28 K-Means Clustering: Chasing the Centroids

28.1 What Does "K" Really Mean? — Understanding the Essence of K-Means

28.1.1 From k-NN to K-Means: A Shift from Supervision to Discovery

The letter *K* in K-Means clustering is not the same as the *k* in the *k*-Nearest Neighbors (*k*-NN) algorithm. In *k*-NN (introduced in Chapter 8), we deal with supervised learning, where every sample has a known label, and *k* represents the number of nearest labeled neighbors used for classification.

In contrast, *K*-Means is an unsupervised learning algorithm. The dataset has no labels, and *K* represents the number of clusters we want the algorithm to discover. The goal is to divide a dataset Ω into *K* clusters.

K-Means assumes that each cluster can be represented by a cluster centroid (also called the mean of the cluster).

For example, if we choose $K = 2$, then the algorithm will learn two centroids, $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, each serving as the "center" of a cluster. If we use Euclidean distance as the similarity measure, then each data point is assigned to whichever centroid it is closest to.

28.1.2 How Distance Defines Belonging

In the schematic in Figure 1, point A is clearly closer to centroid $\boldsymbol{\mu}_1$, so it belongs to cluster C_1. Point *C* is closer to $\boldsymbol{\mu}_2$, so it belongs to cluster C_2.

Point *B* is equally distant from both centroids, meaning it lies exactly on the decision boundary.

In this case, the boundary is simply the perpendicular bisector between $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$. Because Euclidean distance creates circular distance contours, the lines of equal distance around each centroid form concentric circles. Wherever two circles of equal distance intersect, the points lie on the decision boundary between clusters. This is why the midpoint line between $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ is the dividing line.

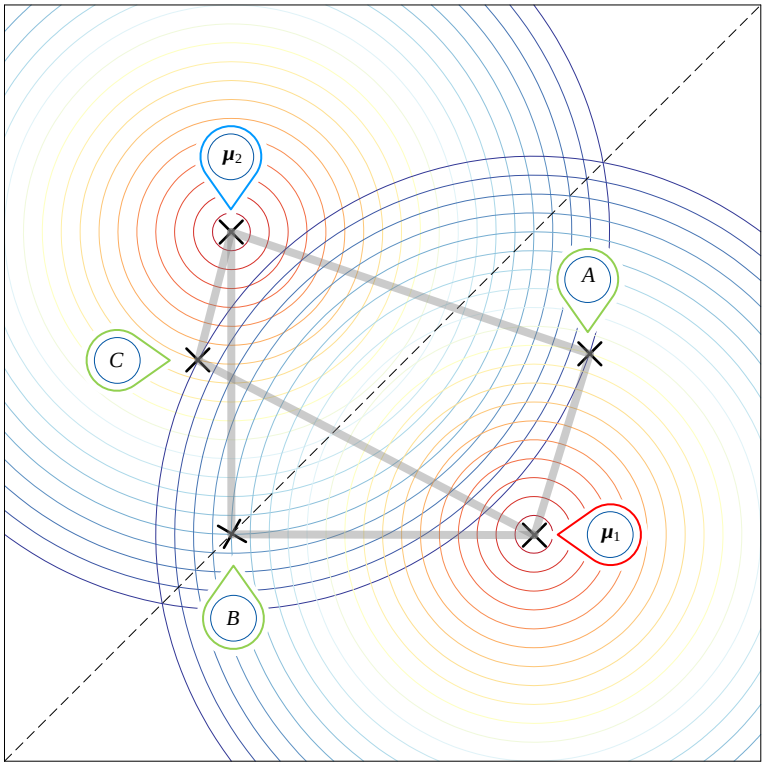

Figure 1. Intuition of *K*-Means: Assigning Points to the Nearest Cluster Centroid

28.2 Behind the Scenes — The Optimization Objective

28.2.1 Minimizing the Sum of Squared Errors (SSE)

The goal of the *K*-Means clustering algorithm is to divide all sample points into K clusters in such a way that points within the same cluster are as close to each other as possible.

Mathematically, *K*-Means minimizes the sum of squared distances between each sample point and the centroid of the cluster it belongs to. When Euclidean distance is used, the optimization objective can be written as:

$$\arg\min_{C} \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} \left\| \boldsymbol{x} - \boldsymbol{\mu}_k \right\|^2 \tag{1}$$

Here, $\boldsymbol{x}$ is a data point (written as a column vector), and $\boldsymbol{\mu}_k$ is the centroid of cluster C_k. In essence, this objective is the same as minimizing the Sum of Squared Errors (SSE). A smaller SSE means the cluster points are more tightly grouped, indicating stronger cohesion inside each cluster.

28.2.2 Voronoi Regions: A Geometric View of Cluster Ownership

When $K = 3$, the space can be divided using three pairwise perpendicular bisectors. As shown in Figure 2, these boundaries form a **Voronoi diagram**, where each region contains all points that are closest to the same centroid.

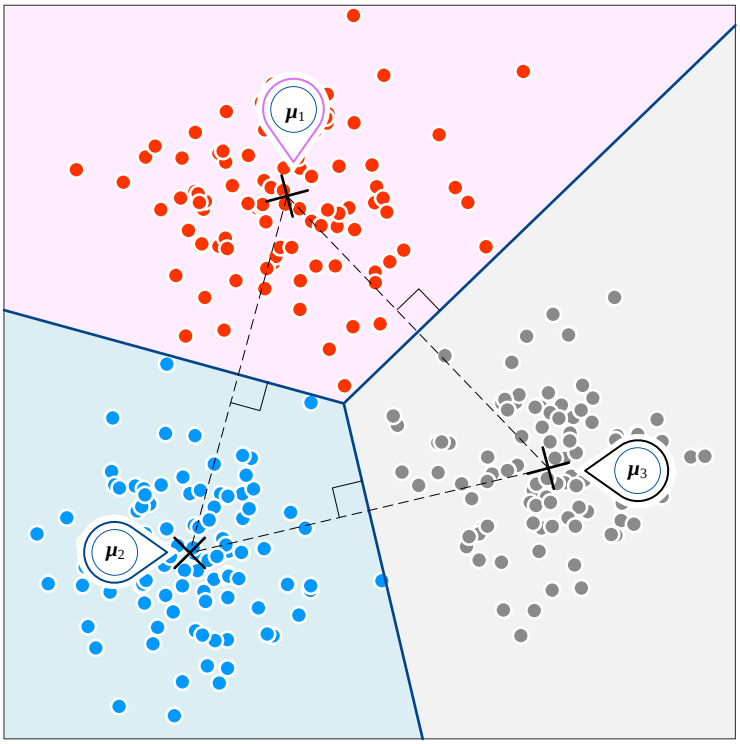

Figure 2. Voronoi Regions Formed by Three Cluster Centroids

If we plot a 3-cluster result in 3D with the *z*-axis representing the predicted cluster label, we can clearly see how the data space is divided into three regions, which is shown in Figure 3. It becomes evident that *K*-Means determines boundaries in a way that is highly similar to the Nearest Centroid Classifier. The key difference is that *K*-Means does not know the centroids in advance—it must learn them through an iterative optimization process, which we will cover in the next section.

When using Euclidean distance, *K*-Means can actually be viewed as a special case of the Gaussian Mixture Model (GMM).

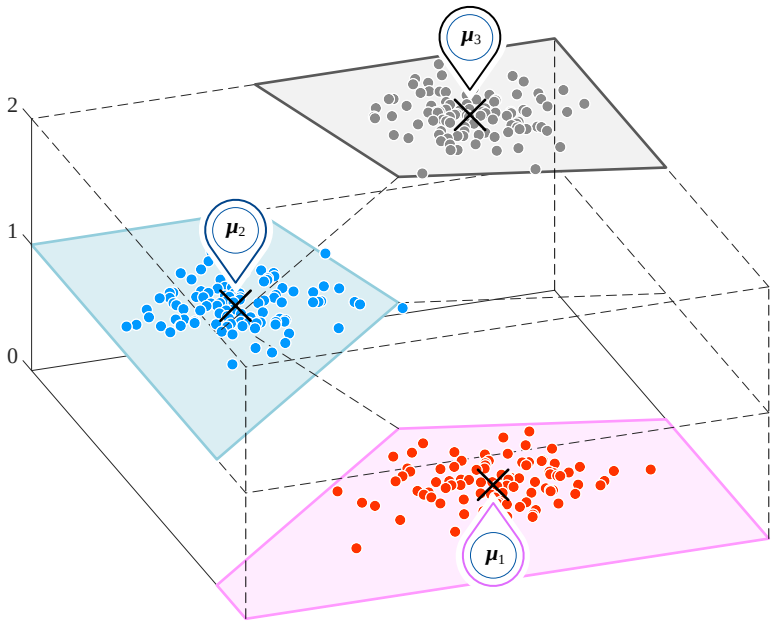

Figure 3. 3D Visualization of Cluster Labels for a Three-Cluster Problem

28.3 Learning by Iteration — How K-Means Finds Its Centers

28.3.1 Step One: Random Beginnings (Initialization)

To understand how *K*-Means actually finds its cluster centers, it is useful to walk through its iterative procedure. In this section, we illustrate the process using the case of $K = 2$, though the same idea applies to any number of clusters.

The algorithm begins with two inputs: the dataset and the chosen number of clusters (for example, $K = 2$). The algorithm then randomly selects two sample points as the initial centroids, denoted by $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$. After the initialization step, *K*-Means repeatedly performs two operations—assigning points to clusters and updating the centroids—until the result stabilizes.

In each iteration, the algorithm first computes the distance from every sample to the current centroids. Each sample is then assigned to whichever centroid it is closest to, which determines the cluster partition for that iteration. Once all samples have been assigned, the centroid of each cluster is recomputed by taking the mean of all points inside that cluster. These updated centroids replace the old ones, and the process repeats. The iterations continue until the centroids stop moving significantly or until a preset limit on the number of iterations is reached. At that point, the algorithm is considered to have converged, and the final cluster assignment is produced.

28.3.2 A Case Study: Watching Centroids Move in the Iris Dataset

Figure 4 shows an example using the iris dataset. Three points (highlighted in yellow) are randomly chosen as the initial centroids $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, and $\boldsymbol{\mu}_3$. Over the course of ten iterations, the centroids gradually shift as the algorithm reassigns points and updates cluster means, eventually settling into stable positions. This movement reflects *K*-Means performing gradient-descent-like optimization on the clustering objective.

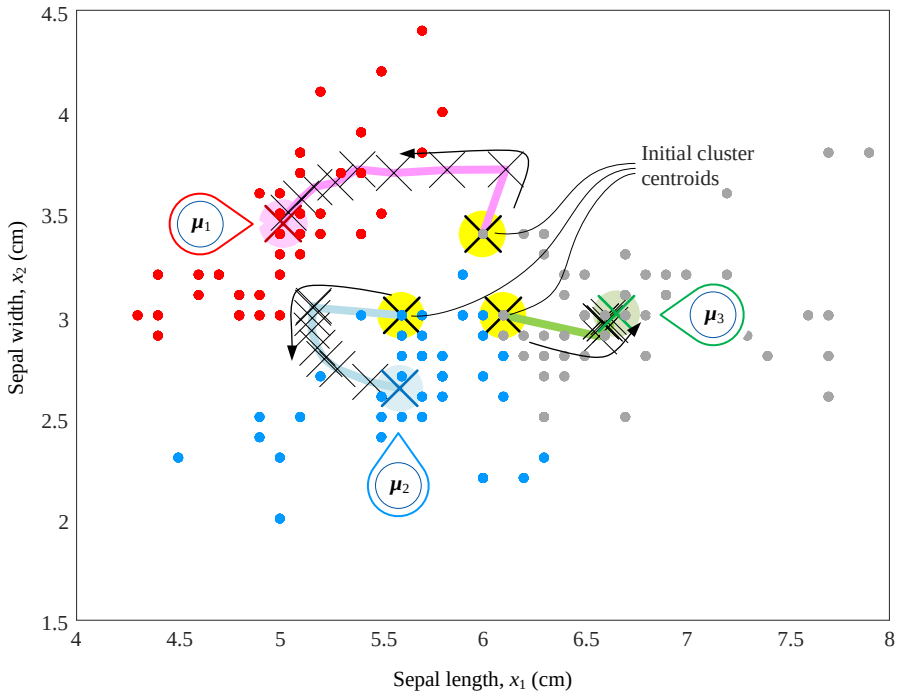

Figure 4. Iterative Movement of Cluster Centroids During *K*-Means Optimization (Iris Dataset)

Figure 5 shows the final clustering result on the iris dataset using the *K*-Means implementation in scikit-learn (sklearn.cluster.KMeans). After fitting the model with model.fit(), the predicted cluster labels can be obtained using model.predict(), and the learned centroids can be accessed through model.cluster_centers_. Even though K-Means has no prior knowledge of species labels, it can still uncover meaningful structure from the unlabeled data.

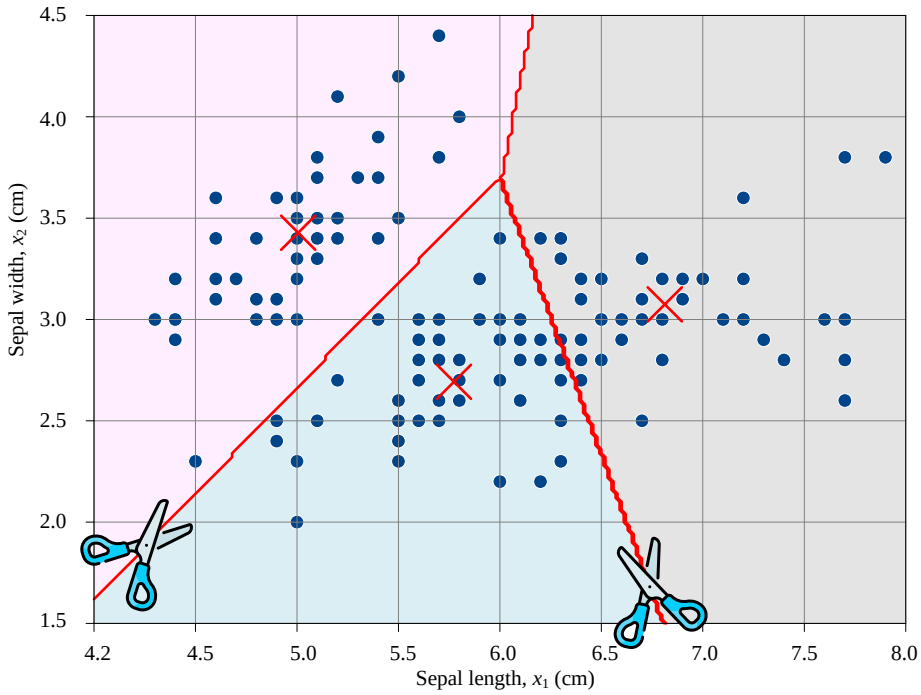

Figure 5. Final *K*-Means Clustering Result on the Iris Dataset. Figure generated by Ch28_01_K-means.ipynb.

28.4 How Many Clusters Are Enough? — The Elbow Method

28.4.1 The Trade-Off Between Fit and Simplicity

In practical applications, one of the main challenges in *K*-Means clustering is deciding how many clusters *K* should be used. A widely adopted heuristic for this purpose is the Elbow Method. The key idea is to examine how the Sum of Squared Errors (SSE) changes as *K* increases.

The SSE, also referred to as inertia, is defined as

$$\text{SSE} = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} \left\| \boldsymbol{x} - \boldsymbol{\mu}_k \right\|^2 \tag{2}$$

This value measures how compact the clusters are—the smaller the SSE, the tighter the points lie around their centroids. As we increase the number of clusters *K*, *K*-Means produces a finer partition of the data.

Consequently, the data points within each cluster tend to be closer to their centroid, and the SSE decreases. In the extreme case, if we set $K = n$ (where every sample becomes its own cluster), the SSE will drop to zero. Although this produces the lowest possible SSE, it is clearly meaningless as a clustering result.

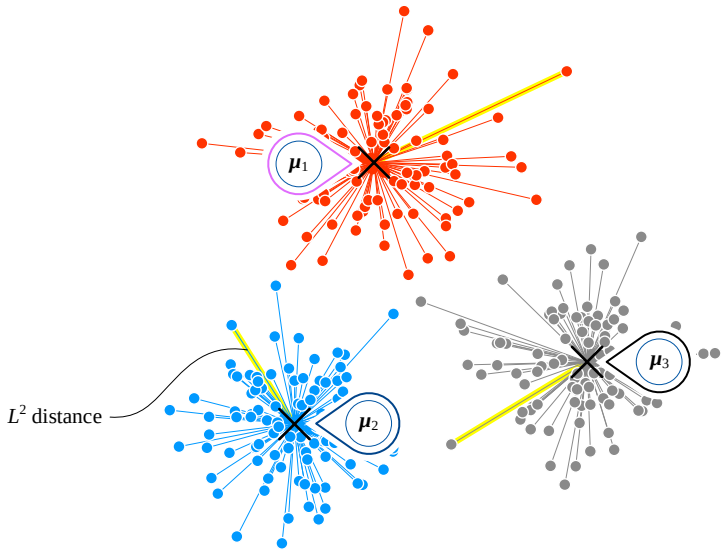

Figure 6. Cluster Centroids and Distance Relationships for Different Values of *K*

28.4.2 Visualizing the Elbow Curve

Figure 7 illustrates how SSE typically changes with different values of *K*. When *K* is too small, each increase in *K* leads to a large drop in SSE, because the model is rapidly improving its ability to fit the data. However, once *K* exceeds the "appropriate" number of clusters, further increases only produce small and diminishing improvements.

The resulting curve resembles a bent arm, forming a visible "elbow" shape. The location of this elbow is taken as the recommended value of *K*, because it balances clustering quality against model simplicity.

In scikit-learn, the current SSE value can be obtained from a fitted *K*-Means model through the attribute model.inertia_.

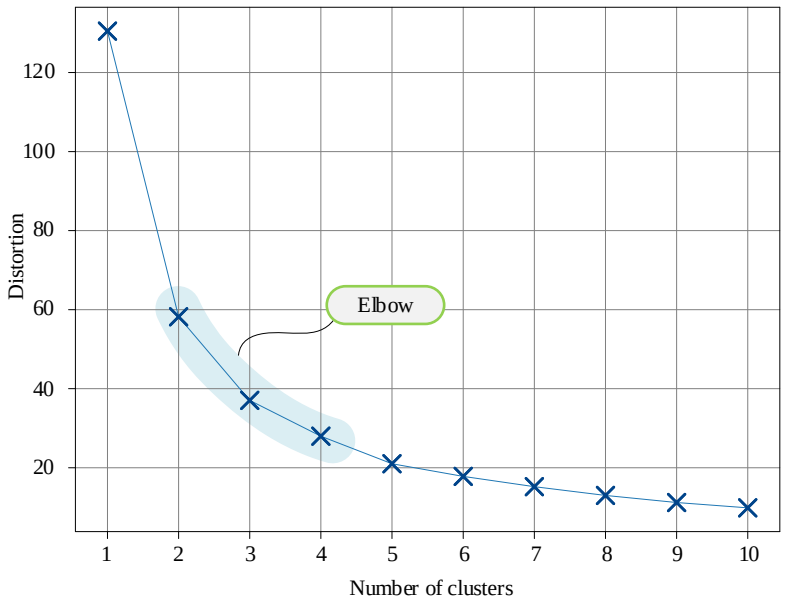

Figure 7. Elbow Curve: SSE versus the Number of Clusters *K*. Ch28_02_elbow_method.ipynb.

28.5 Conclusion

K-Means is an unsupervised clustering algorithm that groups unlabeled data into *K* clusters by assigning each sample to its nearest cluster centroid. Using distance (typically Euclidean distance) as a similarity measure, *K*-Means partitions the space into regions where each region contains all points closest to the same centroid.

Geometrically, this forms Voronoi-style decision boundaries, similar to the Nearest Centroid Classifier, but unlike supervised methods, *K*-Means must learn the centroids directly from the data. The algorithm works through iterative optimization. It begins by choosing *K* initial centroids, then repeats two steps: assigning each point to its closest centroid and recomputing each centroid as the mean of its assigned points. This process continues until the centroids stabilize, minimizing the Sum of Squared Errors (SSE) within clusters.

When visualized, the movement of centroids shows how the algorithm gradually improves cluster cohesion. Because the number of clusters is not known in advance, the Elbow Method is often used to select *K*. By plotting SSE against different values of *K*, the curve forms an "elbow," whose bend indicates a good balance between simplicity and clustering quality. *K*-Means is simple, intuitive, and effective for many real-world datasets.