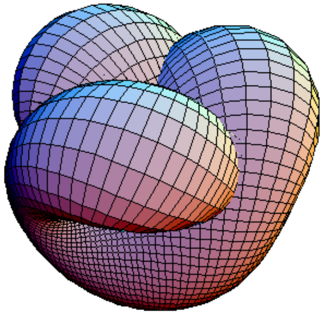
Wolfram

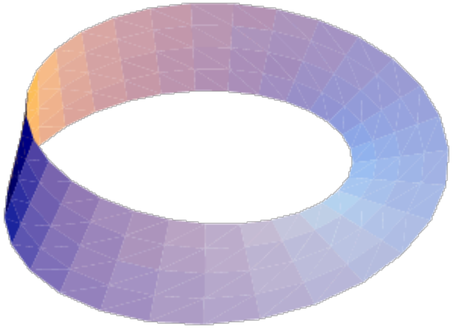
ptglab.com

Examples of manifolds

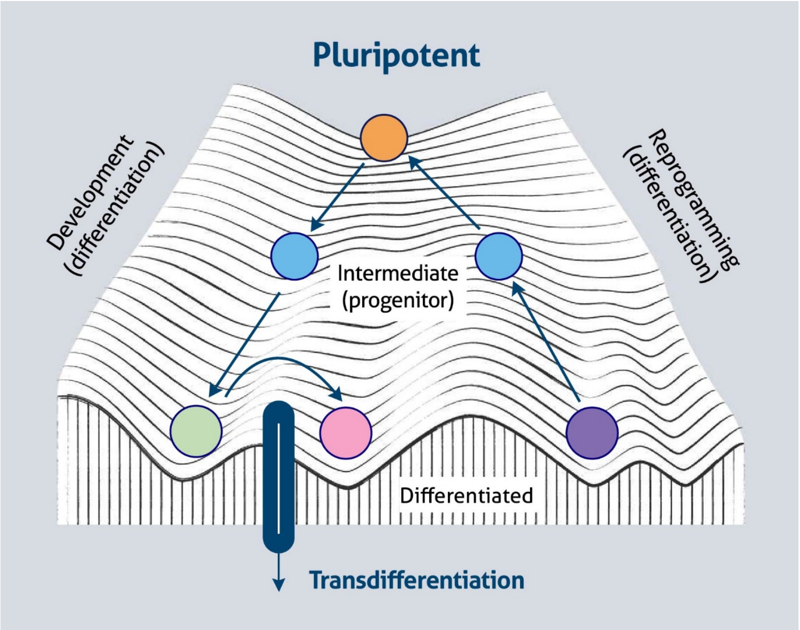








Waddington's landscape



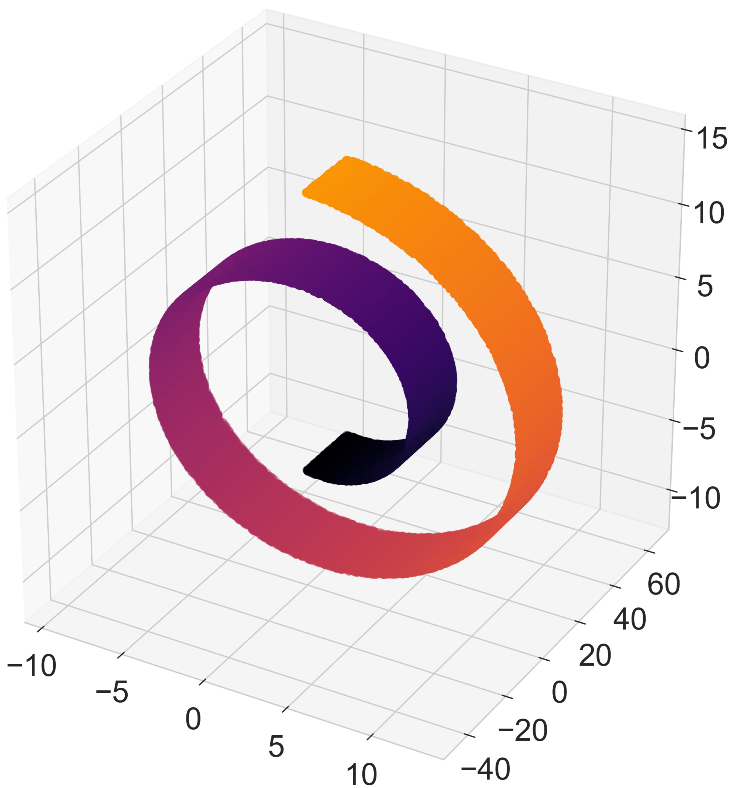
- Smooth transitions between states
- Small changes in gene expression are linear

Wolfram

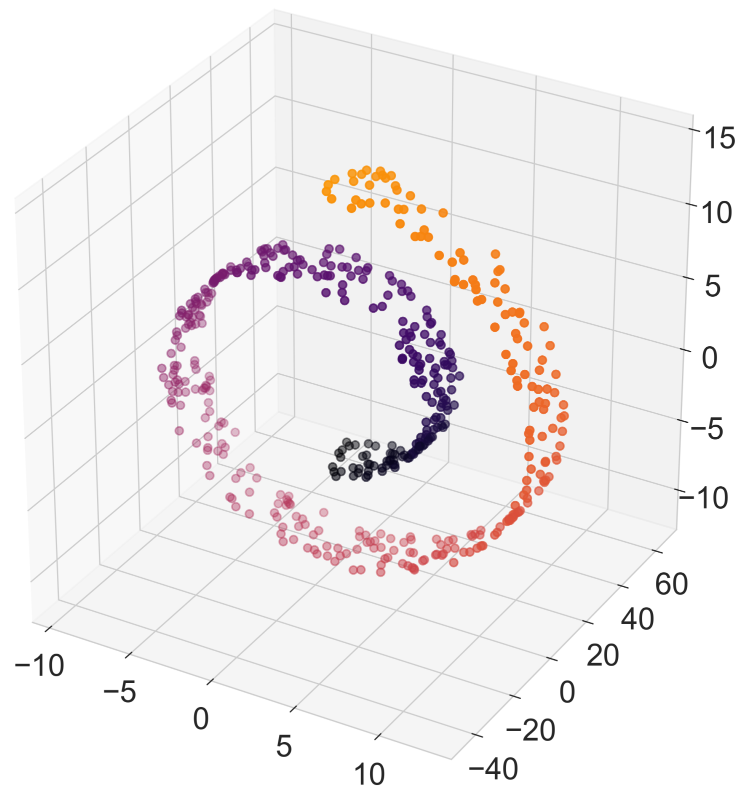
ptglab.com

The swiss roll is a plane ($\mathbb{R}^2$) embedded in $\mathbb{R}^3$

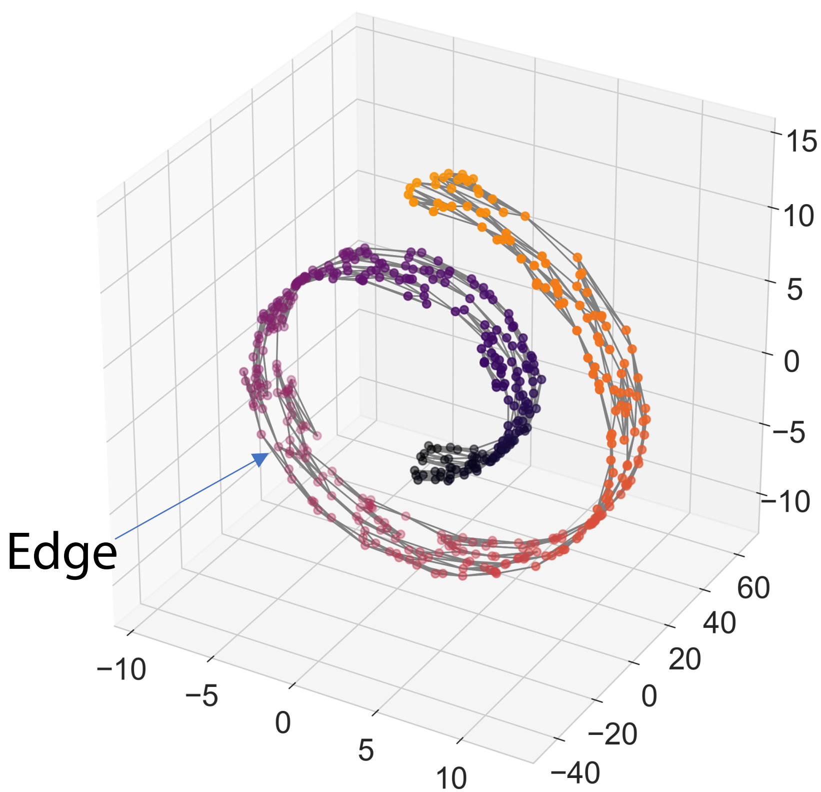
True manifold



Data sampled from manifold



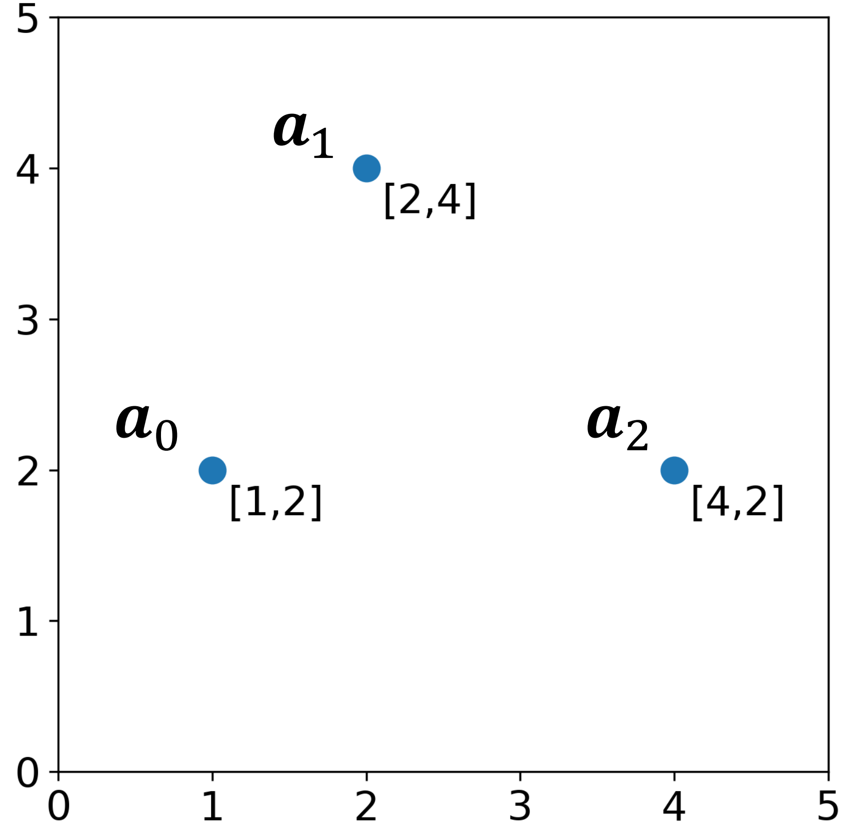
How do we find the data manifold?



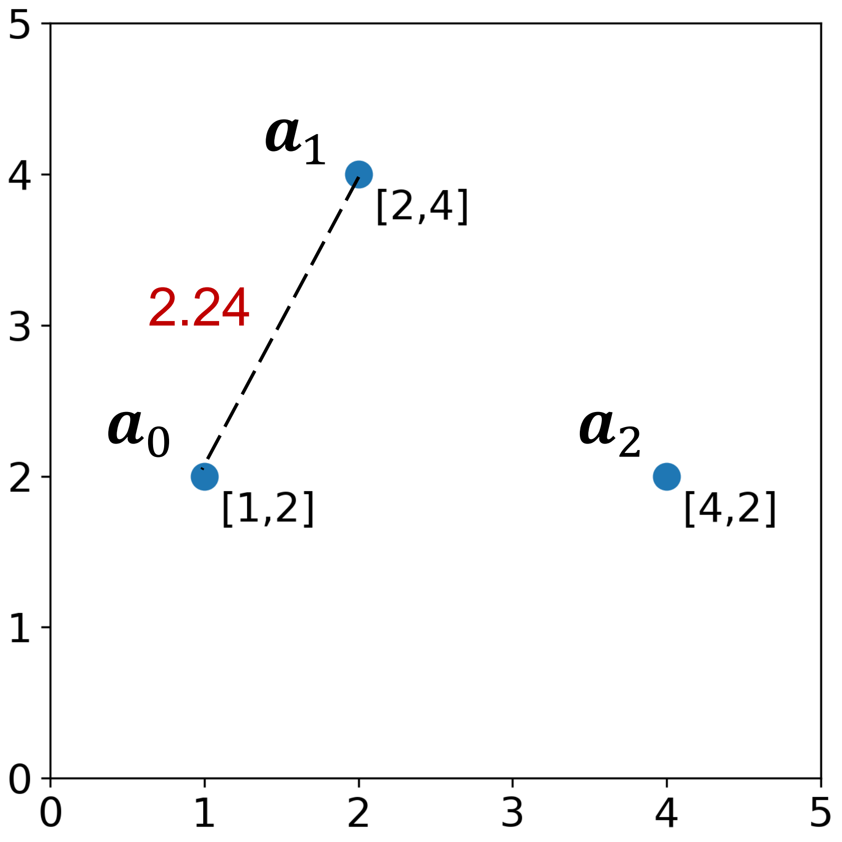
If you connect the nearest neighbors of any data point with a line (also called EDGE) it forms a mesh of the data that can be unrolled to find the shape.

This is called a nearest neighbors graph

How far away is each point from a_0?



Euclidean Distance



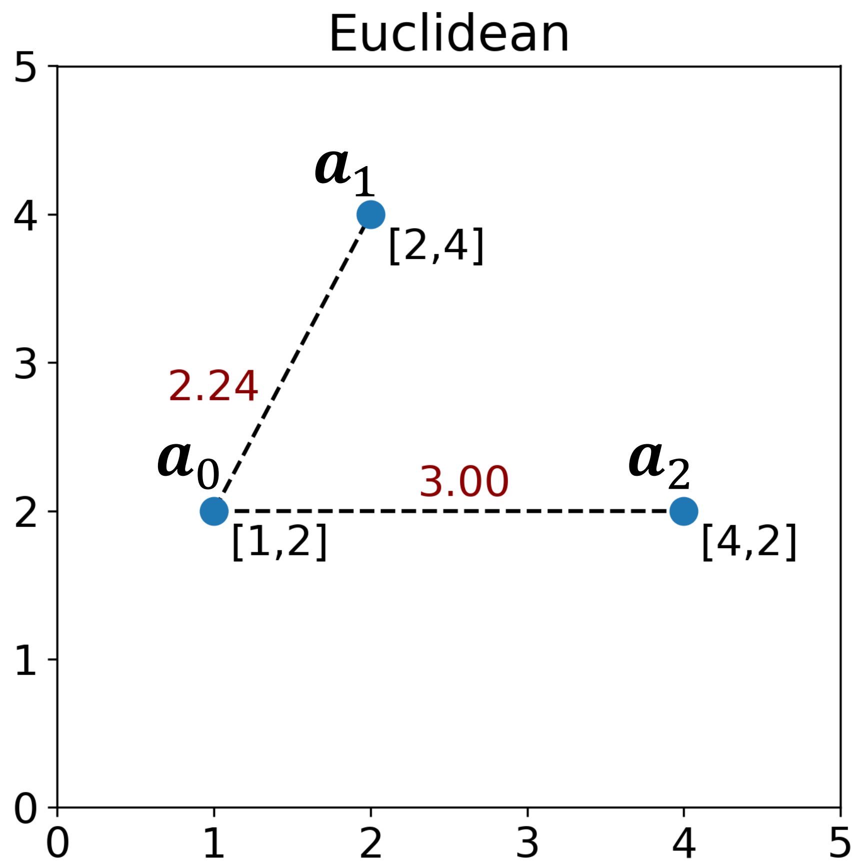
$$d_{euclidean}(a_0, a_1) = \|a_0 - a_1\|_2^2$$

$$= \sqrt{(a_{0,0} - a_{1,0})^2 + (a_{0,1} - a_{1,1})^2}$$

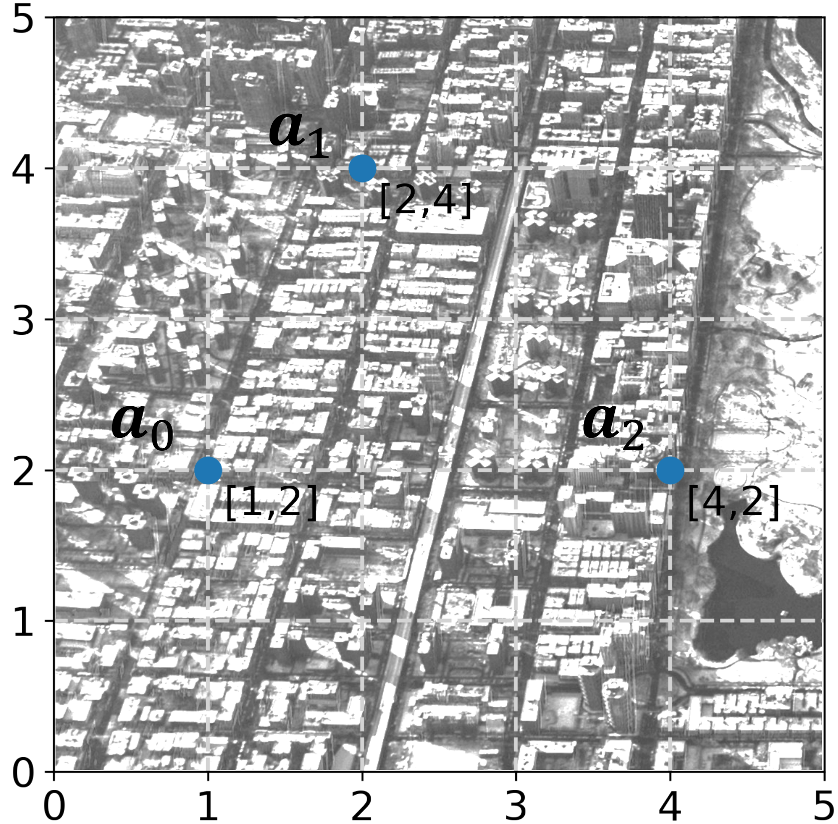
$$= \sqrt{(1 - 2)^2 + (2 - 4)^2}$$

$$\approx 2.24$$

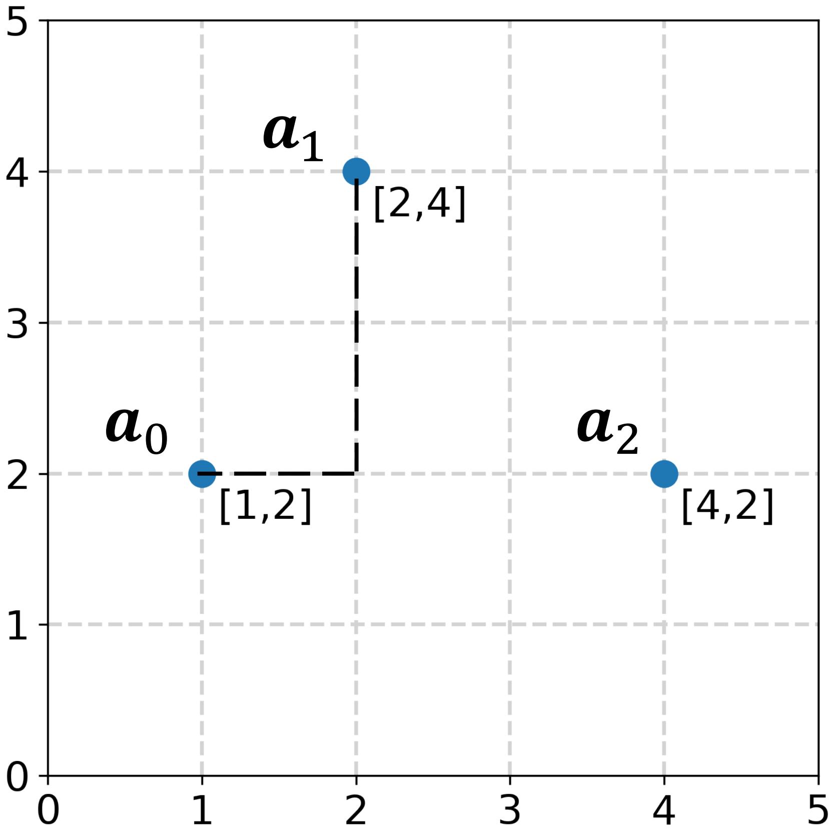
How far away is each point from a_0?



How far away is each point from a_0?



Manhattan Distance



Manhattan distance

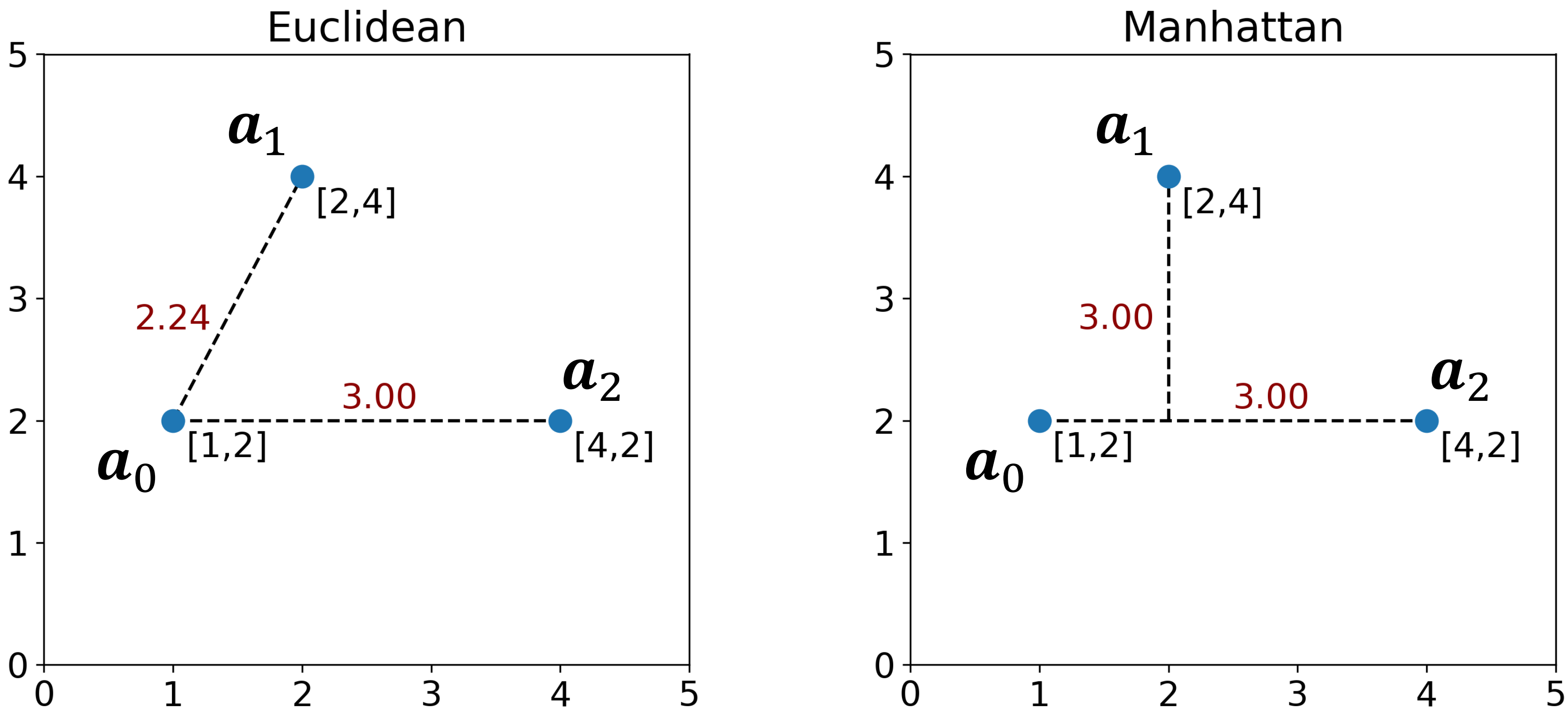
$$d_{manhattan}(a_0, a_1) = |a_{0,1} - a_{1,1}|$$

$$= |a_{0,0} - a_{1,0}| + |a_{0,1} - a_{1,1}|$$

$$= |1 - 2| + |2 - 4|$$

$$= 3$$

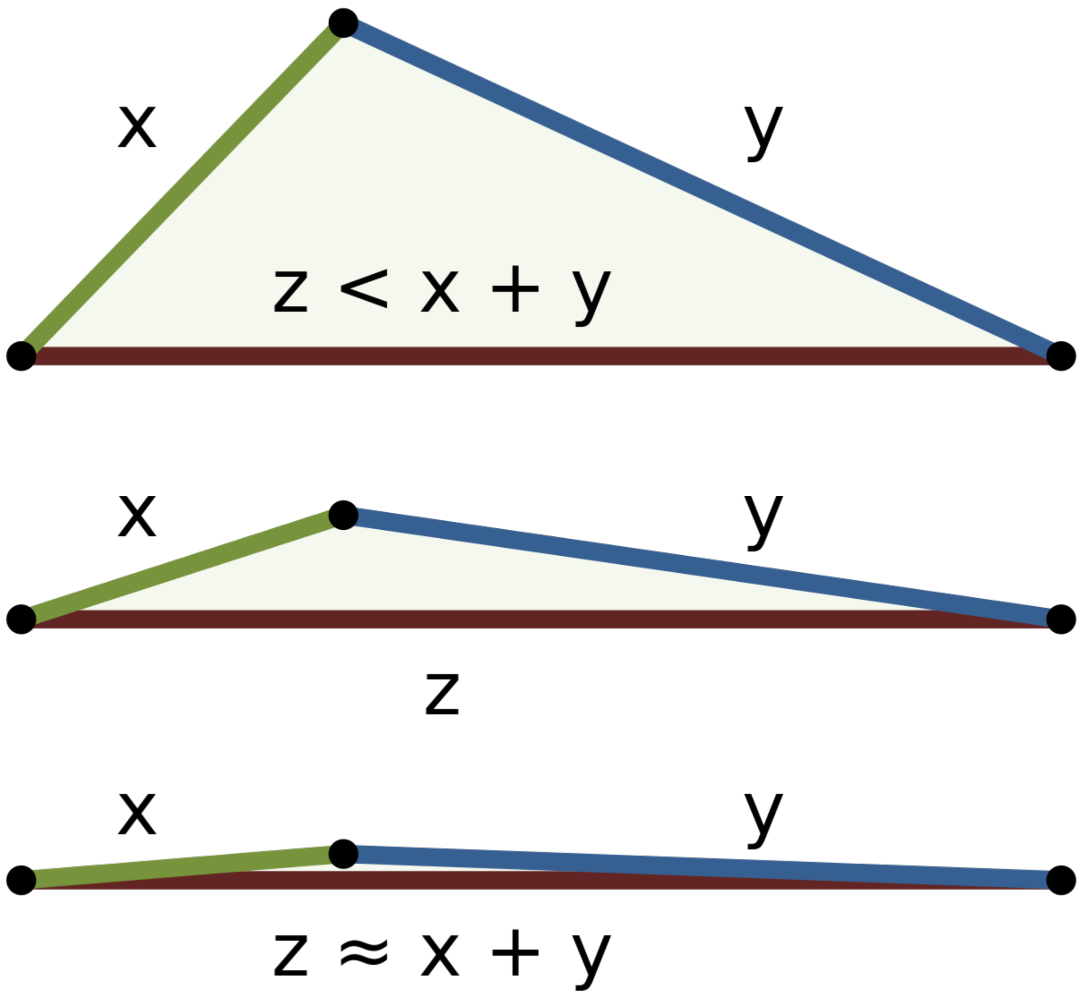
How far away is each point from a_0?



Distances

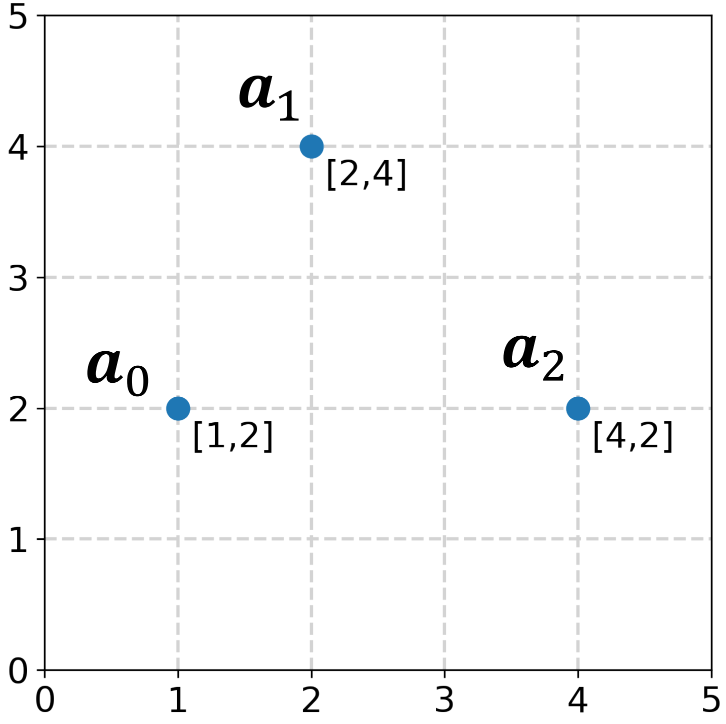
- There are many ways to measure distance
  - Hamming, Euclidean, Cosine
- Distances are functions that take two points and return a real number that is **positive or 0**
- Distances can be any function that is:
  - **Symmetric:** dist(a → b) = dist(b → a)
  - **Non-negative:** dist(a→b)>= 0
  - **Follow triangle inequality:** dist(a→c) <= dist(a→b) + dist(b→c)

Triangle Inequality

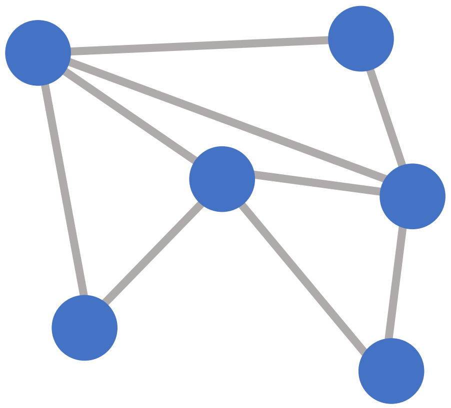


What is a "graph"?

Plot



Graph



$$G = (V, E)$$

= V = vertices / nodes

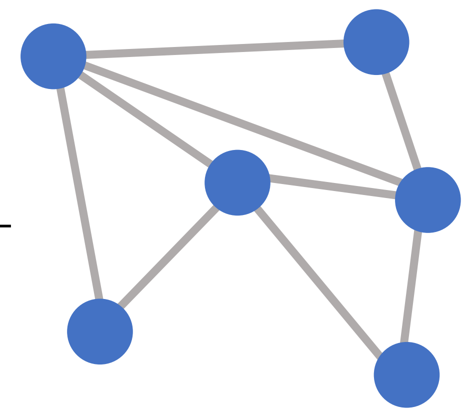
= E = edges

Provide an example of real-world data that could be modeled as a graph. What would be the vertices and edges?

Top

Example

Thing	Vertices	Edges
Internet	Computers	Network connections



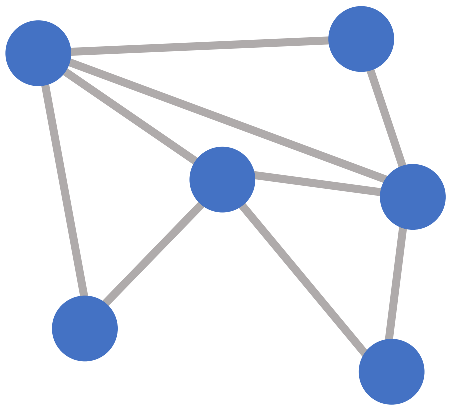
$G = (V, E)$ ● $= V =$ vertices — $= E =$ edges

What is a “graph”?

Things that can be modelled as a graph

Thing	Vertices	Edges
Internet	Computers	Network connections
Traffic	Intersections	Roads
Social network	People	Friendships
Cell similarities	Cells	Similarity relationships

Graph

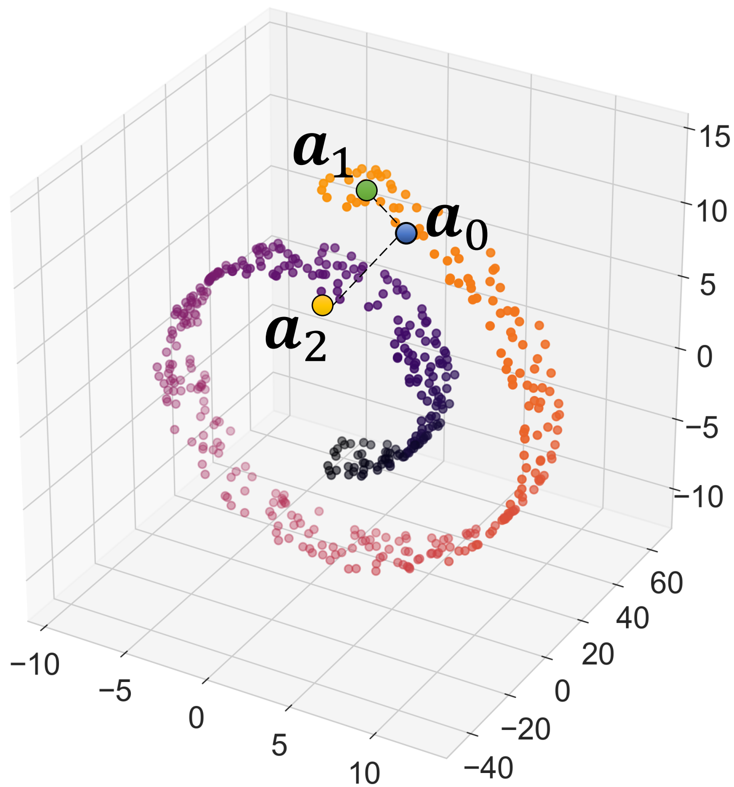


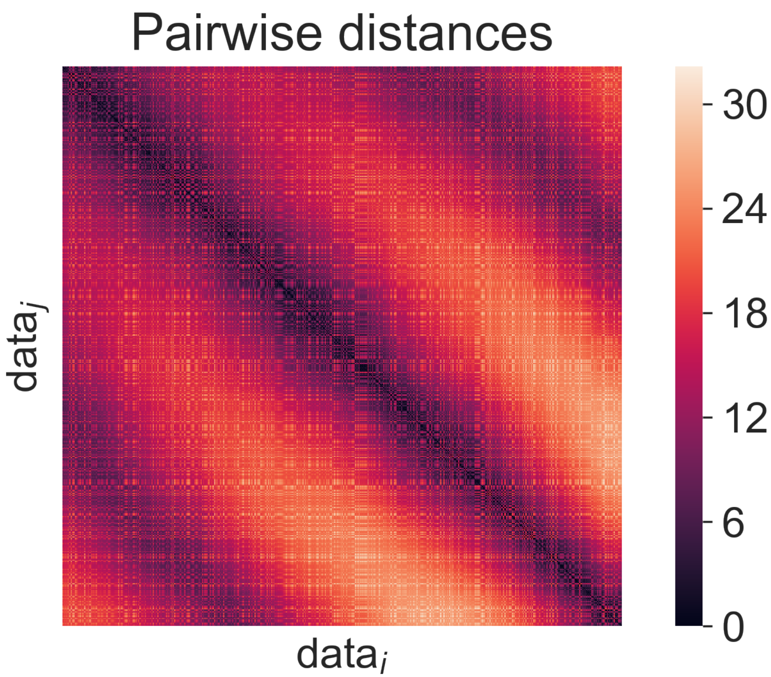
$$G = (V, E)$$

● $= V =$ vertices

— $= E =$ edges

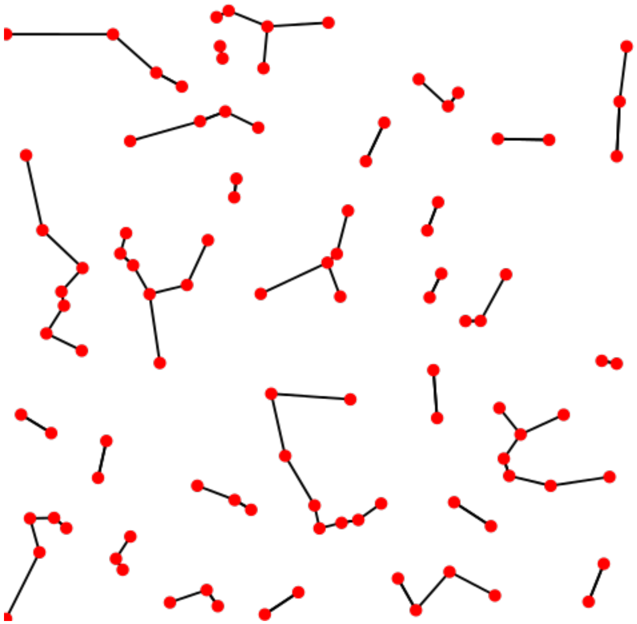
How do we quantify close (local) versus far (global) in a dataset?





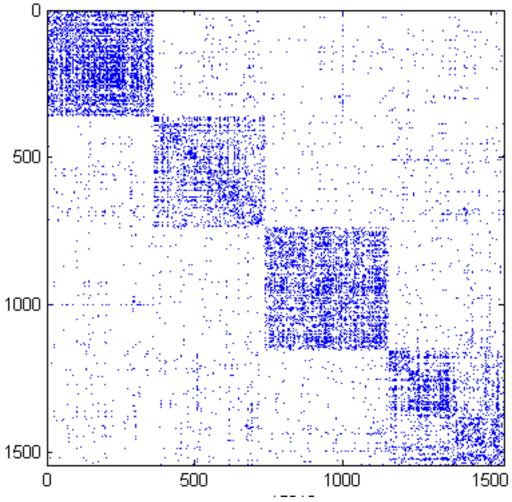
K Nearest Neighbors Graph

- To turn data into a graph, connect the K closest neighbors of a graph
- Measure closeness with your favorite distance

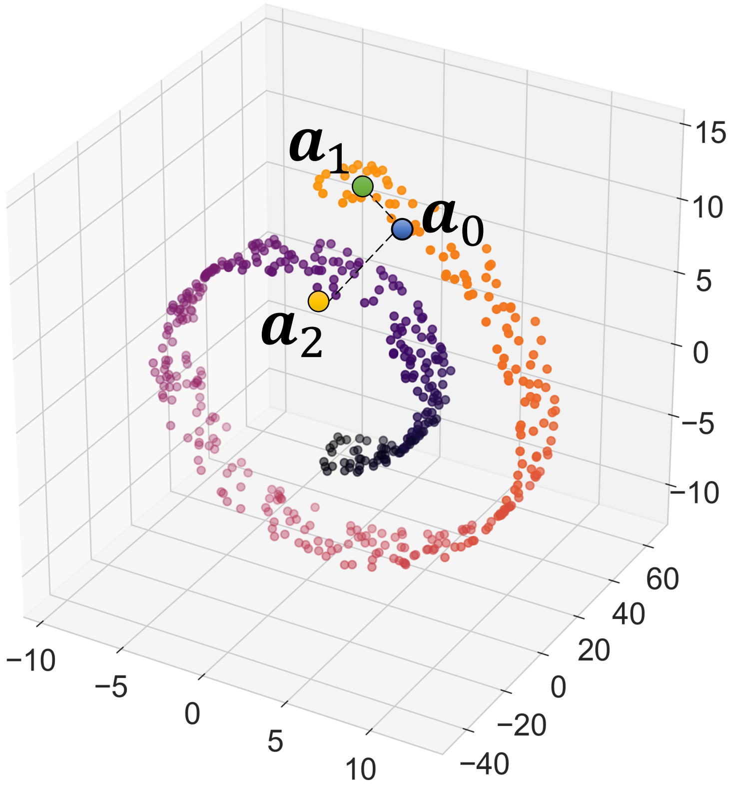


Affinity Matrix as Graph

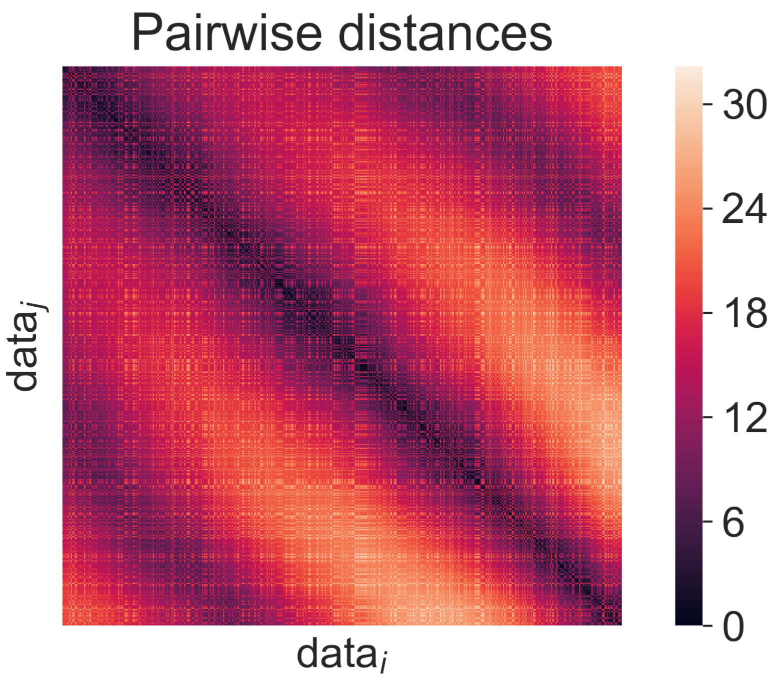
- Fully connected graph with edges weighted by an affinity (negatively proportional to distance)



Swiss Roll

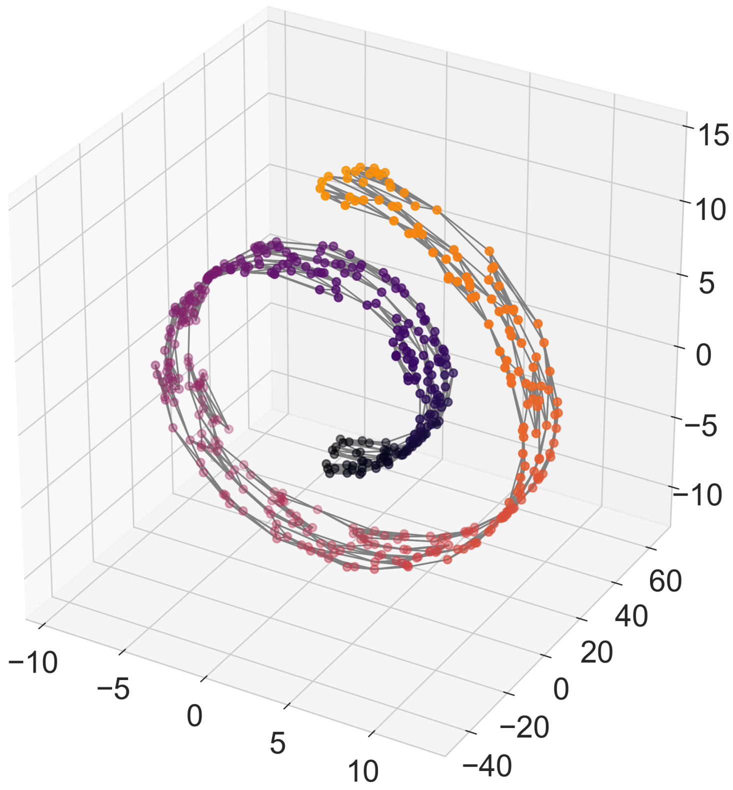


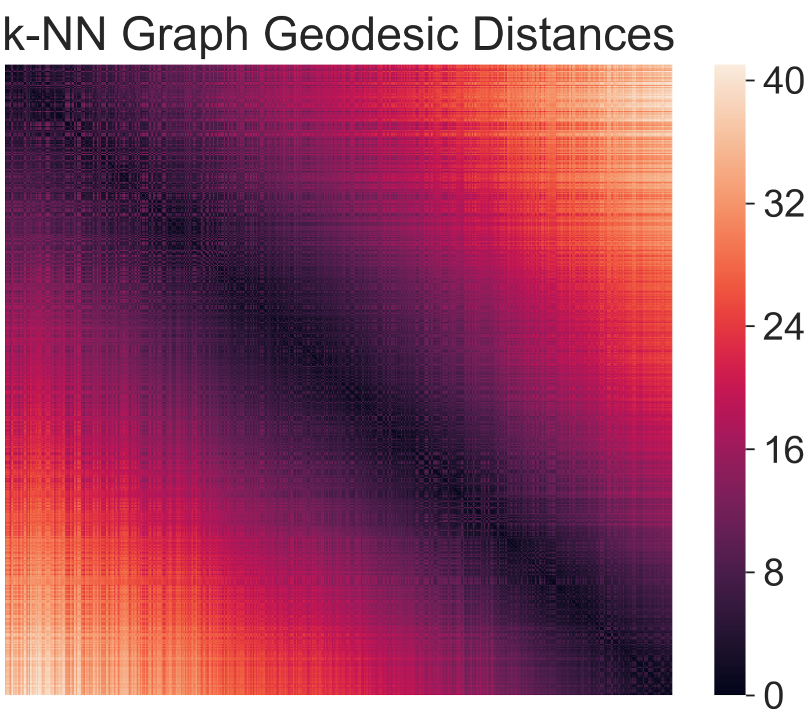
a_1
a_0
a_2
15
10
5
0
−5
−10
60
40
20
0
−20
−40
−10
−5
0
5
10



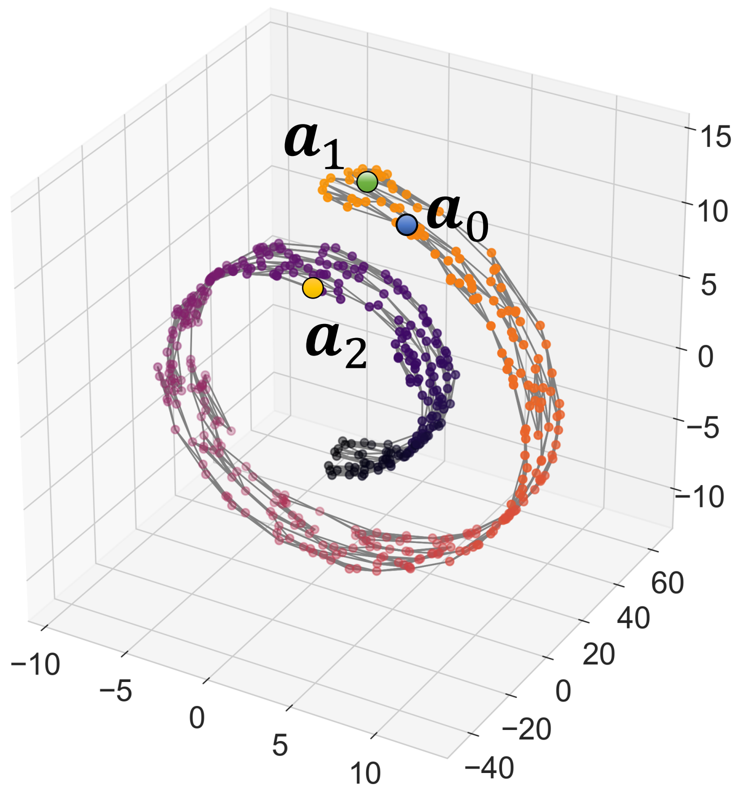
Pairwise distances
data_j
data_i
30
24
18
12
6
0

Swiss Roll K-NN

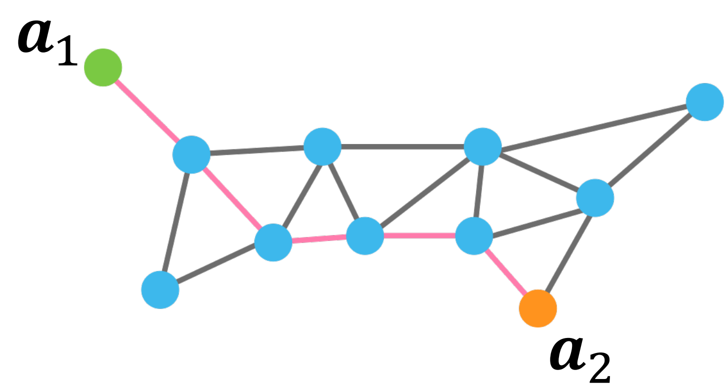




Paths or “walks”on graphs reveal dominant data manifold directions

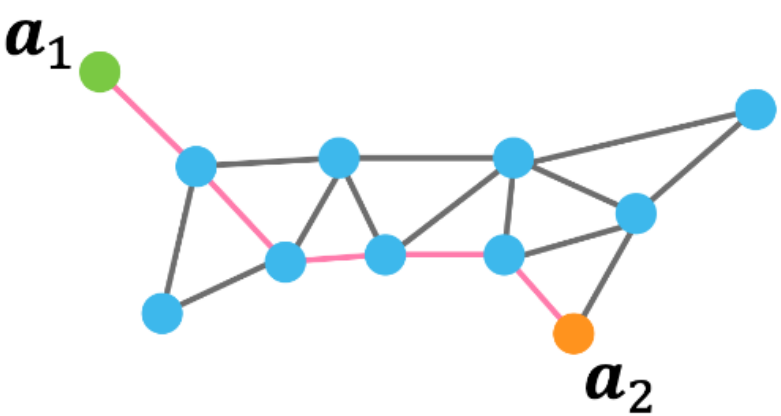


Shortest path between points



What is the shortest path distance between a1 and a2?

Shortest path between points



4

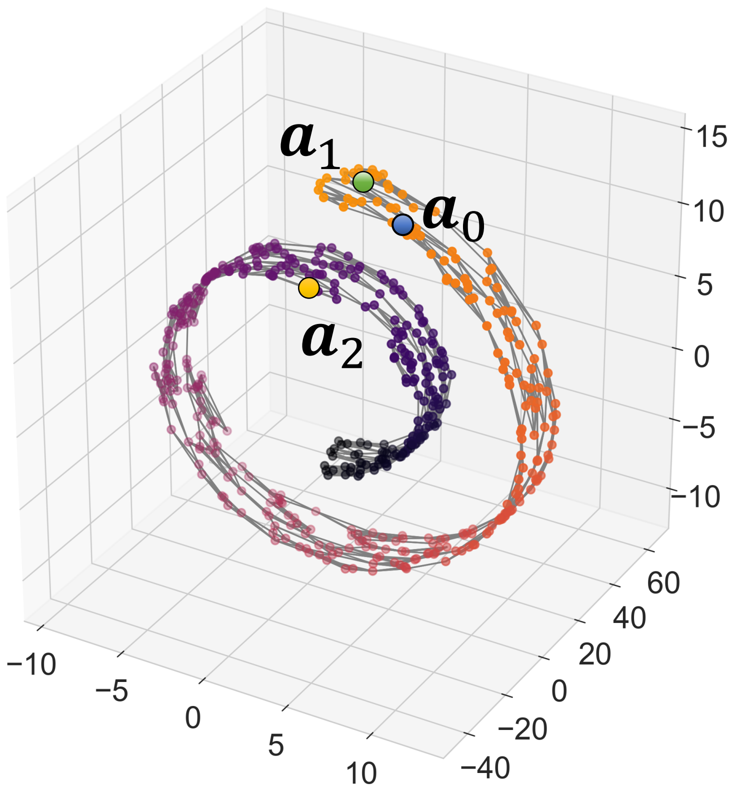
5

6

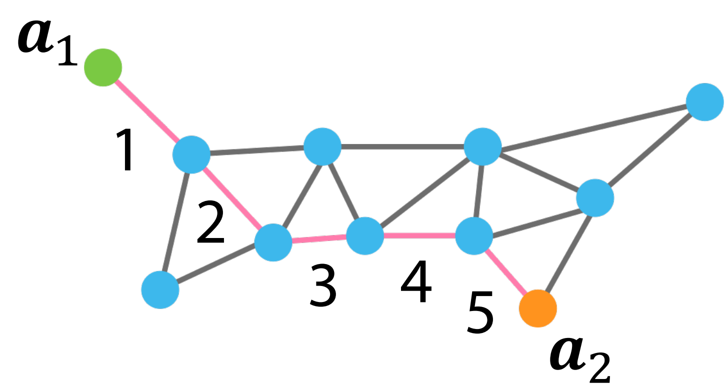
7

8

Graph walks approximate manifold distances

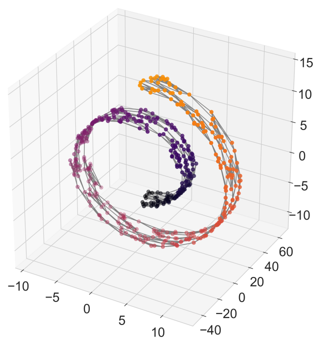


Shortest path between points

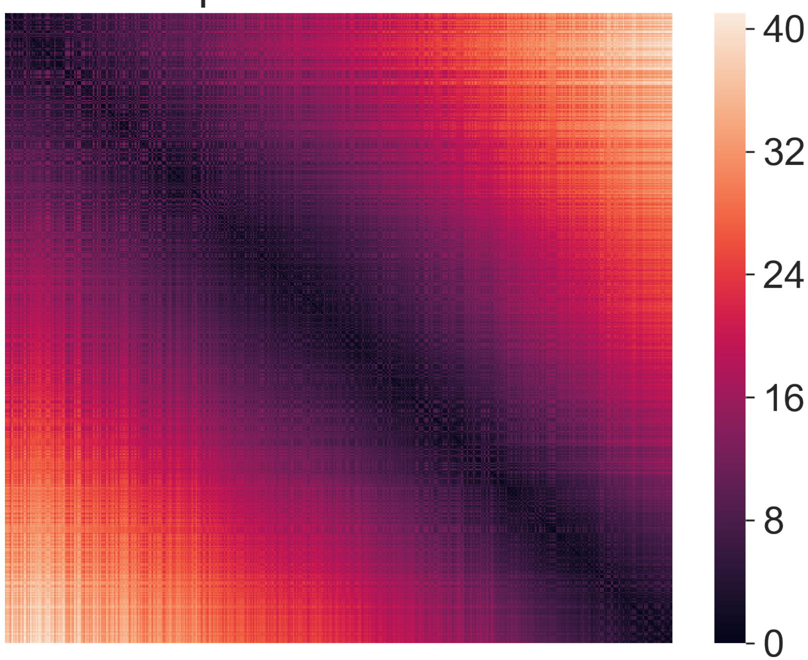


$$d_{geodesic}(a_1, a_2) = \text{shortestpath}(a_1, a_2)$$
$$= 5$$

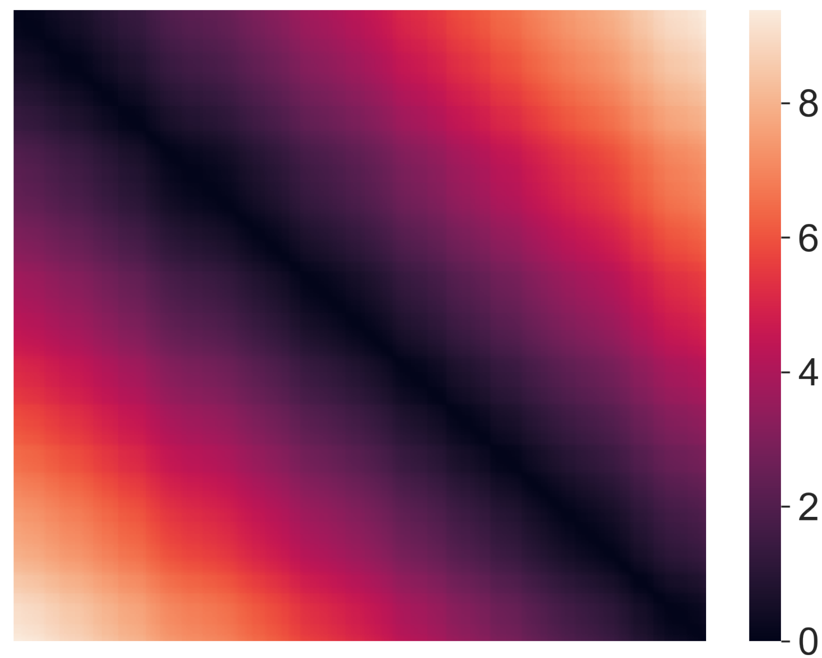
Graph walks approximate manifold distances



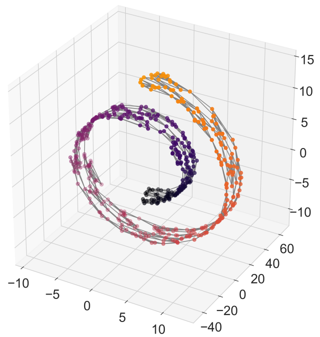
k-NN Graph Geodesic Distances

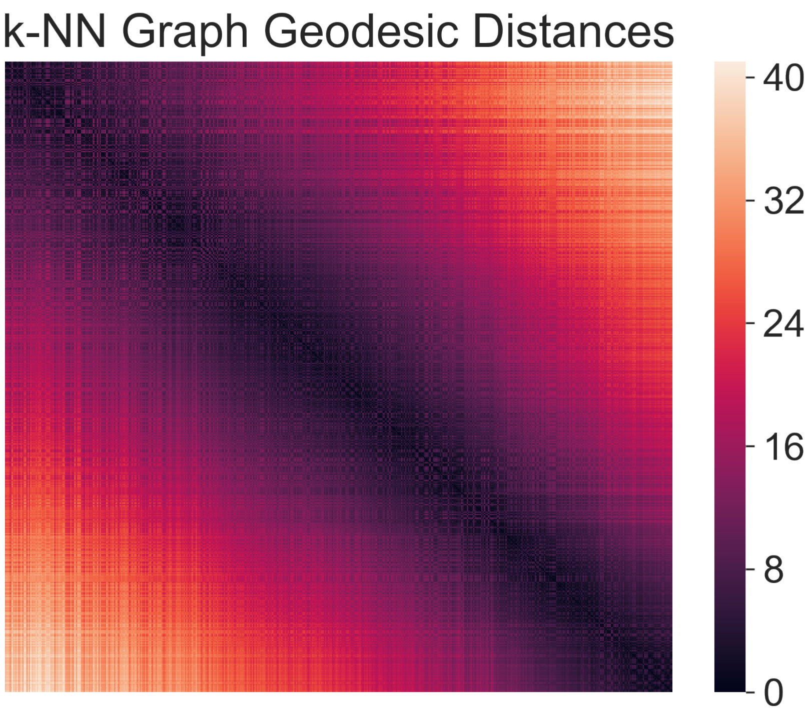


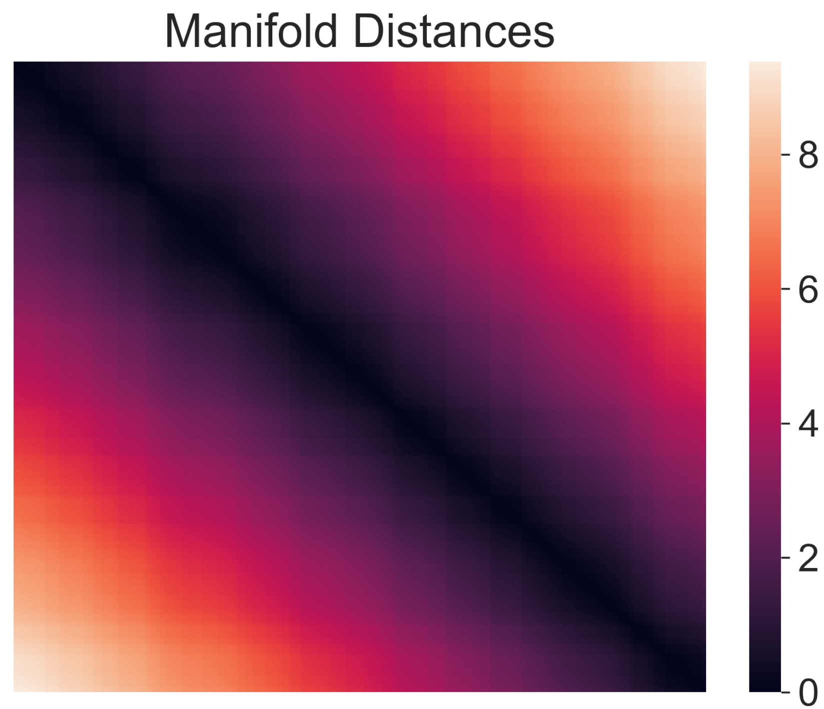
Manifold Distances



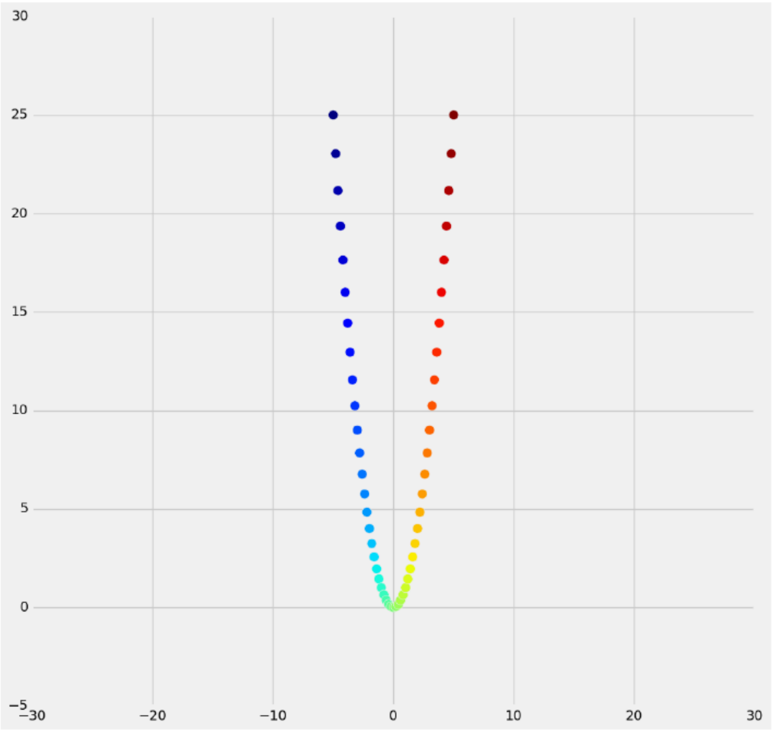
Graph walks approximate manifold distances

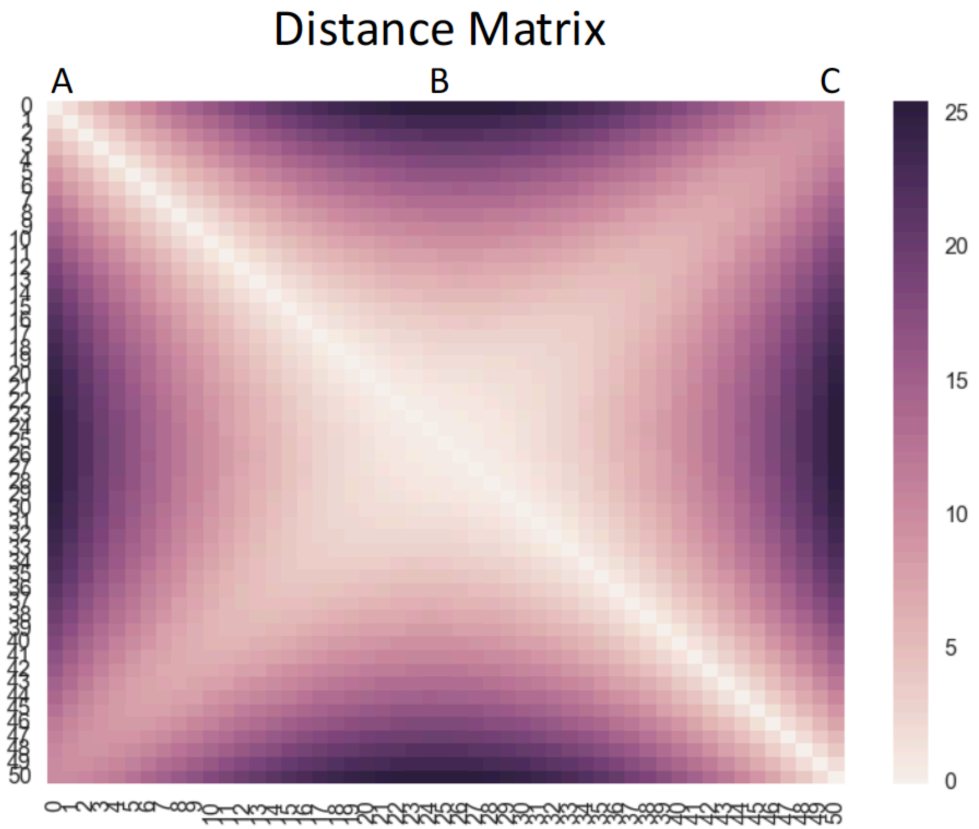




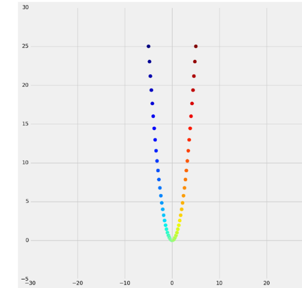


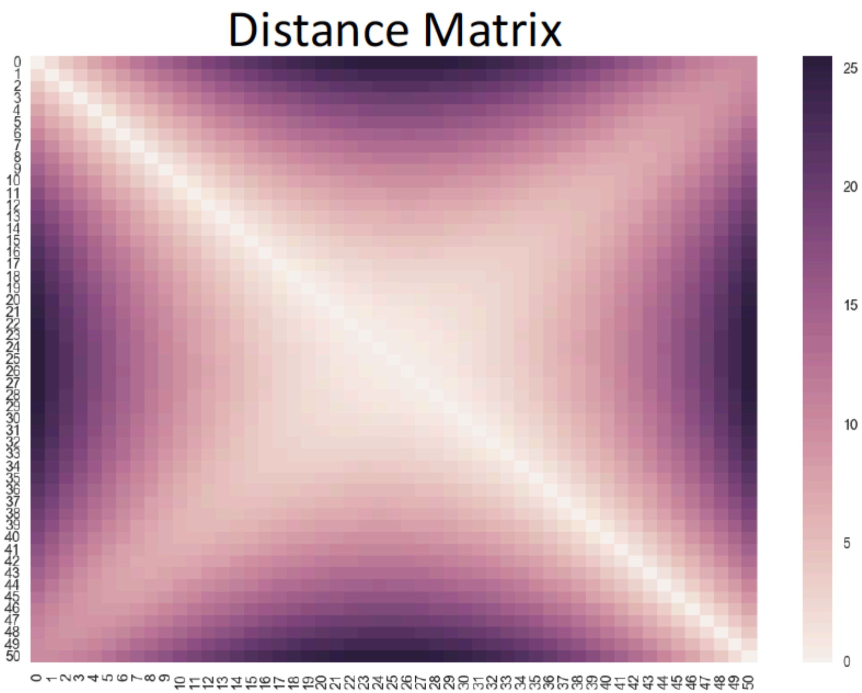
Kernel functions measure “similarity” or “affinity” on a graph

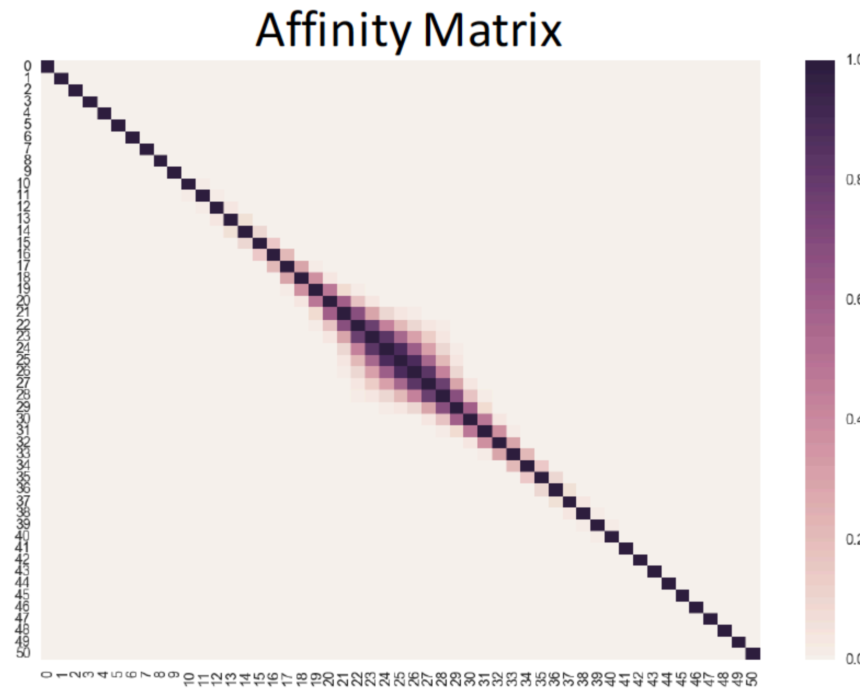




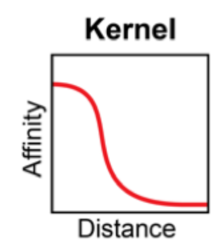
Affinity is the inverse of distance + locality



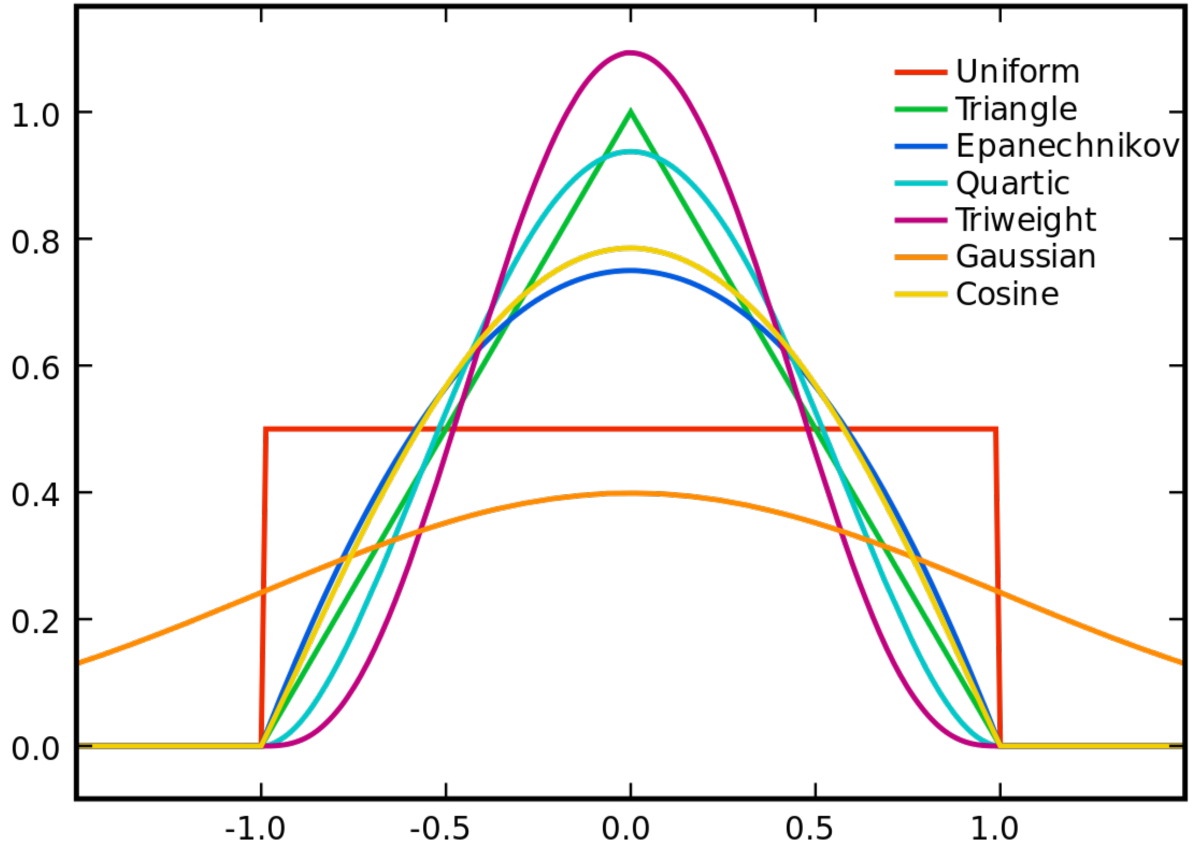




$$Affinity_{i,j} = s_{i,j} = exp\left(-\frac{dist\left(x_i, x_j\right)^2}{2\sigma^2}\right)$$



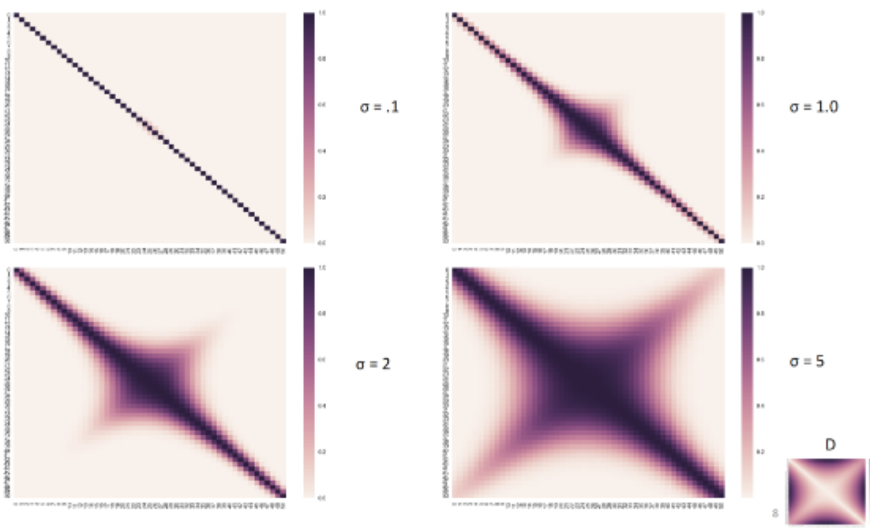
Distance to Affinity via Kernels



Real-valued
Non-negative
Symmetric

Affinities correlations in this hidden hypothetical space

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



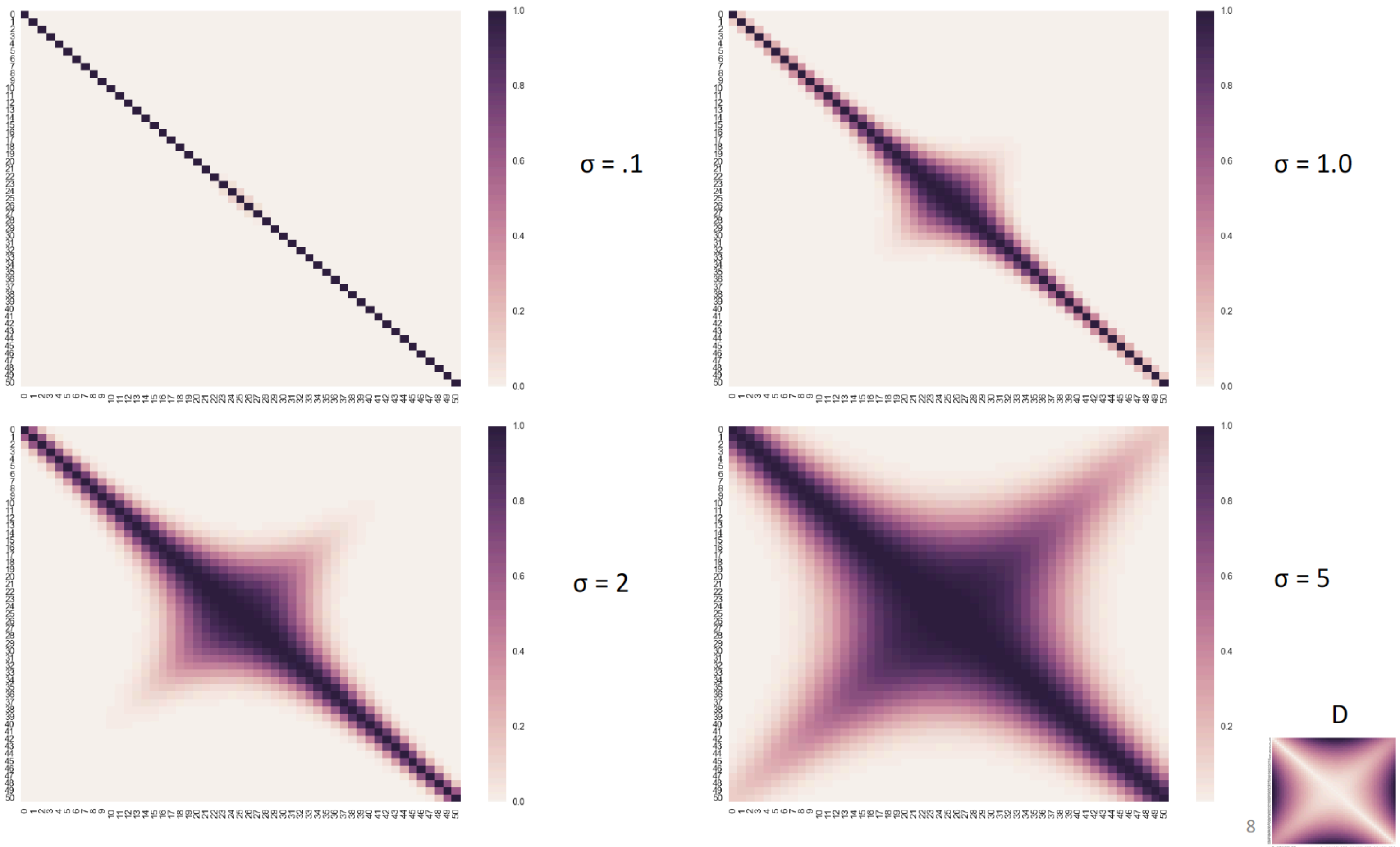
$\sigma = 0.1$

$\sigma = 1$

$\sigma = 2$

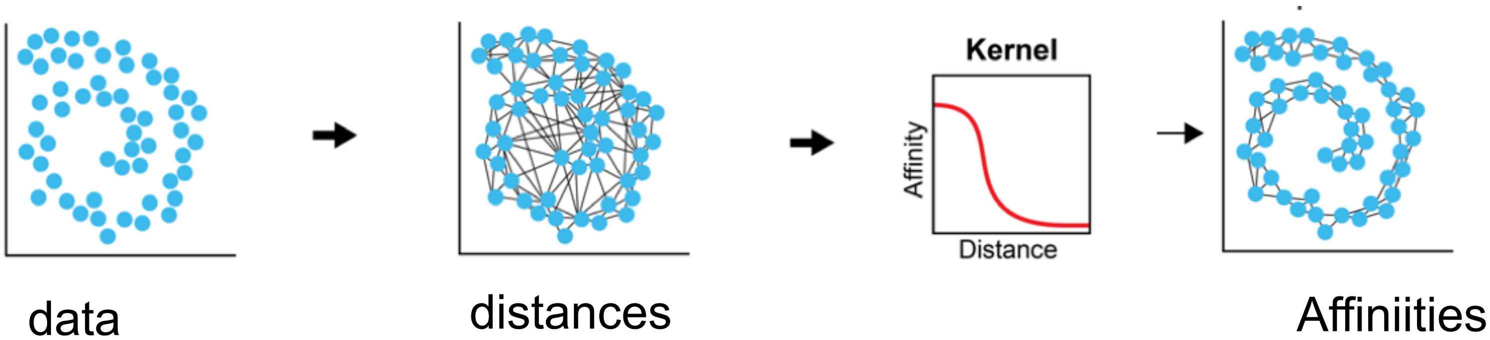
$\sigma = 5$

Effect of Changing Kernel Width



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

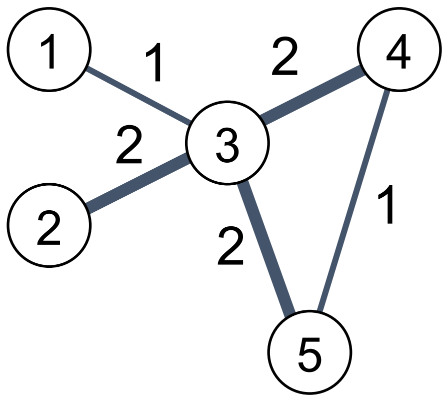
Kernel PCA



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

The graph Laplacian is a matrix representation of a graph

G



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

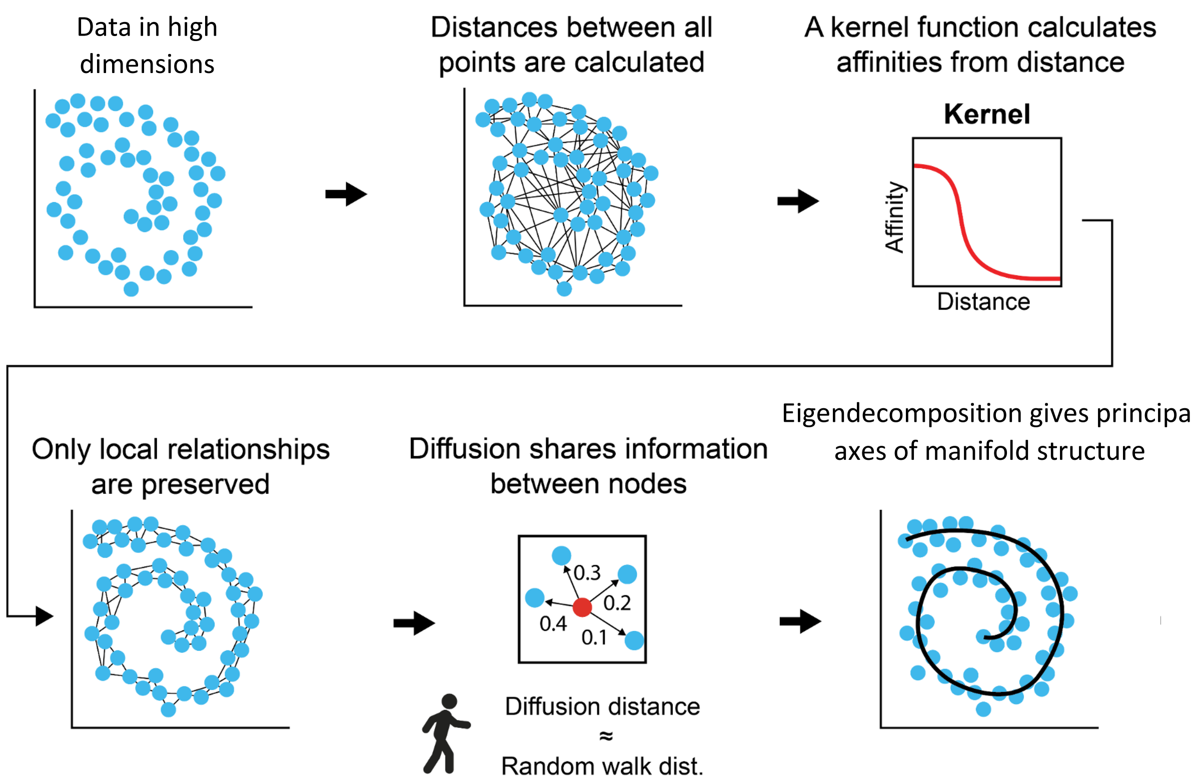
$$\mathcal{L} = \mathbf{D} - \mathbf{A} = \begin{vmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ -1 & -2 & 7 & -2 & -2 \\ 0 & 0 & -2 & 3 & -1 \\ 0 & 0 & -2 & -1 & 3 \end{vmatrix}$$

Laplacian Eigenmaps

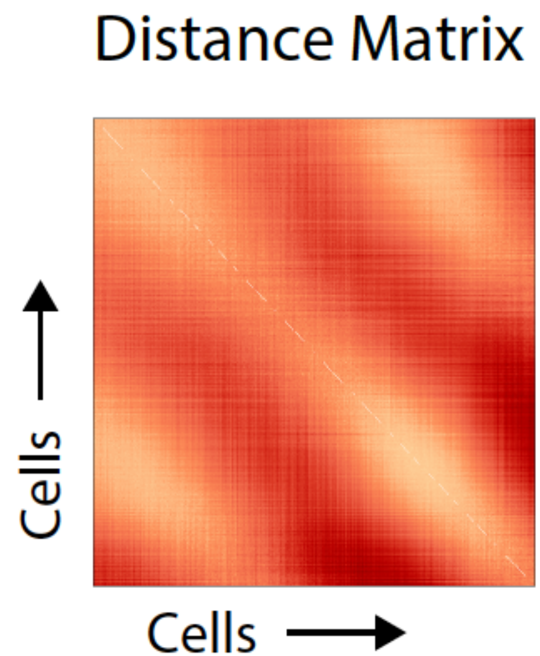
$$\mathcal{L} = \mathbf{D} - \mathbf{A} = \begin{vmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ -1 & -2 & 7 & -2 & -2 \\ 0 & 0 & -2 & 3 & -1 \\ 0 & 0 & -2 & -1 & 3 \end{vmatrix}$$

Eigenvectors of Laplacian matrix form Laplacian Eigenmaps

Diffusion Maps

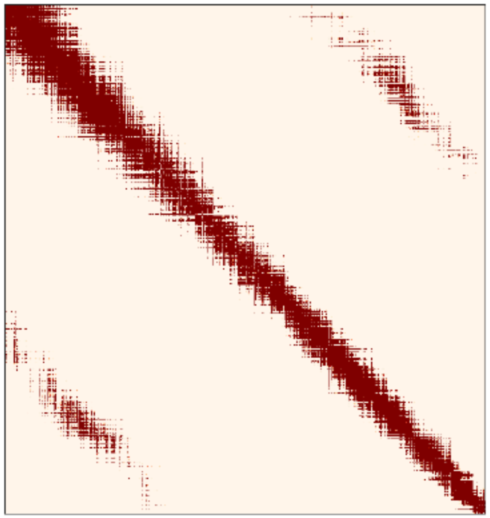


Distance Matrix

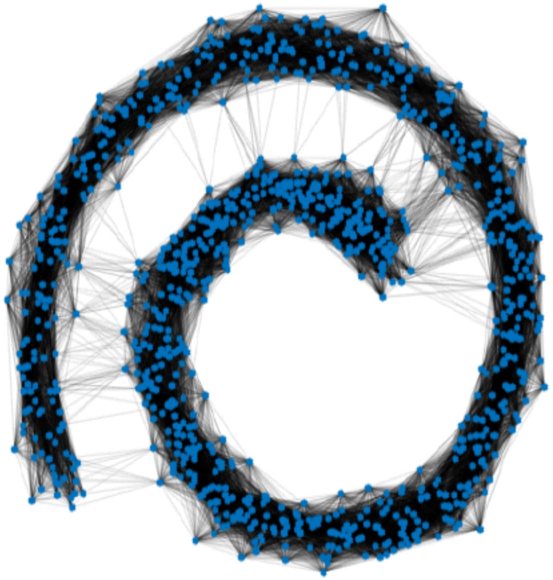


Entries are distances $D(x_i, x_j) = \sqrt{||x_i - x_j}||$

Affinity Matrix



(Gaussian Kernel)



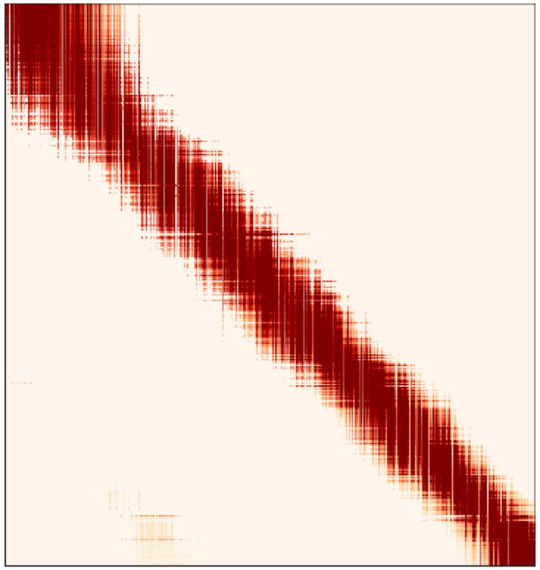
Graph Representation

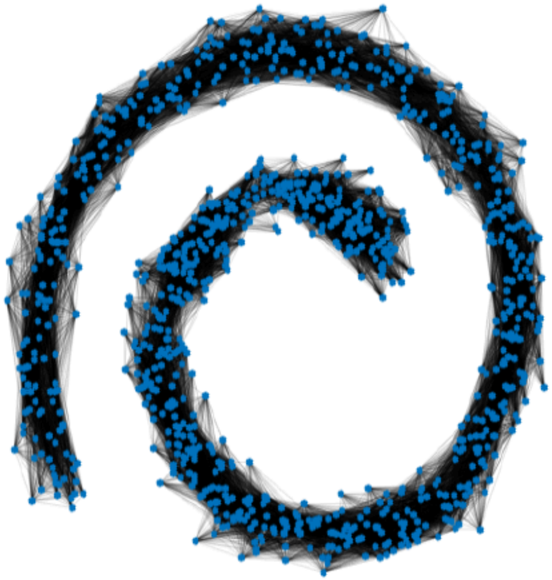
Entries are affinities

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

Powered Markov Matrix

Powered Markov Matrix



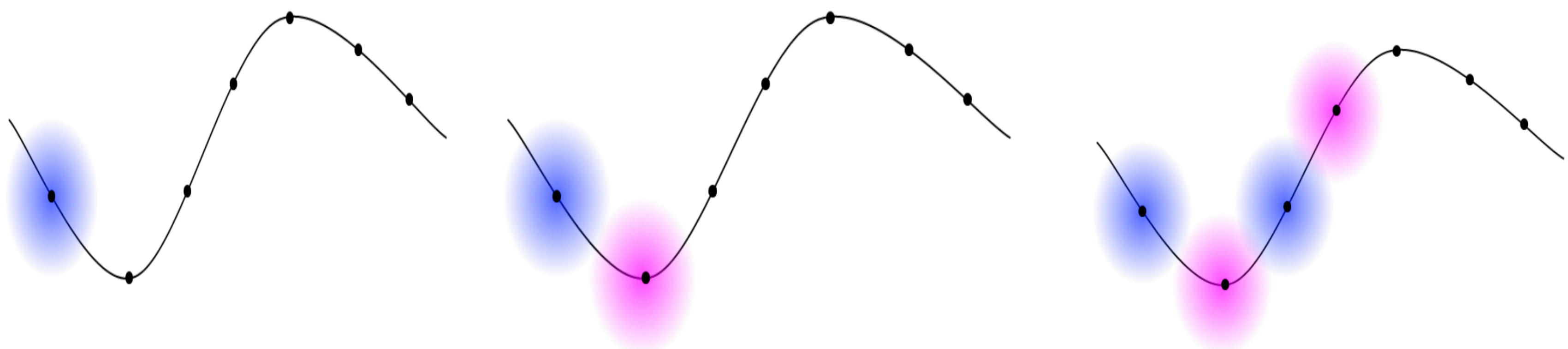


Entries are row normalized affinities

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

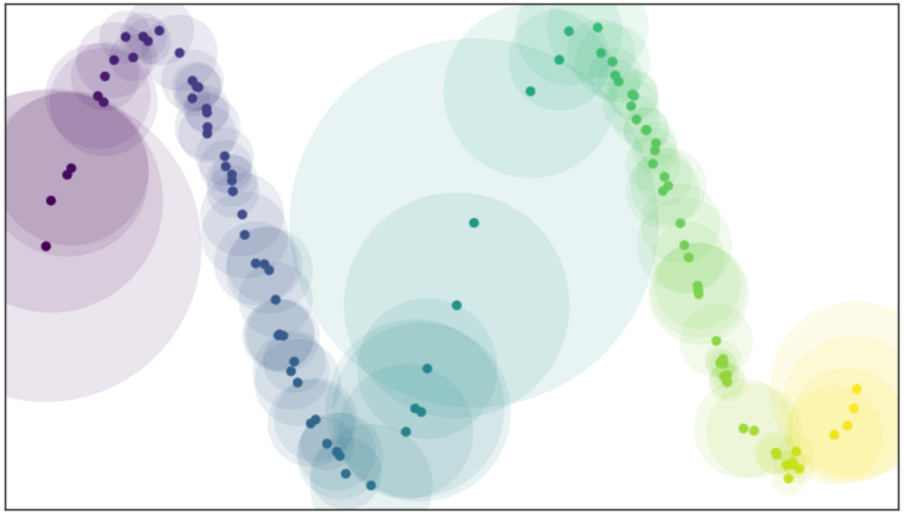
What does Powering Do?

- Smooths and infers connectivity through data



Adaptive neighborhoods

- Neighborhoods can adapt to the density of the local area
- This actually helps diffusion discover manifold geometry by avoiding “dense ditches”



Diffusion maps pros and cons

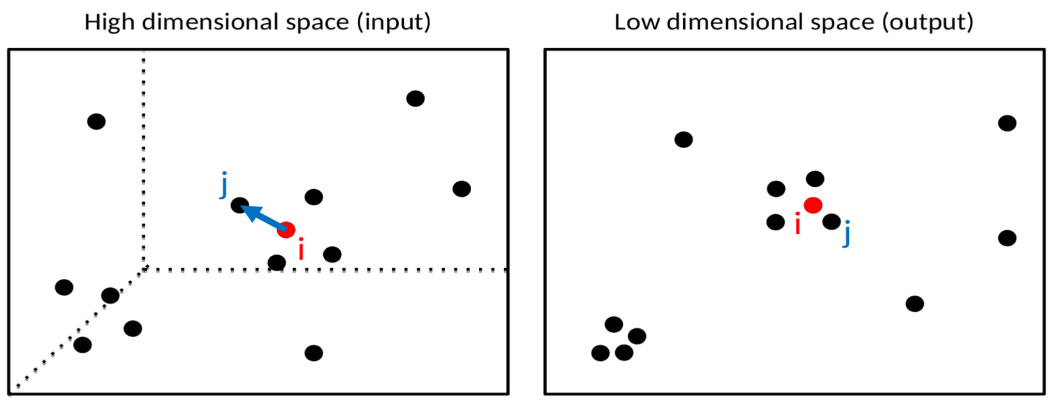
- ✓ Find intrinsic dimensions within noisy data
  - ✓ Denoises data well
- ✓ Captures trajectories or clusters in the data
- ✓ Distances are meaningful in diffusion maps
- ✓ Robust in that runs are the same
  - ✓ Deterministic method

- x Separates each component into different dimensions
  - x Each eigenvector is roughly one trajectory or one cluster ---bad for a 2D visualization (but good for clustering!)
- x Slow runtime---due to eigendecomposition of a large matrix

2D Dimensionality Reductions

- 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 (and UMAP)



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

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

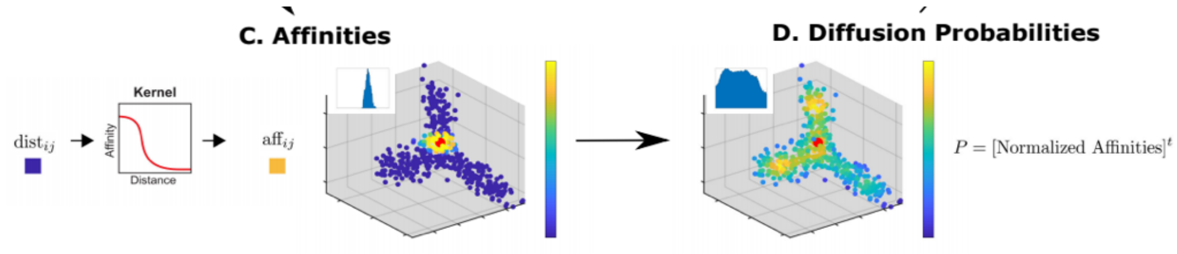
tSNE/UMAP pros and cons

- ✓ Captures information in 2 D
- ✓ Preserves local neighborhoods
- ✓ Shows number of points in a cluster by flattening out density (i.e. does not “overplot”)
- ✓ UMAP fast due to SGD and lack of normalization
  - ✓ tSNE can be slow for large number of points and low dimensions

- x Does not preserve any type of density or distance
- x Does not capture global structure of data
- x Does not denoise data
- x Based on local minima, solution can change run-to-run

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

$C_1 = [\ p_{11}\ p_{12}\ \ldots\ldots.p_{1n}]$
$C_2 = [\ p_{21}\ p_{22}\ \ldots\ldots.p_{2n}]$
…
$C_n = [\ p_{n1}\ p_{n2}\ \ldots\ldots.p_{nn}]$

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

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

......

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

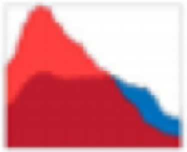
...

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

...

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

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



- Step 4: Embed with multidimensional scaling

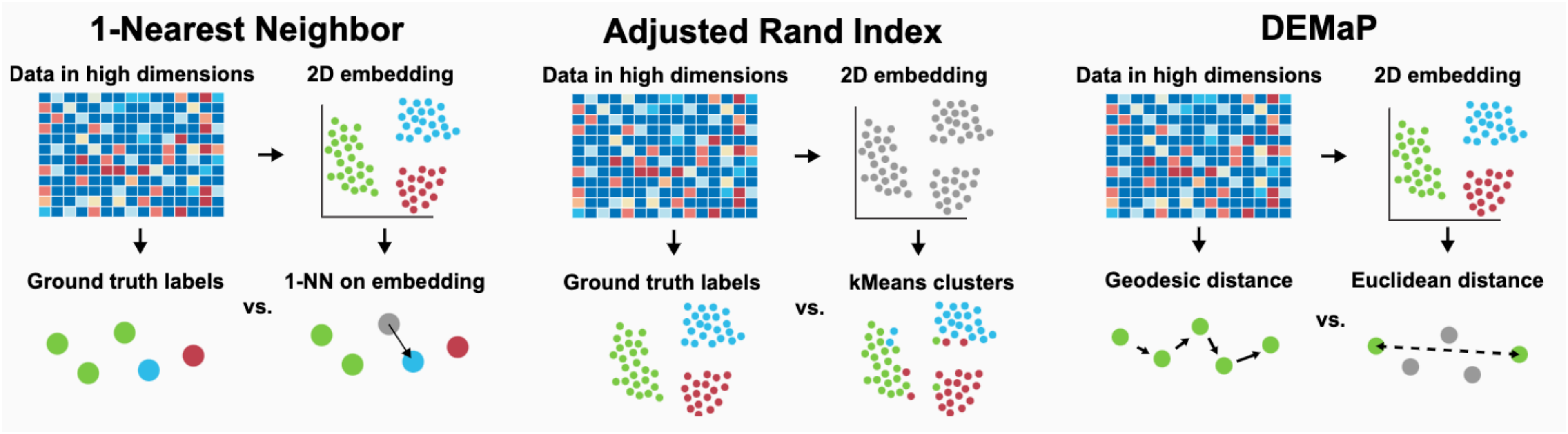
- This reorganizes information in a diffusion map into 2D

PHATE pros and cons

✓ Captures information in 2 D
✓ Denoises data for clean visualization
✓ Preserves local and global manifold structure
✓ Fast due to landmarked diffusion tricks
✓ Distances are meaningful in PHATE (preserves Denoised Manifold Affinity)
✓ Robust, same solution run-to-run

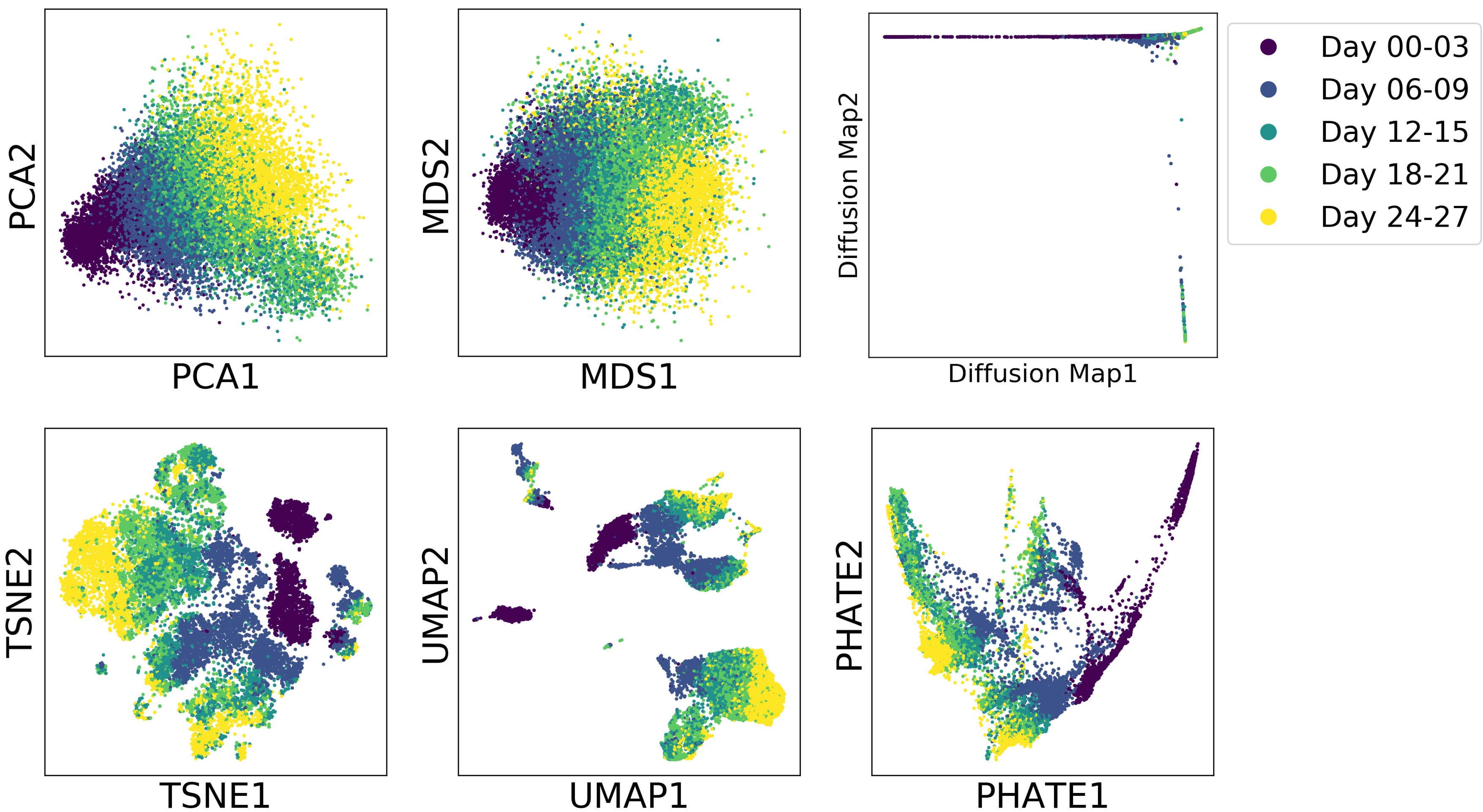
x Overplots data similar points can be very close
x Shows “horseshoe” shapes due to MDS

Quantitative Metrics

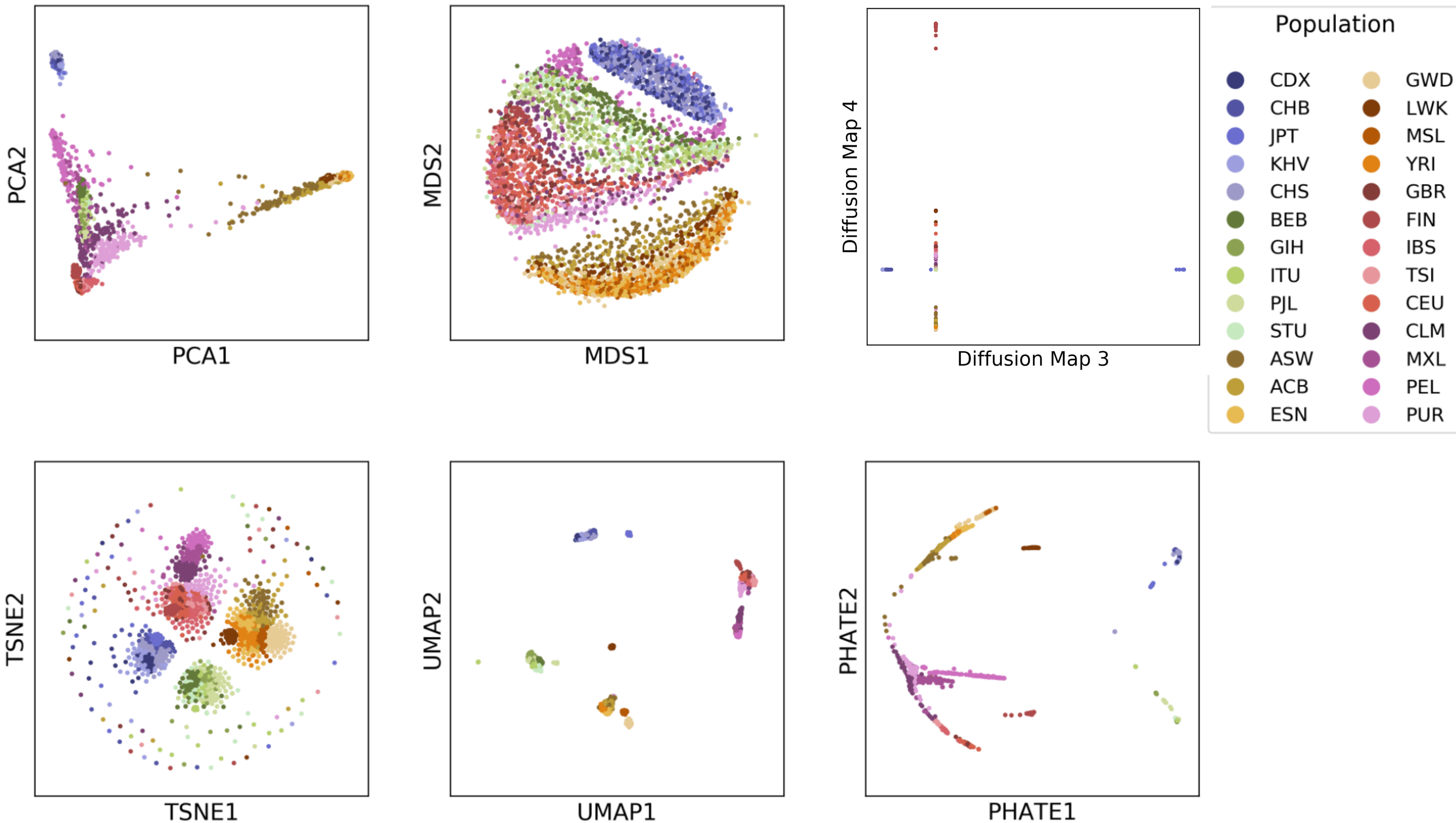


PHATE tends to do better on DeMAP, and ARI tSNE/UMAP on 1-NN, sometimes on ARI on certain Datasets

Visual Information Extraction: Embryoid Body



1000 Genomes



What visualization would be good for understanding the heterogeneity in T cells measured after an infection?

What visualization might be good for deciding if a dataset has a single dominant trajectory of progression?

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 denoised 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