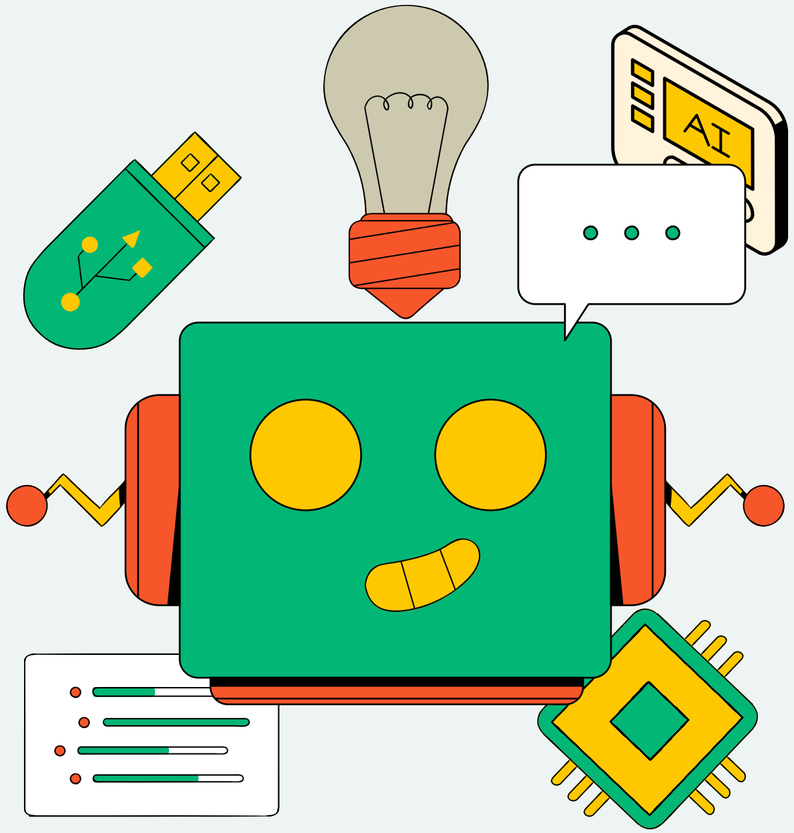
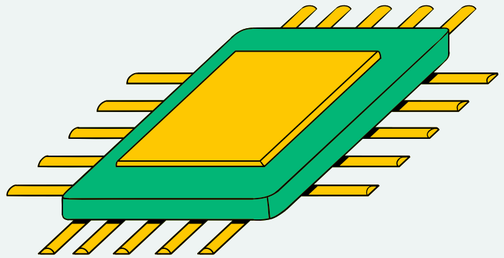


UNSUOERVISED LEARNING & ASSIGNMENT 3



OUTLINE

- Unsupervised learning theory recap
- Oblig 2 help tips and help

DEFINITION UNSUPERVISED LEARNING

Unsupervised learning is a type of machine learning technique **where the algorithm explores the data and identifies hidden structures or relationships within it without being given specific instructions on what to look for.**

Commonly used for tasks such as clustering, dimensionality reduction, and anomaly detection, **where the goal is to understand the inherent structure of the data.**

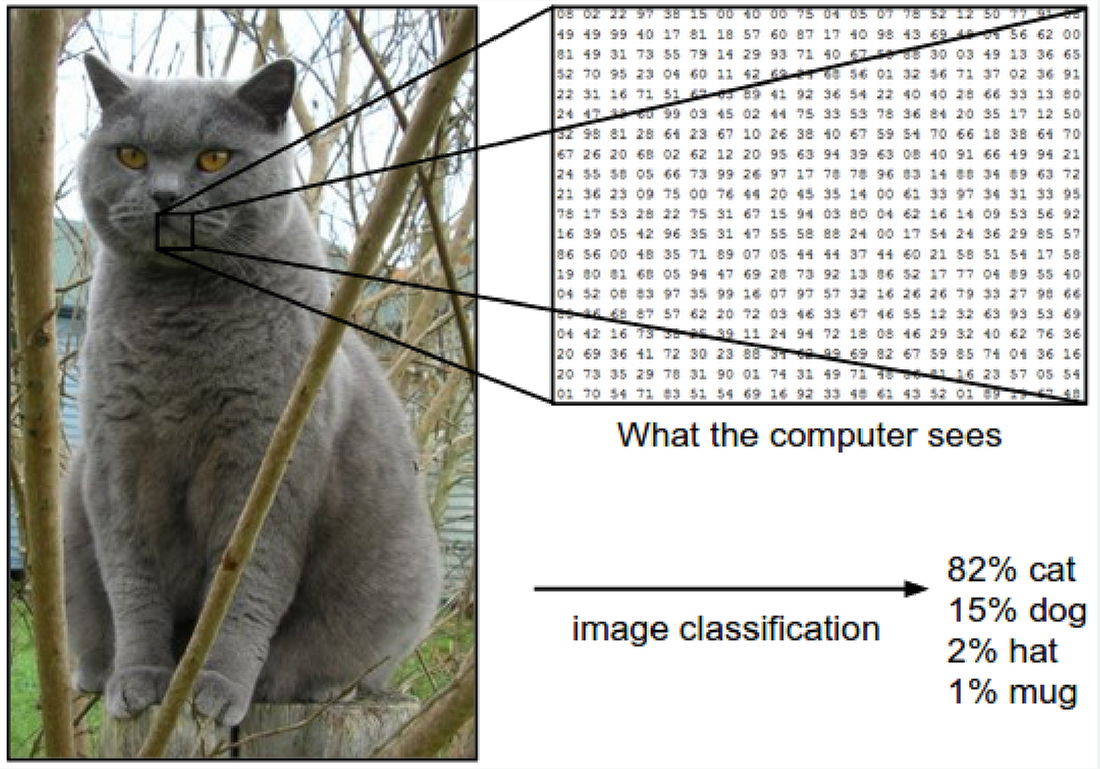
Unlike supervised learning, there are no predefined target variables, and the algorithm must infer the underlying structure of the data on its own.

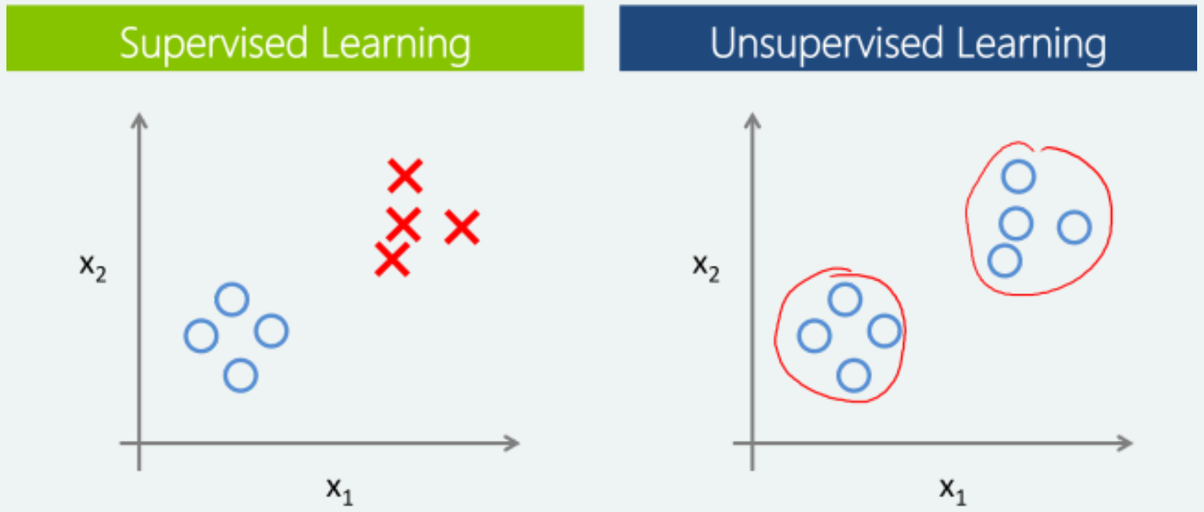
DIFFERENCE BETWEEN SUPERVISED AND UNSUPERVISED LEARNING

Aspect	Supervised Learning	Unsupervised Learning
Training Data	Labeled input-output pairs	Unlabeled data
Goal	Learn a mapping from inputs to outputs	Discover hidden patterns or structures
Example Applications	Image classification, regression, classification	Clustering, dimensionality reduction, anomaly detection
Evaluation	Performance metrics based on predicted outputs	Metrics based on discovered patterns or clusters
Feedback	Direct feedback based on labeled data	Indirect feedback based on data similarity or distribution

DENSITY

We can represent each picture as a point in a 3D space, where the three dimensions represent the amount of red, green, and blue in the picture. So, if a picture has a lot of red and a little bit of green and blue, it would be represented by a point that's mostly towards the red side in our 3D space.

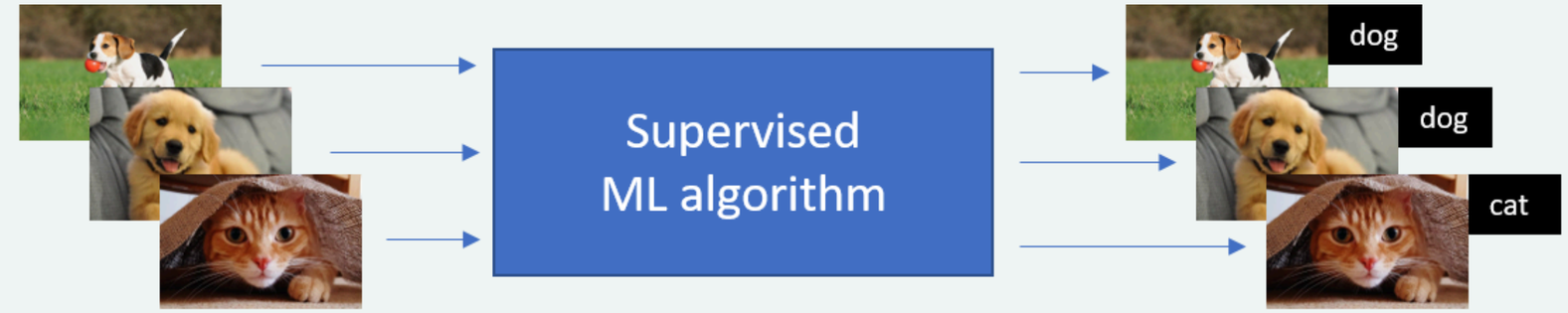


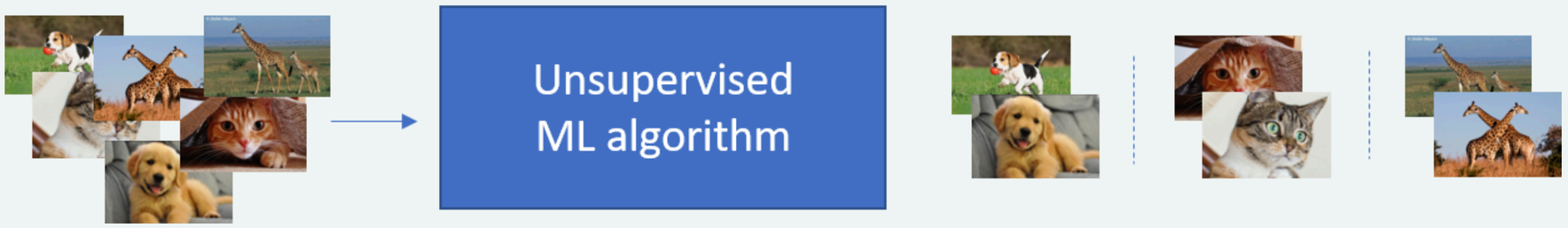


DENSITY

Chances are, the cat pictures will be kind of similar to each other in terms of their colors, and the dog pictures will be kind of similar to each other too. So, the points representing cat pictures will be closer to each other, and the points representing dog pictures will also be closer to each other.

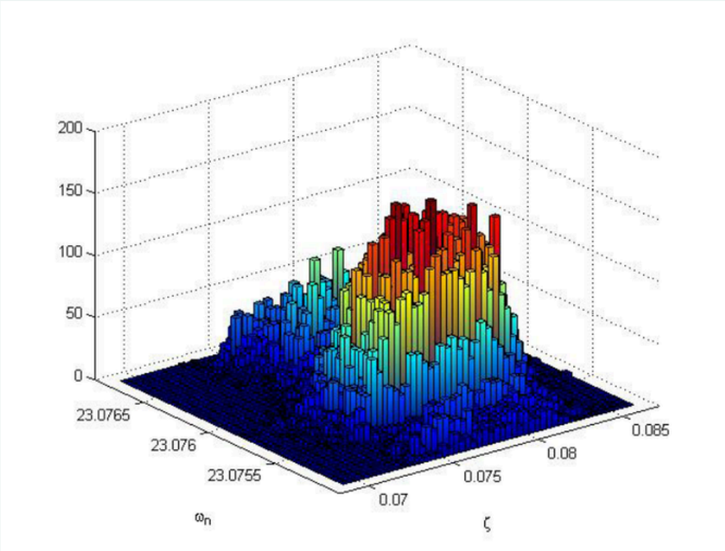
That's the basic idea of how similar features can be located near each other. We look at one aspect of the pictures—in this case, color—and represent each picture as a point based on that aspect. Then, we can see that pictures with similar colors (like all the cat pictures or all the dog pictures) end up being located near each other in our 3D space.



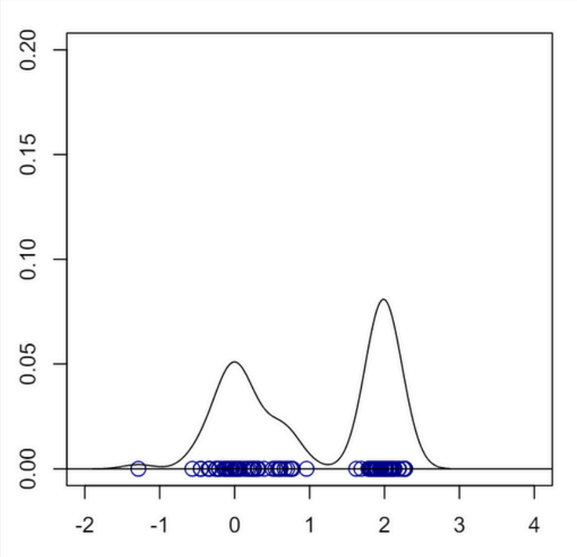


DENSITY FUNCTIONS

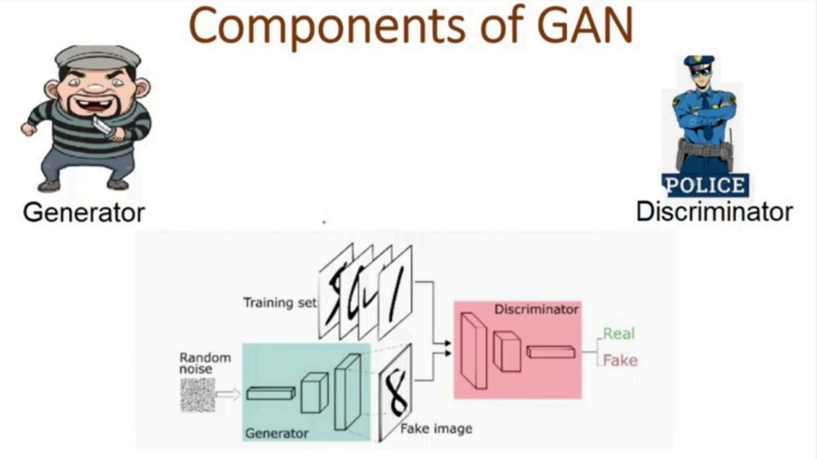
Histogram



Smooth density estimator



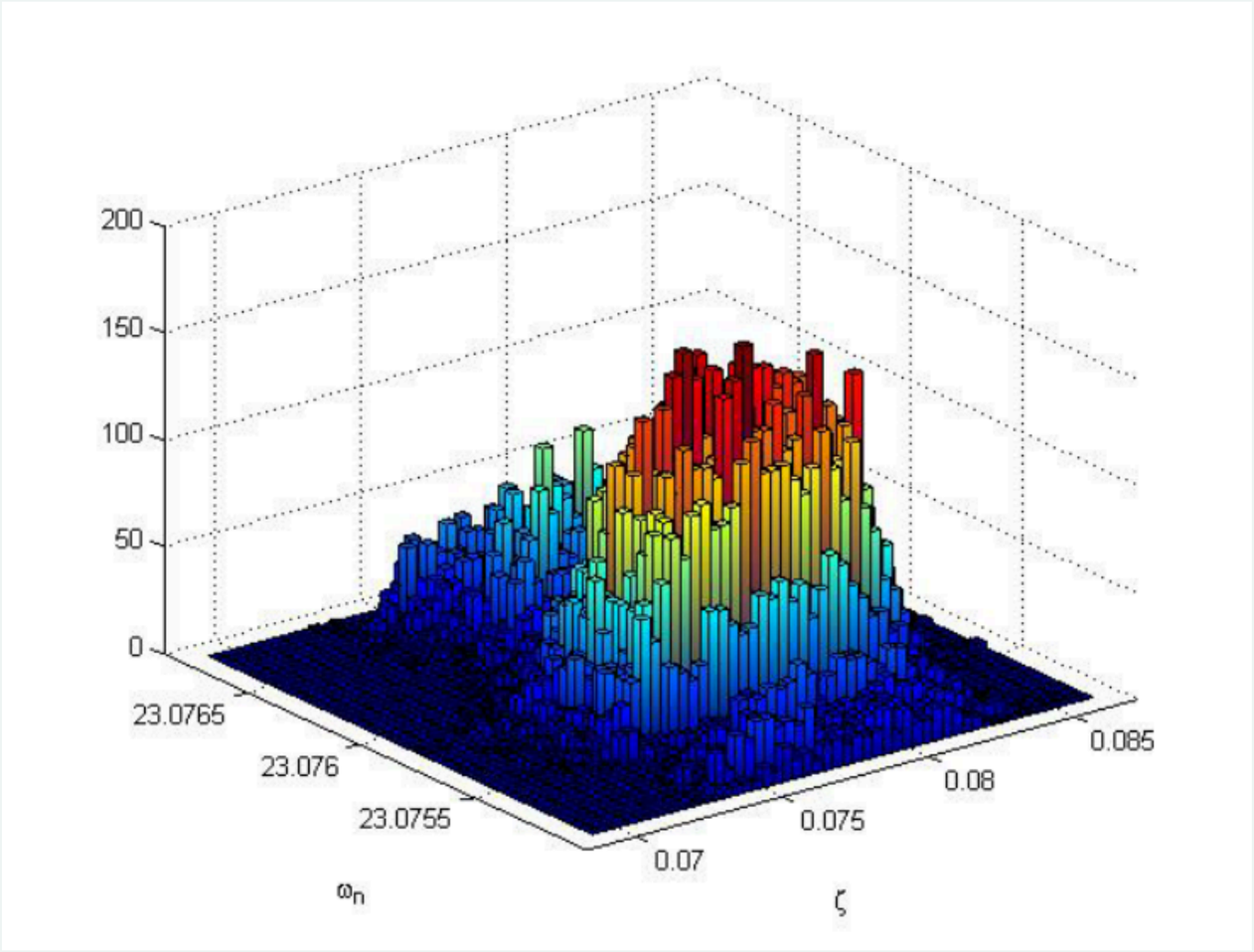
generative adversarial networks (GANs)



DENSITY FUNCTIONS

Histogram

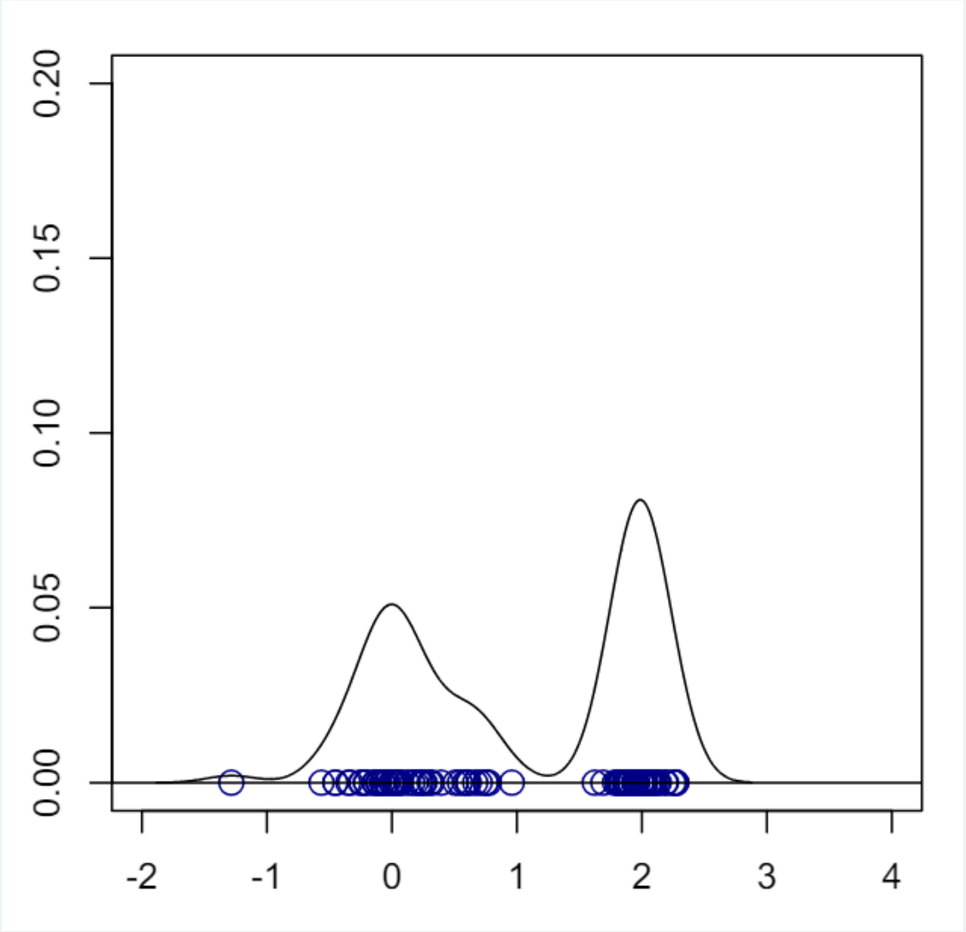
Consists of a series of adjacent bars, where the height of each bar represents the frequency or count of data points falling within a specified range or bin. Essentially, it shows how data is spread out over different intervals or categories.



DENSITY FUNCTIONS

Smooth density estimator

Kernel Density Estimation (KDE) is a statistical method used to estimate the probability density function (PDF) of a dataset by placing a kernel function at each data point and summing up their contributions to create a smooth density estimate.



HOW TO TRAIN KERNEL SMOOTH DENCITY ESTIMATOR

Training data: $x_1, \dots, x_n \in \mathbf{R}$ Test data: x

- Select a kernel function $K(x)$
  - A kernel is a "0-detector" with high positive value for x close to 0 and gradually lower values for x far from 0

- Compute $K(\frac{x-x_i}{h})$ for $i = 1,2, \dots, n.$
  - This detects if x is close to any of the points in the training data set, and h>0 controls the detection sensitivity

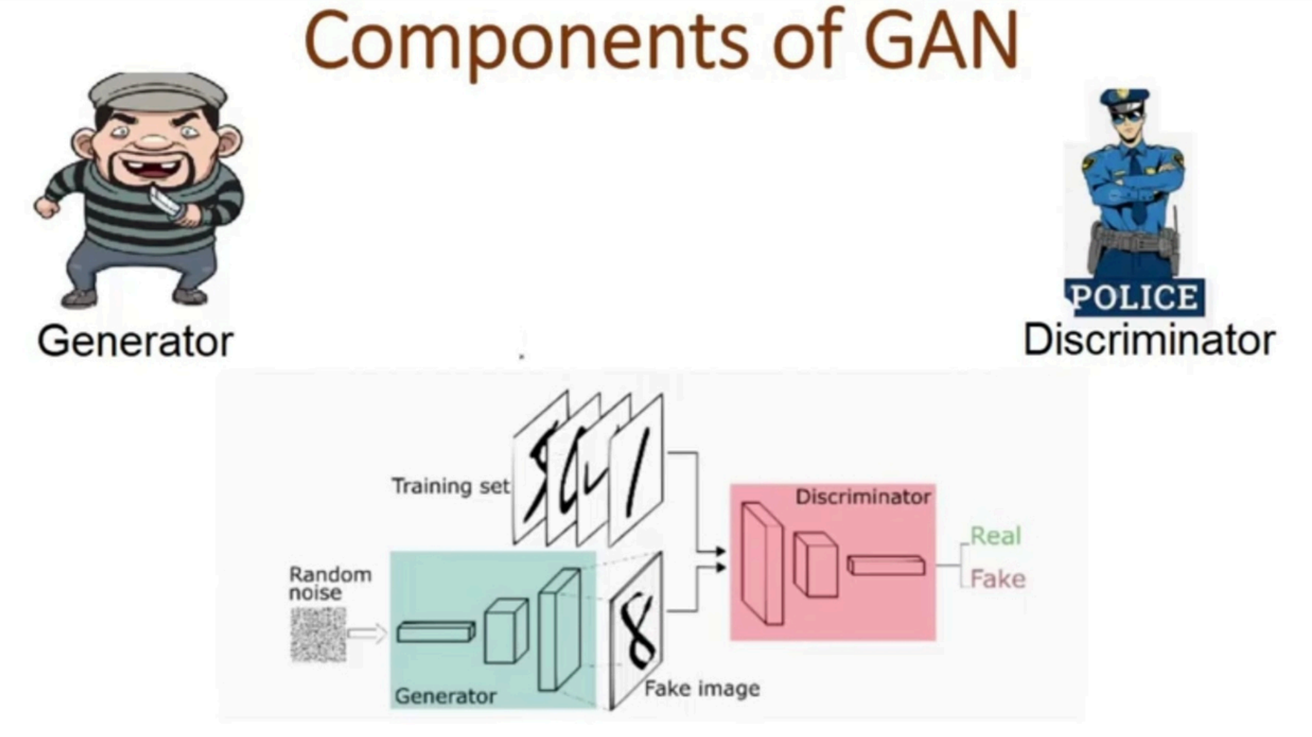
- Let $f(x) = \frac{1}{nh}\sum_i K(\frac{x-x_i}{h})$
  - This adds together all detector values, so $f(x)$ is higher the more training data points there are in neighborhood of x

(Page 25 of lecture 11, 2024)

DENSITY FUNCTIONS

generative adversarial networks (GANs)

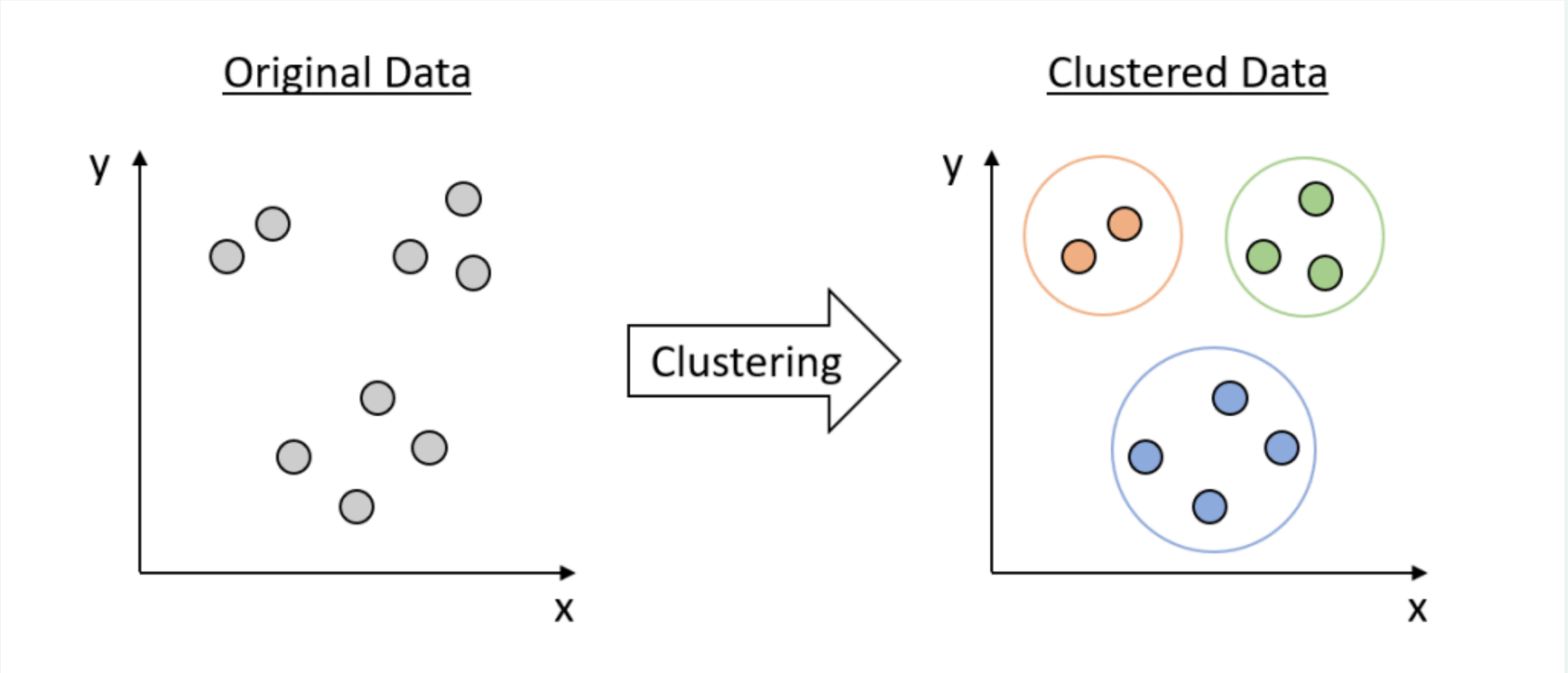
Consists of two neural networks: a generator and a discriminator. The generator learns to create realistic data samples, while the discriminator learns to distinguish between real and fake data. Through this adversarial process, both networks improve until the generator can produce data samples that are indistinguishable from real ones.



CLUSTERING

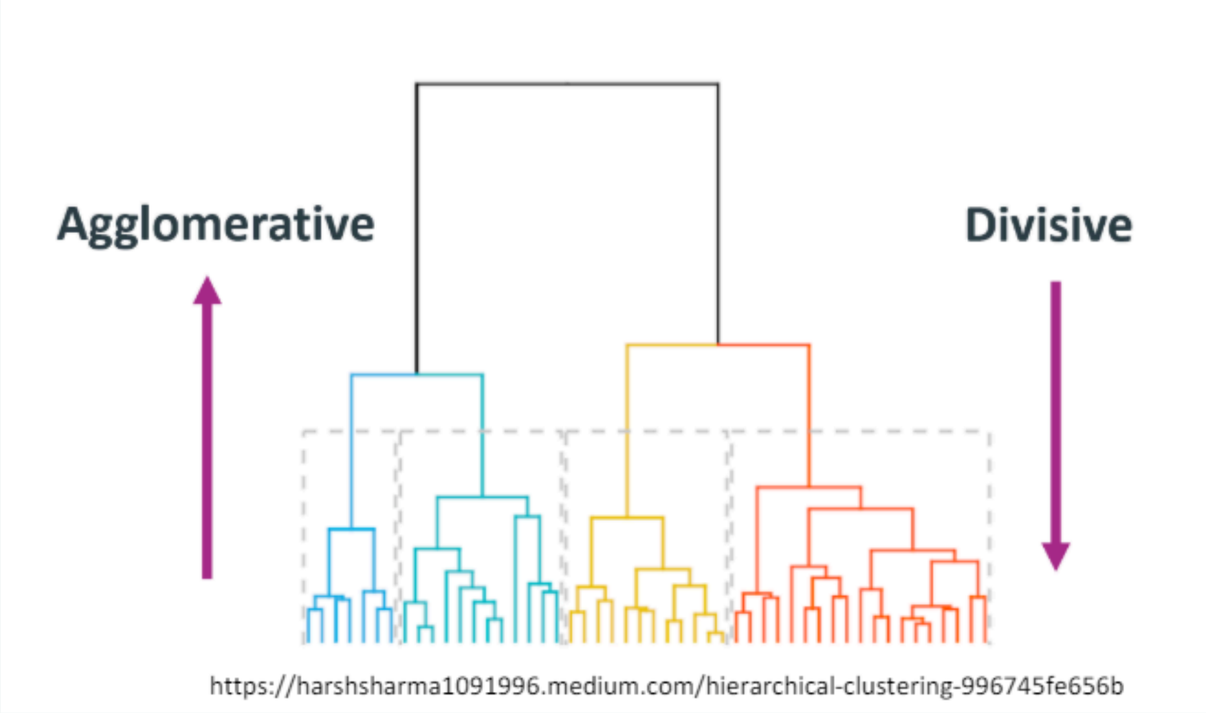
technique in unsupervised learning where the goal is to group similar objects or data points together into clusters, based on their intrinsic characteristics or features.

The main objective of clustering is to partition a dataset into subsets, or clusters, such that data points within the same cluster are more similar to each other than to those in other clusters.



HIERARCHICAL CLUSTERING

Hierarchical clustering is a clustering algorithm that creates a hierarchy of clusters. Unlike other clustering algorithms like k-means, hierarchical clustering does not require the number of clusters to be specified in advance. Instead, it creates a tree-like structure, called a **dendrogram**, where each node represents a cluster.



LINKAGE

Linkage refers to the criterion used to measure the distance or similarity between clusters when deciding which clusters to merge at each step of the algorithm.

TYPES OF LINKAGE IN CLUSTERING

SINGLE LINKAGE

The distance between two clusters corresponds to the minimum distance between two points in each cluster:

$$D(c_1, c_2) = \min D(x_1, x_2)$$

COMPLETE LINKAGE

The distance between two clusters corresponds to the maximum distance between two points in each cluster:

$$D(c_1, c_2) = \max D(x_1, x_2)$$

AVERAGE LINKAGE

The distance between two clusters corresponds to the mean distance of all pairs from each cluster:

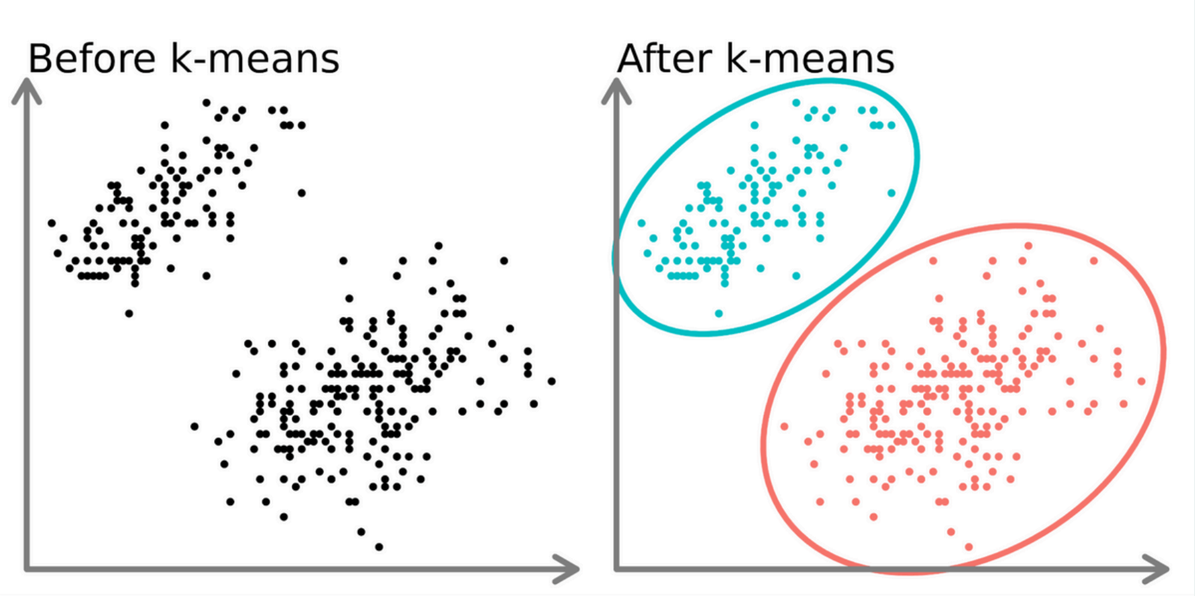
$$D(c_1, c_2) = \frac{1}{|c_1|}\frac{1}{|c_2|}\sum\sum D(x_1, x_2)$$

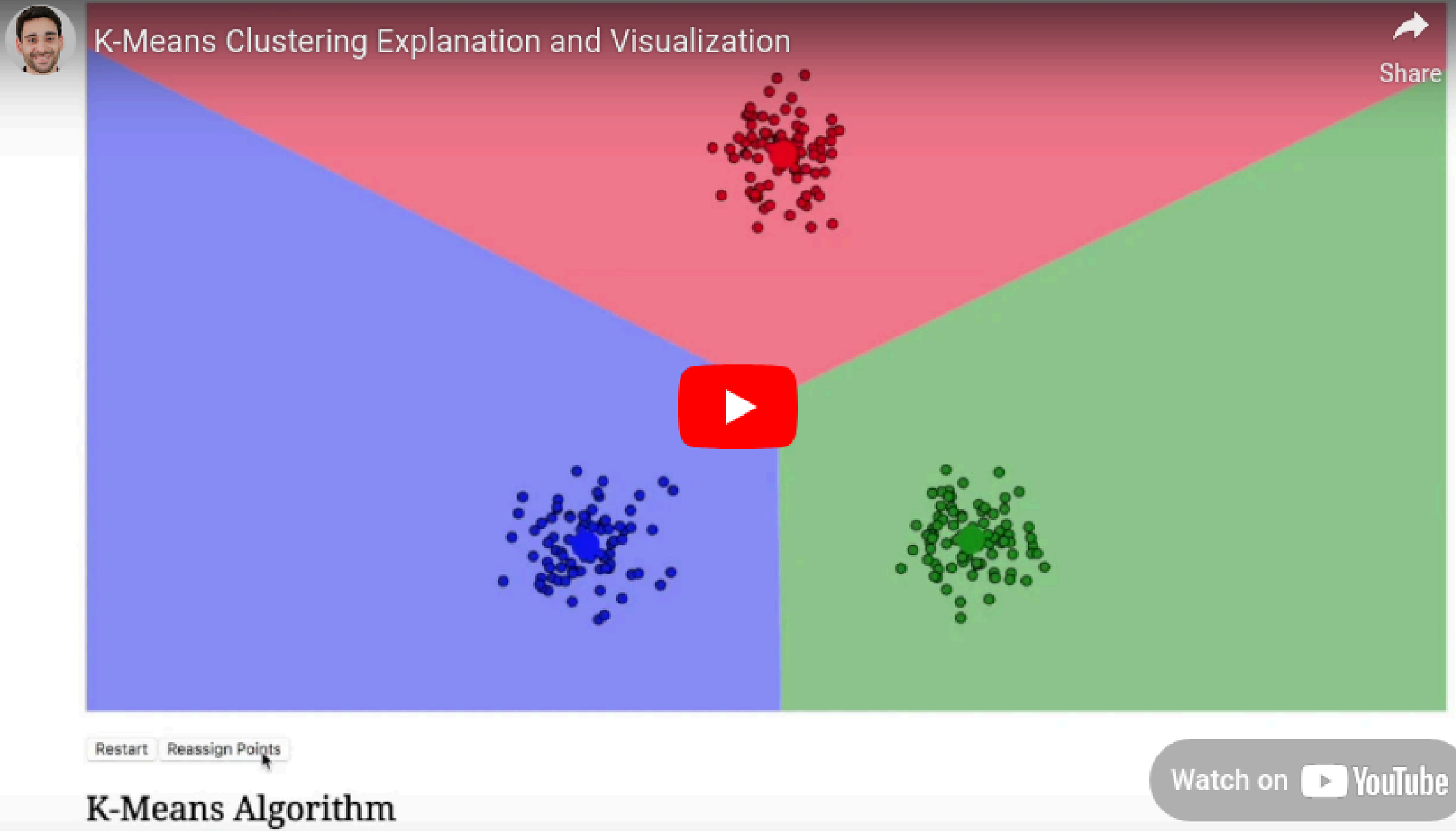
WARD'S LINKAGE

The distance between two clusters corresponds to the increase in the sum of squares after the clusters have been merged. The goal is to minimize the total within cluster variance

K MEANS CLUSTEERING

The algorithm works by iteratively assigning data points to clusters and updating the cluster centroids until convergence. The goal of k-means clustering is to minimize the variance within clusters, meaning that data points within the same cluster are as similar to each other as possible, while data points in different clusters are as dissimilar as possible.




K-Means Clustering Explanation and Visualization
Share
Restart
Reassign Points
K-Means Algorithm
Watch on YouTube

K MEANS CLUSTEERING ALGORITHM

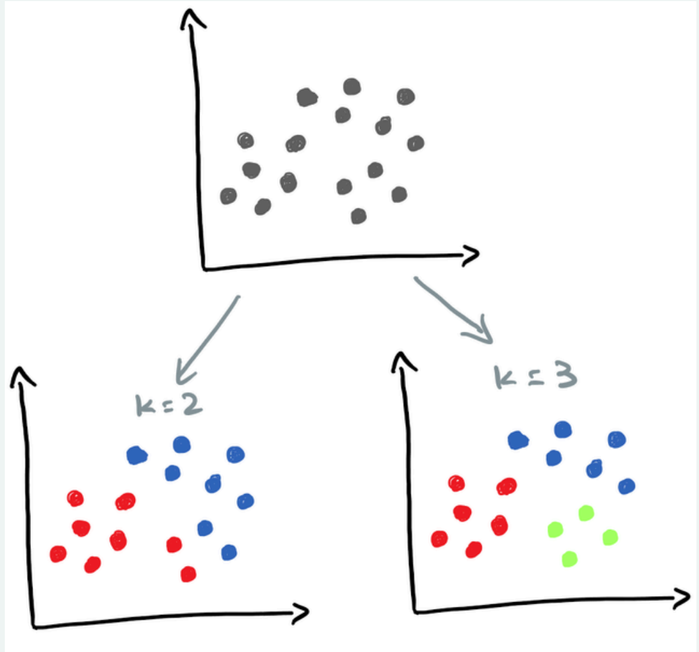
k-means clustering

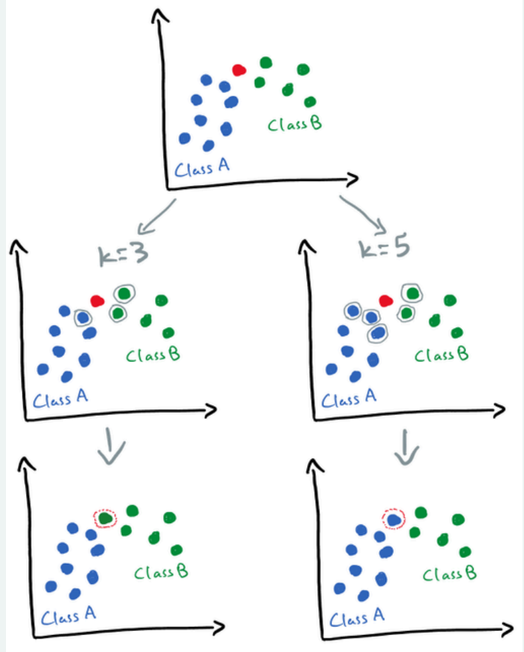
As before we wish to cluster samples $x_1, \ldots, x_n \in \mathbf{R}^p$

1) Randomly select k cluster centers $c_1, \ldots, c_k \in \mathbf{R}^p$
2) Assign each sample to the nearest cluster center
3) Update cluster centers to mean of samples in cluster
4) Repeat steps 2-3 until convergence

DIFFERENCE BETWEEN K MEAN AND K-NEAREST NEIGHBOURS

K-means Clustering	K-nearest Neighbors (KNN)
Unsupervised	Supervised
Partition data into k clusters	Predict the class of a new data point
Dataset with features only	Dataset with features and corresponding labels





More about K means and KNN;

https://ealizadeh.com/blog/knn-and-kmeans/#:~:text=KNN%20is%20a%20supervised%20learning%20algorithm%20mainly%20used%20for%20classification,on%20the%20chosen%20distance%20metric).

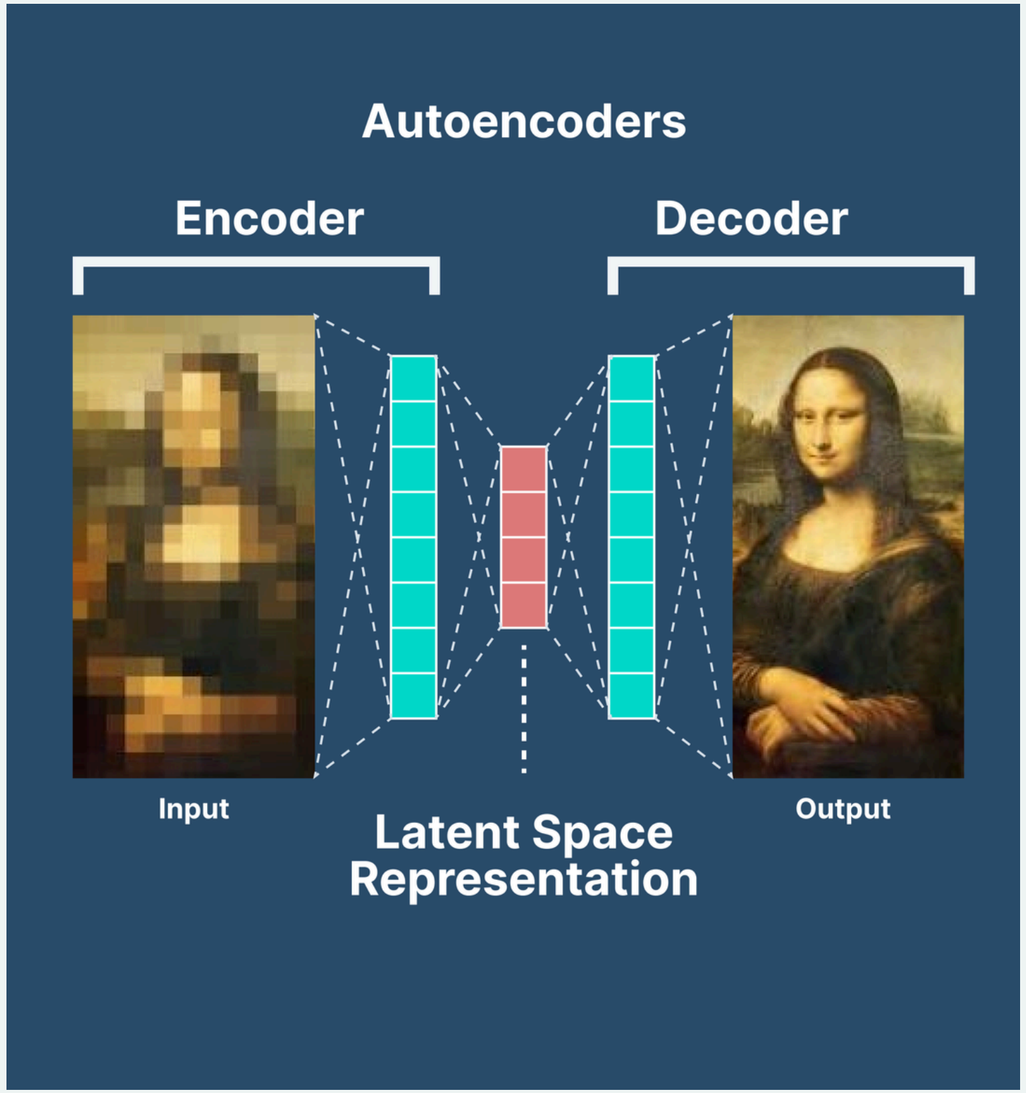
DIFFERENCE BETWEEN K MEAN AND K-NEAREST NEIGHBOURS

- Choose the number of clusters, k, and randomly initialize k cluster centroids.
- For each data point, calculate the distance to each cluster centroid.
- Assign the data point to the cluster whose centroid is closest to it.
- Update each cluster centroid to be the mean of all data points assigned to that cluster.
- Repeat the assignment and update steps iteratively until convergence or until a specified number of iterations is reached.
- Once the algorithm converges, the final cluster assignments are obtained.

- Choose the number of neighbors k and a distance metric.
- Store the training dataset with features and corresponding labels.
- For each new data point:
- Calculate the distance to all data points in the training dataset using the chosen distance metric.
- Select the k nearest neighbors based on the calculated distances.
- Assign the majority class label among the k nearest neighbors to the new data point.
- If regression, predict the average of the k nearest neighbors' labels.

AUTOENCODERS

Autoencoders are a type of artificial neural network used for unsupervised learning. They aim to learn a compressed, or encoded, representation of input data without supervision. The network learns to encode the input data into a lower-dimensional space and then decode it back to its original form.



TYPES OF AUTOENCODERS

Sparse autoencoders: trained to obtain a sparse hidden layer representation (= only a few hidden nodes active at a time)

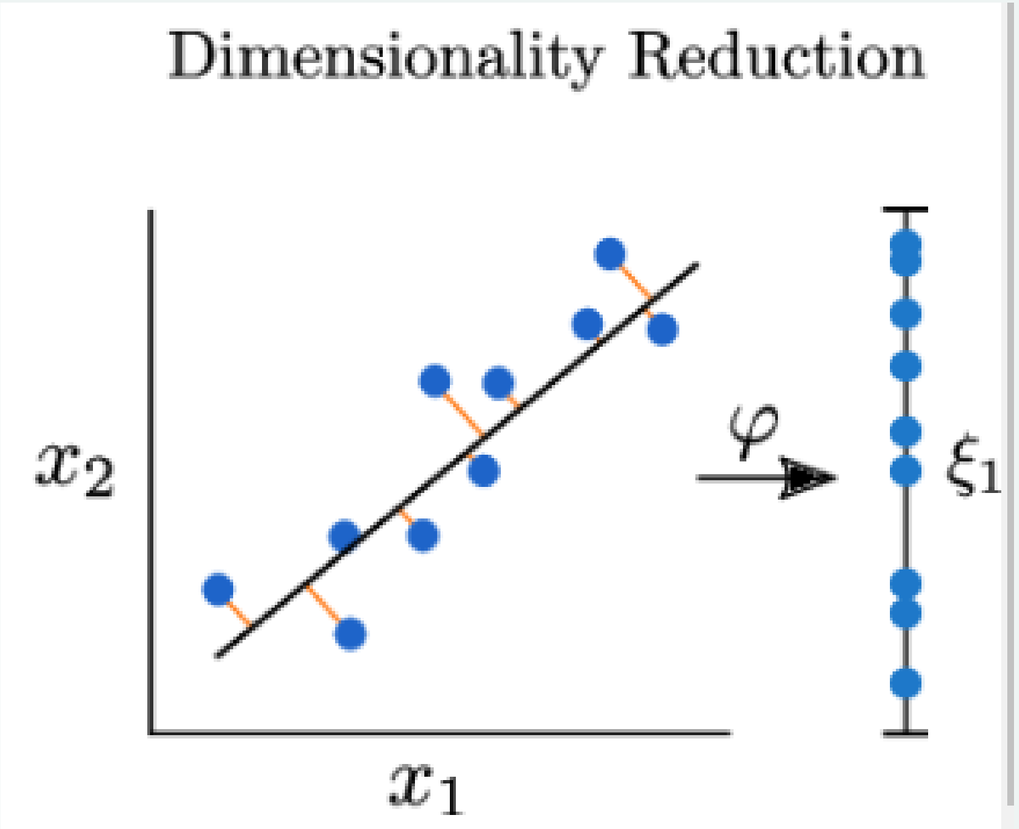
Denoising autoencoders: trained to recapture a denoised version of the input vector

Contractive autoencoders: trained to obtain a hidden layer representation that is robust to small changes in the input

Variational autoencoders: trained to obtain a representation useful also for generating new data from same distribution

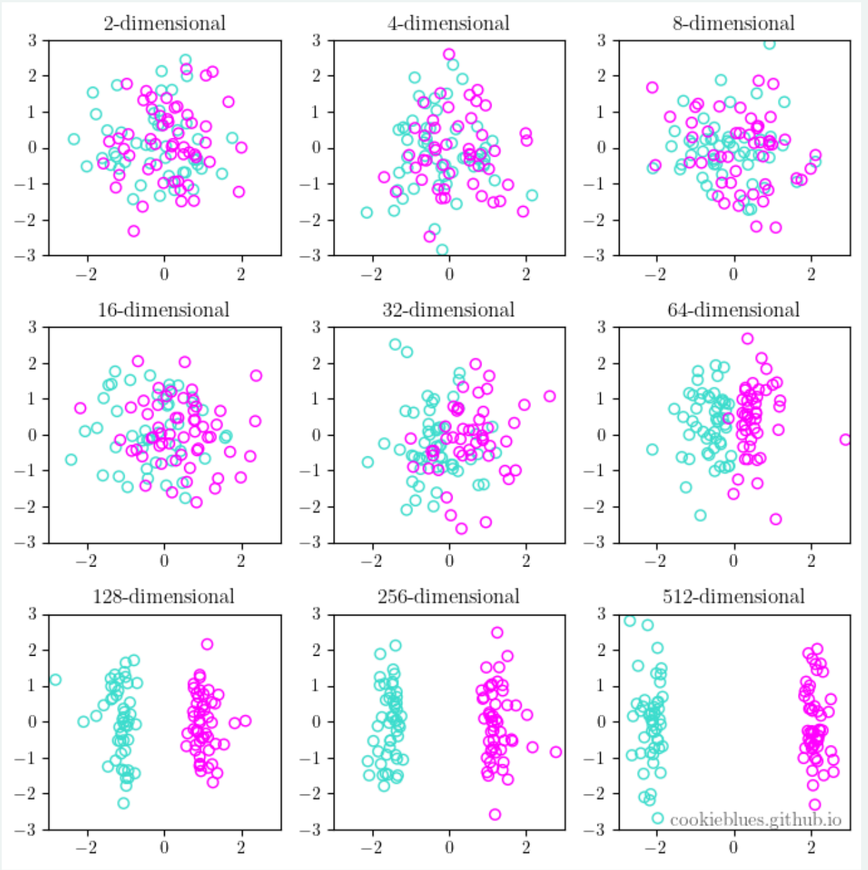
DIMENSIONALITY REDUCTION

Dimensionality reduction is a technique used to reduce the number of input variables or features in a dataset while preserving as much relevant information as possible. It aims to simplify the dataset by transforming it into a lower-dimensional space, making it more manageable and easier to analyze.



CURSE OF DIMENSIONALITY

The curse of dimensionality refers to various problems that arise when working with high-dimensional data, such as increased computational complexity, sparse data distributions, and overfitting in machine learning models.



More about curse of dimensionality:

https://towardsdatascience.com/the-curse-of-dimensionality-5673118fe6d2

DIMENSIONALITY REDUCTION TECHNIQUES

- Principal Component Analysis (PCA)
- t-Distributed Stochastic Neighbour Embedding (t-SNE)
- Linear Discriminant Analysis (LDA)
- Isomap
- Autoencoders
- Random Projection
- Sparse Coding

PCA

Principal Component Analysis (PCA) is a linear dimensionality reduction technique widely used for feature extraction and data visualization. It aims to transform high-dimensional data into a lower-dimensional space while preserving as much variance as possible.

1) Start with a data set:

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{bmatrix} \begin{matrix} \text{sample 1} \\ \text{sample 2} \\ \text{.....} \\ \text{sample n} \end{matrix}$$

2) Center the data:

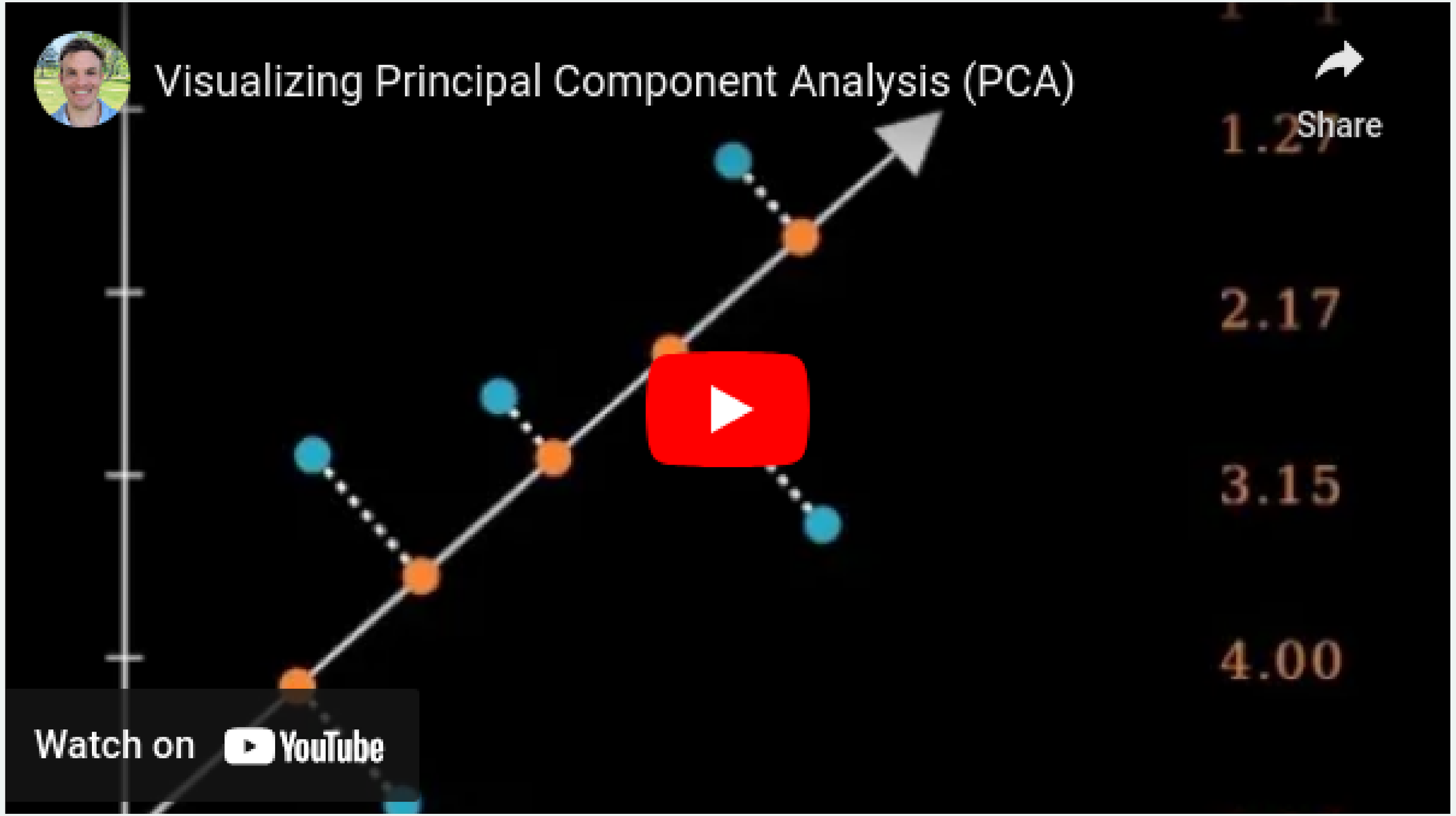
$$Y = \begin{bmatrix} x_{11} - m_1 & x_{12} - m_2 \\ x_{21} - m_1 & x_{22} - m_2 \\ \vdots & \vdots \\ x_{n1} - m_1 & x_{n2} - m_2 \end{bmatrix}$$

3) Find weights $\boldsymbol{w} = (w_1, w_2)$ maximizing length of $Y\boldsymbol{w}$:

$$\widehat{\boldsymbol{w}} = \underset{\|\boldsymbol{w}\|=1}{\operatorname{argmax}} \|Y\boldsymbol{w}\|$$

That's it! The vector $\widehat{\boldsymbol{w}}$ is called the 1st principal component of X.

PCA


Visualizing Principal Component Analysis (PCA)
Share
1.27
2.17
3.15
4.00
Watch on YouTube

EIGENVECTORS AND EIGENVALUES

Eigenvectors:

Imagine you have a dataset with multiple features, like height, weight, and age. An eigenvector represents a direction in this feature space. It's a special direction where, when you look at your data from that direction, it seems to change only in size, not in shape. In PCA, these directions are called principal components, and they help us understand the main patterns in our data.

Eigenvalues:

Now, imagine that each eigenvector has a number attached to it, called an eigenvalue. This number tells you how much the data spreads out along that direction (eigenvector). The bigger the eigenvalue, the more spread out the data is along that direction. So, eigenvalues help us decide which directions (or principal components) are the most important in describing our data's variability.

Detailed explanation of the concepts: https://www.youtube.com/watch?v=PFDu9oVAE-g

LOGISTIC REGRESSION FOR MULTICLASS

Approach 1

Just build five different classifiers with different names. For prediction compute the probabilty each classifier ascribes to the corresponding class and choose the class with the highest probability.

Approach 2

The principle is the same, but we build a wrapper around the five classfiers. The wrapper is a new classifier which constructs and fits the five other classifiers.

Копировать код

```
Class OvRWrapper inherits from NumpyClassifier:

    Method __init__(base_classifier defaults to NumpyLogReg, bias defaults to -1):
        Set the classifier attribute to base_classifier
        Set the bias attribute to bias

    Method fit(X_train, t_train, **kwargs):
        Get unique labels from t_train and store them in labels
        Initialize an empty dictionary base_classifiers
        Initialize an empty list rounds

        For each label in labels:
            Create a binary target vector t_bin_train where current label is 1 and others
            Initialize a new classifier instance clf
            Train clf using X_train and t_bin_train, passing additional keyword arguments
            Store clf in base_classifiers with current label as the key
            Append the number of training rounds from clf to rounds

    Method predict(x):
        For each label in labels:
            Predict probabilities for x using the classifier corresponding to the current
            Reshape predictions to column format

        Concatenate all probability predictions into a matrix
        Return the index of the highest probability in each row (the predicted class)
```