Clustering:

- A clustering is a grouping / assignment of objects (data points) such that objects in the same group / cluster are:
    - similar to one another
    - dissimilar to objects in other groups
- Applications:
    - Outlier detection / anomaly detection
        - Data Cleaning / Processing
        - Credit card fraud, spam filter etc.
    - ●Filling Gaps in your data
        - Using the same marketing strategy for similar people
        - Infer probable values for gaps in the data (similar users could have similar hobbies, likes / dislikes etc.)
- The clustering problem:
    - Given a collection of data points
        - Find a clustering such that:
            - Similar data points are in the same cluster
            - Dissimilar data points are in different clusters
        - Questions:
            - -What does similar mean?
            - -How do we find a clustering?
            - How do we know if we have found a good clustering?
        - Types of clusterings:
            - **Partitional**
                - Each object belongs to exactly one cluster
                - Given **n** data points and a number **k** of clusters: partition the **n** data points into **k** clusters. Suppose we are given all possible ways of distributing these **n** data points into these **k** buckets / clusters. How would we find the best such partition? Recall our goal: **similar** items should belong to the **same cluster** & **dissimilar** items should belong to **different clusters**.
                - A good partition is one where the total dissimilarity of points within each cluster is small.
                    - Example:
                        - Clearly the clustering on the left has smaller intra-cluster distances than the one on the right. That is:
                        - $$\sum_{k}^{K} \sum_{x_i, x_j \in C_k} d(x_i, x_j)$$

- Given a distance function **d**, we can find points (not necessarily part of our dataset) for each cluster called **centroids** that are at the center of each cluster.
  - Q: When **d** is Euclidean, what is the **centroid** (also called **center of mass**) of **m** points **{x_1, … , x_m}** ?
  - A: The mean / average of the points
  - $$\sum_{k}^{K} \sum_{x_i, x_j \in C_k} d(x_i, x_j)^2 = \sum_{k}^{K} |C_k| \sum_{x_i \in C_k} d(x_i, \mu_k)^2$$
- K-means:
  - Given **X = {x_1, … , x_n}** our dataset and **k**
  - Find **k** points **{μ_1, … , μ_k}** that minimize the **cost function**:
    - $$\sum_{i}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2$$
    - When **k=1** and **k=n** this is easy. Why?
    - When **x_i** lives in more than 2 dimensions, this is a very difficult (**NP-hard**) problem
  - Lioyd's Algorithm:
    - 1.Randomly pick **k** centers **{μ_1, … , μ_k}**
    - 2.Assign each point in the dataset to its closest center
    - 3.Compute the new centers as the means of each cluster
    - 4.Repeat 2 & 3 until convergence
  - WIll it always converge:
    - **Proof** (by contradiction): Suppose it does not converge. Then, either:
    - 1.The minimum of the cost function is only reached in the limit (i.e. after an infinite number of iterations).
    - **Impossible** because we are iterating over a finite set of partitions
    - 1.The algorithm gets stuck in a cycle / loop
    - **Impossible** since this would require having a clustering that has a lower cost than itself and we know:

            - -If old ≠ new clustering then the cost has improved
            - -If old = new clustering then the cost is unchanged
            - **Conclusion**: Lloyd’s Algorithm always converges!
    - Initialization:
        - One solution: Run Lloyd’s algorithm multiple times and choose the result with the lowest cost.
        - This can still lead to bad results because of randomness.
        - Another solution: Try different initialization methods
- How to choose the right k:
    - Iterate through different values of k (elbow method)
    - Use empirical / domain-specific knowledge
    - Example: Is there a known approximate distribution of the data? (K-means is good for spherical gaussians)
- K-means variations:
    - K-medians (uses the L_1 norm / manhattan distance)
    - K-medoids (any distance function + the centers must be in the dataset)
    - Weighted K-means (each point has a different weight when computing the mean)
- **Hierarchical**
    - A set of nested clusters organized in a tree
- **Density-Based**
    - Defined based on the local density of points
- **Soft Clustering**
    - Each point is assigned to every cluster with a certain probability