The Krishnaswamy Laboratory

Yale Genetics and Yale SEAS present

Machine Learning

for

Single Cell Analysis

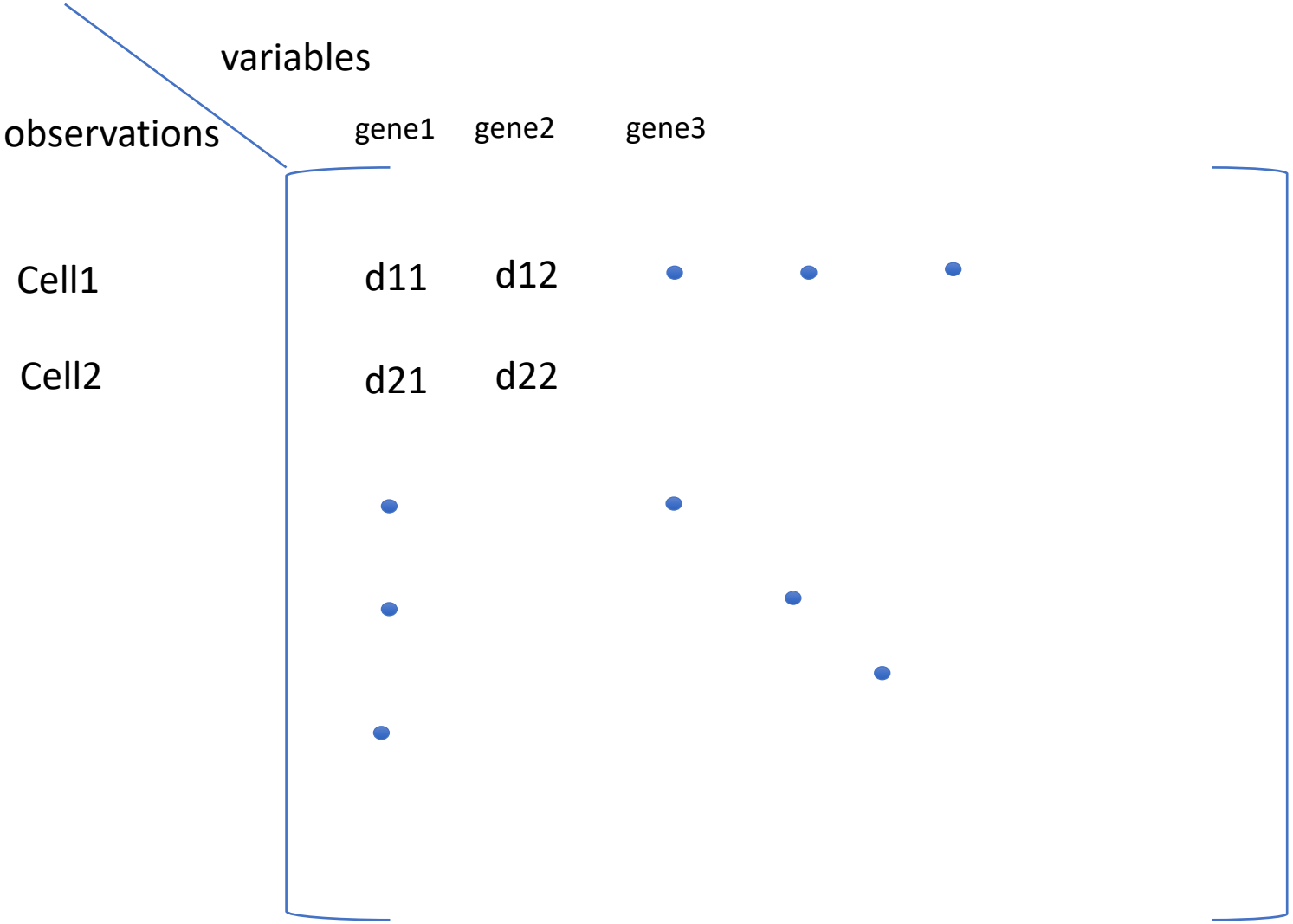
Online - May 20-29, 2020

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

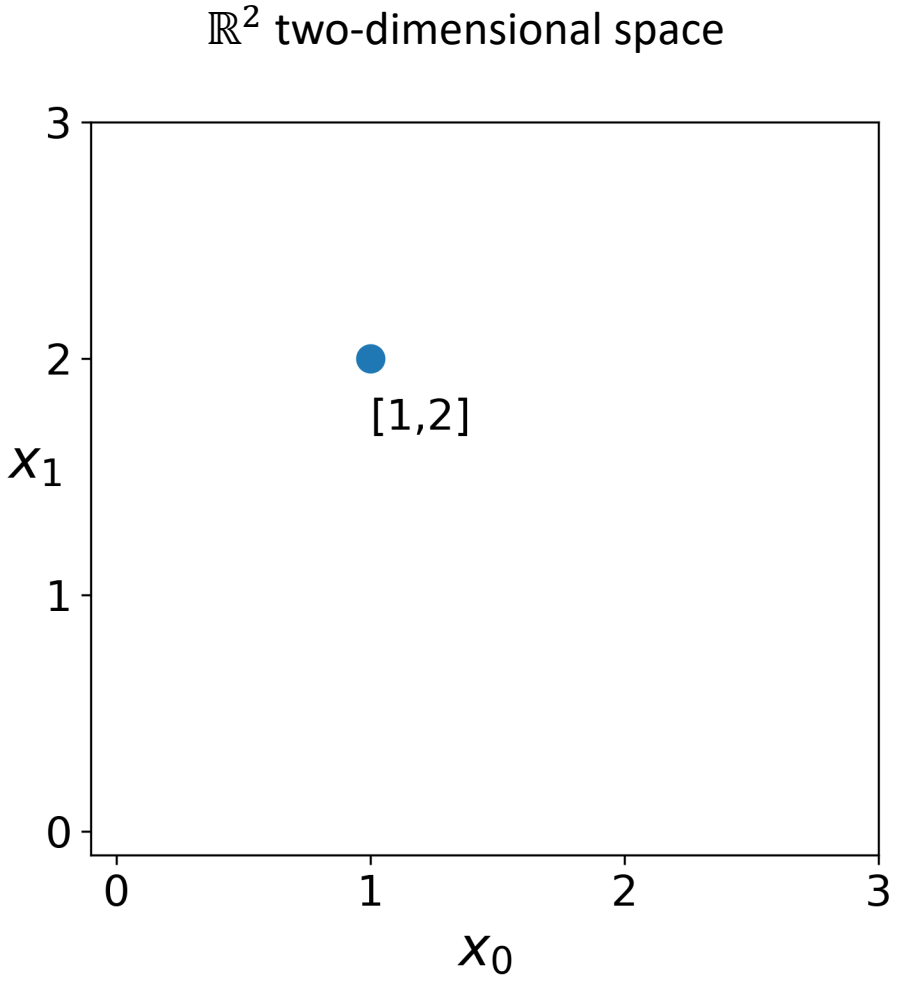
Day 2: Visualization, Dimensionality Reduction and Manifold Learning

Thinking about high dimensional data

Our Data is a High-dimensional Matrix



What does “dimensionality” mean?



Features are dimensions

vector coordinate space

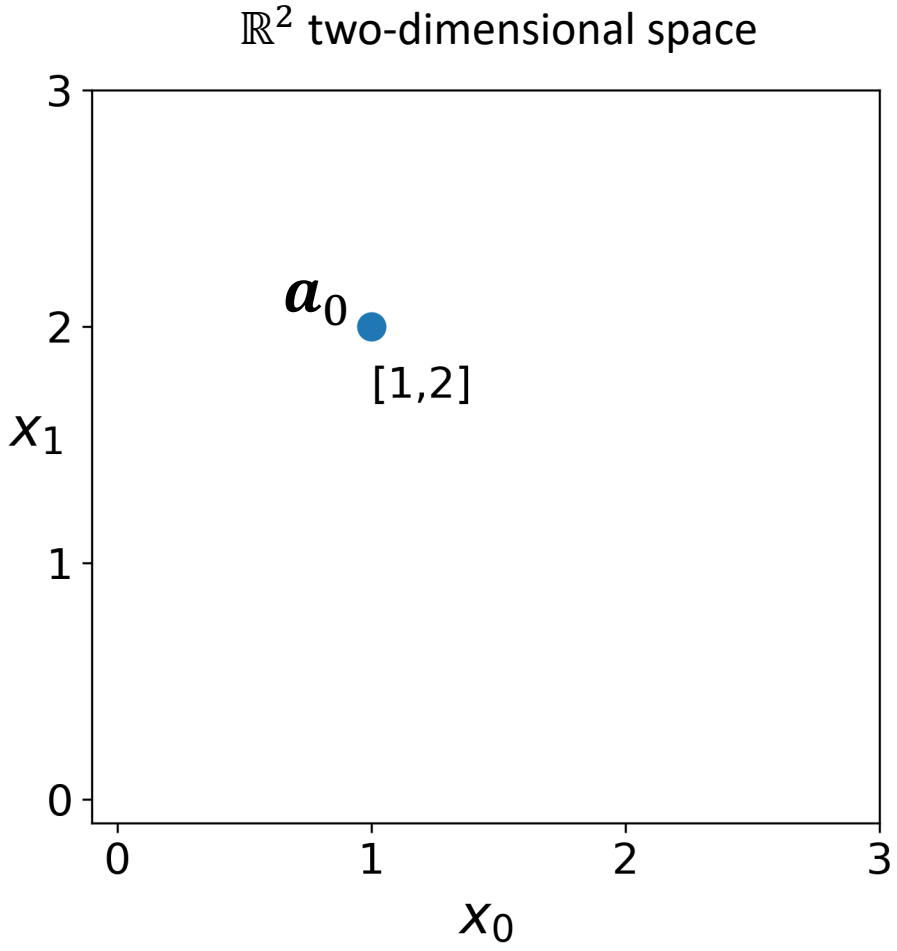
$\boldsymbol{a}_0 \in \mathbb{R}^2$

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

x_0 x_1

features

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



Features are dimensions

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

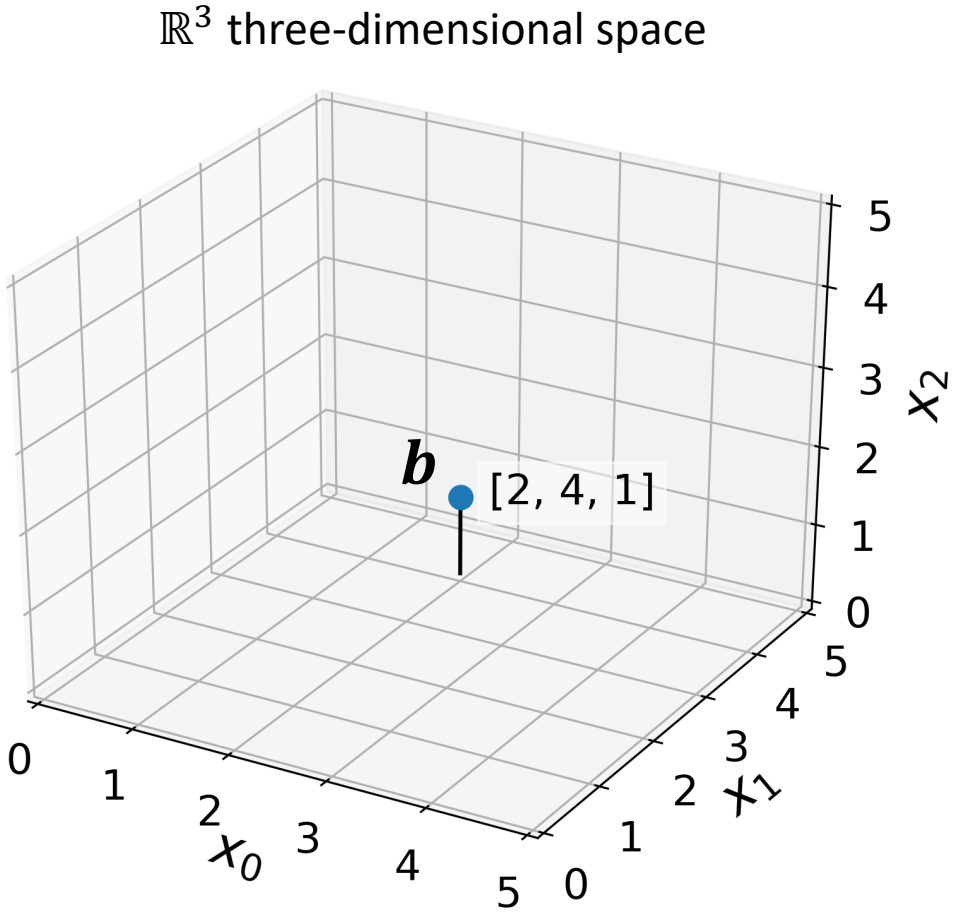
$\boldsymbol{a} = [1,2]$

x_0 x_1

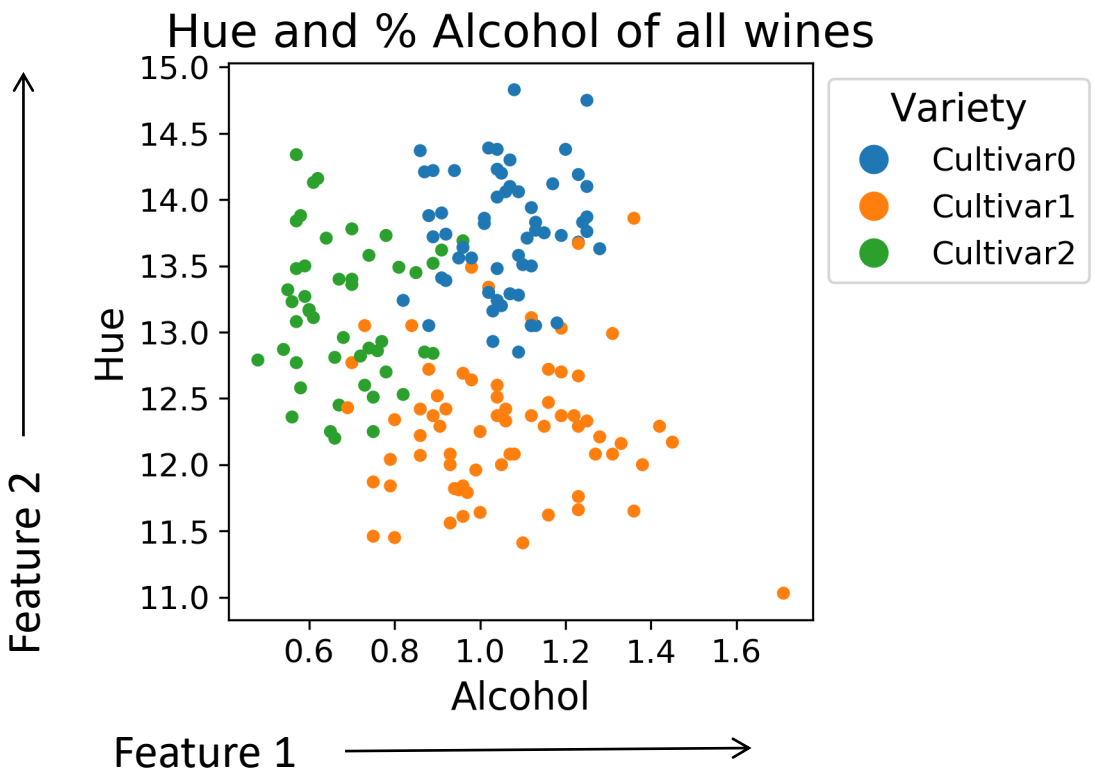
$\boldsymbol{b} \in \mathbb{R}^3$

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

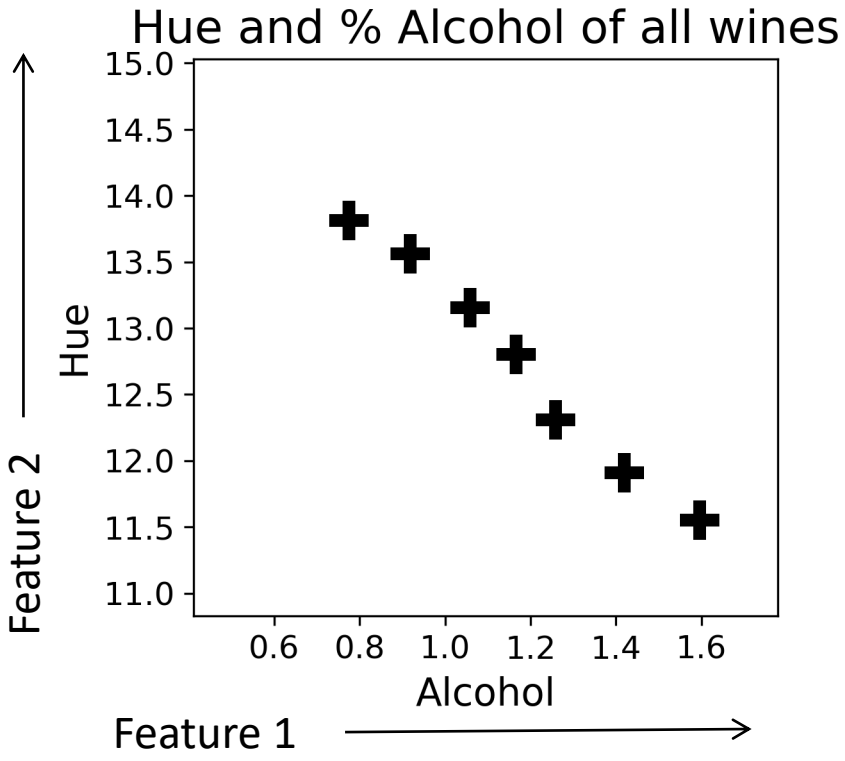
x_0 x_1 x_2



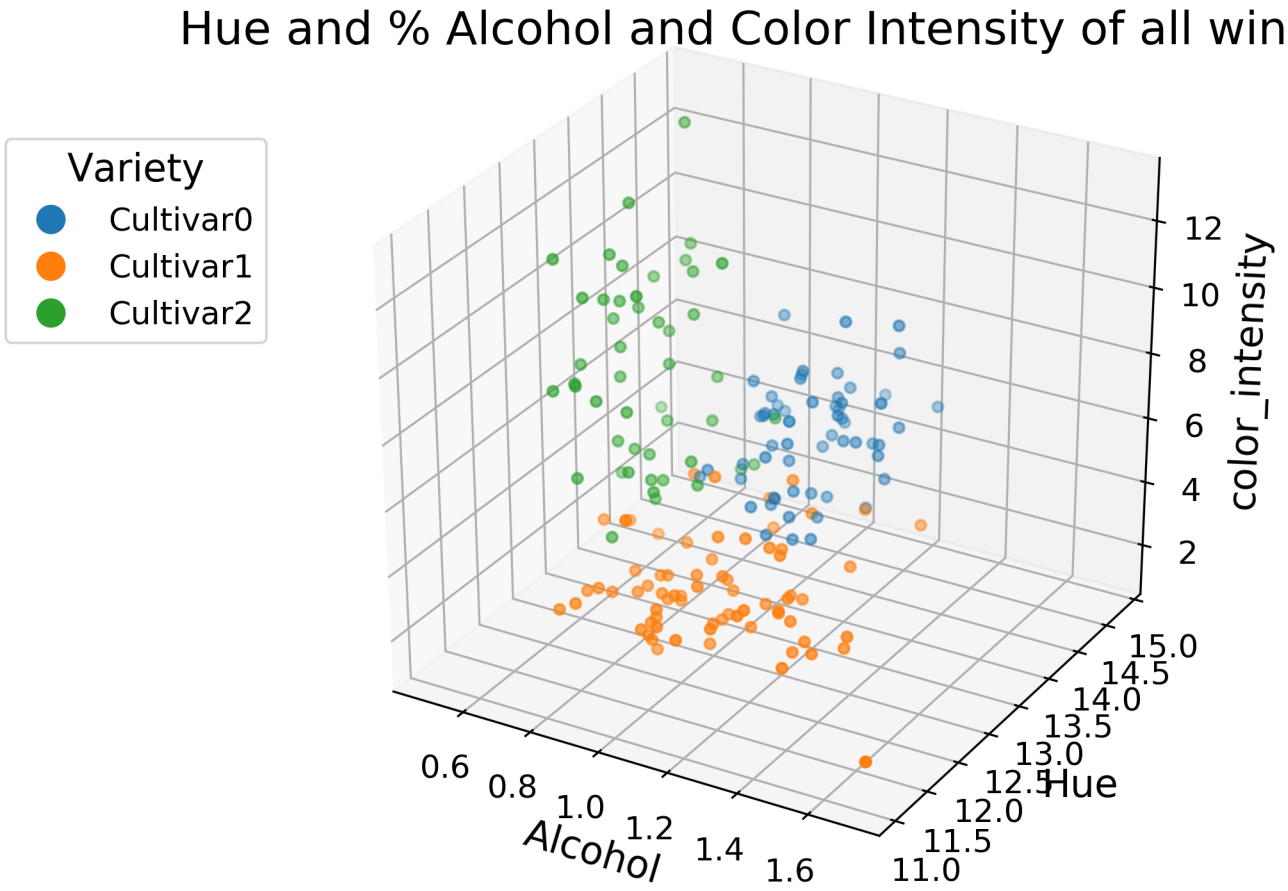
Features reveal structure



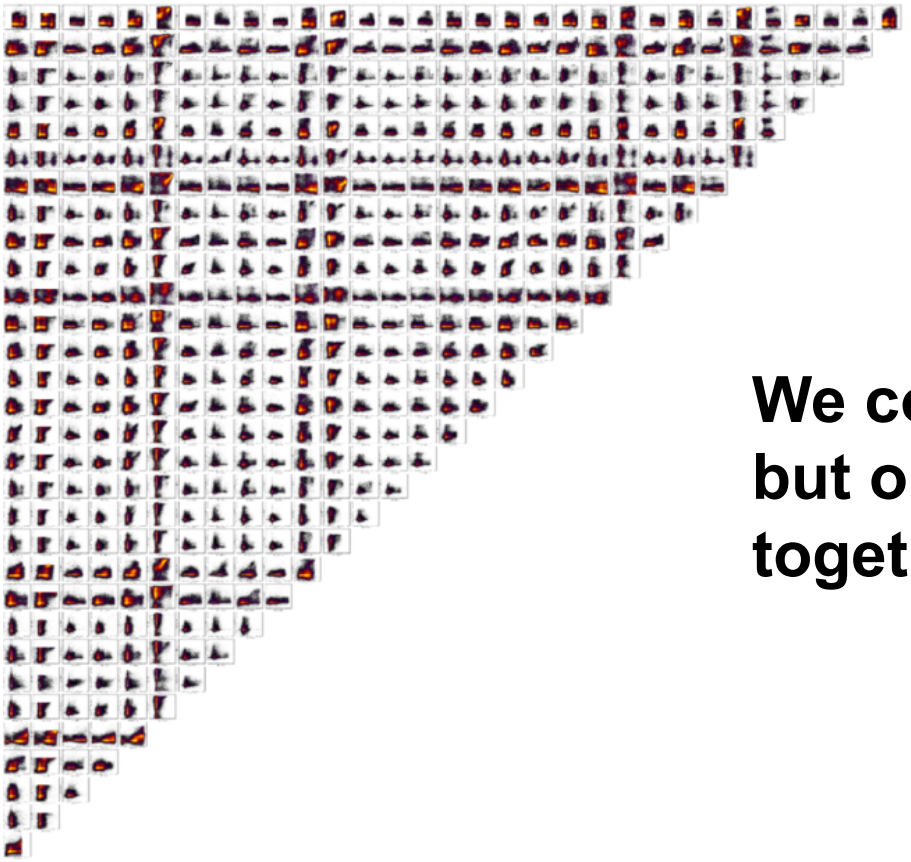
Many types of structure can exist in data



More Dimensions = More Accurate Structure



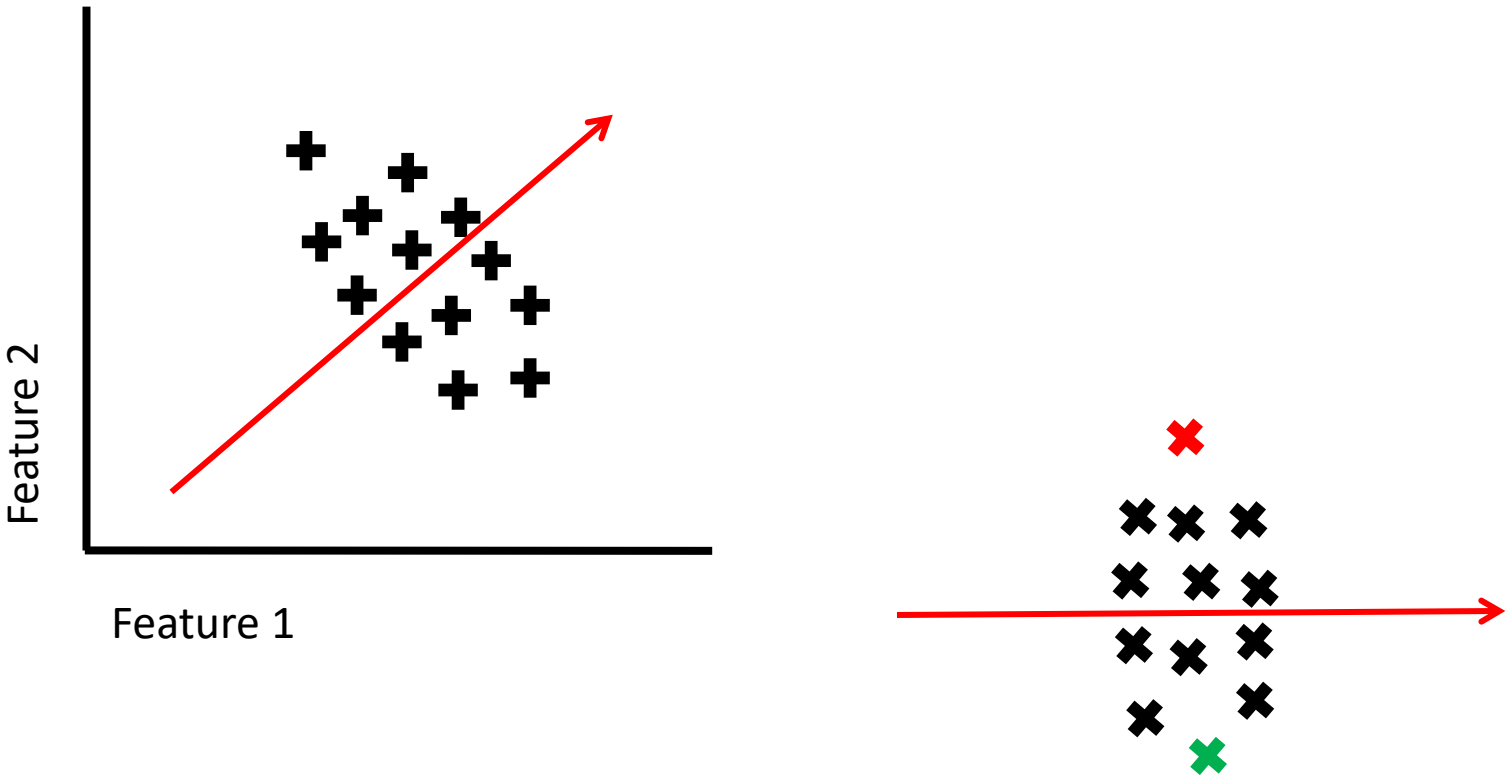
Problem: We can only see in 3D (not 20K D)



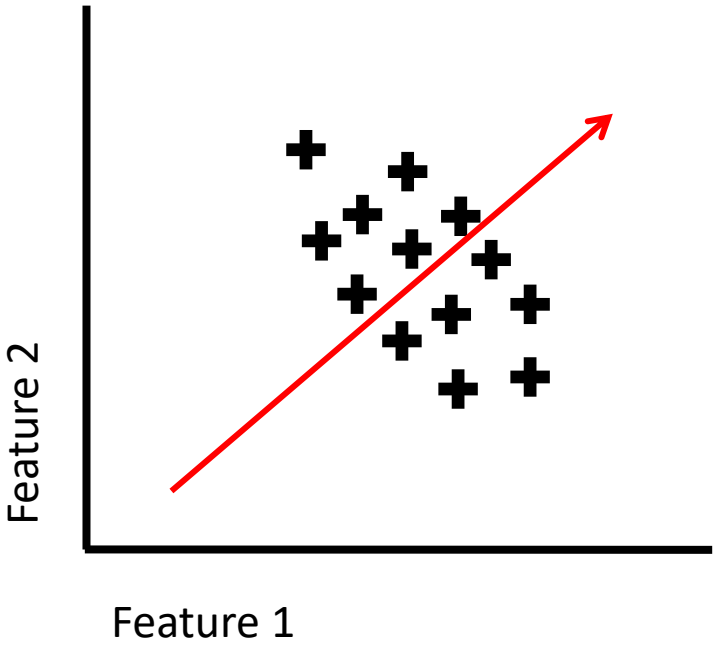
We could try all 2D pairs, but our brains can't put this together

Solution: Dimensionality Reduction

Thought Exercise: Reduce this to 1D

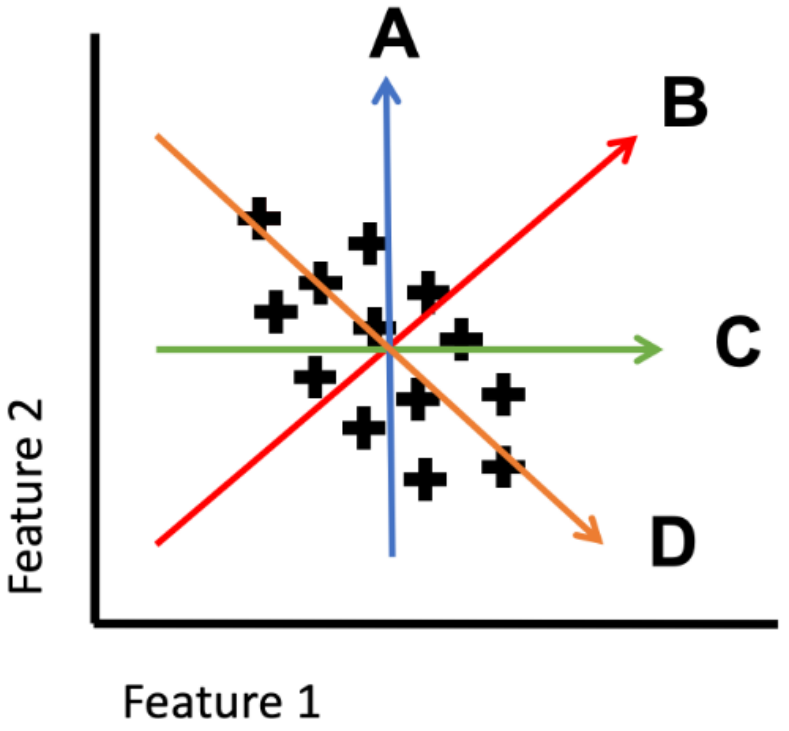


Thought Exercise: Reduce this to 1D



Most distant points on top of each other !

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



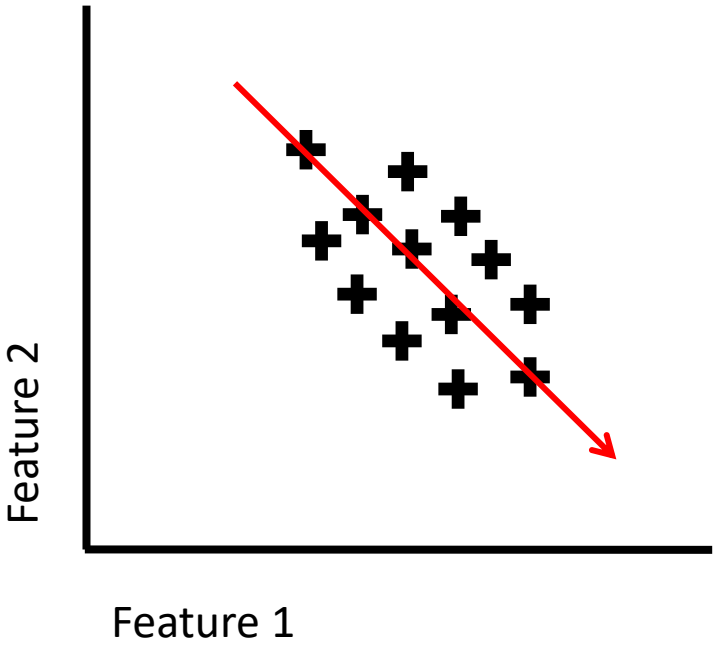
A

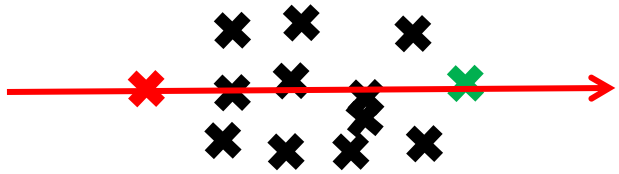
B

C

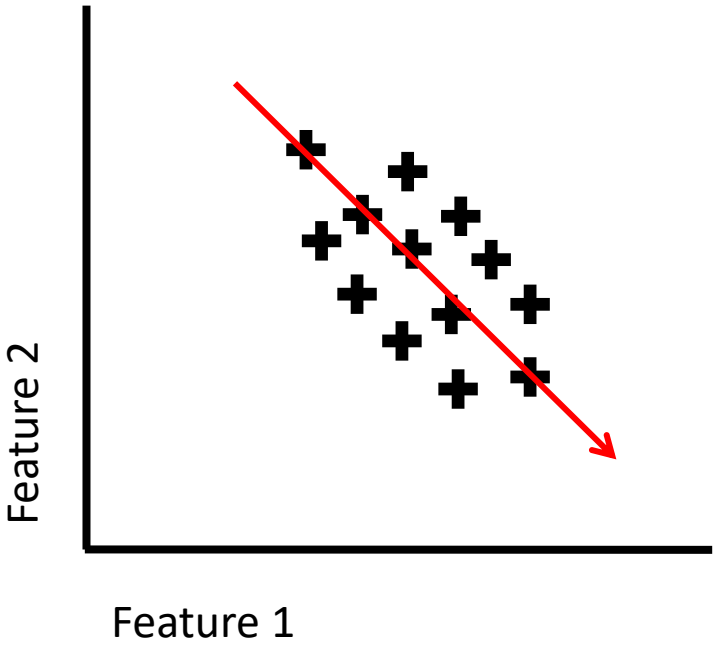
D

Thought Exercise: Reduce this to 1D





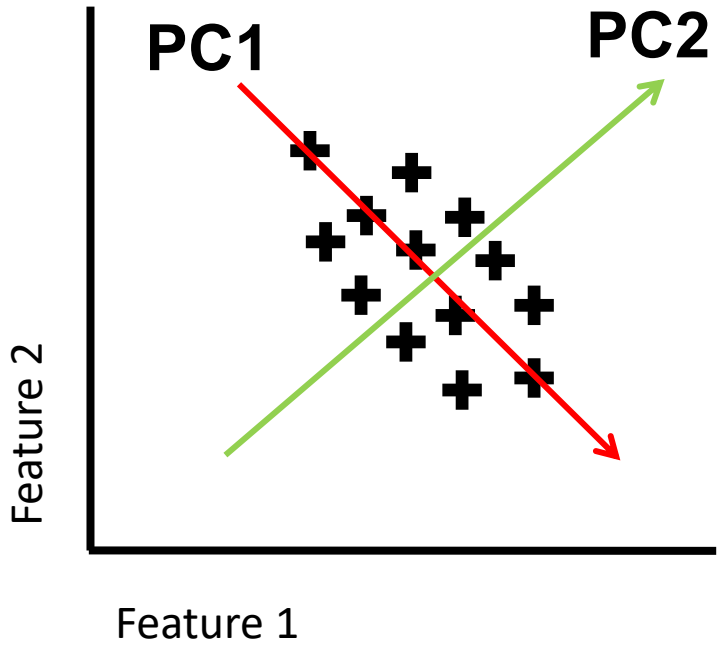
Thought Exercise: Reduce this to 1D



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

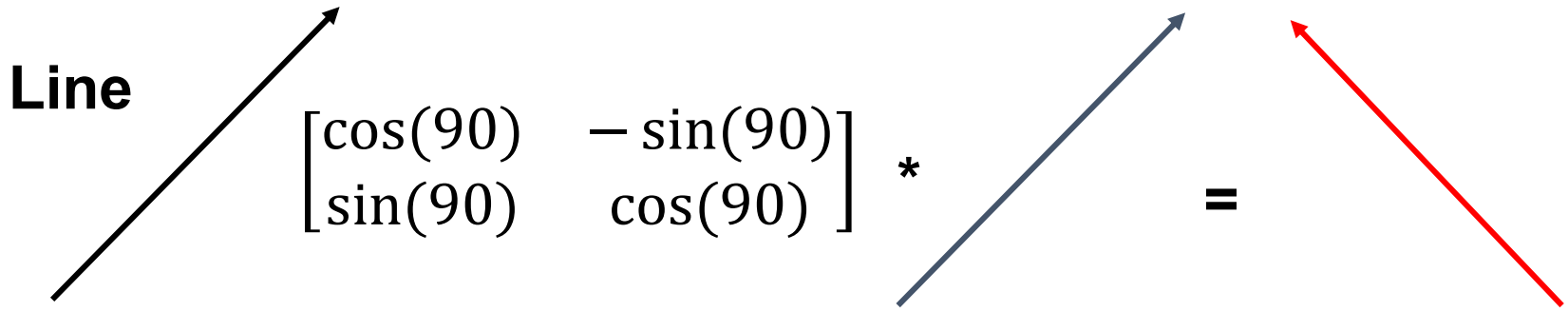
$$\mathbf{X} = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$$

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

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

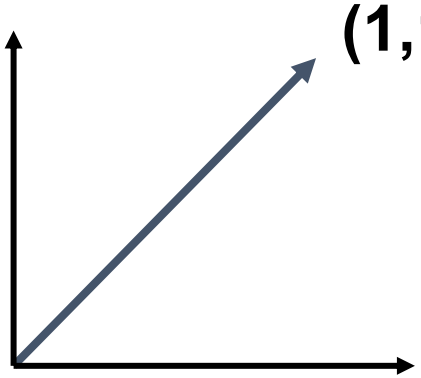
Matrices as Transformations

- 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



Matrix–Vector Notation

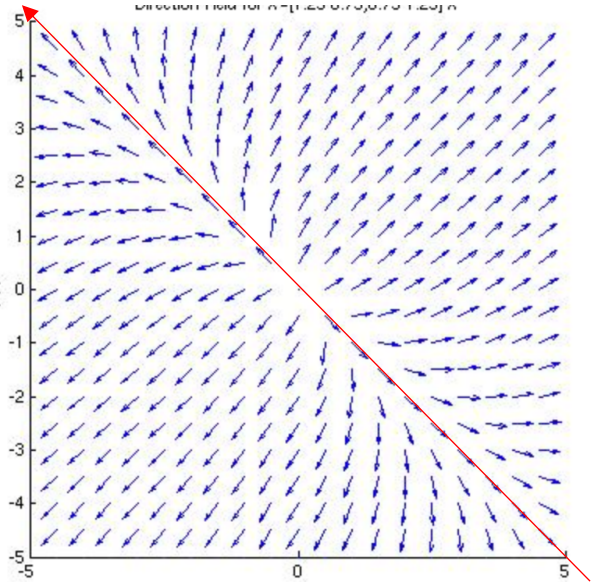
- Lines can be described as vectors from the origin



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

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

Eigenvectors



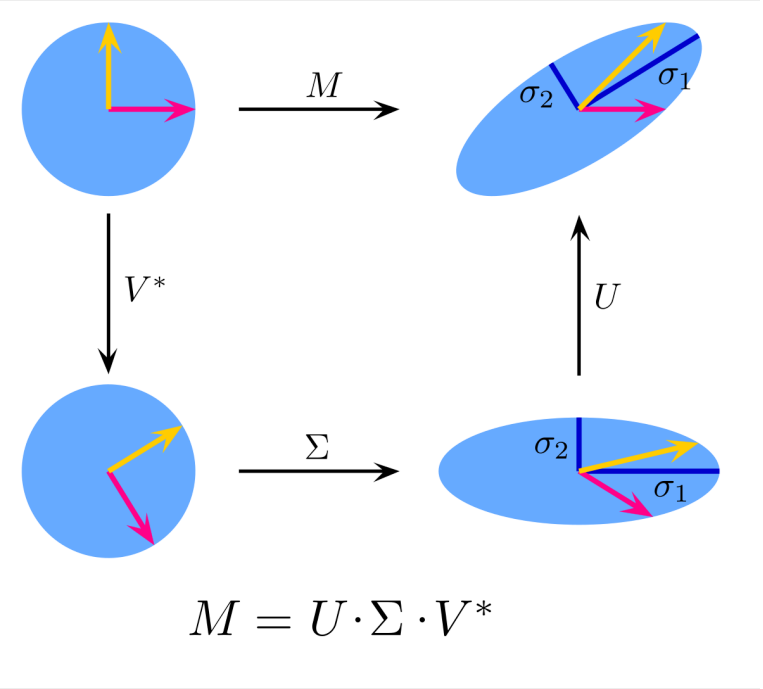
Rotation matrices rotate lines in only one direction

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.

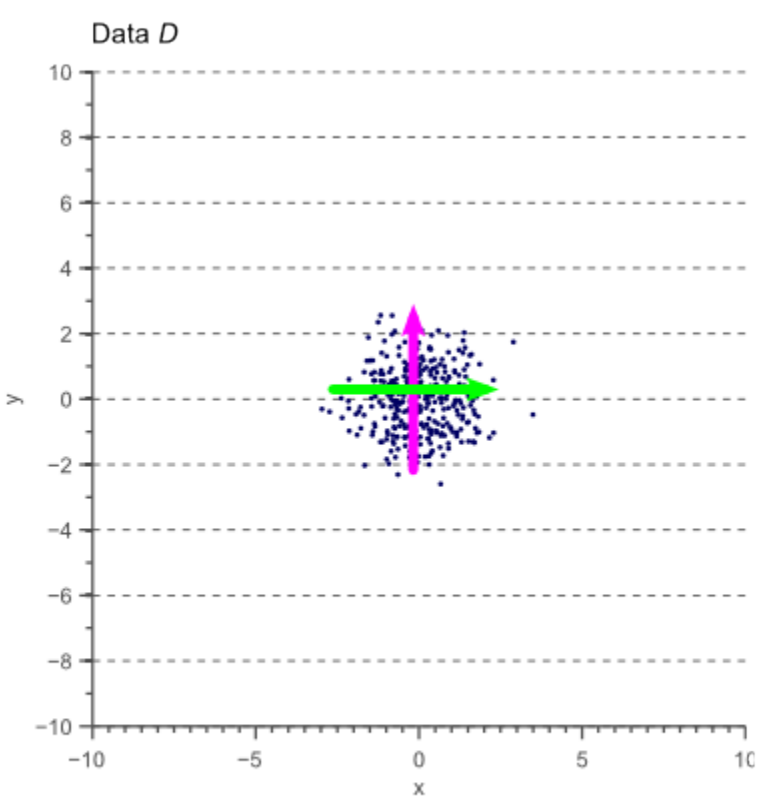
Eigendecomposition

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

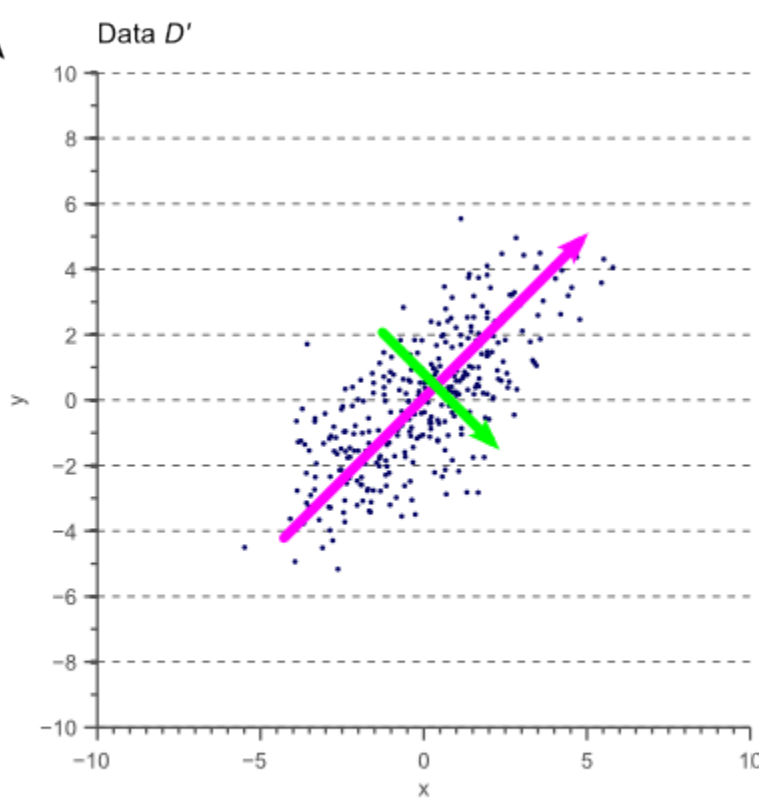


Non-square matrices have similar OP called **Singular value decomposition**

Eigenvectors of Covariance



$$D' = TD = RSD$$

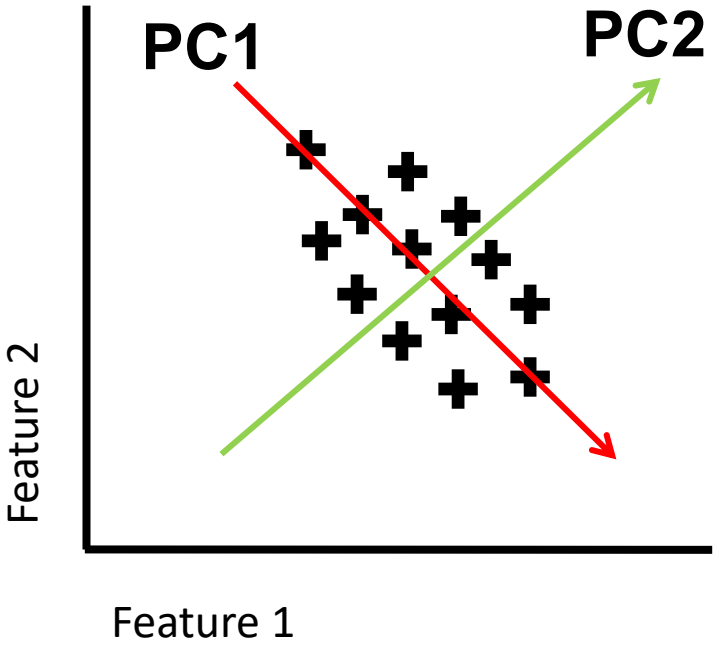


$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma' = \begin{bmatrix} 4.25 & 3.10 \\ 3.10 & 4.29 \end{bmatrix} = RSSR^{T}$$

Matrix creates correlation structure of data on unit blob

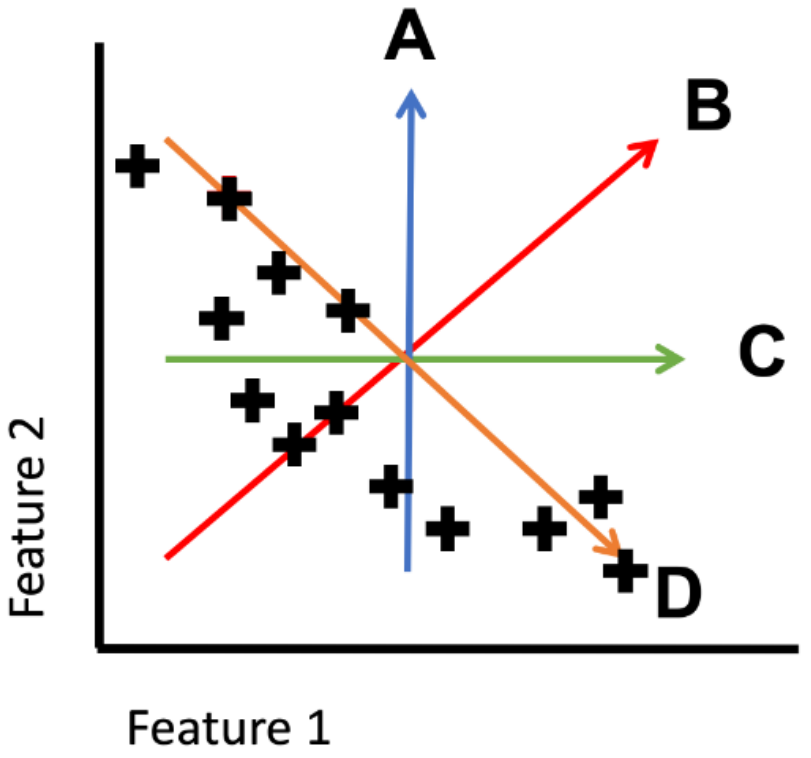
Principle Components Analysis



PC1 is first eigenvector (with Highest eigenvalue)

PC2 is next eigenvector

Which line would be PC1 for the following data?



A

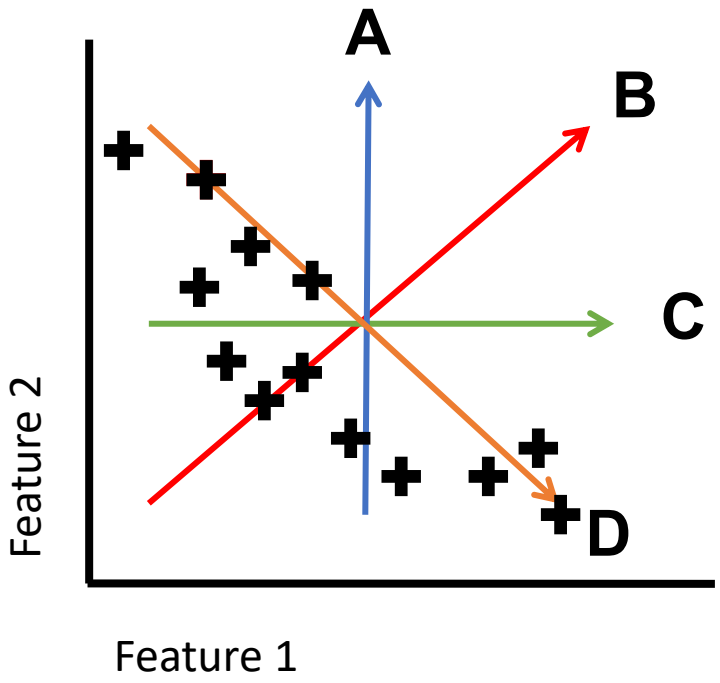
B

C

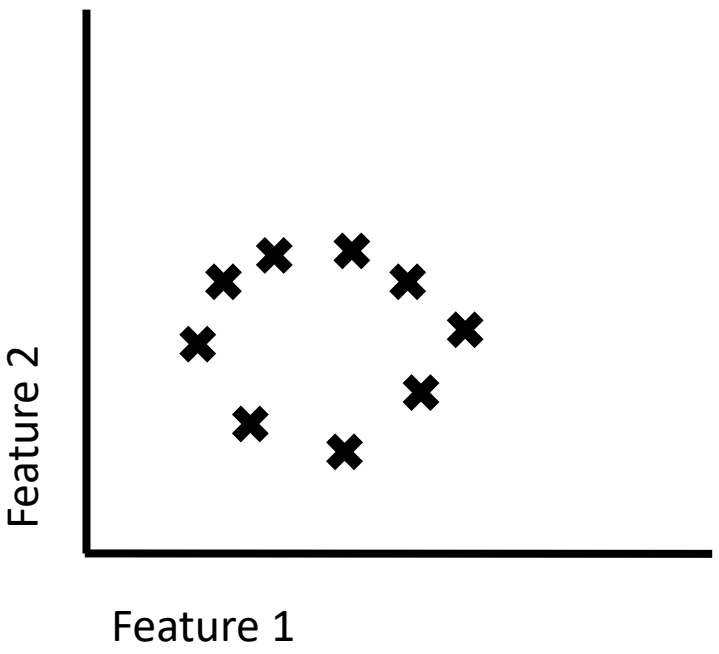
D

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?

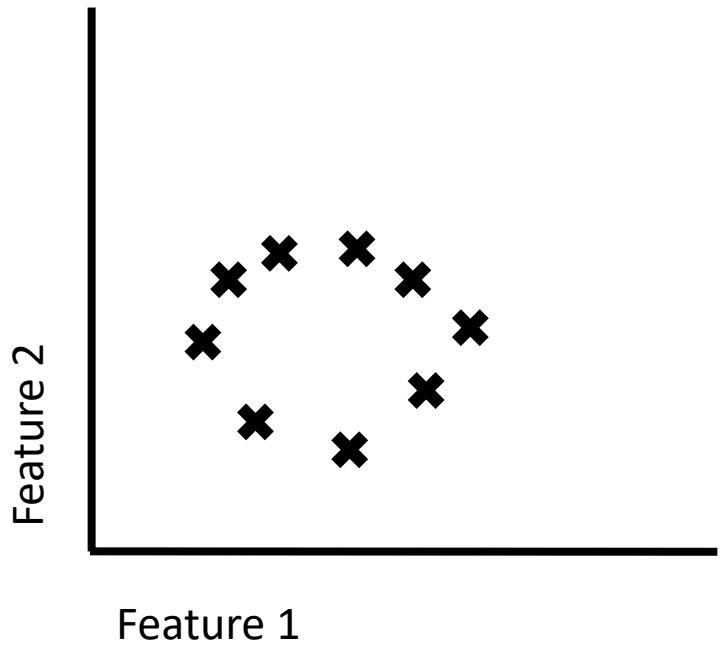
Conclusions

- High dimensional data is often represented using a matrix with observations on the rows and features on the columns
- Visualizing high dimensional data requires selecting features or combinations of features
- PCA uses eigenvectors of the covariance matrix to identify the combination of features that maximize variance

Non-linear dimensionality reduction

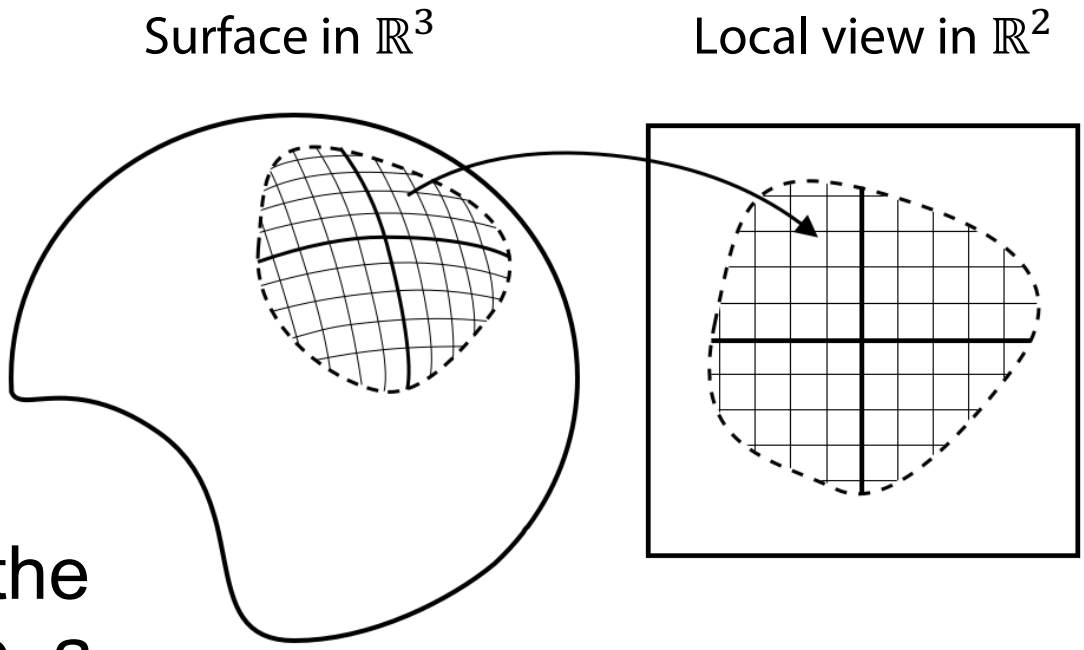
Non-linear dimensions

- 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”

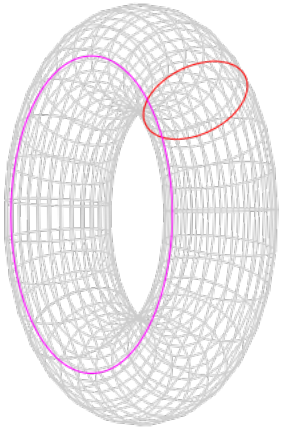


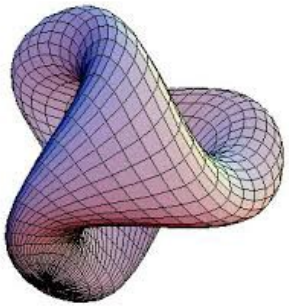
What is a manifold?

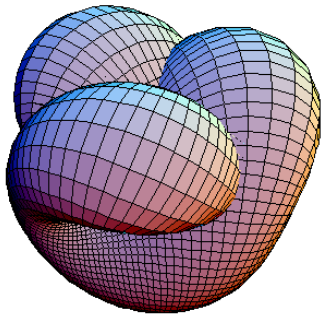
- Locally smooth
- Locally Euclidean
- Generally, lower dimensional than the ambient space (i.e. a subspace)

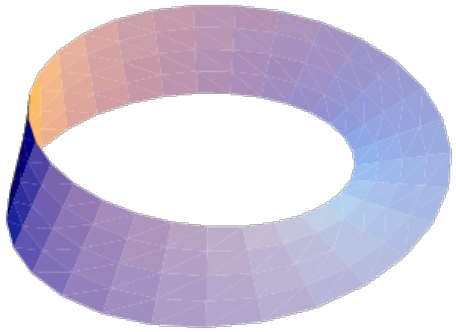


Examples of manifolds

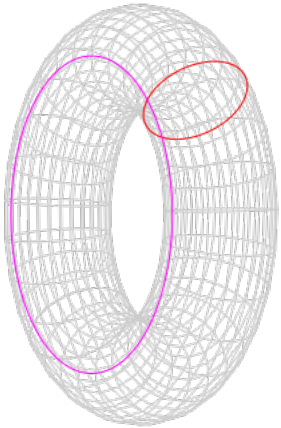


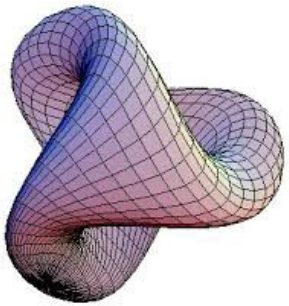


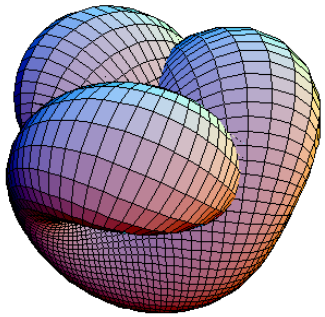


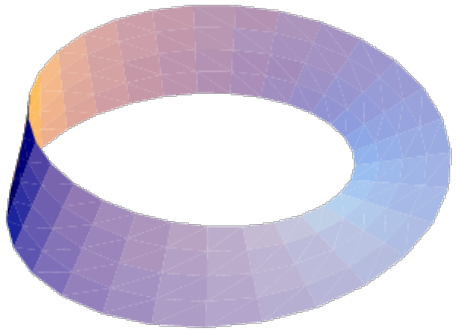


Examples of manifolds

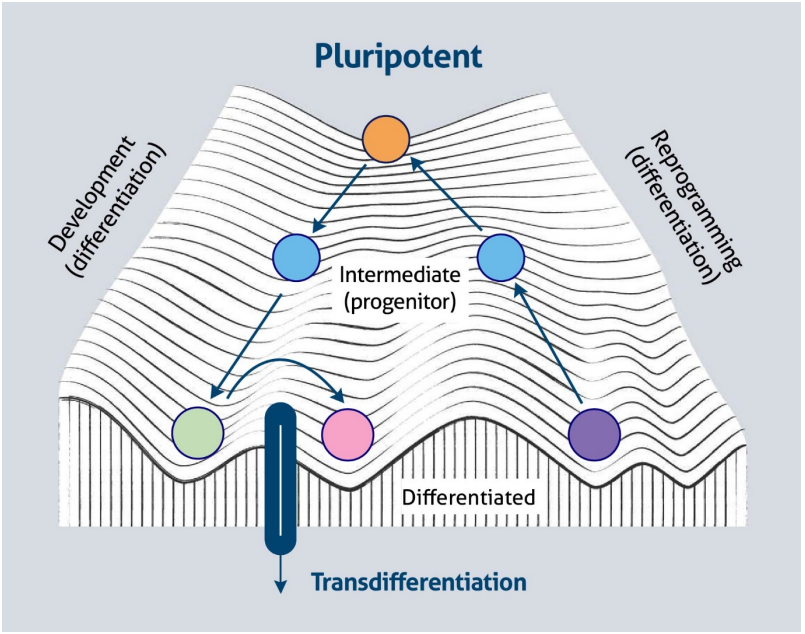




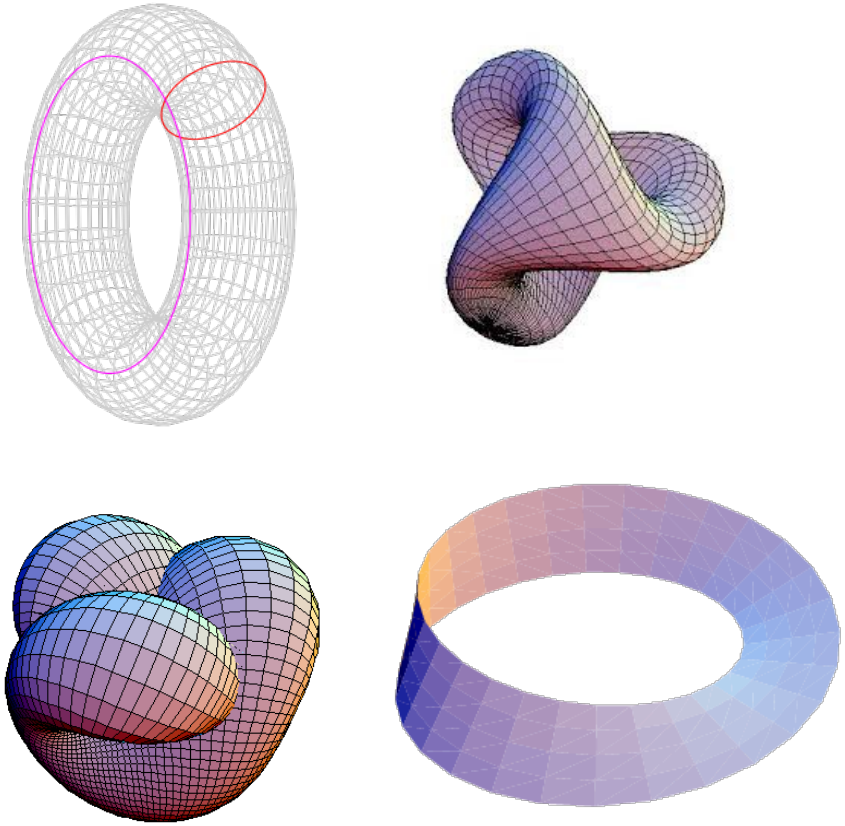




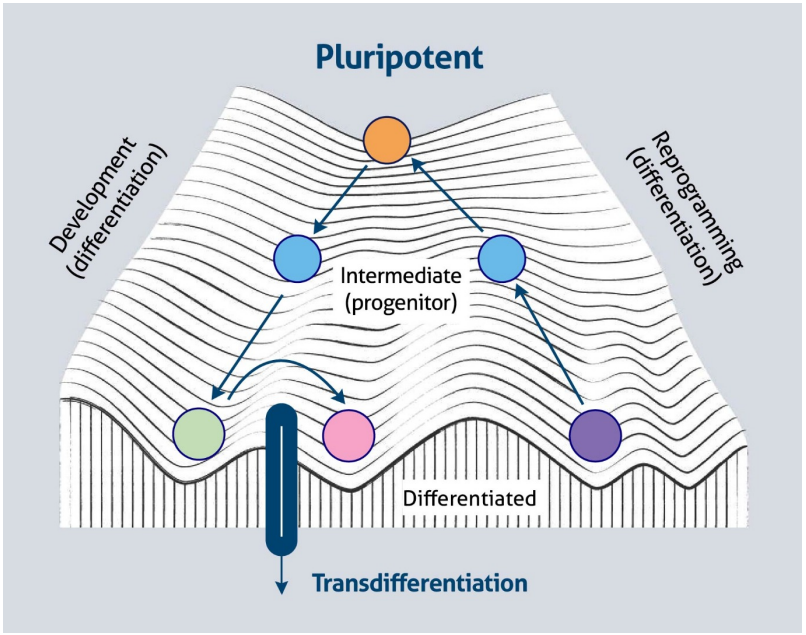
Waddington's landscape



Examples of manifolds



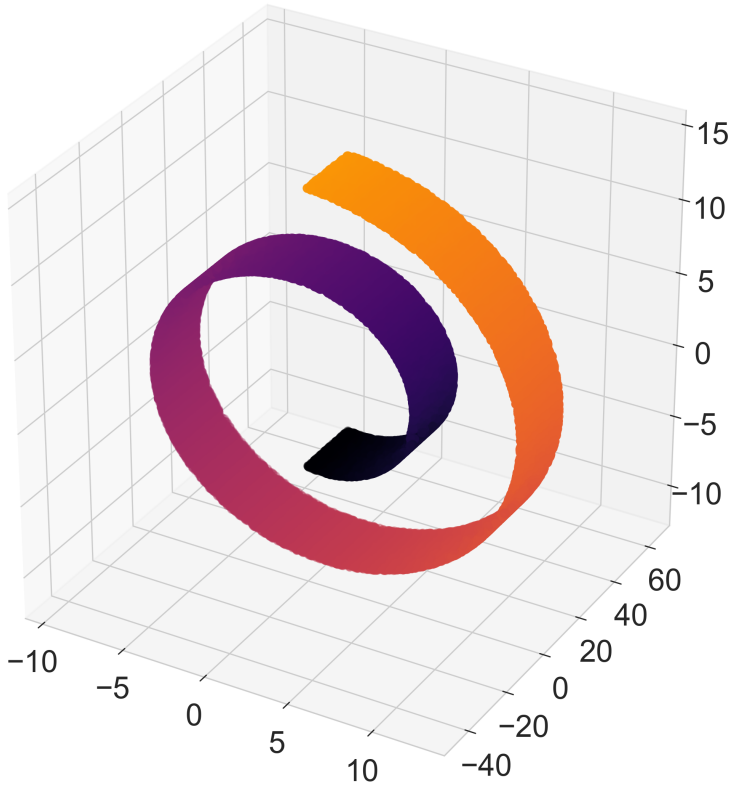
Waddington's landscape



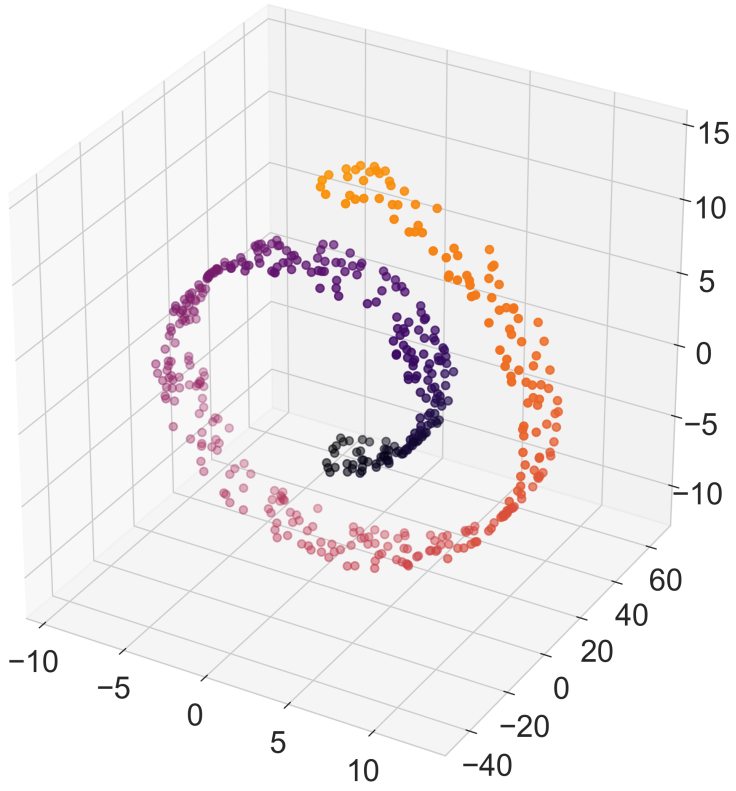
- Smooth transitions between states
- Small changes in gene expression are linear

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

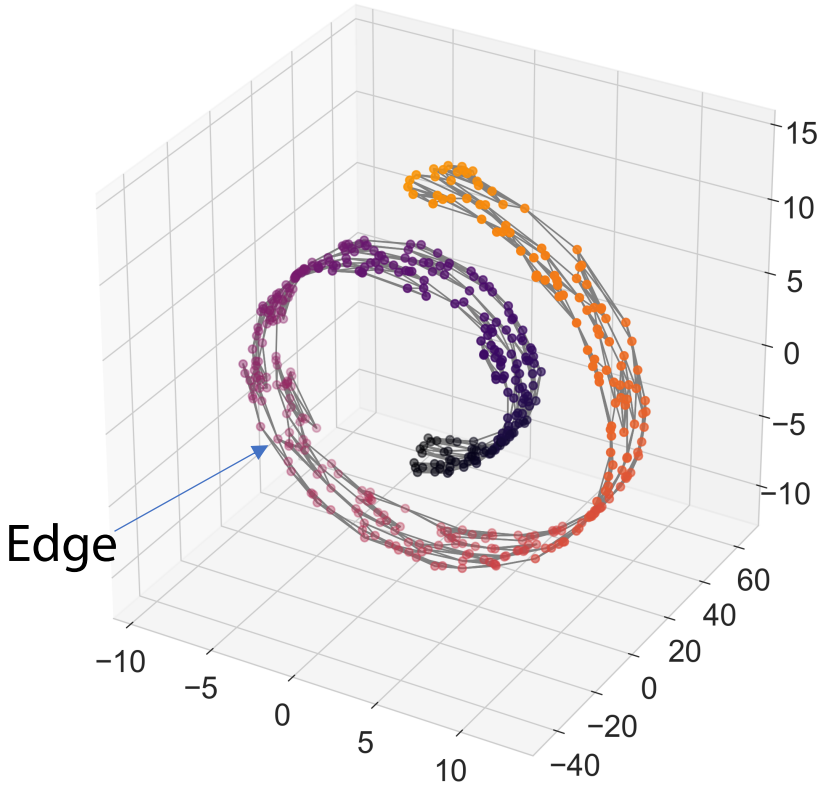
True manifold



Data sampled from manifold



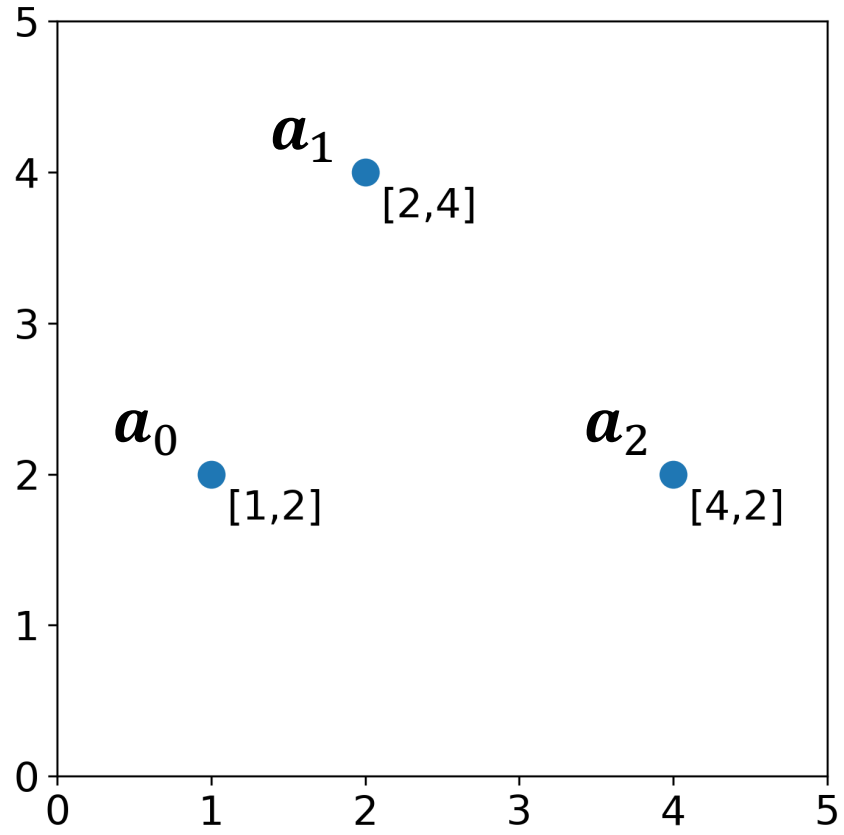
How do we find the data manifold?



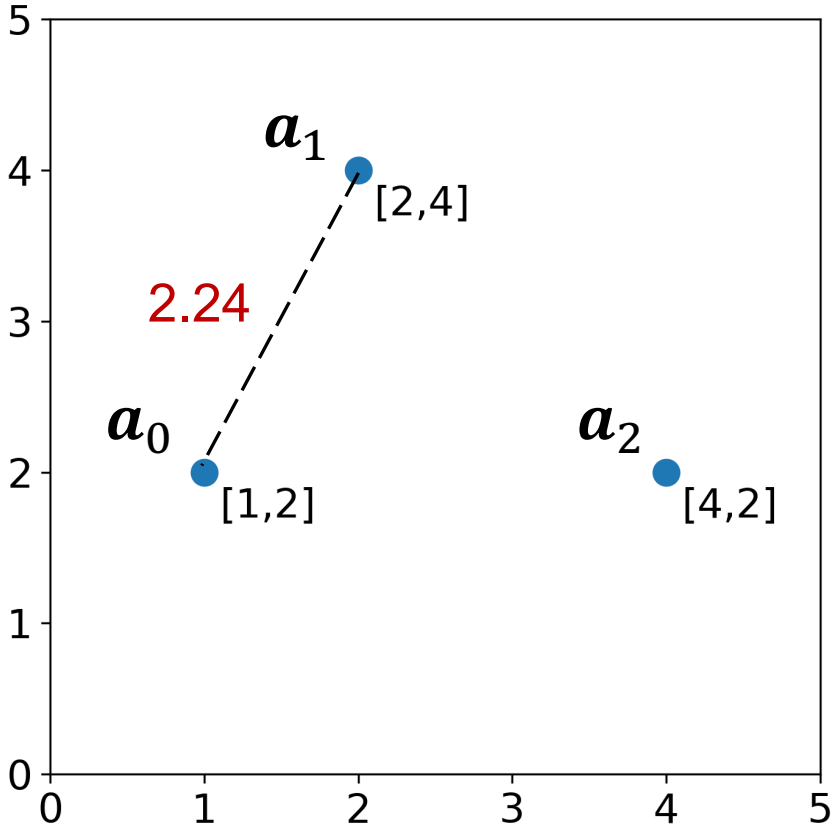
If you connect the nearest neighbors of any data point with a line (also called EDGE) it forms a mesh of the data that can be unrolled to find the shape.

This is called a nearest neighbors graph

How far away is each point from $\boldsymbol{a}_0$?



Euclidean Distance



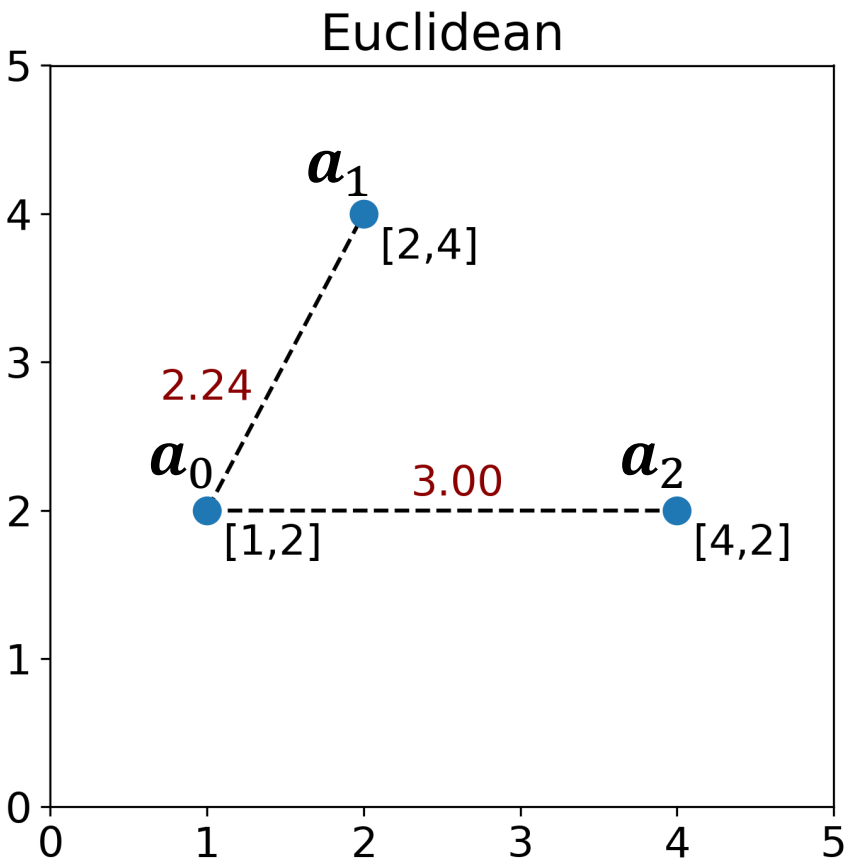
$$d_{euclidean}(a_0, a_1) = \|a_0 - a_1\|_2^2$$

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

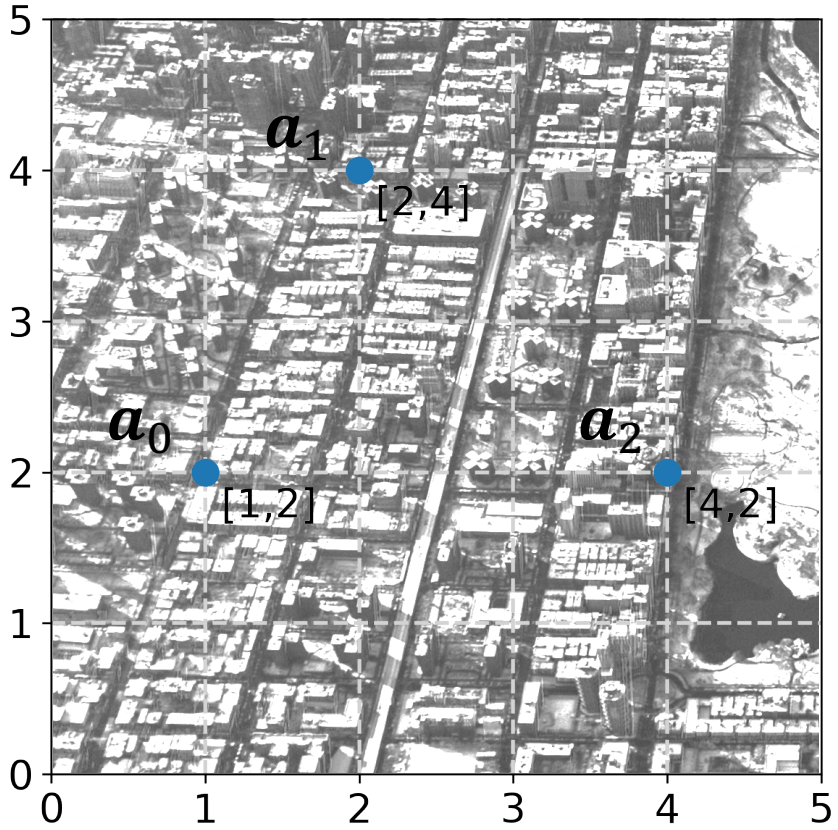
$$= \sqrt{(1 - 2)^2 + (2 - 4)^2}$$

$$\approx 2.24$$

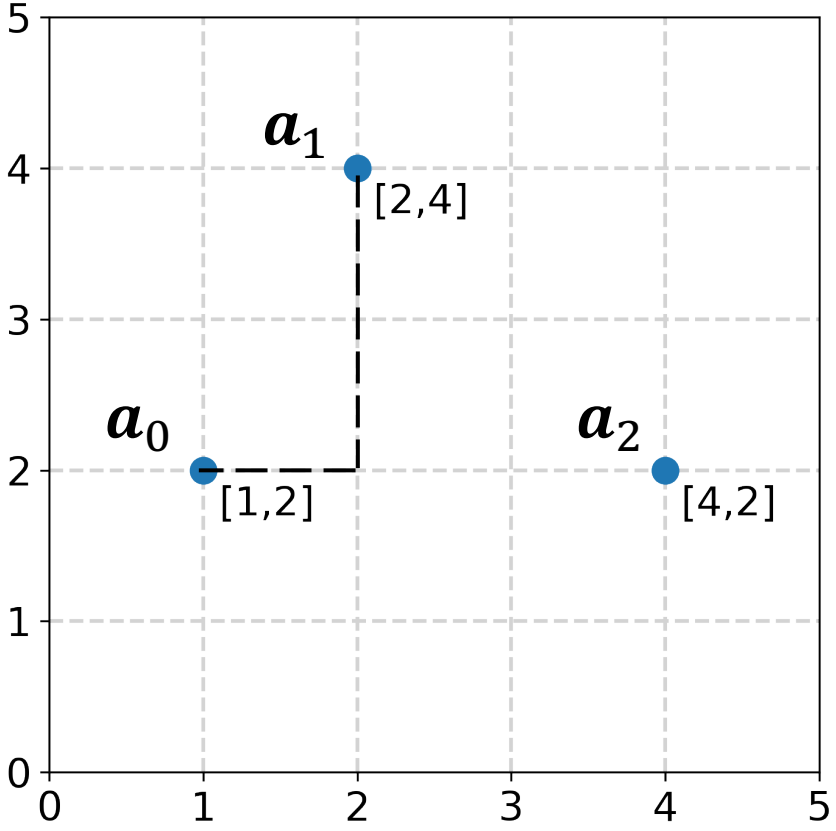
How far away is each point from a_0?



How far away is each point from a_0?



Manhattan Distance



Manhattan distance

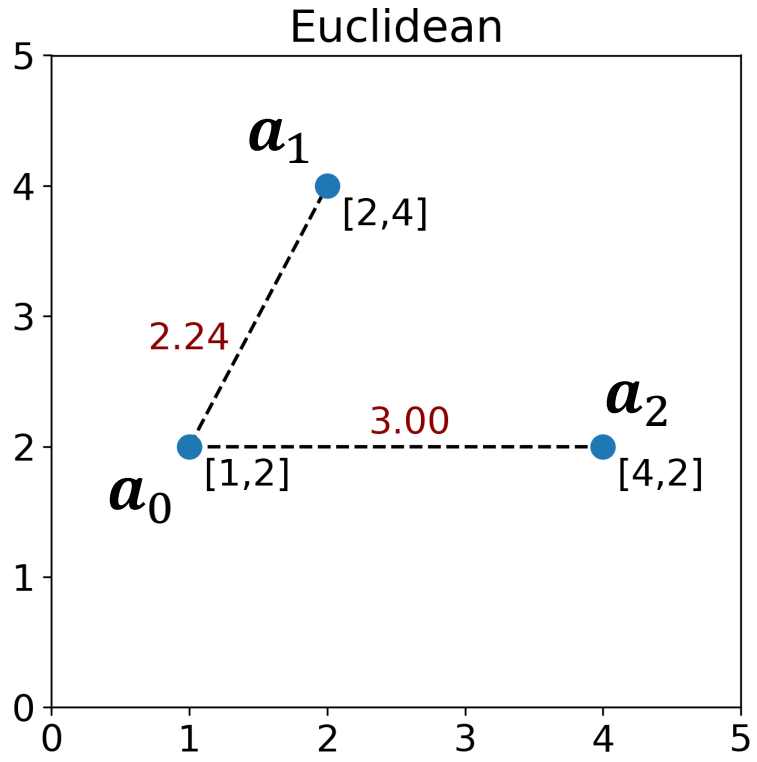
$$d_{manhattan}(a_0, a_1) = |a_{0,1} - a_{1,1}|$$

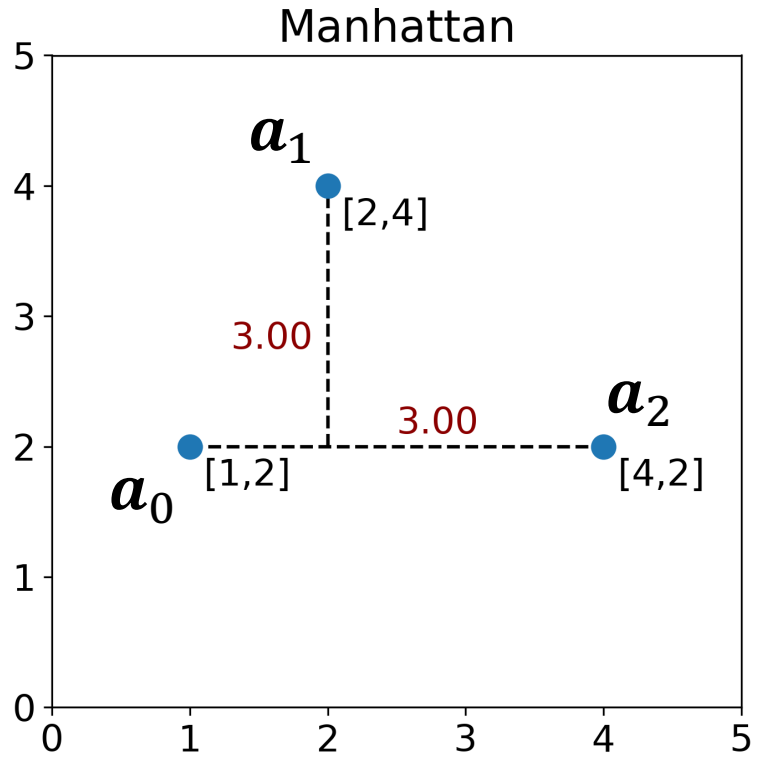
$$= |a_{0,0} - a_{1,0}| + |a_{0,1} - a_{1,1}|$$

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

$$= 3$$

How far away is each point from a_0?



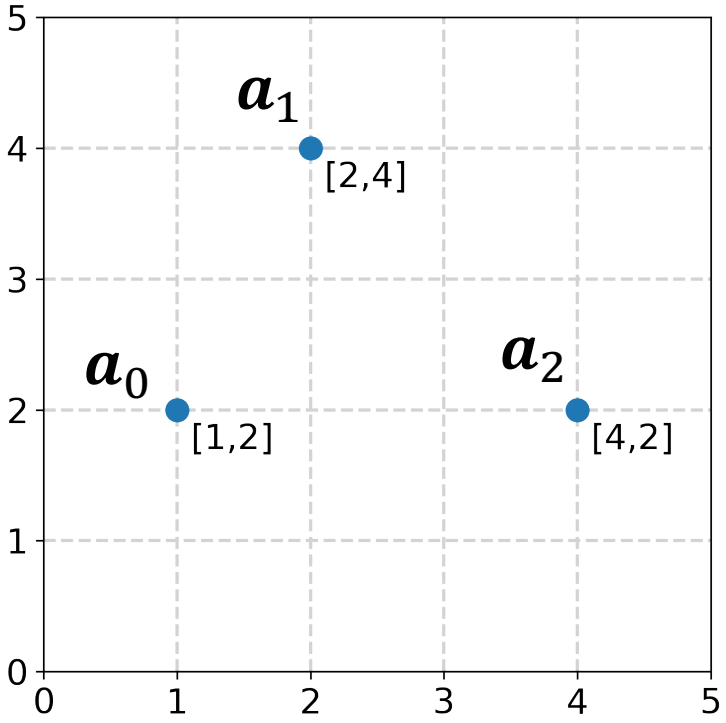


Distances

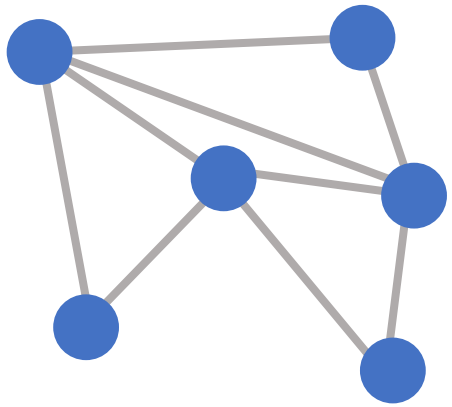
- There are many distances, Hamming, Euclidean, Cosine, etc
- Distances are functions that take two points and return a real number that is **positive or 0**
- Distances can be any function that is:
  - **Symmetric:** dist(a → b)=dist(b → a)
  - **Non-negative:** dist(a→b)>= 0
  - **Follow triangle inequality:**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

What is a “graph”?

Plot



Graph



$G = (V, E)$

● = V = vertices / nodes

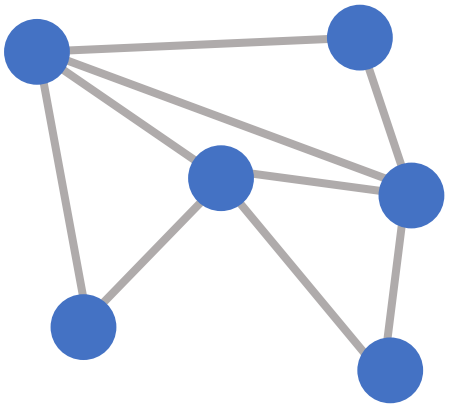
— = E = edges

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



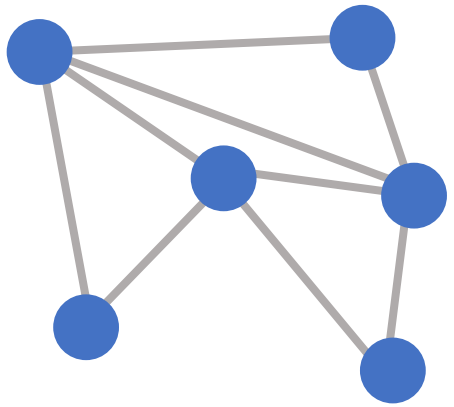
$G = (V, E)$ ● $= V =$ vertices — $= E =$ edges

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

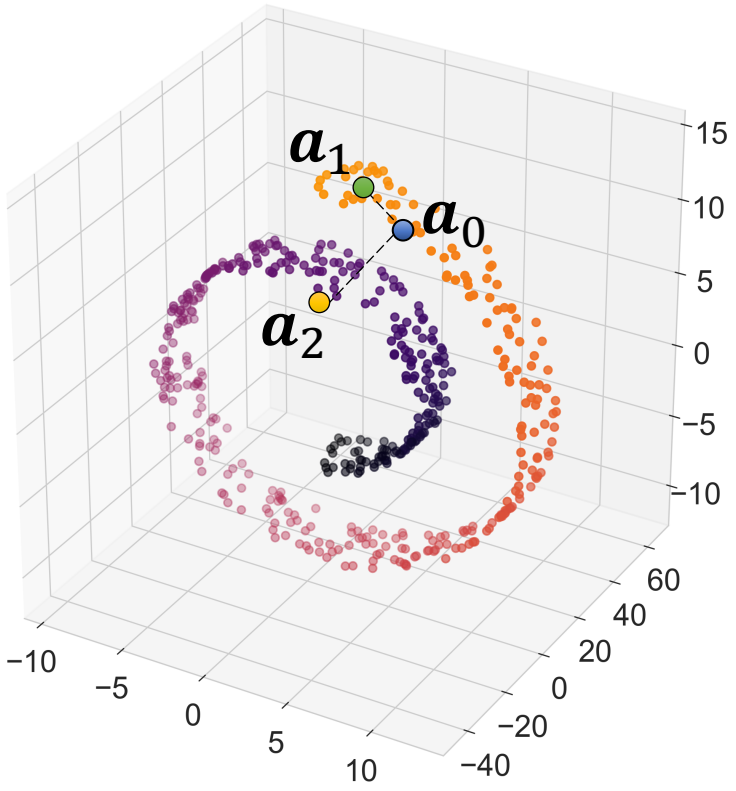


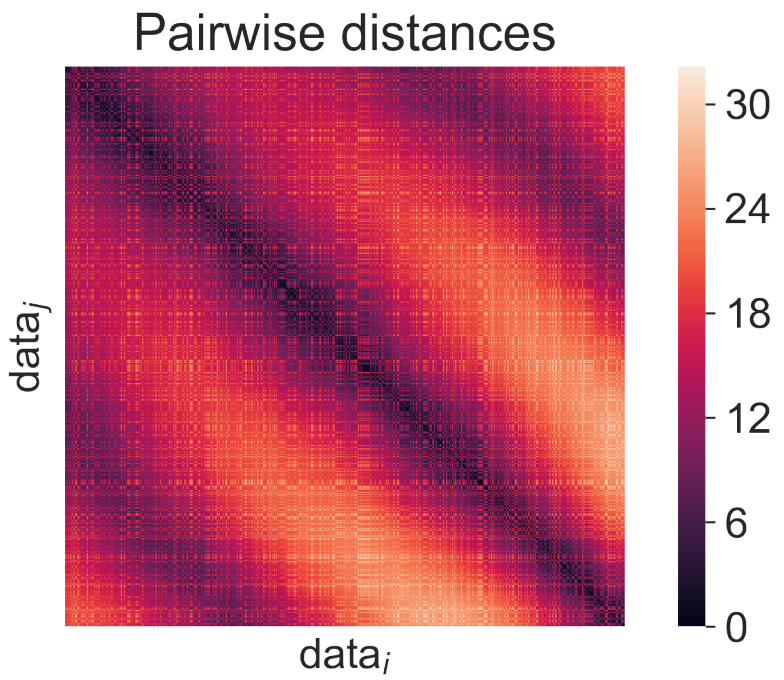
$G = (V, E)$

● $= V =$ vertices

— $= E =$ edges

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

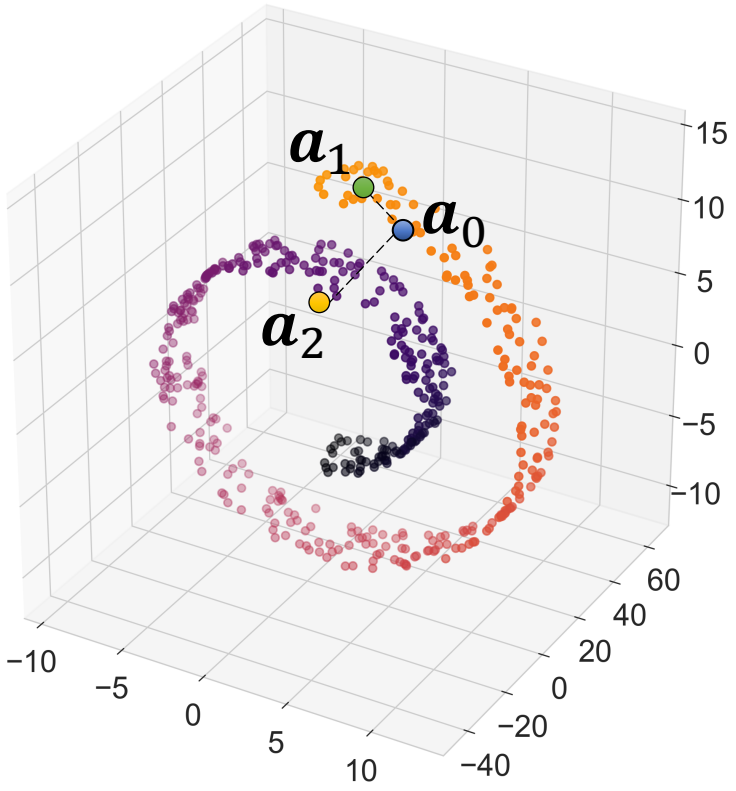


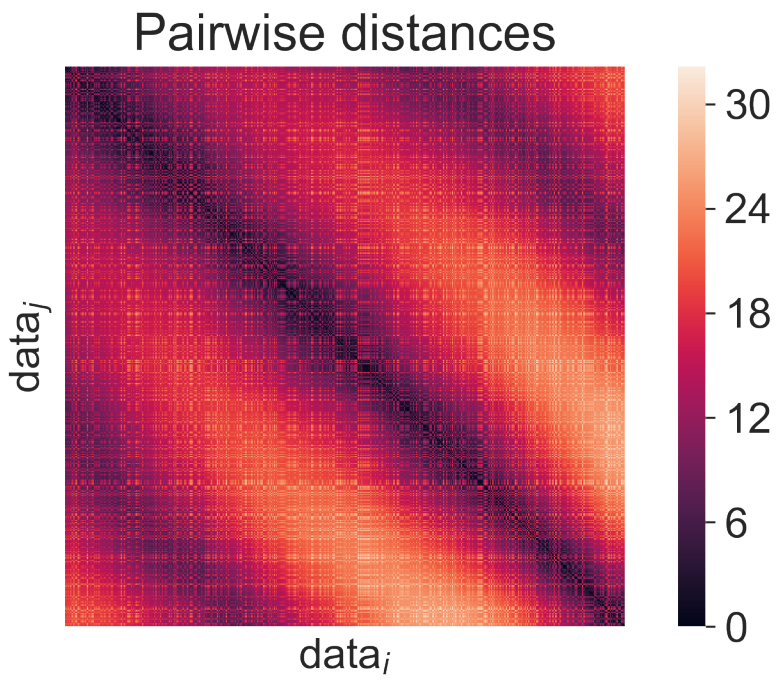


K Nearest Neighbors Graph

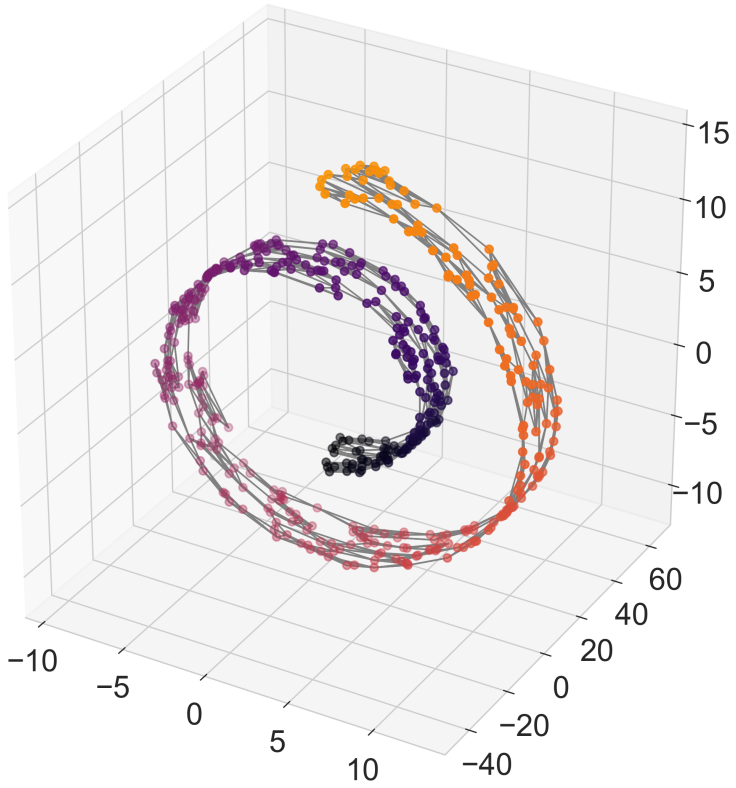
- To turn data into a graph, connect the K closest neighbors of a graph
- Measure closeness with your favorite distance
- Can have a fully connected graph with edges weighted by an affinity (negatively proportional to distance)

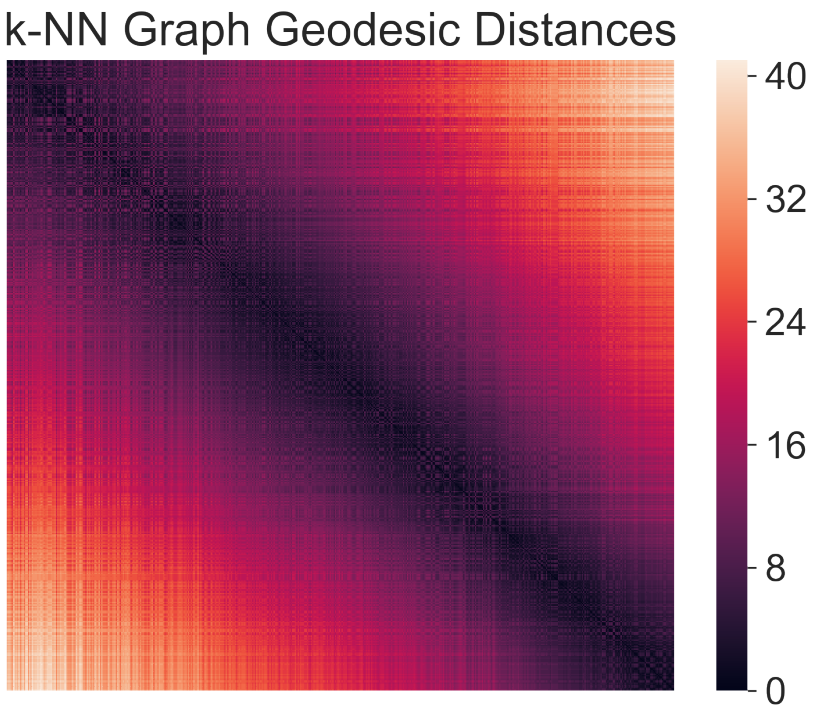
Swiss Roll


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

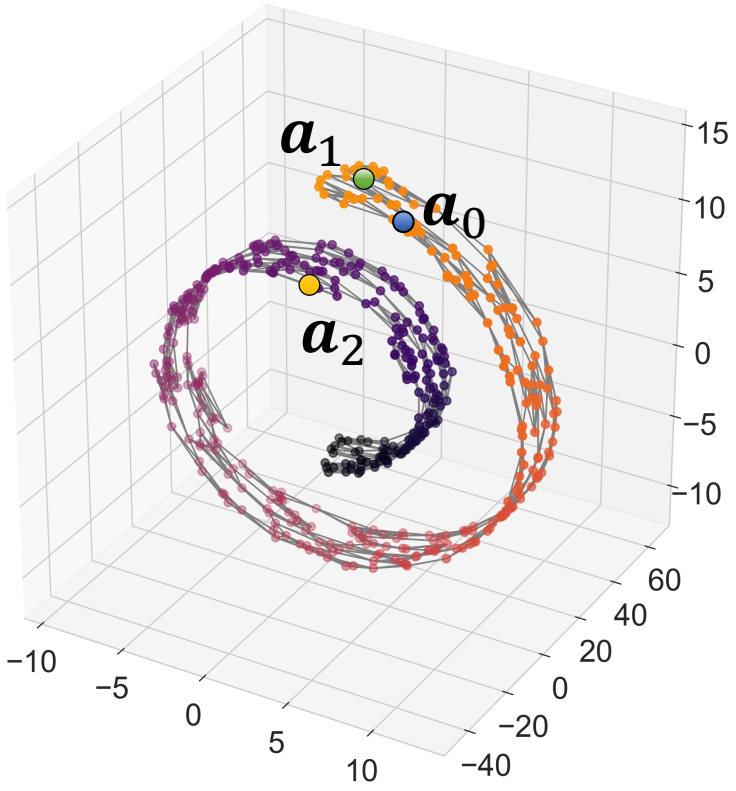

Pairwise distances
data$_j$
data$_i$
30
24
18
12
6
0

Swiss Roll K-NN

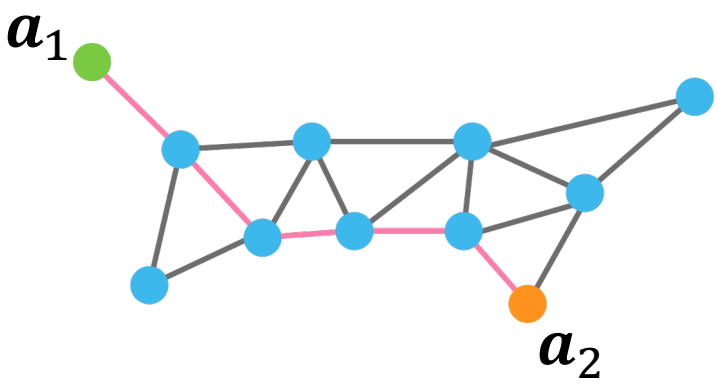




Paths or “walks”on graphs reveal dominant data manifold directions

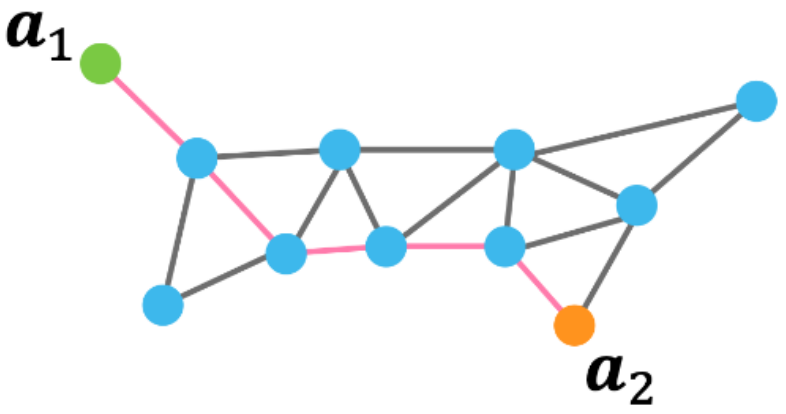


Shortest path between points



What is the shortest path distance between a1 and a2?

Shortest path between points



4

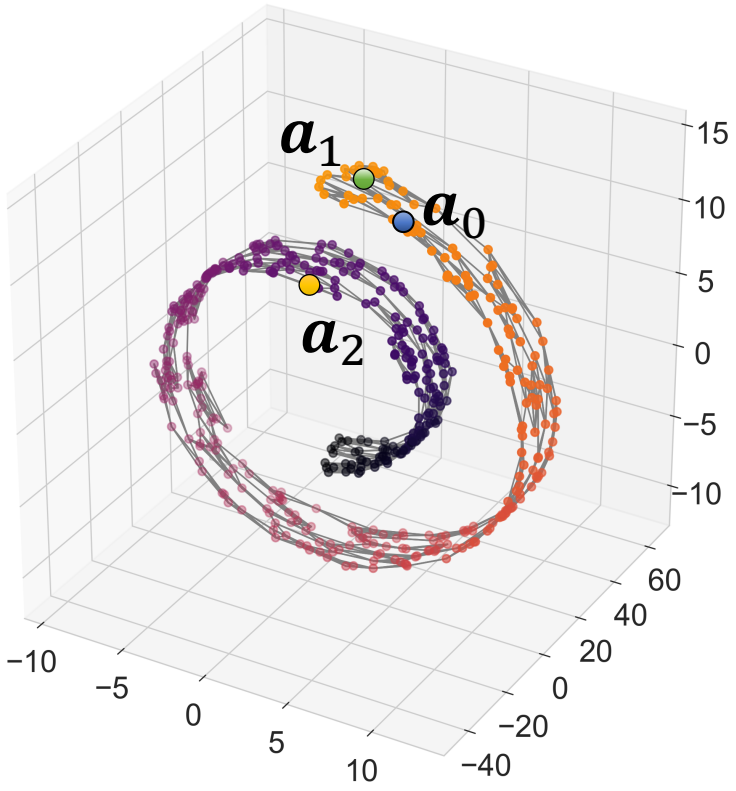
5

6

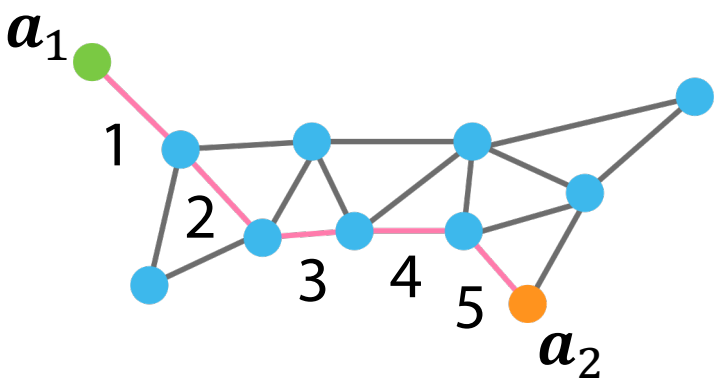
7

8

Graph walks approximate manifold distances

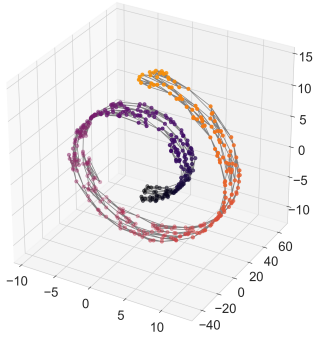


Shortest path between points

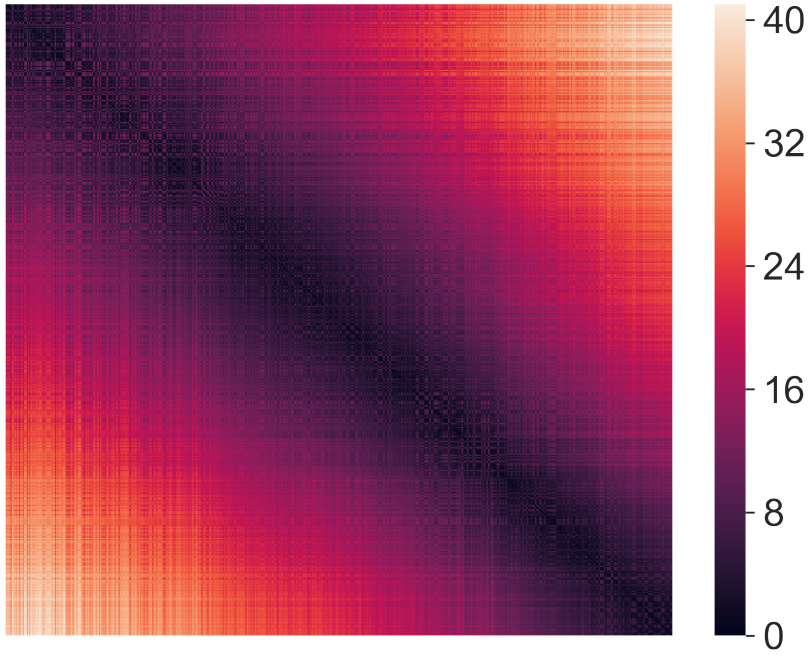


$$d_{geodesic}(a_1, a_2) = \text{shortestpath}(a_1, a_2)$$
$$= 5$$

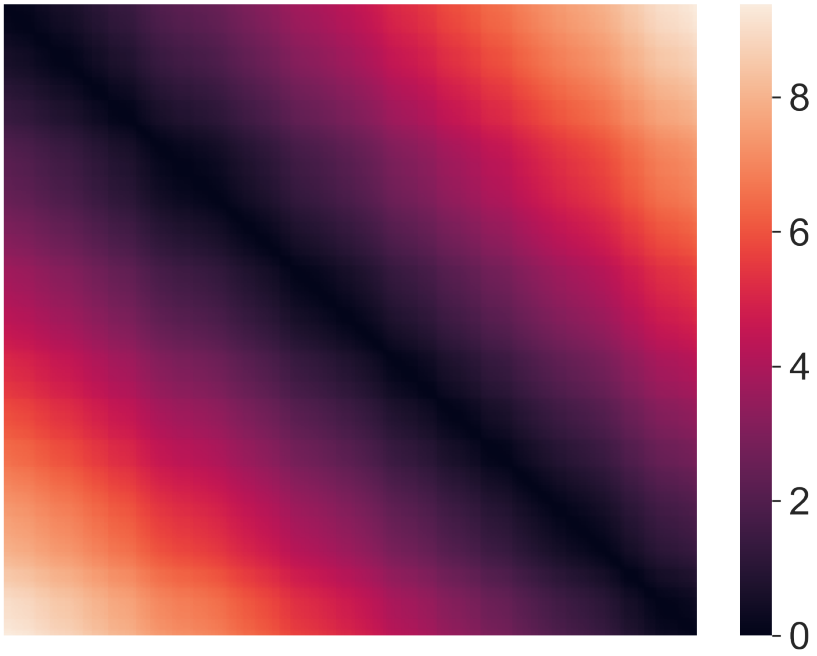
Graph walks approximate manifold distances



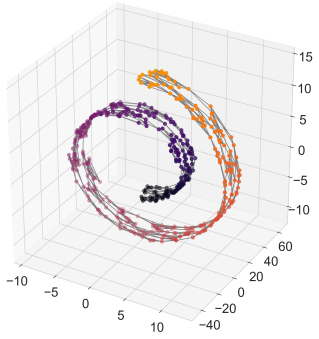
k-NN Graph Geodesic Distances



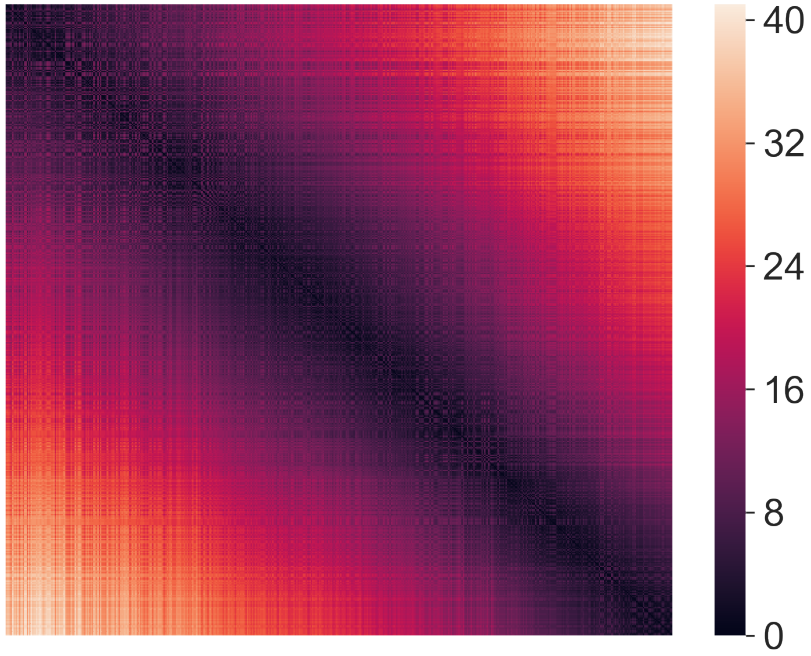
Manifold Distances



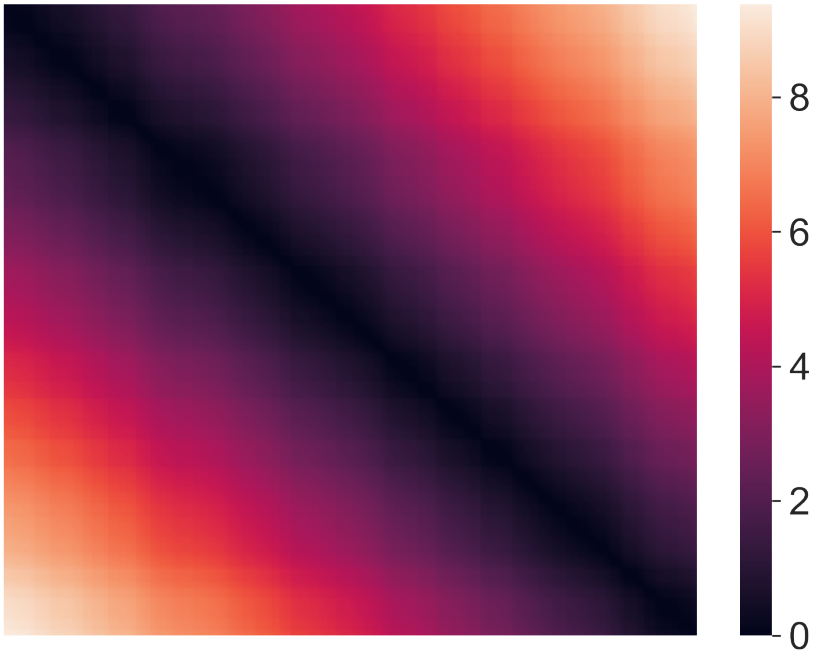
Graph walks approximate manifold distances



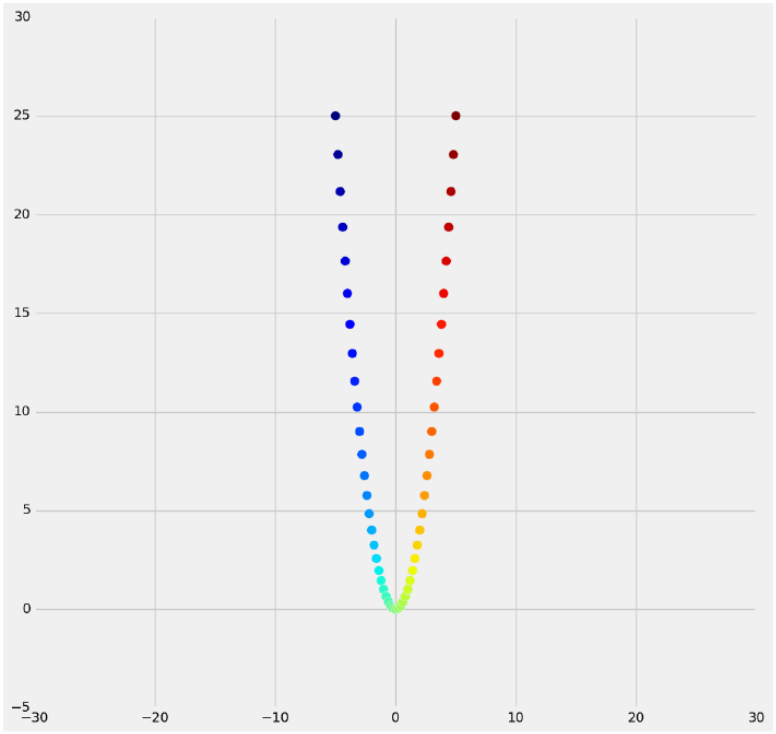
k-NN Graph Geodesic Distances

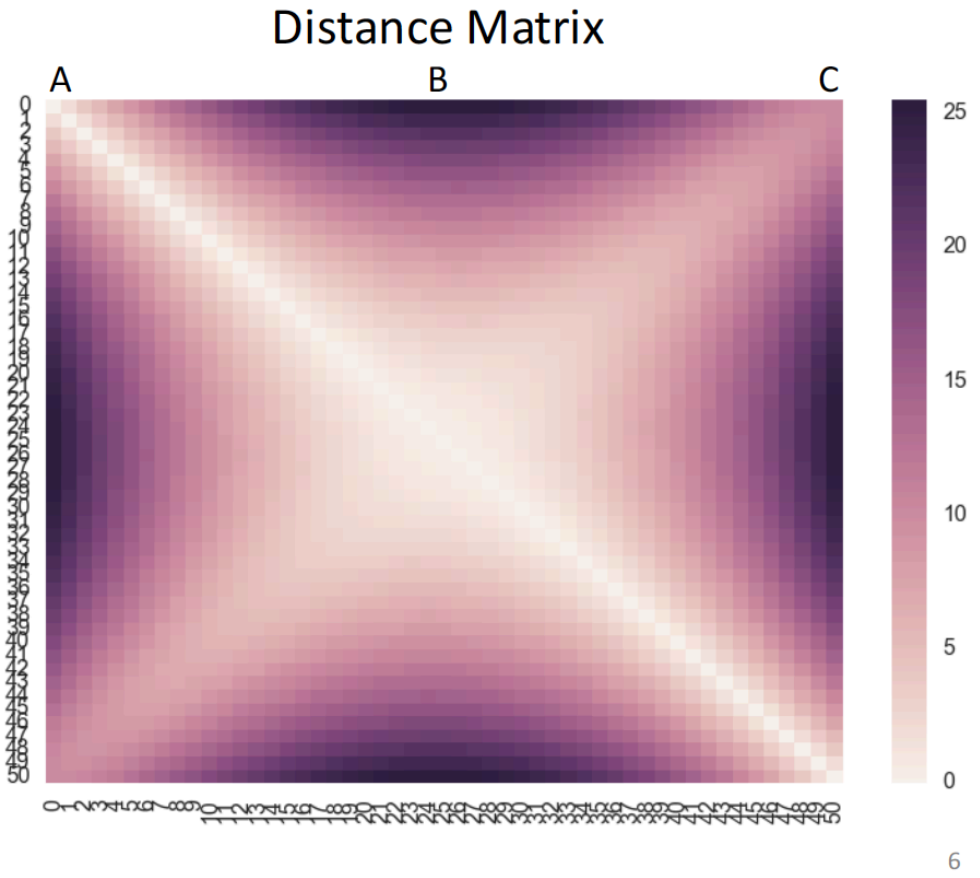


Manifold Distances

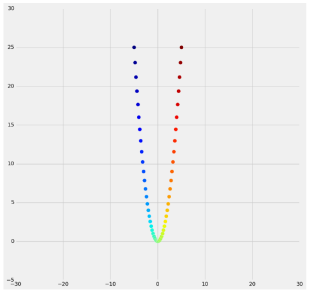


Kernel functions measure “similarity” or “affinity” on a graph

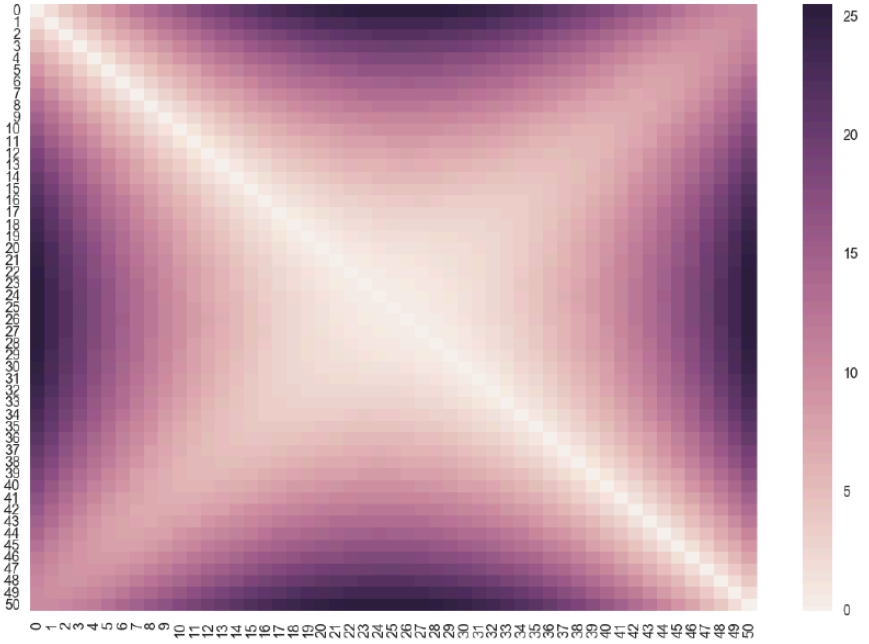




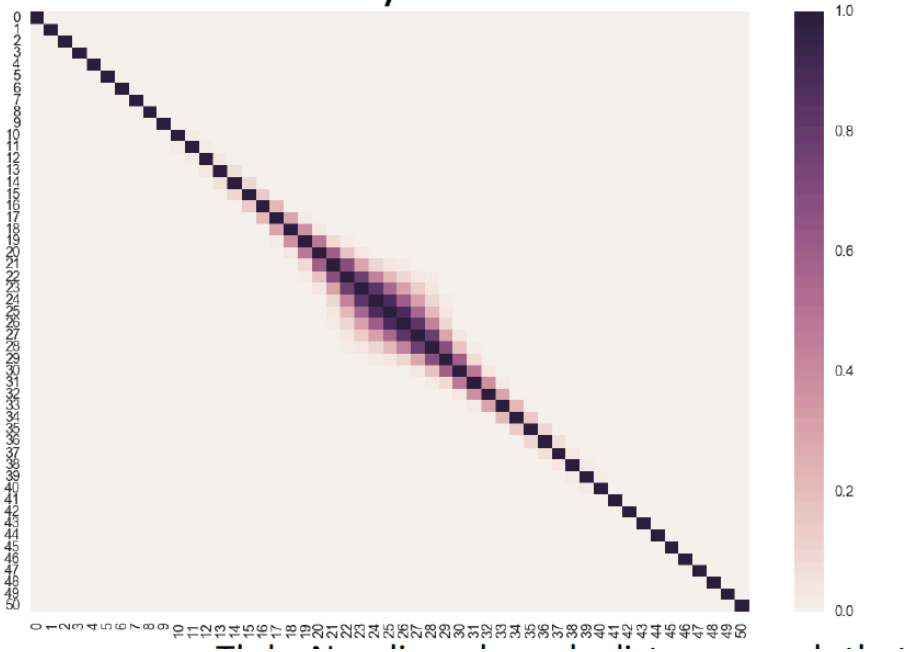
Affinity is the inverse of distance + locality



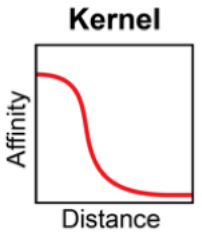
Distance Matrix



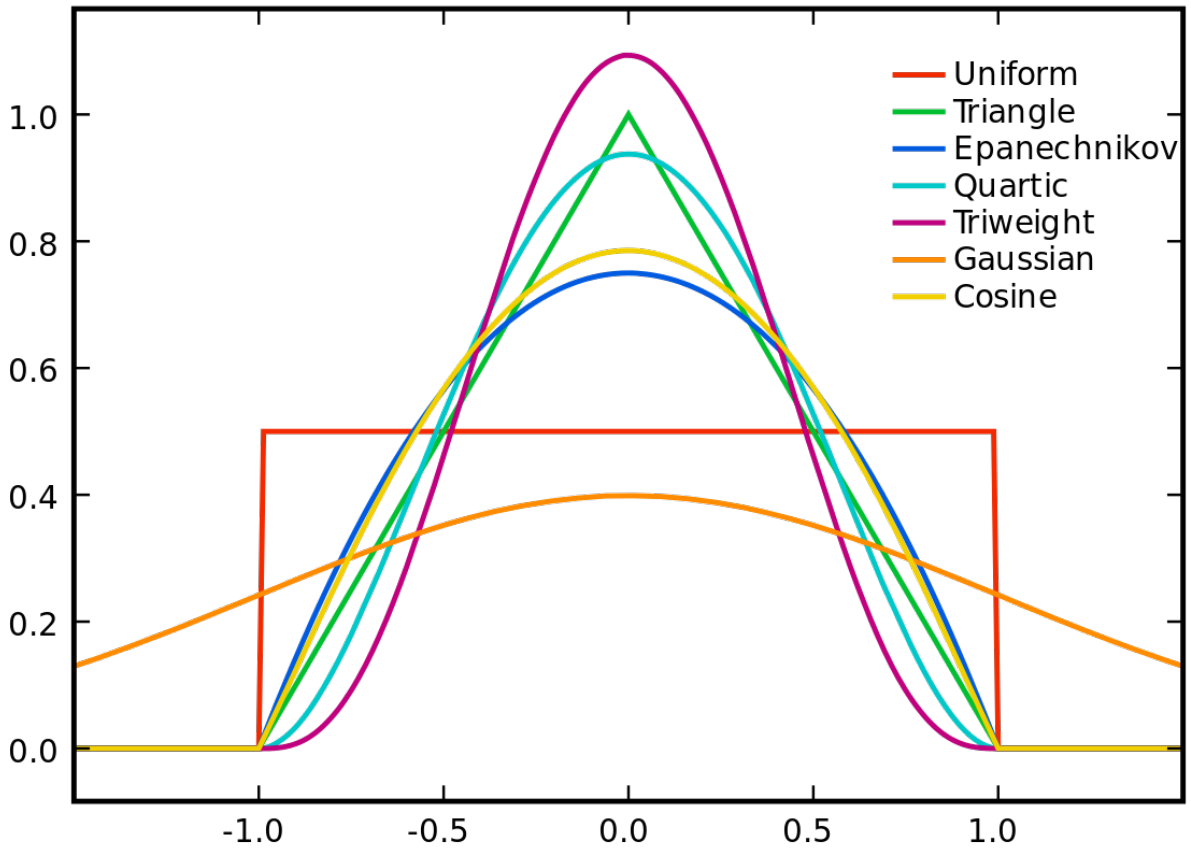
Affinity Matrix



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



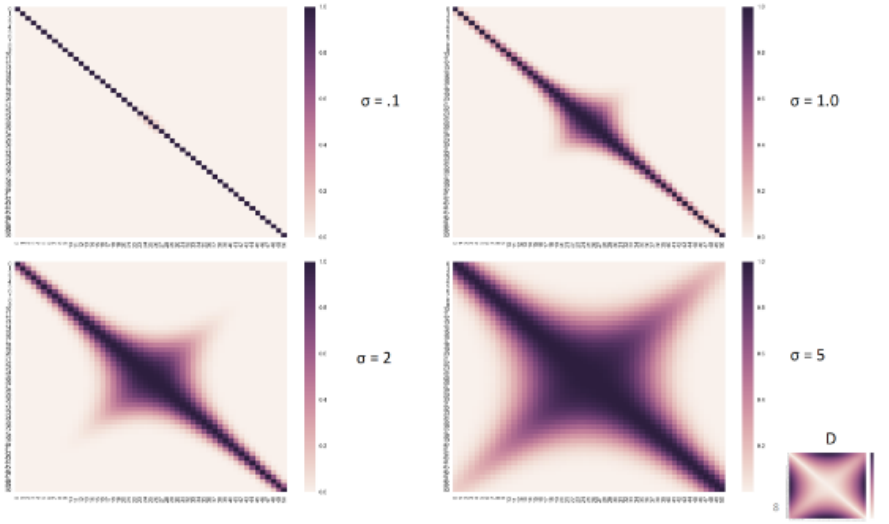
Distance to Affinity via Kernels



Real-valued
Non-negative
Symmetric

Affinities correlations in this hidden hypothetical space

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



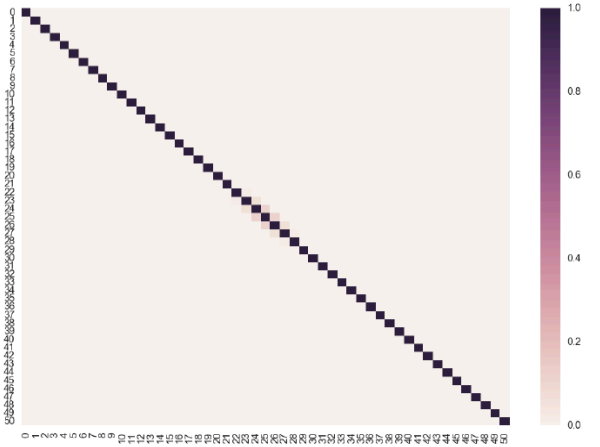
$\sigma = 0.1$

$\sigma = 1$

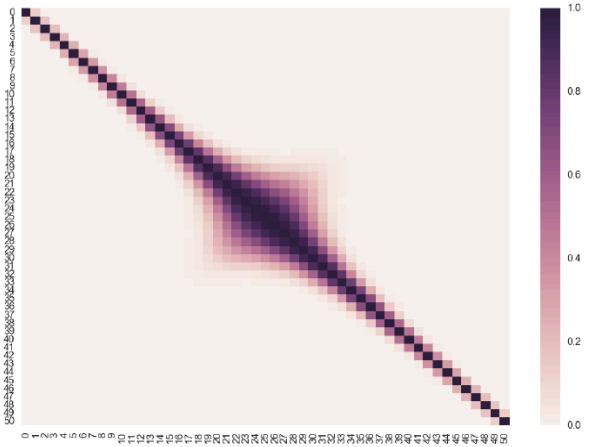
$\sigma = 2$

$\sigma = 5$

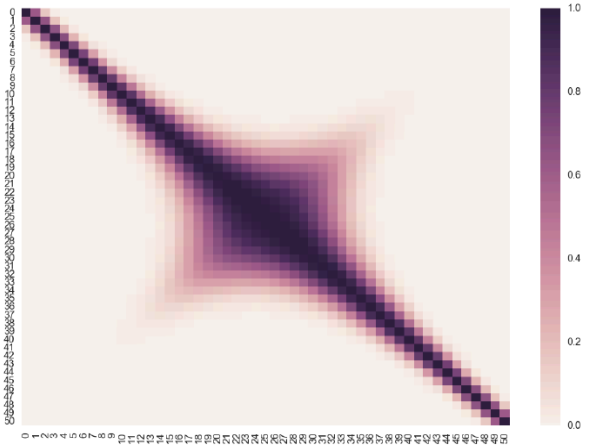
Effect of Changing Kernel Width



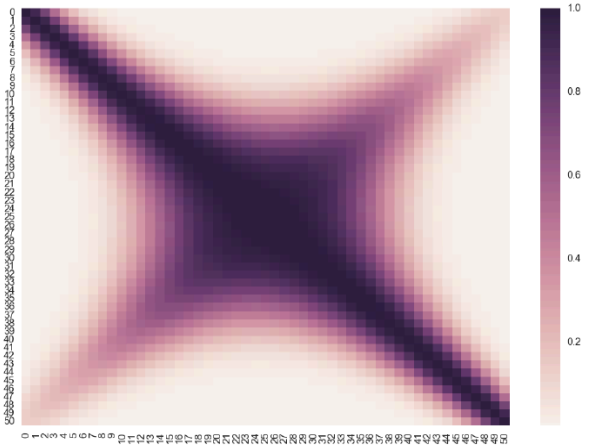
σ = .1



σ = 1.0



σ = 2

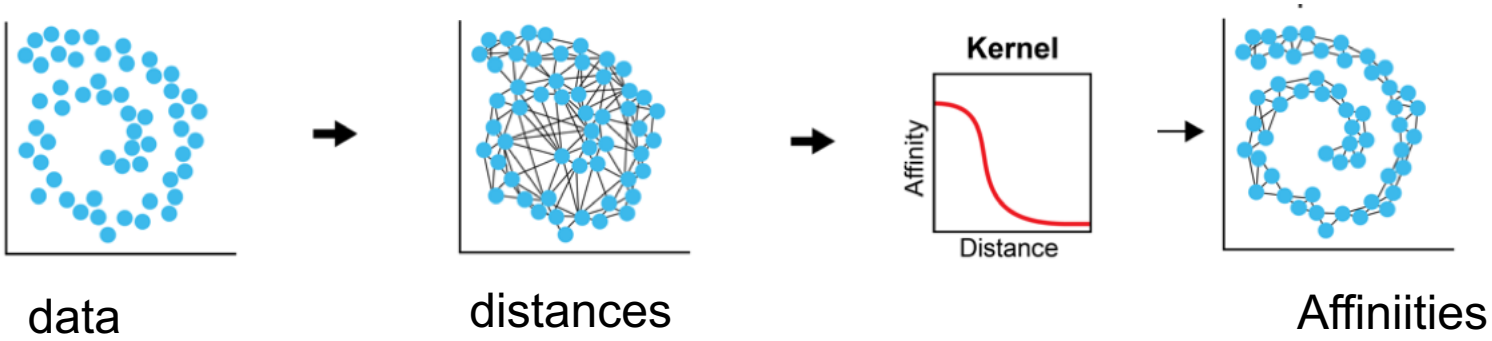


σ = 5

D

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

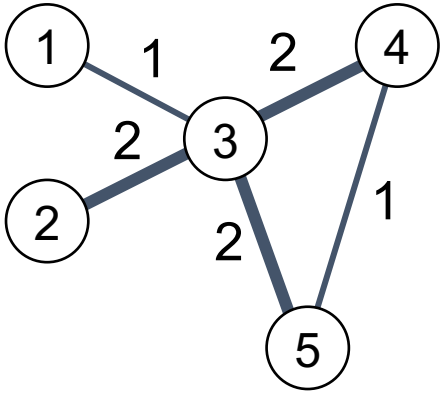
Kernel PCA



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

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

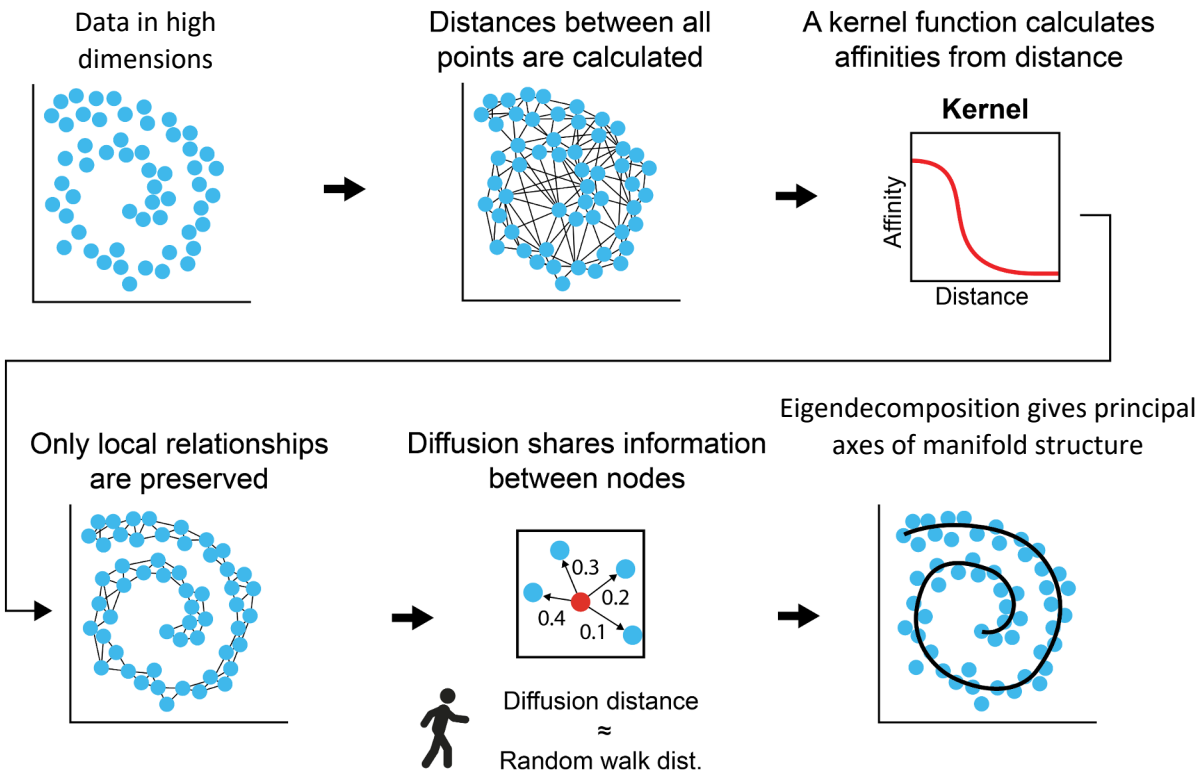
$$\mathcal{L} = \mathbf{D} - \mathbf{A} = \begin{vmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ -1 & -2 & 7 & -2 & -2 \\ 0 & 0 & -2 & 3 & -1 \\ 0 & 0 & -2 & -1 & 3 \end{vmatrix}$$

Laplacian Eigenmaps

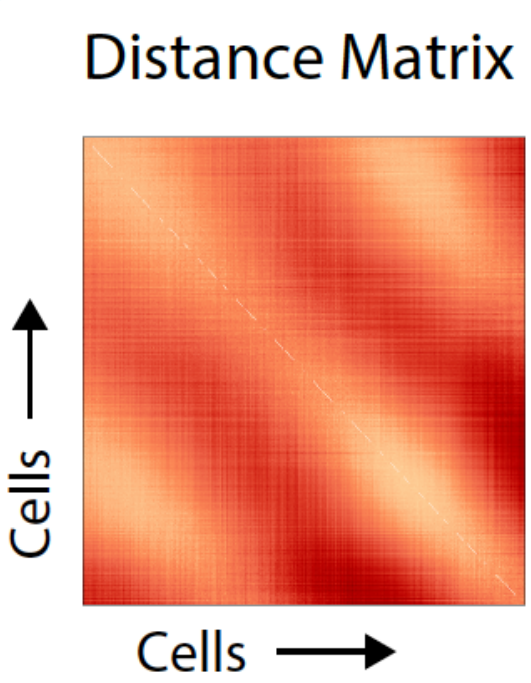
$$\mathcal{L} = \mathbf{D} - \mathbf{A} = \begin{vmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ -1 & -2 & 7 & -2 & -2 \\ 0 & 0 & -2 & 3 & -1 \\ 0 & 0 & -2 & -1 & 3 \end{vmatrix}$$

Eigenvectors of Laplacian matrix form Laplacian Eigenmaps

Diffusion Maps

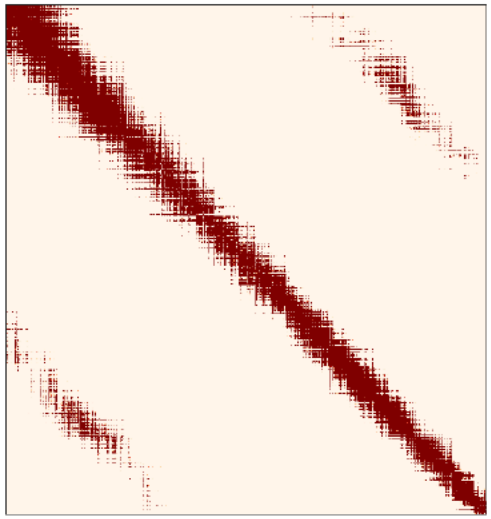


Distance Matrix

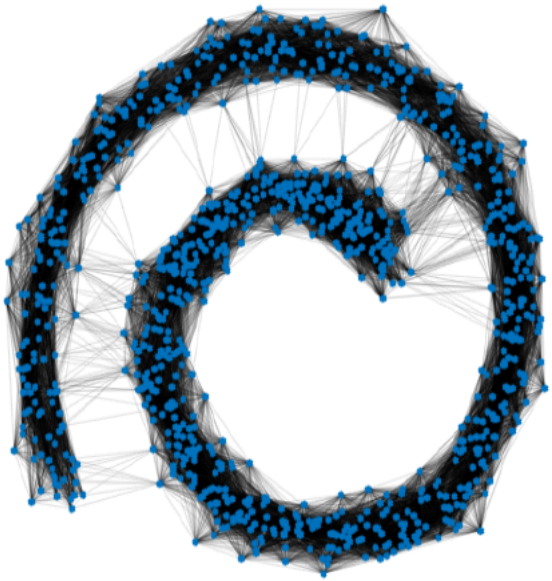


Entries are distances $D(x_i, x_j) = \sqrt{||x_i - x_j||}$

Affinity Matrix



(Gaussian Kernel)



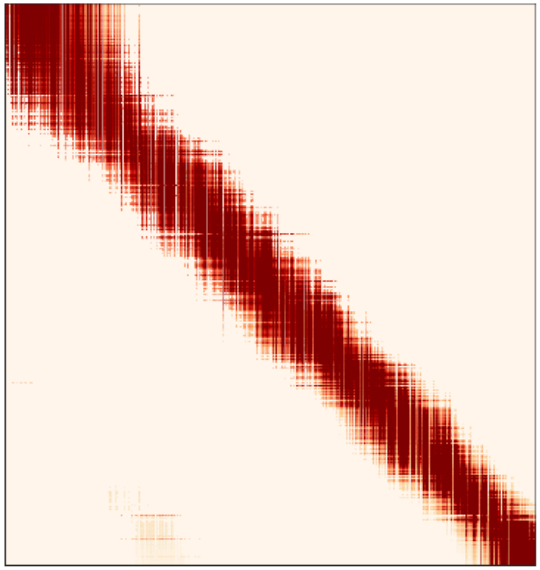
Graph Representation

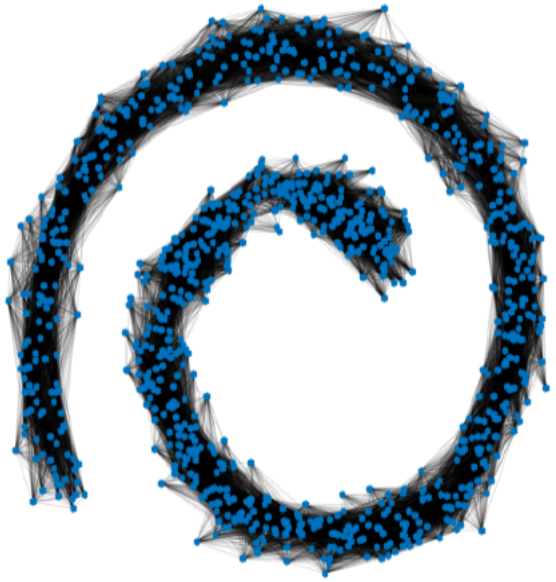
Entries are affinities

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

Powered Markov Matrix

Powered Markov Matrix



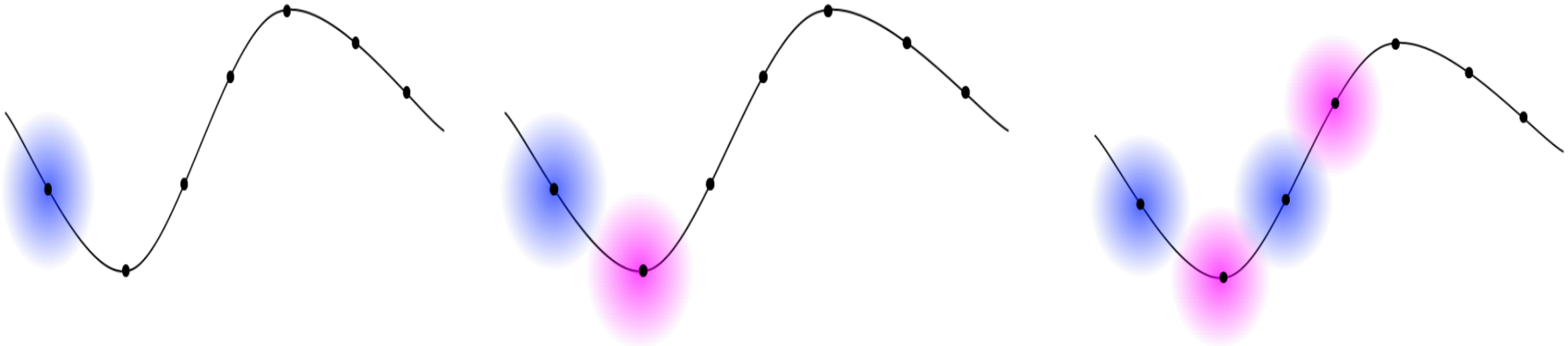


Entries are row normalized affinities

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

What does Powering Do?

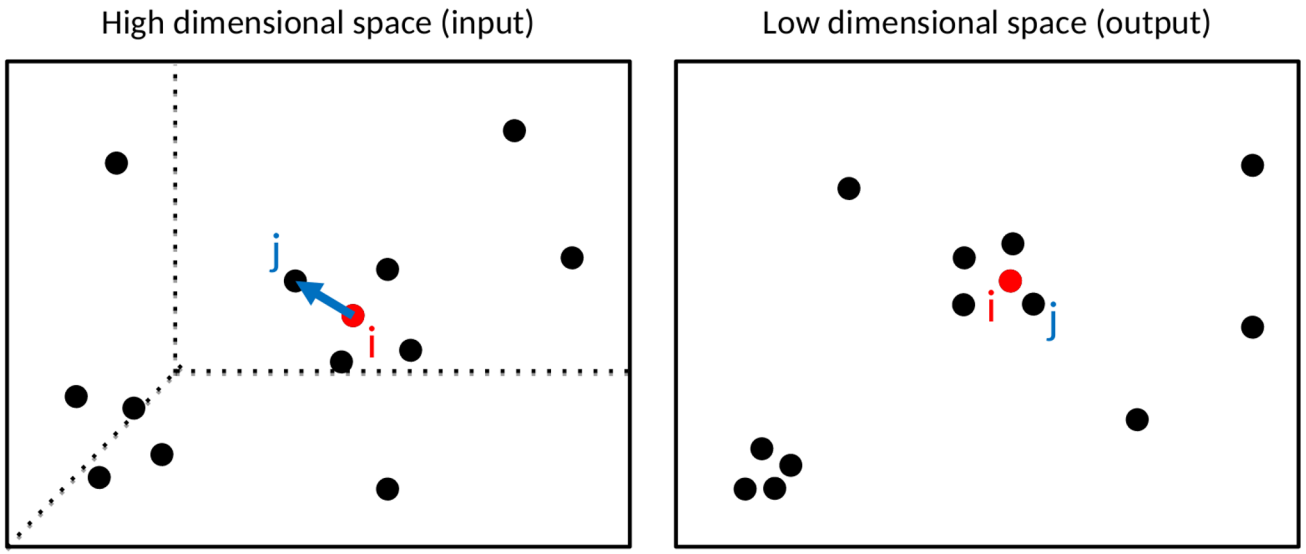
- Smooths and infers connectivity through data



Post Kernel PCA

- These methods something other than eigendecompose the affinity matrix
  - **tSNE:** greedy method for preserving near neighbors
  - **PHATE:** creates a new distance matrix from a distance between diffusion probabilities of points

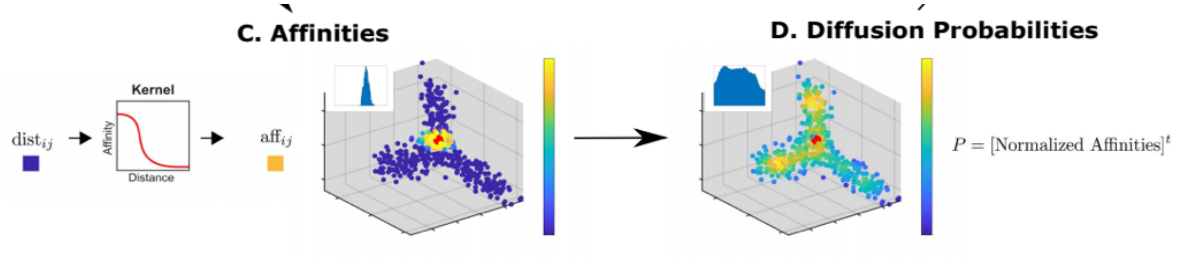
tSNE (and UMAP)



Goal: Maintain the local neighborhood of each cell
Approach: Place points randomly in a 2D plane and move them around using greedy local moves (stochastic gradient descent)

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

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

……

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

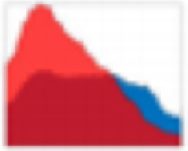
…

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

…

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

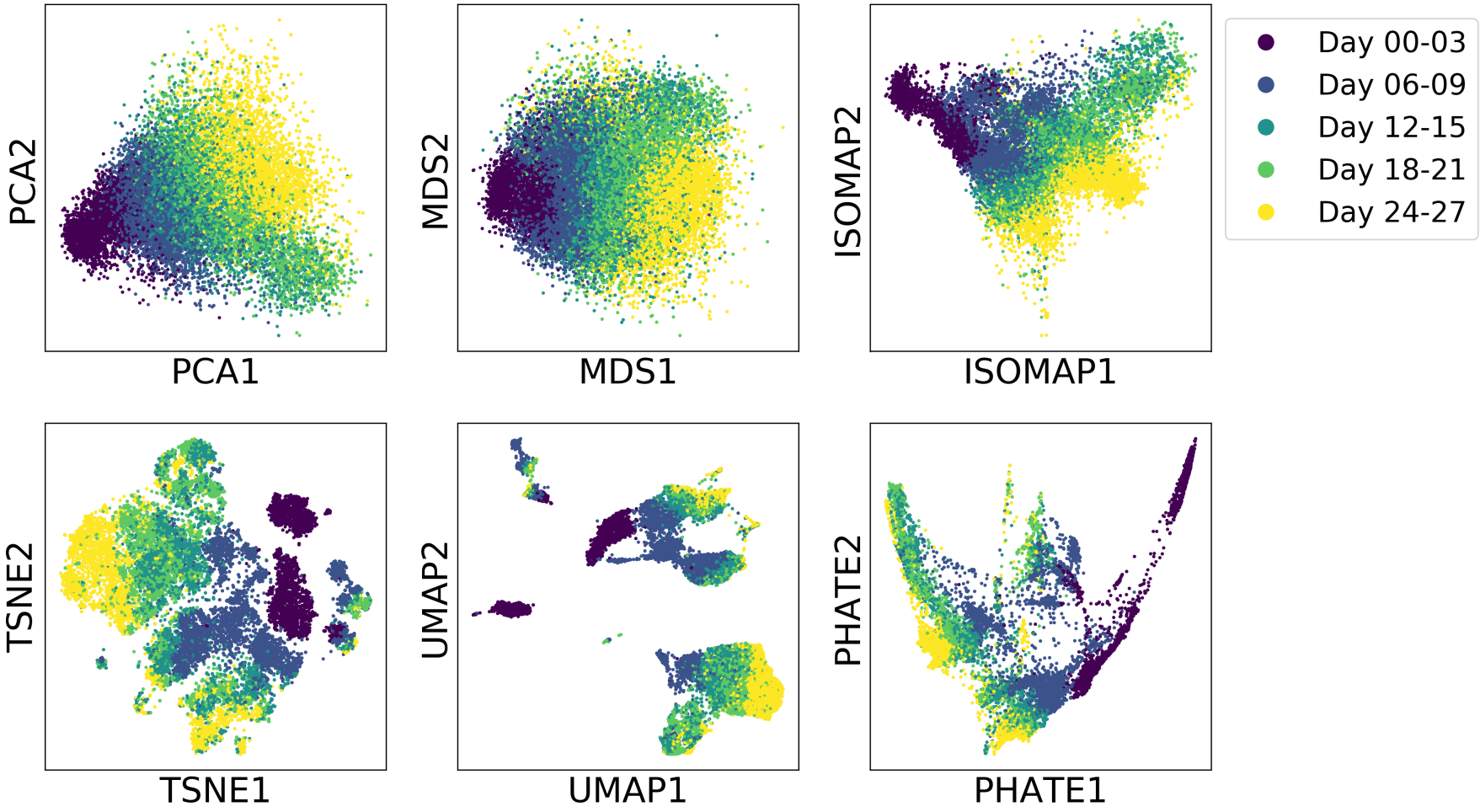
$$\mathrm{dist}_{ij} = \sqrt{\| \log P_i - \log P_j \|^2}$$



- Step 4: Embed with MDS

- This reorganizes information in a diffusion map into 2D

Comparing visualization algorithms on single-cell data



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
- PHATE, UMAP, tSNE are all manifold-learning algorithms for visualizing single-cell data

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

Top