Knowledge Discovery & Data Mining
– Clustering –

Yong Zhuang

Recap: Supervised vs. Unsupervised Learning

- Supervised Learning
  - Data: both the features, x, and a target, y, for each item in the dataset
  - Goal: ‘learn’ how to predict the target from the features, y = f(x)
  - Example: Regression and Classification
- Unsupervised Learning
  - Data: Only the features, x, for each item in the dataset
  - Goal: discover ‘interesting’ things about the dataset
  - Example: Clustering, Dimensionality reduction, Principal Component Analysis (PCA)

Outline

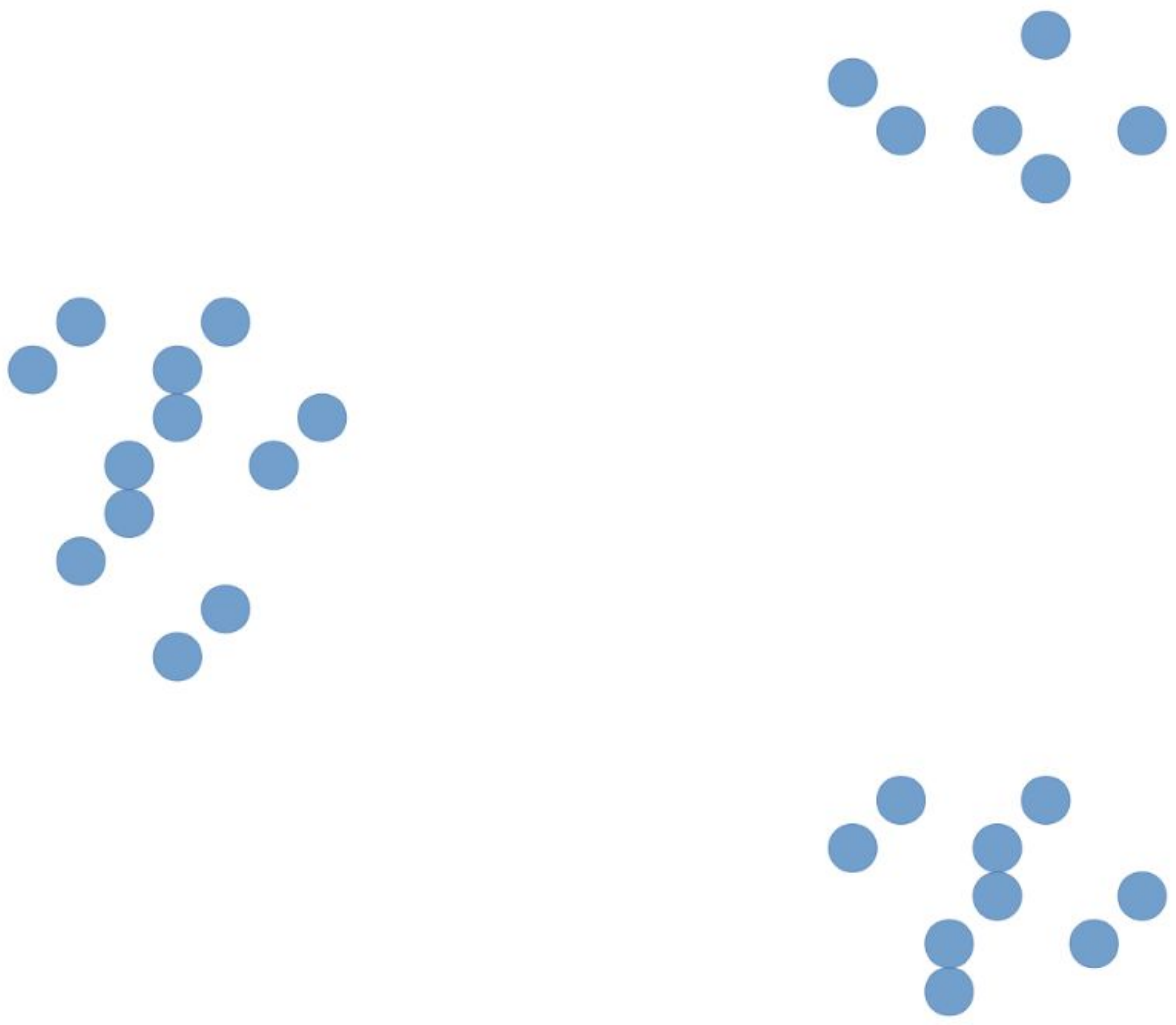
- Introduction to Clustering
- K-Means
  - K-Means Algorithm
  - Limitation of K-Means
  - K-Means Implementation
- Agglomerative Clustering

Clustering

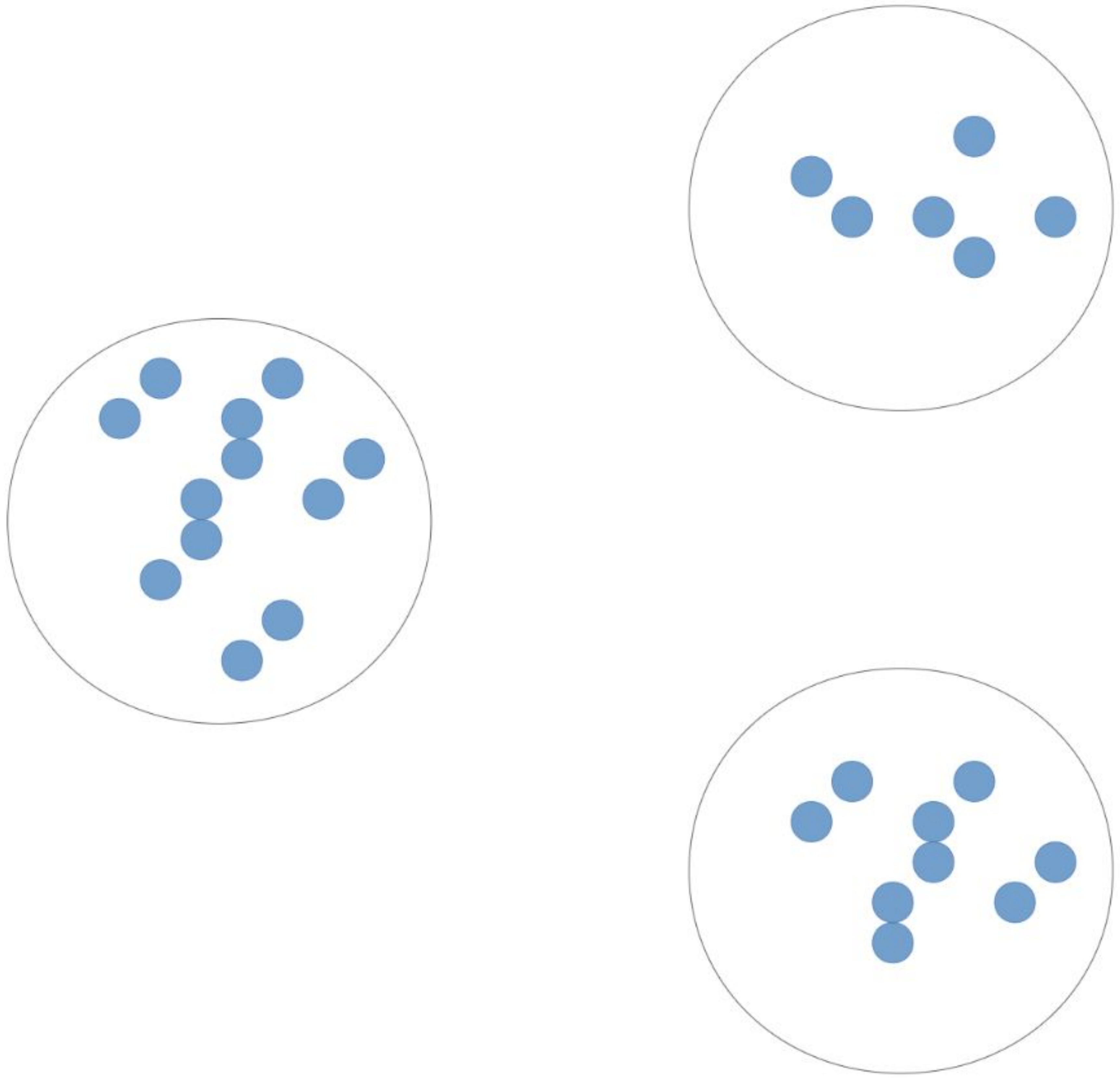
- Clustering is the task of discovering unknown subgroups in data, or clusters
- The goal is to partition the dataset into clusters where ‘similar’ items are in the same cluster and ‘dissimilar’ items are in different clusters
- Example:
  - Social Network Analysis: Clustering can be used to find communities
  - Ecology: cluster organisms that share attributes into species, genus, etc…
  - Handwritten digits where the digits are unknown

Question: What is the difference between Clustering and Classification?

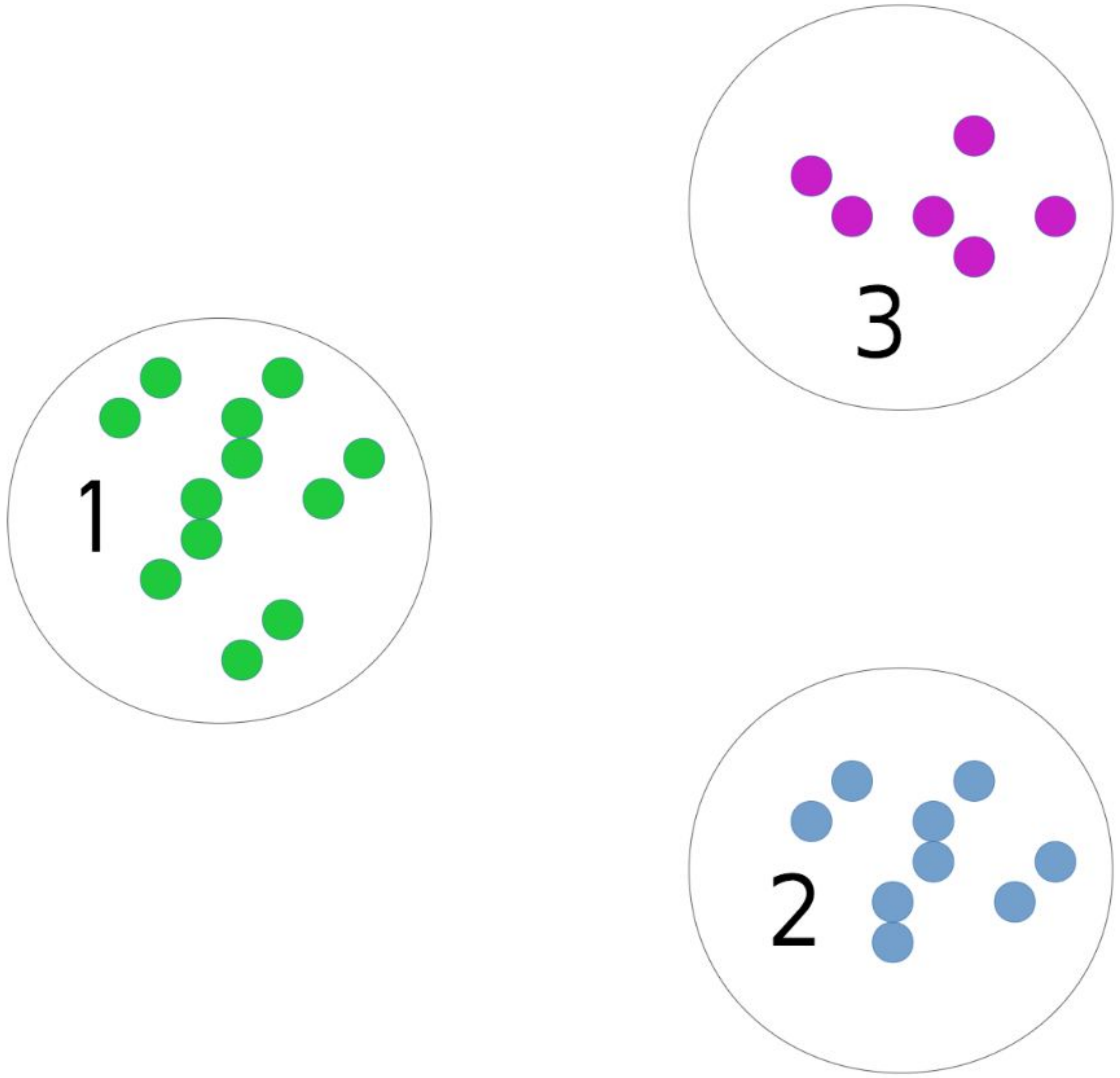
Clustering



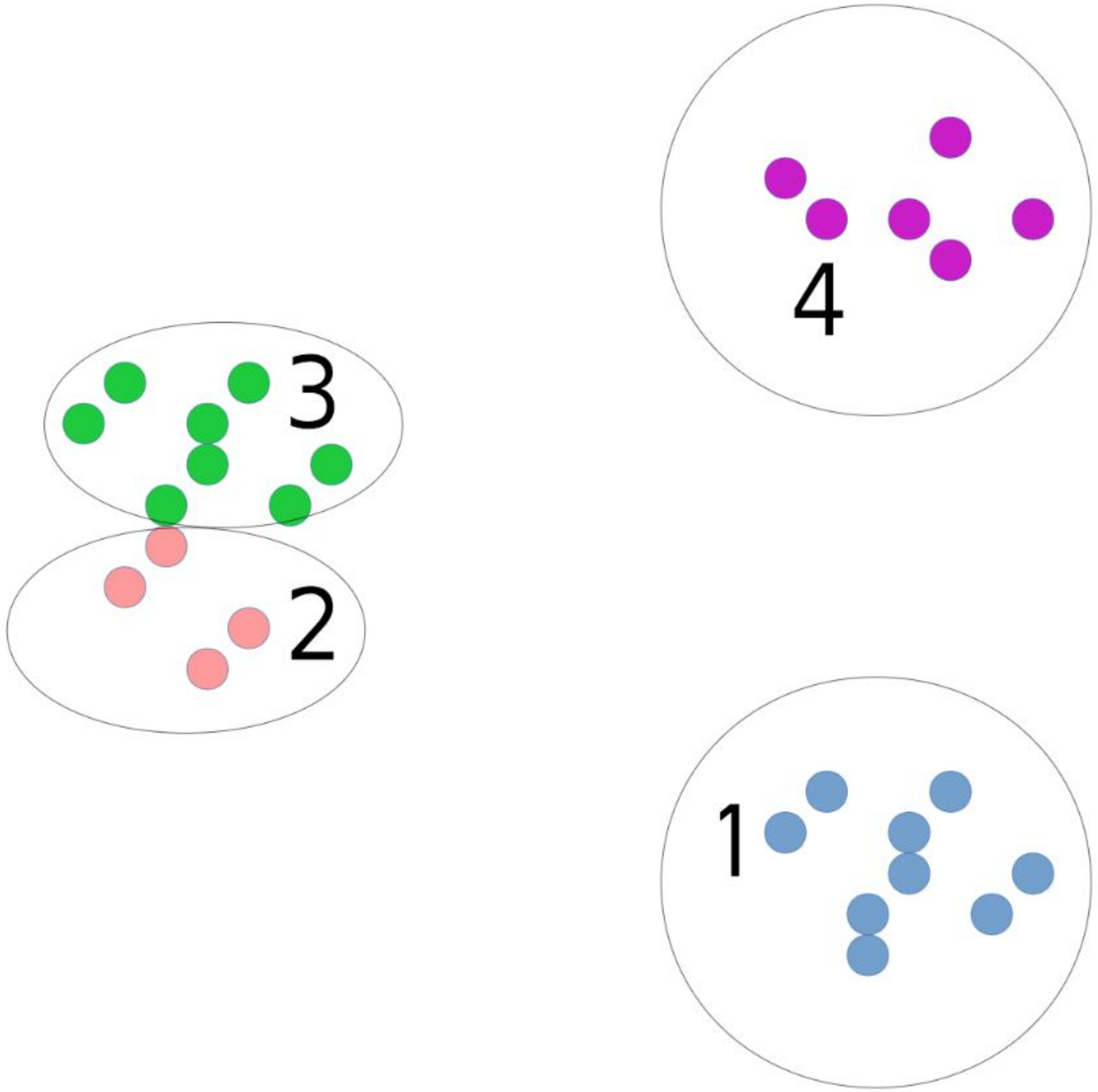
Clustering



Clustering



Clustering



Summary of Clustering

- Partition data into groups (clusters)
- Points within a cluster should be “similar”.
- Points in different cluster should be “different”.

Goal of Clustering

- Data Exploration
  - Are there coherent groups ?
  - How many groups are there ?
- Data Partitioning
  - Divide data by group before further processing

Outline

- Introduction to Clustering
- K-Means
  - K-Means Algorithm
  - Limitation of K-Means
  - K-Means Implementation
- Agglomerative Clustering

Partitioning-based Clustering Problem

- Data: A collection of samples $x_i, for\ i = 1, \ldots, n, where\ x_i \in R^d$
- Partition samples into K clusters, so that each cluster is as much cohesive as possible.

Question: How do we define the “cohesion” of a cluster?

Cohesion
- Cluster is cohesive

→

Similarity
- samples within cluster are similar to the cluster center

→

Closeness
- samples are close to corresponding cluster center

→

Distance
- distance between samples and cluster center are minimized

K-Means: Objective Function

- Assign each sample x_i to its closest cluster C_k with center μ_k , as to minimize the total within-cluster distance:

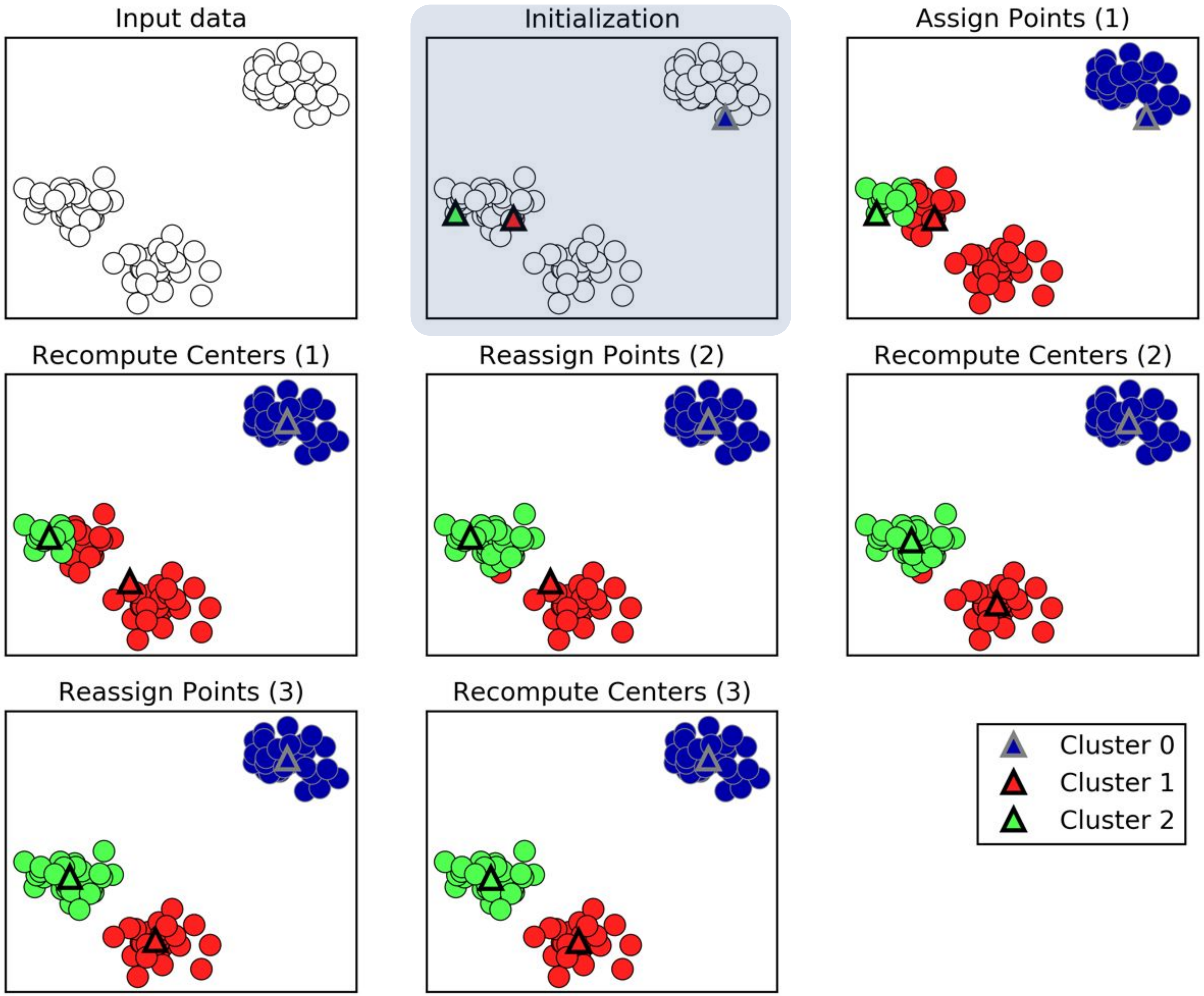
$$\arg\min_{\mu, C} \sum_{k=1}^{K} \sum_{x_i \in C_k} d(x_i, \mu_k)^2$$

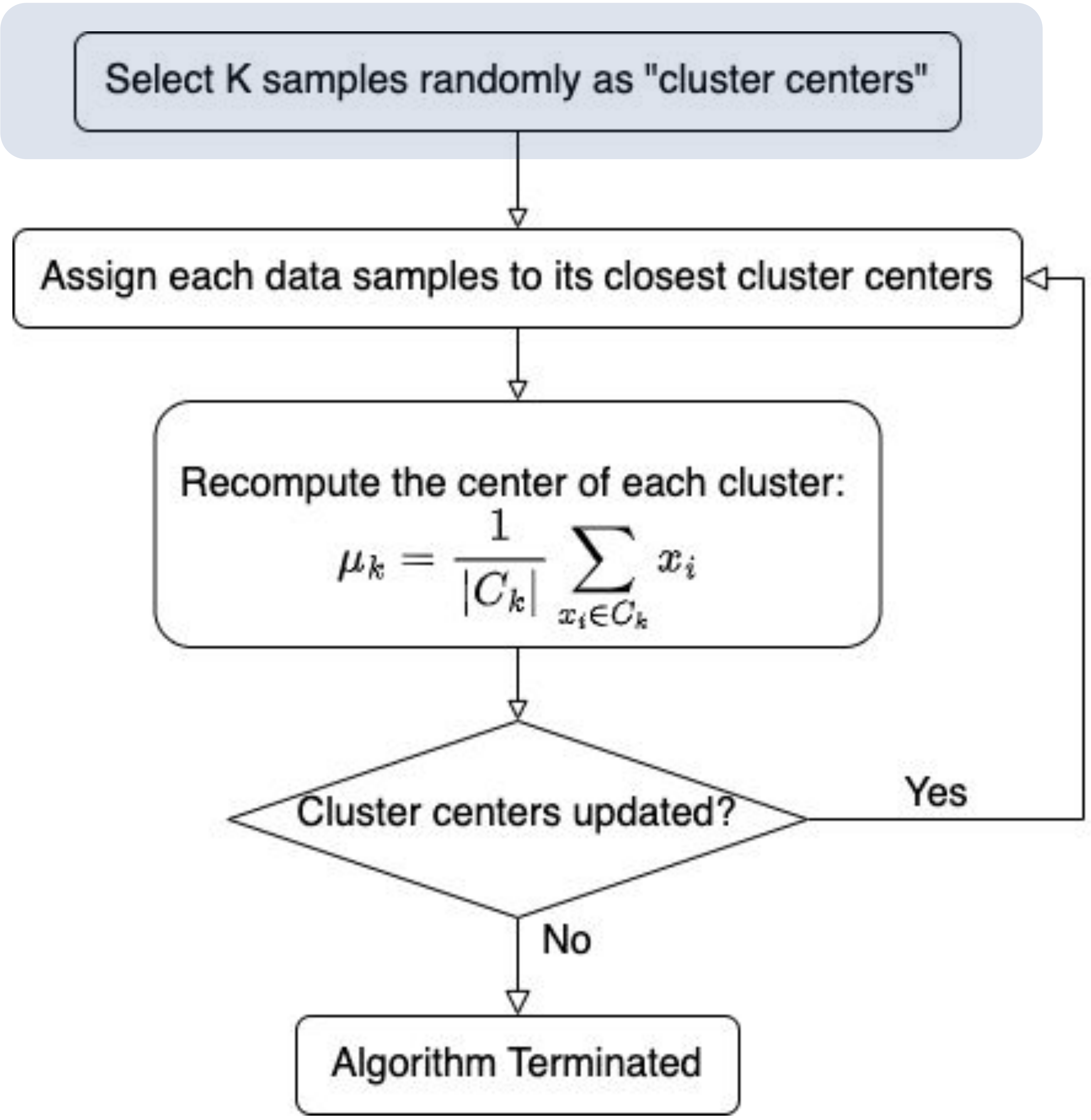
- Where d can be any distance/dissimilarity functions.
- In general, we suppose the data is on a Euclidean space. Then our objective is to minimize the following formula:

$$\arg\min_{\mu, C} \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

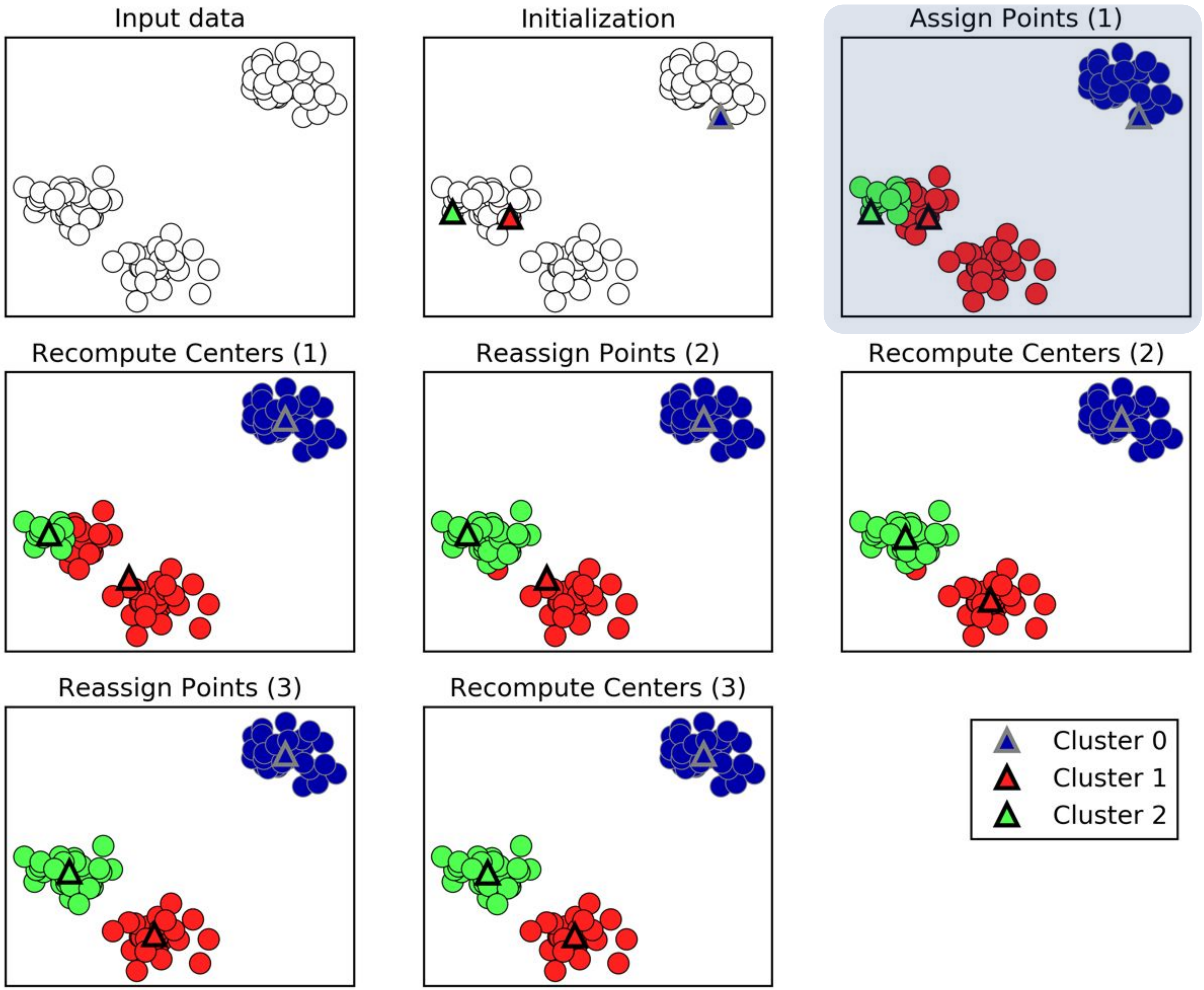
- We minimize the within-cluster Sum of Squared Error (SSE).
- For each point, the error is the distance to its nearest cluster center.

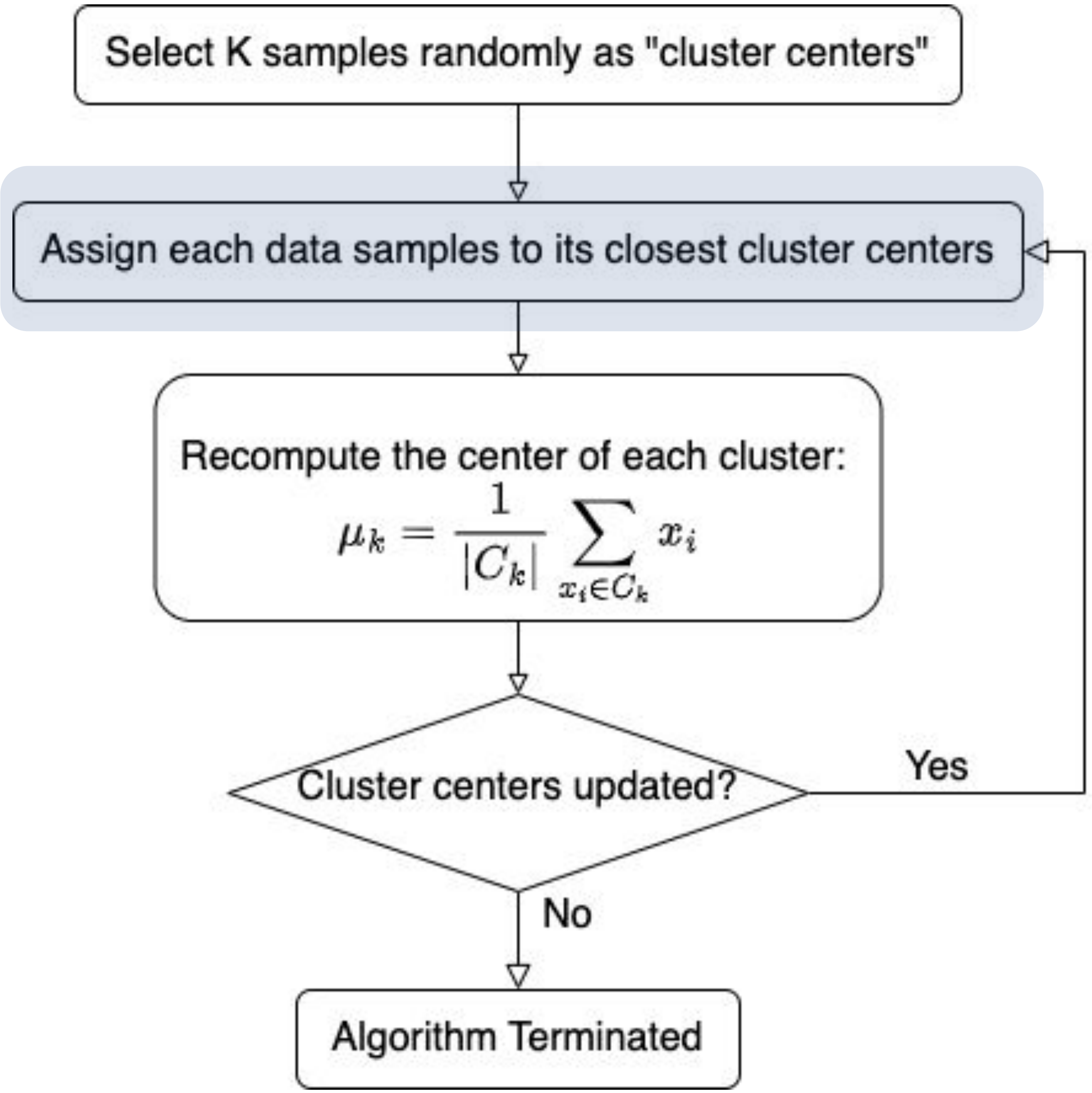
K-Means Algorithm



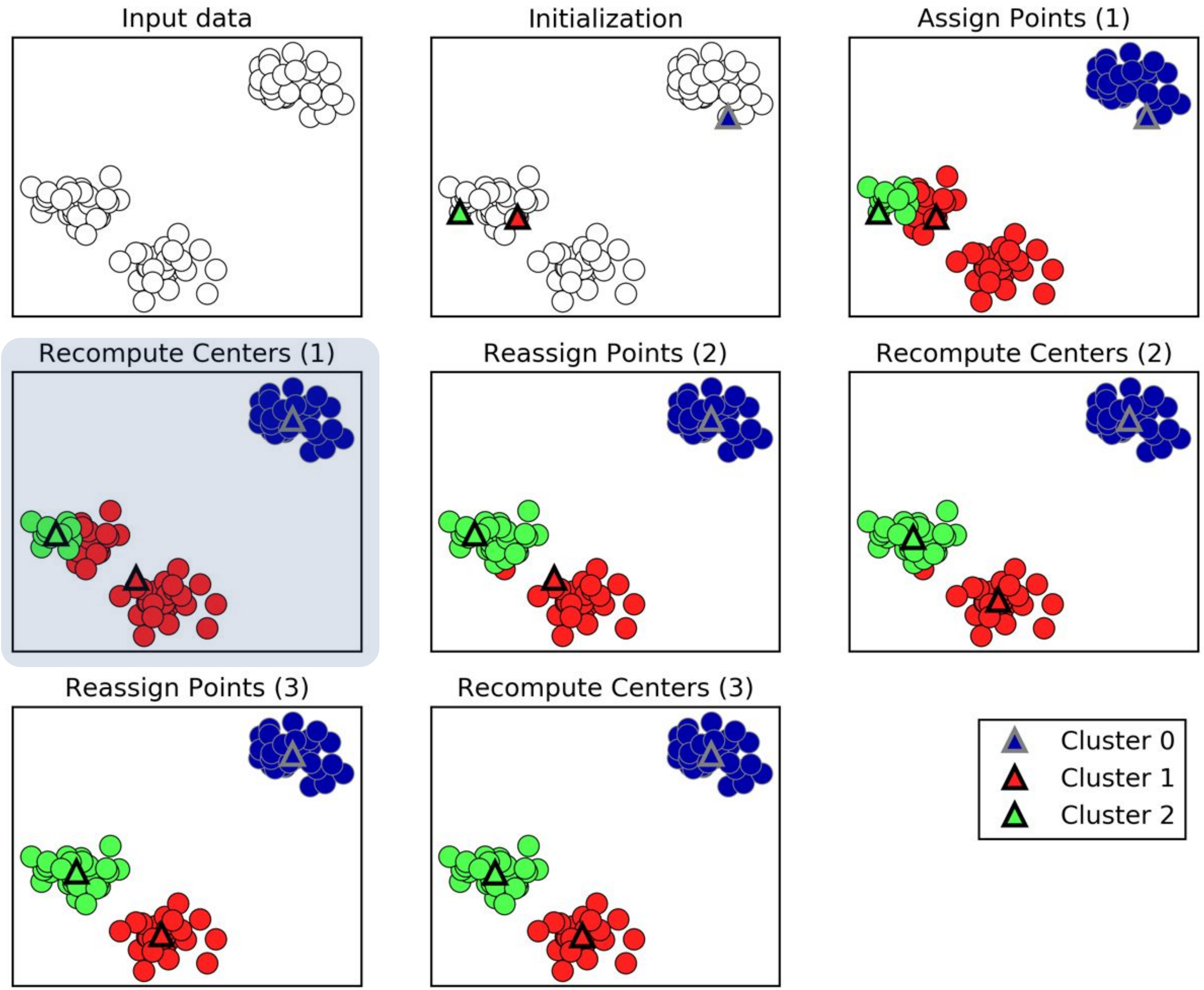


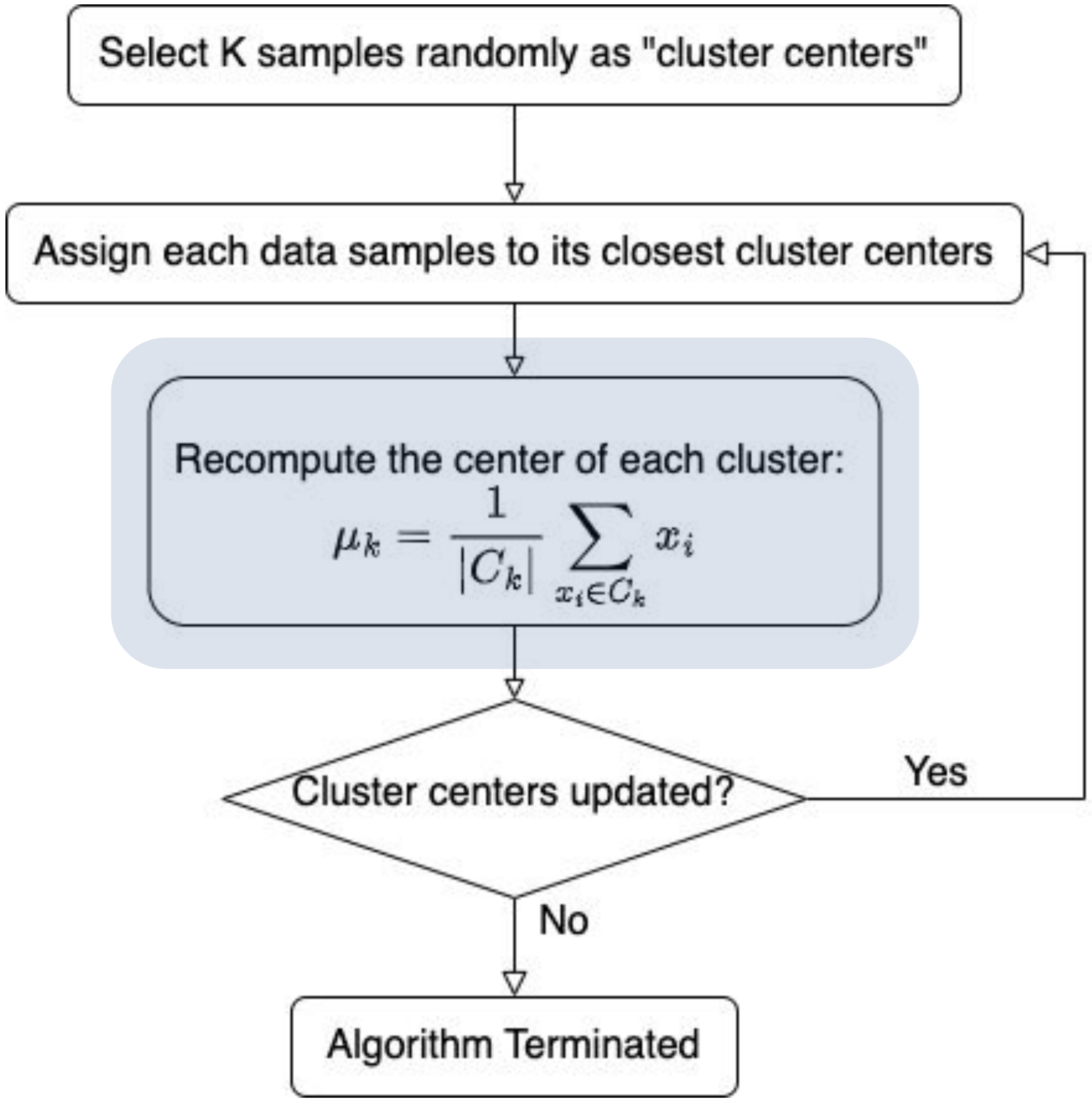
K-Means Algorithm



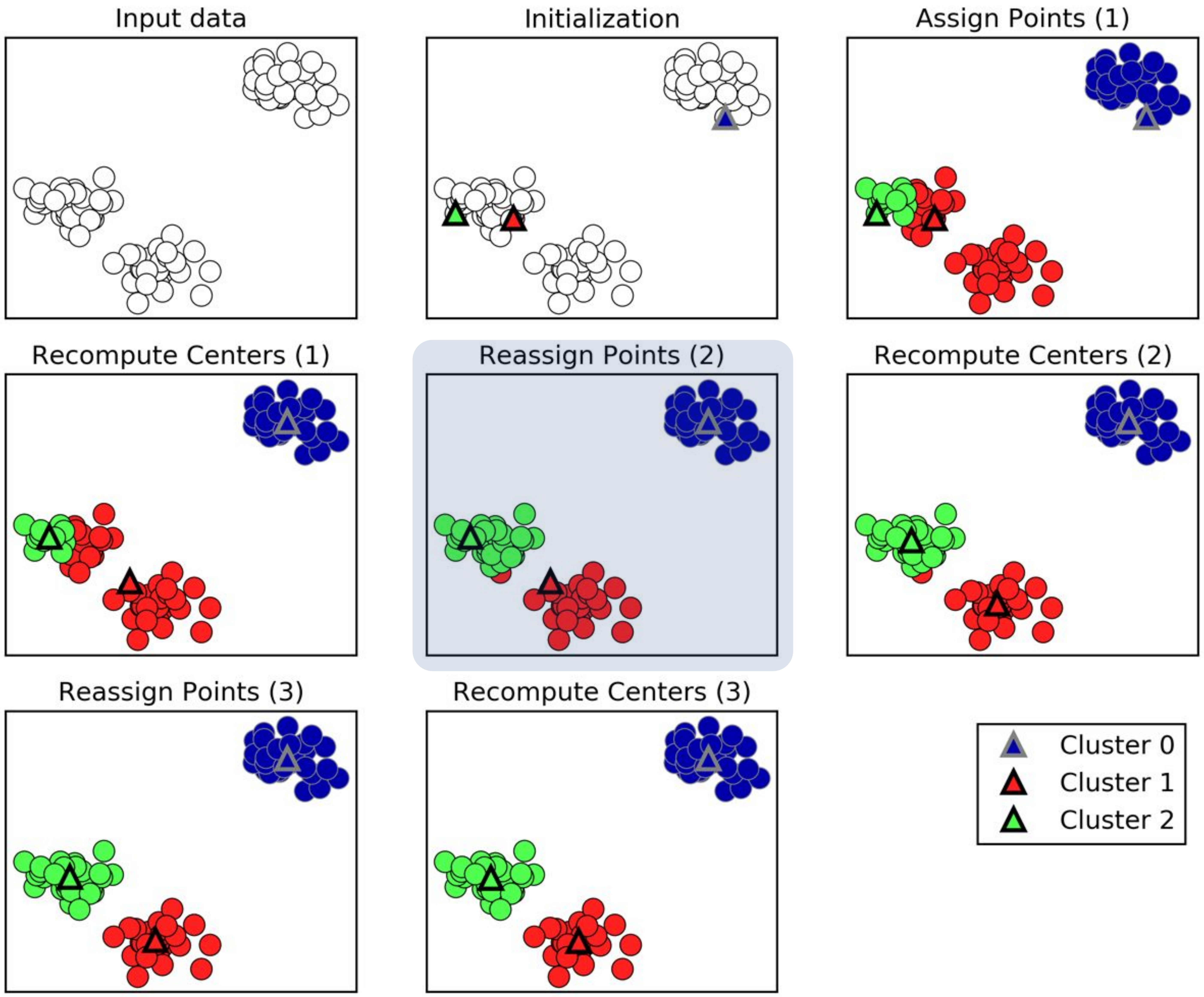


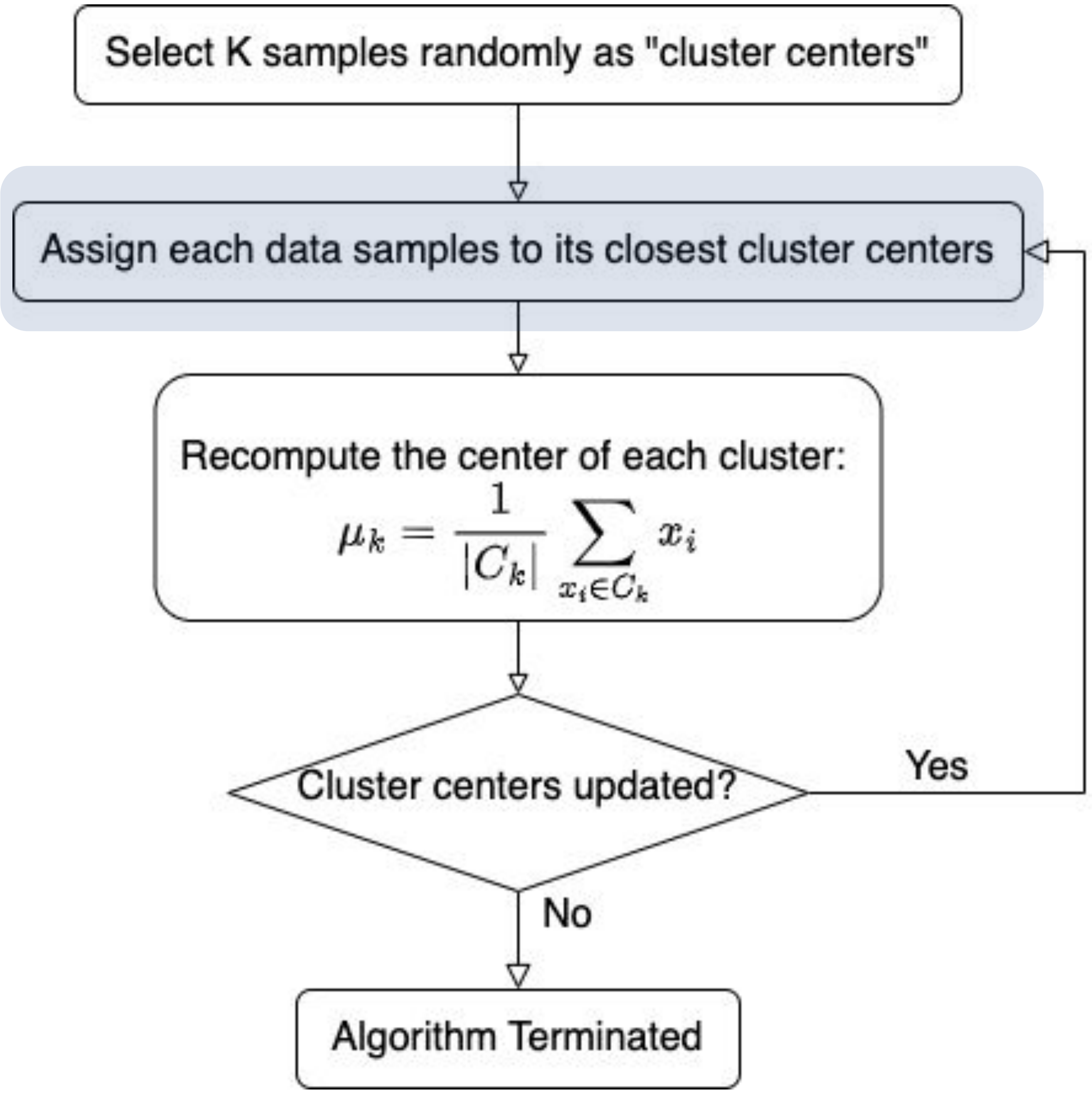
K-Means Algorithm



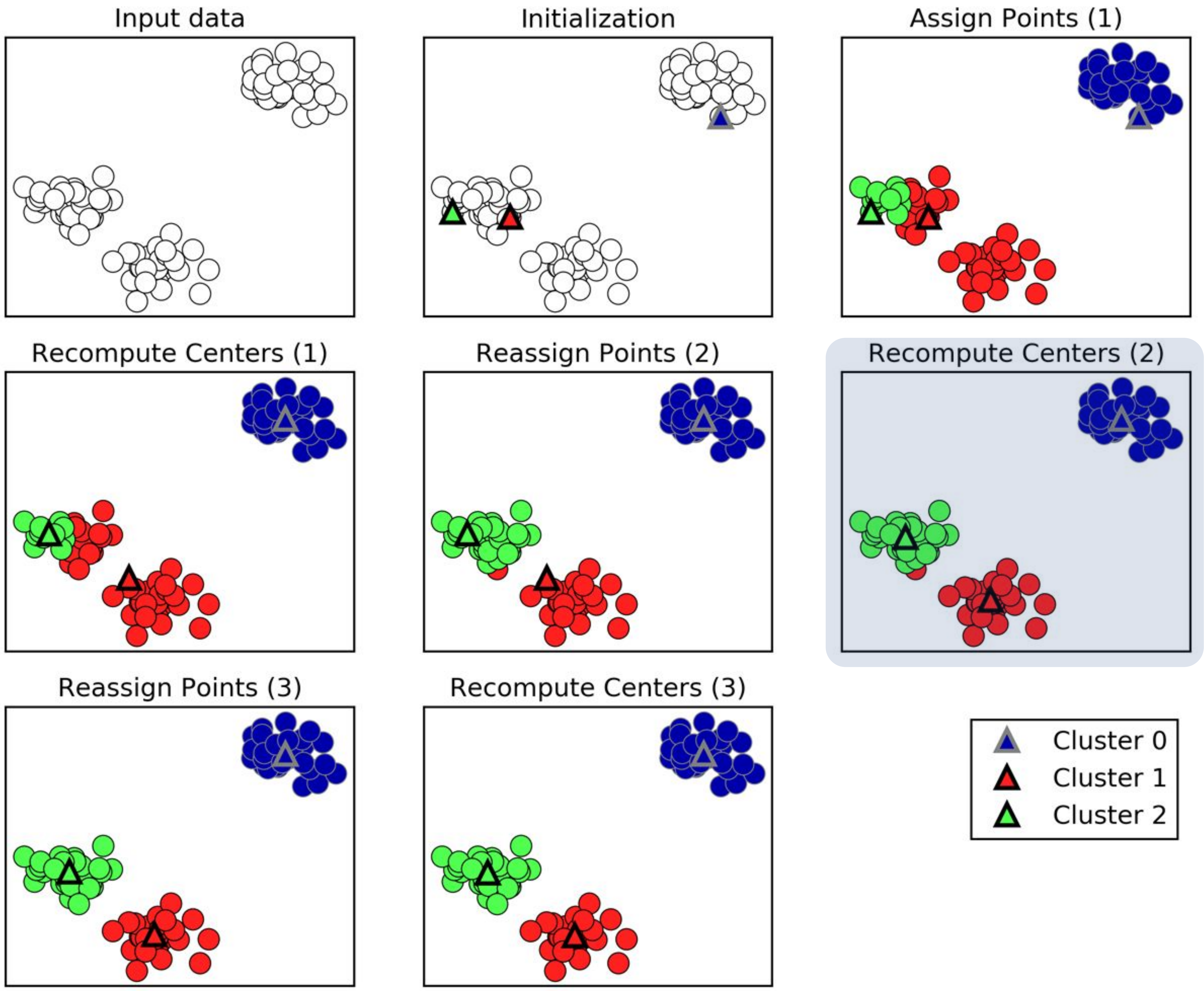


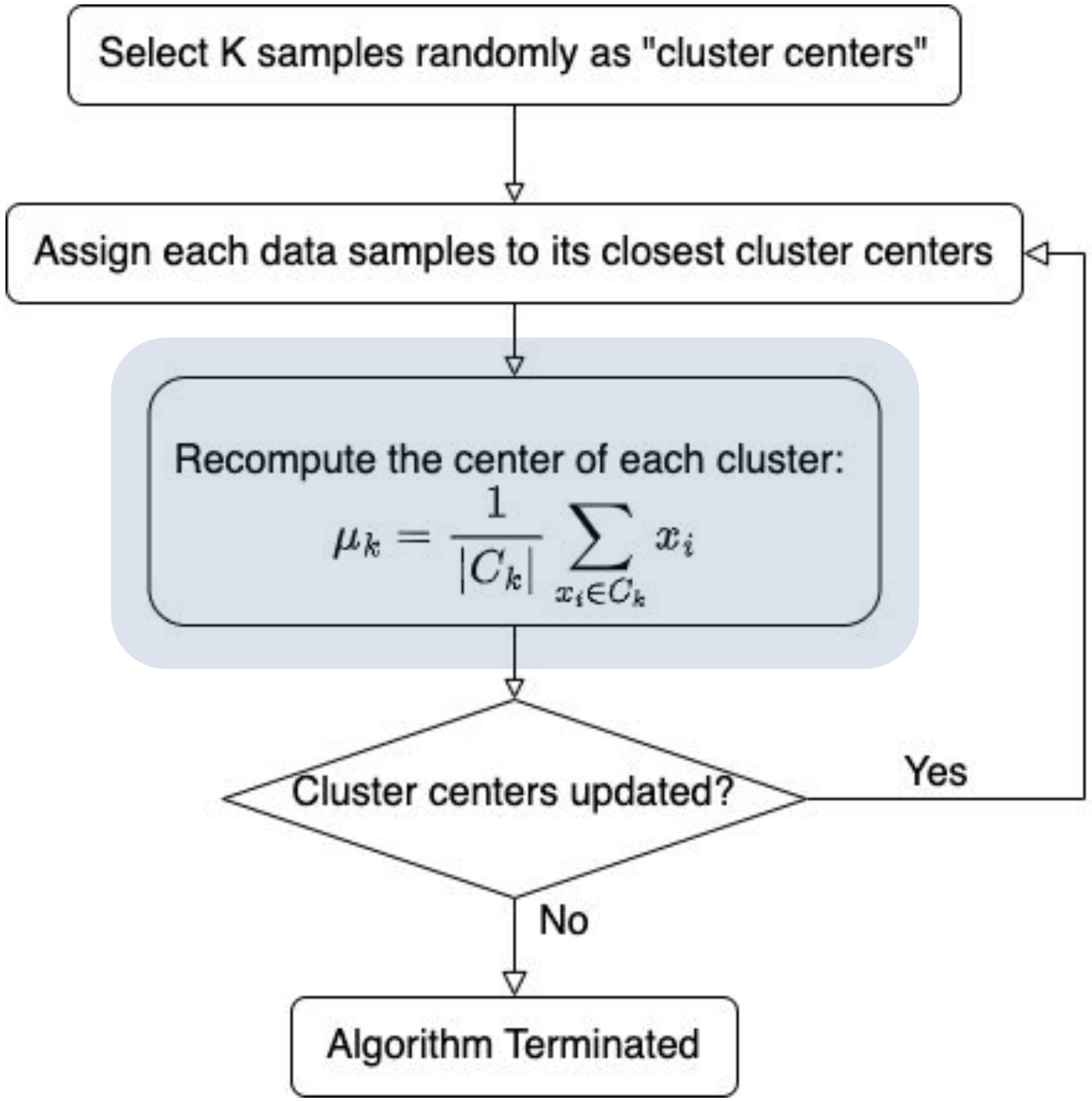
K-Means Algorithm



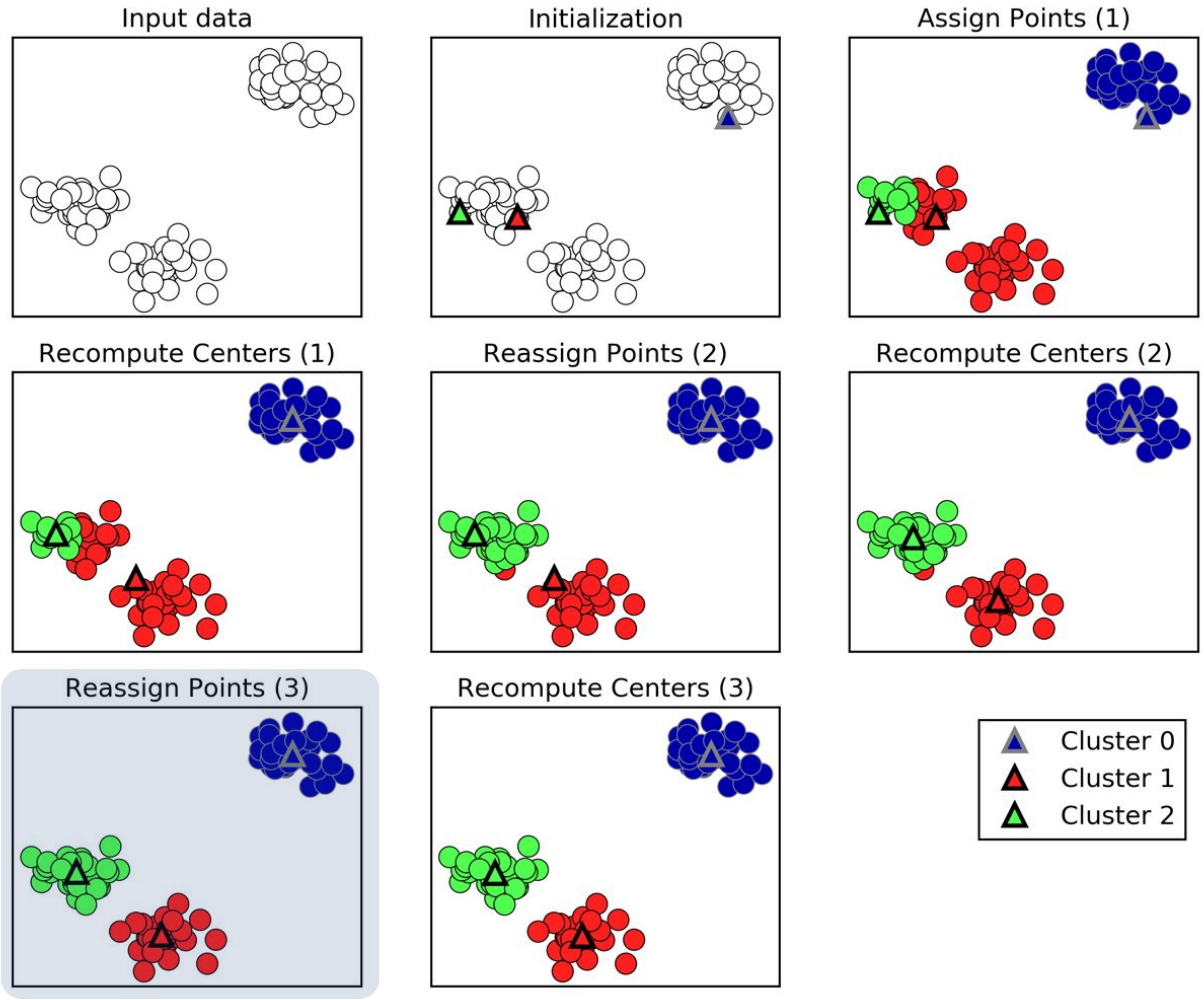


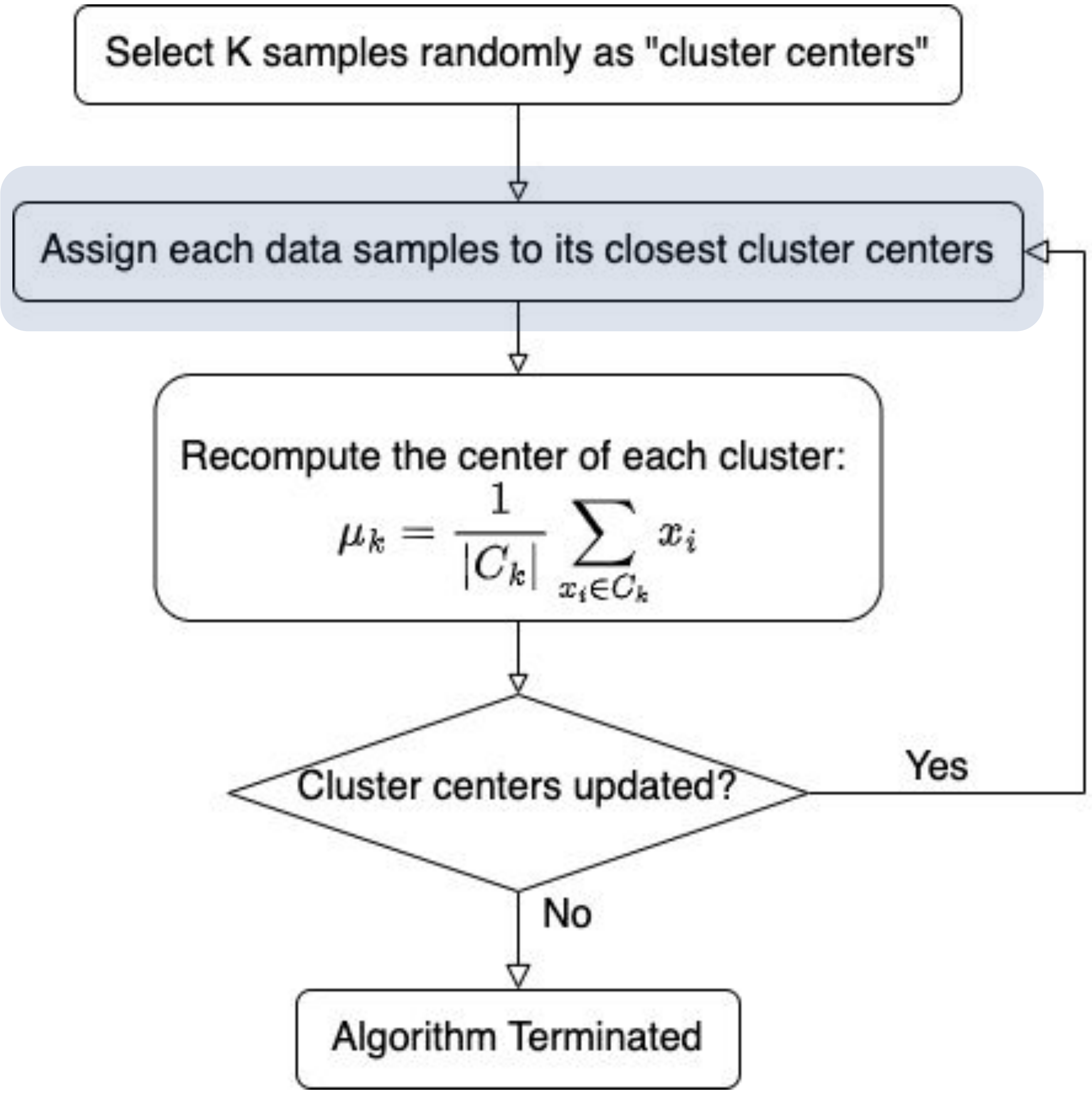
K-Means Algorithm



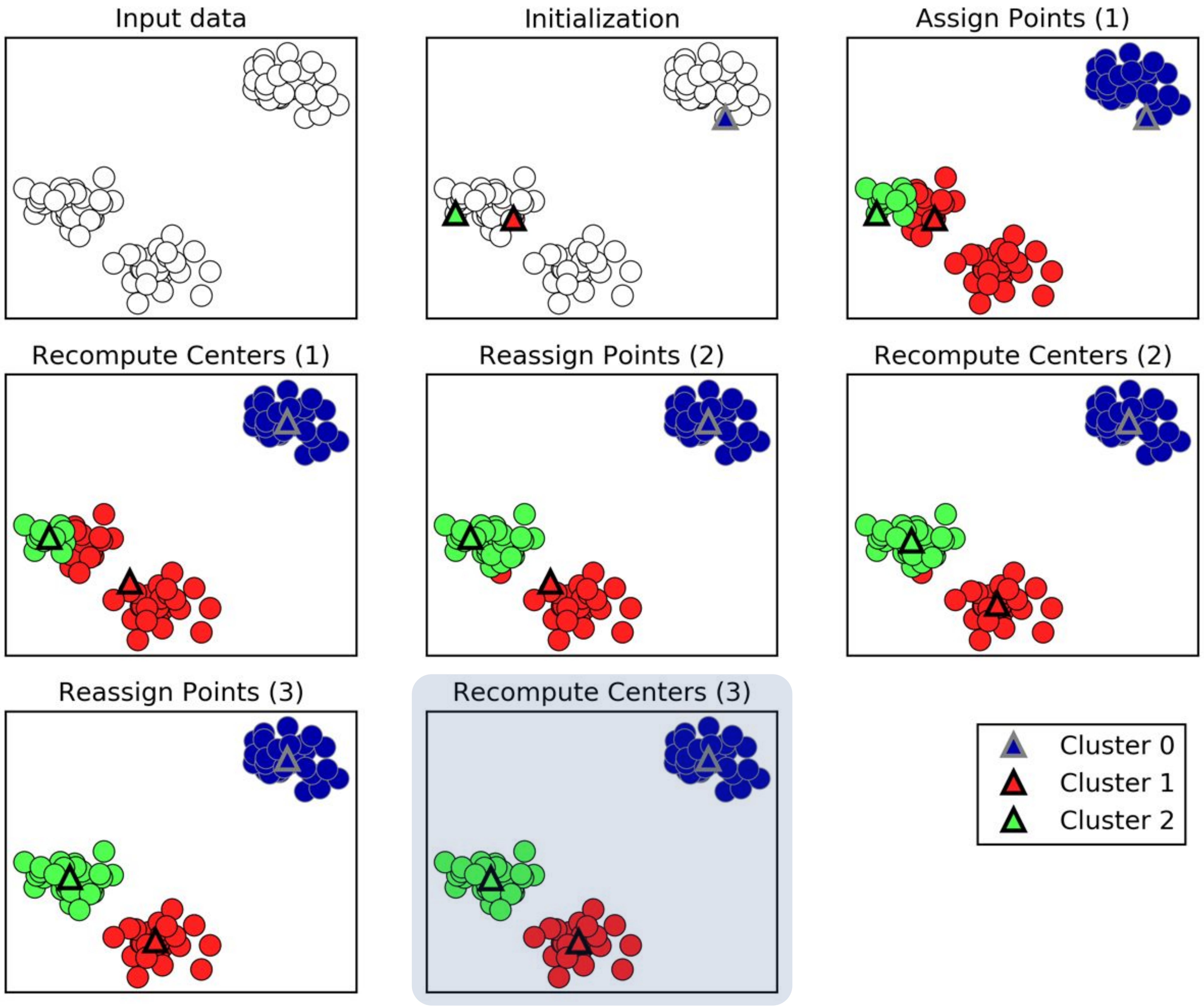


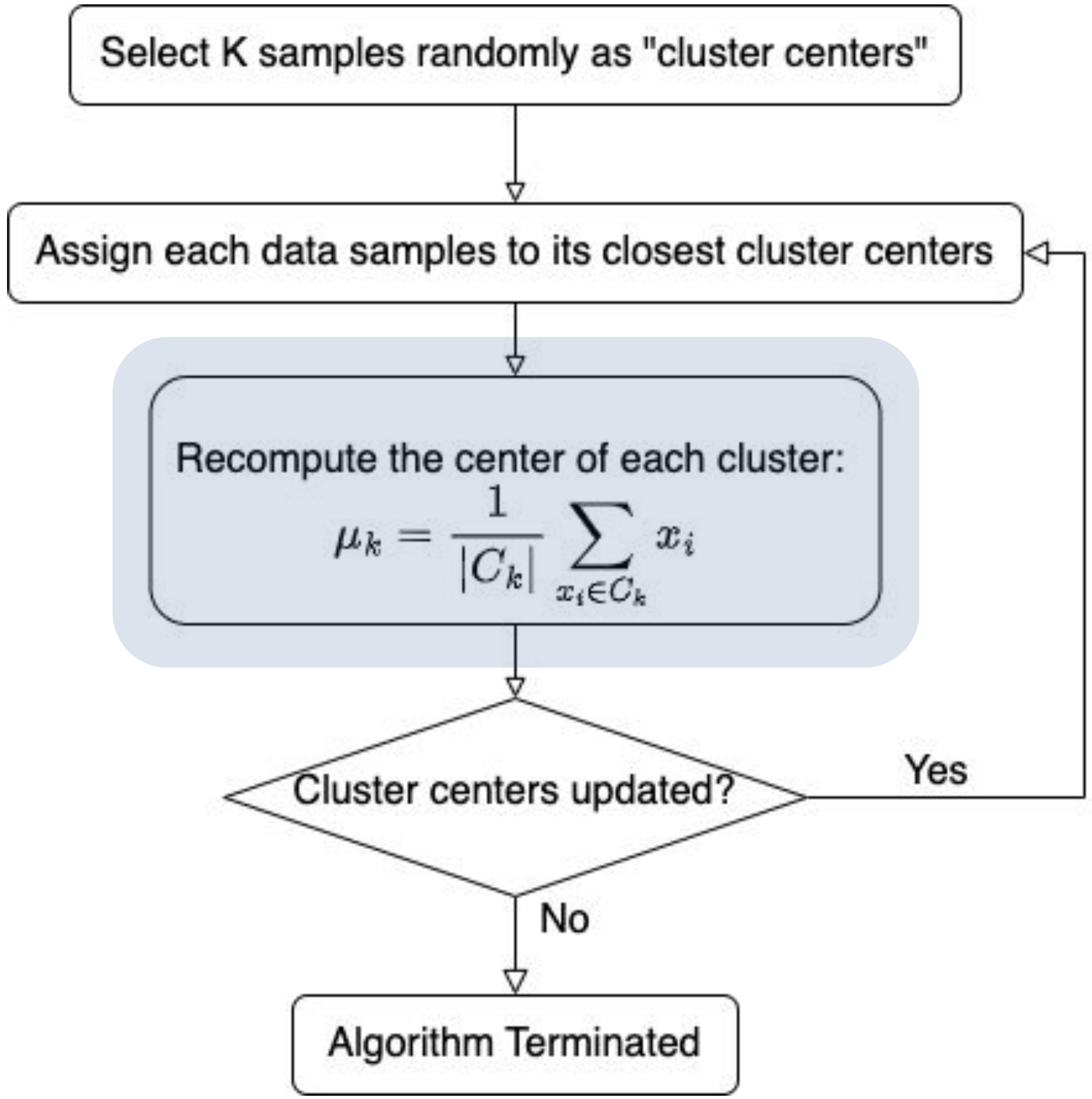
K-Means Algorithm





K-Means Algorithm

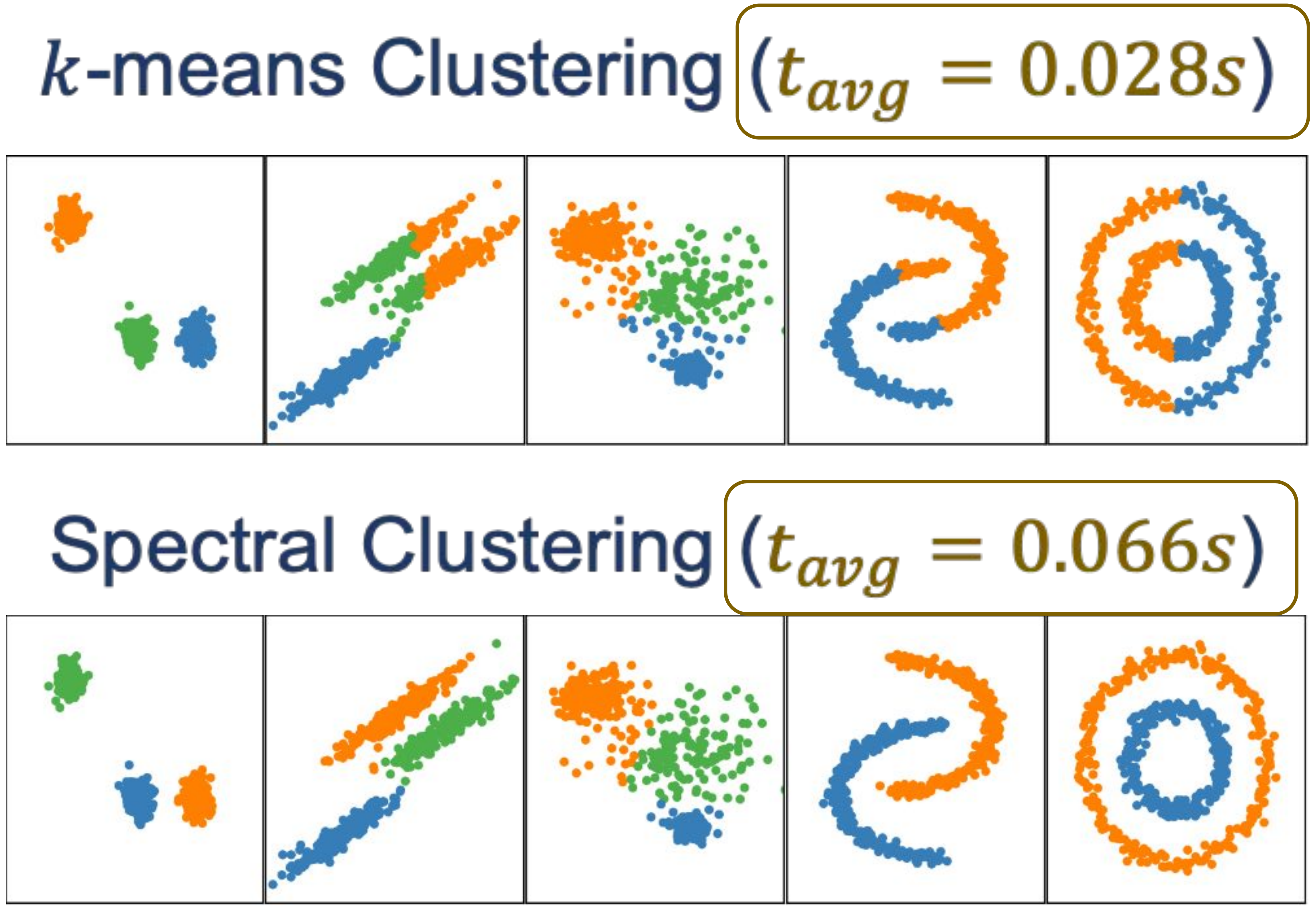




Outline

- Introduction to Clustering
- K-Means
  - K-Means Algorithm
  - Limitation of K-Means
  - K-Means Implementation
- Agglomerative Clustering

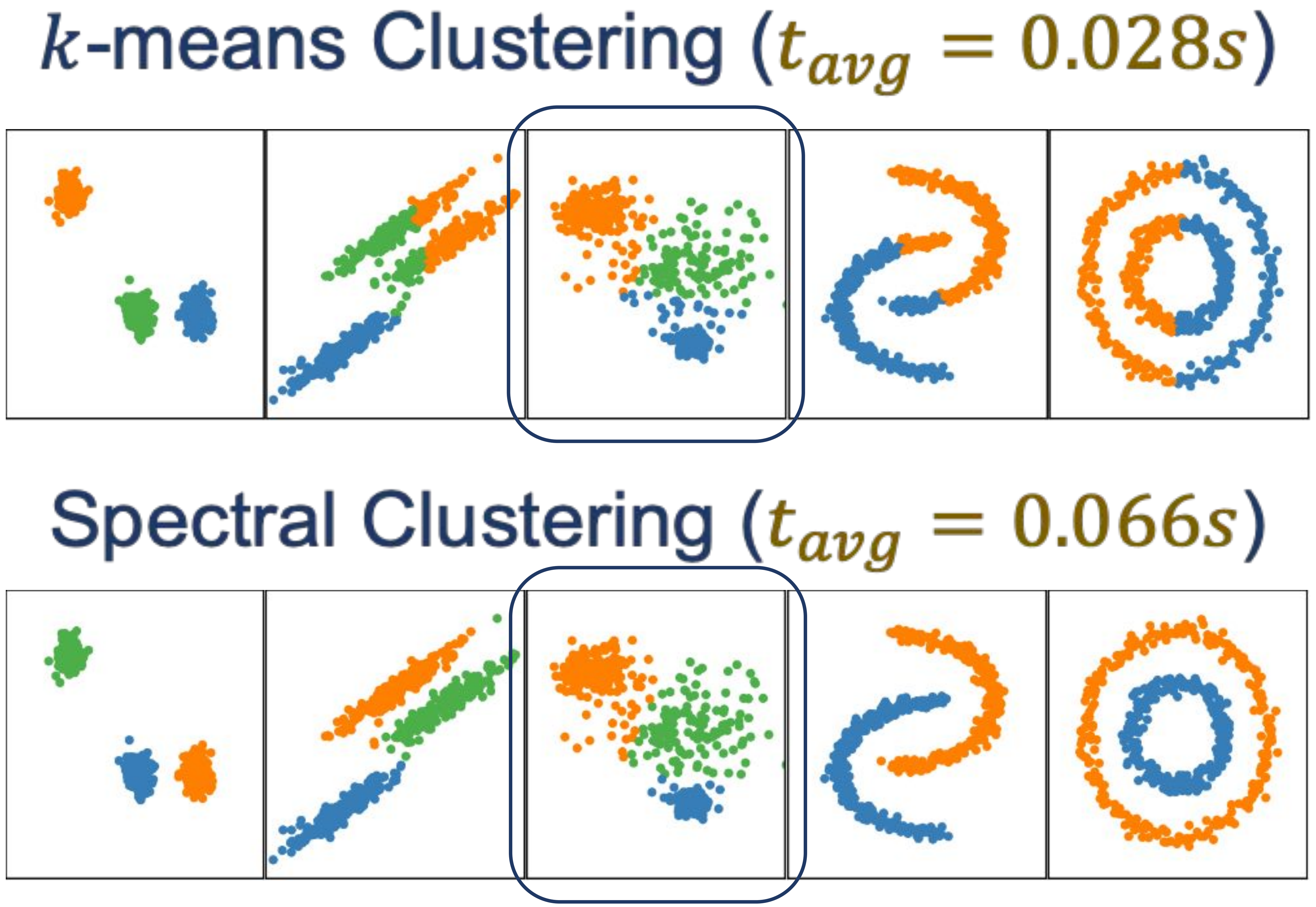
K-Means: Strength and Limitations



Strengths:

1. Simple and Efficient.

K-Means: Strength and Limitations



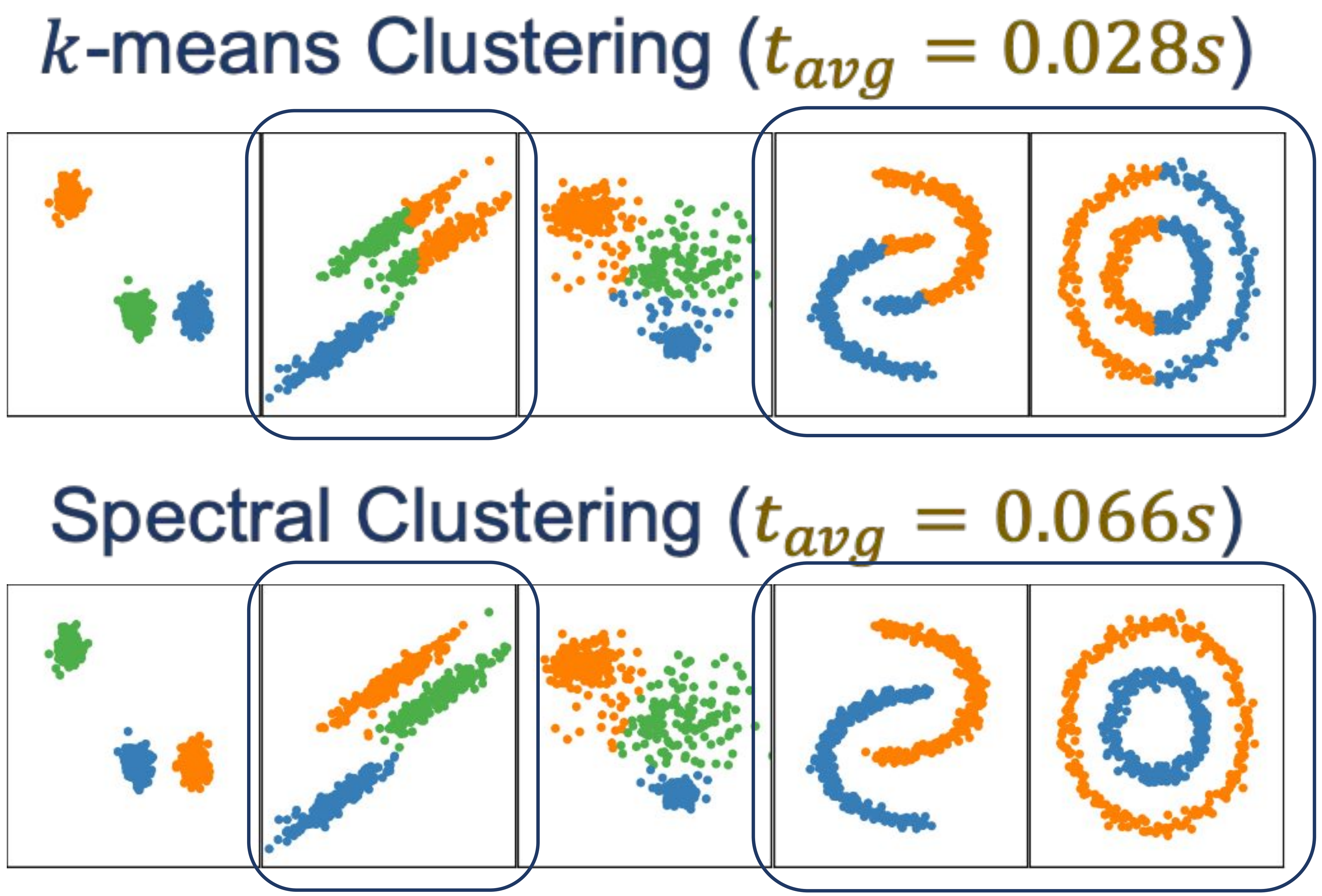
Strengths:

1. Simple and Efficient.

Weaknesses:

1. Clusters with different sizes and densities.

K-Means: Strength and Limitations



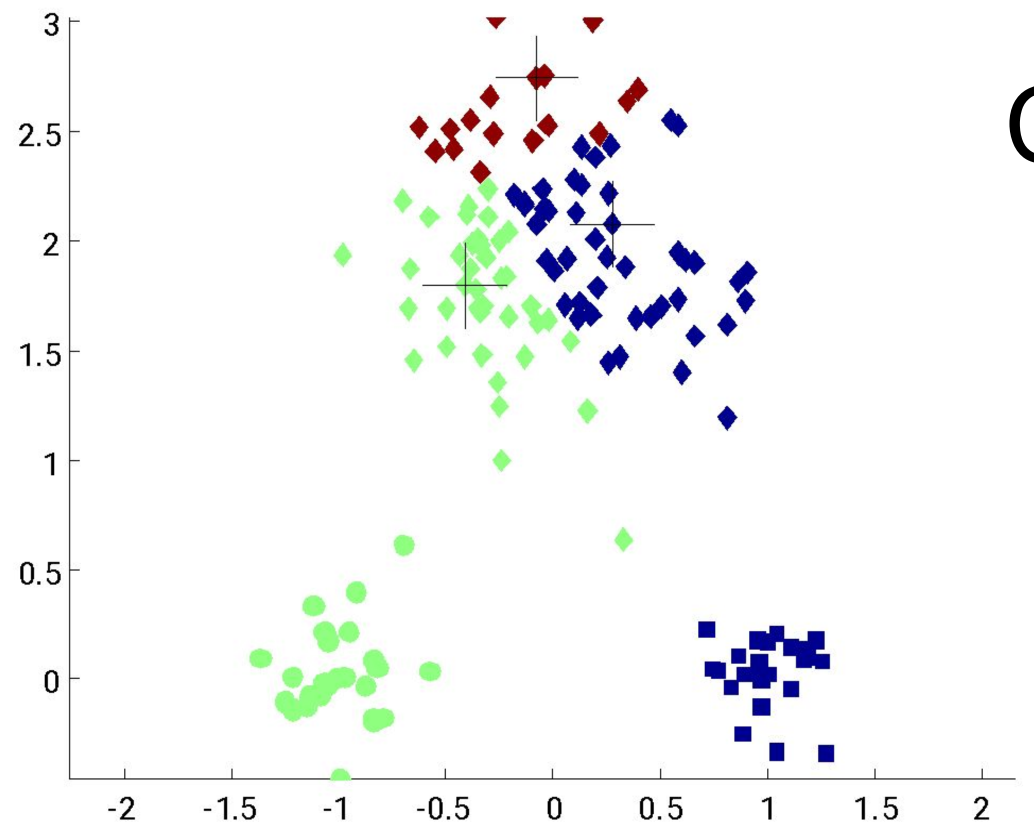
Strengths:

1. Simple and Efficient.

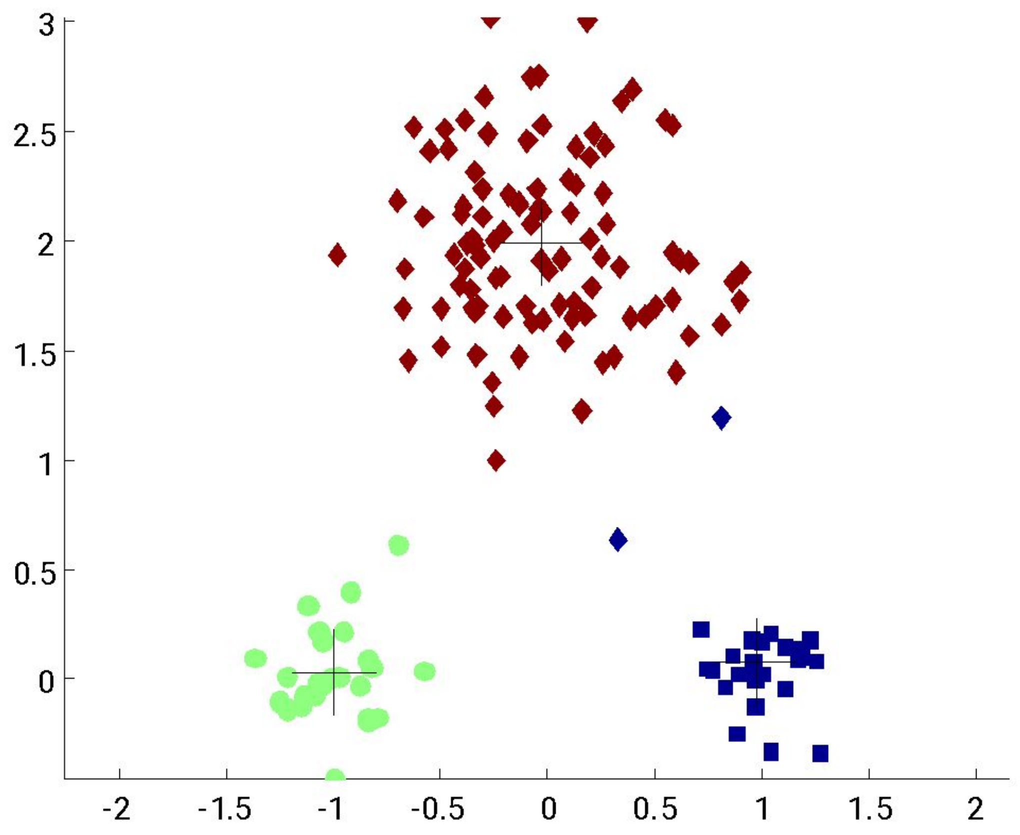
Weaknesses:

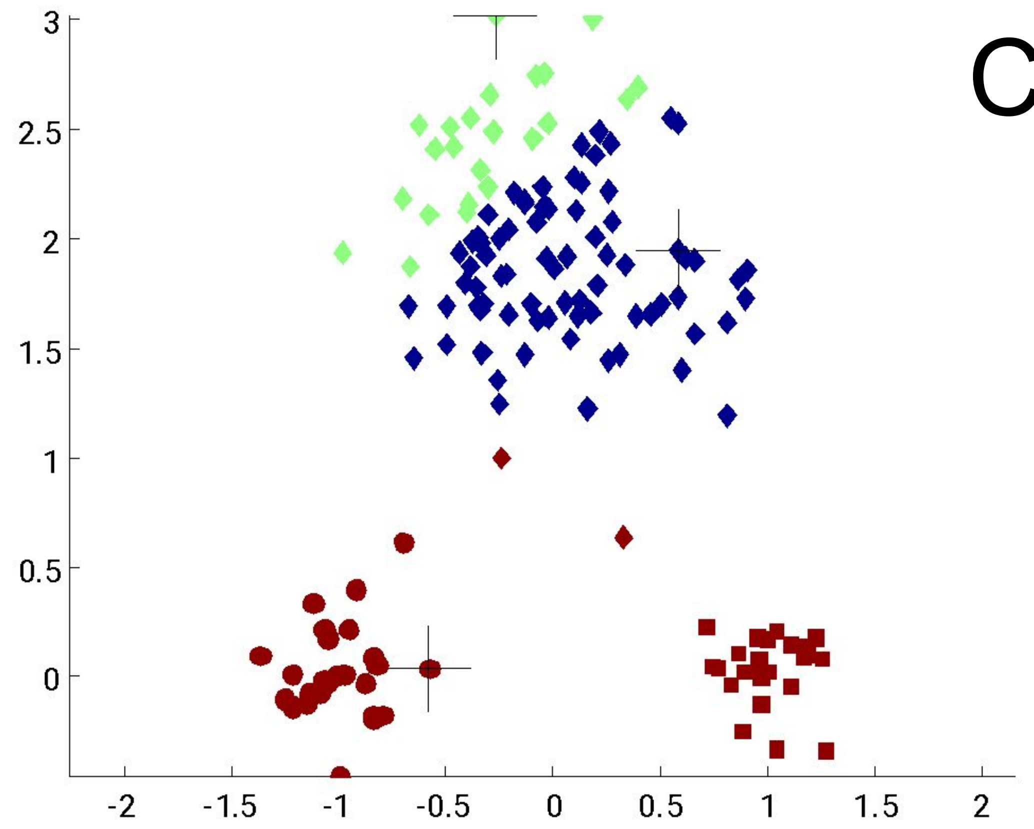
1. Clusters with different sizes and densities.
2. Non-spherical clusters.

K-Means: Strength and Limitations

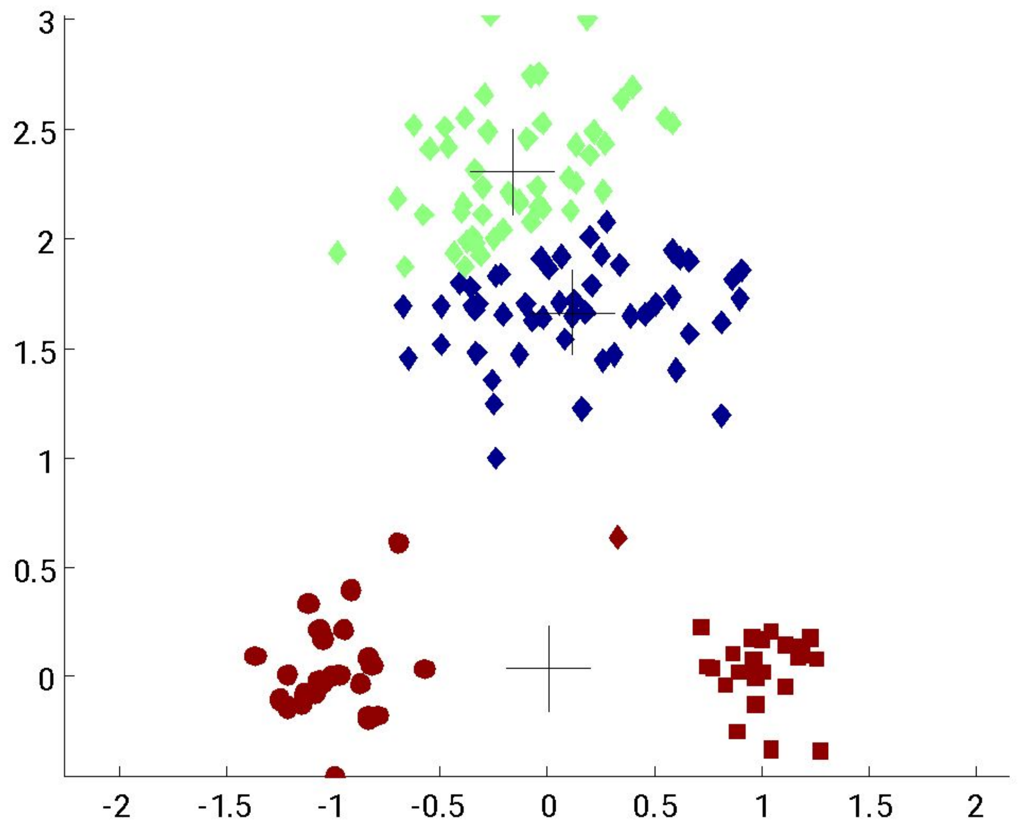


Converge in 6 iterations





Converge in 5 iterations



Strengths:

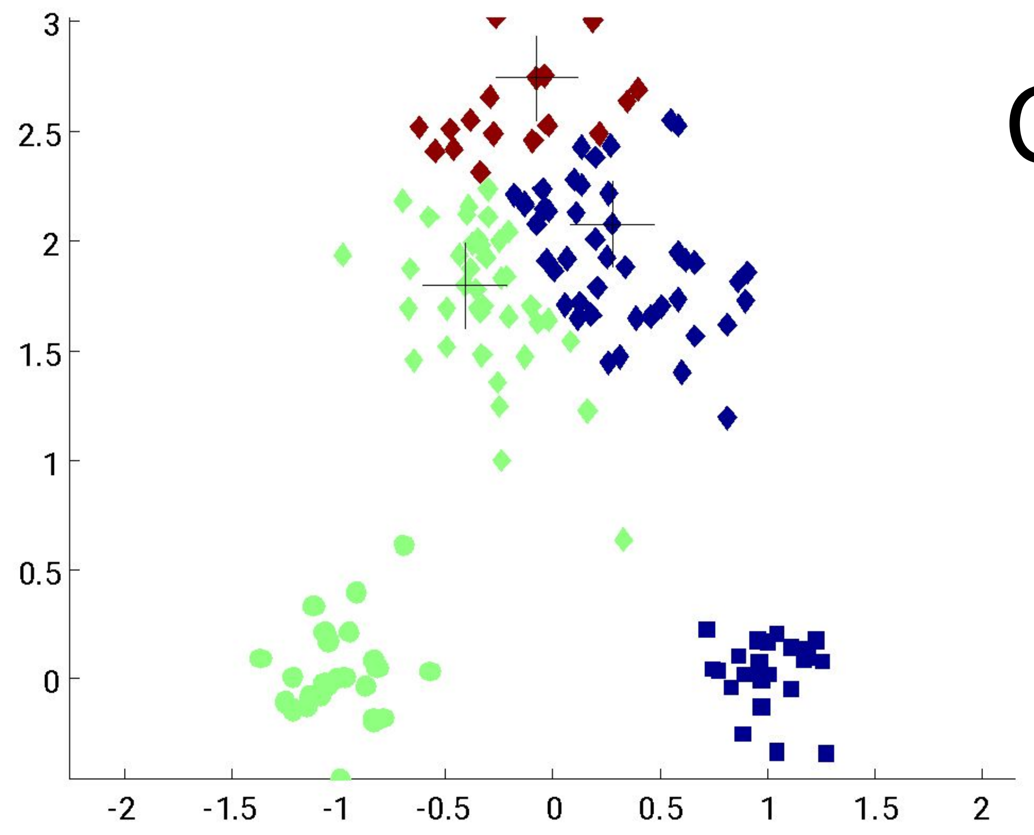
1. Simple and Efficient.

Weaknesses:

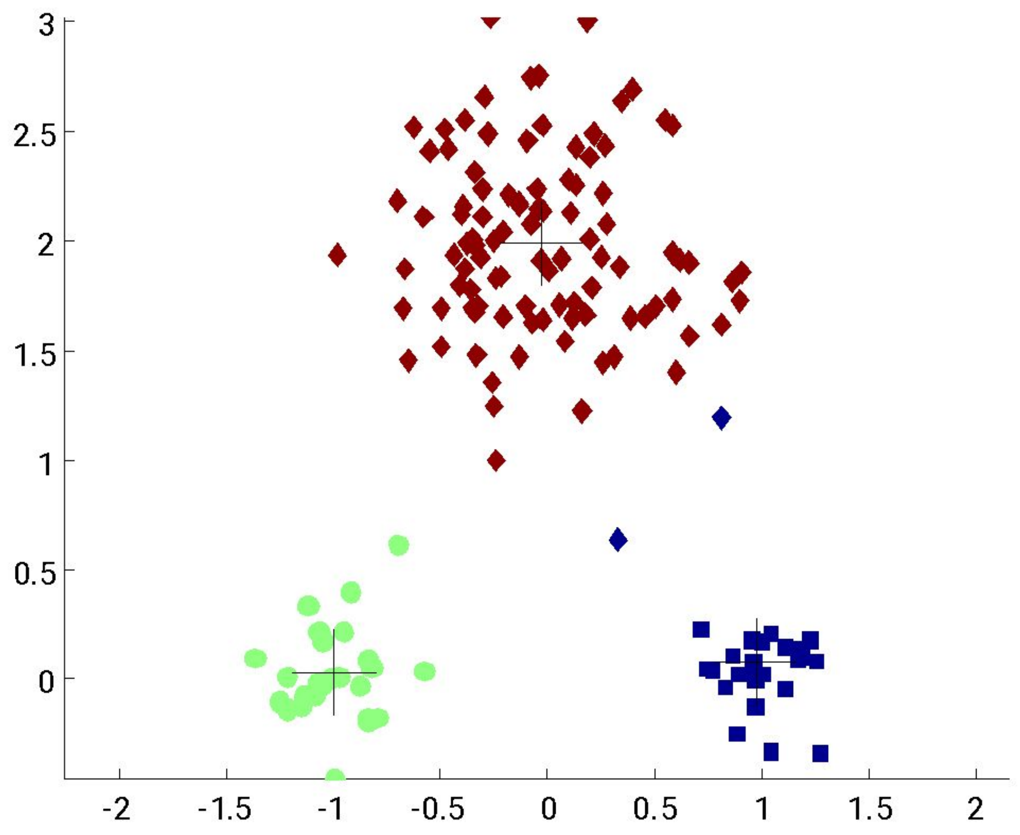
1. Clusters with different sizes and densities.
2. Non-spherical clusters.
3. Sensitive to initial centroids.

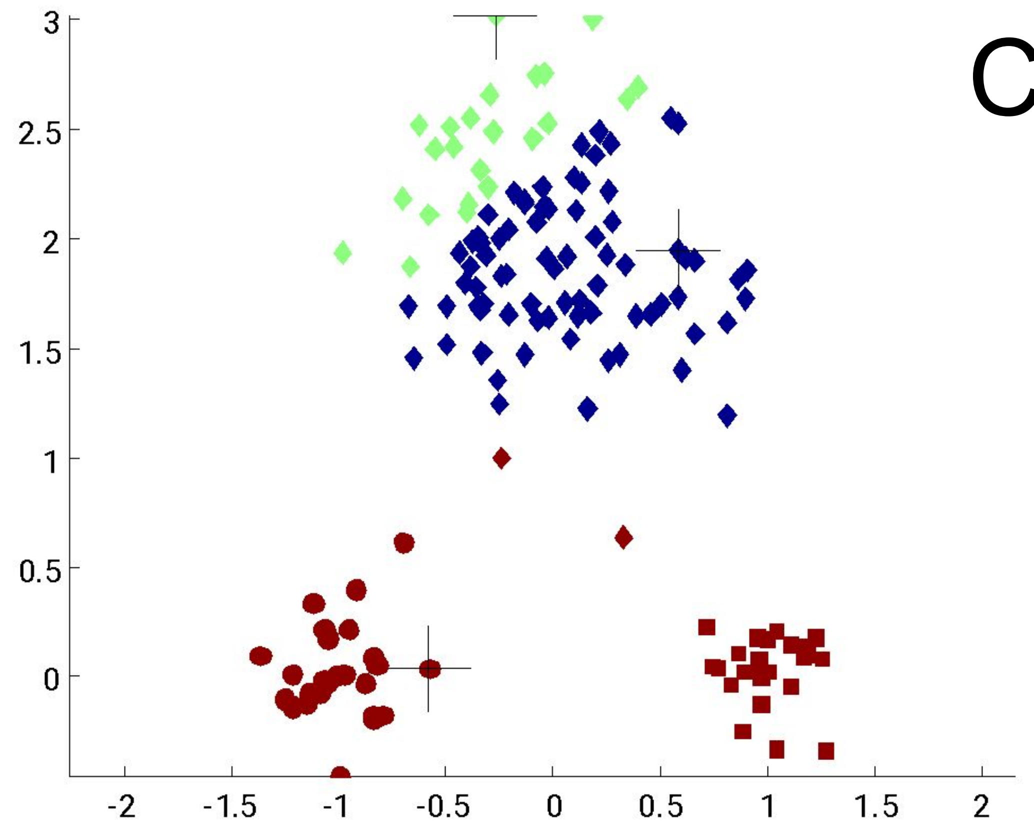
Question: How can we deal with the initialization problem?

K-Means: Strength and Limitations

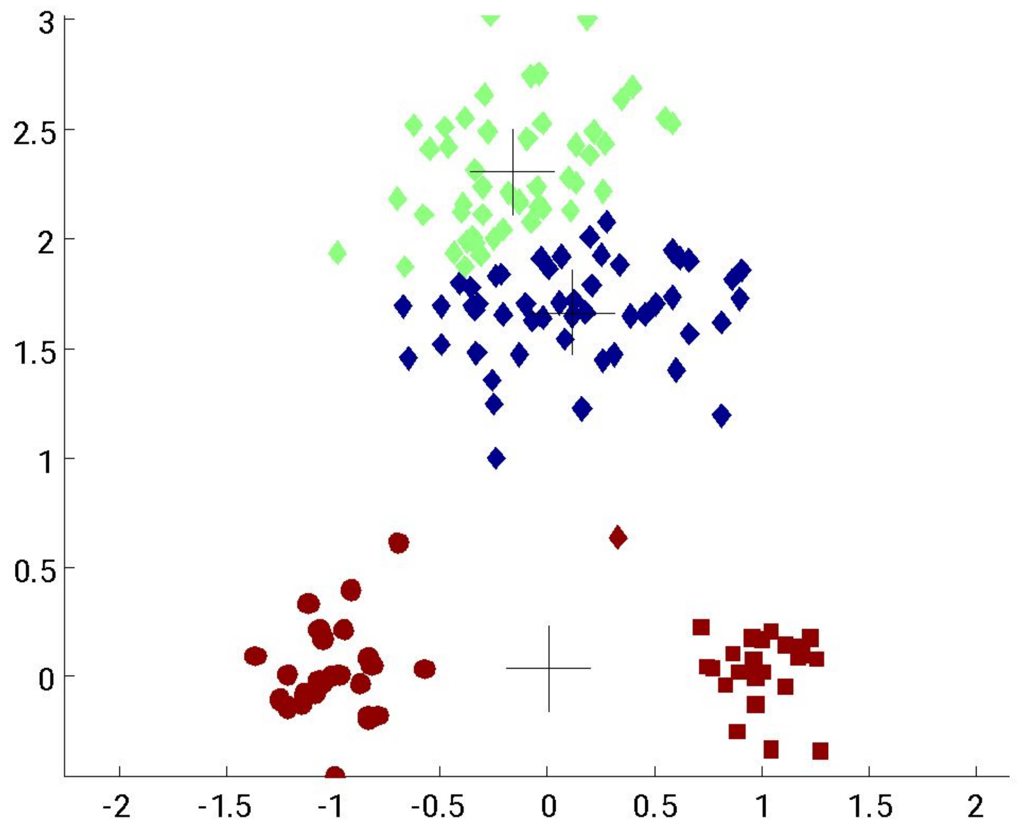


Converge in 6 iterations





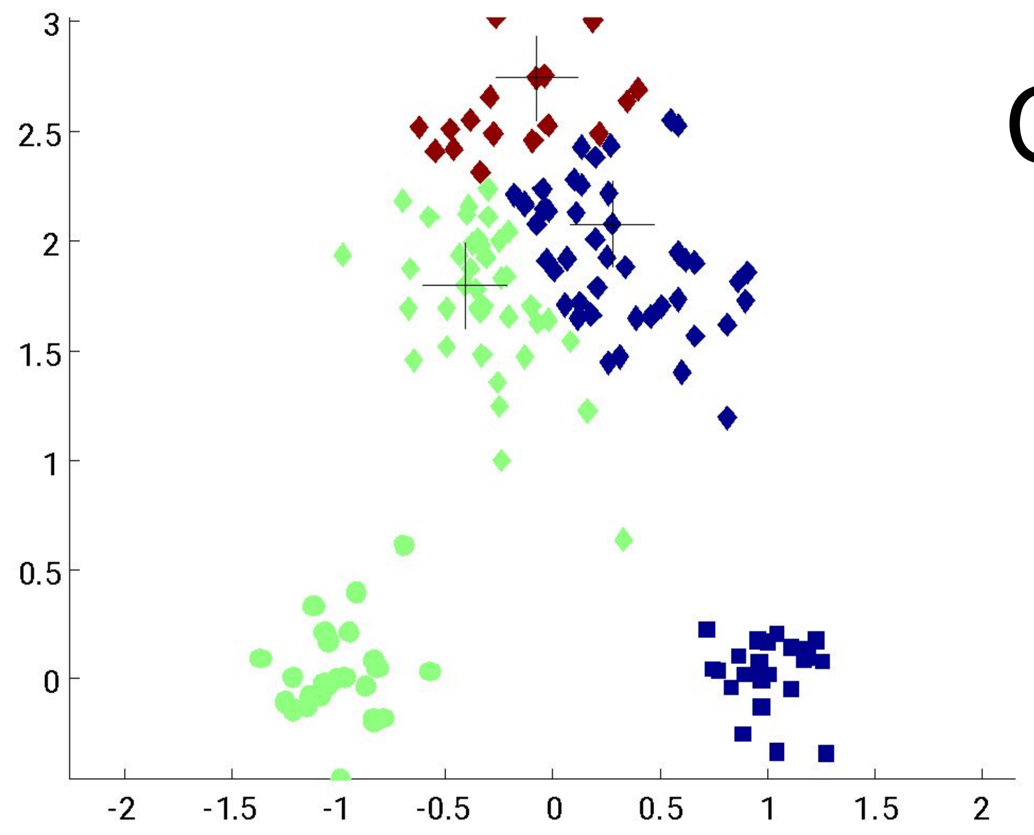
Converge in 5 iterations



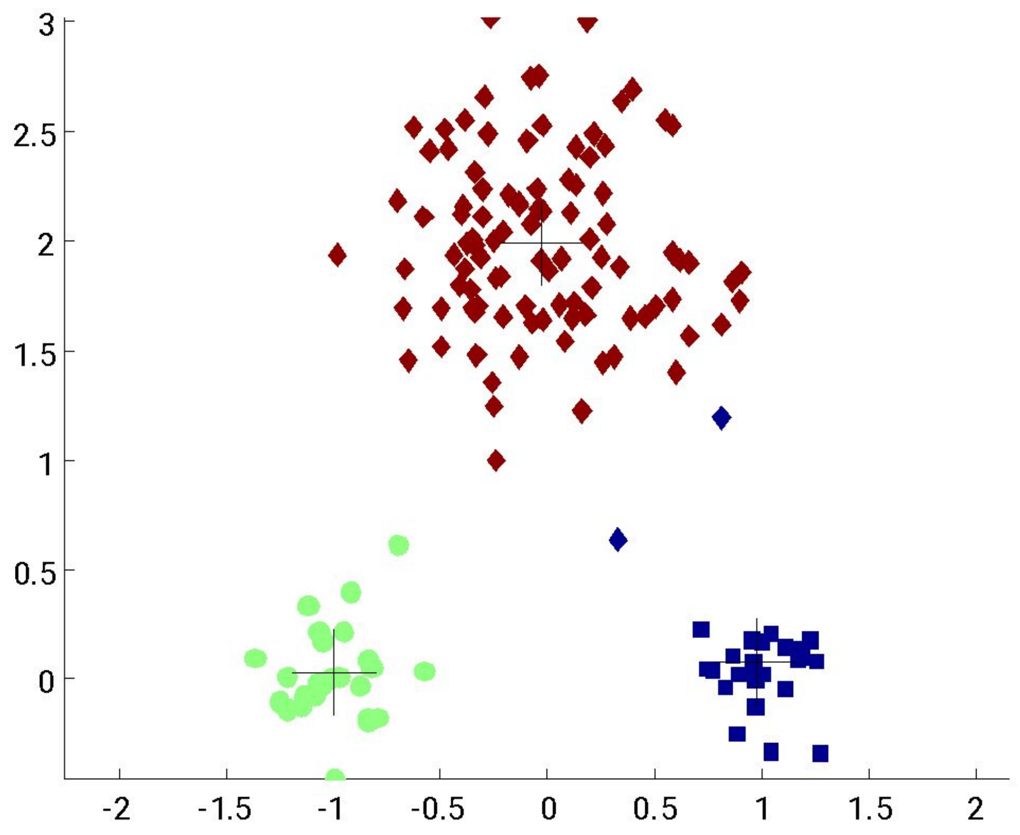
Question: How can we deal with the initialization problem?

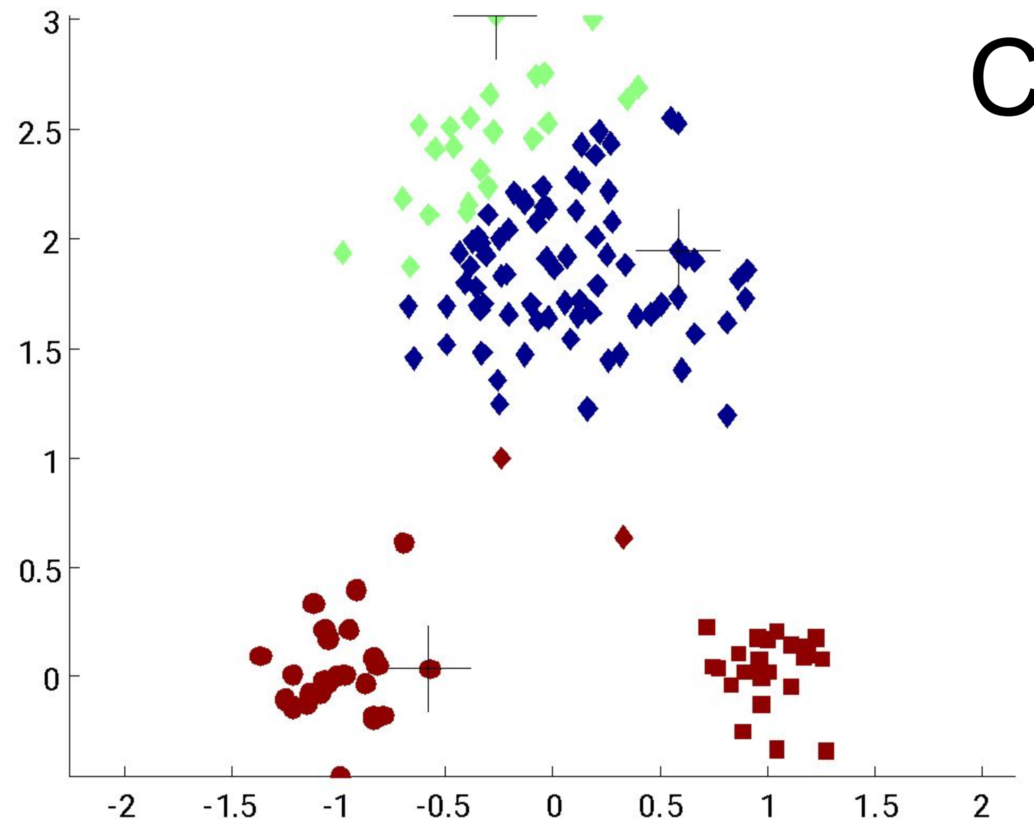
1. Multi-start with best result.

K-Means: Strength and Limitations

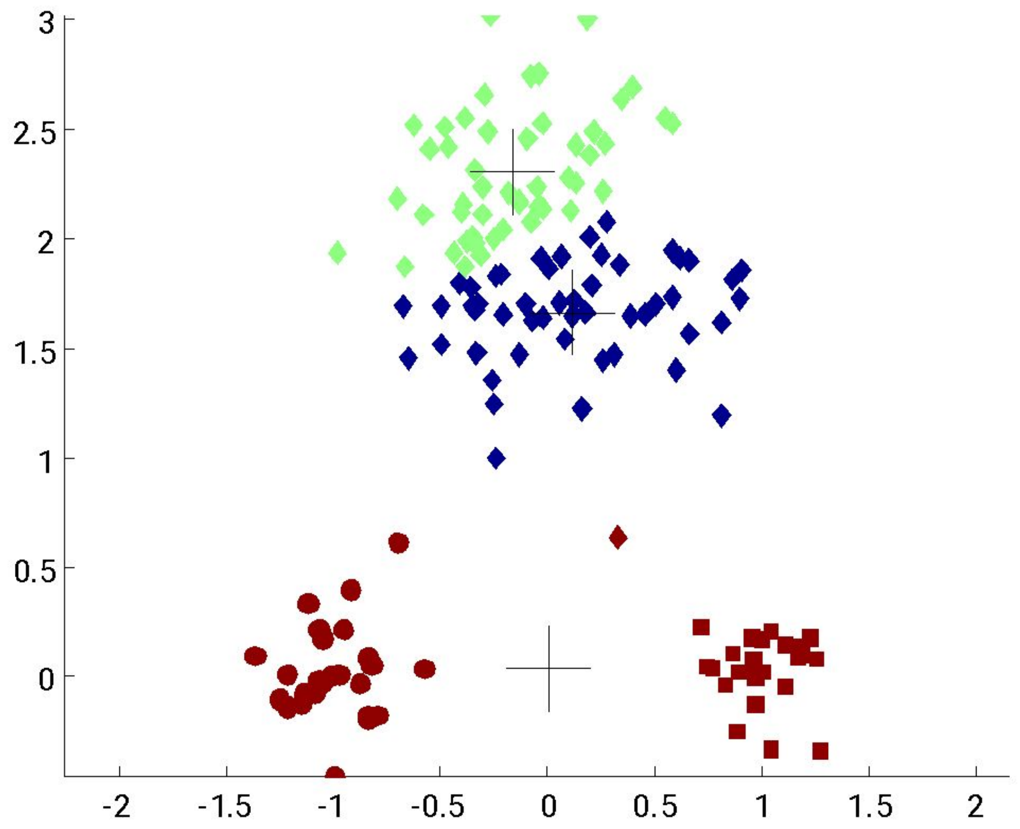


Converge in 6 iterations





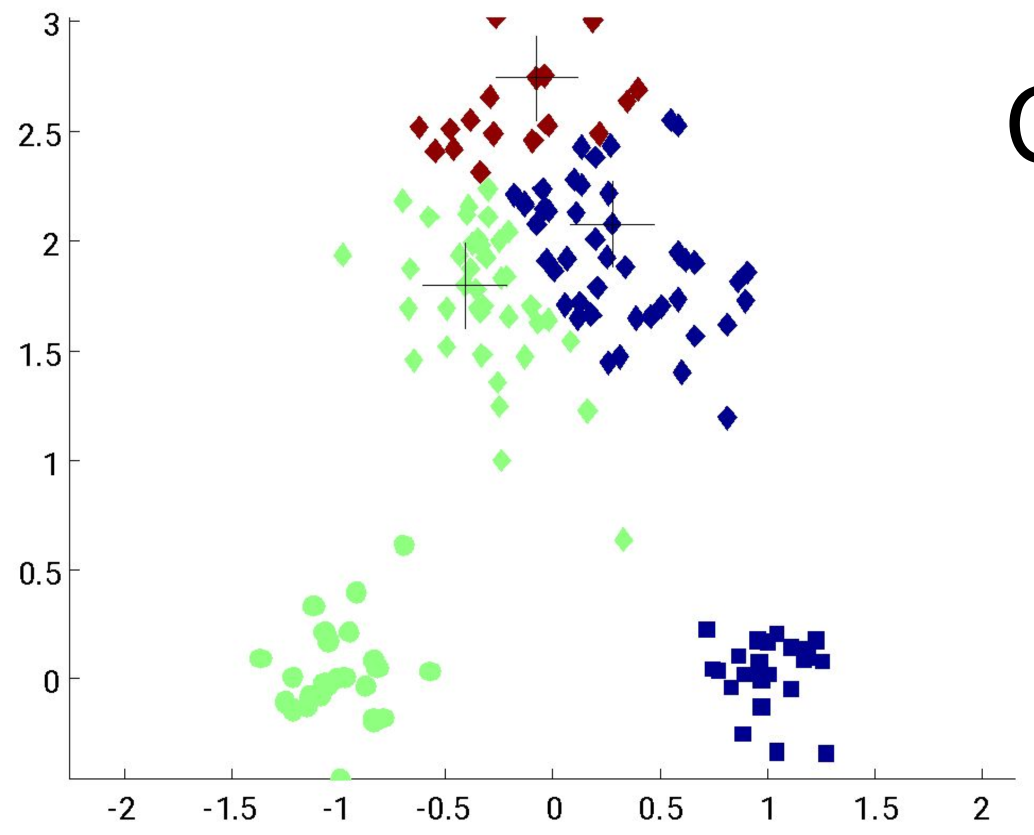
Converge in 5 iterations



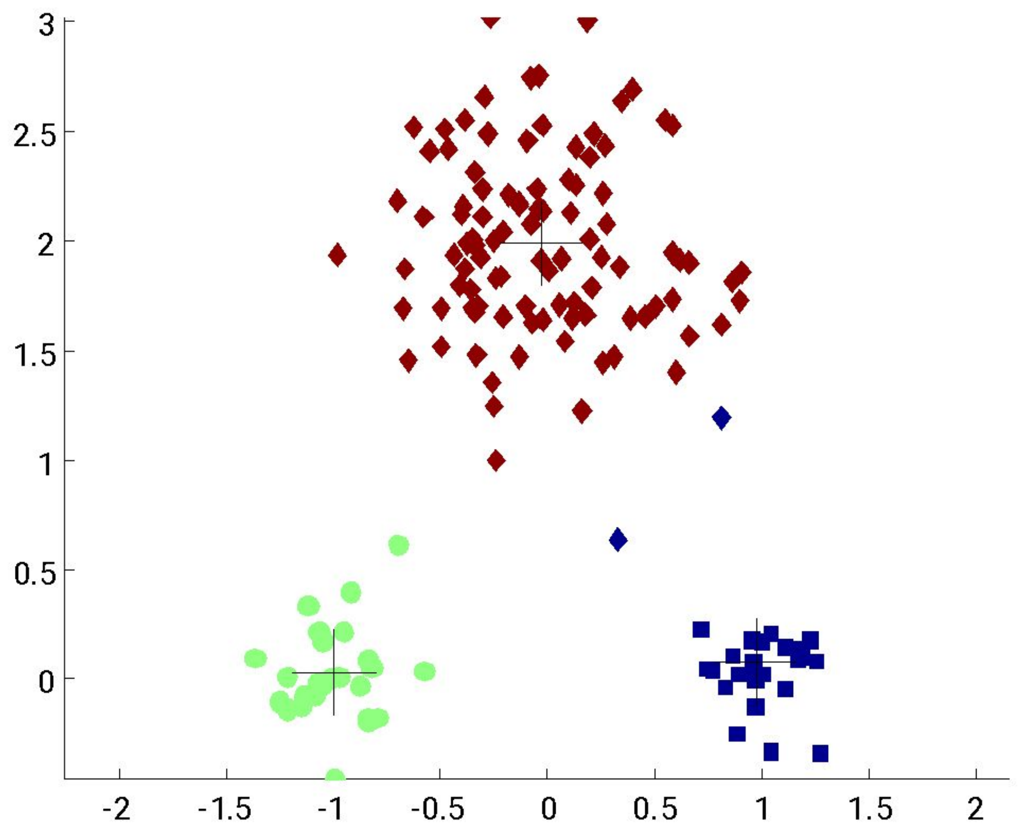
Question: How can we deal with the initialization problem?

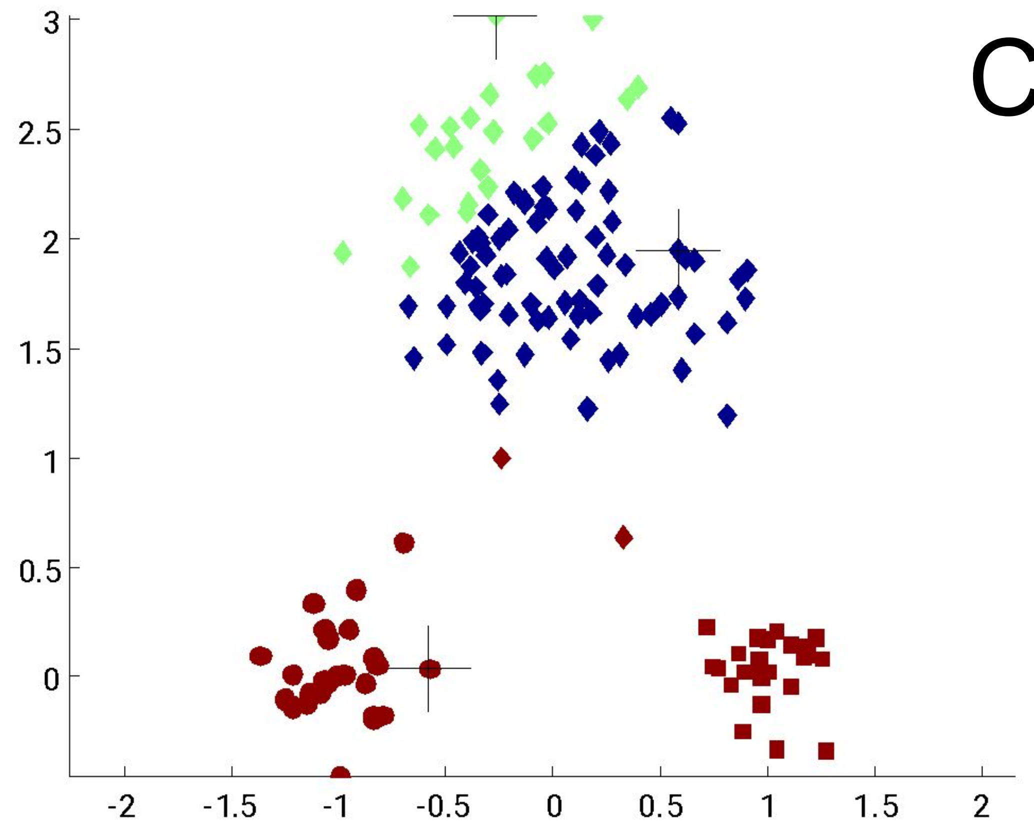
1. Multi-start with best result.
2. Heuristic for initial centers selection: K-Means++

K-Means: Strength and Limitations

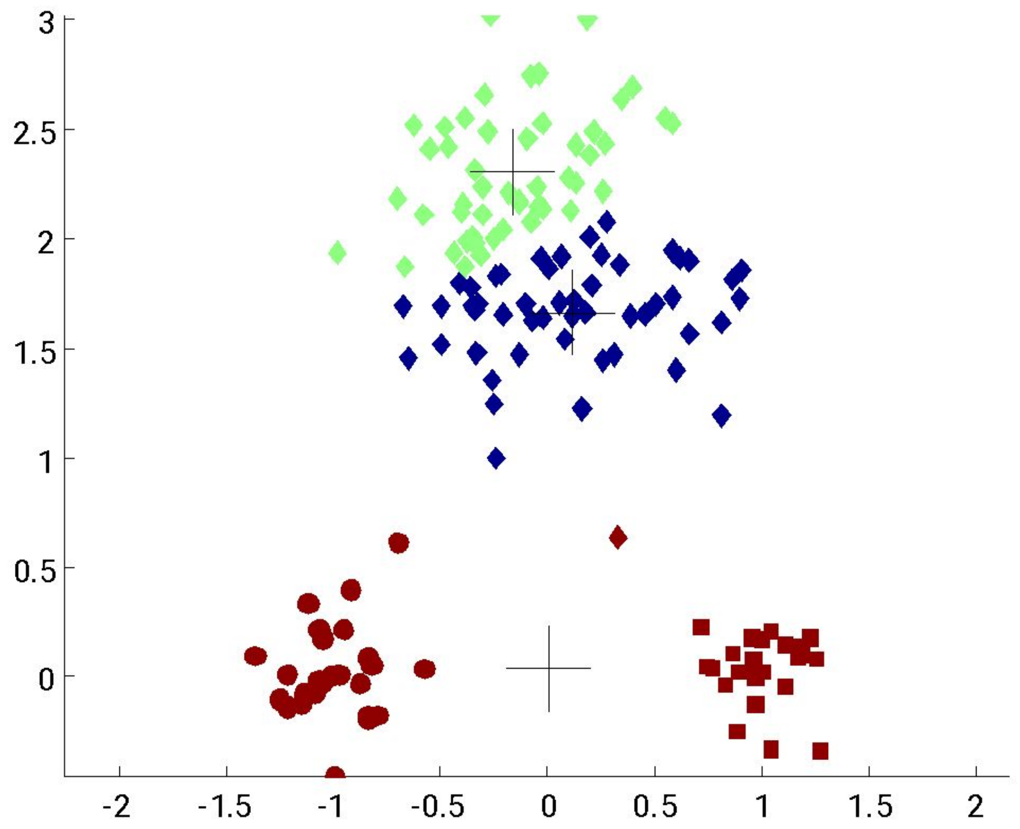


Converge in 6 iterations





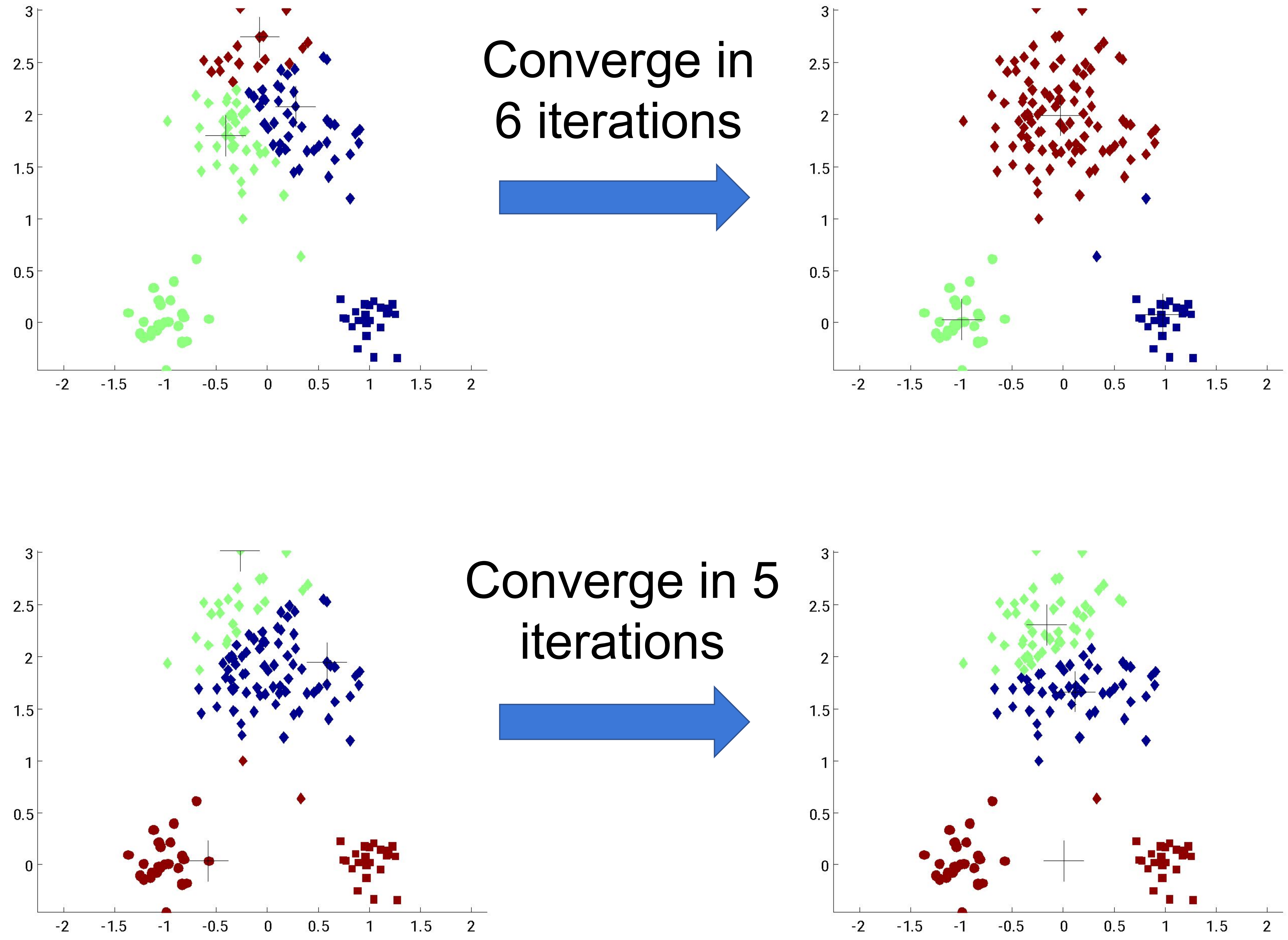
Converge in 5 iterations



Question: How can we deal with the initialization problem?

1. Multi-start with best result.
2. Heuristic for initial centers selection: K-Means++
3. Algorithm invariant to initial selection: Bisecting K-Means

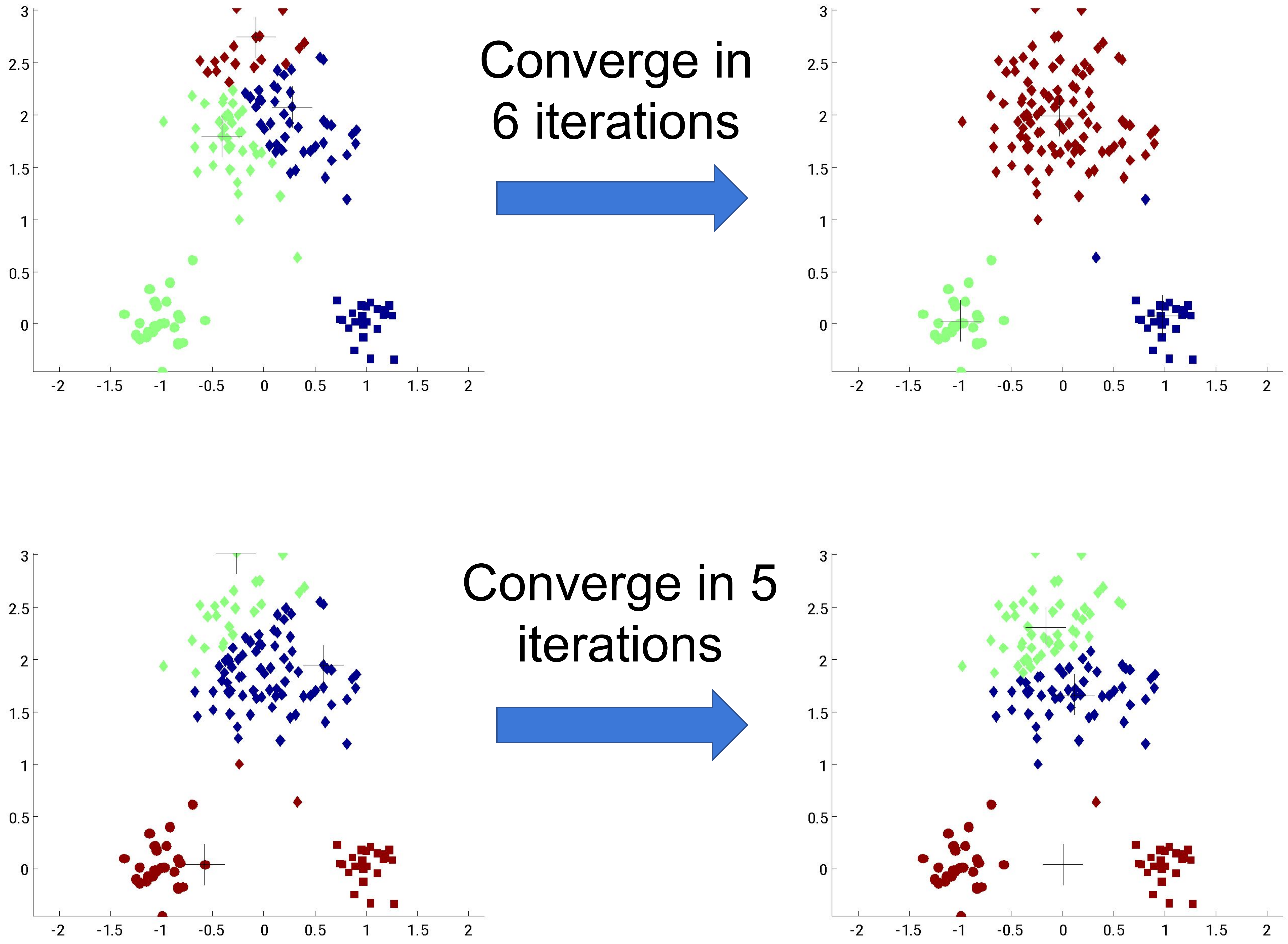
K-Means: Strength and Limitations



Question: How can we deal with the initialization problem?

1. Multi-start with best result.
2. Heuristic for initial centers selection: K-Means++
3. Algorithm invariant to initial selection: Bisecting K-Means
4. Post processing on clusters

K-Means: Strength and Limitations



Question: How can we deal with the initialization problem?

1. Multi-start with best result.
2. Heuristic for initial centers selection: K-Means++
3. Algorithm invariant to initial selection: Bisecting K-Means
4. Post processing on clusters
5. Global optimization

Outline

- Introduction to Clustering
- K-Means
  - K-Means Algorithm
  - Limitation of K-Means
  - K-Means Implementation
- Agglomerative Clustering

K-Means API

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
```

```
X, y = make_blobs(centers=4, random_state=1)
```

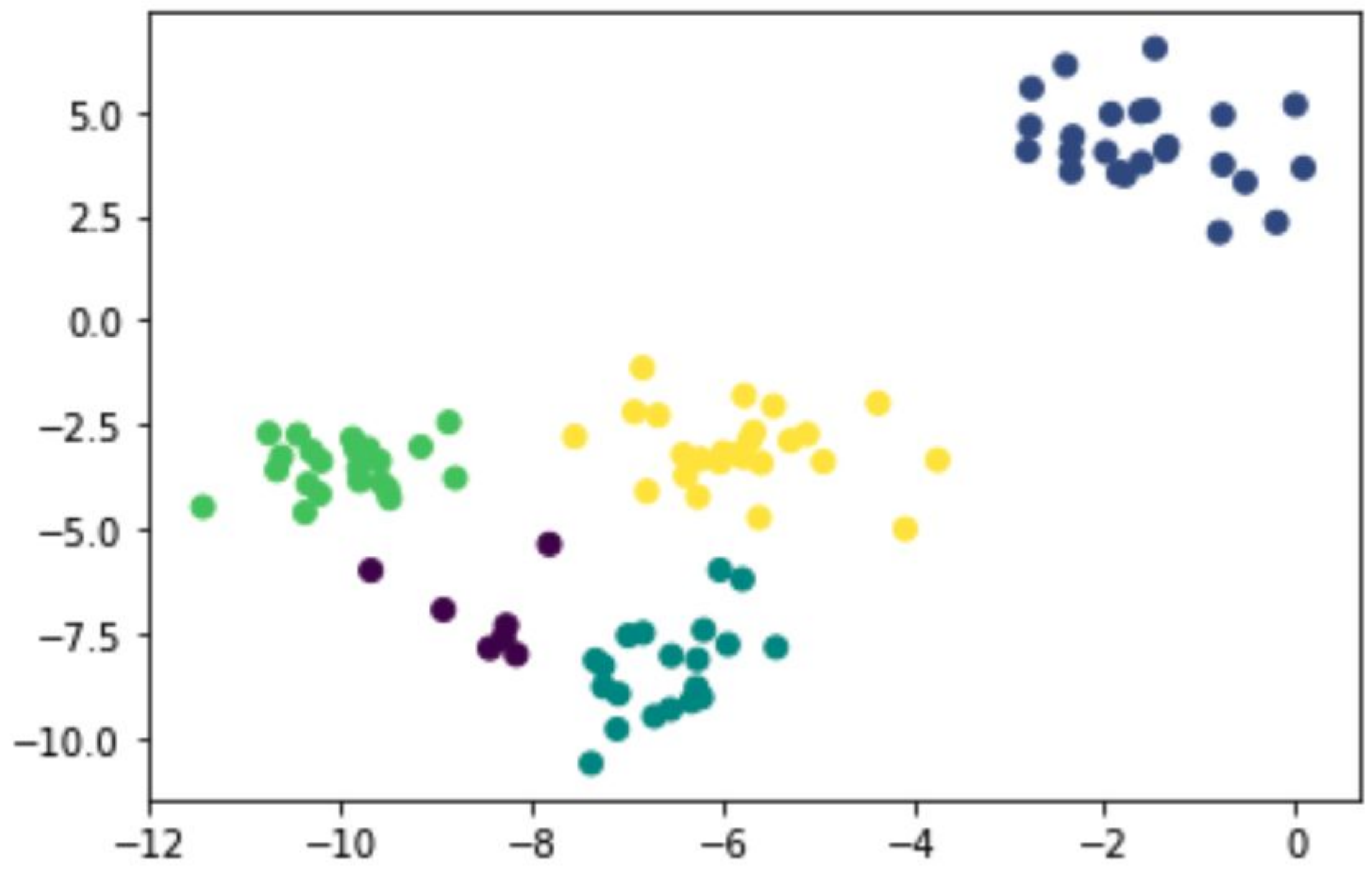
```
km = KMeans(n_clusters=5, random_state=0)
km.fit(X)
print(km.cluster_centers_.shape)
print(km.labels_.shape)
```

```
(5, 2)
(100,)
```

```
# predict is the same as labels_ on training data
# but can be applied to new data
print(km.predict(X).shape)
```

```
(100,)
```

```
plt.scatter(X[:, 0], X[:, 1], c=km.labels_)
```



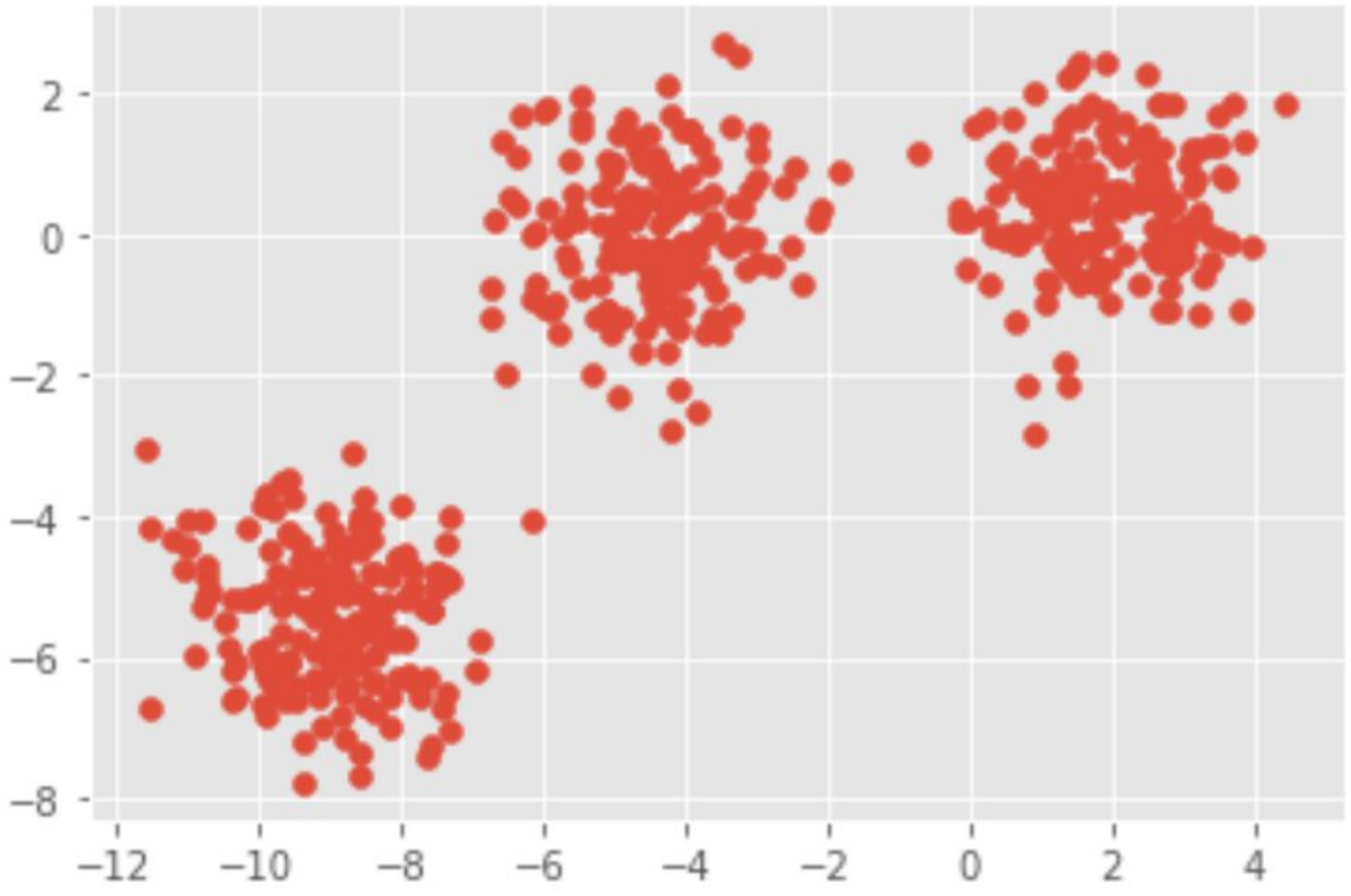
Initialization

- Random centers fast
- K-means++ (default): Greedily add ‘furthest way’ point
- By default K-means in sklearn does 10 random restarts with different initializations
- K-means++ initialization may take much longer than clustering

K-Means Application

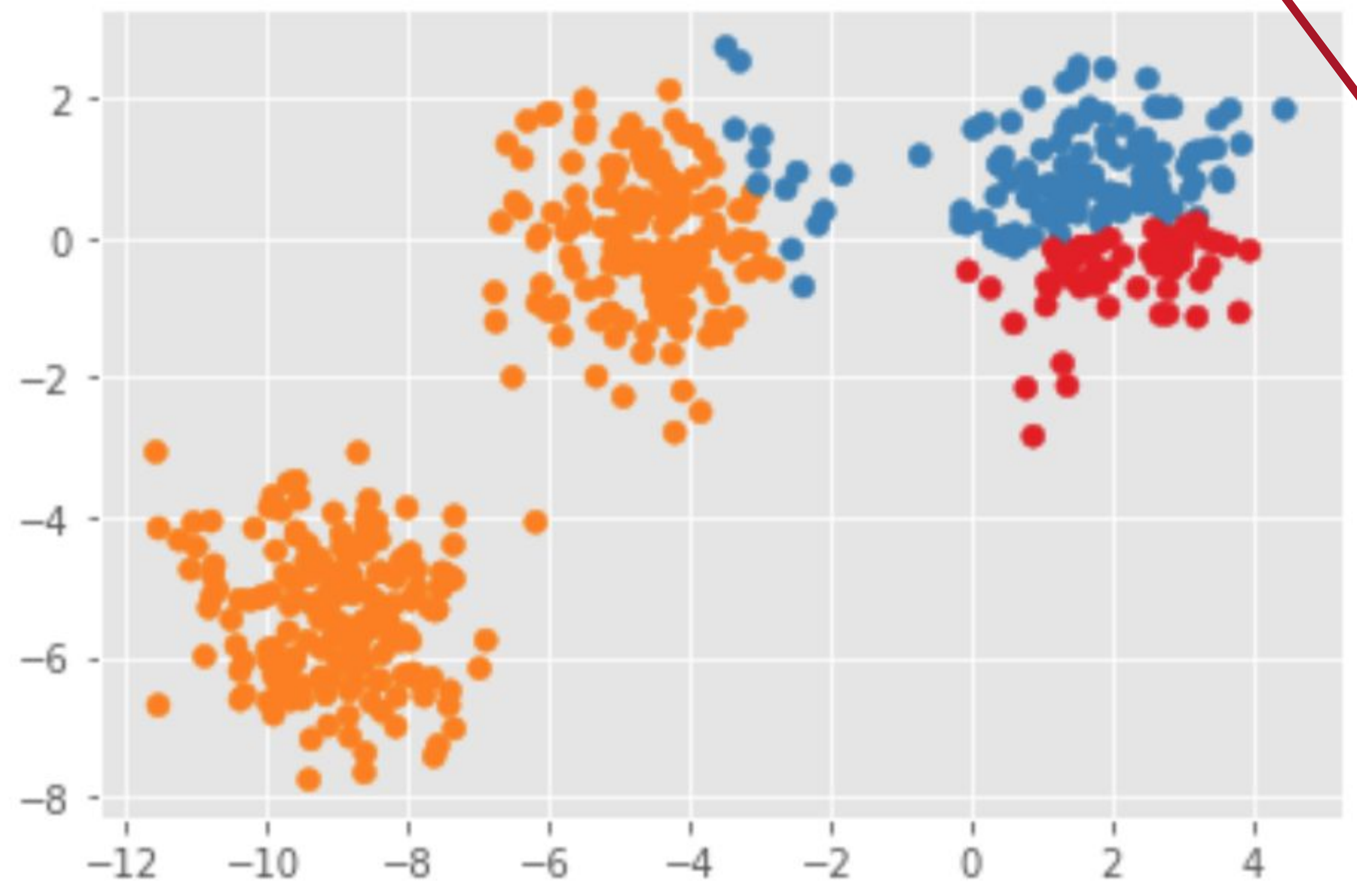
```python
# make_blobs generates gaussian blobs, we create 3 blobs
n_samples = 500
random_state = 170
X, y = make_blobs(n_samples=n_samples, centers=3, random_state=random_state)

# plot data
plt.scatter(X[:, 0], X[:, 1],  marker="o");
```



K-Means Application

```
y_pred = KMeans(n_clusters=3, n_init=1, init='random', max_iter=1).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred,  marker="o", cmap=cmap);
```

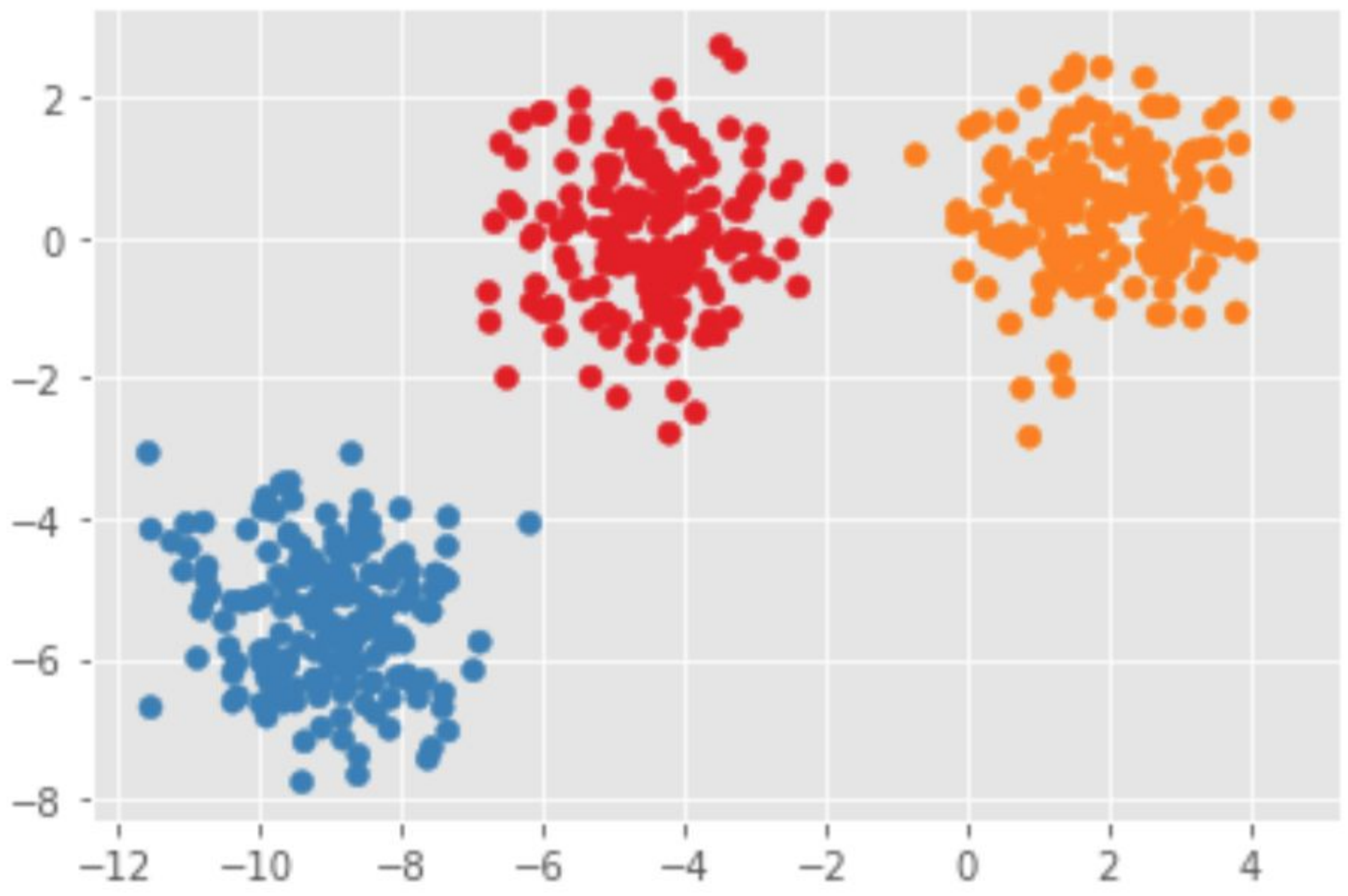


- random
- k-means++ by default

- Number of iterations in each initialization

- Number of initialization
- 10 by default

```
y_pred = KMeans(n_clusters=3, n_init=1, init='random', max_iter=5).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred,  marker="o", cmap=cmap);
```

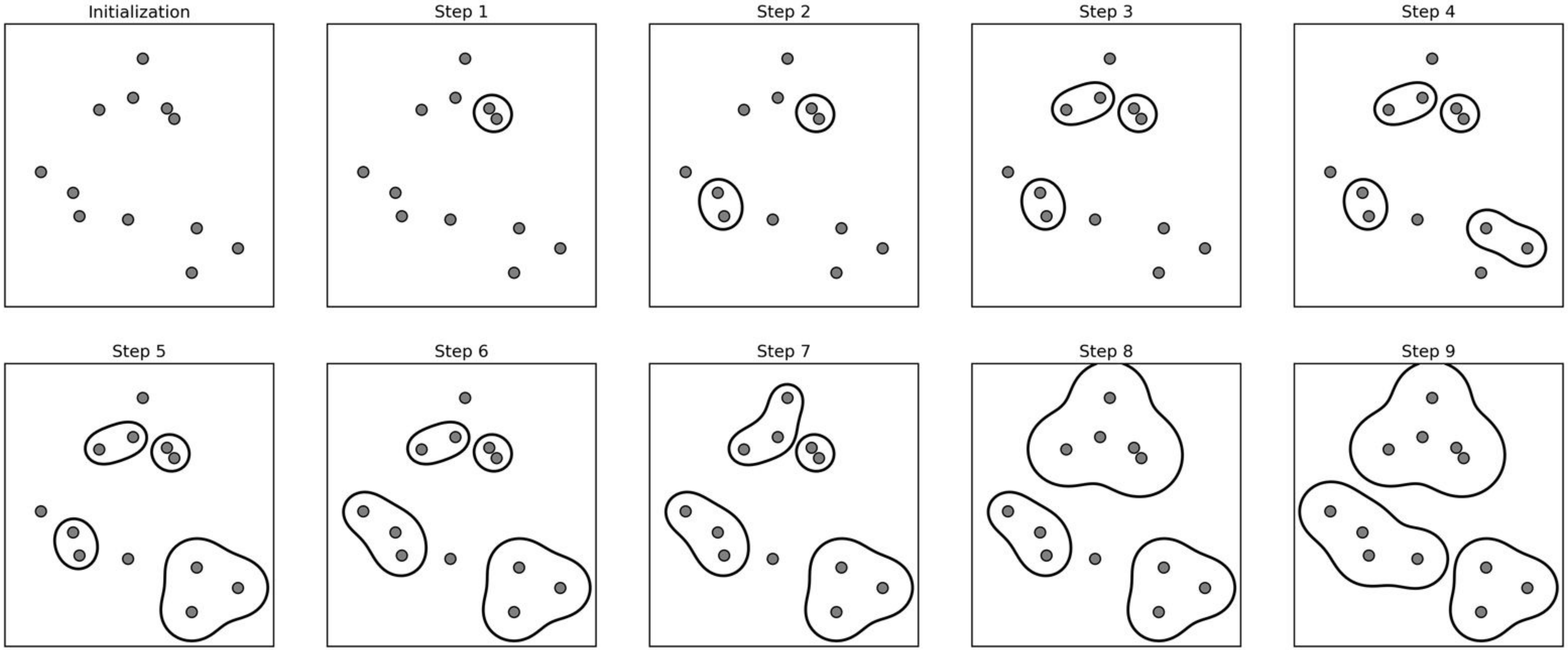


Outline

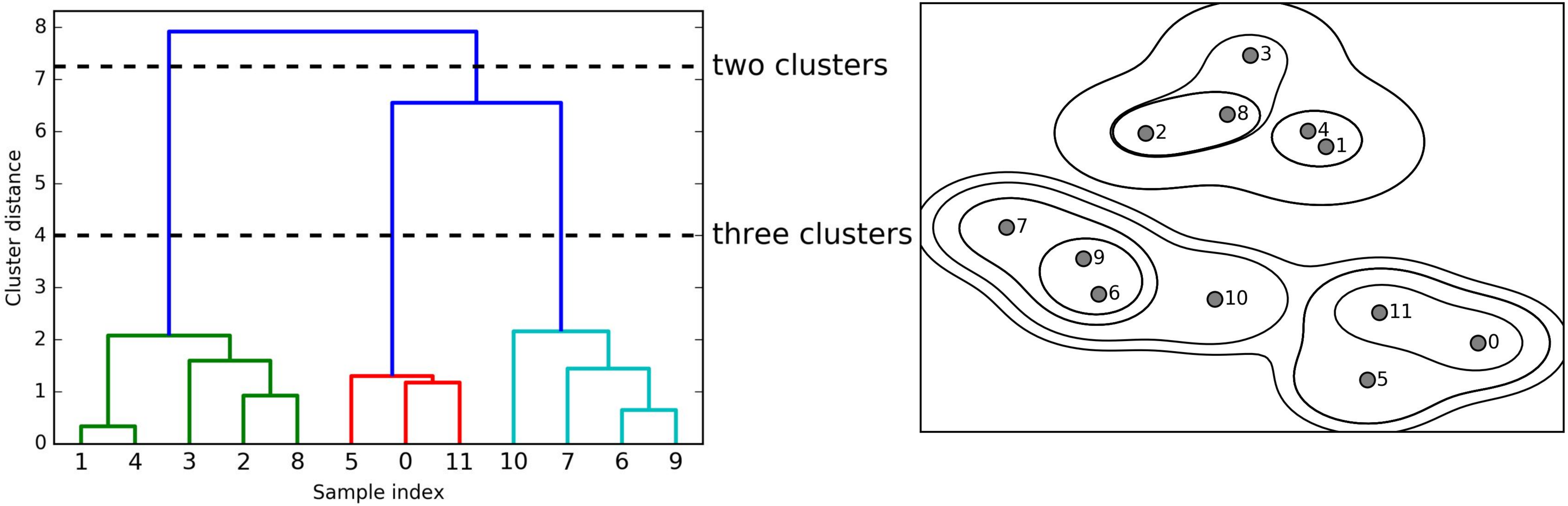
- Introduction to Clustering
- K-Means
  - K-Means Algorithm
  - Limitation of K-Means
  - K-Means Implementation
- Agglomerative Clustering

Agglomerative Clustering

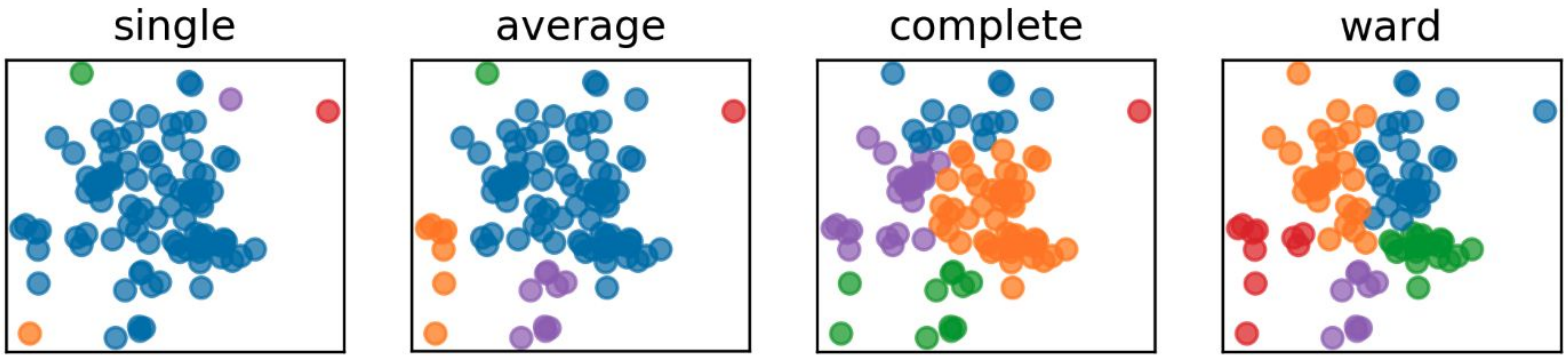
- Start with all points in their own cluster
- Greedily merge the two most similar clusters



Dendrograms



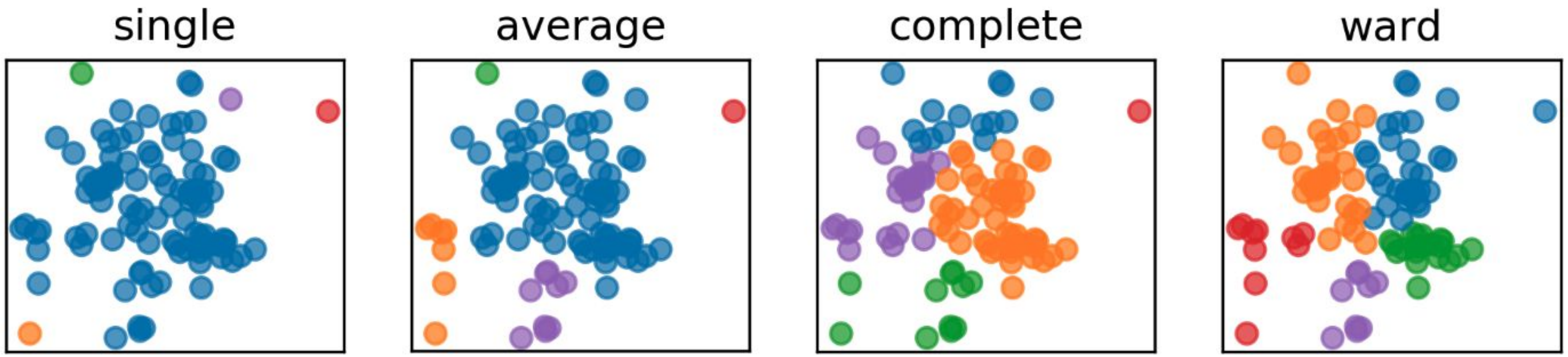
Merging (Linkage) Criteria




```
single :    [96  1  1  1  1]
average :   [82  9  7  1  1]
complete :  [50 24 14 11  1]
ward :      [31 30 20 10  9]
```

- Single Linkage
  - Smallest minimum distance

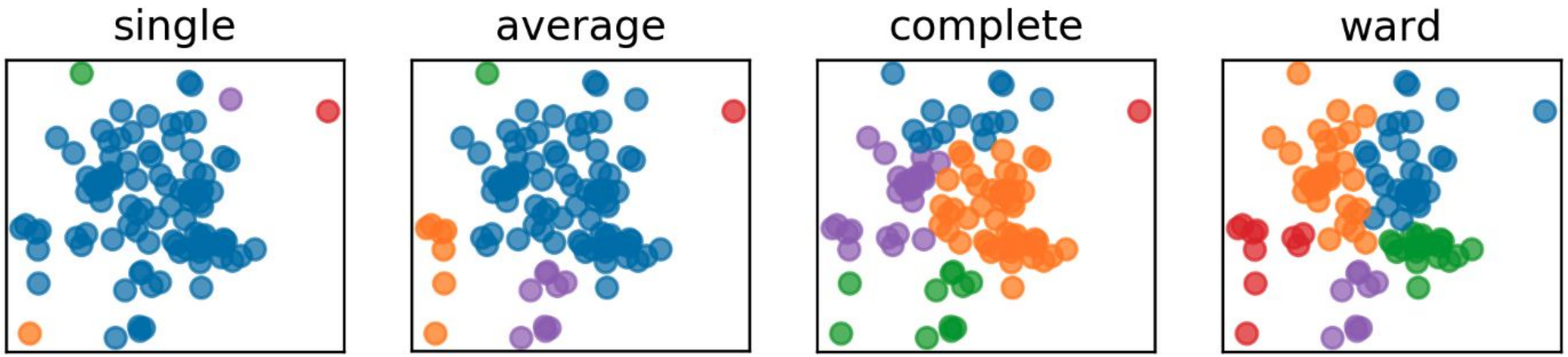
Merging (Linkage) Criteria




```
single :    [96  1  1  1  1]
average :   [82  9  7  1  1]
complete :  [50 24 14 11  1]
ward :      [31 30 20 10  9]
```

- Single Linkage
  - Smallest minimum distance
- Average Linkage
  - Smallest average distance between all pairs in the clusters

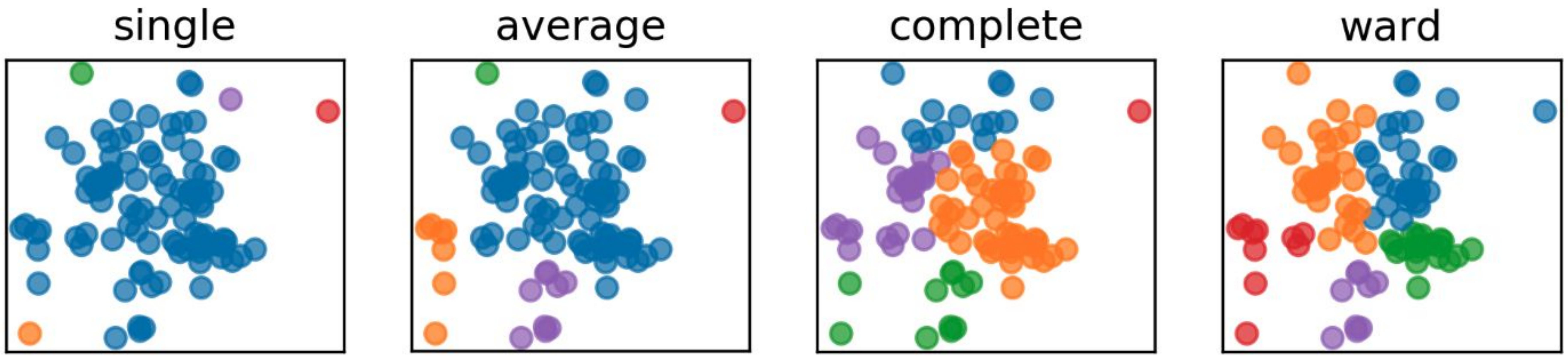
Merging (Linkage) Criteria




```
single :    [96  1  1  1  1]
average :   [82  9  7  1  1]
complete :  [50 24 14 11  1]
ward :      [31 30 20 10  9]
```

- Single Linkage
  - Smallest minimum distance
- Average Linkage
  - Smallest average distance between all pairs in the clusters
- Complete Linkage
  - Smallest maximum distance

Merging (Linkage) Criteria




```
single :     [96  1  1  1  1]
average :    [82  9  7  1  1]
complete :   [50 24 14 11  1]
ward :       [31 30 20 10  9]
```

- Single Linkage
  - Smallest minimum distance
- Average Linkage
  - Smallest average distance between all pairs in the clusters
- Complete Linkage
  - Smallest maximum distance
- Ward (default in sklearn)
  - Smallest increase in within-cluster variance
  - Leads to more equally sized clusters.

Summary

- KMeans
  - Classic, simple and efficient.
  - Algorithm with iterative update of cluster centers.
  - Failed on cluster with non-globularity, various density and size.
  - Issues with the initialization of centers
- Agglomerative
  - Do not need to specify the number of clusters
  - It is sensitive to noise and outliers
  - Time and space complexity is high so not suitable for large dataset