

# Unsupervised Learning

## K-Means, GMMs

Machine Learning Course - CS-433

Nov 20, 2024

Nicolas Flammarion

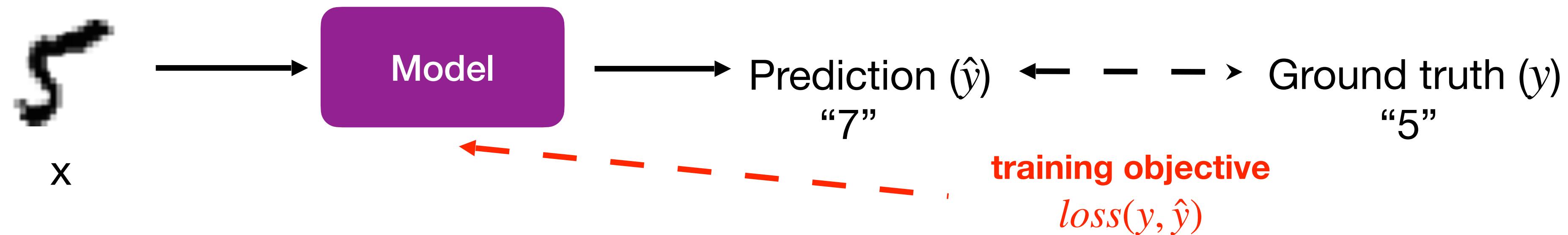


# Supervised Learning

We are given labeled training data  $\mathcal{D} = \{(x_n, y_n)\}_{n \in [N]}$

MNIST Dataset	
Image(x)	Label(y)
5	"5"
7	"7"
3	"3"

The **training objective** is usually the alignment of the model predictions and the given labels



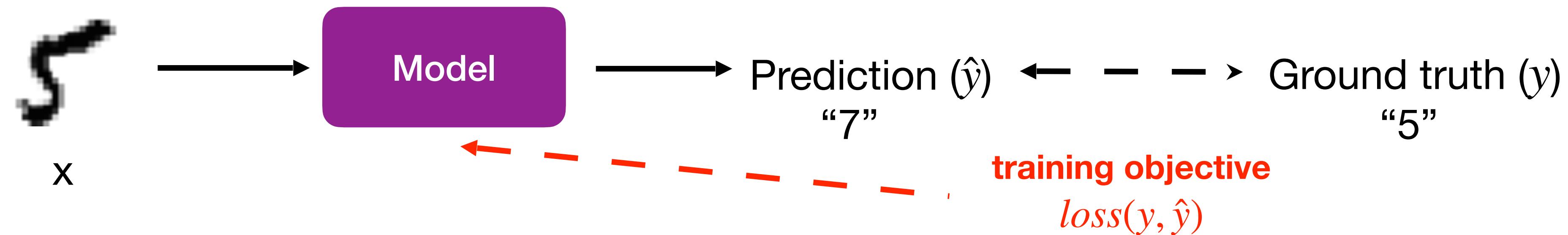
# Supervised Learning

We are given labeled training data  $\mathcal{D} = \{(x_n, y_n)\}_{n \in [N]}$

Labeled data is **costly** and **time-consuming** to obtain, but unlabeled data is **abundant** (e.g. text and images online)

MNIST Dataset	
Image(x)	Label(y)
5	"5"
7	"7"
3	"3"

The **training objective** is usually the alignment of the model predictions and the given labels



# Unsupervised Learning

We are given only unlabeled training data  $\mathcal{D} = \{x_n\}_{n \in [N]}$

**What can we still learn from this data?**

Main learning goals are

- Learning a compact and useful representation of the data
- Density estimation and generative modelling

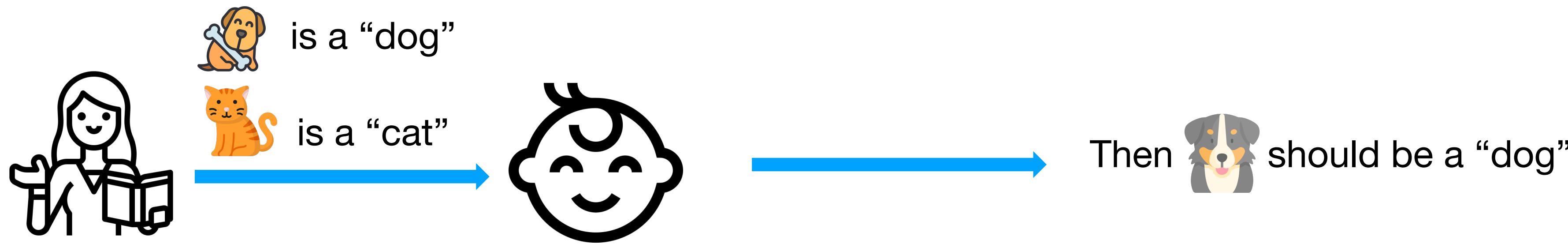
Useful for

- Exploratory data analysis (finding patterns in the data) and anomaly detection
- Learning good features/embeddings for various tasks
- Generate new observations

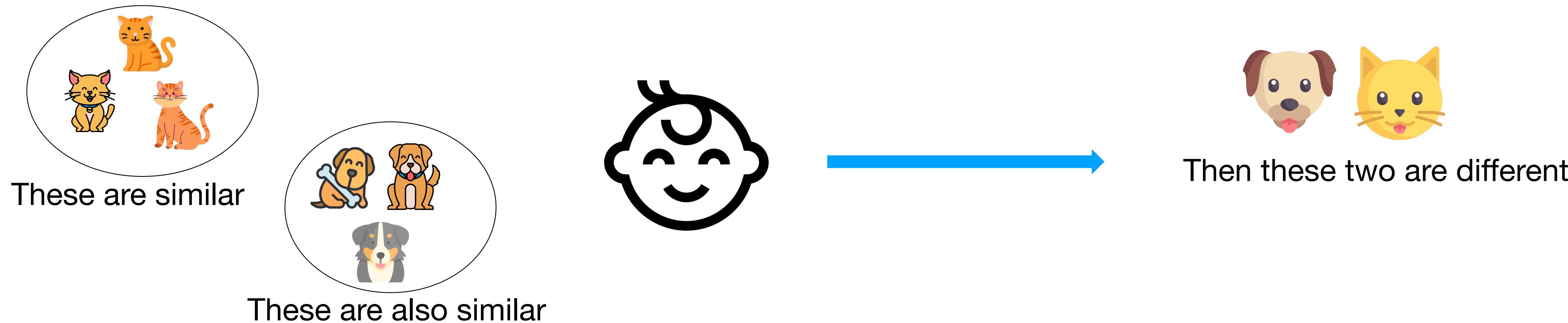
Dataset
Image(x)
5
7
3

# Unsupervised Learning

## Supervised Learning: The Teacher-Guided Approach

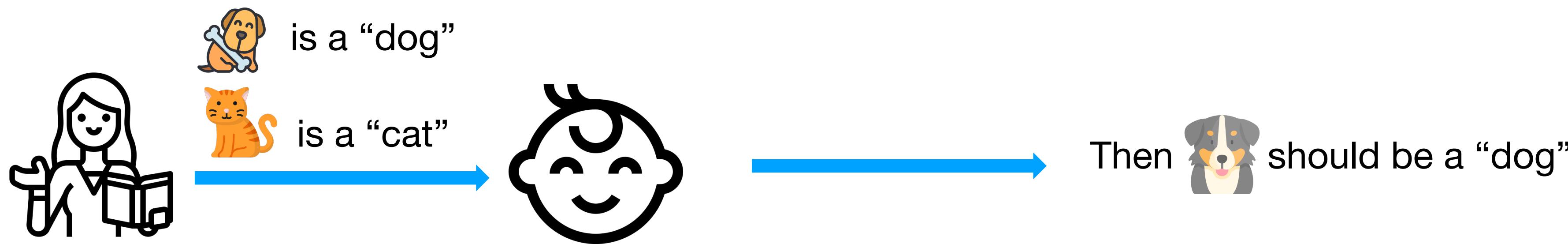


## Unsupervised Learning: The Self-Discovery Approach

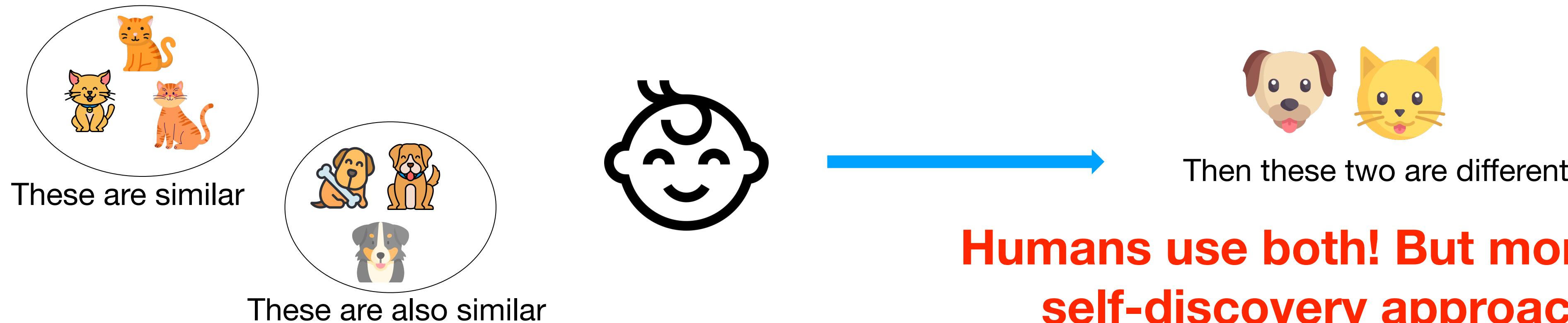


# Unsupervised Learning

## Supervised Learning: The Teacher-Guided Approach



## Unsupervised Learning: The Self-Discovery Approach



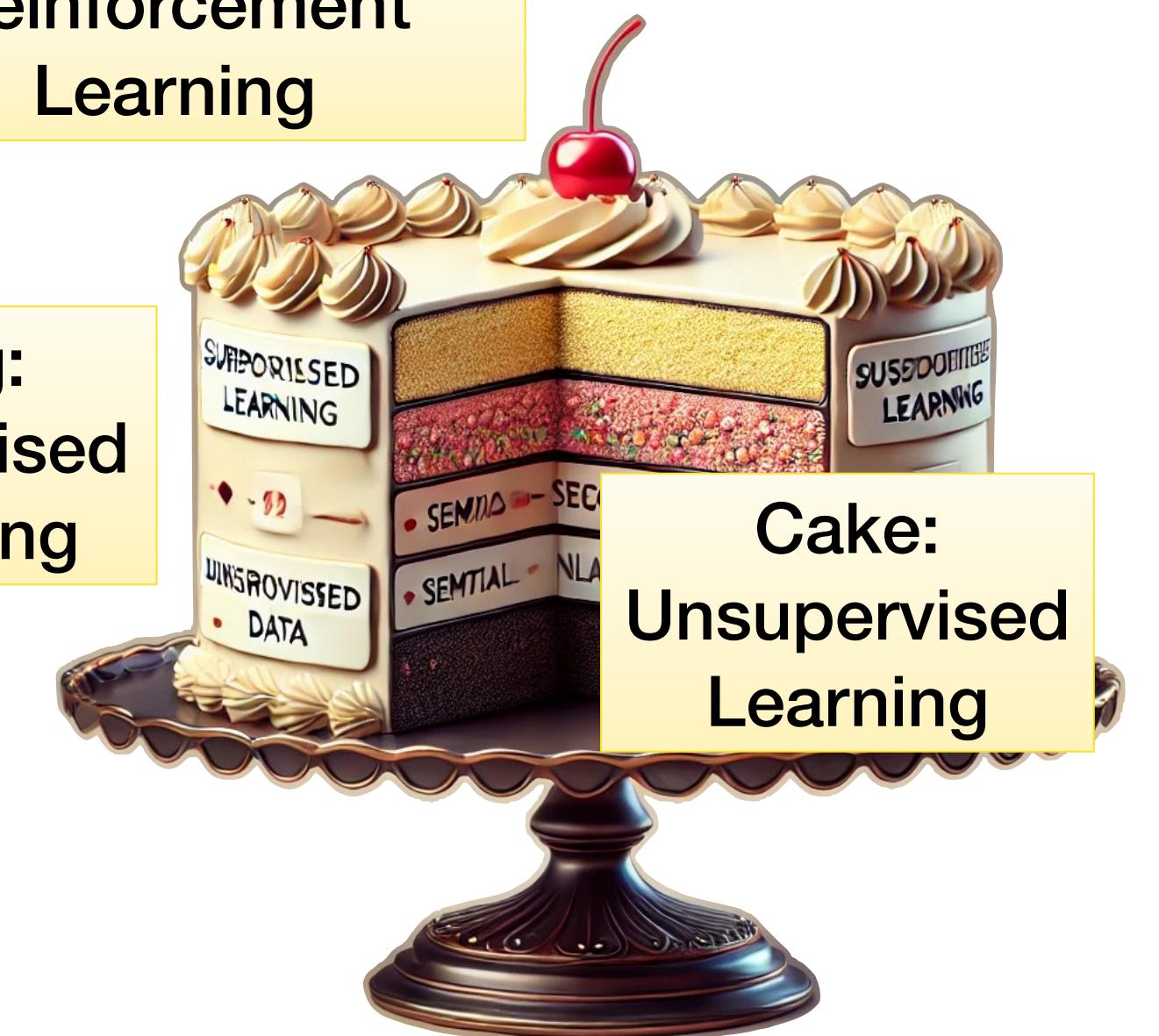
# Deep Learning and Cake Analogy

Unsupervised learning leverages vast amounts of unstructured data without the need for labeled examples

Reinforcement  
Learning

*"Most of human and animal learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake."*

Icing:  
Supervised  
Learning



NeurIPS 2016

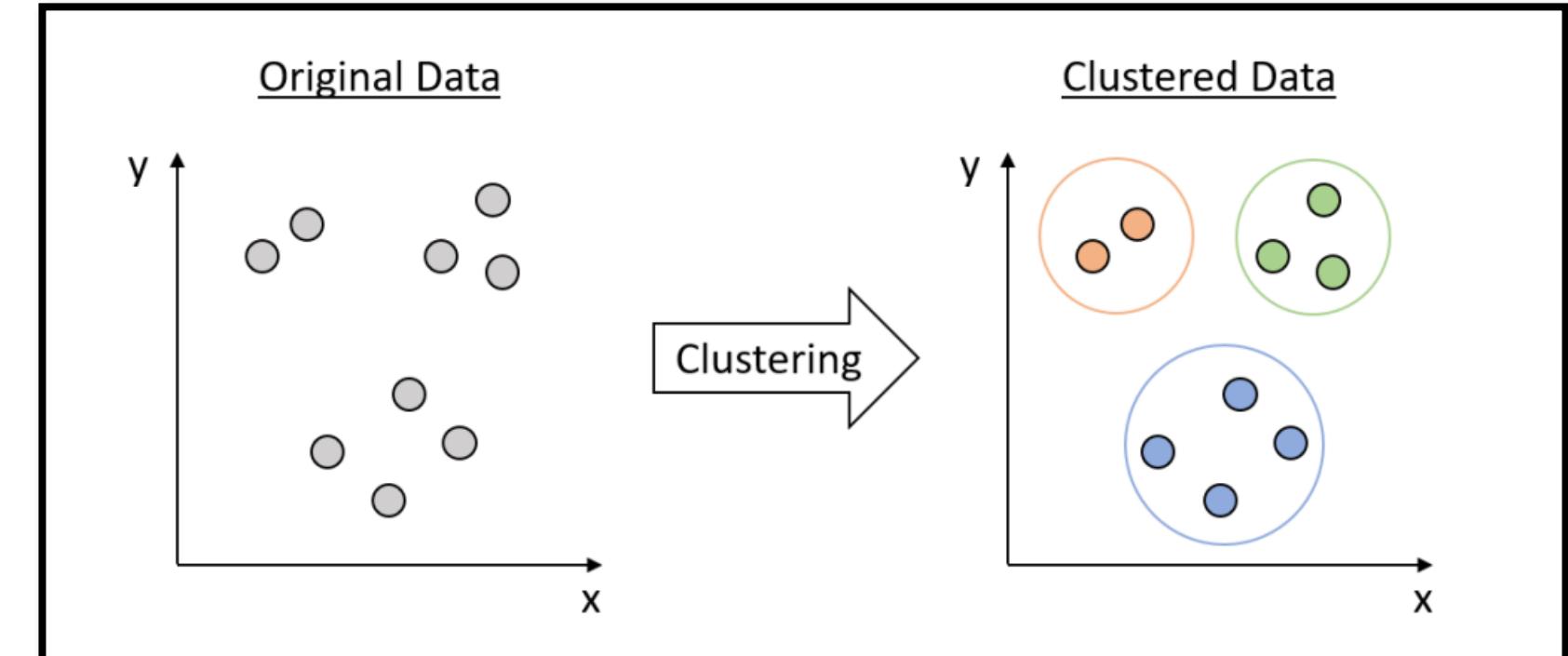


Yann Le Cun

# Unsupervised Learning Examples

**Clustering:** Group the data points into clusters based on their similarities

- K-means, Gaussian Mixture Models (GMMs)

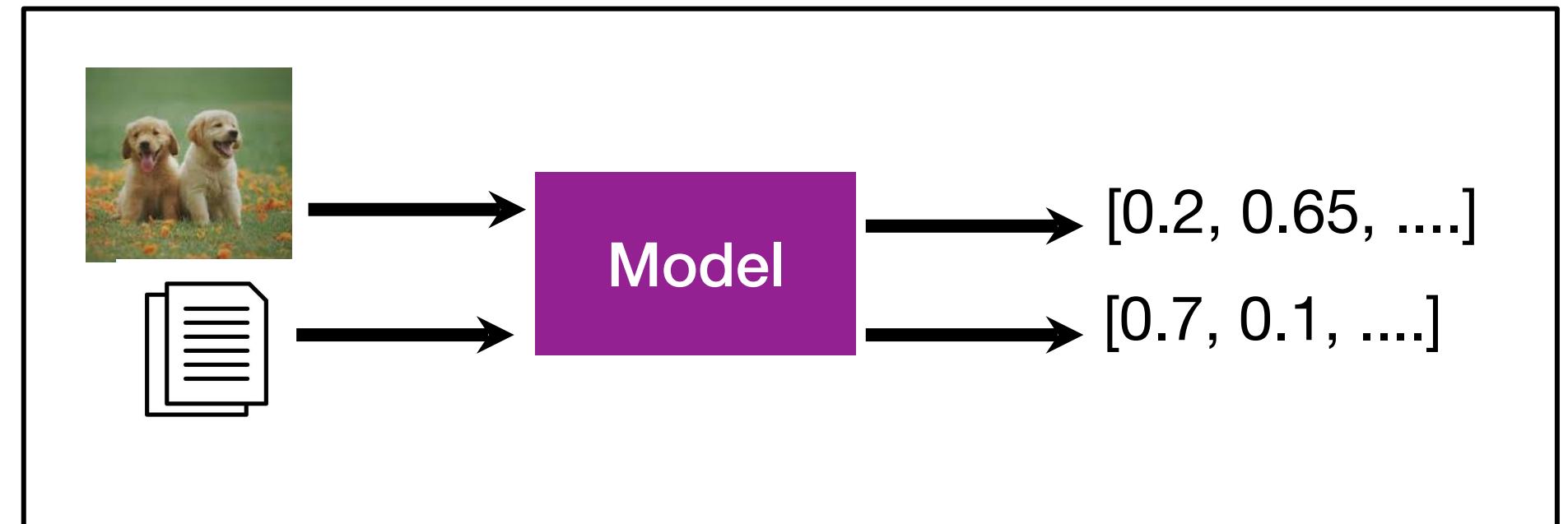


**Representation Learning:** Converting raw data into good numerical features

- PCA
- Matrix factorization, Word2Vec, GloVe, FastText
- BERT
- Contrastive learning (CLIP, SimCLR)

**Generative Modelling**

- Gaussian Mixture Models (GMMs)
- Large Language Models (LLMs) for text generation (e.g. GPT)
- GANs and Diffusion models for image generation

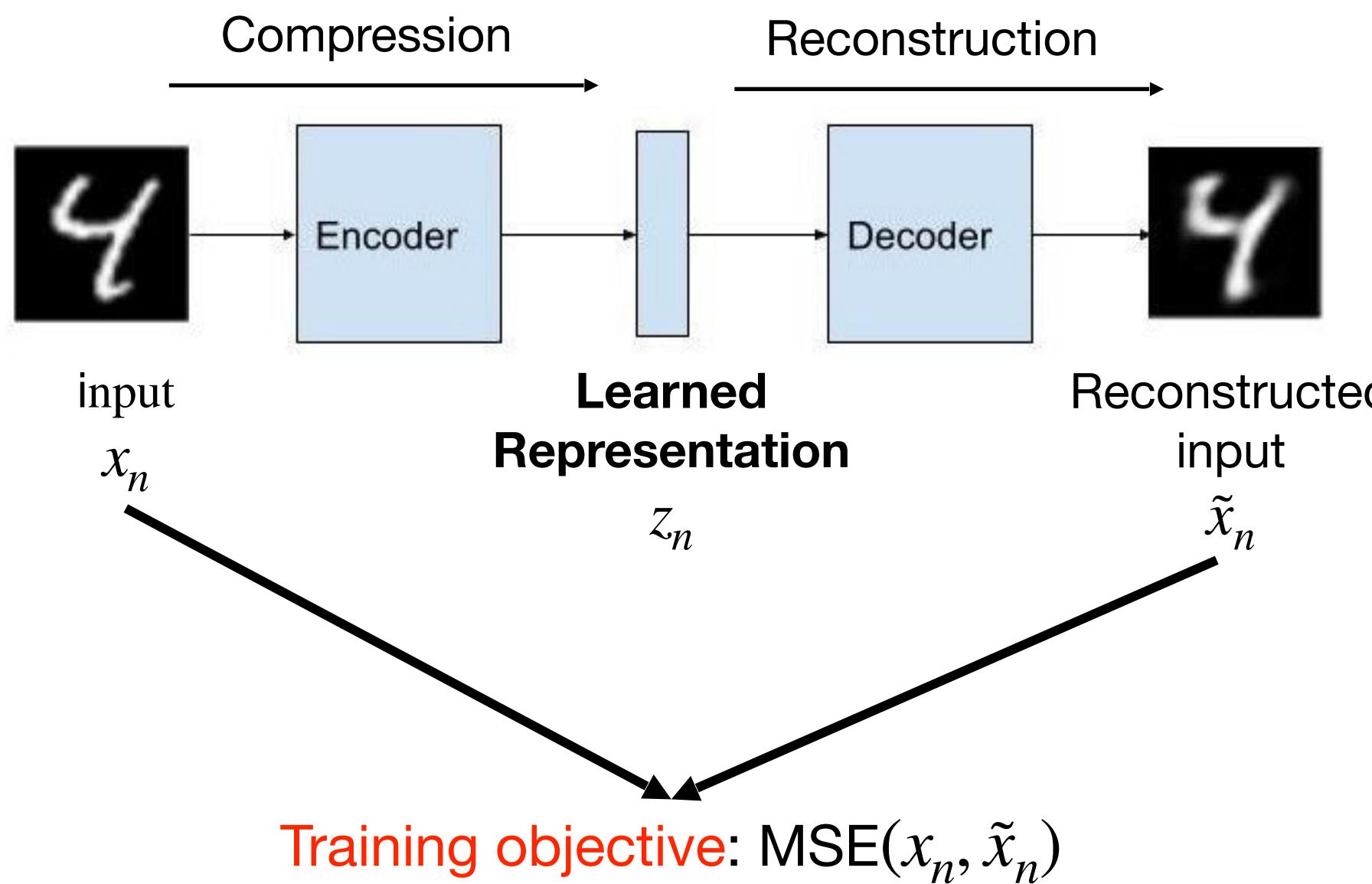


# Unsupervised Learning

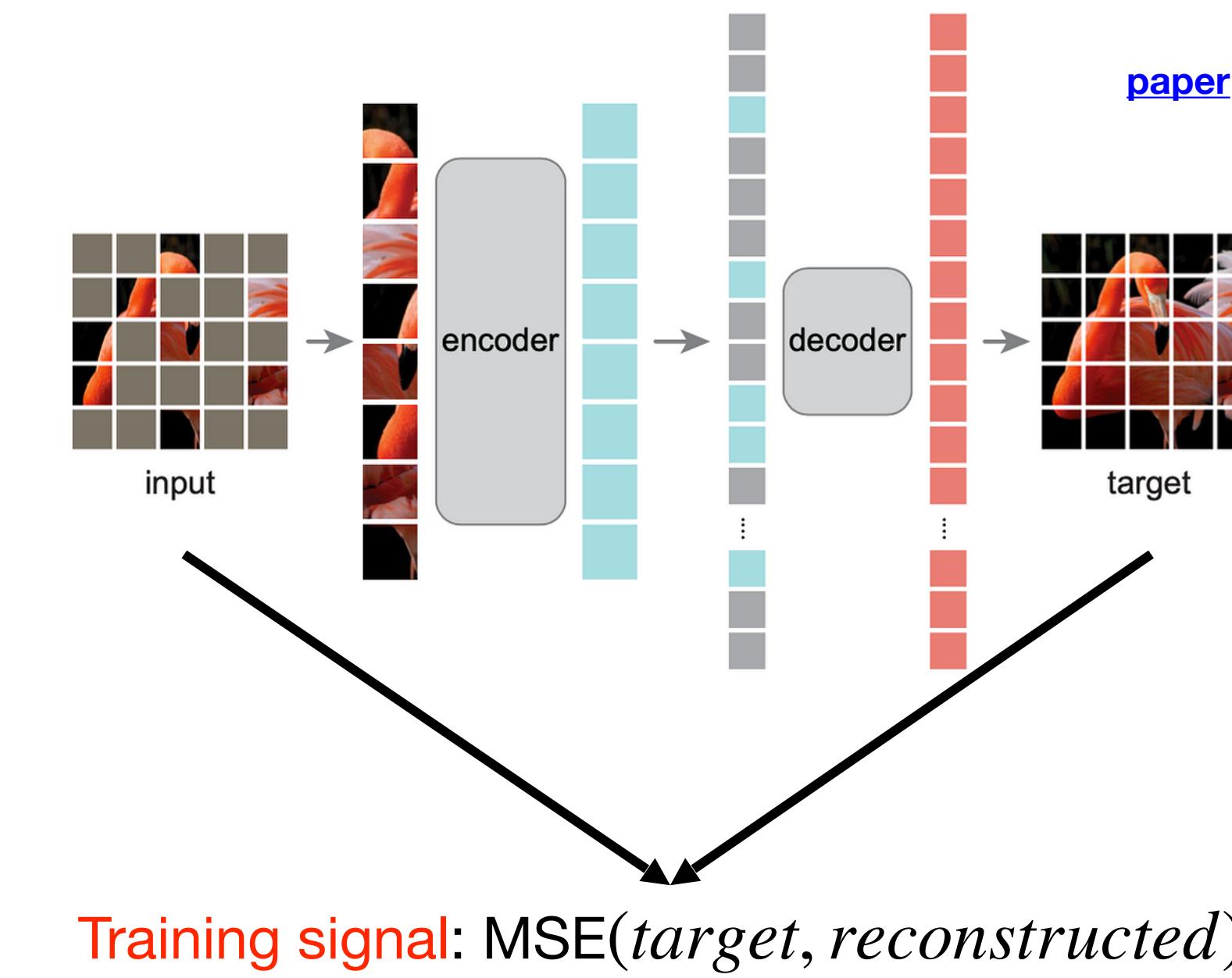
We can seek many ideas to find a **training objective**.

Two examples:

**AutoEncoders**



**Masked AutoEncoders**



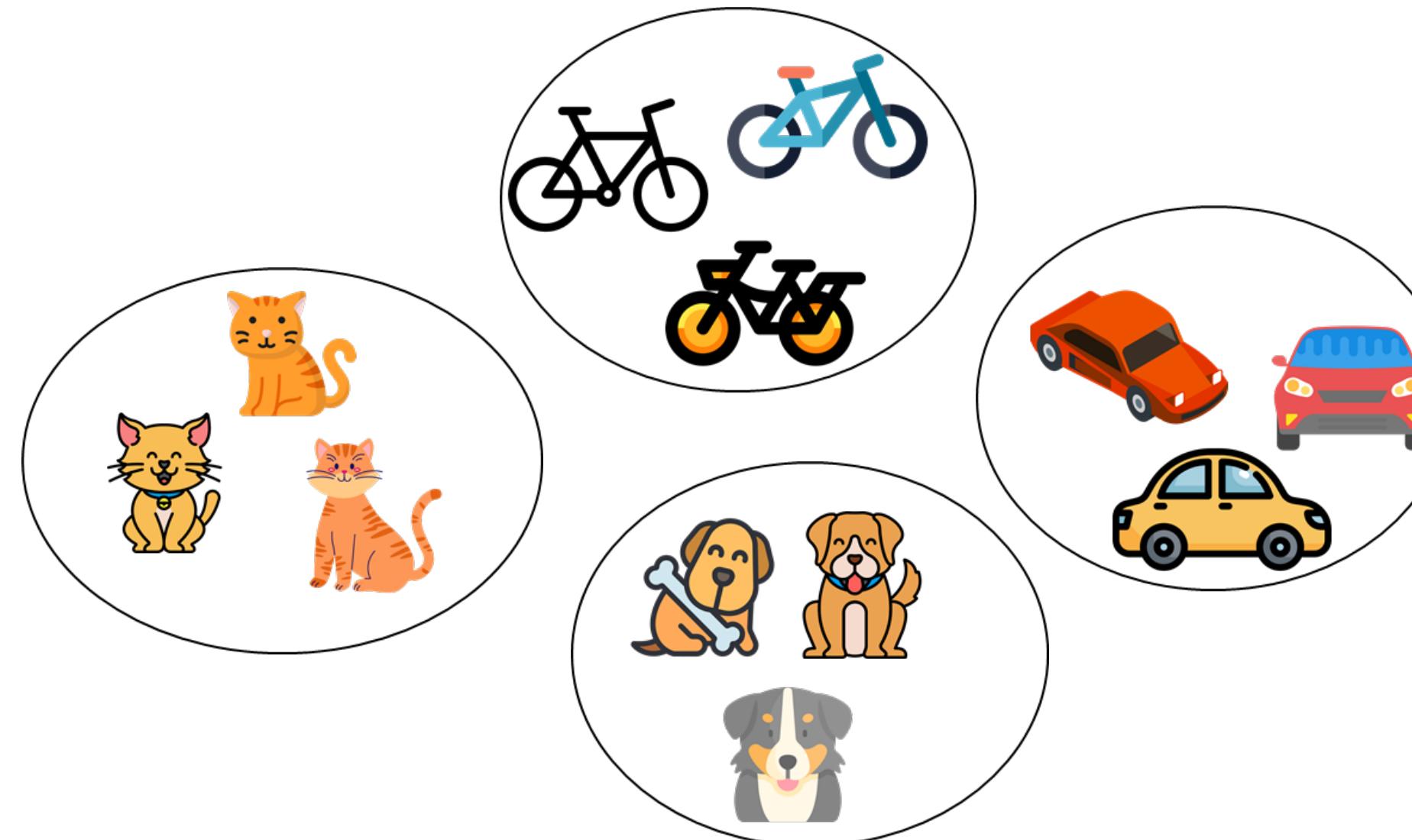
# Clustering

# Clustering

One of the most widely used techniques for exploratory data analysis

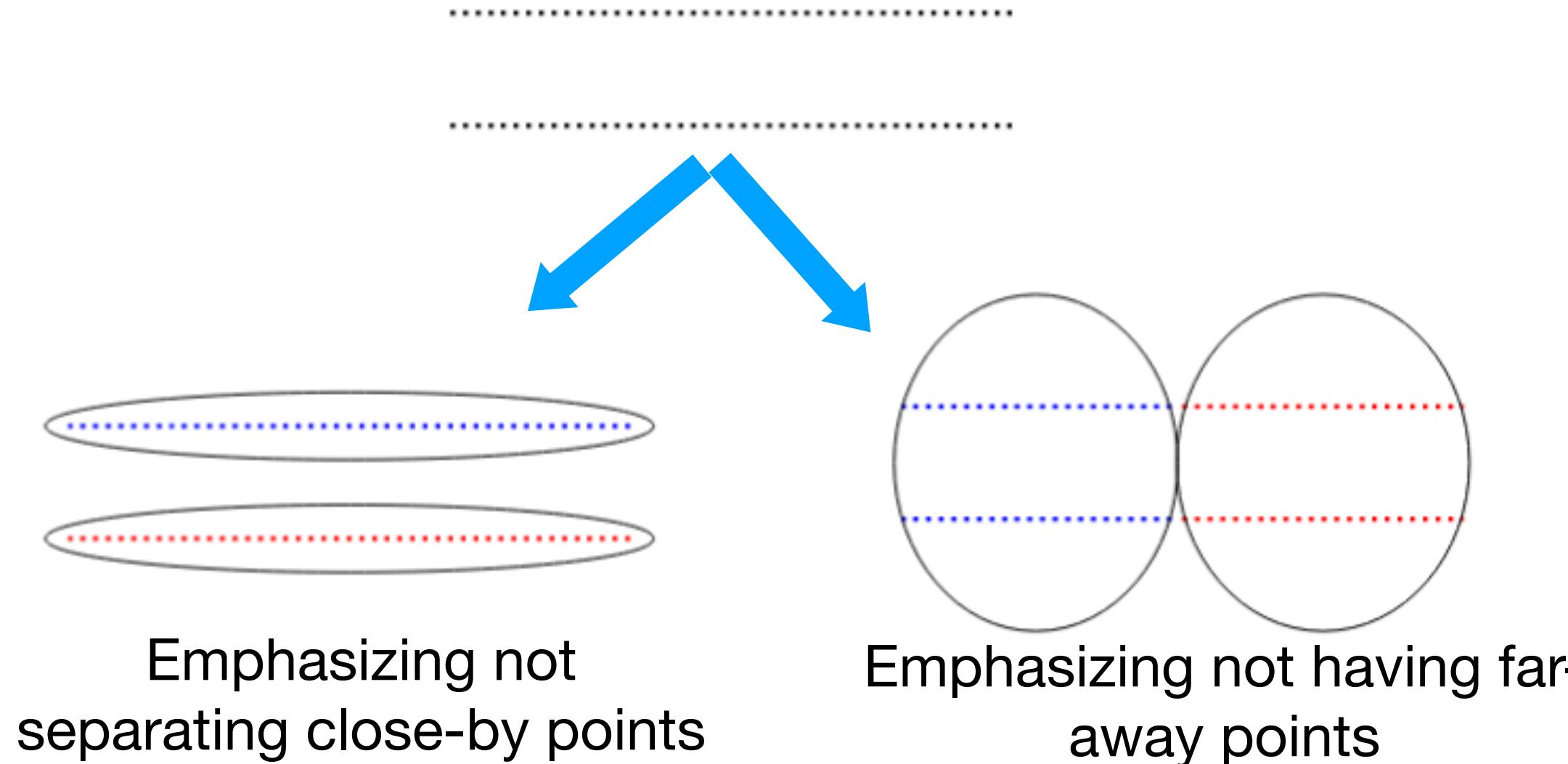
Clustering is the task of grouping a set of objects such that:

- *Similar* objects end up in the same group
- *Dissimilar* objects are separated into different groups

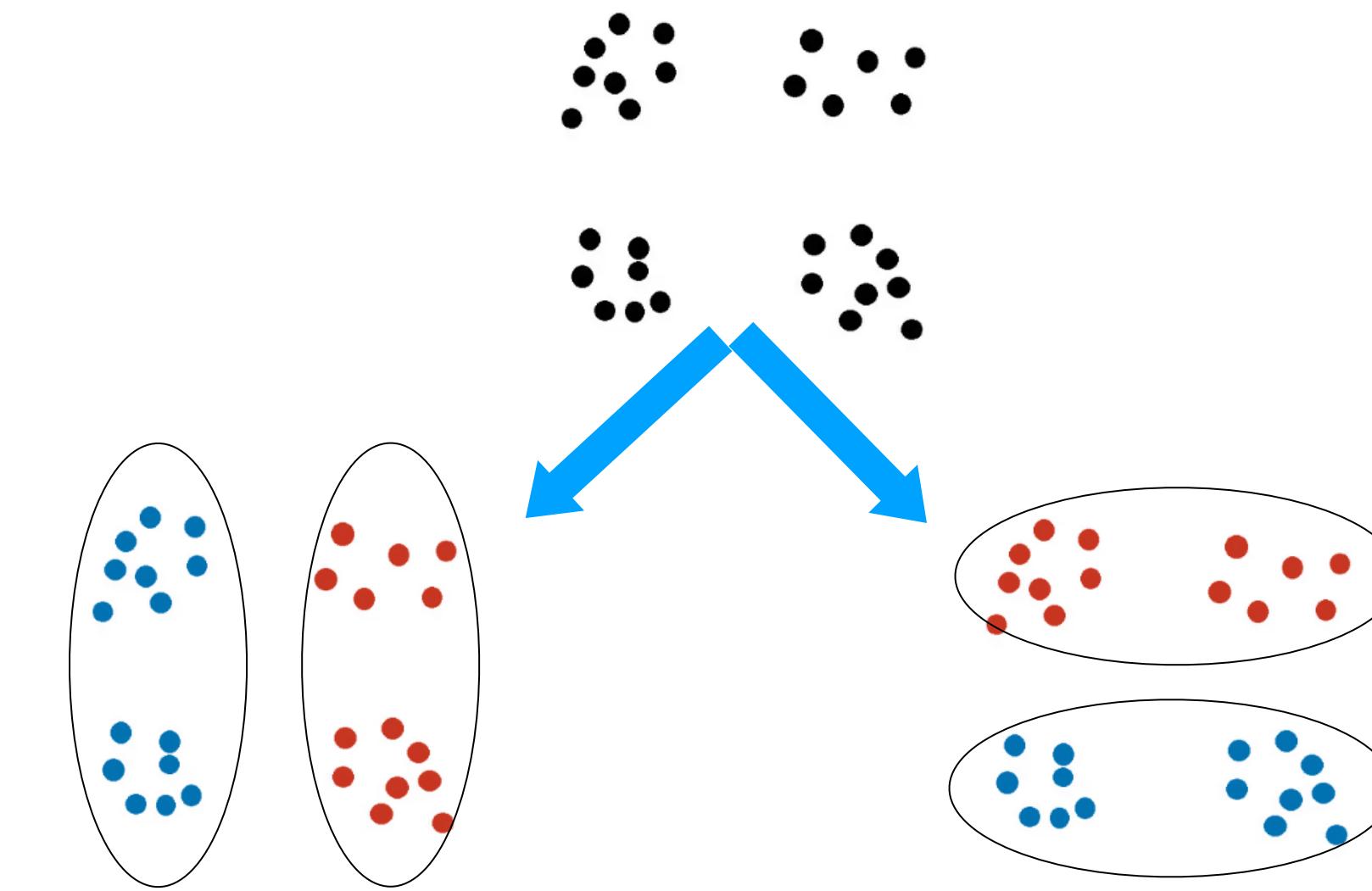


# Challenges in Clustering

## Ambiguity of what is a cluster



## Lack of “ground truth”

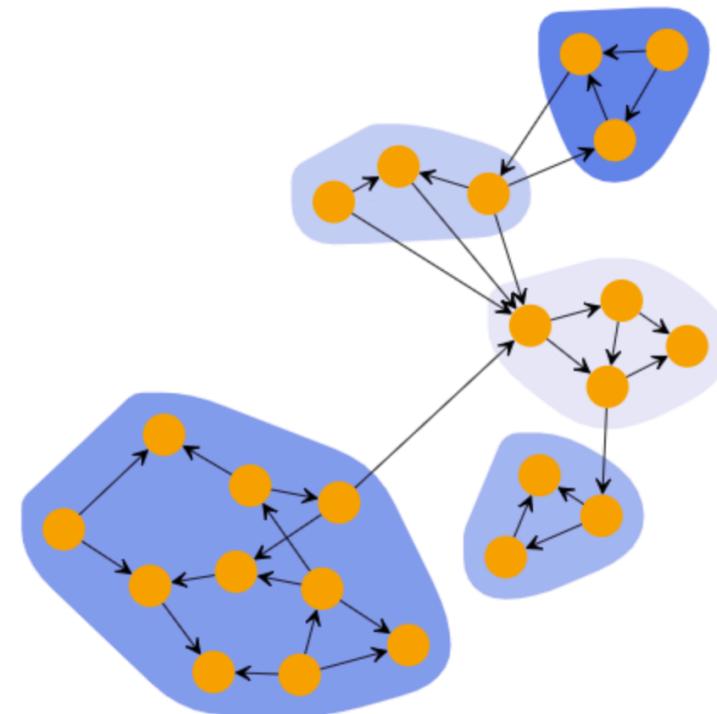


**It is difficult to evaluate the success**

# Different Approaches for Clustering

## Graph-based clustering

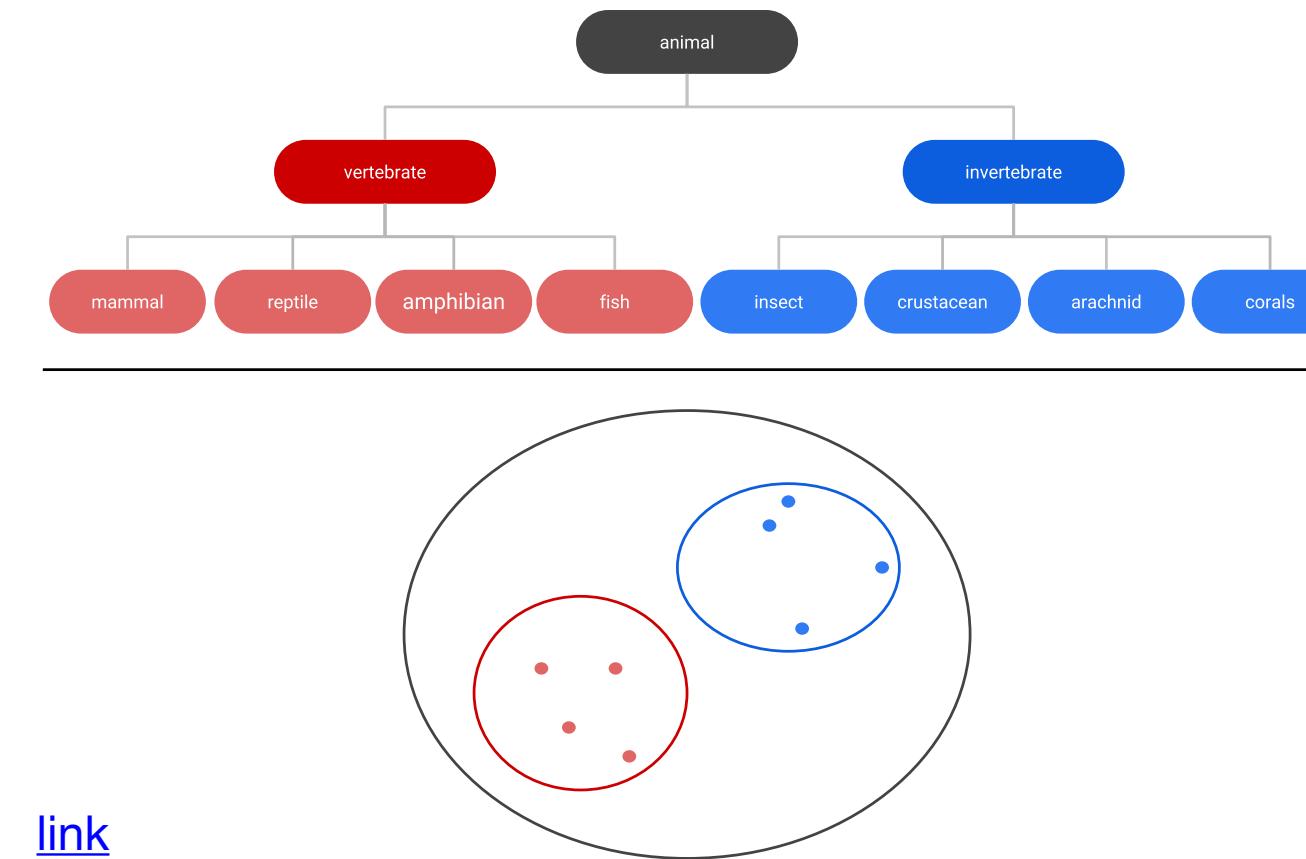
Graph partition to minimize inter-cluster weights and maximize intra-cluster weights



**Examples:** Spectral clustering, graph-cut based approaches

## Hierarchical clustering

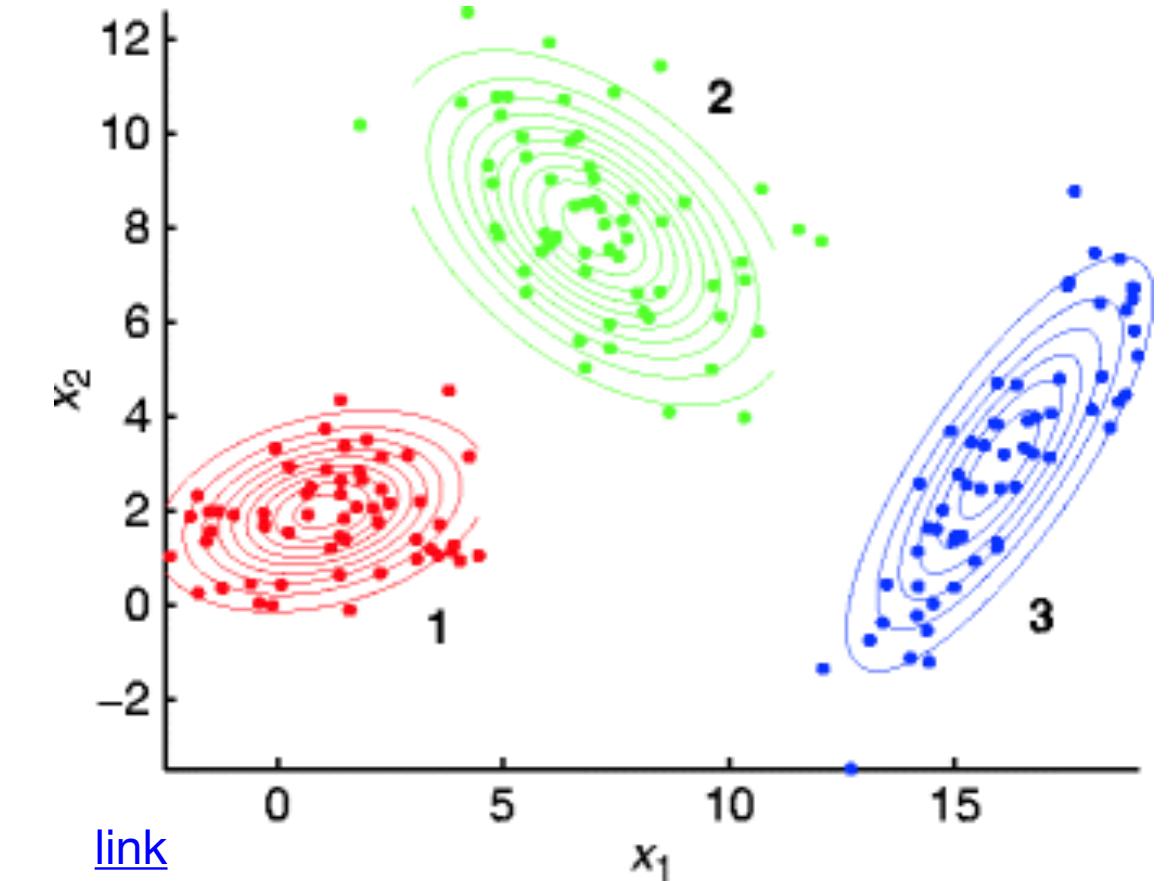
Build a tree (bottom-up or top-down), representing distances among data points



**Examples:** Linkage-based approaches

## Model-based clustering

Maintain cluster models and infer memberships based on them

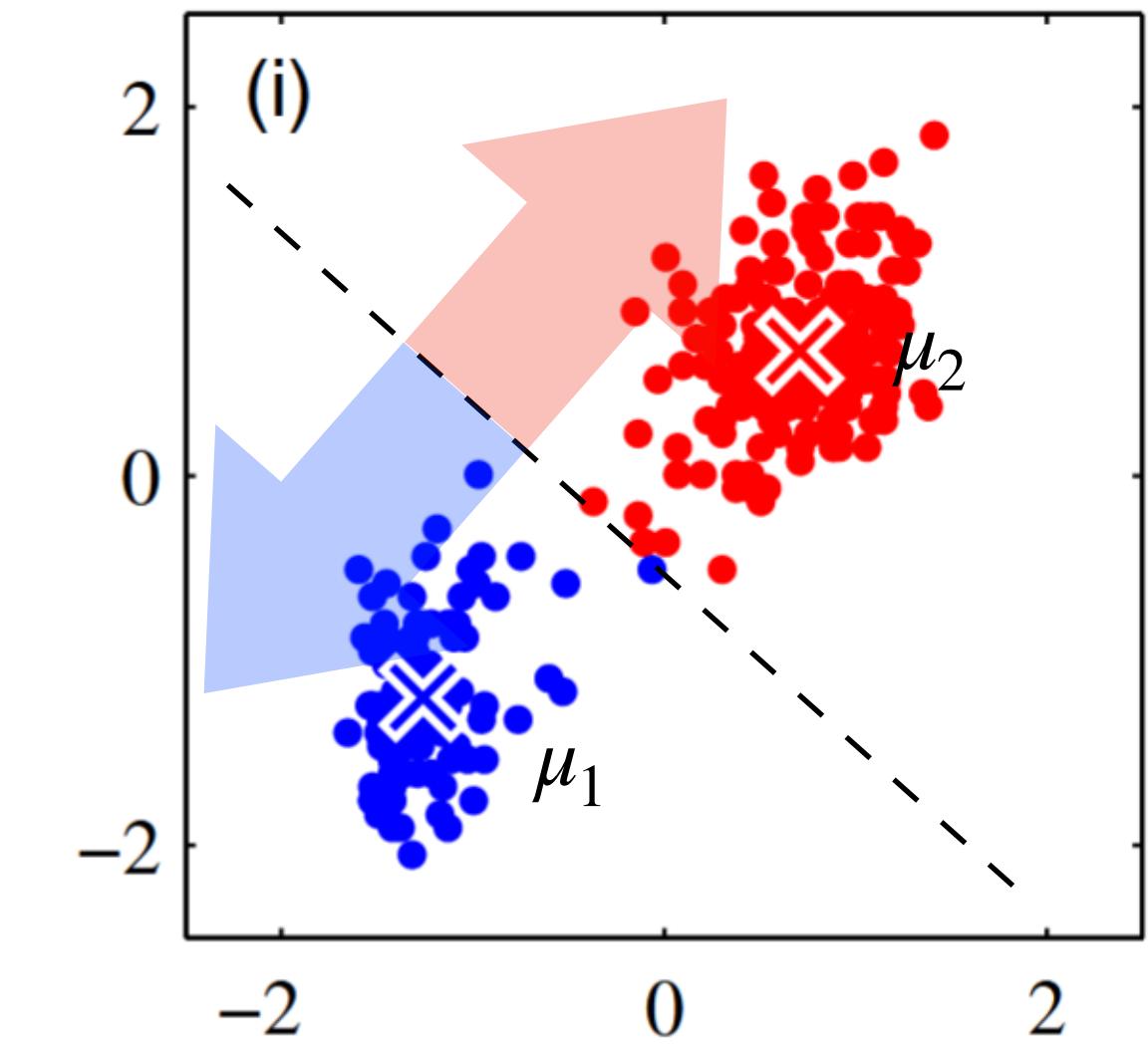


**Examples:** K-means, GMMS

# K-Means

**K-Means** is an **unsupervised learning algorithm** used for **clustering**

**It aims to partition data into K clusters, where each point belongs to the cluster with the closest mean**



Key Concepts:

- **Clusters:** Groups of data points with similar characteristics
- **Centroids:** The center of each cluster, representing the "mean" position of data points in the cluster ( $\mu_1, \mu_2$ )

# K-Means

## Input:

- A dataset  $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  with  $N$  datapoints, each in  $\mathbb{R}^D$
- Number of clusters  $K$

## Output:

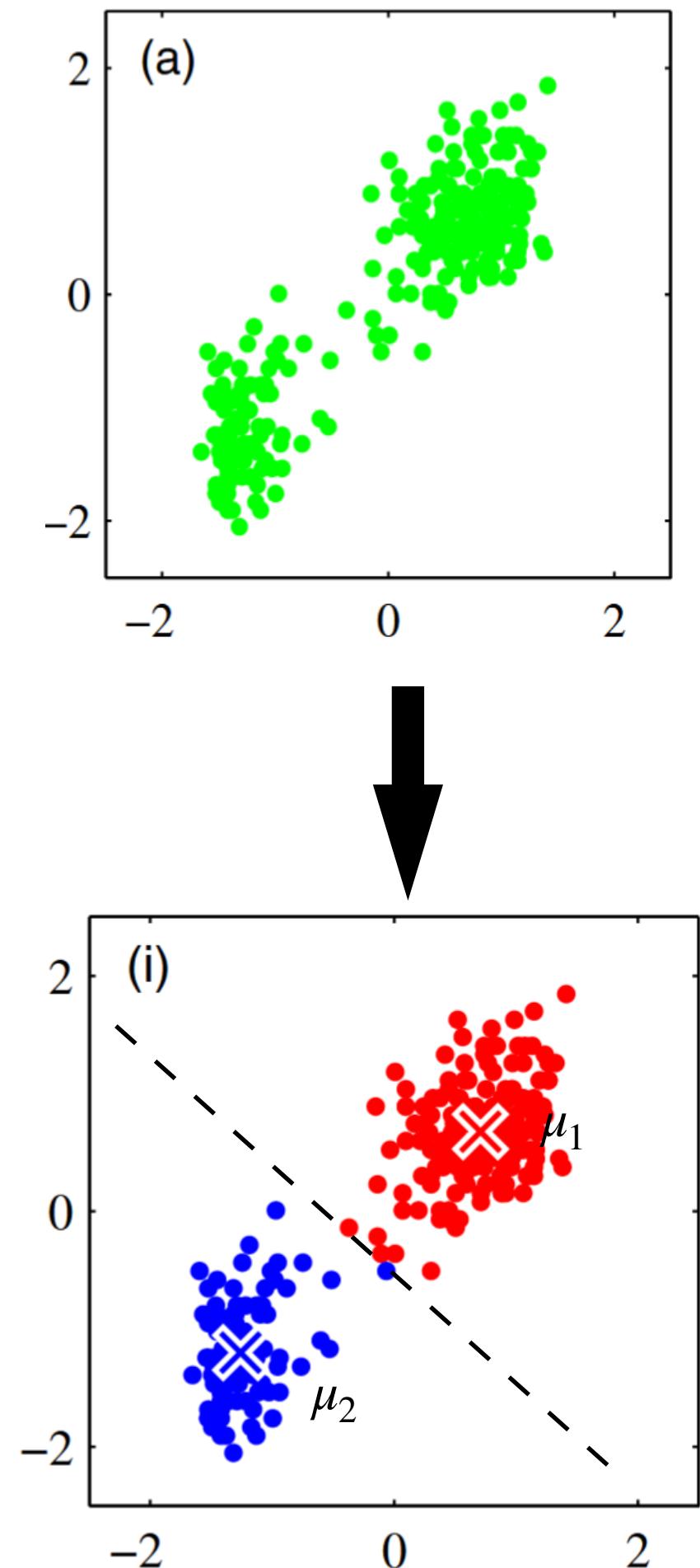
- Cluster centroids  $\mu_1, \mu_2, \dots, \mu_K$
- Cluster assignments  $z_{nk} \in \{0,1\}$  for each datapoint

## Objective function: Within-cluster sum of squares (WCSS)

$$\min_{\mathbf{z}, \mu} \mathcal{L}(\mathbf{z}, \mu) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \mu_k\|_2^2$$

$$\text{s.t. } \mu_k \in \mathbb{R}^D, z_{nk} \in \{0,1\}, \sum_{k=1}^K z_{nk} = 1,$$

where  $\mathbf{z}_n = [z_{n1}, z_{n2}, \dots, z_{nK}]^\top$ ,  $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N]^\top$ , and  $\mu = [\mu_1, \mu_2, \dots, \mu_K]^\top$

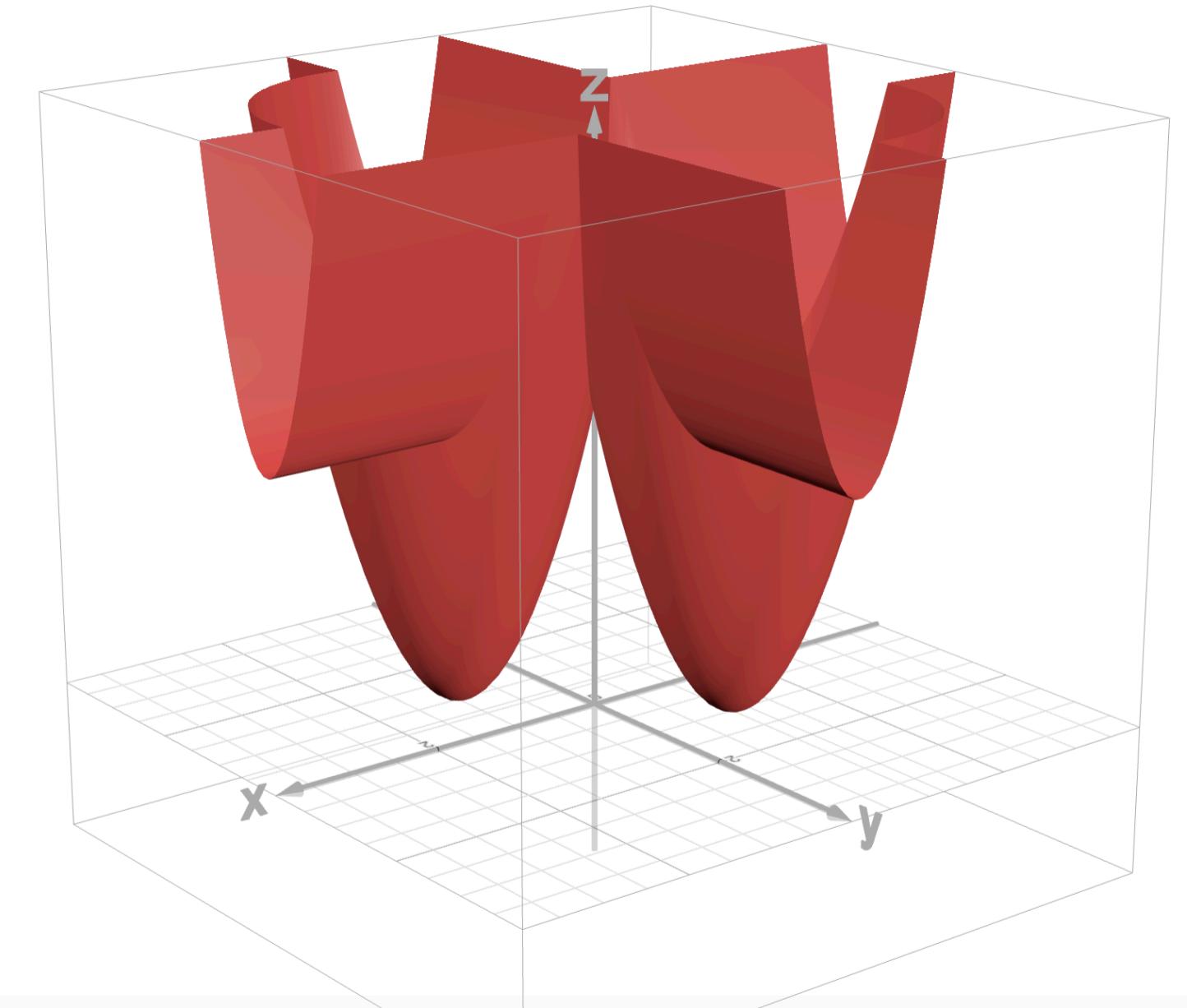


# K-Means Objective

$$\min_{\mathbf{z}, \mu} \mathcal{L}(\mathbf{z}, \mu) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \mu_k\|_2^2$$

s.t.  $\mu_k \in \mathbb{R}^D$ ,  $z_{nk} \in \{0,1\}$ ,  $\sum_{k=1}^K z_{nk} = 1$ ,

where  $\mathbf{z}_n = [z_{n1}, z_{n2}, \dots, z_{nK}]^\top$ ,  $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N]^\top$ ,  
and  $\mu = [\mu_1, \mu_2, \dots, \mu_K]^\top$



- The objective is **non-convex** and there are multiple local minima.
- Some of the optimization variables are **discrete**

# K-Means Algorithm

**Input:** dataset  $\mathcal{D}$ ; Number of clusters  $K$

**Initialize:** Randomly choose initial centroids

$$\mu_1, \mu_2, \dots, \mu_K$$

**Repeat until convergence:**

- Assignment step:

- $\forall n \in [N]$  compute  $z_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \mu_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$

Break ties  
arbitrarily!

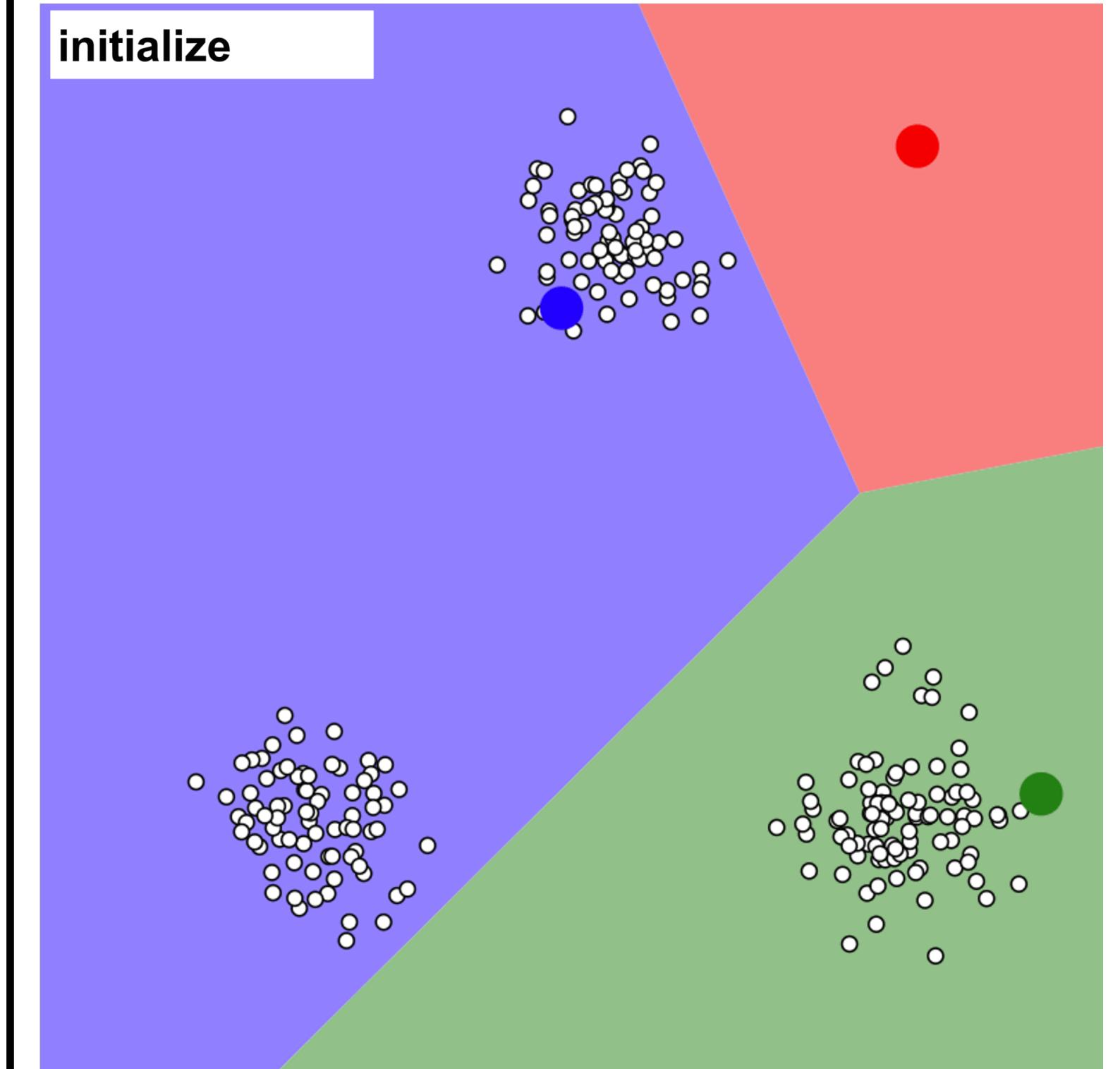
- Update step:

- $\forall k \in [K]$  update  $\mu_k = \frac{\sum_{n=1}^N z_{nk} \mathbf{x}_n}{\sum_{n=1}^N z_{nk}}$

Optimal assignment maps each point to its nearest center

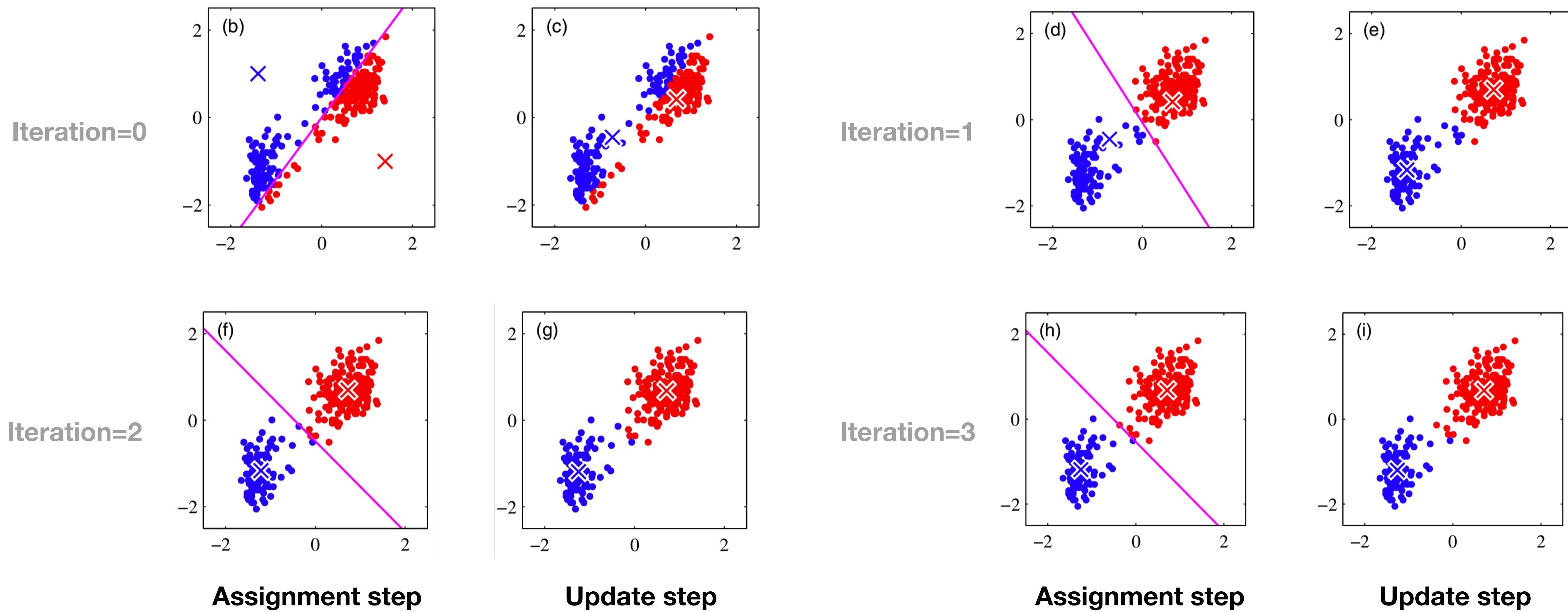


Optimal centers are the means of each cluster



Iteration cost:  $O(NKd)$

# K-Means Algorithm



# Monotonic Decrease of K-Means Objective

**Proposition:** Each iterations of the K-means algorithm decreases the K-means objective function:

$$\mathcal{L}(\mathbf{z}^{(t)}, \boldsymbol{\mu}^{(t)}) \leq \mathcal{L}(\mathbf{z}^{(t-1)}, \boldsymbol{\mu}^{(t-1)})$$

**Proof:**

Assignment step: Each data point is assigned to its nearest centroid (centroids are fixed)

$$\sum_{n=1}^N \sum_{k=1}^K z_{nk}^{(t-1)} \|\mathbf{x}_n - \boldsymbol{\mu}_k^{(t-1)}\|_2^2 \geq \sum_{n=1}^N \sum_{k=1}^K z_{nk}^{(t)} \|\mathbf{x}_n - \boldsymbol{\mu}_k^{(t-1)}\|_2^2$$

Update step: The mean minimizes the sum of squared Euclidean distances to the points in the cluster (assignments are fixed)

$$\sum_{n=1}^N \sum_{k=1}^K z_{nk}^{(t)} \|\mathbf{x}_n - \boldsymbol{\mu}_k^{(t-1)}\|_2^2 \geq \sum_{n=1}^N \sum_{k=1}^K z_{nk}^{(t)} \|\mathbf{x}_n - \boldsymbol{\mu}_k^{(t)}\|_2^2$$

# Convergence of K-Means Algorithm

K-means is a **coordinate descent algorithm**:

$$\mathbf{z}^{(t+1)} = \arg \min_{\mathbf{z}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}^{(t)})$$

$$\boldsymbol{\mu}^{(t+1)} = \arg \min_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}^{(t+1)}, \boldsymbol{\mu})$$

**K-means converges** in a finite number of steps (due to monotonic decrease of the lower-bounded objective function and a finite number of possible assignments)

**No guarantee** on the number of iterations needed for convergence (can be exponential)

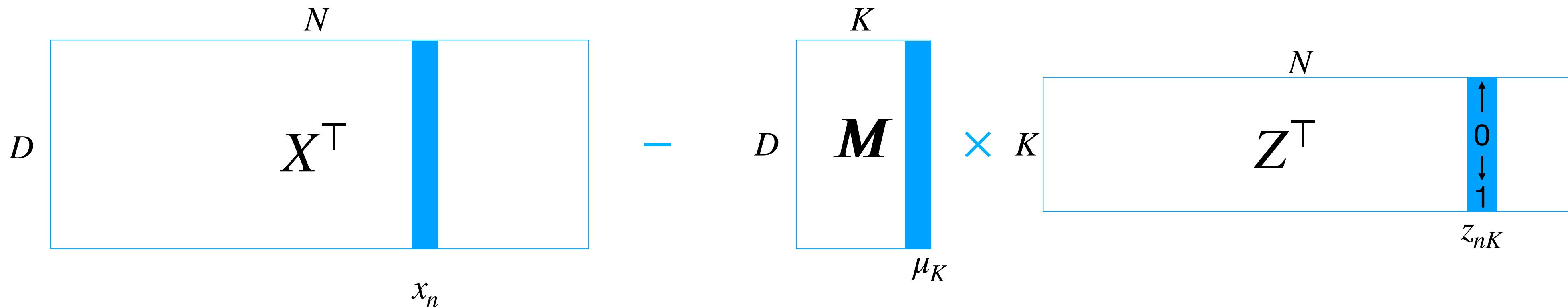
**No nontrivial lower bound** on the gap between the final objective value and the global minimum

**May fail to find the global optimum** (might even converge to a point which is not a local minimum)

# K-Means as Matrix Factorization

We can interpret K-means as Matrix Factorization:

$$\begin{aligned}\min_{\mathbf{z}, \mu} \mathcal{L}(\mathbf{z}, \mu) &= \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \mu_k\|_2^2 \\ &= \|X^\top - MZ^\top\|_{\text{Frob}}^2\end{aligned}$$



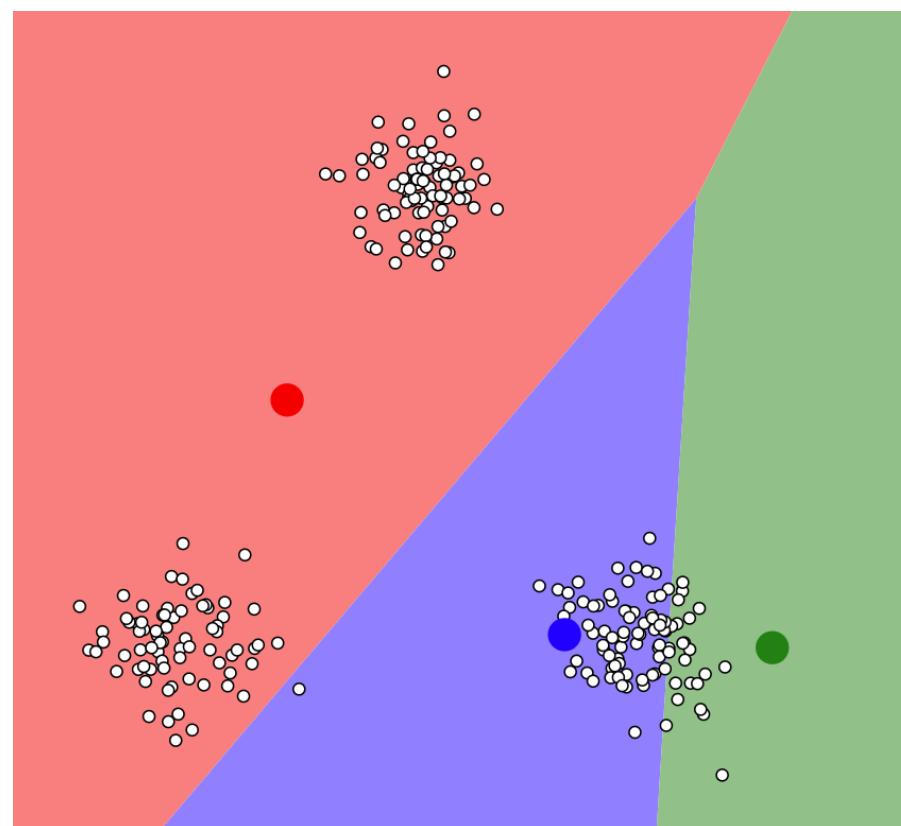
$x_n$  is reconstructed by the mean of closest centroid  $\mu_k$

# K-Means Challenges

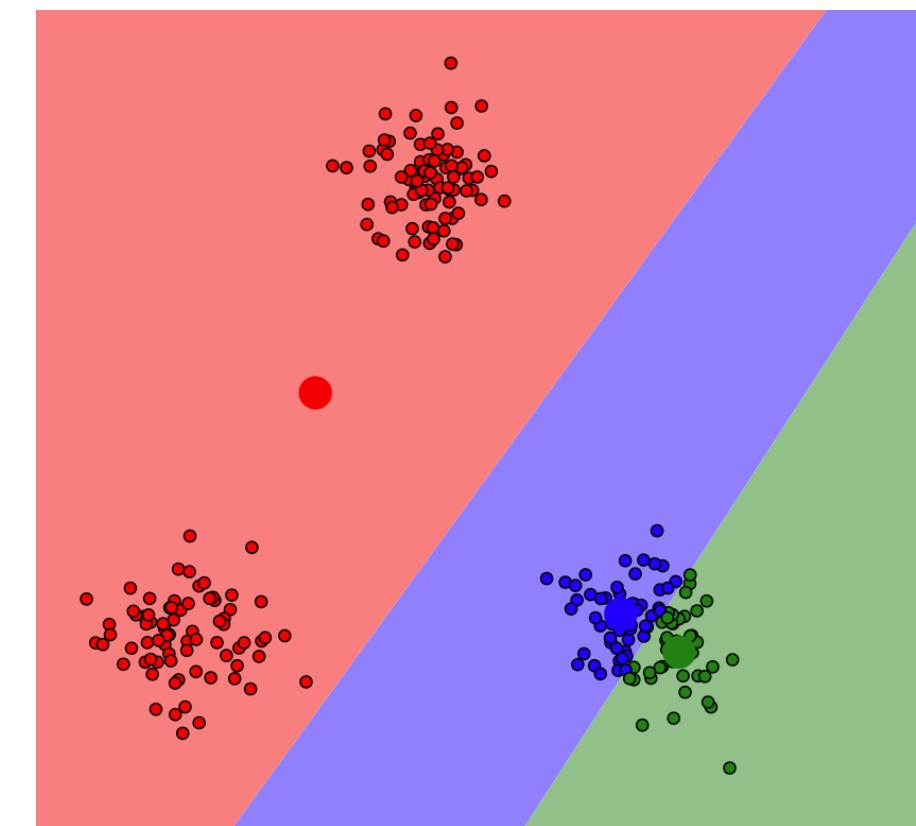
# Challenge 1: Performance strongly depends on the initialization

[Try here!](#)

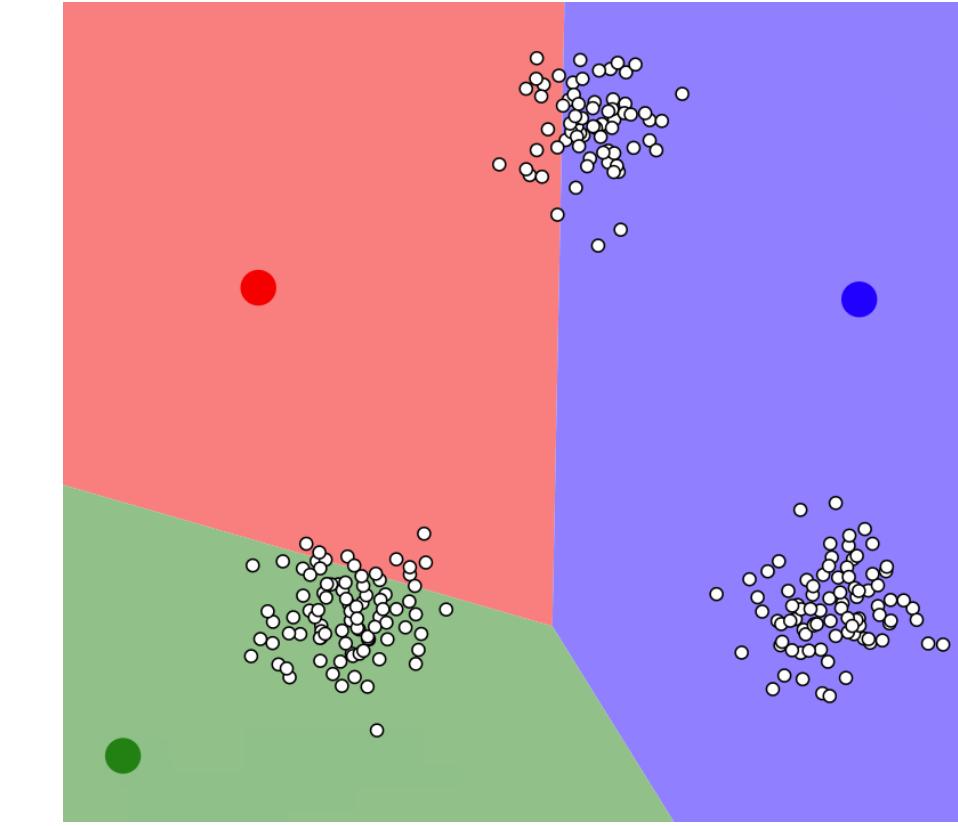
**Bad** initialization



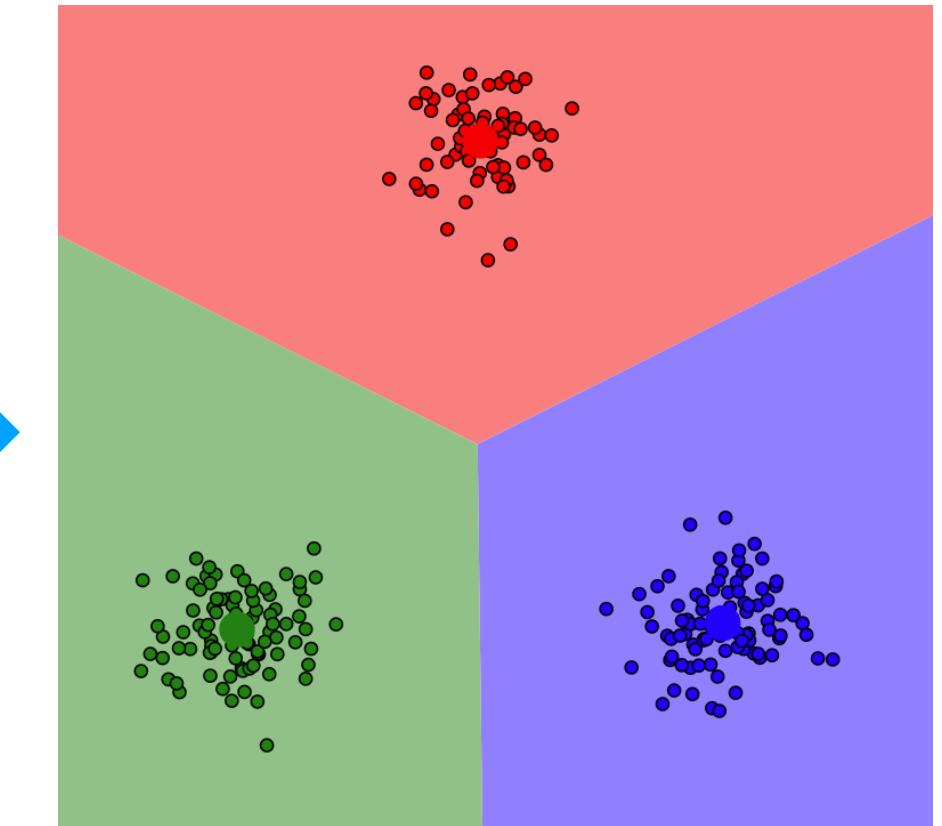
**Local minimum**



**Good** initialization



**Global maximum**



Solutions:

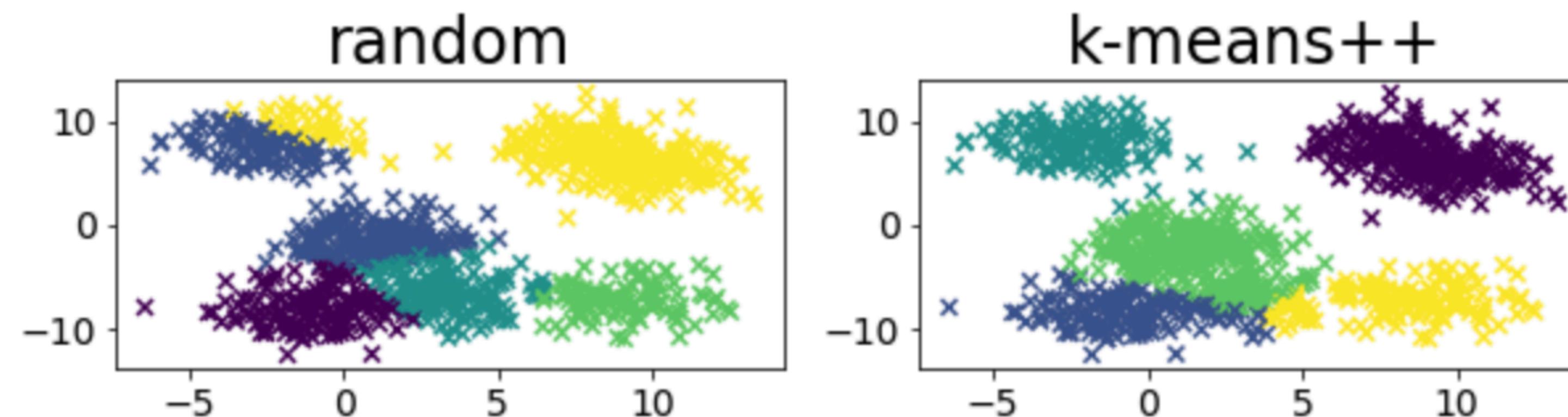
- Multiple random restarts
- Seeding with K-means++

# K-Means++

An improved initialization method for the K-means that helps to **avoid poor initial centroid placements**

The basic idea is to spread out the initial centroids so that they are more representative of the data distribution, rather than choosing them randomly

**Benefits:** Avoids poor local minima, faster convergence, and lower K-means objective (expected cost  $O(\log K)$  of the optimal solution)



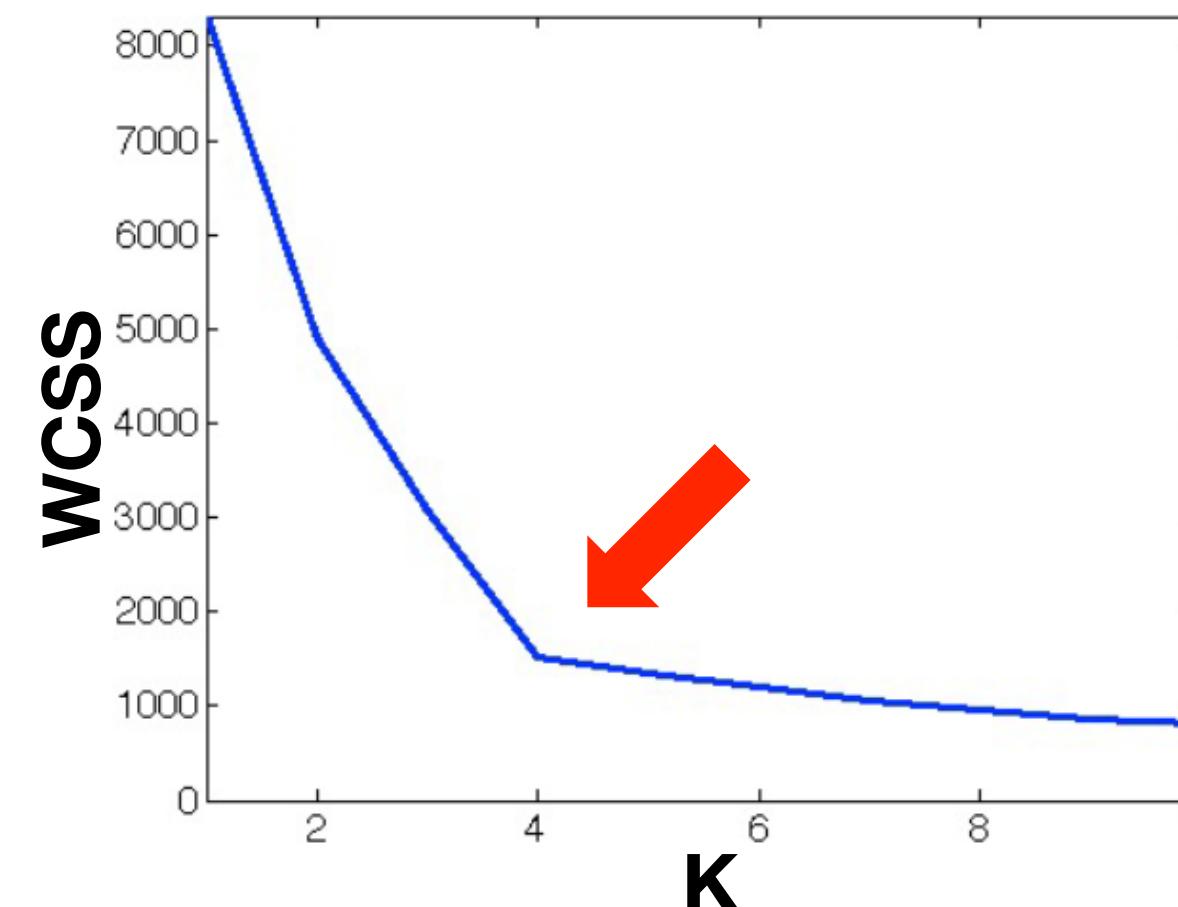
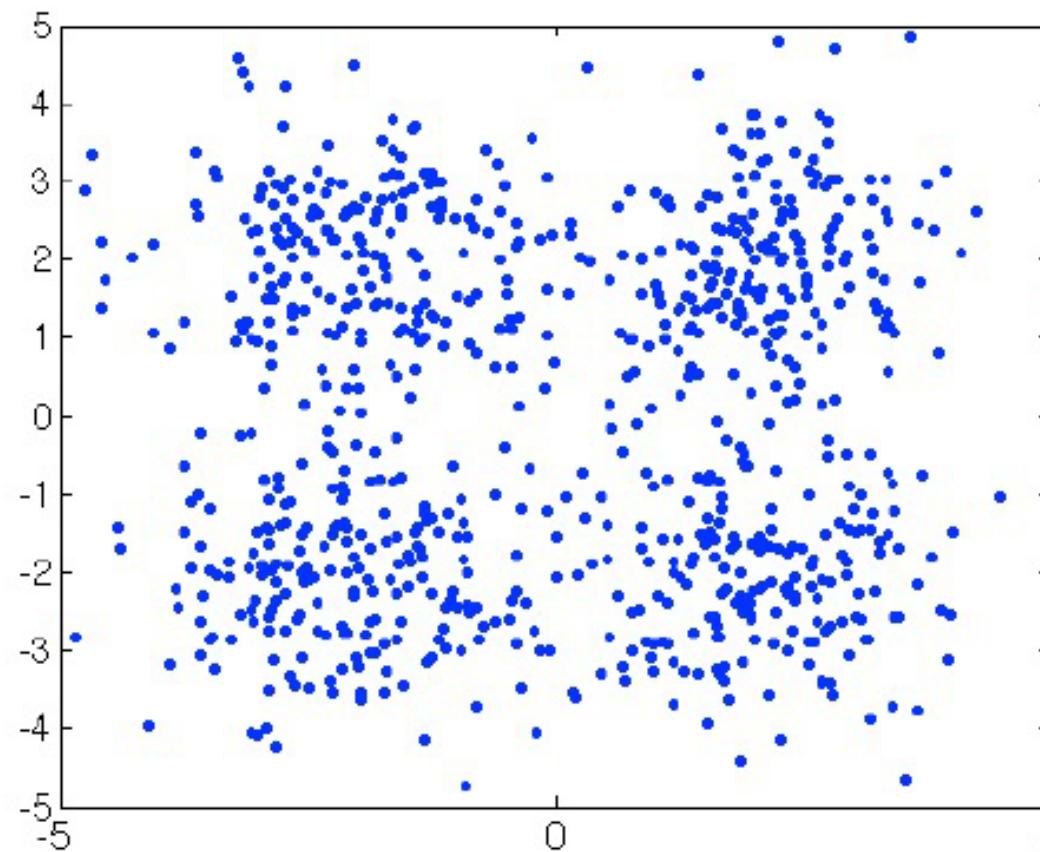
# K-Means++

1. Select the first centroid  $\mu_1$  randomly from the data points in  $\mathcal{D}$
2. Select subsequent centroids probabilistically:
  1. For each point  $x$ , calculate the distance  $D(x) = \min_{j \in \{1, \dots, m\}} \|x - \mu_j\|_2^2$ , where  $\{\mu_1, \dots, \mu_m\}$  represents the centroids that have already been chosen
  2. Select the next centroid  $\mu_{m+1}$  by choosing a point  $x$  from  $\mathcal{D}$  with probability:  $P(x) = \frac{D(x)}{\sum_{x \in \mathcal{D}} D(x)}$
  3. Repeat Step 2 until K centroids have been chosen
  4. Run standard K-means

# Challenge 2: How to define the number of clusters?

Test loss typically keeps decreasing with increasing K so we cannot use cross-validation to determine the number of clusters

- Domain knowledge
- Elbow method

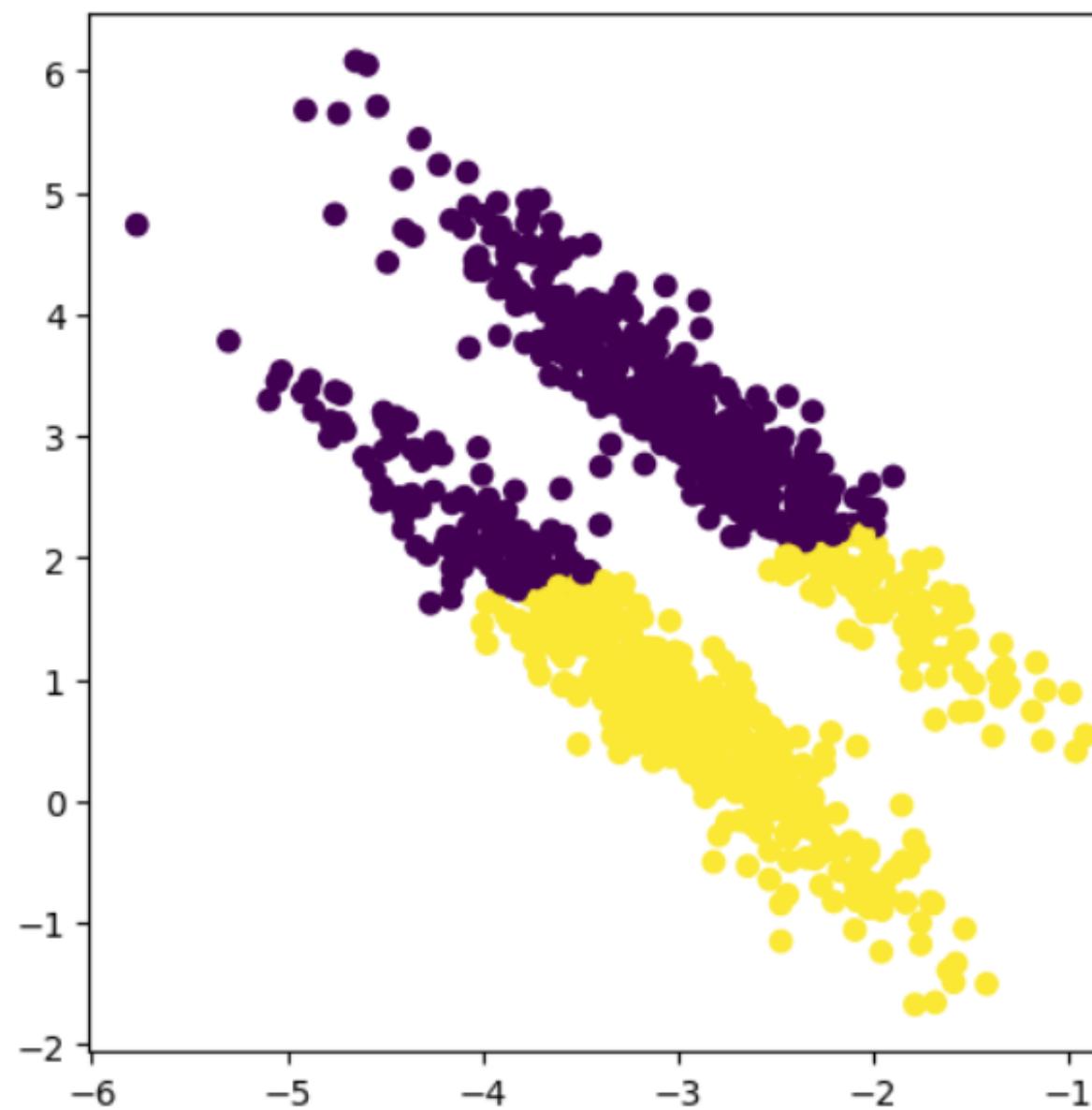


1. Run K-means for different values of K (e.g., 1, 2, ..., 10) and calculate the within-cluster sum of squares (WCSS) for each K
2. Plot K vs the WCSS, and look for the **elbow point** where the rate of decrease sharply slows

**Diminishing returns:** A new cluster significantly reduces loss when there are too few clusters, but offers little improvement once the optimal number is reached

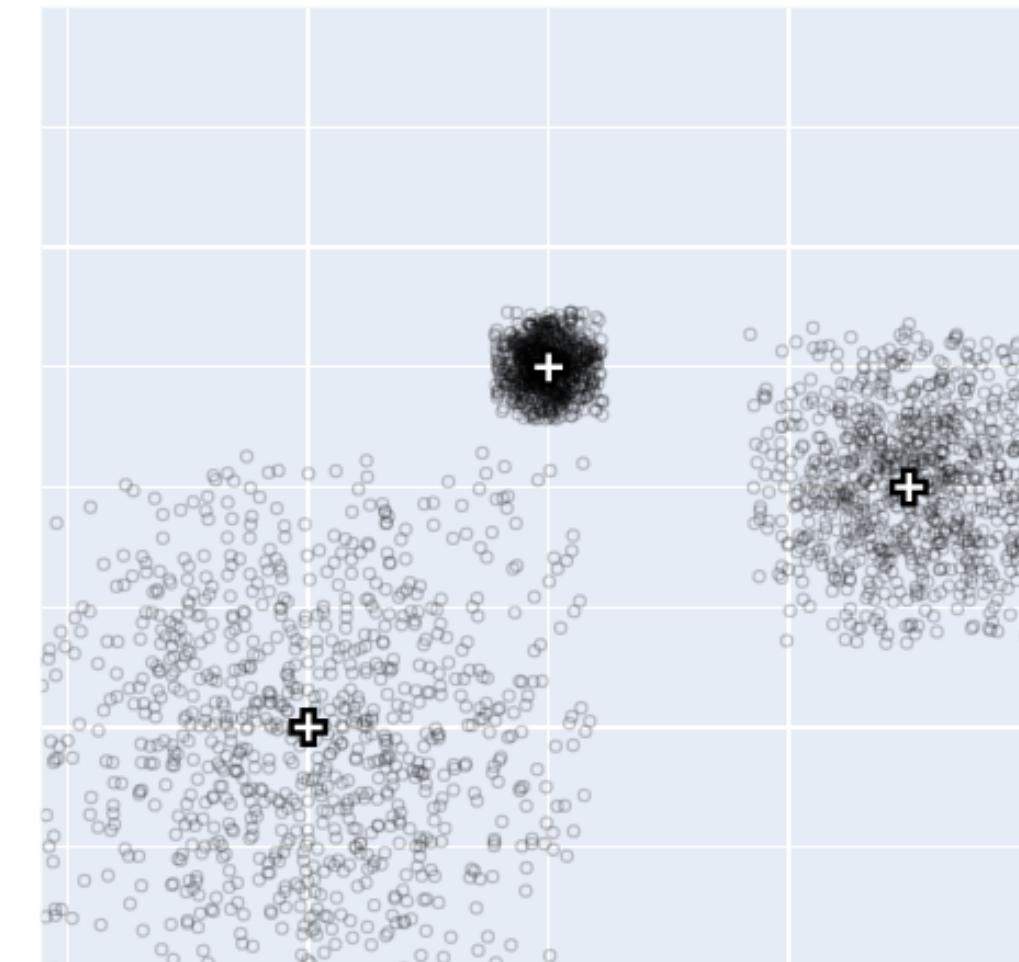
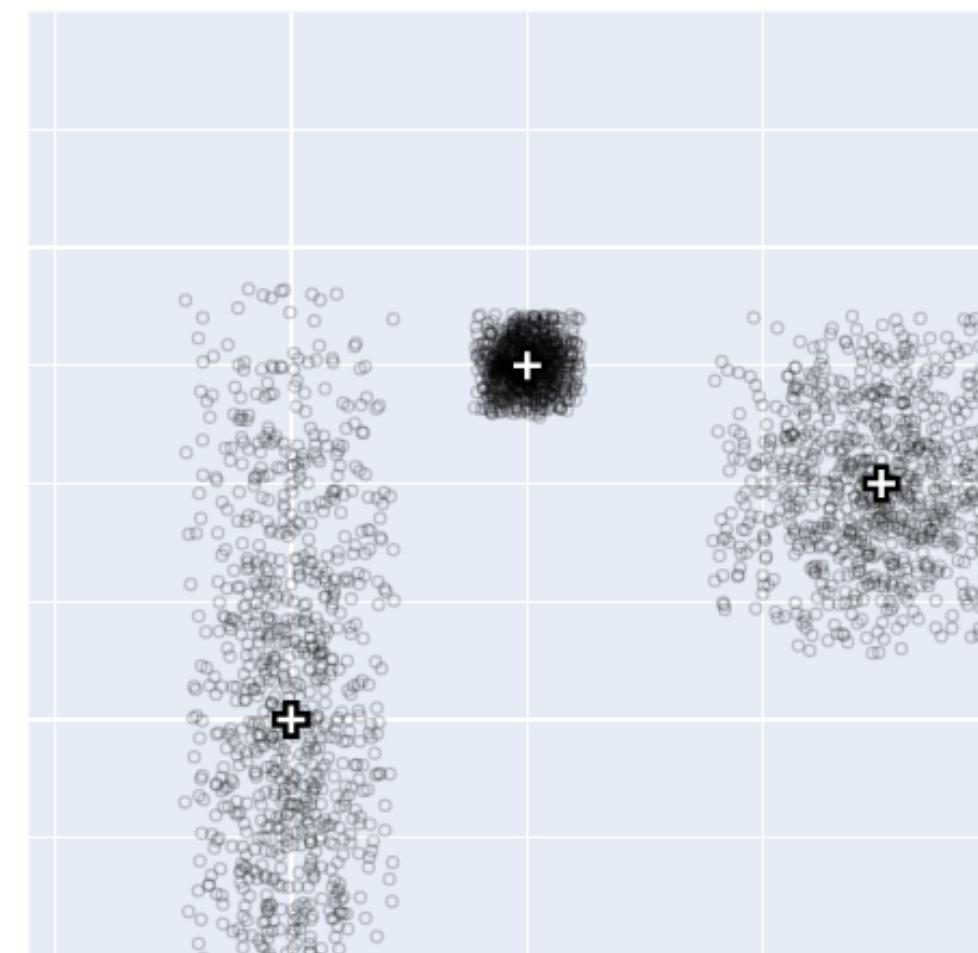
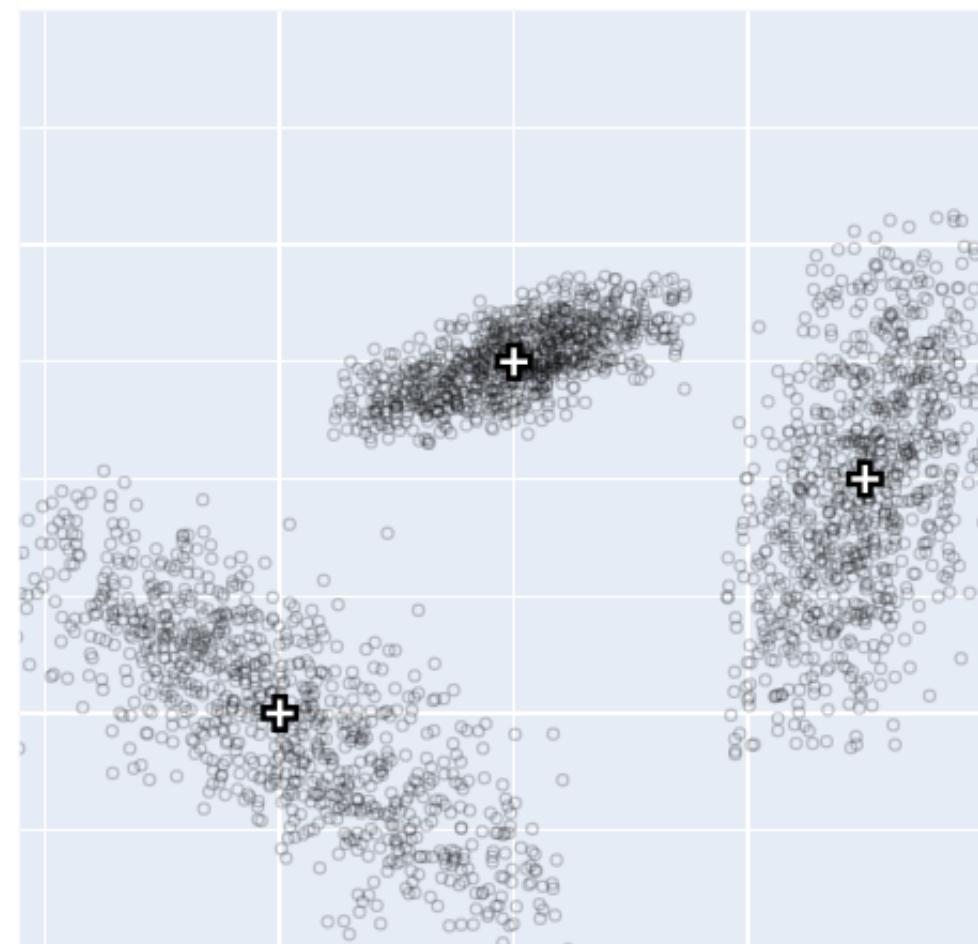
# Challenge 3: It cannot model well clusters of arbitrary shape

K-means forces the clusters to be spherical (e.g., not elliptical) and of equal size

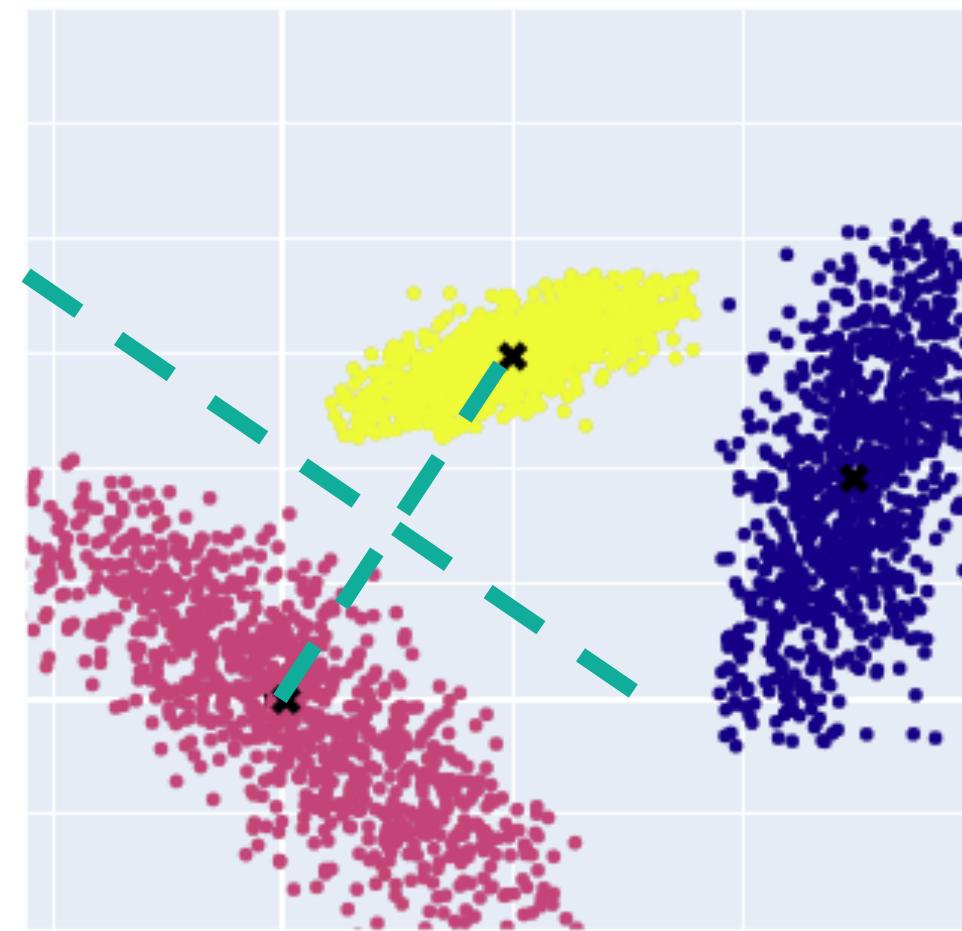


K-means struggles with elliptical clusters

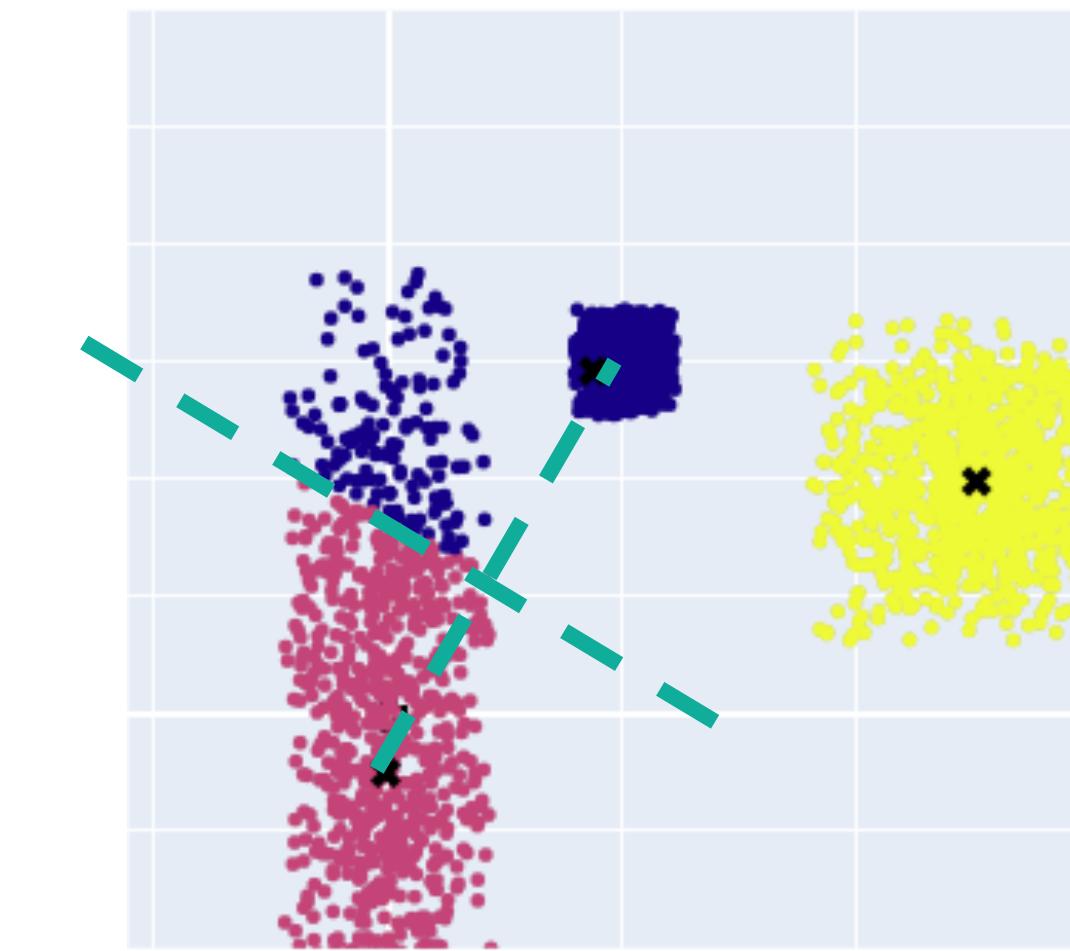
# When Does K-Means Fail?



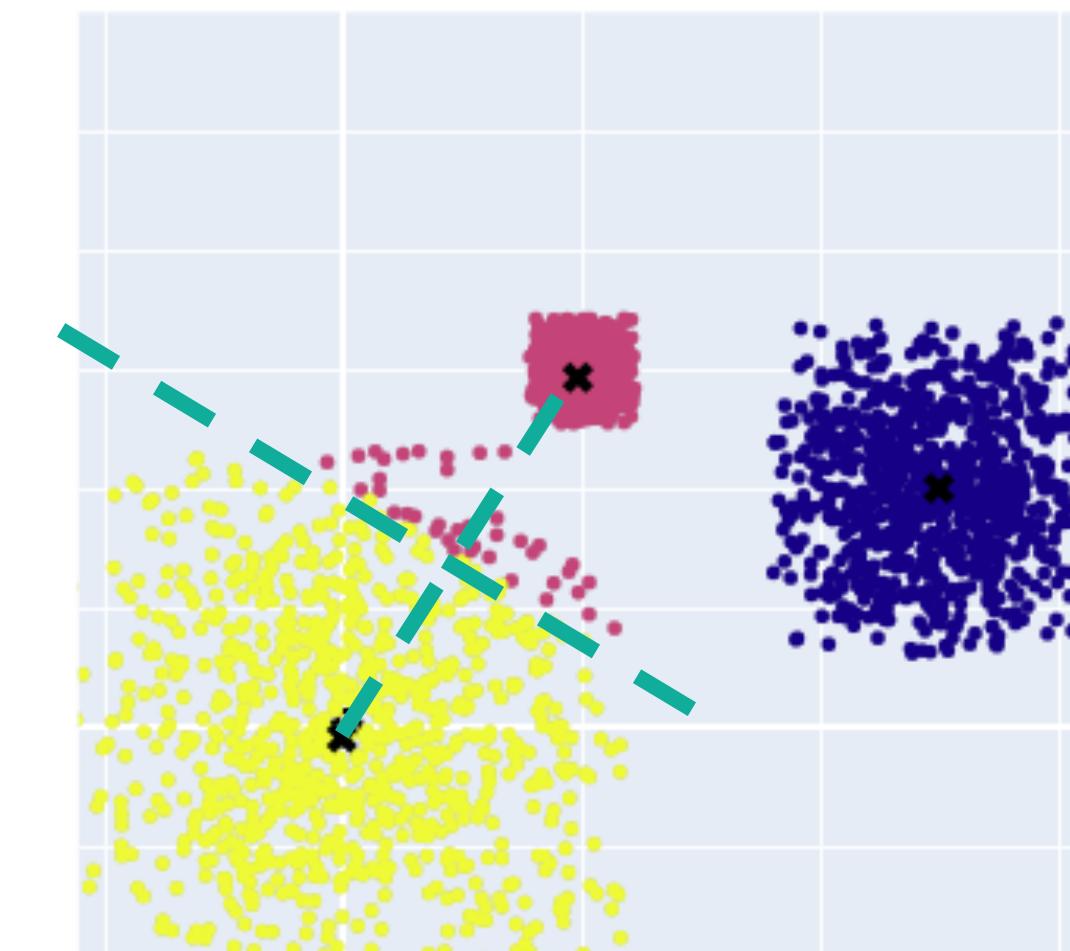
# When Does K-Means Fail?



**Works!**



**Fails!**

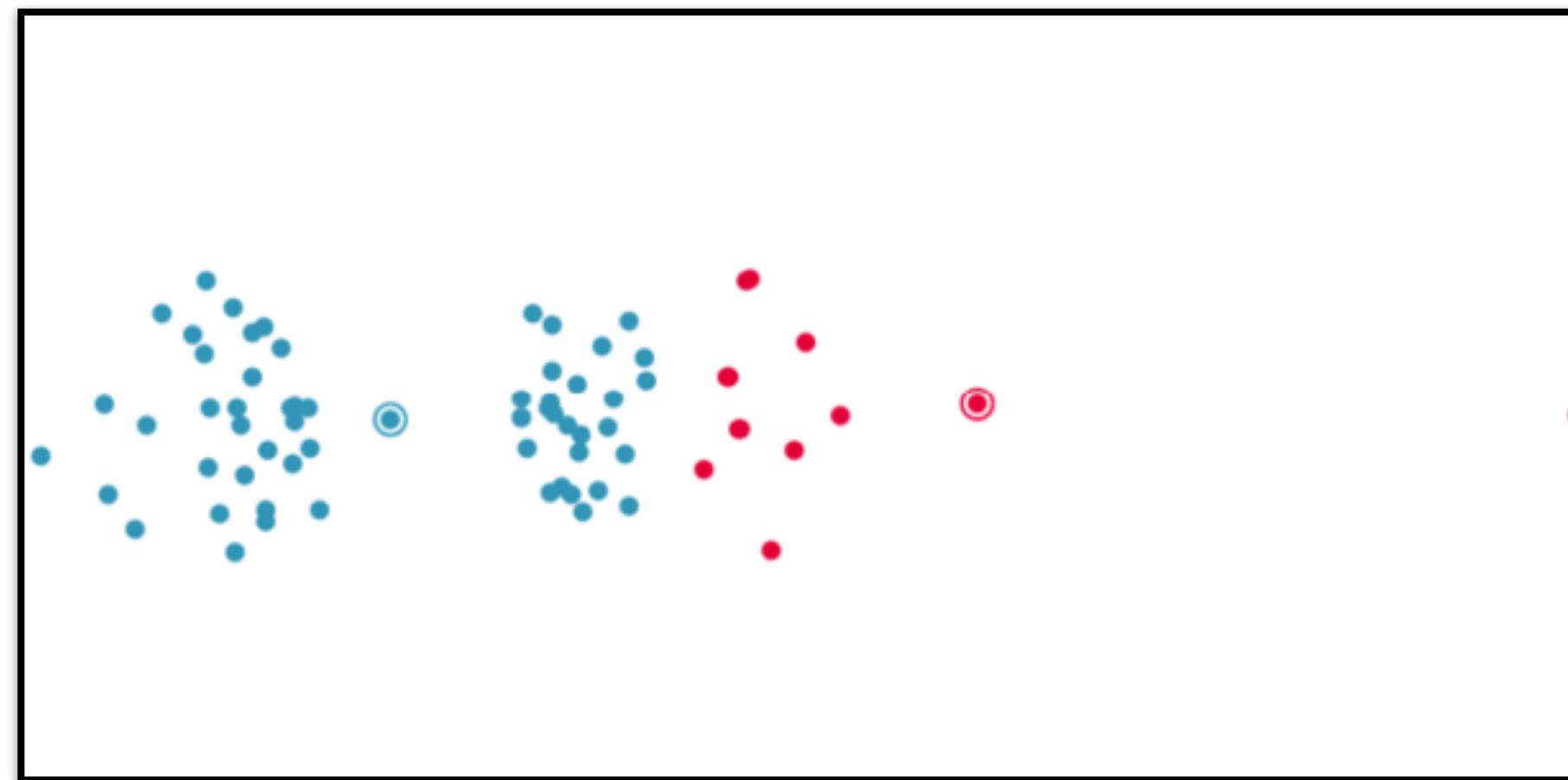


**Fails!**

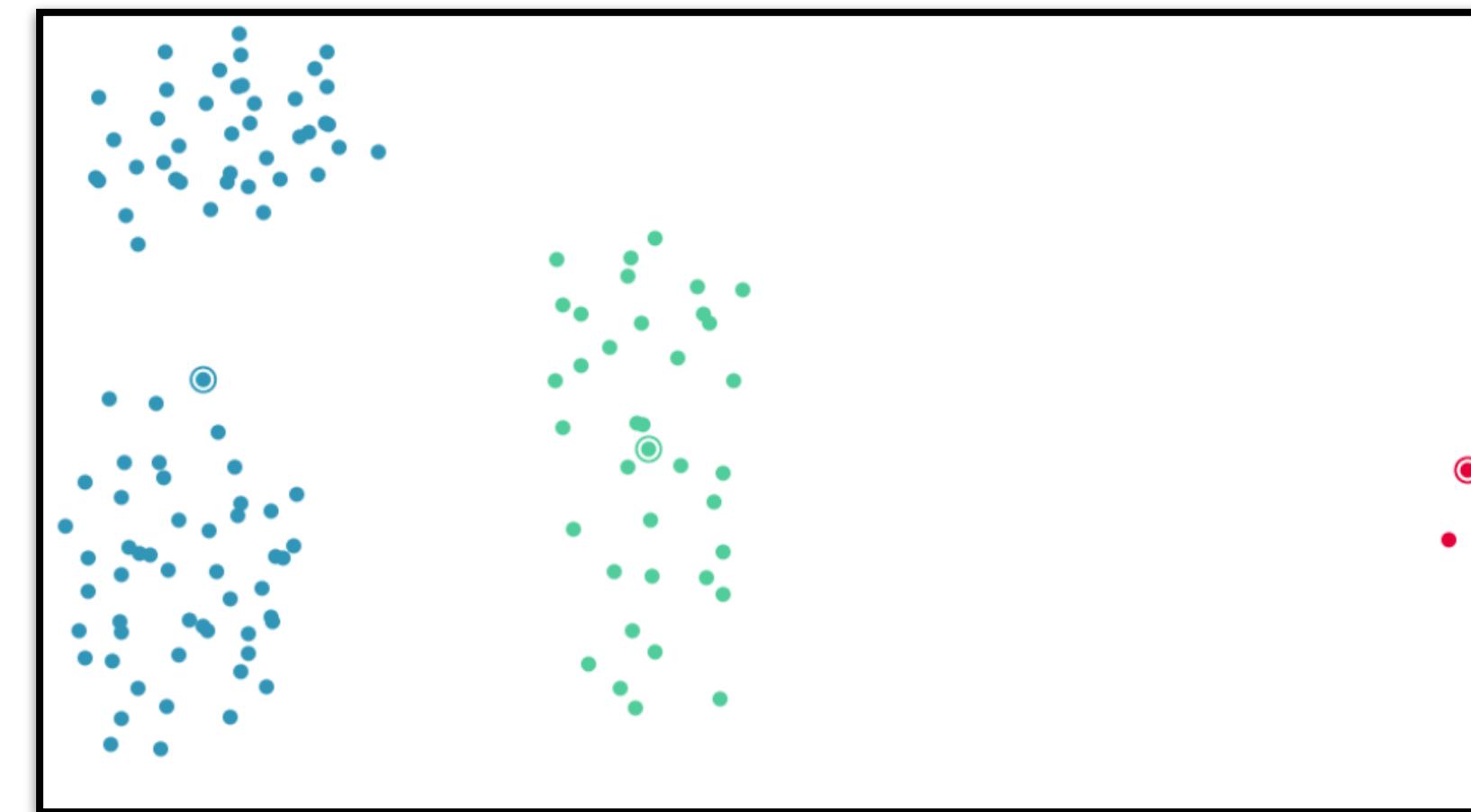
K-means fails when clusters are not separated by the bisectors of the lines connecting their centers

# Challenge 4: Sensitivity to outliers

Outliers can drag the centroids away from the actual dense areas, causing clusters to become stretched or less representative of the majority of data points



Outliers can lead to **suboptimal** results

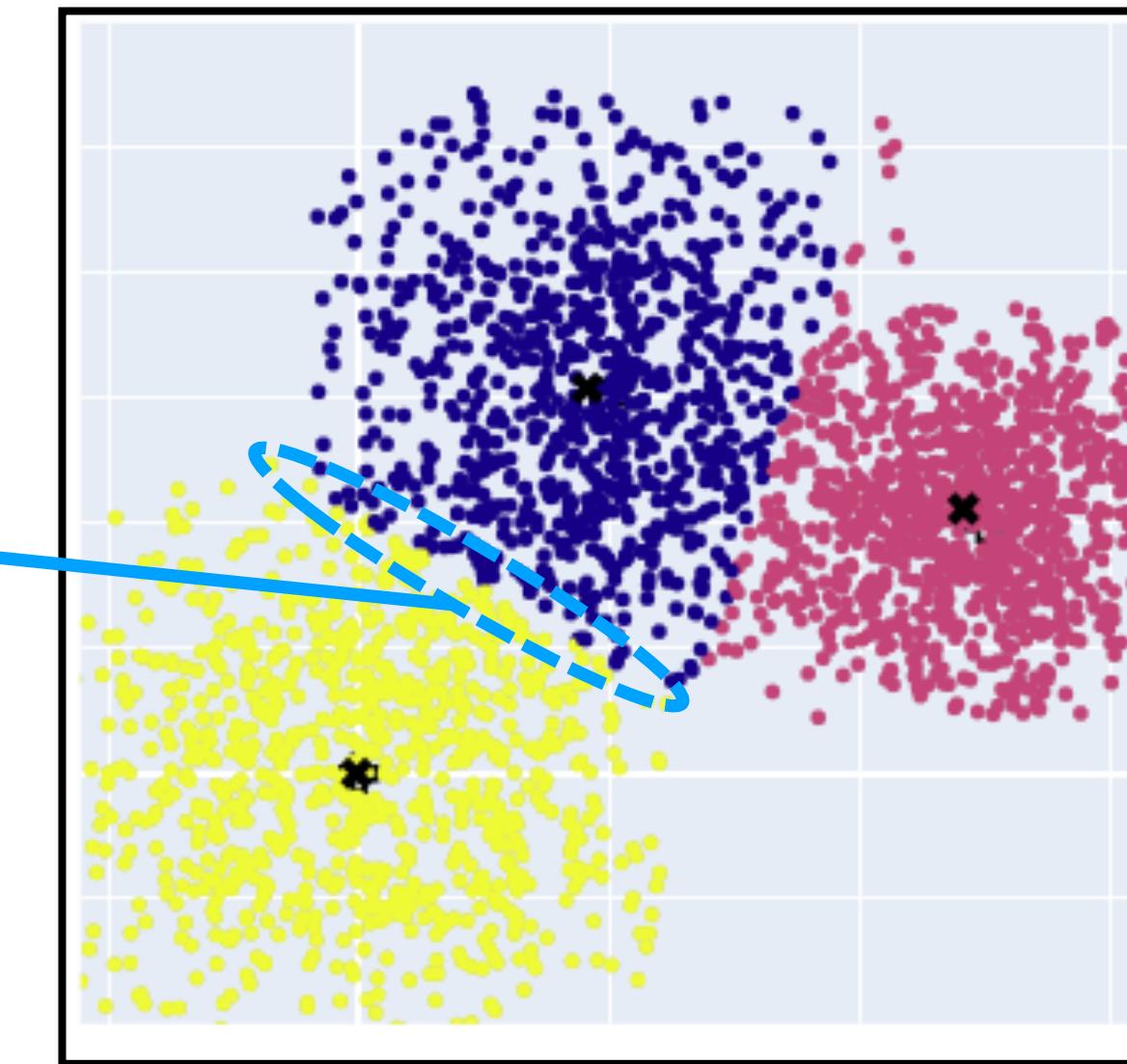


Outliers can lead to **undesirable** clusters

# Challenge 5: Hard assignments in overlapping clusters

In many cases, data has overlapping clusters or points that don't clearly belong to any single cluster

Very similar (close)  
datapoints belong to  
different clusters



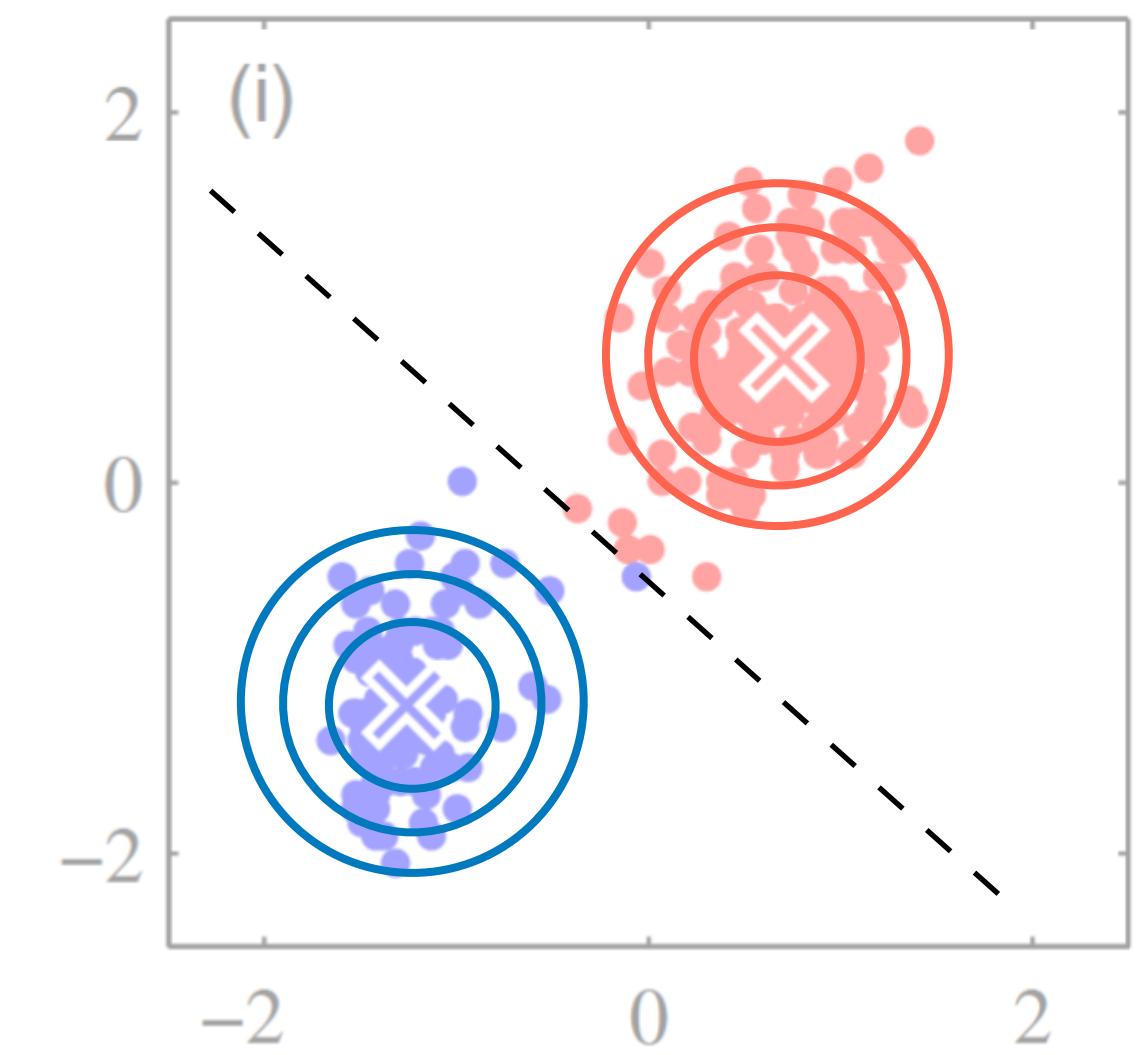
# Probabilistic view on K-Means

Consider each cluster as an isotropic Gaussian:

$$\begin{aligned} p(\mathbf{x}_n | \mu, \mathbf{z}_n) &= \prod_{k=1}^K \mathcal{N}(\mathbf{x}_n | \mu_k, I_D)^{z_{nk}} \\ &= \prod_{k=1}^K c \exp\left(-\frac{1}{2} \|\mathbf{x}_n - \mu_k\|_2^2 z_{nk}\right) \end{aligned}$$

Dataset likelihood will be:

$$\begin{aligned} \log \prod_{n=1}^N p(\mathbf{x}_n | \mu, \mathbf{z}_n) &= \log \prod_{n=1}^N \prod_{k=1}^K c \exp\left(-\frac{1}{2} \|\mathbf{x}_n - \mu_k\|_2^2 z_{nk}\right) \\ &= - \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \mu_k\|_2^2 + c' \\ &= -\mathcal{L}(\mathbf{z}, \boldsymbol{\mu}) + c' \end{aligned}$$



K-means can be interpreted as **maximizing the log-likelihood of the data** with this data model

# GMM

# Gaussian Mixture Models

Data is modeled as a mixture of  $K$  Gaussian distributions

Instance space is  $\mathcal{X} = \mathbb{R}^D$  and each sample  $\mathbf{x}$  is generated as follows:

1. Choose a random component  $k \in \{1, 2, \dots, K\}$ , i.e., let  $Z$  be a multinomial r.v. with  $\mathbb{P}(Z = k) = \pi_k$
2. Sample  $\mathbf{x}$ , given the value of  $Z = k$ , from a Gaussian distribution:

$$p(\mathbf{x} | Z = k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k) \right)$$

# Gaussian Mixture Models

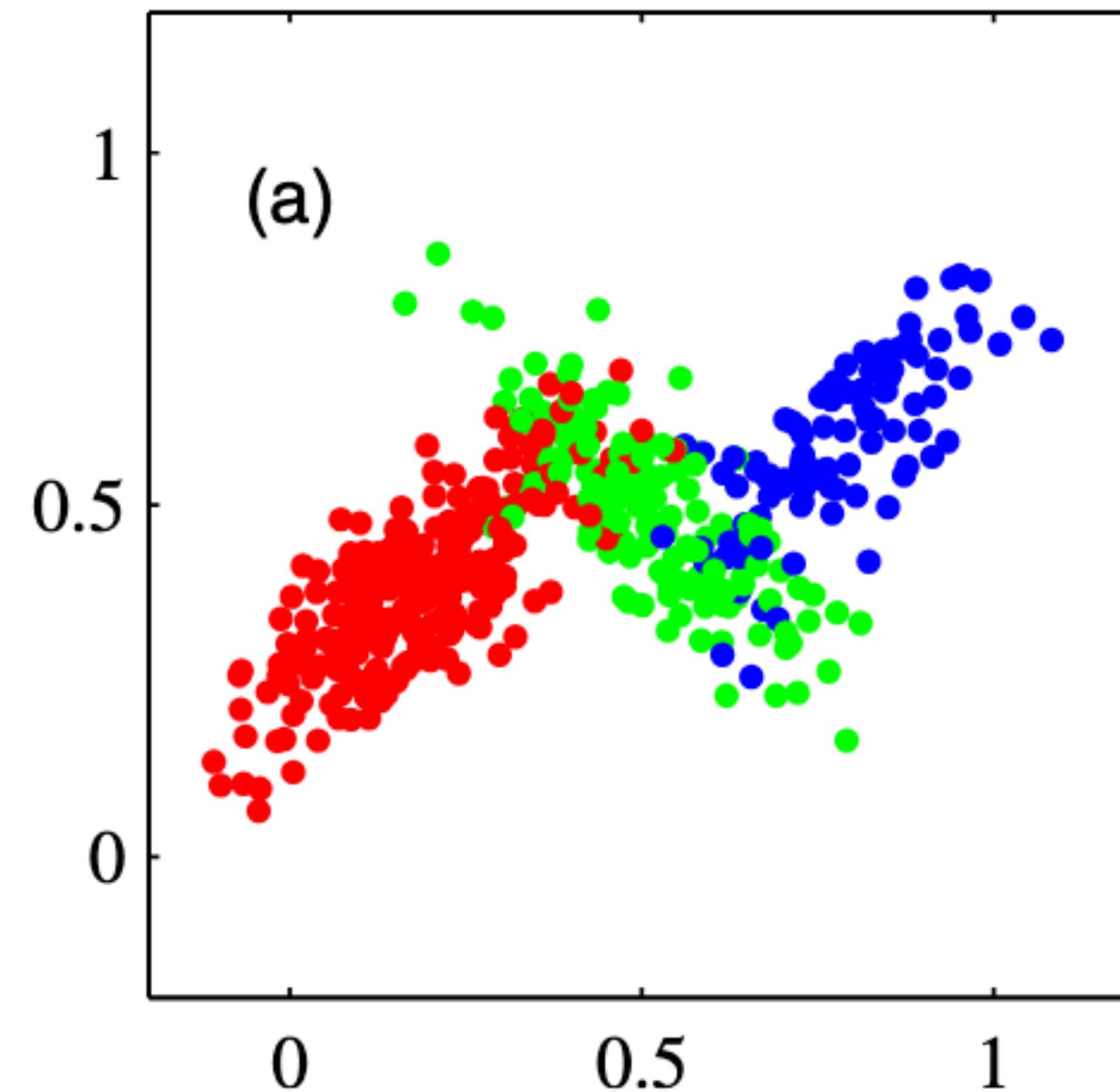
Then we can write the distribution of  $\mathbf{x}$  as:

$$\begin{aligned} p(\mathbf{x}) &= \sum_{k=1}^K \mathbb{P}[Z = k] p(\mathbf{x} | Z = k) \\ &= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k) \\ &= \sum_{k=1}^K \frac{\pi_k}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right) \end{aligned}$$

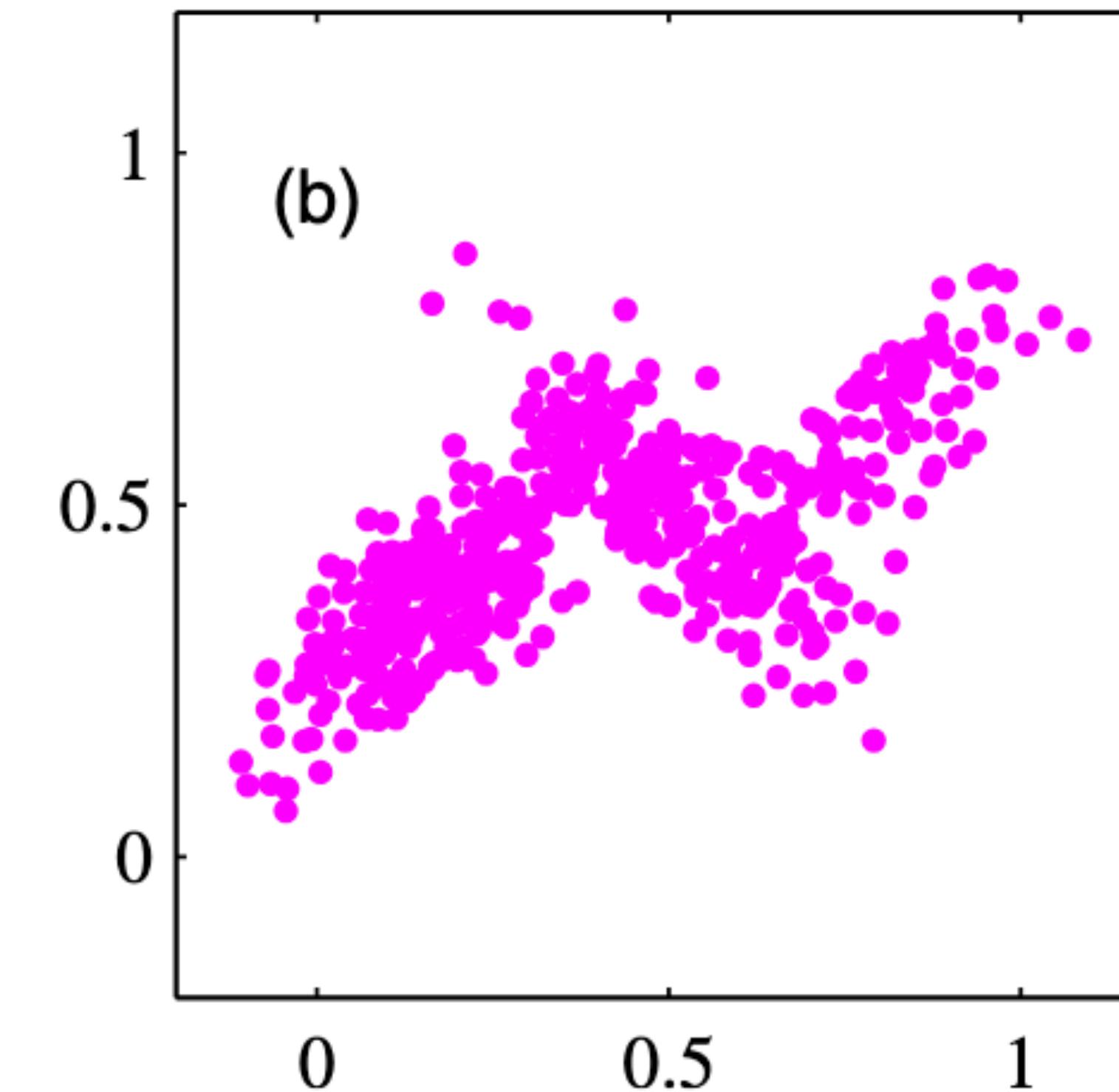
$Z$  is a latent (hidden) variable that is not directly observed

We introduce  $Z$  because it helps us describe a simple parametric form

# Visualizing the Joint and Marginal Distributions of GMM



Samples from the joint distribution  $p(X, Z)$ ,  
with  $Z$  represented by colors



Samples from the marginal distribution  $p(X)$

# Recap

Unsupervised Learning learns patterns from unlabeled data to discover hidden structures

K-means clustering partitions data into K clusters by minimizing the distance between points and cluster centroids

K-means is simple and efficient but assumes well-separated clusters and requires a predefined K

GMM: Data is modeled as a mixture of K Gaussian distributions. Offers great flexibility in modeling as you can freely choose  $\{\pi_k, \mu_k, \Sigma_k\}_{k \in K}$

Our objective is to maximize the log-likelihood of the data to get estimates of the model parameters but this brings new challenges (next week)