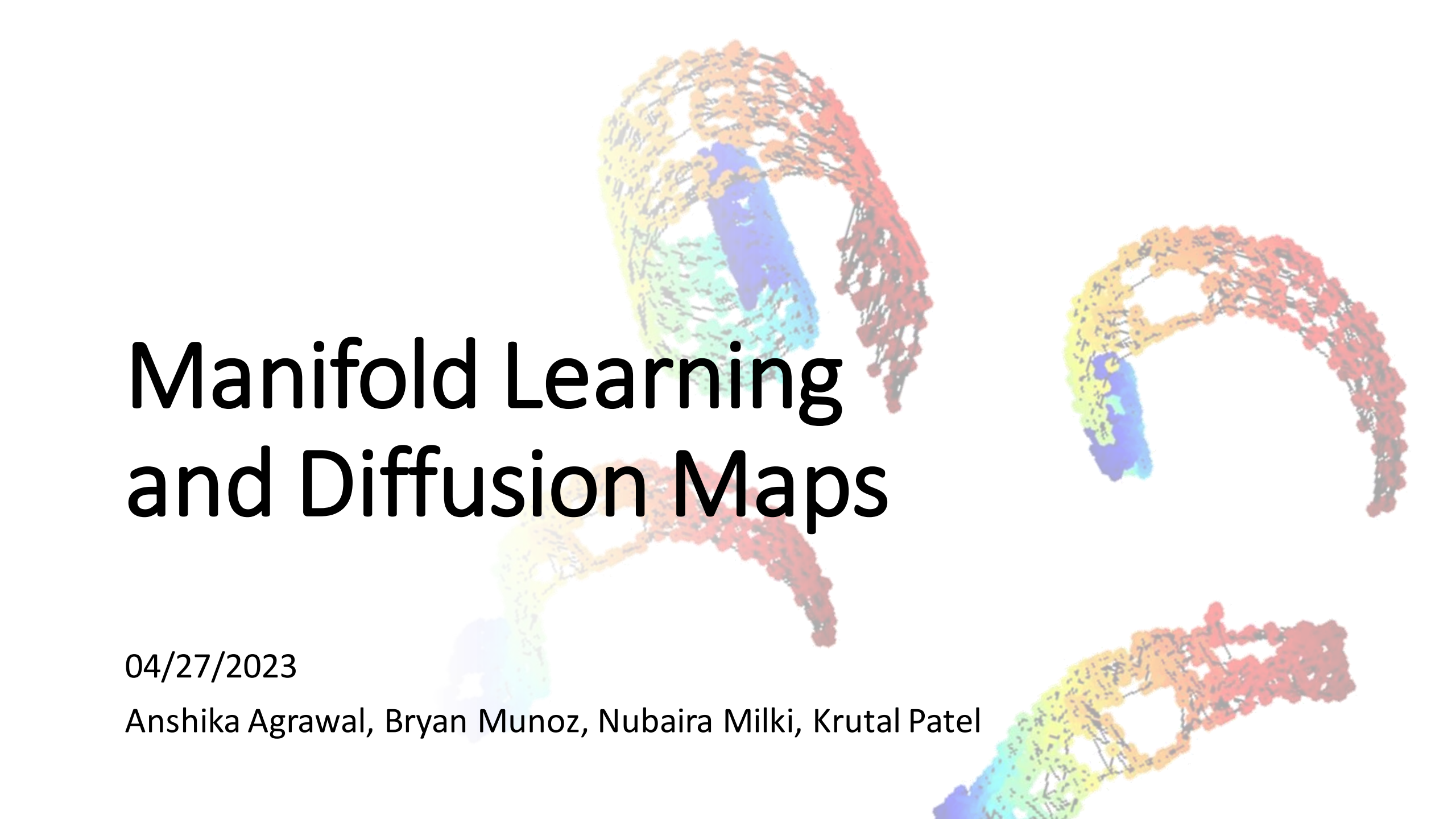


# Manifold Learning and Diffusion Maps

04/27/2023

Anshika Agrawal, Bryan Munoz, Nubaira Milki, Krutal Patel

# Linear Dimensionality Reduction

- In general, we often have a collection of n data points with m dimensions grouped into k clusters

Goal: Determine and map the high dimensional data with their associated clusters to low dimension spaces

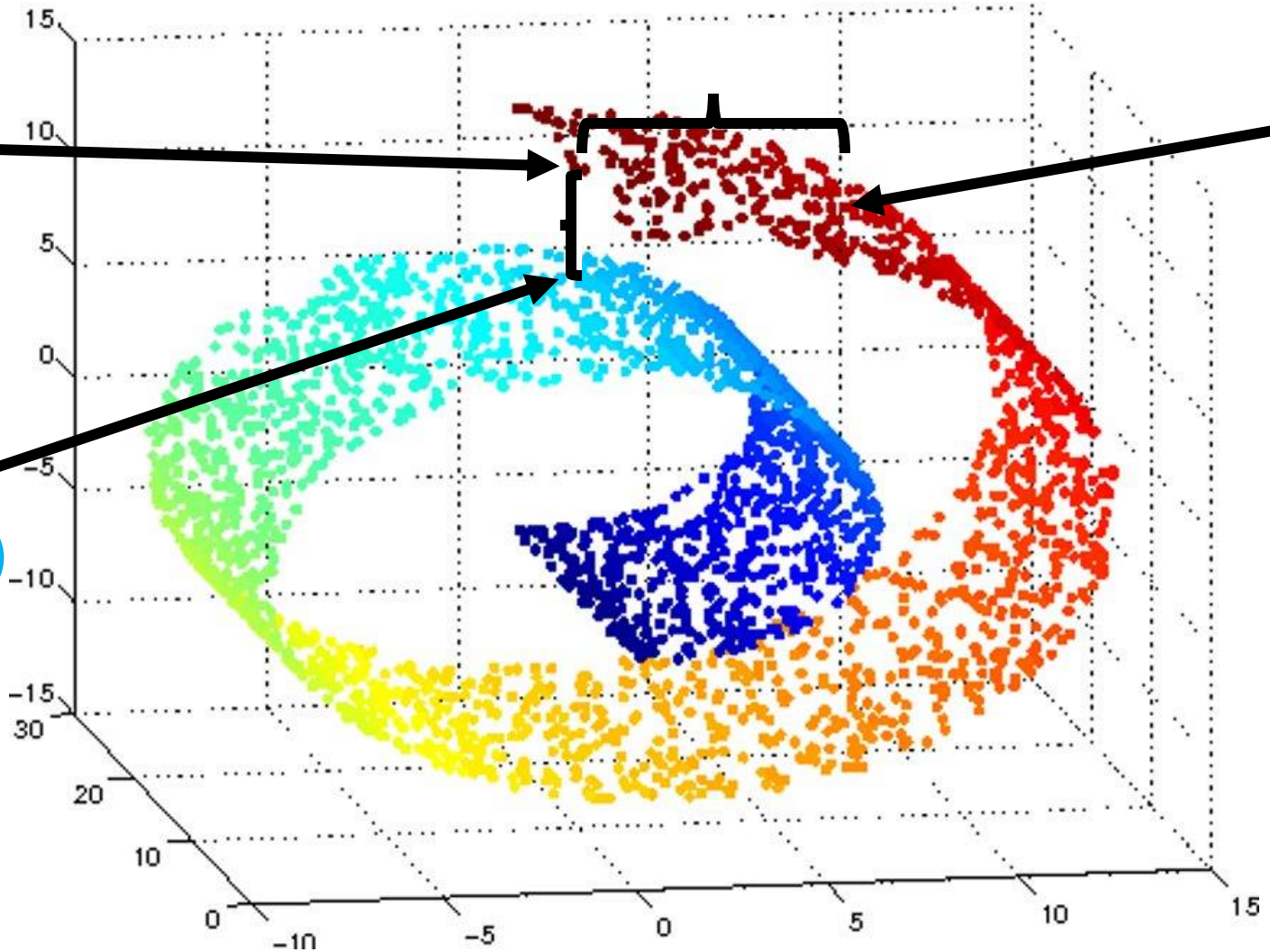
- Consider Linear Dimensionality Reduction Techniques (PCA)
  - Assume the data can be represented by a linear subspace
  - Does not do well at preserving the local structure (i.e. preserving neighbors)
- Computing the pairwise distances of data points to gauge "closeness" or "similarity" is often used in Dimensionality Reduction Methods

# Example: The Swiss Roll Dataset

Point A
(Dark Red Cluster)

Point B
(Light Blue Cluster)

Point C
(Dark Red Cluster)



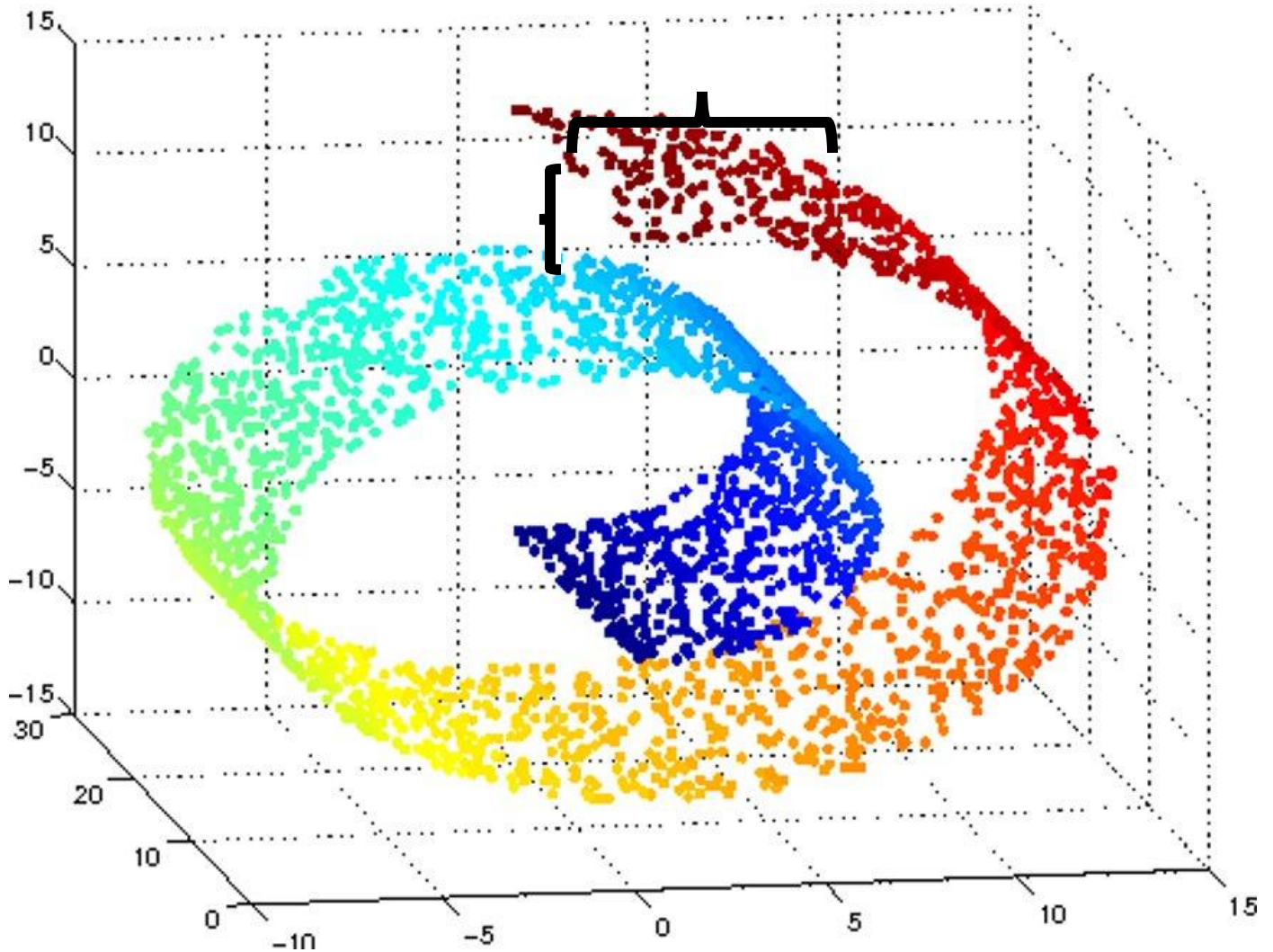
# Example: The Swiss Roll Dataset

- Considering the standard Euclidean distance:

  $||AB|| > ||AC||$

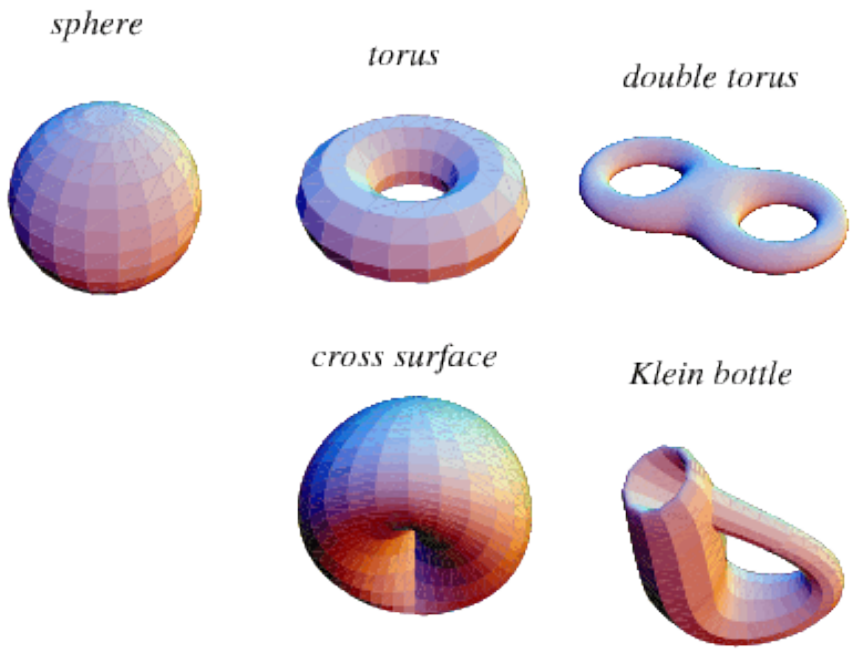
- Using PCA, this will give Point A and B are larger similarity then Point A and C (belong to the same cluster)

- But … this is clearly incorrect!

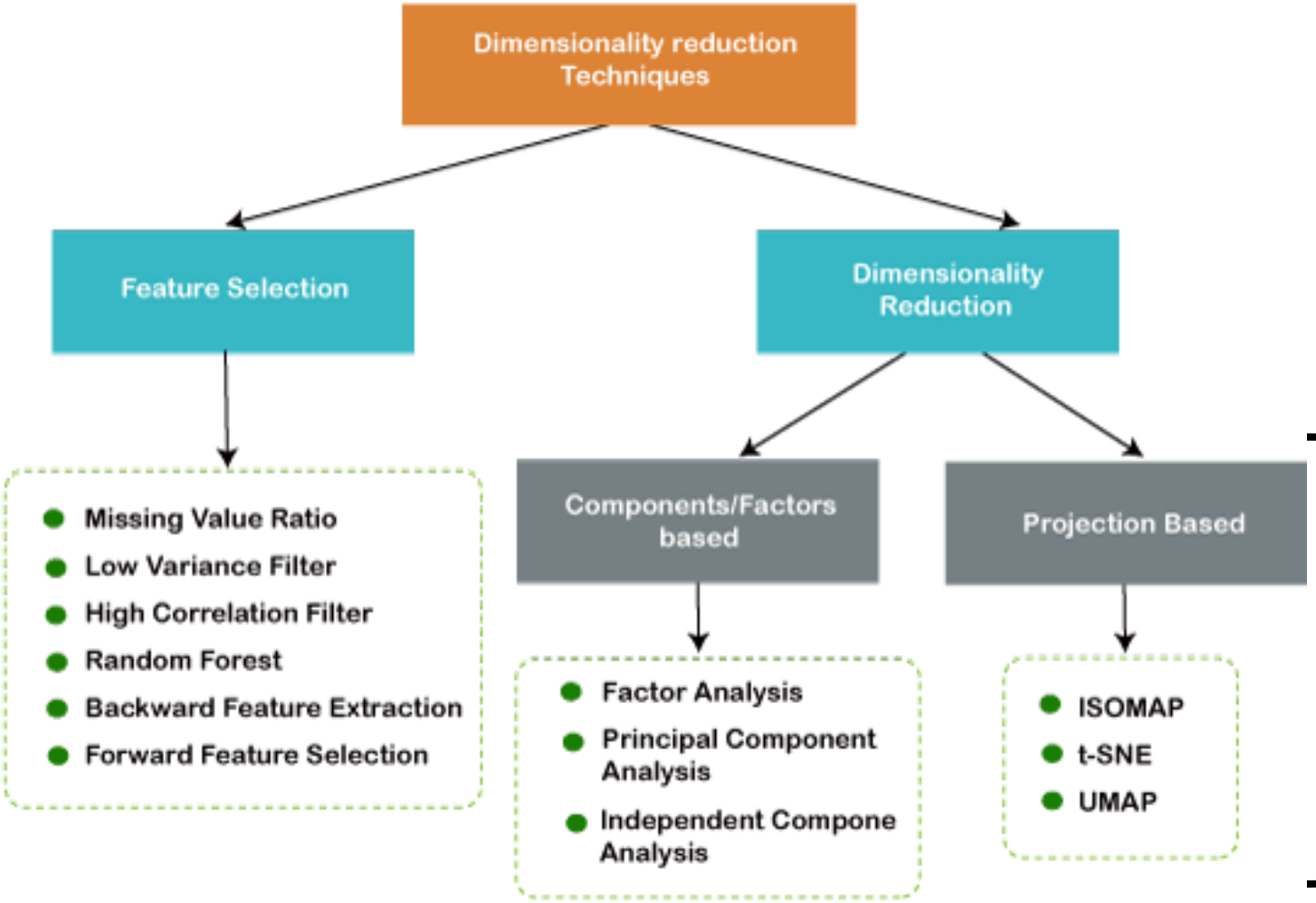


# Introducing Manifold Learning!

- Manifolds are topological spaces that are located on a high dimensional, usually Euclidean, space
  - Example of some manifolds are shown to the right
- Manifold learning is a machine learning algorithm used to dimensionally reduced a dataset that cannot be separated linearly
- The various techniques try to preserve locality between points to conserve the nature of the graph



# Key Manifold Algorithms




1. t-SNE
2. Isomap
3. Laplacian Eigenmap
4. Diffusion maps

# T-distributed Stochastic Neighbor Embedding (t-SNE)

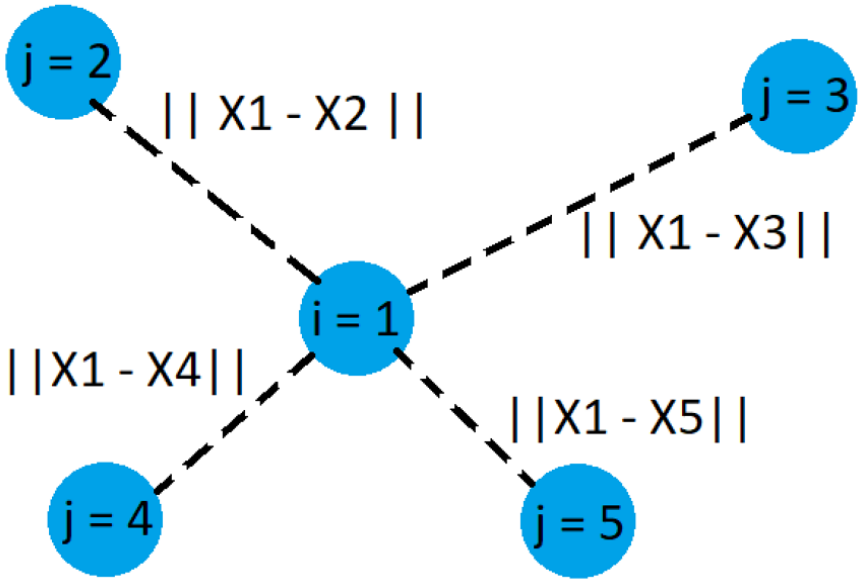
- Developed by Maaten and Hinton in 2008
- Easier to visualize and gain insights into the underlying structure and relationships within the data.
- Effective at capturing non-linear relationships within the data.
- Widely used in a variety of fields, including bioinformatics, computer vision, and natural language processing
- Based off Stochastic Neighbor Embedding

Maaten

Hinton

# SNE Algorithm

Step 1: Compute the pairwise distances (ex. Euclidean) between each data point



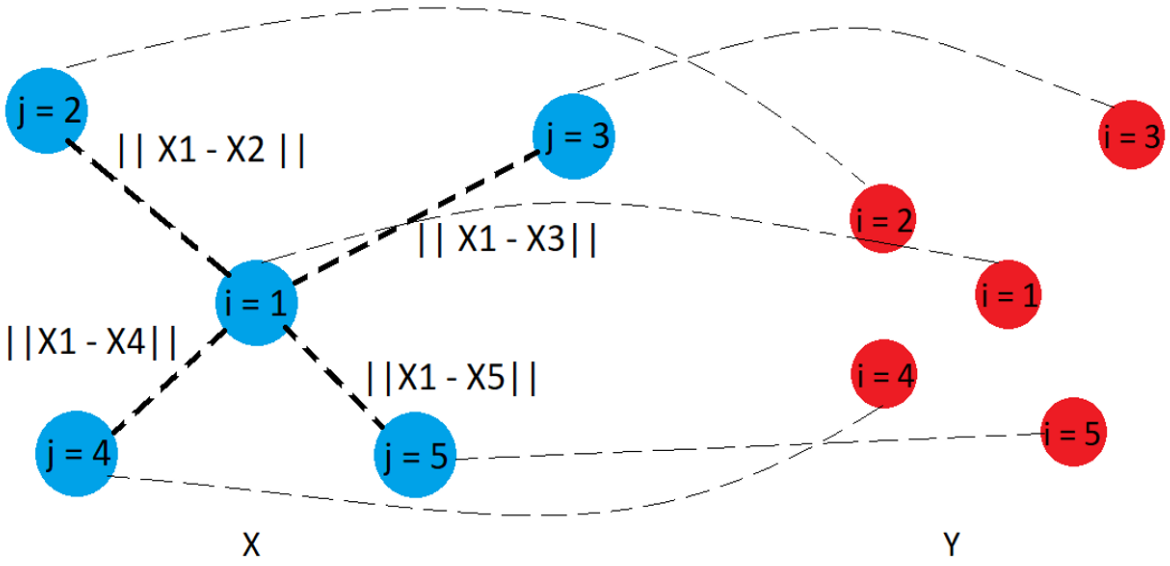

Step 2: Construct a conditional probability distribution centered at each Xi.

Define P(j|i) ~ probability that Xj will choose Xi as its neighbor given distances.

$$P_{j|i|} = \frac{\exp\left\{-\frac{||x_i - x_j||^2}{2\sigma_i^2}\right\}}{\sum_{k \neq i} \exp\left\{-\frac{||x_i - x_k||^2}{2\sigma_i^2}\right\}}$$

# SNE Algorithm

Step 3: Define variables [y1,…,yN], which are the low dimensional representations of [x1,…,xN].




Step 4: Create a conditional probability distribution for each Yi.

Q(j|i) ~ probability that Yj will choose Yi as its neighbor

$$q_{j|i|} = \frac{\exp\left\{-||y_i - y_j||^2\right\}}{\sum_{k \neq i} \exp\left\{-||y_i - y_k||^2\right\}}$$

# SNE Algorithm

Step 5: Goal → Find the optimal yi that minimize q(j|i) and p(j|i) for each data point.

We use the Kullback-Leibler Divergence for the minimization.

Cost function:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Step 6: Gradient Descent to find optimal yi.

Iterative approach until gradient of cost function reaches 0.

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

# SNE + few modifications = t-SNE

1. Use a joint probability distribution, P and Q, instead of individual conditional probability distributions
2. The cost function is then updated as such:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

3. Use the student t distribution instead of gaussian for similarity probabilities

Why?

- To better preserve local structure.
- Heavier tails → Map larger distances to lower probabilities

# Toolboxes & Optimization Methods

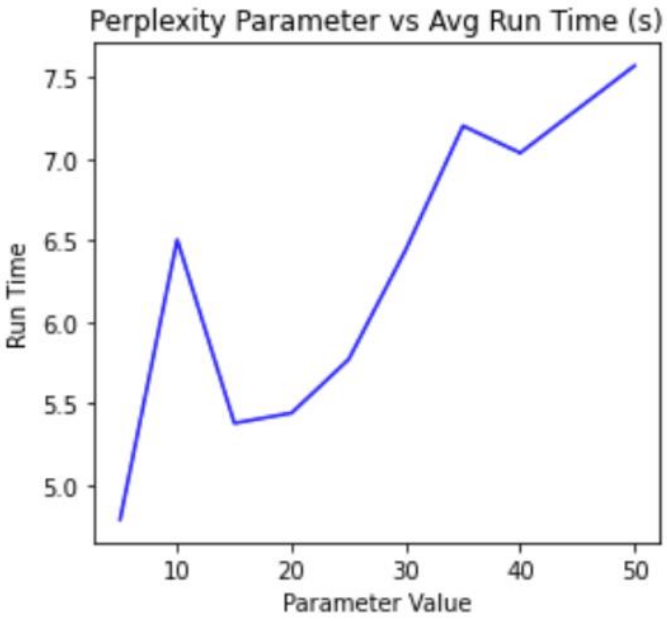
- No analytical solution for t-SNE →iterative optimization approach → larger runtime →need optimal parameter search
- Time complexity: *$O(n^2)$ where n is the number of data points*

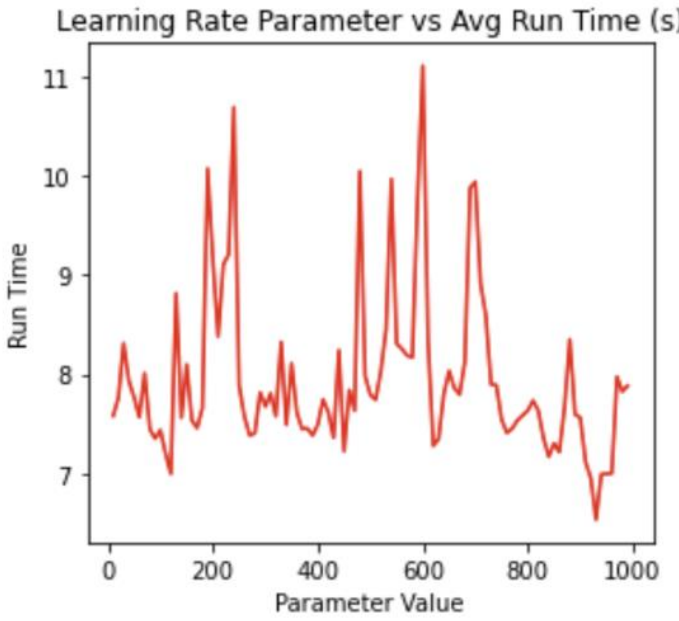
| Sklearn Implementation Parameters | |
|---|---|
| n_components | the number of dimensions in the low-dimensional space. |
| perplexity | a measure of the balance between preserving local and global structure in the embedding; usually between 5 and 50. |
| learning_rate | the step size at each iteration of the gradient descent optimization. |
| n_iter | the number of iterations of the gradient descent optimization. |
| early_exaggeration | a parameter that controls the degree of exaggeration of attractive forces between nearby points in the early stages of the optimization; usually between 2 and 10. |
| metric | the distance metric used to calculate pairwise distances between data points in the high-dimensional space. The default metric is Euclidean distance. |
| init | the initialization strategy for the low-dimensional embedding. The default strategy is to use a random initialization. |

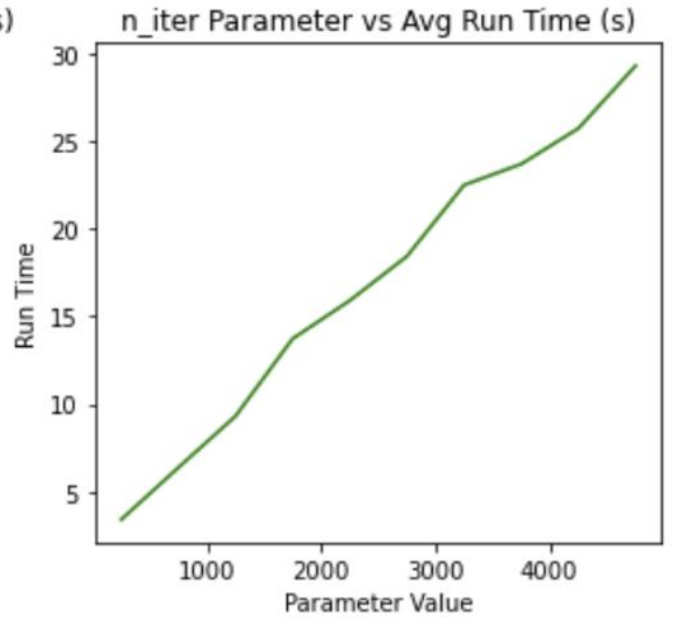
# Parameter Search Example

Applied t-SNE on MNIST Dataset (subset of digits [1,2,3,4])

Runtime vs parameter values



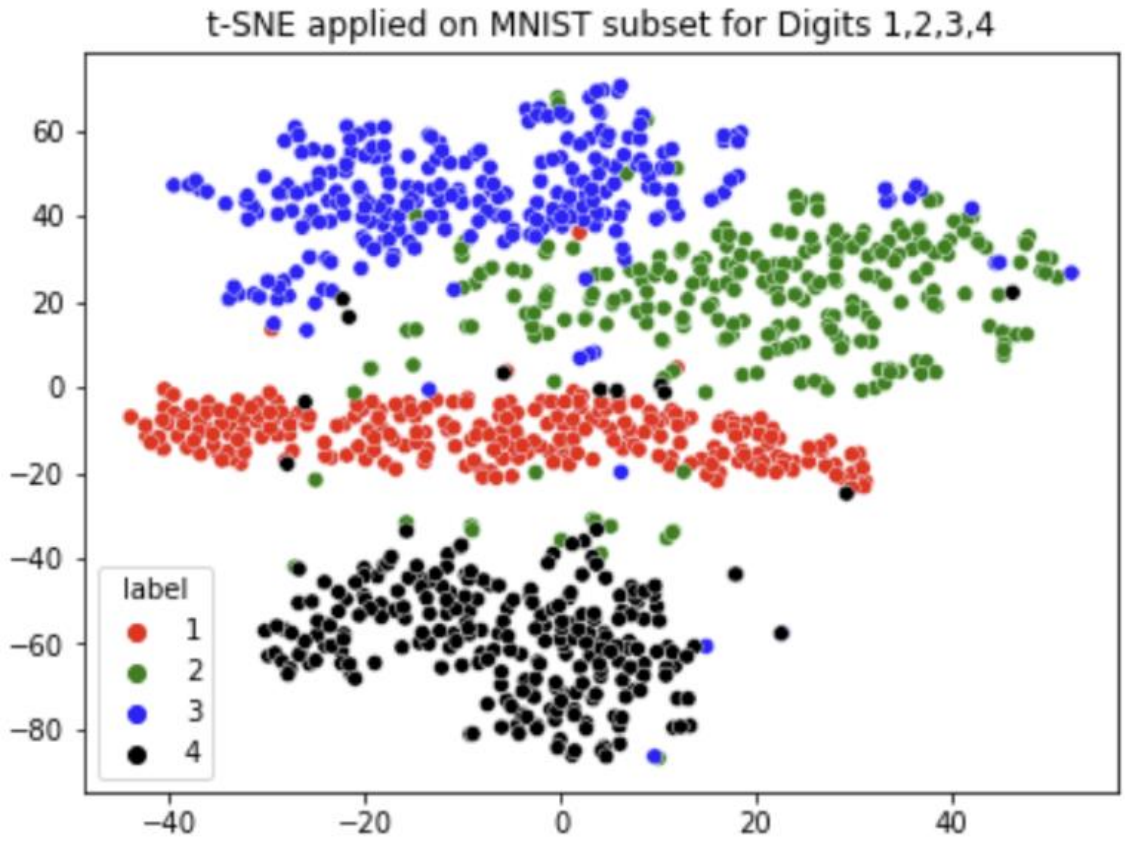



# Parameter Search Example

Applied t-SNE on MNIST Dataset (subset of digits [1,2,3,4])

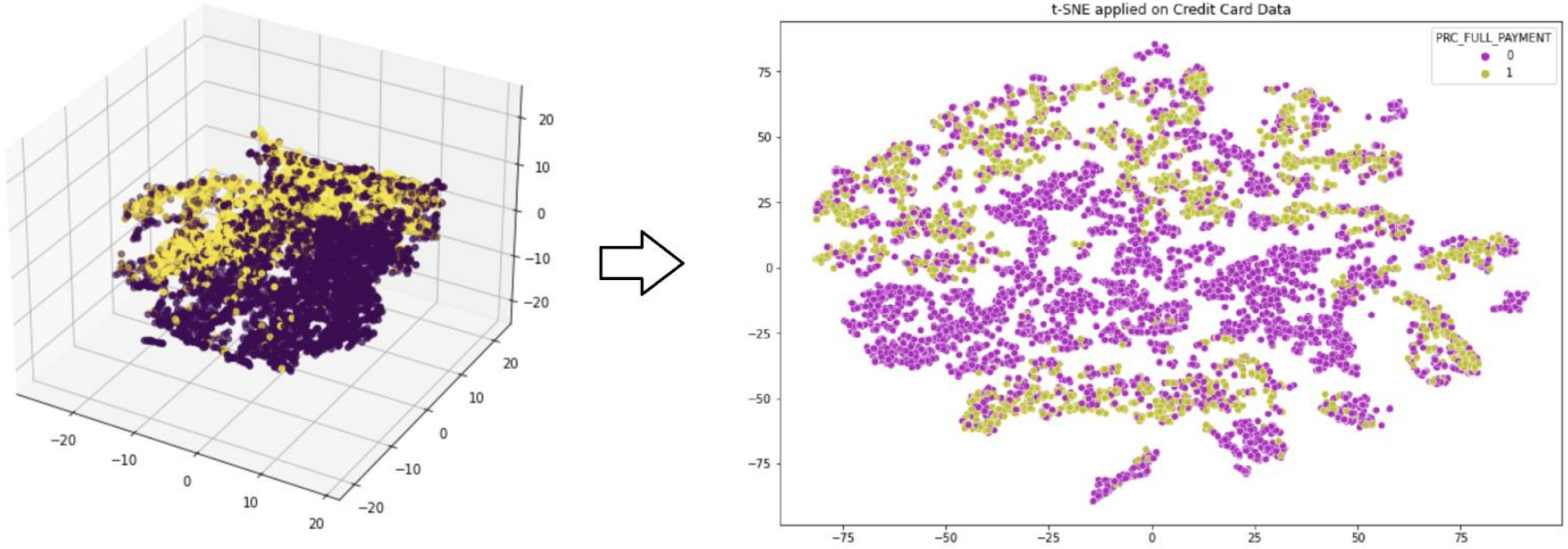
Result of Dimension Reduction:

Original Data: 400 attributes (20 x 20 pixel images)



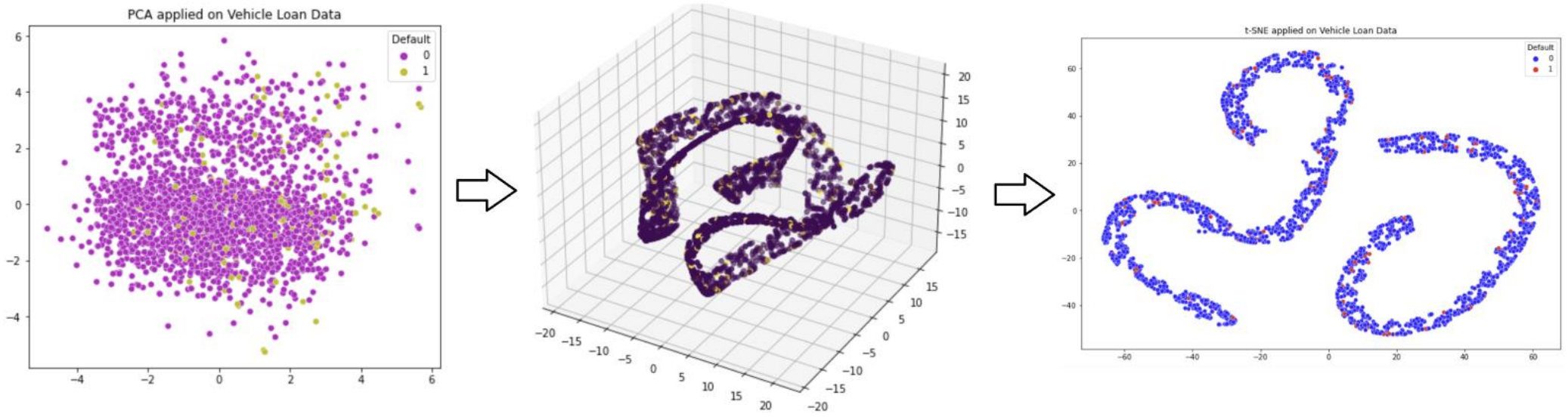
# Application I: Credit Card Data

- This credit card dataset summarizes the usage behavior of approximately 9000 active credit card holders from the last 6 months.
- There are 18 credit card attributes in the dataset, including Balance, Balance Frequency, Cash Advance, and Credit Limit.
- T-SNE is applied on this dataset to reduce the 18-dimension space to 3-dimensions and 2- dimensions.
- Optimal hyperparameters are determined using a standard grid search.
- The 3-dimensional and 2-dimensional transformed spaces are shown below:



# Application II: Vehicle Loan Data

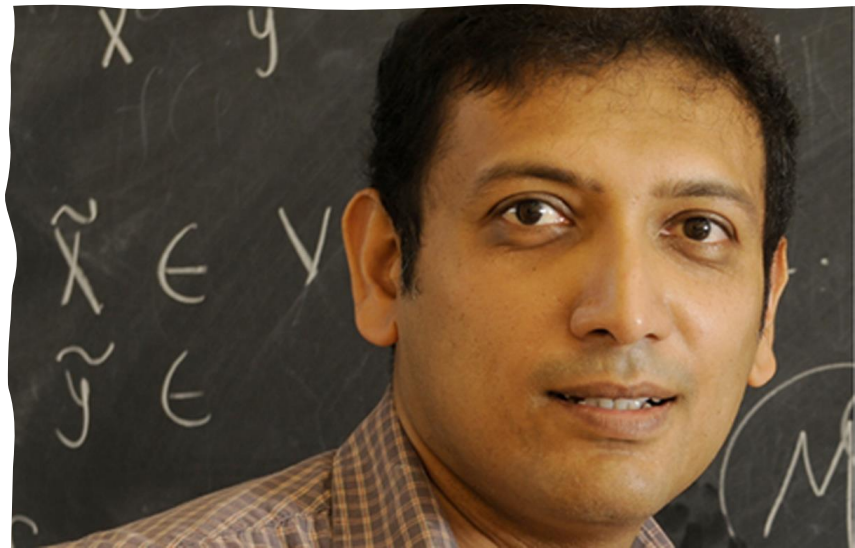
- An NBPC is currently facing profitability issues due to an increase in defaults on vehicle loans.
- Thirty attributes are contained in this dataset and are used to cluster customers by those who will or will not default on their loans.
- The dataset contains 121,800 records from customers in a single year.




- The clusters don't group by default variable
- But it indicates a presence of an underlying output variable given two notable clusters are formed!

# Laplacian Eigenmaps (LE)

- Laplacian Eigenmap's usage in dimensionality reduction was first pioneered by Mikhail Belkin and Partha Niyogi (top and bottom)
- They used the already understood tool of a Laplacian to describe a lower dimensional manifold existing on a higher dimension
- The use of the Laplacian and heat kernel allowed for locality to be conserved
- They made clear connections to clustering techniques, primarily spectral clustering whose algorithm is very similar to LE



# LE Algorithm (Dataset --> Graph)

- Define the data points to be vertices
- Define which vertices have an edge between them
  - n nearest neighbors and ε-neighborhood are used to determine if vertices have edges
- Define the weights of the edges
  - Heat Kernel and Simple Weights are used to determine weights of the edge
  - Equation for heat kernel: $e^{\frac{\left\|x_i - x_j\right\|_2}{t}}$
  - Simple weights define an edge to have a weight of 1 and no edges to have a weight of zero

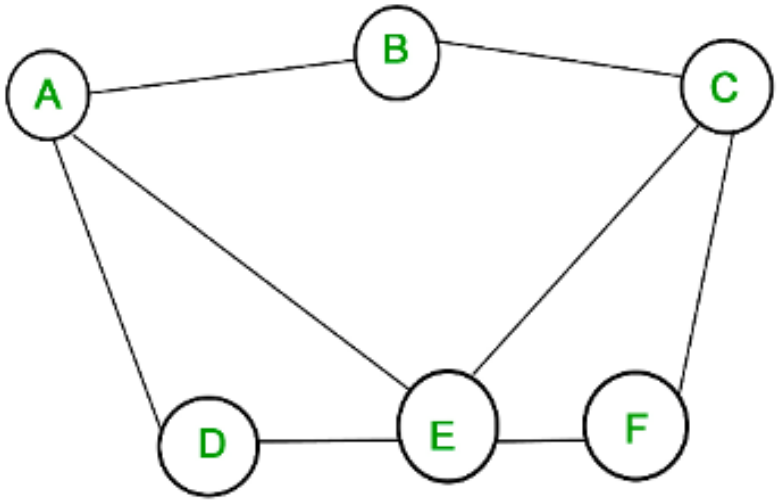
# LE Algorithm (Graph --> Laplacian)

- Define the Adjacency matrix
  - A square matrix whose elements describe the weight of the edge between vertices
- Define the Degree Matrix
  - A square diagonal matrix whose diagonal entries are the summation of the corresponding row of the adjacency matrix
- Solve for the Laplacian Matrix
  - L = D - W

# Example

- Assuming simple weights so that edges have a weight of 1
- Solving for the Adjacency Matrix:

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




- Solving the Degree Matrix:

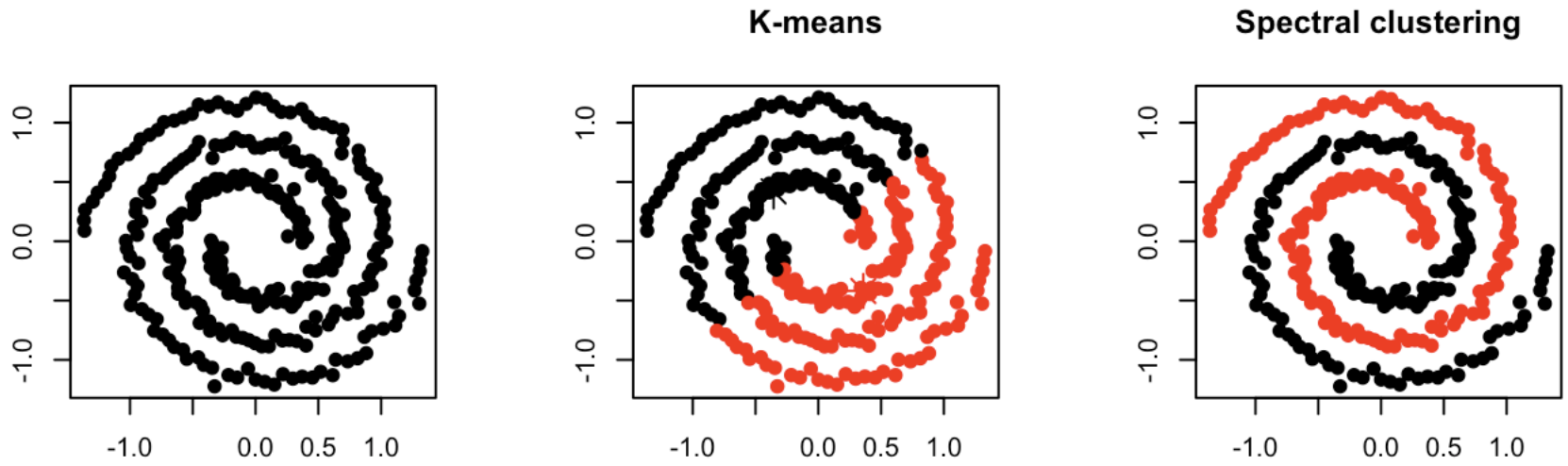
$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

# LE Algorithm (Laplacian --> Reduced Dimensions)

- Solve for the eigenvectors and eigenvalues of the Laplacian
  - Solving: $D^{-1}Lf = \lambda f$ , where $f$ is the eigenvector and $\lambda$ is the eigenvalue
  - Eigenvalues must be ordered from least to greatest
  - Corresponding eigenvectors must be listed in the same order
- Obtain the set of k eigenvectors corresponding to the first k, nonzero eigenvalues

# LE Algorithm (Connection to Spectral Decomposition)

- On the reduced basis do a simple clustering algorithm, like K-Means, to get a list of the labels of the group
- Graph the original dataset using the list of labels to differentiate points to a cluster
- Example of Spectral Clustering vs K-means:



# SKLearn Toolbox for LE (Spectral Embedding)

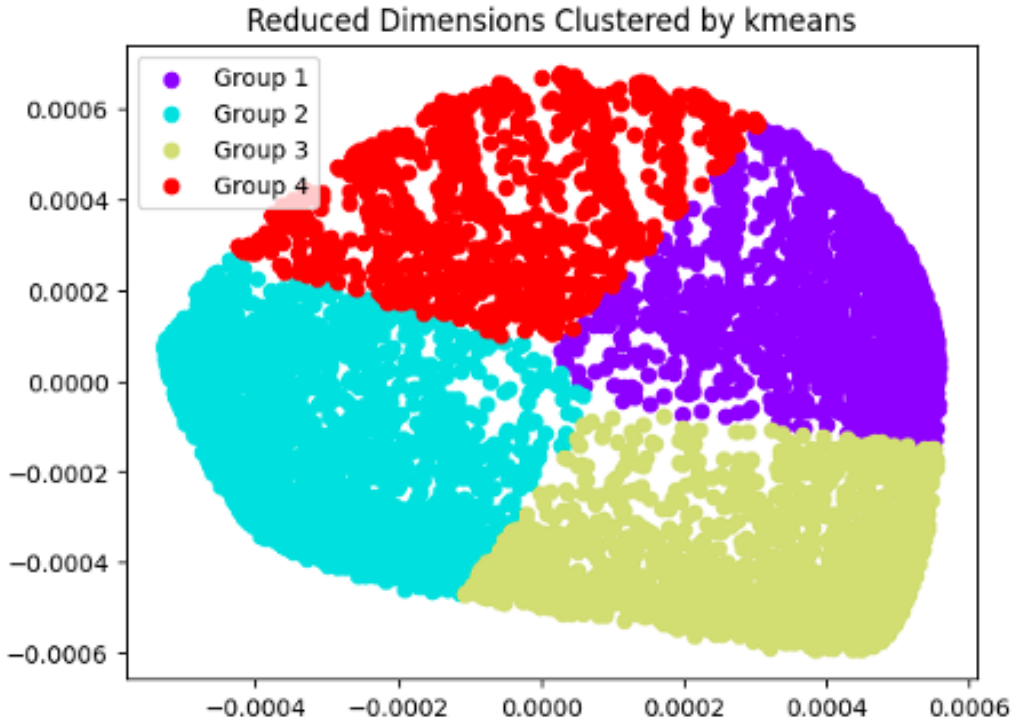
The complexity of this algorithm composes (where n is the number of data points):

- Finding the pairwise distances between all the points: $O(n^2)$
- Solving for the eigenvectors and eigenvalues of the Laplacian: $O(n^3)$
- Completing K-means clustering algorithm: $O(n^2)$

| Sklearn Implementation Parameters | |
|---|---|
| n_neighbors | This specifies the number of neighbors that will be used to each data point to construct the neighborhood structure. This is then used to determine geodesic distances between data points. The default is n_neighbors = 5 |
| affinity | Describes technique used to obtain neighbors. The default is nearest_neighbors |
| gamma | determines value of t in the heat kernel, where the default value is 1/len(data_set) |
| n_components | The number of dimensions in the reduced space. The default is n_components = 2 |
| eigen_solver | This parameter determines the method to solve eigenvalue of the Laplacian |
| tol | Specifies the tolerance of the convergence of the eigenvalue solver. The default is tol = 0. |

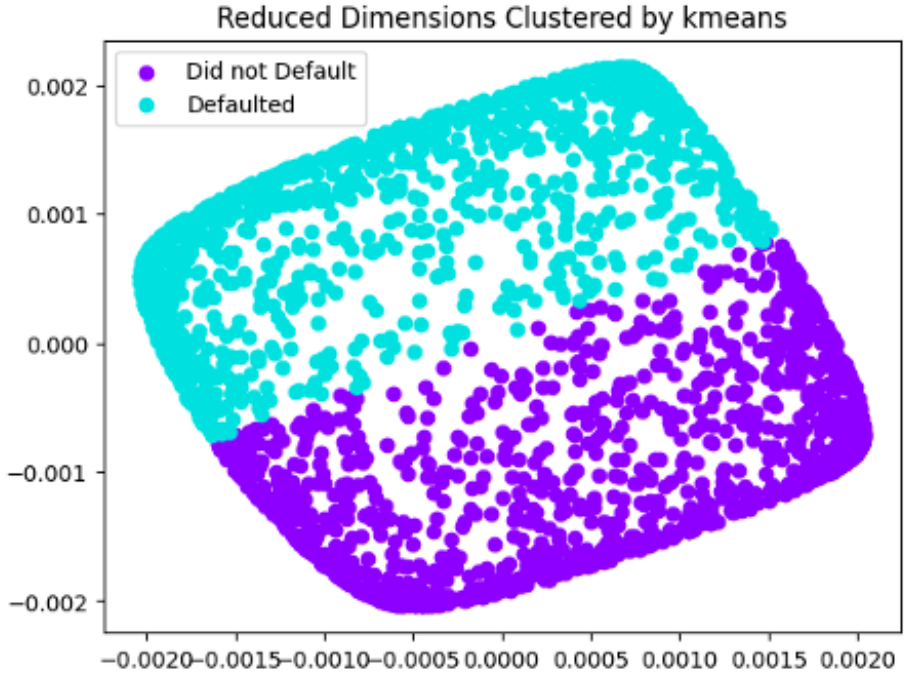
# Application of LE: Credit Card Data

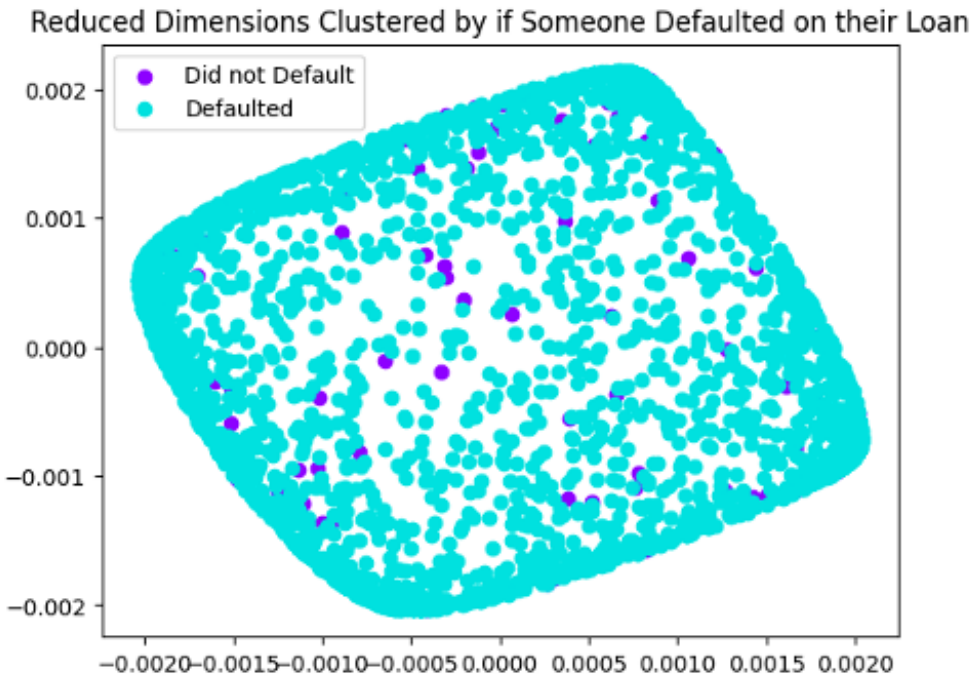
- The data set below is a credit card dataset summarizing the usage behavior of approximately 9000 active credit card holders from the last 6 months
- There are 18 credit card attributes in the dataset, including Balance, Balance Frequency, Cash Advance, and Credit Limit
- Laplacian Eigenmaps was used to reduce the dataset to two dimensions and then k-means clustering was used to divide the data up into 4 different groups



# Application of LE: Credit Card Data

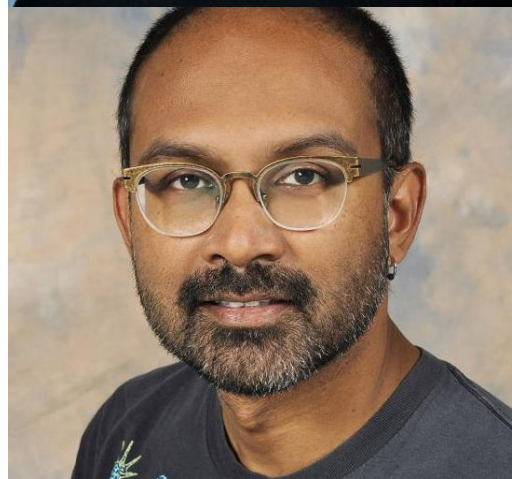
- The dataset below describes 121,800 different people's data about their loan, with 30 different attributes, and if they would default on their payment or not
- The dataset is focused on having a pre-existing cluster of those who will default and those who won't
- As shown, the data did not cluster properly showing that with this current information it is not possible to predict if someone will default on their loan



# Isometric Feature Mapping (IsoMap)

- Developed by Tenenbaum, Langford, and Silva in 2000
- Unsupervised dimensionality reduction method that maintains geodesic distances and non-linear relationships between datapoints.
- Characterized as an extension to multi-dimensional scaling or kernel PCA.
- Data is represented as a network or a graph with nodes representing original data and distances representing the similarities.
- IsoMap is used in several contexts including image processing, bioinformatics, and social network analysis.

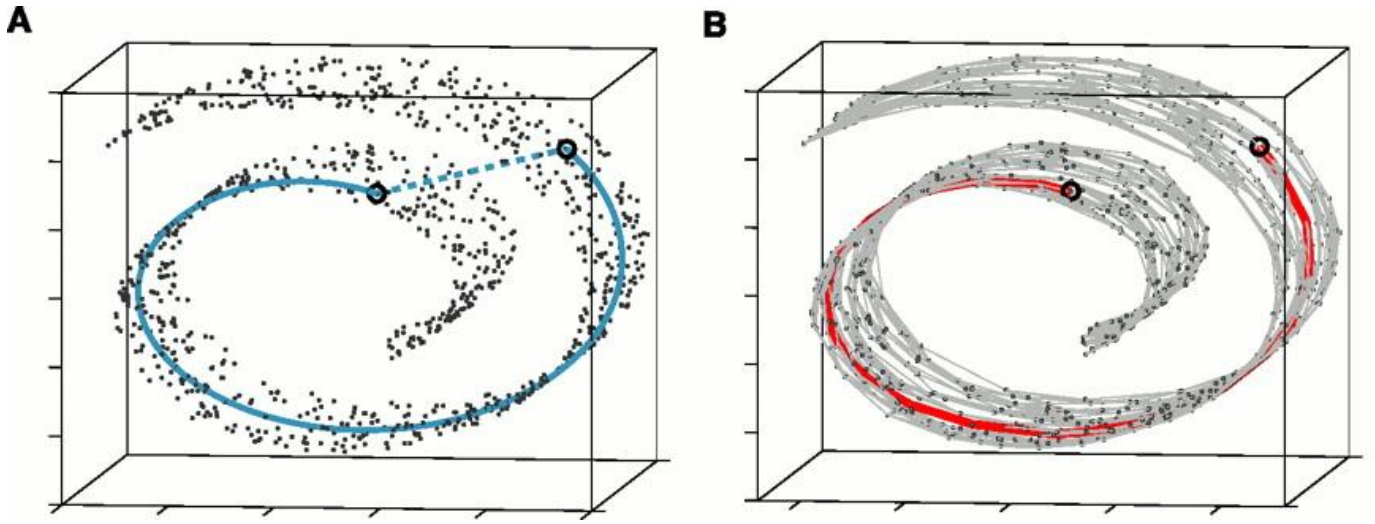


# IsoMap Algorithm

Stage 1: Nearest Neighbor Search
Find neighbors on the manifold (*M*) in the dataset of selected points within some fixed radius. Present as a weighted graph with data as nodes relation as edges.

Stage 2: Shortest Path Graph Search

Estimate geodesic distances between pairs and compute shortest distances.



# IsoMap Algorithm

Stage 3: *d*-Dimensional Embedding

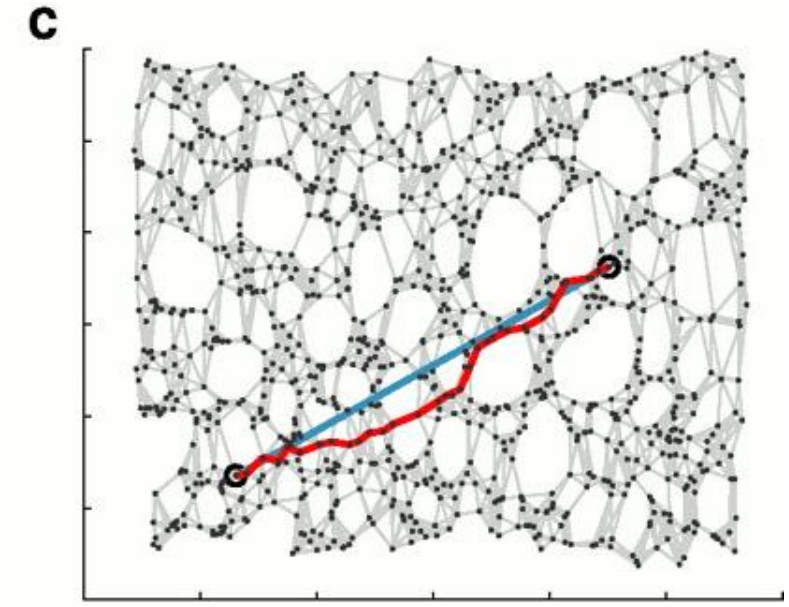
Applies traditional multidimensional scaling (MDS) to a matrix of distances in which the *d*-dimensional Euclidian space is created.

The embedding maintains preserves the shortest path distance.

Matrix Embedding: $\psi(x) = \left(\sqrt{\lambda_1}\phi_i(x), \ \ldots \ , \ \sqrt{\lambda_k}\phi_k(x)\right) \in \mathbb{R}^k$

from Euclidian distances:

$\{d_Y(i, \ j) = ||y_i - y_j||\}$



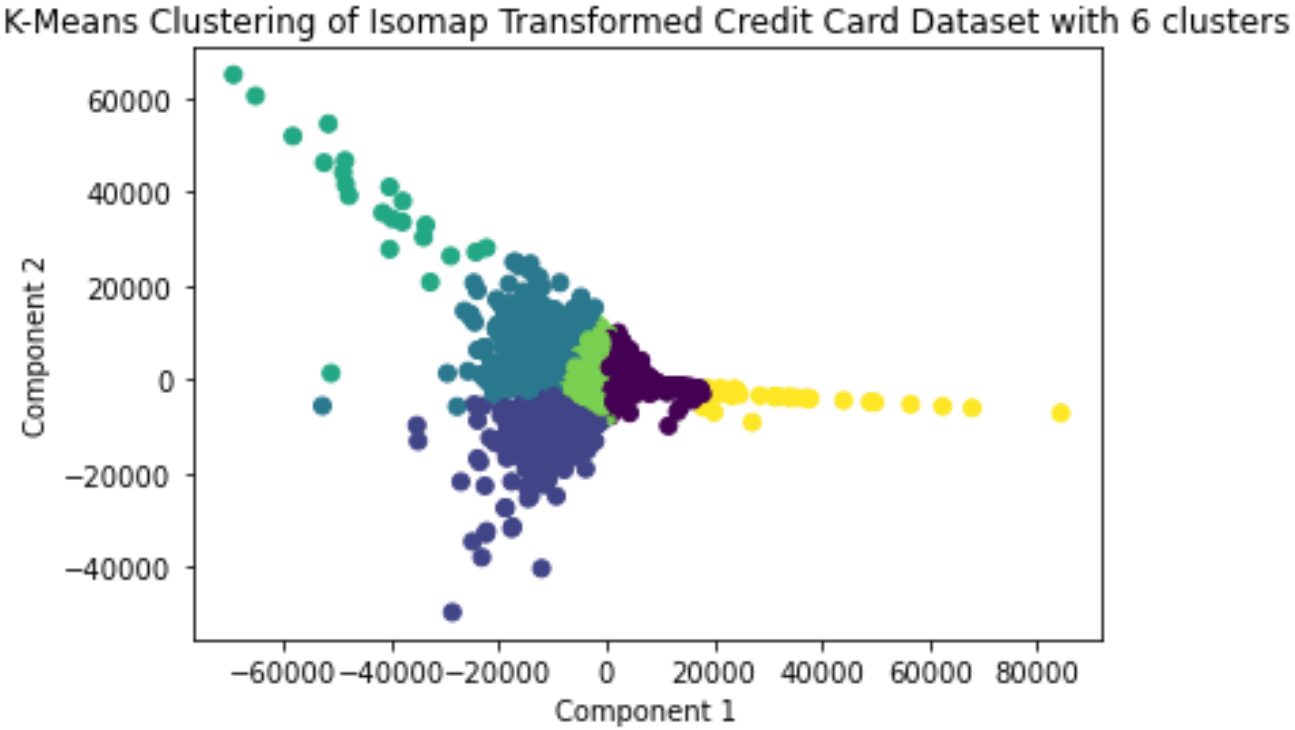
# Toolboxes & Optimization Methods

- IsoMap also uses an optimization approach → larger runtime →need optimal parameter search
- Overall Isomap Compelxity: $O[D\log(k)N\log(N)] + O\left[N^2(k + \log(N))\right] + O\left[dN^2\right]$

*where N – number of training data points; D – input dimension; k – neighbors; d – output dimension*

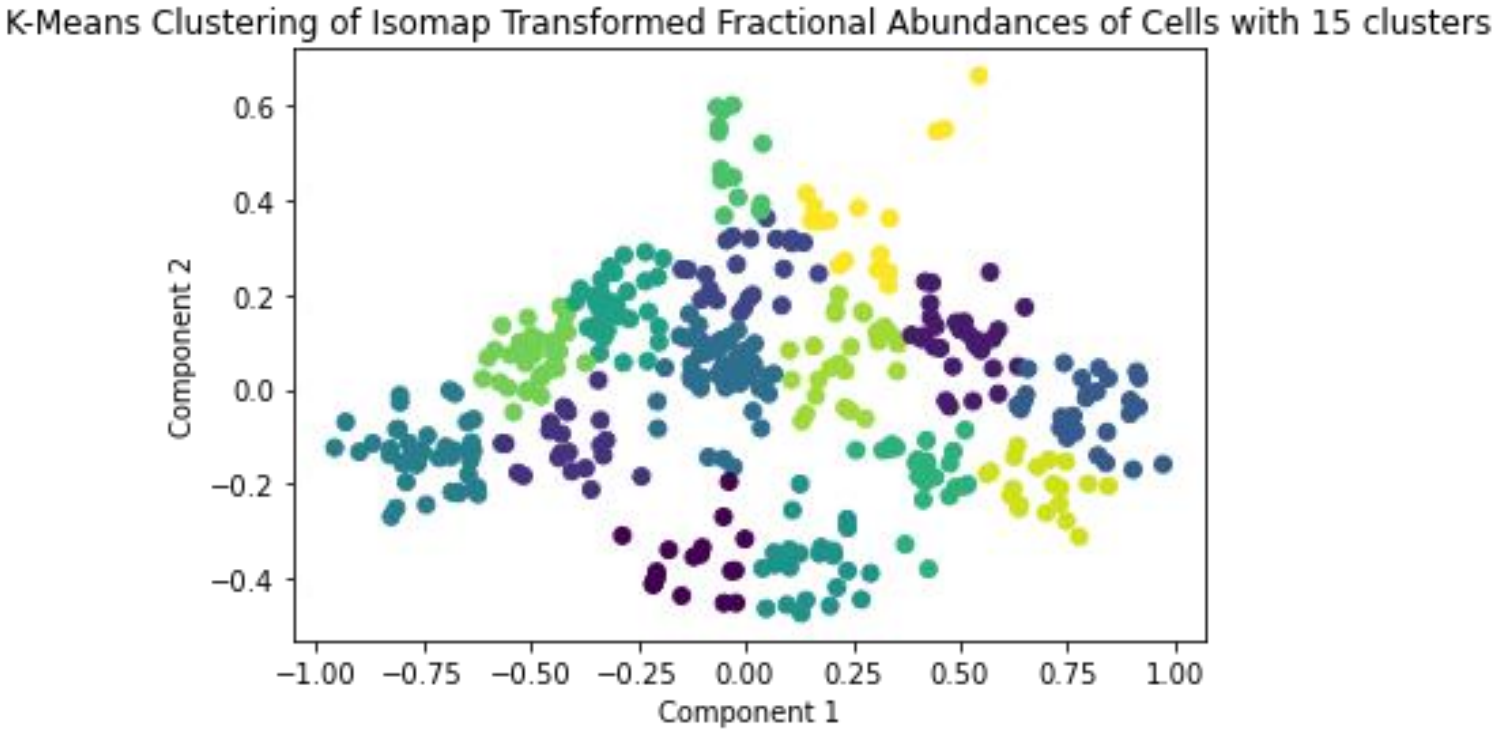
| Sklearn Implementation Parameters | |
|---|---|
| n_neighbors | number of neighbors used to construct the neighborhood strucutre. |
| n_components | number of dimensions in reduced space |
| eigen_solver | method used to solve eigenvalue problem for geodesic distances |
| tol | tolerance of the convergence of the eigenvalue solver |
| max_iter | maximum number of iterations for eigenvalue solver |
| path_method | parameter specifies method used to calculate geodesic distances with options “auto”, “FW” (Floyd-Warshall algorithm), and “D” (Dijakstra’s Algorithm). |
| neighbors_algorithm | specifies algorithm chosen to compute nearest neighbors of each data point with options “auto”, “brute”, “kd_tree”, and “ball tree”. |

# Application I: Credit Card Data



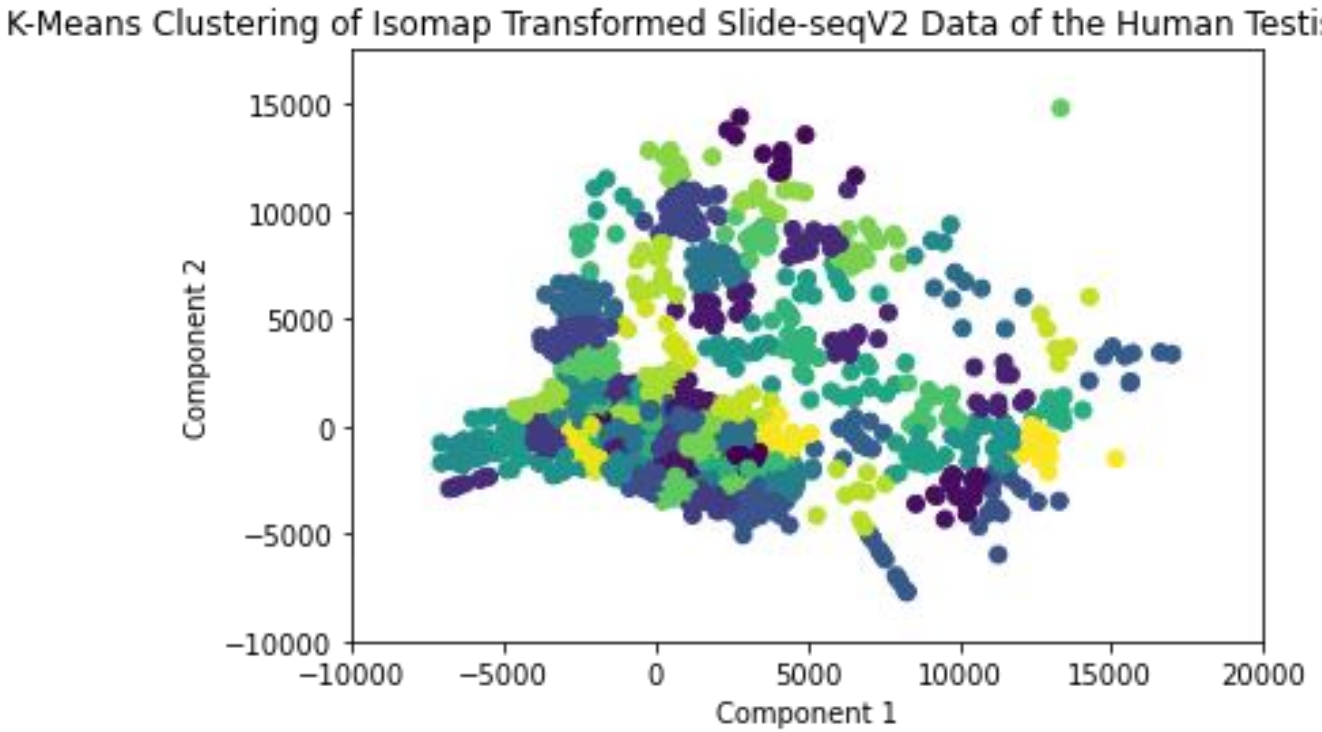

- Dataset summarizes usage of 9000 active credit card users from the past 6 months
- IsoMap reduced this 19-dimensional space to a 2-dimensional space
- K-means clustering optimization (*Elbow Method*) determined an optimal 6 clusters.

# Application II: Fractional Abundances of Cells Classified within 2.5 Hours of Movement




- Dataset presents fractional abundance of various cell types (ovarian cancer cells, germinal center b cells, human dermal fibroblasts) within 2.5 hours of *x-y* motility tracking under different conditons
- IsoMap reduced this data to a 2-dimensional dataset.
- K-means clustering optimization (*Elbow Method*) determined an optimal 15 clusters.

# Application III: Slide-seqV2 Data of Human Testis




- Dataset summarizes the testing of human testis cells.
- Slide-seq specifically analyzes and harvests data regarding the spatial organization of different cells.
- IsoMap reduced the dataset including genome classifications for over 21,000 human genomes and ~6000 targeted genes to a 2-dimensional space
- K-means clustering optimization (*Elbow Method*) determined an optimal 85 clusters.

# Diffusion Maps

- Introduced by Ronald Coifman and Stephane Lafon in 2005
- A strategy for finding a lower dimension embedding of high-dimensional data
- As the diffusion process occurs, paths along the underlying geometry will have the highest contribution to the diffusion distance
- Calculating diffusion distances is computationally expensive, so instead the data can be mapped into Euclidean space according to the diffusion metric
  - The diffusion distance in this new diffusion space becomes the Euclidean distance
- This map is used to reduce dimensions, as afterwards fewer coordinates are needed to describe the same data points
  - Preserves intrinsic geometry as well

# How it Works

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

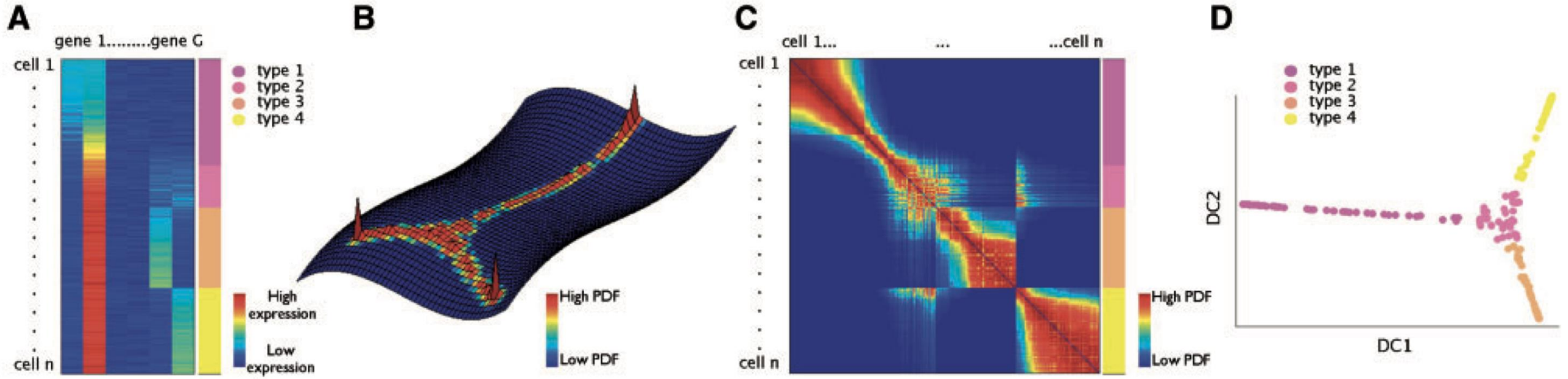
$$P^2 = \begin{bmatrix} p_{11}p_{11} + p_{12}p_{21} & p_{12}p_{22} + p_{11}p_{12} \\ p_{21}p_{12} + p_{22}p_{21} & p_{22}p_{22} + p_{21}p_{12} \end{bmatrix}$$

- Diffusion matrices have each entry $p_{ij}$ represent the probability of jumping between data point i to j
- Like the chance of jumping from one point to another on a random walk
- A diffusion metric is calculated as: $D_t(X_i, X_j)^2 = \sum_{u \epsilon X} |p_t(X_i, u) - p_t(X_j, u)|^2 = \sum_k \left|P^t_{ik} - P^t_{kj}\right|^2 .$

- Probabilities $P^t$ are calculated for increasing t as the diffusion process progresses
- As t increases, the probability of following a path along the geometry of the data set is more likely as that is where points should be denser and more connected

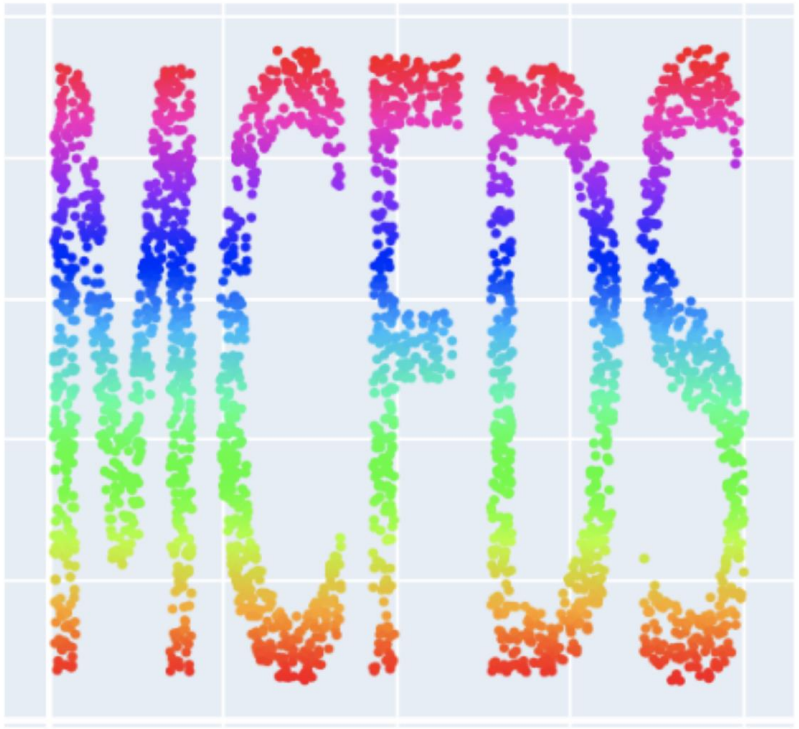
# Application I: Single Cell Genomics

- Sometimes cells might not be clustered distinctly but can be classified
- Analyzing random walks between cells near each other with a Gaussian kernel can reveal underlying structure
- The Markovian transition matrix for the cell holds the probabilities of transitioning to any other cell and is proportional to a Gaussian kernel
  - First eigenvectors of the Markov transition matrix are called the diffusion components
- Walks are all superimposed to create a diffusion map

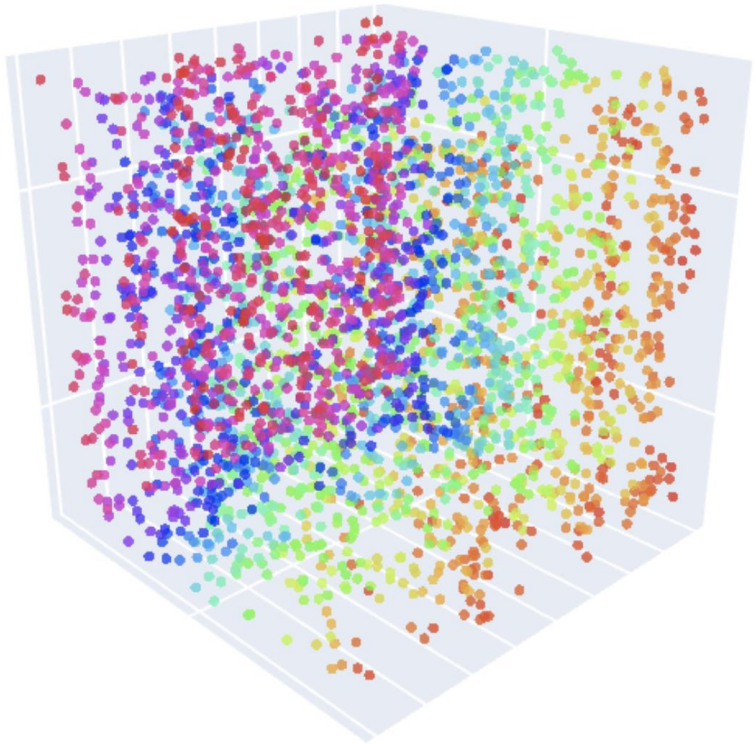


# Application II

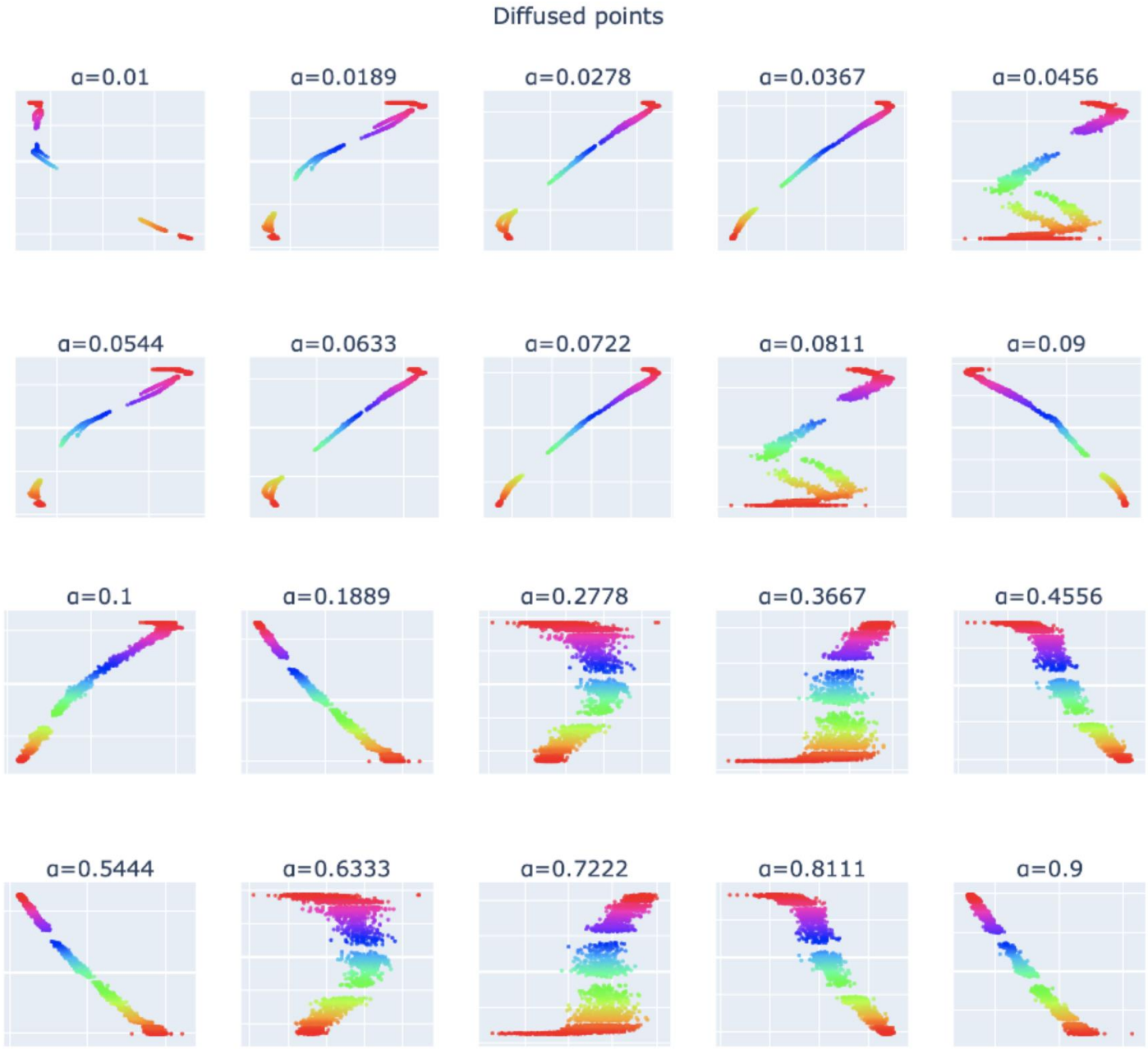
Original shape

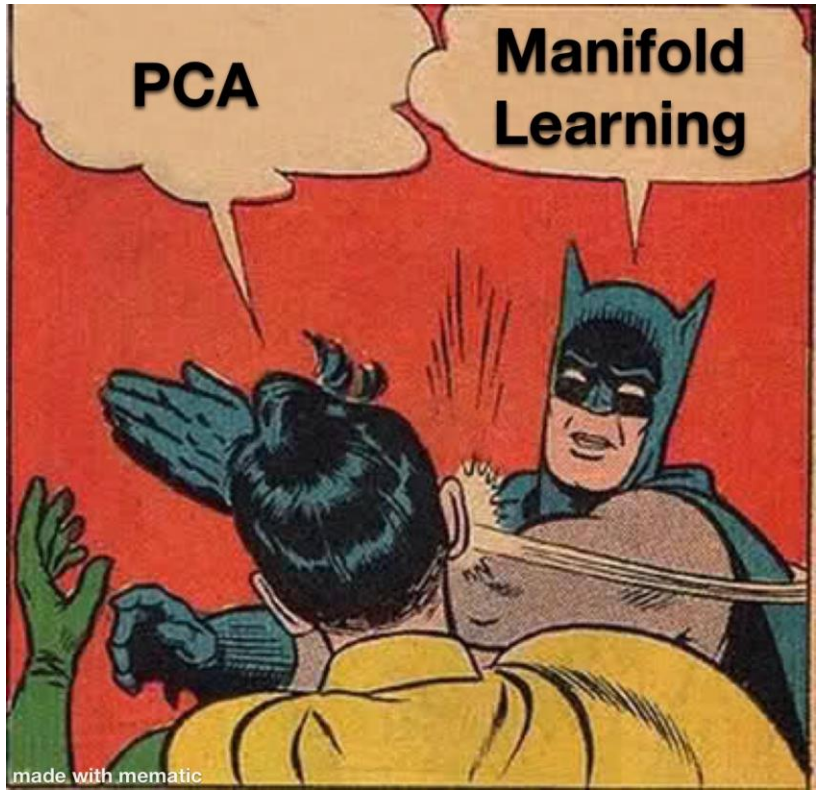


Synthetic 3D Datapoints

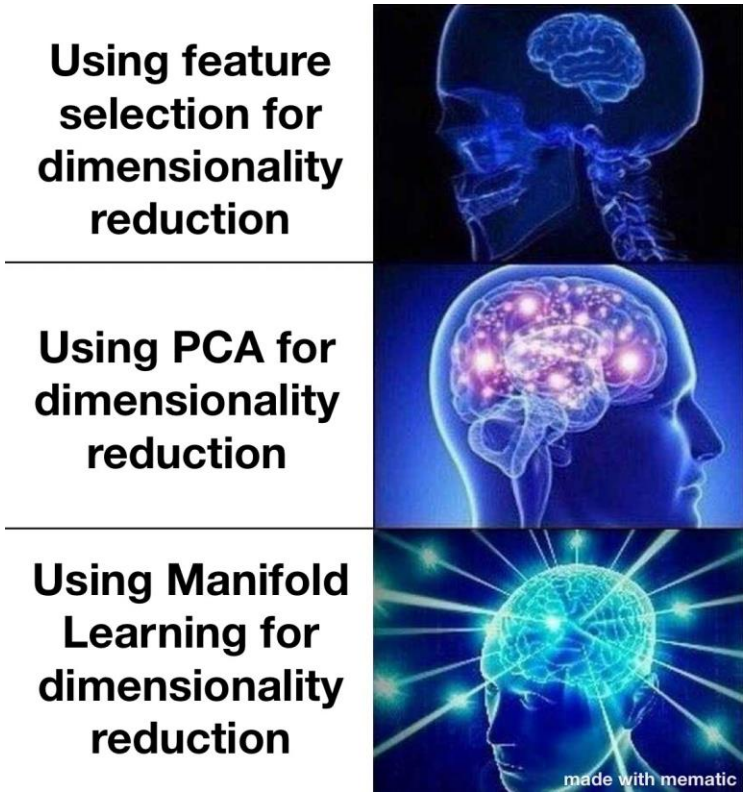


- Alpha is a parameter that scales the exponent used for left normalization when creating the diffusion map.
- Alpha values of 0.01-0.9 were tested as shown:




PCA
Manifold Learning
made with mematic

# Thank you!


Using feature selection for dimensionality reduction
Using PCA for dimensionality reduction
Using Manifold Learning for dimensionality reduction
made with mematic