

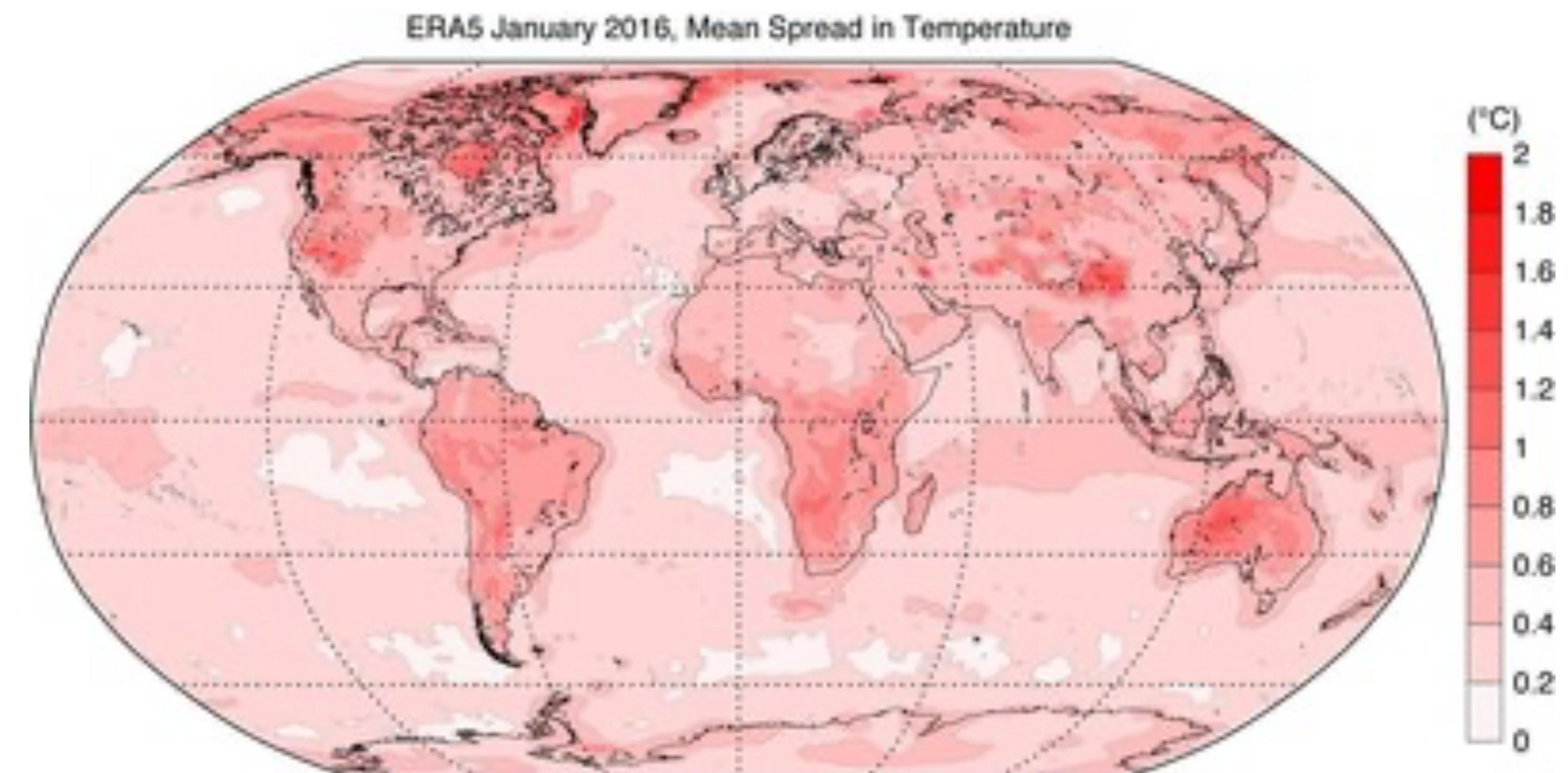
# K-Means Clustering

# Recap

- So far, we have discussed **supervised learning**

- Given. A labeled dataset  $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Goal. Learn  $f(\cdot)$  such that  $f(\mathbf{x}) \approx y$

- $\mathbf{x}$ : Time & Location
- $y$ : temperature
- Predict temperature at new time & location



# **Unsupervised Learning**

# Unsupervised Learning

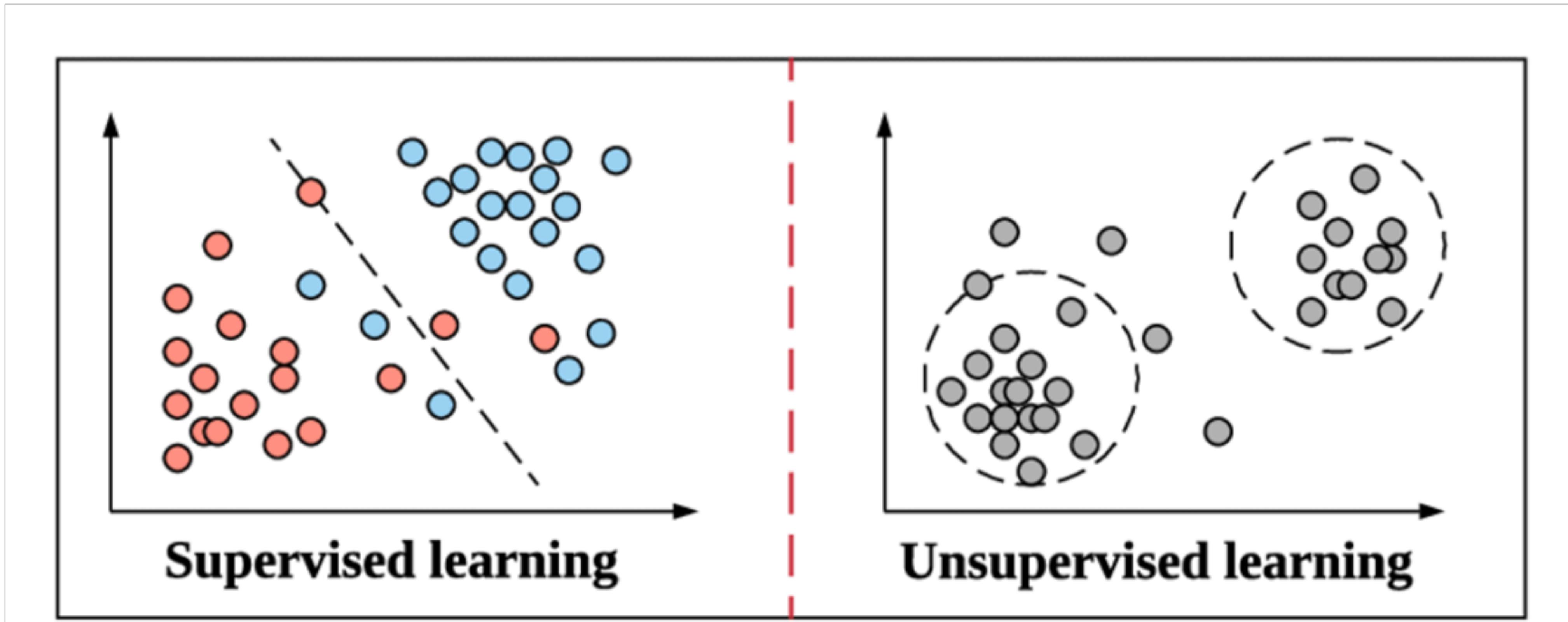
- Given. An **unlabeled dataset**  $D = \{\mathbf{x}_i\}_{i=1}^n$ 
  - No labeling cost needed
    - Can utilize very large datasets!
  - Examples. Most web-crawled datasets
    - Text on Web
      - GPTs are being trained on these!
    - Images on Flickr
    - Codes on GitHub
    - Videos on YouTube

Language	# tokens	# words
Russian	102,825,040,945	14,679,750
Japanese	92,827,457,545	9,073,245
Spanish	72,493,240,645	10,614,696
French	68,358,270,953	12,488,607
German	65,648,657,780	19,767,924
Italian	36,237,951,419	10,404,913
Portuguese	35,841,247,814	8,370,569
Chinese	30,176,342,544	17,599,492
Polish	21,859,939,298	10,209,556
Czech	13,070,585,221	8,694,576
Finnish	6,059,887,126	9,782,381
Hindi	1,885,189,625	1,876,665

Table 3: Comparison across languages of the size of the datasets obtained using the Common Crawl. The second column indicates the vocabulary size of the models trained on this data.

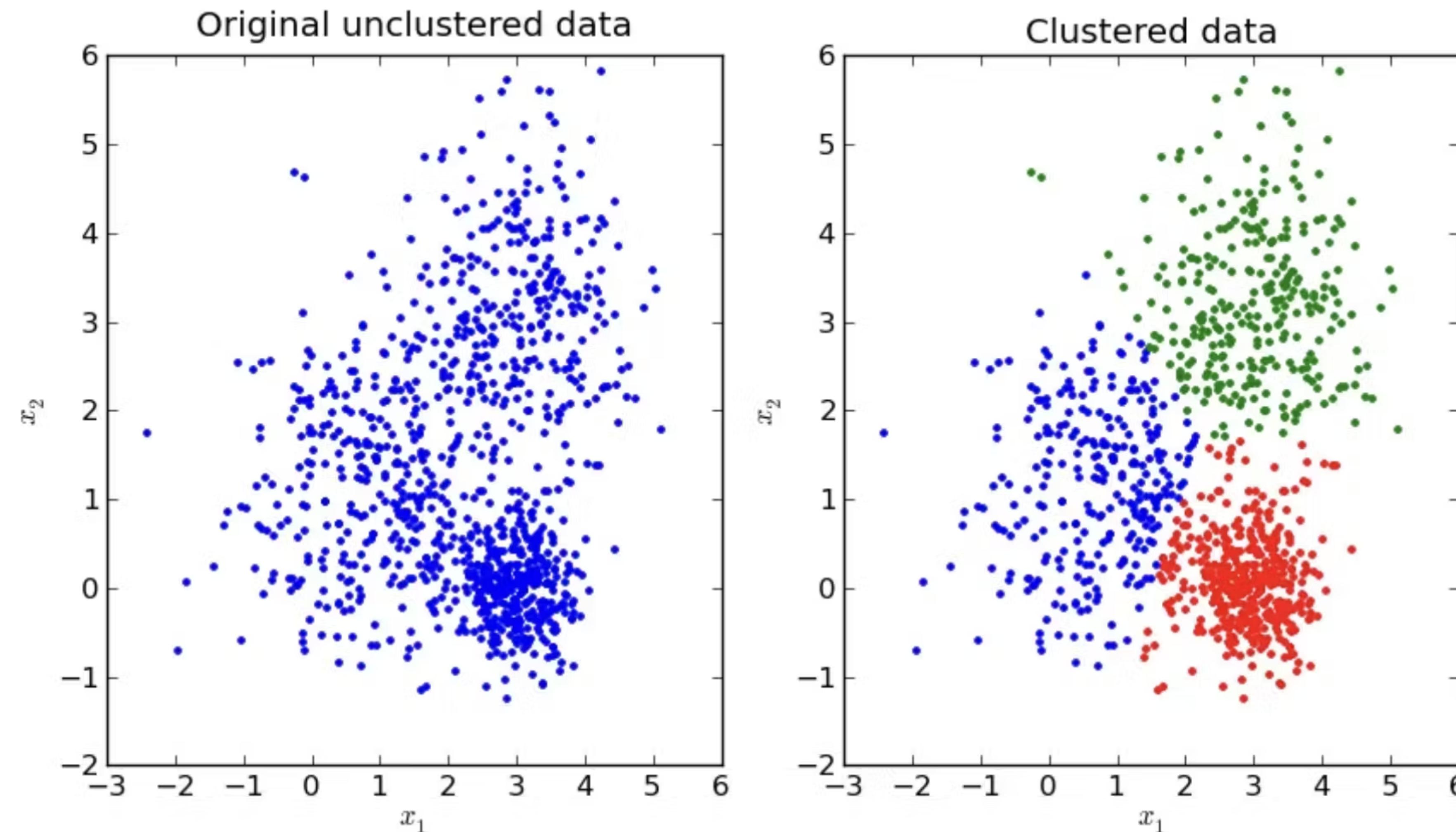
# Unsupervised Learning

- Goal. Get insights from data – underlying **structures, causality, relation**
  - Learned structures can be used for supervised learning tasks
    - e.g., learning a feature map  $\Phi(\cdot)$



# What can unsupervised learning do?

- 1957. People were clustering data points for classification



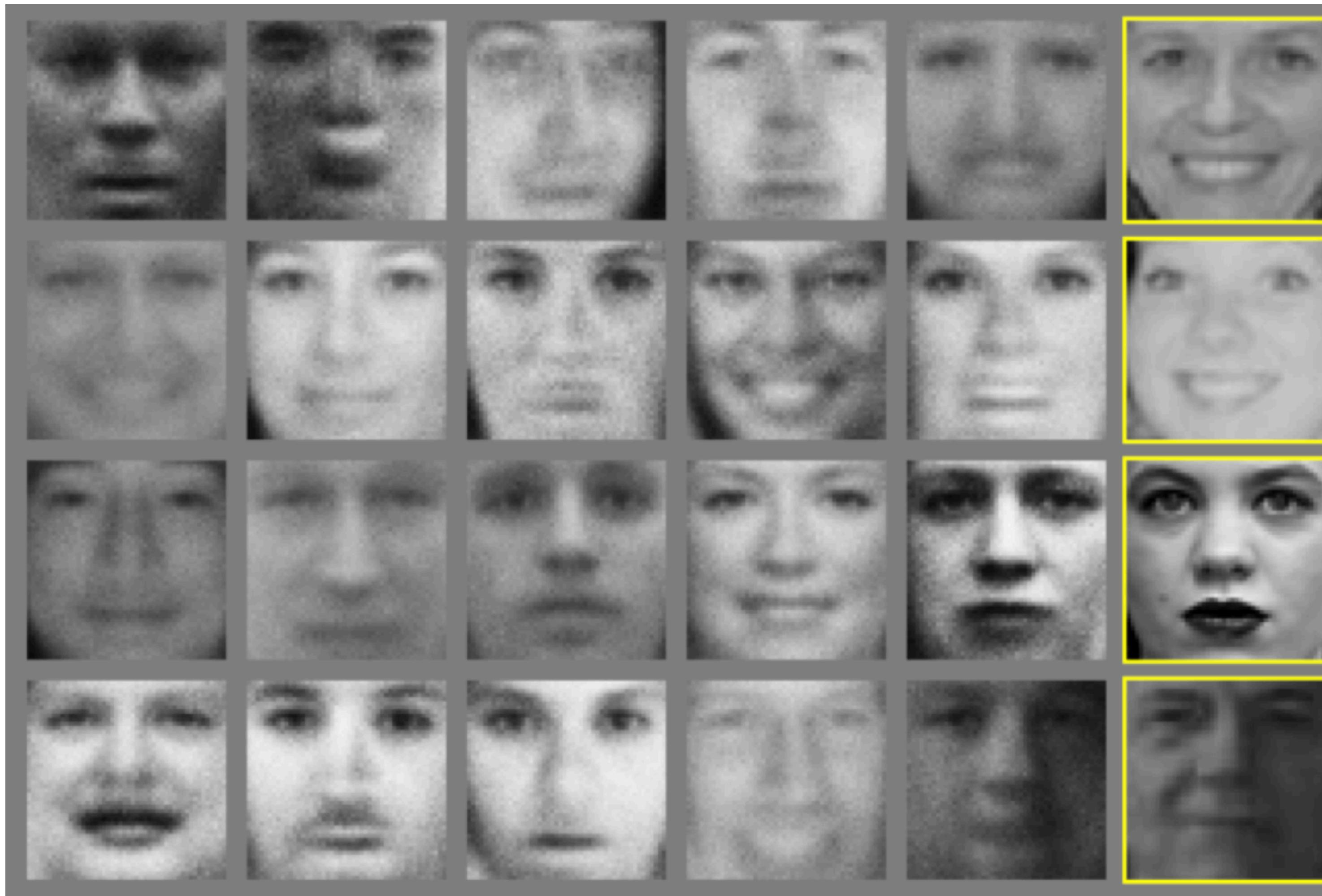
# What can unsupervised learning do?

- 2012. Discovered patterns from YouTube videos, without any supervision
  - Useful for classification & generation



# What can unsupervised learning do?

- 2014. Used face images to generate new faces



# What can unsupervised learning do?

- 2025. Multimodal generator with rich knowledge

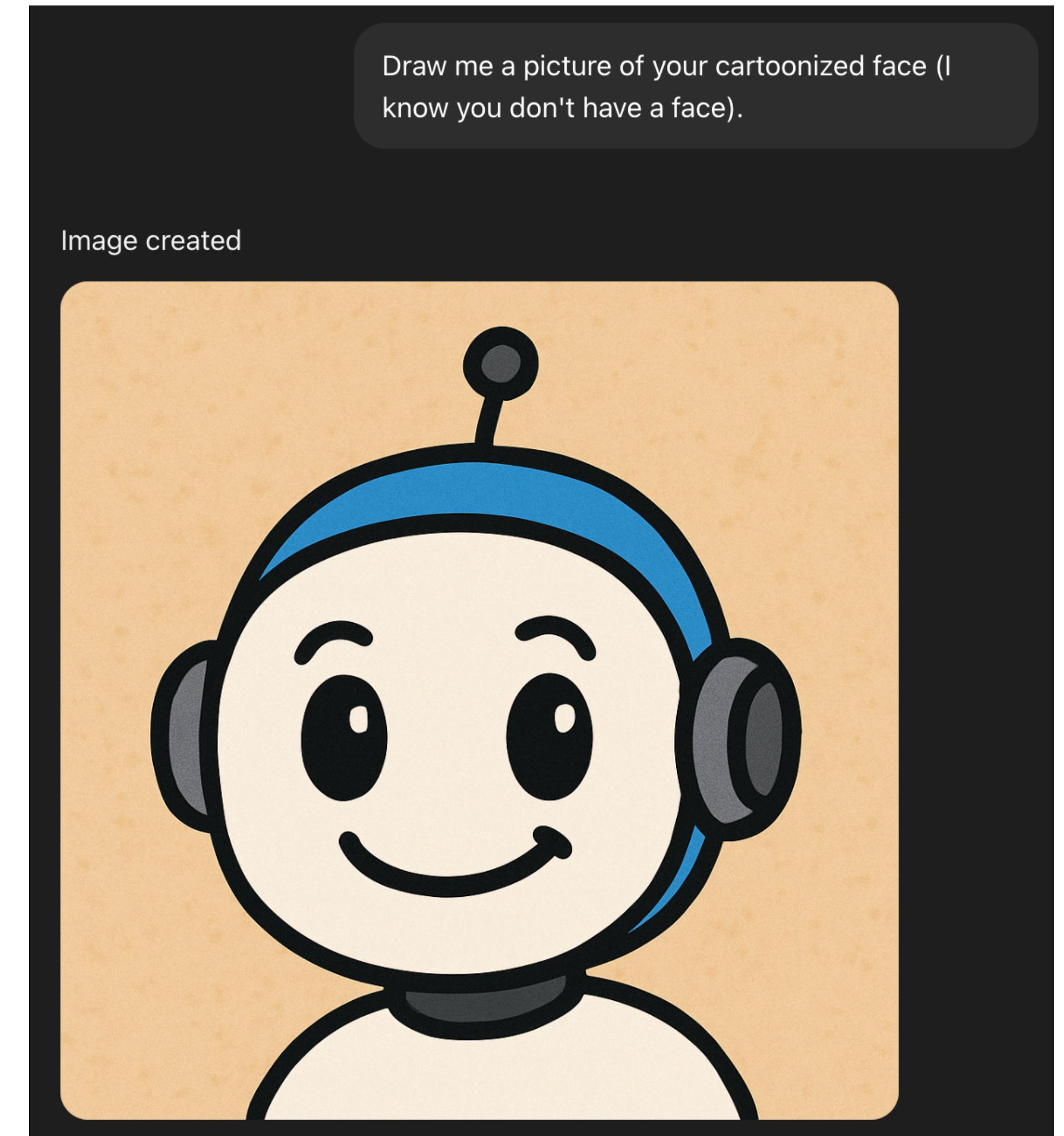
User

What is the pH of a 0.10 M solution of  $\text{NH}_4\text{F}$ ? The  $K_a$  of  $\text{NH}_4^+$  is  $5.6 \times 10^{-10}$  and the  $K_a$  of HF is  $6.8 \times 10^{-4}$ .

Hide chain of thought ^

First, let's understand what is being asked.

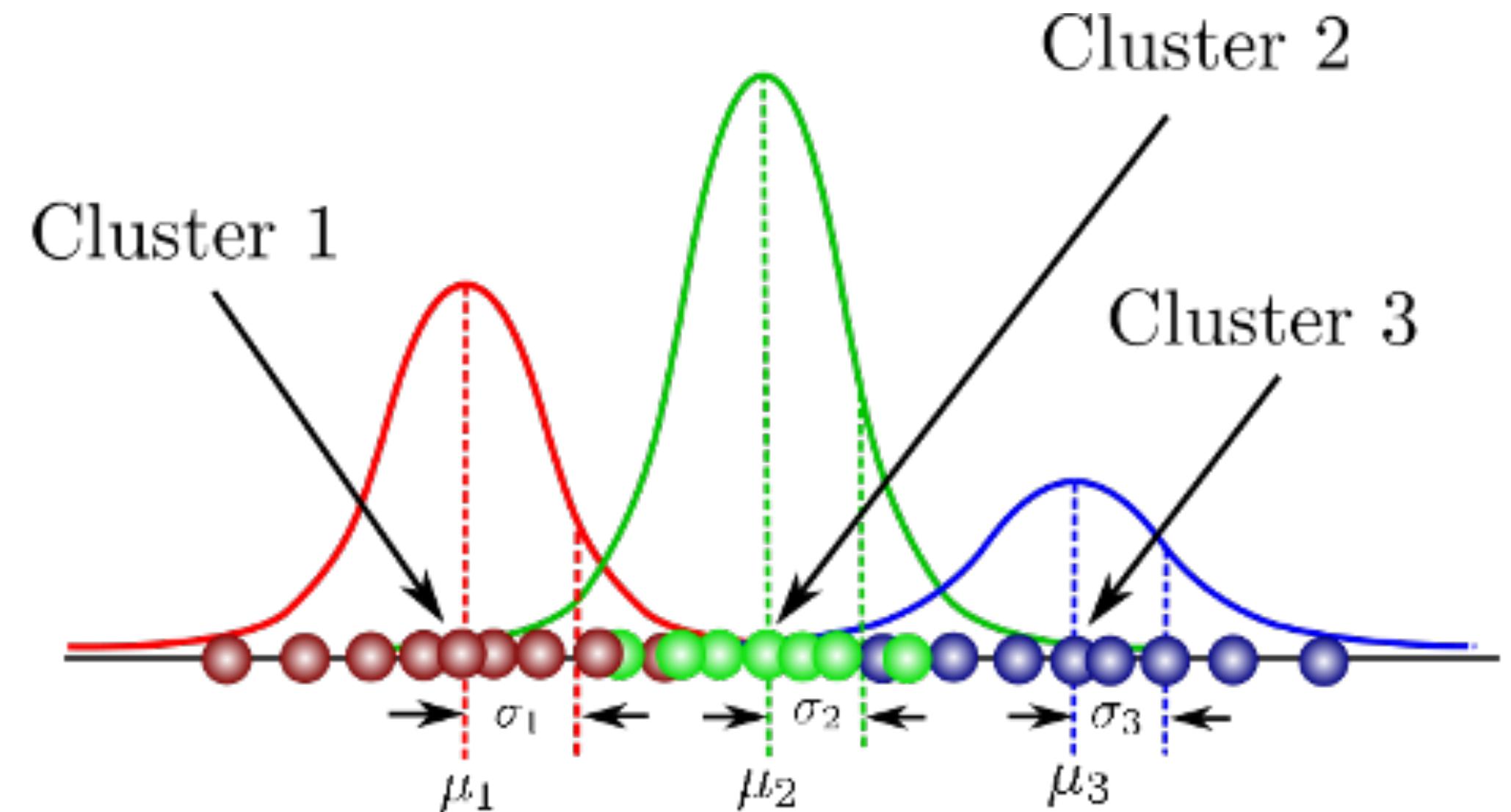
We need to find the pH of a 0.10 M solution of ammonium fluoride,  $\text{NH}_4\text{F}$ .



# K-Means Clustering

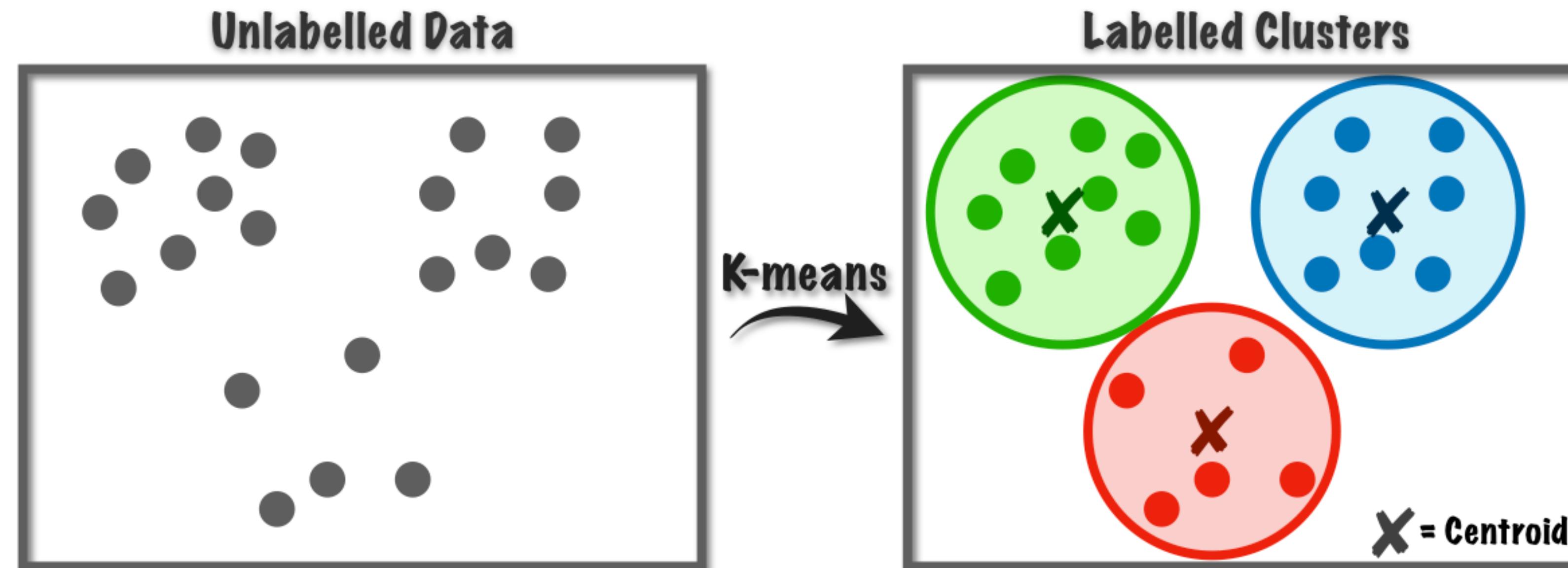
# Clustering

- Assigning **(imaginary) group identities** to a set of unlabeled data points
  - K-Means
  - Hierarchical / Spectral Clustering
  - Gaussian Mixture Models
- Requires some notion of similarity
  - e.g., geometric distance between data
- Maximize the intra-group similarity
  - Optionally, minimize the inter-group similarity



# K-Means

- Each cluster is represented by a **single point** in space
  - called “centroid”
  - The loss is measured by
$$\text{dist}(\text{data}, \text{centroid})$$
  - Centroid-data similarity as a proxy for intra-group similarity



# K-Means

- Formally, suppose that we have a dataset

$$D = \{\mathbf{x}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d$$

- We will make  $K$  clusters, by making two decisions
  - Decide their **centroids**

$$\mu_1, \dots, \mu_K \in \mathbb{R}^d$$

- Decide the **assignments** (1: belonging, 0: otherwise)

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

# K-Means

- **Goal.** Choose nice  $\{\mu_k\}$ ,  $\{r_{ik}\}$  so that we can minimize the **mean-squared distance** of each data to centroids

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

- Joint minimization of two variables
  - Discrete: Assignments
  - Continuous: Centroids
- **Problem.** Cannot apply a useful tool: critical point analysis

# Strategy

- More generally, can be written as solving

$$\min_x \min_y f(x, y)$$

- **Typical strategy.** Tackle two subproblems separately:

- Fix  $x$  and optimize  $y$
- Fix  $y$  and optimize  $x$
- Repeat

- Known as **alternating minimization** procedure

# Strategy

- More formally, define the following algorithm:

Initialize  $x^{(0)}, y^{(0)}$

For  $t = 1, 2, \dots$ , repeat:

$$x^{(t)} = \arg \min_x f(x, y^{(t-1)})$$

$$y^{(t)} = \arg \min_y f(x^{(t)}, y)$$

- If we assume that  $f(x, y) \geq 0$ , we can prove that
  - $f(x^{(i+1)}, y^{(i+1)}) \leq f(x^{(i)}, y^{(i)})$
  - the loss converges

# Strategy

- **Proof Sketch.** 
- Note. Not guaranteeing that it will converge to optimum!

# Algorithm

- Come back to the problem of K-means

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

- Applying the alternating minimization, we have:
  - Fix  $\{\mu_k\}$  and optimize  $\{r_{ik}\}$
  - Fix  $\{r_{ik}\}$  and optimize  $\{\mu_k\}$
  - Repeat!

# Algorithm

- Thankfully, each subproblem is easy to solve
- **Subproblem 1.** Optimize assignment  $\{r_{ik}\}$ , given centroids  $\{\mu_k^{(t-1)}\}$

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

- **Solution.** Assign the nearest centroid:

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

# Algorithm

- **Subproblem 2.** Optimize centroids  $\{\mu_k\}$  assignment  $\{r_{ik}^{(t)}\}$ , given

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

- **Solution.** Take an average of all assigned points:

- If  $\mathbf{x}_{(1)}, \dots, \mathbf{x}_{(n_k)}$  are assigned to cluster  $k$ , then

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

# Algorithm

- Summing up, the final algorithm becomes:

---

## Algorithm 1 $k$ -means algorithm

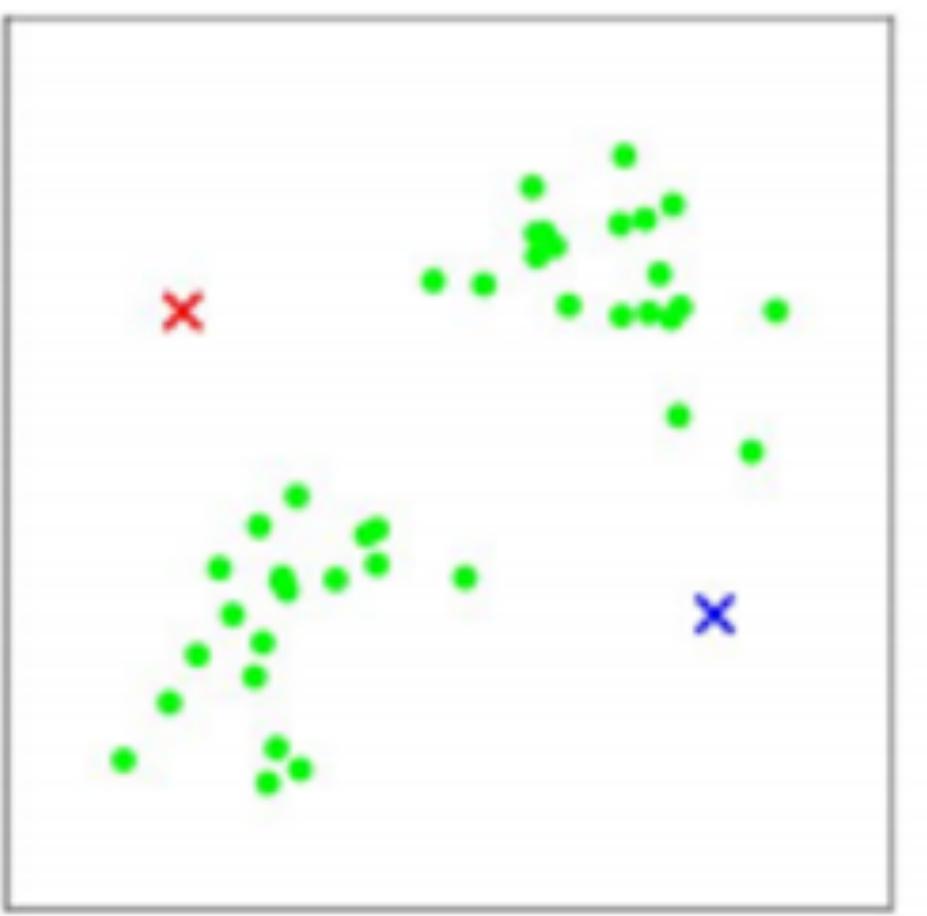
---

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

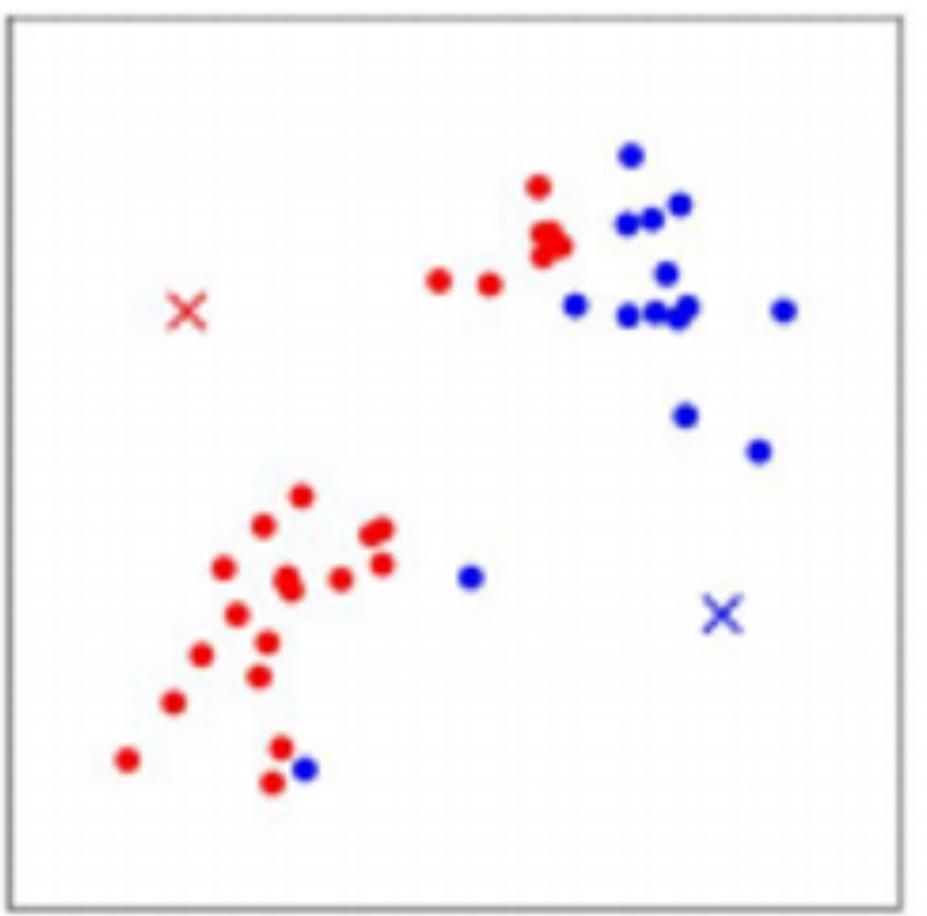
- Known as **Lloyd's algorithm** (or simply K-means)
  - Originally proposed for pulse-code modulation
  - Special case of expectation-maximization



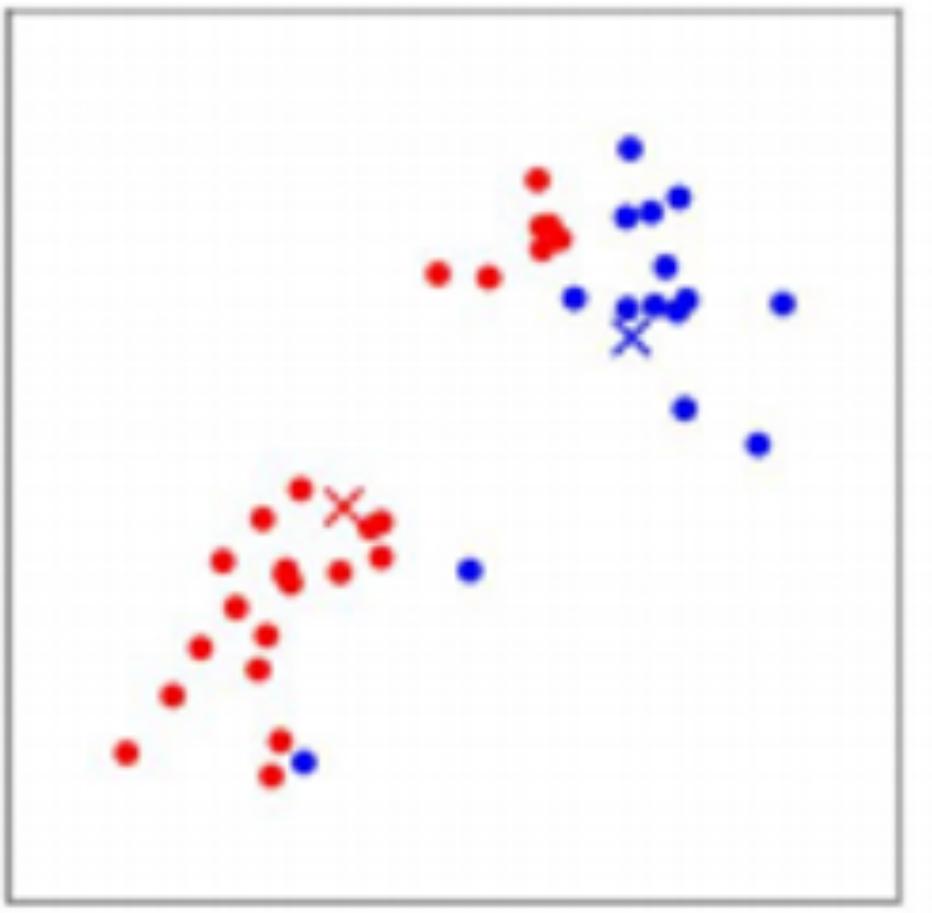
(a)



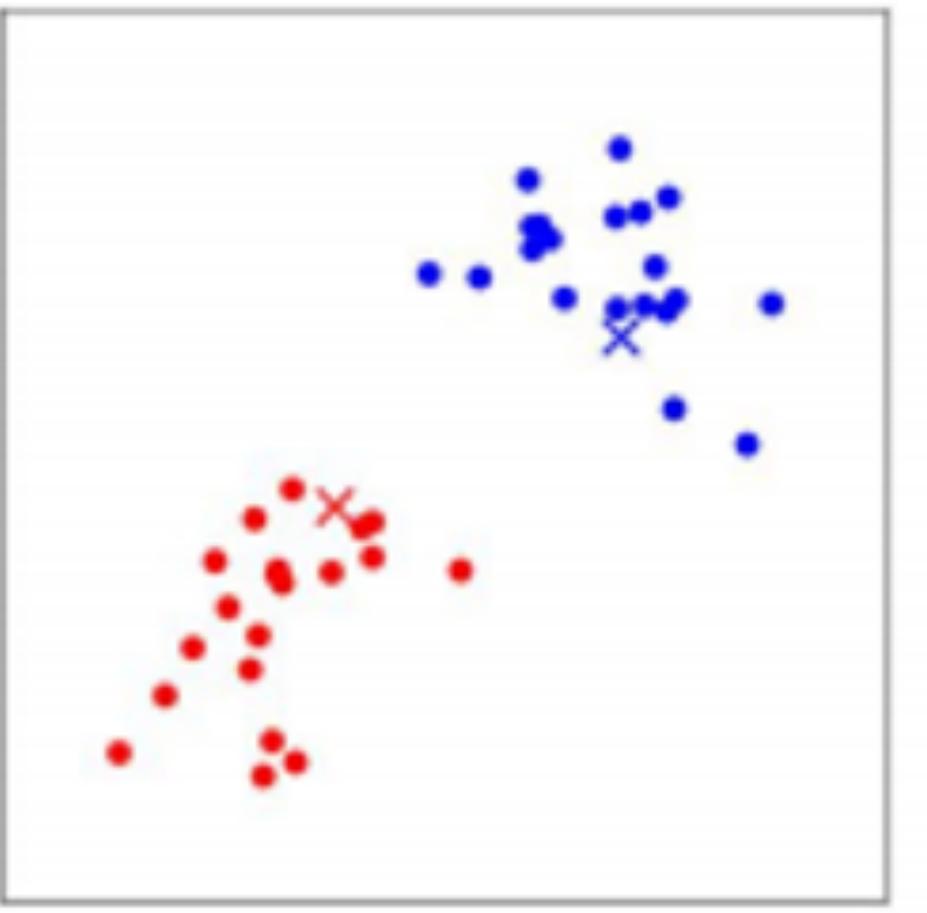
(b)



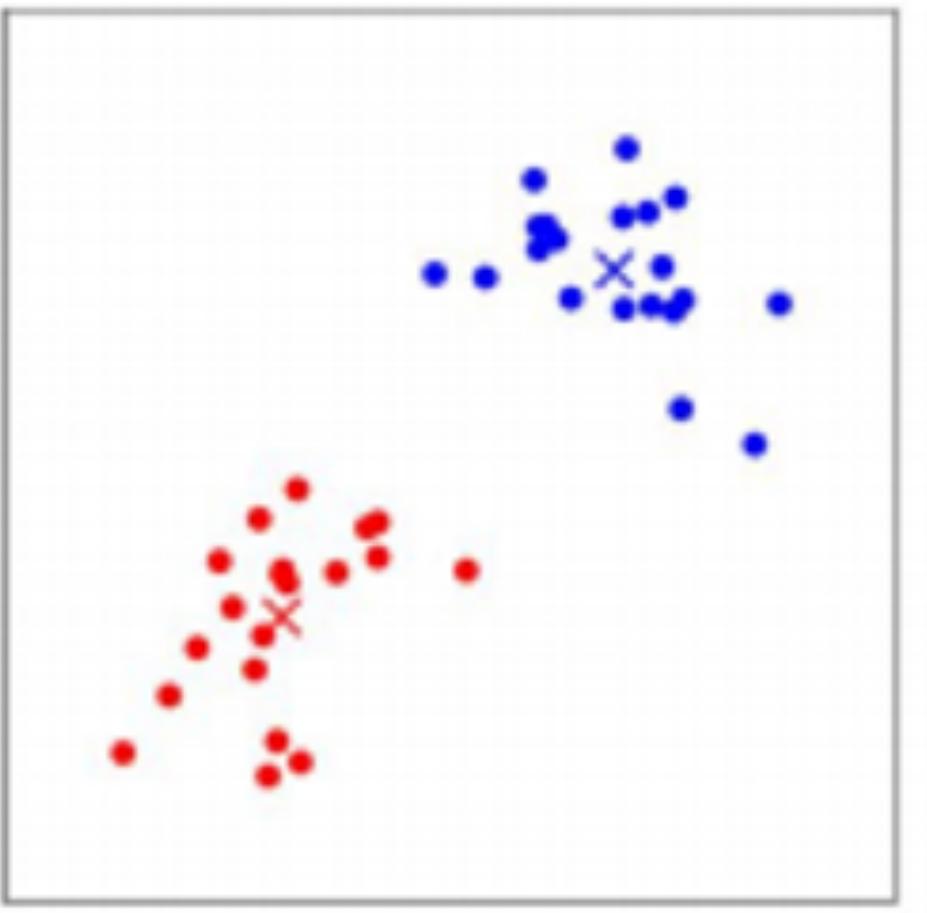
(c)



(d)



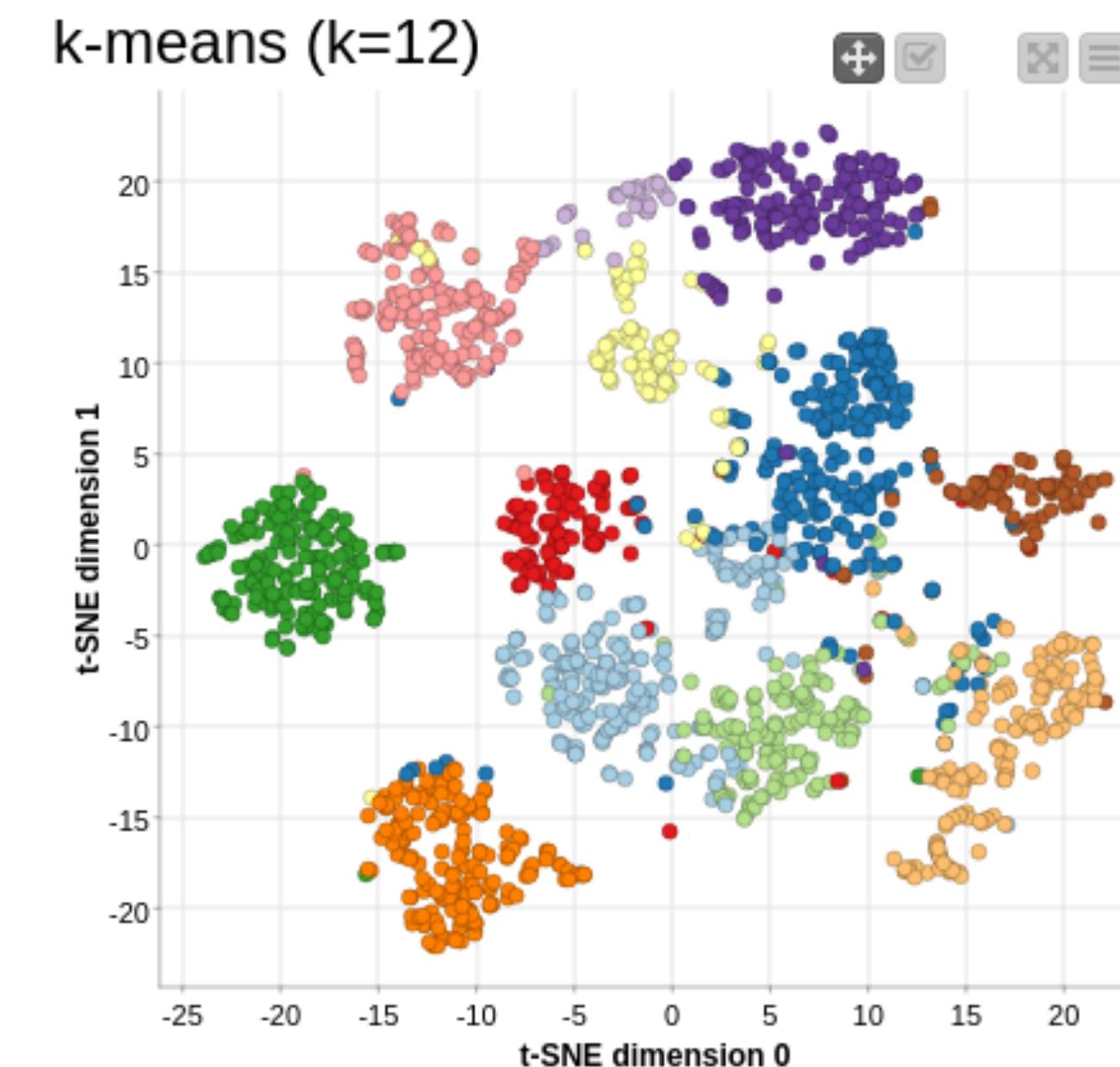
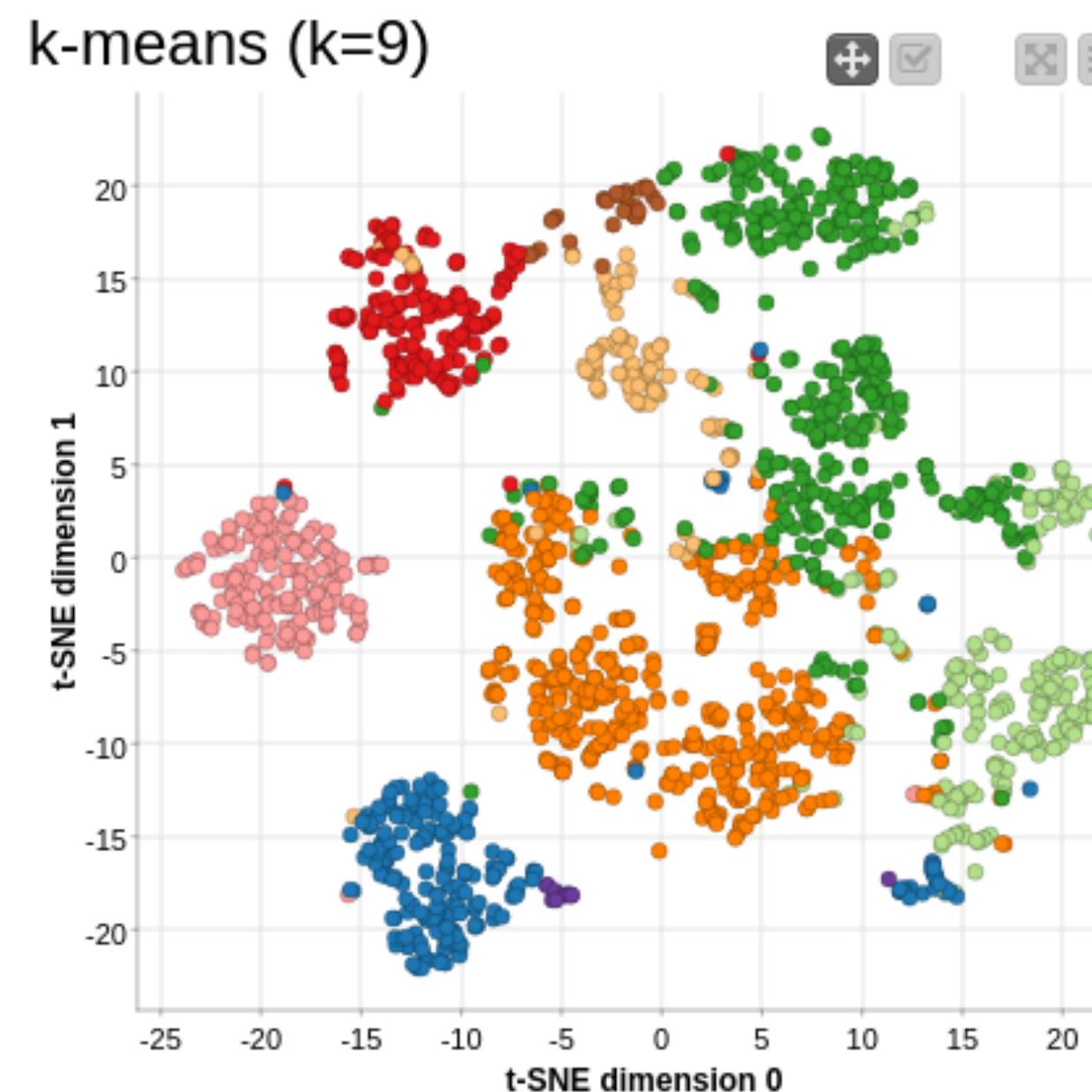
(e)



(f)

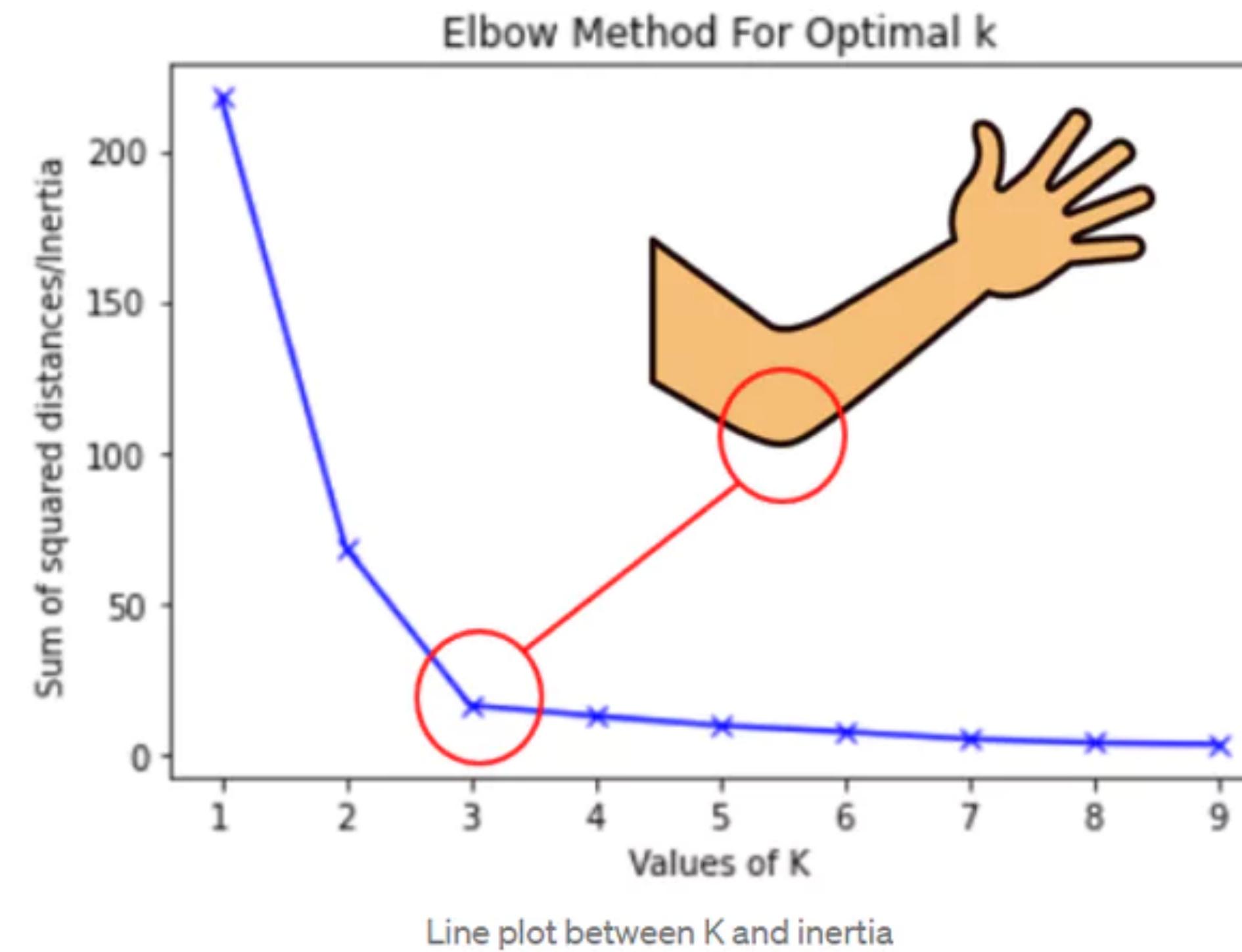
# Hyperparameter

- The number of clusters –  $K$  – is the key hyperparameter
- **Problem.** Previous strategy of “run all, select the best” does not work
  - Reason. Larger  $K$  means finer clustering
    - Thus almost always smaller in loss



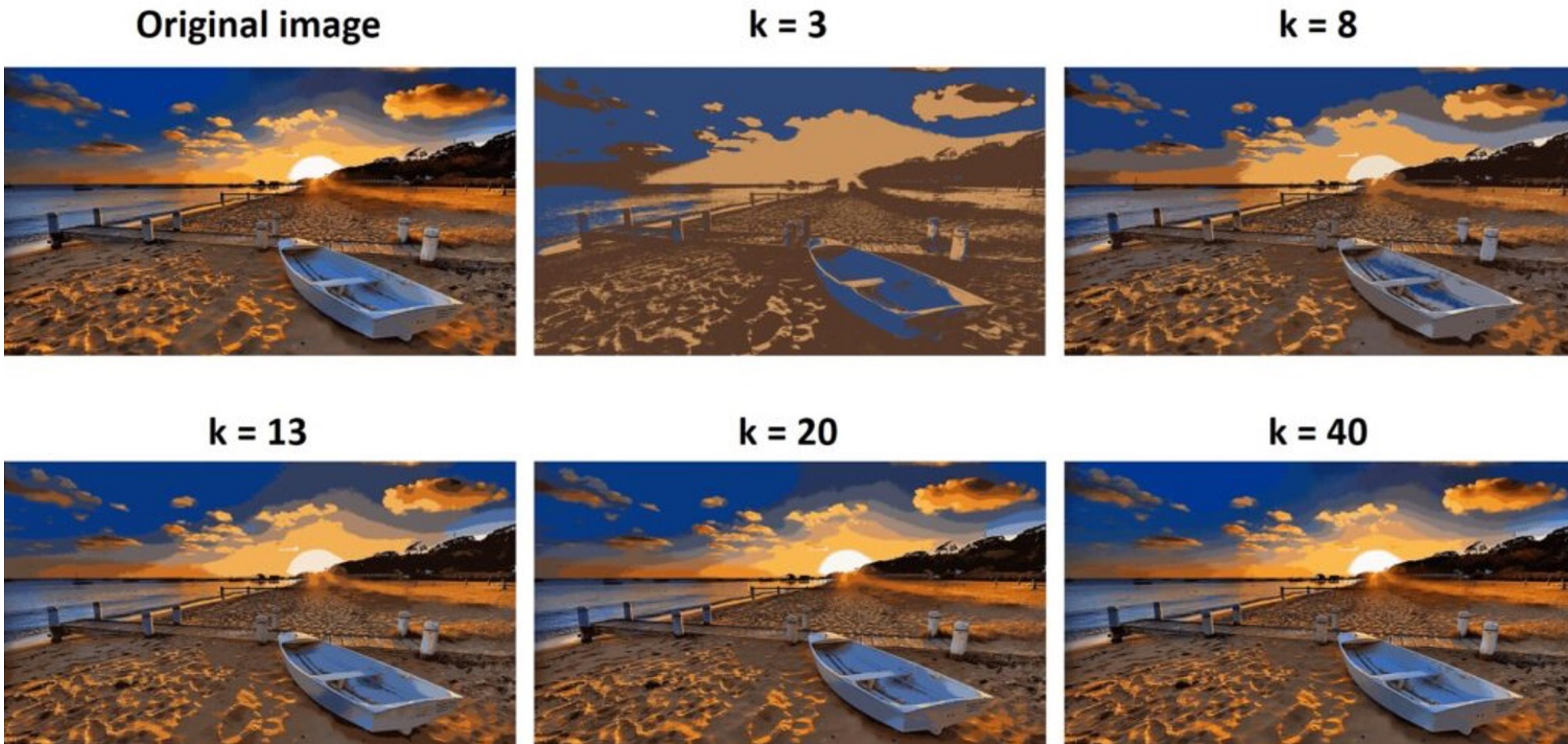
# Hyperparameter

- **Solution.** Many approaches, but the popular heuristic is:  
“Stop adding if the **marginal gain is small**”
- Example. “elbow method”
  - Advanced methods include Akaike information criterion



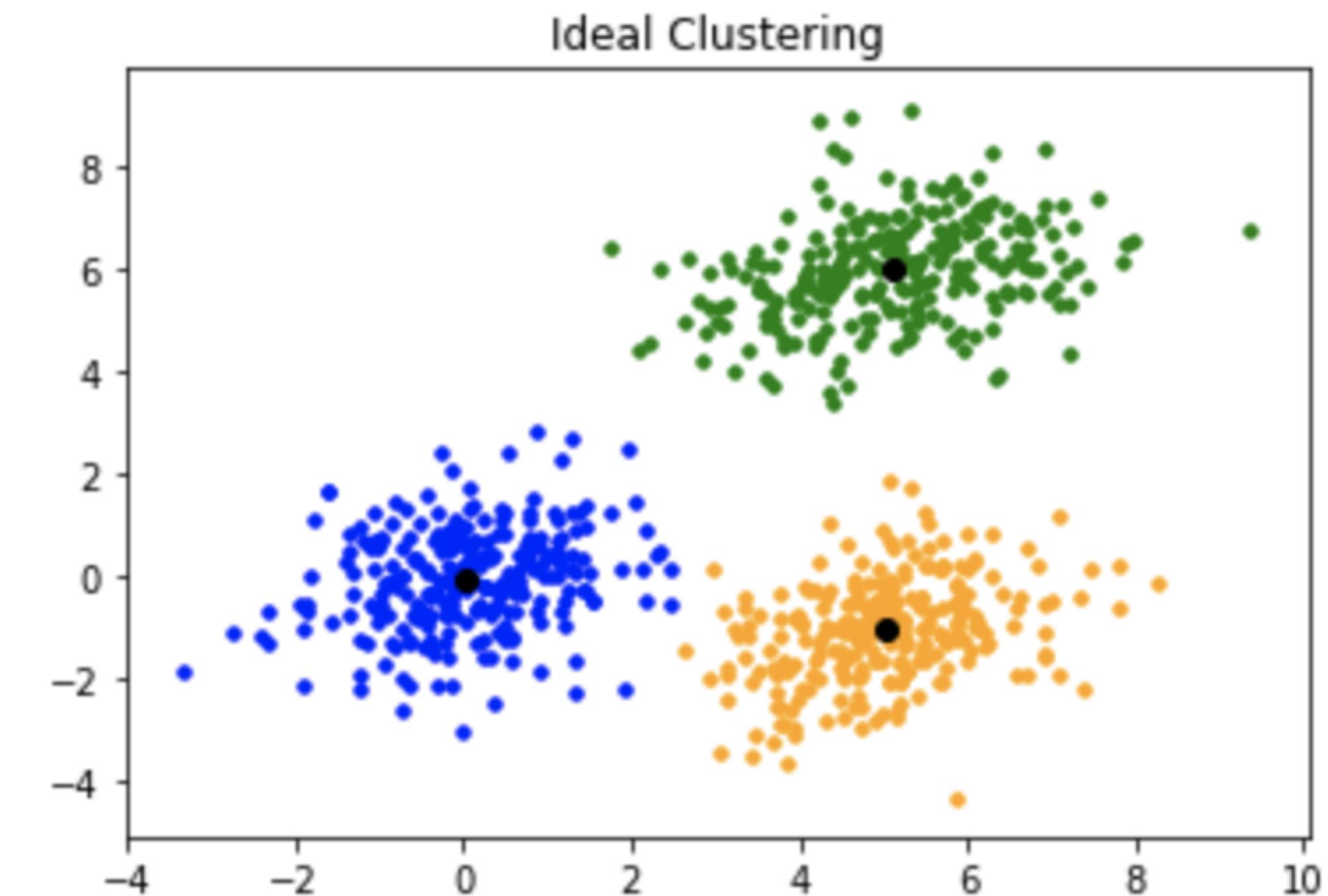
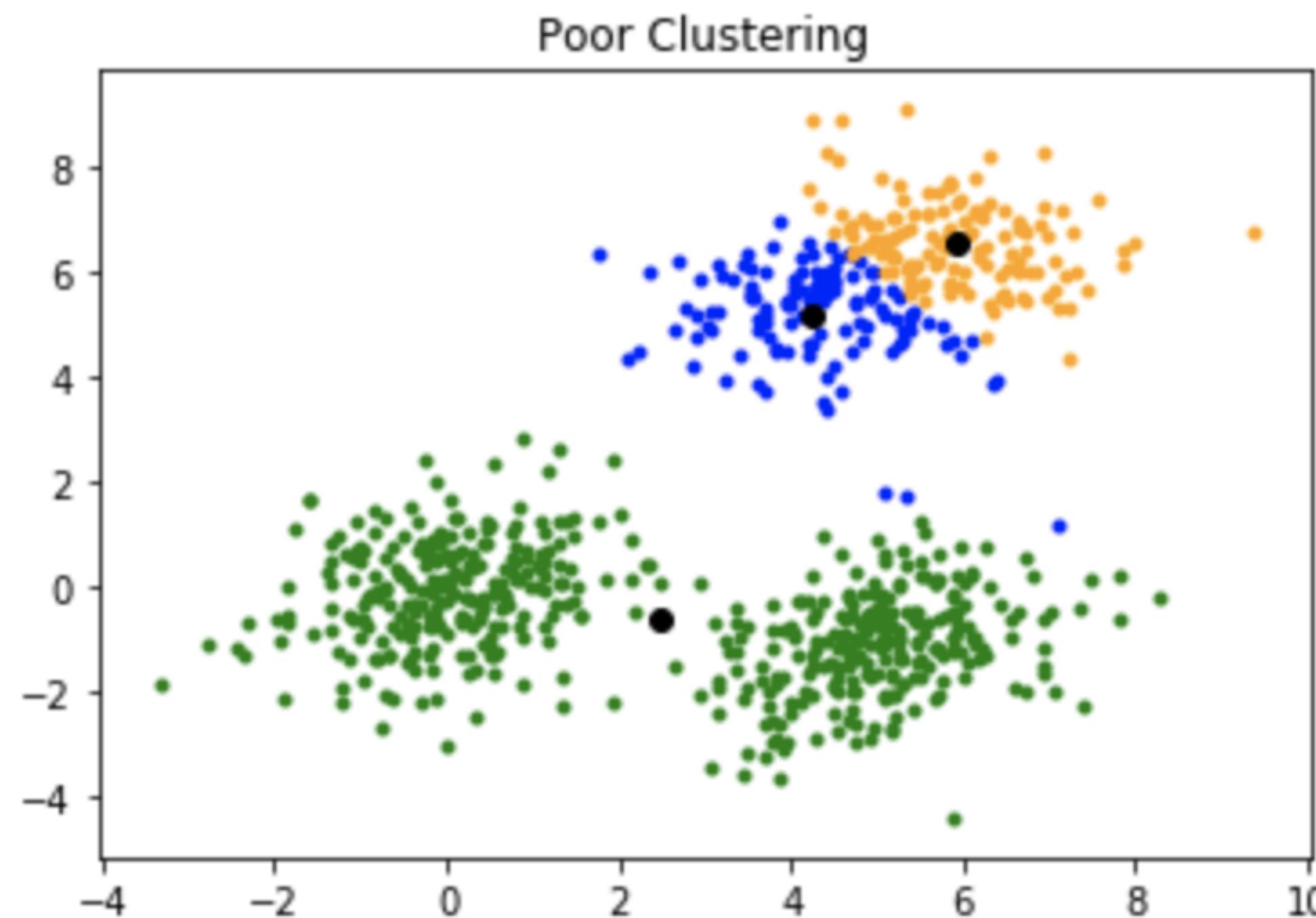
# Application

- One can apply this to compress an image
  - Cluster in the RGB space ( $\subseteq \mathbb{R}^3$ )
  - Reduce the number of colors used; representable with low bit



# Limitations

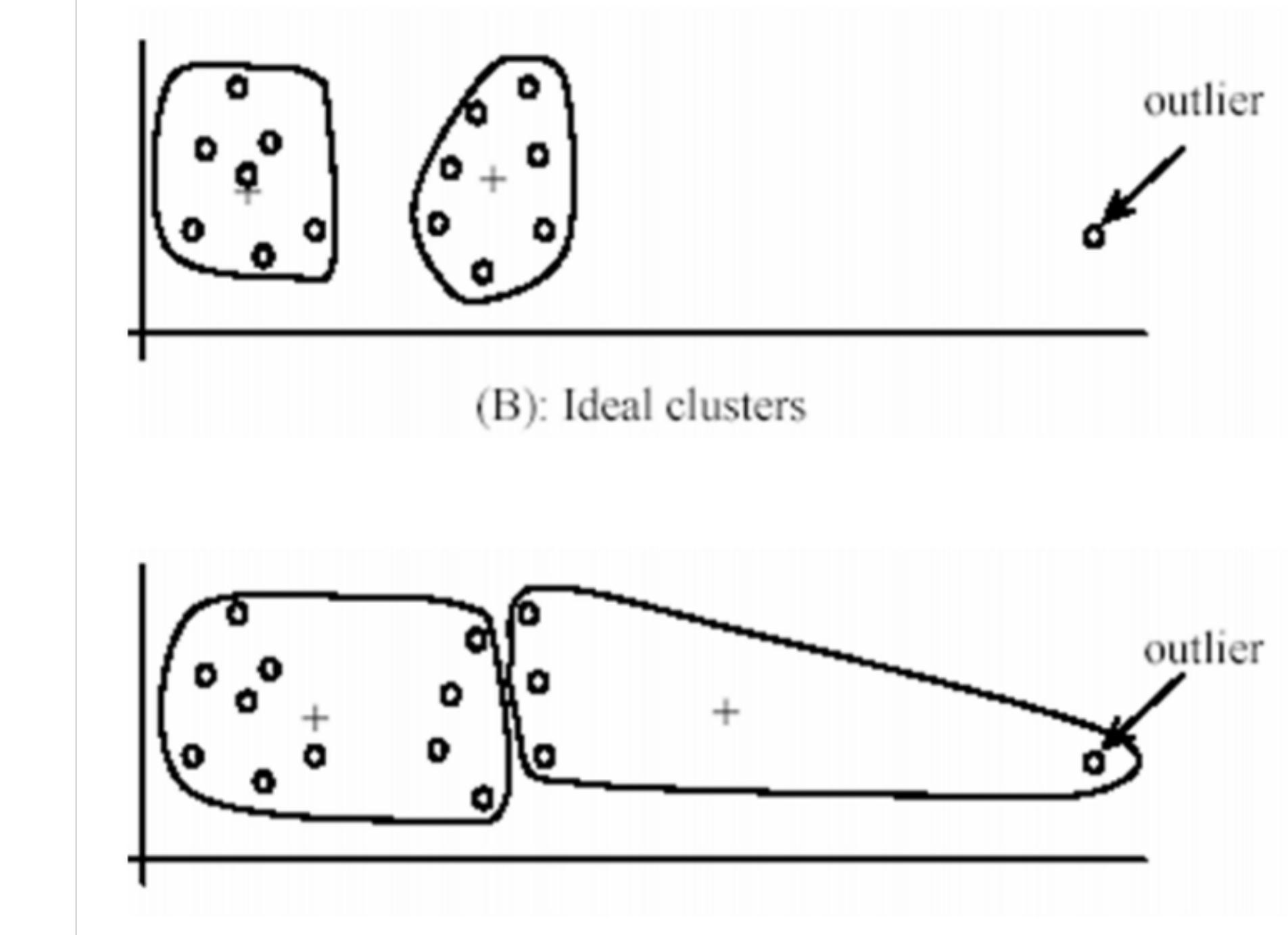
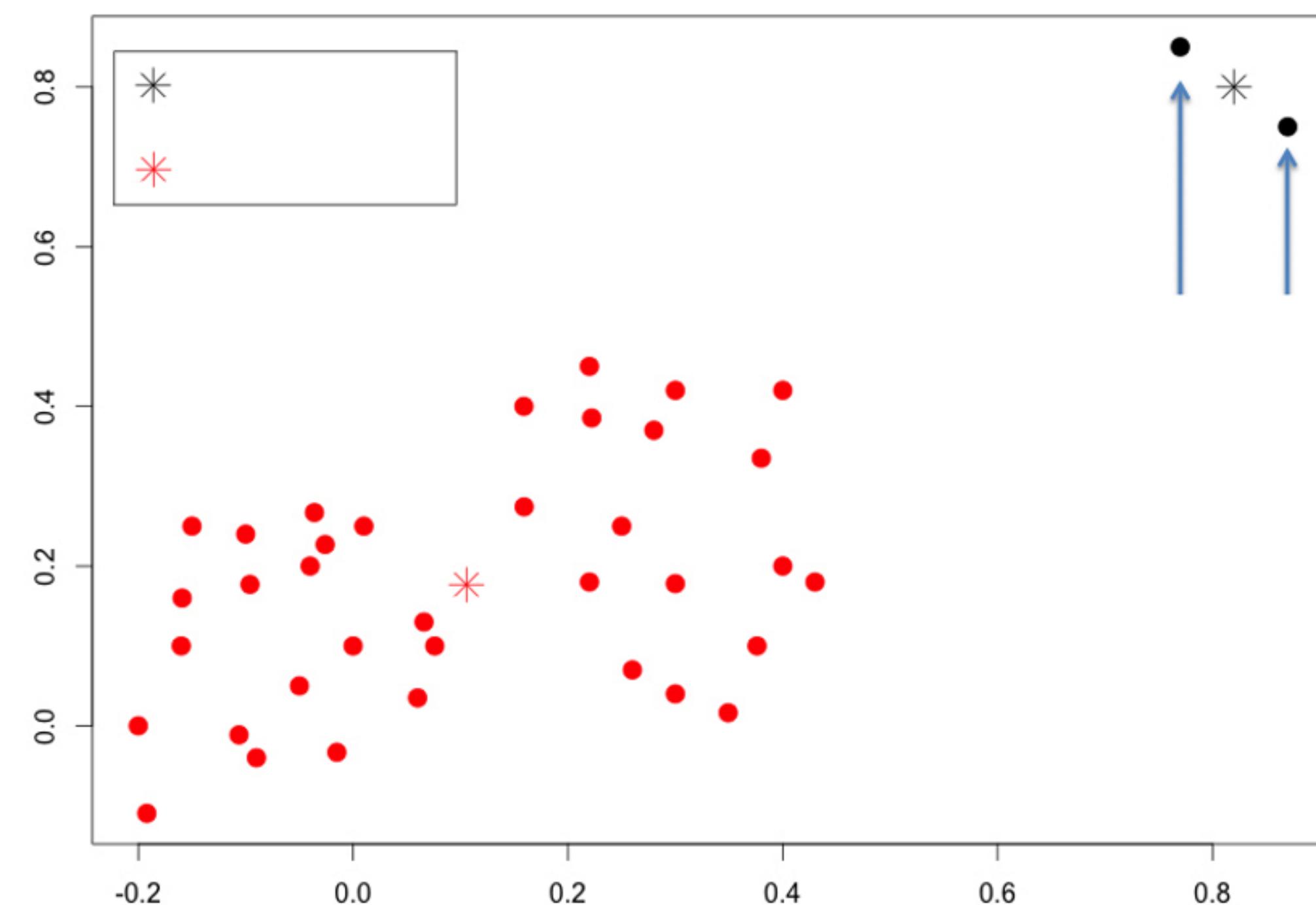
- **Sensitivity to initialization**
  - With random initialization, some chance to fall to a local minima



# Limitations

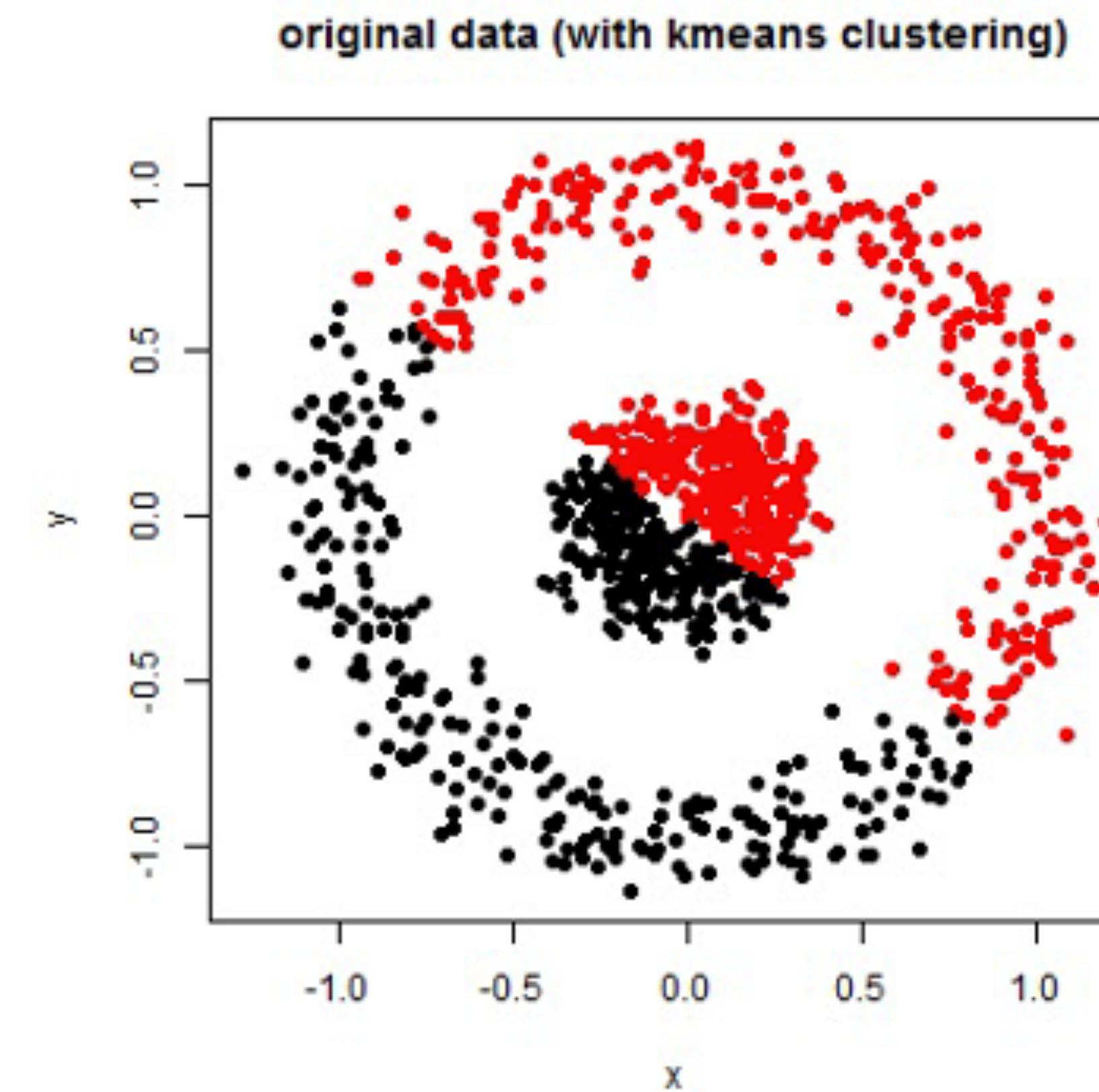
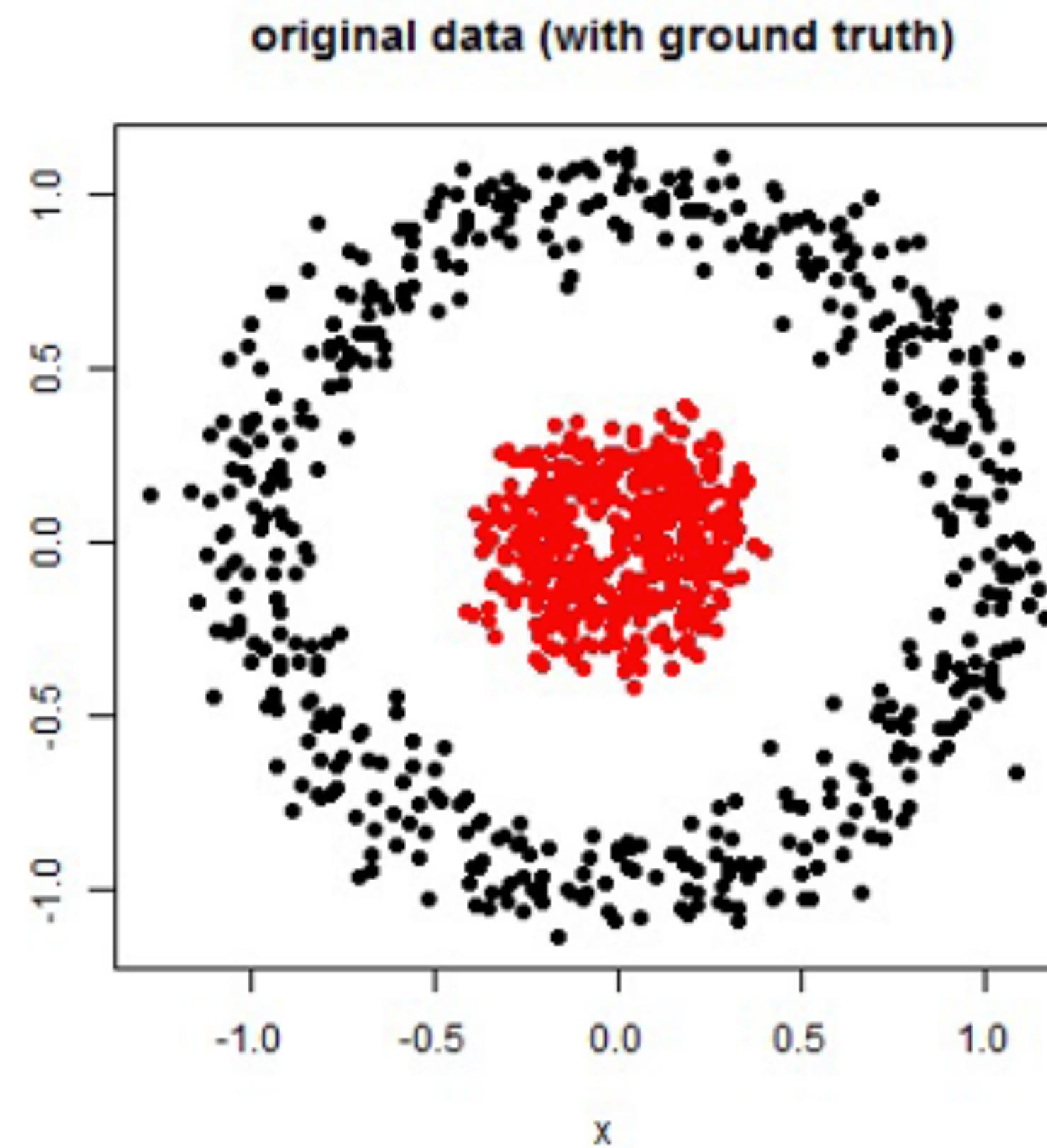
- **Sensitivity to outliers**

- Under outliers, leads to suboptimal clustering
- Assigning additional cluster is undesirable in some cases
  - e.g., image compression –  $2^b$  clusters is efficient



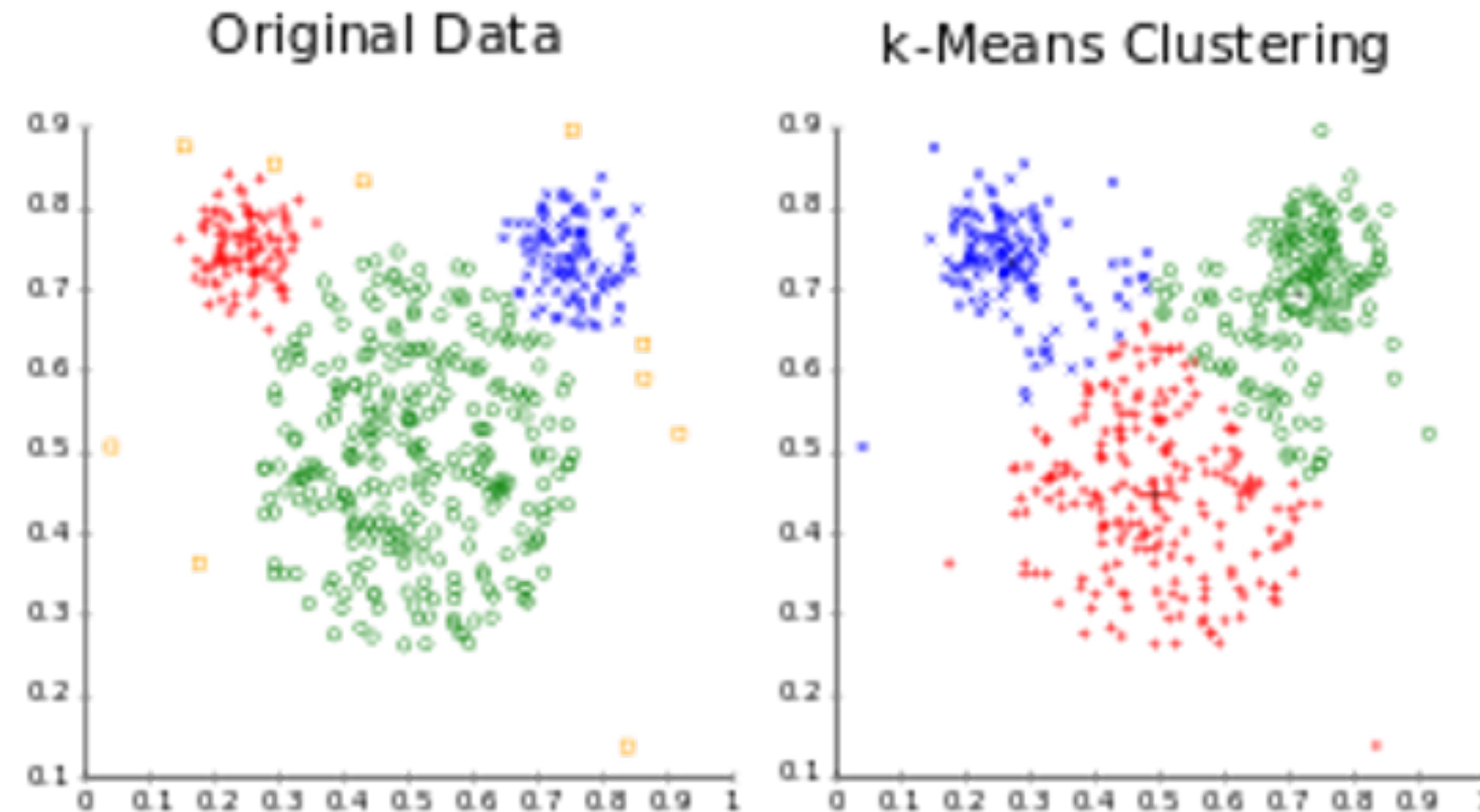
# Limitations

- **Nonlinear data**
  - Difficult to apply to nonlinear data



# Limitations

- **Nonuniform cluster sizes**
  - Even further, the ground-truth cluster sizes may not be linear



# **K-Means++**

# K-Means++

- Resolve the initialization-sensitivity
- Idea. Have the initial centroids well-spread
  - Choose  $\mu_1$  uniformly at random, among all data points
  - For  $i = 2, \dots, \mu$ 
    - For all data, compute the distance-to-nearest centroid

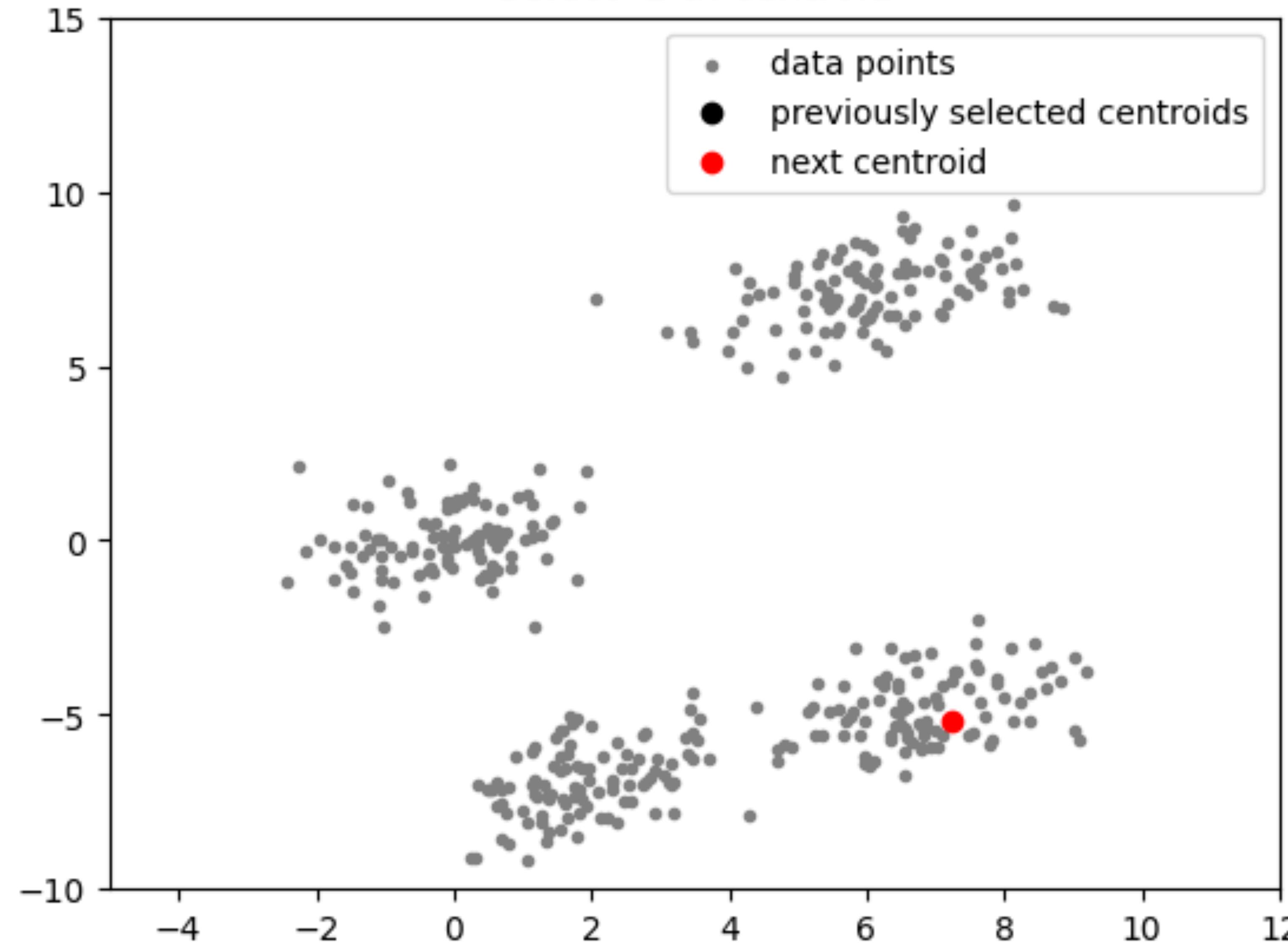
$$D(\mathbf{x}_j) = \min_{j \in \{1, \dots, i-1\}} \|\mathbf{x}_j - \mu_i\|^2$$

- Draw with new centroid with inverse probability

$$p(\mu_i = \mathbf{x}_j) \propto 1/D(\mathbf{x}_j)$$

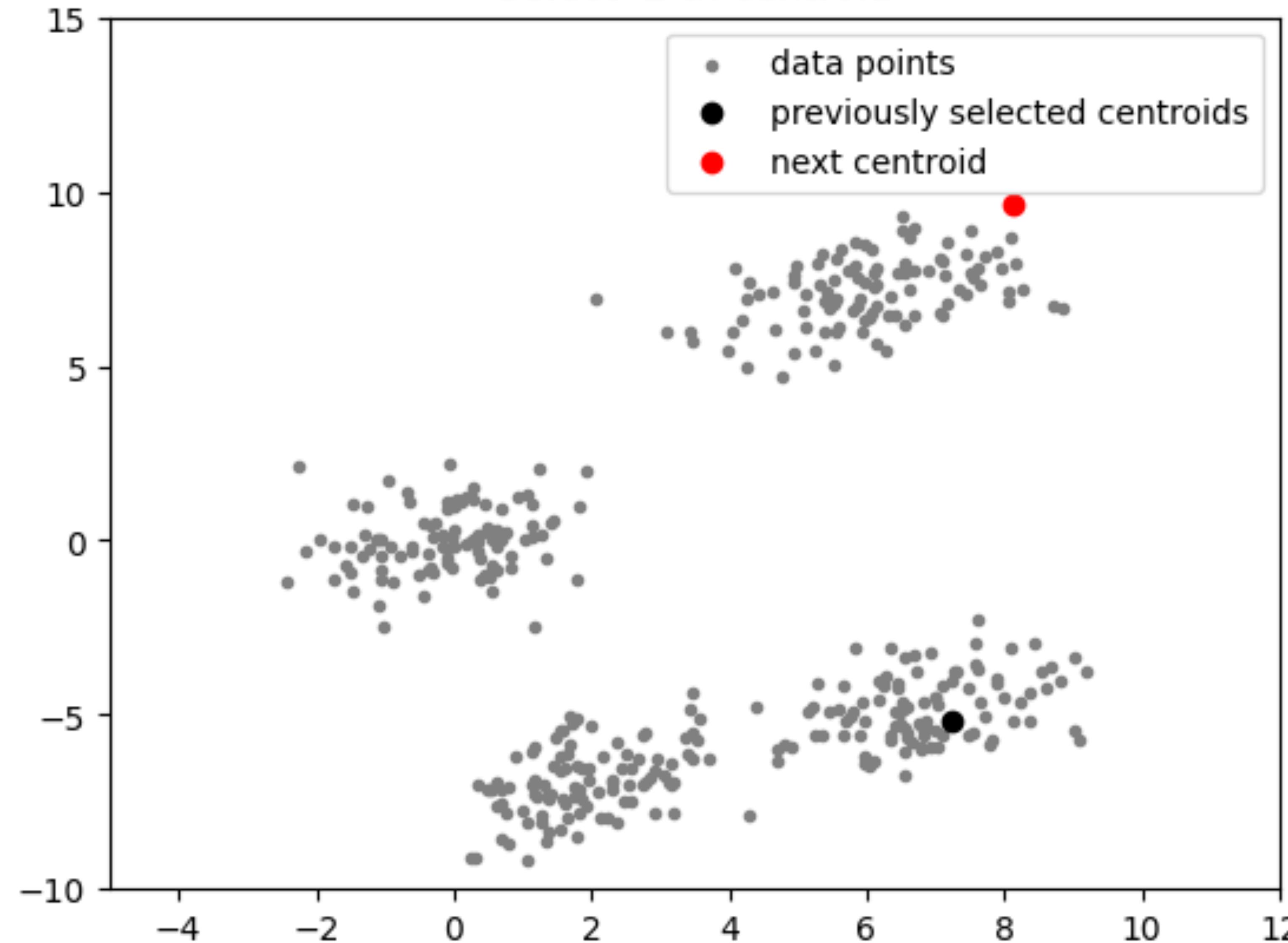
# K-Means++

Select 1 th centroid



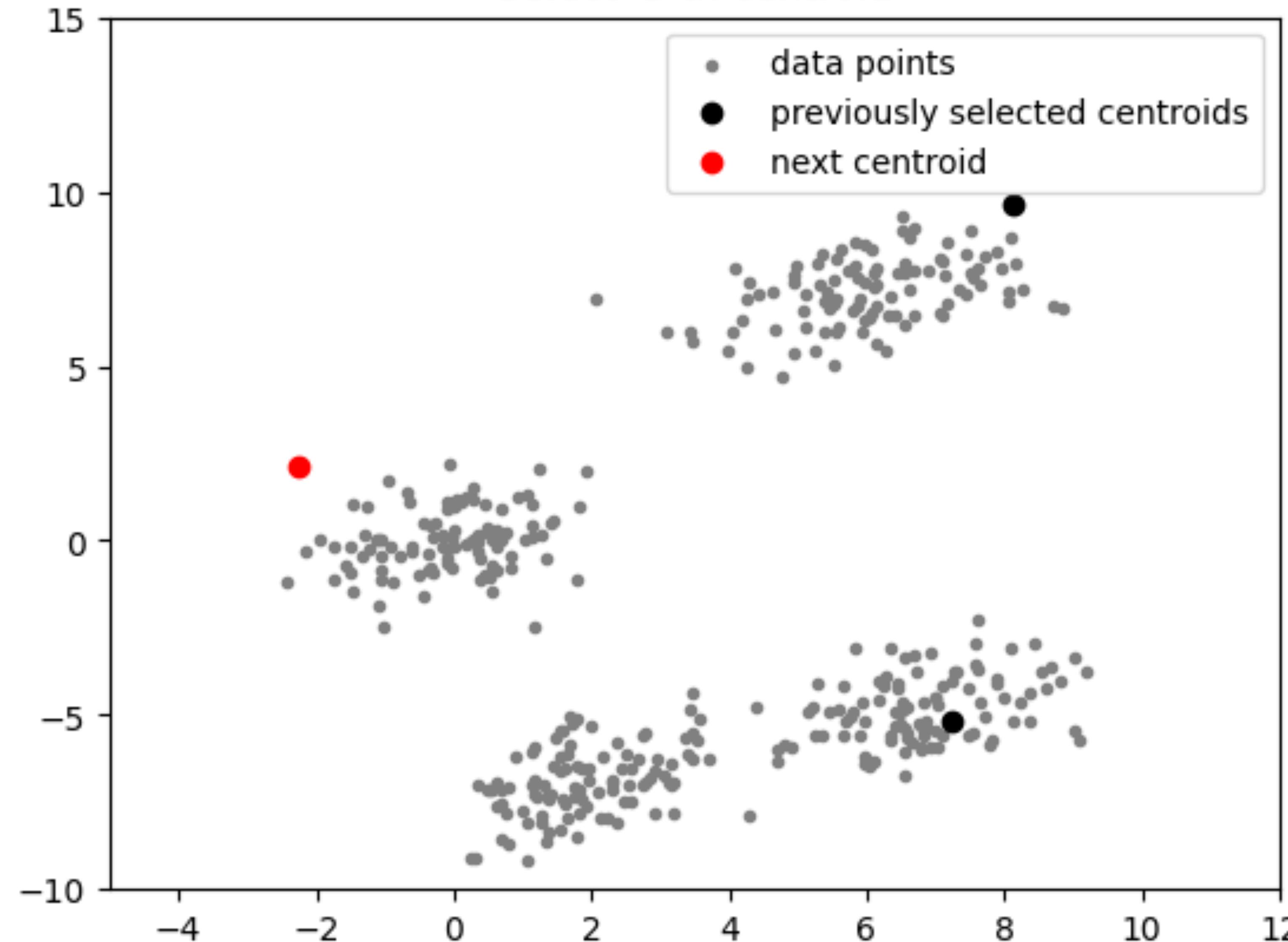
# K-Means++

Select 2 th centroid



# K-Means++

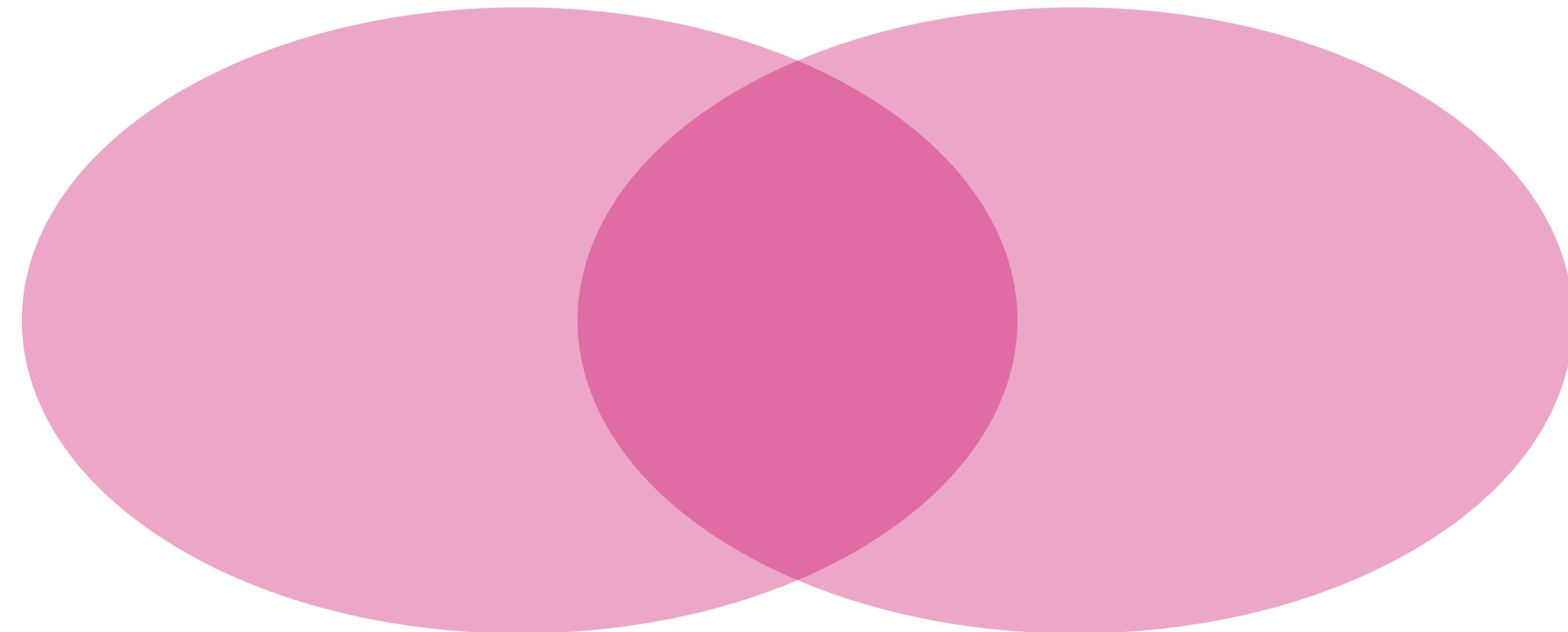
Select 3 th centroid



# **Soft K-Means**

# Soft K-Means

- A version of K-means that allows room for overlapping clusters
  - where should the “middle” go?



# Soft K-Means

- **Idea.** Make soft assignments (“responsibility”) instead of hard one
  - Hard. A point belongs to a specific cluster

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

- Soft. A point may belong to 90% to a cluster, 10% to another

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

# Soft K-Means

- Again, the optimization can be written as

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

- **Problem.** Solving this will always lead to hard assignments!
  - Want to introduce some fuzziness purposefully
  - Solution. Replace with Bayesian objective & Gaussian model
    - Will be discussed in detail in the next lecture

# Soft K-Means

- **Assignment.** Larger responsibility for a closer centroid
  - For some hardness hyperparameter  $\beta > 0$ , we let

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

- This function is known as **softmax**, i.e., a soft version of argmax
  - Letting  $\beta \rightarrow \infty$ , we recover the hard K-means

# Soft K-Means

- **Update.** Take a **weighted average** of the assigned data

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

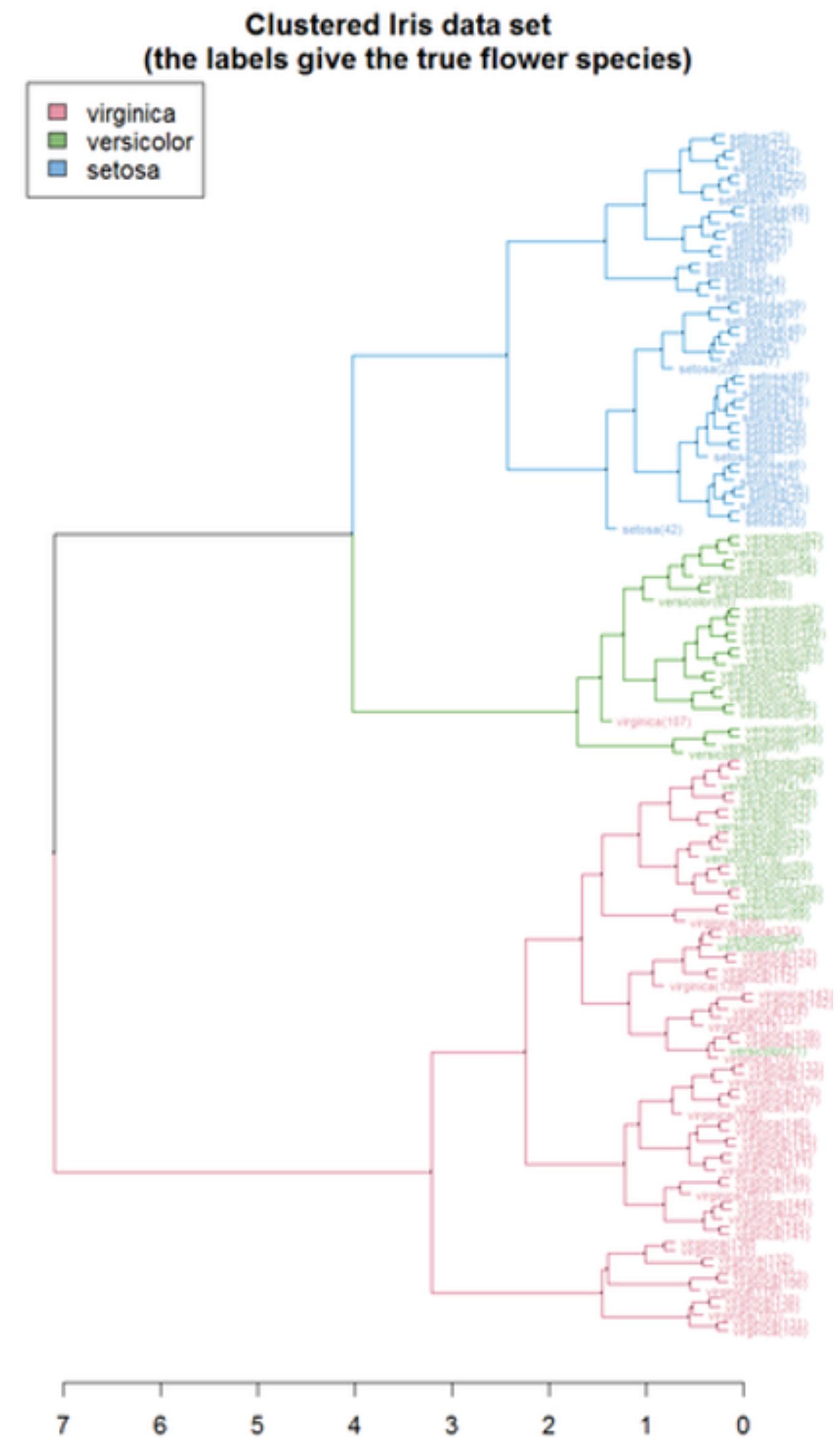
- Can be derived as a solution to the subproblem

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

# **Other variants (informally)**

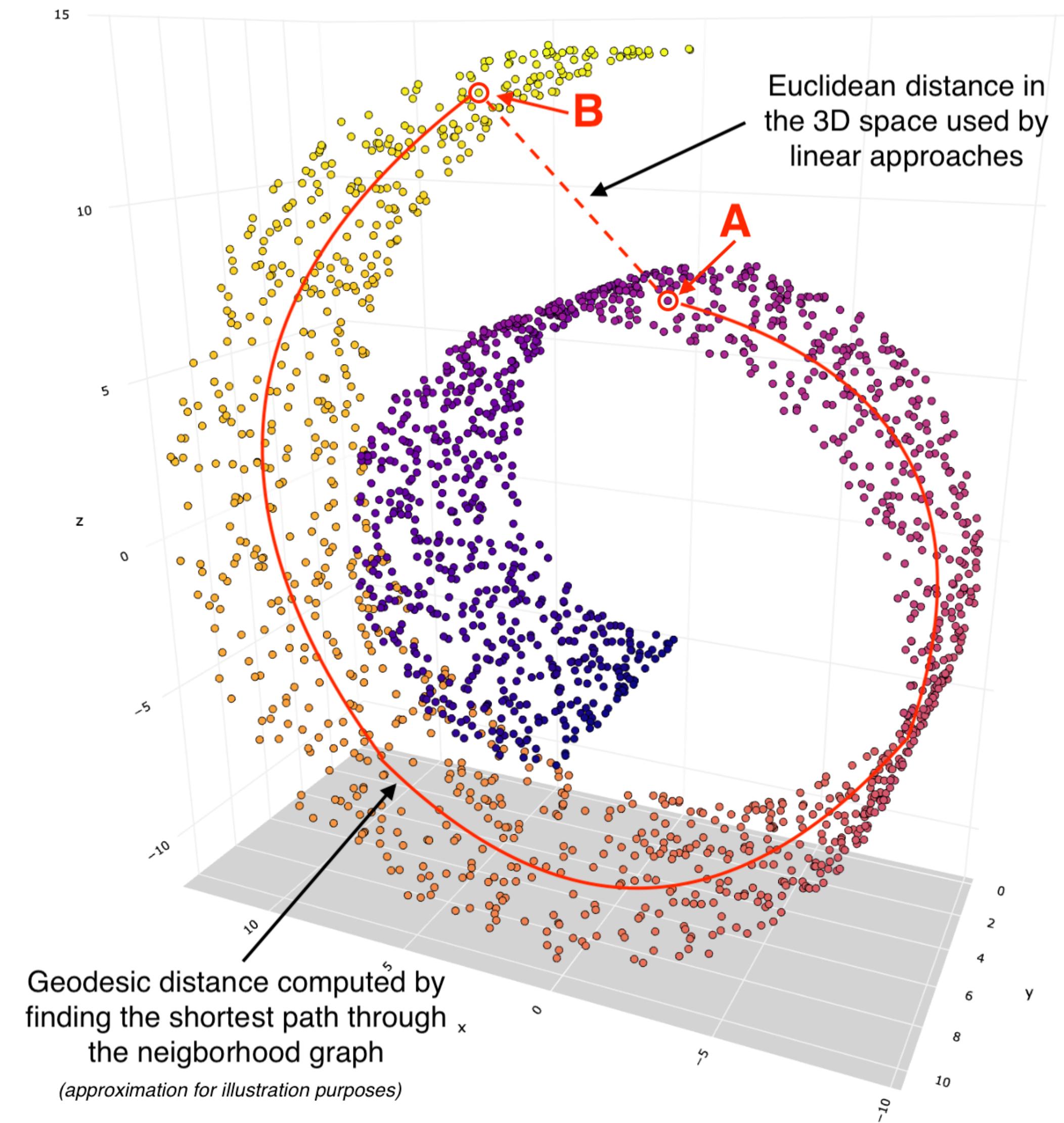
# Hierarchical Clustering

- **Idea.** Clusters inside clusters
  - Can discover hierarchical structures
  - Waive strict decision of  $K$
  - Can handle “different-sized” clusters
  - Can be done top-down or bottom-up



# Spectral Clustering

- **Idea.** Cluster using **graph distance**
  - Construct neighbor graphs
  - Use graph distance for cluster
- Can handle nonlinear data



# Next up

- Gaussian mixture models

**</lecture 6>**