CS 337, Fall 2023
K-means Clustering

Scribes: Guramrit Singh, Anshul Verma, Erata Maharshi, Sayantika Mandal
Atishay Jain, Akshat Singh, Aryan Kamat [*]
Edited by: Parshant Arora

October 5, 2023

Disclaimer. Please note this document has not received the usual scrutiny that formal publications enjoy. This may be distributed outside this class only with the permission of the instructor.

1 Unsupervised Learning

Recall that until now we have been dealing with ***labeled*** datasets. *For example*, in K-class classification, datapoints were of the form $(\boldsymbol{x}_i, y_i)$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, 2, \ldots, K\}$ was the corresponding class label for each $i \in \{1, 2, .., N\}$.

Now, we shift our focus to **unsupervised learning**. *The natural first question to ask is -*
What is Unsupervised Learning?
Unsupervised learning, also known as *unsupervised machine learning*, uses machine learning algorithms to analyze and cluster ***unlabeled*** datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for *exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.*[1]

Unsupervised learning models are utilized for three main tasks :

- Clustering (eg: K-means Clustering)
- Dimensionality reduction (eg: Principal Component Analysis)
- Association Rule Mining (eg: Apriori algorithm)

In this lecture, we will focus on Clustering and more specifically *K-means Clustering.*

2 Clustering

Clustering is a specific task within unsupervised learning that involves grouping similar data points together into clusters or categories based on some similarity or distance metric.

[*] All scribes get equal credit for the final notes.
[1] Source : IBM

2.1 Clustering setting

Consider a set of points $\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_n$ where each $\boldsymbol{x}_i$ belongs to Euclidean space $\mathbb{R}^d$ and is equipped with Euclidean norm. Assume that each datapoint $\boldsymbol{x}_i$ belongs to one of the $\boldsymbol{K}$ clusters where $\boldsymbol{K}$ is a hyperparameter.
Note : We want the points within each cluster to be "similar", i.e. close to one another in terms of Euclidean distances.

Next question as you would have already figured out is how do we identify such clusters of datapoints?
Before proceeding to address the above question, let's look at some of the scenarios where *clustering* algorithms play a crucial role.

(a) **Stock sector analysis**: Group stocks into sectors or industries based on similarities in financial metrics, business models, or other relevant factors.

(b) **Customer segmentation**: Identify groups of customers with similar purchasing behavior, allowing businesses to tailor marketing strategies for each segment.

(c) **Social Network**: Recognize communities or groups within a social network, helping to understand patterns of connections and interactions.

(d) **Topic Clustering in documents**: A crucial Natural Language Processing (NLP) task that involves grouping documents with common themes or subject matter together.

(e) **Stars / Celestial Bodies** - Astronomers employ clustering to categorize celestial bodies based on their spectral characteristics, luminosity, or evolutionary stages. Mostly, the astronomical data is unlabelled.

(f) **Image Segmentation** - In computer vision, it can be used for separating an image into regions with similar characteristics, which is useful in object recognition, medical imaging, and more.

2.2 Motivation

- **Labelling Efficiency**: Unlabeled dataset can be labeled by assigning the datapoints with labels corresponding to their respective cluster indices.
- **Outlier detection**: Outliers can manifest as distinct clusters containing only a single datapoint. For instance, in a dataset comprising 1000 datapoints with 5 outliers, and assuming K is set to 6, the optimal cluster assignment would segregate the 5 outlier datapoints into separate clusters, while the remaining 995 datapoints would form a single cluster.
- **Compression (Lossy)**: Using the K-means algorithm the data can be represented using fewer bits by encoding each point with the index of the cluster to which it belongs. Large datasets can be simplified by representing them with a smaller number of cluster centroids.

3 K-means clustering

The K-Means clustering algorithm is one of the most popular, easy and elegant algorithm used for clustering data. In this algorithm, before starting with the process of clustering, we have to assume that there are **K** clusters of the dataset. Thus, **K** here acts as a parameter to be decided initially.

3.1 Intuition

Assume that there are K clusters. We have the following observations:

- Every datapoint is close to the center (i.e. mean) (*also referred to as centroid*) of its assigned cluster.
- If we keep the *cluster centers fixed*, assigning each datapoint becomes straightforward. We simply calculate the Euclidean distance between each datapoint and every cluster center, and assign the datapoint to the cluster whose center is closest in terms of the Euclidean distance.
- Conversely, once the *assignments are fixed*, determining the cluster centers is a straightforward process. It involves computing the mean of all the points assigned to that particular cluster.

This scenario exemplifies the classic chicken-and-egg problem!
We will discuss a heuristic algorithm i.e. fix one set of parameters, optimize for the other, and then alternate between them (***Alternating Minimization***) as a solution to the above problem.

3.2 Objective

Find the cluster centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$; $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and assignments $\boldsymbol{C}^1, \boldsymbol{C}^2, \ldots, \boldsymbol{C}^n$ such that the sum of squared distances of all datapoints $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$; $\boldsymbol{x}_i \in \mathbb{R}^d$ from their assigned cluster centers is minimized.

Here each $\boldsymbol{C}^i$ is a *one-hot encoding* of size K. For example, if $\boldsymbol{x}_i$ is assigned j^{th} cluster then

$$\boldsymbol{C}^i = (0, \ldots, \underbrace{1}_{j^{th}}, 0, \ldots, 0)^T \quad \text{i.e.} \quad C^i_l = 1 \quad \text{for} \quad l = j \quad \text{and} \quad C^i_l = 0 \quad \text{otherwise}$$

Formally,

$$\min_{\{\mu_k\}} \min_{\{C^i\}} \sum_{i=1}^{n} \sum_{k=1}^{K} C^i_k \|\mu_k - x_i\|^2$$

where $C^i_k = 1$ [iff x_i is assigned to cluster k]

which means that we want to find both $\{\mu_k\}$ (set of cluster centers) and $\{C^i\}$ (set of assignments) which minimize the above cost function. The optimal solution for this problem is **NP-Hard**, which is why K-Means follow the easy heuristic as discussed before -

K-Means Heuristic: Fix one, optimize for the other

3.3 Heuristics

During each iteration, we perform two steps. We either fix one of the assignments or the means, then optimize the other, and continue this alternating process until convergence. [2]

Step-1: Fix $\boldsymbol{\mu}$; Optimize the assignment. i.e. for the i^{th} datapoint optimize the following function:

$$\sum_{k=1}^{K} C^i_k \ \|\boldsymbol{\mu}_k - \boldsymbol{x}_i\|^2$$

[2] When no assignments change during an iteration, we consider the algorithm to have converged.

The optimal value for the i^{th} point is obtained at:

$$C_k^i = \begin{cases} 1 & \text{if } k = argmin_j \; \|\boldsymbol{\mu}_j - \boldsymbol{x}_i\|^2 \\ 0 & \text{otherwise} \end{cases}$$

Step-2: Fix the assignments $\boldsymbol{C}$ and optimize for the cluster means

$$\text{For } k = 1, 2 \ldots K$$

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{i=1}^{n} \sum_{j=1}^{K} C_j^i \; \|\boldsymbol{\mu}_j - \boldsymbol{x}_i\|^2 = 0$$

$$\implies 2 \sum_{i=1}^{n} C_k^i \; (\boldsymbol{\mu}_k - \boldsymbol{x}_i) = \mathbf{0}$$

$$\implies \boxed{\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} C_k^i \; \boldsymbol{x}_i}{\sum_{i=1}^{n} C_k^i}}$$

3.4 Algorithm

```
Algorithm 1: K-Means Clustering Algorithm
Data: Dataset $X$, Number of clusters $K$
Result: Cluster centers $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K\}$, Cluster assignments $\{\boldsymbol{C}^1, \boldsymbol{C}^2, \ldots, \boldsymbol{C}^n\}$
Initialisation: Initialise the cluster centers randomly $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\}$;
repeat
    Assignment: // Assign each datapoint to its nearest cluster center
        for $i = 1, 2, \ldots, n \quad C_k^i = \begin{cases} 1 & \text{if } k = argmin_j \; \|\boldsymbol{\mu}_j - \boldsymbol{x}_i\|^2 \\ 0 & \text{otherwise} \end{cases}$
    Update: // Recompute the cluster centres to be the mean of all datapoints assigned to that cluster
        for $k = 1, 2, \ldots, K \quad \boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} C_k^i \; \boldsymbol{x}_i}{\sum_{i=1}^{n} C_k^i}$
until convergence i.e. assignments do not change;
```

Given the NP-hard nature of the actual optimization problem, we do not expect the K-means algorithm to attain the global optimum. Instead, we hope for it to converge towards a local minimum.

Question: Is the algorithm guaranteed to converge and terminate?

To answer this, we have the following two observations:

- The cost reduces (or remains the same) after every iteration of the algorithm.
- The number of cluster assignments is finite since both datapoints and clusters are *finite* in number.

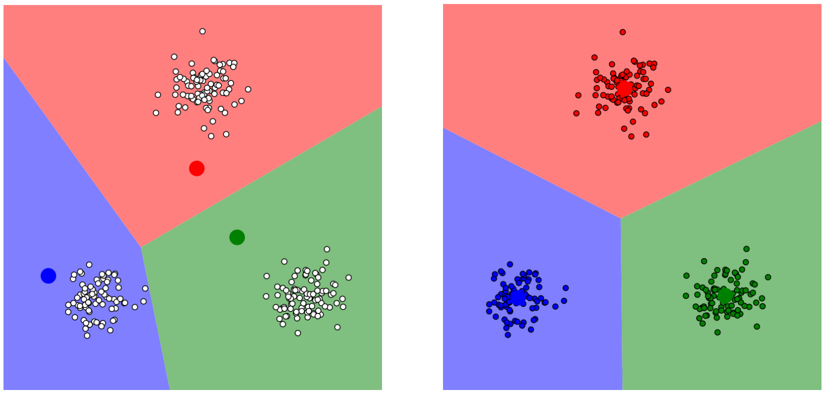
The above observations guarantee the convergence and termination of the algorithm. Note that convergence is guaranteed to a local minimum but the number of iterations to achieve it can be very large.

3.5 Drawbacks

One of the *drawbacks* of the standard K-means algorithm is that it is **highly sensitive to the initialization** of centroids or the mean points.

- Initializing more than one centroid within the same cluster can result in suboptimal clustering.
- If a centroid is initialized to a remote location, it runs the risk of being orphaned with no associated points, and concurrently, multiple clusters may become affiliated with a singular centroid.

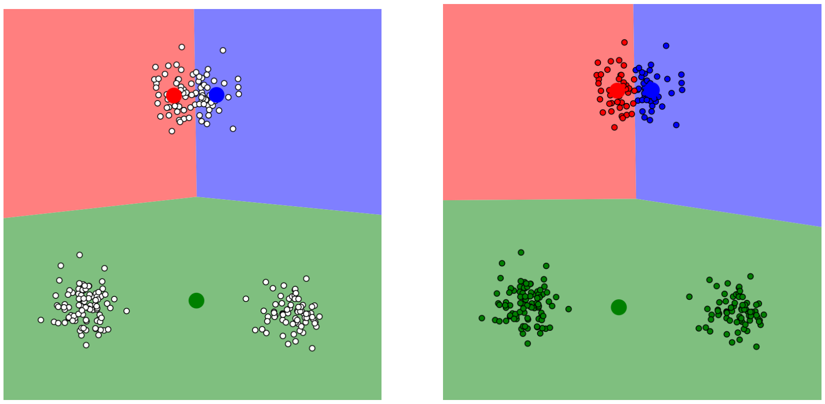
As an illustration, let's examine the figures provided below.



(a) Initialization of Cluster centers

(b) Correctly clustered the data

Figure 1: K-Means on a Gaussian Mixture dataset



(a) Initialization of Cluster centers

(b) Poorly clustered by K-Means

Figure 2: K-Means on same dataset with different initialization

- In Figure 1, the K-Means algorithm converged to separate the dataset into 3 groups as expected
- But in Figure 2, it already converged before arriving at the correct clusters due to the dense cluster at the top, because of which there was no motivation for the cluster centers to move further.
- This shows that the algorithm is highly **sensitive** to the initialization of the cluster centers.

- Another example of K-Means inefficiency is that if we use it on a dataset which has a circular ring with groups inside it (for example, a smiley face), it can never classify it because of the shape of the dataset
- For using K-Means on such dataset, one way is to use the appropriate transformation of data. Another way is to use a large enough value of **K** and then merge some clusters, which is difficult.

4 Further Exploration: K-means^{++} (Additional Reading)

4.1 Overview

To address the limitation associated with centroid initialization, the K-means^{++} method [3] (proposed in *2007 by David Arthur and Sergei Vassilvitskii*) offers an intelligent alternative. This approach aims to distribute initial centroids effectively. The first cluster center is chosen uniformly at random from the data points after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center. After the centroid initialization, we proceed with the K-means algorithm described in **??**

4.2 Algorithm

Algorithm 2: K-Means^{++} Clustering Algorithm

```
Data: Dataset $X$, Number of clusters $K$
Result: Cluster centers $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K\}$, Cluster assignments $\{\boldsymbol{C}^1, \boldsymbol{C}^2, \ldots, \boldsymbol{C}^n\}$
Initialisation: Initialise $\boldsymbol{\mu}_1$ uniformly at random from the data points;
repeat
    for $\boldsymbol{x}$ such that $\boldsymbol{x}$ is not already a centre do
        $$D(\boldsymbol{x}) = \min_{\boldsymbol{\mu} \in \Gamma} \|\boldsymbol{x} - \boldsymbol{\mu}\| \text{ where } \Gamma \text{ represents set of already chosen centers}$$
    choose new center from $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \setminus \Gamma$ with probability of each $\boldsymbol{x}$ given by $P(\boldsymbol{x}) \propto D(\boldsymbol{x})^2$
until K centres are not chosen;
repeat
    Assignment: // Assign each datapoint to its nearest cluster center
        $$\text{for } i = 1, 2, \ldots, n \quad C_k^i = \begin{cases} 1 & \text{if } k = argmin_j \ \|\boldsymbol{\mu}_j - \boldsymbol{x}_i\|^2 \\ 0 & \text{otherwise} \end{cases}$$
    Update: // Recompute the cluster centers to be the mean of all datapoints assigned to that cluster
        $$\text{for } k = 1, 2, \ldots, K \quad \boldsymbol{\mu}_k = \frac{\sum_{i=1}^n C_k^i \ \boldsymbol{x}_i}{\sum_{i=1}^n C_k^i}$$
until convergence i.e. assignments do not change;
```

[3] Source : Wikipedia

4.3 Approximation guarantees

The K-means^{++} algorithm guarantees an approximation ratio $\boldsymbol{O(log(K))}$ in expectation (over the randomness of the algorithm), where K is the number of clusters used.[4]

5 Soft Assignments

So far, K-means is a HARD Assignment, meaning that each data-point is assigned to a single cluster. In other words, the assignment vector is one-hot encoded. Soft assignment refers to a probabilistic assignment where each data point's assignment vector has a distribution over clusters. That is, each data-point is assigned to all the clusters with different weights or probabilities. This is useful because it keeps the possibility of a point belonging to more than 1 cluster, especially when the data is not easily separable and complex. For soft assignments, we use the assignment vector C_k^i as :

$$\boxed{C_k^i = \frac{\exp(-\beta \|\mu_k - x_i\|^2)}{\sum_j \exp(-\beta \|\mu_j - x_i\|^2)}}$$

which is the **SoftMax** function, with a parameter β introduced. β reflects the amount of flexibility in our soft assignment and is called the stiffness parameter. As $\beta \to \infty \implies$ Hard K-Means, because essentially c^i follows a Gaussian distribution, so β can be seen as an inverse variance. Hence, larger values of β will cause the Gaussian to be narrower. So, at infinite, it would resemble the one hot encoded vector of Hard K-Means. However, learning the parameter β is challenging, which leads to Gaussian Mixture models and EM algorithm, which we will learn in the next class.

[4] This is in contrast to vanilla K-means, which can generate clusterings arbitrarily worse than the optimum.