Unsupervised Learning

Prof. Dr. M. Elif Karslıgil
Yıldız Teknik Üniversitesi - Bilgisayar Mühendisliği Bölümü
Akıllı Sistemler Laboratuvarı

Introduction

- Supervised Learning: Data has labels
  - Data : <Input, Output>
  - Input: Features, Output: Labels

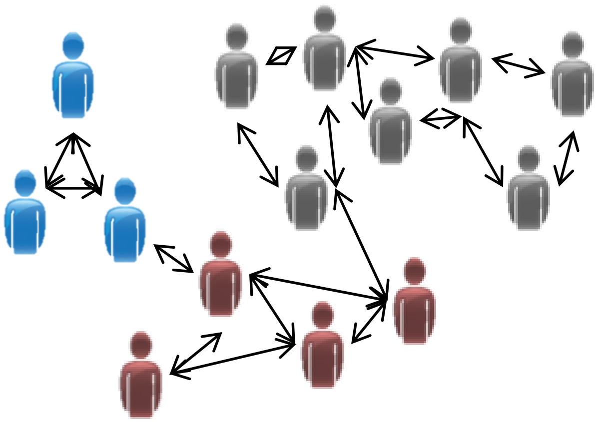
- If we don't have labels? Data : <X>

- If labeling too many examples is time-consuming.

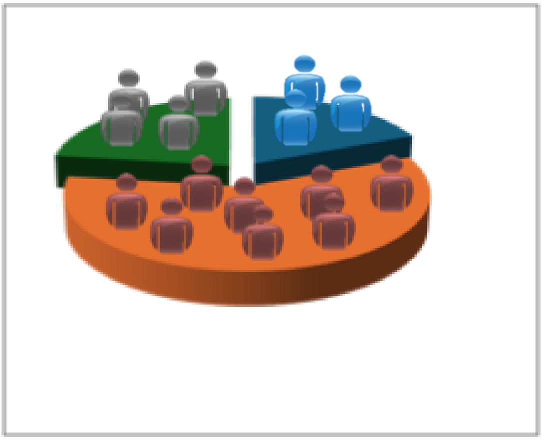
Motivating Examples

- Determine groups of people in image above
  - based on clothing styles
  - gender, age, etc

Motivating Examples

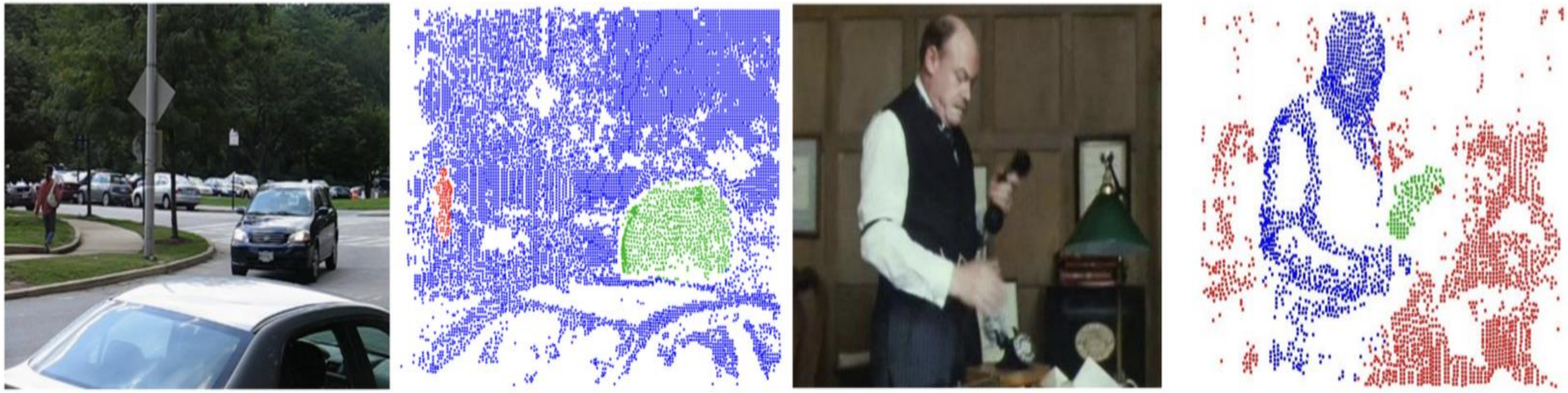


Social Network Analysis



Customer Segmentation

Motivating Examples

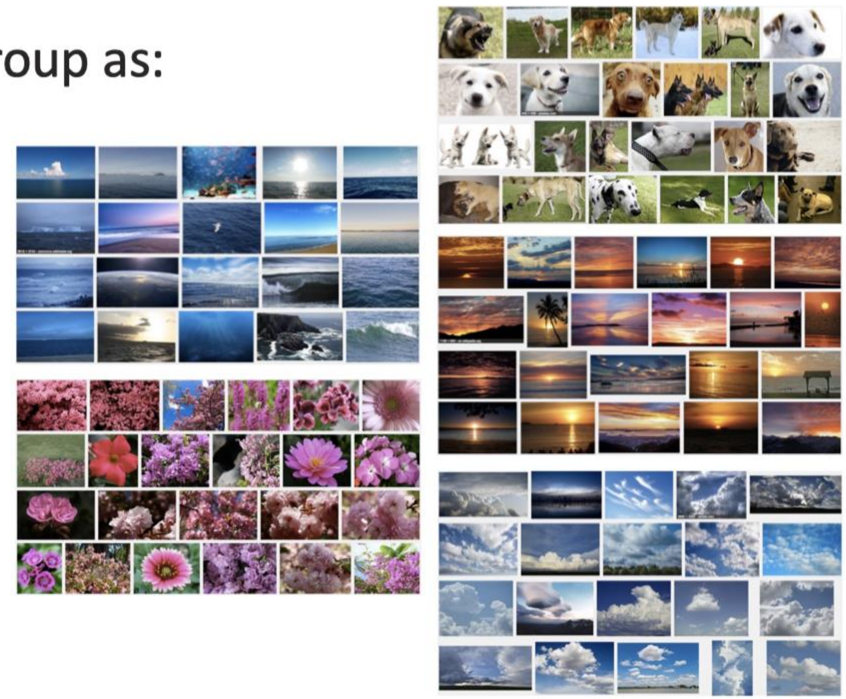


- Determine moving objects in videos

Motivating Examples

Clustering images

- For search, group as:
  - Ocean
  - Pink flower
  - Dog
  - Sunset
  - Clouds
  - …



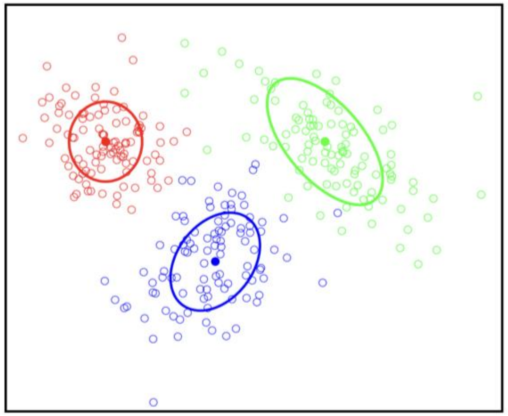
What is Clustering

Clustering

No labels provided

...uncover cluster structure from input alone

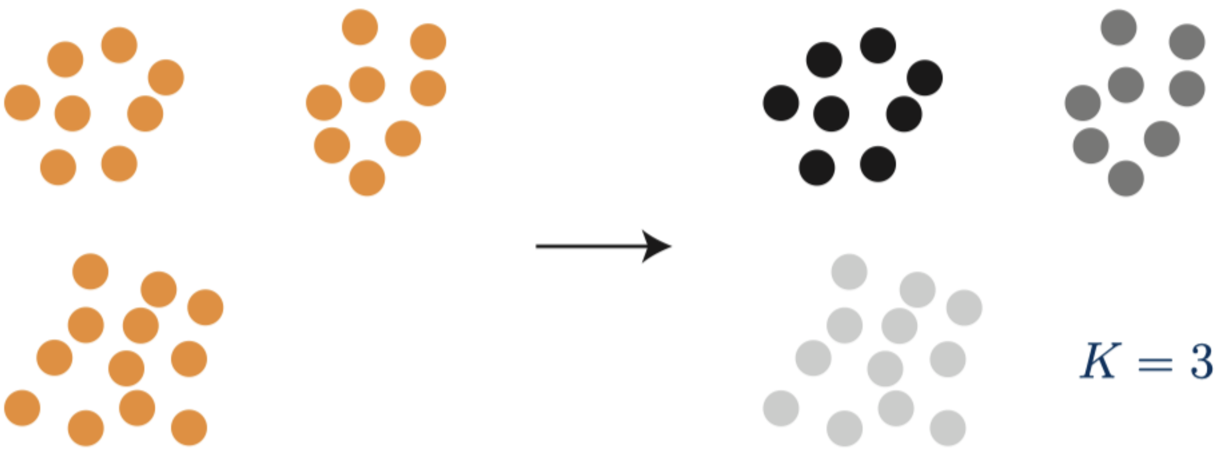
An unsupervised learning task



Input: docs as vectors x_i

Output: cluster labels z_i

What is Clustering



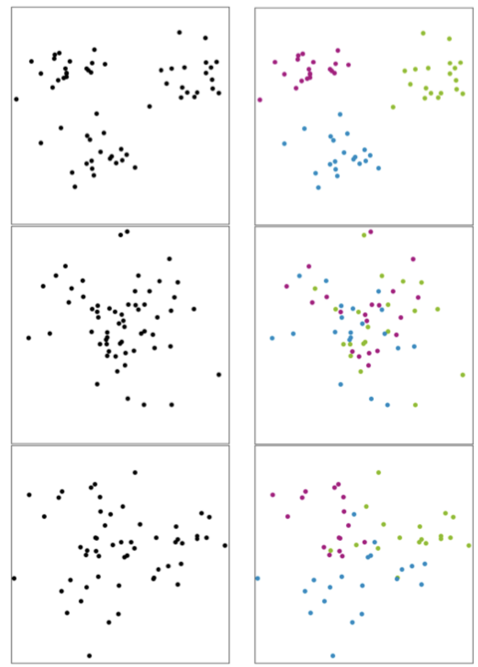
What is Clustering

Hope for unsupervised learning

Easy

Impossible

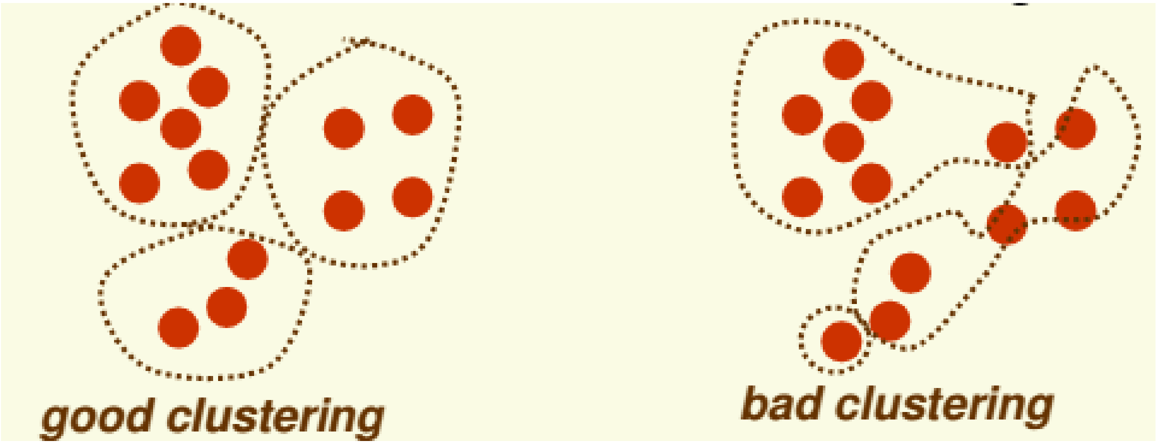
In between



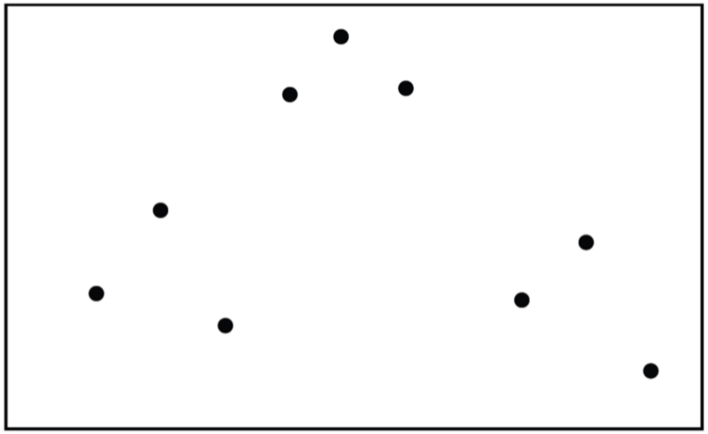
10

Unsupervised Learning

- Good clustering requires
  - **low** between-class similarity
  - **high** within-class similarity.



What is Clustering

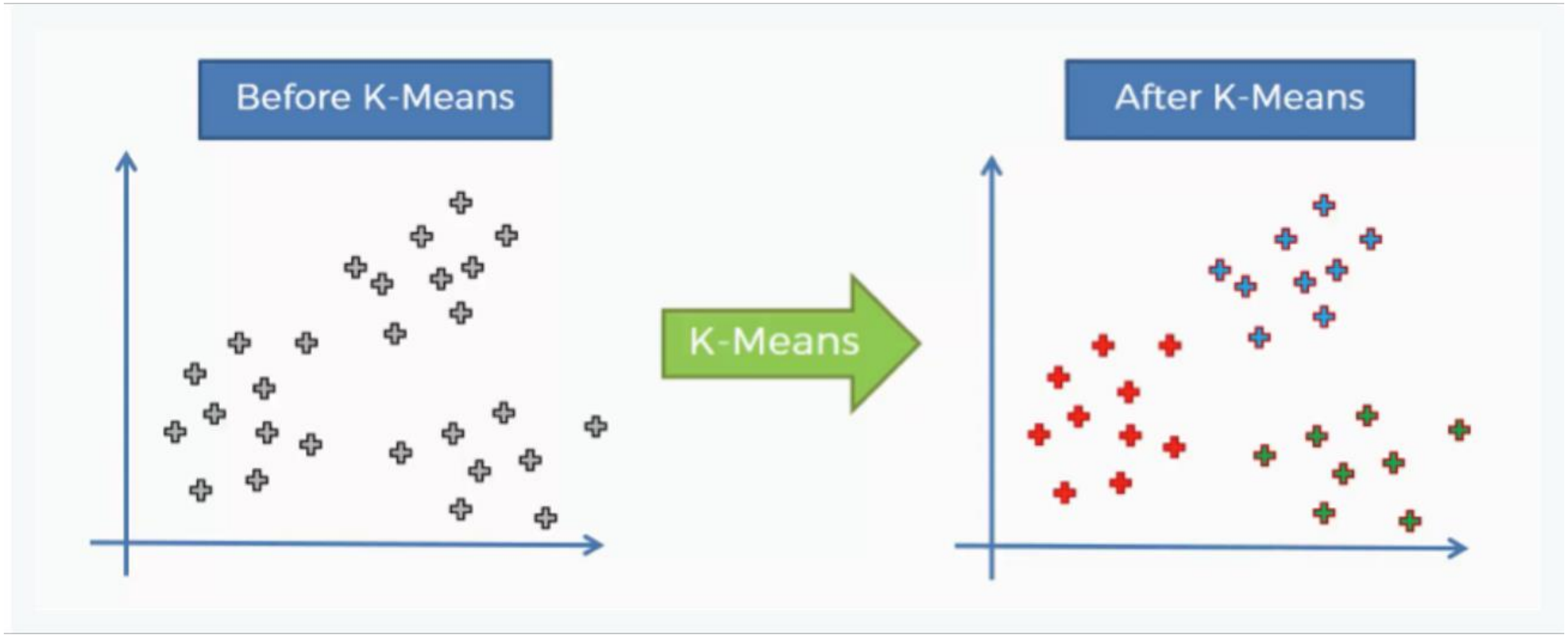


- Assume the data $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ lives in a Euclidean space, $\mathbf{x}^{(n)} \in \mathbb{R}^d$.
- Assume the data belongs to K classes (patterns)
- Assume the data points from same class are similar, i.e. close in euclidean distance.
- How can we identify those classes (data points that belong to each class)?

k-Means Clustering

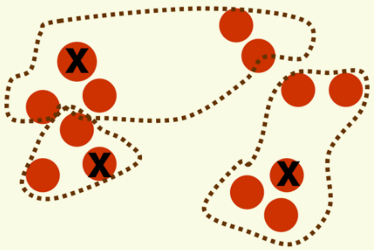
- K-means assumes there are k clusters, and each point is close to its cluster center (the mean of points in the cluster).
- If we knew the cluster assignment we could easily compute means.
- If we knew the means we could easily compute cluster assignment.
- Chicken and egg problem!
- Can show it is NP hard.
- Very simple (and useful) heuristic - start randomly and alternate between the two!

k-Means Clustering

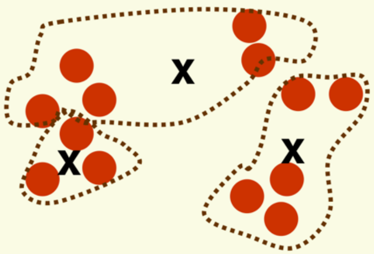


k-Means Clustering

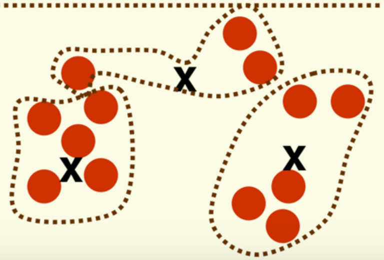
1. Initialize
   - pick ***k*** cluster centers arbitrary
   - assign each example to closest center



2. compute sample means for each cluster



3. reassign all samples to the closest mean



4. if clusters changed at step 3, go to step 2

k-Means Clustering

Input: A collection of data points $\mathcal{D} = \{x_1, \ldots, x_N\}$ where $x_n \in \mathbb{R}^D$.

Output: K cluster centers $\mu_k \in \mathbb{R}^D$ and an **assignment** $z_n \in \{1, \ldots, K\}$ of each data point to one of the cluster centers.

- The means μ_k are **model parameters** of the model ($z_n$can be computed from μ) that are **fit to the data.**
- K is a **hyperparameter** which we need to select.

Objective: Each data point should be close to its assigned center.

$$\arg\min_{\mu, z} \sum_{n=1}^{N} \left\| x_n - \mu_{z_n} \right\|_2^2$$

$(\mathrm{L}_2\text{−Norm})^2$

Euclidean Distance Squared

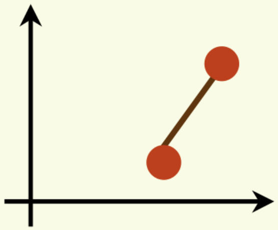
$$\|a\|_2^2 = \sum_{d=1}^{D} a_d^2$$

Distance Metrics

- Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

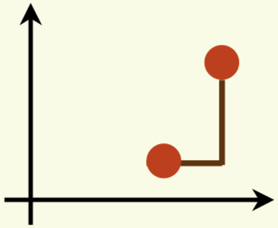
  - translation invariant



- Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

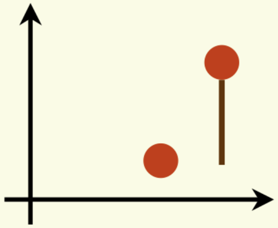
  - approximation to Euclidean distance, cheaper to compute



- Chebyshev distance

$$d(x_i, x_j) = \max_{1 \le k \le d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

  - approximation to Euclidean distance, cheapest to compute



k-Means Cluster (Lloyd's Algorithm)

Initialization: Choose K points at random to be the initial cluster centers μ.

Iterate until convergence:

1. **Update Assignments:** Assign each point to its nearest cluster center.

$$z = \arg\min_{z} \sum_{n=1}^{N} \left\| x_n - \mu_{z_n} \right\|_2^2$$

2. **Update Centers:** Recompute centers by averaging assigned points.

Why is this the mean of each cluster?

$$\mu = \arg\min_{\mu} \sum_{n=1}^{N} \left\| x_n - \mu_{z_n} \right\|_2^2$$

k-Means Cluster
Updating Cluster Centers

What is the value that minimizes?

$$\arg\min_{\mu} \sum_{n=1}^{N} \left\| x_n - \mu_{z_n} \right\|_2^2$$

We can re-arrange the objective to optimize with respect to μ:

$$\sum_{n=1}^{N} \left\| x_n - \mu_{z_n} \right\|_2^2 = \sum_{n=1}^{N} \sum_{d=1}^{D} \left(x_{nd} - \mu_{z_n d} \right)^2 = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \sum_{d=1}^{D} \left(x_{k_i d} - \mu_{kd} \right)^2$$

k_i is the i^{th} **point in cluster k**

Sum over **points in cluster k**

Sum over **clusters**

Convergence of k-Means

Is the k-means (Lloyd's) algorithm guaranteed to converge?

- Yes ☺! Why?

Alternating minimization always decreases the objective

$$z = \arg\min_{z} \sum_{n=1}^{N} \left\| x_n - \mu_{z_n} \right\|_2^2 \quad \text{and} \quad \mu = \arg\min_{\mu} \sum_{n=1}^{N} \left\| x_n - \mu_{z_n} \right\|_2^2$$

How do we know when it has converged?

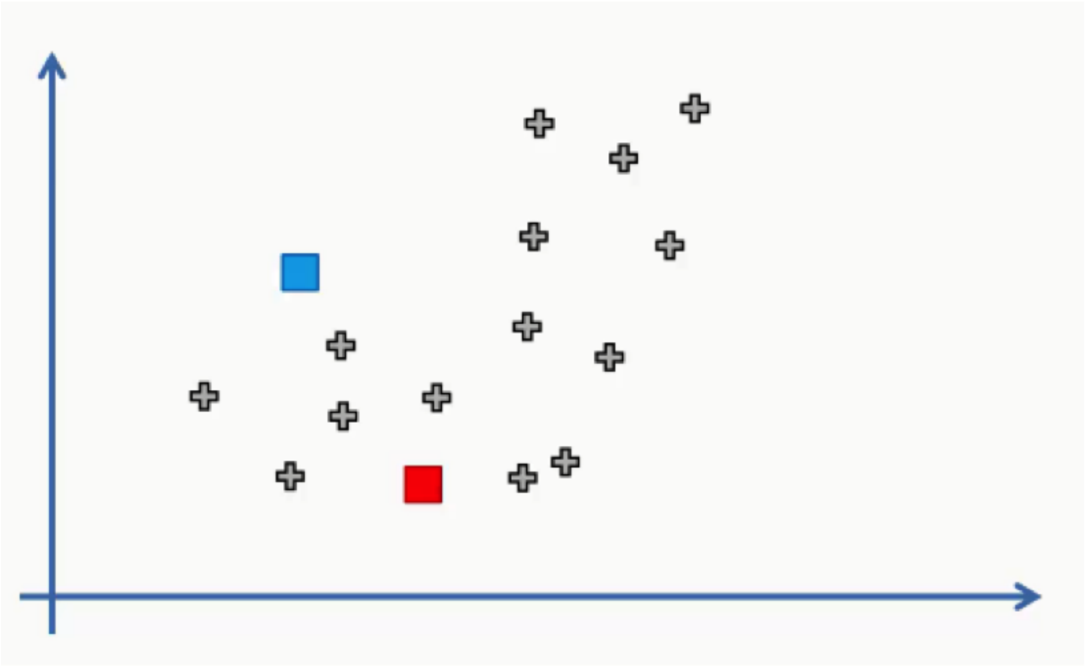
- The **cluster assignments** (z) stop changing.

Do the final z^* and μ^* minimize the objective?

- No ☹! Can be **local minima**.

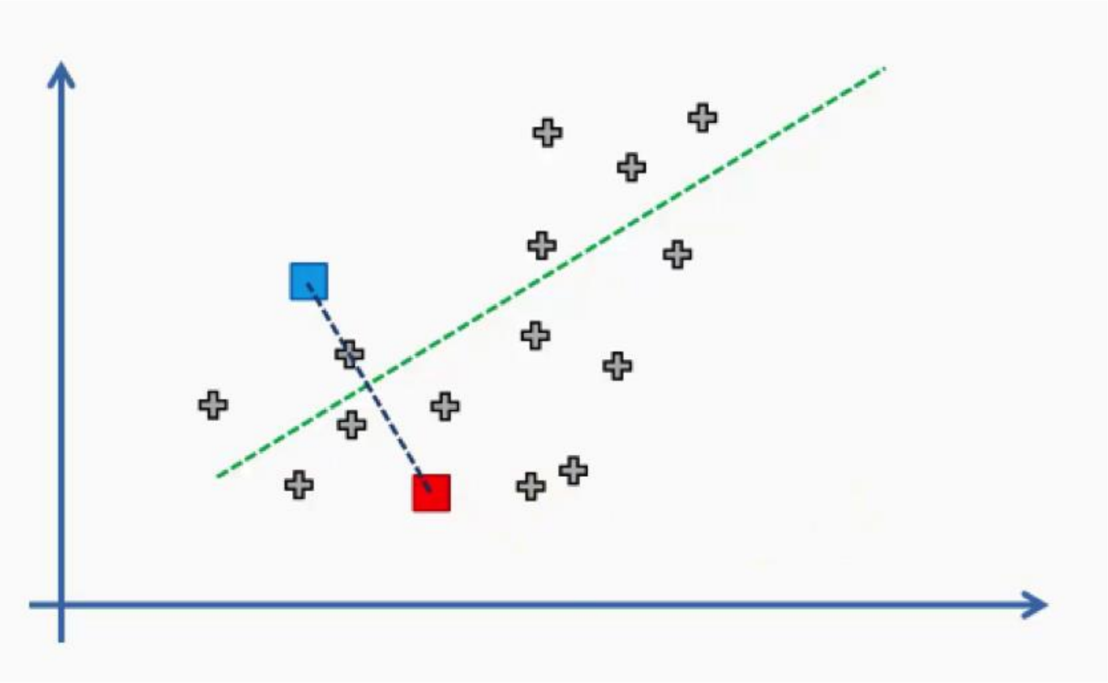
k-Means Clustering

1. Choose k points at random to be the cluster centers



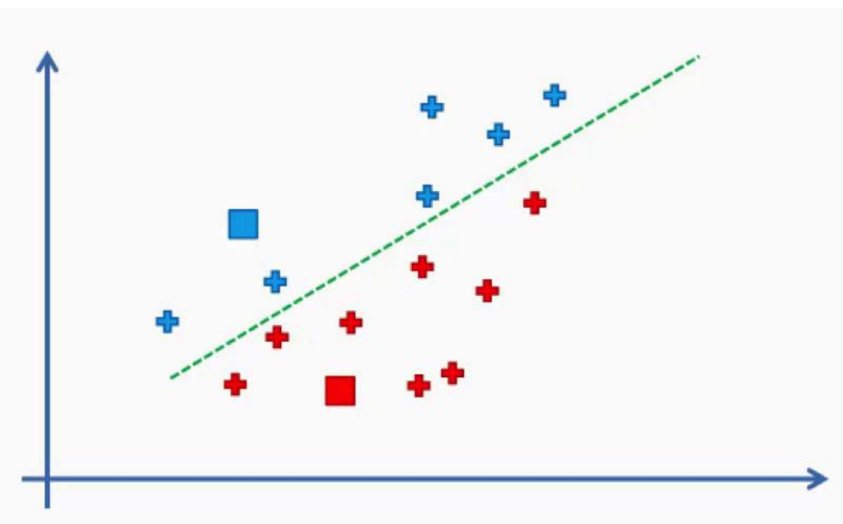
k-Means Clustering

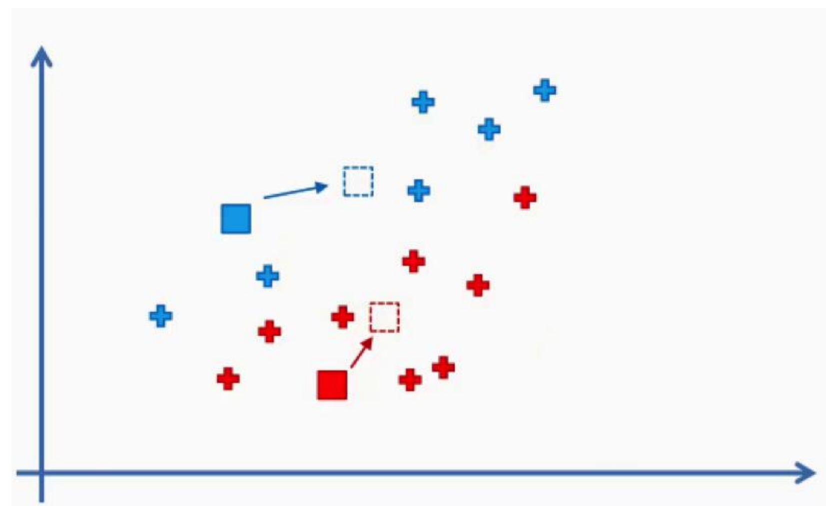
2. Assign each sample to its nearest cluster center



k-Means Clustering

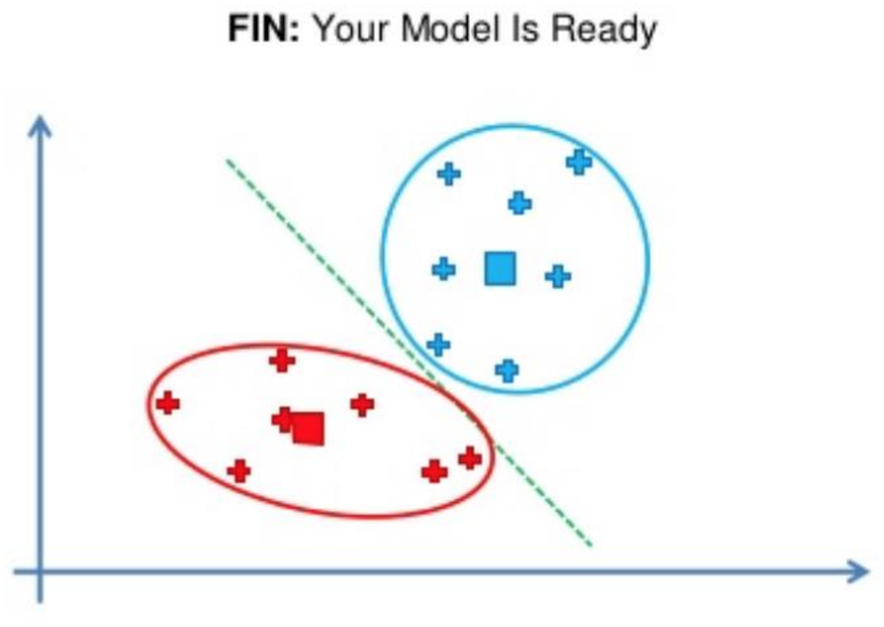
2. Recompute cluster centers by averaging assigned points





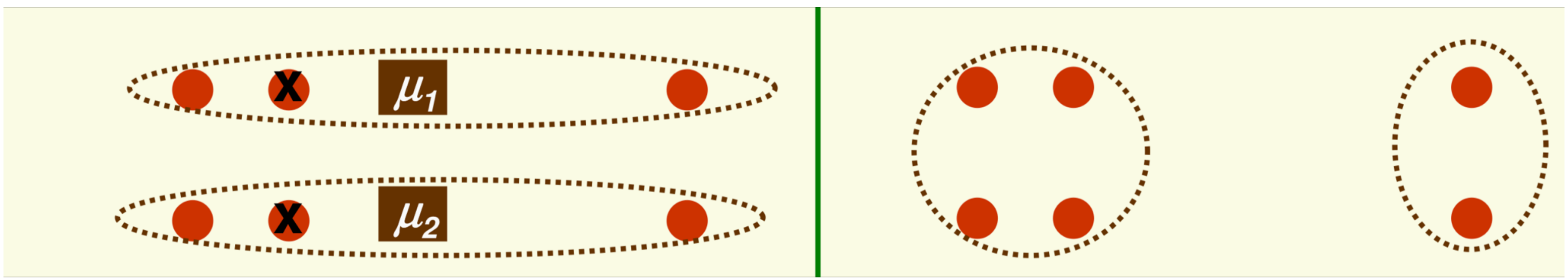
k-Means Clustering

Convergence:

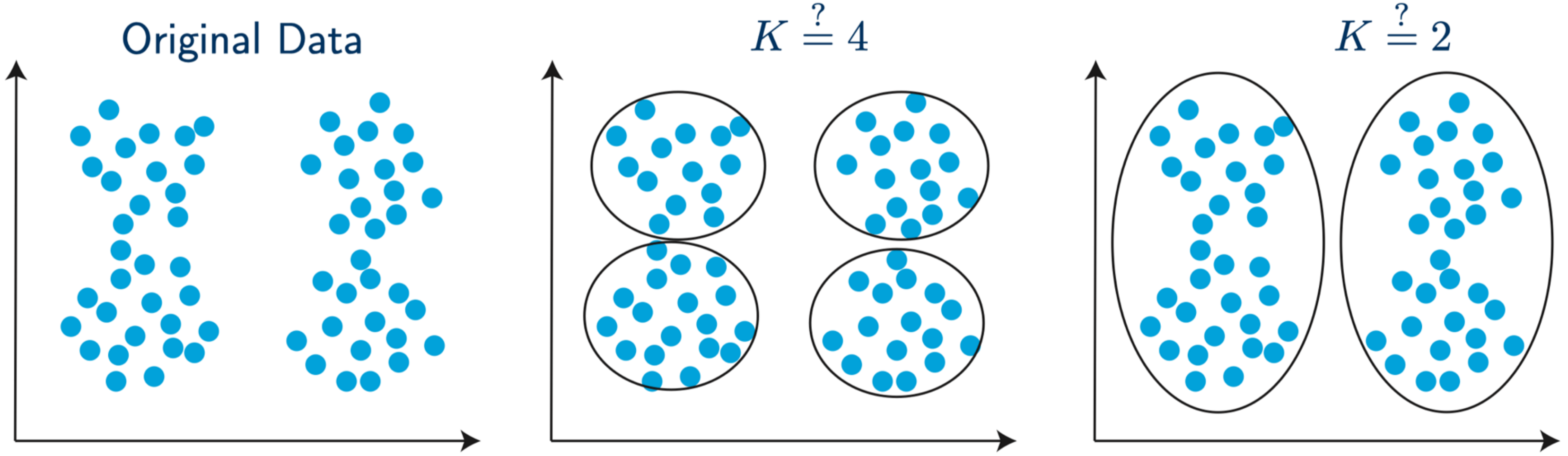


k-Means Clustering

It doesn't always work well.



How to determine number of clusters?



How to determine optimal number of clusters?

Silhouette Score:

Measures how well each point fits in its cluster vs. neighboring clusters.

Silhouette score for a point:

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

- a = average distance to points in same cluster
- b = average distance to closest other cluster

Score range:

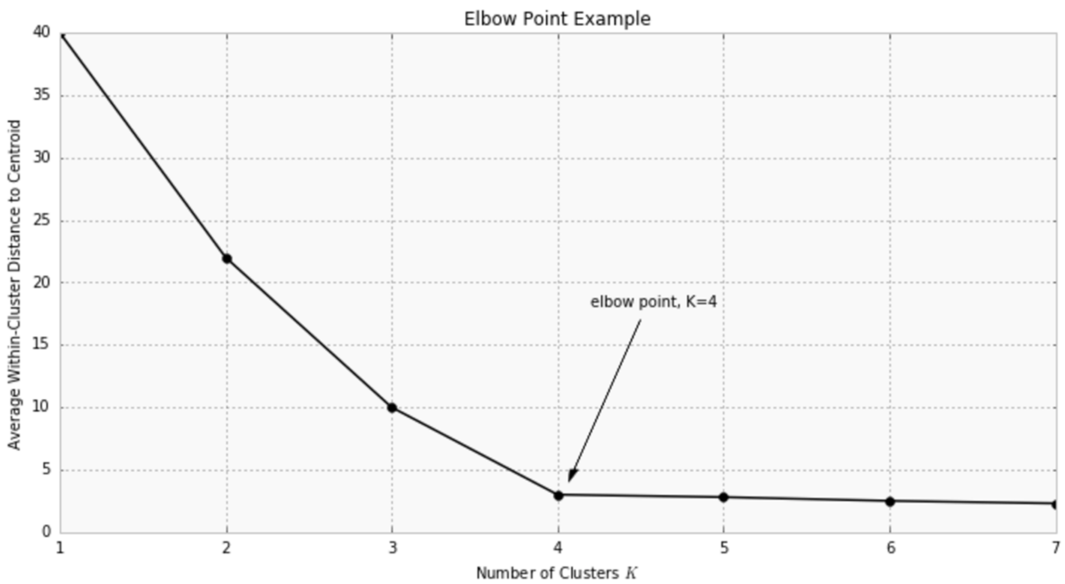
- ~1 → well-clustered
- ~0 → overlapping
- < 0 → probably wrong cluster

Choose k with highest average silhouette score.

How to determine optimal number of clusters?

Elbow Method :

- Compute **within-cluster sum of squared errors (WCSS)** for different k values.
- Plot k vs. WCSS. Look for an elbow



Elbow Method

Steps:

1. Run K-means for k = 1, 2, ..., Kmax.
2. Compute:

$$\mathrm{WCSS}(k) = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

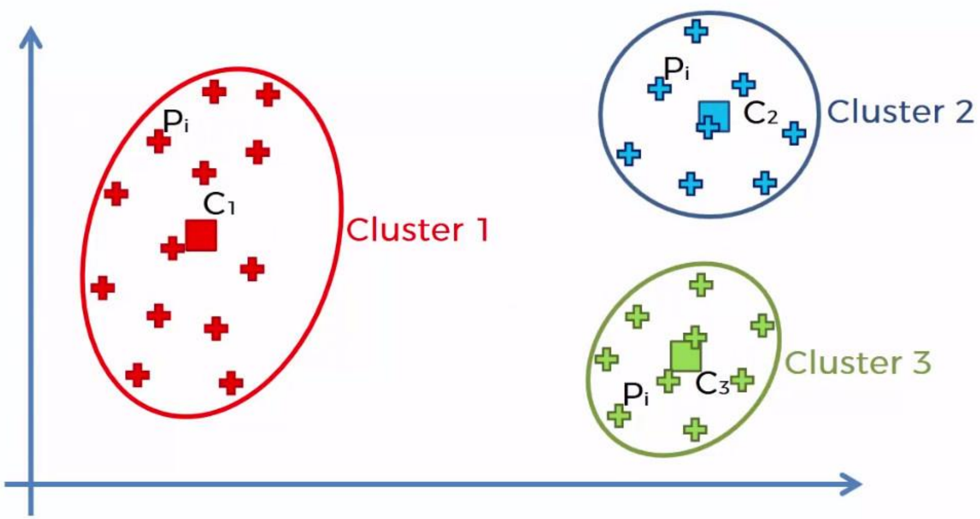
3. Choose the k at the “bend”.

Good for:

- Well-separated clusters

Elbow Method

- WCSS : within clusters sum of errors :



$$\mathrm{WCSS} = \sum_{\mathrm{P_i\ in\ Cluster\ 1}} \mathrm{distance}(\mathrm{P_i}, \mathrm{C_1})^2 + \sum_{\mathrm{P_i\ in\ Cluster\ 2}} \mathrm{distance}(\mathrm{P_i}, \mathrm{C_2})^2 + \sum_{\mathrm{P_i\ in\ Cluster\ 3}} \mathrm{distance}(\mathrm{P_i}, \mathrm{C_3})^2$$

Elbow Method

If all the samples are in different clusters, the error will be zero.