Centroid Clustering

2026-05-06 · cheerful mango Haubentaucher

Same Same but Different.

The Why

In the previous chapter, we established that distance is a choice that is to be engineered[1]: it encodes what similar means in a given feature space. But similarity is a local, pairwise concept so far.

As a human looking at data, we immediately see clusters C of points that belong together, and we have an intuition for which new points would fit into which cluster. The process of finding them is called clustering.

[1]

A partition or a clustering assigns every point to exactly one cluster, with no cluster left empty. In this lecture we will learn about K-Means[2], which produces a *hard* partition: membership is binary, not graded.

[2]

Binary membership is a strong assumption that later clustering methods will soften.

Given that means to encode similarity over whole sets of samples, we can ask a more complex question, one that goes beyond pairs of points. Being able to answer which point belongs to which cluster is a predictive power that allows us to generalize from observed data to new instances.

Figure 1: Clustering allows us to move from a) and b) are similar to a) and b) are of the same. *Image is generated with NanoBanana.*

In order to move from representation to leveraging pattern we have good reason to find ways to,

- compute partitions, of the data, given a distance measure.
- formalize what makes a well-defined partition by specifying an appropriate objective.
- recognize the geometric assumptions and finding ways to represent cluster characteristics.

Hands On Experience

CONSIDER twelve data points in two dimensions. Which points belong together? How many clusters do you see?

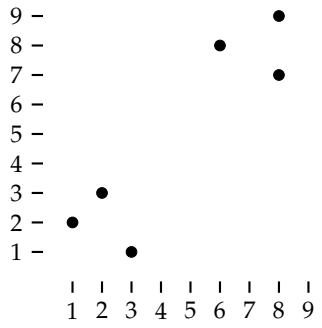


Figure 2: Six points in 2D. How many clusters do you see?

HOW MANY CLUSTERS do you see? Two seems natural, but could you argue for one? For six?

WITHOUT GROUND TRUTH, there is no single right answer, as a context for any answer could be engineered. However, some answers are better than others, and it does not come as a surprise when I tell you that most people see two clusters in this data.

THE FUNDAMENTAL QUESTION this chapter answers is how do we find the clusters mathematically? We will see how K-Means assigns each point to a cluster; and how cluster centroids function as representative points for each cluster.

WHERE WOULD YOU PLACE a representative point for each cluster? Could you think of other measures to characterize and represent clusters, beyond a single point?

A CENTROID c is the mean position of all points assigned to a cluster. It need not be an actual data point; it is the center of mass of the cluster. The centroid exists in the same space as the data. With a different distance, a different notion of center is needed. For sake of simplicity and interpretability, we tie ourselves to Euclidean distance and arithmetic means. But it is worth getting an intuition for other distances as well.

THE LEARNING OBJECTIVES of this lecture:

- Understand the K-Means objective and algorithm, including convergence.
- Apply initialization strategies and K-selection methods.
- Identify when K-Means fails and understand why.

The K-Means Algorithm

A DATA POINT is a vector $x_i \in \mathbb{R}^d$, where d is the number of features and $i = 1, \ldots, n$ indexes the observations.

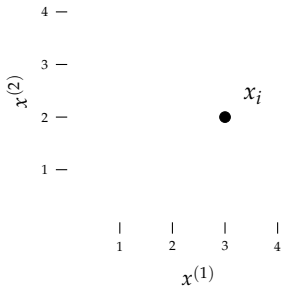


Figure 3: A single data point x_i in $\mathbb{R}^2$.

A CLUSTER is a non-empty subset $C_k \subseteq \{x_1, \ldots, x_n\}$. Each point belongs to exactly one cluster; this is the hard assignment assumption, meaning clusters are disjoint:

$$C_j \cap C_k = \emptyset \quad \text{for } j \neq k$$

A clustering of the data is a collection of K clusters $\{C_1, \ldots, C_K\}$ that is exhaustive, every point is accounted for:

$$\bigcup_{k=1}^{K} C_k = \{x_1, \ldots, x_n\}$$

K-MEANS finds a clustering of n data points into K clusters by minimizing the total squared distance from each point to its cluster's centroid. This quantity is called inertia or within-cluster sum of squares (WCSS):

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \tag{1}$$

where the mean $\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$ is the centroid of cluster C_k.

MINIMIZING J is NP-hard (effort exponential in n and K) in general. K-Means finds a local optimum, not necessarily the global one. The result depends on initialization, and is thus non-deterministic. Shout out numerical methods.

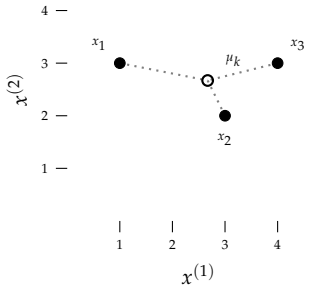


Figure 4: Cluster $C_k = \{x_1, x_2, x_3\}$ with centroid μ_k. Dotted lines show the distances whose squares sum to J.

THE ALGORITHM ALTERNATES BETWEEN TWO STEPS until the assignments no longer change. It remains one of the most cited algorithms in machine learning, in part because it is easy to implement, easy to interpret, and fast enough to run on large datasets.

Algorithm 1: K-Means

STANDARD ALGORITHMIC FORM was described by Lloyd

Require: Data $X = \{x_1, \ldots, x_n\}$, number of clusters K, initial centroids $\mu_1, \ldots, \mu_K$
Ensure: Clustering $\{C_1, \ldots, C_K\}$, centroids $\{\mu_1, \ldots, \mu_K\}$

1: **repeat**
2: **Assignment:** $c_i \leftarrow \arg\min_k \|x_i - \mu_k\|^2$ for each x_i
3: **Update:** $\mu_k \leftarrow \frac{1}{|C_k|} \sum_{x_i : c_i = k} x_i$ for each k
4: **until** assignments no longer change

A TRACE THROUGH THE ALGORITHM on three points with K=2 makes this concrete. With initial centroids at μ_1=x_1=$(1,3)$ and μ_2=x_3=$(4,3)$:

- For each point, the assignment step asks: which centroid is closest?
- For each cluster, the update step asks: given the current assignments, where is the new centroid?

STEP 1: INITIALIZATION. We place two centroids at $\mu_1 = (1,4)$ and $\mu_2 = (4,1)$, away from any data point. At this moment the positions of the centroids is arbitrary, and the mean variable is not meaningful.

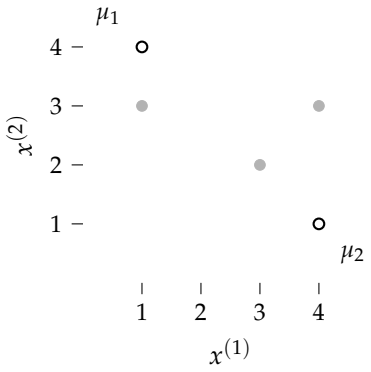


Figure 5: Initialization: two centroids placed away from the data. All points are unassigned.

STEP 2: ASSIGNMENT. Each point joins the cluster of its nearest centroid. For that, we calculate squared distances.

$$
\begin{aligned}
d^2(x_1,\mu_1) &= 0+1 = 1\\
d^2(x_1,\mu_2) &= 9+4\\
&= 13 \;\rightarrow\; C_1\\
d^2(x_2,\mu_1) &= 4+4 = 8\\
d^2(x_2,\mu_2) &= 1+1\\
&= 2 \;\rightarrow\; C_2\\
d^2(x_3,\mu_1) &= 9+1 = 10\\
d^2(x_3,\mu_2) &= 0+4\\
&= 4 \;\rightarrow\; C_2
\end{aligned}
$$

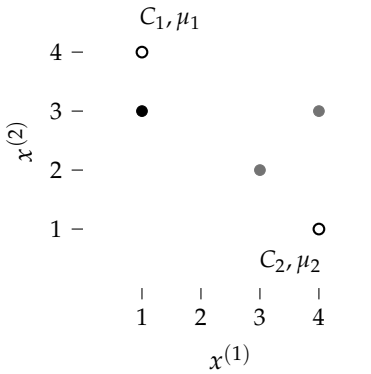


Figure 6: Assignment: $x_1 \rightarrow C_1$ (dark), x_2 and $x_3 \rightarrow C_2$ (grey).

STEP 3: UPDATE. Each centroid moves to the mean of its cluster.

$$
\begin{aligned}
\mu_1' &= x_1\\
&= (1,\ 3)\\
\mu_2' &= \tfrac{1}{2}\big((3,2)+(4,3)\big)\\
&= (3.5,\ 2.5)
\end{aligned}
$$

ON ′ NOTATION. The prime symbol (′) usually denotes updates in iterative algorithms.

CONVERGENCE is guaranteed in finite steps. The argument rests on three observations: first, each assignment step can only decrease

LOCAL OPTIMUM. Convergence does not mean the algorithm found the best clustering. It found a clustering. A different initialization may yield a better one.

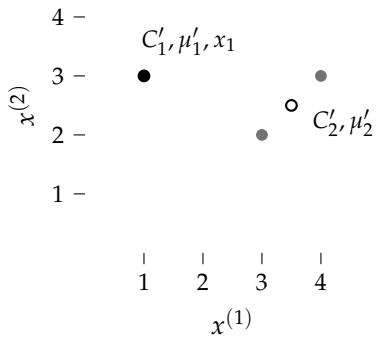


Figure 7: Update: both centroids move. μ_1 coincides with $x_1 = (1, 3)$; μ_2 jumps from $(4, 1)$ to $(3.5, 2.5)$. Re-running assignment yields the same clusters: converged at J=1.

J, since every point moves to an equal or closer centroid; second, each update step can only decrease J, since the mean minimizes squared distance to its members; third, there are at most K^n possible assignment configurations. Since J is non-increasing and the number of states is finite, the algorithm must reach a fixed assignment and stop.

Initialization Matters

K-Means is sensitive to where centroids start. The algorithm is thus non-deterministic. This is because the objective is non-convex: it has many local minima, and the algorithm may get stuck in one of them. If two initial centroids fall inside the same natural cluster, this cluster might get split. While at the same time, a different cluster might be left without a centroid, getting merged with a nearby cluster instead, which results in a high-inertia J clustering.

Educated InitializationK-Means++ for instance features educated initialization. Centroids seeded onto data points x_i. The first centroid is chosen uniformly at random. Each subsequent centroid is chosen with probability proportional to its squared distance to the nearest already-chosen centroid: $p(x_i) = D(x_i)^2 / \sum_j D(x_j)^2$, where $D(x_i) = \min_{j<k} \|x_i - c_j\|$. Points far from all current centroids are more likely to be chosen next. This gives a theoretical guarantee on the expected inertia relative to the global optimum.

Running the same three points with μ_1=$(2, 2)$ and μ_2=$(4, 4)$ shows how a different start leads to a different, worse result.

Multiple restarts. Running K-Means n times with different random initializations, keep the result with the lowest J. Typical default: n=10, but checking the variance or standard deviation of the resulting Inertias and making an informed decision is recommended.

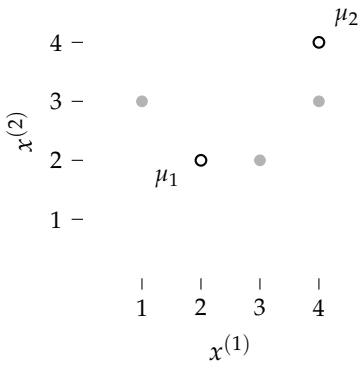


Figure 8: Initialization: centroids at μ_1=(2, 2) and μ_2=(4, 4). All points unassigned.

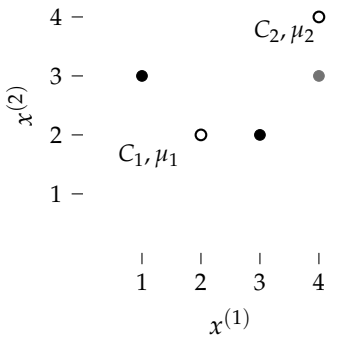


Figure 9: Assignment: x_1 and $x_2 \rightarrow C_1$ (dark), $x_3 \rightarrow C_2$ (grey).

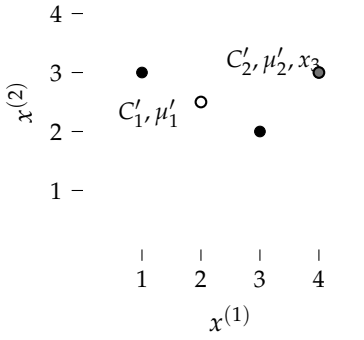


Figure 10: Update: μ_1 moves to (2, 2.5), μ_2 drops to (4, 3). Re-running assignment yields the same clusters: converged at J=2.5, worse than the earlier trace (J=1).

Choosing K

In 2D the number of clusters is often obvious, but in higher dimensions and with more data, or when facing fuzzy clusters or cluster borders, it is not. While the choice for K is often guided by domain knowledge, it is not uncommon to have no such guidance. Then we have to rely on heuristics.

The elbow method plots inertia J against K and looks for a kink: the point where adding one more cluster yields diminishing returns. For our three points, J drops from K=1 to K=2, then reaches zero at K=3.

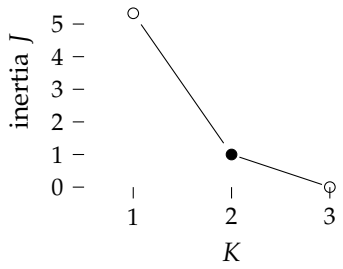


Figure 11: Inertia J as a function of K for the three trace points. The kink at K=2 (filled dot) matches the two natural clusters. K=3 trivially gives J=0 - jap, that is overfitting.

The elbow is not always clear. On real data, the curve may decrease smoothly with no obvious kink. The elbow is a heuristic, not a guarantee.

The silhouette score builds on two distances defined for each point x_i. This allows us to evaluate the clustering on a per-point scale. Think about how interpretable and beautiful that would look when colour-coded.

- **Intra-cluster distance** $a(i)$: mean distance from x_i to all other points in its own cluster. Small $a(i)$ means the point fits tightly with its neighbours.
- **Inter-cluster distance** $b(i)$: mean distance from x_i to all points in the nearest other cluster. Large $b(i)$ means the point is far from the next-best alternative.

A well-placed point has small $a(i)$ and large $b(i)$. The silhouette score combines both into a single value per point:

$$s(i) = \frac{b(i) - a(i)}{\max\big(a(i),\, b(i)\big)}$$

$s(i)$ ranges from -1; inter $<$ intra: point fits better in a neighbouring cluster. To $+1$; intra $\ll$ inter: point is tightly placed and well-separated The mean silhouette over all points is a quality score for the full clustering. Choose K that maximizes it.

Simplifications in per-cluster metrics. In practice, $a(i)$ is often computed as the distance to the cluster centroid, rather than the mean distance to all other points. And b(i) is often computed as the distance to the nearest other centroid, rather than the mean distance to all points in the nearest cluster. And even more simplifications are possible, when only calculating the silhouette for the centroids themselves.

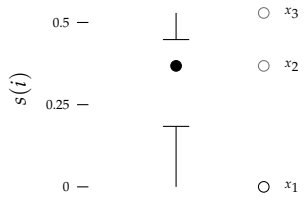
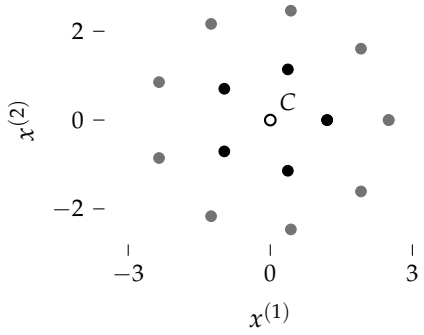


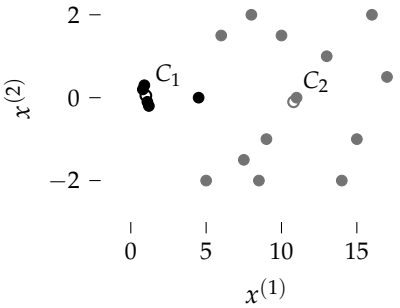
Figure 12: Silhouette scores for K=2. Box plot (left): median ≈0.37, mean ≈0.30, IQR [0.18, 0.45]. Individual scores per point (right, coloured by cluster).

When K-Means Fails

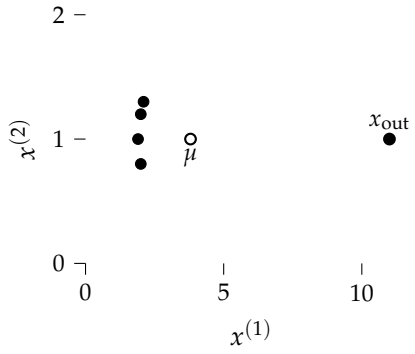
K-Means assumes clusters are roughly spherical, equally sized, and equally dense. Remember that distances measures are a choice.



Non-Convex Shapes
Figure 13: The natural clusters are concentric rings. Both share the same centroid at the origin, so K-Means cannot separate them; it will split the data along a straight boundary through the centre instead.



Cluster Density and Gravity
Figure 14: C_1 has four tightly packed points near $x = 1$; C_2 has thirteen points spread from $x = 5$ to $x = 17$, pulling its centroid to $x \approx 10.8$. The decision boundary falls near $x \approx 6$, so the point at $x = 4.5$ is assigned to C_1 even though it sits at C_2's left edge—a direct consequence of K-Means optimising inertia rather than visual proximity.



Outliers
Figure 15: Sensitivity to outliers: four tight points form a natural cluster near $(2, 1)$, but a single outlier ($\times$) pulls the centroid μ to $(3.8, 1)$, well outside the dense group. K-Means uses the mean, which is not robust to extreme points.

When K-Means struggles, the underlying issue is always the same: the Euclidean distance to a centroid is the wrong measure of belonging for that data. Can you think of another failure mode? Let me know.

K-Medoids uses actual data points as cluster centers instead of means, making it robust to outliers and compatible with non-Euclidean distances.

Mini-Batch K-Means updates centroids on random subsets of the data only, allowing us to scale to millions of points at the cost of a slightly noisier result.

Examples & Exercises

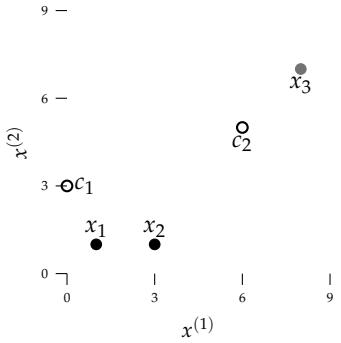


Figure 16: Three coffee samples x_1, x_2, x_3 with initial centroids $\mu_1{=}(0,3)$ and $\mu_2{=}(6,5)$ (circles).

COMPUTING BY HAND builds the intuition that no amount of function calls can replace. Step through the arithmetic slowly. Implementing from scratch comes right after, but let's anchor the mechanics first.

GIVEN THREE POINTS $x_1{=}(1,1)$, $x_2{=}(3,1)$, $x_3{=}(8,7)$, run one full K-Means iteration with $K{=}2$. The initial centroids are $c_1{=}(0,3)$ and $c_2{=}(6,5)$, neither coinciding with a data point. Your tasks: perform the assignment step, the update step, verify convergence, and evaluate the quality of the clustering by computing the inertia.

ASSIGNMENT STEP. For each point compute $d^2(x, c_k) = \sum_j (x_j - c_{kj})^2$ and assign to the nearest centroid. Here is the full trace for point $x_1{=}(1,1)$:

$$\begin{aligned} d^2(x_1,\, c_1) &= (1-0)^2 + (1-3)^2 \\ &= 1 + 4 \\ &= 5 \end{aligned}$$

$$\begin{aligned} d^2(x_1,\, c_2) &= (1-6)^2 + (1-5)^2 \\ &= 25 + 16 \\ &= 41 \end{aligned}$$

Since $5 < 41$, point x_1 is assigned to C_1. Now compute the distances for x_2 and x_3 yourself, then verify against the table below.

Point	$d^2(\cdot, \mu_1)$	$d^2(\cdot, \mu_2)$	**Cluster**
x_1 $(1,1)$	**5**	41	C_1
x_2 $(3,1)$	**13**	25	C_1
x_3 $(8,7)$	80	**8**	C_2

Table 1: Assignment step with $c_1{=}(0,3)$, $c_2{=}(6,5)$. Bold values indicate the minimum in each row; each point is assigned to the corresponding cluster.

UPDATE STEP. Recompute each centroid as the mean of its assigned points. Here is the full calculation for μ_1':

$$\begin{aligned}\mu_1' &= \frac{1}{|C_1|}\sum_{x\in C_1} x \\ &= \tfrac{1}{2}\big((1,1)+(3,1)\big) \\ &= \tfrac{1}{2}(4,2) \\ &= (2,1)\end{aligned}$$

Now derive μ_2' yourself (C_2 contains only a single point) and then check:

$$\begin{aligned}\mu_2' &= x_3 \\ &= (8,7)\end{aligned}$$

CONVERGENCE CHECK. Run the assignment step once more with μ_1'=(2,1) and μ_2'=(8,7): you should recover the same two clusters C_1=$\{x_1, x_2\}$, C_2=$\{x_3\}$. The algorithm has converged after a single iteration.

The inertia at convergence is:

$$\begin{aligned}J &= \|x_1-\mu_1'\|^2 + \|x_2-\mu_1'\|^2 + \|x_3-\mu_2'\|^2 \\ &= \big((1{-}2)^2+(1{-}1)^2\big) + \big((3{-}2)^2+(1{-}1)^2\big) + 0 \\ &= 1+1+0 \\ &= 2\end{aligned}$$

Calculate the inertia for $K = 1$ yourself. Assume that the single centroid is at the mean of all three points, and is thus quick to compute. What is the Inertia for $K = 3$, is there a need to compute it, and why?

CHOOSING K WITH THE ELBOW METHOD. The table in the margin gives the optimal inertia for $K = 1, 2, 3$ on the same three points. Sketch the inertia curve and identify the elbow: the value of K at which adding another cluster stops yielding a large reduction. Which K do you select, and why?

Note that K=3 always reaches J=0 by placing one centroid per point; this tells us nothing about structure in the data.

Inertia values:

K	J
1	50
2	2
3	0

COMPUTE THE SILHOUETTE SCORES for the three trace points $x_1{=}(1,3)$, $x_2{=}(3,2)$, $x_3{=}(4,3)$ with $K{=}2$, where $C_1{=}\{x_1\}$ and $C_2{=}\{x_2, x_3\}$. Recall that $a(i)$ is the mean intra-cluster distance and $b(i)$ the mean nearest-cluster distance for point x_i.

For x_1 (singleton cluster C_1): a single point has no intra-cluster structure to evaluate, so by convention:

$$s(x_1) = 0$$

For a singleton, $a(x_i)$ is the mean distance to zero other points, i.e. $0/0$. Setting $a{=}0$ by convention makes the formula return $s{=}1$, as if the point were perfectly clustered. The override $s{=}0$ corrects this: without at least two points in a cluster, the silhouette score is undefined and should not inflate the mean. $s{=}0$ is the neutral midpoint of $[-1, 1]$ and carries no bias.

For x_2 and x_3 (both in C_2, with $d(x_2, x_3) = \sqrt{2} \approx 1.41$):

$$\begin{aligned} a(x_2, C_2) &= d(x_2, x_3) \approx 1.41 \\ b(x_2, C_1) &= d(x_2, x_1) \\ &= \sqrt{5} \\ &\approx 2.24 \\ s(x_2) &= \frac{2.24 - 1.41}{2.24} \approx 0.37 \end{aligned}$$

$$\begin{aligned} a(x_3, C_2) &= d(x_3, x_2) \approx 1.41 \\ b(x_3, C_1) &= d(x_3, x_1) \\ &= \sqrt{9} \\ &= 3 \\ s(x_3) &= \frac{3 - 1.41}{3} \approx 0.53 \end{aligned}$$

The mean silhouette score is $\bar{s} = (0 + 0.37 + 0.53)/3 \approx 0.30$, matching the box plot in the theory section. Note that x_1 drags the mean down: singleton clusters always score $s{=}0$. A score well below 1 indicates that points are not much closer to their own cluster than to the nearest rivaling cluster.

A COFFEE AROMA DATASET TO EXPLORE. Load the dataset of 1,797 two-dimensional aroma fingerprints from ten brewing methods and scatter-plot the two aroma coordinates, coloring each sample by its true variety. Implement the assignment step, the update step, and the inertia function from scratch, then combine them into a complete K-Means run. Run K-Means with $K{=}10$ twenty times using different random seeds and compare the resulting inertia values.

CODE will be provided as a Python notebook. Use it as a starting point, break things, and observe what changes.

Then vary K from 1 to 15, record the best inertia across three restarts for each, and plot the elbow curve.

RANDOM SEEDS. Random seeds are integers that initialize the pseudo-random number generator, ensuring reproducibility.

WHAT TO OBSERVE. Inertia varies noticeably across the twenty runs at $K{=}10$: some seeds place two starting centroids inside the same brewing group and the algorithm never recovers, producing a measurably worse partition. The elbow curve bends near $K{=}10$, though

not sharply, because several brewing varieties overlap in the two-dimensional fingerprint space. Adding clusters beyond the elbow yields diminishing inertia reduction, confirming that the algorithm is splitting existing groups rather than discovering new ones.

Self-Reflection and Recap

SELF-REFLECTION questions to guide your thinking:

- What does the K-Means objective actually minimize, and why is that a reasonable definition of a good clustering?
- Why does K-Means always converge, and why does convergence not guarantee the best solution?
- Which two improvements of K-means did we mention around initialization?
- How can we mitigate the effect of outliers on K-Means?
- When would you use the elbow method? When would you not?
- How does the silhouette score measure something different from inertia?
- What are the three most notorious failures of K-Means, and as a teaser how can we mitigate those?
- Given a new dataset, what would be your first three steps before running K-Means?

RECAP of Key Concepts:

- K-Means minimizes inertia by alternating assignment (points to nearest centroid) and update (centroid to cluster mean)
- Multiple random initializations is a common heuristics to mitigate local minima
- The elbow and silhouette score are complementary heuristics for choosing K
- K-Means assumes spherical, equally sized clusters; it fails on rings, elongated shapes, and density differences

K-MEANS IS FLAWED. Every failure mode in this chapter reduces to the same problem; Euclidean distance to a centroid is the wrong measure of membership for that data. But there is a more structural

limitation: K-Means must commit to a single K. With K=3 you see roast regions; with K=10 you see brewing methods. You cannot see both at once, and the elbow heuristic only helps you pick one level and drop the remaining pattern.

Teaser How can we built the full hierarchy of clusterings first, capture the complete pattern across all levels, and allow for post hoc selections?

feedback