# 8. K-Means Clustering

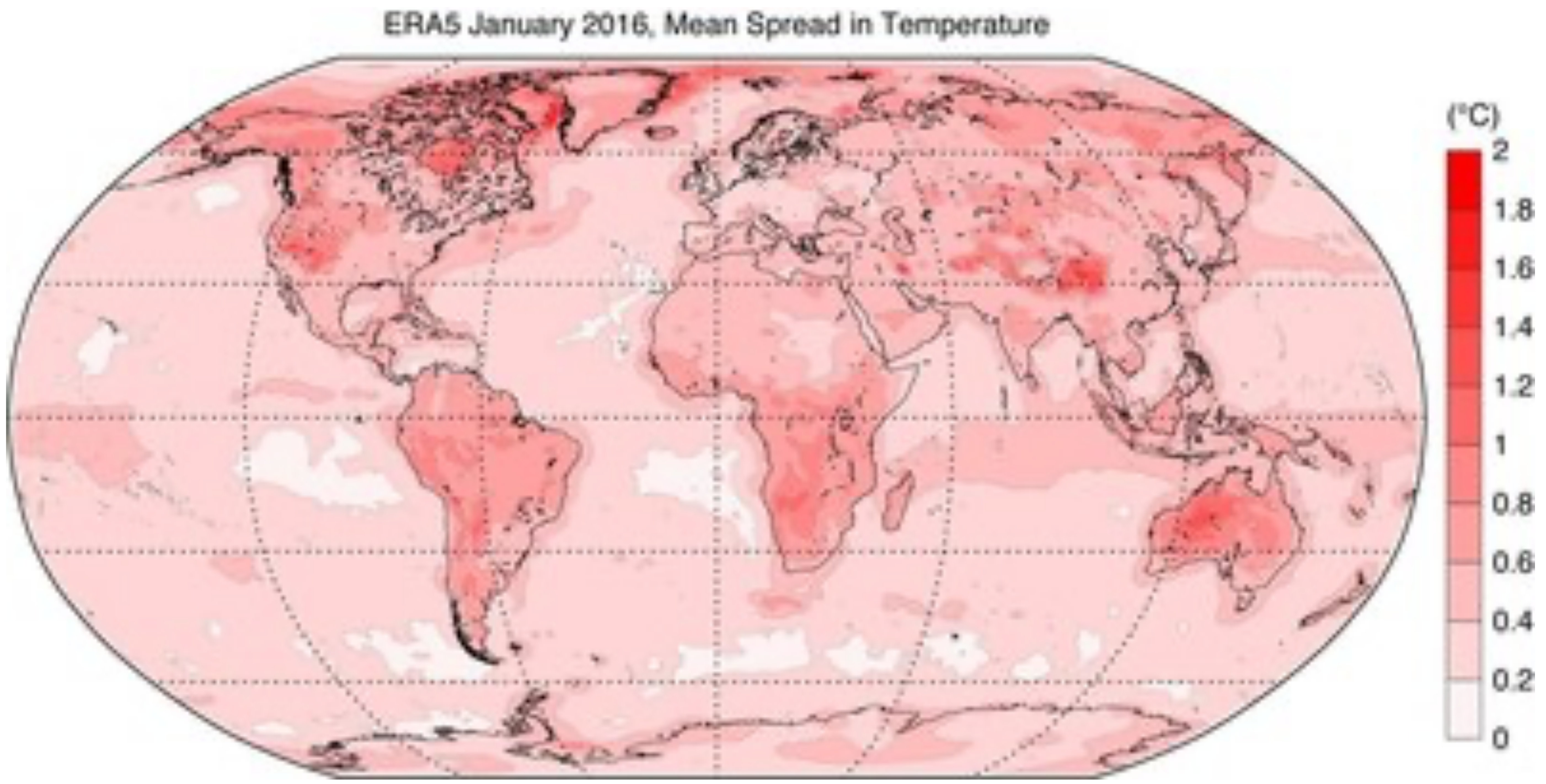
## EEC454 Introduction to Machine Learning Systems

2023 Fall, Jaeho Lee

# Recap: Supervised Learning

- **What we have.** A labeled dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- **Want to do.** Learn $f(\cdot)$ such that $f(\mathbf{x}) \approx y$.
    - *Example*. ERA5 dataset
        - $\mathbf{x}$: time & location
        - $y$: temperature.

$\Rightarrow$ Train a model for temperature prediction



# Unsupervised Learning

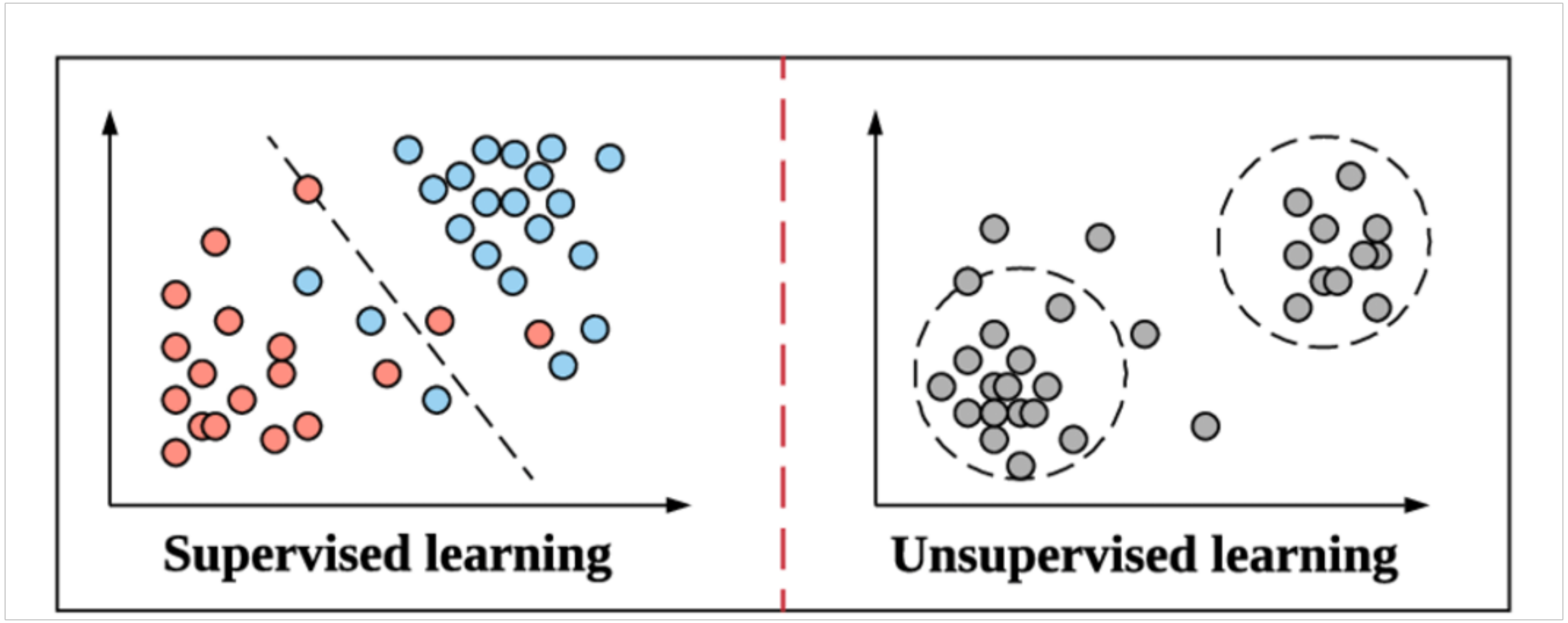
# Unsupervised Learning

- **What we have.** An unlabeled dataset $D = \{\mathbf{x}_i\}_{i=1}^{n}$
  - No labeling cost—typically very large!
  - *Example*. Common Crawl — petabytes of web-crawled sentences.

$\Rightarrow$ Most Language Models trained on these!

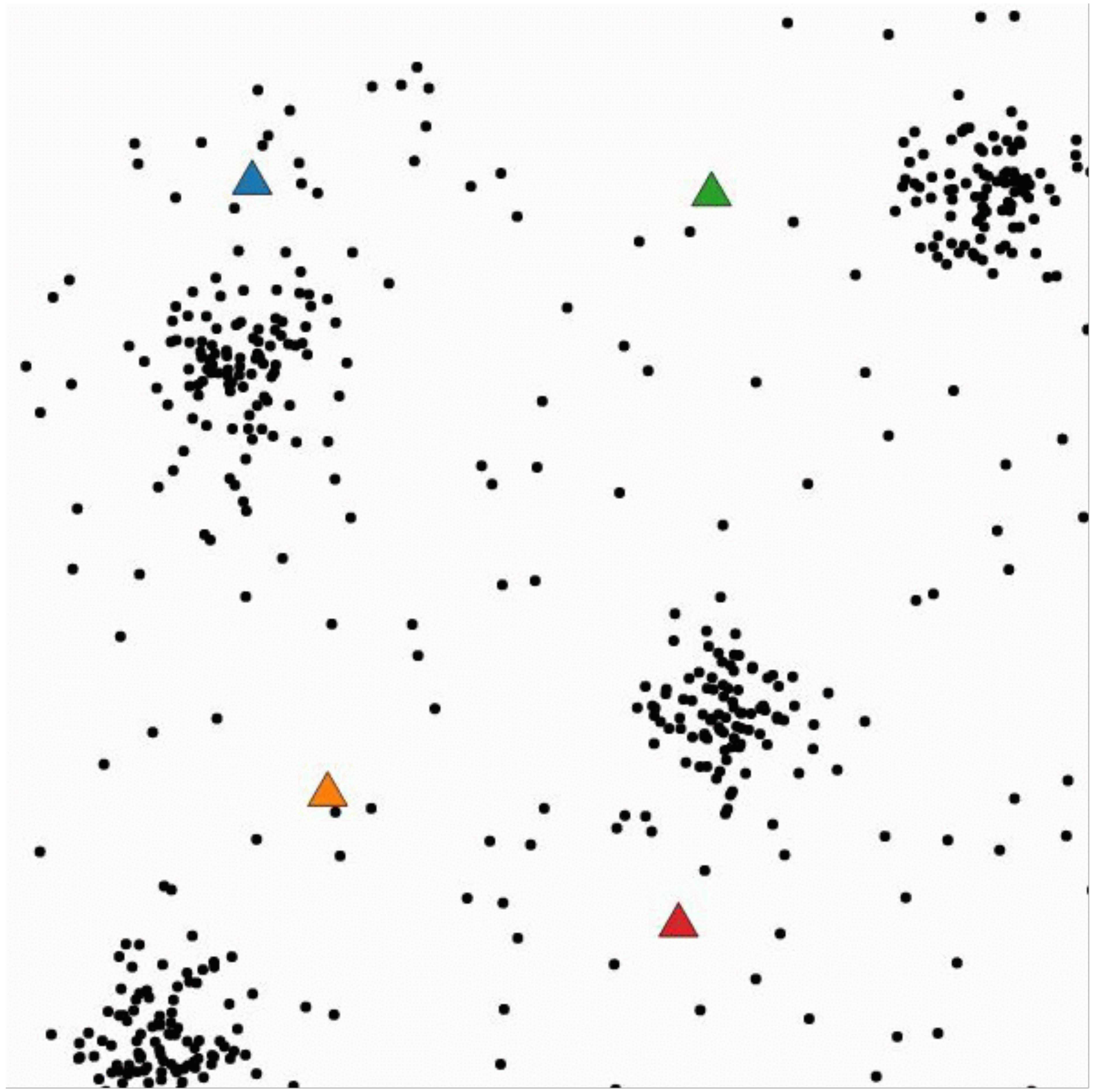
# Unsupervised Learning

- **Want to do.** Get insights from data, by discovering underlying structure, cause, or statistical relation
  - Learned structure can be used for supervised learning tasks.
    (e.g., learning a feature map $\Phi(\,\cdot\,)$)



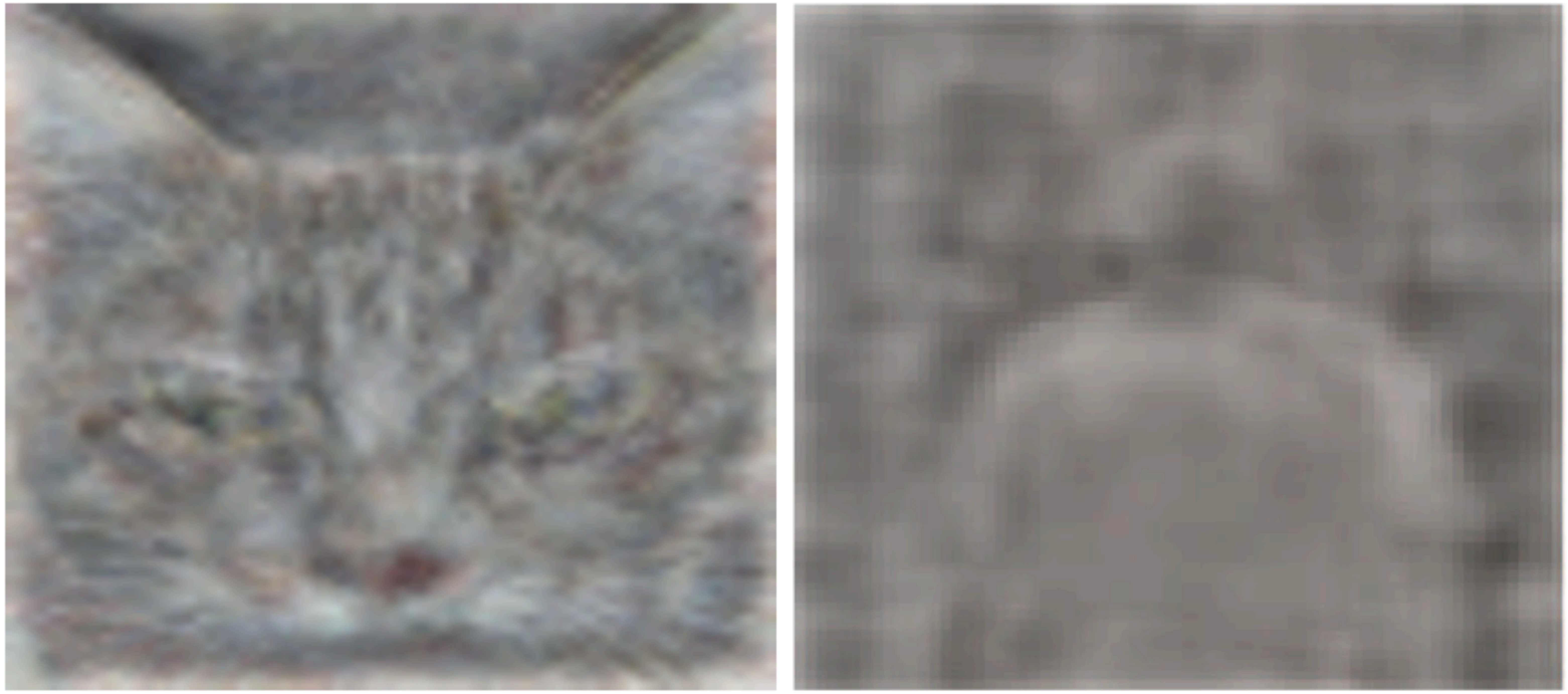
# What can unsupervised learning do?

- **1957.** People were clustering many data points.



# What can unsupervised learning do?

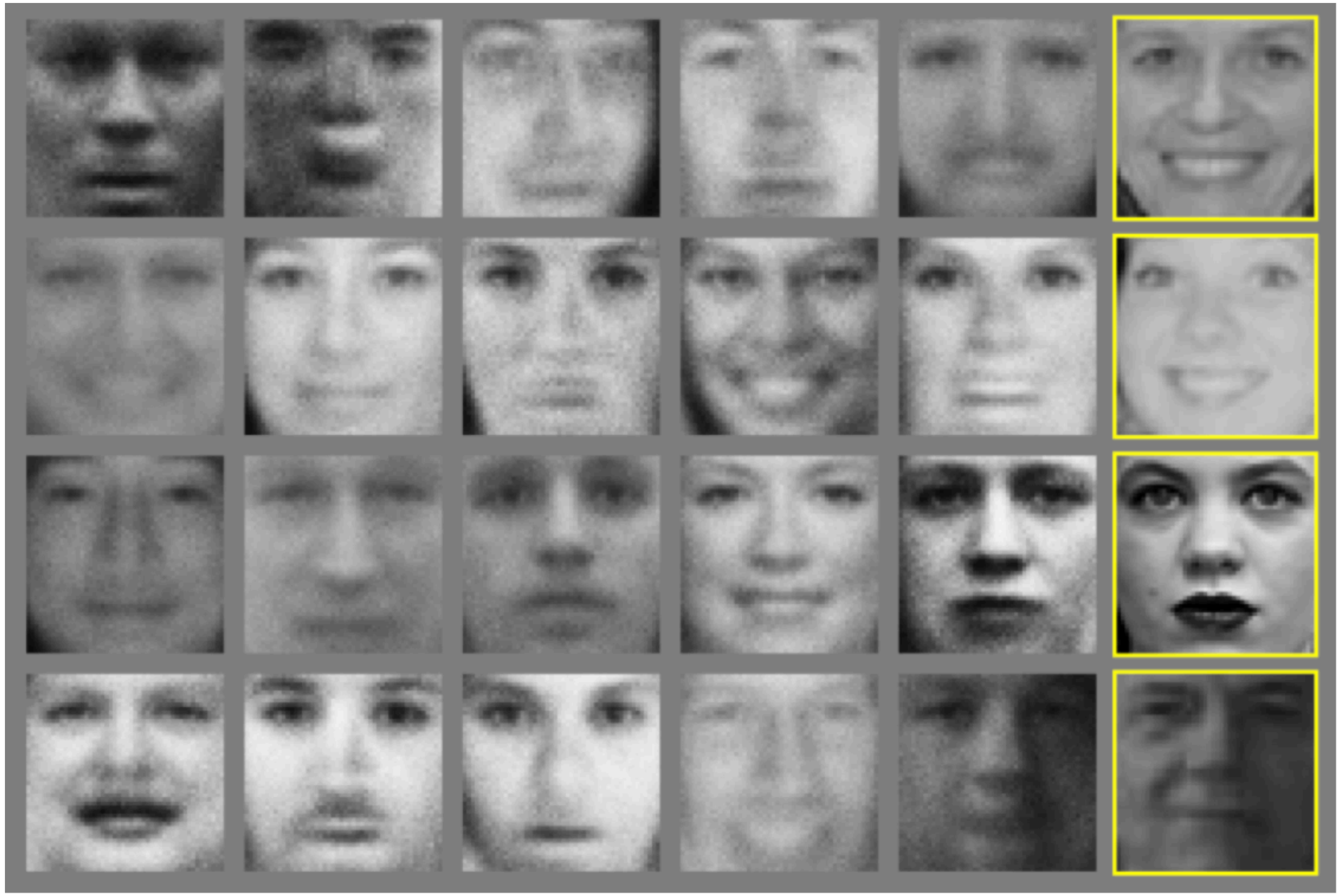
- **2012.** Youtube videos (w/o labels) can be used to discover useful patterns for classification.



Q. V. Le "Building High-Level Features Using Large Scale Unsupervised Learning," ICASSP 2013

# What can unsupervised learning do?

- **2014.** People used face images to generate realistic(?) new faces.



Goodfellow et al., “Generative Adversarial Nets” NeurIPS 2014

# What can unsupervised learning do?

- **2023.** People are training awesome chatbots.

GPT-4

JA

can you write a python script for performing principal component analysis?

Sure, I can provide a basic example of performing Principal Component Analysis (PCA) using Python. This script will make use of the `pandas` library for handling data and `scikit-learn` library for PCA.

In this example, I will use the Iris dataset which is built into scikit-learn for demonstration.

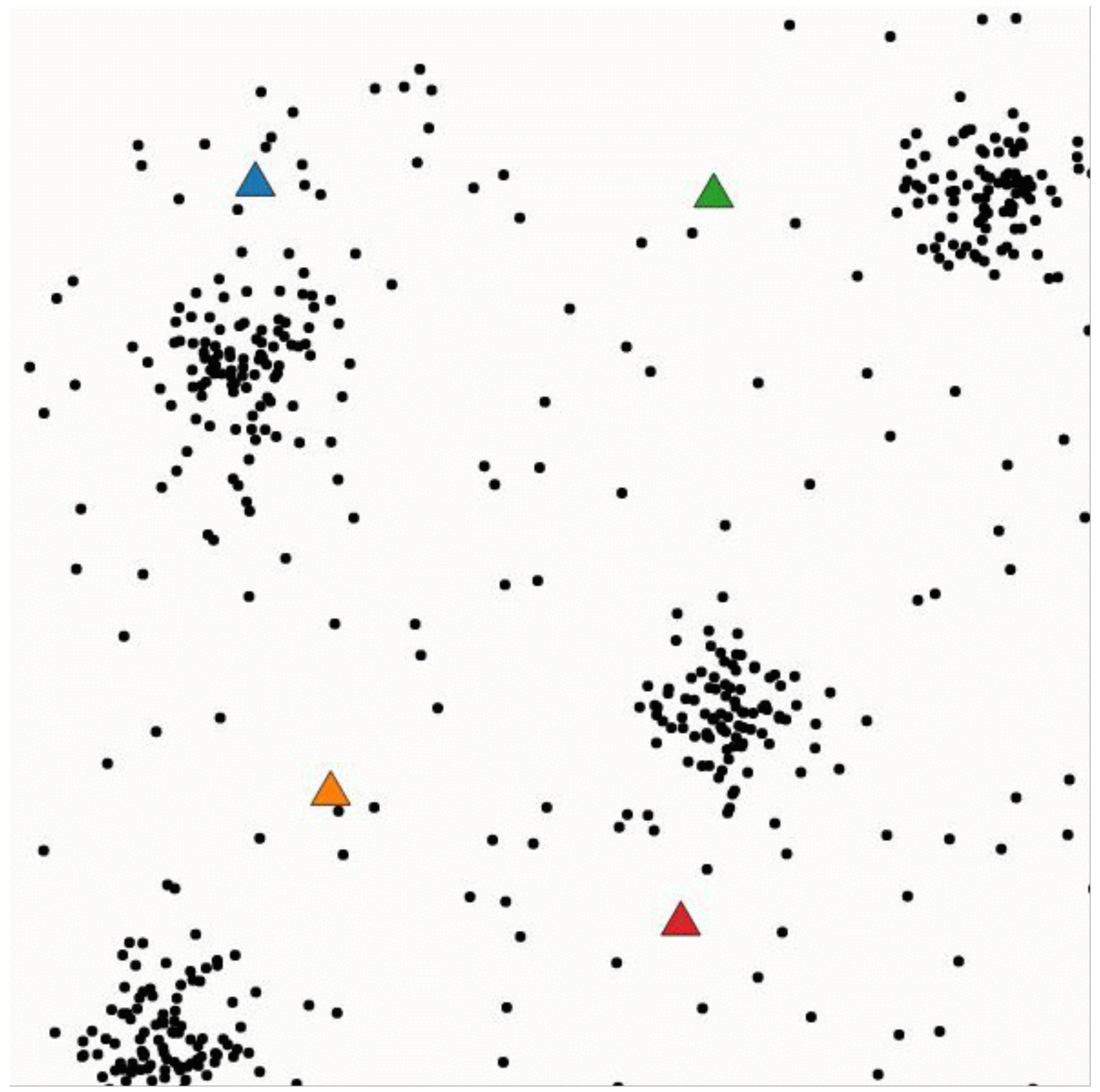
python                                                                 Copy code

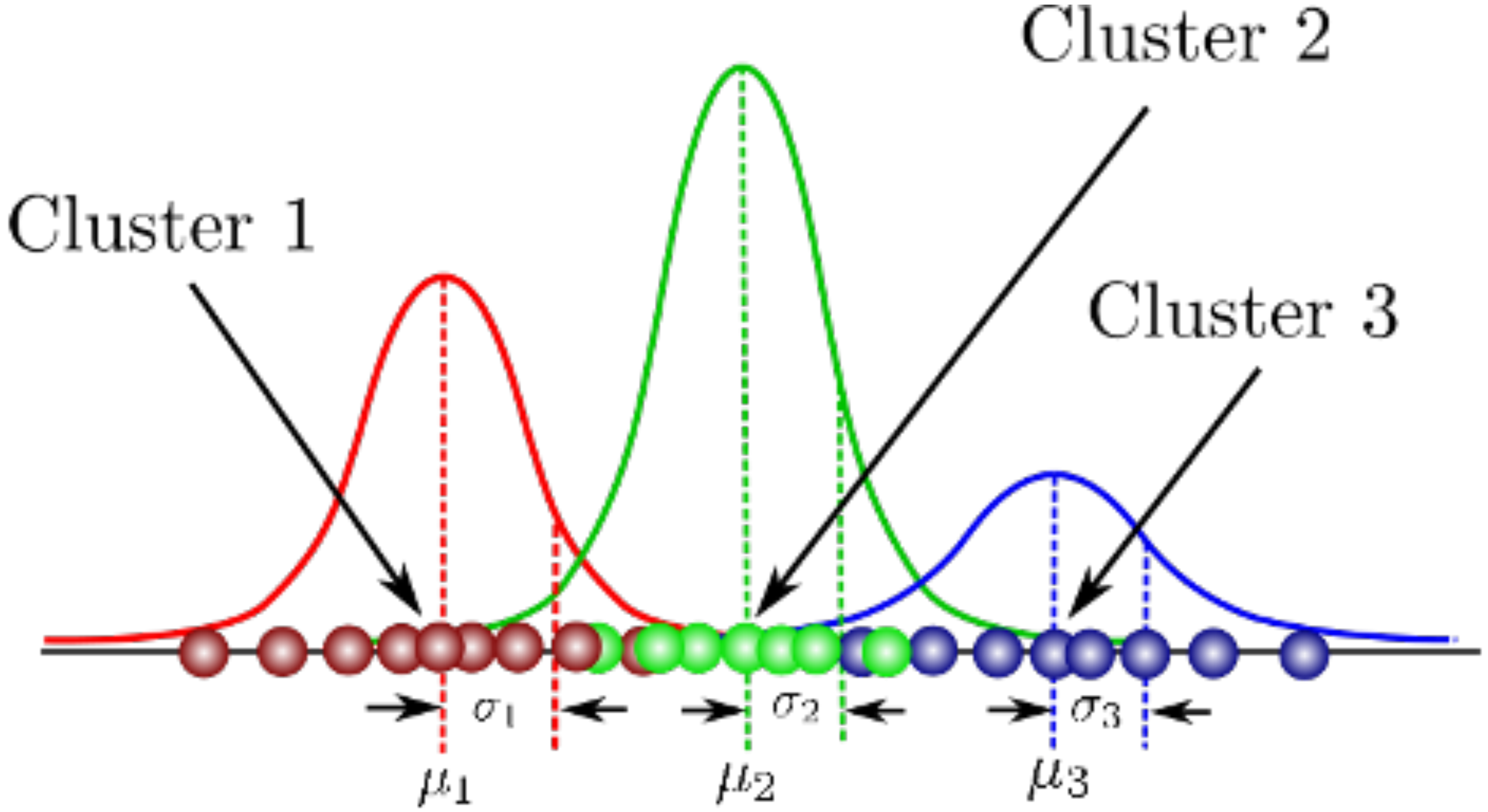
```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
```

# K-means clustering

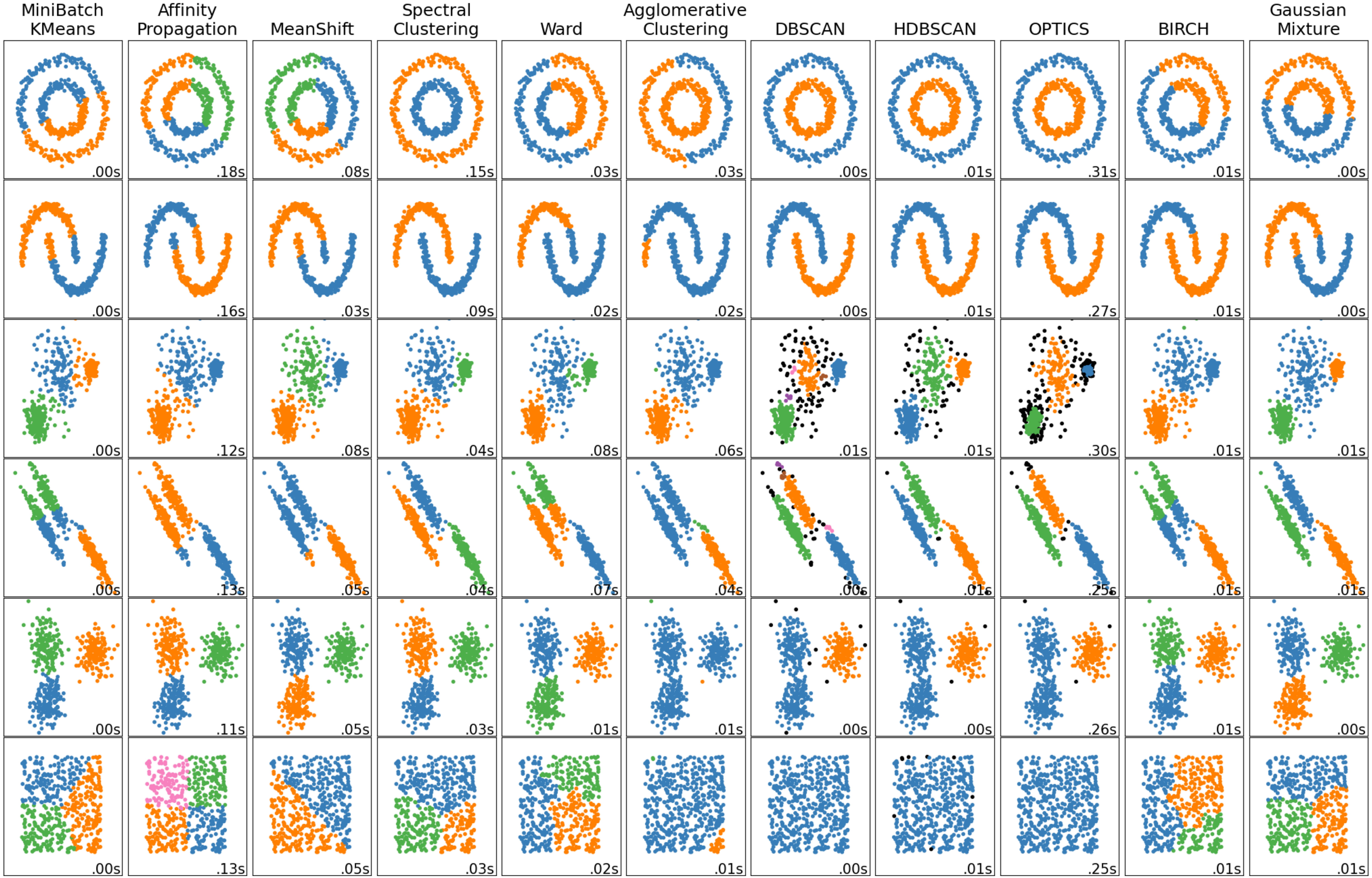
# Clustering

- Partitioning a set of ***unlabeled*** data points into pre-specified #groups
    - K-means, Gaussian mixture models, Hierarchical, Spectral, …
    - Implicitly assumes some notion of "similarity"





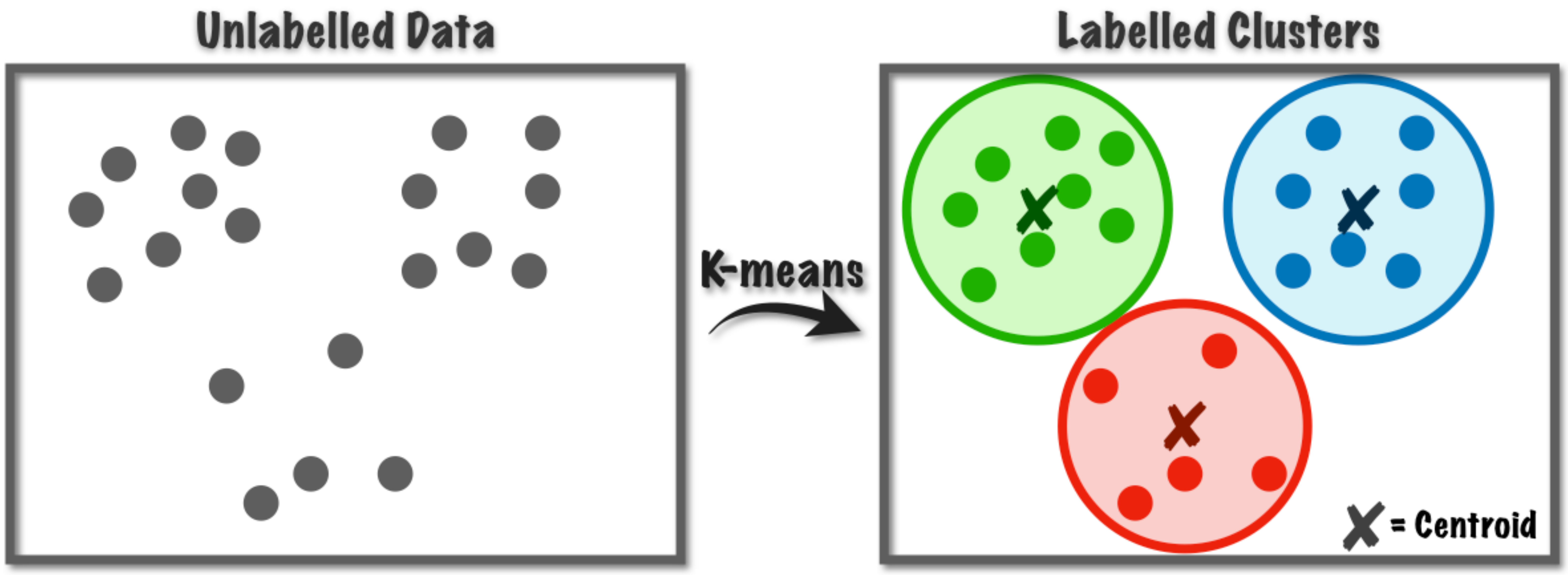

https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html




https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

# K-Means

- Assign each data point to one of K clusters.
  - Each cluster is represented by a single point, called centroid.
  - The loss is measured by the distance(data,centroid).



# K-Means

- Suppose that we have a dataset $D = \{\mathbf{x}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d.$
- We make **two decisions**:
  - We make $K$ clusters.
    - We decide the corresponding centroids $\mu_1, \ldots, \mu_K \in \mathbb{R}^d.$
  - We assign each data to clusters.
    - We decide the assignment $r_{ik} \in \{0,1\}, \quad \sum_{k=1}^{K} r_{ik} = 1$

      ($r_{ik} = 1$ means $\mathbf{x}_i$ belongs to $k$-th cluster)

# K-Means

- We want to choose nice $\{\mu_k\}$ and $\{r_{ik}\}$ so that we can minimize the mean squared distance from data point to the centroid, i.e.,

$$\min_{\{\mu_k\}} \min_{\{r_{ik}\}} \sum_{i=1}^{n} r_{ik} \|\mathbf{x}_i - \mu_k\|_2^2$$

(One can use other distances instead of $\|\cdot\|_2^2$)

- How do we solve this optimization problem?

# Principle #1: Centroid -> Assignment

*"Always assign to a cluster with the closest centroid"*

- Given the centroids, the optimal assignment is obvious:

$$r_{ik} = \begin{cases} 1 & \cdots \quad k = \text{argmin}_k \|\mathbf{x}_i - \mu_k\|_2^2 \\ 0 & \cdots \quad \text{otherwise} \end{cases}$$

# Principle #2: Assignment -> Centroid

*"Select the average of assigned points as a centroid"*

- Given the assignments, the optimal centroid is obvious:
  - If $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(n_k)}$ are the data assigned to $k$-th cluster,

$$\mu_k = \mathrm{argmin}_{\mu \in \mathbb{R}^d} \sum_{i=1}^{n_k} \|\mu - \mathbf{x}_{(i)}\|_2^2$$

$$= \frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_{(i)}$$

(check!)

# The Famous K-Means Algorithm

- The optimal solution should satisfy both principles:
  - P1: Data points are assigned to nearest centroids.
  - P2: Centroids should be the average of assigned points.

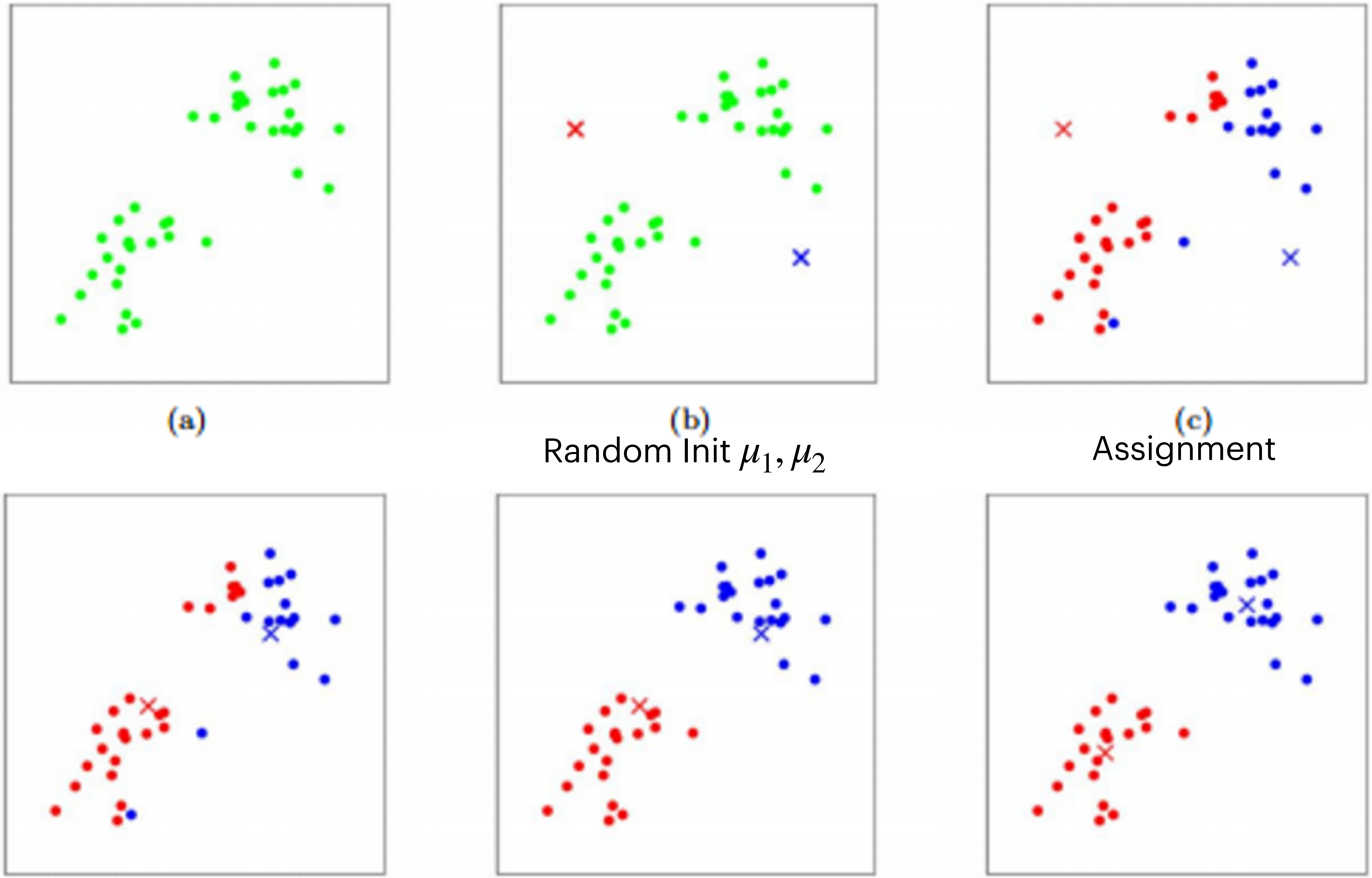
- **Question.** How do we find a solution that satisfies P1 and P2?

# The Famous K-Means Algorithm

- **A working way.** Apply P1, Apply P2, Apply P1, ..., until convergence.
  - Assignment Step: Given $\{\mu_k\}$, find $\{r_{ik}\}$.
  - Update Step: Given $\{r_{ik}\}$, find $\{\mu_k\}$.

**Algorithm 1** $k$-means algorithm

Specify the number $k$ of clusters to assign.
Randomly initialize $k$ centroids.
**repeat**
    **expectation:** Assign each point to its closest centroid.
    **maximization:** Compute the new centroid (mean) of each cluster.
**until** The centroid positions do not change.



(a)

(b) Random Init $\mu_1, \mu_2$

(c) Assignment

(d) Update

(e) Assignment

(f) Update

**Original image**

**k = 3**

**k = 8**

**k = 13**

**k = 20**

**k = 40**

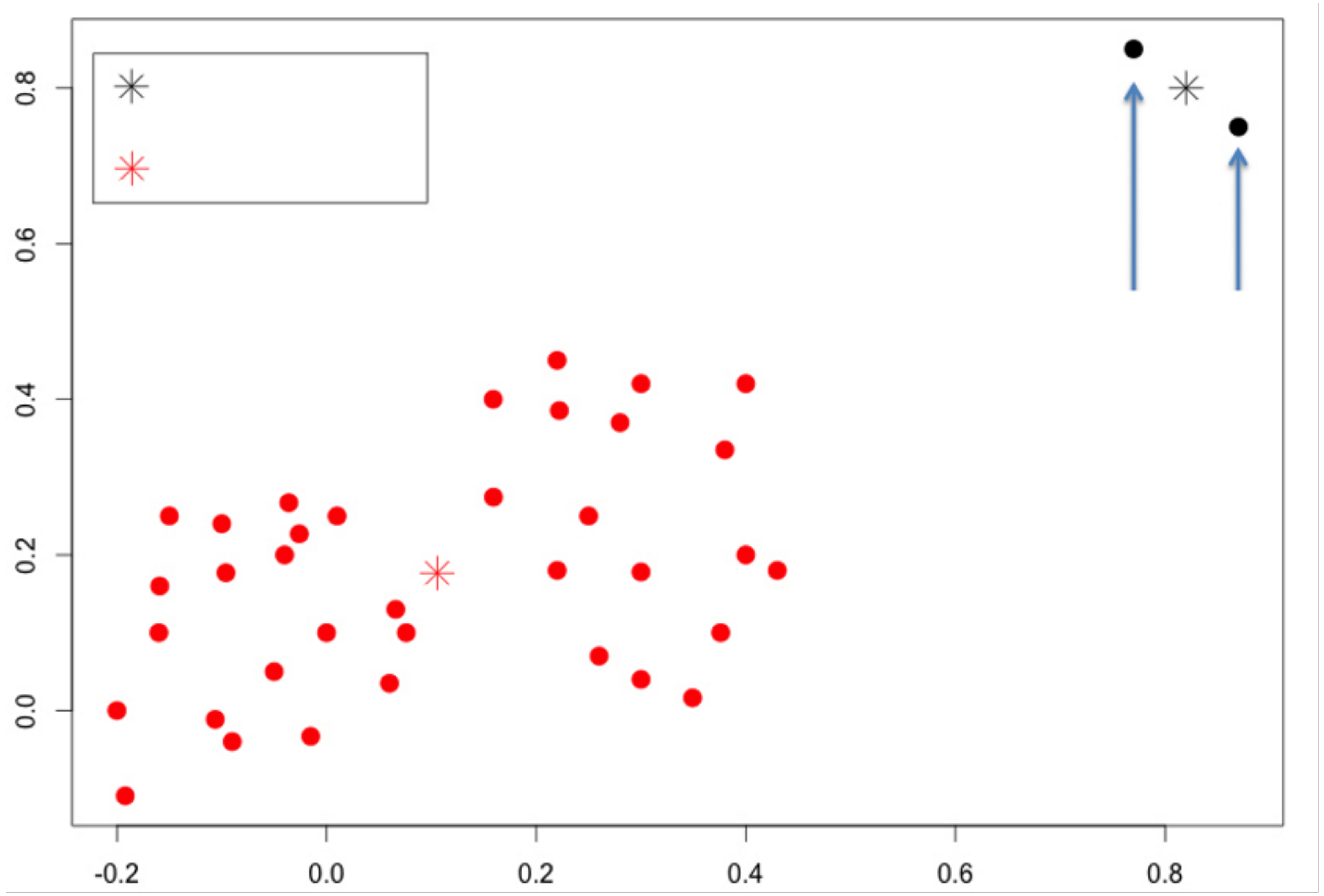
Data Point = Each Pixel's RGB

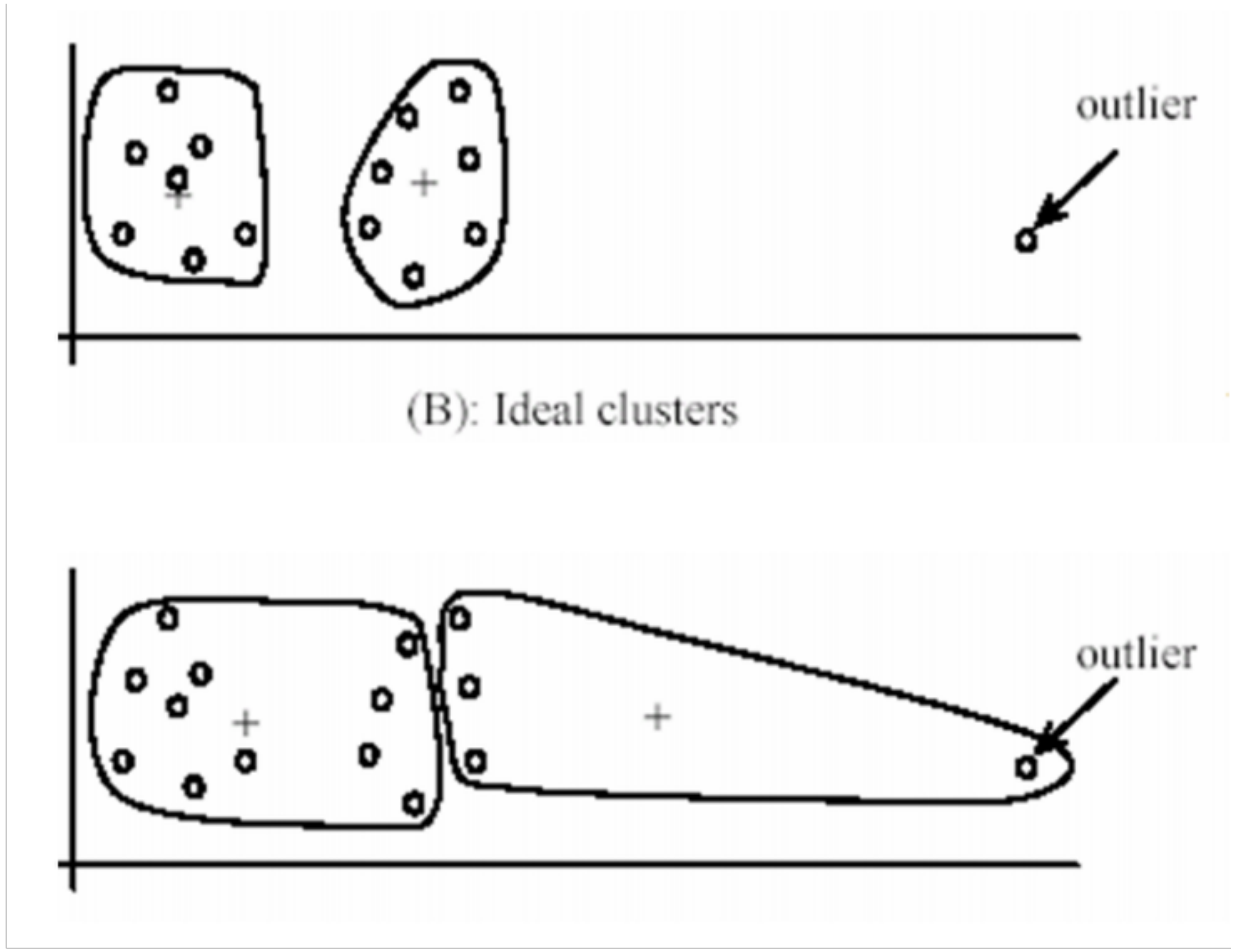
# Properties of K-Means

- Local optimum is found.
    - Sensitive to initialization—use K-means++ for better results
- Convergence within finite number of iterations is guaranteed.
- Computational Complexity.
    - **Assignment.** $\mathcal{O}(d \cdot k \cdot n)$
    - **Update**. $\mathcal{O}(n)$

# Limitations

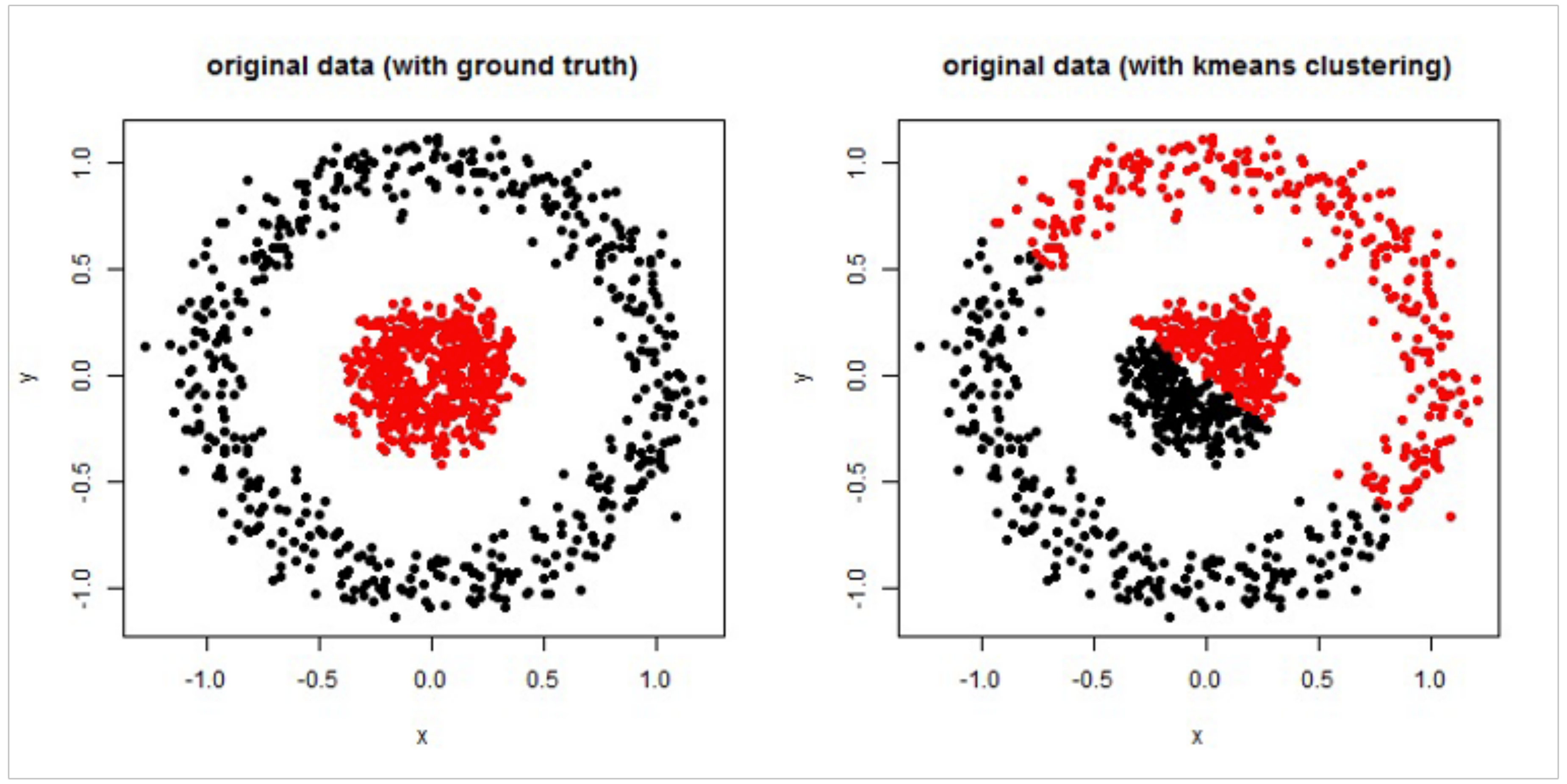
- Quite sensitive to outliers

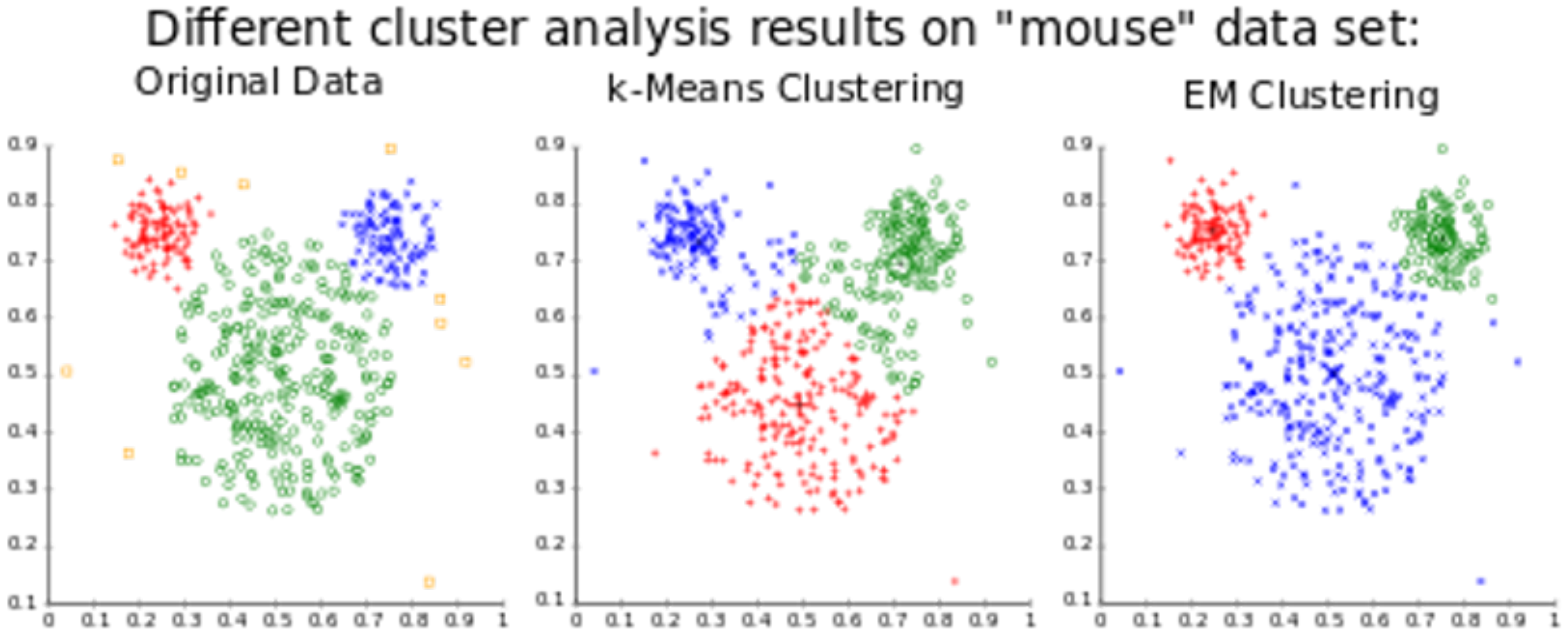


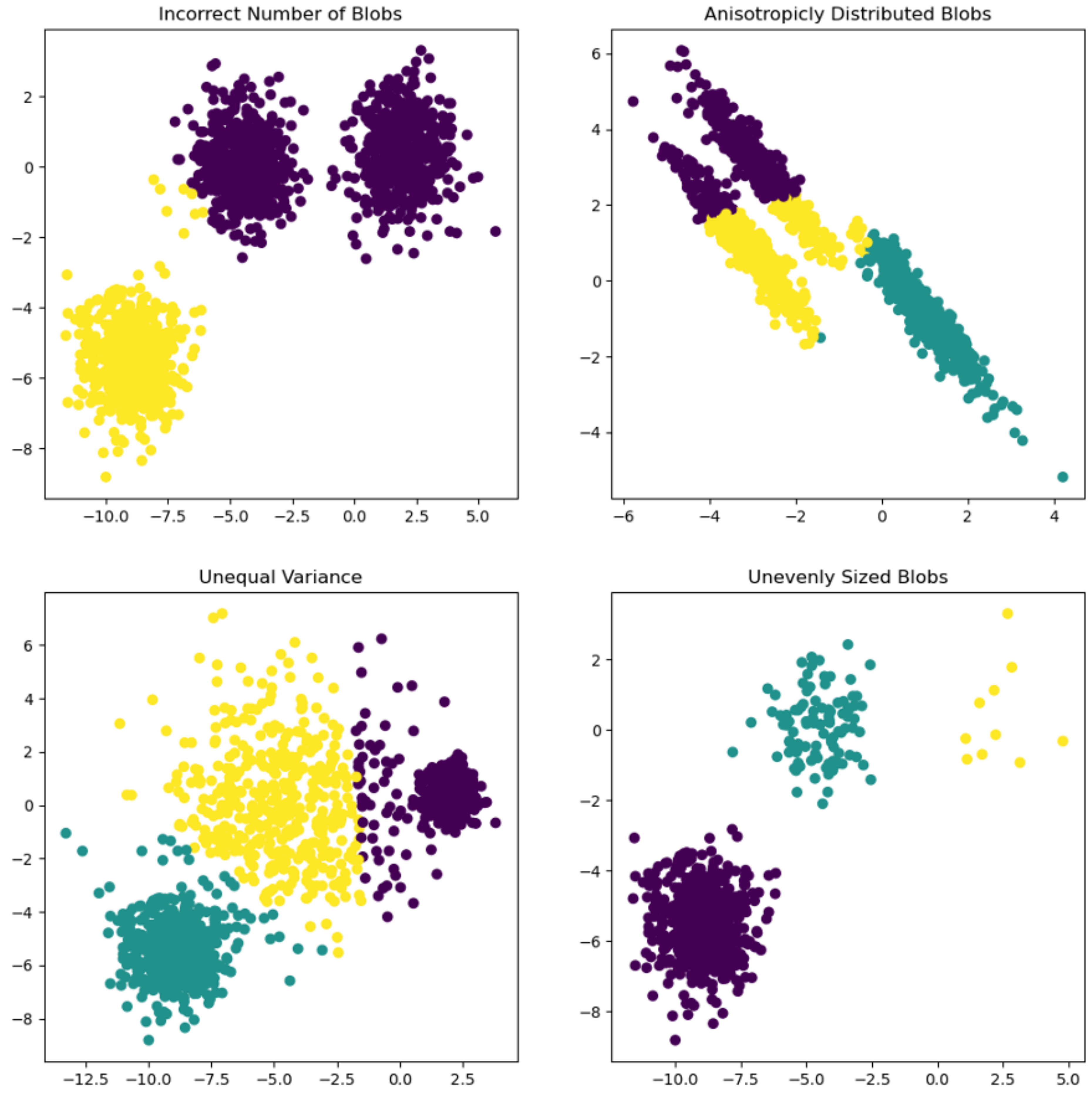



# Limitations

- May not work for certain datasets



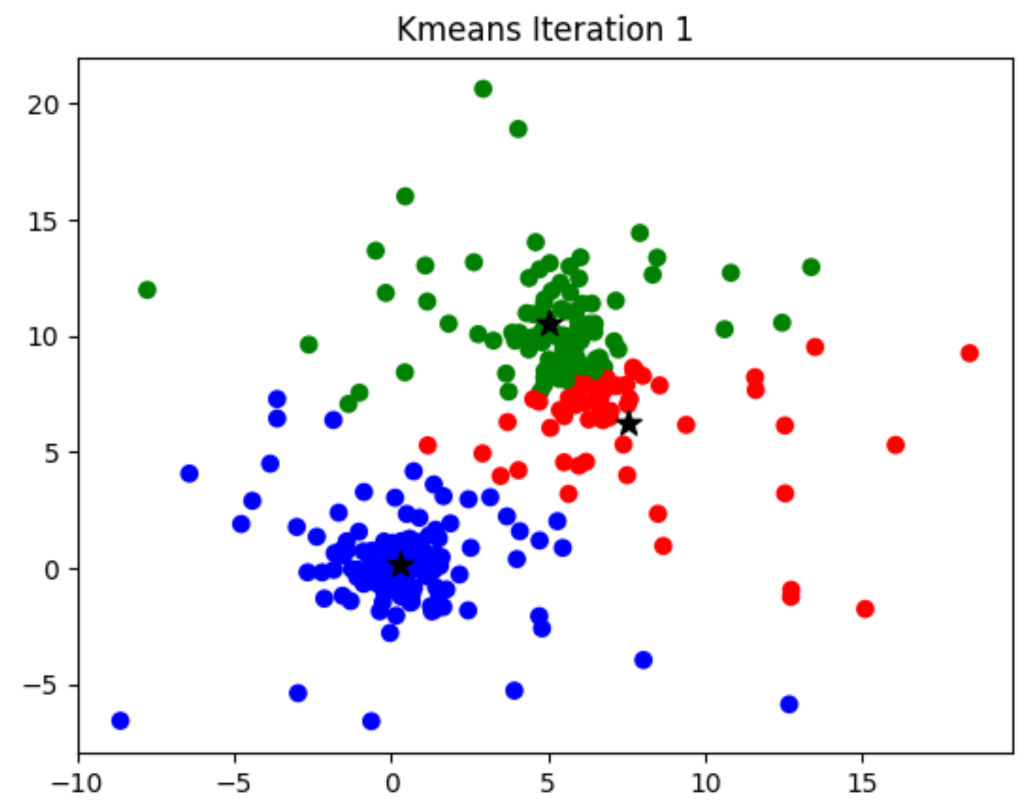



# Soft K-Means

# Soft Clustering

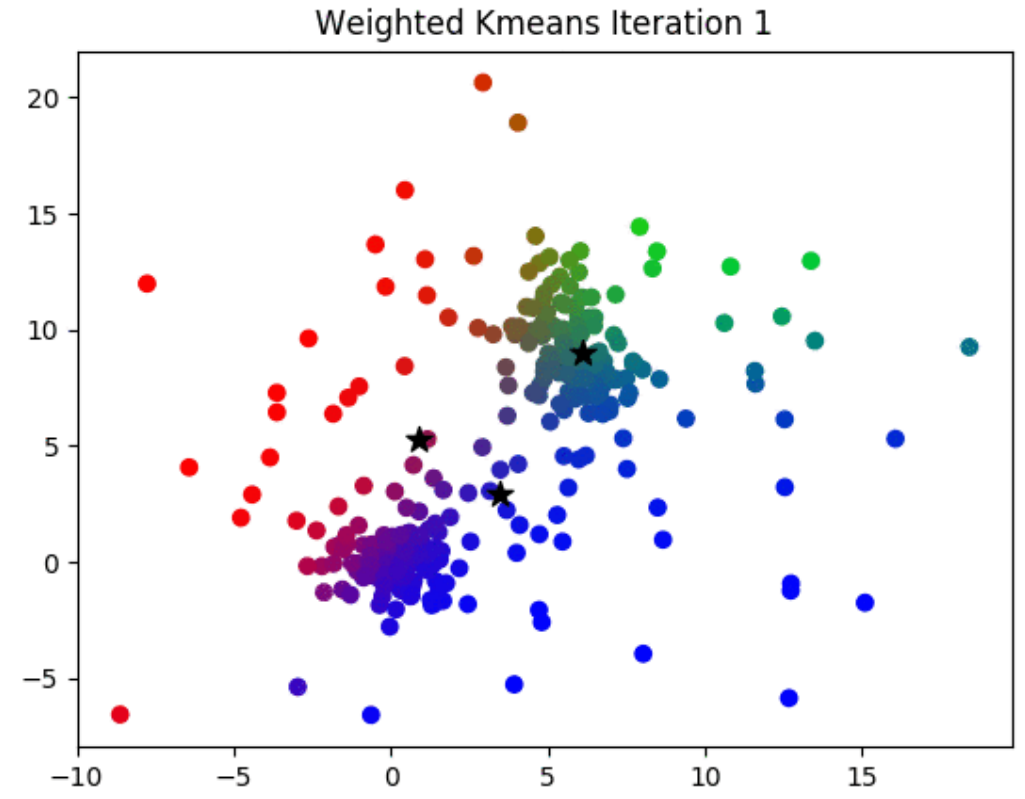
- Clustering, but the assignment is soft.
  - **Hard.** A point 100% belongs to a specific cluster

$$r_{ik} \in \{0,1\}, \quad \sum_{k=1}^{K} r_{ik} = 1$$




  - **Soft.** A point may 90% belong to one, and 10% to another, ...

$$r_{ik} \in [0,1], \quad \sum_{k=1}^{K} r_{ik} = 1$$



# The Soft K-Means Algorithm

- **Assignment.** The larger “responsibility” for closer point, with some $\beta$.

$$r_{ik} = \frac{\exp(-\beta\|\mathbf{x}_i - \mu_k\|_2^2)}{\sum_j \exp(-\beta\|\mathbf{x}_i - \mu_j\|_2^2)}$$

**Note.** This becomes the hard assignment

$$r_{ik} = \begin{cases} 1 & \cdots \quad k = \operatorname{argmin}_k \|\mathbf{x}_i - \mu_k\|_2^2 \\ 0 & \cdots \quad \text{otherwise} \end{cases}$$

if we let $\beta \to \infty$ (thus called softmax).

# The Soft K-Means Algorithm

- **Update.** A weighted average of data, where the weight comes from the responsibility.

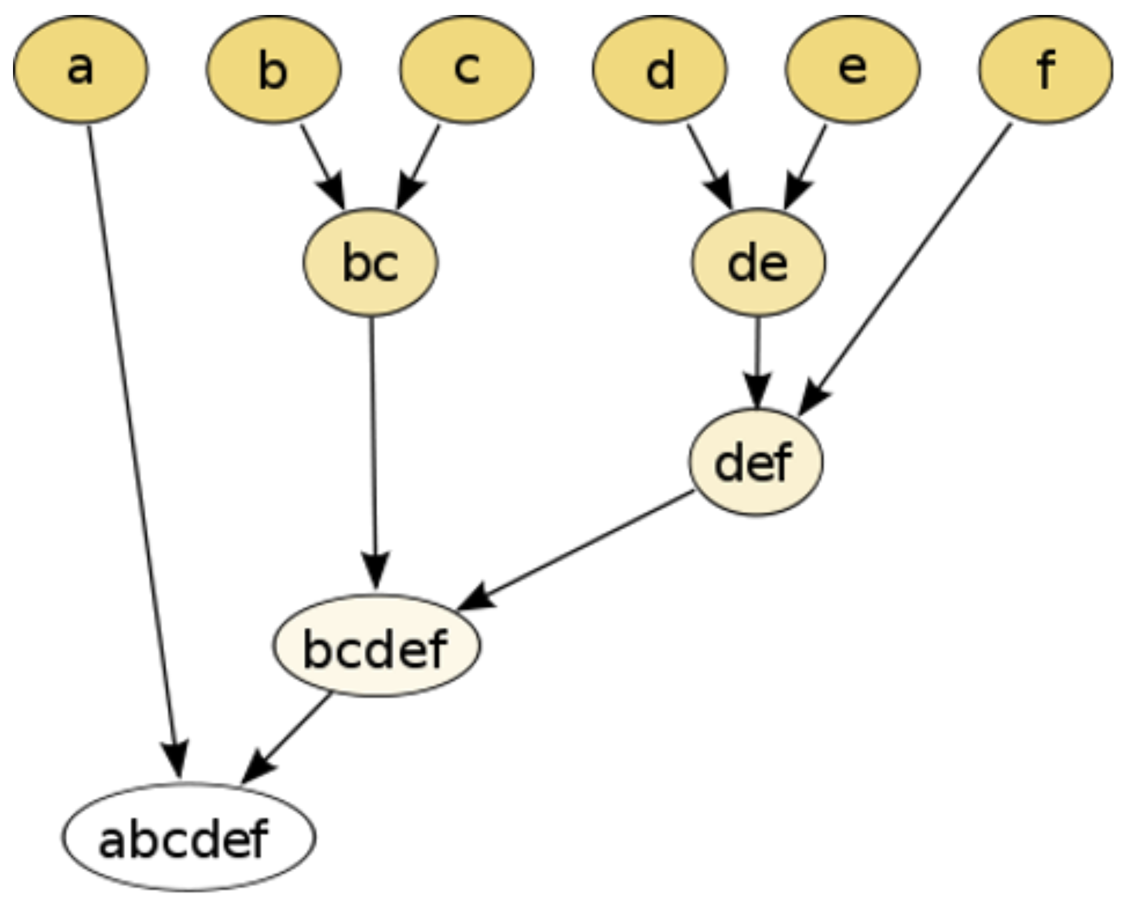
$$\mu_k = \frac{\sum_i r_{ik} \mathbf{x}_i}{\sum_j r_{jk}}$$

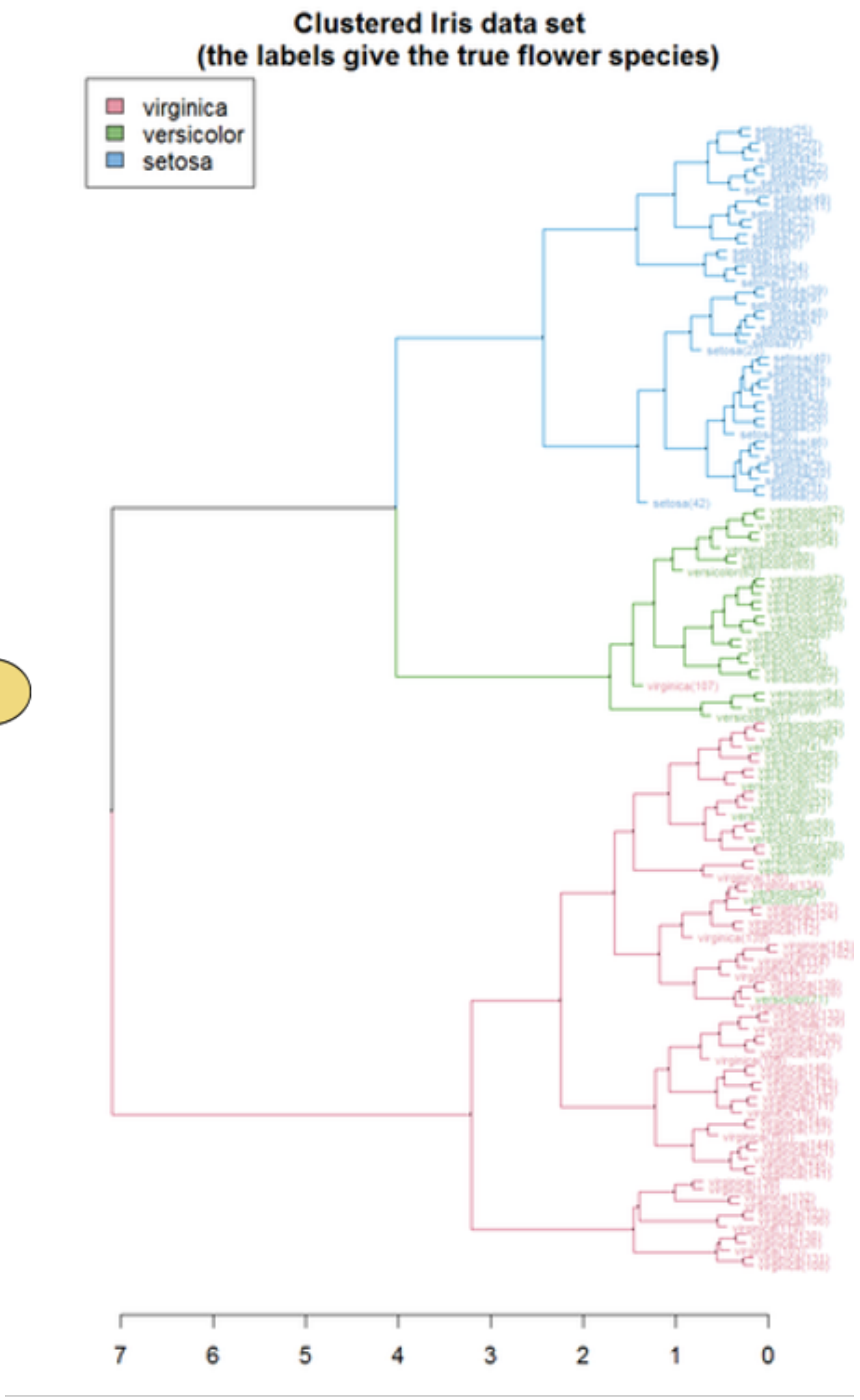
(can be derived similarity as in hard K-means)

# Others

# Hierarchical Clustering

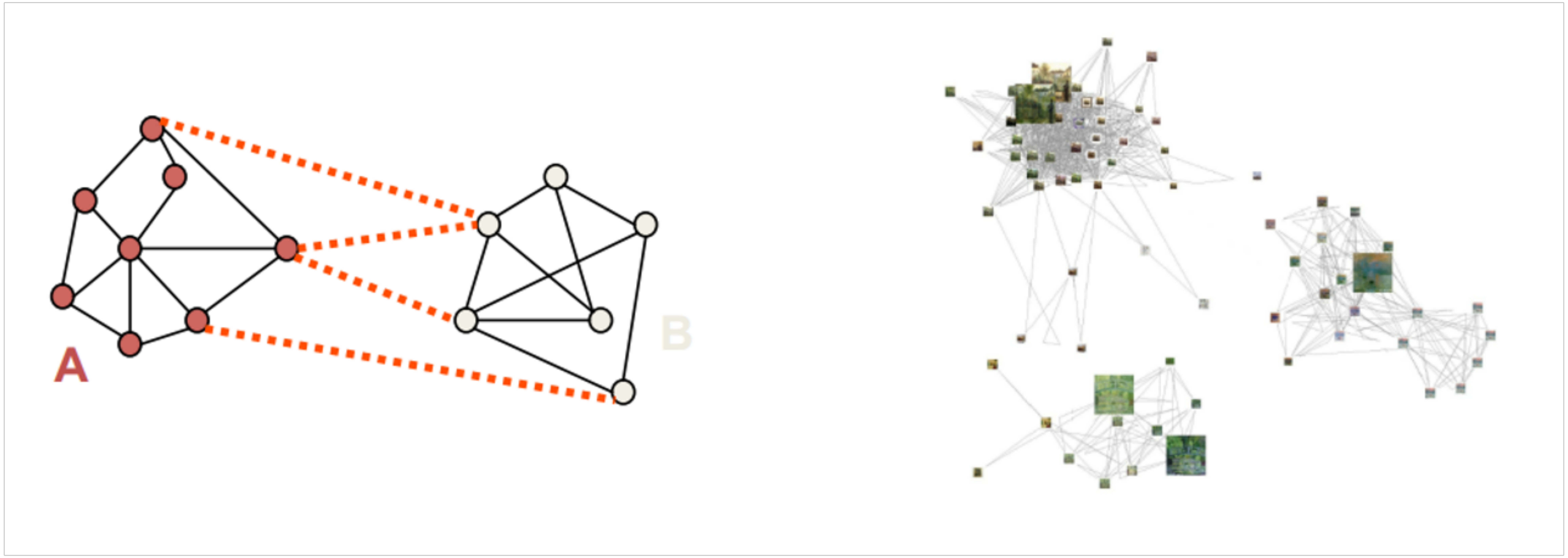
- Clusters inside clusters
  - Discovers hierarchical structures
  - Relax strict assumptions (e.g., distributions)
  - Leverage faster heuristic algorithms
  - Waive strict decision of $K$.
- Two ways to construct—
  - Divisive: Top-Down
  - Agglomerative: Bottom-up.



# Spectral Clustering

- Data lies on graph—similarity by distance on graph.
  - Solve via graph algorithms, e.g., min-cut



# Cheers

- *Next up.* Mixture models