

DELPHYS 2025

**DEep Learning for PHYSics
Workshop**



Unsupervised Learning

Poemwai Chainakun

Suranaree University of Technology

pchainakun@g.sut.ac.th

You are given a dataset of cats:

- **No teacher**
- **No instruction**
- **No answer key**

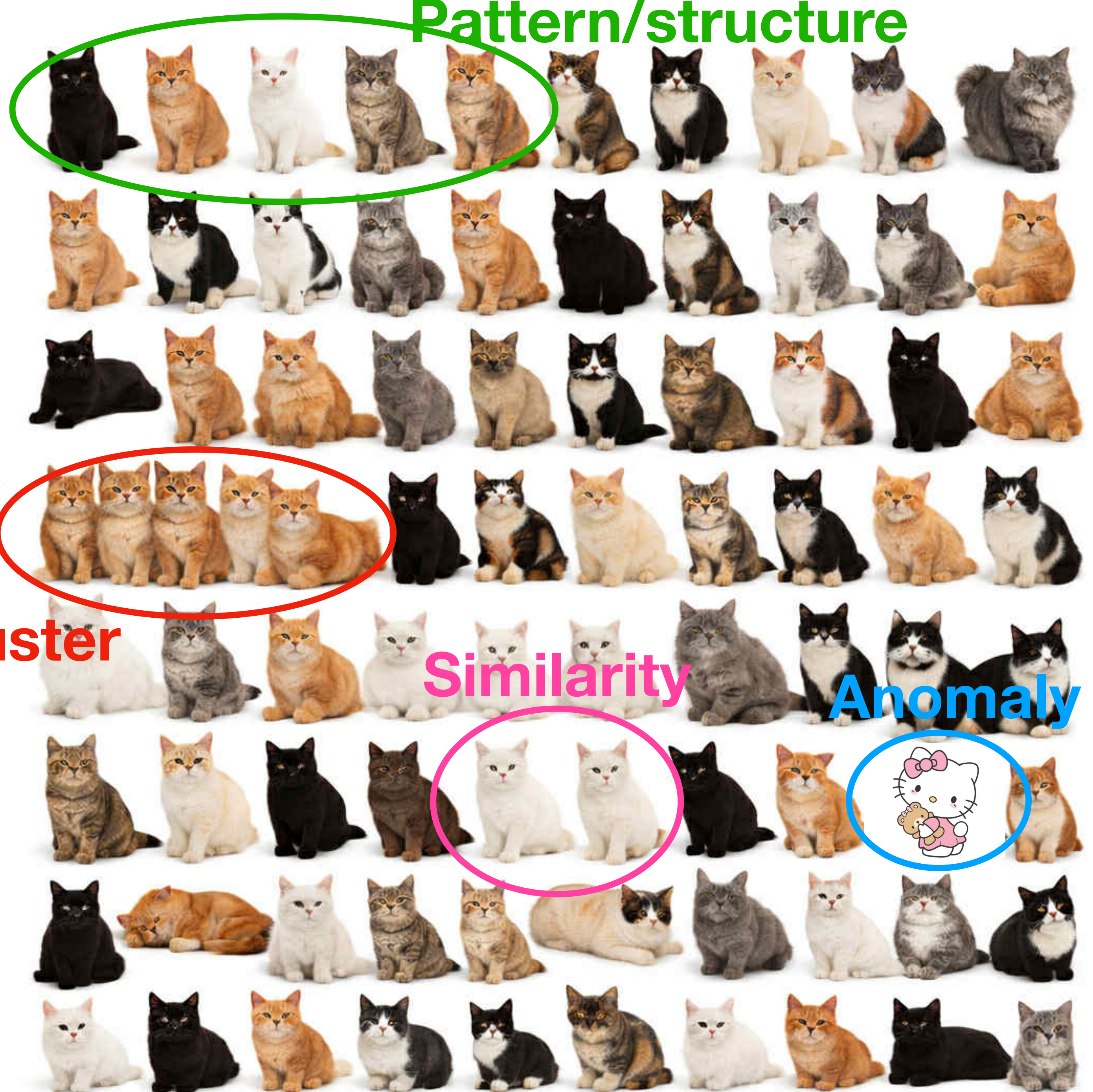
**What can you
still learn?**



You are given a dataset of cats:

- No teacher
- No instruction
- No answer key

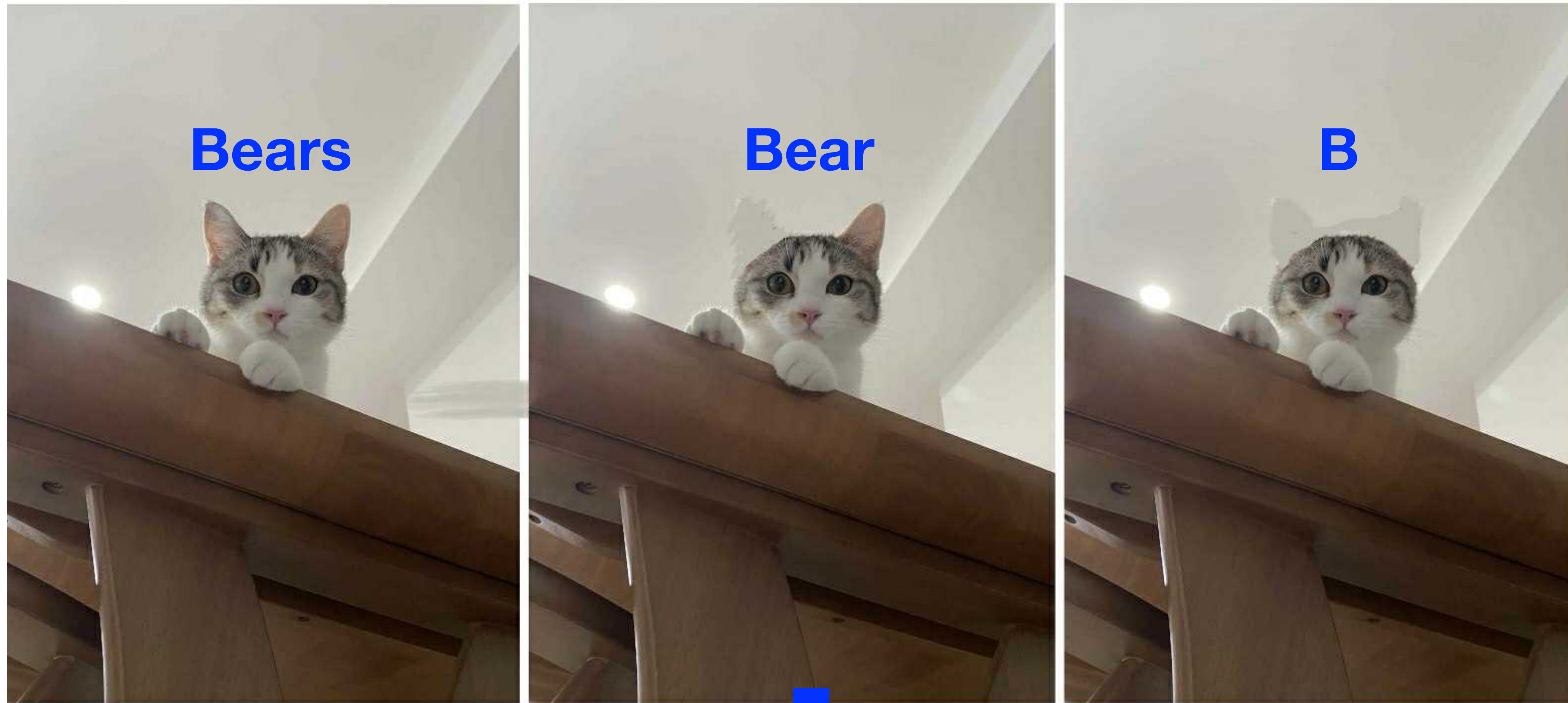
What can you
still learn?



**When you don't know what to look for,
you look for structure.**

**Most datasets live in many dimensions (features),
but we can reduce the size of the data by keeping only
the most informative directions**

**Most datasets live in many dimensions (features),
but we can reduce the size of the data by keeping only
the most informative directions**



Dimensionality reduction

Unsupervised Learning

- Typical tasks: Clustering, dimensionality reduction, and anomaly detection

Supervised



- With guidance
- Learning with **labeled data**

v.s.



Unsupervised



- Without guidance
- Learning from **unlabeled data**

Unsupervised Learning

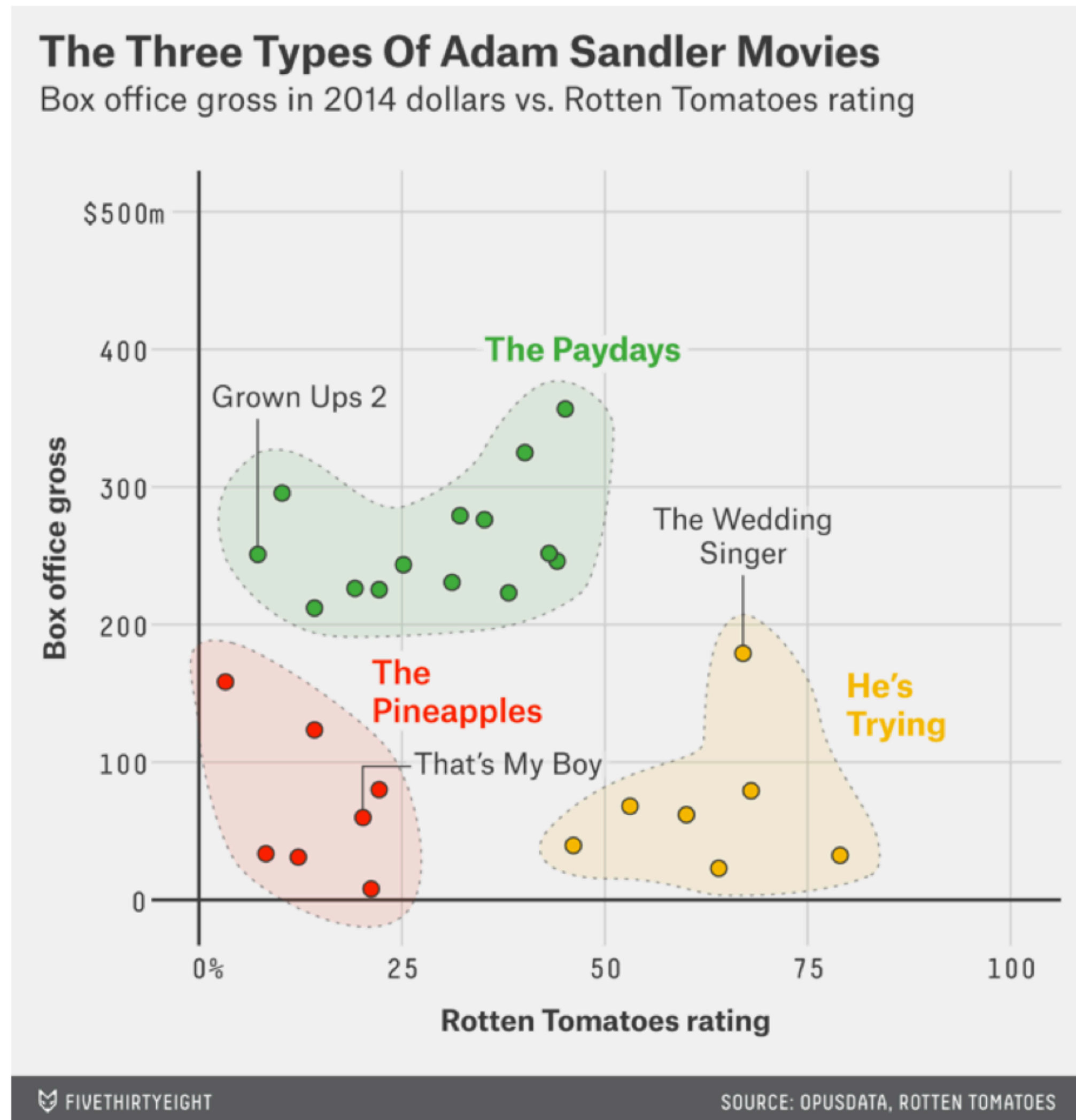
- Typical tasks: Clustering, dimensionality reduction, and anomaly detection
- Data are unlabeled
- Learning focuses only on features vectors X
- No output (response) variable y
- Goal: Discover interesting patterns/properties of the data (e.g. visualizing or interpreting high-dimensional data)

Supervised v.s. Unsupervised

	Supervised Learning	Unsupervised learning
Objective	To approximate a function that maps inputs to outputs based out example input-output pairs.	To build a concise representation of the data and generate imaginative content from it.
Accuracy	Highly accurate and reliable.	Less accurate and reliable.
Complexity	Simpler method.	Computationally complex.
Classes	Number of classes is <i>known</i> .	Number of classes is <i>unknown</i> .
Output	A desired output value (also called the supervisory signal).	No corresponding output values.

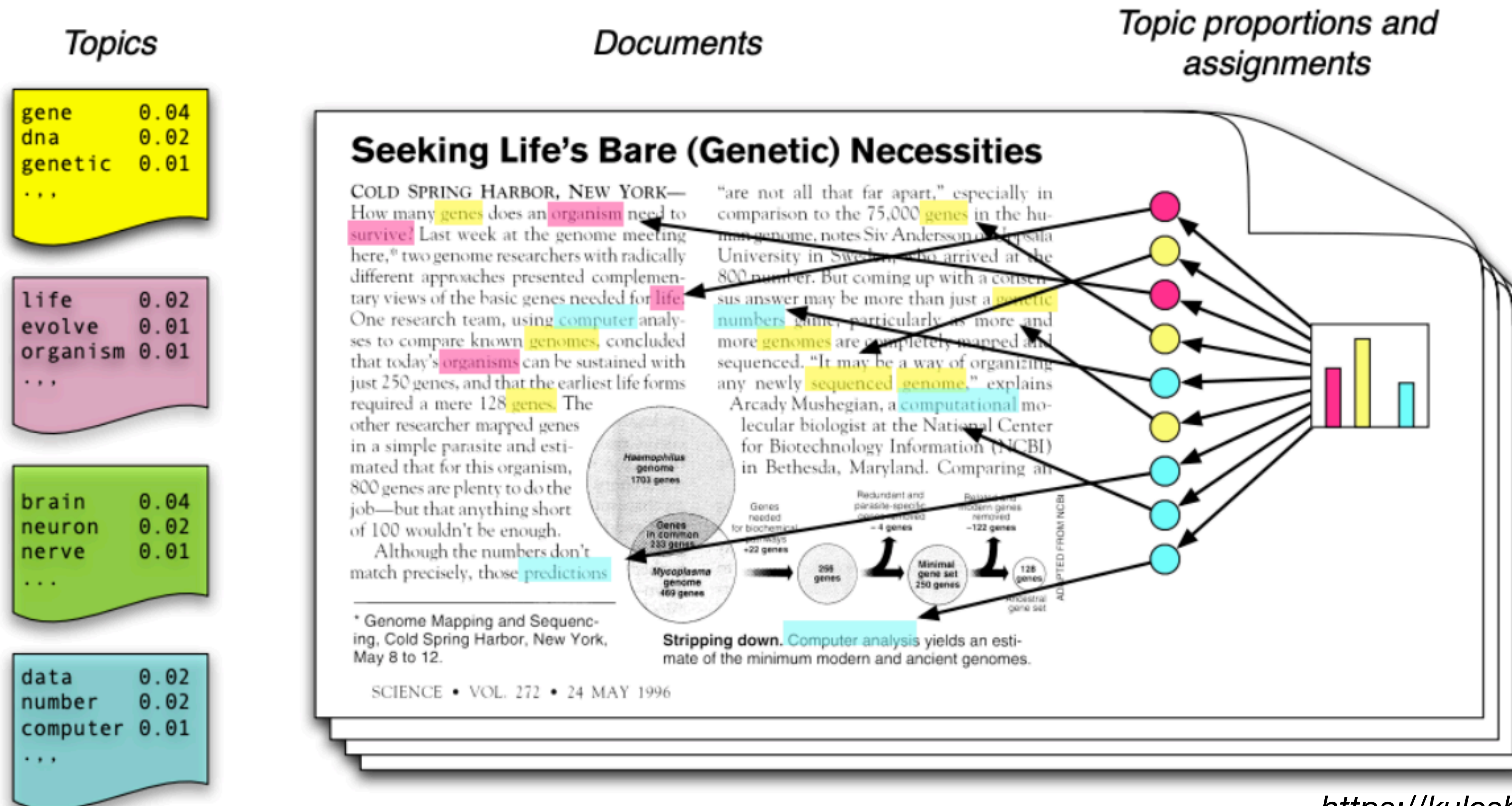
Applications of Unsupervised Learning

- Data clustering: identify sets of data about the same properties



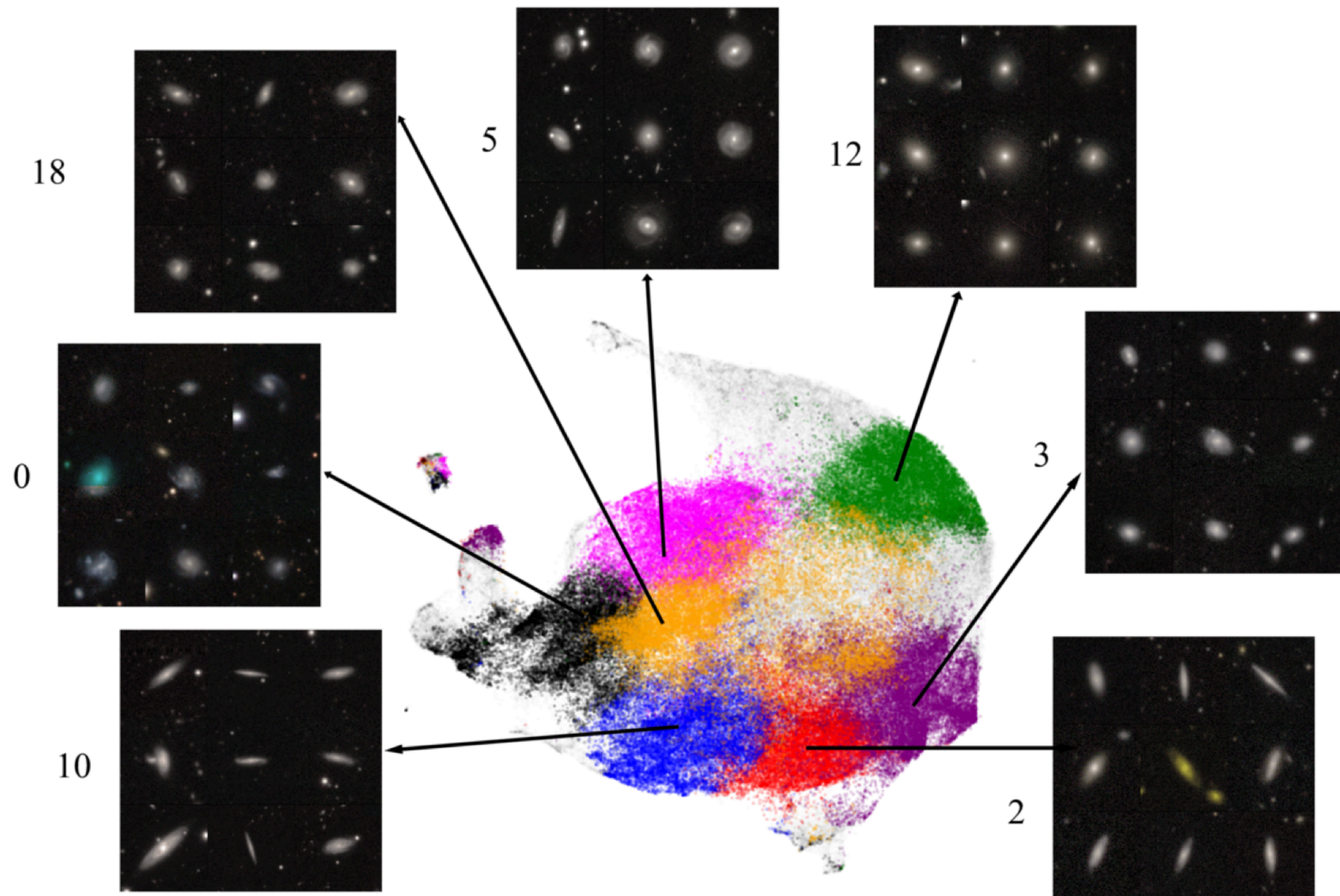
Applications of Unsupervised Learning

- Data clustering: clusters of related words and/or topics



Applications of Unsupervised Learning

- Data clustering: cluster samples of galaxies



UMAP plot of cluster samples
- *Mohale & Lochner 2023*

A Recipe for Unsupervised Learning



- We first collect a dataset and define a learning algorithm
- The result of running the algorithm on the data is an unsupervised model
- The outputs of this model are interesting properties of the data
 - > An unsupervised model does not make predictions but instead uncovers underlying structure in the data (e.g., identifying clusters of related data points)

Challenges of Unsupervised Learning

- Working with high-dimensional data
- Goal is not always clearly defined
- ‘Right answer’ unknown

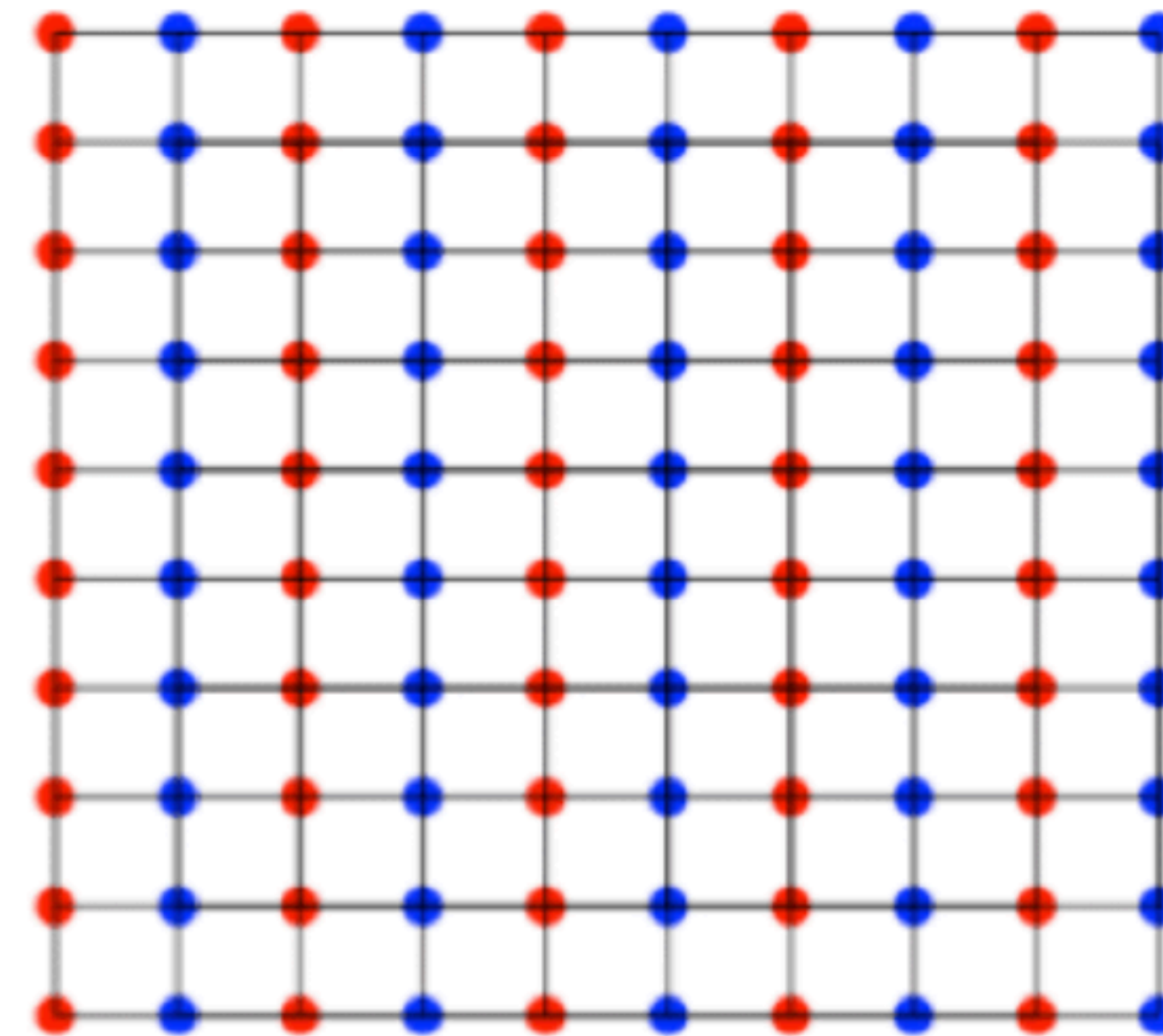
The Curse of Dimensionality

- Richard Bellman first introduced this idea identifying a problem with the relationship between dimensionality and volume
- As dimensionality increases, the volume of the space grows much faster than the number of data points can fill it
- Previously collected sample points become sparse and leave most of the space empty
- By adding extra dimensions, data points needed to cover the entire volume increase exponentially

10 evenly spaced points cover it



(A) 1-D



(B) 2-D

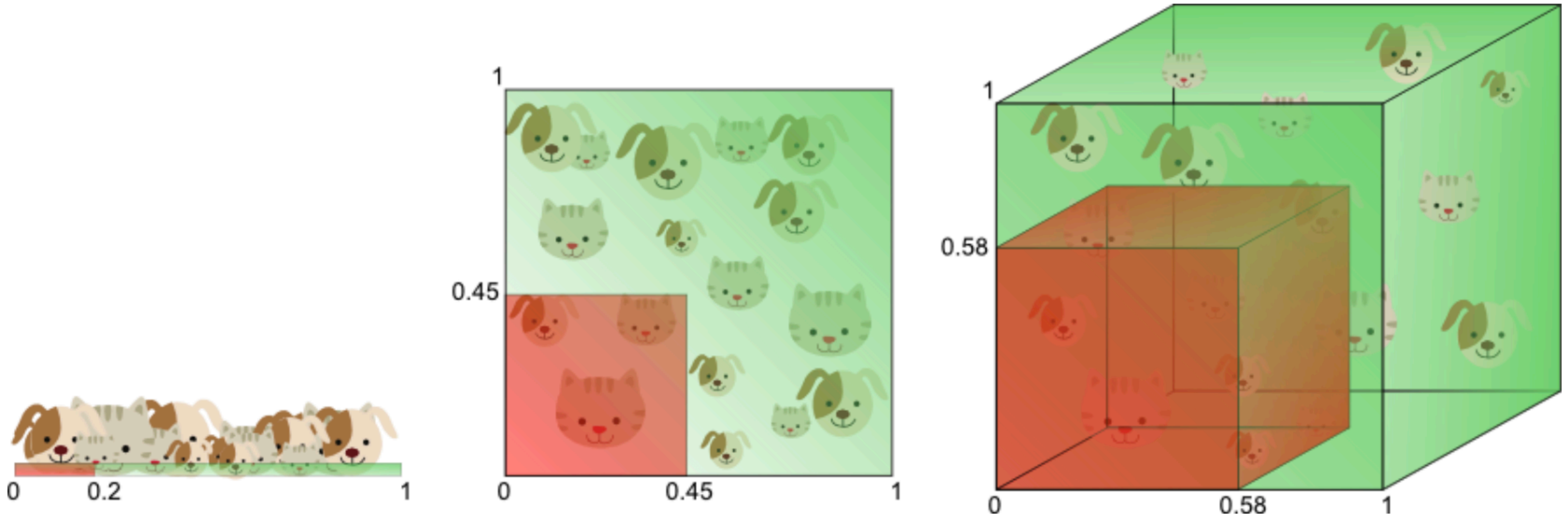
10 x 10
evenly spaced points
cover it



- Credit image: wikipedia

- Credit image: Tiange Cui

Amount of training data needed to cover 20% of the feature range grows exponentially with the number of dimensions



Types of Unsupervised Learning

- **Dimensionality reduction**: finding a low-dimensional representation to characterize and visualize the data
 - Principle Component Analysis (PCA)
 - t-distributed Stochastic Neighbor Embedding (t-SNE)
 - Uniform Manifold Approximation and Projection (UMAP)
 - Autoencoder
- **Clustering**: categorizes data into groups according to similarity
 - K-means Clustering
 - Hierarchical Clustering
 - K-nearest Neighbors

Dimensionality reduction: Autoencoder (Computational Intensity!)

THE ASTROPHYSICAL JOURNAL, 980:257 (14pp), 2025 February 20

© 2025. The Author(s). Published by the American Astronomical Society.

OPEN ACCESS

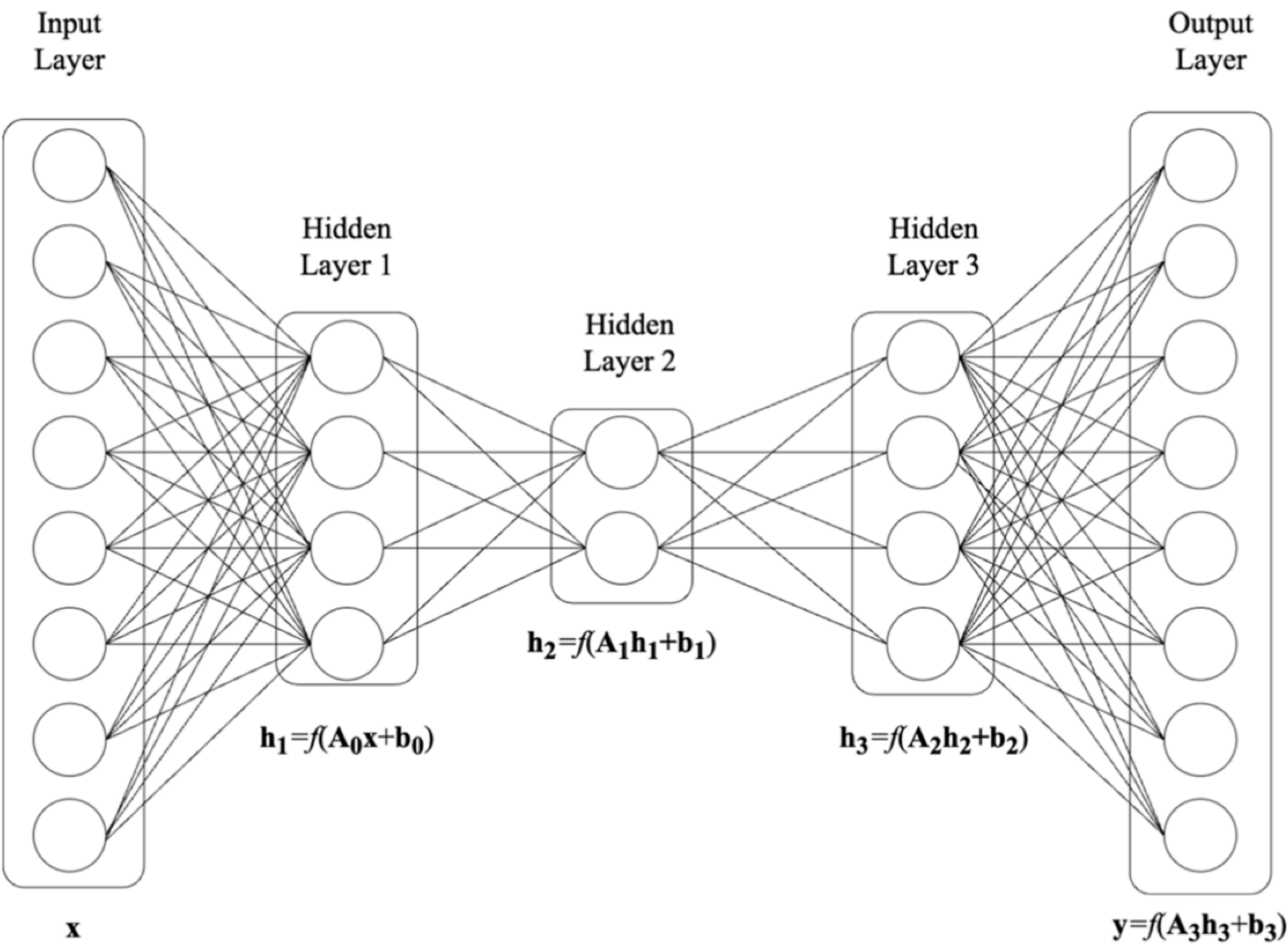
<https://doi.org/10.3847/1538-4357/adae85>



Extracting the X-Ray Reverberation Response Functions from the Active Galactic Nucleus Light Curves Using an Autoencoder

Sanhanat Deesamutara¹ , Poemwai Chainakun^{1,2} , Tirawut Worrakitpoonpon^{1,2} , Kamonwan Khanthasombat¹,
Wasutep Luangtip^{3,4} , Jiachen Jiang^{5,6} , Francisco Pozo Nuñez⁷ , and Andrew J. Young⁸

- Autoencoders are a type of artificial neural network designed to encode data into a lower-dimensional representation and then decode it back into its original form
- Input and output have the same dimension
- The narrowest layer should compress the most important data information



Dimensionality reduction: Autoencoder (Computational Intensity!)

THE ASTROPHYSICAL JOURNAL, 980:257 (14pp), 2025 February 20

© 2025. The Author(s). Published by the American Astronomical Society.

OPEN ACCESS

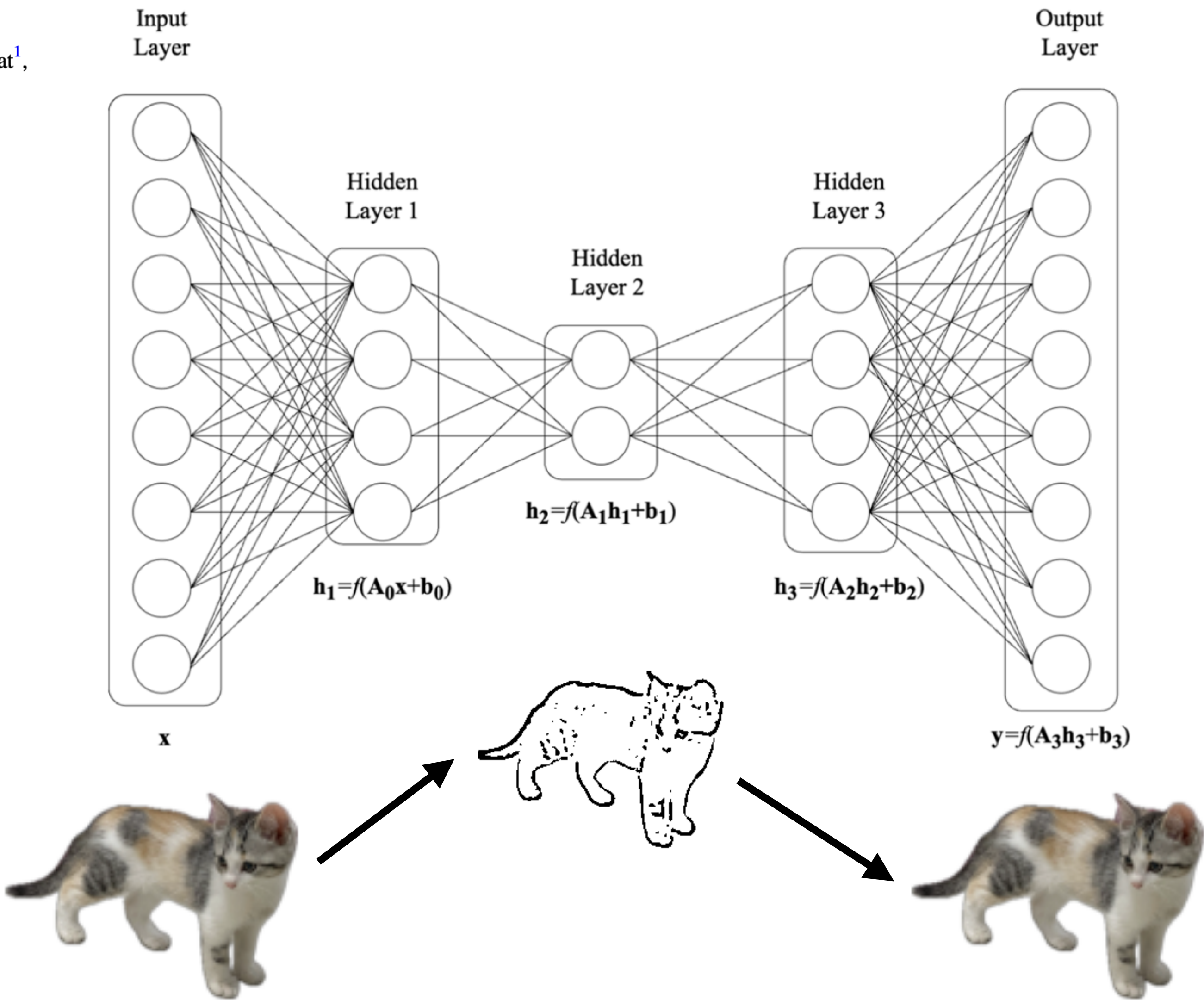
<https://doi.org/10.3847/1538-4357/adae85>



Extracting the X-Ray Reverberation Response Functions from the Active Galactic Nucleus Light Curves Using an Autoencoder

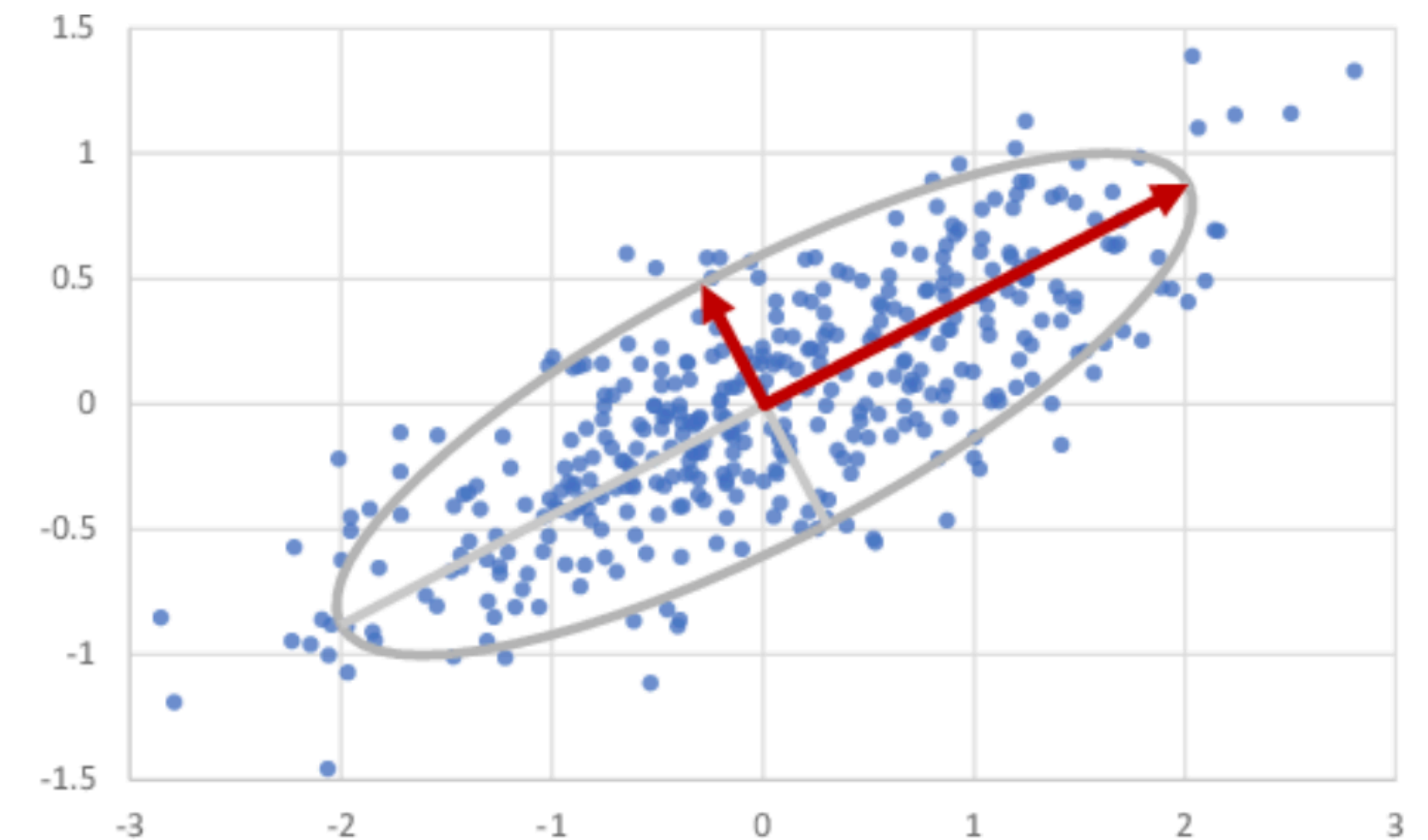
Sanhanat Deesamutara¹, Poemwai Chainakun^{1,2}, Tirawut Worrakitpoonpon^{1,2}, Kamonwan Khanthasombat¹,
Wasutep Luangtip^{3,4}, Jiachen Jiang^{5,6}, Francisco Pozo Nuñez⁷, and Andrew J. Young⁸

- Autoencoders are a type of artificial neural network designed to encode data into a lower-dimensional representation and then decode it back into its original form
- Input and output have the same dimension
- The narrowest layer should compress the most important data information

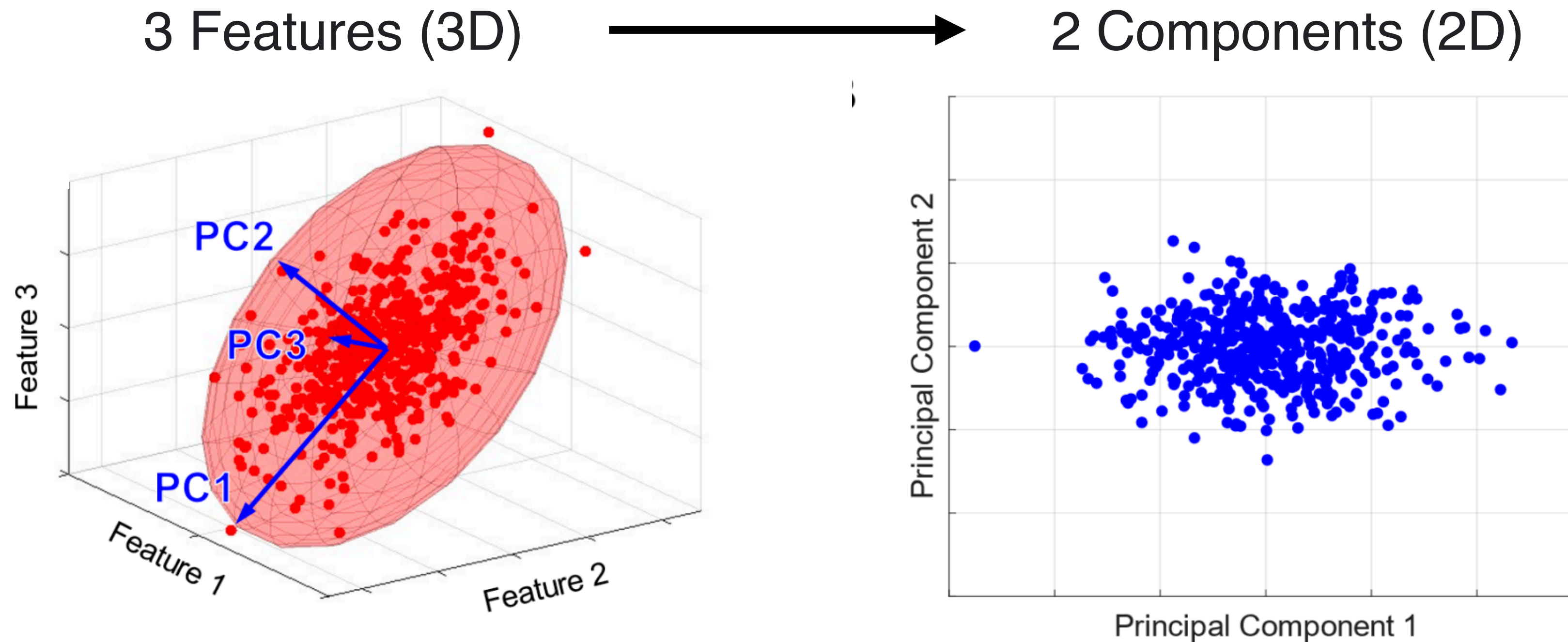


Dimensionality reduction: PCA

- Principal Component Analysis (PCA) is a simple and widely used method for dimensionality reduction
- Data cloud of all observations is considered or approximated as an ellipsoid
- PCA finds the directions in which this ellipsoid is **most stretched (greatest variance)**.
- These directions become the **new axes** (the new basis).
 - Each axis points in a direction where the data varies the most
 - All axes are **orthogonal** (perfectly independent)
- Components are uncorrelated and ranked by explained variance
- Components are linear combinations of original attributes



Dimensionality reduction: PCA



By keeping only the first few PCs and discarding the rest, we obtain a lower-dimensional representation with little loss of the original data's variance.

Dimensionality reduction: PCA

Mathematically speaking!

- Principal components are an ordered set of vectors that provide the best linear approximation of the data:

$$f(\lambda) = \mu + V_q \lambda$$

- μ : location (mean) vector in $\mathbb{R}^p$
- V_q : $p \times q$ matrix of **q orthogonal unit vectors** (principal components)
- λ : q -dimensional vector of parameters (component scores)

- Find the components that best reconstruct the original data by minimizing the total reconstruction error:

$$\min_{\mu, \{\lambda_i\}, V_q} \sum_{i=1}^N \|x_i - \mu - V_q \lambda_i\|^2$$

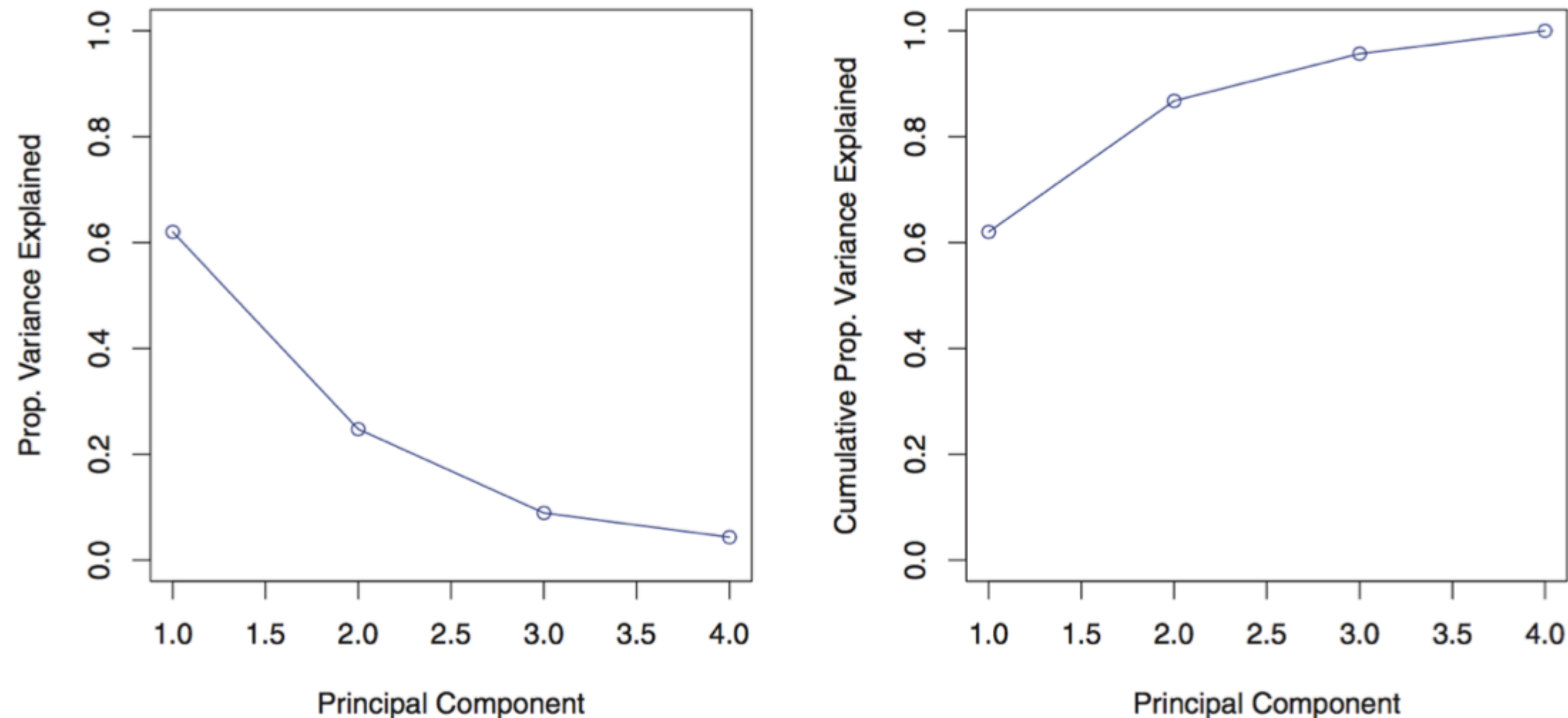
✓ Interpretation

Term	Meaning
μ	The "center" of the dataset
V_q	The new axes (principal components)
λ	How far along each new axis a point is
$f(\lambda)$	The reconstructed point in original dimension

Dimensionality reduction: PCA

- How many principal components?
 - > Check out the variance explained by each component
 - > In practice, we choose the number of principal components such that we can explain 70-90% of the initial data variance

Scree plot



Dimensionality reduction: PCA

Example: PCA - Fisher's iris dataset

<https://drive.google.com/file/d/1HIWUtduiosCvNMpYY0OH1iUBvILfVtCo/view?usp=sharing>



Iris setosa



Iris versicolor



Iris virginica

Original features:

- sepal length (cm)
- sepal width (cm)
- petal length (cm)
- petal width (cm)



- Use a Decision Tree to classify Iris flower species based on the original features
- Apply PCA to reduce the features to two principal components, then perform the classification again
- Check out the percent of variance that can be explained by each of the selected components

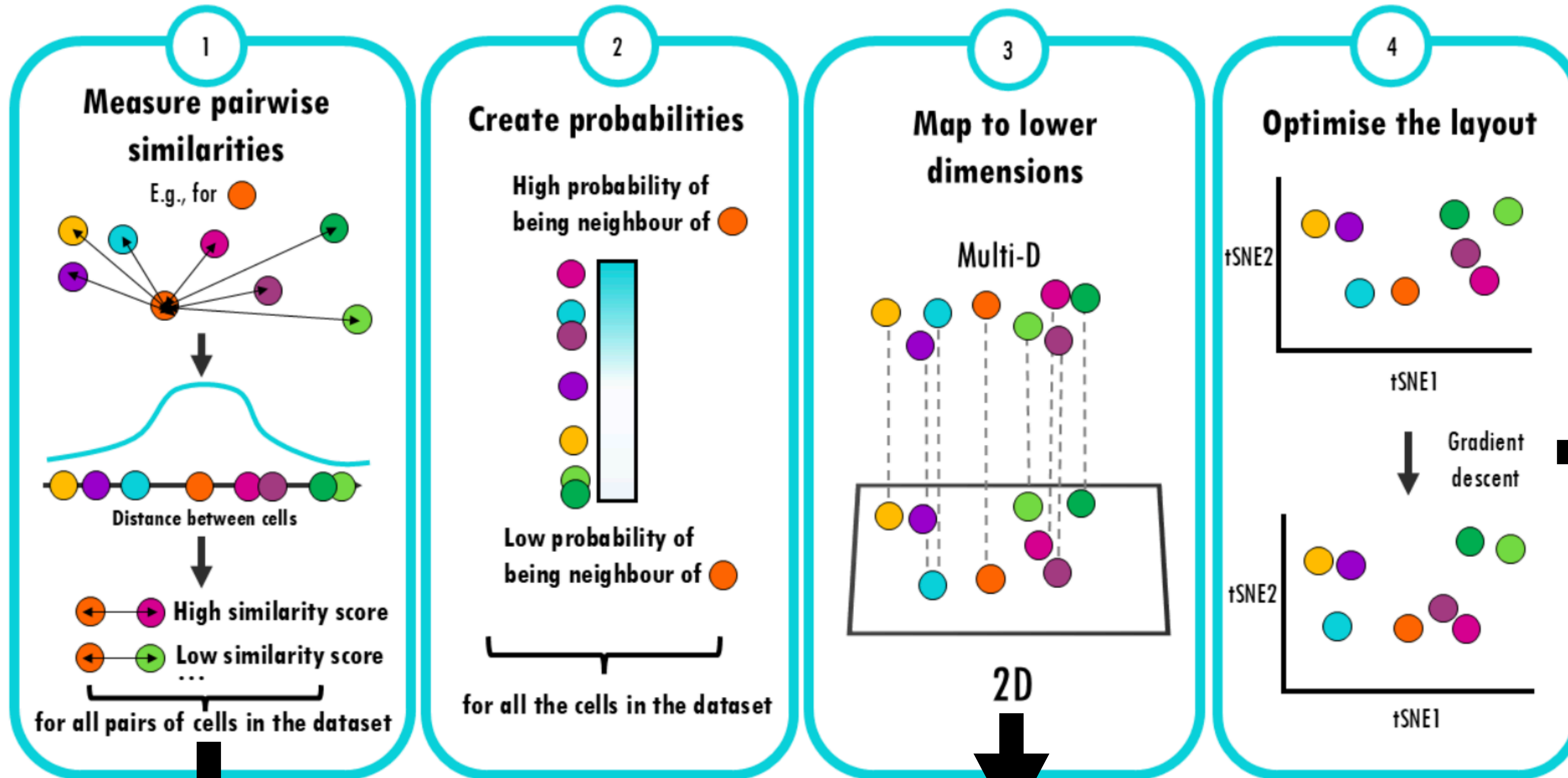
Dimensionality reduction: t-SNE

- **t-distributed Stochastic Neighbor Embedding (t-SNE)** is an unsupervised **non-linear** dimensionality reduction
- Algorithm allows us to separate data that cannot be separated by a straight line

Characteristic	t-SNE	PCA
Type	Non-linear dimensionality reduction	Linear dimensionality reduction
Goal	Preserve local pairwise similarities	Preserve global variance
Best used for	Visualizing complex, high-dimensional data	Data with linear structure
Output	Low-dimensional representation	Principal components
Use cases	Clustering, anomaly detection, NLP	Noise reduction, feature extraction
Computational intensity	High	Low
Interpretation	Harder to interpret	Easier to interpret

Dimensionality reduction: t-SNE

How does t-SNE summarise many dimensions into 2?



t-SNE calculates how similar each pair of cells is to each other. These similarities are turned into probabilities.

Place points so that similar cells in the original space remain close in the low-dimensional space, while dissimilar cells are far apart.

Use gradient descent to adjust point positions so that similarity relationships in the low-dimensional space match those in the original space.

Dimensionality reduction: t-SNE

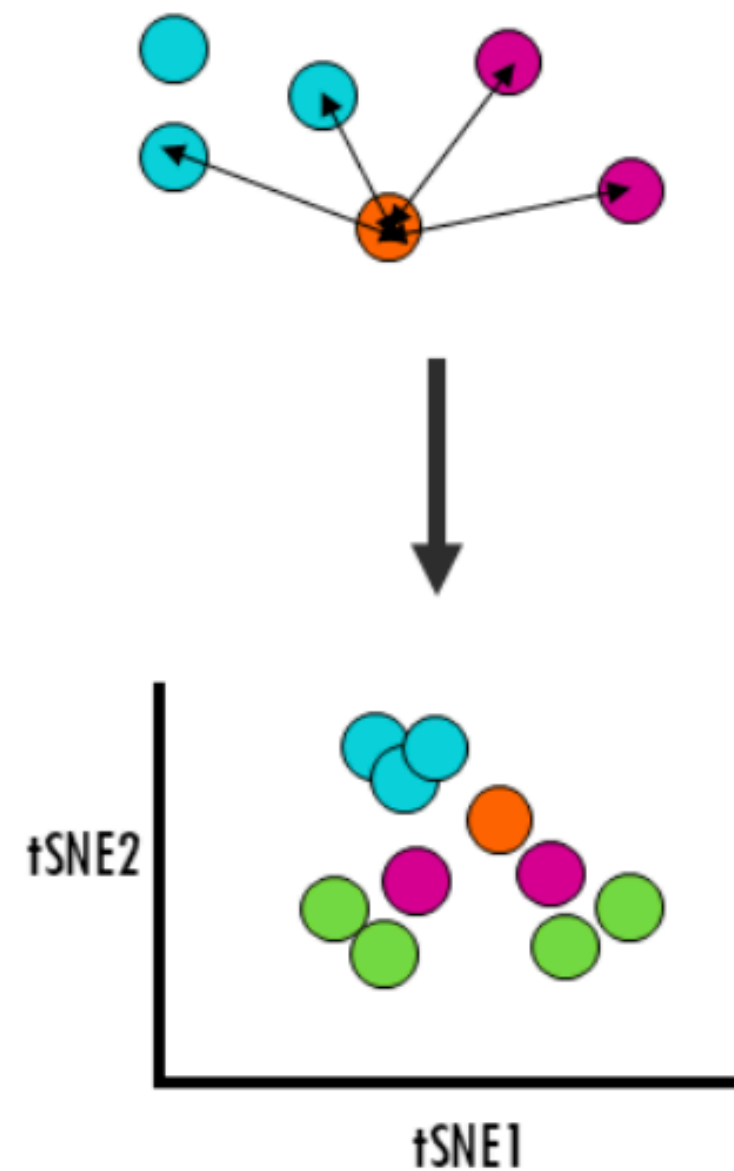
- Main hyperparameter of t-SNE is **perplexity**, which is related to how many neighbours each point is considering when building its similarity map
- **Perplexity** controls how the algorithm balances local versus global relationships in the data

Low perplexity

E.g., for ●

Few neighbours

Focus on local relationships

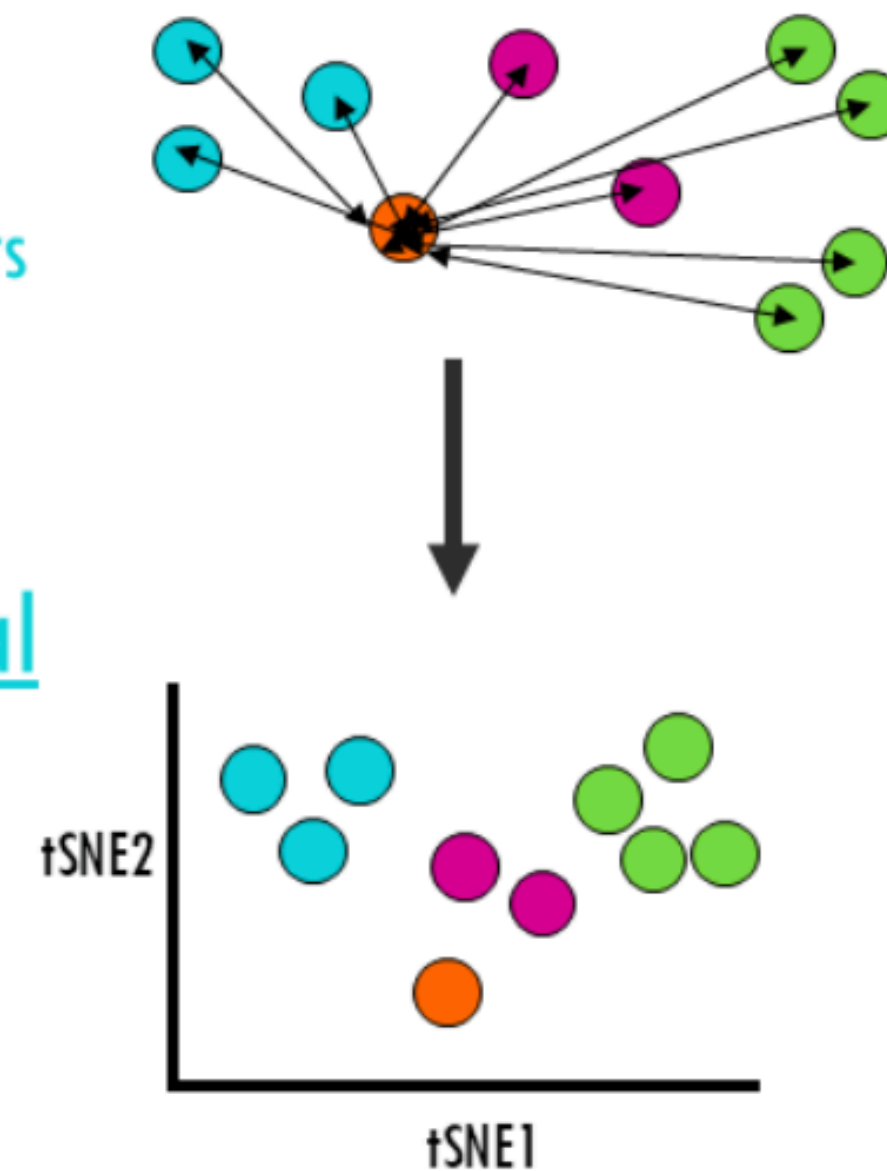


High perplexity

E.g., for ●

Many neighbours

Focus on global relationships



Dimensionality reduction: UMAP

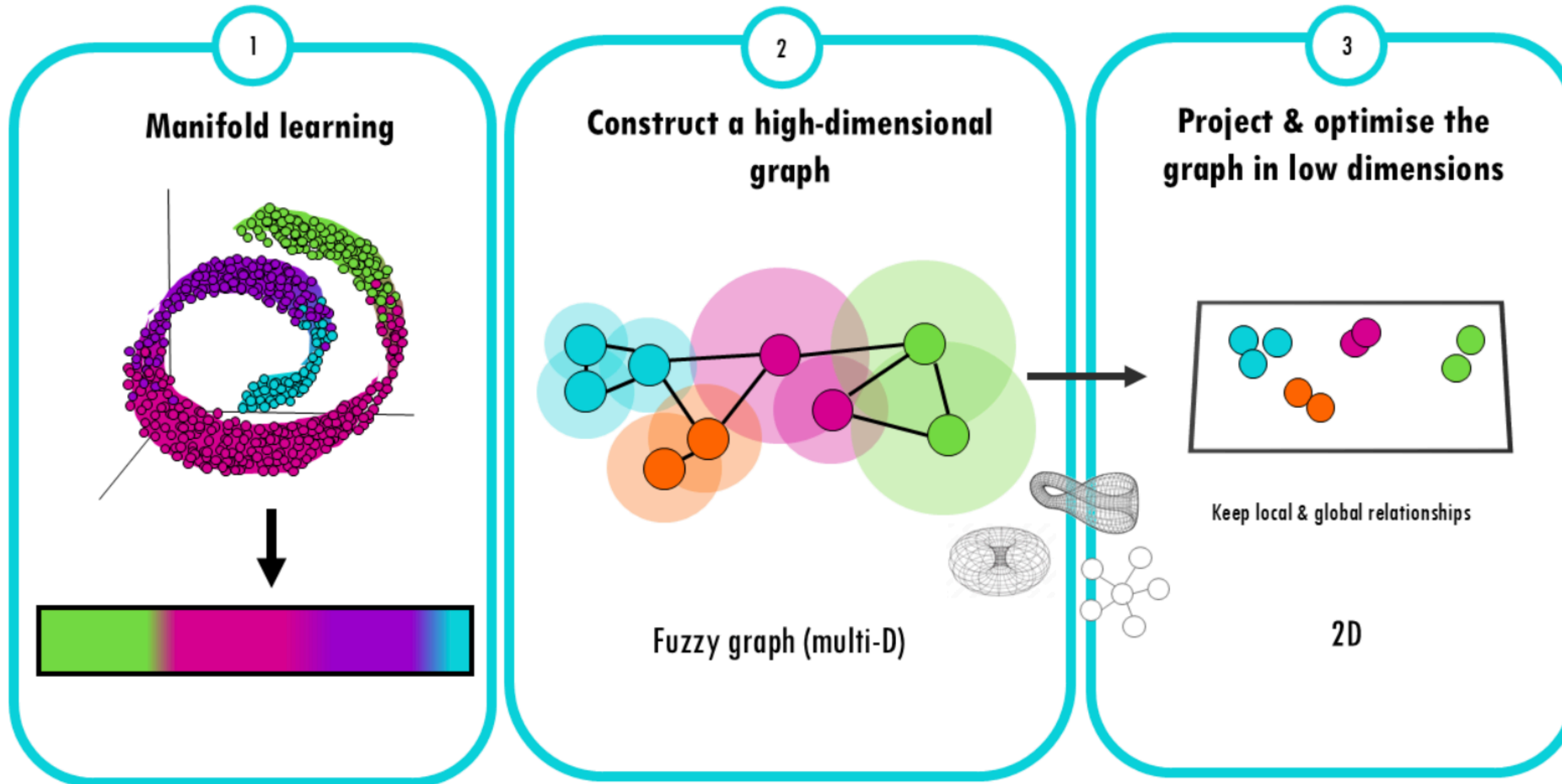
- **Uniform Manifold Approximation and Projection (UMAP)** has emerged as a popular alternative to t-SNE
- UMAP works by constructing a high-dimensional graph of the data and then finding the closest low-dimensional representation, effectively grouping similar data points together

Technique	Computational complexity	Features	Suitability for large datasets
t-SNE	$O(N^2)$	Preserves local structure, highly customizable	Moderate (slow for large datasets)
UMAP	$O(N \log N)$	Balances local and global structure, faster	High (handles large datasets efficiently)
PCA	$O(Nd^2)$	Linear reduction, interpretable components	High (very efficient)

- N : Number of data points
- d : Number of features in the data

Dimensionality reduction: UMAP

How does UMAP summarise many dimensions into 2?



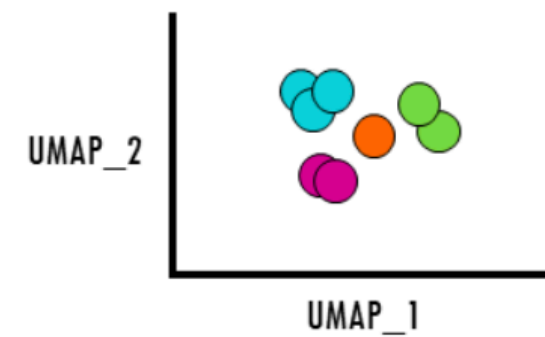
- UMAP optimizes the layout of a 2D graph to preserve the connections computed in the original high-dimensional fuzzy graph.
- Unlike t-SNE, which is based on probability distributions, UMAP relies on concepts from topology (the study of shapes) to optimize the graph.

Dimensionality reduction: UMAP

- Main hyperparameter of UMAP is **n_neighbors**, which controls how much of the local versus global structure is preserved
- Another important hyperparameter of UMAP is **min_dist**, which is the minimum distance between points in the low-dimensional space

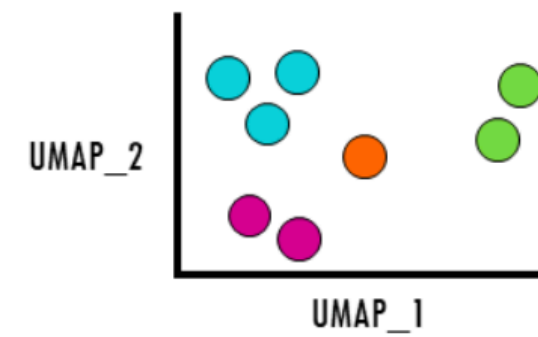
Low n_neighbours

Focus on local relationships



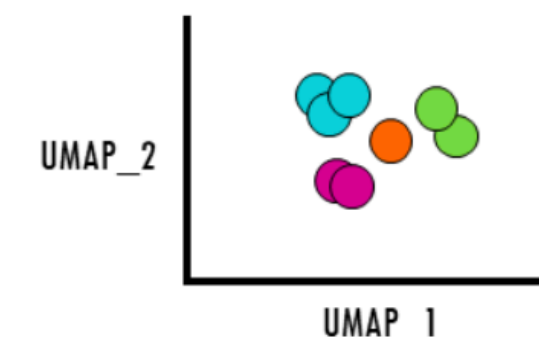
High n_neighbours

Focus on global relationships



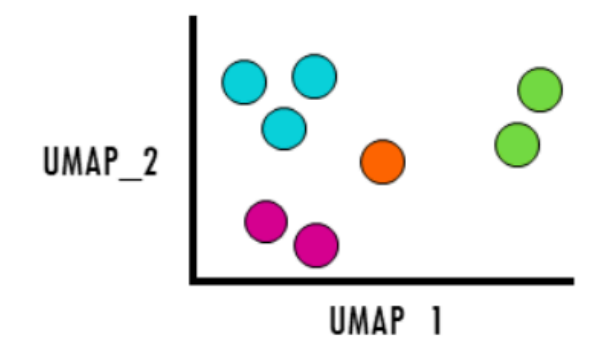
Low min_dist

Focus on local relationships



High min_dist

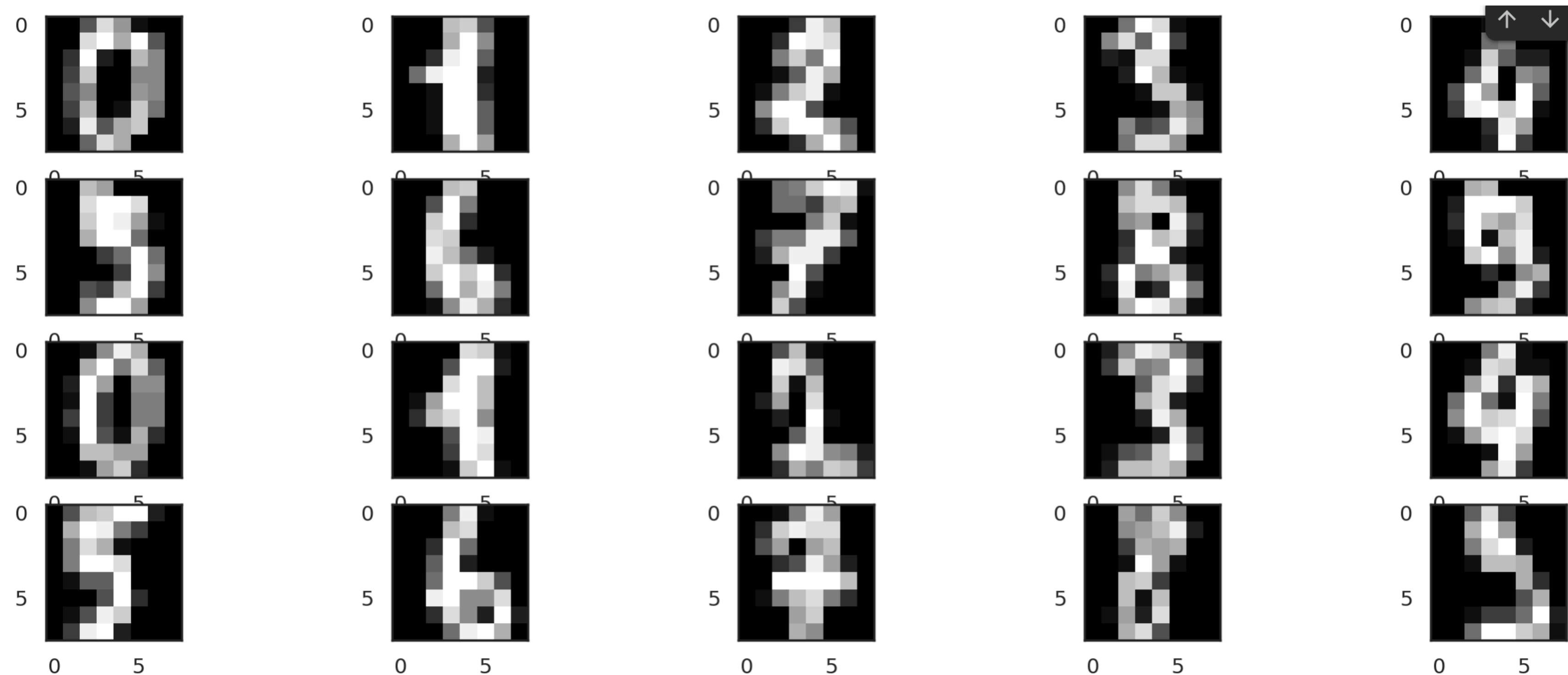
Focus on global relationships



Dimensionality reduction

Example: PCA+t-SNE+UMAP - Handwritten numbers dataset

https://drive.google.com/file/d/1x8Z6dpgYZIU_McorNMEzWZSR7OF7KrQs/view?usp=sharing



Dimensionality reduction

Exercise 1: breast cancer

Predict the diagnosis of breast tissue (malignant vs. benign) using Logistic Regression with an appropriately selected feature set.

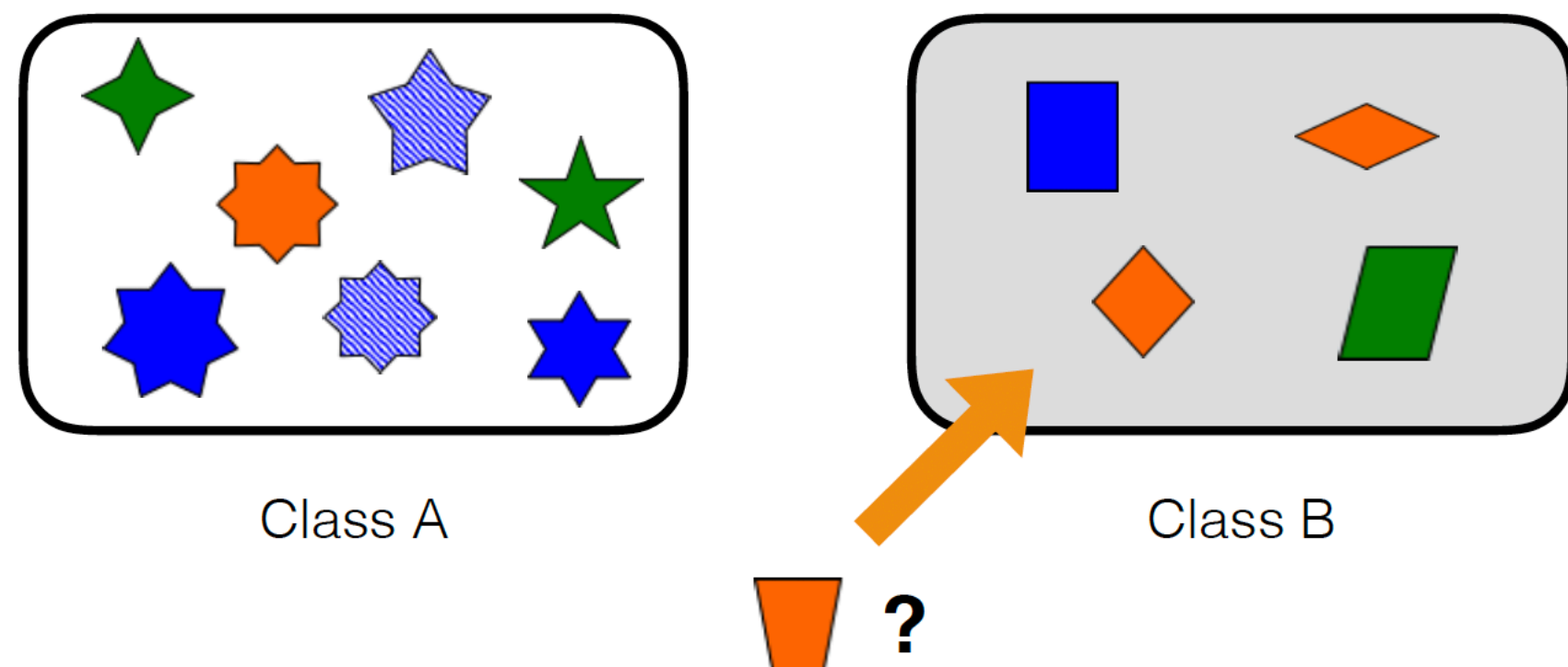
<https://drive.google.com/file/d/1KNDEc5cOHHr4dV11HHBkTuTQEKI-d9Bq/view?usp=sharing>

- Explore your data
- Use PCA to reduce the dimension of the data
- Select an appropriate number of principal components that explain $\sim 90\%$ of the data variance
- Train and test the model to predict the diagnosis
- Show confusion matrix
- Reconstruct data and check Information loss
- Use t-SNE and UMAP, compared the results with those obtained using PCA

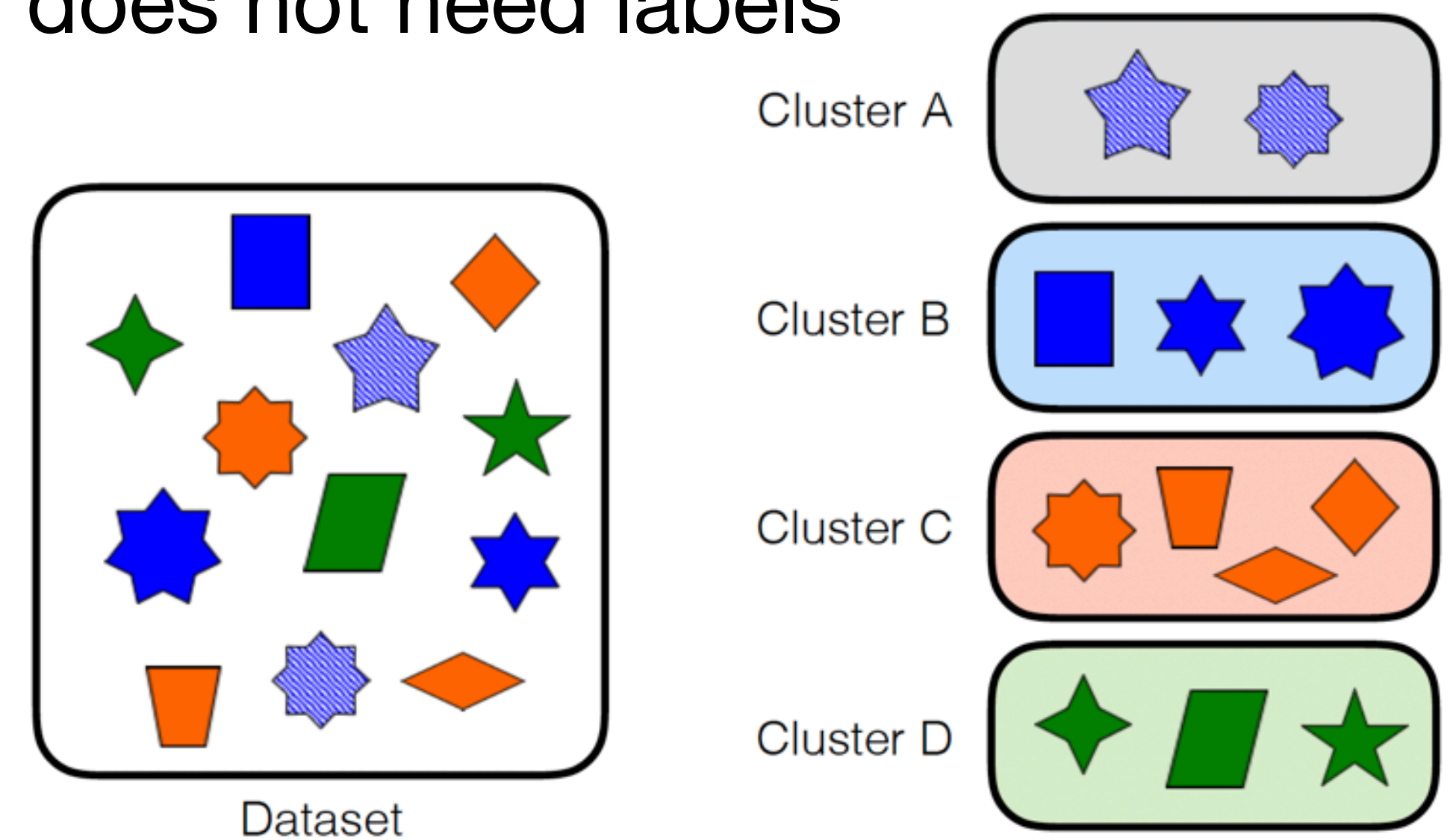
Clustering

Clustering v.s. Classification

Classification is a supervised-learning task, so it requires labeled data for training



Clustering is an unsupervised learning task, so it does not need labels



There are many clustering algorithms available today, but we will focus on two main algorithms:

- K-means Clustering
- Gaussian Mixture Model (GMM)

K-means Clustering

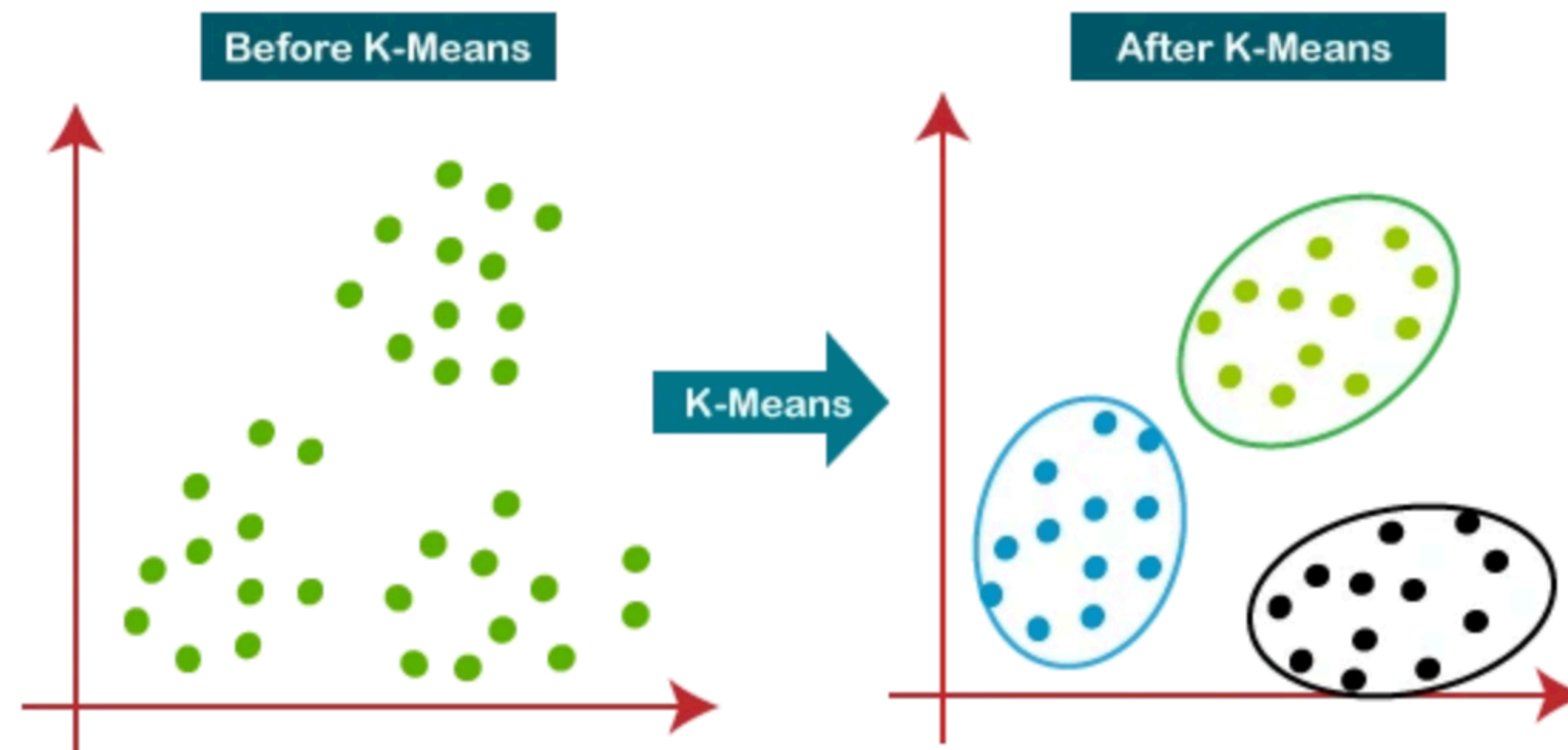
- K-means is a centroid-based algorithm, or distance-based algorithm, where we calculate the distances to assign a point to a cluster
- Some well-known choice of the distance functions are

$$\text{Dist}(\mathbf{x}, \mathbf{x}') = \begin{cases} \|\mathbf{x} - \mathbf{x}'\| = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}, & \text{(Euclidean)} \\ \sum_{i=1}^d |x_i - x'_i|, & \text{(Manhattan)} \\ \max_{i=1, \dots, d} |x_i - x'_i|, & \text{(Maximum)} \\ \sum_{i=1}^d \mathbb{I}(x_i \neq x'_i), & \text{(Hamming distance for binary features)} \end{cases}$$

K-means Clustering

Partition the data into K clusters that satisfy two properties:

1. Each observation belongs to at least one of the K clusters
2. Clusters are non-overlapping. No observation belongs to more than one cluster



Goal of good clustering:
minimize variation within each cluster

K-means Clustering

- Good clustering → minimize within-cluster variation
- Let each cluster center be C_k , we define $W(C_k)$ as the within-cluster variation
- K-means objective: find clusters $C_1, \dots, C_k$ that minimize the total variation
- So, K-means aim to solve

$$\min_{C_1, \dots, C_K} \sum_{k=1}^K W(C_k)$$

- The most common choice to measure within-cluster variation is squared Euclidean distance (measure Euclidean distances between k centroids and individual data points)

$$W(C_k) = \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2$$

K-means Clustering Algorithm

- In practice, we can optimize K-means with an iterative algorithm

Step 1: Choose the number of clusters K

Step 2: Randomly select K initial centroids

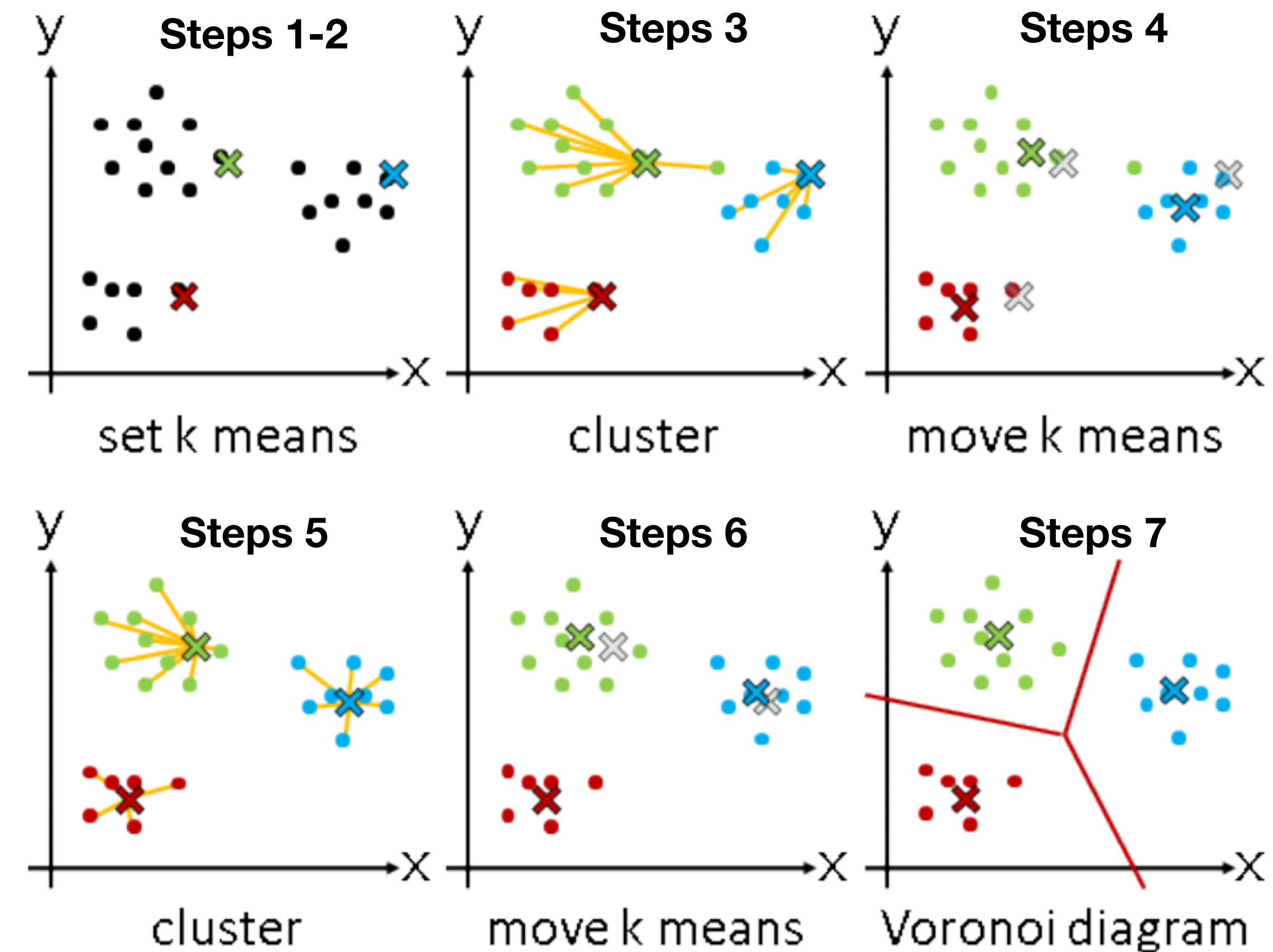
Step 3: Assign each data point to the nearest centroid

Step 4: Compute new centroids for all clusters

Step 5: Reassign points to the nearest updated centroid

Step 6: Repeat Steps 4–5 until no points change clusters

Step 7: Finish - final clusters are formed



K-means Clustering Algorithm

In Python, the scikit-learn library provides an implementation of K-Means

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

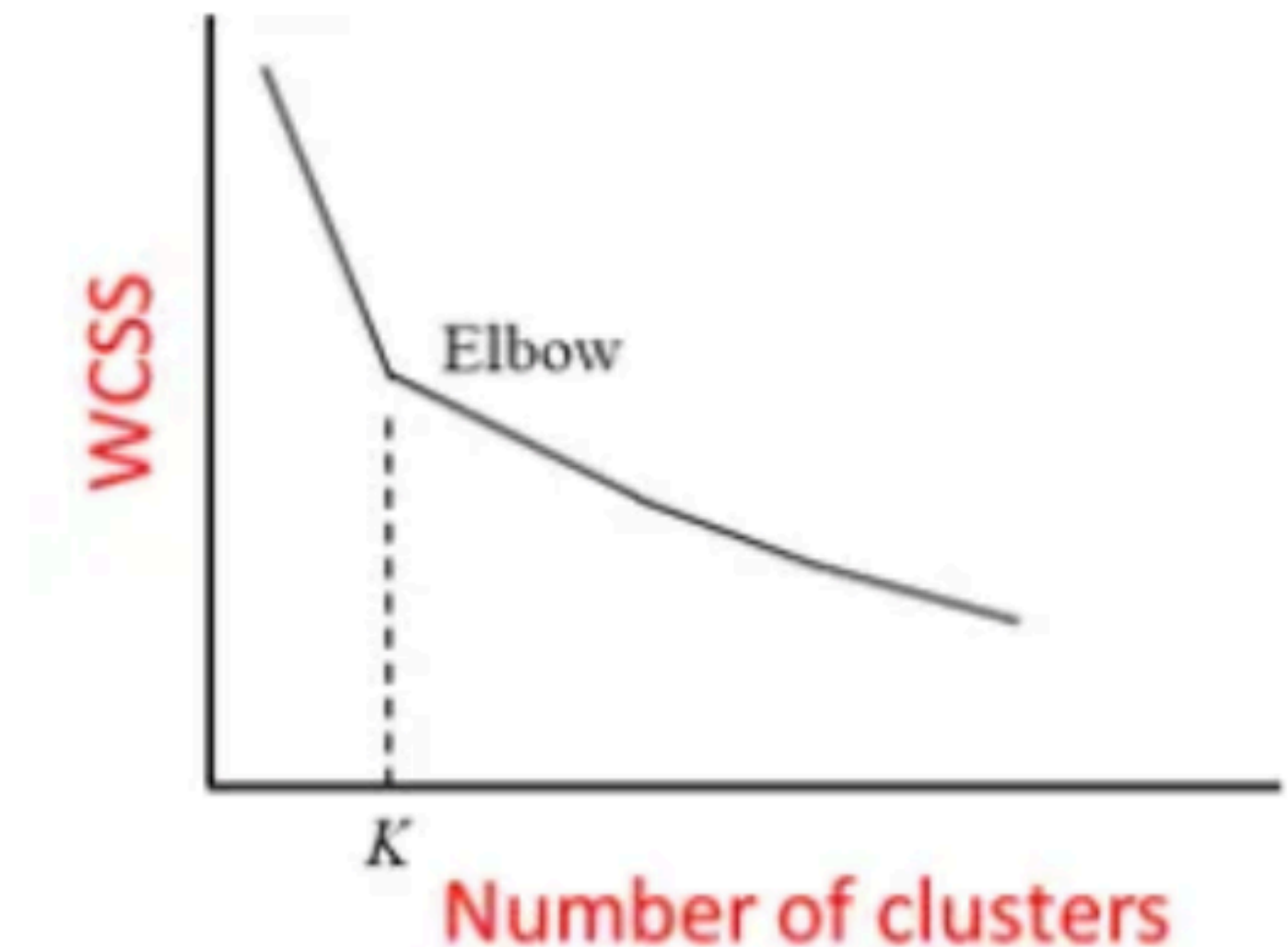
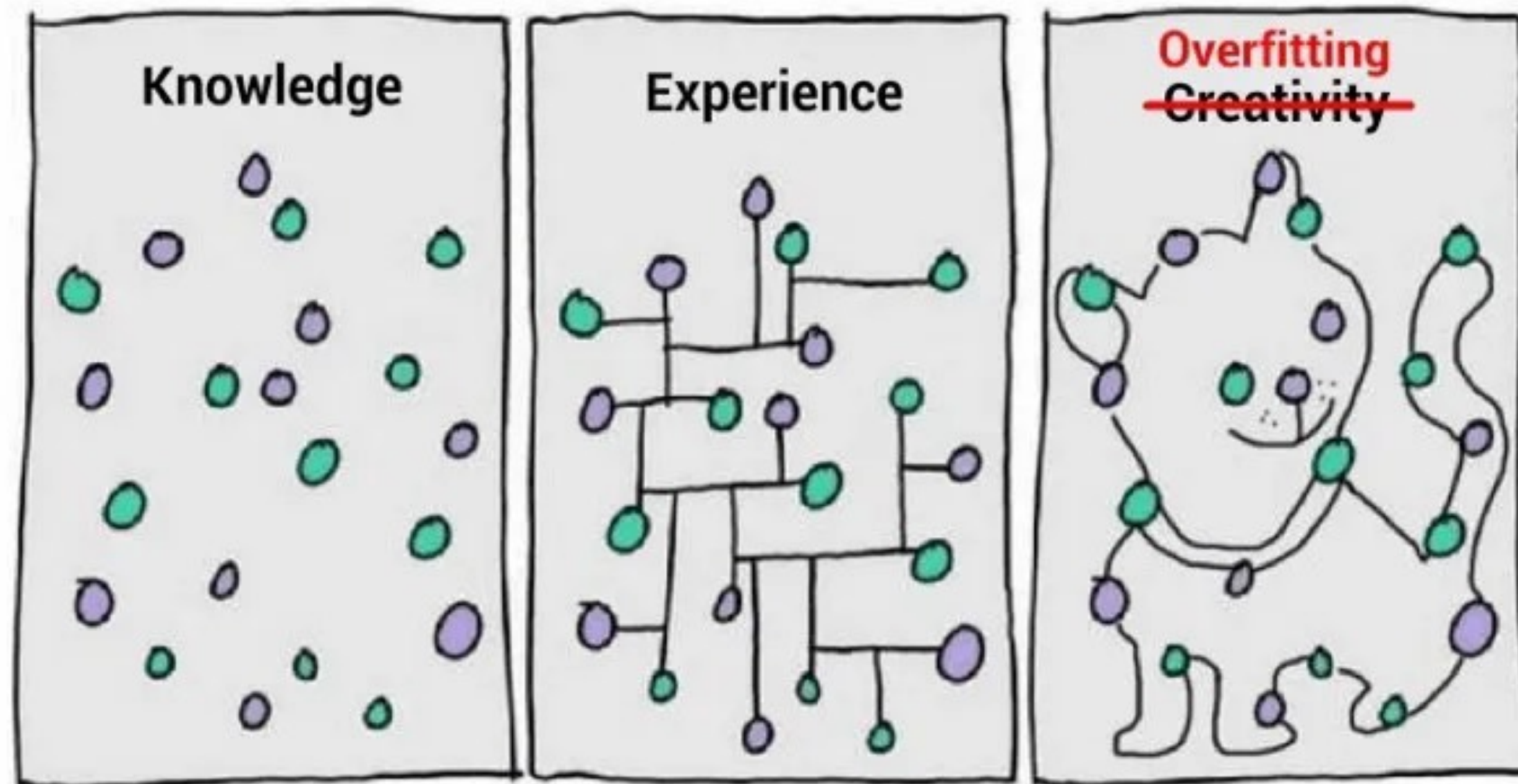
KMeans



```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++',  
n_init='auto', max_iter=300, tol=0.0001, verbose=0, random_state=None,  
copy_x=True, algorithm='lloyd')
```

How to determine $K = ?$

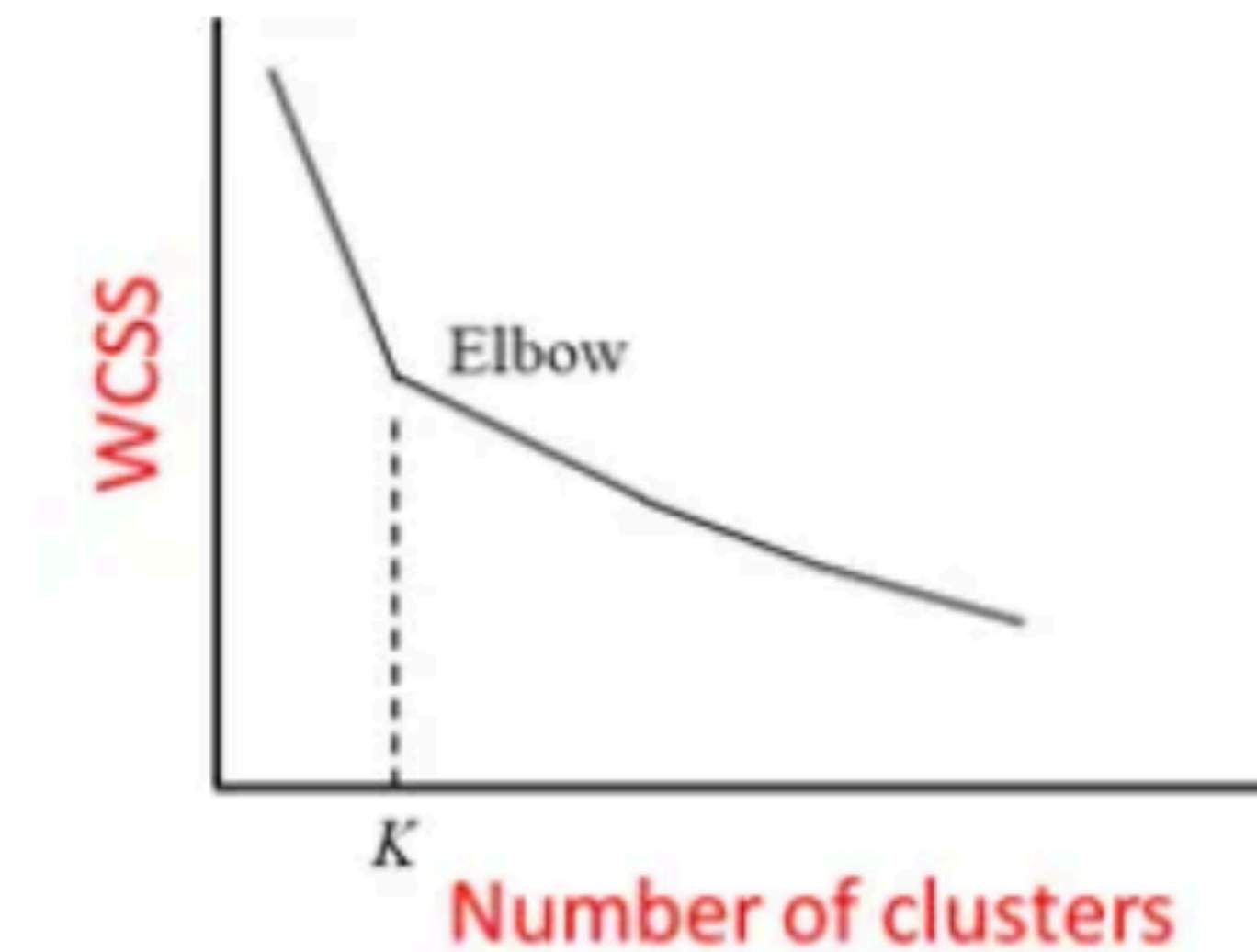
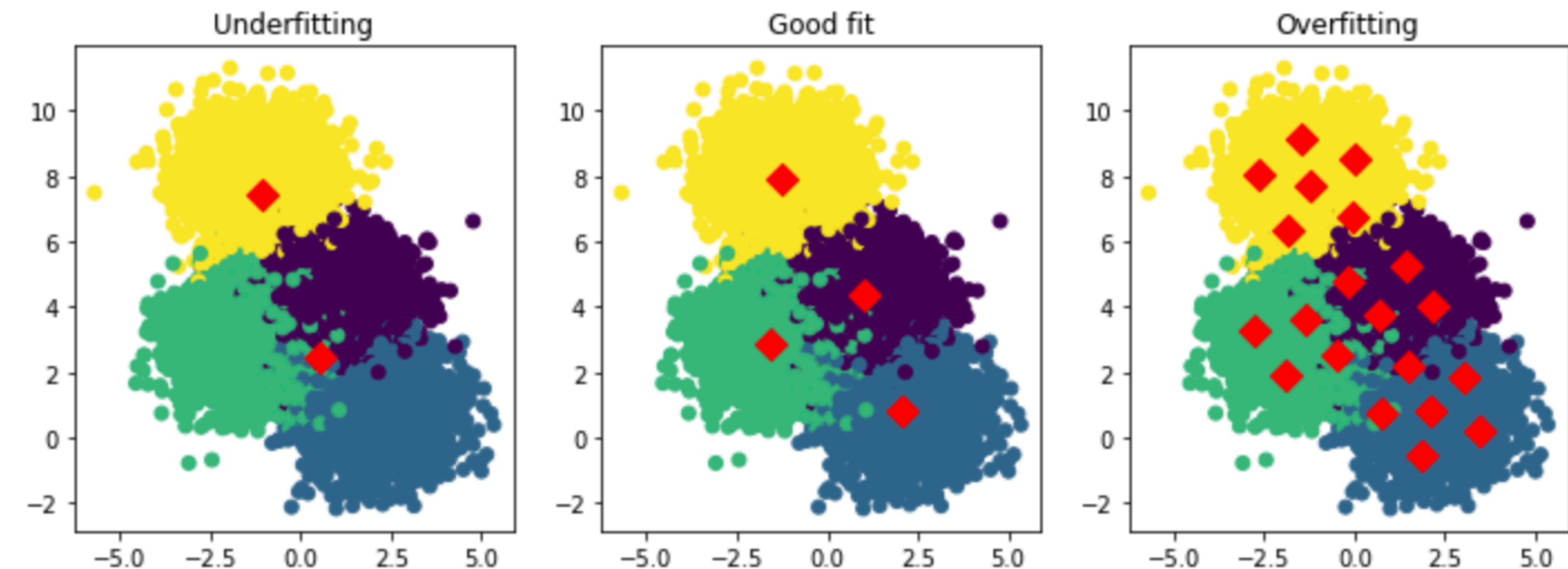
- A common problem: OVERFITTING



- In K-means clustering, the elbow method can be used to determine the optimal number of clusters (K) and avoid overfitting

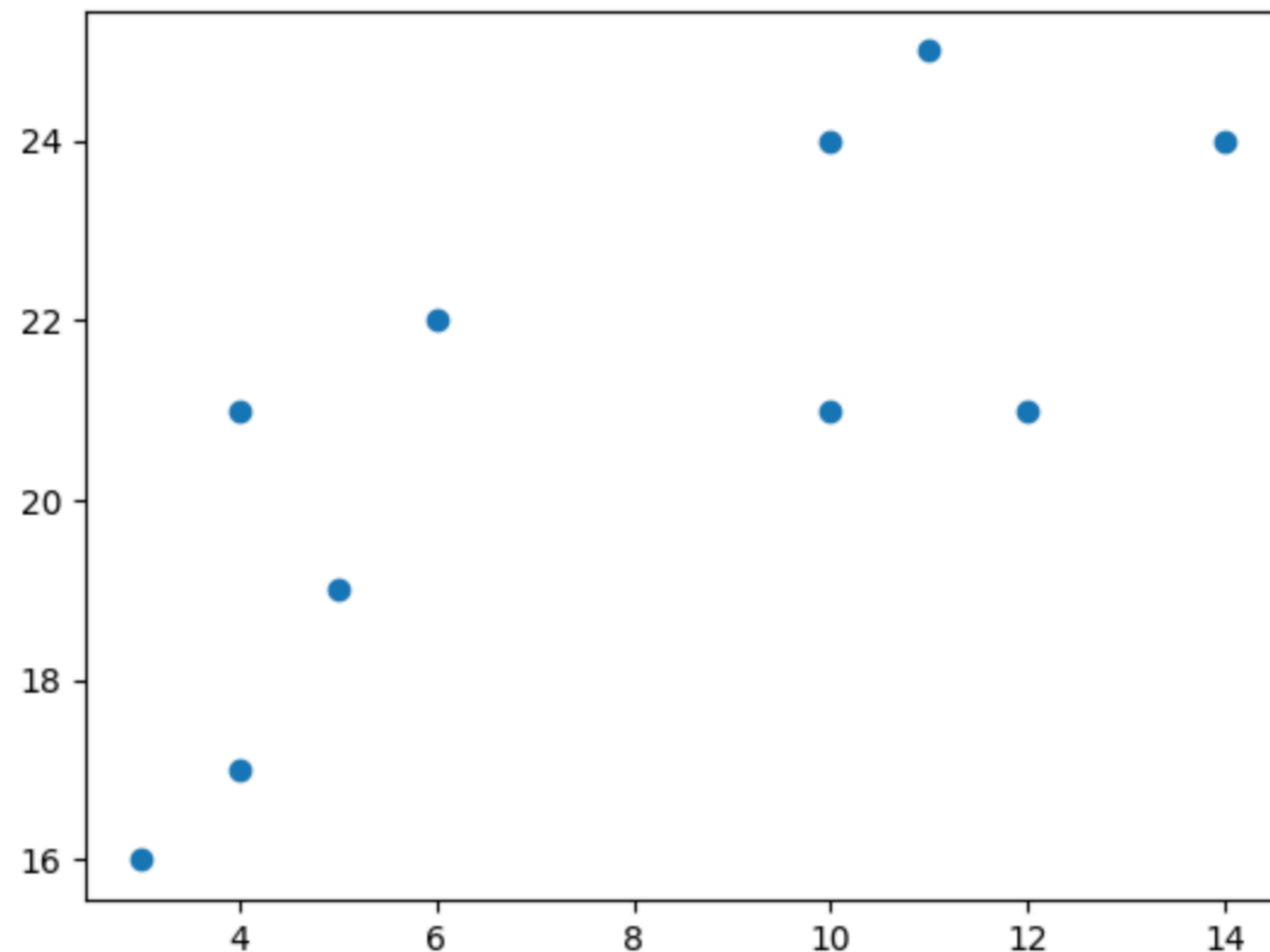
Elbow method to determine K

- Calculate the sum of squared distances of samples to their closest cluster center, referred to as the Within-Cluster Sum of Squares (WCSS)
- Plot the WCSS against different K values
- As K increases, WCSS decreases (clusters fit better)
- The elbow point indicates where adding more clusters yields diminishing returns
- Typically, K at the elbow point is chosen as the optimal number of clusters that provides a balance between model accuracy and simplicity —> avoid **overfitting**
- We can use `kmeans.inertia_` from `sklearn.cluster.KMeans` to find the WCSS



Example: K-means 101

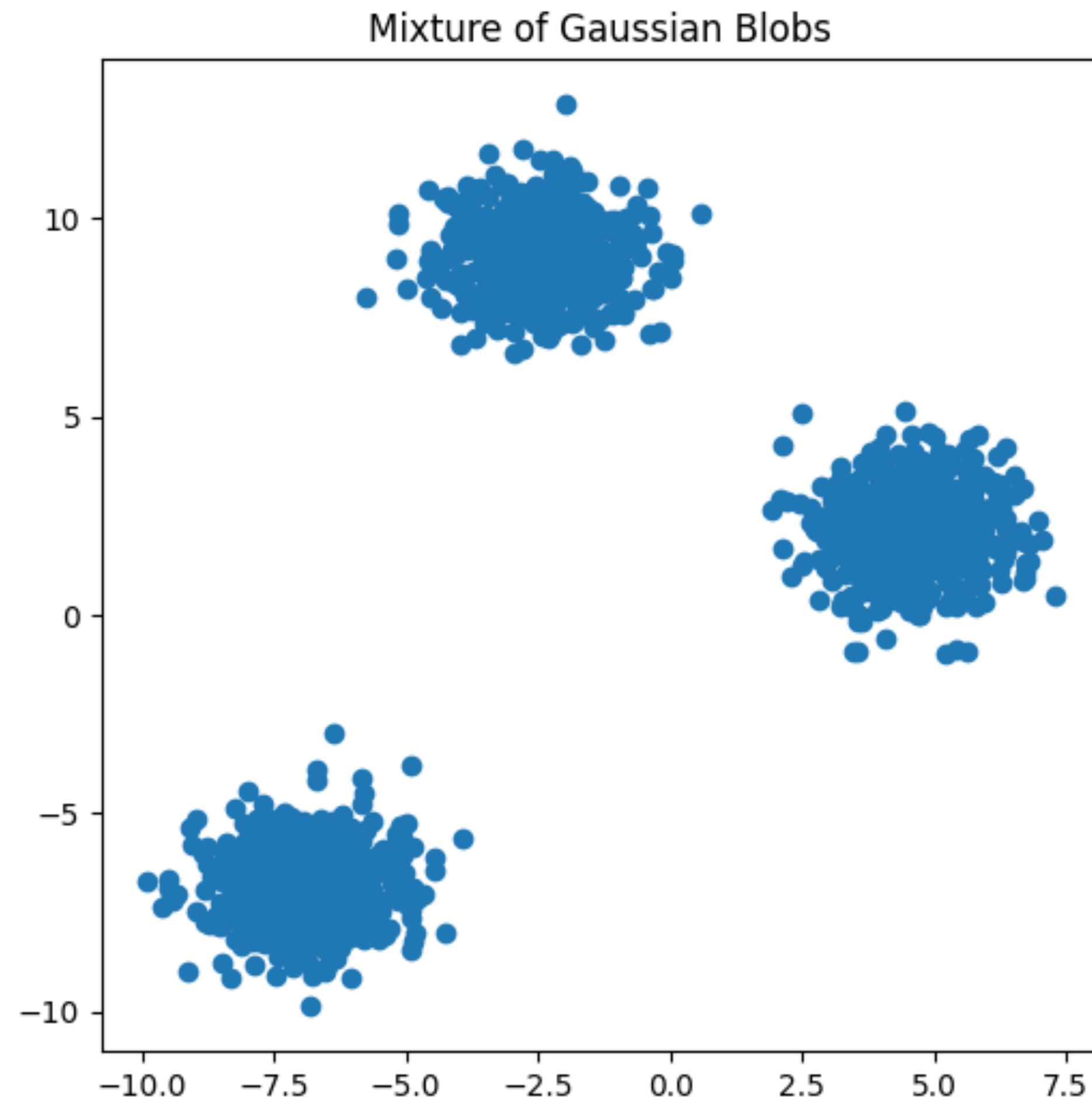
https://drive.google.com/file/d/19cKk1EbKt5lt44ISGF-zwaQNOc_FtKE6/view?usp=sharing



- Perform basic K-Means clustering using the scikit-learn library
- Investigate the optimal number of clusters (K) using the elbow method
- Plot the center of the clusters

Example: K-means - Gaussian blobs

https://drive.google.com/file/d/1ApUZCwlgqBfkICVDa6N4B_v44n_wiZkq/view?usp=sharing



- Generate 3 clusters with isotropic (spherical) gaussian blobs
- Perform suitable clustering using the scikit-learn library
- Investigate the optimal number of clusters (K) using the elbow method
- Plot the center of the clusters

KMeans

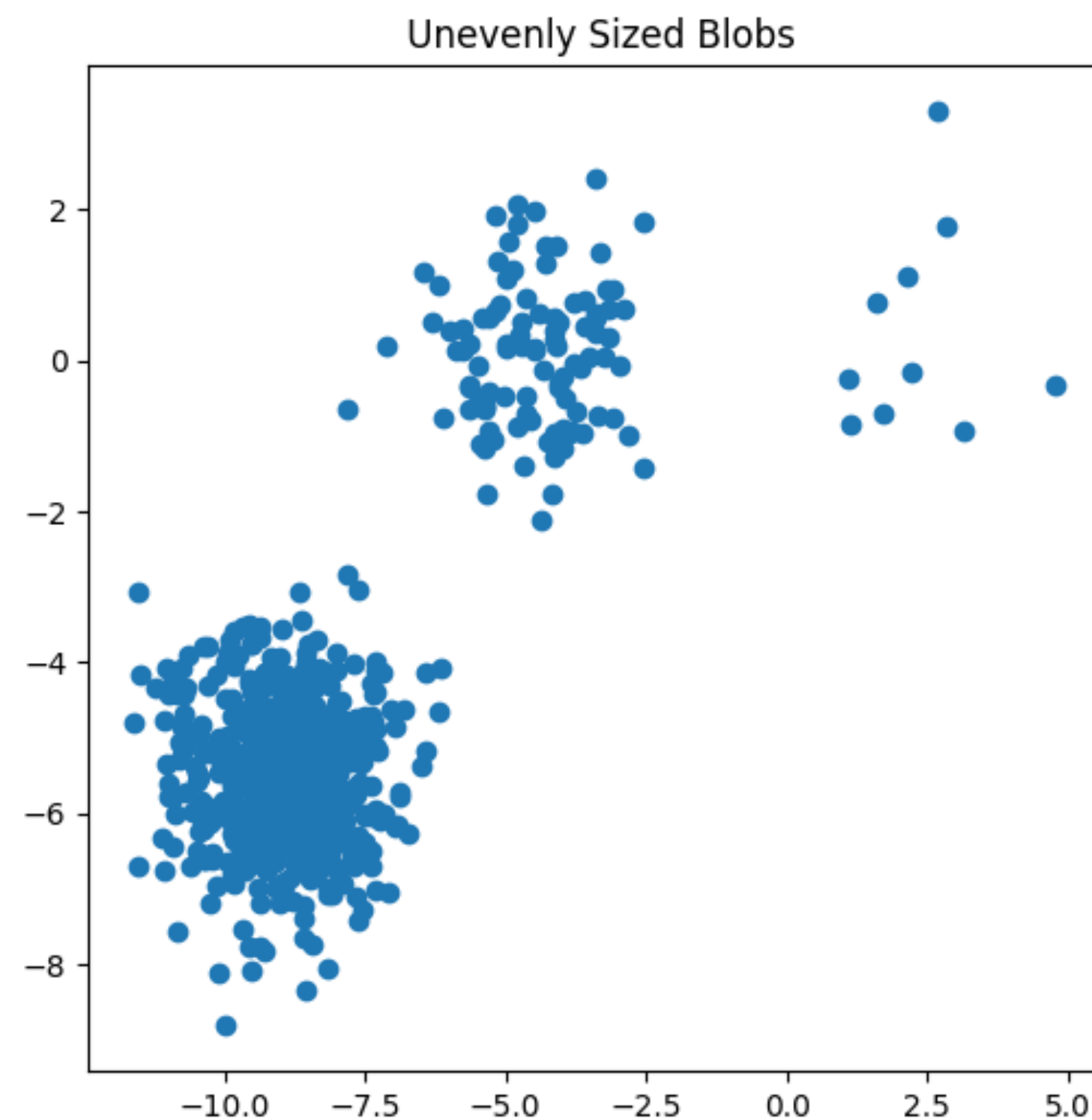
```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++',  
n_init='auto', max_iter=300, tol=0.0001, verbose=0, random_state=None,  
copy_x=True, algorithm='lloyd')
```

[\[source\]](#)

- Random centroid initialization can lead to different results on different runs
- K-means can get stuck in poor local optima, especially with uneven cluster sizes/densities or bad initial centroids.
- Increasing the number of random initializations (*n_init*) reduces this risk by restarting the algorithm several times with different centroids.
- The algorithm then selects and uses the run that results in the lowest WCSS, leading to a more robust and generally better overall clustering result

Exercise 2: K-means - Unevenly sized blobs

<https://drive.google.com/file/d/1jWksZfd9OMuKI0mSYeZbUPFUB9eqjaUg/view?usp=sharing>



- Generate 3 clusters with unevenly sized blobs
- Perform suitable clustering using the scikit-learn library
- Investigate the optimal number of clusters (K) as well as **random centroid initialization**
- Plot the center of the clusters

Exercise 3: K-means - Customer Segmentation

https://drive.google.com/file/d/17-a_sbKHiQ5zLRss_t2jOoONIoS3jhNA/view?usp=sharing

- Exploration of data
- Data Visualization
- Clustering using K-Means
- Selection of Clusters
- Plotting the Clusters and Cluster Boundary
- 3D Plot of Clusters

K-means practical problems

- Practical problems:
 - Sensitive to initialization; multiple random runs often required
 - Choosing the optimal K equires experimentation
 - Struggles with clusters of varying size, variance, density, or presence of outliers
 - May split or merge non-spherical clusters
 - No guarantee of optimal solution quality
 - High spatial complexity; not suitable for large datasets

Gaussian Mixture Model (GMM)

<https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture>

GaussianMixture

```
class sklearn.mixture.GaussianMixture(n_components=1, *,  
covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1,  
init_params='kmeans', weights_init=None, means_init=None,  
precisions_init=None, random_state=None, warm_start=False, verbose=0,  
verbose_interval=10)
```

[\[source\]](#)

- A probabilistic model assuming data is generated from [a mixture of several Gaussian distributions](#)
- Each component (cluster) has its own mean, covariance, and mixing weight
- Can model complex, non-spherical cluster, and suitable for data with overlapping clusters
- Number of components (K) often chosen via Bayesian information criterion (BIC) and Akaike information criterion (AIC)

Gaussian Mixture Model (GMM)

AIC (Akaike Information Criterion)

- Focuses on predictive accuracy
- Formula: $AIC = -2 \ln(L) + 2k$
- Applies a lighter penalty for model complexity
- Often used in machine learning and model tuning
- Lower AIC $\rightarrow$ better model

BIC (Bayesian Information Criterion)

- Focuses on model simplicity and evidence
- Formula: $BIC = -2 \times \ln(L) + k \ln(n)$
- Applies a stronger penalty as sample size increases
- Common in statistical inference and GMM clustering
- Lower BIC $\rightarrow$ better model

L = Likelihood (measures how well the model explains the data)

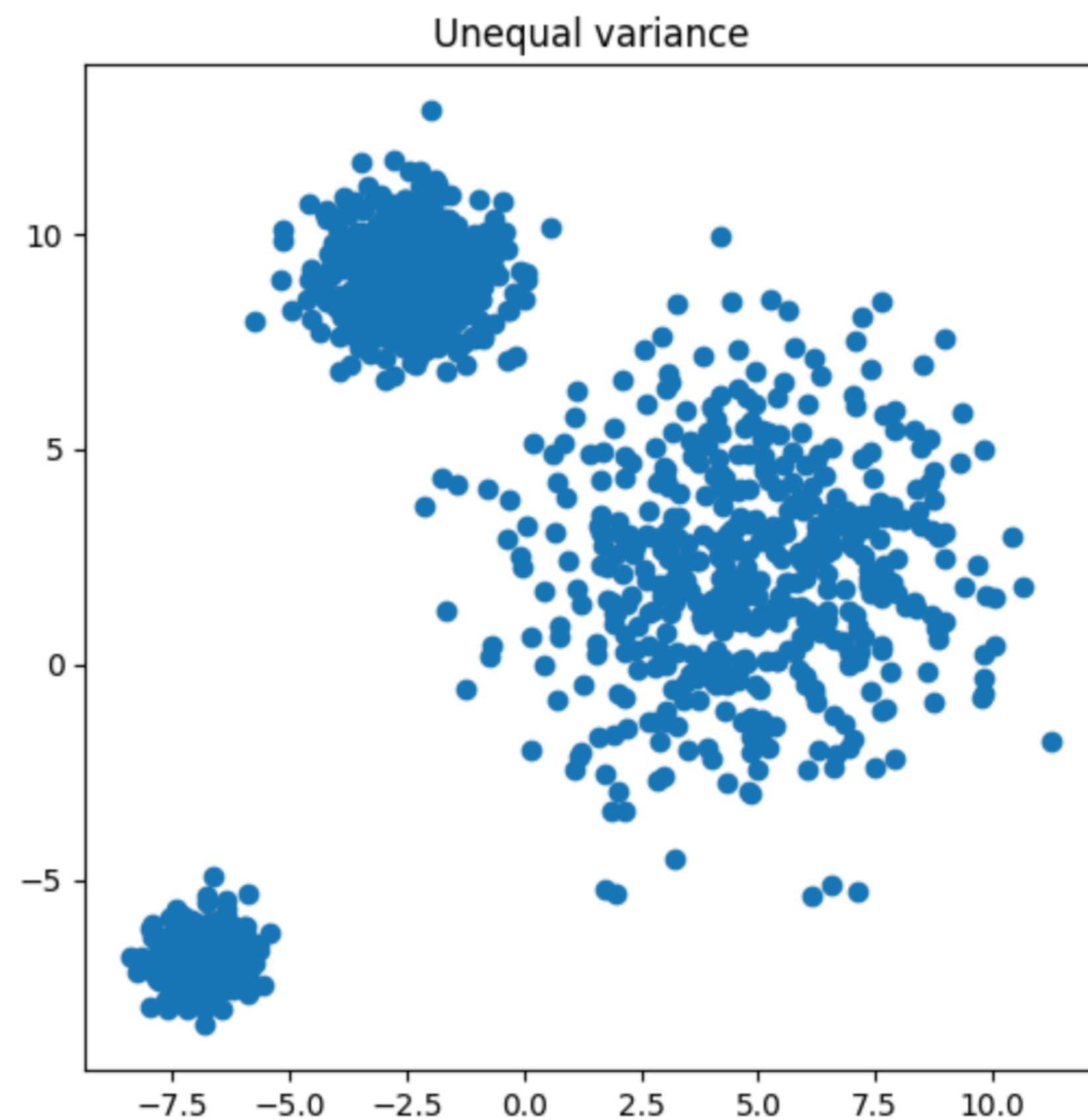
k = the number of parameters

n = the number of observations in the dataset

Calculating the AIC and BIC is straightforward, as both are available as built-in methods in the **Scikit-Learn GaussianMixture** class

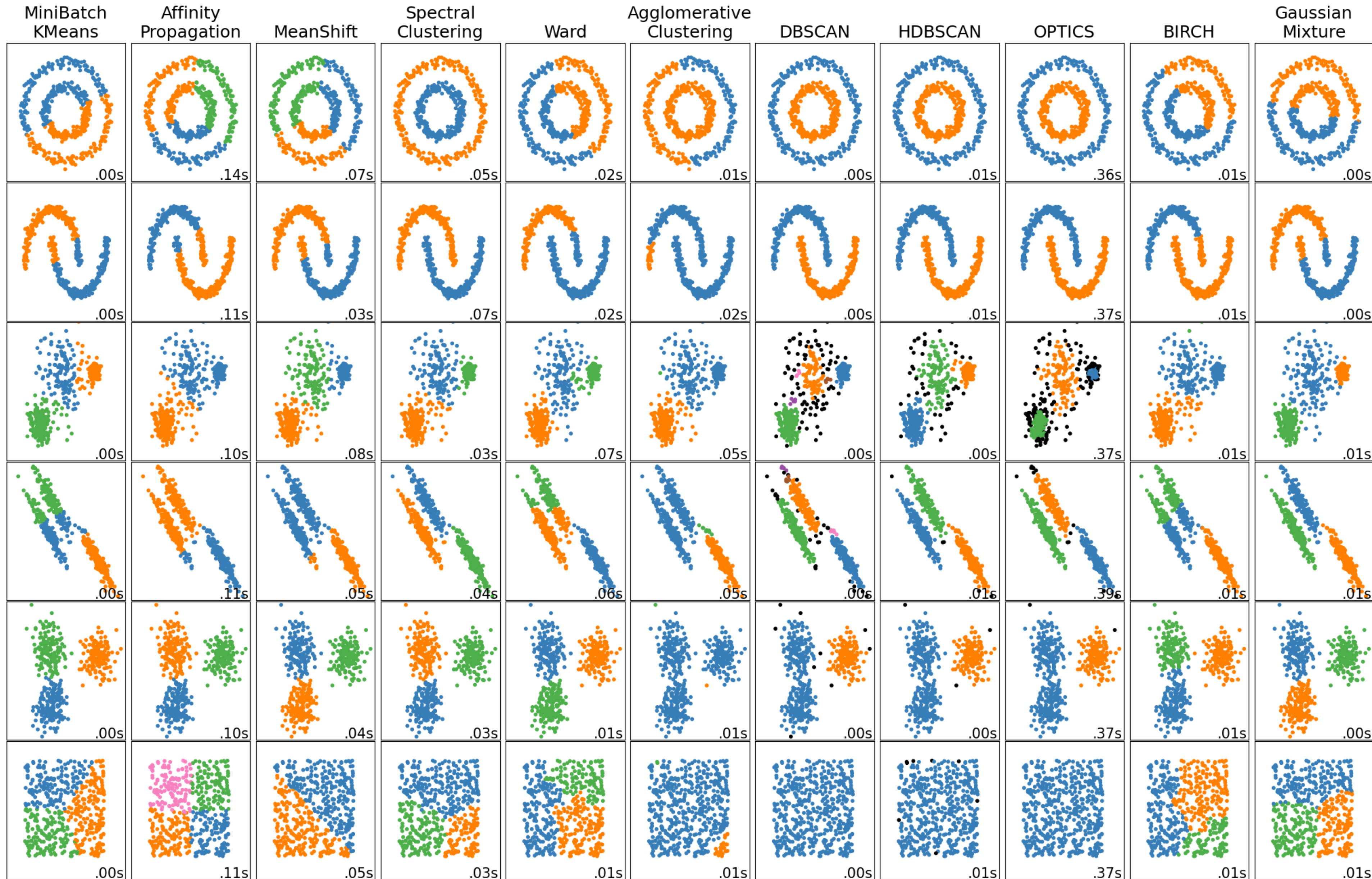
Exercise 4: clusters with unequal variance

<https://drive.google.com/file/d/19zUdfdoGYqG1kdPoFUz4FEbli3amWSc3/view?usp=sharing>



- Compare the clustering results from K-mean and GMM
- Plot the BIC and AIC values together
- Plot the clusters and their center

Comparing different clustering algorithms on toy datasets

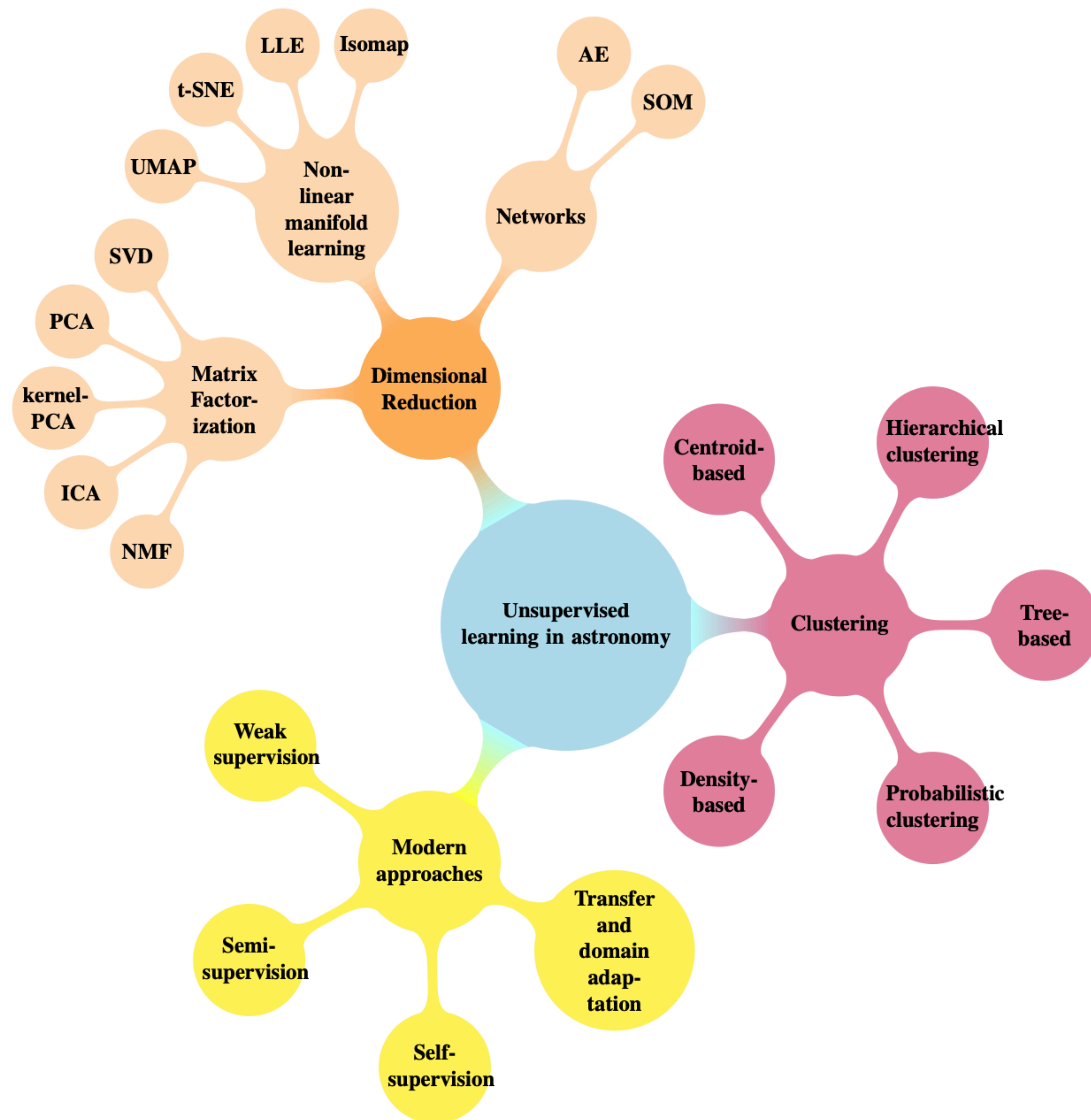


[https://scikit-learn.org/stable/
auto_examples/cluster/
plot_cluster_comparison.html#sphx-glr-
auto-examples-cluster-plot-cluster-
comparison-py](https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py)

Exercise 5: wine dataset

- **Use PCA, t-SNE, and UMAP to explore the number of distinct wine classes**
- **Investigate the statistical properties of features within each cluster**
- **Evaluate feature importance to determine which feature contributes most strongly to the model or clustering structure**

<https://drive.google.com/file/d/1NpISeEZQP-h1xox7H3gu-xjEixCz4dDx/view?usp=sharing>



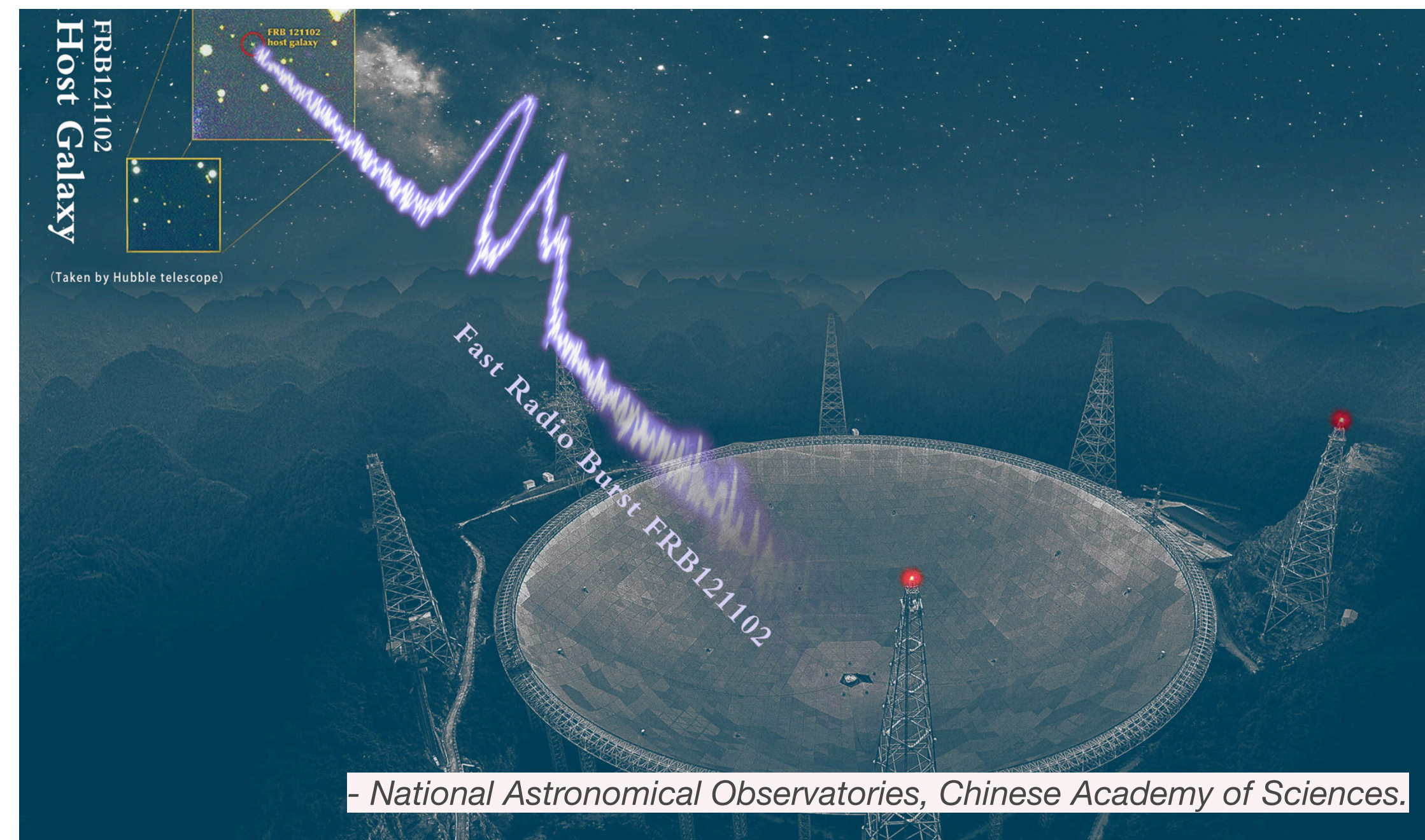
Exercise 6: Fast Radio Burst (FRB) dataset

- Use PCA, t-SNE, and UMAP to explore the number of distinct FRB classes
- Investigate the statistical properties of features within each cluster
- Evaluate feature importance to determine which feature contributes most strongly to the model or clustering structure
- Interpret the results

<https://drive.google.com/file/d/1PZ0fM2JTMVYYMs3ggpZ4zT6tRcMZXbYA/view?usp=sharing>

ไฟล์ข้อมูล FRB ประกอบด้วยคอลัมน์ดังนี้:

- Source: รหัสหรือชื่อที่ใช้ระบุ FRB
- DM_snr: ค่าการกระเจิงของคลื่นวิทยุ (Dispersion Measure, หน่วย $\text{pc}\cdot\text{cm}^{-3}$)
- Flux_density: ความเข้มสูงสุดของพัลส์ในช่วงที่เกิดการปะทุ โดยวัดเป็นหน่วย Jansky (Jy)
- Width: ความกว้างของพัลส์หรือช่วงเวลาที่สัญญาณเกิดขึ้นในแต่ละ FRB โดยวัดในหน่วย มิลลิวินาที (ms)
- Freq_peak: ความถี่ของคลื่นวิทยุที่ความเข้มสูงสุดของพัลส์ (peak frequency) ในหน่วย เมกะเฮิรตซ์ (MHz)
- Fluence: ปริมาณพลังงานรวมที่ได้รับจากพัลส์ในช่วงเวลาสั้น ๆ นั้น วัดในหน่วย $\text{Jy}\cdot\text{ms}$



Semi-supervised Learning

Revealing Hidden Repeaters in the CHIME/FRB Catalog: Semi-Supervised Insights into the Fast Radio Burst Population

N. Mankatwit,¹ P. Thongkongsing,¹ S. Loekkesee,² P. Chainakun,^{1,3}★ W. Luangtip,²† S. Sanpa-arsa⁴‡

¹*School of Physics, Institute of Science, Suranaree University of Technology, 111 University Avenue, Muang, Nakhon Ratchasima, 30000, Thailand*

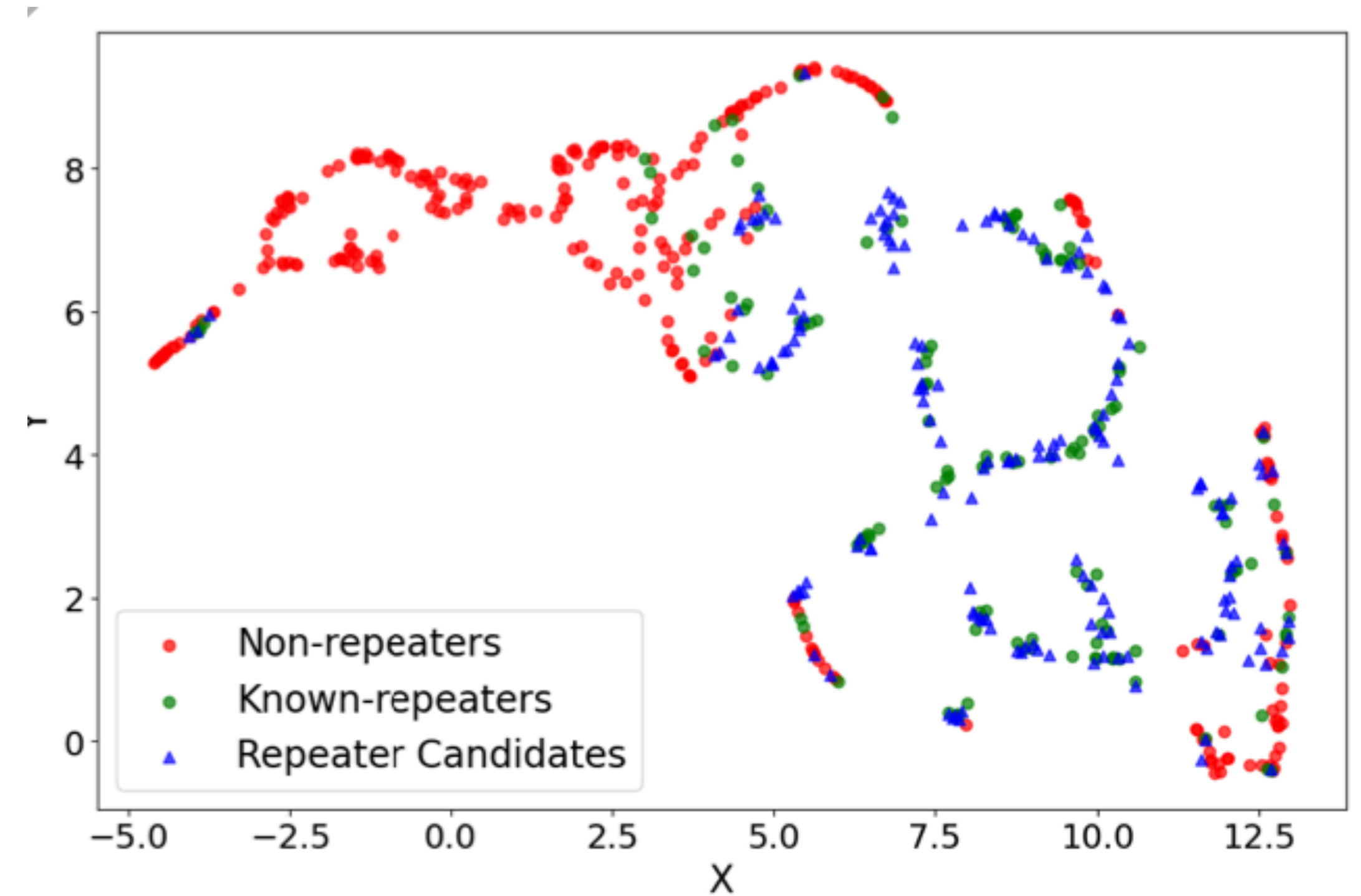
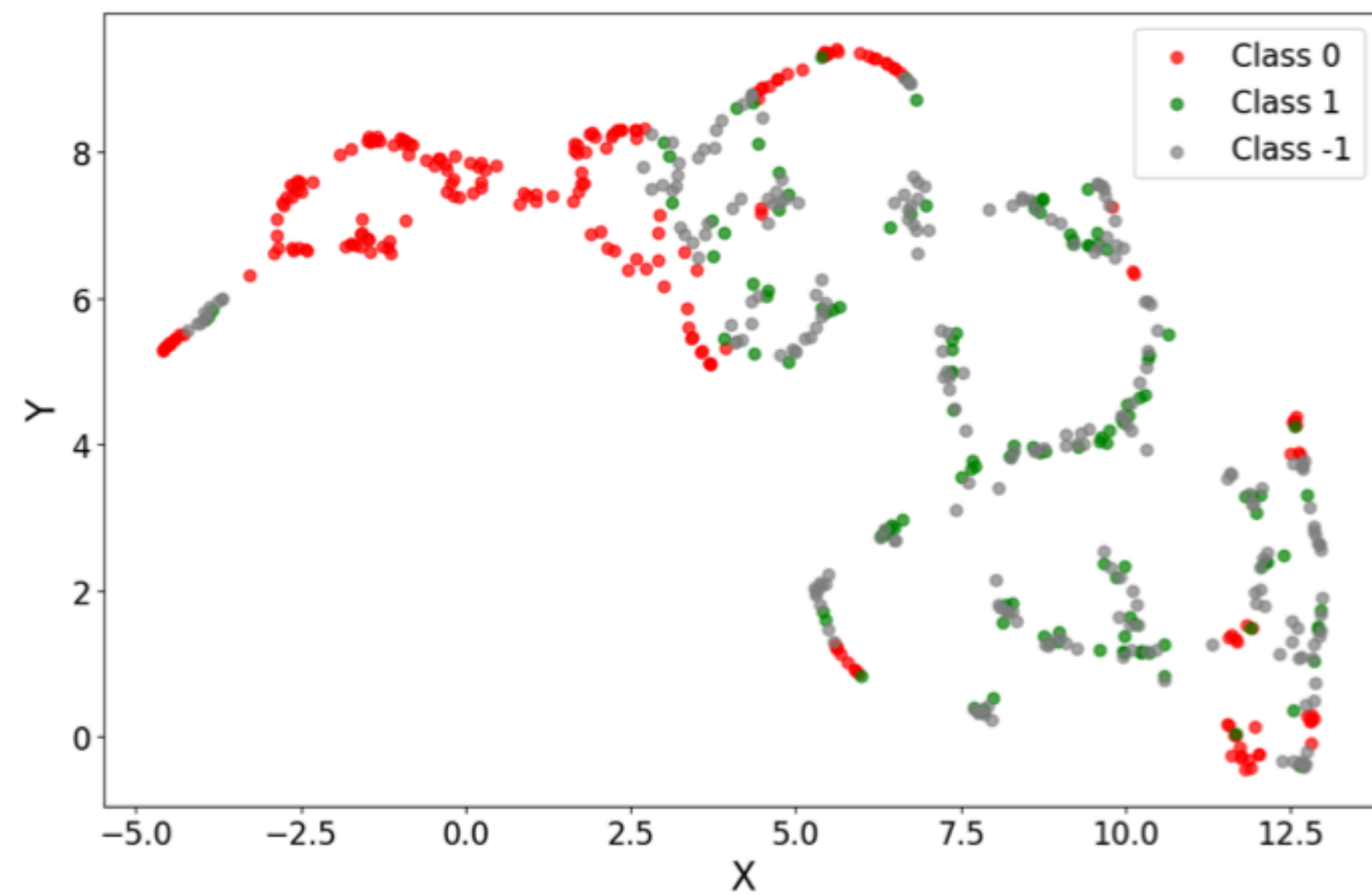
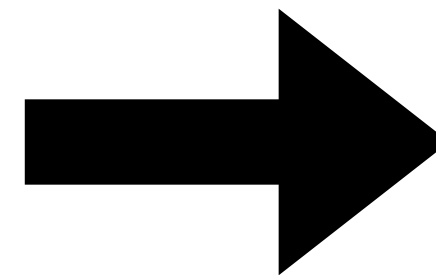
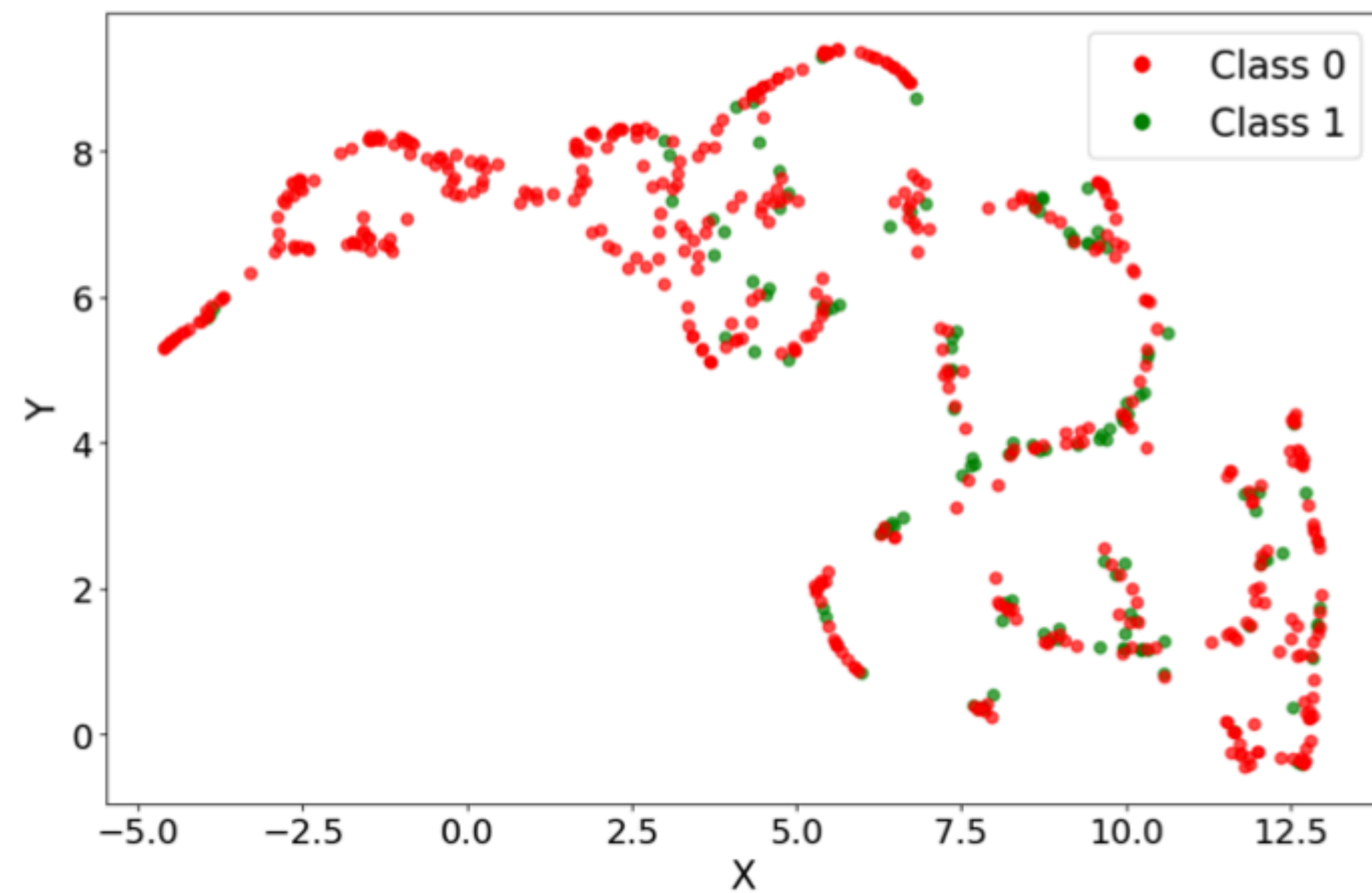
²*Department of Physics, Faculty of Science, Srinakharinwirot University, Bangkok 10110, Thailand*

³*Centre of Excellence in High Energy Physics and Astrophysics, Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand*

⁴*National Astronomical Research Institute of Thailand (Public Organization), 260 Moo 4, Donkaew, Mae Rim, Chiang Mai, 50180, Thailand*

Class 0 = non-repeater

Class 1 = repeater



- Mankatwit et al., MNRAS (accepted, in press)

Credit for the data sources and example codes:

- <https://www.geeksforgeeks.org/machine-learning/implementing-pca-in-python-with-scikit-lear/>
- https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html#sphx-glr-auto-examples-cluster-plot-kmeans-assumptions-py
- <https://www.kaggle.com/code/kushal1996/customer-segmentation-k-means-analysis>
- <https://towardsdatascience.com/gaussian-mixture-models-for-clustering-3f62d0da675/>
- <https://www.kdnuggets.com/2023/05/principal-component-analysis-pca-scikitlearn.html>