Lecture 05

Question: Will K-means always converge?

Yes. Each iteration reduces the sum of squared distances between points and their assigned centroids, and since there are only a finite number of possible cluster assignments, the algorithm eventually reaches a state where assignments no longer change.

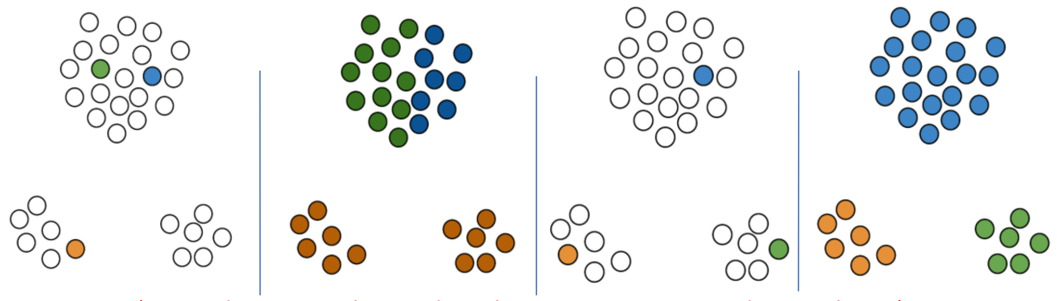
Question: Will K-means always converge to an optimal solution?

No. The K-Means objective function is **non-convex**. It may get stuck in a local minimum depending on the initial placement of centroids, so the final clustering might not be the best possible arrangement of points. Initialization methods like K-Means++ can help improve the chances of finding a better solution.

Farthest First Traversal: is a deterministic initialization method for clustering algorithms like K-Means. The idea is to choose initial centroids that are as far apart as possible to improve clustering quality. Works by

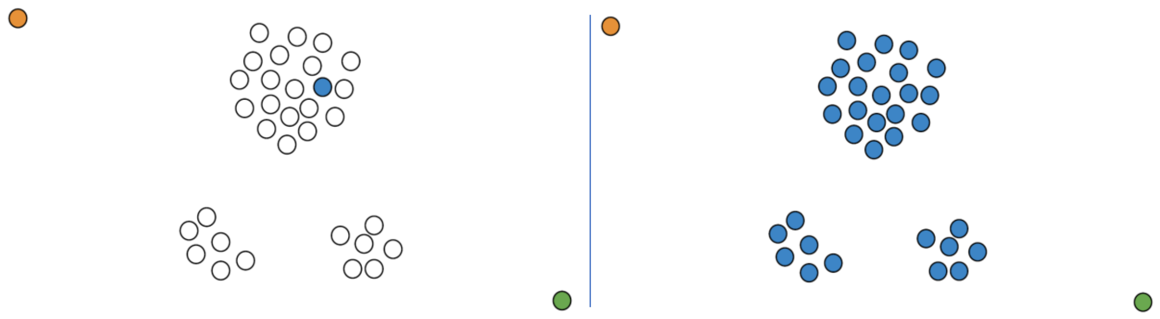
1. **Picking the first centroid** randomly from the dataset.
2. **Iteratively selecting the next centroid** as the point farthest from all previously chosen centroids (using Euclidean distance or another metric).
3. Repeating until **k centroids** are selected.

Helps avoiding poor local minima by spreading initial centroids across the dataset. Often leads to better clustering than purely random initialization.



(a – random center, b – random clusters, c – FFT center, d – FFT Cluster)

However sometimes random might be better.



K – means++

It is an improved initialization method for K-Means that aims to select initial centroids in a way that increases the likelihood of converging to a good clustering. The procedure is:

- Choose the first centroid randomly from the dataset.
- For each remaining point, compute its distance squared to the nearest already chosen centroid.
- Select the next centroid randomly, with probability proportional to its squared distance.
- Repeat until k centroids are chosen, then proceed with standard K-Means iterations.
  <u>Example:</u>
  Suppose we have 5 points: $X = \{1,2,4,8,9\}$. We want to initialize $k = 2$, using k – means++

1. Pick random centroid, let $C_1 = 1$
2. Next compute distances squared to nearest centroid

Point	**Distance to nearest centroid**	**Distance²**
2	2 – 1 = 1	1
4	4 – 1 = 3	9
8	8 – 1 = 7	49
9	9 – 1 = 8	64

3. Select next centroid with probability proportional to distance²
   Compute total distance²: $1 + 9 + 49 + 64 = 123$
   Probability for each point to be chosen as next centroid:

Point	**Distance²**	**Probabilities**
2	1	1/123 = 0.008
4	9	9/123 = 0.073
8	49	49/123 = 0.398
9	64	64/123 = 0.520

4. Based on these probabilities 9 is picked randomly as C_2
5. Proceed with K – means with centroids as $C = \{1,9\}$

Black Box

- It refers to the **randomized selection step** for picking new centroids.
- After the first centroid is chosen randomly, each subsequent centroid is chosen using a probability distribution based on squared distances.

- The exact point that gets picked is determined by a randomized procedure (the **black box**) that samples from this probability distribution.
- In practice, this is done by generating a random number between 0 and 1 and mapping it onto the cumulative probability distribution of the squared distances.

Thus, black box is the random sampling mechanism:

- Input: list of probabilities proportional to squared distances.
- Output: one data point chosen as the next centroid.

Continuing from previous example:

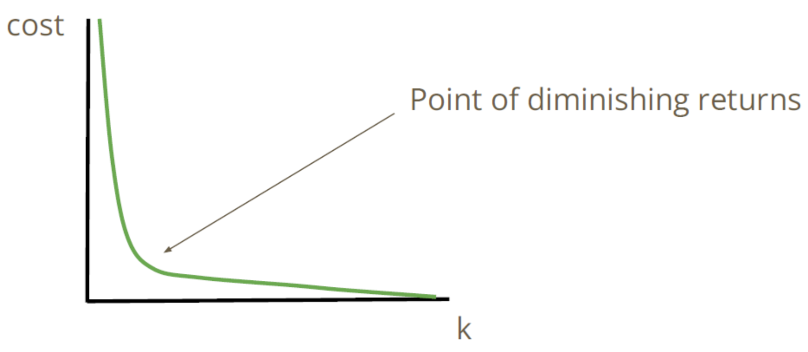
6. After getting all the probabilities we build Cumulative Distribution (CDF)

Point	Probabilities	Probabilities
2	0.008	0.008
4	0.073	0.008 + 0.073 = 0.081
8	0.398	0.081 + 0.398 = 0.479
9	0.520	0.479 + 0.520 = 0.999 (~1.0)

7. Using Black Box
   a. Generate random number r between 0 and 1
   b. Let $r = 0.72$
   c. There 0.72 falls between 0.479 and 0.999. Therefore point 9 is chosen.

Choosing the Right k

1. **Elbow Method:** Plot the within-cluster sum of squares (WCSS) or total squared error against different values of k. Look for an "elbow" point where adding more clusters does not significantly reduce WCSS.



2. **Silhouette Score:** Measures how similar a point is to its own cluster compared to other clusters. Silhouette ranges from -1 to 1:
   a. 1 → well-clustered
   b. 0 → on the boundary
   c. Negative → misclassified

Compute the average silhouette score for different k values. Choose k that maximizes the score.

3. **Gap Statistic:** Compares WCSS of your clustering with WCSS of randomly distributed points. The optimal k maximizes the gap between your clustering and random clustering.
4. **Domain Knowledge:** Sometimes, the best k is informed by the specific application or business problem rather than pure statistics.

Gap Statistic

The Gap Statistic is a method to choose the optimal number of clusters k by comparing your clustering result to random data.

- If your clustering is meaningful, the within-cluster dispersion should be much smaller than what you'd get by clustering randomly distributed points.
- It tries to measure how much better your clusters are than random noise.

Understanding Gap Statistics

Imagine We are sorting candies into jars.

- We have a bunch of candies (our data points).
- We want to put them into k jars (clusters) so that similar candies go together.
- After putting candies in jars, we check: "How close are the candies in each jar?"
- If candies are close → jar is neat → small WCSS.
- If candies are spread out → messy → big WCSS.
- Now, imagine if candies were thrown randomly on the table.
- Even if we try to put them in k jars, the jars won't be neat.
- Measure the messiness (WCSS) of these random jars.
- Gap = "messiness of random jars" – "messiness of your actual jars"
- Big gap → our clustering is much better than random
- Small gap → our clustering is not much better than random

Steps:

- Compute WCSS for your data: WCSS = "within-cluster sum of squares" (sum of squared distances from points to their cluster centroids).
- Generate reference datasets: These are random points with the same shape as your data (same number of points and dimensions), usually uniformly distributed in the bounding box of your original data.
- Cluster the reference datasets: For each k, compute WCSS for the reference datasets.

- Compute the gap:

$$Gap(k) = \frac{1}{B}\sum_{b=1}^{B} \log(W_{kb}^{*}) - \log(W_k)$$

  Where,
  - W_k = WCSS for your real data with k clusters
  - W_{kb}^{*} = WCSS for the b_{th} reference dataset with k clusters
  - B = number of reference datasets
  - $\log(W_{kb}^{*}) - \log(W_k)$: bigger gap means your clusters are much better than random.
- Choose the k with maximum gap.

Unlike WCSS or Elbow, Gap adds a **baseline for comparison**. It asks: "*Is my clustering actually meaningful, or would random points look just as good?*" it avoids overestimating k.

Evaluation Metrics

1. Centroid Distance

- **What it is:** Distance between cluster centres.
- **Goal:** Larger distance → clusters are well-separated.

2. Silhouette Score

- **What it is:** Measures how close each point is to its own cluster vs nearest other cluster.
- **Range:** -1 to 1
- **Goal:** Higher → points well-clustered; lower → overlapping clusters.

3. Davies-Bouldin Index (DBI)

- **What it is:** Ratio of cluster spread to distance between clusters.
- **Goal:** Lower → clusters are tight and far apart; higher → clusters overlap.

4. Between-Cluster Sum of Squares (BCSS)

- **What it is:** Sum of squared distances from cluster centroids to overall mean.
- **Goal:** Higher → clusters are spread out from global mean; lower → clusters close together.

5. WCSS (Within-Cluster Sum of Squares)

- **What it is:** Sum of squared distances of points to their own cluster centroid.
- **Goal:** Lower → clusters are tight (cohesive).

6. Gap Statistic

- **What it is:** Compares WCSS of real data to WCSS of random points.
- **Goal:** Larger gap → clusters much better than random → optimal k.