

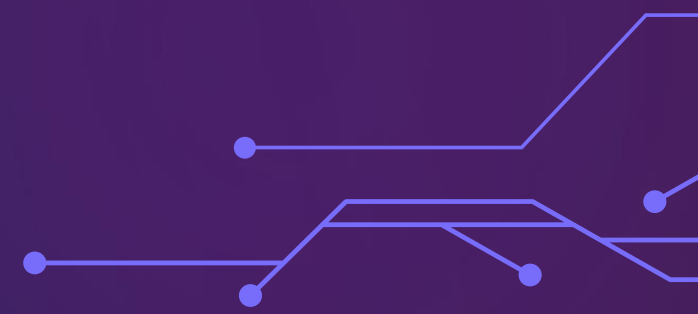


Data Clustering

19 Oct 2566

Sarun Gulyanon

Agenda



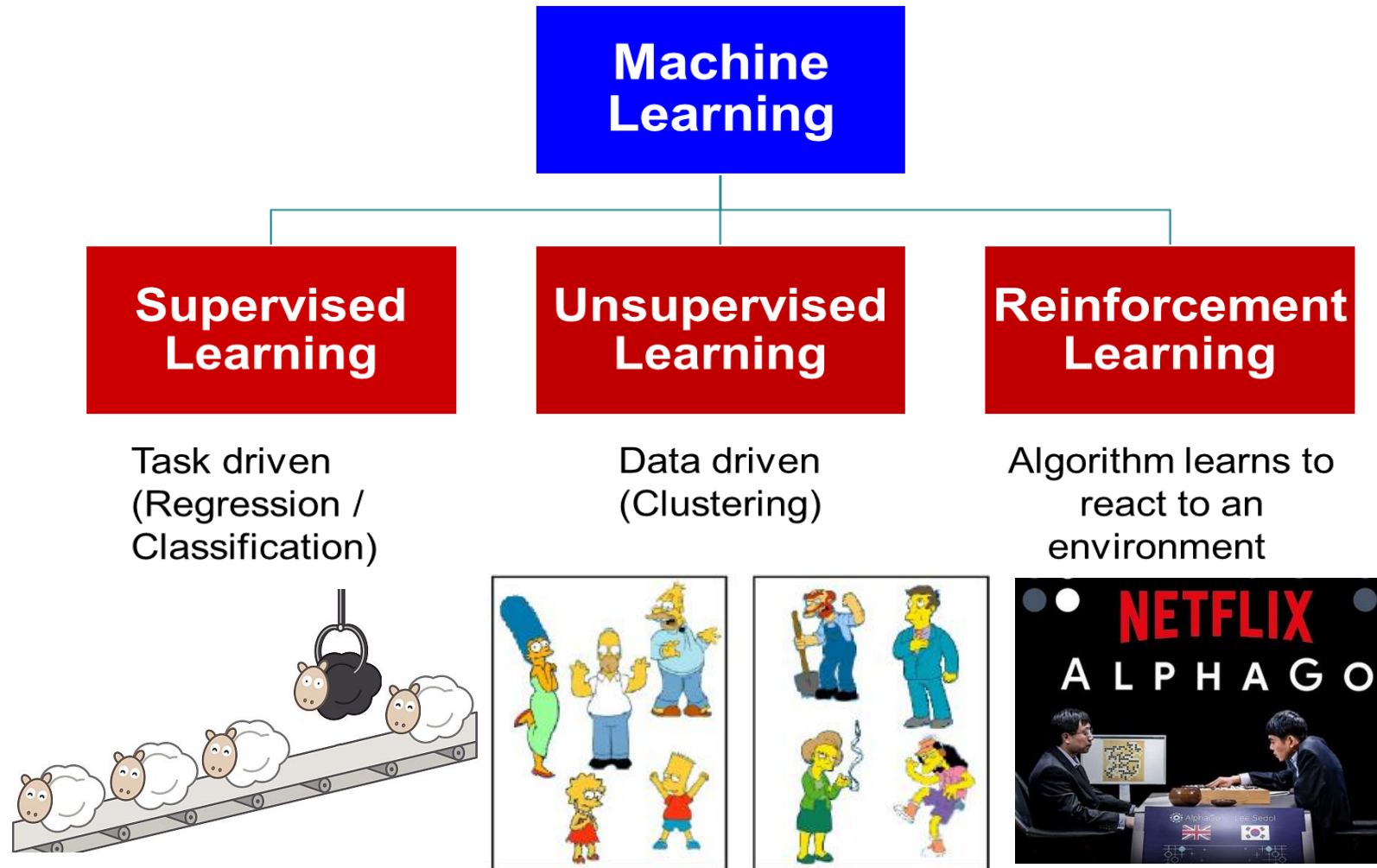
- What is Clustering?
- Clustering Techniques
 - K-Means
 - Gaussian Mixtures
- Applications



1.

What is Clustering?

Types of ML: Tasks



Clustering

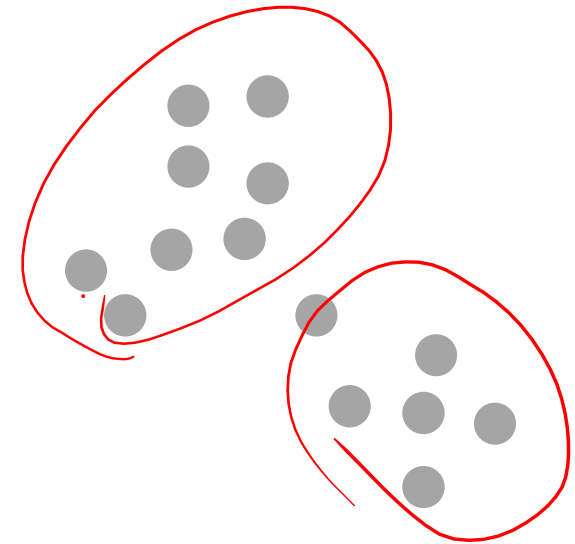


- Vast majority of the available data is unlabeled: we have the input features \mathbf{X} , but we do not have the labels \mathbf{y} .
- *Focus on* unsupervised learning tasks:
 1. *Clustering*: Group similar instances together into *clusters*.
 2. *Anomaly detection*: Learn what “normal” data looks like, and then use that to detect abnormal instances.
 3. *Dimensionality Reduction*: Reduce the number of input features while retaining the most important information.
 4. *Association Rule Learning*: Discover interesting relations between attributes.

Clustering



- Group similar instances together into *clusters*.
- There is no universal definition of what a cluster is: it really depends on the context.
- How to group? → Many possible approaches:
 - Representative-Based Clustering
 - Density-Based Clustering
 - Hierarchical Clustering





2.

Clustering Techniques

Representative-Based Clustering



- Given a dataset with n points in a d -dimensional space and given the number of desired clusters k ,
- **Goal:** partition the dataset into k groups or clusters
- There is a representative point that summarizes each cluster C_i , a common choice being the mean (or the *centroid*) μ_i

$$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} \mathbf{x}_j \quad \text{where } n_i = \text{number of points in cluster } C_i.$$

n k

- Exhaustive search of all possible clusterings is not practically feasible



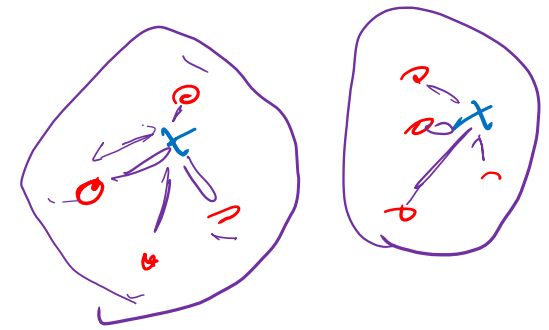
K-Means Clustering

K-Means Clustering

- K-means algorithm finds the clusters that minimize the sum of squared errors (SSE):

Score Function:

$$\text{SSE}(\mathcal{C}) = \sum_{i=1}^k \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mu_i\|^2$$



- Let μ_i denote a d-dimensional vector, $i = 1, \dots, k$, in which μ_i is the center/mean of the i^{th} cluster.
- Hard clustering = assign each point to a single cluster
- Assign data point to its closest vector μ_i

K-Means Method

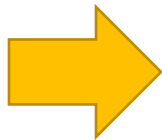


- K-means employs a greedy iterative approach
 1. Initializes the cluster means μ_i by randomly generating k points in the data space.
 2. Each iteration of K-means consists of two steps:
 - a) cluster assignment, and
 - b) centroid update
 3. K-means has converged if the centroids do not change from one iteration to the next (or less than the convergence threshold).

$$\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$$

Notation

- Each data point x_j has a corresponding set of binary indicator variables $r_{ij} \in \{0, 1\}$, describing whether the data point x_j is assigned to the i^{th} cluster.
- If data point x_j is assigned to i^{th} cluster then $r_{ij} = 1$, and $r_{kj} = 0$ for $i \neq k$, known as the 1-of-K coding or one-hot encoding.



Cluster	Cluster 1	Cluster 2	Cluster 3
1	1	0	0
1	1	0	0
2	0	1	0
3	0	0	1

Clustering Assignment



- Rewrite the objective function

$$J(\mathcal{C}) = \sum_{i=1}^k \sum_{j=1}^N r_{ij} \|\mathbf{x}_j - \mu_i\|^2$$

- This step finds $\{r_{ij}\}$ that minimize J . (Fix μ_i)

$$r_{ij} = \begin{cases} 1 & \text{if } i = \arg \min_i \|\mathbf{x}_j - \mu_i\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

Centroid Update



- This step finds the value $\{\mu_i\}$ to minimize J . (Fix $\{r_{ij}\}$)

$$J(\mathcal{C}) = \sum_{i=1}^k \sum_{j=1}^N r_{ij} \|\mathbf{x}_j - \mu_i\|^2$$

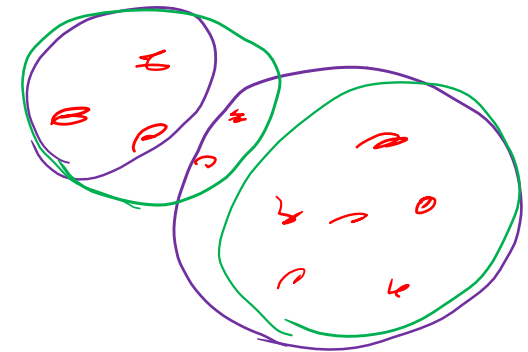
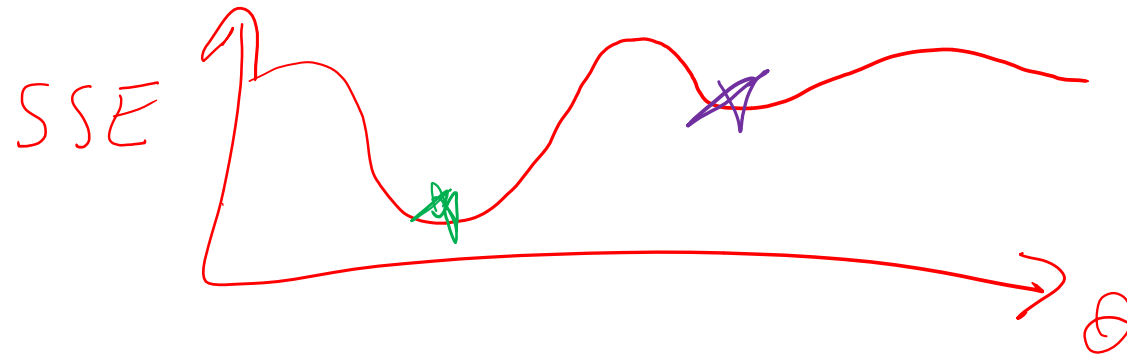
$$\frac{\partial}{\partial \mu_i} J(\mathcal{C}) = 2 \sum_{j=1}^N r_{ij} (\mathbf{x}_j - \mu_i) = 0$$

$$\mu_i = \frac{\sum_{j=1}^N r_{ij} \mathbf{x}_j}{\sum_{j=1}^N r_{ij}}$$

K-Means Clustering



- The two steps — cluster assignment, and centroid update — are repeated in turn until there is no further change in the assignments (or until some maximum number of iterations is exceeded).
- Because each phase reduces the value of the objective function J , convergence of the algorithm is assured.
- However, it may converge to a local rather than global minimum of J .



K-Means Clustering



- We can generalize the K-means algorithm by introducing a more general dissimilarity measure $V(x, x')$ between two vectors x and x'

$$\tilde{J} = \sum_{i=1}^k \sum_{j=1}^N r_{ij} \mathcal{V}(\mathbf{x}_j, \mu_i)$$

- <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>



Gaussian Mixture Model

Gaussian Mixtures



- A *Gaussian mixture model* (GMM) is a probabilistic model that assumes that the instances were generated from a mixture of several Gaussian distributions whose parameters are unknown.
- Generalize the K-means approach
- *Soft* clustering approach = each point has a probability of belonging to each cluster.

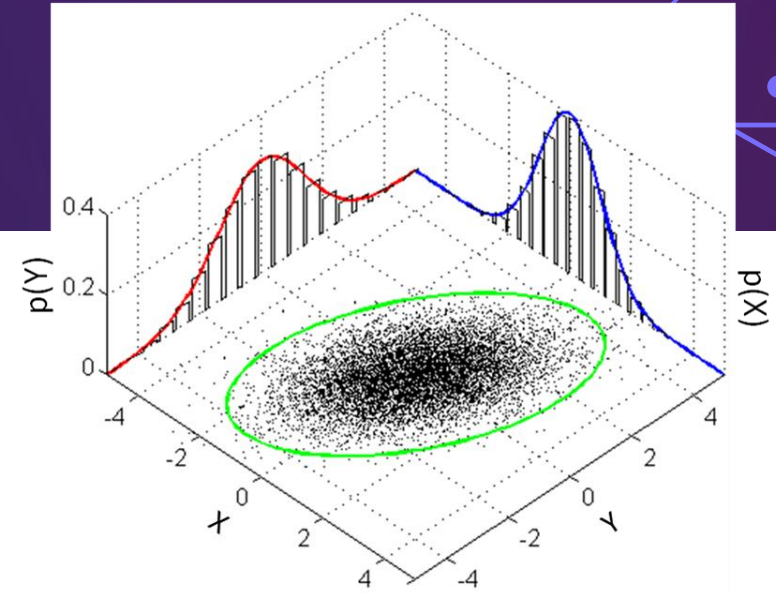
Gaussian Mixtures

- The number of Gaussian distributions, k , is given.
- Each cluster C_i is characterized by a multivariate normal distribution

$$f_i(\mathbf{x}) = f(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp \left\{ -\frac{(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)}{2} \right\}$$

- The probability density function of \mathbf{x} as a GMM with k clusters,

$$f(\mathbf{x}) = \sum_{i=1}^k f_i(\mathbf{x}) P(C_i) = \sum_{i=1}^k f(\mathbf{x}|\mu_i, \Sigma_i) P(C_i)$$



Mixture Parameters

- $P(C_i)$ are called the *mixture parameters*, where the prior probability of each class must satisfy the condition:

$$\sum_{i=1}^k P(C_i) = 1$$

- The mean μ_i , the covariance matrix Σ_i , and the mixture probability $P(C_i)$ for each of the k normal distributions are all the model parameters θ

Maximum Likelihood Estimation (MLE)

- Given the dataset $\mathbf{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the *likelihood* of \mathbf{D} given θ is:

$$P(\mathbf{D}|\theta) = \prod_{j=1}^n f(\mathbf{x}_j)$$

- MLE picks θ that maximize the likelihood $\theta^* = \arg \max_{\theta} \{P(\mathbf{D}|\theta)\}$
- In practice, use log-likelihood:

$$\ln P(\mathbf{D}|\theta) = \sum_{j=1}^n \ln f(\mathbf{x}_j) = \sum_{j=1}^n \ln \left(\sum_{i=1}^k \underbrace{f(\mathbf{x}_j|\mu_i, \Sigma_i)}_{\text{Gaussian}} \underbrace{P(C_i)}_{\text{Mixture}} \right)$$

Expectation-Maximization (EM) Algorithm

- Maximizing the log-likelihood over θ is hard
- EM is a two-step iterative approach that starts from an initial guess for the parameters θ
- 1. E-Step: expectation step computes the cluster posterior probabilities $P(C_i | \mathbf{x}_j)$
- 2. M-Step: maximization step re-estimates θ using $P(C_i | \mathbf{x}_j)$
- EM algorithm has converged if the centroids do not change from one iteration to the next (or less than the convergence threshold).

$$\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$$

Initialization



- For each cluster C_i , with $i = 1, 2, \dots, k$, randomly initialize parameters:
- The d -dimensional mean vector μ_i , for each dimension, is selected uniformly at random from the range of *possible value*.
- The covariance matrix is initialized as the $d \times d$ identity matrix, $\Sigma_i = \mathbf{I}$
- One simplification is to assume that all dimensions are independent, which leads to a diagonal covariance matrix.
- The cluster prior probabilities are initialized to $P(C_i) = 1/k$, so that each cluster has an equal probability.

Expectation Step



- Compute the posterior probability $P(C_i | \mathbf{x}_j)$ of cluster C_i given point \mathbf{x}_j :

$$P(C_i | \mathbf{x}_j) = \frac{f_i(\mathbf{x}_j) \cdot P(C_i)}{\sum_{a=1}^k f_a(\mathbf{x}_j) \cdot P(C_a)}$$

- $P(C_i | \mathbf{x}_j)$ can be considered as the weight or contribution of the point \mathbf{x}_j to cluster C_i .
- So use the notation $\mathbf{w}_i = (w_{i1}, w_{i2}, \dots, w_{in})^T$ to denote the weight vector for cluster C_i , across all the n points.

Maximization Step

- The mean

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^n w_{ij} \cdot \mathbf{x}_j}{\sum_{j=1}^n w_{ij}}$$

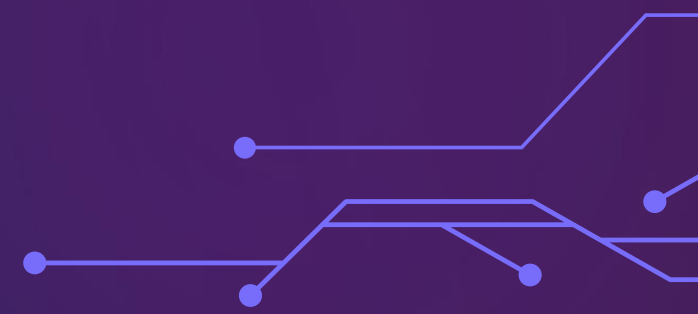
- The covariance matrix

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^n w_{ij} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^n w_{ij}}$$

- the prior probability

$$P(C_i) = \frac{\sum_{j=1}^n w_{ij}}{n}$$

Estimating the Prior Probability



- Introduce Lagrange multiplier α for the constraint $\sum_{a=1}^k P(C_a) = 1$

$$\frac{\partial}{\partial P(C_i)} \left(\ln(P(\mathbf{D}|\boldsymbol{\theta})) + \alpha \left(\sum_{a=1}^k P(C_a) - 1 \right) \right) = \left(\sum_{j=1}^n \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{f(\mathbf{x}_j)} \right) + \alpha$$

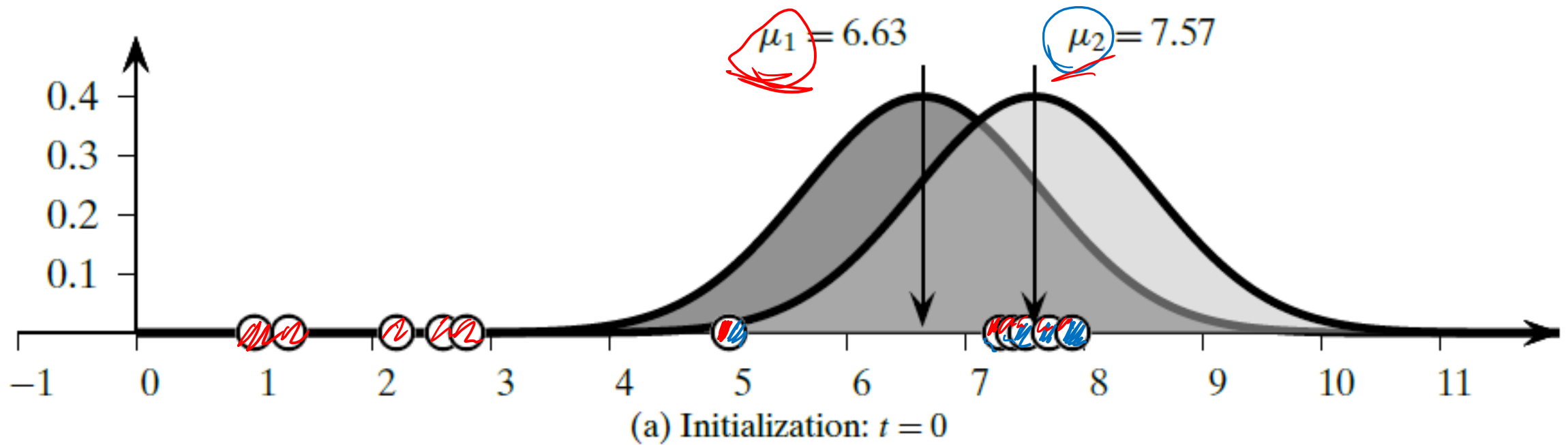
- multiplying on both sides by $P(C_i)$,

$$\sum_{j=1}^n \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i)}{f(\mathbf{x}_j)} = -\alpha P(C_i)$$

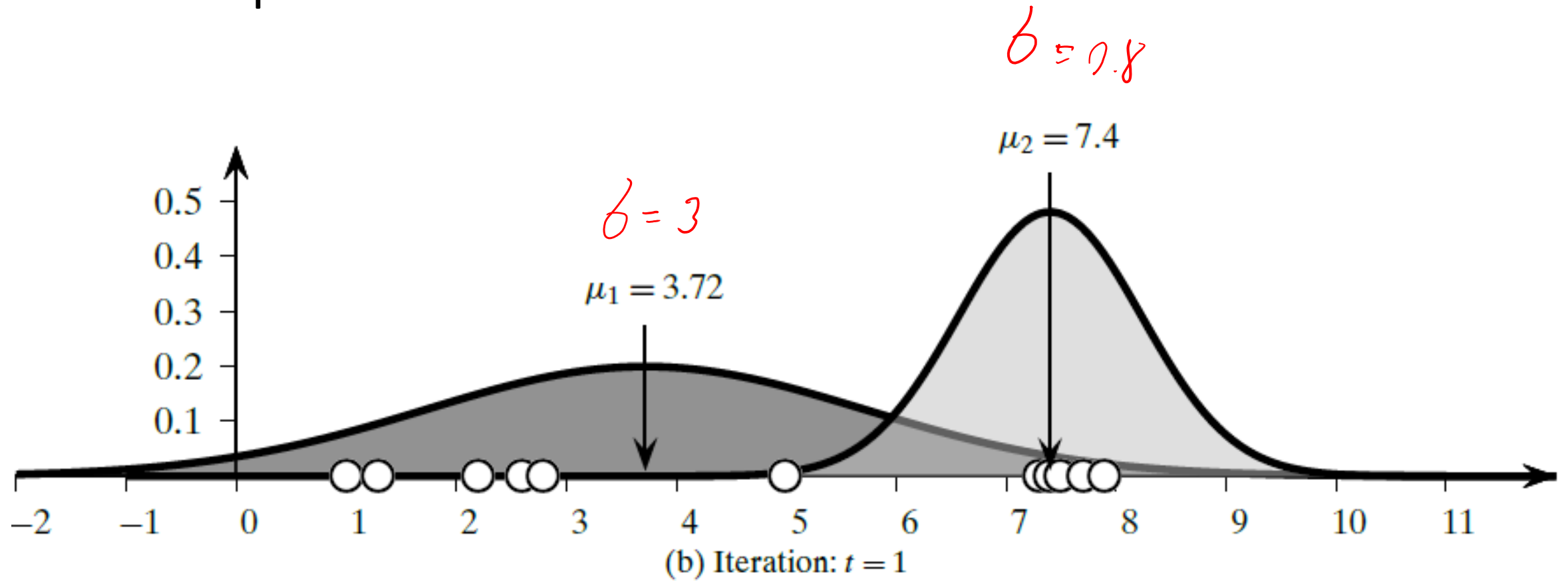
$$\sum_{j=1}^n w_{ij} = -\alpha P(C_i) \quad \text{where } n = -\alpha$$

Example

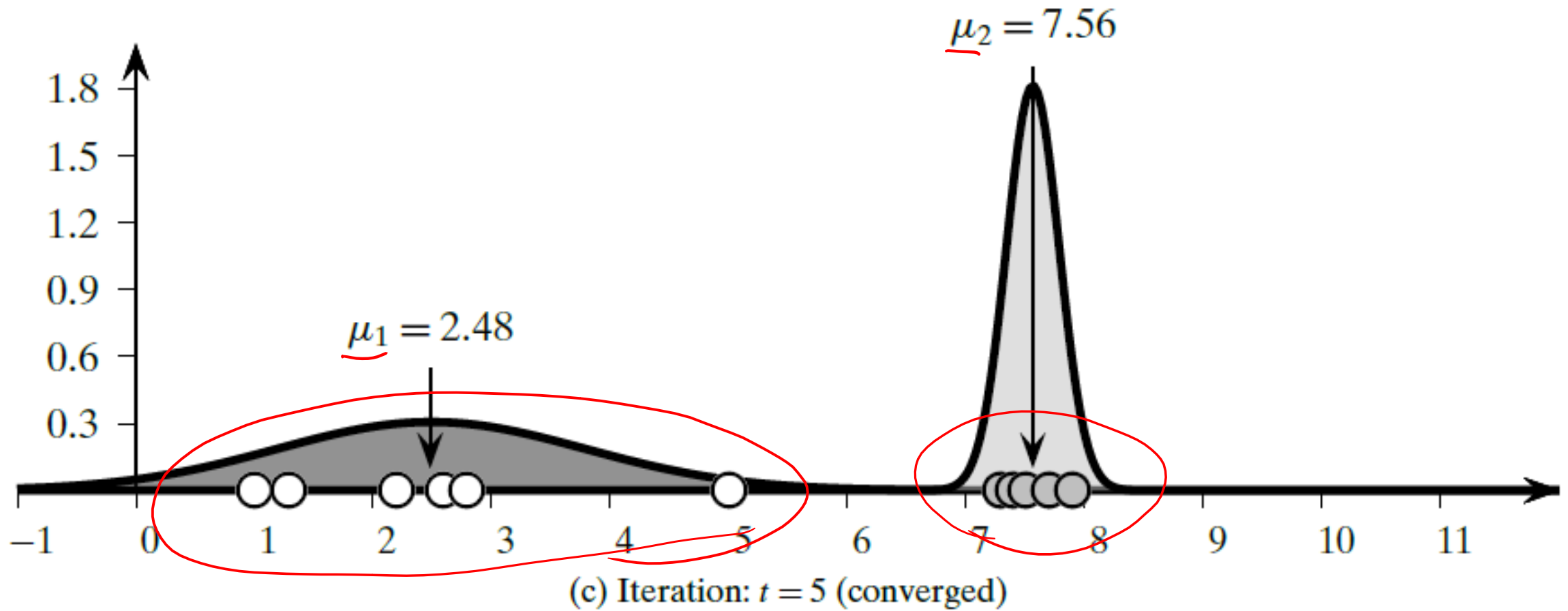
$$\underline{p(C_1) = \frac{1}{2}} \quad p(C_2) = \frac{1}{2}$$



Example



Example



Gaussian Mixture Properties

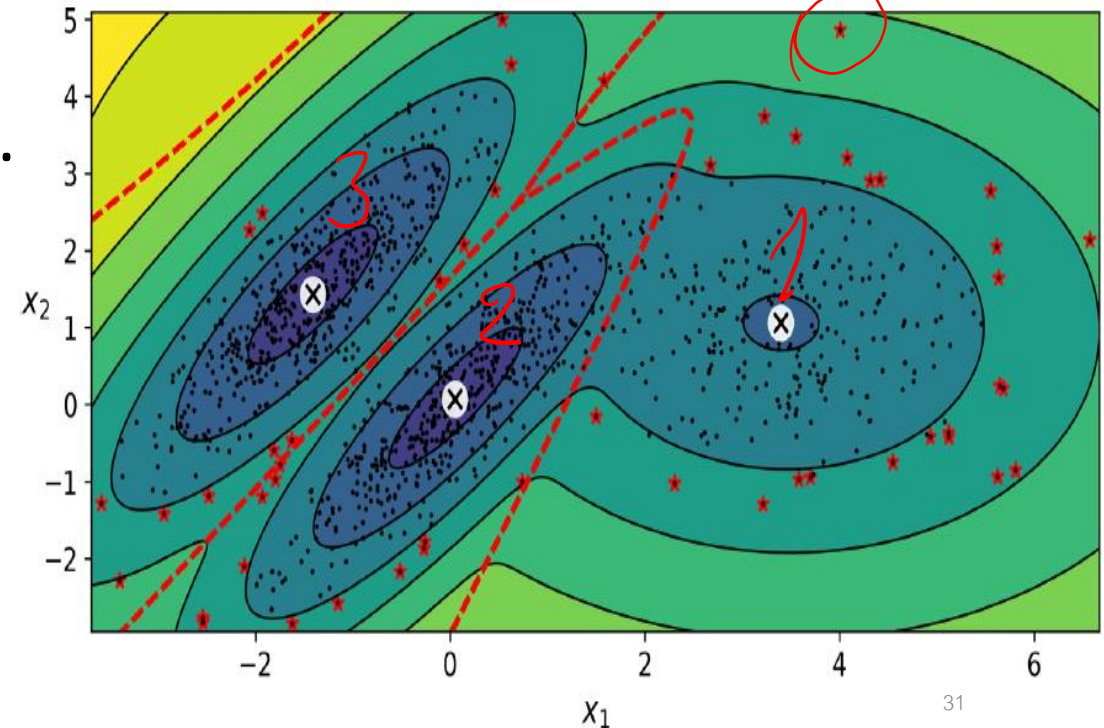


- K-means can be considered as a special case of the EM algorithm.
- Gaussian Mixture can be used to approximate the density estimation of the random process that generated the dataset.
 - Fit GMM on the dataset generated by such process.

Anomaly Detection Using GMM

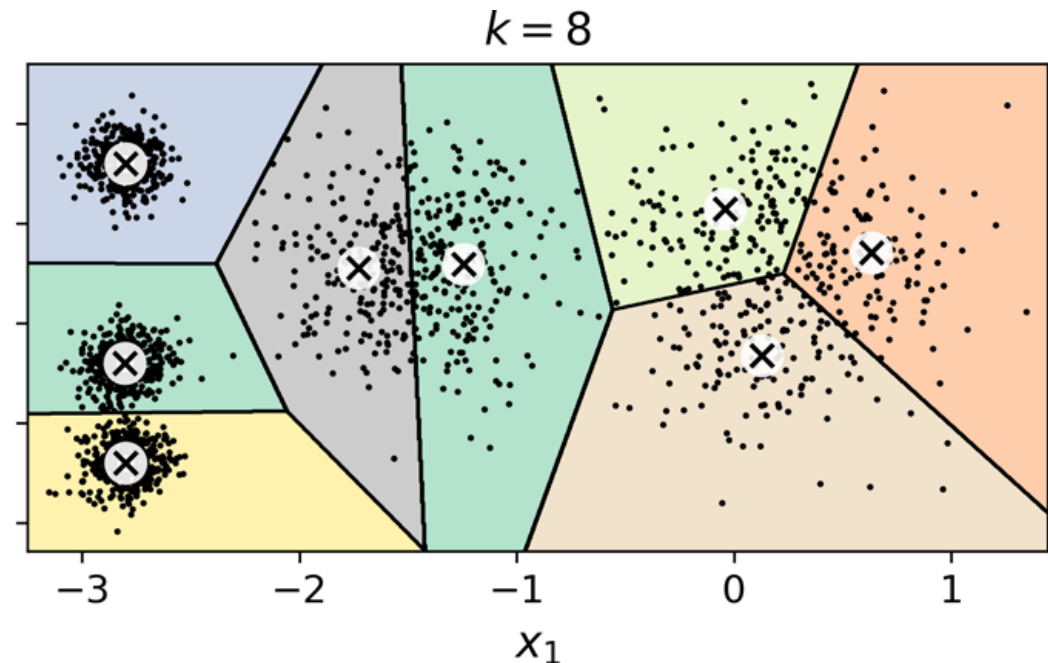
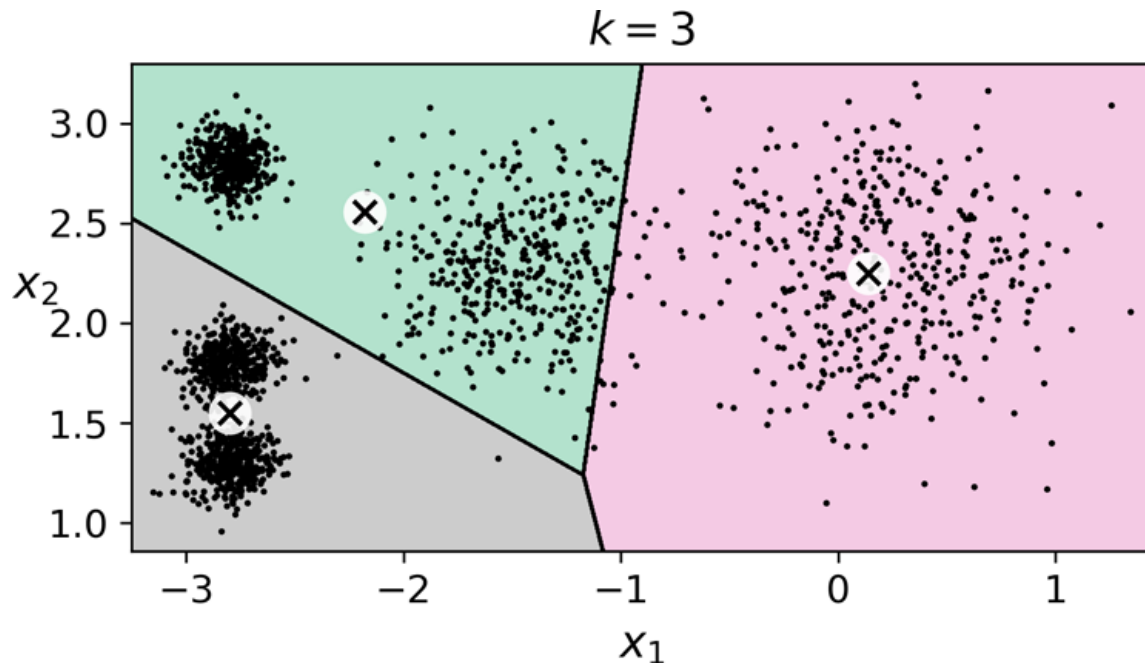
- *Anomaly detection* is the task of identifying observations that are statistically different from the rest of the observations
- Any instance located in a low-density region can be considered an anomaly.
- Manually define the density threshold.
- Empirically adjust the threshold
- Too many FP, lower the threshold.
- Too many FN, higher the threshold.

<https://towardsdatascience.com/understanding-anomaly-detection-in-python-using-gaussian-mixture-model-e26e5d06094b>



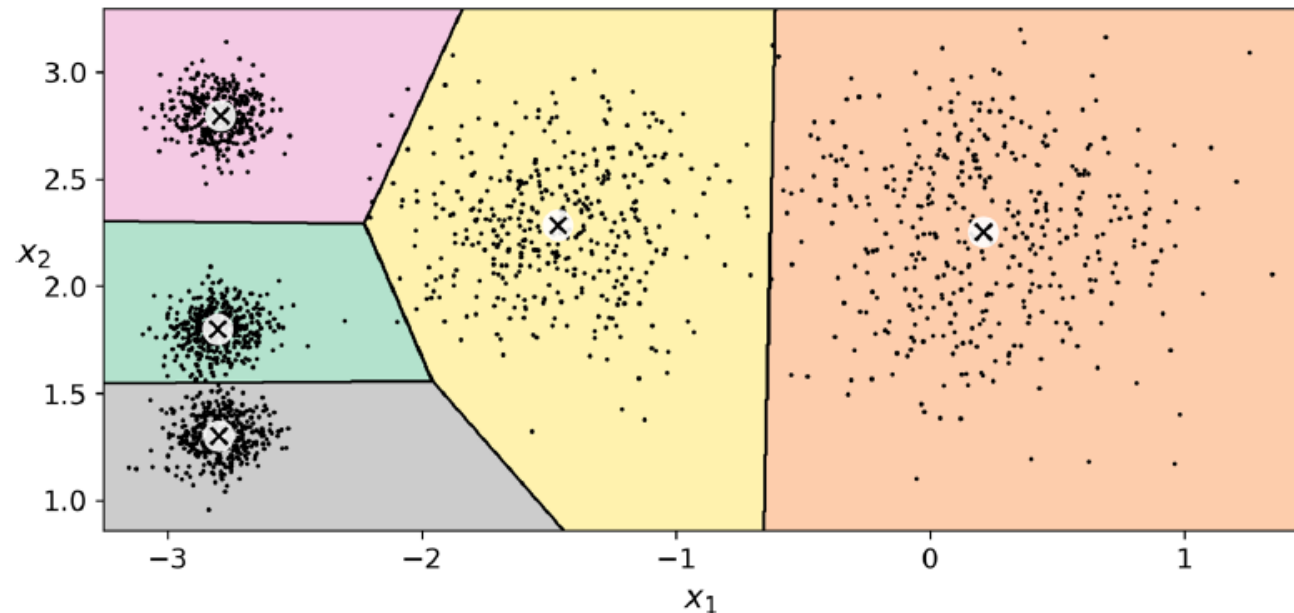
Finding The Optimal Number of Clusters

- Bad choices for the number of clusters:
 - when k is too small, separate clusters get merged
 - when k is too large, some clusters get chopped into multiple pieces



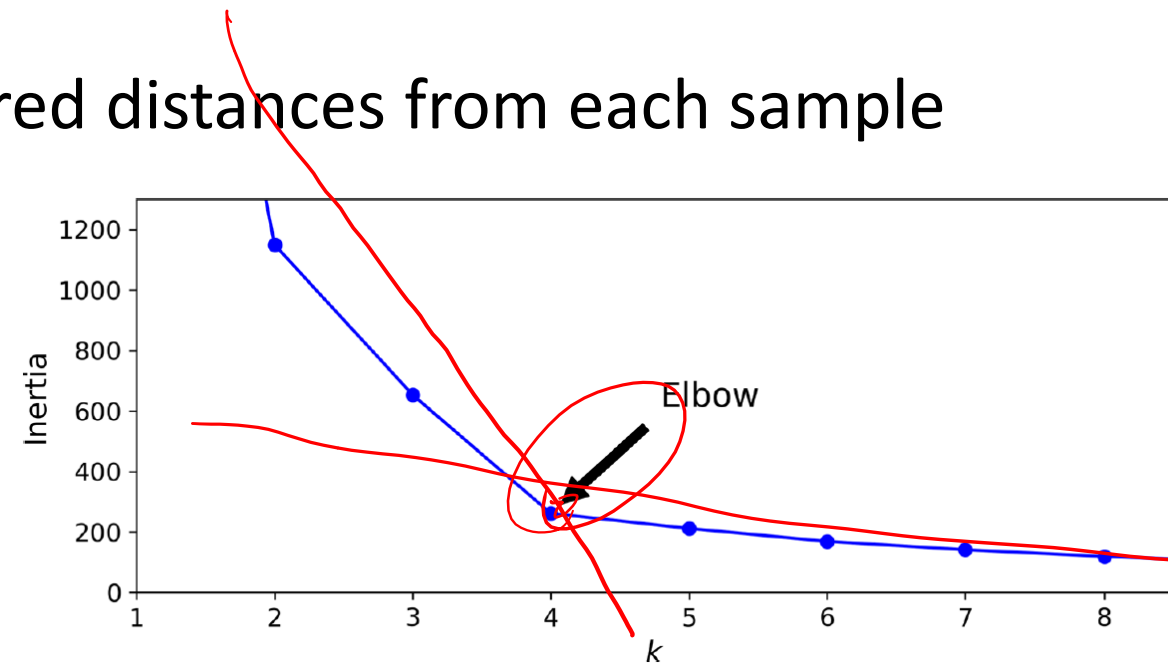
Finding The Optimal Number of Clusters

- Bad choices for the number of clusters:
 - when k is too small, separate clusters get merged
 - when k is too large, some clusters get chopped into multiple pieces
- Two methods:
 - Elbow method
 - Silhouette score/
Silhouette diagram



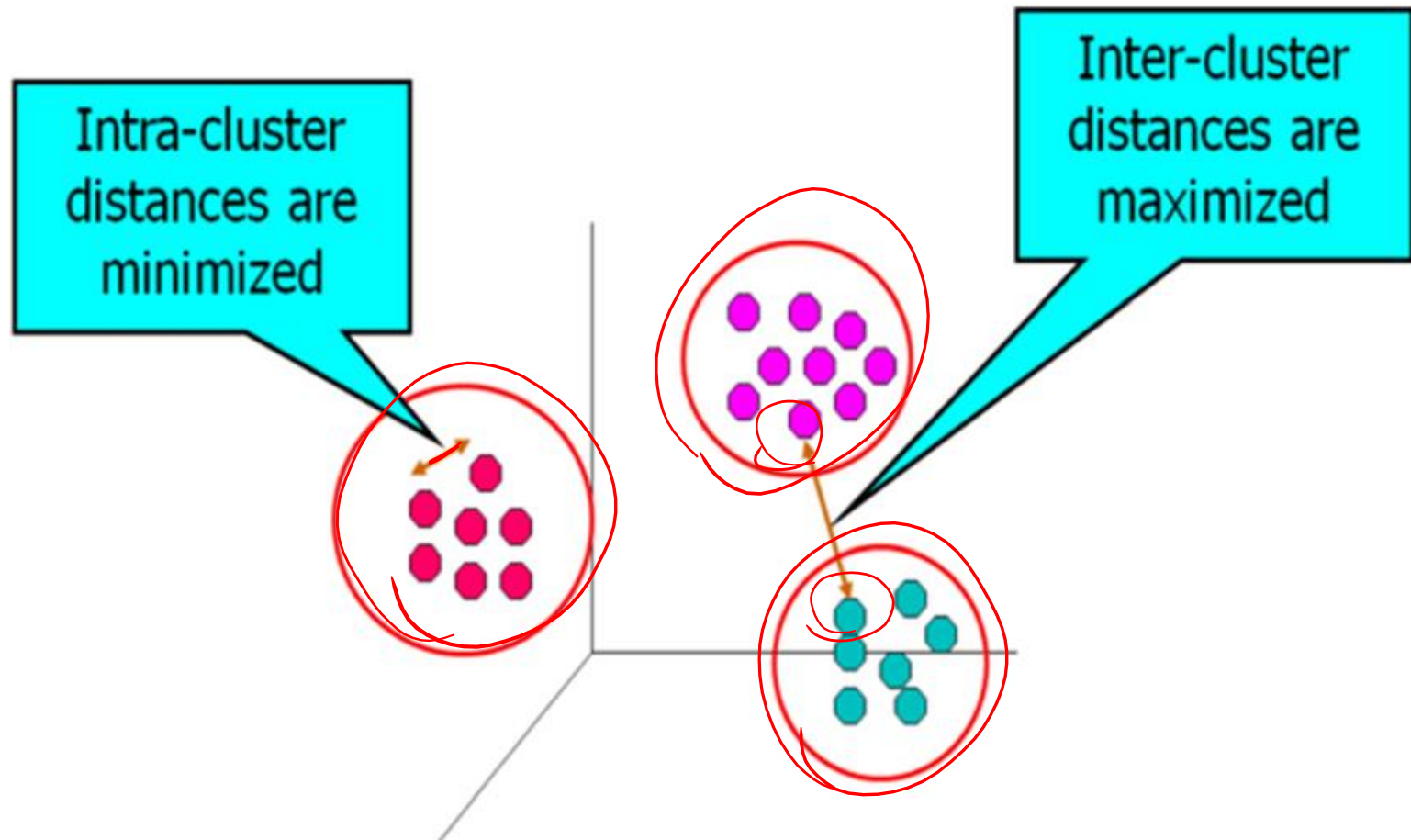
Elbow Method

- A heuristic for determining the number of clusters in a data set.
- Choose #clusters that adding another cluster doesn't give much better modeling of the data.
- Distortion/Inertia = the sum of squared distances from each sample to its cluster center.
- Select the value of k at the “elbow”
= distortion starts decreasing in a linear fashion.



Intra-Cluster VS Inter-Cluster Distances

- **Intra-cluster distance** is the distance among members of a cluster
- **Inter-cluster distance** is the distance between two different clusters



Silhouette Score



- The mean silhouette coefficient over all the instances
- The Silhouette Coefficient for a sample i is

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

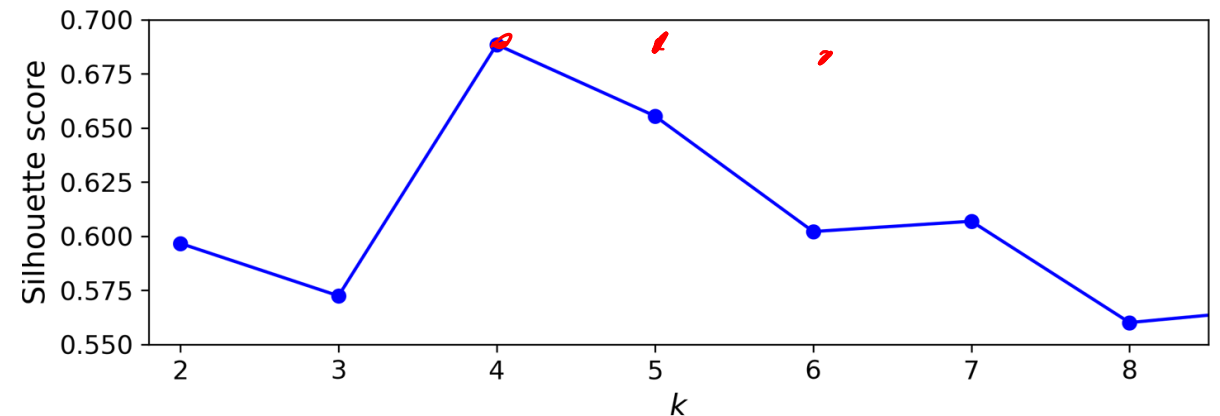
1.00
100%

- $a(i)$ = The average distance of i to all other data points in the same cluster mean (**intra-cluster distance**)
- $b(i)$ = The average distance of i to all data points in the nearest cluster (**inter-cluster distance**).

Silhouette Coefficient



- The silhouette coefficient can vary between -1 and $+1$
 - $+1$ = the instance is well inside its own cluster and far from other clusters
 - 0 = it is close to a cluster boundary
 - -1 = the instance may have been assigned to the wrong cluster.
- Give score for each value of k
- Plot every instance's silhouette coefficient
→ a silhouette diagram

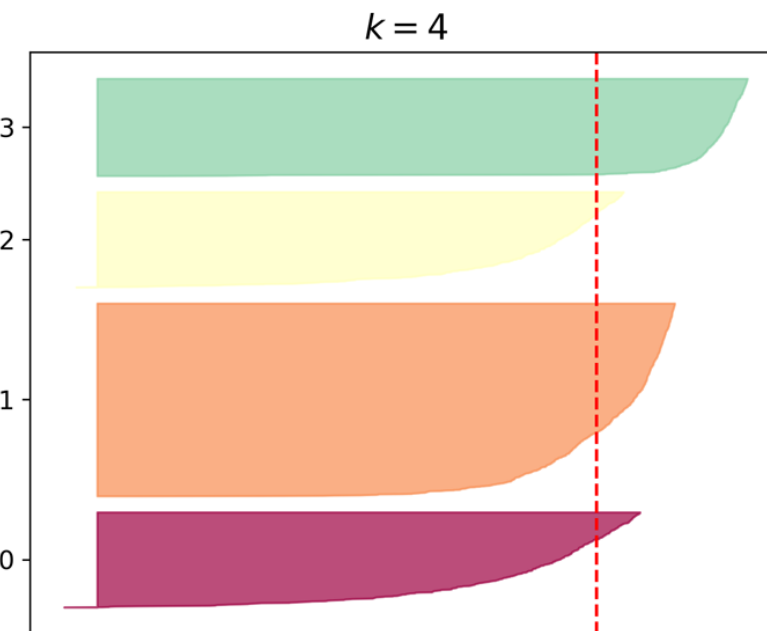
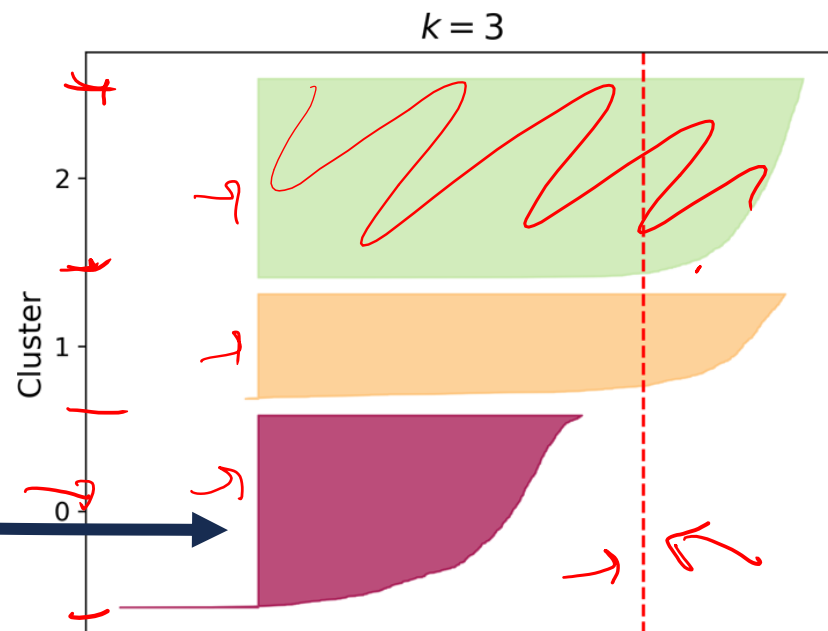


Silhouette Diagram

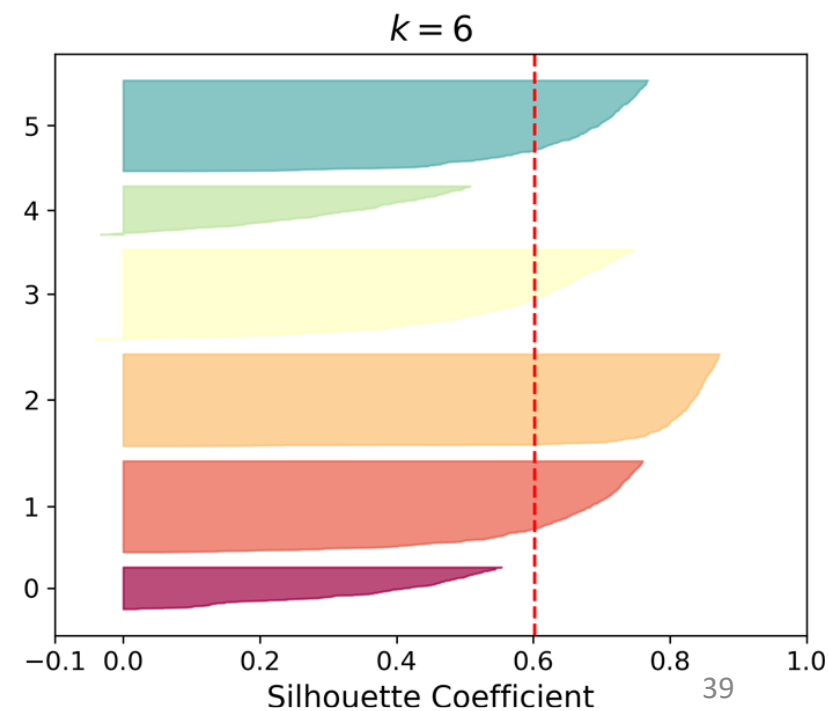
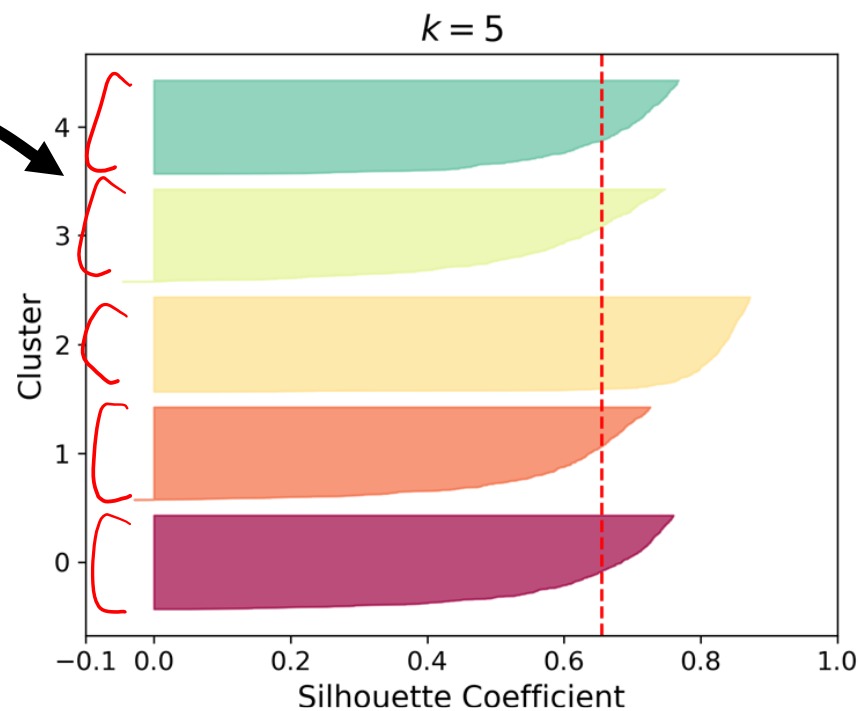



- Silhouette diagram contains one knife shape per cluster.
- The shape's height indicates the number of instances the cluster contains
- The shape's width represents the sorted silhouette coefficients of the instances in the cluster (wider is better).
- The dashed line indicates the mean silhouette coefficient.
- Demo: https://colab.research.google.com/drive/1sN_Kowf9w-FDwjt3Zje5hciWJm-tOsHL?usp=sharing

The cluster is bad if it stops short of the dashed line
→ Tts instances are much too close to other clusters



We prefer that all clusters have similar sizes





3.

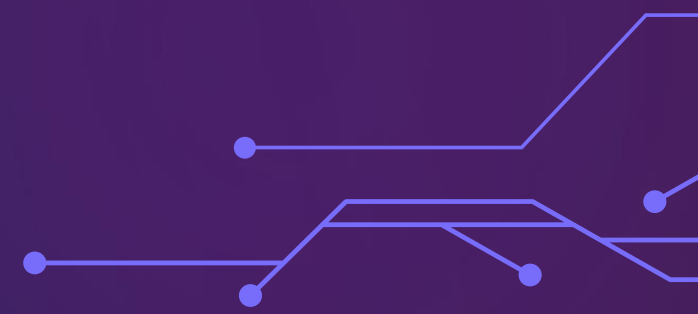
Applications

Clustering Applications



- Anomaly Detection Using Gaussian Mixtures
- Using Clustering for Image Segmentation
- Using Clustering for Semi-Supervised Learning

Color Segmentation



- Color segmentation = assign pixels to the same segment if they have a similar color.
- A simplified version of image segmentation, the task of partitioning an image into multiple segments.
- Image segmentation is used to locate objects and boundaries (lines, curves, etc.) in images.



<https://stackoverflow.com/questions/30303151/image-color-segmentation-with-opencv-c>

Original image



10 colors



8 colors



6 colors



4 colors



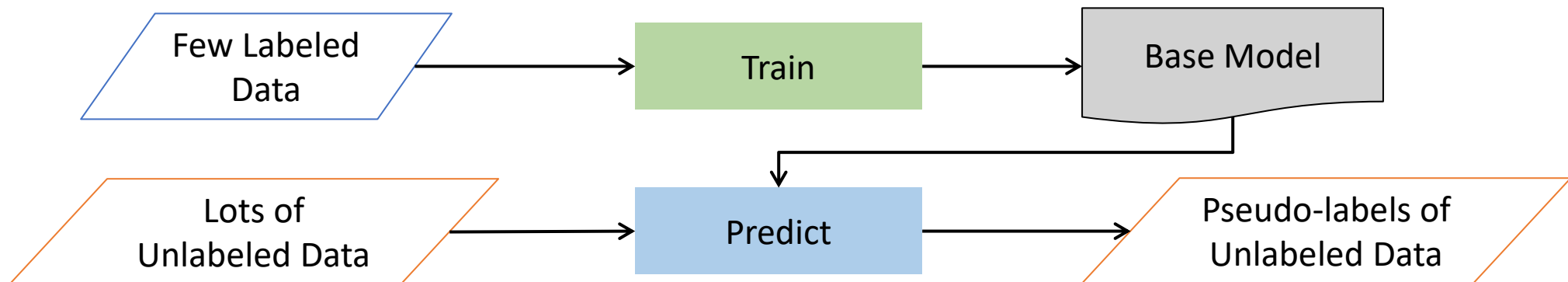
2 colors



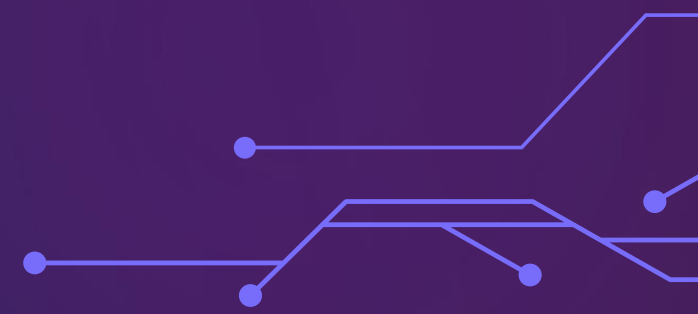
Semi-Supervised Learning



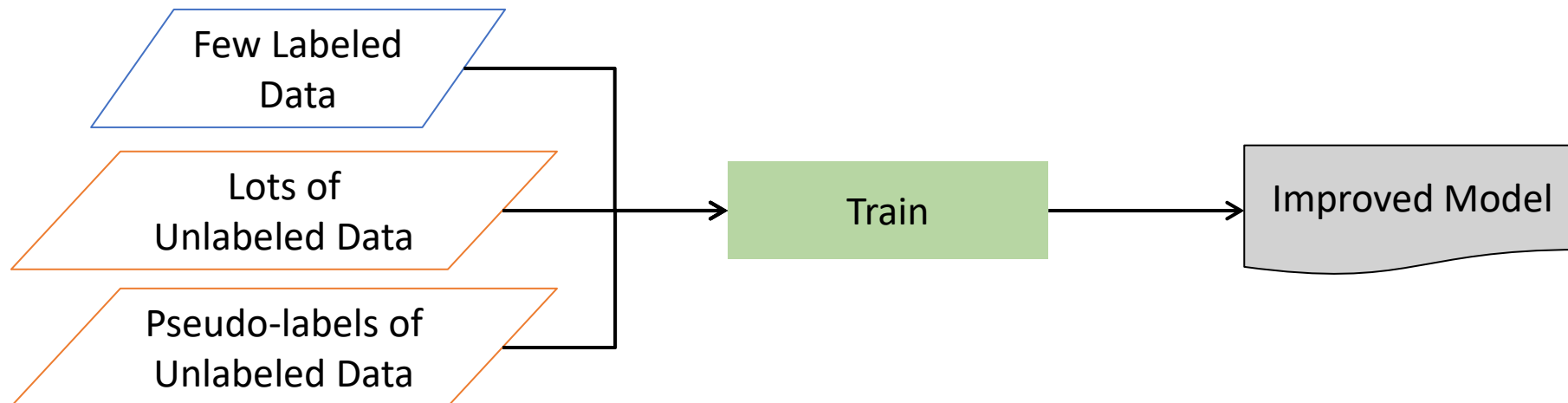
- Semi-Supervised Learning = when we have plenty of unlabeled instances and very few labeled instances.
- One of the simplest of semi-supervised learning is **self-training**.
 1. Use few labeled data to train the base model
 2. Use base model to predict the pseudo-labels of unlabeled data
 3. Use the few labeled data and unlabeled data with pseudo-labels as the new dataset



Self-Training Example

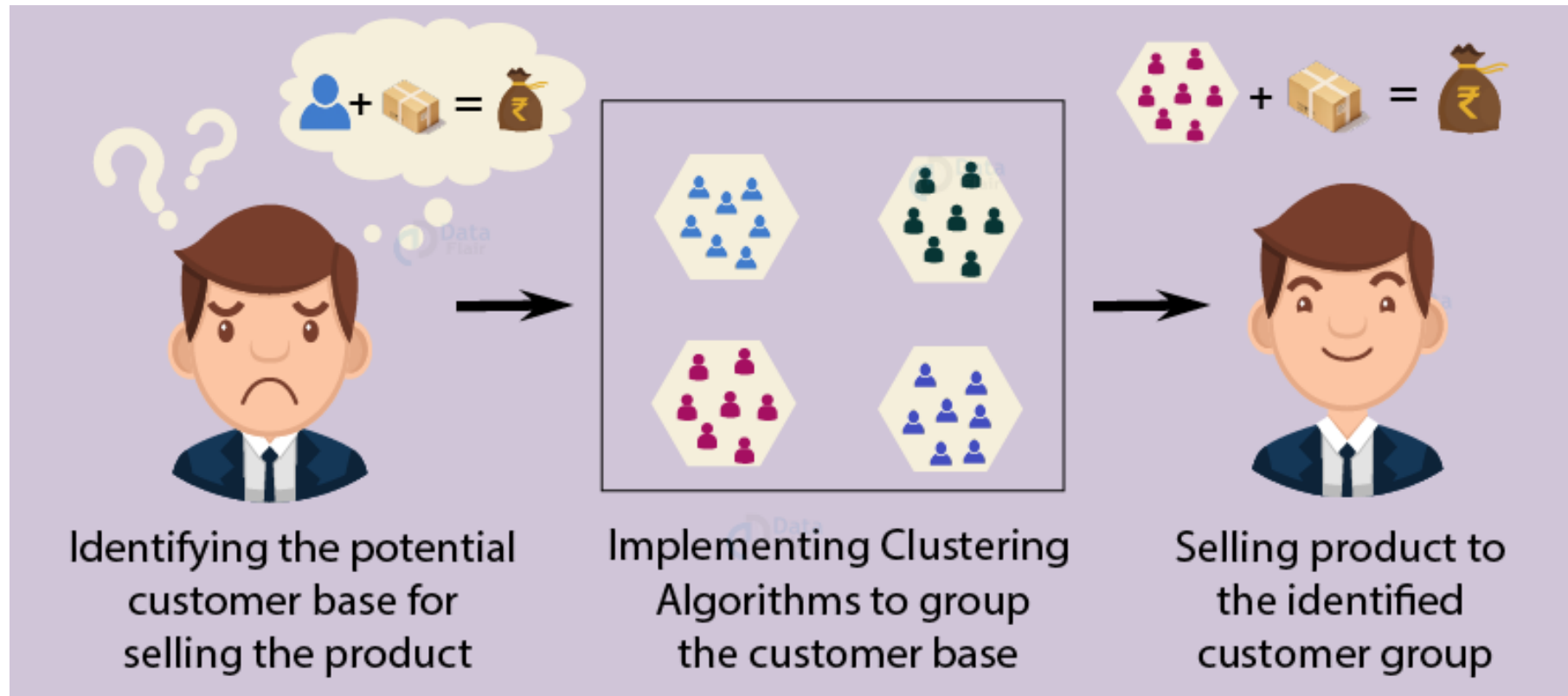


1. One possible approach is using k-means clustering on the labeled data.
2. Assign label to each cluster representative (may use voting).
3. Assign cluster to unlabeled data (may ignore instances far from the centroids).
4. Assign pseudo-label based on the cluster they belong.
5. Train a new model on the labeled data and unlabeled data + pseudo-labels.



Exercise: Customer Segmentation

- Dataset: <https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python>



Reference



- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 3rd ed by Aurélien Géron, 2022
Ch 9