Foundations of Data Science & Machine Learning

TERM PAPER

Devansh Singh Rathore
111701011

B.Tech. in Computer Science & Engineering
Indian Institute of Technology Palakkad

May 12, 2021

Abstract

In the following chapter we will discuss about clustering in data science, particularly about K-means clustering. For the same, we will look into two different algorithms namely Lloyd's algorithm and Ward's algorithm. While discussing Lloyd's algorithm, we will study about Farthest Traversal method for cluster's center initialisation and then will study about convergence proof of this algorithm. We also look into the proof of executing k-means clustering in polynomial time complexity for $\mathbb{R}^1$ case. Further, we discuss kernelised K-means clustering technique. In the end, we discuss different real life applications of K-means clustering.

1 K-means Clustering

Definition: The unsupervised learning method used for dividing data points in different clusters is called **Clustering**. Mathematically, we can imagine set of data points $D = \{r_1, r_2, ..., r_n\}$, where a data point $r_i = (x_1^i, x_2^i, ..., x_d^i)$ is in d - dimensional space i.e. $\forall i \; r_i \in \mathbb{R}^d$. The main goal of a clustering algorithm is to allocate each data point r_i to its *closest* cluster c_j from a set of clusters $C = \{c_1, c_2, ..., c_m\}$.

$\rightarrow$ *Closeness* is decided in terms of Euclidean distance. Particularly in K-means clustering we calculate distance between *centroid* v of a cluster and a point r, to estimate their closeness.

$$Dis(v, r) = |v - r| = \sqrt{\sum_{l=1}^{d} (v_l - x_l)^2}$$

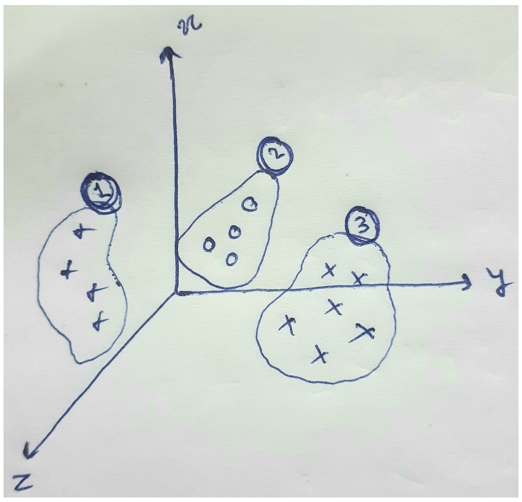

Fig 1.0 Clustering example in $\mathbb{R}^3$

$\rightarrow$ So our basic aim is to assign a cluster to each data point s.t. its distance with the center of the cluster is minimized. i.e. we try to minimize loss (L):

$$L = \sum_{j=1}^{k} \sum_{i=1}^{n} a_{ij} |center_j - r_i|^2$$

$$where\ a_{ij} = \begin{bmatrix} 1 & ,if\ r_i\ is\ assigned\ to\ cluster\ j \\ 0 & ,otherwise \end{bmatrix}$$

assuming k clusters.

1.1 Lloyd's Algorithm

$\rightarrow$ The brute-force method of K-means clustering is Lloyd's algorithm.

$\rightarrow$ Steps:

* Take k different centers $\{v_1, v_2, ..., v_k\}$.

* For every data point r_i, find closest center v_l and put it in that cluster c_l. i.e. $\forall i \in [n]$:

$$l = argmin_{s \in [k]} |v_s - r_i|$$

$$cluster(r_i) = c_l$$

* Calculate centroid of each cluster and appoint it as the new center of that cluster. Note that the centroid is defined as the mean of all points in a cluster. (It can also be taken as the data point closest to the mean.)

* Repeat the last two steps until the set of centers do not change.

$\rightarrow$ Note that Lloyd's algorithm does not find globally optimal solution but a locally optimal solution. So clustering can vary according to the initial centers that were chosen.

$\rightarrow$ Hence there can be cases where a cluster might get empty.

$\rightarrow$ A known strategy for finding the initial k centers is called **"Farthest Traversal"**. Similar initialisation method is used in **K-means++** algorithm. Method:

* Choose c_1 as a random center. Keep it in a set C. So initially $C = \{c_1\}$.

* i^{th} step: Choose c_i to be the farthest data point from points in set C. Add c_i in C.

* Repeat until we get k centers in C. i.e. $|C| = k$

$$prob(choose\ r_t\ as\ centroid\ in\ i^{th}\ iteration) \propto \sum_{j=1}^{i-1} |r_t - c_j|^2$$

Lemma 1.0: If we have a set of data points $D = \{r_1, r_2, ..., r_n\}$ with centroid c, then the sum of square of distances of any point x from each of data point in D is equal to sum of square of distances between each data point in D and centroid c plus n times the square of distance between centroid c and x. i.e.

$$\sum_{i=1}^{n} |x - r_i|^2 = \sum_{i=1}^{n} |c - r_i|^2 + n * |c - x|^2$$

where

$$c = (1/n) \sum_{i=1}^{n} r_i$$

Proof:

$$\begin{aligned}\sum_{i=1}^{n} |x - r_i|^2 &= \sum_{i=1}^{n} |(r_i - c) + (c - x)|^2 \\ &= \sum_{i=1}^{n} (|r_i - c|^2 + 2(r_i - c)(c - x) + |c - x|^2) \\ &= \sum_{i=1}^{n} (|r_i - c|^2) + 2(c - x) \sum_{i=1}^{n} (r_i - c) + \sum_{i=1}^{n} (|c - x|^2) \\ &= \sum_{i=1}^{n} (|r_i - c|^2) + 2(c - x) \sum_{i=1}^{n} (r_i - c) + n(|c - x|^2)\end{aligned}$$

Since c is the centroid,

$$\sum_{i=1}^{n} (r_i - c) = \sum_{i=1}^{n} (r_i) - n * c = n * c - n * c = 0$$

So,

$$\sum_{i=1}^{n} |x - r_i|^2 = \sum_{i=1}^{n} (|r_i - c|^2) + n(|c - x|^2) \qquad -(*)$$

Corollary 1.0: If x is the centroid of the cluster, then we can minimize the sum of square of distance between each data point in cluster and point x. In the RHS of the expression (*), $\sum_{i=1}^{n} (|r_i - c|^2)$ is constant (independent of x) and the term $n(|c - x|^2)$ can only be minimized to 0 and when $c = x$.

Claim: Lloyd's algorithm will always converge.

Proof of convergence:

Subclaim 1: sum of square of distance of each data point in a cluster from cluster's center always decreases.

Subclaim 1 can be proved using the *Corollary 1.0* and the fact that in each iteration we are modifying center to the mean of all the data points in a cluster. Since the centroid brings the lowest sum (*Corollary 1.0*), we can say that *Subclaim 1* is true. Hence for every cluster either their sum of square of distance from center decreases or otherwise remains same. Therefore the total loss L also decreases.

Subclaim 2: At each iteration, even if some data points shift to a new cluster, the loss L decreases.

For supporting our *Subclaim 2*, let's suppose a data point r_t is in cluster c_{l1} with total loss L_1 and now shifts to c_{l2} giving total loss L_2. So,

$$L_2 = L_1 - |r_t - center_{l1}|^2 + |r_t - center_{l2}|^2$$

since

$$|r_t - center_{l1}| > |r_t - center_{l2}|$$

therefore

$$L_2 < L_1$$

We can generalize the same for n' number of points to be shifted in a iteration, which gives the same result.

Using *Subclaim 1* and *Subclaim 2* we can clearly conclude that in each iteration the total loss L reduces, hence the Lloyd's algorithm will always converge.

1.2 Ward's Algorithm

→ Also known as greedy K-means algorithm.

→ Steps:

* Keep all points r_i in their own cluster. i.e. we have n clusters for n data points.

$$cluster(r_i) = c_i$$

* If we define cost function as:

$$cost(C) = \sum_{a_i \in C} dis^2(c, a_i), \text{ where c is the centroid of C}$$

Then, merge the pair of clusters (C, C') if it minimizes: $cost(C \cap C') - cost(C) - cost(C')$

* Repeat until we reduce the number of clusters to k.

→ This algorithm tries to minimize the *immediate increase in k-means score.*

1.3 K-means Clustering in $\mathbb{R}^1$

$\rightarrow$ When the data points lie on a line i.e. in $\mathbb{R}^1$, then we can compute K-means clustering in polynomial time using Dynamic Programming (DP) technique.

Task: Compute K-means cluster for n data points in polynomial time, assuming the data points $\{r_1, r_2, ..., r_n\}$ are sorted i.e. $r_1 \leq r_2 \leq ... \leq r_n$.

Obs.: A cluster will contain only consecutive points.

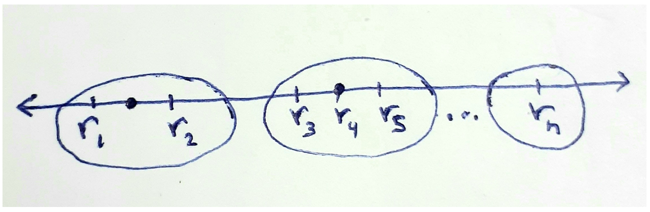

Fig 1.1 Clustering example in $\mathbb{R}^1$

Base Case: For n = 1, we can compute optimal k' means clustering $\forall k' \leq k$. Trivial Case.

Hypothesis: Let's assume that for every $n \leq i$ points, we can find an optimal k' means clustering $\forall k' \leq k$.

Goal: To prove that the claim is true for $n = (i + 1)$ as well.

Strategy for n = (i+1): for a given k':

Let

$$A_j^{k'} = \sum_{p=j}^{p=(i+1)} |r_p - \mu_j|$$

$$where\ \mu_j = (1/(i - j + 2)) \sum_{s=j}^{s=(i+1)} r_s$$

and

$$B_{j-1}^{k'-1} = L(k' - 1, j - 1)$$

where $L(k'-1, j-1)$ = cost of optimal $k'-1$ clustering among $\{r_1, ..., r_j - 1\}$ then,

$$L(k, (i + 1)) = min_j(A_j^k + B_{j-1}^{k-1}) \ \ , 1 \leq j \leq (i + 1)$$

Time Complexity: This has total time complexity of $O(kn)$ for a given i. Hence the total time complexity of this DP based algorithm is $O(kn^2)$.

1.4 Kernalised K-means

$\rightarrow \Phi : \mathbb{R}^d \rightarrow H$

$\rightarrow$ Algorithm:

* Initialisation: we choose k centers $c_1, c_2, ..., c_k$ from Hilbert space H.

* Keep the same iteration steps of finding closest center for each data point, recalculating the centers and then re-clustering. Just that for calculating distance use $|\Phi(r_i) - c_j|^2$. i.e.

$$l = argmin_{s \in [k]} |\Phi(r_i) - c_s|$$

then

$$cluster(r_i) = c_l$$

1.5 Applications:

* Recommendation Systems: According to a user's preference in different fields(dimensions), we can form a data point for each user specifying his/her parameter of interest as each coordinate of this data point. Then we can find a cluster closest to this user's data point and use other points in that cluster as recommendations.

* Image Segmentation: Cluster pixels with close to each other and with similar pixel values into a common cluster to segment a particular object in a picture.