Unsupervised Learning: Clustering

A Deep Dive into Finding Structure in Unlabeled Data

Nipun Batra

IIT Gandhinagar

2025-10-29

Outline

Motivation

Today's Journey

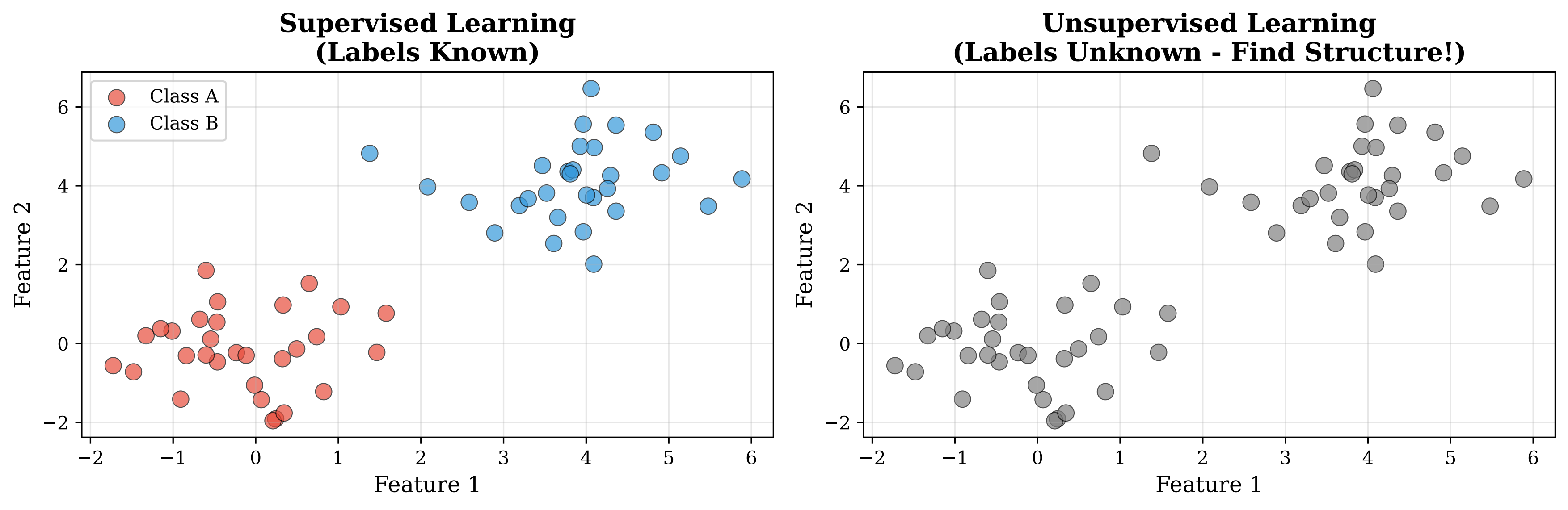
Section	Focus
1. Motivation	Why unsupervised learning + clustering intuition
2. K-Means Deep Dive	Derive objective + visualize + algorithm
3. Practical Issues	Initialization, scaling, complexity, variants
4. Hierarchical	Alternative approach + dendrogram

Today's Journey

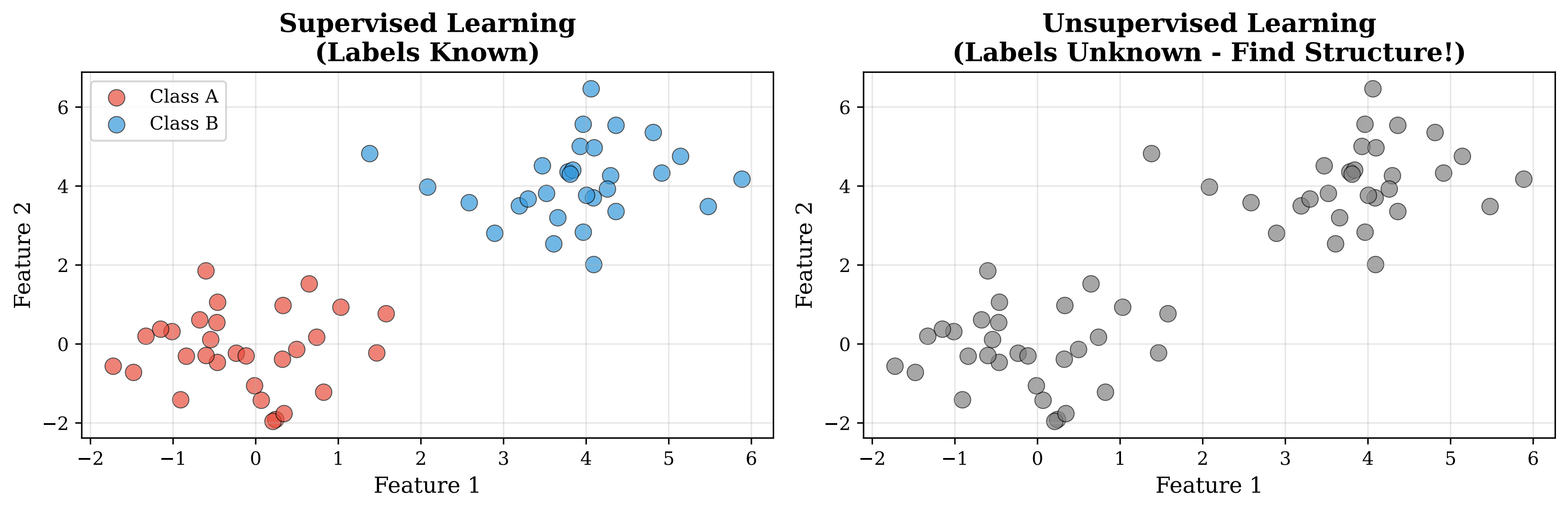
Section	Focus
1. Motivation	Why unsupervised learning + clustering intuition
2. K-Means Deep Dive	Derive objective + visualize + algorithm
3. Practical Issues	Initialization, scaling, complexity, variants
4. Hierarchical	Alternative approach + dendrogram

Key Philosophy: Intuition $\longrightarrow$ Visualization $\longrightarrow$ Mathematics $\longrightarrow$ Code

Supervised vs Unsupervised Learning



Supervised vs Unsupervised Learning



Key Difference: We discover patterns, not predict labels!

Why Unsupervised Learning?

Three Main Reasons:

1. Labels are expensive or impossible to obtain

- Medical images: Need expert radiologists ($$$)
- Customer behavior: No “true” groupings exist
- Exploratory analysis: Don’t know what to look for yet

Why Unsupervised Learning?

Three Main Reasons:

1. Labels are expensive or impossible to obtain

- Medical images: Need expert radiologists ($$$)
- Customer behavior: No “true” groupings exist
- Exploratory analysis: Don’t know what to look for yet

2. Discover hidden patterns

- Find new disease subtypes from patient data
- Identify market segments you didn’t know existed
- Detect anomalies (fraud, network intrusion)

Why Unsupervised Learning?

Three Main Reasons:

1. Labels are expensive or impossible to obtain

- Medical images: Need expert radiologists ($$$)
- Customer behavior: No "true" groupings exist
- Exploratory analysis: Don't know what to look for yet

2. Discover hidden patterns

- Find new disease subtypes from patient data
- Identify market segments you didn't know existed
- Detect anomalies (fraud, network intrusion)

3. Data preprocessing

- Dimensionality reduction before supervised learning
- Feature extraction, data compression

Real-World Applications

Business & Marketing

- Customer segmentation
- Product recommendation
- Market basket analysis

Healthcare

- Disease subtype discovery
- Patient stratification
- Gene expression analysis

Computer Vision

- Image segmentation
- Object discovery
- Facial recognition preprocessing

Text & NLP

- Document clustering
- Topic modeling
- News article grouping

Clustering: The Core Task

AIM: Find groups/subgroups in a dataset

REQUIREMENTS: A notion of similarity/dissimilarity

Clustering: The Core Task

AIM: Find groups/subgroups in a dataset

REQUIREMENTS: A notion of similarity/dissimilarity

Central Question: What makes two data points "similar"?

- **Euclidean distance**: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = \sqrt{\sum_d (x_{id} - x_{jd})^2}$
- **Cosine similarity**: $\cos(\theta) = \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \cdot \|\boldsymbol{x}_j\|}$
- **Manhattan distance**: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1 = \sum_d |x_{id} - x_{jd}|$

Clustering: The Core Task

AIM: Find groups/subgroups in a dataset

REQUIREMENTS: A notion of similarity/dissimilarity

Central Question: What makes two data points "similar"?

- **Euclidean distance**: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = \sqrt{\sum_d \left(x_{id} - x_{jd}\right)^2}$
- **Cosine similarity**: $\cos(\theta) = \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \cdot \|\boldsymbol{x}_j\|}$
- **Manhattan distance**: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1 = \sum_d |x_{id} - x_{jd}|$

Today we'll focus on **Euclidean distance** (most common)

K-Means Deep Dive

K-Means: The Workhorse Algorithm

K-Means: Problem Setup

Given:

- N points: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$
- Number of clusters: K (specified in advance)

K-Means: Problem Setup

Given:

- N points: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$
- Number of clusters: K (specified in advance)

Find: Partition into K clusters $C_1, C_2, \ldots, C_K$ such that:

1. Every point belongs to exactly one cluster:

$$C_1 \cup C_2 \cup \ldots \cup C_K = \{1, 2, \ldots, n\}$$

2. Clusters don't overlap (hard assignment):

$$C_i \cap C_j = \emptyset \text{ for } i \neq j$$

K-Means: Problem Setup

Given:

- N points: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$
- Number of clusters: K (specified in advance)

Find: Partition into K clusters $C_1, C_2, \ldots, C_K$ such that:

1. Every point belongs to exactly one cluster:

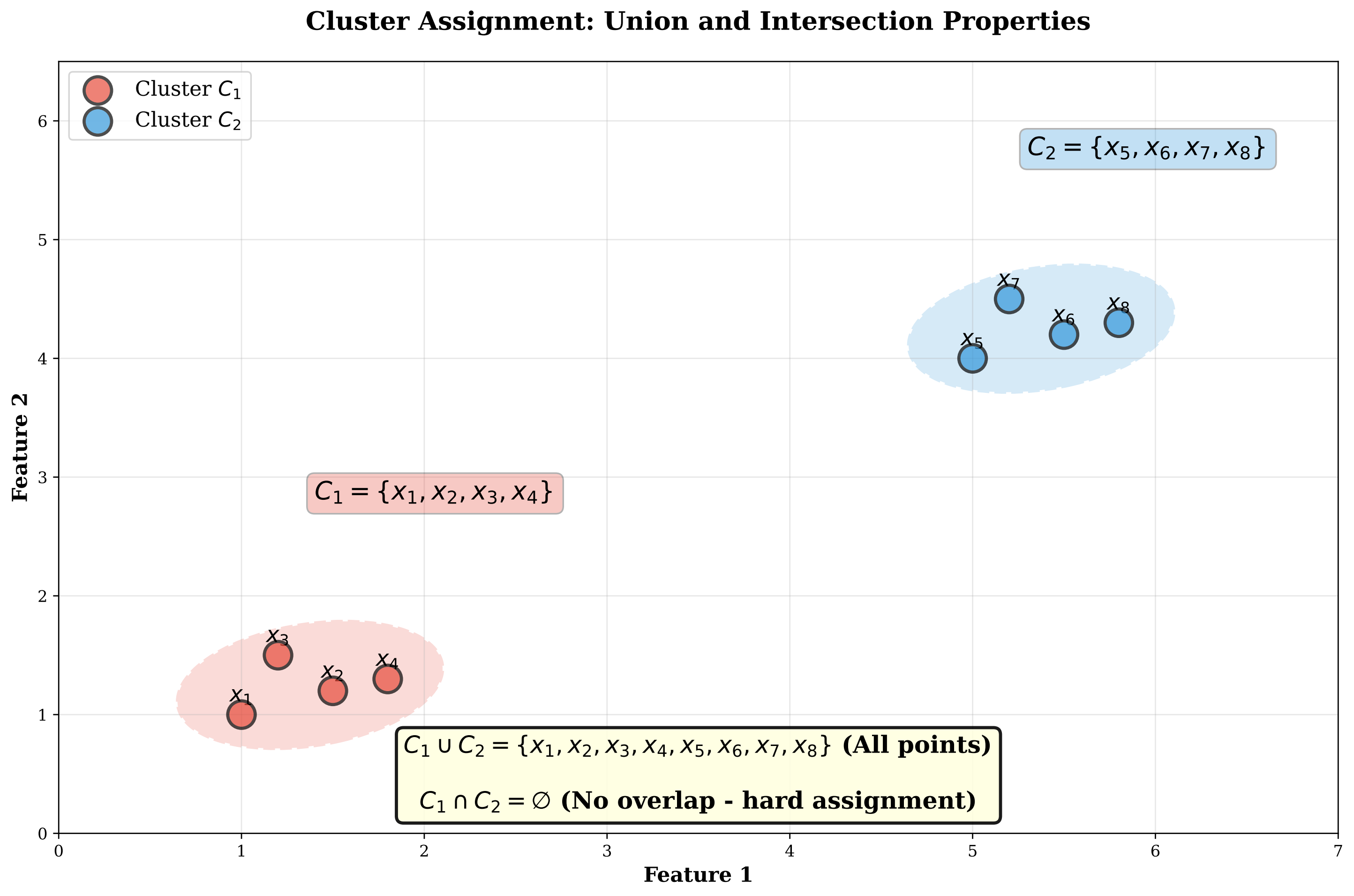
$$C_1 \cup C_2 \cup \ldots \cup C_K = \{1, 2, \ldots, n\}$$

2. Clusters don't overlap (hard assignment):

$$C_i \cap C_j = \emptyset \text{ for } i \neq j$$

Each point gets assigned to **exactly one** cluster (hard clustering)

Cluster Assignment: Visualized



K-Means: Objective Function

Goal: Minimize the total objective Φ

$$\Phi = \min_{C_1, \ldots, C_K} \sum_{i=1}^{K} \mathrm{WCSS}(C_i)$$

where:

- Φ = total objective (sum across ALL clusters)
- $\mathrm{WCSS}(C_i)$ = Within-Cluster Sum of Squares for cluster C_i

K-Means: Objective Function

Goal: Minimize the total objective Φ

$$\Phi = \min_{C_1,\ldots,C_K} \sum_{i=1}^{K} \mathrm{WCSS}(C_i)$$

where:

- Φ = total objective (sum across ALL clusters)
- $\mathrm{WCSS}(C_i)$ = Within-Cluster Sum of Squares for cluster C_i

Intuition: Make each cluster tight $\longrightarrow$ minimize Φ

WCSS: Definition

For cluster C_i, define:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$$

where:

- $|C_i|$ = number of points in cluster C_i
- $\|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$ = squared Euclidean distance

WCSS: Definition

For cluster C_i, define:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$$

where:

- $|C_i|$ = number of points in cluster C_i
- $\|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$ = squared Euclidean distance

Interpretation: Average of all pairwise squared distances within the cluster

WCSS: Definition

For cluster C_i, define:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$$

where:

- $|C_i|$ = number of points in cluster C_i
- $\|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$ = squared Euclidean distance

Interpretation: Average of all pairwise squared distances within the cluster

Problem: This has $O(|C_i|^2)$ terms! Can we simplify?

WCSS Simplification: The Centroid Form (1/3)

Theorem: WCSS can be rewritten using centroids:

$$\mathrm{WCSS}(C_i) = 2 \sum_{a \in C_i} \|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{a \in C_i} \boldsymbol{x}_a$ is the **centroid** (mean) of cluster C_i

WCSS Simplification: The Centroid Form (1/3)

Theorem: WCSS can be rewritten using centroids:

$$\mathrm{WCSS}(C_i) = 2 \sum_{a \in C_i} \|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{a \in C_i} \boldsymbol{x}_a$ is the **centroid** (mean) of cluster C_i

Advantage: Only $O(|C_i|)$ terms instead of $O(|C_i|^2)$!

WCSS Simplification: The Centroid Form (1/3)

Theorem: WCSS can be rewritten using centroids:

$$\mathrm{WCSS}(C_i) = 2 \sum_{a \in C_i} \|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{a \in C_i} \boldsymbol{x}_a$ is the **centroid** (mean) of cluster C_i

Advantage: Only $O(|C_i|)$ terms instead of $O(|C_i|^2)$!

Next: Let's prove this equivalence...

WCSS Equivalence Proof (2/3): Setup

Start with pairwise form:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$$

WCSS Equivalence Proof (2/3): Setup

Start with pairwise form:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$$

Expand the squared norm:

$$\|\boldsymbol{x}_a - \boldsymbol{x}_b\|^2 = (\boldsymbol{x}_a - \boldsymbol{x}_b)^\top (\boldsymbol{x}_a - \boldsymbol{x}_b)$$

WCSS Equivalence Proof (2/3): Setup

Start with pairwise form:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a - \boldsymbol{x}_b\|_2^2$$

Expand the squared norm:

$$\|\boldsymbol{x}_a - \boldsymbol{x}_b\|^2 = (\boldsymbol{x}_a - \boldsymbol{x}_b)^\top (\boldsymbol{x}_a - \boldsymbol{x}_b)$$

$$= \boldsymbol{x}_a^\top \boldsymbol{x}_a - 2\boldsymbol{x}_a^\top \boldsymbol{x}_b + \boldsymbol{x}_b^\top \boldsymbol{x}_b$$

$$= \|\boldsymbol{x}_a\|^2 - 2\boldsymbol{x}_a^\top \boldsymbol{x}_b + \|\boldsymbol{x}_b\|^2$$

WCSS Equivalence Proof (3/3): Expand and Simplify

Substitute expanded norm into WCSS:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \left(\|\boldsymbol{x}_a\|^2 - 2\boldsymbol{x}_a^\top \boldsymbol{x}_b + \|\boldsymbol{x}_b\|^2 \right)$$

WCSS Equivalence Proof (3/3): Expand and Simplify

Substitute expanded norm into WCSS:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \sum_{a \in C_i} \sum_{b \in C_i} \left(\|\boldsymbol{x}_a\|^2 - 2\boldsymbol{x}_a^\top \boldsymbol{x}_b + \|\boldsymbol{x}_b\|^2 \right)$$

Separate the three sums:

$$= \frac{1}{|C_i|} \left[\sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a\|^2 - 2 \sum_{a \in C_i} \sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b + \sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_b\|^2 \right]$$

WCSS Equivalence Proof (4/5): Simplify First and Third Terms

For the first term: $\sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a\|^2$

- Inner sum over b: $\|\boldsymbol{x}_a\|^2$ doesn't depend on b, so we get $|C_i| \cdot \|\boldsymbol{x}_a\|^2$
- Result: $\sum_{a \in C_i} |C_i| \, \|\boldsymbol{x}_a\|^2 = |C_i| \sum_{a \in C_i} \|\boldsymbol{x}_a\|^2$

WCSS Equivalence Proof (4/5): Simplify First and Third Terms

For the first term: $\sum_{a\in C_i}\sum_{b\in C_i}\|\boldsymbol{x}_a\|^2$

- Inner sum over b: $\|\boldsymbol{x}_a\|^2$ doesn't depend on b, so we get $|C_i|\cdot\|\boldsymbol{x}_a\|^2$
- Result: $\sum_{a\in C_i}|C_i|\ \|\boldsymbol{x}_a\|^2 = |C_i|\sum_{a\in C_i}\|\boldsymbol{x}_a\|^2$

Similarly, the third term: $\sum_{a\in C_i}\sum_{b\in C_i}\|\boldsymbol{x}_b\|^2 = |C_i|\sum_{b\in C_i}\|\boldsymbol{x}_b\|^2$

WCSS Equivalence Proof (4/5): Simplify First and Third Terms

For the first term: $\sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_a\|^2$

- Inner sum over b: $\|\boldsymbol{x}_a\|^2$ doesn't depend on b, so we get $|C_i| \cdot \|\boldsymbol{x}_a\|^2$
- Result: $\sum_{a \in C_i} |C_i| \ \|\boldsymbol{x}_a\|^2 = |C_i| \sum_{a \in C_i} \|\boldsymbol{x}_a\|^2$

Similarly, the third term: $\sum_{a \in C_i} \sum_{b \in C_i} \|\boldsymbol{x}_b\|^2 = |C_i| \sum_{b \in C_i} \|\boldsymbol{x}_b\|^2$

So we have:

$$\mathrm{WCSS}(C_i) = \frac{1}{|C_i|} \left[|C_i| \sum_{a \in C_i} \|\boldsymbol{x}_a\|^2 - 2 \sum_{a \in C_i} \sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b + |C_i| \sum_{b \in C_i} \|\boldsymbol{x}_b\|^2 \right]$$

WCSS Equivalence Proof (5/8): Simplify Double Sum

Current: $\text{WCSS}(C_i) = 2\sum_{a \in C_i} \|\boldsymbol{x}_a\|^2 - \frac{2}{|C_i|}\sum_{a \in C_i}\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b$

WCSS Equivalence Proof (5/8): Simplify Double Sum

Current: $\mathrm{WCSS}(C_i) = 2\sum_{a \in C_i} \|\boldsymbol{x}_a\|^2 - \frac{2}{|C_i|}\sum_{a \in C_i}\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b$

Focus on: $\sum_{a \in C_i}\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b$

For fixed a, factor out from inner sum:

$$\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b = \boldsymbol{x}_a^\top \left(\sum_{b \in C_i} \boldsymbol{x}_b \right)$$

WCSS Equivalence Proof (5/8): Simplify Double Sum

Current: $\text{WCSS}(C_i) = 2\sum_{a \in C_i} \|\boldsymbol{x}_a\|^2 - \frac{2}{|C_i|}\sum_{a \in C_i}\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b$

Focus on: $\sum_{a \in C_i}\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b$

For fixed a, factor out from inner sum:

$$\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b = \boldsymbol{x}_a^\top \left(\sum_{b \in C_i} \boldsymbol{x}_b \right)$$

WHY double → single: $\boldsymbol{x}_a$ doesn't depend on b, so pull it out!

WCSS Equivalence Proof (6/8): Use Centroid Definition

We have: $\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b = \boldsymbol{x}_a^\top \left(\sum_{b \in C_i} \boldsymbol{x}_b \right)$

WCSS Equivalence Proof (6/8): Use Centroid Definition

We have: $\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b = \boldsymbol{x}_a^\top \left(\sum_{b \in C_i} \boldsymbol{x}_b \right)$

Centroid definition: $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{b \in C_i} \boldsymbol{x}_b$

Rearrange: $\sum_{b \in C_i} \boldsymbol{x}_b = |C_i| \; \boldsymbol{\mu}_i$

WCSS Equivalence Proof (6/8): Use Centroid Definition

We have: $\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b = \boldsymbol{x}_a^\top \left(\sum_{b \in C_i} \boldsymbol{x}_b \right)$

Centroid definition: $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{b \in C_i} \boldsymbol{x}_b$

Rearrange: $\sum_{b \in C_i} \boldsymbol{x}_b = |C_i| \; \boldsymbol{\mu}_i$

Substitute:

$$\sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b = \boldsymbol{x}_a^\top (|C_i| \; \boldsymbol{\mu}_i) = |C_i| \; \boldsymbol{x}_a^\top \boldsymbol{\mu}_i$$

WCSS Equivalence Proof (7/8): Plug Back In

Now sum over a:

$$\sum_{a\in C_i}\sum_{b\in C_i}\boldsymbol{x}_a^\top\boldsymbol{x}_b = \sum_{a\in C_i}|C_i|\ \boldsymbol{x}_a^\top\boldsymbol{\mu}_i = |C_i|\sum_{a\in C_i}\boldsymbol{x}_a^\top\boldsymbol{\mu}_i$$

WCSS Equivalence Proof (7/8): Plug Back In

Now sum over a:

$$\sum_{a\in C_i}\sum_{b\in C_i}\boldsymbol{x}_a^\top\boldsymbol{x}_b = \sum_{a\in C_i}|C_i|\ \boldsymbol{x}_a^\top\boldsymbol{\mu}_i = |C_i|\sum_{a\in C_i}\boldsymbol{x}_a^\top\boldsymbol{\mu}_i$$

Plug into WCSS:

$$\mathrm{WCSS}(C_i) = 2\sum_{a\in C_i}\|\boldsymbol{x}_a\|^2 - \frac{2}{|C_i|}\cdot|C_i|\sum_{a\in C_i}\boldsymbol{x}_a^\top\boldsymbol{\mu}_i$$

WCSS Equivalence Proof (7/8): Plug Back In

Now sum over a:

$$\sum_{a \in C_i} \sum_{b \in C_i} \boldsymbol{x}_a^\top \boldsymbol{x}_b = \sum_{a \in C_i} |C_i| \; \boldsymbol{x}_a^\top \boldsymbol{\mu}_i = |C_i| \sum_{a \in C_i} \boldsymbol{x}_a^\top \boldsymbol{\mu}_i$$

Plug into WCSS:

$$\text{WCSS}(C_i) = 2 \sum_{a \in C_i} \|\boldsymbol{x}_a\|^2 - \frac{2}{|C_i|} \cdot |C_i| \sum_{a \in C_i} \boldsymbol{x}_a^\top \boldsymbol{\mu}_i$$

$|C_i|$ cancels:

$$= 2 \sum_{a \in C_i} \|\boldsymbol{x}_a\|^2 - 2 \sum_{a \in C_i} \boldsymbol{x}_a^\top \boldsymbol{\mu}_i$$

WCSS Equivalence Proof (8/8): Complete the Square

Current: $\mathrm{WCSS}(C_i) = 2\sum_{a\in C_i} \llbracket |\boldsymbol{x}_a\|^2 - \boldsymbol{x}_a^\top \boldsymbol{\mu}_i]$

WCSS Equivalence Proof (8/8): Complete the Square

Current: $\mathrm{WCSS}(C_i) = 2\sum_{a \in C_i} [\![|\boldsymbol{x}_a\|^2 - \boldsymbol{x}_a^\top \boldsymbol{\mu}_i]$

Goal: Get $\|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|^2 = \|\boldsymbol{x}_a\|^2 - 2\boldsymbol{x}_a^\top \boldsymbol{\mu}_i + \|\boldsymbol{\mu}_i\|^2$

We have first 2 terms, missing: $\|\boldsymbol{\mu}_i\|^2$

WCSS Equivalence Proof (8/8): Complete the Square

Current: $\text{WCSS}(C_i) = 2\sum_{a \in C_i} [\![|\boldsymbol{x}_a\|^2 - \boldsymbol{x}_a^\top \boldsymbol{\mu}_i]$

Goal: Get $\|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|^2 = \|\boldsymbol{x}_a\|^2 - 2\boldsymbol{x}_a^\top \boldsymbol{\mu}_i + \|\boldsymbol{\mu}_i\|^2$

We have first 2 terms, missing: $\|\boldsymbol{\mu}_i\|^2$

WHERE we add/subtract: Inside the sum, add $+\|\boldsymbol{\mu}_i\|^2 - \|\boldsymbol{\mu}_i\|^2$

$$= 2 \sum_{a \in C_i} [\![|\boldsymbol{x}_a\|^2 - \boldsymbol{x}_a^\top \boldsymbol{\mu}_i + \|\boldsymbol{\mu}_i\|^2 - \|\boldsymbol{\mu}_i\|^2]$$

WCSS Equivalence Proof (8/8): Complete the Square

Current: $\mathrm{WCSS}(C_i) = 2\sum_{a \in C_i} [\![|\boldsymbol{x}_a\|^2 - \boldsymbol{x}_a^\top \boldsymbol{\mu}_i]\!]$

Goal: Get $\|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|^2 = \|\boldsymbol{x}_a\|^2 - 2\boldsymbol{x}_a^\top \boldsymbol{\mu}_i + \|\boldsymbol{\mu}_i\|^2$

We have first 2 terms, missing: $\|\boldsymbol{\mu}_i\|^2$

WHERE we add/subtract: Inside the sum, add $+\|\boldsymbol{\mu}_i\|^2 - \|\boldsymbol{\mu}_i\|^2$

$$= 2\sum_{a \in C_i} [\![|\boldsymbol{x}_a\|^2 - \boldsymbol{x}_a^\top \boldsymbol{\mu}_i + \|\boldsymbol{\mu}_i\|^2 - \|\boldsymbol{\mu}_i\|^2]\!]$$

Group first 3 terms:

$$= 2\sum_{a \in C_i} \|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|^2 - 2\sum_{a \in C_i} \|\boldsymbol{\mu}_i\|^2$$

But $\|\boldsymbol{\mu}_i\|^2$ doesn't depend on a: $\sum_{a \in C_i} \|\boldsymbol{\mu}_i\|^2 = |C_i|\ \|\boldsymbol{\mu}_i\|^2$

Actually this term is 0 in the original! Final: $2\sum_{a \in C_i} \|\boldsymbol{x}_a - \boldsymbol{\mu}_i\|^2$ ✓

K-Means: Final Objective

Combining everything, K-Means minimizes:

$$\min_{C_1,\ldots,C_K} \sum_{i=1}^{K} \sum_{\boldsymbol{x}\in C_i} \|\boldsymbol{x}-\boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mu}_i = \frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i} \boldsymbol{x}$ is the centroid of cluster i

K-Means: Final Objective

Combining everything, K-Means minimizes:

$$\min_{C_1,\ldots,C_K} \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$ is the centroid of cluster i

Notation:

- $\Phi = \sum_{i=1}^{K} \mathrm{WCSS}(C_i)$ is the total objective
- $\mathrm{WCSS}(C_i)$ is the measure for a single cluster C_i

K-Means: Final Objective

Combining everything, K-Means minimizes:

$$\min_{C_1,\ldots,C_K} \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$ is the centroid of cluster i

Notation:
- $\Phi = \sum_{i=1}^{K} \mathrm{WCSS}(C_i)$ is the total objective
- $\mathrm{WCSS}(C_i)$ is the measure for a single cluster C_i

Alternative names for the total objective Φ:
- **Inertia** (scikit-learn terminology)
- **Total distortion**

All mean: total sum of squared distances to centroids

Finding the Optimal Centroid

Question: For fixed cluster assignments C_i, what is the best centroid?

Take derivative and set to zero:

$$\frac{\partial}{\partial \boldsymbol{\mu}_i} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2 = 0$$

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$$

Finding the Optimal Centroid

Question: For fixed cluster assignments C_i, what is the best centroid?

Take derivative and set to zero:

$$\frac{\partial}{\partial \boldsymbol{\mu}_i} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2 = 0$$

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$$

Result: The optimal centroid is simply the **mean** of all points in the cluster!

K-Means Algorithm

Key Idea: Alternate between two steps:

K-Means Algorithm

Key Idea: Alternate between two steps:

E-Step (Assignment): Fix centroids, assign points to nearest centroid

M-Step (Update): Fix assignments, recompute centroids as means

K-Means Algorithm

Key Idea: Alternate between two steps:

E-Step (Assignment): Fix centroids, assign points to nearest centroid

M-Step (Update): Fix assignments, recompute centroids as means

Repeat until convergence (assignments don't change)

K-Means Algorithm

Key Idea: Alternate between two steps:

E-Step (Assignment): Fix centroids, assign points to nearest centroid

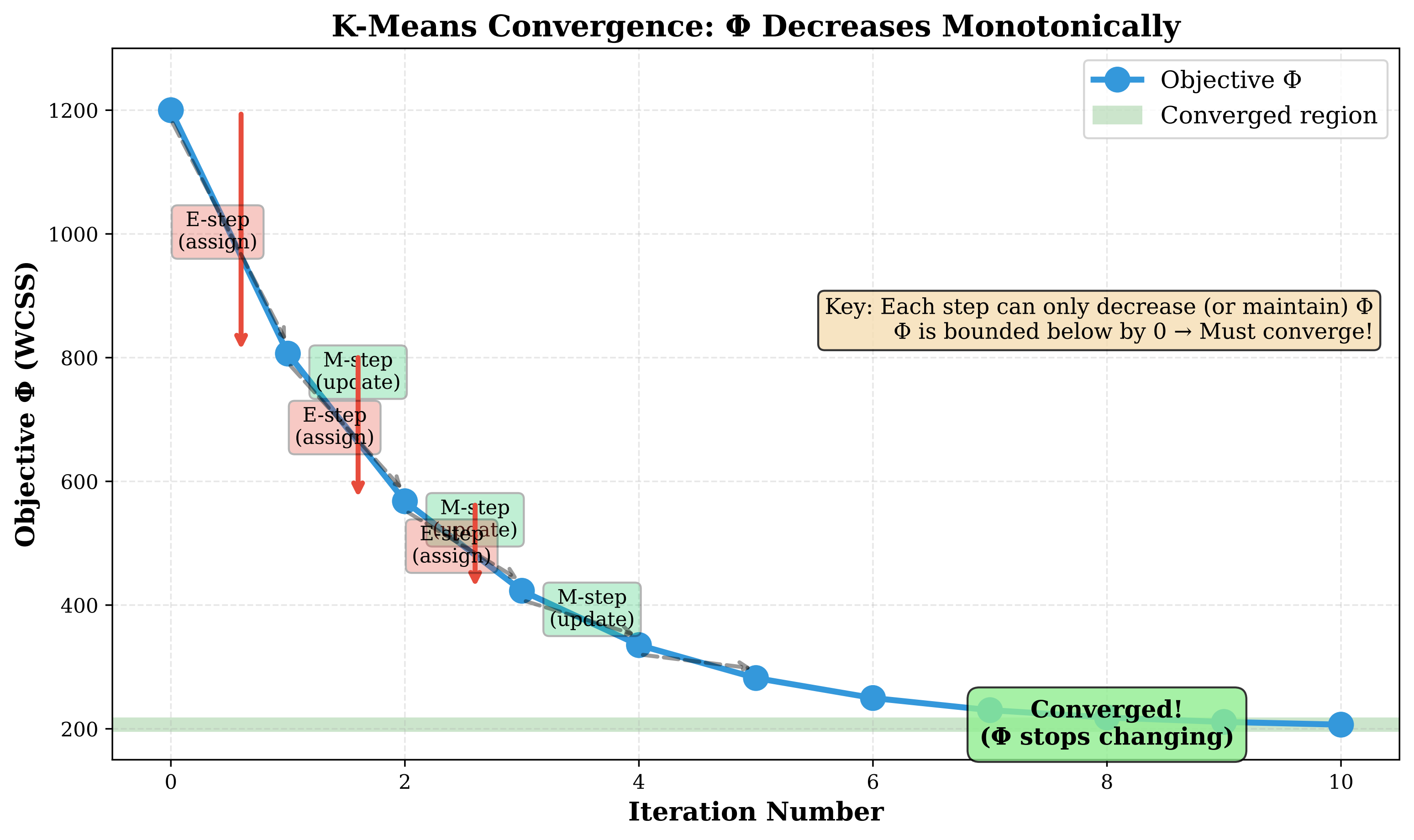
M-Step (Update): Fix assignments, recompute centroids as means

Repeat until convergence (assignments don't change)

Guarantee: Each step decreases (or maintains) the objective

Algorithm converges to a **local minimum**

Why K-Means Converges



Why K-Means Converges

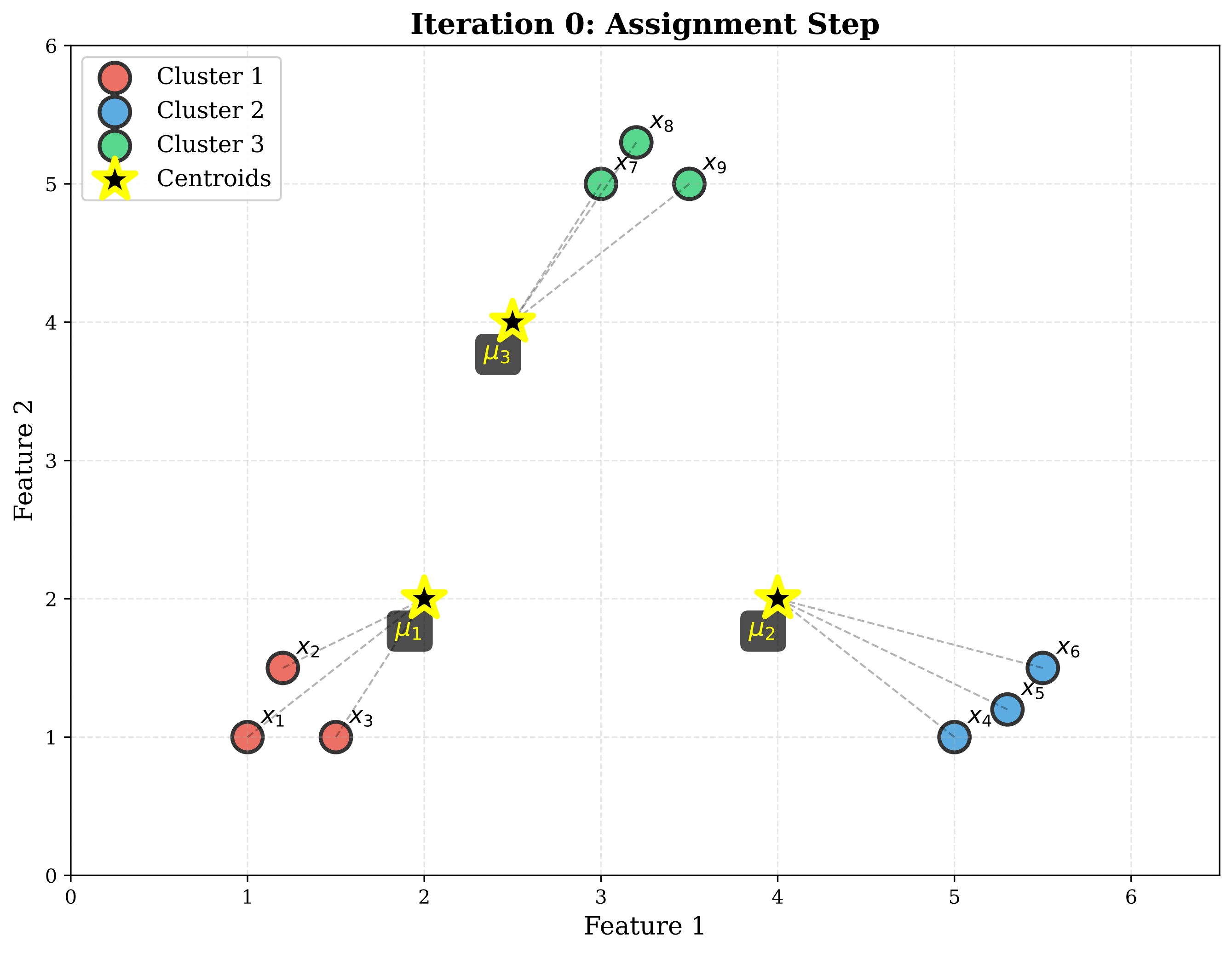
Key insight: Each step can only **decrease** (or maintain) Φ

- **E-step**: Assign to nearest centroid $\longrightarrow$ Φ decreases
- **M-step**: Move centroid to cluster mean $\longrightarrow$ Φ decreases

Convergence: $\Phi \geq 0$ (bounded below) + monotone decrease $\longrightarrow$ must stop!

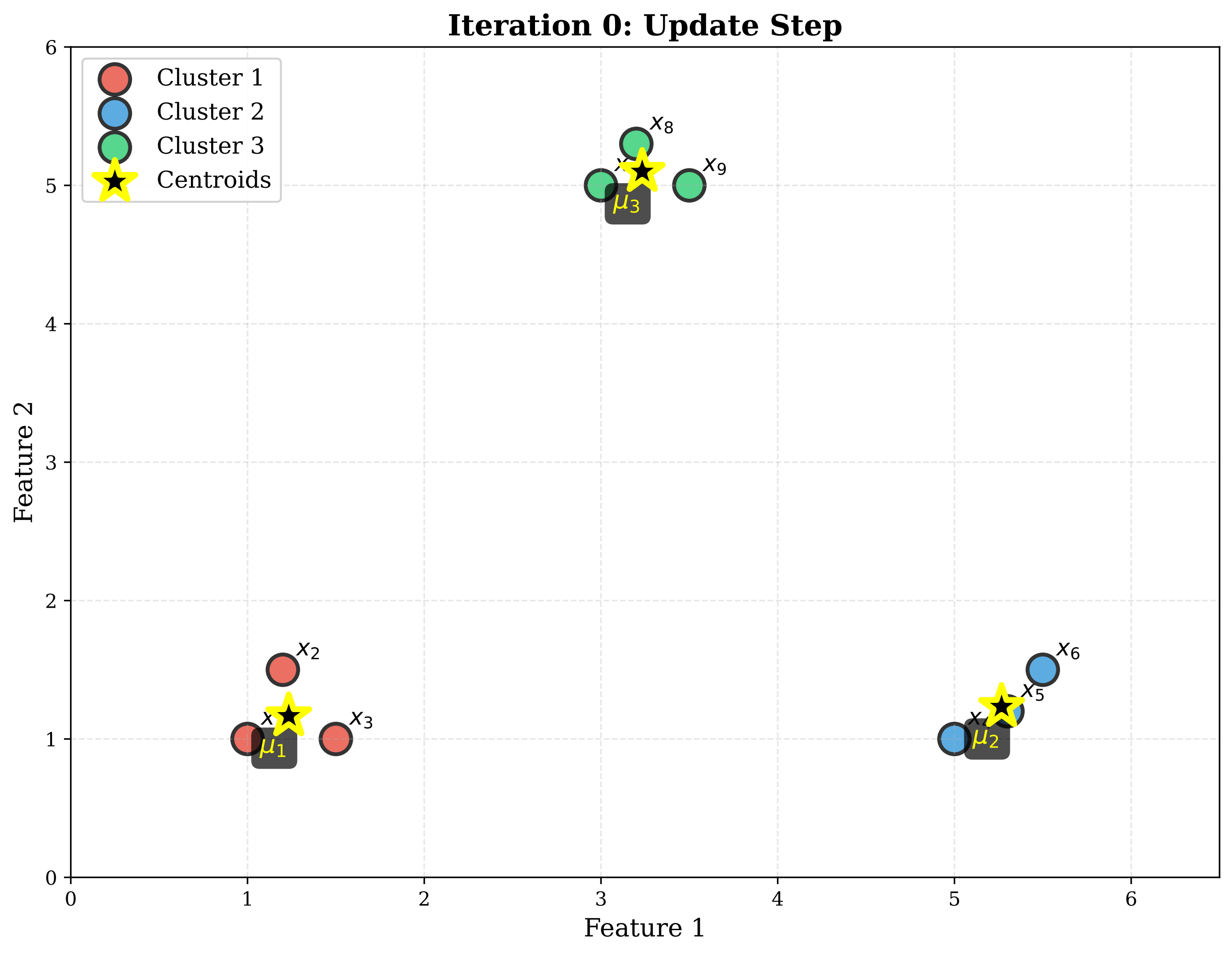
K-Means: Iteration 0 (Initialization)

K-Means: Iteration 0 (Initialization)



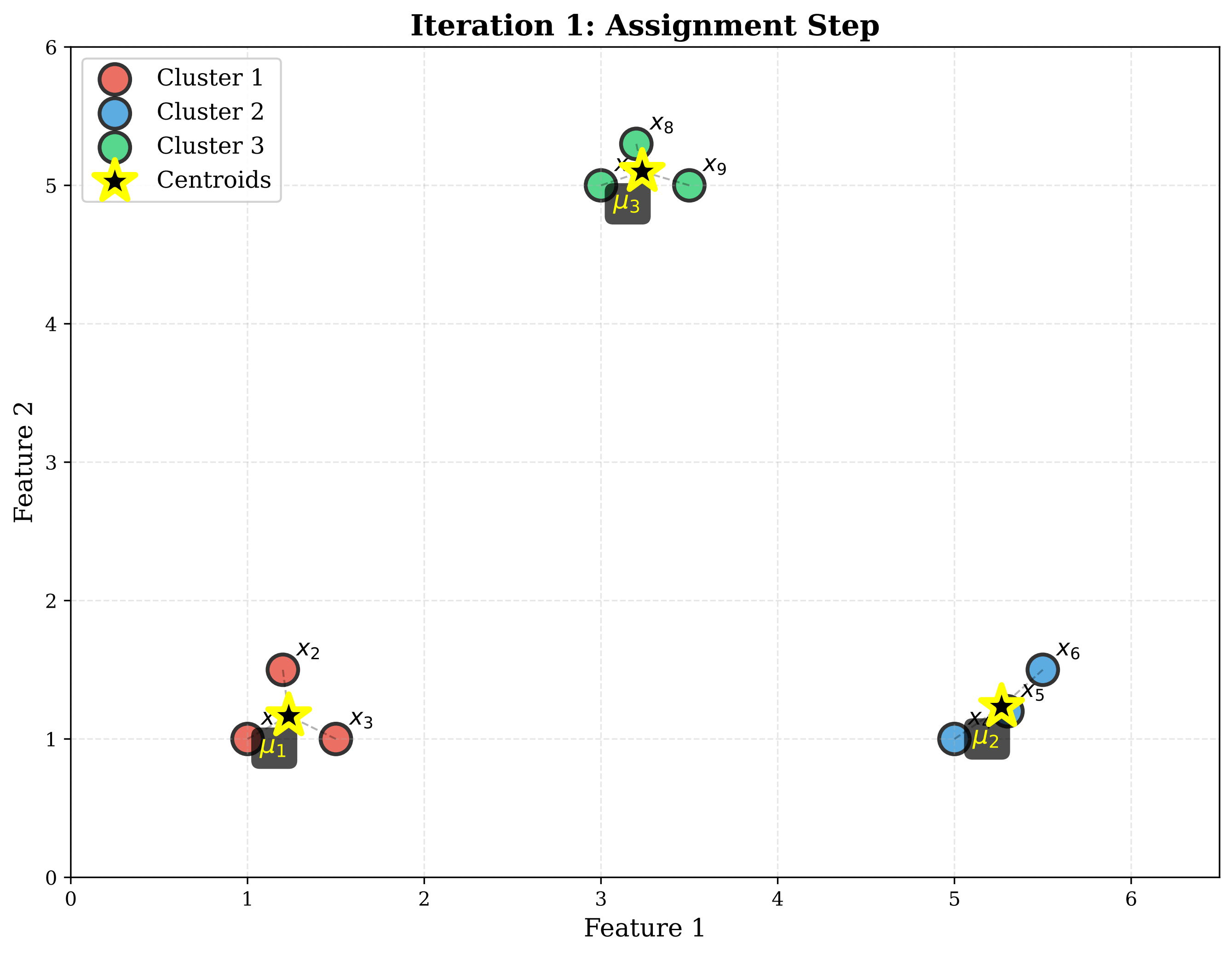
K-Means: Iteration 0 (Update)

K-Means: Iteration 0 (Update)



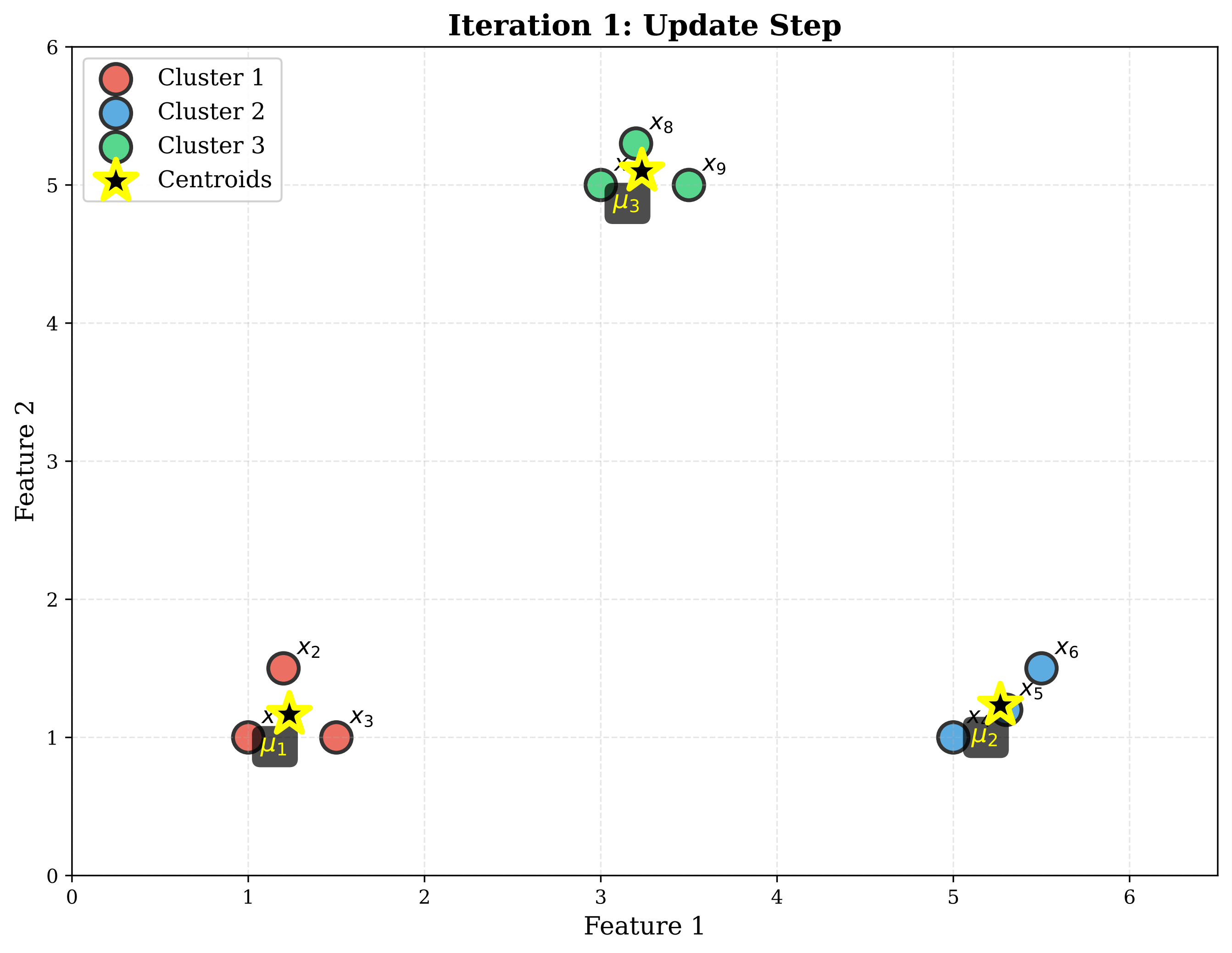
K-Means: Iteration 1 (Assignment)

K-Means: Iteration 1 (Assignment)



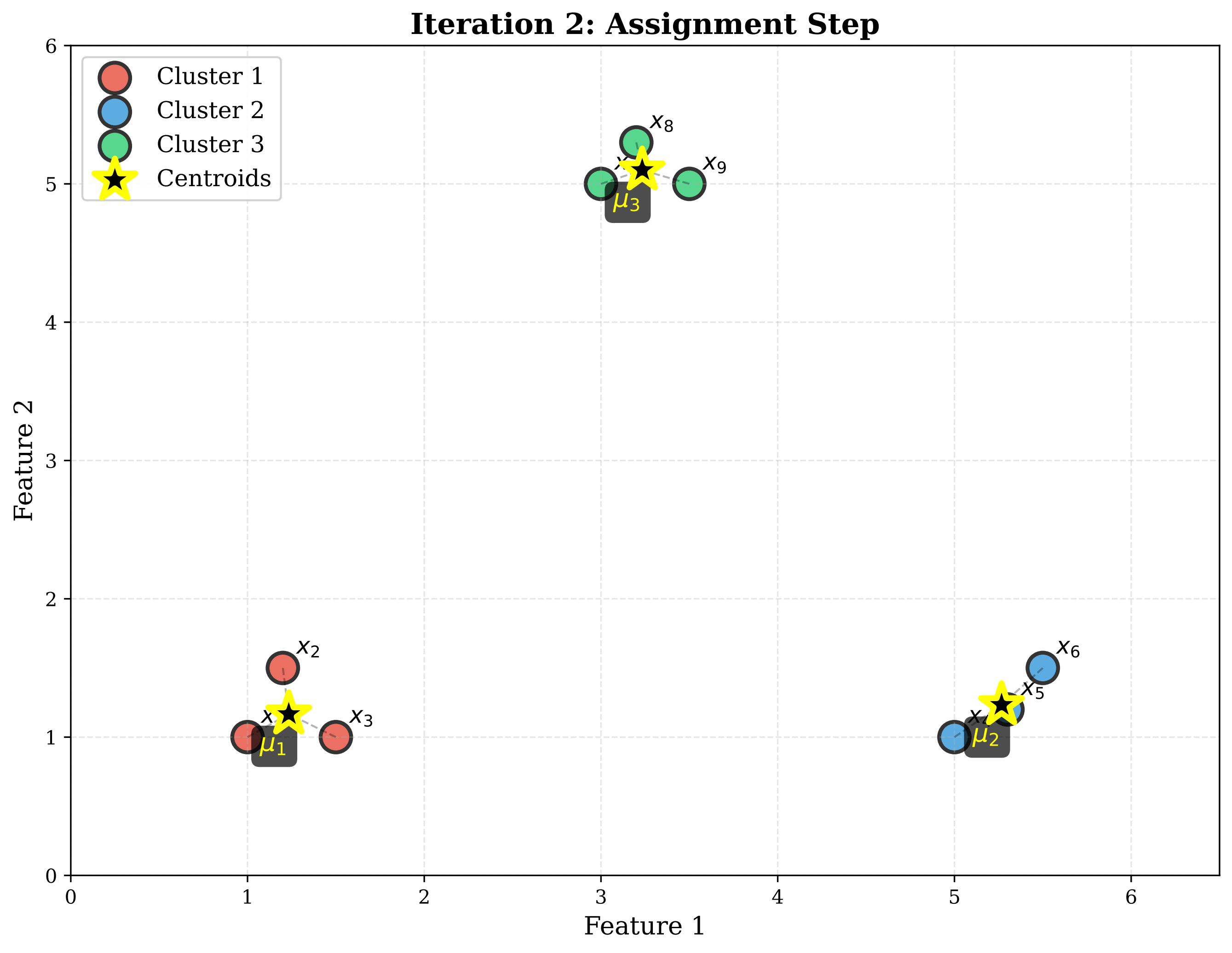
K-Means: Iteration 1 (Update)

K-Means: Iteration 1 (Update)



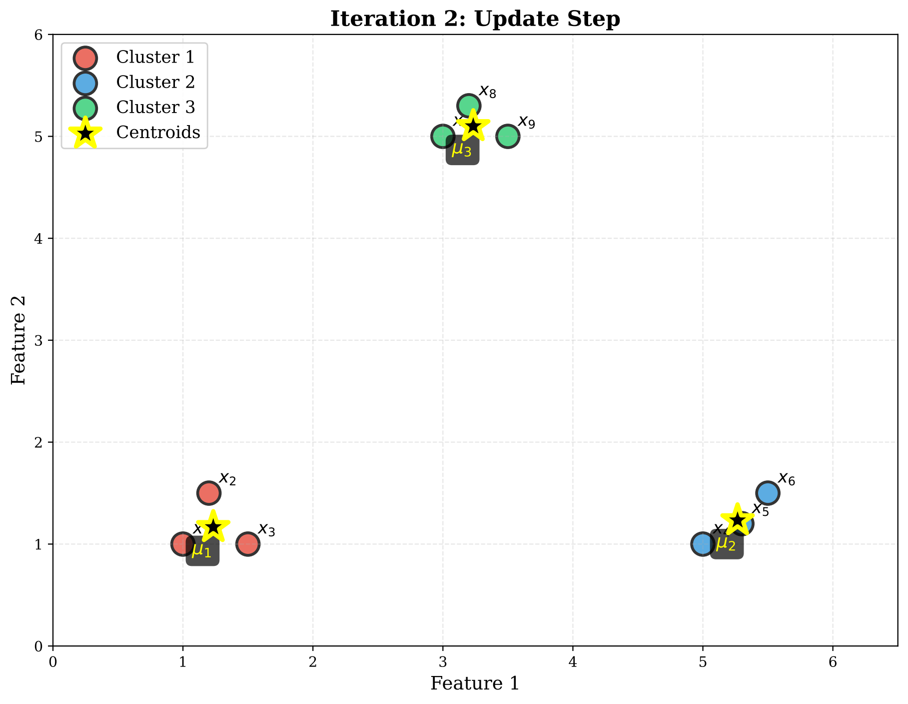
K-Means: Iteration 2 (Assignment)

K-Means: Iteration 2 (Assignment)



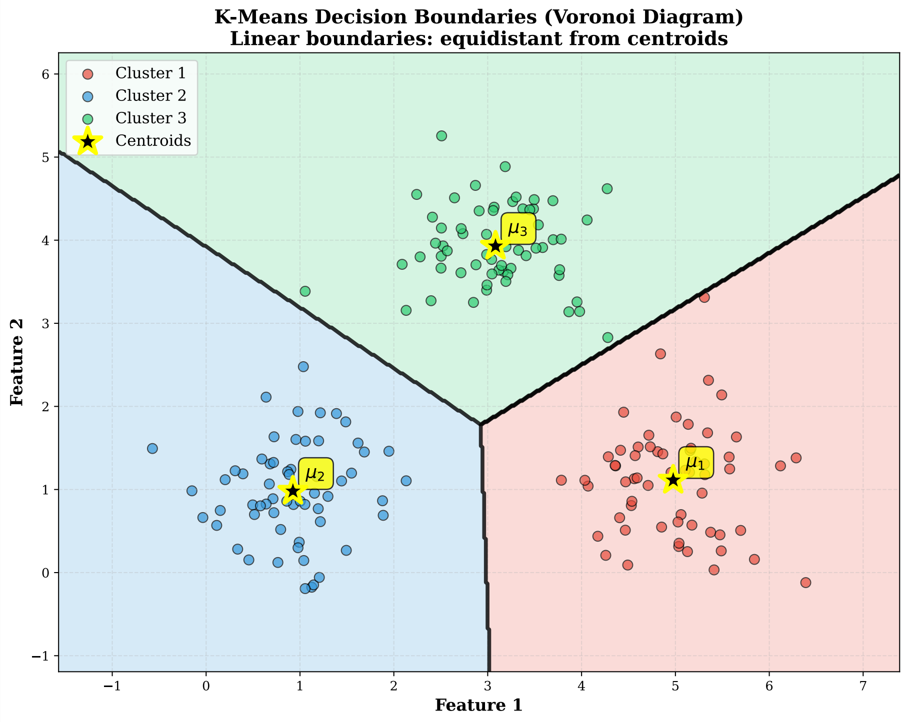
K-Means: Convergence & Decision Boundaries

Convergence



Assignments don't change $\longrightarrow$ Algorithm terminates!

Voronoi Diagram



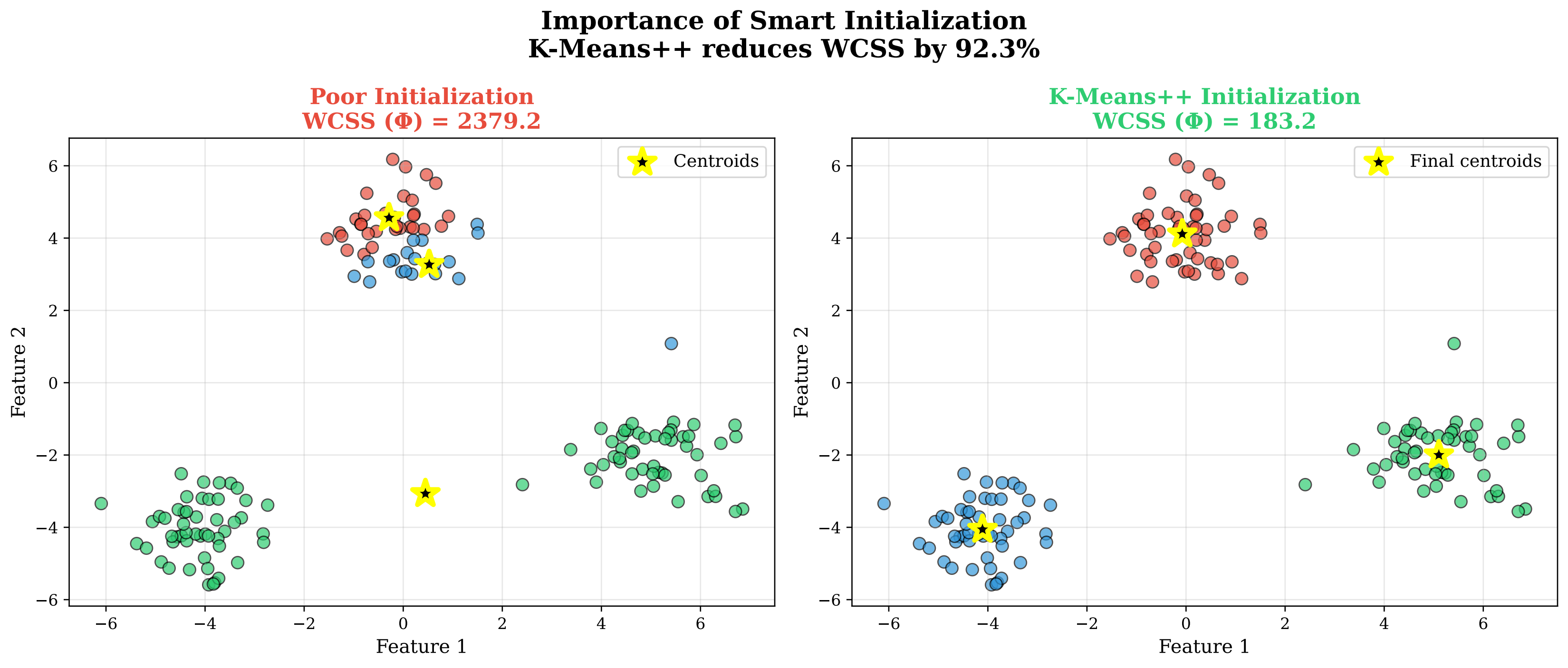
K-Means creates **linear decision boundaries**

Practical Issues

Practical Issues & Advanced Variants

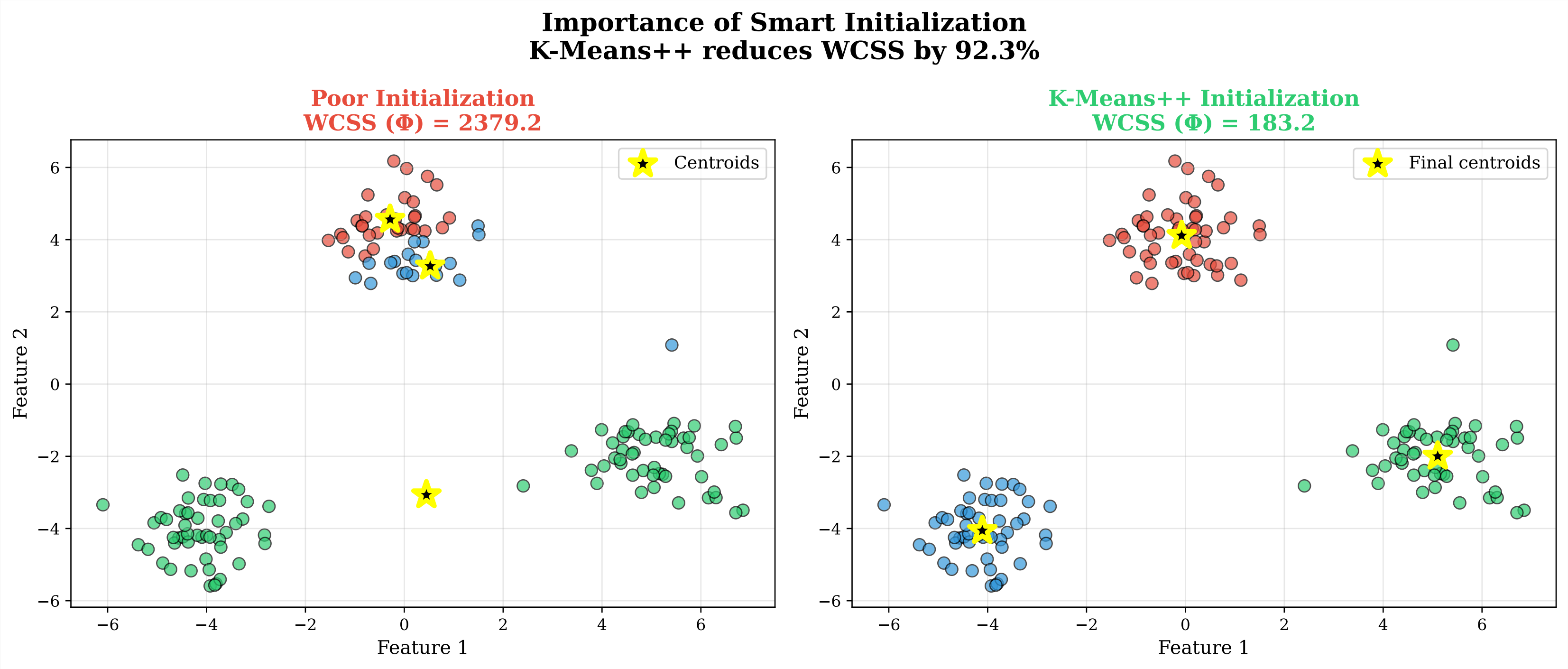
Issue #1: Poor Initialization

Problem: Bad initialization leads to worse local minima (higher WCSS/Φ)



Issue #1: Poor Initialization

Problem: Bad initialization leads to worse local minima (higher WCSS/Φ)



Observation: Poor init ⟶ much higher Φ (worse clustering!)

Solution: K-Means++ (smart initialization)

K-Means++ Algorithm

Algorithm:

1. Choose first centroid $\boldsymbol{\mu}_1$ uniformly at random from data
2. For $k = 2, 3, ..., K$:
   - For each point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$ = distance to nearest centroid so far
   - Choose next centroid $\boldsymbol{\mu}_k$ with probability $\propto D(\boldsymbol{x})^2$
3. Run standard K-Means with these K initial centroids

K-Means++ Algorithm

Algorithm:

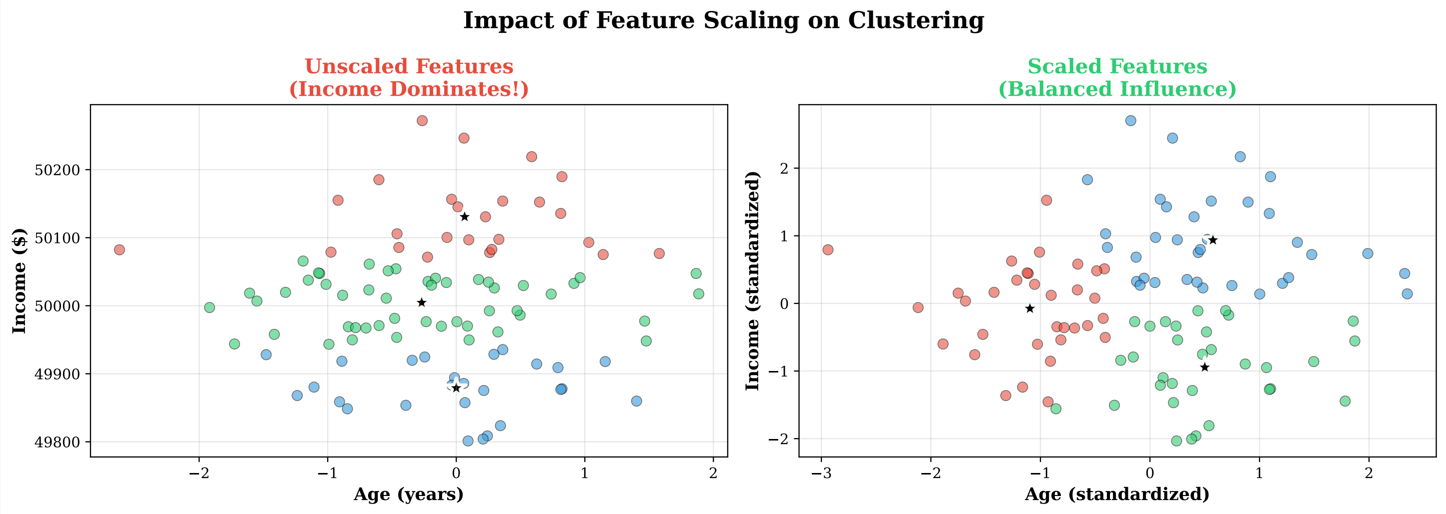
1. Choose first centroid $\boldsymbol{\mu}_1$ uniformly at random from data
2. For $k = 2, 3, \ldots, K$:
   - For each point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$ = distance to nearest centroid so far
   - Choose next centroid $\boldsymbol{\mu}_k$ with probability $\propto D(\boldsymbol{x})^2$
3. Run standard K-Means with these K initial centroids

Theorem (Arthur & Vassilvitskii, 2007):

K-Means++ is $O(\log K)$-competitive with optimal clustering

Issue #2: Feature Scaling

Problem: Features with large ranges dominate distance calculations



Always standardize: $x'_j = \frac{x_j - \mu_j}{\sigma_j}$ (z-score)

Issue #3: Time Complexity

K-Means complexity: $O(n \cdot K \cdot d \cdot T)$

where:

- n = number of points
- K = number of clusters
- d = dimensionality
- T = number of iterations (typically 10-100)

Time Complexity: Detailed Breakdown

Each iteration has two steps:

Time Complexity: Detailed Breakdown

Each iteration has two steps:

1. Assignment Step (E-step):

- For each point $\boldsymbol{x}_i$ (n points):
  - Compute distance to each centroid $\boldsymbol{\mu}_k$ (K centroids)
  - Distance computation: $\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 = \sum_{j=1}^{d} \left(x_{ij} - \mu_{kj}\right)^2$
  - Cost per distance: $O(d)$ (sum over d dimensions)
- **Total**: $n \times K \times d = O(nKd)$

Time Complexity: Detailed Breakdown

Each iteration has two steps:

1. Assignment Step (E-step):

- For each point $\boldsymbol{x}_i$ (n points):
  - Compute distance to each centroid $\boldsymbol{\mu}_k$ (K centroids)
  - Distance computation: $\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 = \sum_{j=1}^{d} \left(x_{ij} - \mu_{kj}\right)^2$
  - Cost per distance: $O(d)$ (sum over d dimensions)
- **Total**: $n \times K \times d = O(nKd)$

Why? n points × K centroids × d operations per distance

Time Complexity: Update Step

2. Update Step (M-step):

- For each cluster k (K clusters):
  - Compute new centroid: $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i$
  - Need to sum all points in cluster C_k across d dimensions
- Total points across all clusters: n
- **Total**: $n \times d = O(nd)$

Time Complexity: Update Step

2. Update Step (M-step):

- For each cluster k (K clusters):
  - Compute new centroid: $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i$
  - Need to sum all points in cluster C_k across d dimensions
- Total points across all clusters: n
- **Total**: $n \times d = O(nd)$

Why? Each point contributes once to its cluster's centroid (across d dimensions)

Time Complexity: Update Step

2. Update Step (M-step):

- For each cluster k (K clusters):
  - Compute new centroid: $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i$
  - Need to sum all points in cluster C_k across d dimensions
- Total points across all clusters: n
- **Total**: $n \times d = O(nd)$

Why? Each point contributes once to its cluster's centroid (across d dimensions)

Per iteration total: $O(nKd) + O(nd) = O(nKd)$ (since $K \geq 1$)

Time Complexity: Overall Algorithm

Total complexity: $O(T \cdot nKd)$

- T iterations needed for convergence
- Typically $T \approx$ 10-100 (often converges quickly)
- Each iteration: $O(nKd)$

Time Complexity: Overall Algorithm

Total complexity: $O(T \cdot nKd)$

- T iterations needed for convergence
- Typically $T \approx$ 10-100 (often converges quickly)
- Each iteration: $O(nKd)$

Practical implications:

- Linear in $n \longrightarrow$ scales well with data size!
- Linear in $K \longrightarrow$ more clusters = slower
- Linear in $d \longrightarrow$ high dimensions = slower
- Can be slow for large n (millions of points)

Time Complexity: Overall Algorithm

Total complexity: $O(T \cdot nKd)$

- T iterations needed for convergence
- Typically $T \approx$ 10-100 (often converges quickly)
- Each iteration: $O(nKd)$

Practical implications:

- Linear in $n \longrightarrow$ scales well with data size!
- Linear in $K \longrightarrow$ more clusters = slower
- Linear in $d \longrightarrow$ high dimensions = slower
- Can be slow for large n (millions of points)

Mini-Batch K-Means: Faster variant for large datasets

- Use when $n > 10,000$
- Next slide: detailed algorithm

Mini-Batch K-Means: Detailed Algorithm

Idea: Use random samples instead of all data each iteration

Mini-Batch K-Means: Detailed Algorithm

Idea: Use random samples instead of all data each iteration

Algorithm:

1. Initialize centroids (using K-Means++)
2. **For each iteration**:
   - Sample b random points (mini-batch) from dataset
   - **E-step**: Assign these b points to nearest centroids
   - **M-step**: Update centroids using **only** these b points
3. Repeat until convergence

Mini-Batch K-Means: Detailed Algorithm

Idea: Use random samples instead of all data each iteration

Algorithm:

1. Initialize centroids (using K-Means++)
2. **For each iteration**:
   - Sample b random points (mini-batch) from dataset
   - **E-step**: Assign these b points to nearest centroids
   - **M-step**: Update centroids using **only** these b points
3. Repeat until convergence

Complexity: $O(T \cdot bKd)$ where $b \ll n$ (e.g., $b = 100$, $n = 1,000,000$)

Mini-Batch K-Means: Detailed Algorithm

Idea: Use random samples instead of all data each iteration

Algorithm:

1. Initialize centroids (using K-Means++)
2. **For each iteration**:
   - Sample b random points (mini-batch) from dataset
   - **E-step**: Assign these b points to nearest centroids
   - **M-step**: Update centroids using **only** these b points
3. Repeat until convergence

Complexity: $O(T \cdot bKd)$ where $b \ll n$ (e.g., $b = 100$, $n = 1,000,000$)

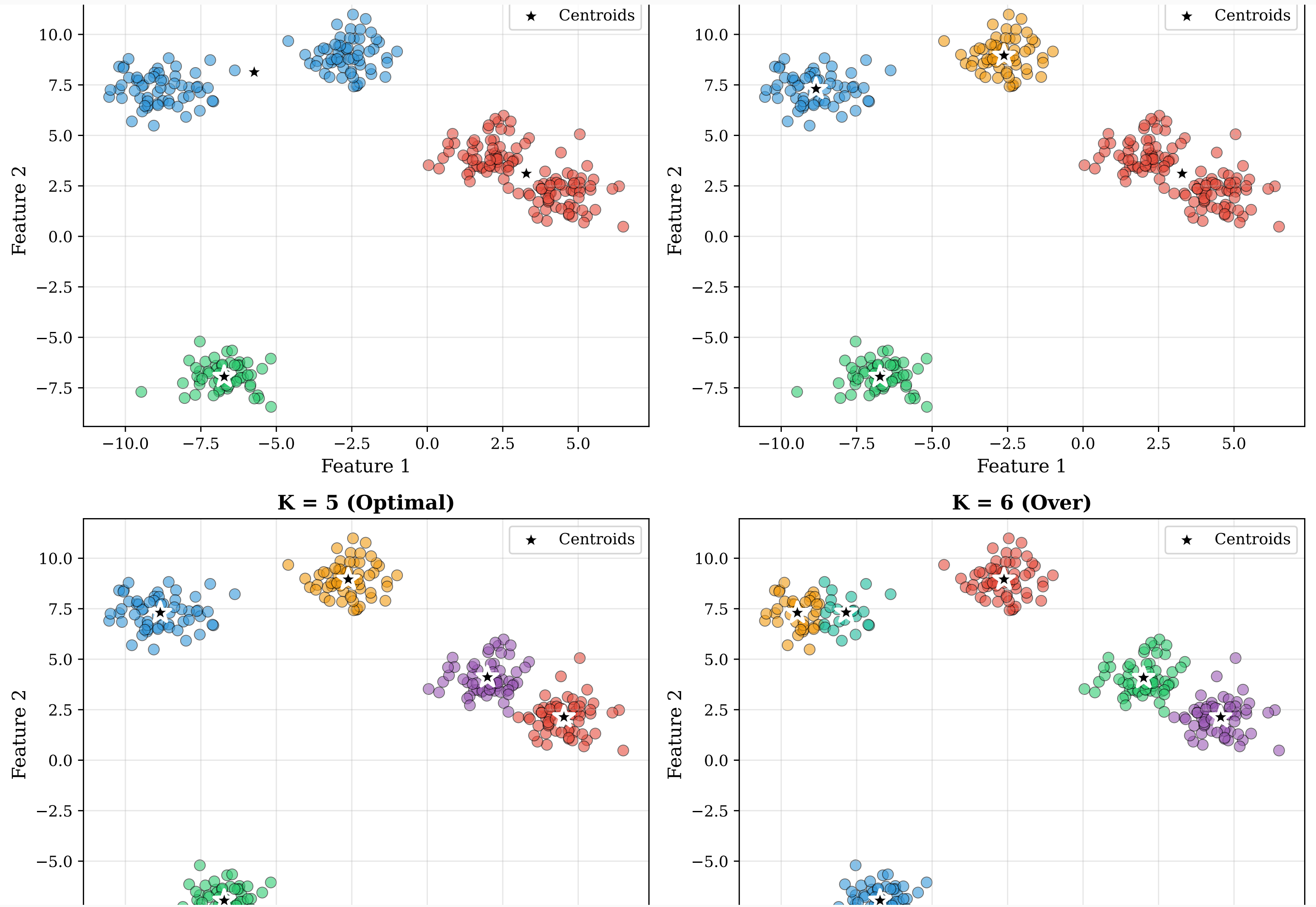
Trade-off:

- 10-100× faster than standard K-Means
- Slight accuracy loss (usually < 5%)
- Good for: large datasets, streaming data, online learning

Issue #4: Choosing K

Problem: How many clusters should we find?

Issue #4: Choosing K

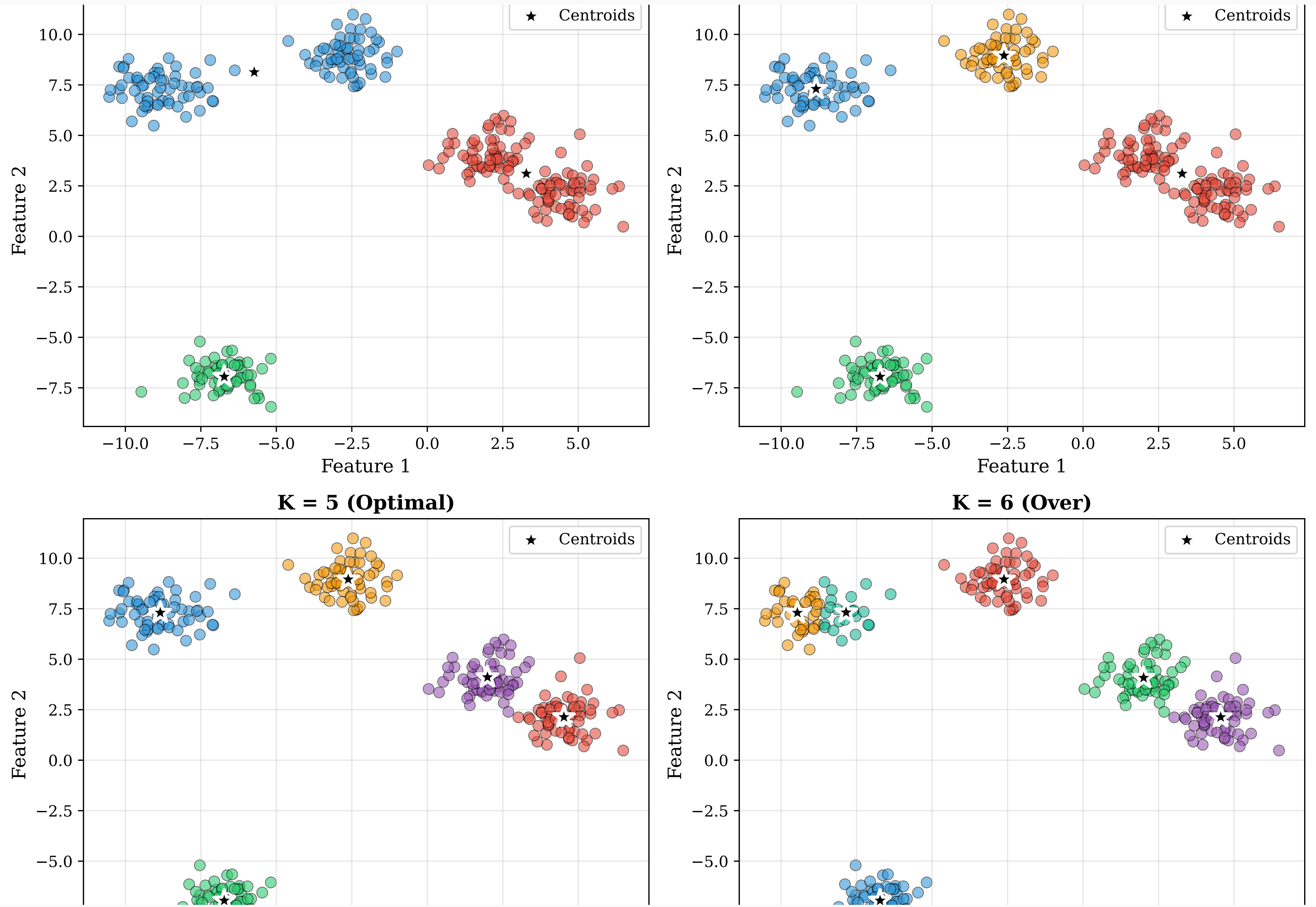


Issue #4: Choosing K

Issue #4: Choosing K

Problem: How many clusters should we find?

Issue #4: Choosing K

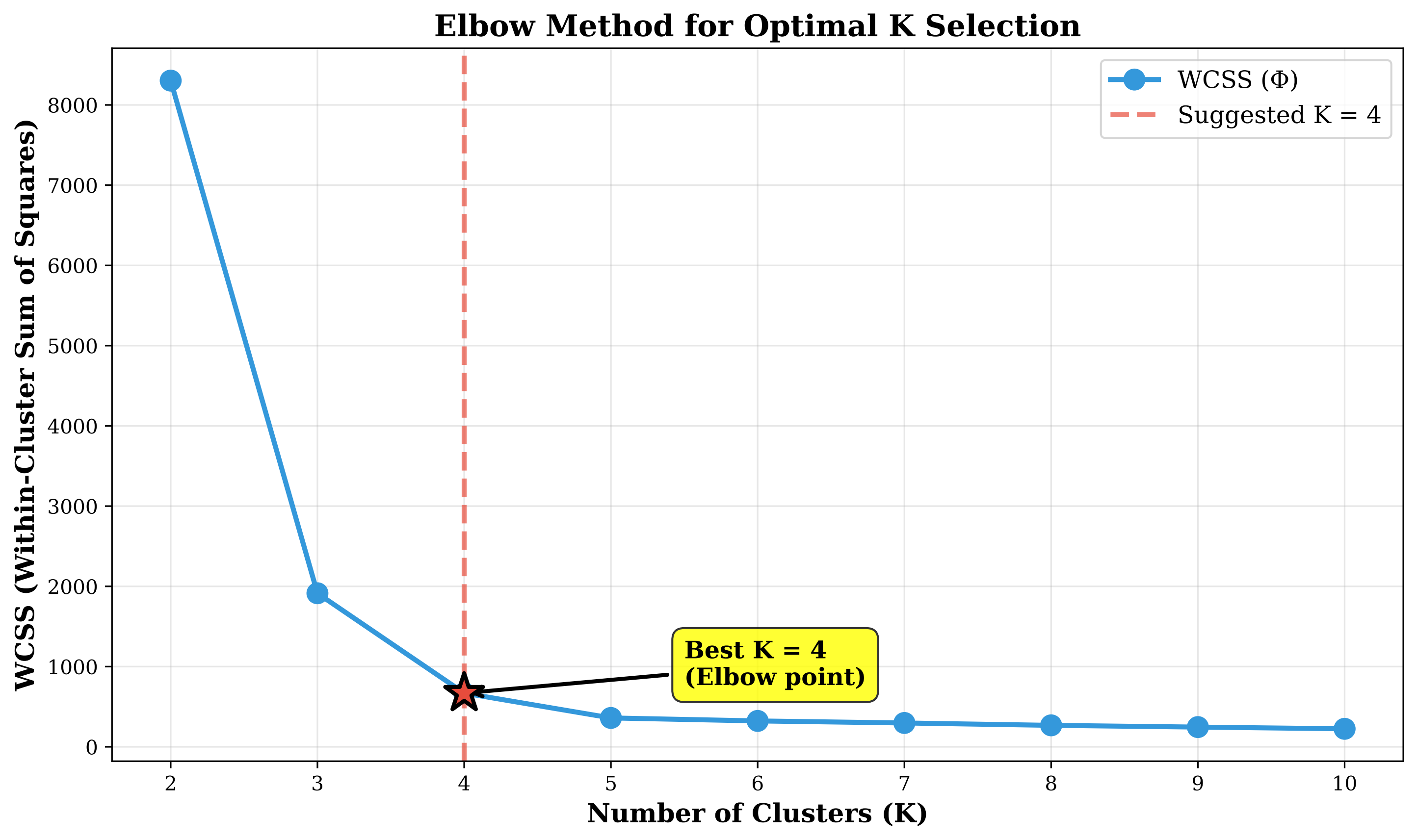


Key Observation: Different K values give different clusterings!

We need a **quantitative** way to measure cluster quality.

Choosing K: The Elbow Method

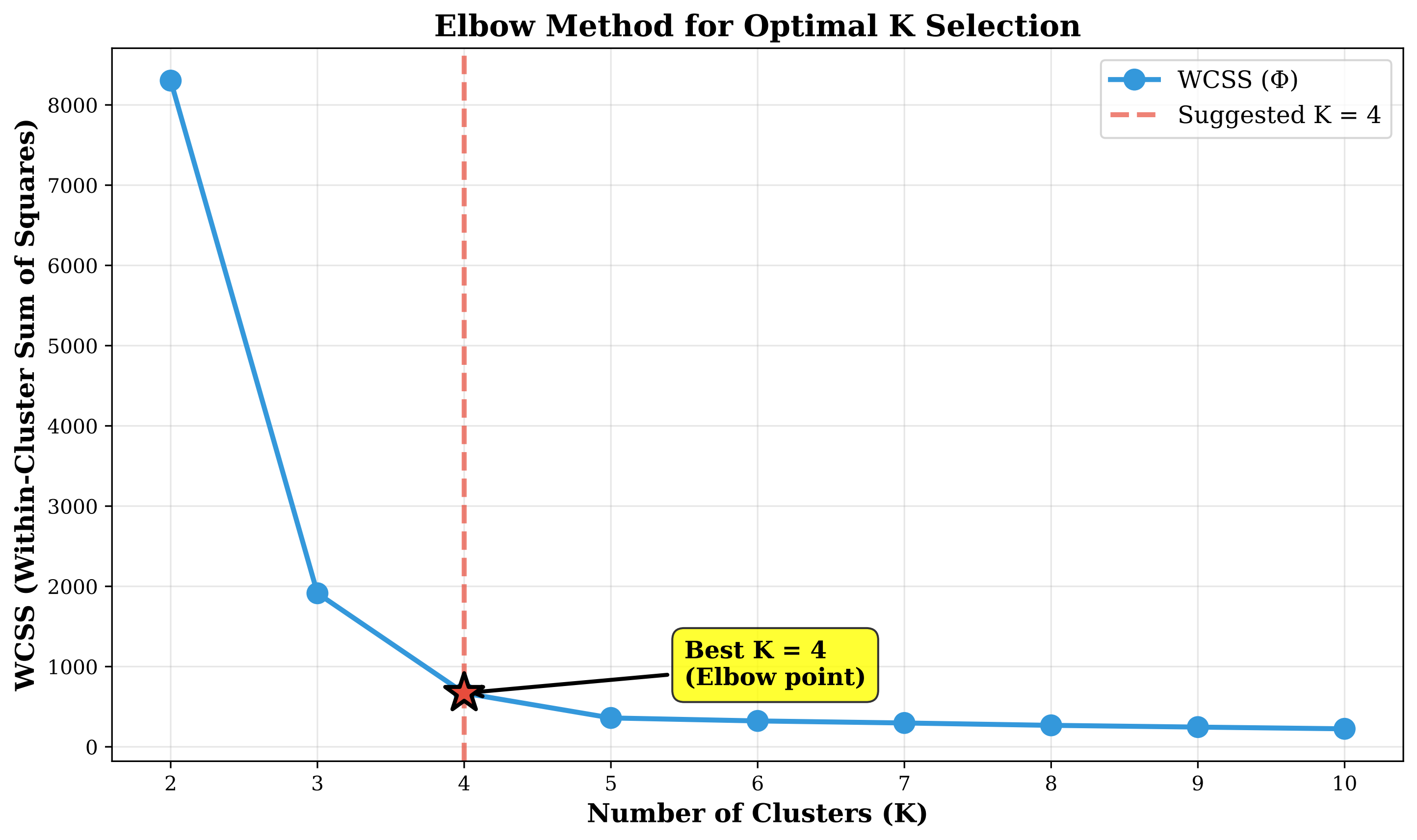
Approach: Plot total objective $\Phi = \sum_{i=1}^{K} \mathrm{WCSS}(C_i)$ vs. K



Choosing K: The Elbow Method

Choosing K: The Elbow Method

Approach: Plot total objective $\Phi = \sum_{i=1}^{K} \mathrm{WCSS}(C_i)$ vs. K

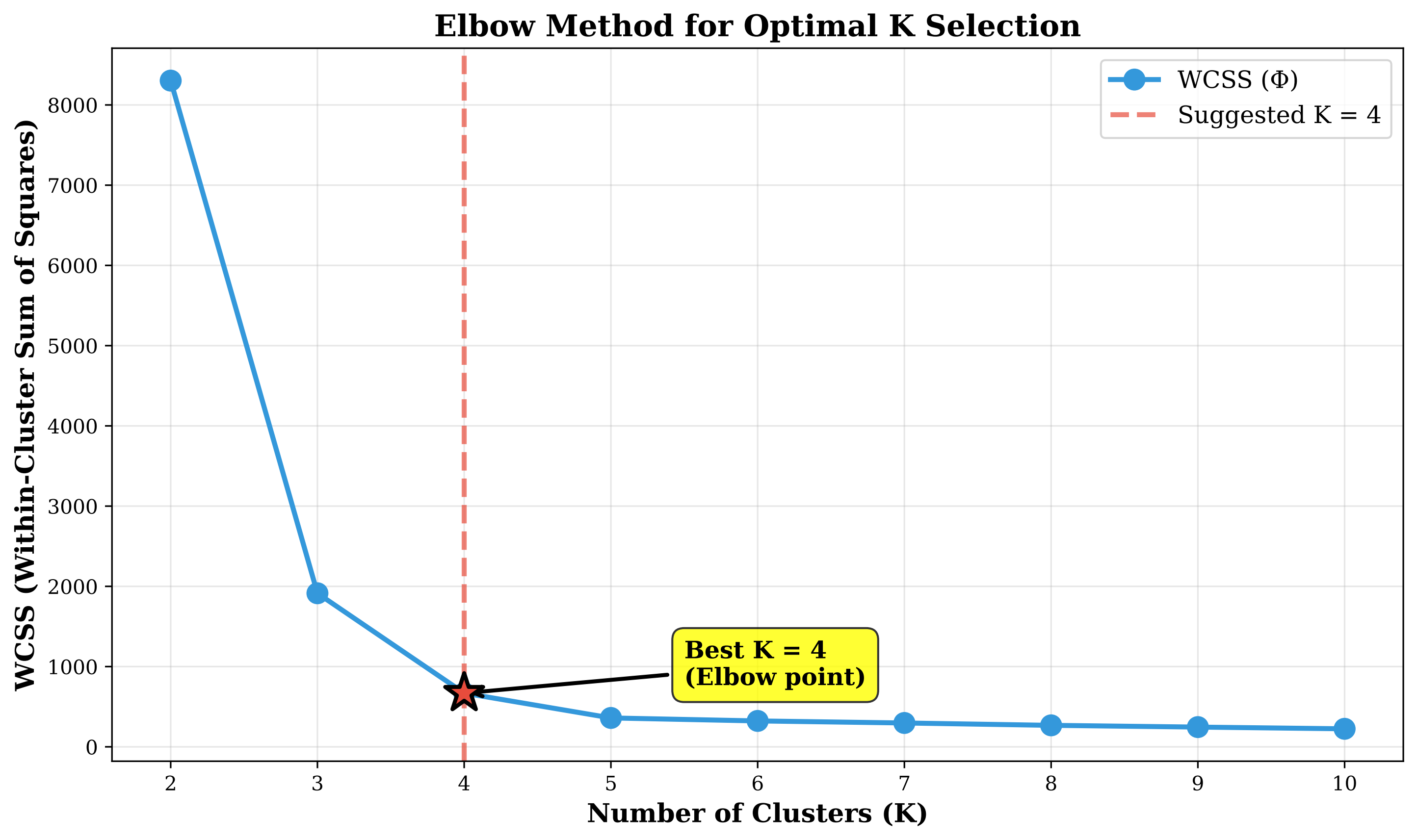


Choosing K: The Elbow Method

Observation: Elbow suggests $K = 4$ (but true K = 5!)

Choosing K: The Elbow Method

Approach: Plot total objective $\Phi = \sum_{i=1}^{K} \mathrm{WCSS}(C_i)$ vs. K



Choosing K: The Elbow Method

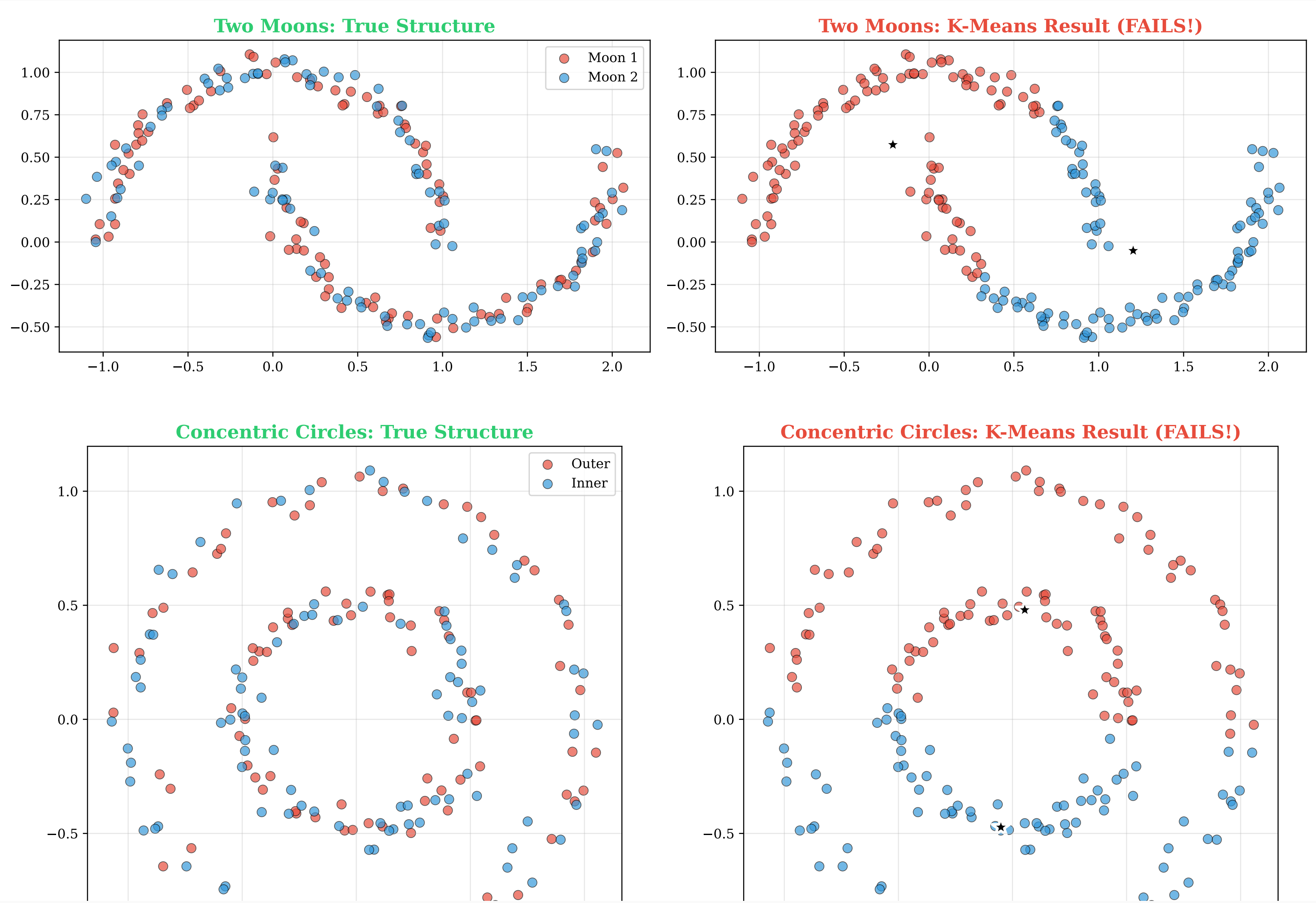
Observation: Elbow suggests $K = 4$ (but true K = 5!)

Warning: “Elbow” can be subjective in practice

Issue #5: Non-Convex Shapes

K-Means Assumption: Clusters are convex, isotropic (spherical)

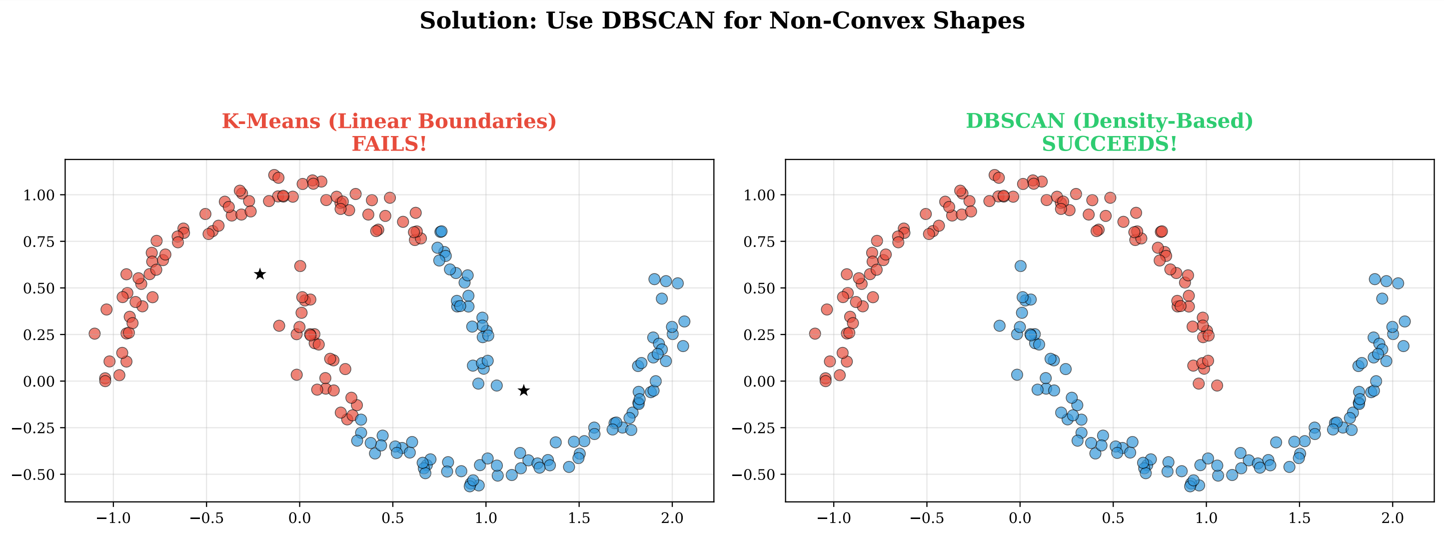
Issue #5: Non-Convex Shapes



Issue #5: Non-Convex Shapes

Limitation: K-Means uses **linear decision boundaries**

Solution: DBSCAN for Non-Convex Shapes



DBSCAN: Density-based, no K needed, finds arbitrary shapes and noise

Summary: K-Means Techniques

Technique	Problem	When to Use
K-Means++	Poor initialization	Always!
Standardization	Feature scale mismatch	Different units/ranges
Mini-Batch	Large datasets	$n > 10,000$
Elbow/Silhouette	Choosing K	K unknown
DBSCAN/Spectral	Non-convex shapes	Arbitrary shapes
GMM	Elliptical clusters	Soft assignments

Hierarchical Clustering

Hierarchical Clustering

When K-Means Fails

Problems with K-Means:

- Need to specify K in advance
- Assumes spherical, equal-sized clusters
- No hierarchy

When K-Means Fails

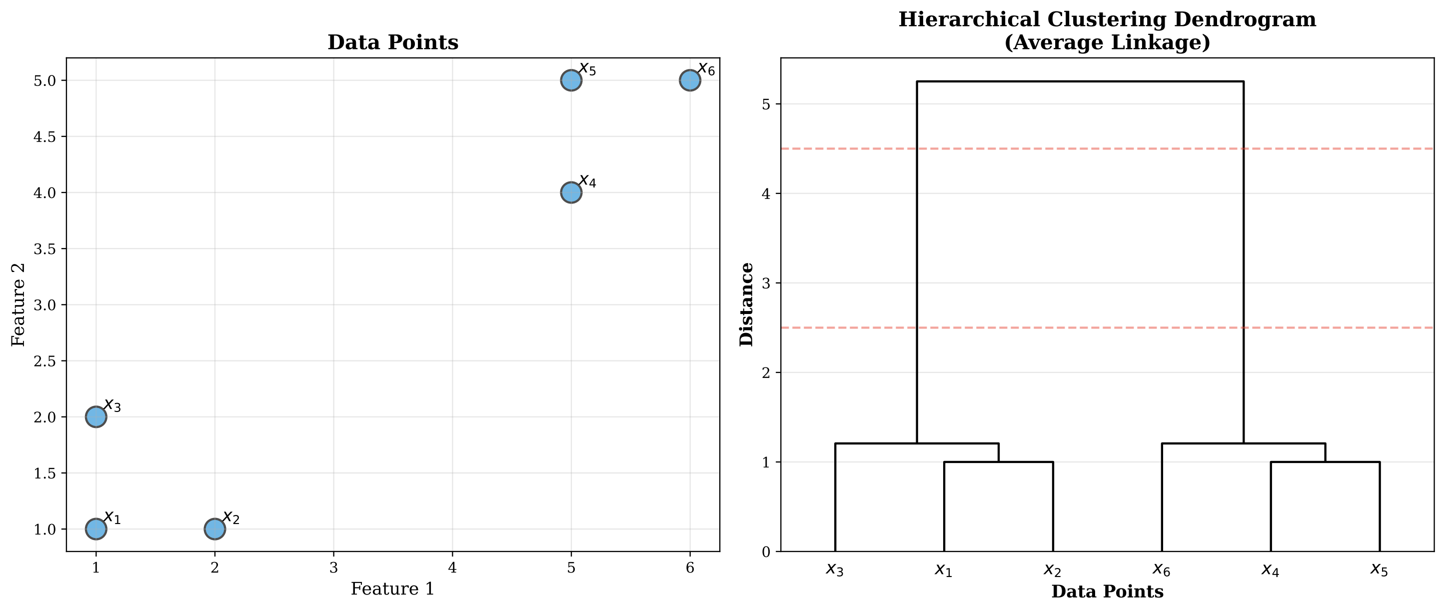
Problems with K-Means:

- Need to specify K in advance
- Assumes spherical, equal-sized clusters
- No hierarchy

Hierarchical Clustering: Builds a **tree** (dendrogram)

- No need to specify K in advance!
- Get clustering at multiple granularities
- Deterministic (no random initialization)

Hierarchical Clustering: Dendrogram



Reading: Leaves = points, Height = merge distance, Cut = choose K

Hierarchical Clustering: Example Setup

Data: 6 points in 2D

- $\boldsymbol{x}_1 = (1, 1)$, $\boldsymbol{x}_2 = (2, 1)$, $\boldsymbol{x}_3 = (1, 2)$
- $\boldsymbol{x}_4 = (5, 4)$, $\boldsymbol{x}_5 = (5, 5)$, $\boldsymbol{x}_6 = (6, 5)$

Hierarchical Clustering: Example Setup

Data: 6 points in 2D

- $\boldsymbol{x}_1 = (1, 1)$, $\boldsymbol{x}_2 = (2, 1)$, $\boldsymbol{x}_3 = (1, 2)$
- $\boldsymbol{x}_4 = (5, 4)$, $\boldsymbol{x}_5 = (5, 5)$, $\boldsymbol{x}_6 = (6, 5)$

Algorithm: Agglomerative (bottom-up)

1. Start: Each point is its own cluster (6 clusters)
2. Repeat: Merge the two closest clusters
3. Stop: When all merged into one tree

Hierarchical Clustering: Example Setup

Data: 6 points in 2D

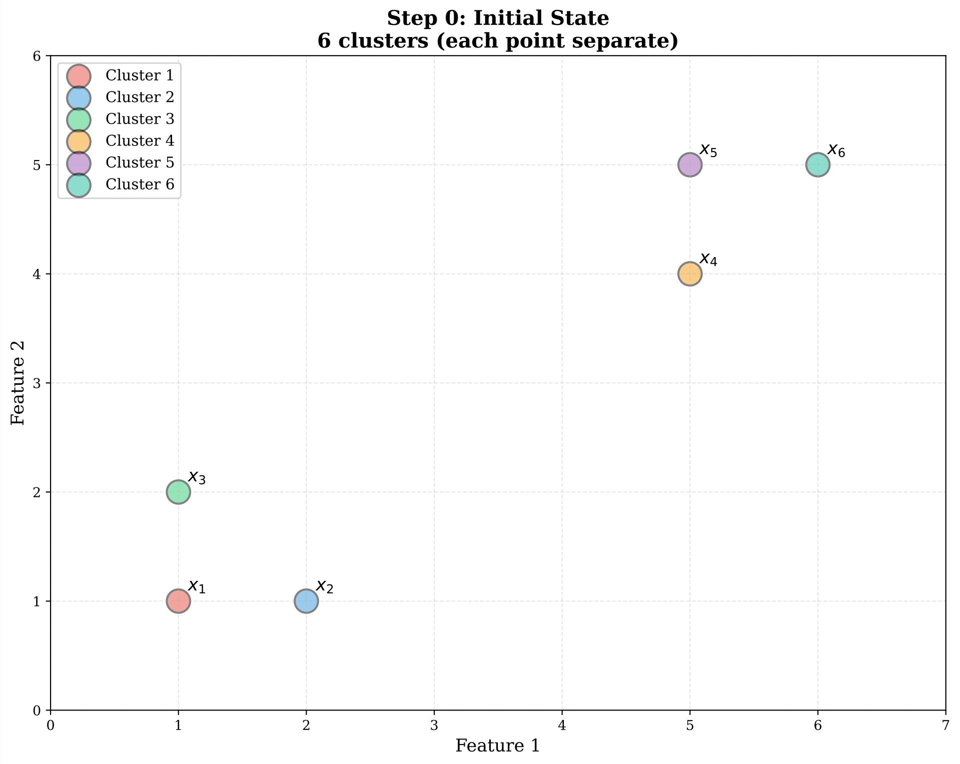
- $\boldsymbol{x}_1 = (1, 1)$, $\boldsymbol{x}_2 = (2, 1)$, $\boldsymbol{x}_3 = (1, 2)$
- $\boldsymbol{x}_4 = (5, 4)$, $\boldsymbol{x}_5 = (5, 5)$, $\boldsymbol{x}_6 = (6, 5)$

Algorithm: Agglomerative (bottom-up)

1. Start: Each point is its own cluster (6 clusters)
2. Repeat: Merge the two closest clusters
3. Stop: When all merged into one tree

Linkage: Average linkage (mean distance between all pairs)

Step 0: Initial State



Calculate distances:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$$

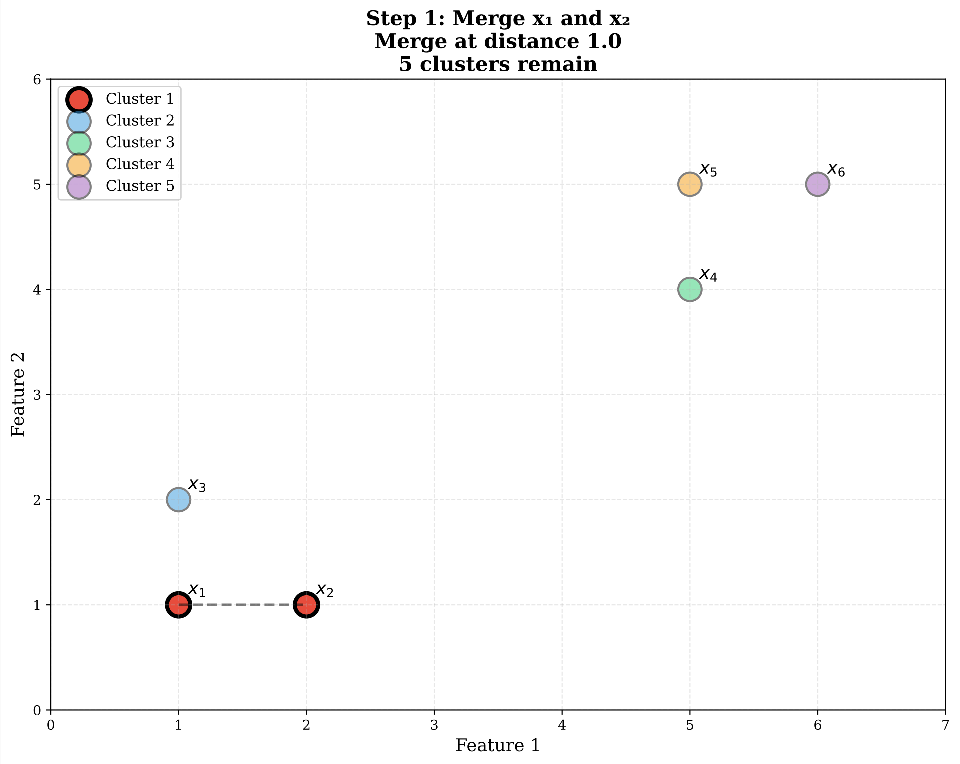
Closest pairs:

- $d(\boldsymbol{x}_1, \boldsymbol{x}_2) = 1.0$
- $d(\boldsymbol{x}_1, \boldsymbol{x}_3) = 1.0$
- $d(\boldsymbol{x}_4, \boldsymbol{x}_5) = 1.0$

Action: Merge x_1 and x_2

Dendrogram height = 1.0

Step 1: After Merging x_1 and x_2



Current clusters (5):

- $\{x_1, x_2\}$
- $\{x_3\}$, $\{x_4\}$, $\{x_5\}$, $\{x_6\}$

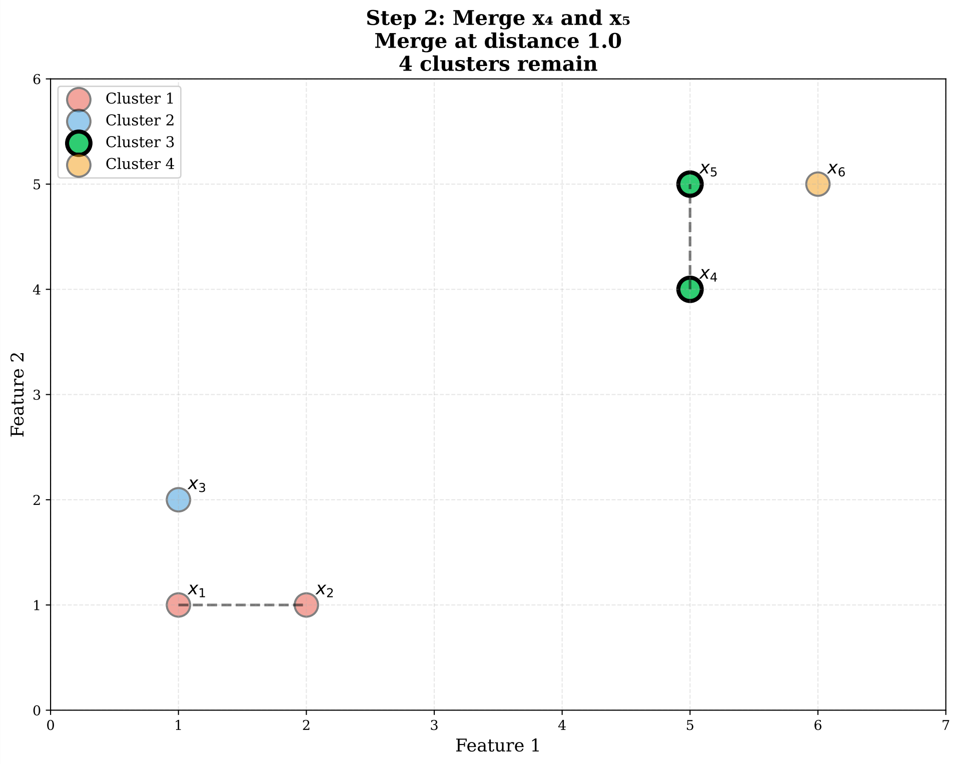
New distance (avg linkage):

$$d(\{x_1, x_2\}, \{x_3\}) = \frac{1.0+1.41}{2} = 1.21$$

Action: Merge x_4 and x_5

Dendrogram height = 1.0

Steps 2-4: Continuing Merges



Step 2: Merge $\{x_1, x_2\}$ with x_3

- Height = 1.21
- 3 clusters left

Step 3: Merge $\{x_4, x_5\}$ with x_6

- Height = 1.27
- 2 clusters left

Step 4: Final merge

- Height ≈ 5.2
- **Large jump** $\longrightarrow$ 2 natural clusters!

Understanding the Dendrogram Y-Axis

Y-axis = Distance at which clusters merge

Understanding the Dendrogram Y-Axis

Y-axis = Distance at which clusters merge

Low values (0-1.5): Points that are close → merge early

- x_1 and x_2 merge at height 1.0 (very close)
- x_4 and x_5 merge at height 1.0 (very close)

Understanding the Dendrogram Y-Axis

Y-axis = Distance at which clusters merge

Low values (0-1.5): Points that are close → merge early

- x_1 and x_2 merge at height 1.0 (very close)
- x_4 and x_5 merge at height 1.0 (very close)

Medium values (1.5-3): Nearby clusters merge

- $\{x_1, x_2\}$ merges with x_3 at height 1.21

Understanding the Dendrogram Y-Axis

Y-axis = Distance at which clusters merge

Low values (0-1.5): Points that are close ⟶ merge early

- x_1 and x_2 merge at height 1.0 (very close)
- x_4 and x_5 merge at height 1.0 (very close)

Medium values (1.5-3): Nearby clusters merge

- $\{x_1, x_2\}$ merges with x_3 at height 1.21

High values (>4): Distant clusters merge

- Left group $\{x_1, x_2, x_3\}$ merges with right group $\{x_4, x_5, x_6\}$ at height 5.2
- **Large jump** ⟶ suggests 2 natural clusters!

Understanding the Dendrogram Y-Axis

Y-axis = Distance at which clusters merge

Low values (0-1.5): Points that are close → merge early

- x_1 and x_2 merge at height 1.0 (very close)
- x_4 and x_5 merge at height 1.0 (very close)

Medium values (1.5-3): Nearby clusters merge

- $\{x_1, x_2\}$ merges with x_3 at height 1.21

High values (>4): Distant clusters merge

- Left group $\{x_1, x_2, x_3\}$ merges with right group $\{x_4, x_5, x_6\}$ at height 5.2
- **Large jump** → suggests 2 natural clusters!

Horizontal cut at height h → Choose K by counting branches below the cut

Linkage Criteria

Problem: Distance between two **clusters**?

Linkage Criteria

Problem: Distance between two **clusters**?

- **Single**: $\min_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} \|\boldsymbol{x} - \boldsymbol{y}\|$ (closest points)
- **Complete**: $\max_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} \|\boldsymbol{x} - \boldsymbol{y}\|$ (farthest points)
- **Average**: $\frac{1}{|C_i|\,|C_j|} \sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{y} \in C_j} \|\boldsymbol{x} - \boldsymbol{y}\|$ (all pairs)

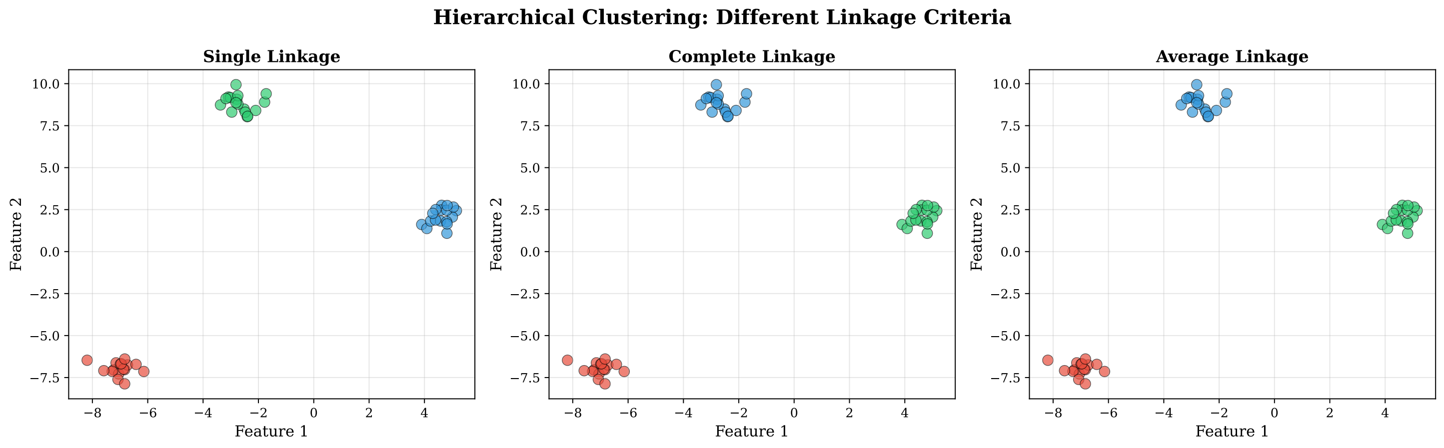
Linkage Criteria

Problem: Distance between two **clusters**?

- **Single**: $\min_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} \|\boldsymbol{x} - \boldsymbol{y}\|$ (closest points)
- **Complete**: $\max_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} \|\boldsymbol{x} - \boldsymbol{y}\|$ (farthest points)
- **Average**: $\frac{1}{|C_i|\,|C_j|} \sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{y} \in C_j} \|\boldsymbol{x} - \boldsymbol{y}\|$ (all pairs)

Choice matters! Different linkages $\longrightarrow$ different dendrograms

Linkage Comparison



Single: chains, **Complete**: compact, **Average**: compromise

Hierarchical vs K-Means

Aspect	**K-Means**	**Hierarchical**
K specified?	Yes	No (from dendrogram)
Shape	Spherical	More flexible
Scalability	Fast: $O(nKdT)$	Slow: $O(n^2 \log n)$
Deterministic?	No	Yes
Best for	Large data, K known	Small data, explore K

Hierarchical vs K-Means

Aspect	K-Means	Hierarchical
K specified?	Yes	No (from dendrogram)
Shape	Spherical	More flexible
Scalability	Fast: $O(nKdT)$	Slow: $O(n^2 \log n)$
Deterministic?	No	Yes
Best for	Large data, K known	Small data, explore K

Recommendation: $n < 10,000$ + hierarchy $\longrightarrow$ Hierarchical, else K-Means

Summary: Key Takeaways

1. K-Means is the workhorse:

- Minimize within-cluster sum of squares
- E-step + M-step, converges to local minimum
- Always use K-Means++ and standardize!

Summary: Key Takeaways

1. K-Means is the workhorse:

- Minimize within-cluster sum of squares
- E-step + M-step, converges to local minimum
- Always use K-Means++ and standardize!

2. Practical workflow:

- Visualize (PCA/t-SNE), standardize, try K-Means++
- Choose K via elbow/silhouette
- If fails: diagnose (scaling? non-convex?)

Summary: Key Takeaways

1. K-Means is the workhorse:

- Minimize within-cluster sum of squares
- E-step + M-step, converges to local minimum
- Always use K-Means++ and standardize!

2. Practical workflow:

- Visualize (PCA/t-SNE), standardize, try K-Means++
- Choose K via elbow/silhouette
- If fails: diagnose (scaling? non-convex?)

3. Alternatives:

- **Hierarchical**: No K, builds tree, slow
- **DBSCAN**: Non-convex, finds outliers
- **GMM**: Soft assignments, elliptical

Questions?

Nipun Batra

IIT Gandhinagar

nipun.batra@iitgn.ac.in

All visualizations generated with Python

Code available in course repository