Clustering

Sandro Cumani

sandro.cumani@polito.it

Politecnico di Torino

Clustering

Partition a set of input vector into homogeneous groups

Allows identifying subsets that share common characteristics

Unsupervised — we do not have data labels

Rather, we want to assign a label to each sample (we are partitioning the data)

- Similar samples will have the same label (same group, or cluster)
- Similar samples should present common patterns
- Different samples will have a different label

Clustering

Data exploration — we want to infer some structure of our data

- Group documents according to topic similarity, without knowing the possible topics in advance
- Group images that contain similar objects
- Image segmentation — find regions of an image that are "similar"

Clustering

Clustering can also be employed as a pre-processing step for classification

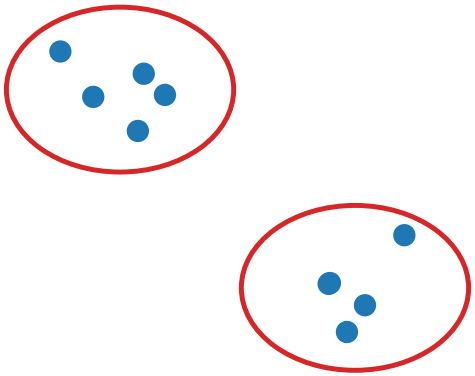
Example: biometric classification

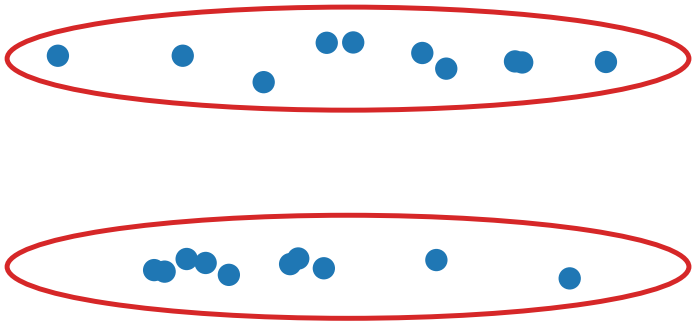
- Lots of unlabeled data available, labeling samples is expensive
- Build a bootstrap classifier using labeled data
- Define similarity based on the classification rule
- Cluster the unlabeled samples
- Train a new classifier using the pseudo-labels provided by clustering

Clustering

We want to group together similar samples

How do we define similar?

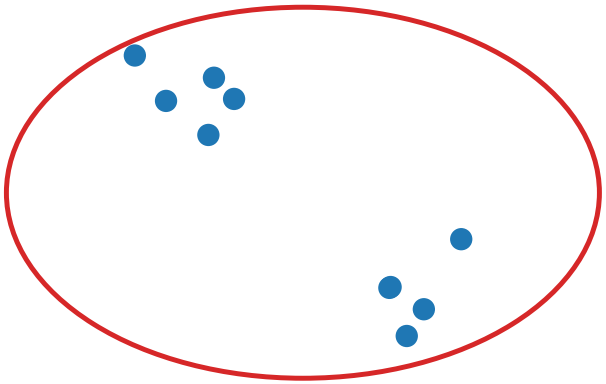


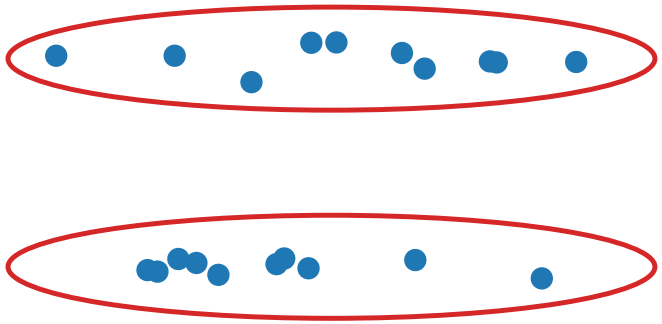


Clustering

We want to group together similar samples

How do we define similar?

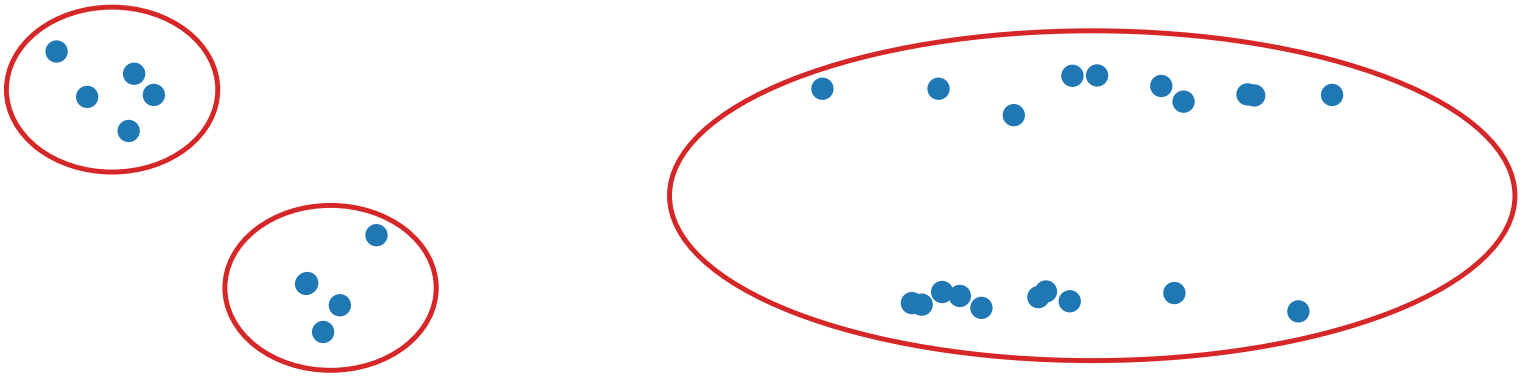




Clustering

We want to group together similar samples

How do we define similar?



Clustering

We want to group together similar samples

How do we define similar?

- Geometrical similarity: Euclidean distance
- Points are similar if they have small Euclidean distance

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$$

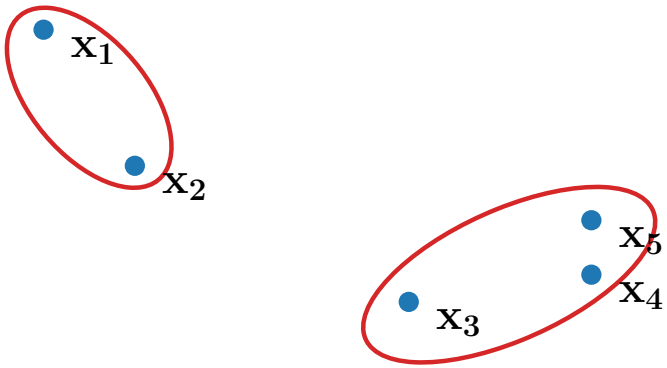
Different measures of similarity (or distance) may lead to different results

Partitional Clustering

Divide the data into a given number K of partitions

Each object belongs to a single partition

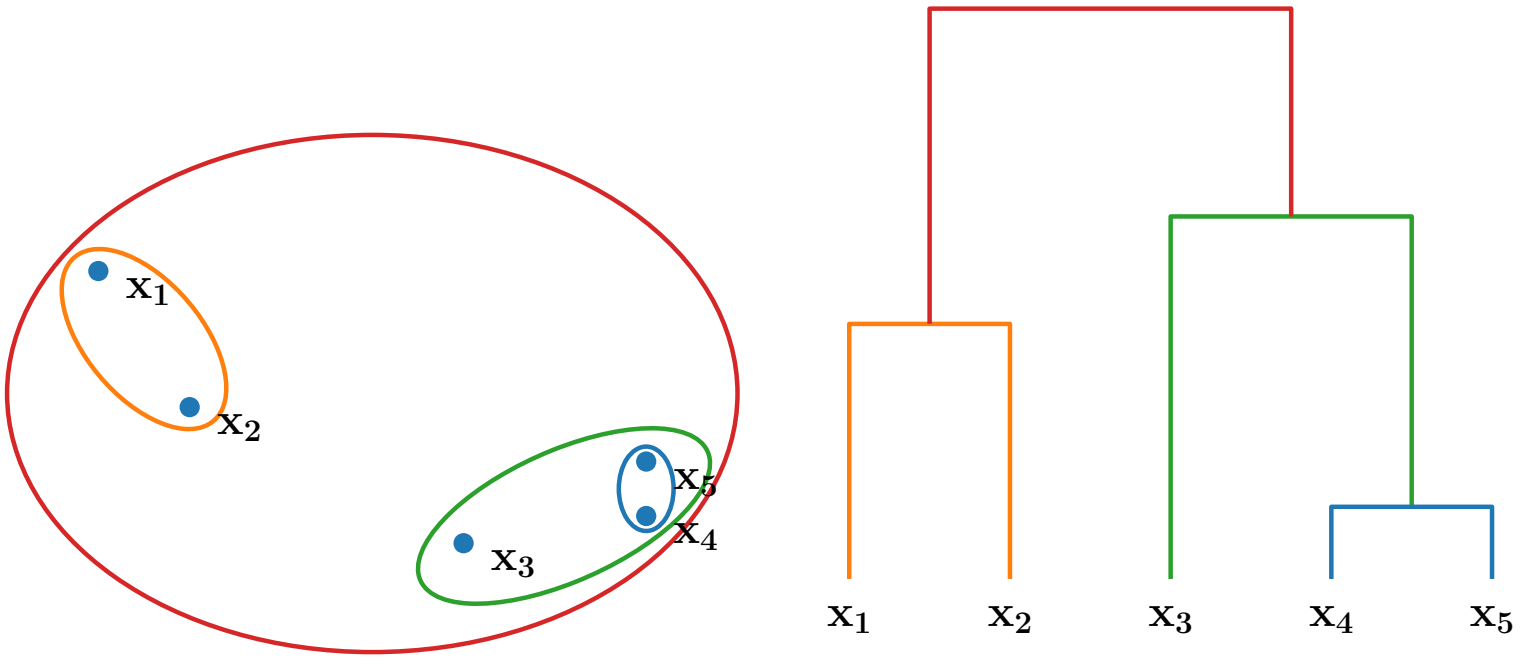
- K–Means
- Gaussian Mixture Models
- Spectral clustering



Hierarchical Clustering

Build a hierarchy of nested sub-clusters

- Bottom-up (agglomerative)
- Top-down (divisive)



K–Means

Partitional clustering algorithm

Requires defining the desired number of clusters K

Each cluster is represented by a centroid μ_c

We need to jointly identify optimal centroids and assignments of patterns to clusters

How do we define optimal?

K–Means

K–Means: minimize the average squared distance of each sample from the centroid of the corresponding cluster

As we will see, in practice we need to find centroids as to minimize the average squared distance of each point from the closest centroid

K–Means

Let $\boldsymbol{x}_1, \ldots \boldsymbol{x}_N \in \mathbb{R}^D$ be a set of N, D-dimensional, samples that we want to cluster

Let $\boldsymbol{\mu}_k \in \mathbb{R}^D, k = 1 \ldots K$ denote the cluster centroids

We encode cluster assignments by associating to each sample a K–dimensional vector $\boldsymbol{r}_i \in \mathbb{R}^K$:

- each element of $\boldsymbol{r}_i$ is 0, except for a single element, whose value is 1

$$\boldsymbol{r}_i = \left[0, \ldots 0, 1, 0, \ldots 0\right]$$

- $\boldsymbol{x}_i$ belongs to cluster $j \iff \boldsymbol{r}_{i,j} = 1$

i.e., the position of the non-zero element of $\boldsymbol{r}_i$ corresponds to the numerical index of the cluster for sample $\boldsymbol{x}_i$

$\boldsymbol{r}_i$ is also called a 1-of-K encoding

K–Means

Our objective is to minimize the (average) squared distance of samples from the corresponding centroids (also called distortion)

$$J(\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_K, \boldsymbol{r}_1 \dots \boldsymbol{r}_N) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{r}_{i,k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

Direct minimization of J is hard

The K–Means algorithm proceeds iteratively from an initial set of centroids:

1. Given centroids $\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_K$, minimize J with respect to cluster assignments $\boldsymbol{r}_1 \dots \boldsymbol{r}_N$
2. Given cluster assignments $\boldsymbol{r}_1 \dots \boldsymbol{r}_N$, minimized J with respect to centroids $\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_K$
3. Iterate from 1. until convergence (J does not change)

Minimize J w.r.t. cluster assignments $\boldsymbol{r}_1 \ldots \boldsymbol{r}_N$:

- Each $\boldsymbol{r}_i$ appears only once in the external sum, and the function is a linear combination of terms $\boldsymbol{r}_i$
- Minimization of J w.r.t. all $\boldsymbol{r}_i$'s is obtained by independently minimizing each term

$$\sum_{k=1}^{K} \boldsymbol{r}_{i,k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

- Since $\boldsymbol{r}_i$ must contains a single non-zero element (1-of-K encoding), the optimal solution consists in setting $\boldsymbol{r}_{i,k} = 1$ for cluster k whose centroid is closest to $\boldsymbol{x}_i$

$$\boldsymbol{r}_{i,k} = 1 \iff k = \arg\min_k \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

i.e., we assign the each sample to the closest centroid

K–Means

Minimize J w.r.t. centroids $\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_K$:

- The function is quadratic in each $\boldsymbol{\mu}_k$
- We can simply set the gradient of J w.r.t each $\boldsymbol{\mu}_k$ equal to 0:

$$\nabla_{\boldsymbol{\mu}_k} J = 2 \sum_{i=1}^{N} \boldsymbol{r}_{i,k} (\boldsymbol{\mu}_k - \boldsymbol{x}_i) = 0$$

- The optimal solution is then given by

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \boldsymbol{r}_{i,k} \boldsymbol{x}_i}{\sum_{i=1}^{N} \boldsymbol{r}_{i,k}}$$

- The numerator is the sum of samples assigned to cluster k
- The denominator is the number of samples assigned to cluster k
- $\boldsymbol{\mu}_k$ is the mean of the samples that are currently assigned to cluster k

K–Means

The K–Means algorithm proceeds iteratively from an initial set of centroids:

1. Assign each sample to the cluster whose centroid is closest
2. Recompute the centroids as the means of the sample that are associated to each cluster
3. Iterate from 1. until convergence

The algorithm converges after a finite number of steps:

- The possible cluster assignments $\boldsymbol{r}_i$ are finite
- Unless we are at convergence, at each step we reduce the value of J

In practice, we can stop the algorithm when cluster assignments do not change

K–Means

K–Means requires specifying an initial set of centroids

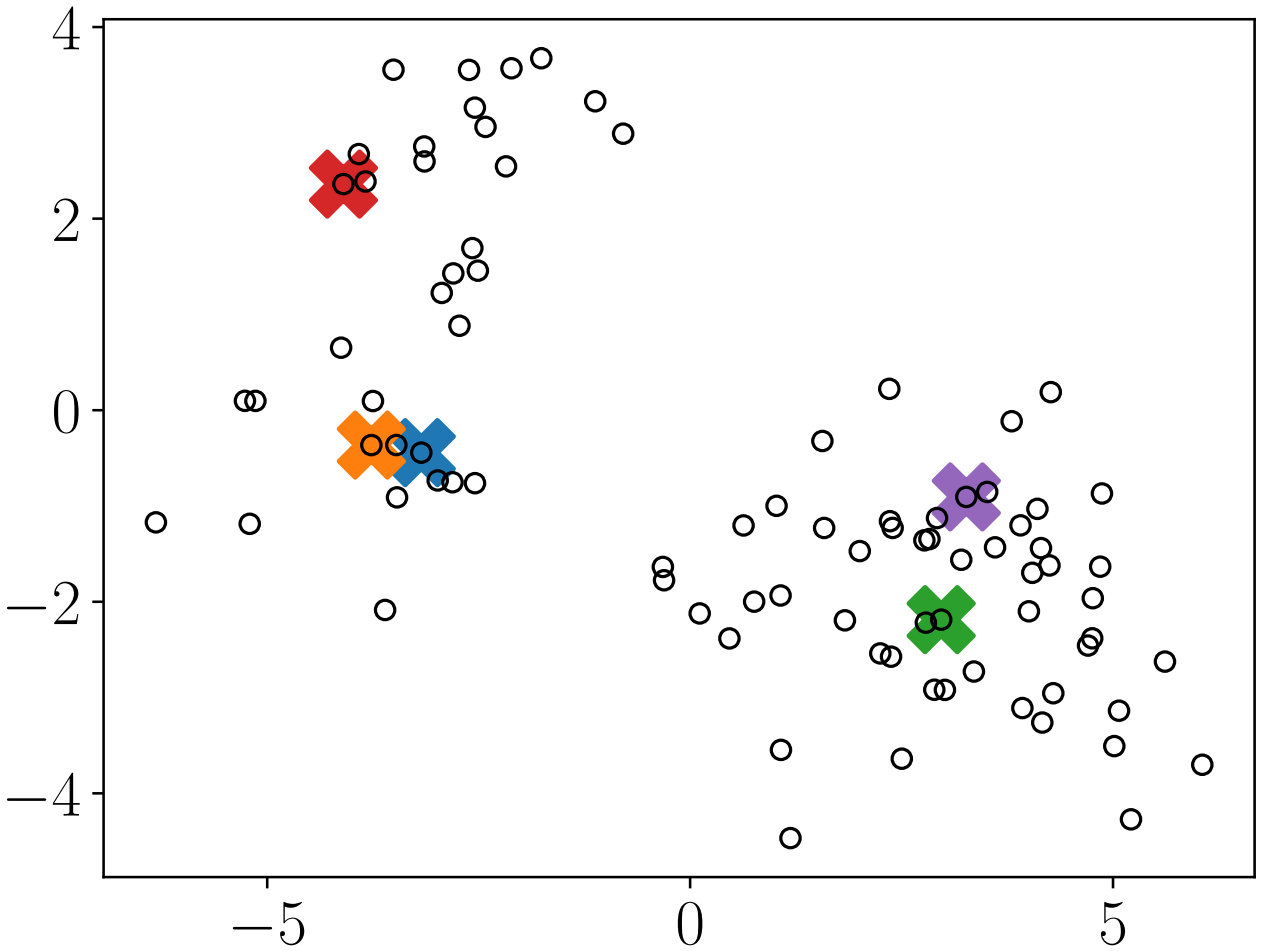
- Random
- K–Means++
- Other heuristics

Since J is in general non-convex, K–Means finds local optima

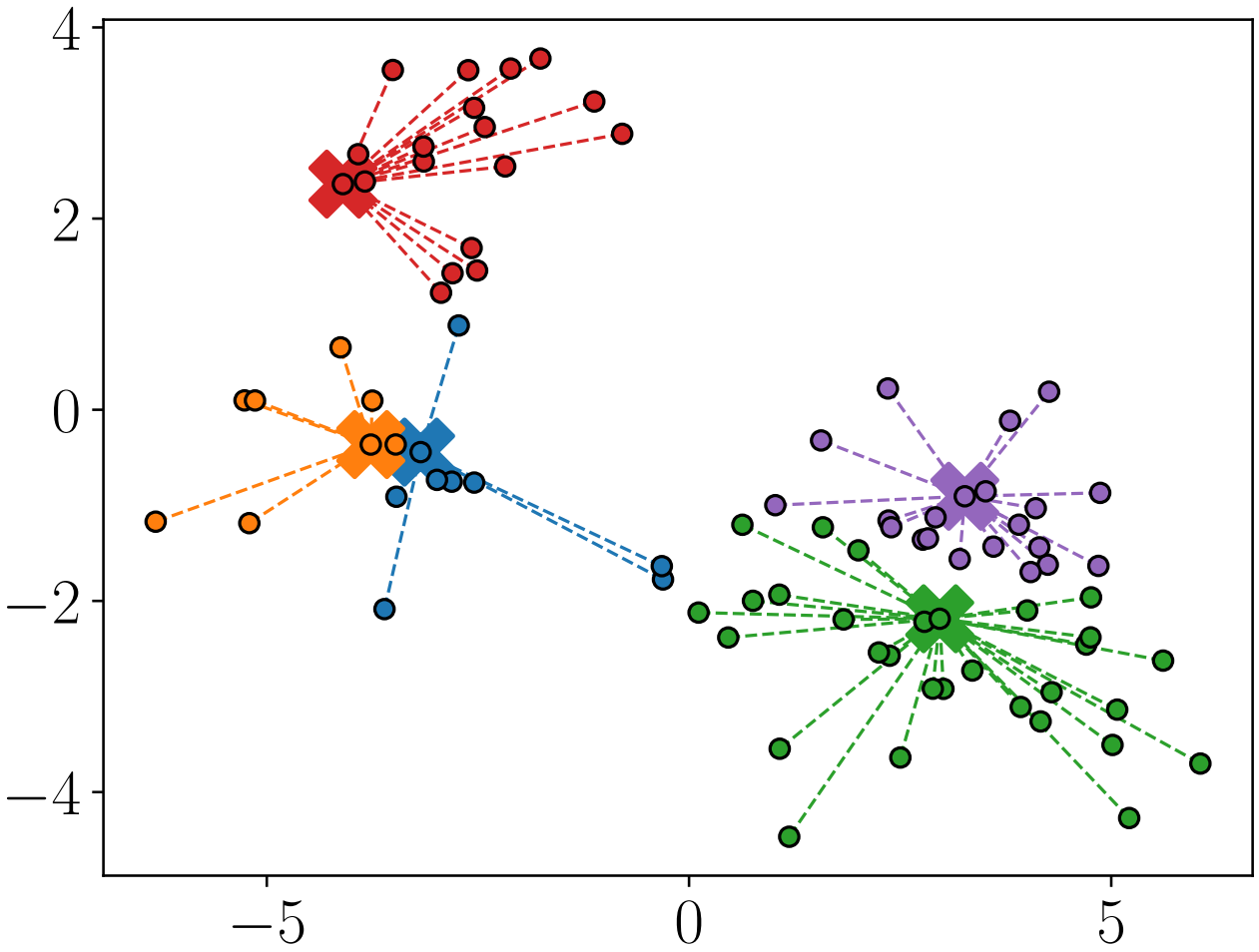
- Run multiple instances with different initial values

K–Means

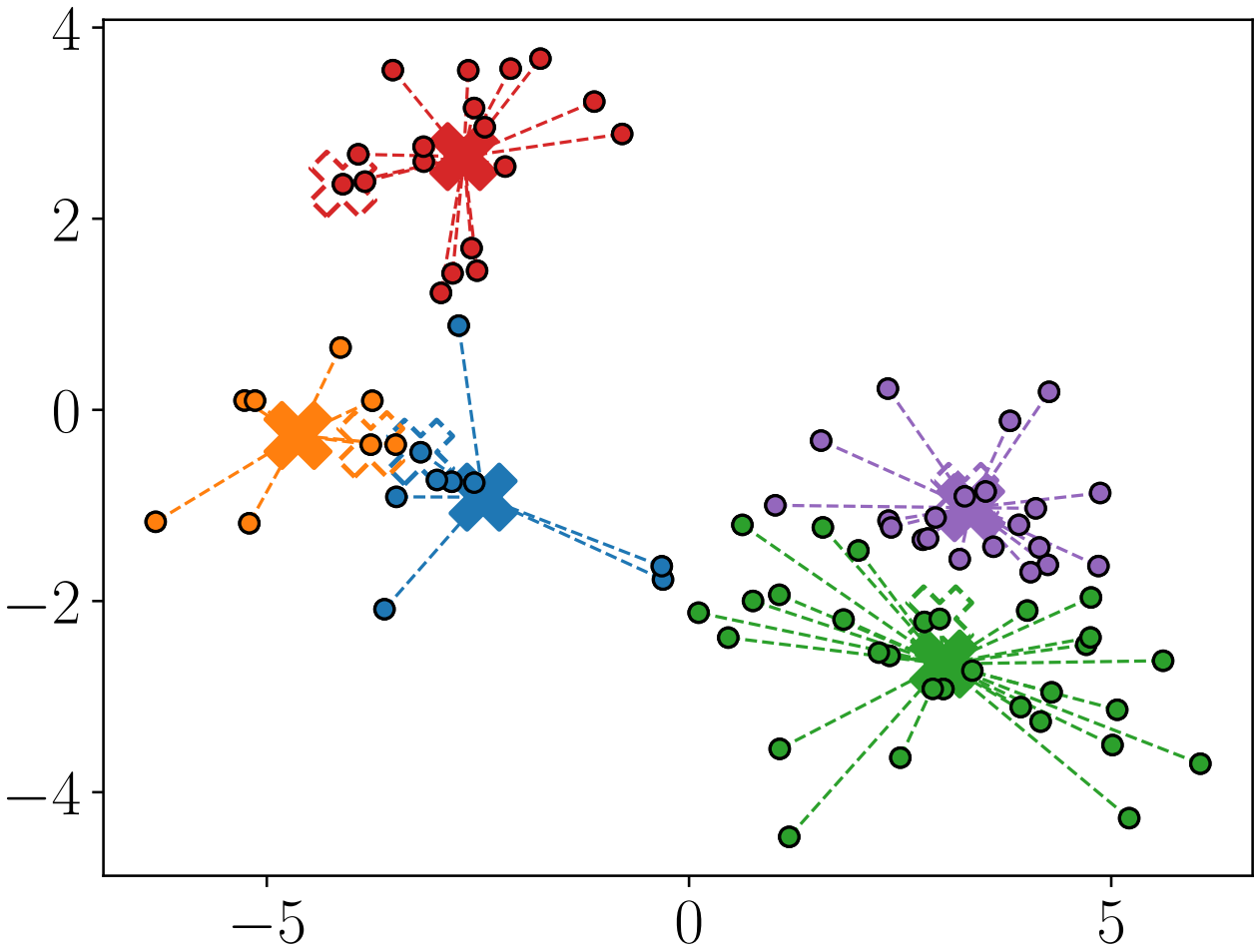
Random initialization: randomly select K samples as centroids



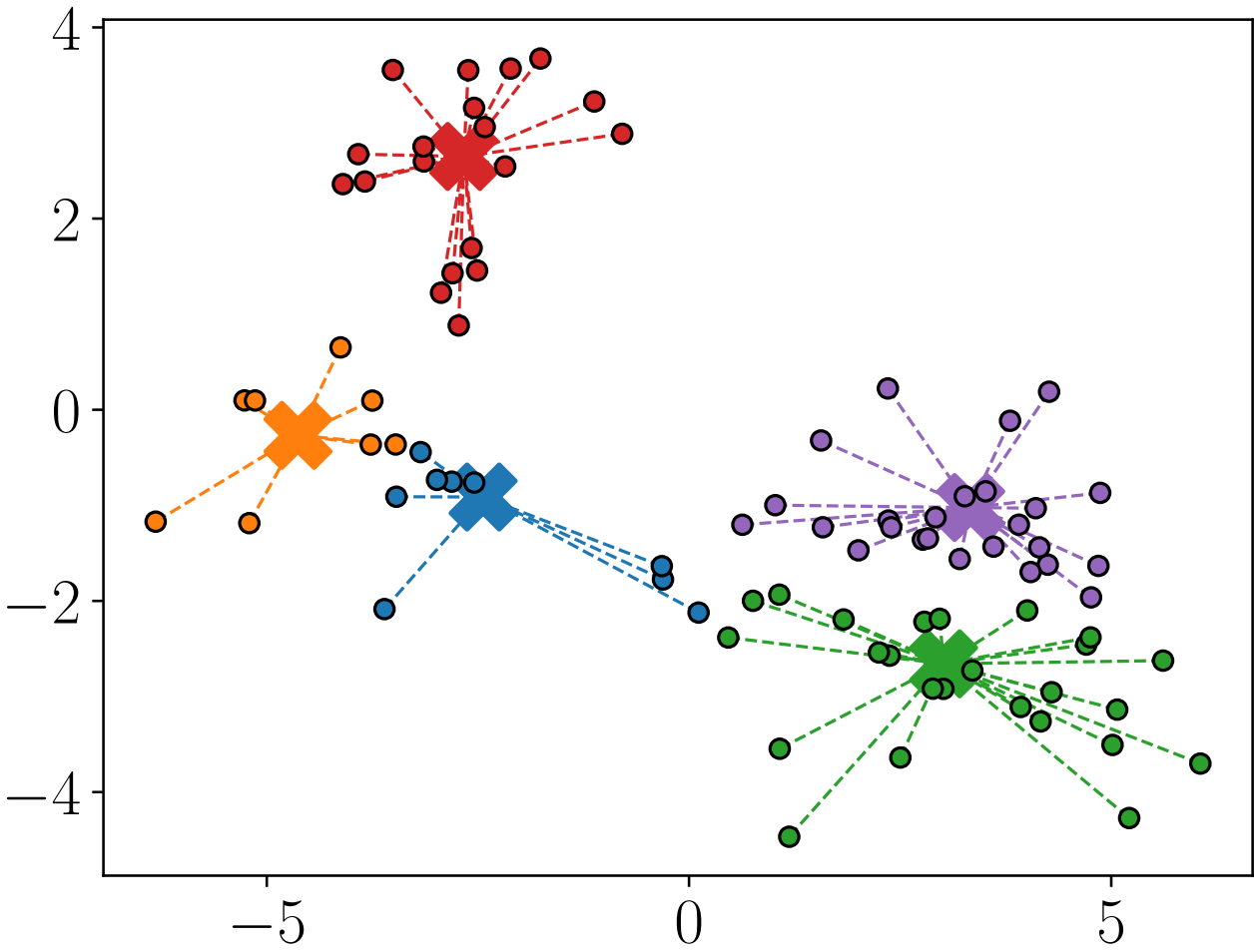
K–Means: Assign samples to nearest centroid



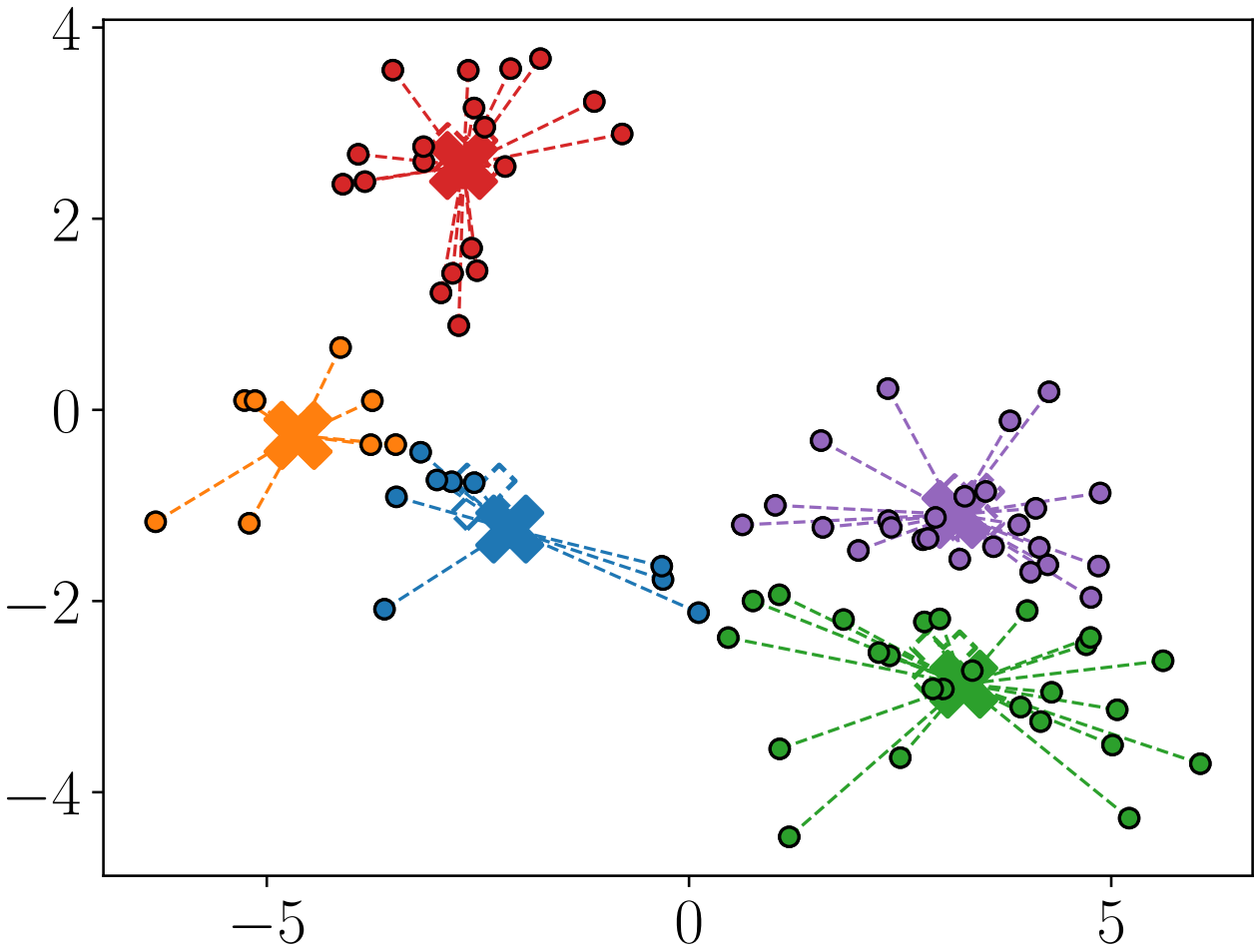
K–Means: Re-estimate centroids



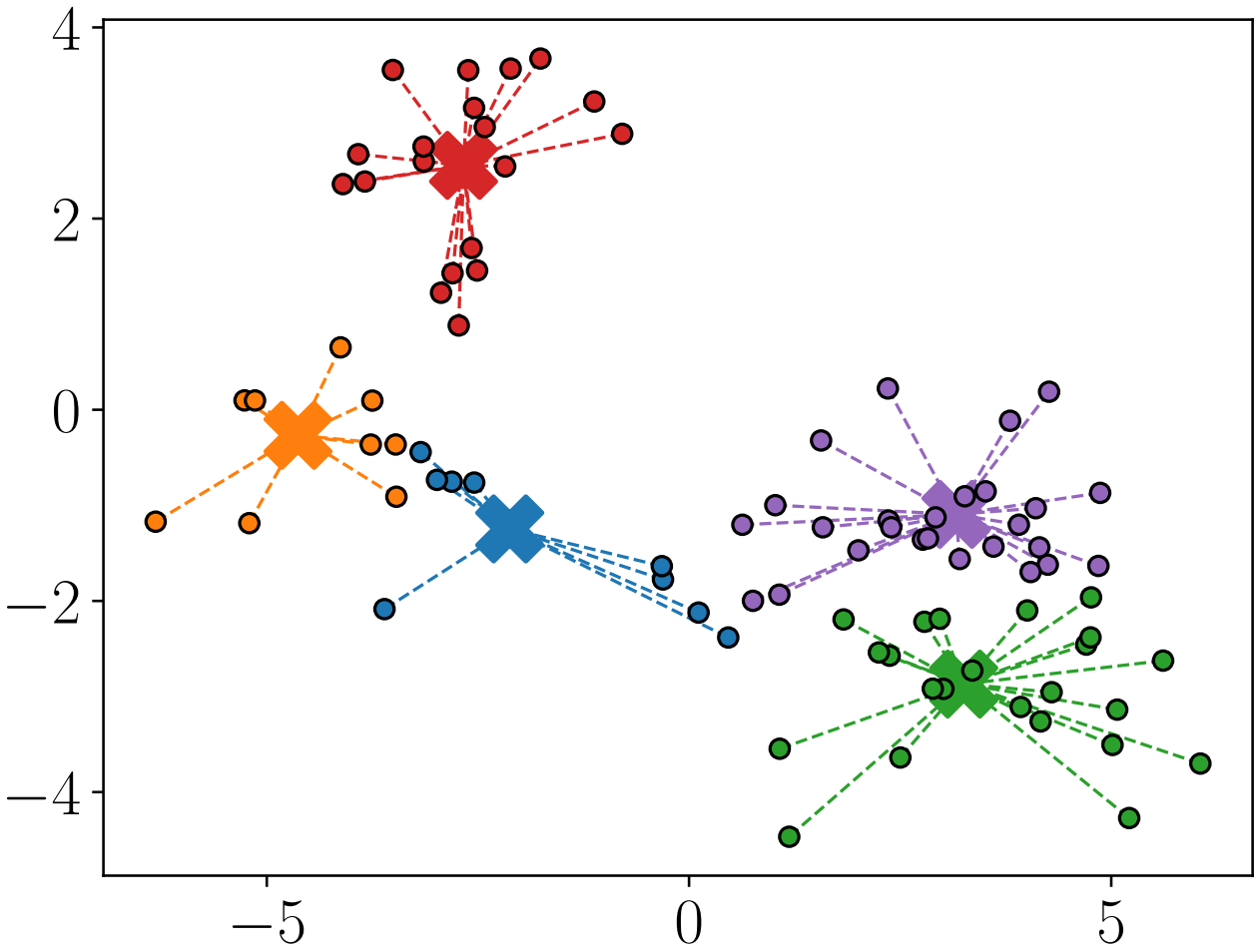
K–Means: Assign samples to nearest centroid



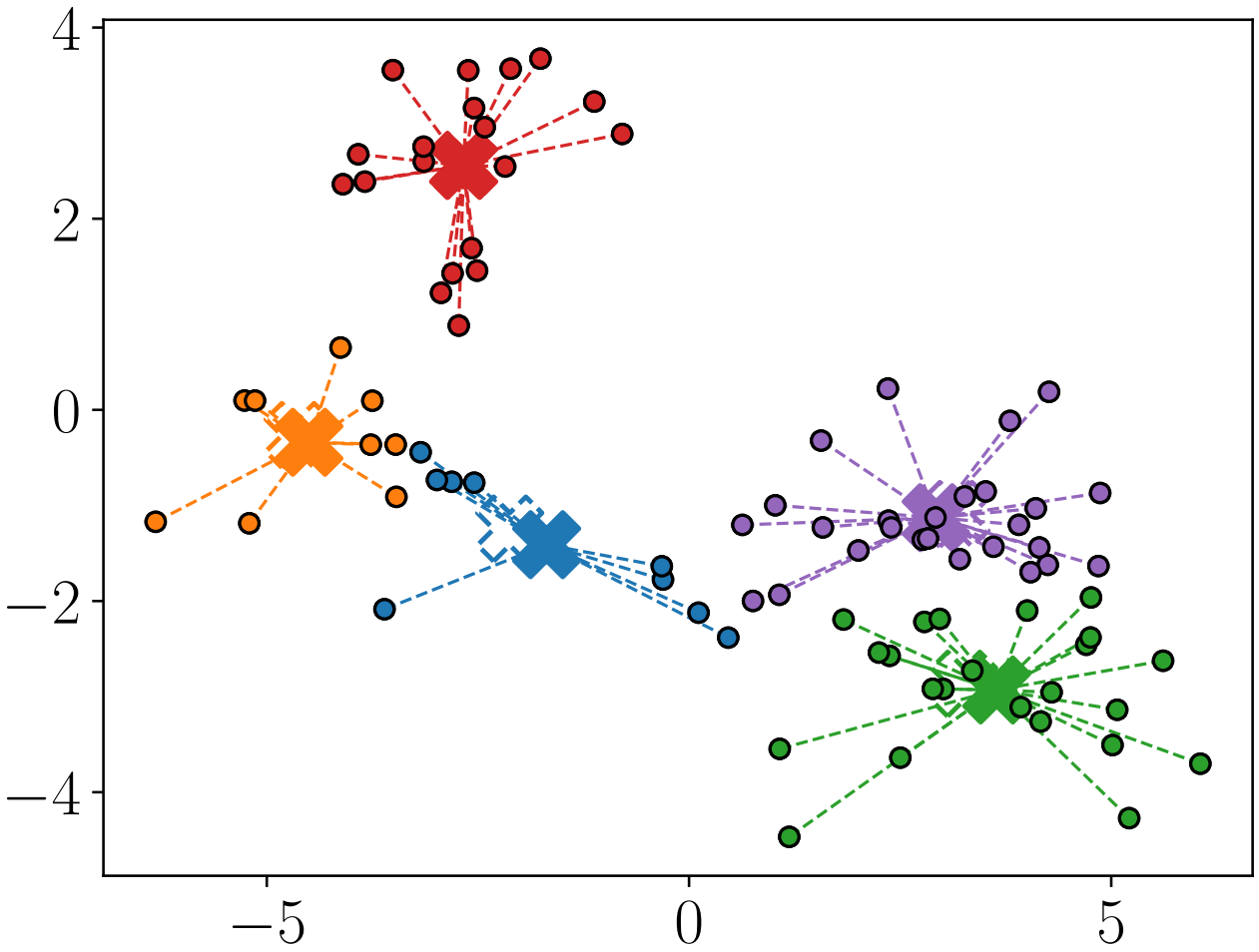
K–Means: Re-estimate centroids



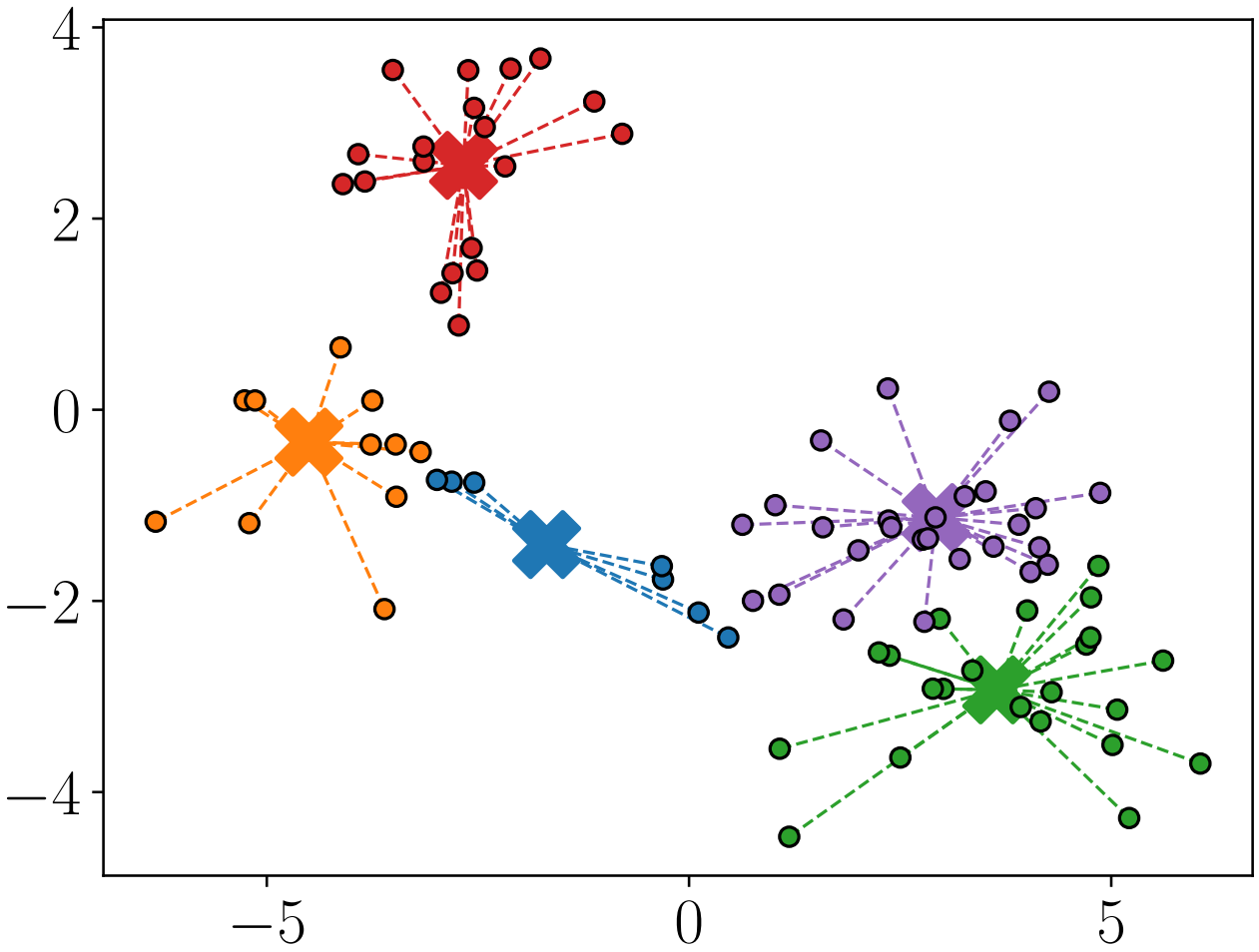
K–Means: Assign samples to nearest centroid



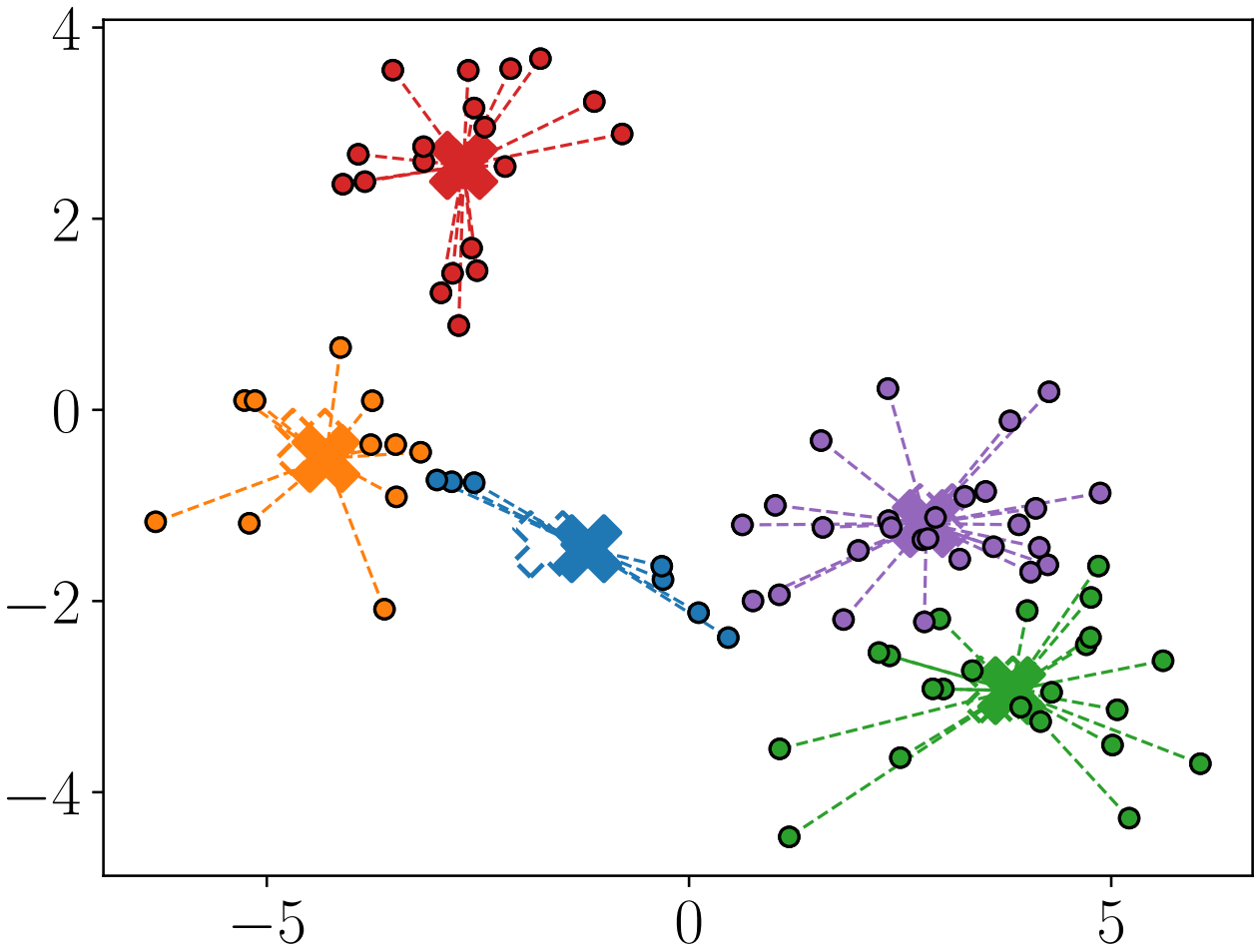
K–Means: Re-estimate centroids



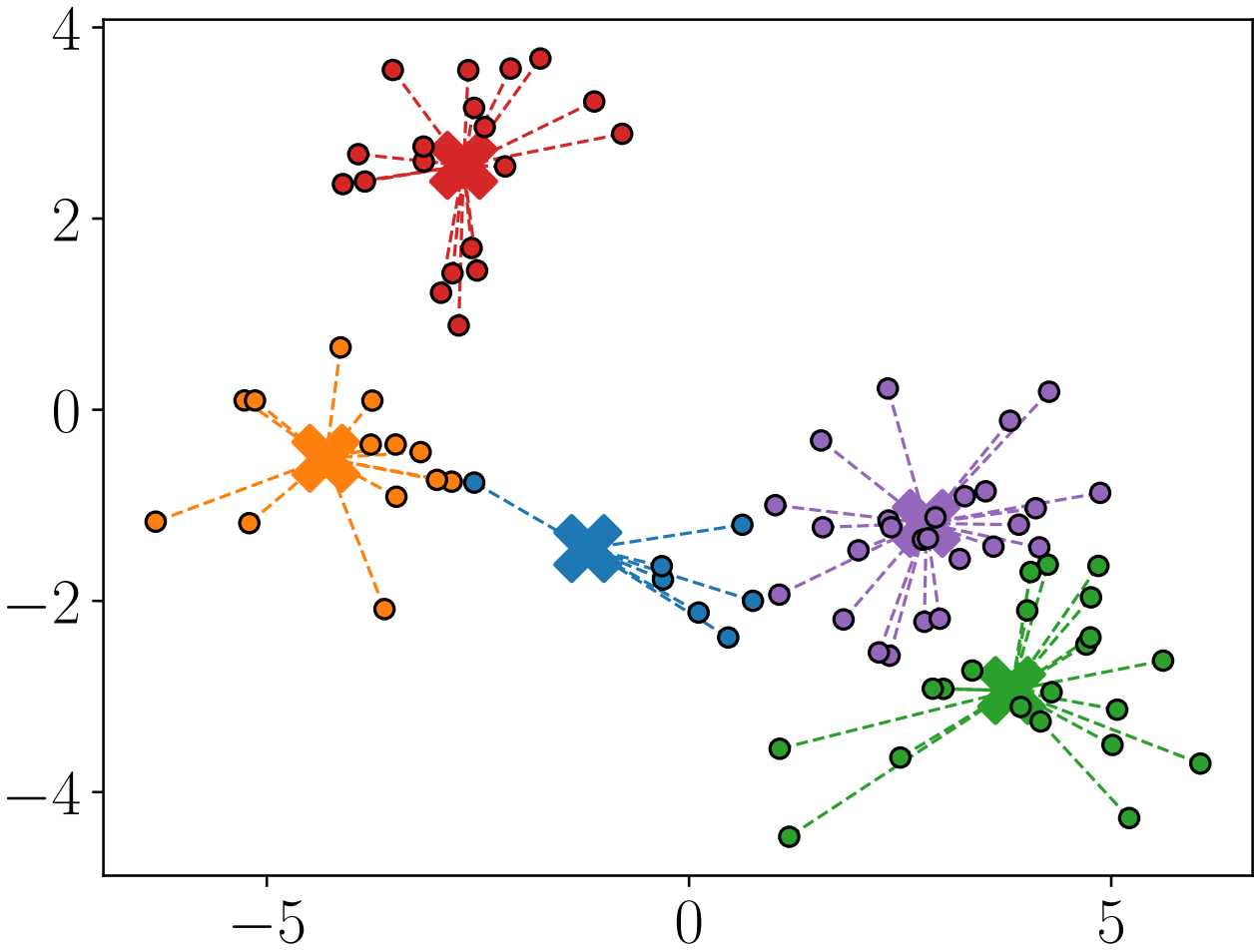
K–Means: Assign samples to nearest centroid



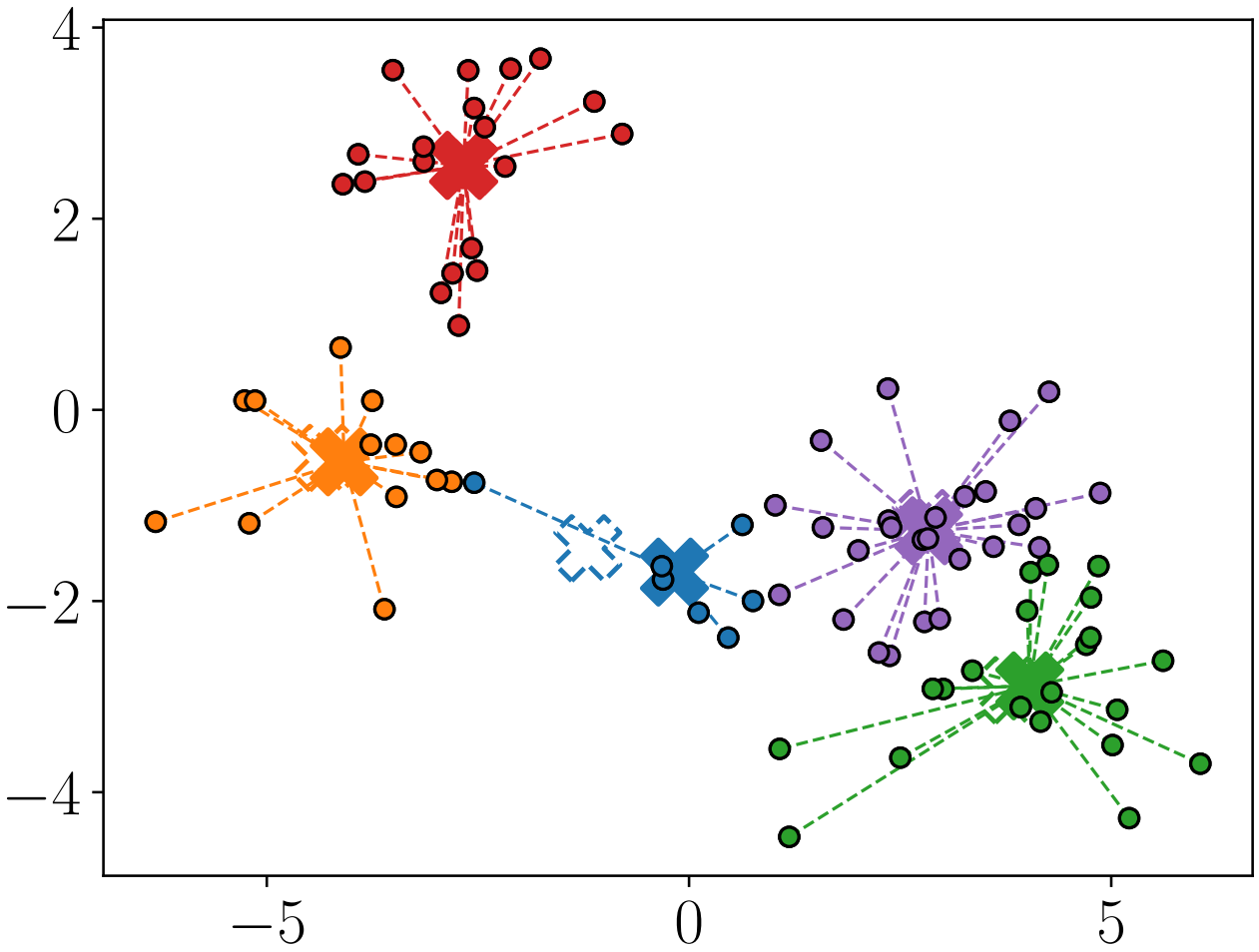
K–Means: Re-estimate centroids



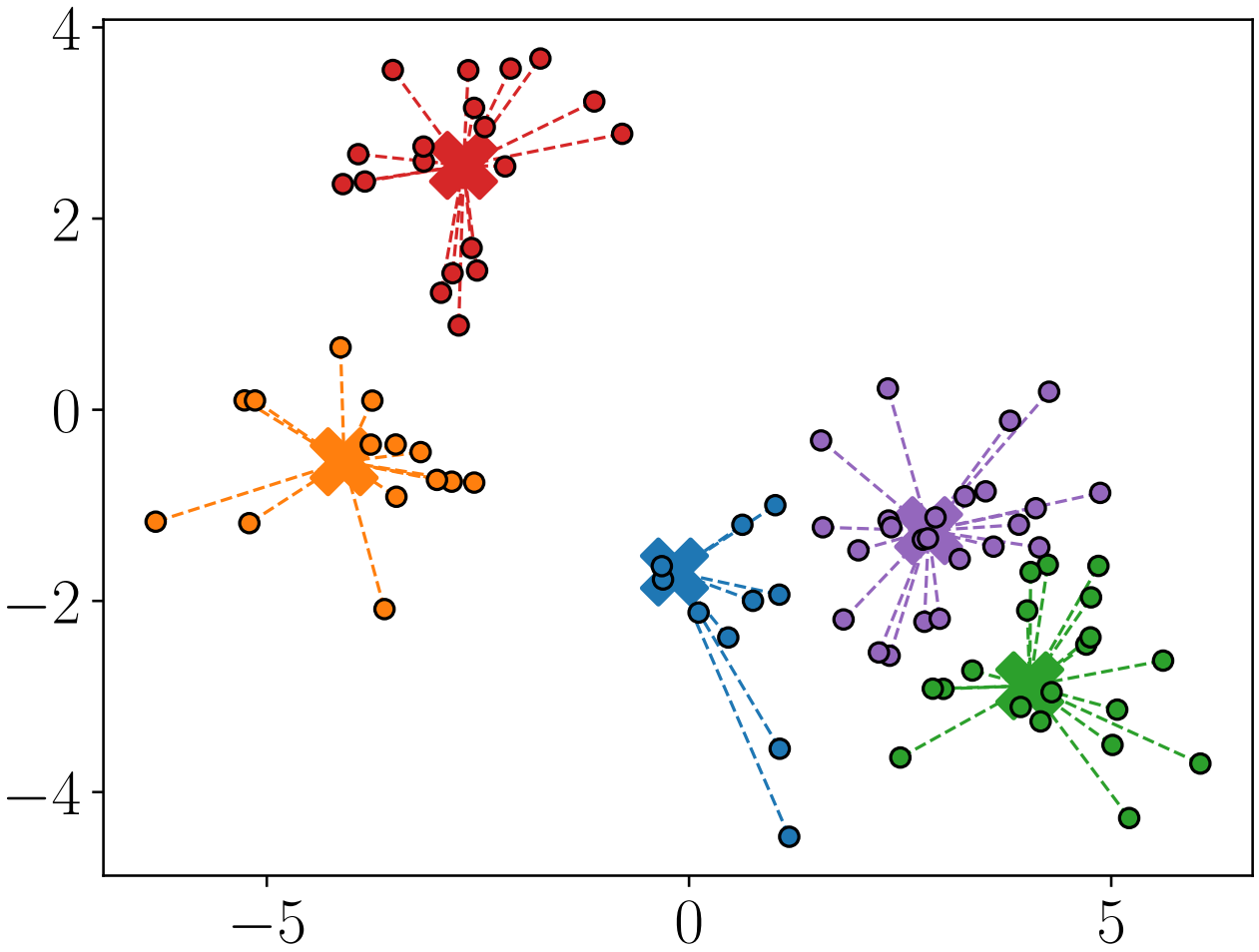
K–Means: Assign samples to nearest centroid



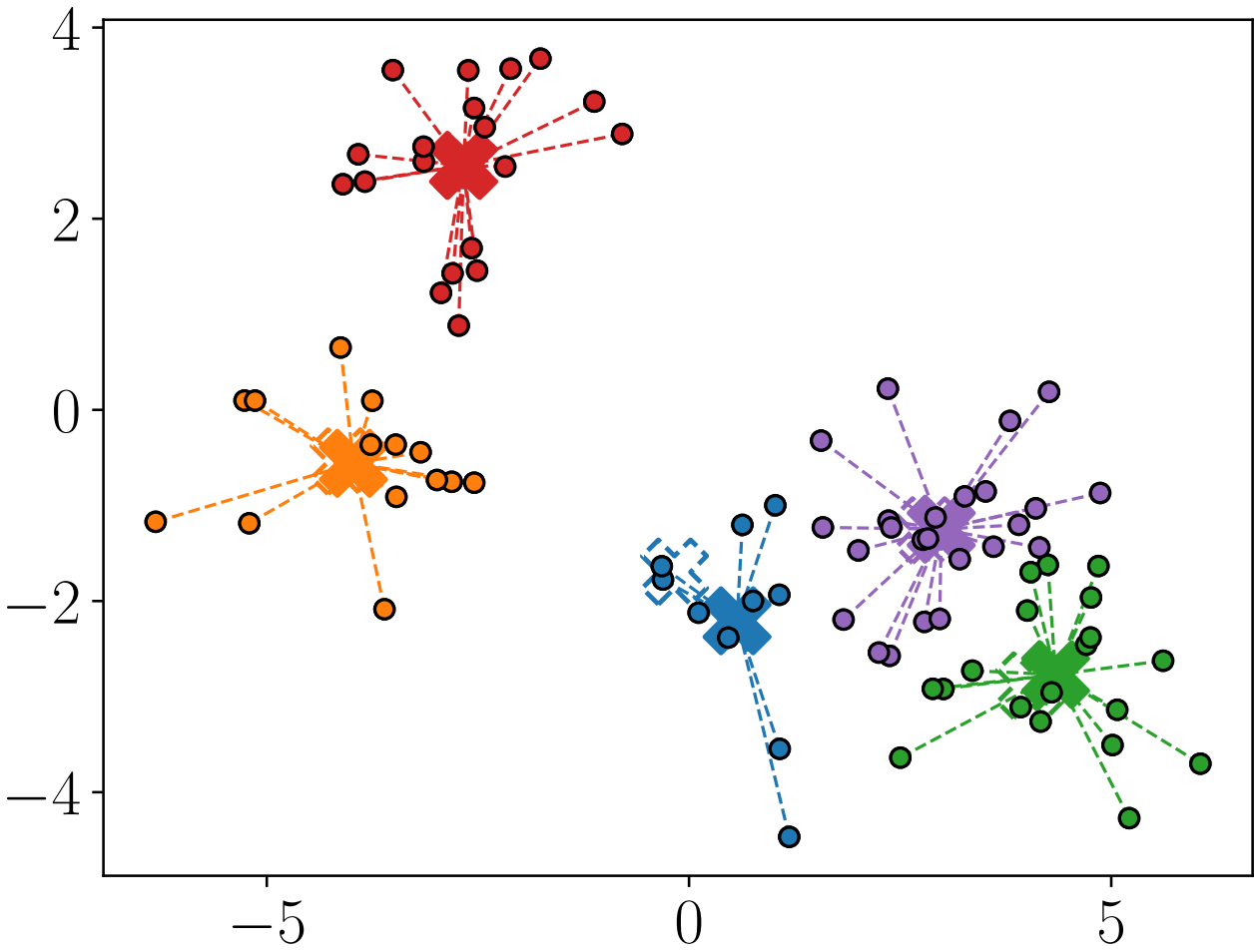
K–Means: Re-estimate centroids



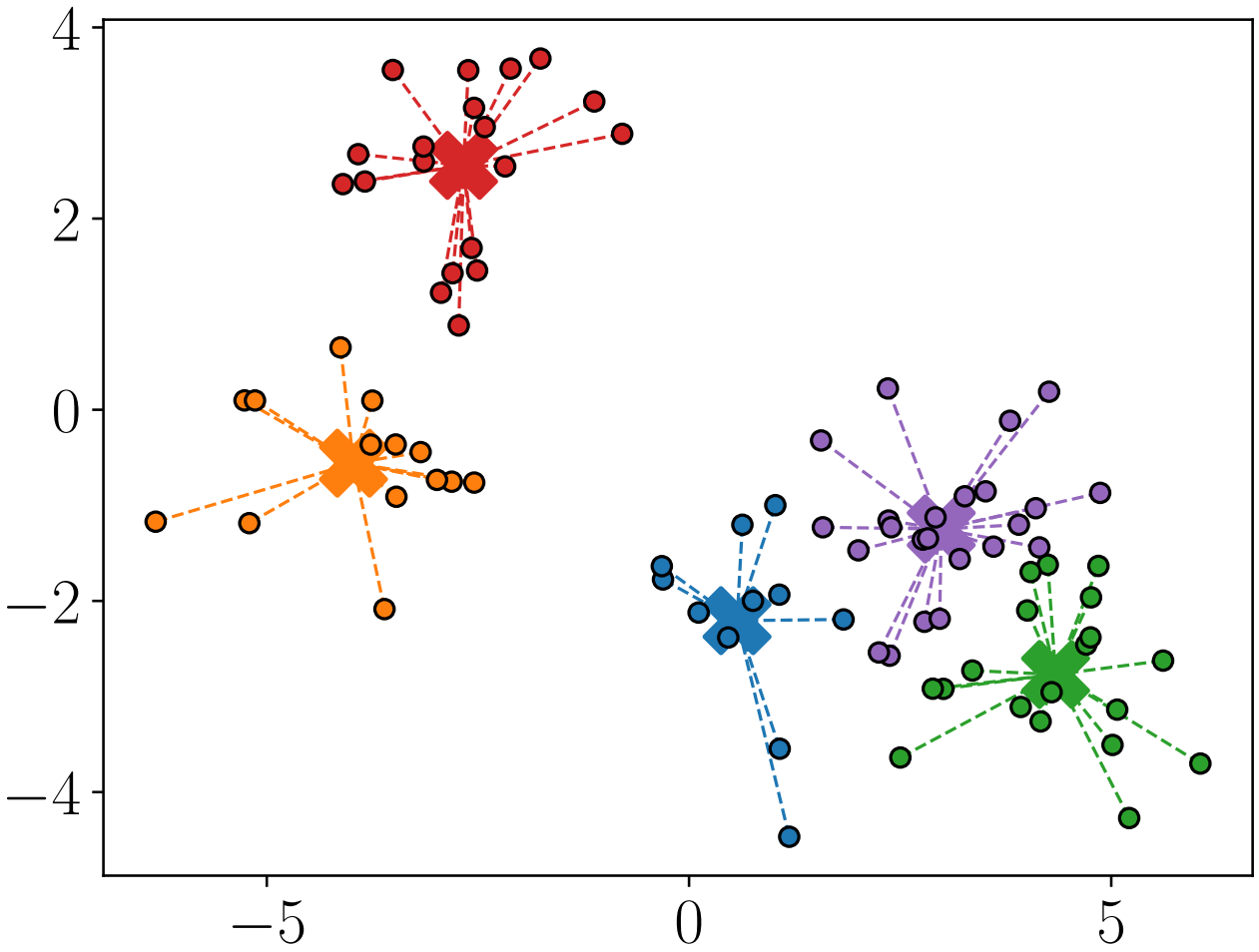
K–Means: Assign samples to nearest centroid



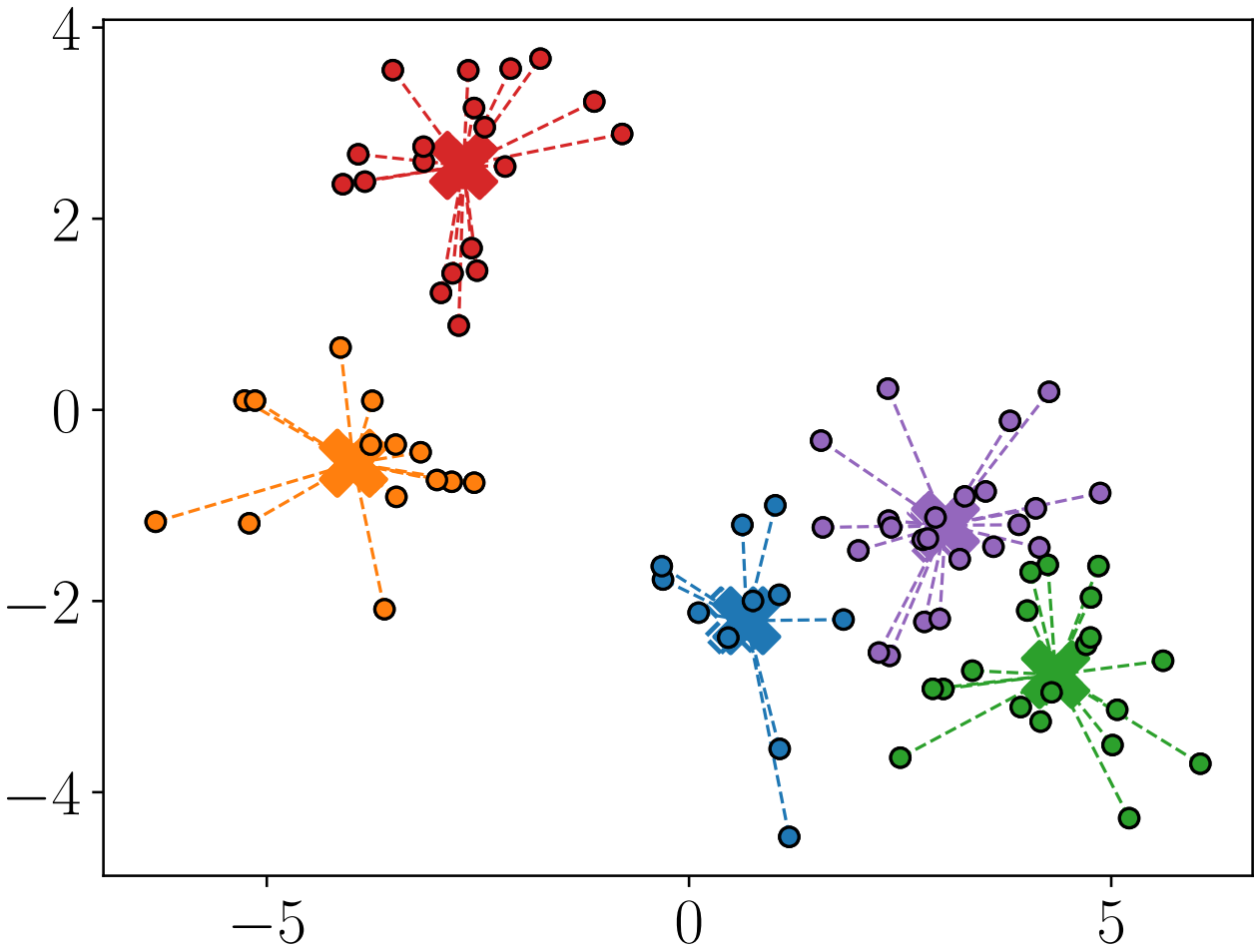
K–Means: Re-estimate centroids



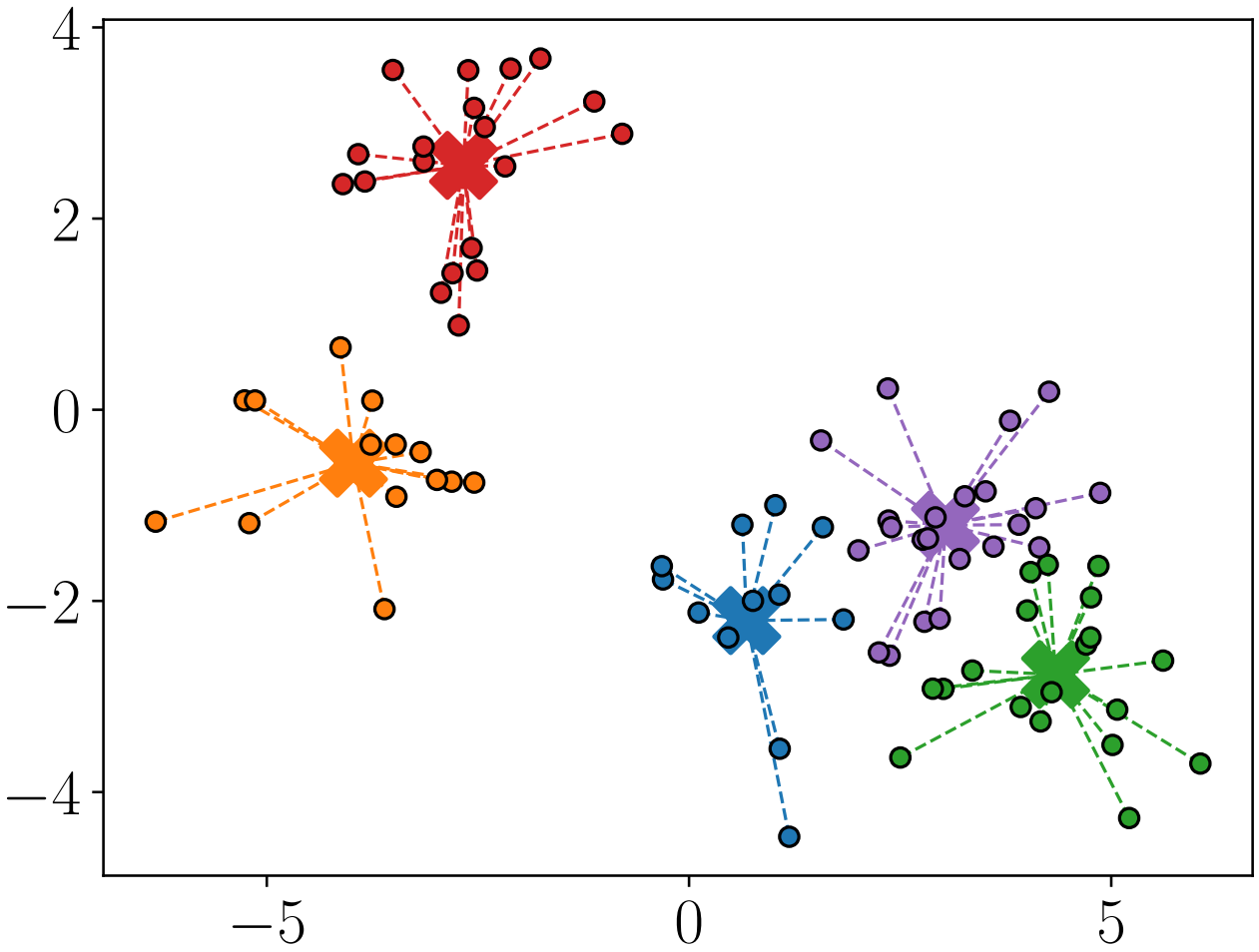
K–Means: Assign samples to nearest centroid



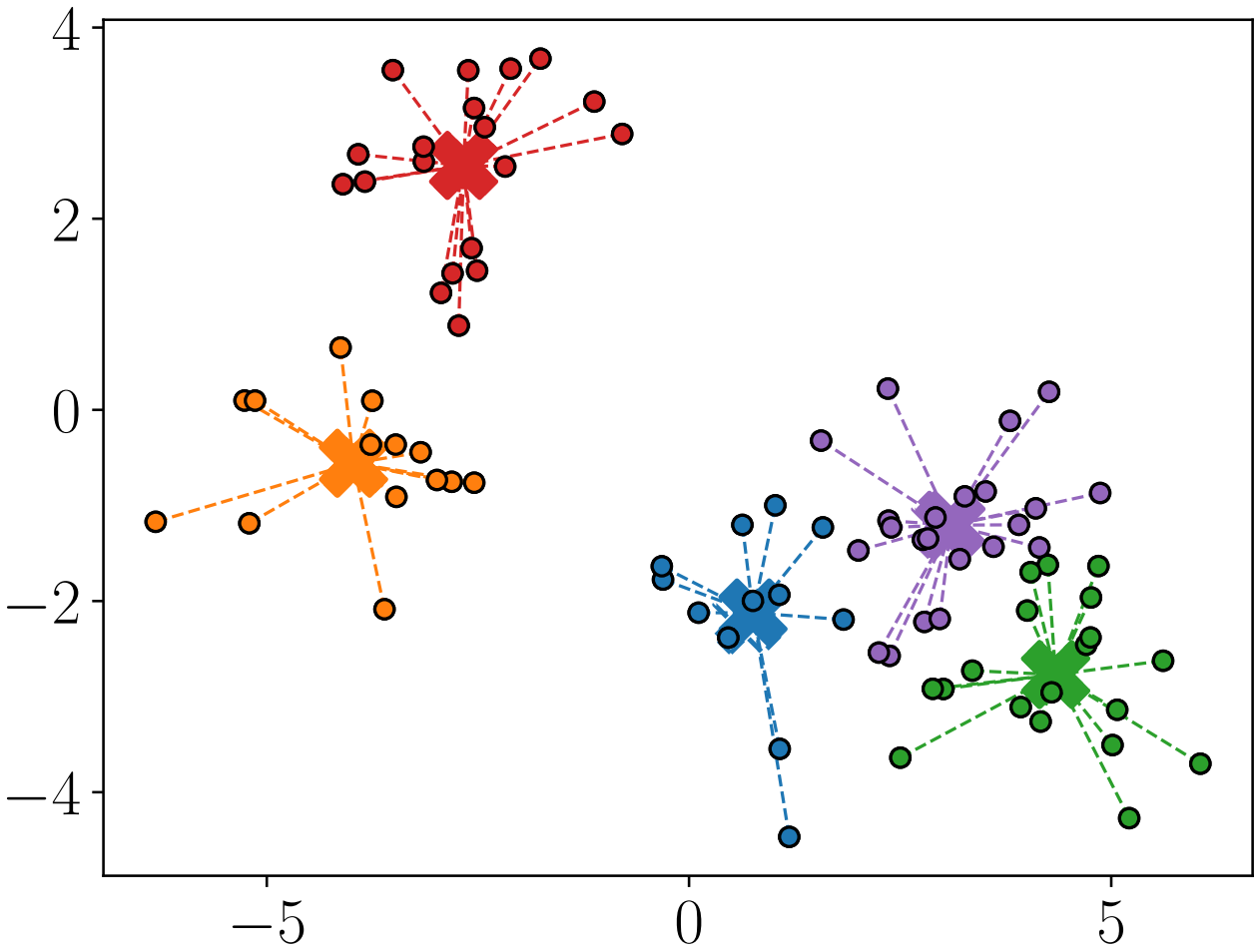
K–Means: Re-estimate centroids



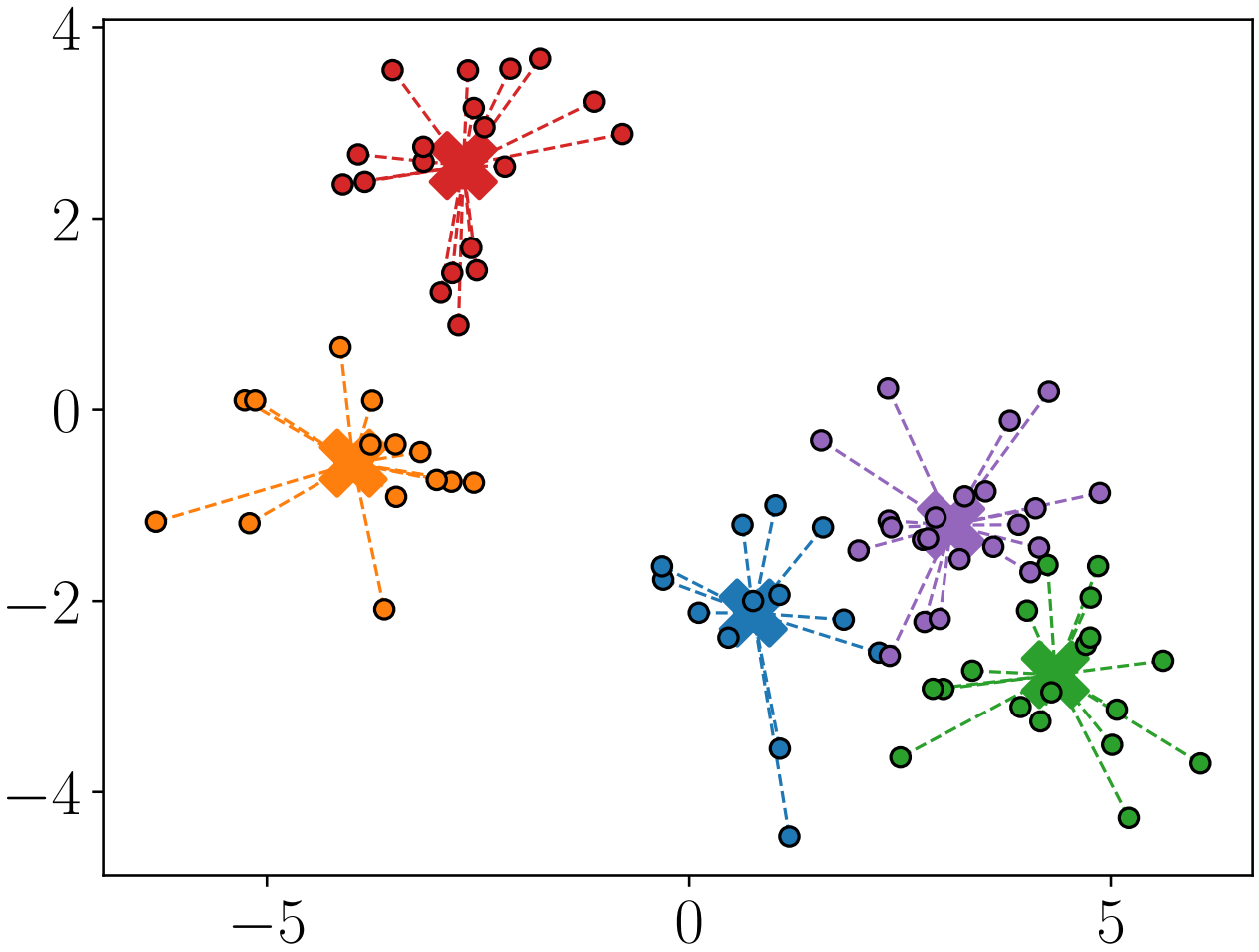
K–Means: Assign samples to nearest centroid



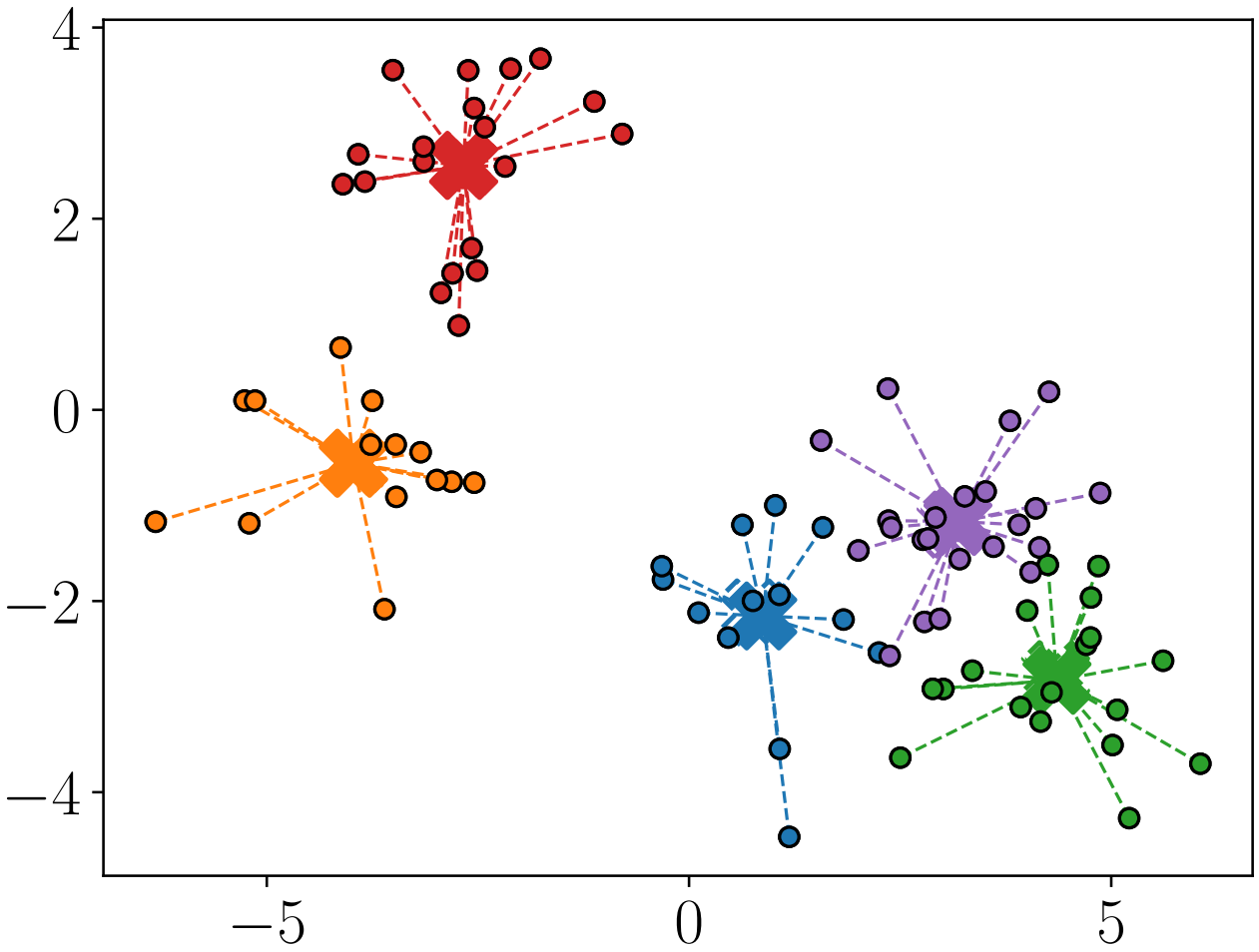
K–Means: Re-estimate centroids



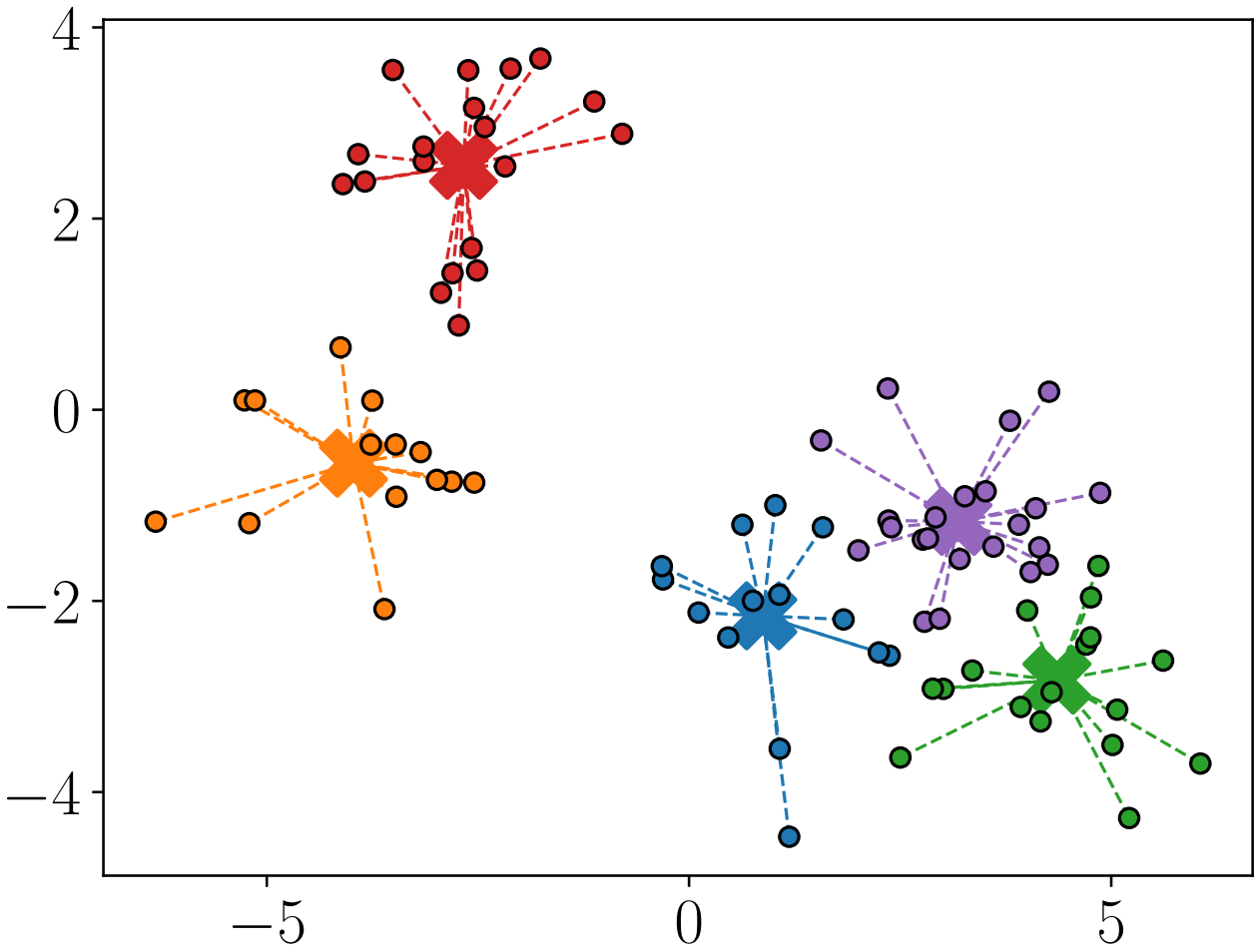
K–Means: Assign samples to nearest centroid



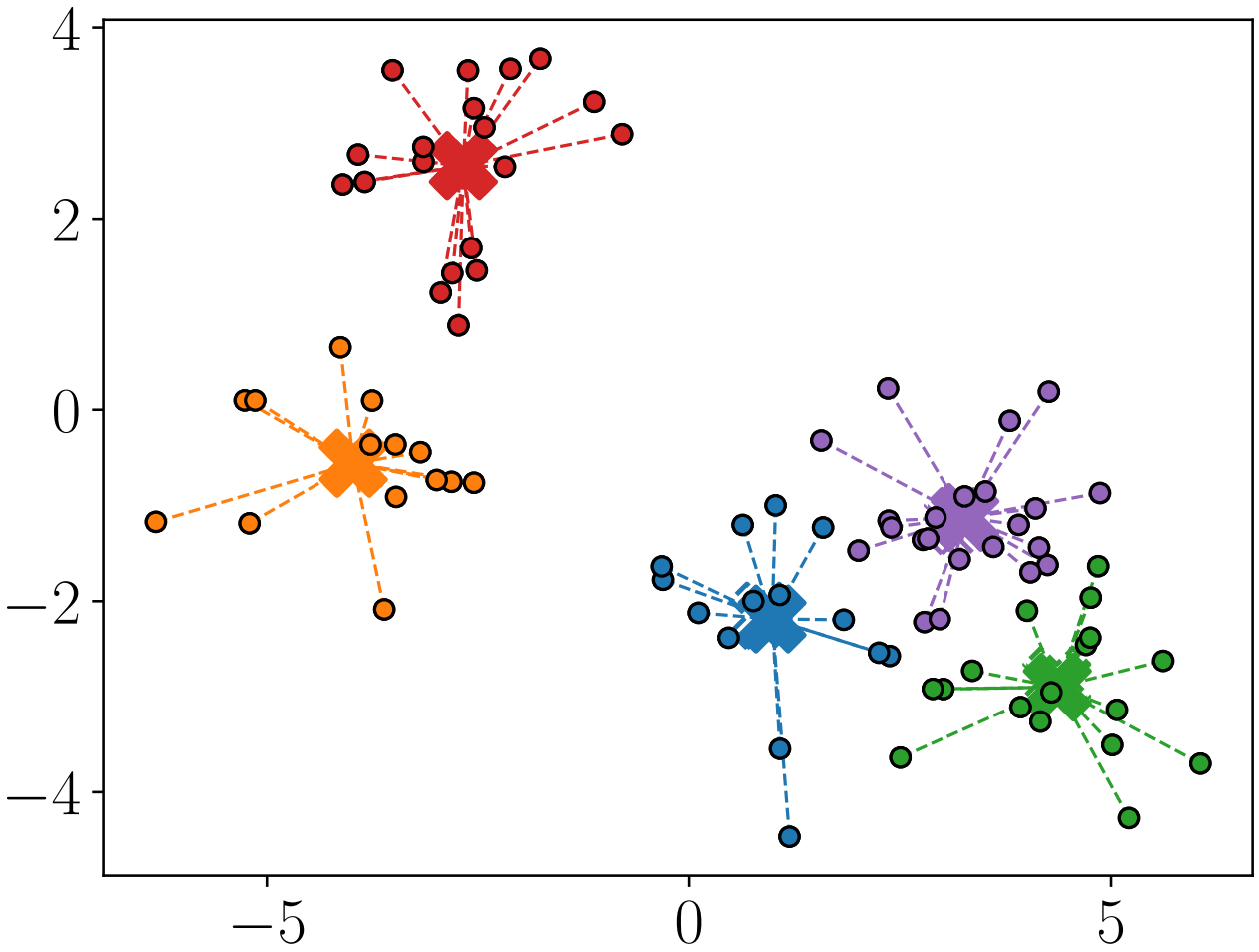
K–Means: Re-estimate centroids



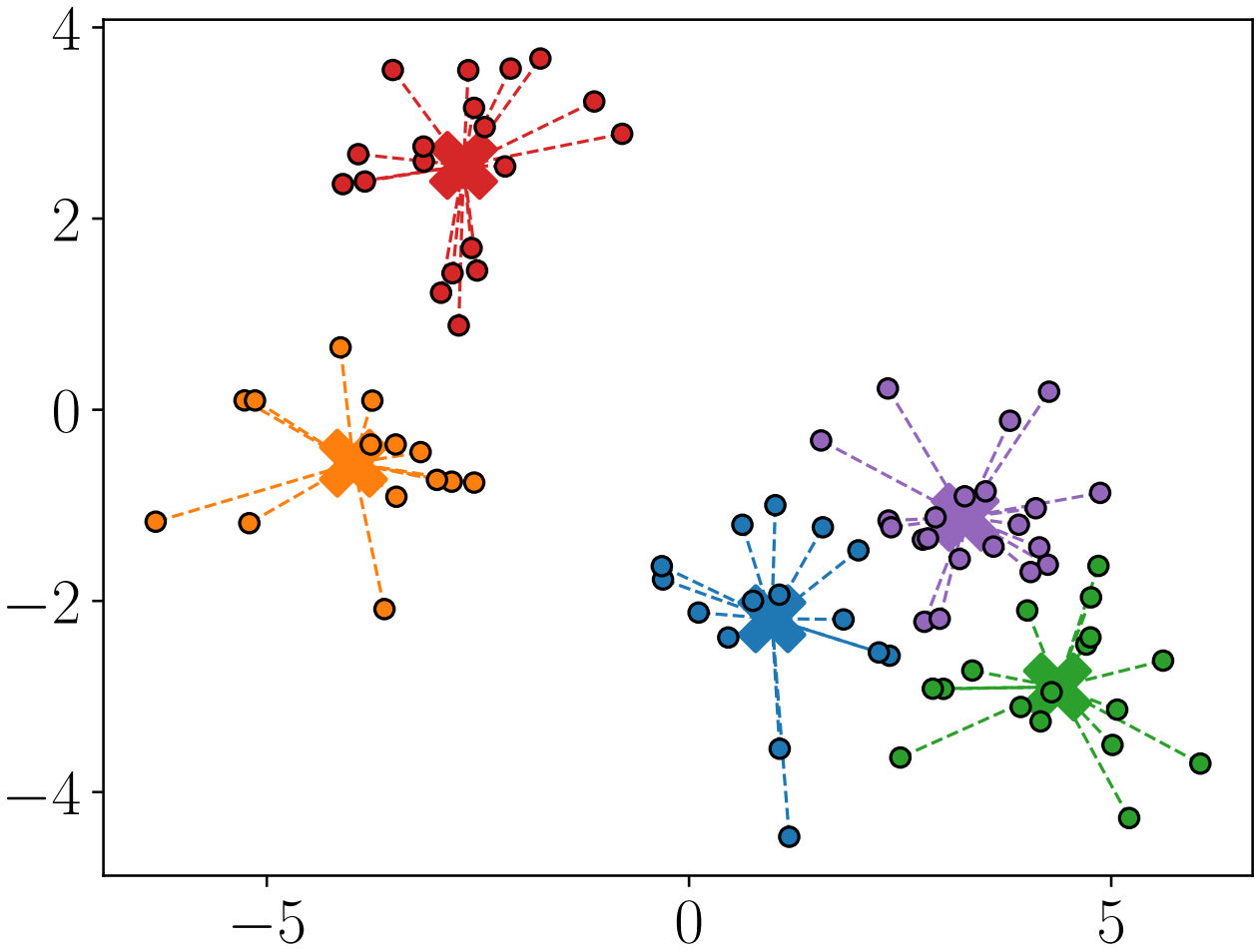
K–Means: Assign samples to nearest centroid



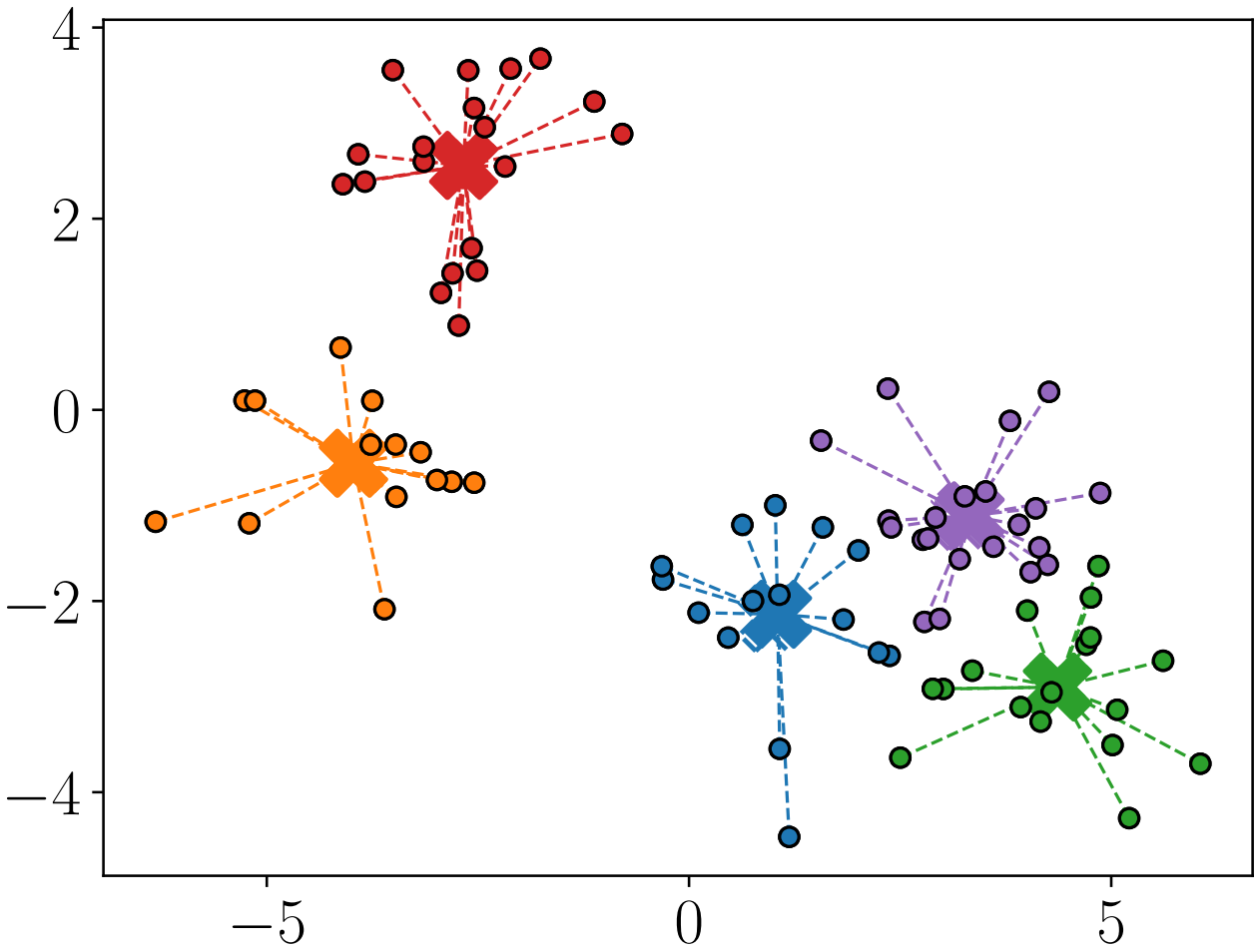
K–Means: Re-estimate centroids



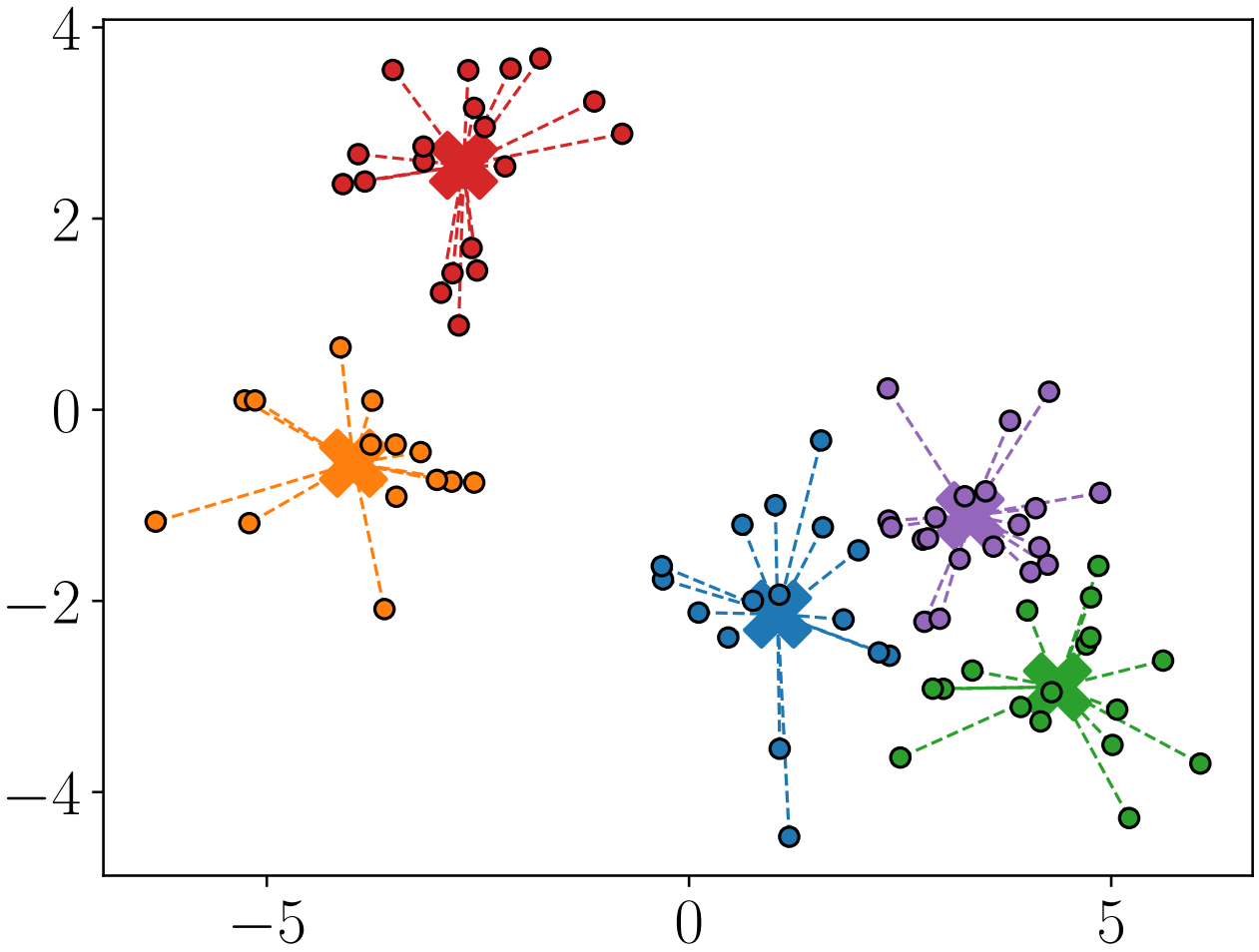
K–Means: Assign samples to nearest centroid



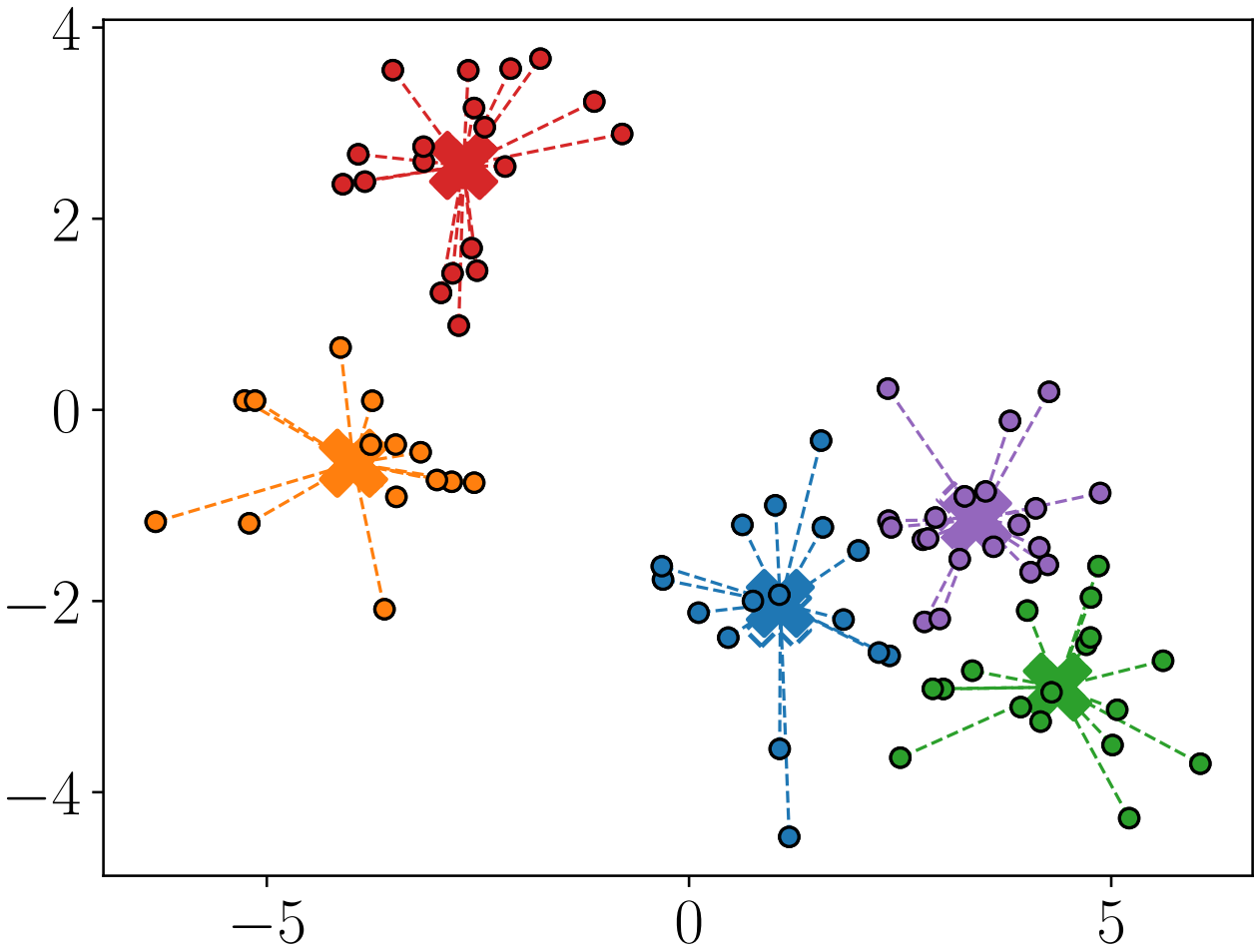
K–Means: Re-estimate centroids



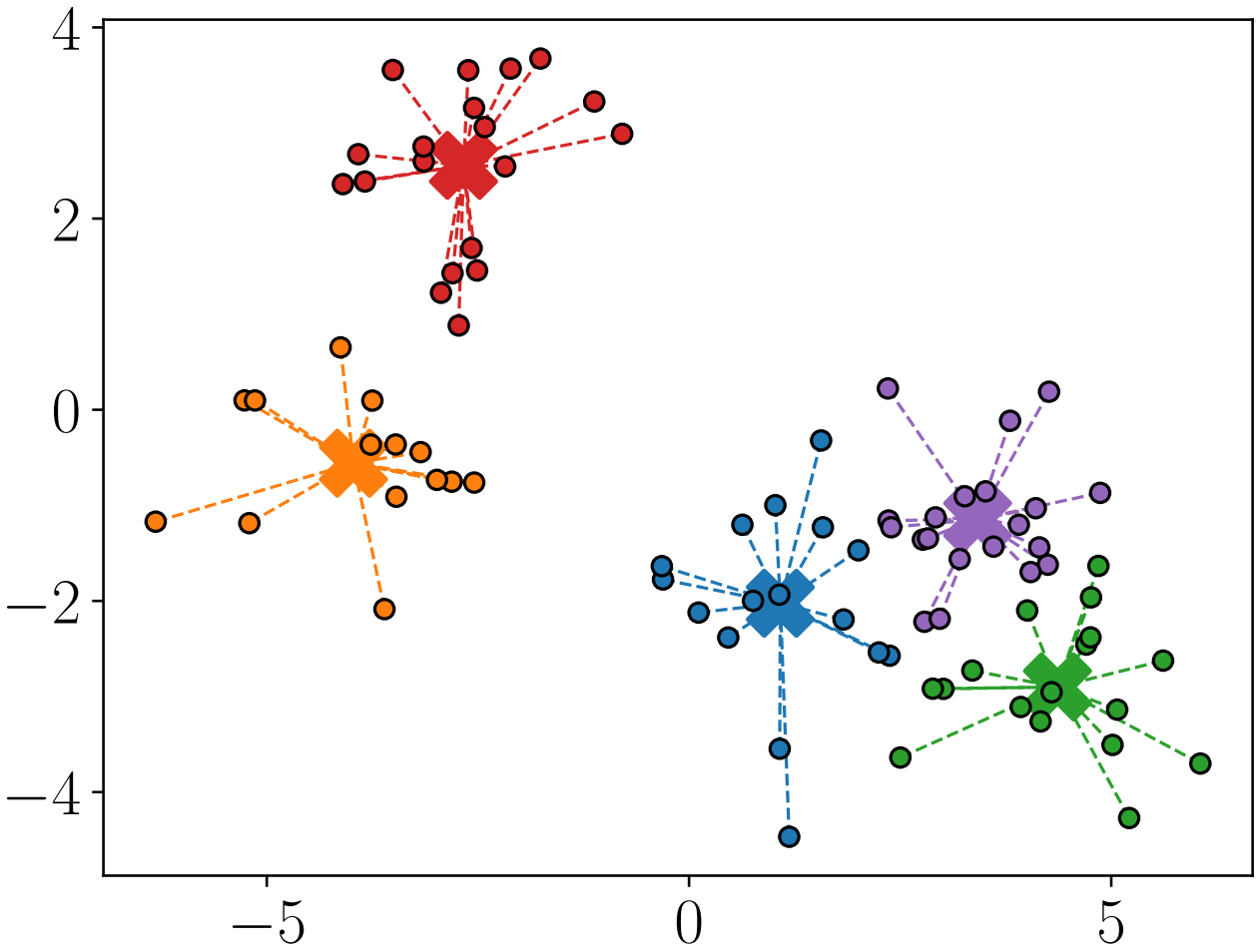
K–Means: Assign samples to nearest centroid



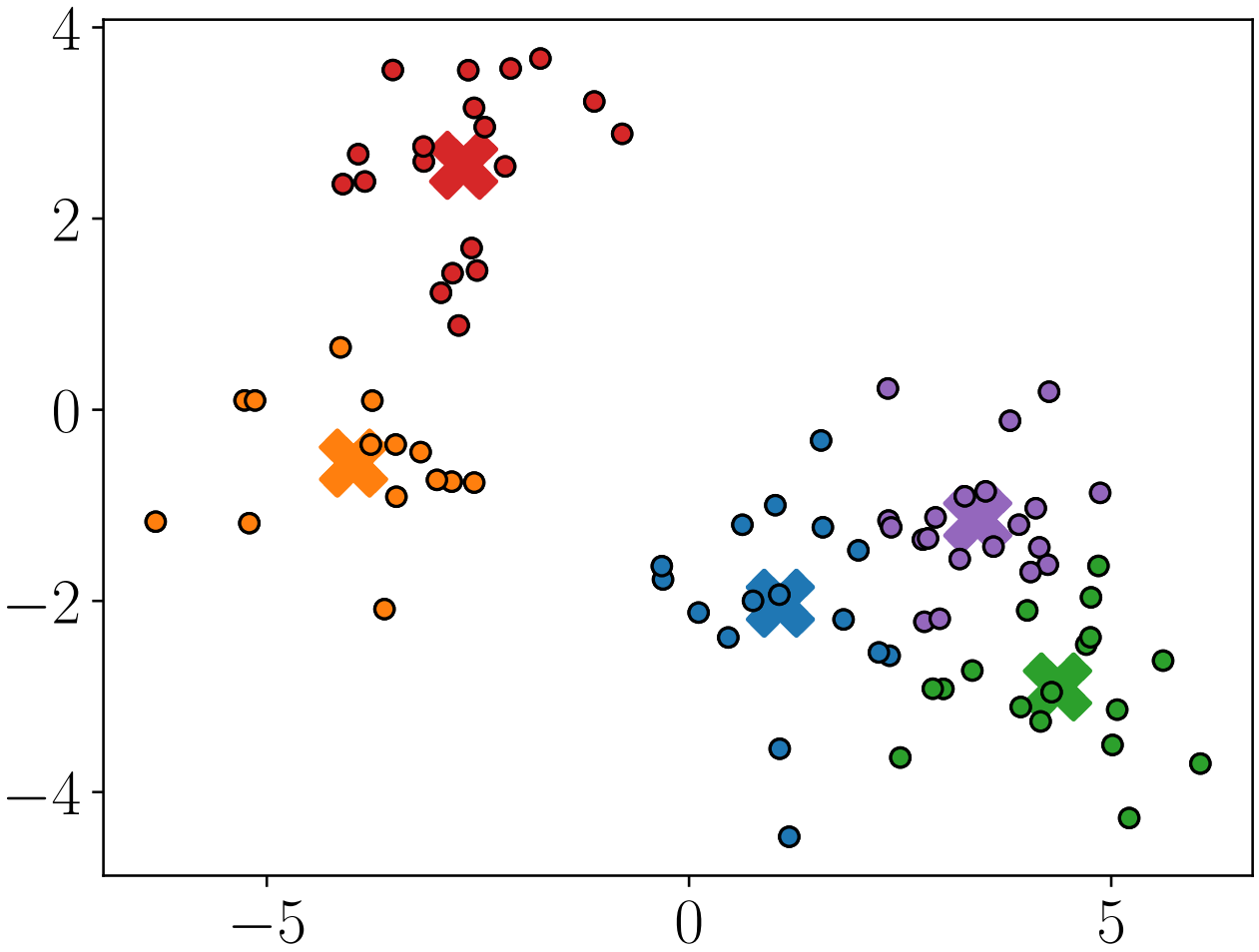
K–Means: Re-estimate centroids



K–Means: Assign samples to nearest centroid

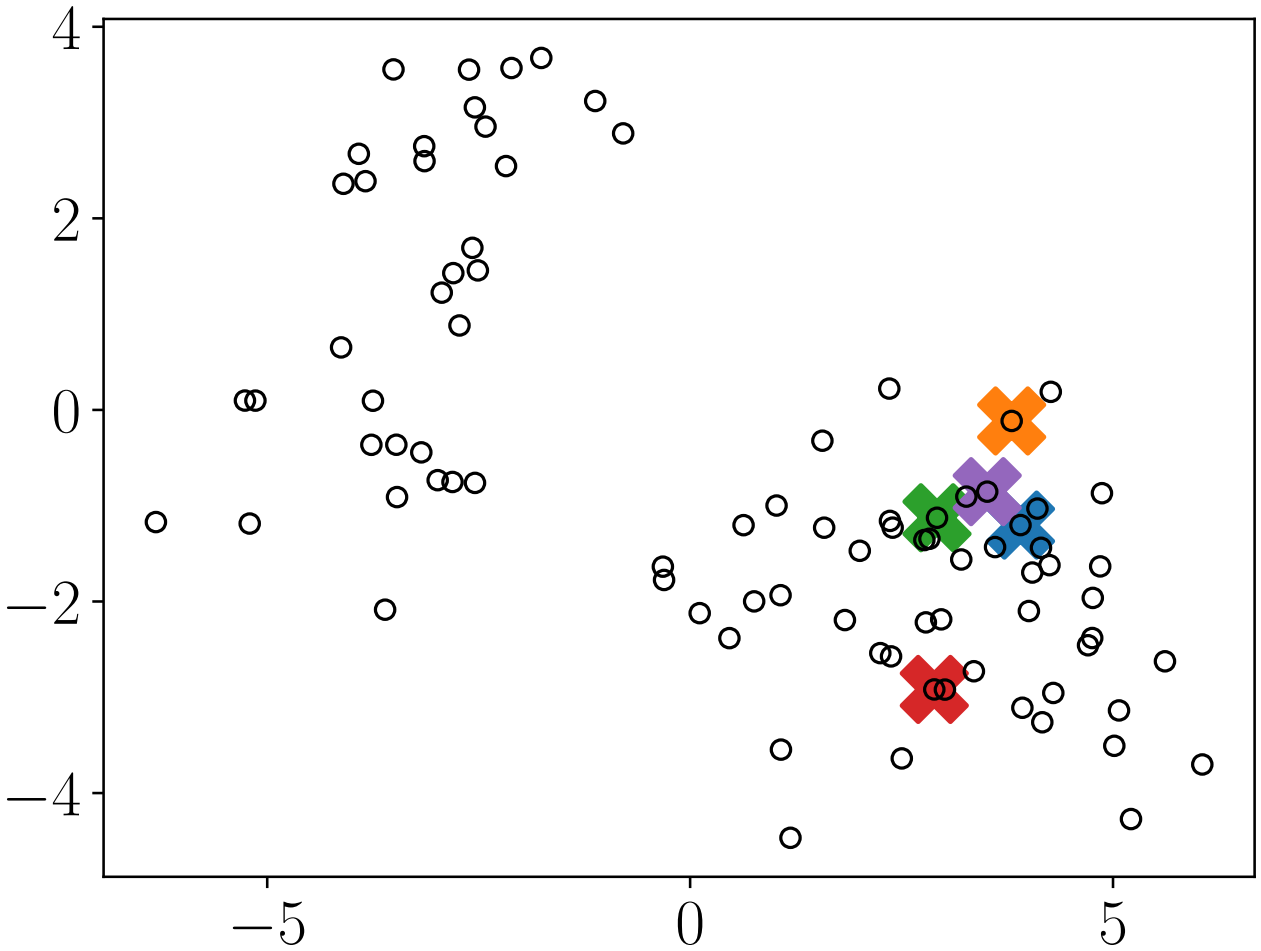


Assignments didn't change — final result



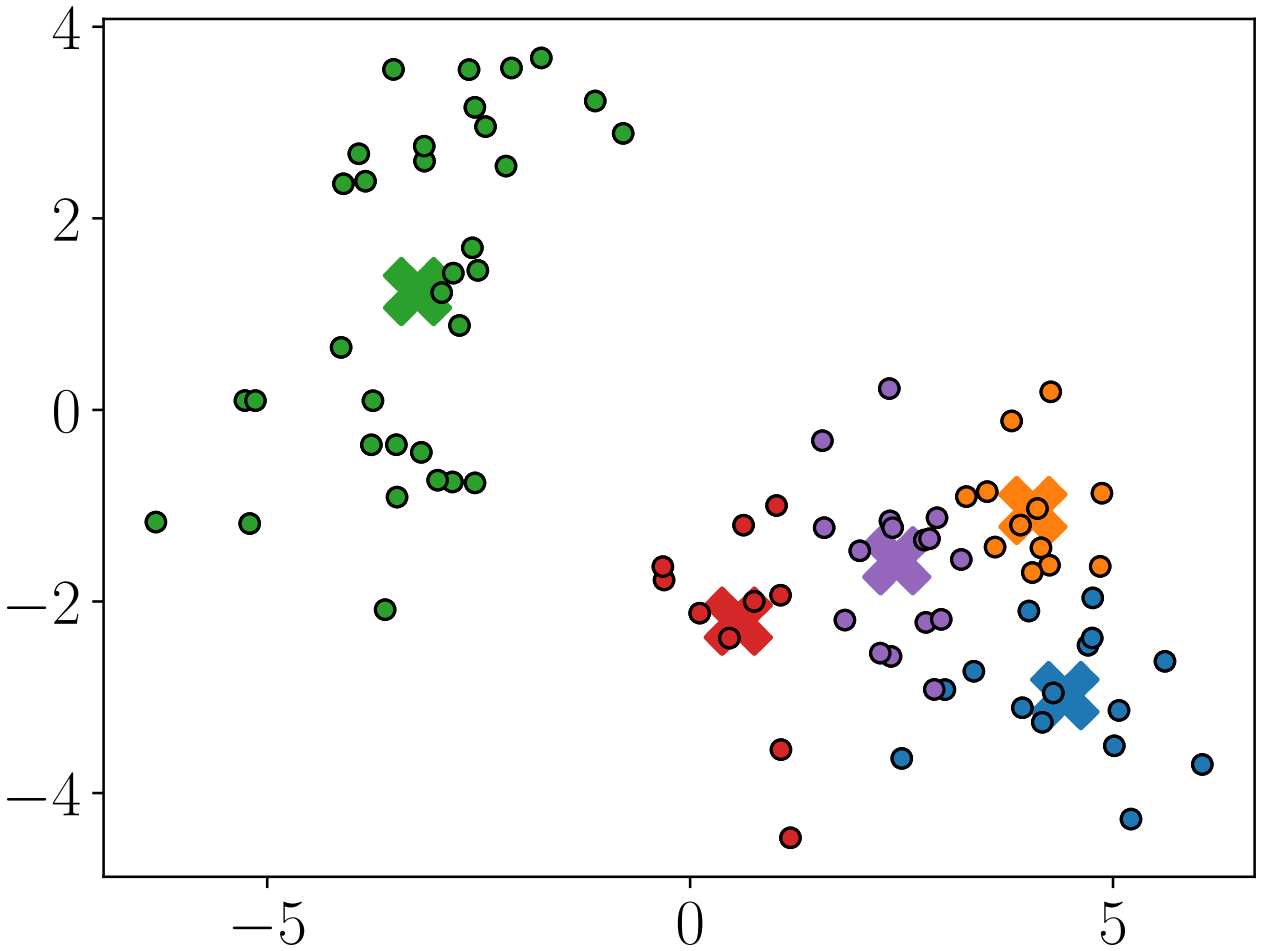
K–Means

K–Means is sensitive to intialization



K–Means

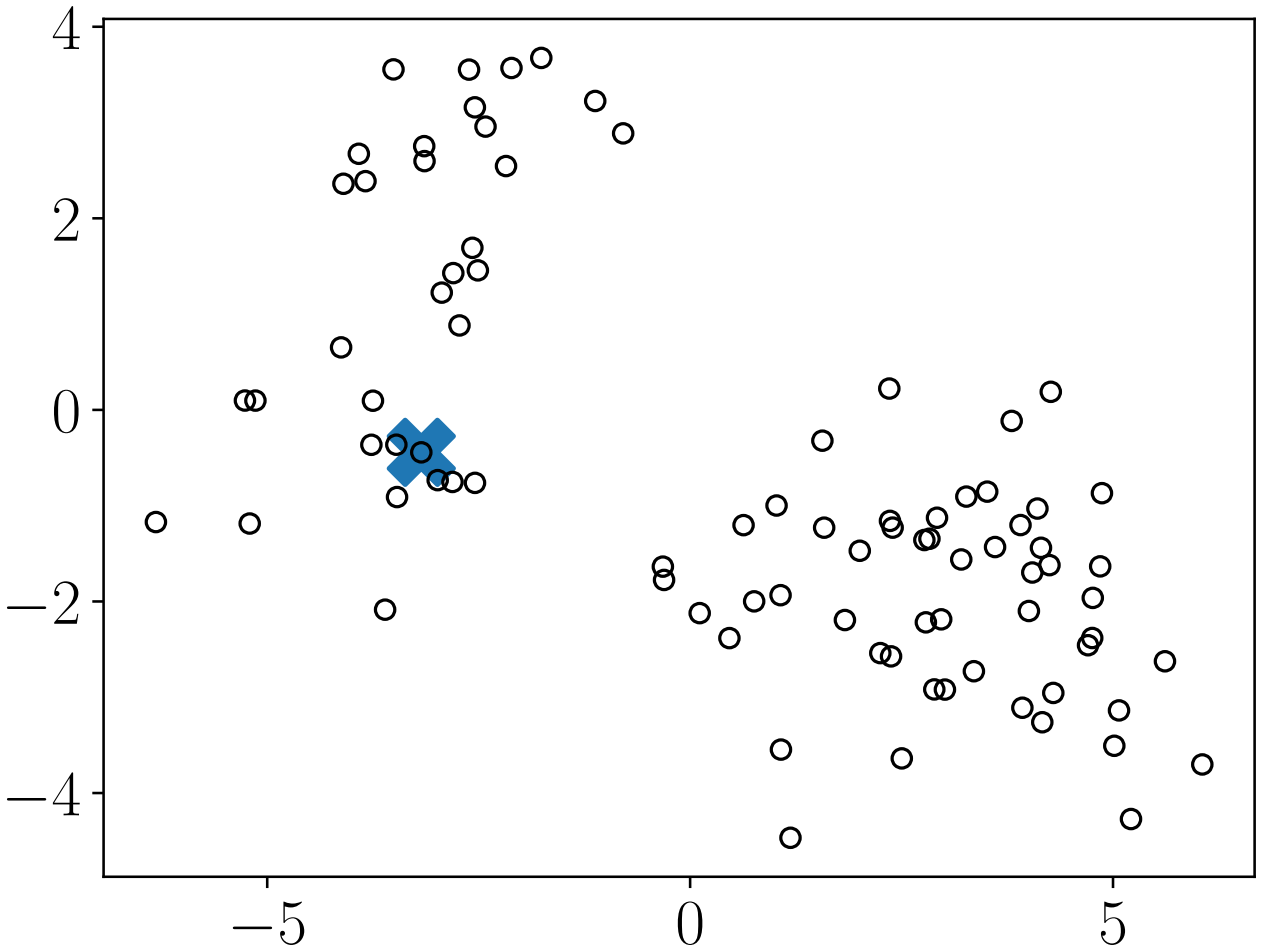
K–Means is sensitive to intialization



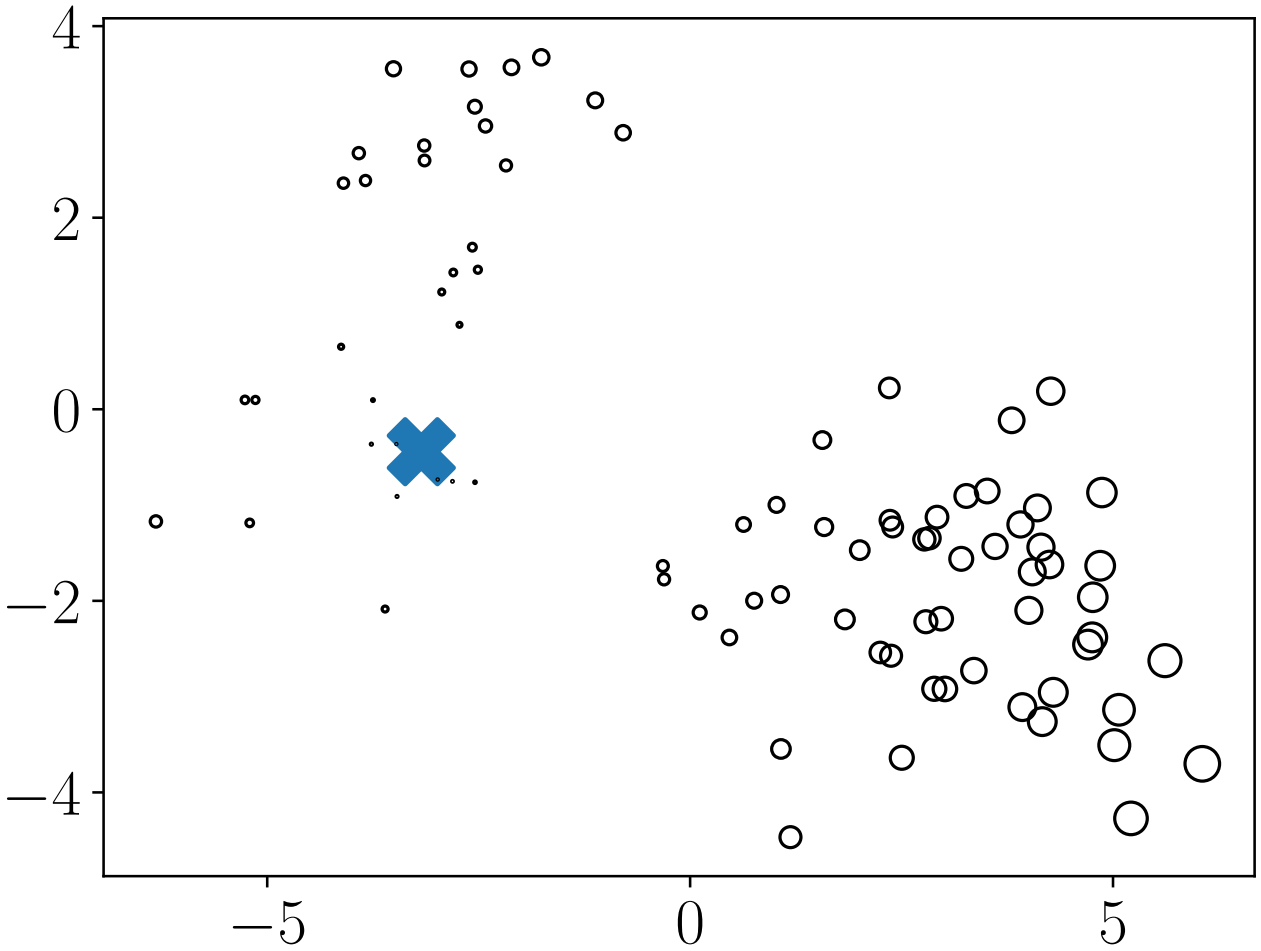
K–Means++: modify initial selection so that centroids are better spread

1. Select randomly, with uniform probability, the first centroid from the data
$$M = \{\boldsymbol{x}_j\},\ j \sim \mathcal{U}\{1, N\}$$
2. Compute the distance $d(\boldsymbol{x}_i)$ of all points $\boldsymbol{x}_i$ that have not been selected from the closest centroid
$$d(\boldsymbol{x}_i) = \min_{\boldsymbol{\mu} \in M} d(\boldsymbol{x}_i, \boldsymbol{\mu})$$
3. Select randomly the next centroid from the dataset using a probability distribution over samples proportional to $d(\boldsymbol{x})^2$:
$$M = M \cup \{\boldsymbol{x}_j\},\ P(j) \propto d(\boldsymbol{x}_j)^2$$

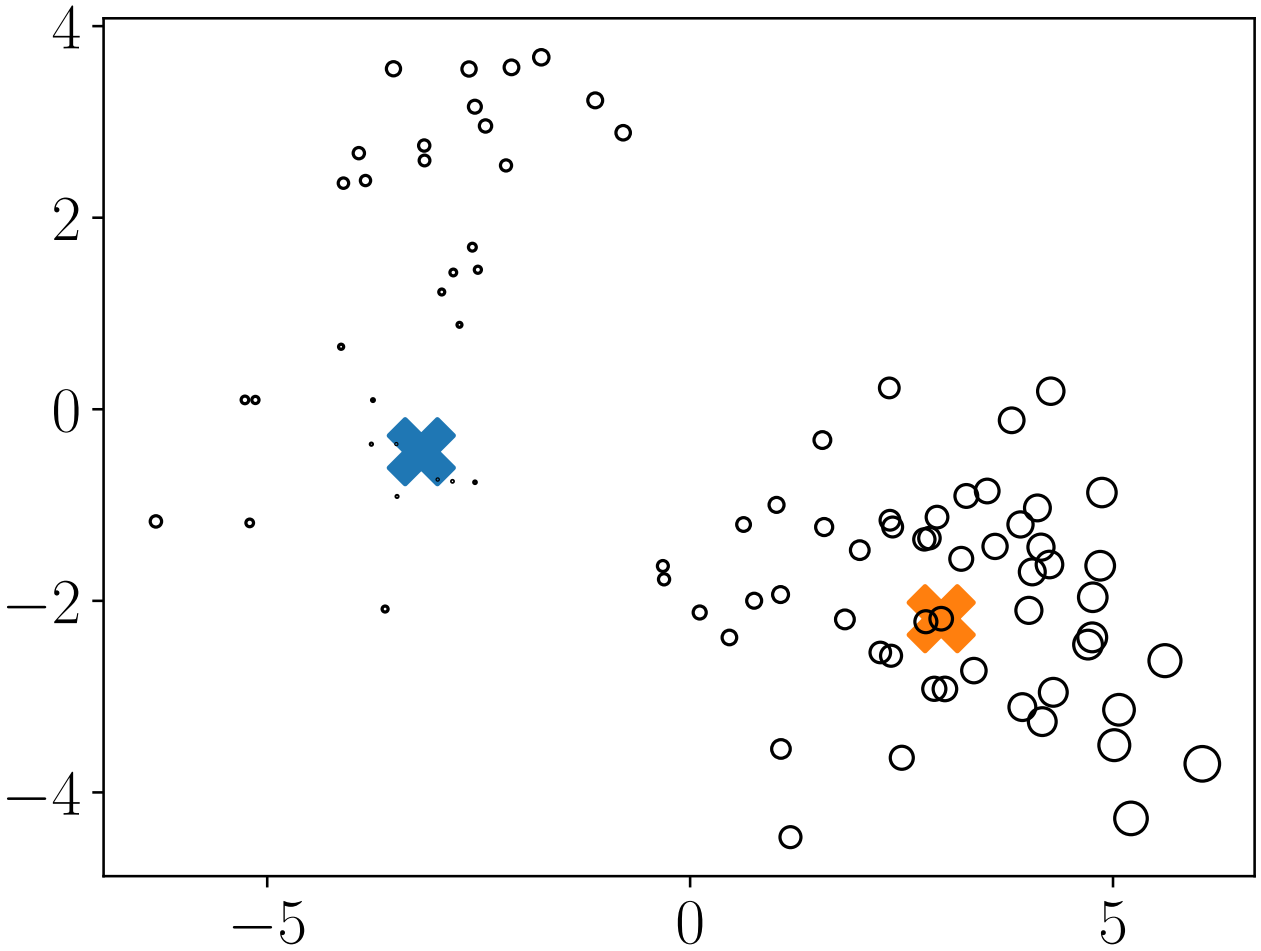
K–Means++: pick uniformly randomly the initial centroid



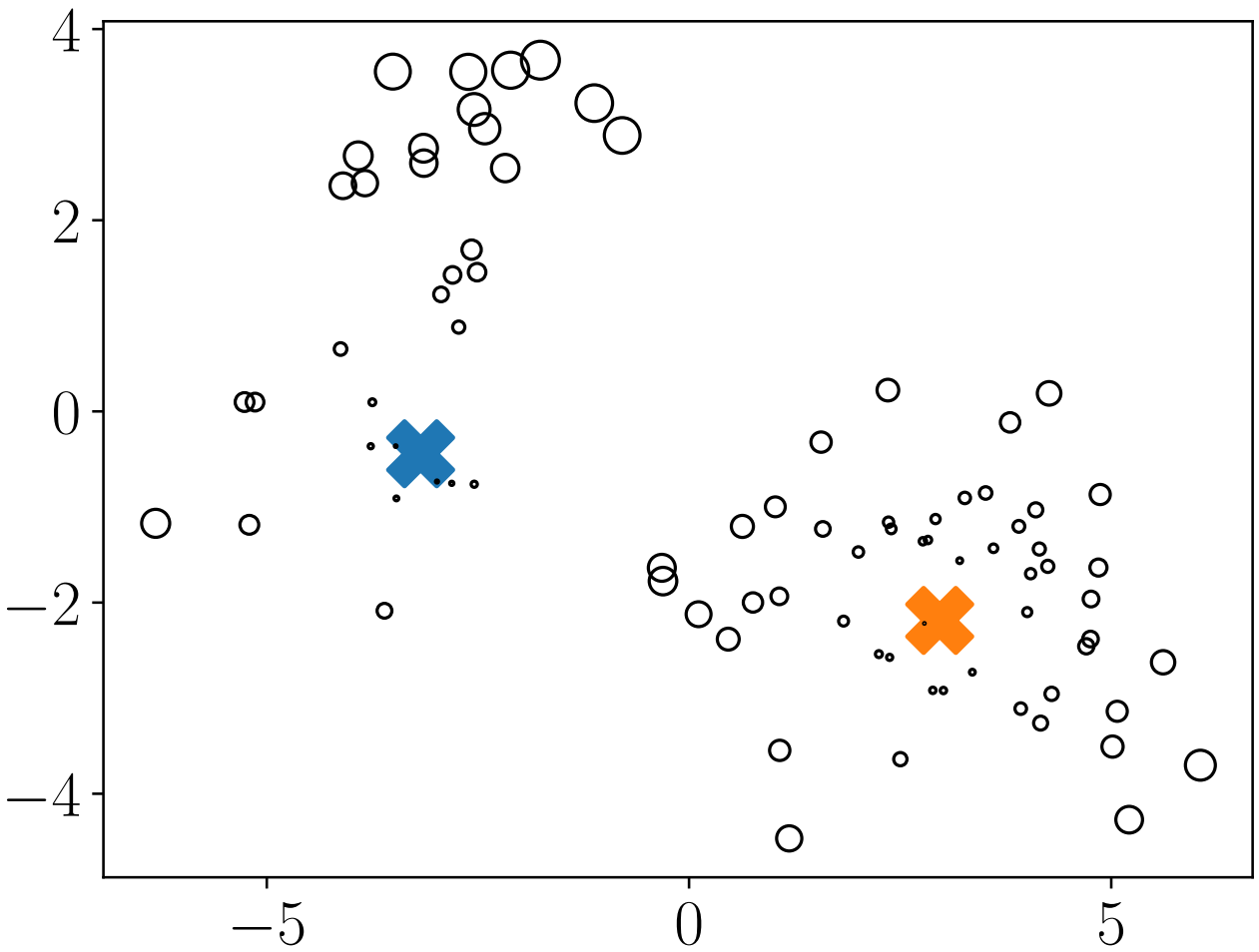
K–Means++: update distribution over samples $P(j)$



K–Means++: pick randomly a sample according to $P(j)$

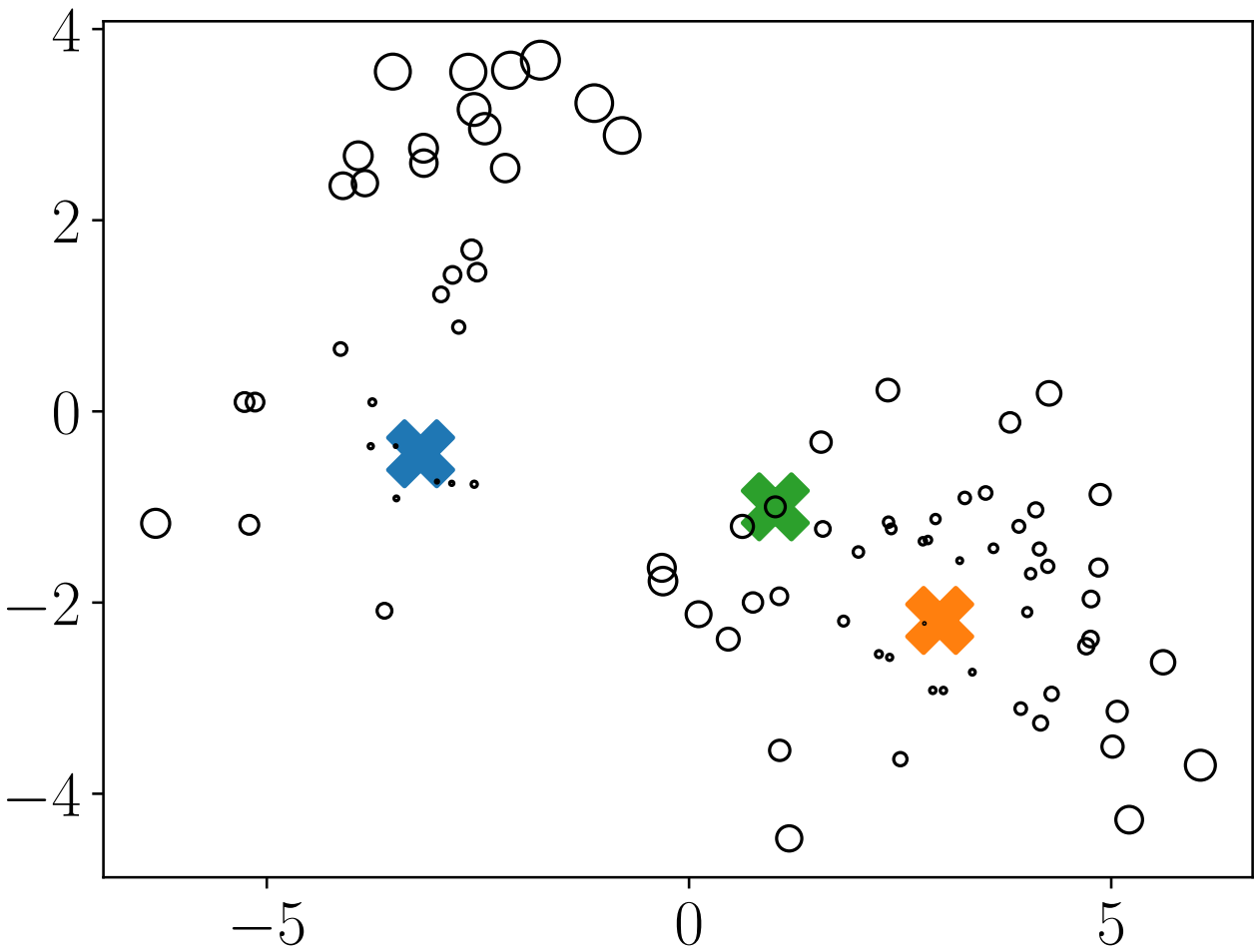


K–Means++: update distribution over samples $P(j)$

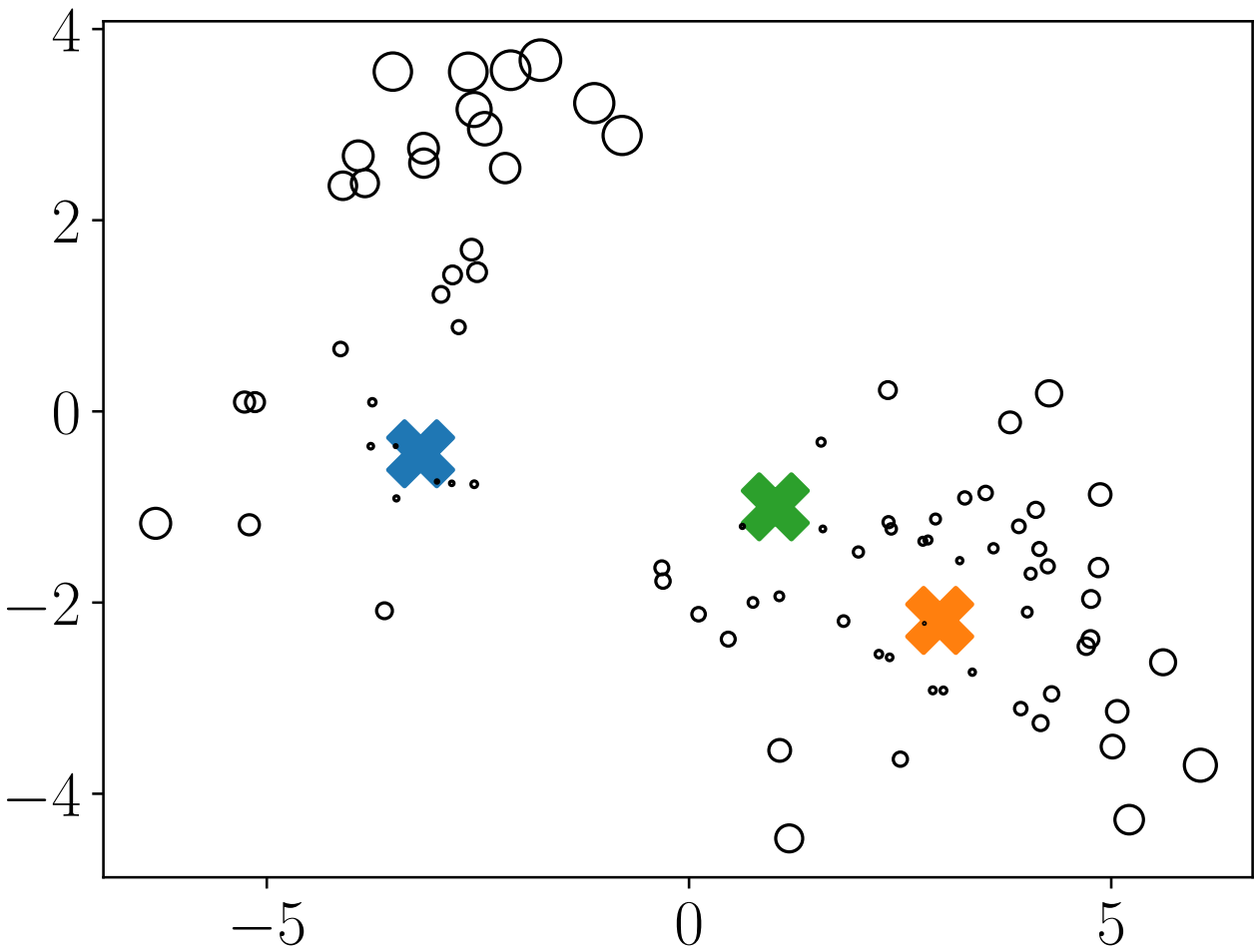


K–Means

K–Means++: pick randomly a sample according to $P(j)$

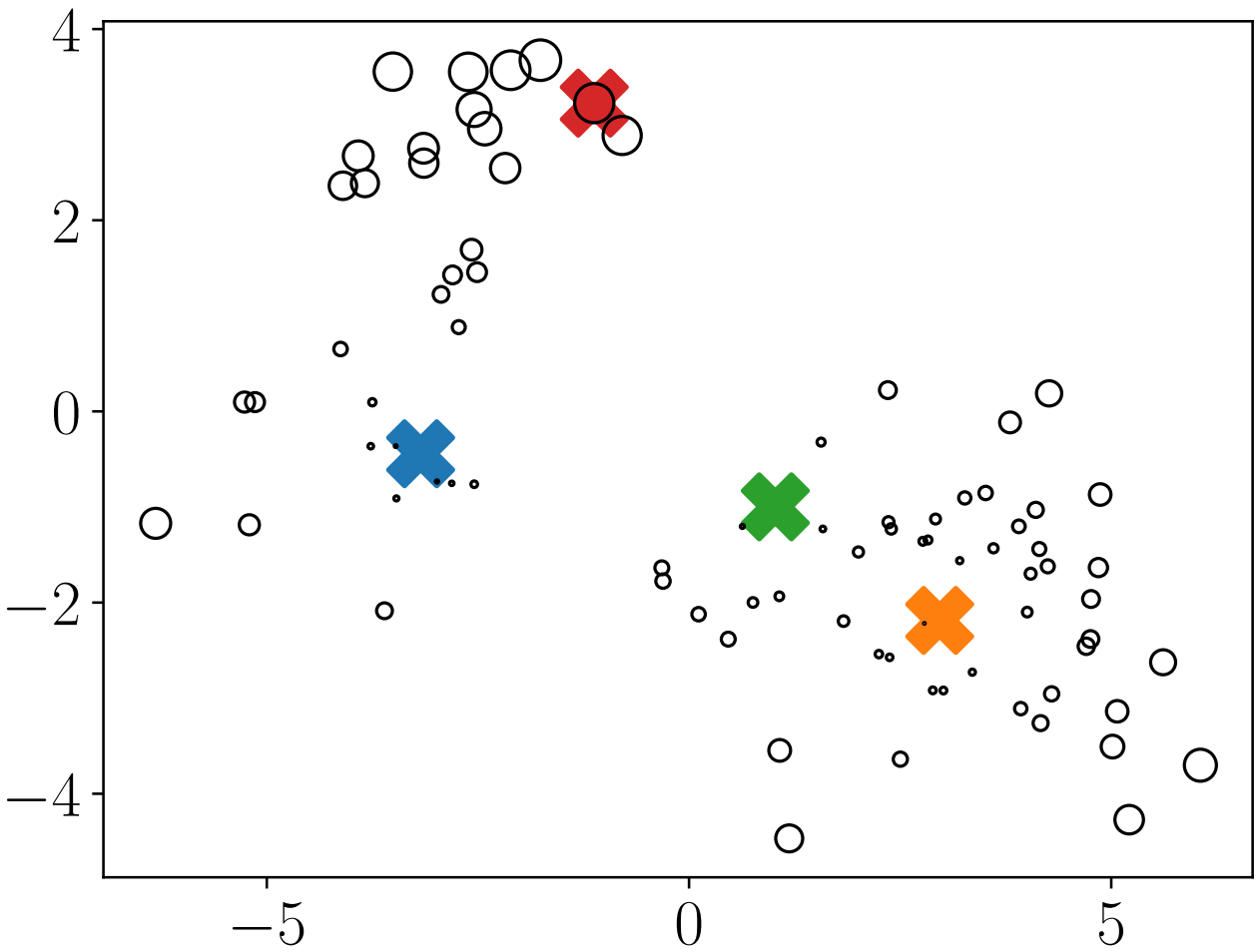


K–Means++: update distribution over samples $P(j)$

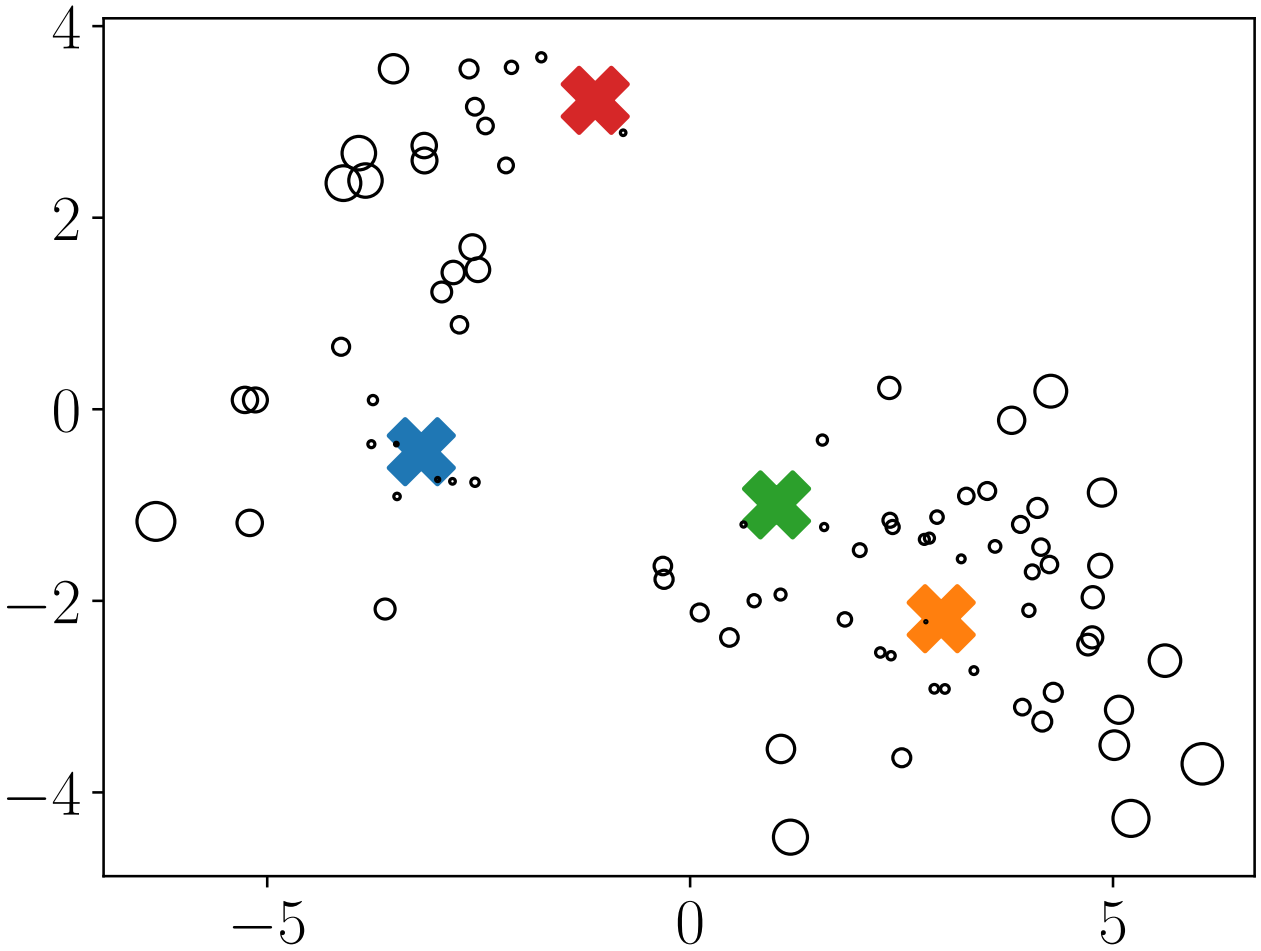


K–Means

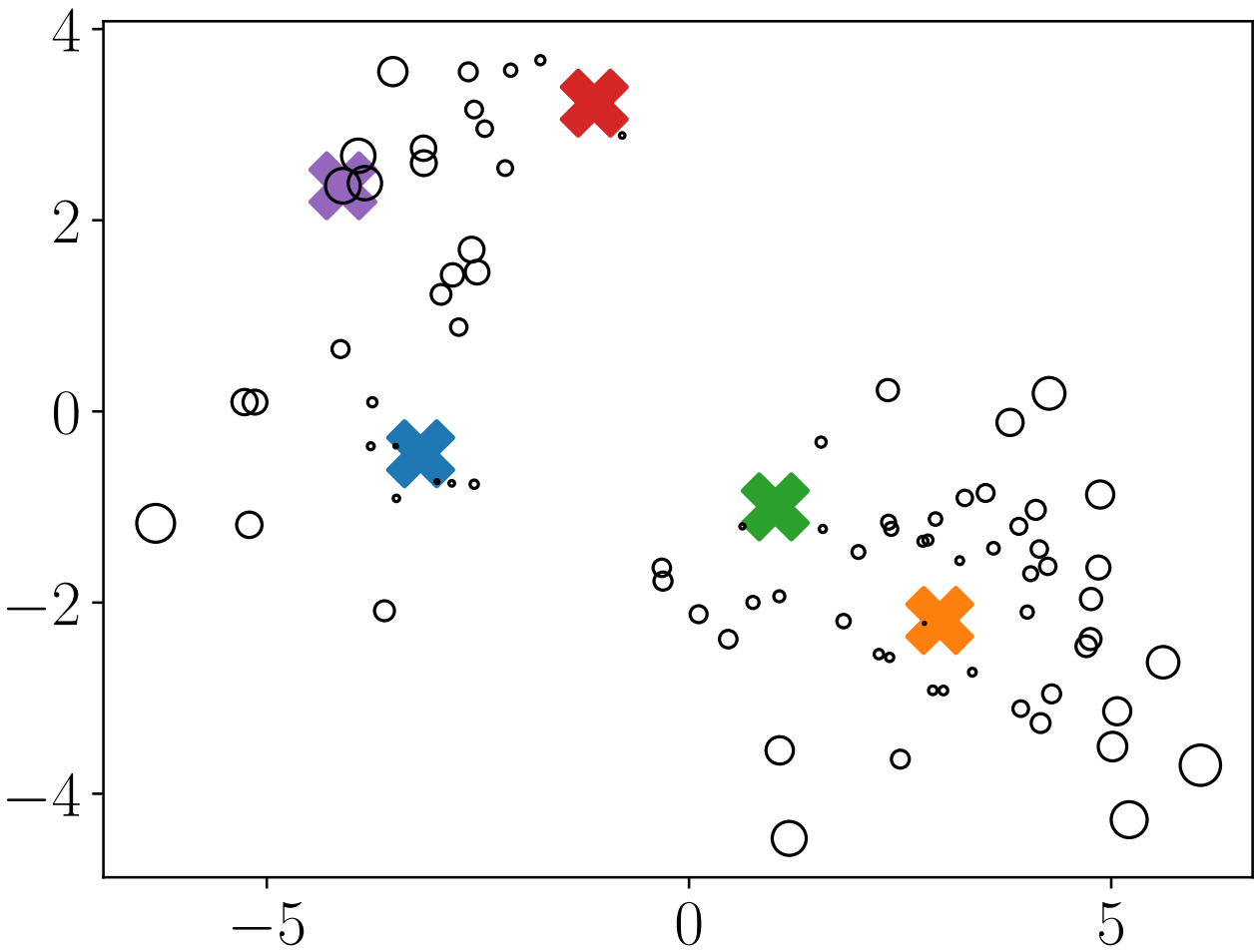
K–Means++: pick randomly a sample according to $P(j)$



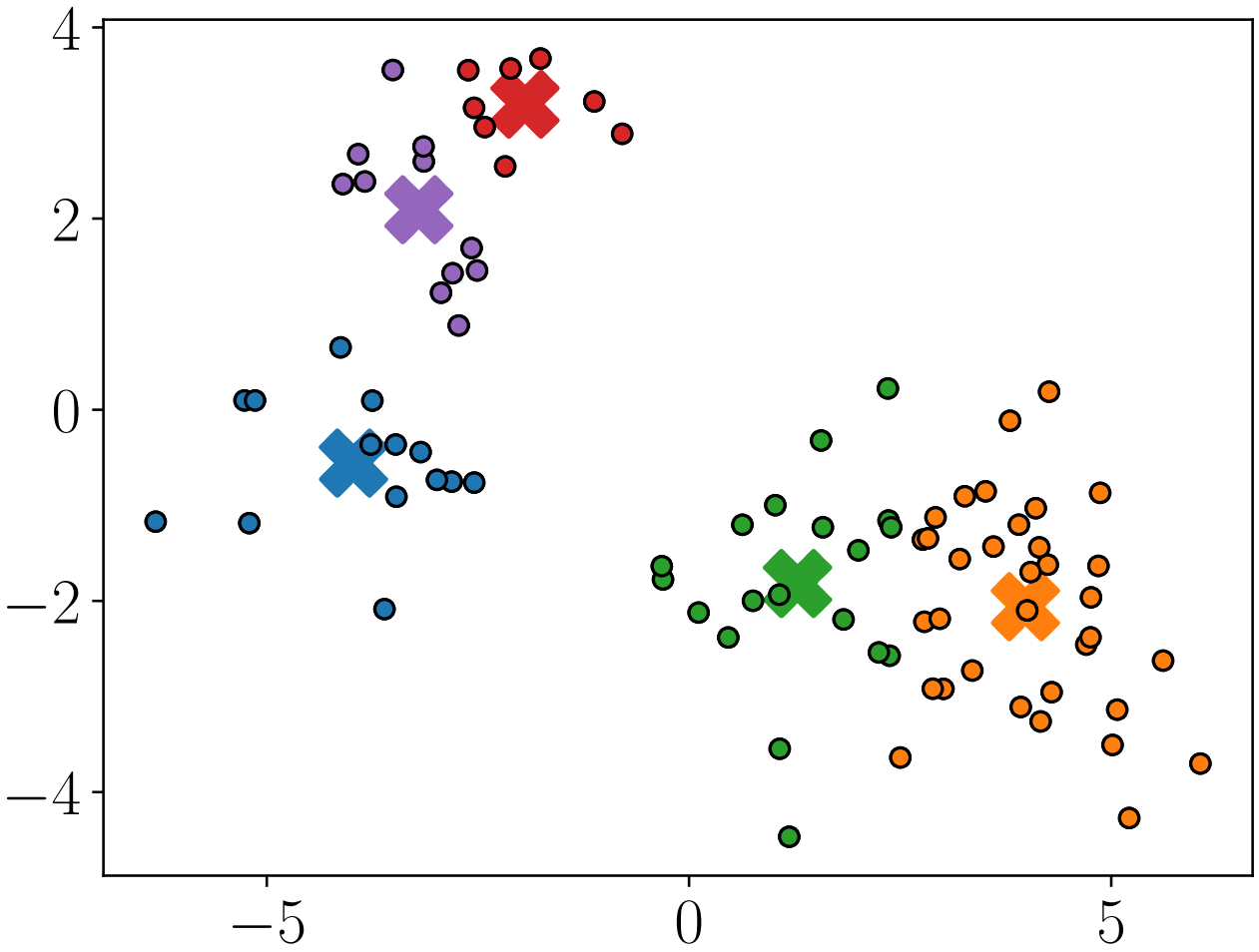
K–Means++: update distribution over samples $P(j)$



K–Means++: pick randomly a sample according to $P(j)$



Apply the K–means algorithm



K–Means

K–Means works well for spherical, similar-shaped clusters (limiting case of isotropic, tied covariance Gaussian mixture models)

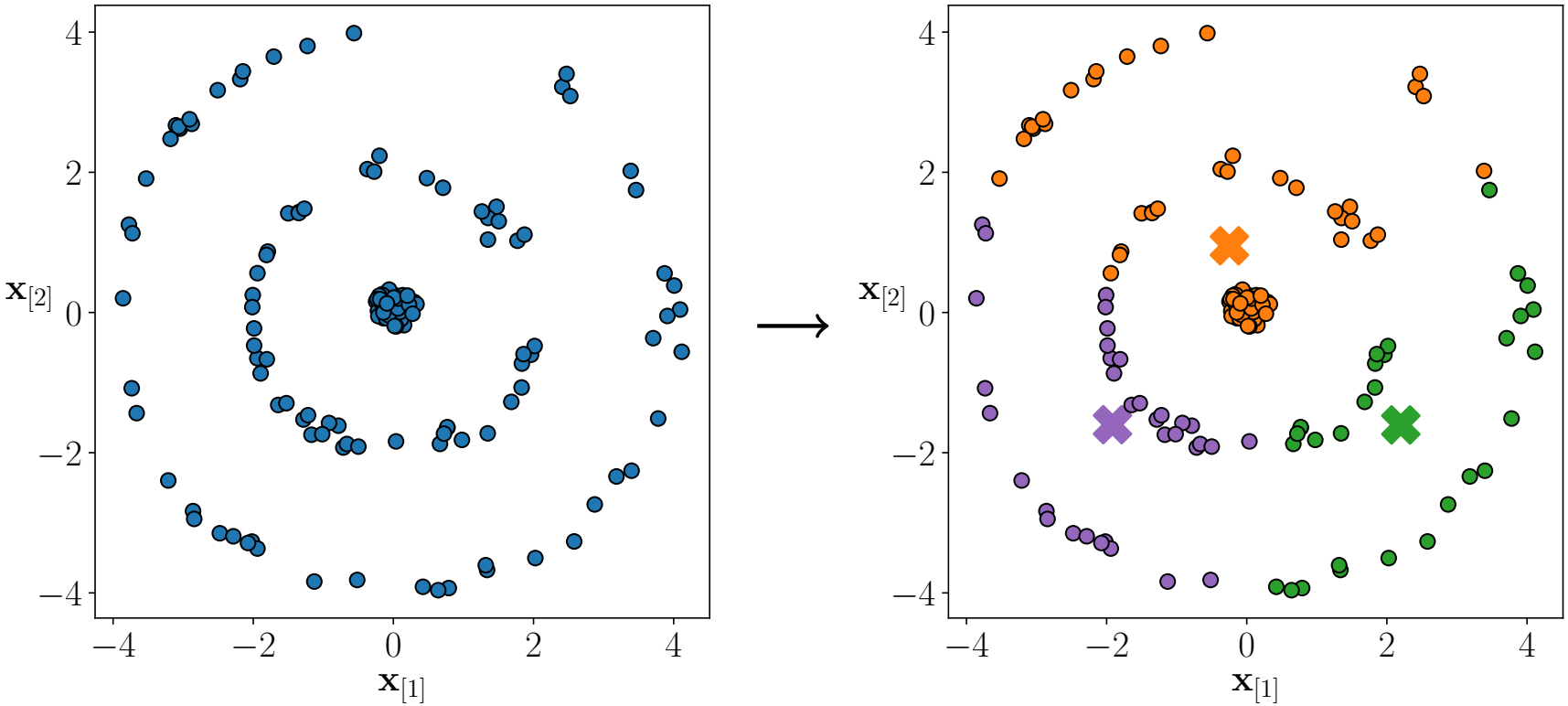
Poor performance for clusters of different size or complex shape

Not invariant to re-scaling

Feature transformations may help (we are changing the definition of distance)

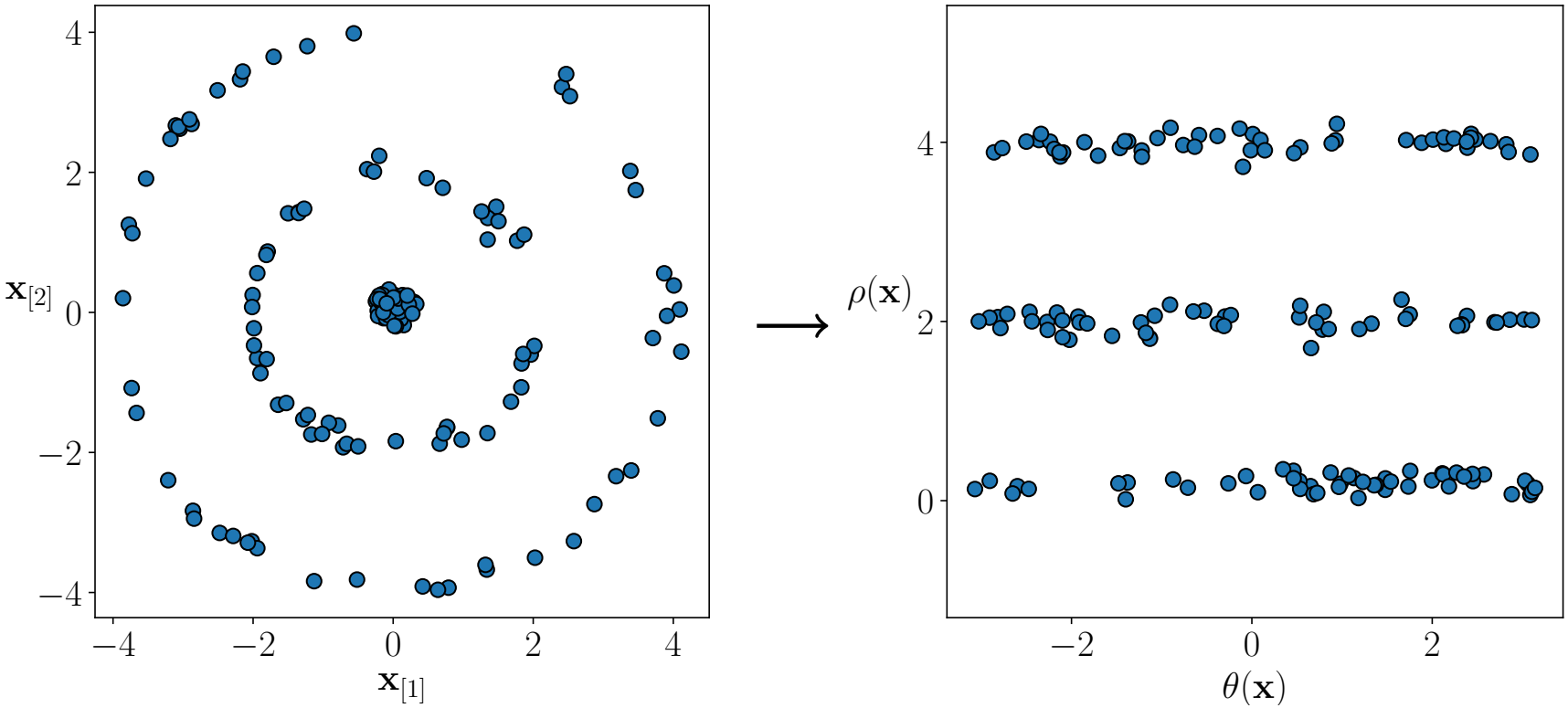
K–Means

K–Means has issues with complex shapes:



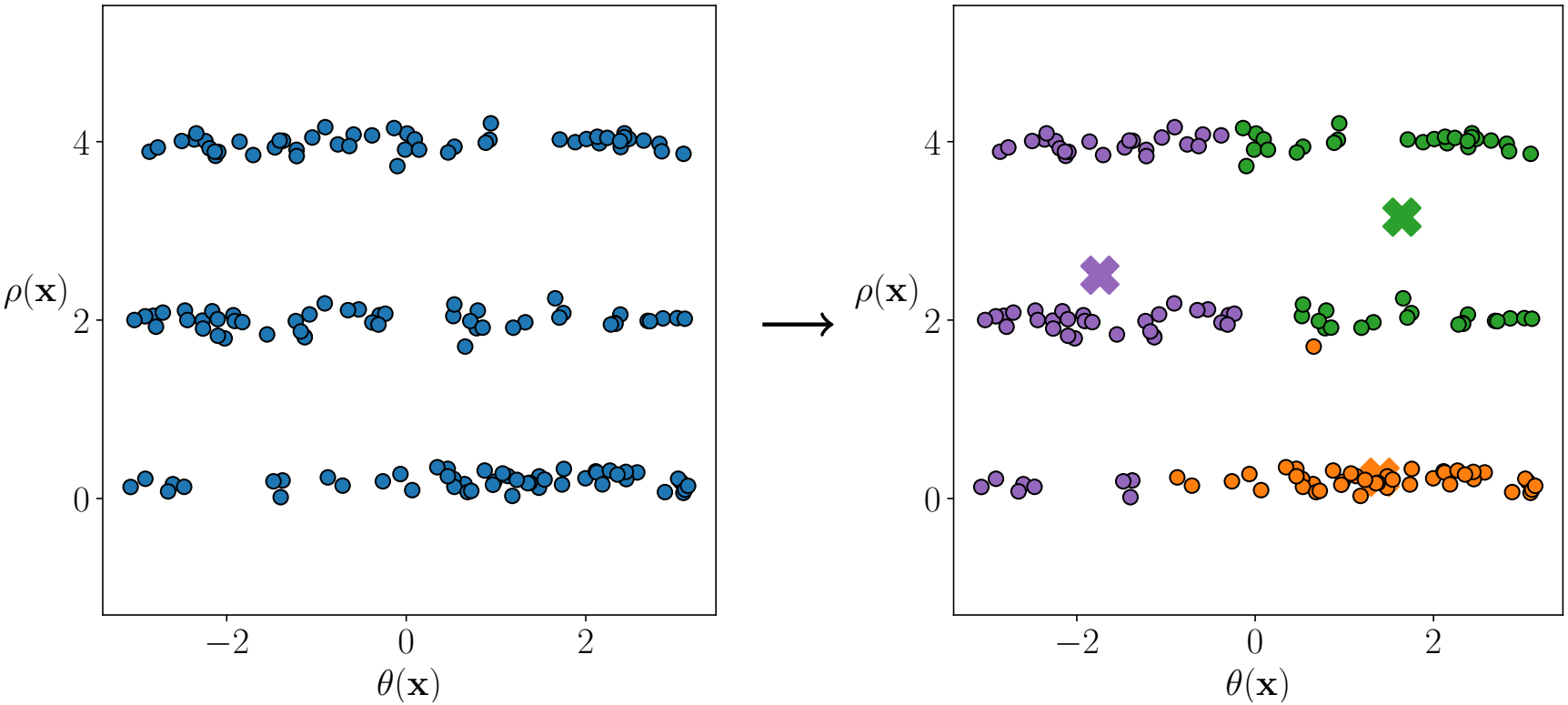
Non-linear data transformation: polar coordinates

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \boldsymbol{x}_{[2]} \end{bmatrix} \rightarrow \begin{bmatrix} \rho(\boldsymbol{x}) = \|\boldsymbol{x}\| \\ \theta(\boldsymbol{x}) = \text{atan2}(\boldsymbol{x}) \end{bmatrix}$$



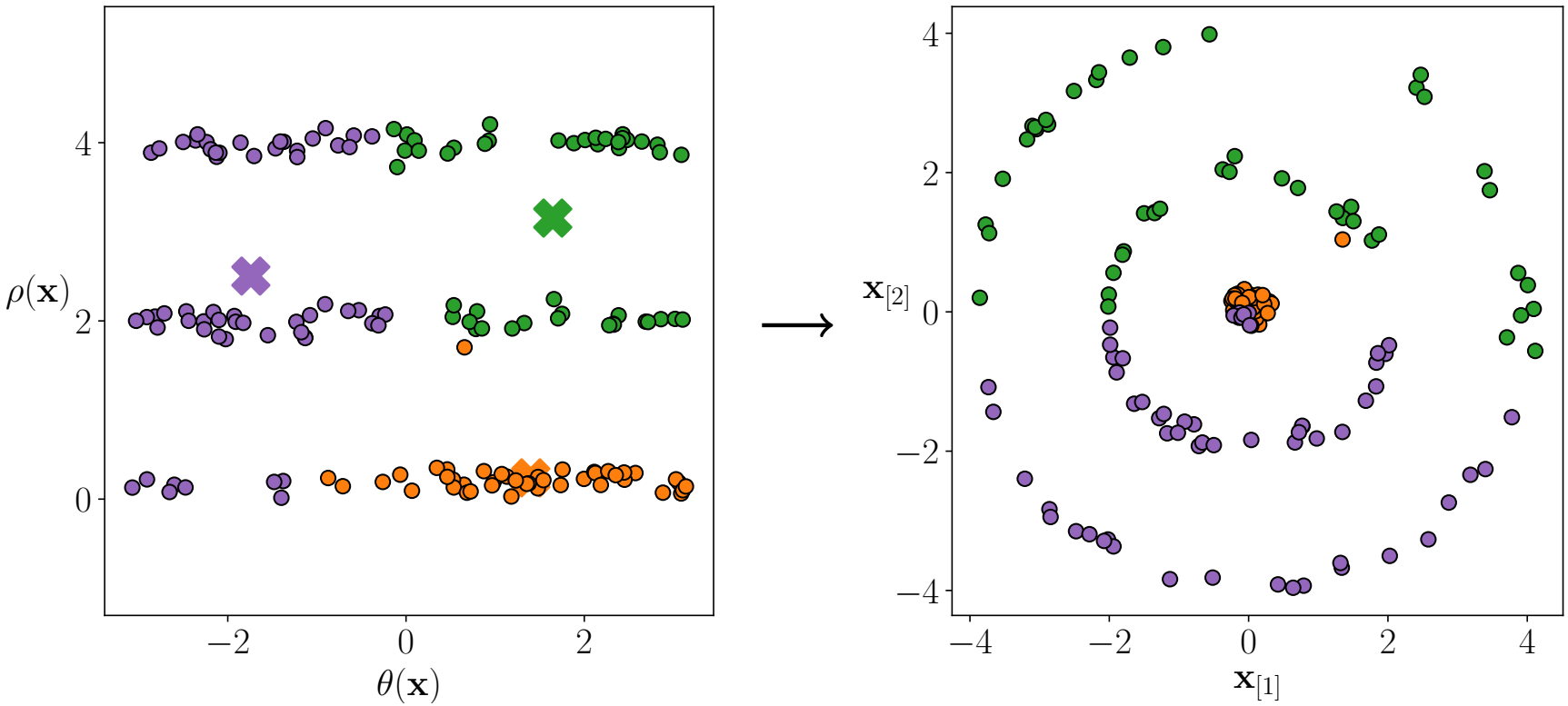
Non-linear data transformation: polar coordinates

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \boldsymbol{x}_{[2]} \end{bmatrix} \rightarrow \begin{bmatrix} \rho(\boldsymbol{x}) = \|\boldsymbol{x}\| \\ \theta(\boldsymbol{x}) = \text{atan2}(\boldsymbol{x}) \end{bmatrix}$$



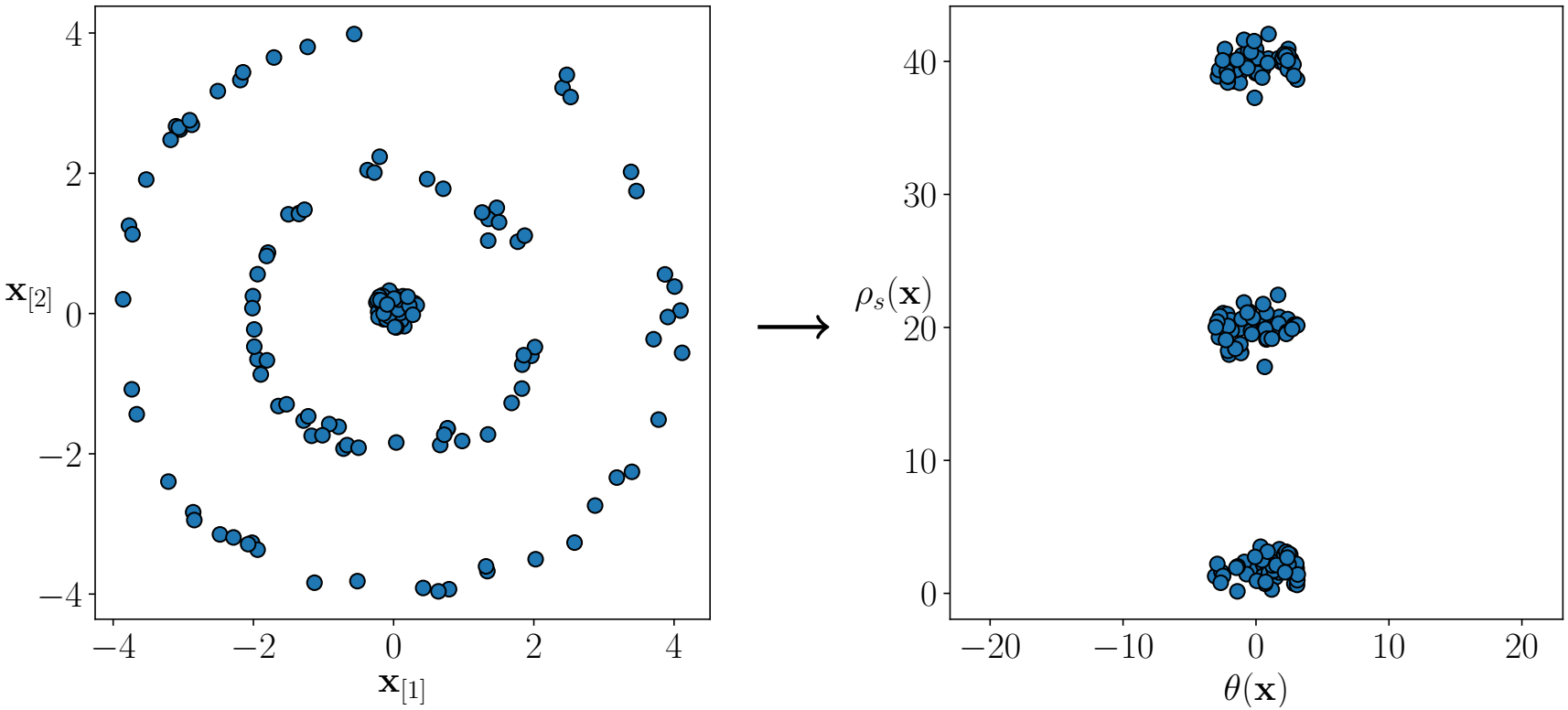
Non-linear data transformation: polar coordinates

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \boldsymbol{x}_{[2]} \end{bmatrix} \rightarrow \begin{bmatrix} \rho(\boldsymbol{x}) = \|\boldsymbol{x}\| \\ \theta(\boldsymbol{x}) = \text{atan2}(\boldsymbol{x}) \end{bmatrix}$$



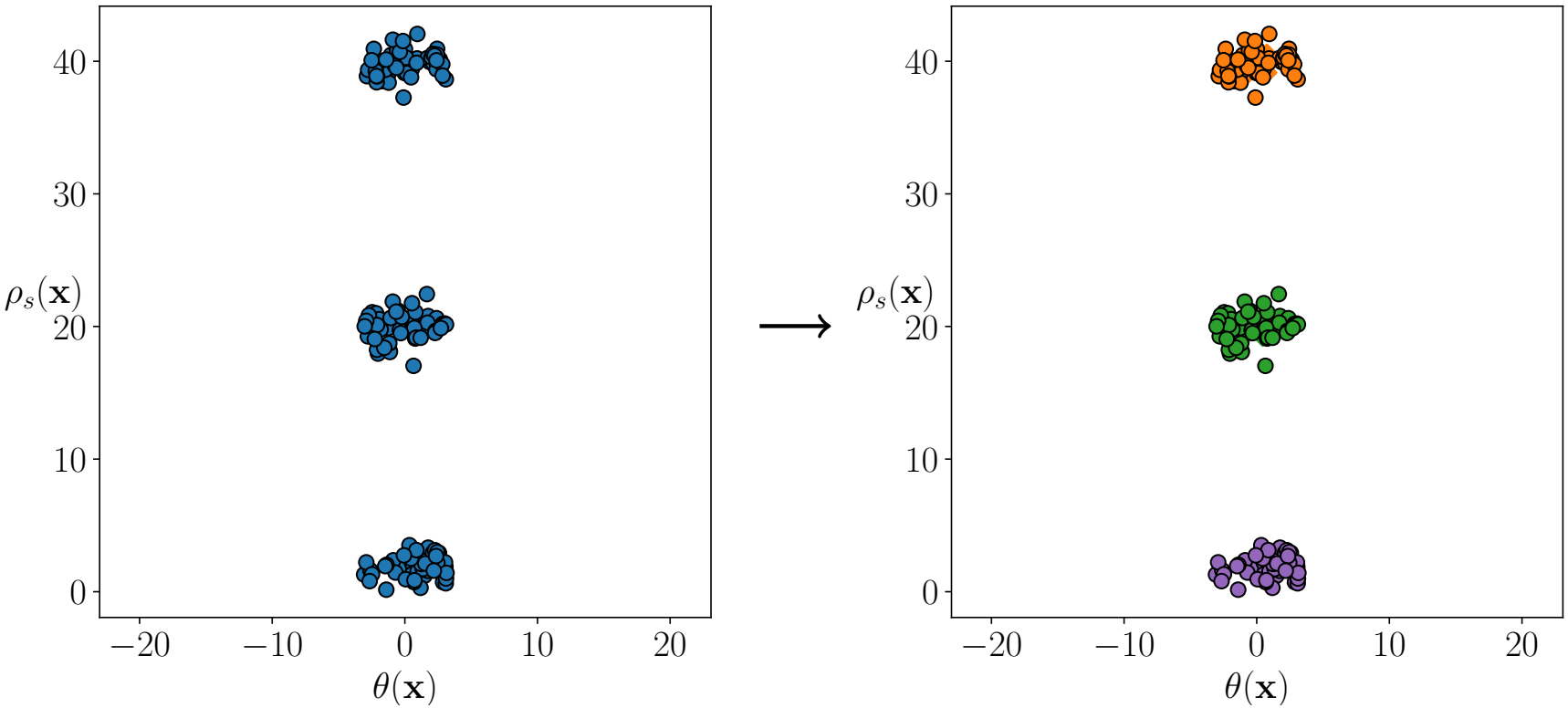
Non-linear data transformation: polar coordinates + scaling

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \boldsymbol{x}_{[2]} \end{bmatrix} \rightarrow \begin{bmatrix} \rho_s(\boldsymbol{x}) = 10\|\boldsymbol{x}\| \\ \theta(\boldsymbol{x}) = \text{atan2}(\boldsymbol{x}) \end{bmatrix}$$



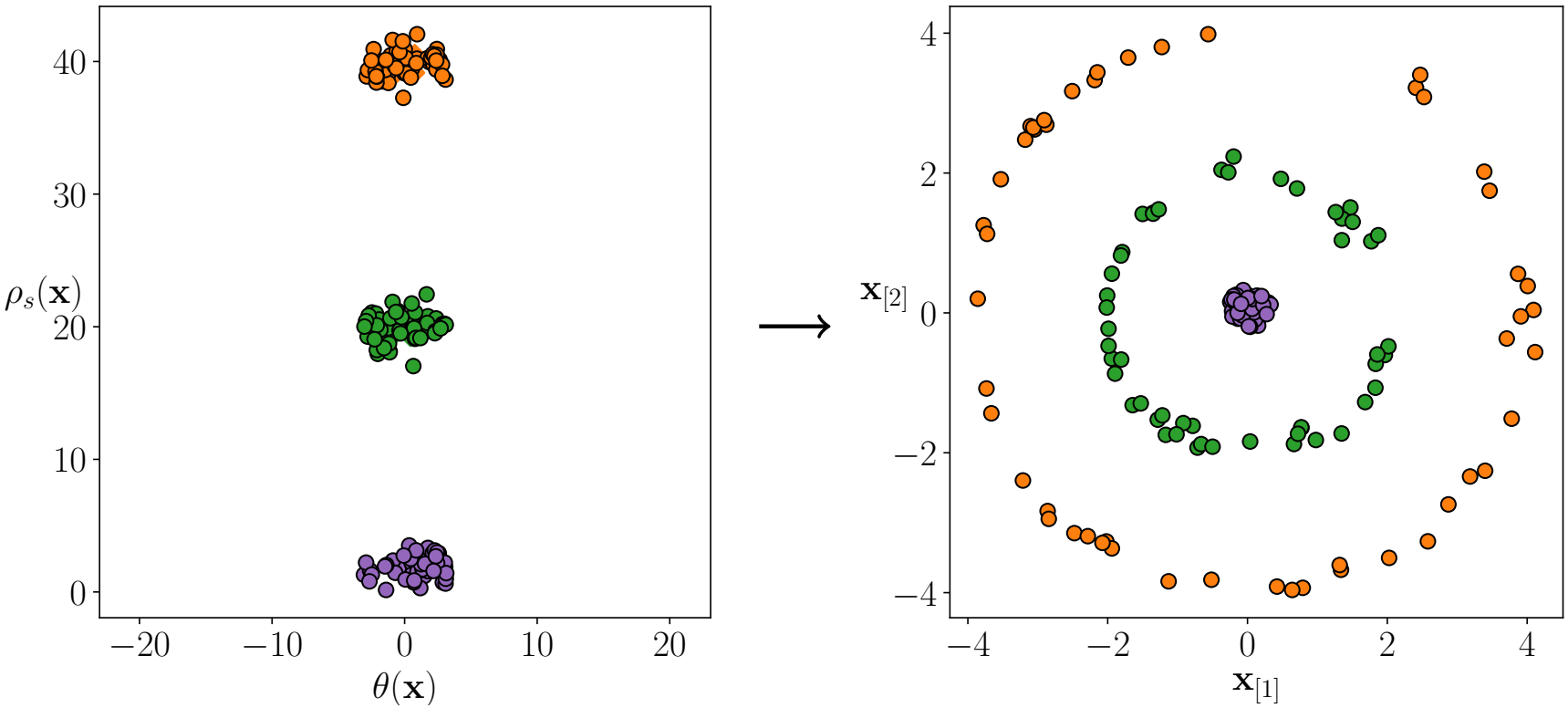
Non-linear data transformation: polar coordinates + scaling

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \boldsymbol{x}_{[2]} \end{bmatrix} \rightarrow \begin{bmatrix} \rho_s(\boldsymbol{x}) = 10\|\boldsymbol{x}\| \\ \theta(\boldsymbol{x}) = \text{atan2}(\boldsymbol{x}) \end{bmatrix}$$



Non-linear data transformation: polar coordinates + scaling

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \boldsymbol{x}_{[2]} \end{bmatrix} \rightarrow \begin{bmatrix} \rho_s(\boldsymbol{x}) = 10\|\boldsymbol{x}\| \\ \theta(\boldsymbol{x}) = \text{atan2}(\boldsymbol{x}) \end{bmatrix}$$



K–Means

Optimal number of clusters?

Usually based on heuristics

- “Elbow” method (explained variance / distortion)
- Silhouette

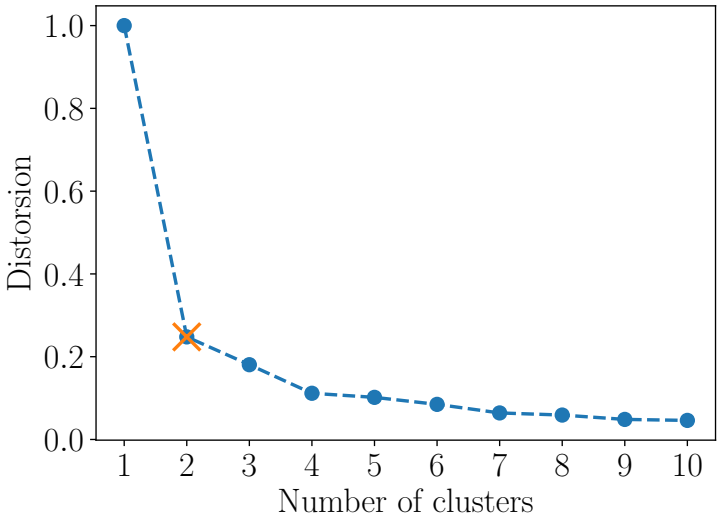
Since these are heuristics, there is no guarantee to obtain good results

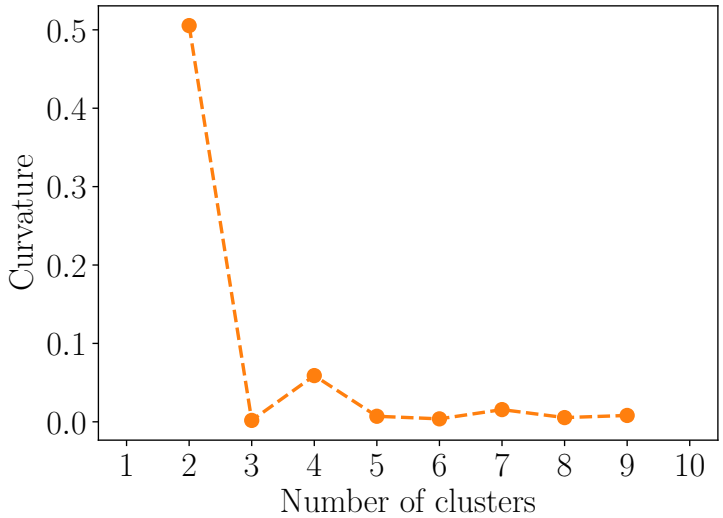
Elbow method

- Try with different number of clusters
- As the number of clusters increases, the clusters become more compact
- Measure the within-cluster variation: mean squared distance of each sample from the centroid of its cluster (also called distortion)
- Plot the distortion as a function of the number of clusters
- Heuristic: choose the number of clusters corresponding to an “elbow”

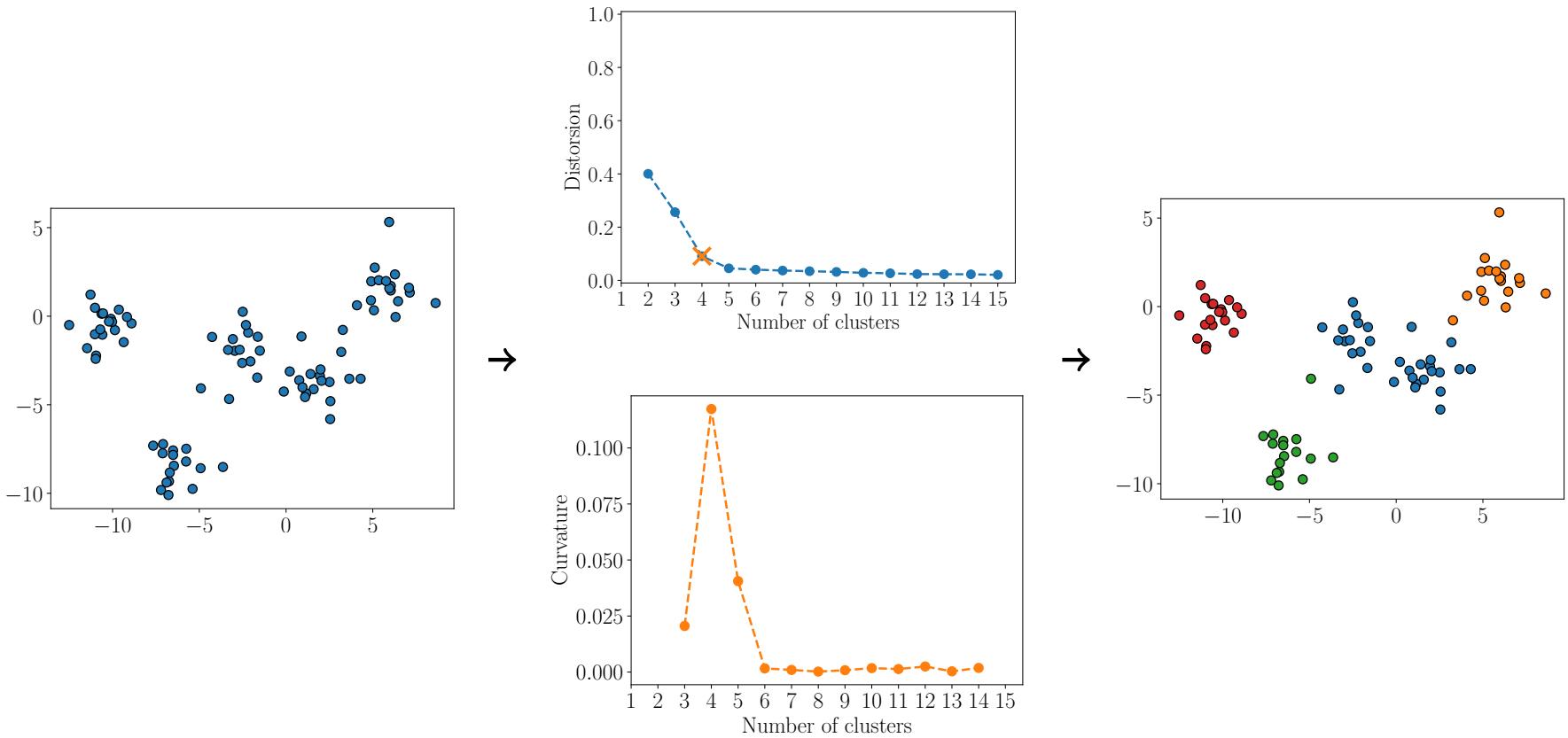
Elbow method

- Up to the elbow, increasing clusters significantly reduces the distortion
- After the elbow, the gains become relatively smaller
- The elbow can be located as a point of maximum curvature





Elbow method — example:



Silhouette

Silhouette: measure of similarity of a sample to its own cluster, compared to other clusters

Can be applied to other approaches (e.g. agglomerative clustering)

Allows computing a silhouette score for a given number of clusters

To select the optimal number of clusters, we try different values and select the value that gives the highest silhouette score

Silhouette

For each point $\boldsymbol{x}_i$ compute:

- Mean intra-cluster distance a_i — distance of $\boldsymbol{x}_i$ w.r.t. other points of the same cluster:

$$a_i = \frac{1}{|C(\boldsymbol{x}_i)| - 1} \sum_{\boldsymbol{x}_j \in C(\boldsymbol{x}_i) \setminus \{\boldsymbol{x}_i\}} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Mean nearest-cluster distance b_i — distance of $\boldsymbol{x}_i$ from the nearest cluster C_J

$$b_i = \min_{C | C \neq C(\boldsymbol{x}_i)} \frac{1}{|C|} \sum_{\boldsymbol{x}_j \in C} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Define the silhouette value for sample $\boldsymbol{x}_i$ as

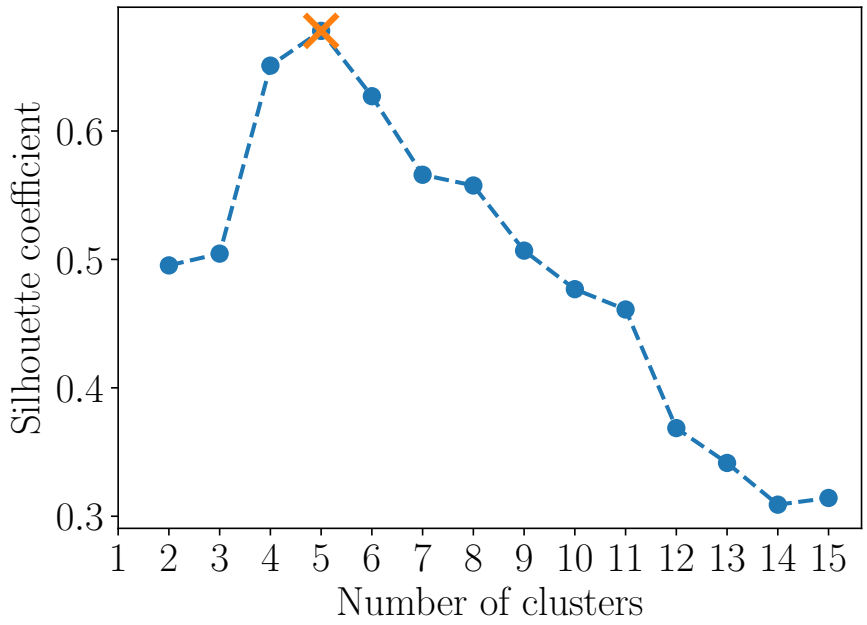
$$s_i = \begin{cases} \frac{b_i - a_i}{\max(a_i, b_i)} & |C(\boldsymbol{x}_i)| > 1 \\ 0 \quad \text{otherwise} \end{cases}$$

Silhouette

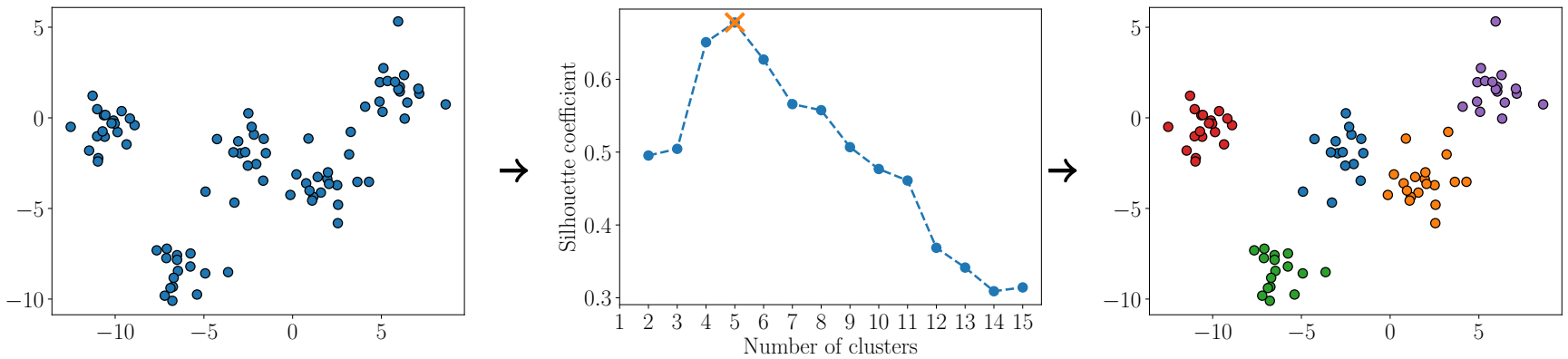
The silhouette score is the average over all samples of the silhouette values s_i:

$$s = \frac{1}{N} \sum_i s_i$$

Select the number of clusters correspoding to the largest silhouette value



Silhouette method — example:



Agglomerative clustering

Build a hierarchy of nested sub-clusters

With suitable linkage functions can use arbitrary distance metrics

At the beginning, all samples are put in their own cluster. The set of clusters is $\mathcal{C}$

$$C_i = \{\boldsymbol{x}_i\}\ , \forall i \in 1 \dots n\ , \quad \mathcal{C} = \{C_1 \dots C_n\}$$

Let $d^{clu}(C_i, C_j)$ be a distance between clusters (linkage function)

Agglomerative clustering

Iteratively merge the two "closest" clusters, until a single cluster remains

- Compute all pairwise cluster distances $d^{clu}(C_i, C_j)$ between clusters in $\mathcal{C}$
- Merge the clusters that have smallest cluster distance until a single cluster remains

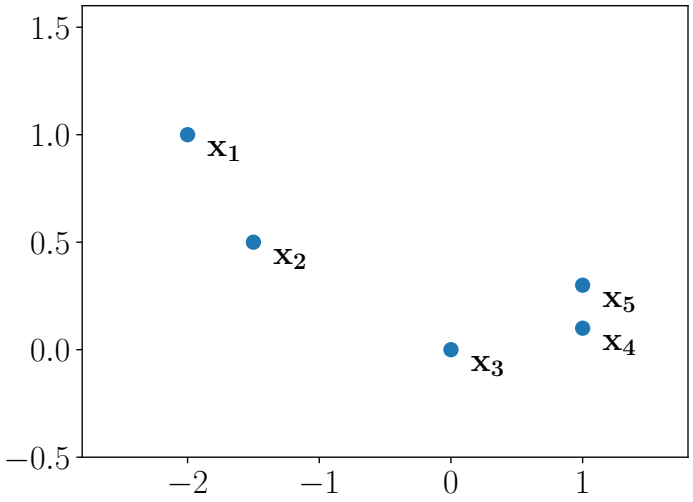
$$C_a, C_b = \arg\min_{C_i, C_j \in \mathcal{C}} d(C_i, C_j)$$

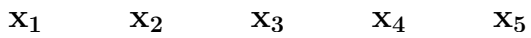
$$C_{new} = C_a \cup C_b$$

$$\mathcal{C} = (\mathcal{C} \setminus \{C_a, C_b\}) \cup \{C_{new}\}$$

Agglomerative clustering

Iteratively merge the two “closest” clusters, until a single cluster remains

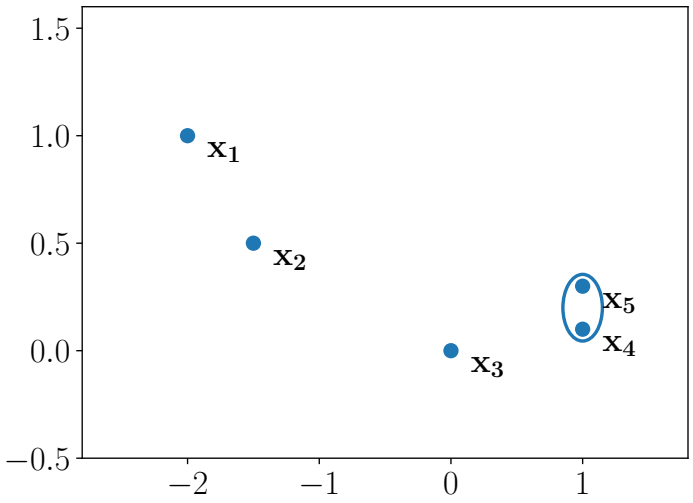


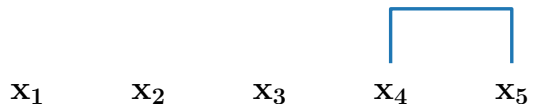


$$\mathcal{C} = \{\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4\}, \{\boldsymbol{x}_1\}\}$$

Agglomerative clustering

Iteratively merge the two “closest” clusters, until a single cluster remains

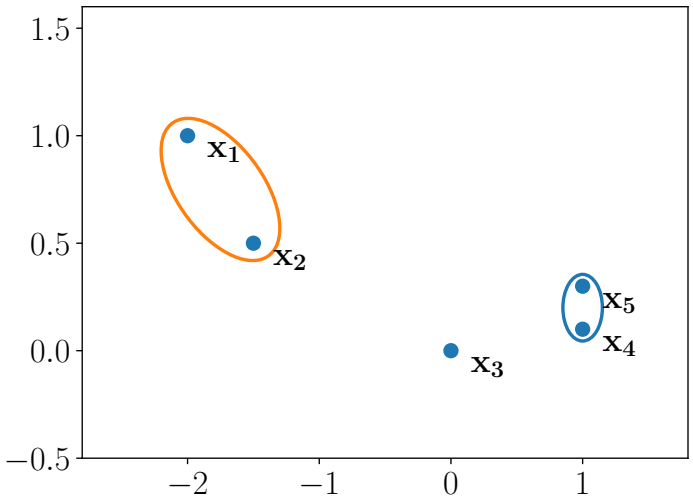


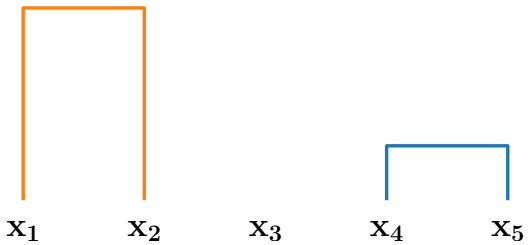


$$\mathcal{C} = \{\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}\}$$

Agglomerative clustering

Iteratively merge the two “closest” clusters, until a single cluster remains

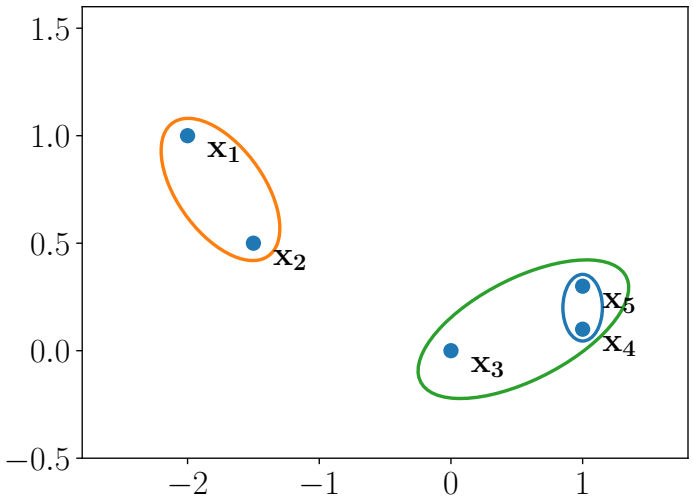


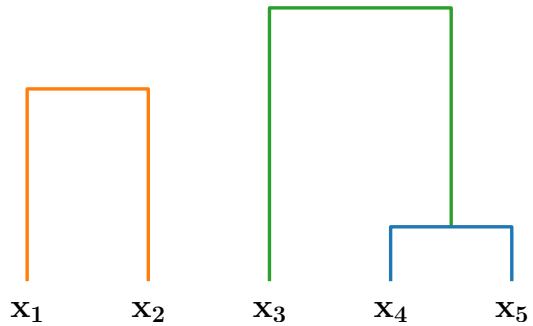


$$\mathcal{C} = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}\}$$

Agglomerative clustering

Iteratively merge the two “closest” clusters, until a single cluster remains

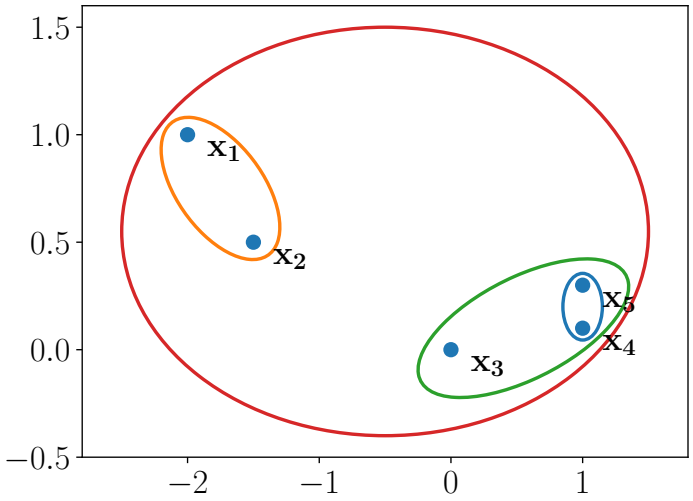


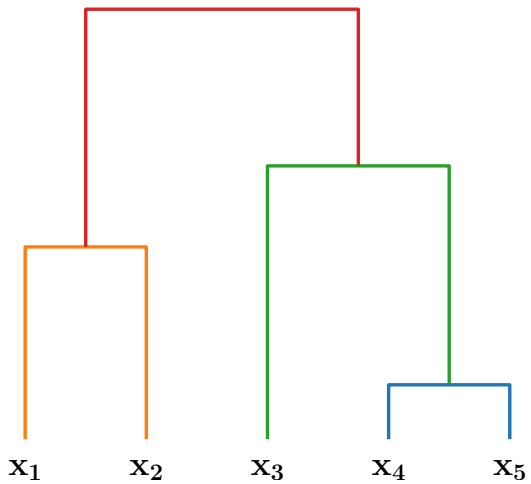


$$\mathcal{C} = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}\}$$

Agglomerative clustering

Iteratively merge the two “closest” clusters, until a single cluster remains





$$\mathcal{C} = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}\}$$

Agglomerative clustering

Let $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ denote the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

How can we define a "distance" between clusters?

For singleton clusters it's trivial: $d^{clu}(\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2\}) = d(\boldsymbol{x}_1, \boldsymbol{x}_2)$

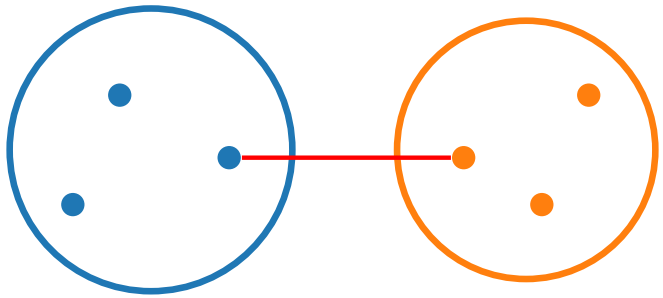
For clusters that contain several points we have different possibilities

- Single linkage
- Complete linkage
- Average linkage (UPGMA)
- Centroid
- ...

Agglomerative clustering

Single linkage: smallest of all pairwise distances of cluster samples (the distance of the closest points of the two clusters)

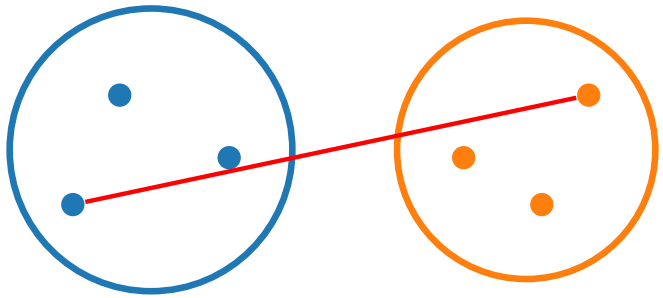
$$d^{clu}(C_i, C_j) = \min_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y})$$



Agglomerative clustering

Complete linkage: largest of all pairwise distances of cluster samples (the distance of the farthest points of the two clusters)

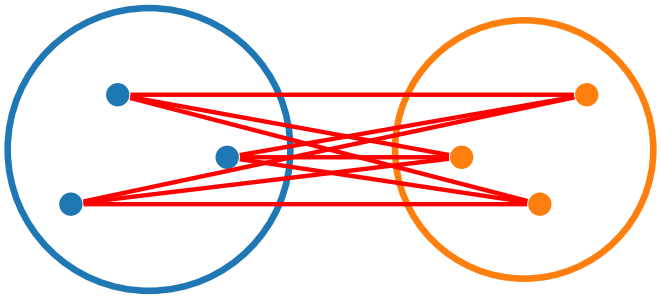
$$d^{clu}(C_i, C_j) = \max_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y})$$



Agglomerative clustering

Unweighted average linkage (UPGMA): average of all pairwise distances of cluster samples

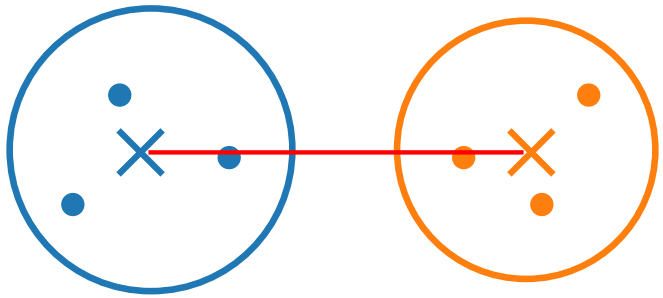
$$d^{clu}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y})$$



Agglomerative clustering

Centroid (Euclidean distance): Euclidean distance between the cluster centroids

$$d^{clu}(C_i, C_j) = \left\| \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x} - \frac{1}{|C_j|} \sum_{\boldsymbol{y} \in C_j} \boldsymbol{y} \right\|$$



Agglomerative clustering

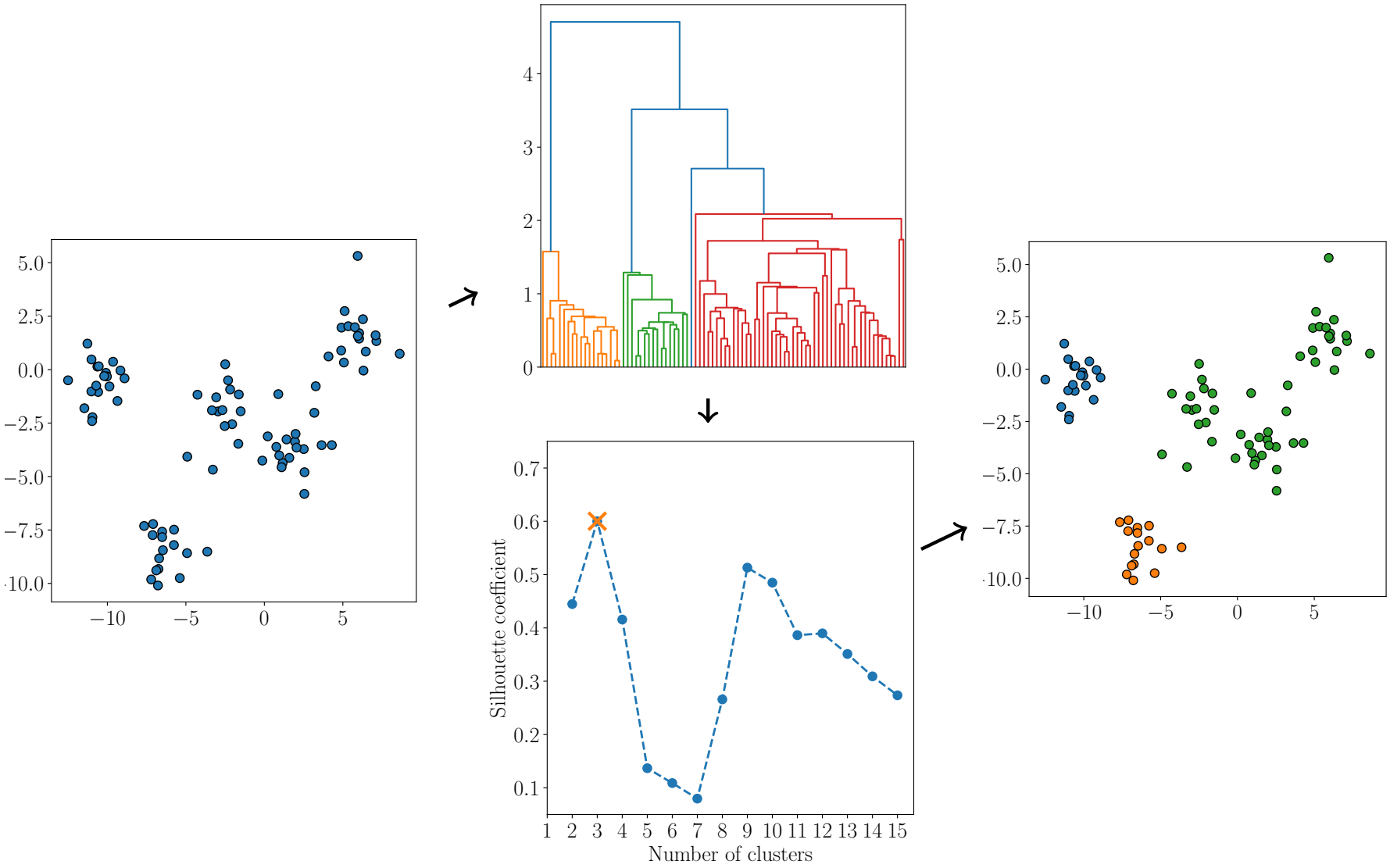
The dendrogram shows the cluster hierarchy

The height at which two clusters are merged corresponds to the distance between the two clusters (cophenetic distance)

Flat clusters can be obtained by cutting the dendrogram at a given level (e.g. through silhouette analysis)

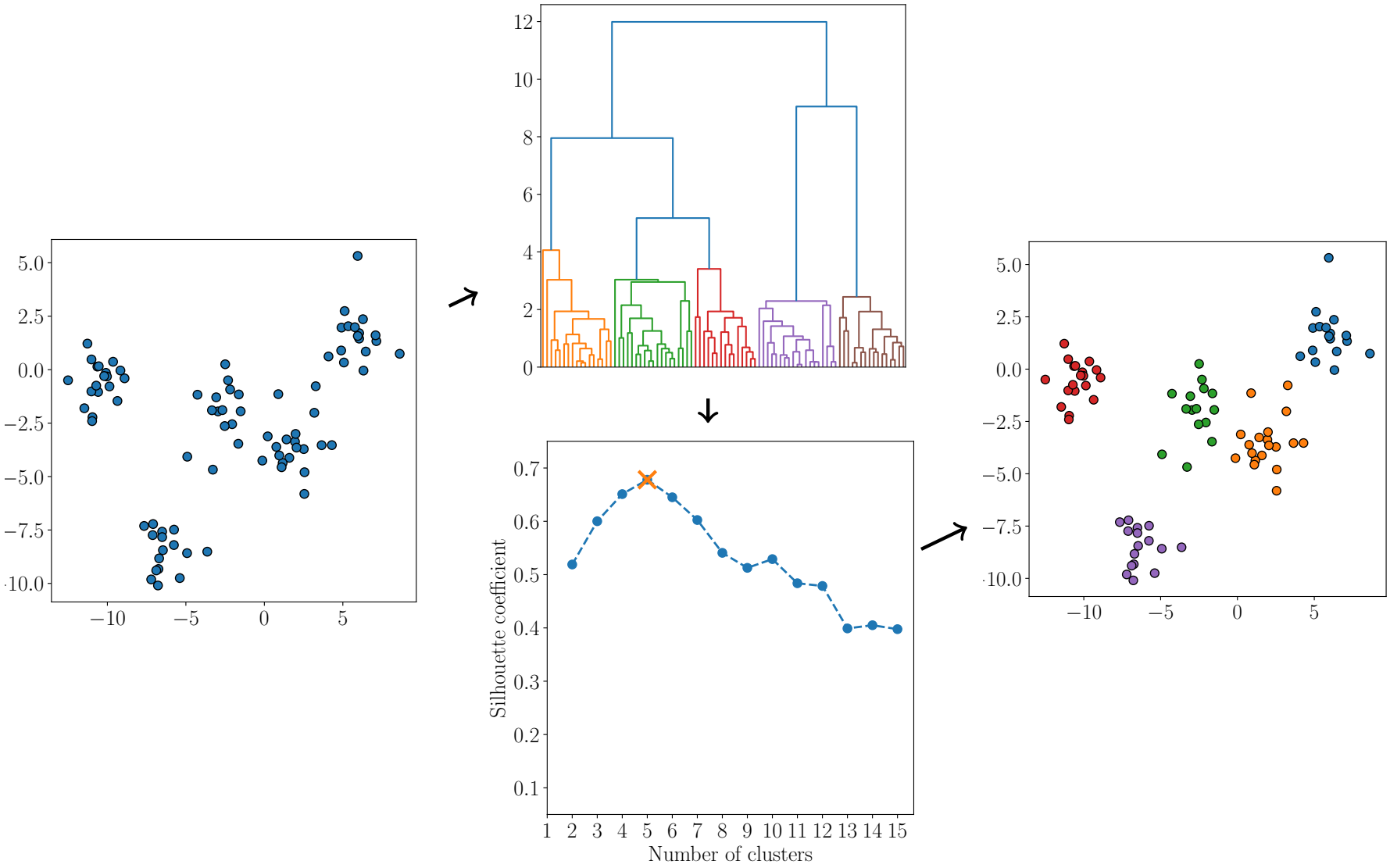
Agglomerative clustering

Linkage: single



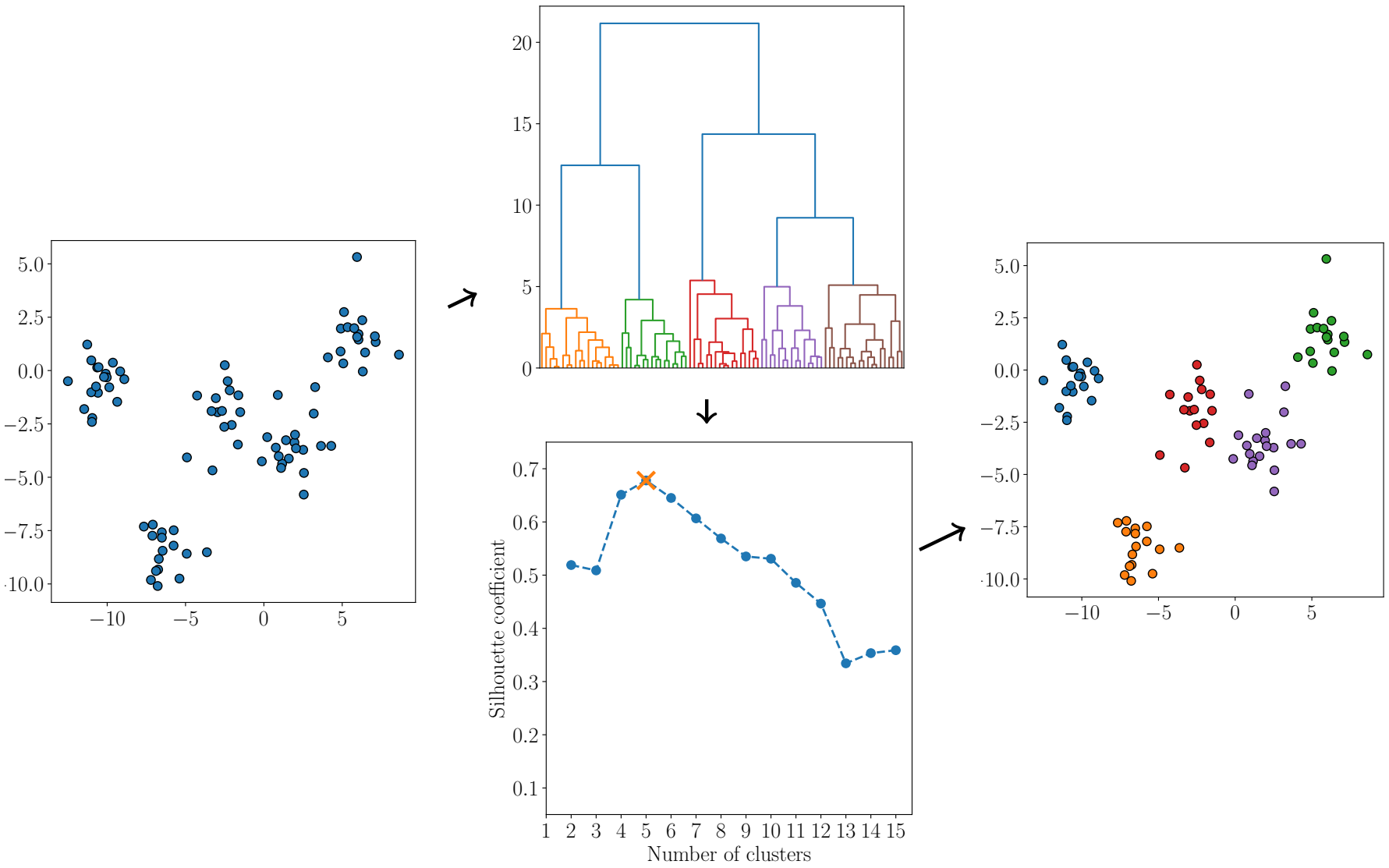
Agglomerative clustering

Linkage: average



Agglomerative clustering

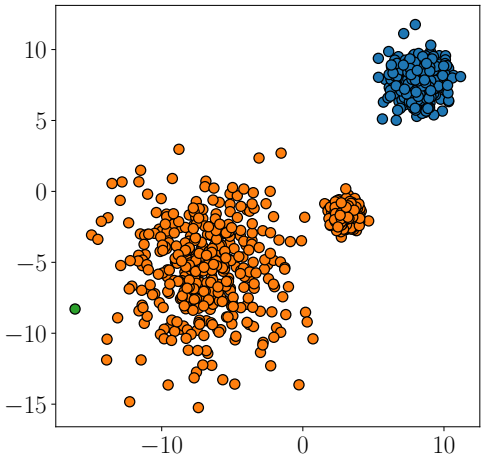
Linkage: complete



Agglomerative clustering

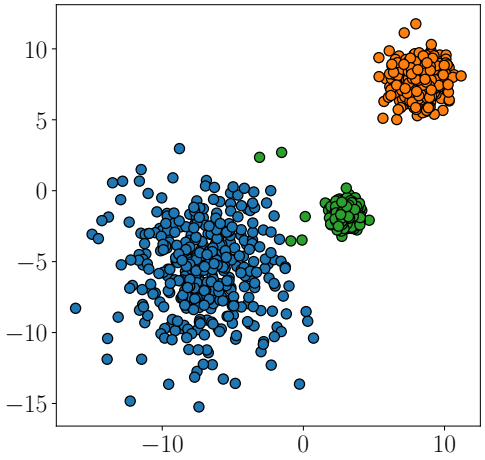
Single linkage:

- Non-globular clusters
- Sensitive to noise



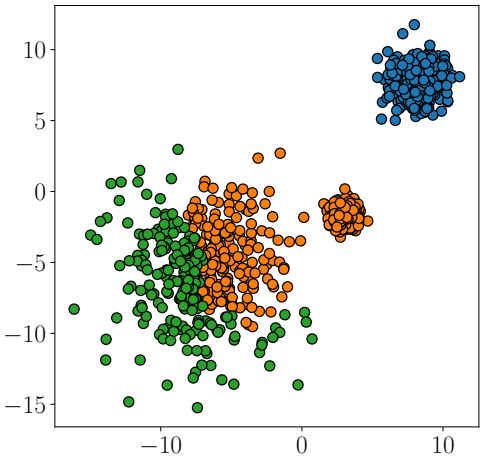
Average linkage:

- Globular clusters
- Less sensitive to noise



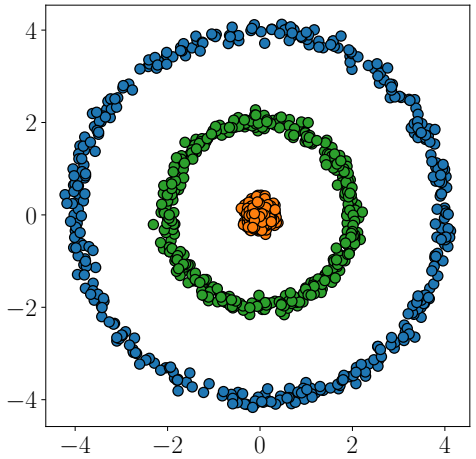
Complete linkage:

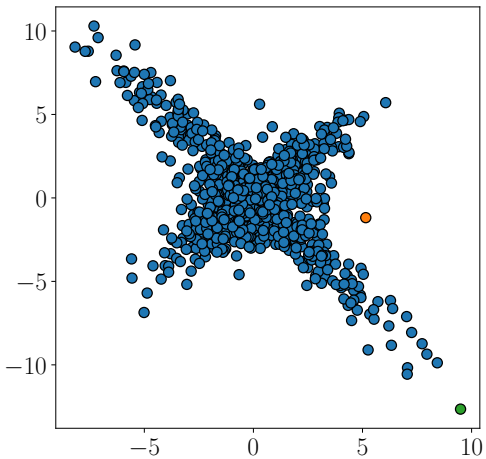
- Globular, compact clusters
- Tends to break large clusters



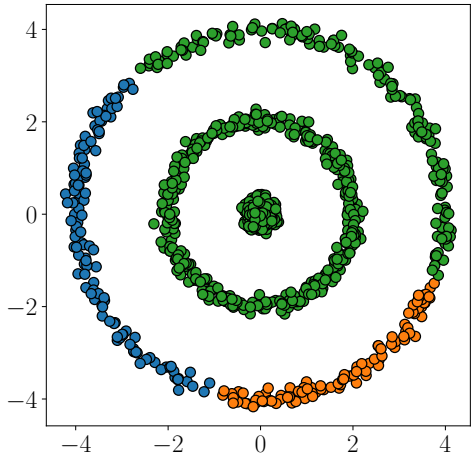
Agglomerative clustering

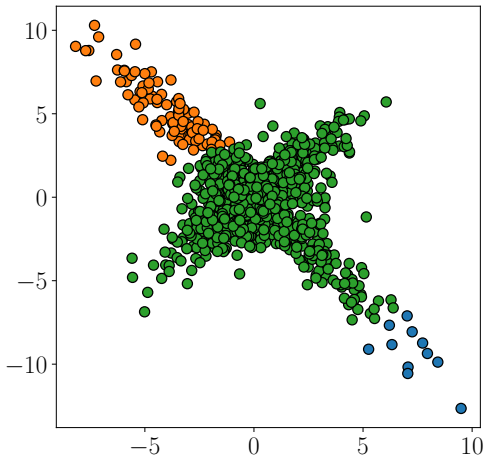
Single linkage:



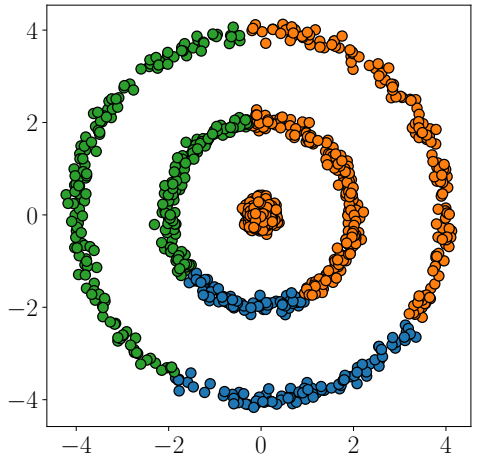


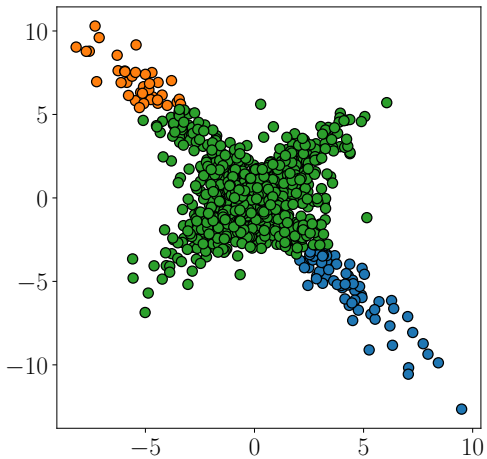
Average linkage:





Complete linkage:





Agglomerative clustering

Can be extended to work with similarity measures (not necessarily associated to distances)

Similarity measures may be derived from classification rules (e.g. classification scores) to guide clustering

For example, a face recognizer can compare two face images and provide a score that represents a measure of whether the pictures belong to the same person or not (high or low similarity)

The scores can be used to cluster a set of unlabeled face images according to the person identity

Density-based clustering

We have seen K-Means and agglomerative clustering

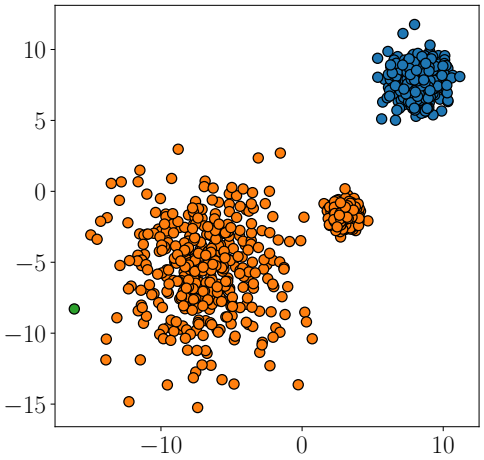
Other approaches based on different frameworks exist

For example, density-based clustering algorithms like DBSCAN group together points that are in high-density regions

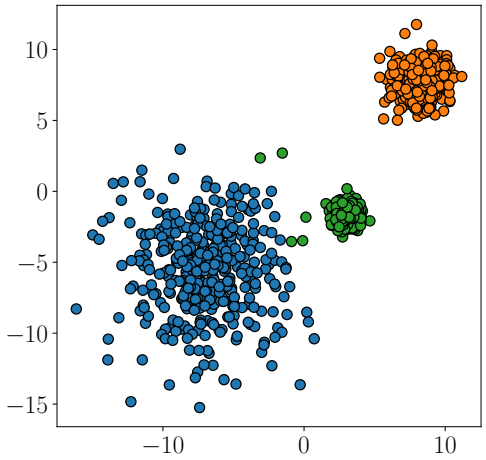
- Density measured in terms of number of close neighbours of a point
- Automatic selection of the number of clusters, but not all samples may be assigned to a cluster
- Issues when clusters have different densities, or with large dimensional data

Agglomerative clustering

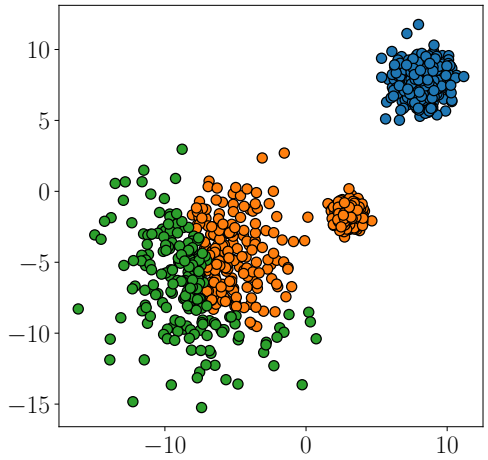
Single linkage

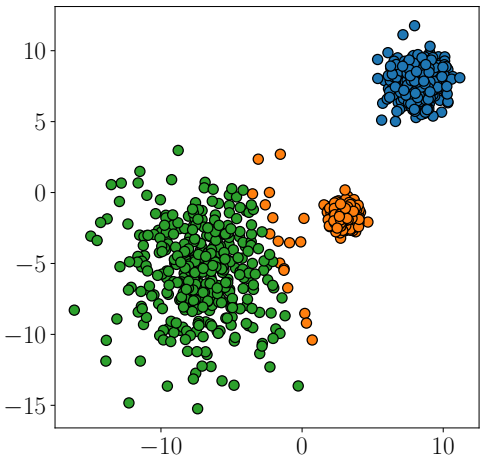


Average linkage

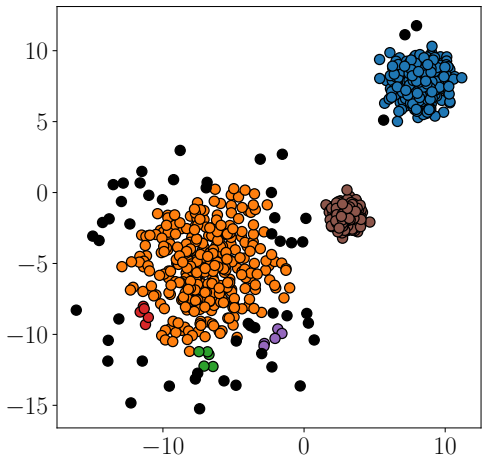


Complete linkage

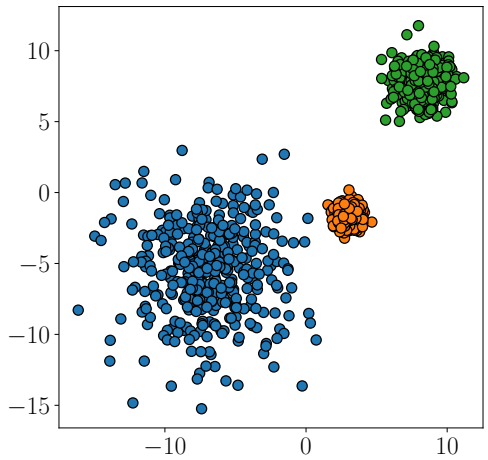




K-means



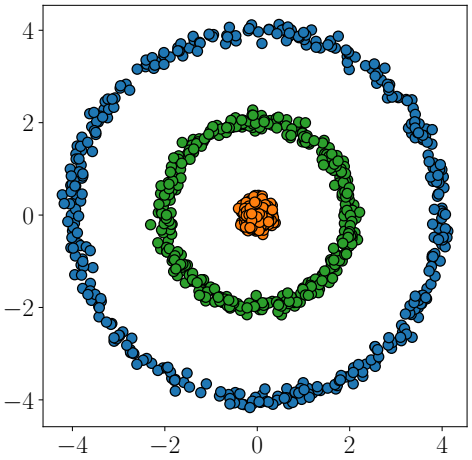
DBSCAN



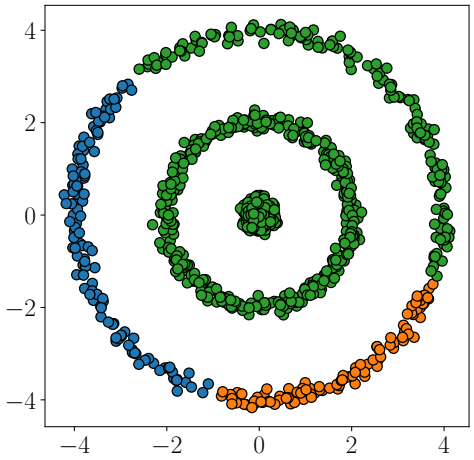
Gaussian mixtures

Agglomerative clustering

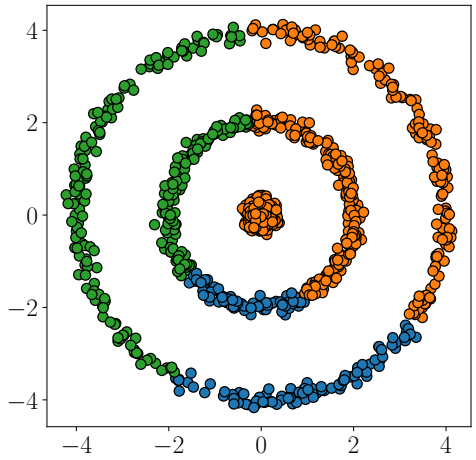
Single linkage

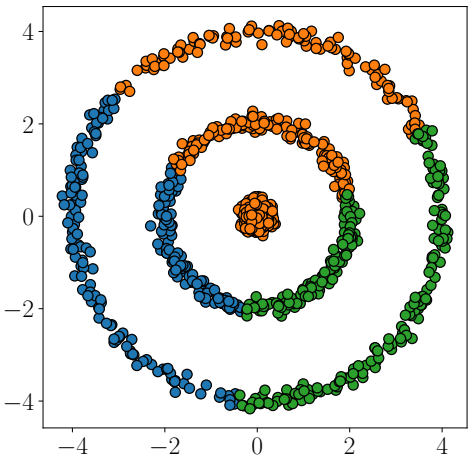


Average linkage

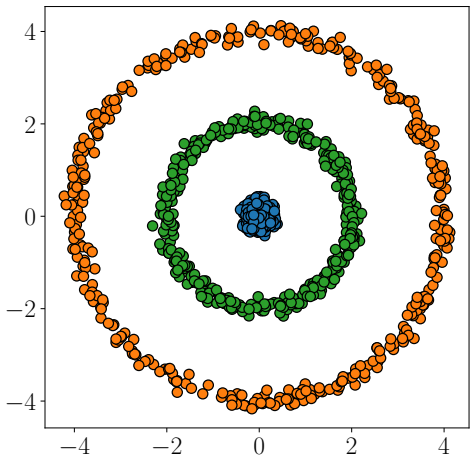


Complete linkage

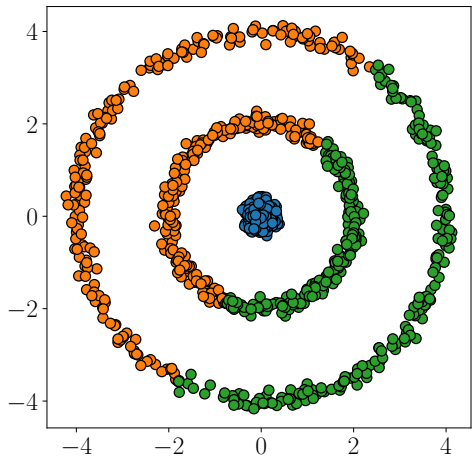




K-means



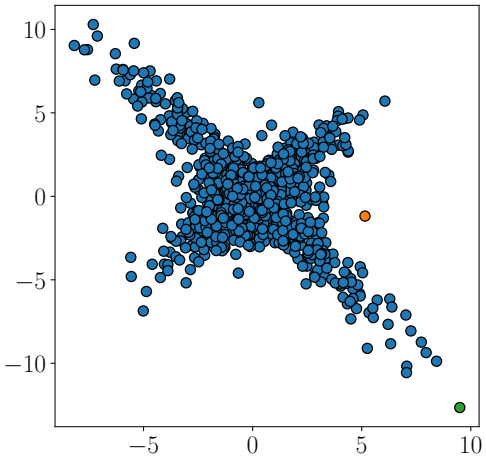
DBSCAN



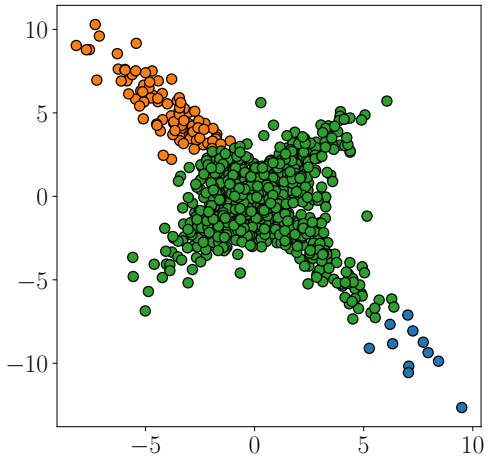
Gaussian mixtures

Agglomerative clustering

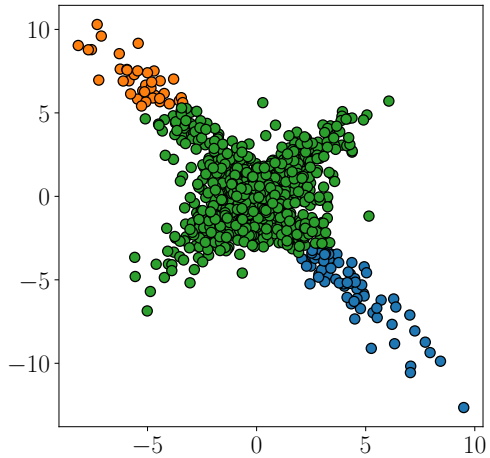
Single linkage

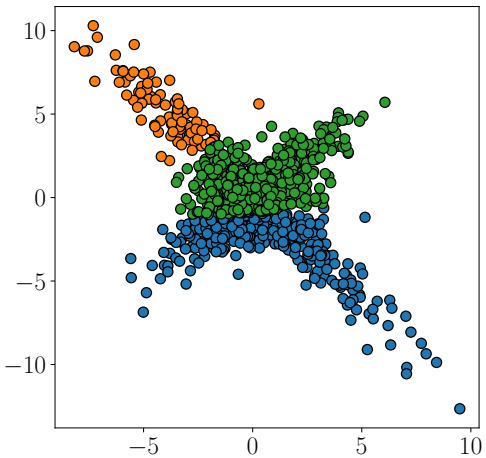


Average linkage

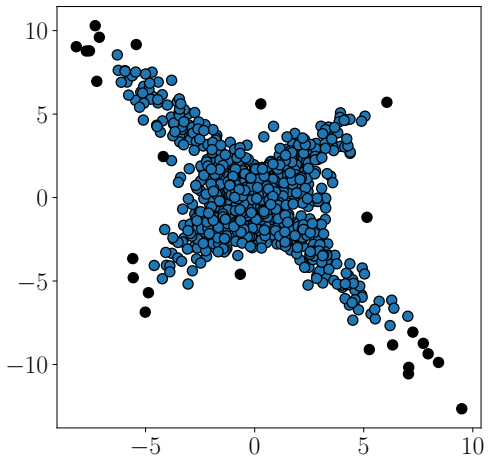


Complete linkage

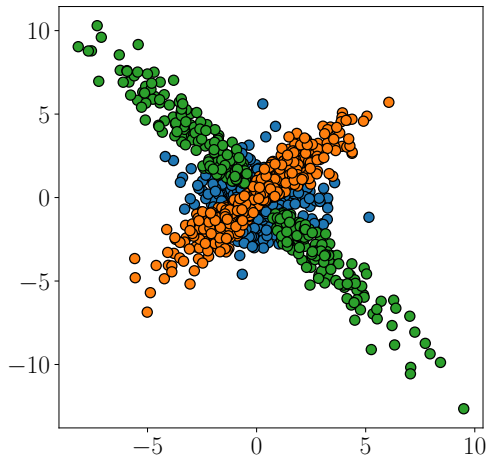




K-means



DBSCAN



Gaussian mixtures

Cluster evaluation

To evaluate the cluster quality we can resort to unsupervised measures such as the silhouette score

If we have the ground-truth cluster labels, we can compare the cluster labels with the ground-truth labels

- Purity / Inverse purity
- Rand Index (RI) / Adjusted Rand Index (ARI)
- Entropy-based measures
- ...

Typically, we evaluate the goodness of the algorithm on a labeled dataset, with the goal of applying the method on unlabeled data

Cluster evaluation

Let

$$C = \{C_1 \dots C_K\}$$

denote the set of clusters

Each cluster also defines a cluster label c_i for sample $\boldsymbol{x}_i$, i.e. $c_i = j \iff \boldsymbol{x}_i \in C_j$

Let

$$L = \{L_1 \dots L_M\}$$

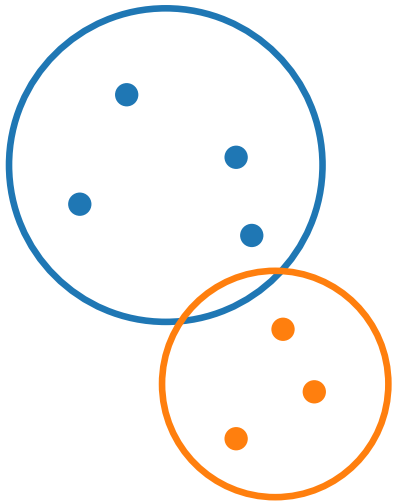
denote the partitioning of samples into ground-truth classes

Again, if l_i is the label of sample $\boldsymbol{x}_i$, then $\boldsymbol{x}_i \in L_j \iff l_i = j$

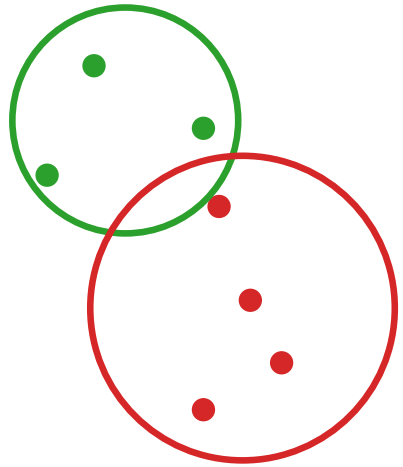
We denote with N the number of samples

Cluster evaluation

Both C (clustering output) and L (labels) are partitions of the data



L (labels)



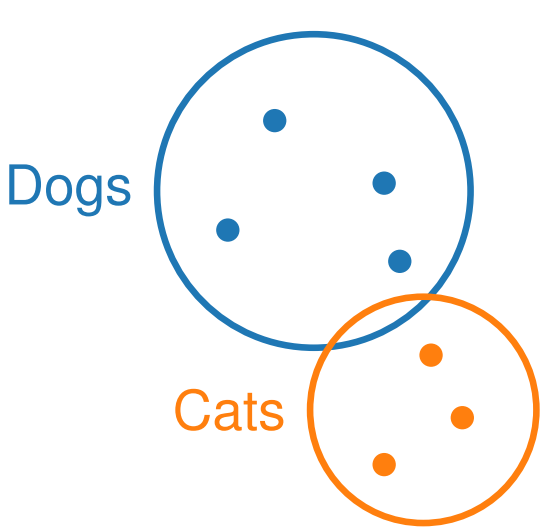
C (cluster output)

Cluster evaluation

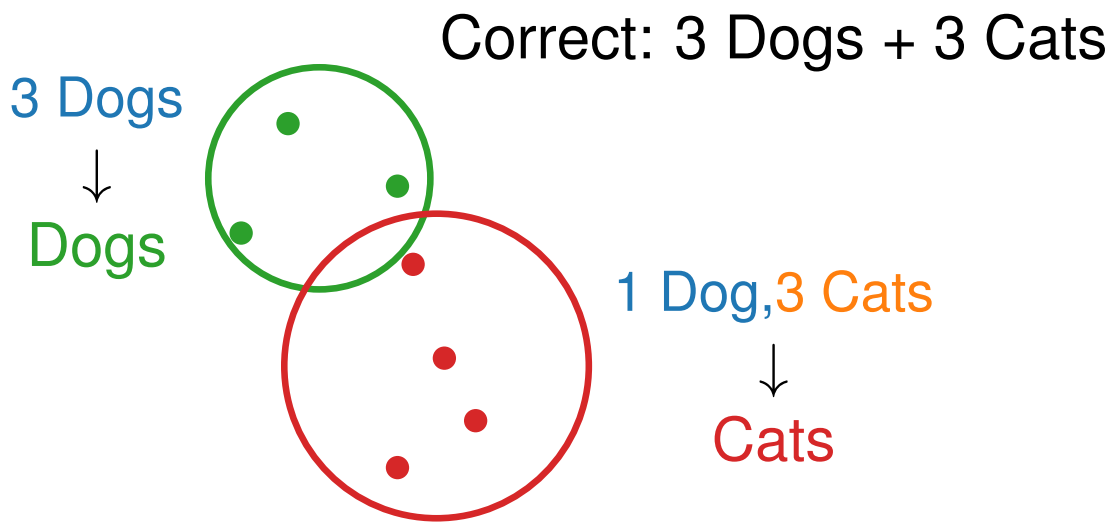
Purity: measure how "pure" is a cluster, i.e. how much the cluster contains elements of a single class

Assign to each cluster the class that is mostly present in the cluster

Compute the fraction of samples that belong to the correct, labeled cluster



L (labels)



C (cluster output)

Cluster evaluation

Purity: measure how "pure" is a cluster, i.e. how much the cluster contains elements of a single class

Assign to each cluster the class that is mostly present in the cluster

Compute the fraction of samples that are correctly labeled

$$P(C, L) = \frac{1}{N} \sum_{c \in C} \max_{l \in L} |c \cap l|$$

$|c \cap l|$ denotes the number of samples that are both in cluster c and in class l

The maximum over $l \in L$ corresponds to labeling the cluster with the label of the class that is most represented

Cluster evaluation

Purity lies in the range $(0, 1]$ (0 cannot be attained since we will always have at least 1 correctly labeled point)

Purity tends to increase with the number of clusters

In agglomerative clustering, purity will increase as we threshold the dendrogram at lower levels (more clusters)

With 1-sample clusters, purity becomes 1 (all clusters contain patterns of a single class)

Cluster evaluation

We need to look at the other side of the problem

Inverse purity (or label purity): measure how much samples of the same class are grouped together

Label each class with the cluster label of the cluster that contains most objects of that class

Compute the fraction of correct cluster labels in the ground-truth partition

It corresponds to purity computed inverting the role of C and L

$$IP(C, L) = P(L, C) = \frac{1}{n} \sum_{l \in L} \max_{c \in C} |c \cap l|$$

The maximum over $c \in C$ corresponds to linking class l with the cluster that contains most of its elements

Inverse purity is 1 when all samples are assigned to a single cluster

Cluster evaluation

We can also define impurity and inverse-impurity

$$I_{clu}(C, L) = 1 - P(C, L)$$

$$I_{lab}(C, L) = 1 - IP(C, L) = 1 - P(L, C)$$

When using hierarchical algorithms we can plot how the impurities change as we change the number of clusters (cutting the dendrogram at different levels)

We can compute the impurities[1] $I_{clu}(k), I_{lab}(k)$ for each possible number of clusters $k \in \{1 \dots n\}$

We can plot the pairs $(I_{clu}(k), I_{lab}(k))$ as a parameteric curve

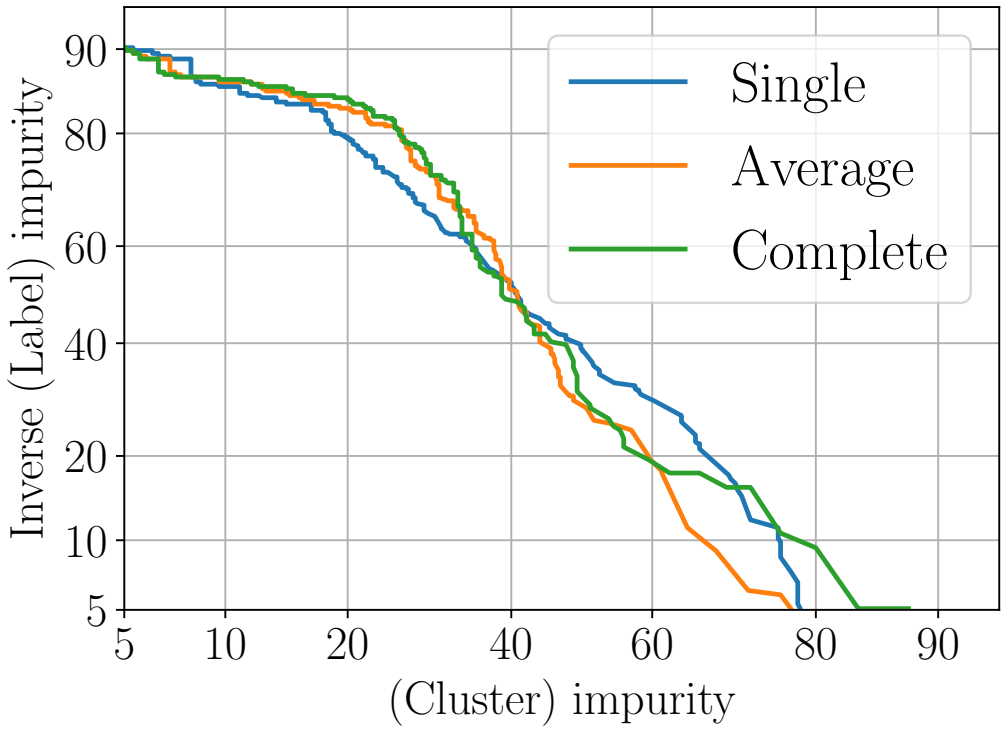
Similar to ROC and DET curves for classification (we will introduce these later)

[1] With an abuse of notation, we identify with $I_{clu}(k)$ the impurity computed from the set of clusters $C(k)$ obtained from the dendrogram cut at a threshold that gives k clusters $I_{clu}(k) = I_{clu}(C(k), L)$. Similarly $I_{lab}(k) = I_{lab}(C(k), L)$.

Cluster evaluation

Changing the number of clusters k allows for a trade-off between the two kinds of error

Can be used to select the best linkage for a given use case — in the following example:



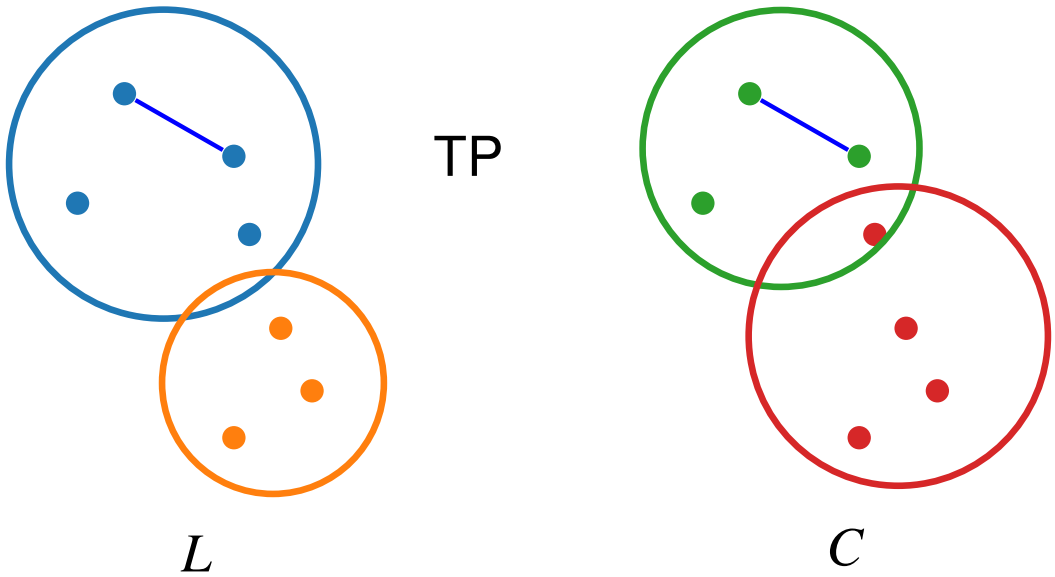
- Single: better if we want lower cluster impurity
- Average: better if we want lower label impurity

Cluster evaluation

Rand index: classification accuracy on pairs for samples

A pair of samples $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is pairwise-correctly clustered if either:

- $c_i = c_j$ and $l_i = l_j$ (i.e., they are in the same cluster and have the same ground-truth label - True Positive (TP))
- $c_i \neq c_j$ and $l_i \neq l_j$ (i.e., they are in different clusters and have different labels - True Negative (TN))

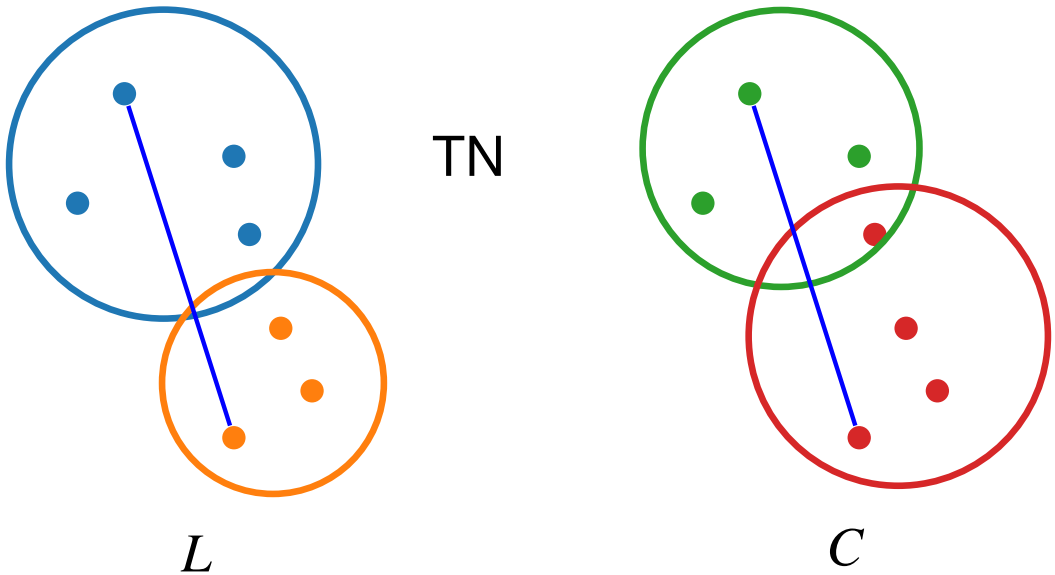


Cluster evaluation

Rand index: classification accuracy on pairs for samples

A pair of samples $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is pairwise-correctly clustered if either:

- $c_i = c_j$ and $l_i = l_j$ (i.e., they are in the same cluster and have the same ground-truth label - True Positive (TP))
- $c_i \neq c_j$ and $l_i \neq l_j$ (i.e., they are in different clusters and have different labels - True Negative (TN))

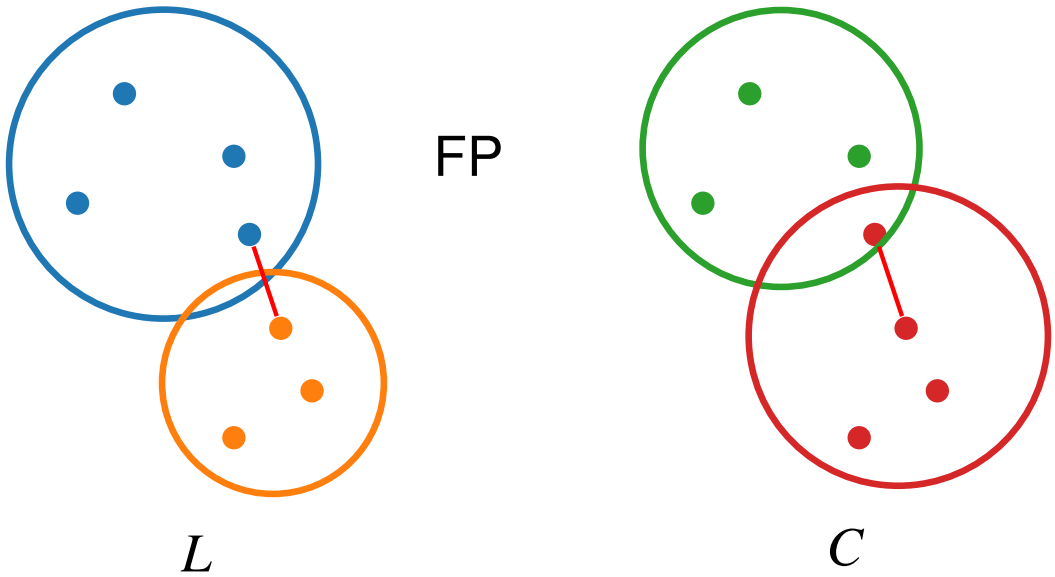


Cluster evaluation

Rand index: classification accuracy on pairs for samples

A pair of samples $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is pairwise-incorrectly clustered if either:

- $c_i = c_j$ and $l_i \neq l_j$ (i.e., they are in the same cluster but have different ground-truth label - False Positive (FP))
- $c_i \neq c_j$ and $l_i = l_j$ (i.e., they are in different clusters but have the same label - False Negative (FN))

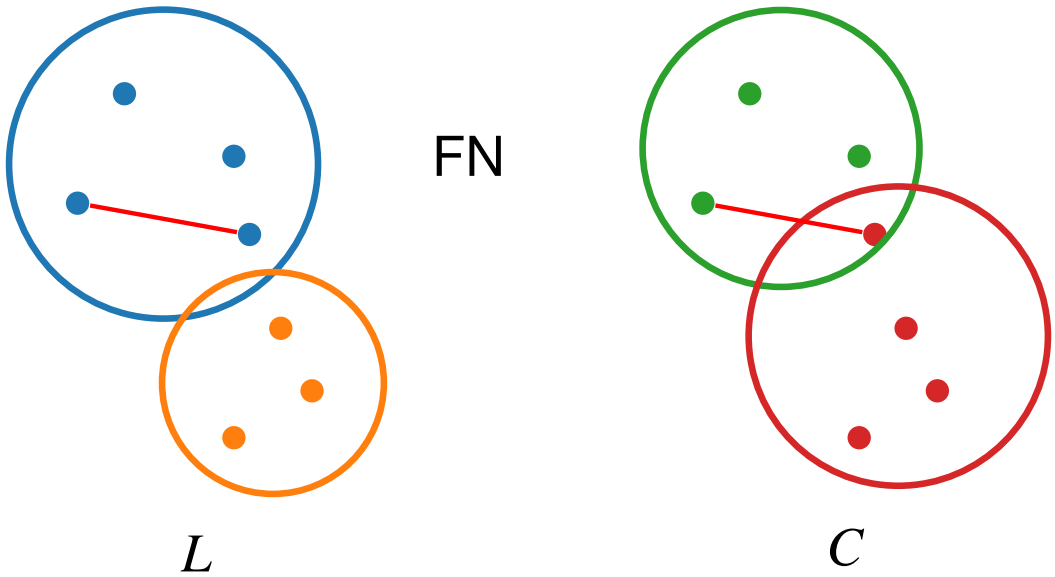


Cluster evaluation

Rand index: classification accuracy on pairs for samples

A pair of samples $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is pairwise-incorrectly clustered if either:

- $c_i = c_j$ and $l_i \neq l_j$ (i.e., they are in the same cluster but have different ground-truth label - False Positive (FP))
- $c_i \neq c_j$ and $l_i = l_j$ (i.e., they are in different clusters but have the same label - False Negative (FN))



Cluster evaluation

Rand index: classification accuracy on pairs for samples

$$RI = \frac{TP + TN}{TP + FN + TN + FP}$$

RI takes values between 0 (complete disagreement) and 1 (complete agreement)

However, if we were to assign random cluster labels, RI would not be, on average, 0

Adjusted Rand Index (ARI): normalize ARI to account for expected RI under a random clustering model

$$ARI = \frac{RI - \mathbb{E}[RI]}{1 - \mathbb{E}[RI]}$$

ARI is close to 0 for random clusters, and to 1 for perfect clustering

Other information-theoretical measures may also employed (mutual information, normalized mutual information)