k-Means Clustering

Victorian England

1854 Cholera Outbreak

A COURT FOR KING CHOLERA.

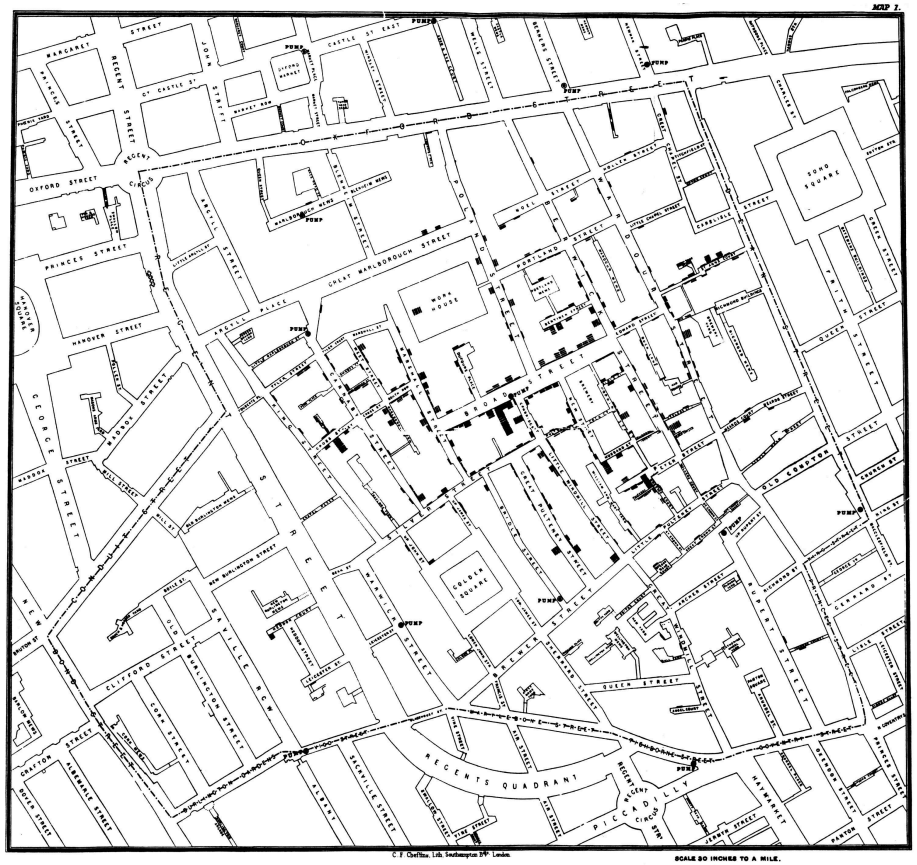
1

1. Public Domain, https://commons.wikimedia.org/w/index.php?curid=680455

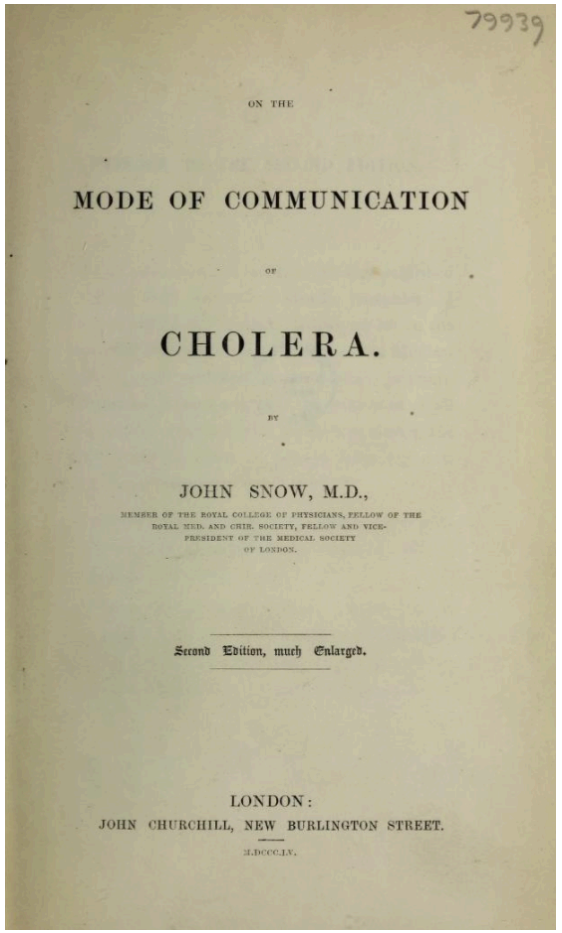
John Snow

John Snow

John’s Snow Map



London cholera outbreak



J. Snow Book

1. By John Snow - Published by C.F. Cheffins, Lith, Southhampton Buildings, London, England, 1854 in Snow, John. On the Mode of Communication of Cholera, 2nd Ed, John Churchill, New Burlington Street, London, England, 1855.

Clustering

Clustering

Clustering Continued

Supervised vs Unsupervised

Supervised methods: Data items have labels, and we want to learn a function that correctly assigns labels to new data items.

Unsupervised methods: Data items do not have labels, and we want to learn a function that extracts important patterns from the data.

Applications of Clustering

- Image Processing
  - Cluster images based on their visual content.
  - Compress images based on color clusters.
- Web Mining
  - Cluster groups of users based on webpage access patterns.
  - Cluster web pages based on their content.
- Bioinformatics
  - Cluster similar proteins together (by structure or function).
  - Cluster cell types (by gene activity).
- And many more …

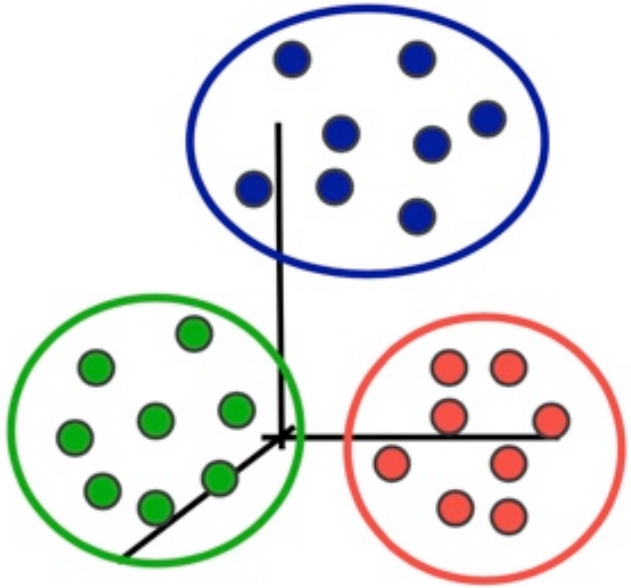
The Clustering Problem

When we apply clustering, what problem are we trying to solve?

The Clustering Problem, continued

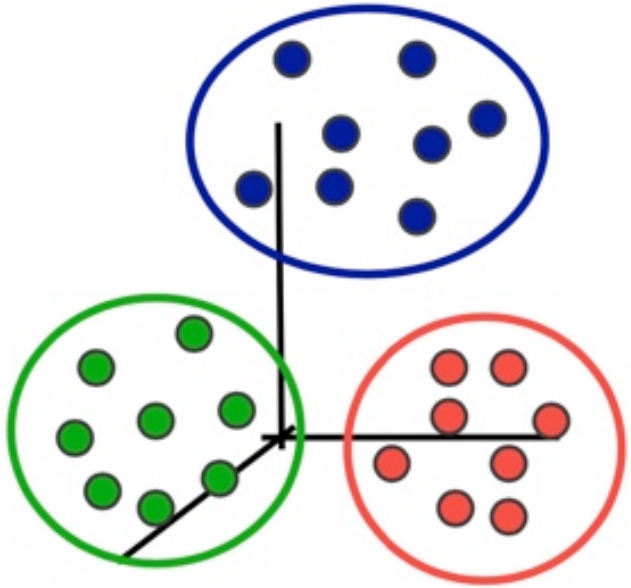
So we want our clustering algorithm to:

- minimize intra-cluster distances.
- maximize inter-cluster distances.



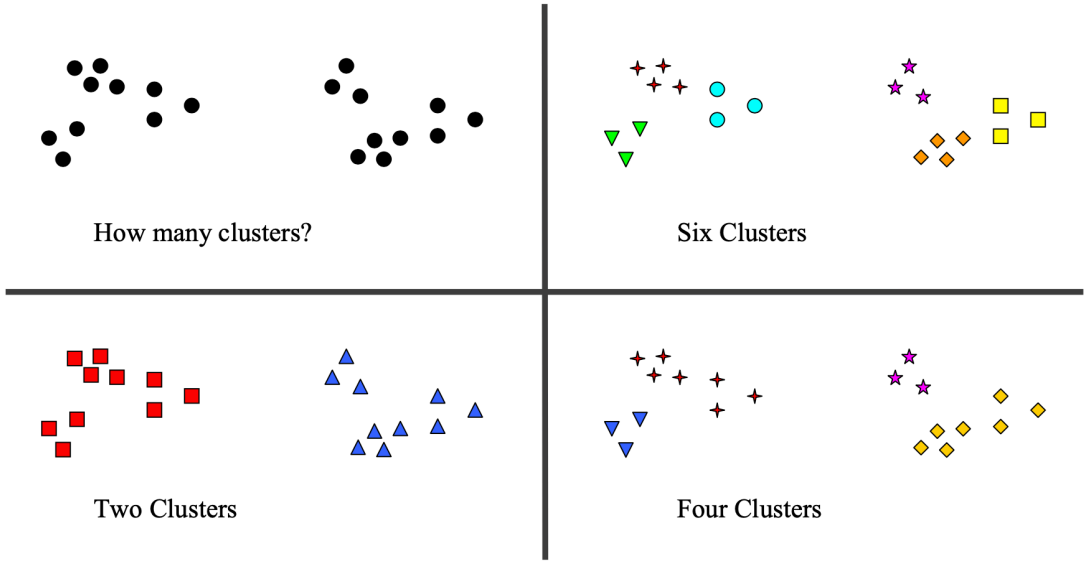
The Clustering Problem, continued

Here are the basic questions we need to ask about clustering:



The Clustering Problem, continued

Now note that even with our more-formal discussion, the criteria for deciding on a “best” clustering can still be ambiguous.



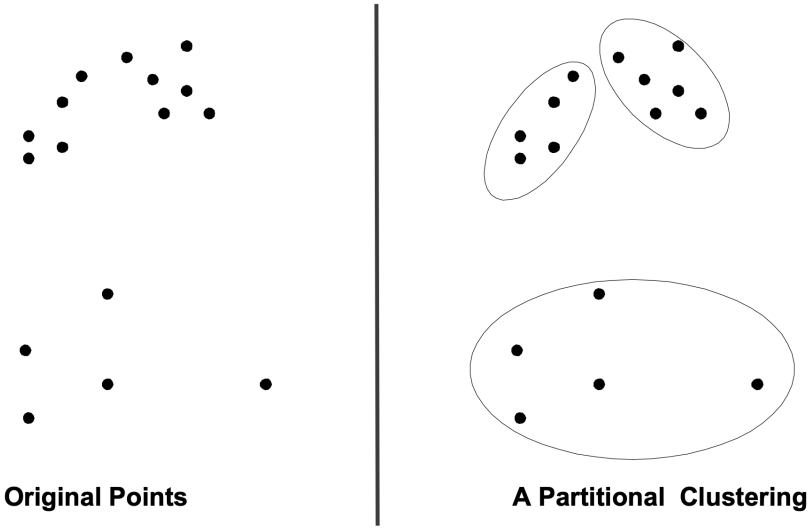
Partitional Clustering

For now, we'll focus on **partitional** clustering.

Points are divided into a set of **non-overlapping** groups:

- Each object belongs to **one**, and only one, cluster. (mutually exclusive)
- The set of clusters covers all the objects. (exhaustive)

We are going to assume for now that the number of clusters, k, is given in advance.



The k-means Algorithm

Assumptions

Now, we are ready to state our first formalization of the clustering problem.

We will assume that

- data items are represented by points in d-dimensional space, $\mathbb{R}^d$, i.e., has d features,
- the number of points, N, is given, and
- the number of clusters K is given.

Minimizing a Cost Function

Find K disjoint clusters C_k, each described by points $c_1, \dots, c_K$ (called *centers*, *centroids*, or *means*) that minimizes

$$\underset{C_1,\dots,C_K}{\arg\min} \sum_{k=1}^{K} \sum_{x \in C_k} \|x - c_k\|^2.$$

- The *Within-Cluster Sum of Squares* (WCSS) or the *Inertia* of the clustering.

See Norms in Distances and Timeseries for a refresher.

We now have a formal definition of a clustering.

This is not the only definition possible, but it is an intuitive and simple one.

How hard is it to solve this problem?

- $k = 1$ and $k = n$ are easy special cases. Why?
- But, this problem is **NP-hard** if the dimension of the data is at least 2.
  - We don't expect that there is any exact, efficient algorithm in general.

Nonetheless, there is a simple algorithm that works quite well in practice!

Aside: k-means as NP-hard

For context, **NP-hard** is a term used in the study of computational complexity.

Problems in **P** are those that can be solved in polynomial time, e.g. for n items, the time to solve the problem is $O(n^2)$ or $O(n^3)$.

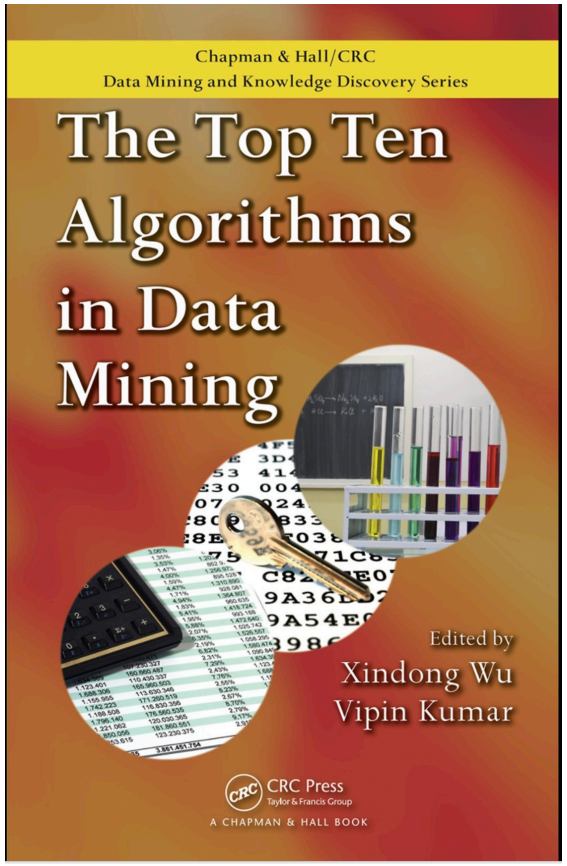
Problems in **NP** are those that can be *verified* in polynomial time but not necessarily *solved* in polynomial time.

NP-hard problems are those that are at least as hard as the hardest problems in **NP**, not necessarily verifiable in polynomial time.

NP-complete problems are those that are both NP-hard and in NP, e.g. hardest problems in NP and verifiable in polynomial time.

You can prove that the k-means problem is NP-hard by finding a reduction from a known NP-hard problem such as the **Partition Problem**.

The k-means Algorithm



- There is a "classic" algorithm for this problem.
- It was voted among the **top-10 algorithms** in data mining![1]
- It is such a good idea that it has been independently discovered multiple times.
- It was first discovered by Lloyd in 1957, so it is often called Lloyd's algorithm.
- It is called the "k-means algorithm."
- Not to be confused with the k-means problem of which this is a heuristic solution.

The other from the top 10 were SVM, Apriori, EM, PageRank, AdaBoost, kNN, Naive Bayes, and CART.

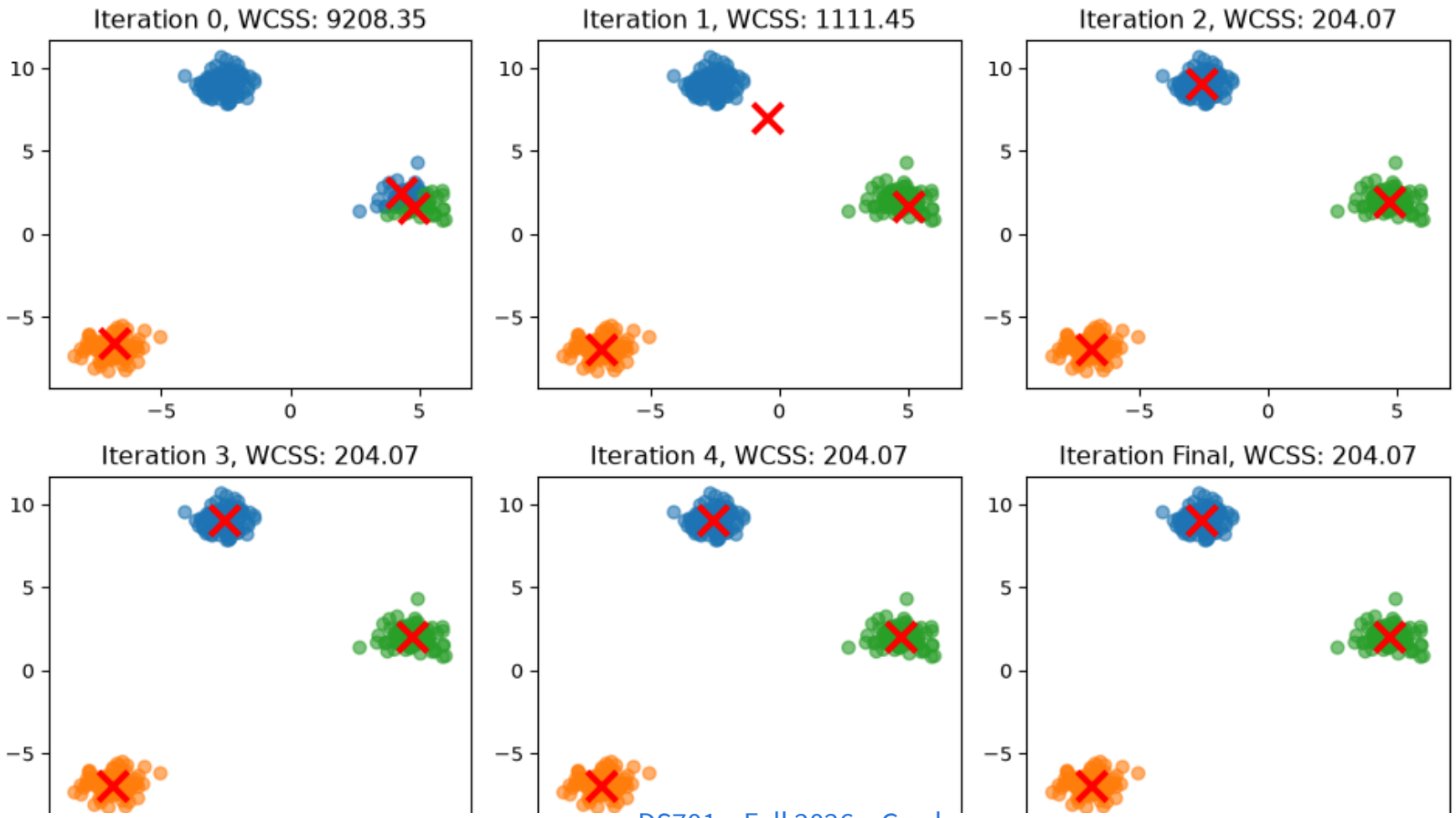
1. As determined at the 2006 IEEE International Conference on Data Mining

The k-means algorithm

1. Pick K cluster centers $\{c_1, \dots, c_K\}$.
    - These can be randomly chosen data points, or by some other method such as k-means++
2. For each j, define the cluster C_j as the set of points in X that are closest to center c_j.
    - Nearer to c_j than to any other center.
3. For each j, update c_j to be the center of mass of cluster C_j.
    - In other words, c_j is the mean of the vectors in C_j.
4. Repeat (i.e., go to Step 2) until convergence.
    - Either the cluster centers change below a threshold, or inertia changes below a threshold or a maximum number of iterations is reached.

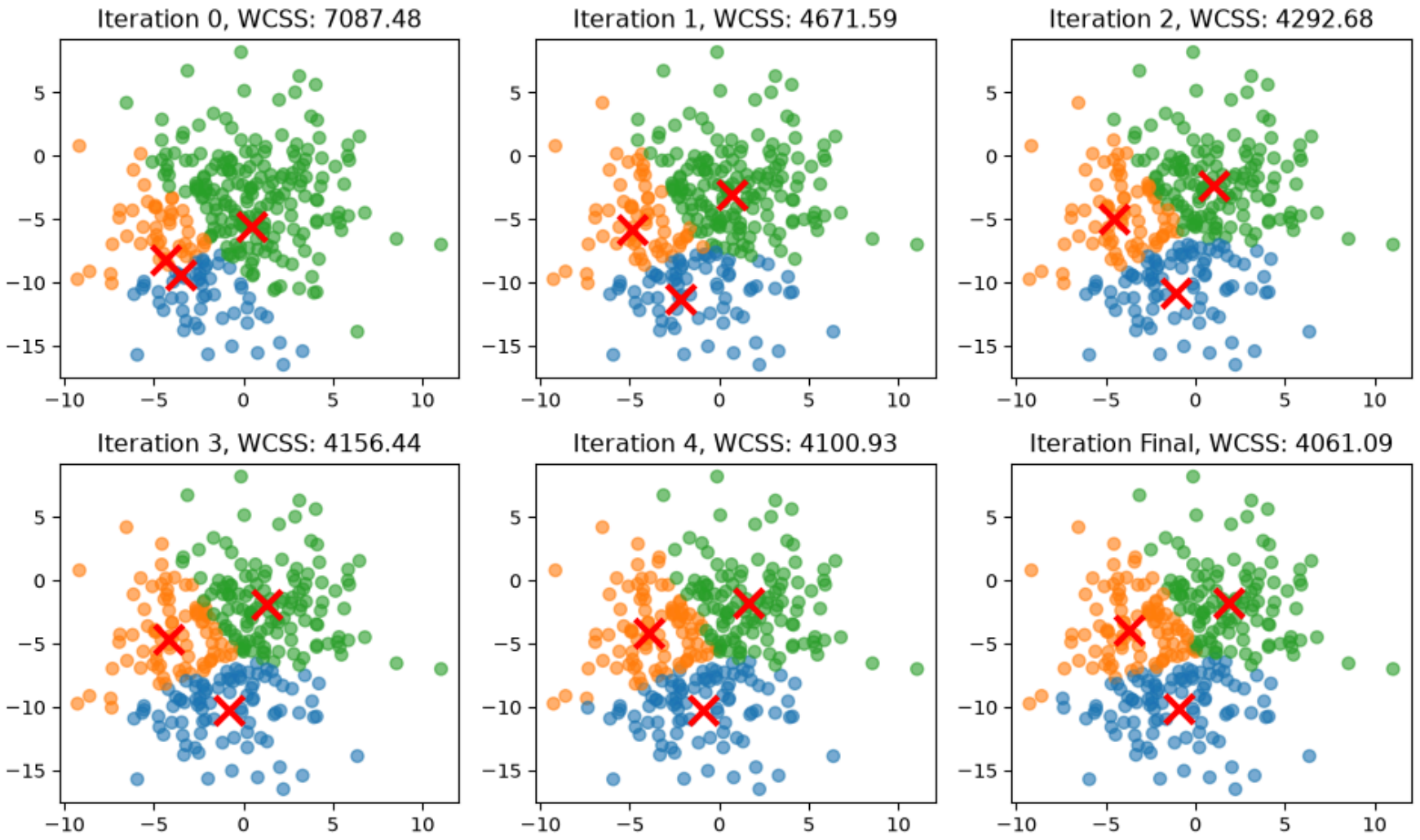
Let’s see this in practice with well separated clusters and also look at the Within-Cluster Sum of Square (WCSS).

▶ Code



Let’s see this in practice with overlapping clusters and also look at the WCSS.

▶ Code



Limitations of k-means

As you can see, k-means can work very well.

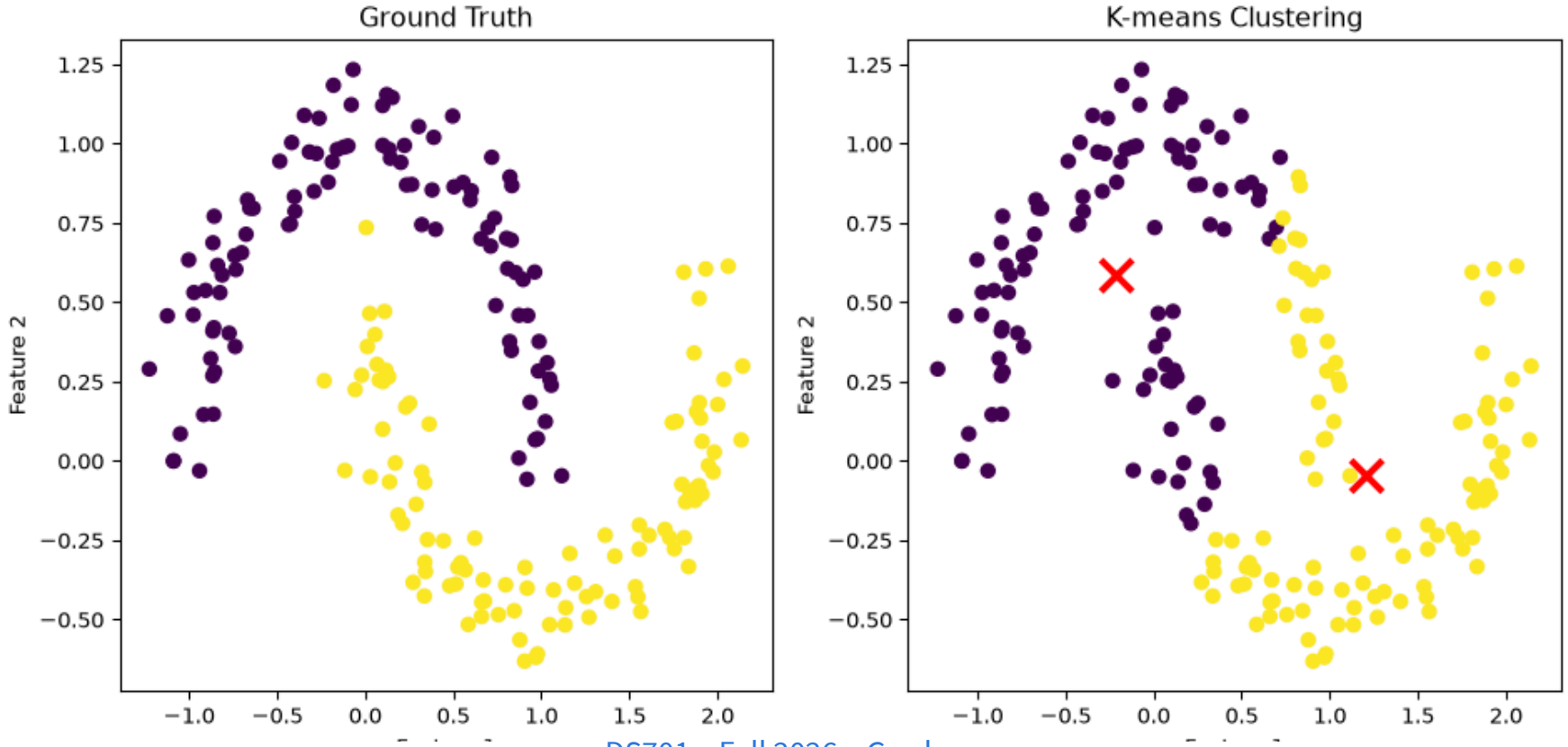
However, we don't have any guarantees on the performance of k-means.

In particular, there are various settings in which k-means can fail to do a good job.

Limitation: non-spherical clusters

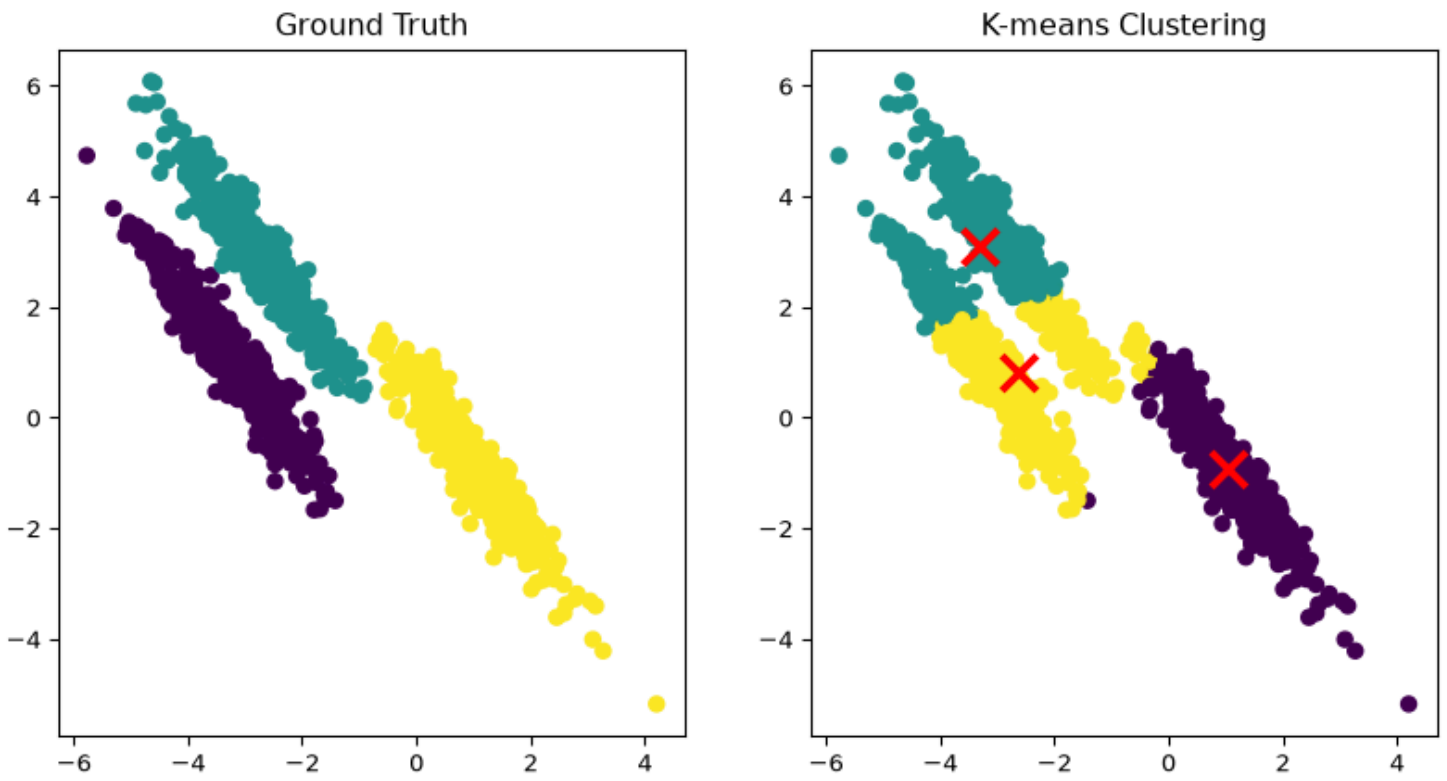
Because each point is assigned to its closest center, the points in a cluster are implicitly assumed to be arranged in a sphere around the center.

▶ Code



Limitation: anisotropic clusters

▶ Code

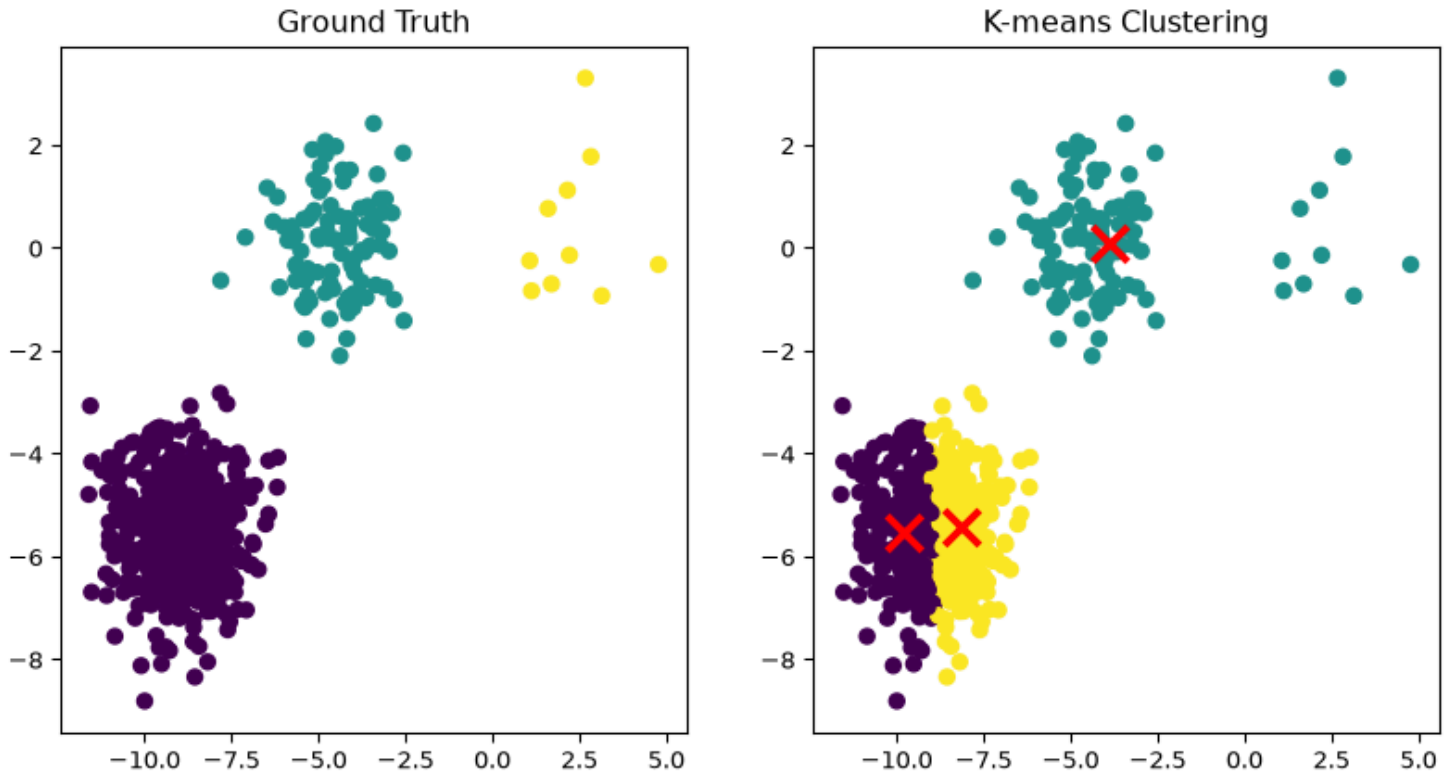


K-means clustering on a dataset with anisotropic gaussian clusters.

Limitation: unequal-sized clusters

For the same reason, k-means tends to try to balance the sizes of the clusters.

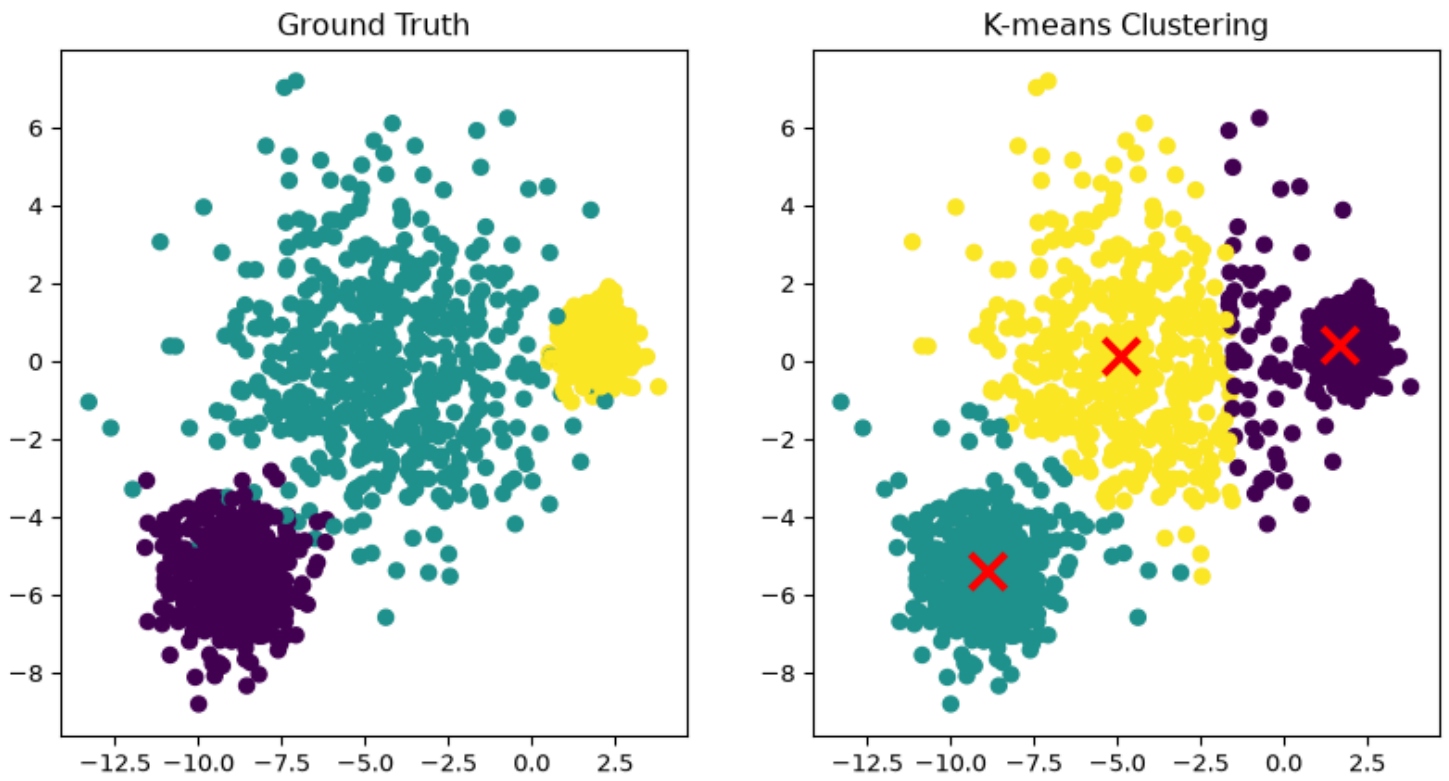
▶ Code



K-means clustering on a dataset with unequal-sized clusters.

Limitation: unequal variance clusters

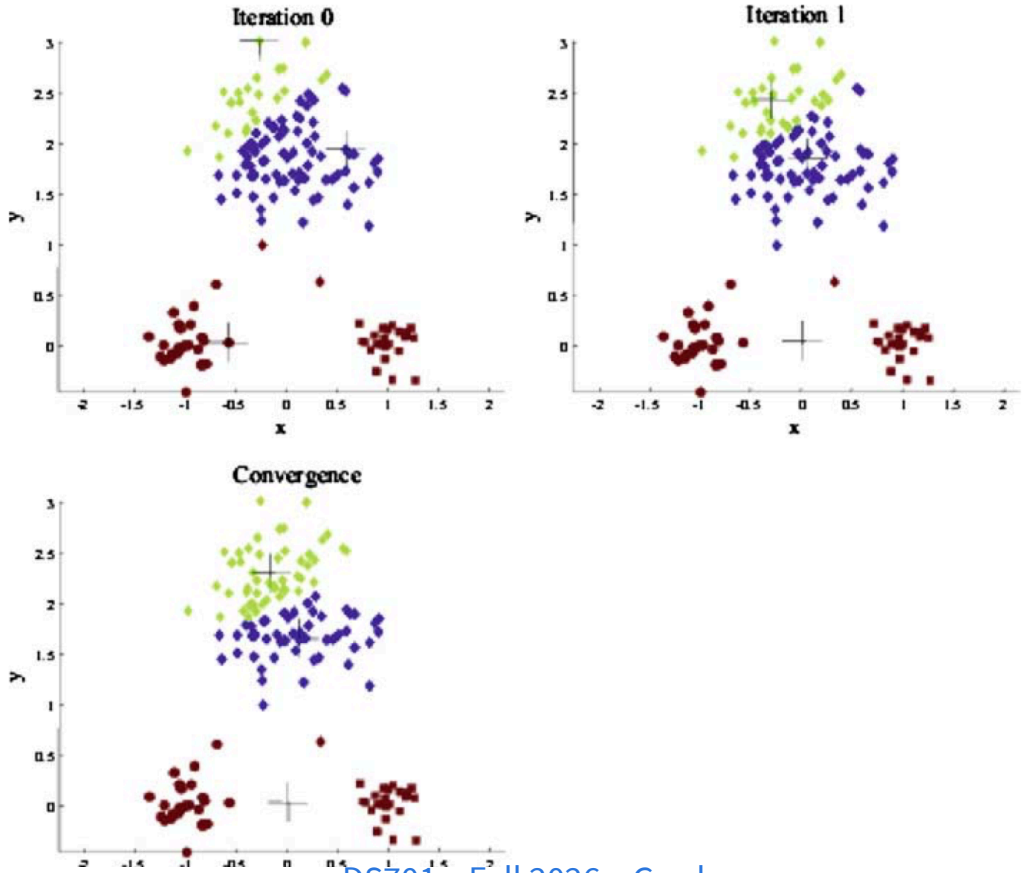
▶ Code



K-means clustering on a dataset with unequal variance clusters.

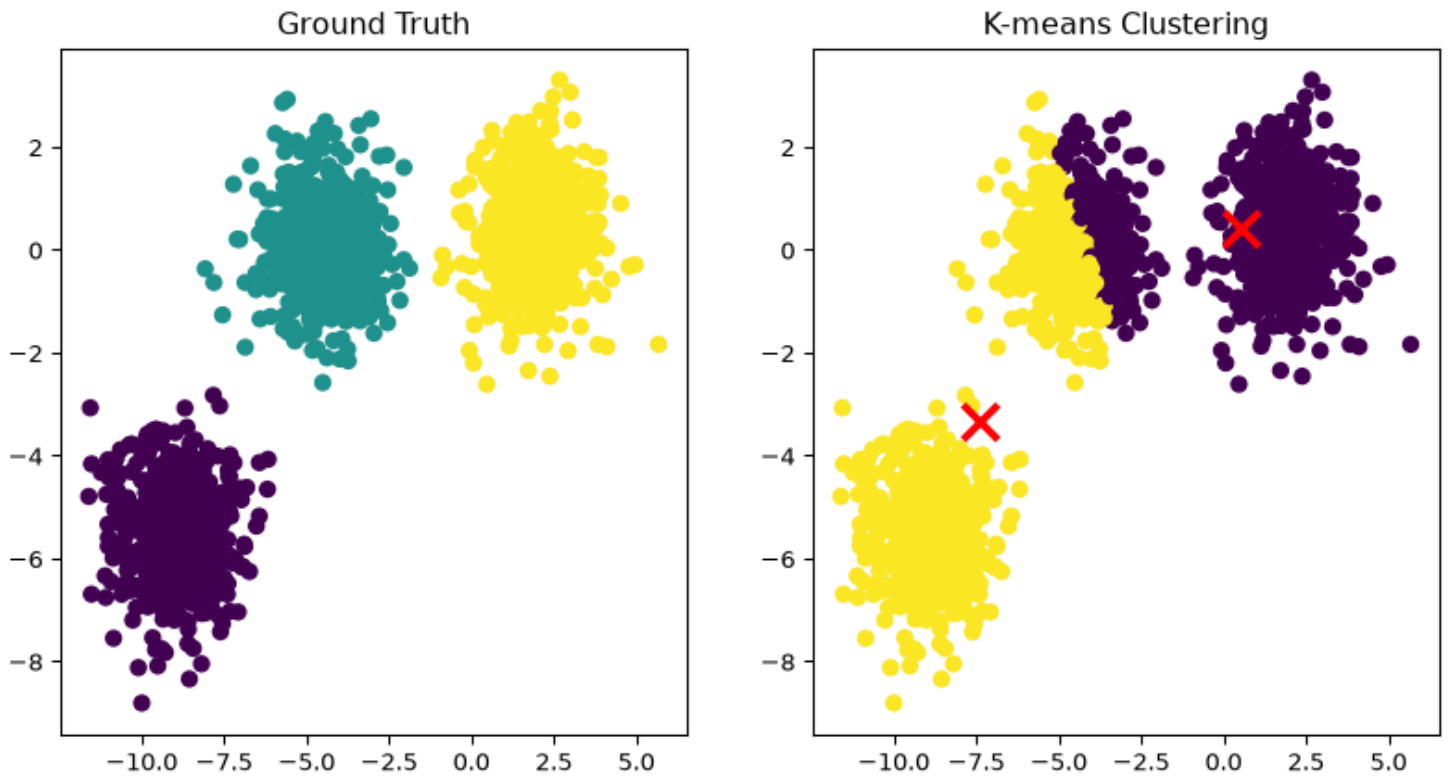
Limitation: sensitive to starting cluster centers

If the initial guess (Step 1) is a bad one, k-means may get "stuck" in a bad solution.



Limitation: sensitive to number of clusters

▶ Code



Choosing a Good Initialization

- How can we avoid the kind of bad initialization we just saw?
- A good strategy is to pick points that are distant to each other.
- This strategy is called "k-means++".
- It works very well in practice, and the `scikit-learn` implementation uses it by default.
- We will explore it in more detail in the next lecture.

Choosing the right k

Given some K, the k-means algorithm "learns" the cluster centers.

Feature Scales

Finally, given the tendency of k-means to look for spherical clusters, we should consider the scales of the various features.

In fact, in general when constructing or selecting a distance metric, one needs to think carefully about the scale of the features being used.

Unscaled Features

For example, consider the case where we are clustering people based on their age, income, and gender.

We might use age in years, income in dollars, and assign gender to the values $\{0, 1\}$.

Thus, the following records:

- Joe Smith, age 27, income USD 75,000, male
- Eve Jones, age 45, income USD 42,000, female

Would be encoded in feature space as:

$$\begin{bmatrix} 27 \\ 75000 \\ 0 \end{bmatrix}, \begin{bmatrix} 45 \\ 42000 \\ 1 \end{bmatrix}$$

Unscaled Features, Continued

What would happen if we used Euclidean distance as our dissimilarity metric in this feature space?

(This is what k-means uses.)

Feature Scaling

The basic idea is to rescale each feature separately, so that its range of values is about the same as all other features.

For example, one may choose to:

- shift each feature independently by subtracting the mean over all observed values.
  - This means that each feature is now centered on zero.
- Then rescale each feature so that the standard deviation overall observed values is 1.
  - This means that the feature will have about the same range of values as all the others.

k-means Clustering Examples

Example 1: Customer Segmentation

Customer Segmentation

Another powerful application of k-means clustering is **customer segmentation** in e-commerce.

E-commerce Customer Data

Let's synthesize e-commerce data to segment customers based on their purchasing behavior.

High-value, frequent buyers; Moderate buyers; Bargain hunters; Occasional buyers.

Segment	Percentage	Annual Spending	Average Order Value	Total Orders
High-value	20%	120 ± 20	15 ± 3	450 ± 80
Moderate	40%	60 ± 15	8 ± 2	250 ± 50
Bargain	25%	25 ± 8	12 ± 4	150 ± 30
Occasional	15%	15 ± 5	3 ± 1	300 ± 60

E-commerce Customer Data

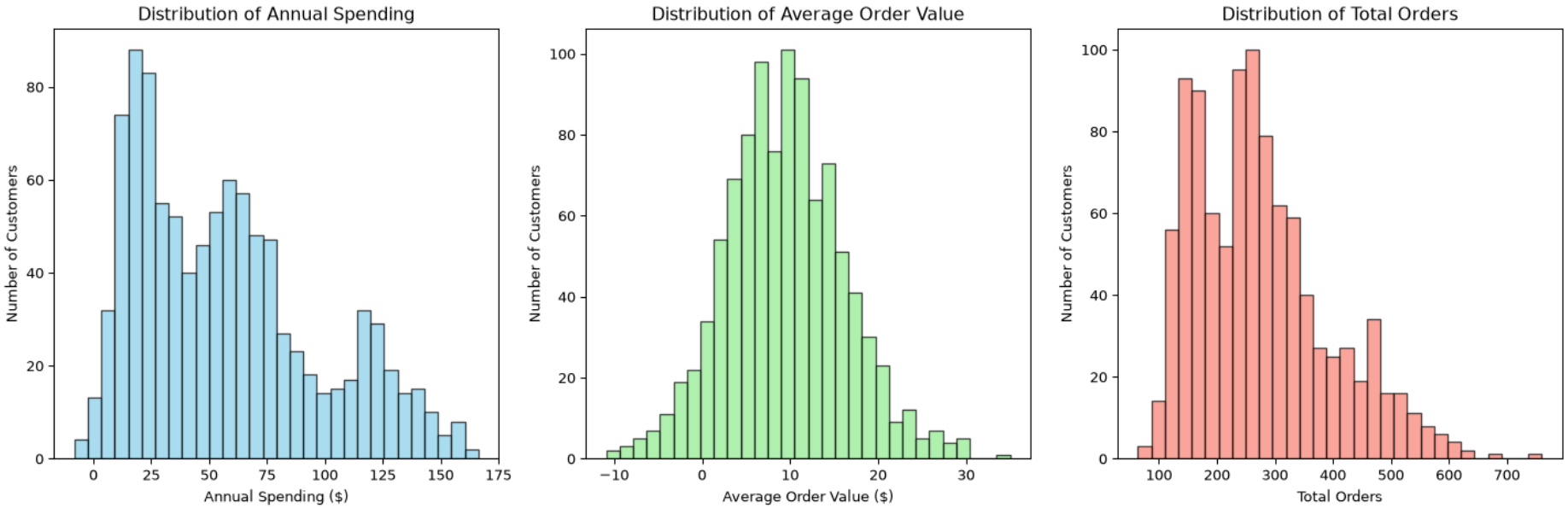
▶ Code

```
E-commerce Customer Data Head:
   Annual_Spending  Avg_Order_Value  Total_Orders  Customer_ID
0             0.27             1.67        325.07            1
1            27.74             4.78        127.63            2
2            43.86             6.75        262.73            3
3           132.78             9.23        345.36            4
4            83.19             9.52        277.84            5


E-commerce Customer Data Summary    :
       Annual_Spending  Avg_Order_Value  Total_Orders  Customer_ID
count          1000.00          1000.00       1000.00      1000.00
mean             57.12             9.39        275.20       500.50
std              39.33             6.80        116.85       288.82
min              -8.21           -10.90         63.74         1.00
25%              23.26             4.87        178.72       250.75
50%              50.96             9.26        257.75       500.50
75%              78.22            13.76        337.32       750.25
max             166.33            35.01        760.47      1000.00
```

Understanding Customer Features

▶ Code



Distribution of customer features

> Can you see clusters?

Feature Scaling is Critical

Before clustering, we need to scale our features since they have very different ranges:

▶ Code

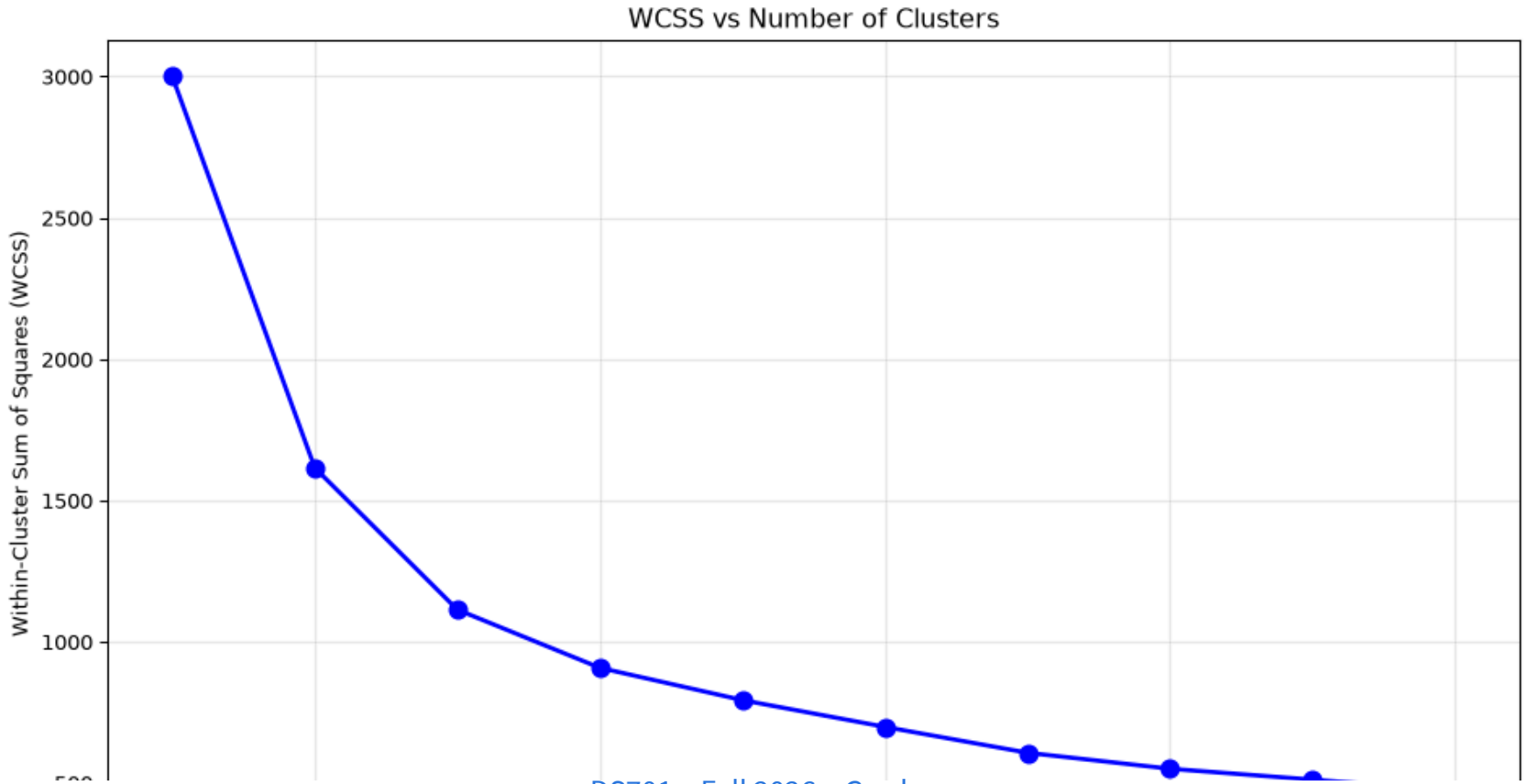
```
Original feature ranges:
Annual Spending: $-8 - $166
Avg Order Value: $-11 - $35
Total Orders: 64 - 760

Scaled feature ranges:
Annual Spending: -1.66 - 2.78
Avg Order Value: -2.99 - 3.77
Total Orders: -1.81 - 4.16
```

Searching Cluster Numbers

What cluster number should we pick?

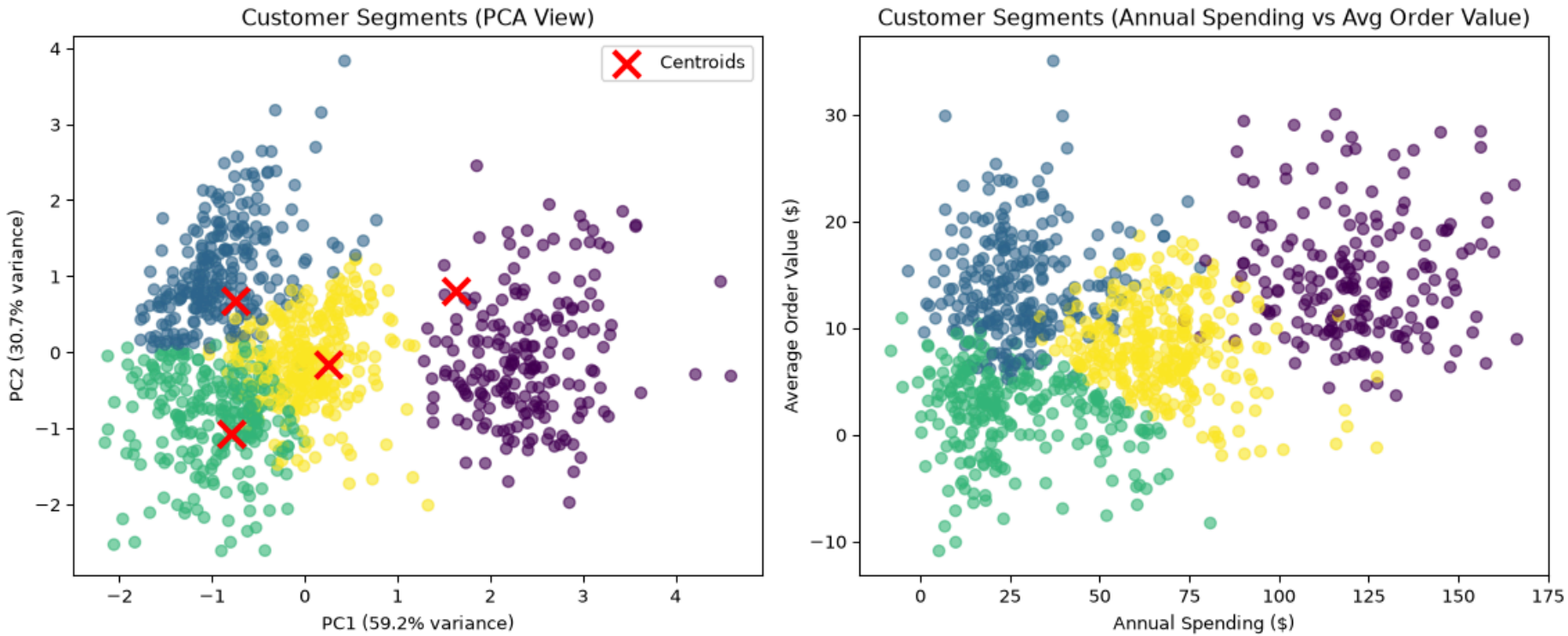
▶ Code



Customer Segmentation Results

Visualize in 2D using PCA which we'll cover later.

▶ Code



K-means customer segmentation results

Customer Segment Profiles

▶ Code

```
Customer Segment Profiles:
============================================================

Segment 0 (193 customers):
  • Average Annual Spending: $121
  • Average Order Value: $15
  • Average Total Orders: 458.9

Segment 1 (251 customers):
  • Average Annual Spending: $28
  • Average Order Value: $14
  • Average Total Orders: 161.1

Segment 2 (255 customers):
  • Average Annual Spending: $26
  • Average Order Value: $2
  • Average Total Orders: 268.5

Segment 3 (301 customers):
  • Average Annual Spending: $67
```

Segment	Percentage	Annual Spending	Average Order Value	Total Orders
High-value	20%	120 ± 20	15 ± 3	450 ± 80

Example 2: Plant Classification Based on Morphology

Plant Species Classification with Iris Data

Let’s explore another real-world application using the famous **Iris dataset** from scikit-learn.

Loading and Exploring the Iris Dataset

▶ Code

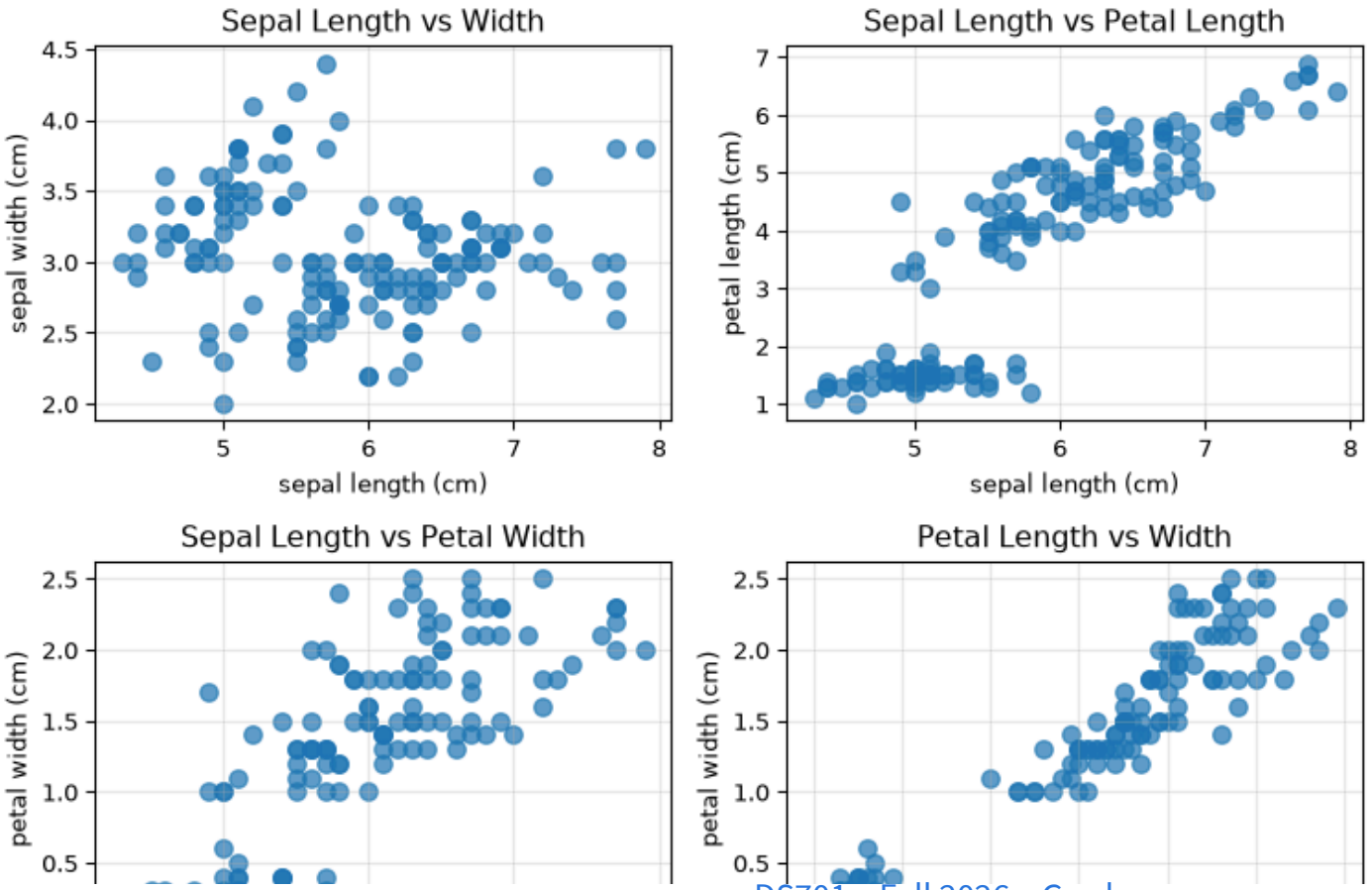
```
Iris Dataset Information:
=========================================
Number of samples: 150
Number of features: 4
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']
Number of species: 3

First 5 samples:
Sepal Length Sepal Width Petal Length Petal Width Species
-----------------------------------------------------------
5.1          3.5         1.4          0.2         setosa
4.9          3.0         1.4          0.2         setosa
4.7          3.2         1.3          0.2         setosa
4.6          3.1         1.5          0.2         setosa
5.0          3.6         1.4          0.2         setosa
```

Visualizing Iris Data in 2D

Let’s compare features in pairs. Do you see clusters?

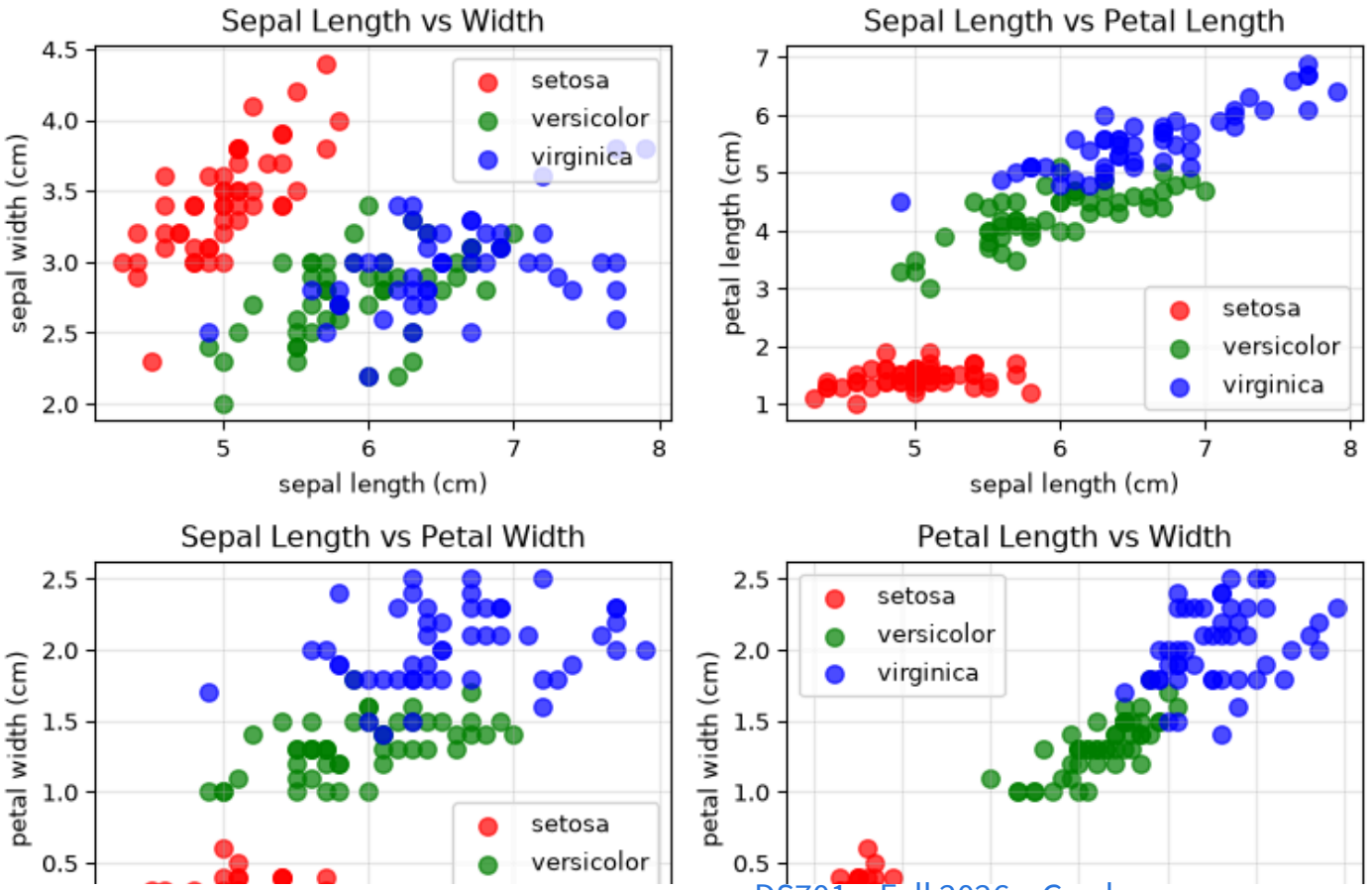
▶ Code



Visualizing Iris Data in 2D

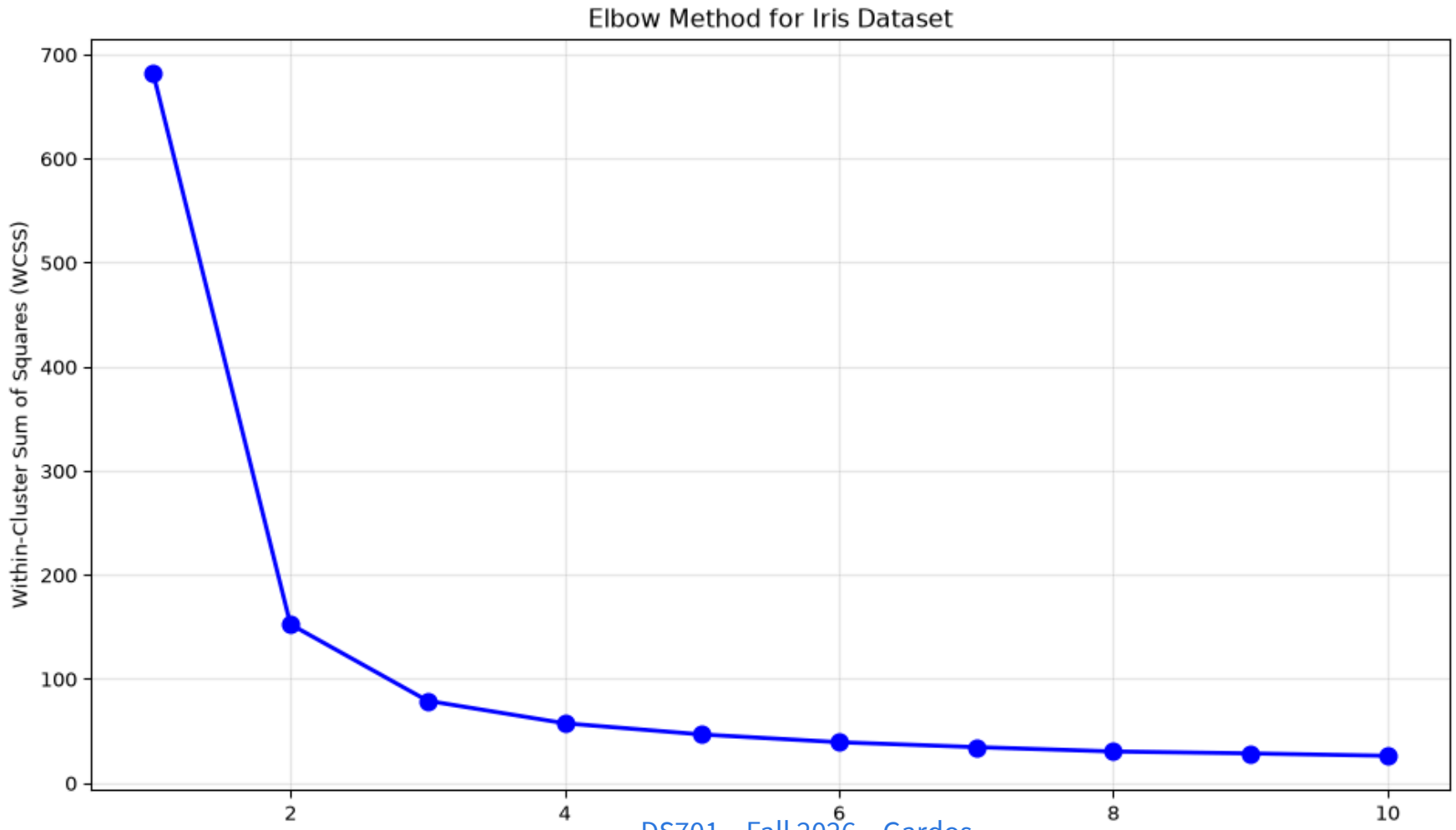
Feature pairs with ground truth labels.

▶ Code



Try different cluster numbers

▶ Code

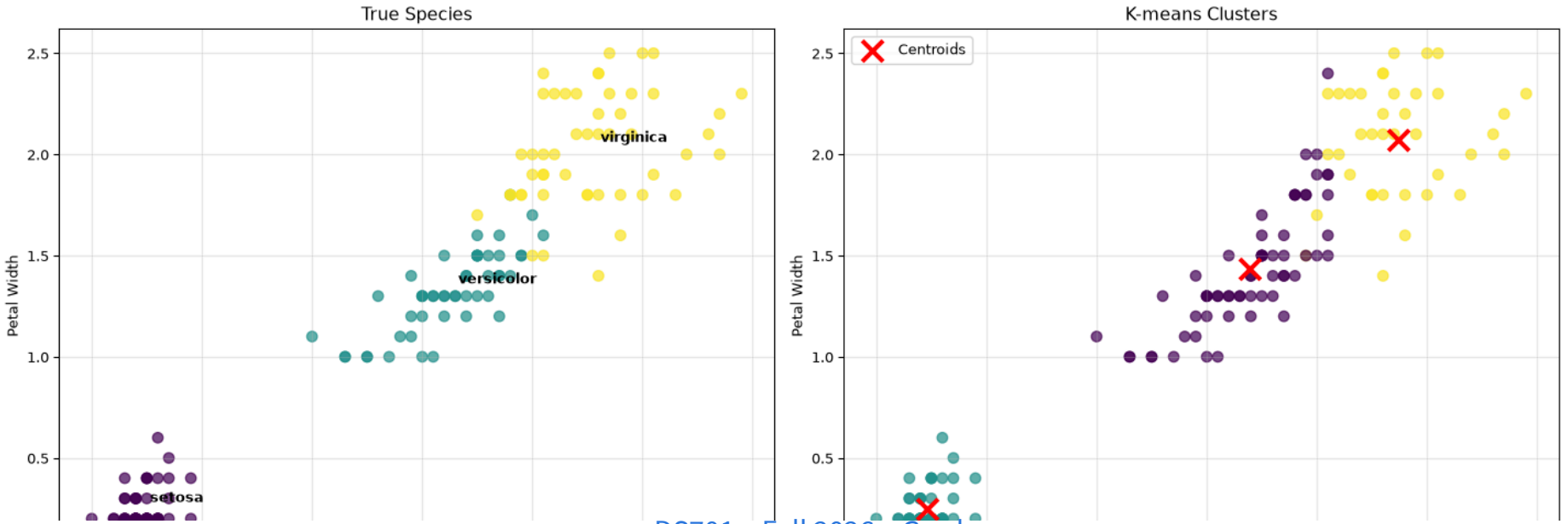


K-means Clustering on Iris Data

Results for $k = 3$ clusters.

▶ Code

```
Confusion Matrix:
Rows: True species, Columns: Predicted clusters
            Cluster 0  Cluster 1  Cluster 2
setosa              0         50          0
versicolor         48          0          2
virginica          14          0         36
```



Analysis

▶ Code

```
Cluster Analysis:
================================================

Cluster 0:
  • Size: 62 samples
  • Dominant species: versicolor (77.4%)
  • Average petal length: 4.4 cm
  • Average petal width: 1.4 cm
  • Average sepal length: 5.9 cm
  • Average sepal width: 2.7 cm

Cluster 1:
  • Size: 50 samples
  • Dominant species: setosa (100.0%)
  • Average petal length: 1.5 cm
  • Average petal width: 0.2 cm
  • Average sepal length: 5.0 cm
  • Average sepal width: 3.4 cm

Cluster 2:
```

Key Insights:

- K-means successfully identified the three iris species
- Setosa is perfectly separated from the other species
- Versicolor and Virginica show some overlap, which is realistic
- Petal measurements are more discriminative than sepal measurements
- Overall clustering quality pretty good!

Why K-means Works Well on Iris Data

- **Well-separated clusters**: The three iris species have distinct characteristics
- **Spherical cluster shapes**: Each species forms roughly spherical groups in feature space
- **Appropriate feature scaling**: All measurements are in the same units (centimeters)
- **A Priori knowledge**: We know the true number of species (cluster number)

Example 3: k-means image compression

k-means image compression

Here is a simple example of how k-means can be used to reduce color space and compress data.

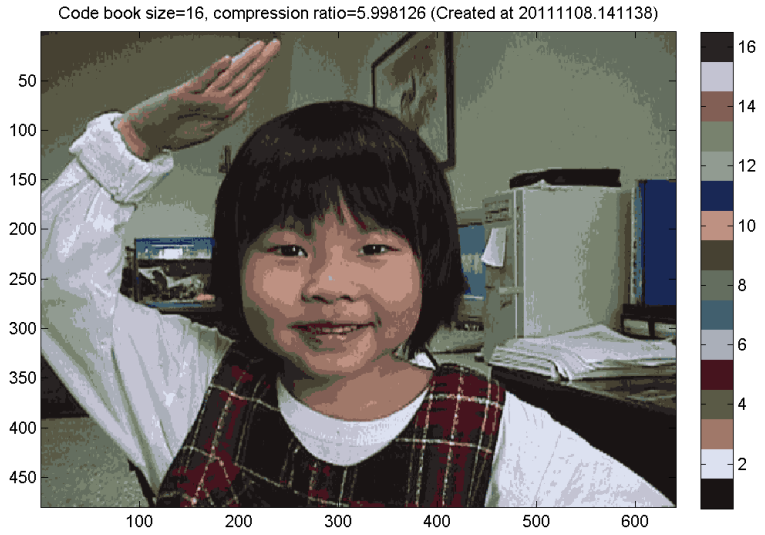
Consider the following image.

- Each color in the image is represented by an integer.
- Typically we might use 24 bits for each integer (8 bits for R, G, and B).

Example, Continued

Now find $k = 16$ clusters of pixels in three dimensional (R, G, B) space and replace each pixel by its cluster center.

Because there are 16 centroids, we can represent by a 4-bit mapping for a compression ratio of $24/4 = 6\times$.

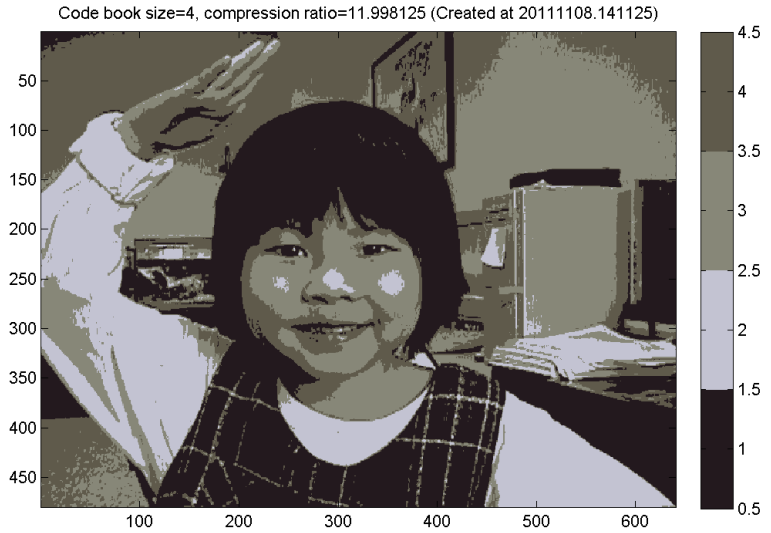


Example, Continued

Here we cluster into 8 groups (3 bits) for a compression ratio $24/3 = 8\times$.

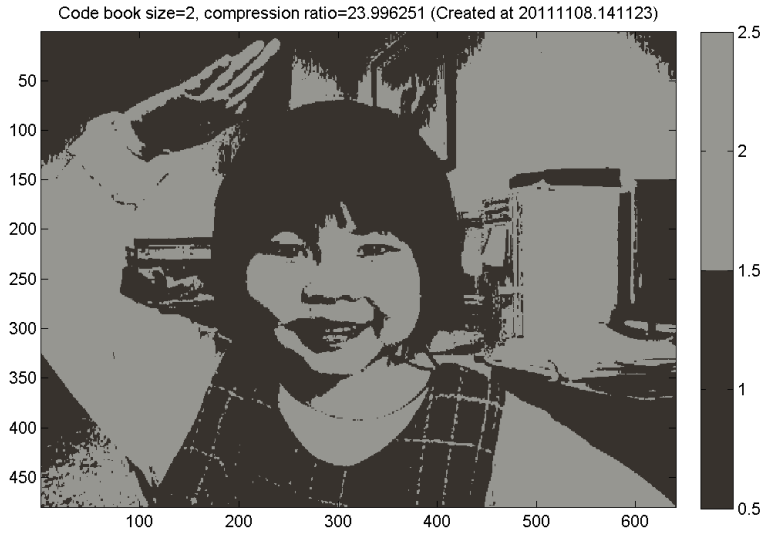
Example, Continued

Here we cluster into 4 groups (2 bits) for a compression ratio around $24/2 = 12\times$.



Example, Continued

Finally, we use 1 bit (2 color groups) for a compression ratio of $24\times$.



In-Class Activity: k-Means Clustering

Time: 10 minutes
Materials: Pencil and paper
Objective: Understand the k-means algorithm through 1D visualization

Activity: “Clustering 1D Data”

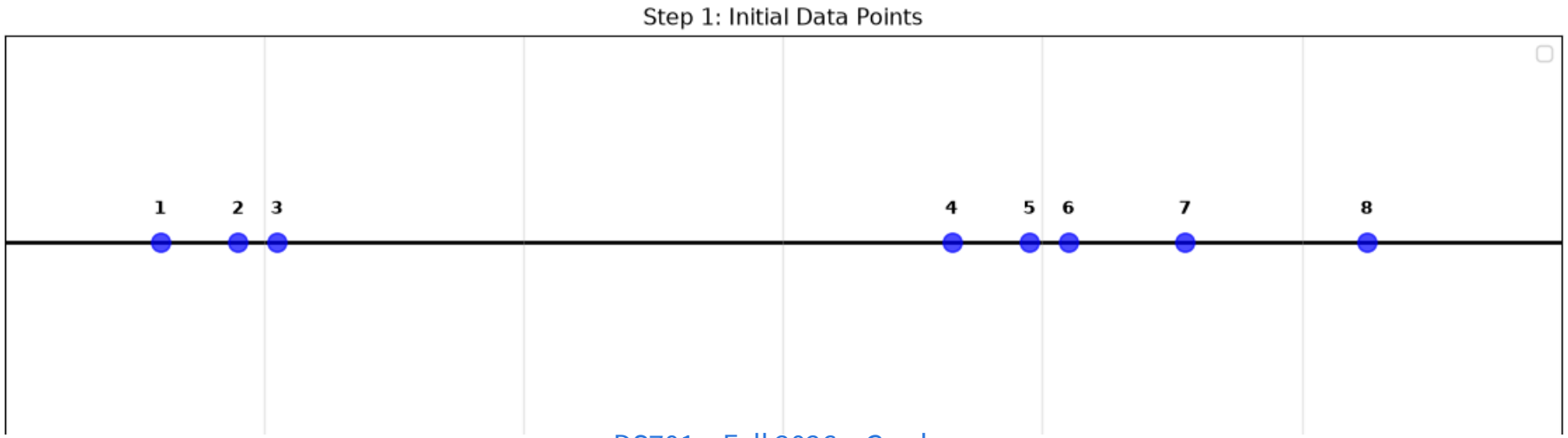
The Data

We have 8 data points on a number line that form 2 natural clusters:

Data Points: 1.2, 1.8, 2.1, 7.3, 7.9, 8.2, 9.1, 10.5

▶ Code

```
/var/folders/j_/hxxgy5dd7655k_416s6t9kgc0000gq/T/ipykernel_5415/3134341457.py:23: UserWarning: No artists with
labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend()
is called with no argument.
  ax.legend(loc='upper right')
```



The Steps

1. Pick K cluster centers $\{c_1, \dots, c_K\}$.
2. For each j, define the cluster C_j as the set of points in X that are closest to center c_j.
3. For each j, update c_j to be the center of mass of cluster C_j.
4. Repeat (i.e., go to Step 2) until convergence.

The Steps

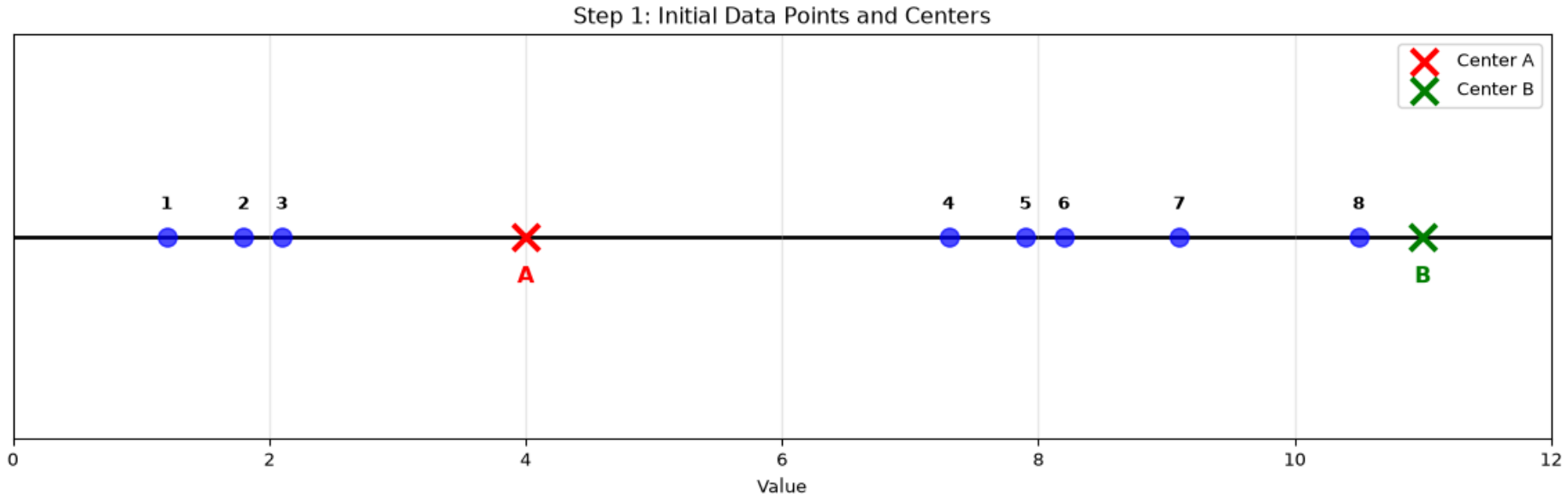
1. **Pick K cluster centers $\{c_1, \dots, c_K\}$.**
2. For each j, define the cluster C_j as the set of points in X that are closest to center c_j.
3. For each j, update c_j to be the center of mass of cluster C_j.
4. Repeat (i.e., go to Step 2) until convergence.

Step 1: Cluster Centers Initialization

Assume our randomly chosen initial centers are:

- Center A = 4.0, Center B = 11.0

▶ Code



The Steps

1. Pick K cluster centers $\{c_1, \dots, c_K\}$.
2. **For each j, define the cluster C_j as the set of points in X that are closest to center c_j.**
3. For each j, update c_j to be the center of mass of cluster C_j.
4. Repeat (i.e., go to Step 2) until convergence.

Step 2: Assignment (2 minutes)

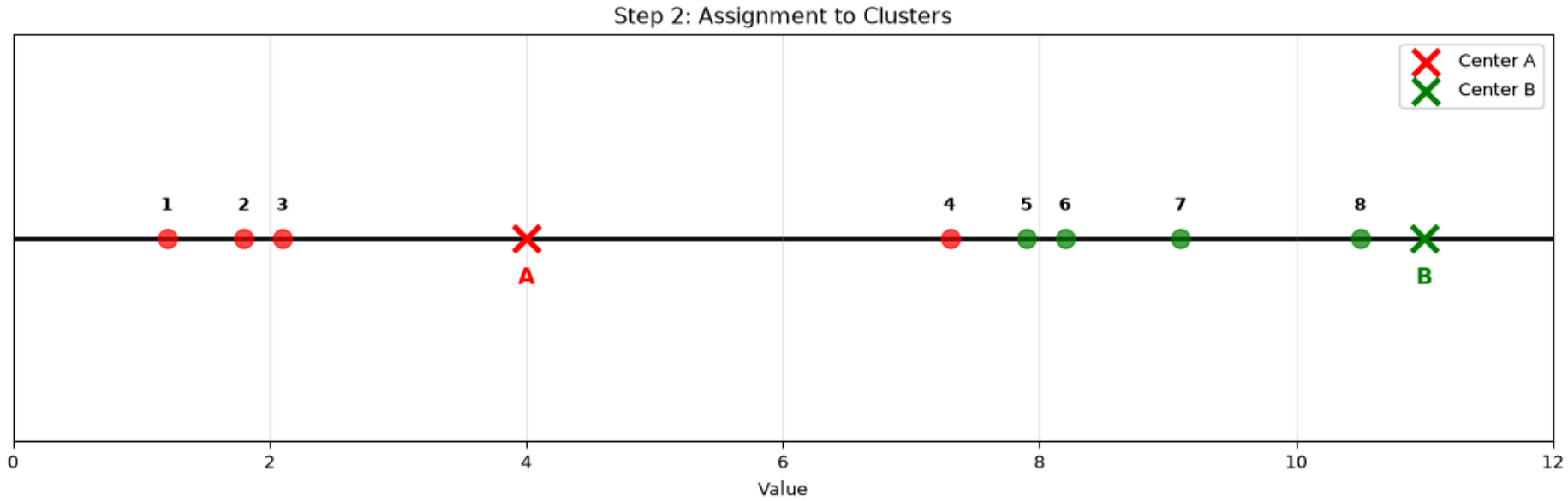
For each data point, calculate the distance to both centers and assign it to the closer one.

Complete the table:

Point	Value	Distance to A (4.0)	Distance to B (11.0)	Assigned to
1	1.2	2.8	9.8	A
2	1.8			
3	2.1			
4	7.3			
5	7.9			
6	8.2			
7	9.1			
8	10.5			

Assignment Visualization

▶ Code



Assignment visualization with cluster colors

The Steps

1. Pick K cluster centers $\{c_1, \dots, c_K\}$.
2. For each j, define the cluster C_j as the set of points in X that are closest to center c_j.
3. **For each j, update c_j to be the center of mass of cluster C_j.**
4. Repeat (i.e., go to Step 2) until convergence.

Step 3: Update Centers (2 minutes)

Calculate the new centers based on the assigned points.

Cluster A (points: 1.2, 1.8, 2.1, 7.3):

- New center A =

Cluster B (points: 7.9, 8.2, 9.1, 10.5):

- New center B =

Step 3: Update Centers

Calculate the new centers based on the assigned points.

Cluster A (points: 1.2, 1.8, 2.1, 7.3):

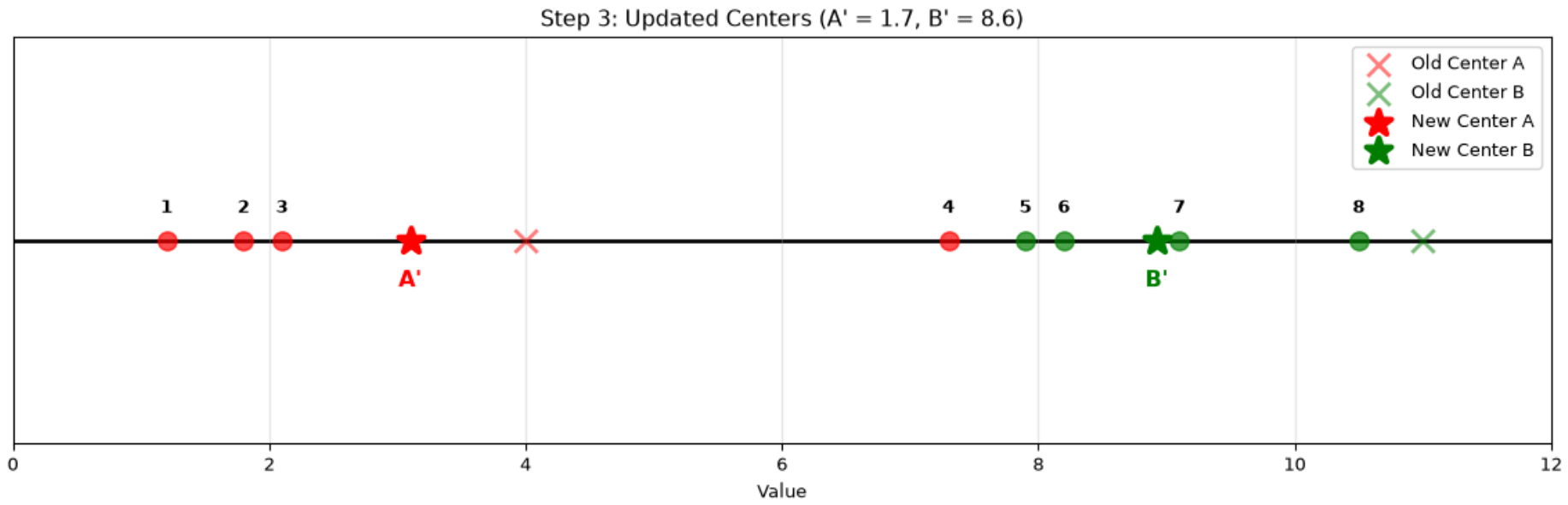
- New center A = (1.2 + 1.8 + 2.1 + 7.3) ÷ 4 = 3.1

Cluster B (points: 7.9, 8.2, 9.1, 10.5):

- New center B = (7.9 + 8.2 + 9.1 + 10.5) ÷ 4 = 8.9

Step 3: Update Centers Visualization

▶ Code




```
New Center A: 3.1
New Center B: 8.9
```

The Steps

1. Pick K cluster centers $\{c_1, \dots, c_K\}$.
2. For each j, define the cluster C_j as the set of points in X that are closest to center c_j.
3. For each j, update c_j to be the center of mass of cluster C_j.
4. **Repeat (i.e., go to Step 2) until convergence.**

The Steps

1. Pick K cluster centers $\{c_1, \dots, c_K\}$.
2. **For each j, define the cluster C_j as the set of points in X that are closest to center c_j.**
3. For each j, update c_j to be the center of mass of cluster C_j.
4. Repeat (i.e., go to Step 2) until convergence.

Step 4: Second Iteration Cluster Assignment (2 minutes)

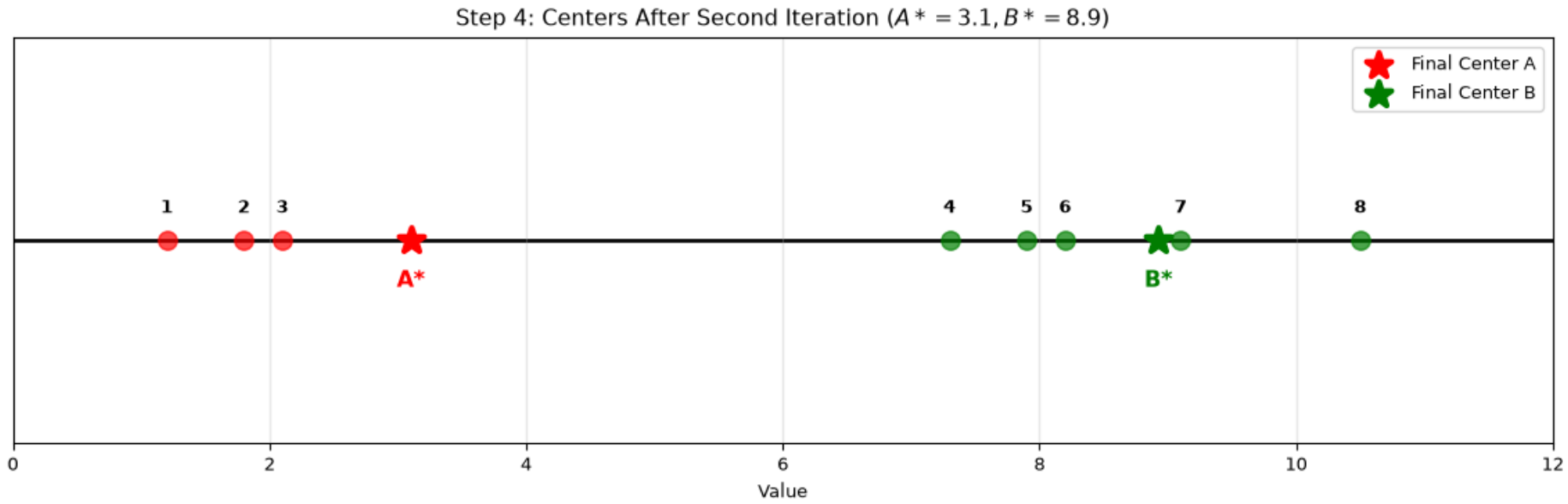
Repeat the assignment with the new centers ($A' = 3.1, B' = 8.9$).

Complete the table:

Point	Value	Distance to A'	Distance to B'	Assigned to
1	1.2	1.9	8.3	A
2	1.8			
3	2.1			
4	7.3			
5	7.9			
6	8.2			
7	9.1			
8	10.5			

Step 4: Second Iteration Cluster Assignment Visualization

▶ Code




```
New Center A: 3.1
New Center B: 8.9
```

Convergence Check

One type of convergence is when the centers stop changing.

1st Iteration Center	2nd Iteration Center	Abs Difference
A = 4	A = 3.1	0.9
B = 11	B = 8.9	2.1

Should we do another iteration?

The Steps

1. Pick K cluster centers $\{c_1, \dots, c_K\}$.
2. For each j, define the cluster C_j as the set of points in X that are closest to center c_j.
3. **For each j, update c_j to be the center of mass of cluster C_j.**
4. Repeat (i.e., go to Step 2) until convergence.

2nd Iteration Update Centers

Cluster A (points: 1.2, 1.8, 2.1):

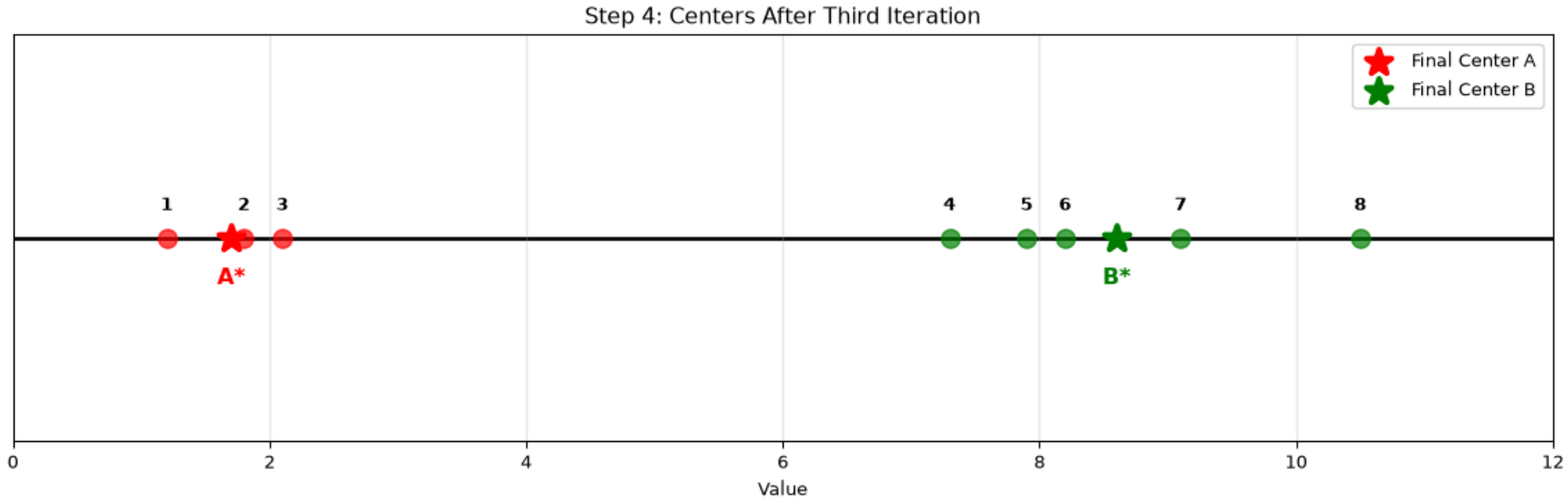
- New center A = (1.2 + 1.8 + 2.1) ÷ 3 = 1.7

Cluster B (points: 7.3, 7.9, 8.2, 9.1, 10.5):

- New center B = (7.3 + 7.9 + 8.2 + 9.1 + 10.5) ÷ 5 = 8.6

Step 4: Second Iteration Visualization

▶ Code



Step 5: Check for Convergence (1 minute)

Questions:

- Would cluster assignments change if we continued?
- Would the centers change if we continued?
- Have we found the optimal clustering?

Key Learning Points

1. **Algorithm Steps**: Initialize → Assign → Update → Repeat
2. **Distance Calculation**: |point - center| in 1D
3. **Center Updates**: Average of assigned points
4. **Convergence**: Centers stop changing when optimal
5. **Initialization Matters**: Different starts can lead to different results

Quick Assessment

Answer on your paper:

1. **What is the correct order of the 4 steps of k-means?**
    - [ ] Repeat until convergence
    - [ ] Assign points to closest center
    - [ ] Initialize centers
    - [ ] Update centers to average of assigned points
2. **How do you update a center?**
    - Answer:
3. **What does “convergence” mean?**
    - Answer:

Recap and Next

Today we covered:

- k-means clustering
- strengths and weaknesses
- Importance of initialization and cluster number
- Feature scaling
- Example applications:
  - Customer segmentation
  - Plant species classification with Iris data
  - Image color compression

Coming up next, we'll look at some practical aspects of applying k-means.