K-Means Clustering and K-Means++

date: 20261103
status: #finished
tags: #machine_learning #clustering #k_means #k_means_plus_plus #elbow_method

---

0. Setup

Conceptually, we think our data contain K clusters denoted by $\mathcal{K} = \{1, 2, ..., K\}$. Let $\mathcal{D}_{\text{data}} = \{x_1, x_2, ..., x_n\}$ where $x_i \in \mathbb{R}^d$ for all i. What we want is to find the cluster label $c_i \in \mathcal{K}$ for each data point x_i.

Each cluster $k \in \mathcal{K}$ is associated with a centroid $\mu_k \in \mathbb{R}^d$. We denote $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ as the set of the data's cluster labels and $\mu = \{\mu_1, \mu_2, ..., \mu_K\}$ as the set of the clusters' centroids.

---

1. The Algorithm

K-means clustering is an algorithm that helps us identify $\mathcal{C}$ and μ. Moreover, it's an iterative algorithm, and the procedure is as follows.

- Randomly initialize the centroids μ.
- Repeat until $\mathcal{C}$ and μ converge: {

1. Assign the cluster label for each data point by

$$c_i = \arg\min_{k \in \mathcal{K}} \|x_i - \mu_k\|^2.$$

2. Update the centroid by

$$\mu_k = \frac{1}{|\Omega_k|} \sum_{i \in \Omega_k} x_i \text{ where } \Omega_k = \{i : c_i = k\}.$$

}

2. Discussion

(i) Is the convergence guaranteed?

Yes. The whole procedure minimizes the objective

$$J_K(\mathcal{C}, \mu) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{c_i = k\} \|x_i - \mu_k\|^2,$$

which decreases monotonically. Note that $J_K(\mathcal{C}, \mu)$ is also known as the **distortion function**. The underlying optimization problem is

$$\min_{\mathcal{C} \in \mathcal{K}^n} \min_{\mu \in \mathbb{R}^{K \times d}} J_K(\mathcal{C}, \mu).$$

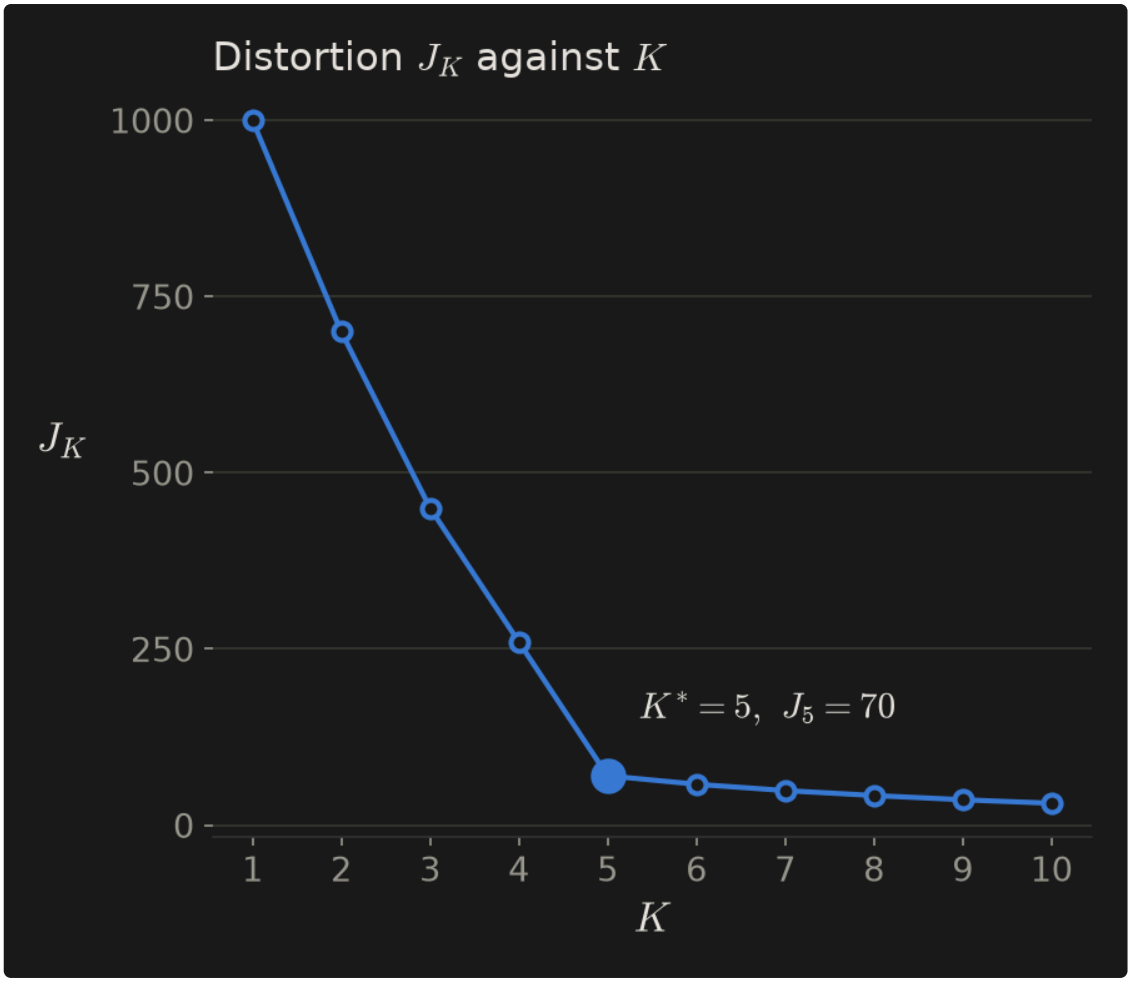
> **Remark**
>
> This objective is **non-convex**, so K-means clustering is not guaranteed to reach the global minimum.

(ii) Is there any problem with the random initialization?

Yes. Different initializations can have different convergence rates. Since the global minimum is not guaranteed, different initializations also yield distinct local minima.

(iii) How to determine the number of centroids K?

K is a hyperparameter that needs to be tuned, and one way to do it is the **elbow method**. Specifically, we generate a 2-D plot where the x-axis is the value of K, from 1 to $K_{\max}$, and the y-axis is the corresponding value of the distortion function, from J_1 to $J_{K_{\max}}$, which looks like the following where $K_{\max} = 10$.



The optimal K^* is the K after which J_K decreases only slowly. The intuition is that although J_K decreases in K, for $K > K^*$ each step decreases it much less, because each extra centroid only splits a cluster that is already tight.

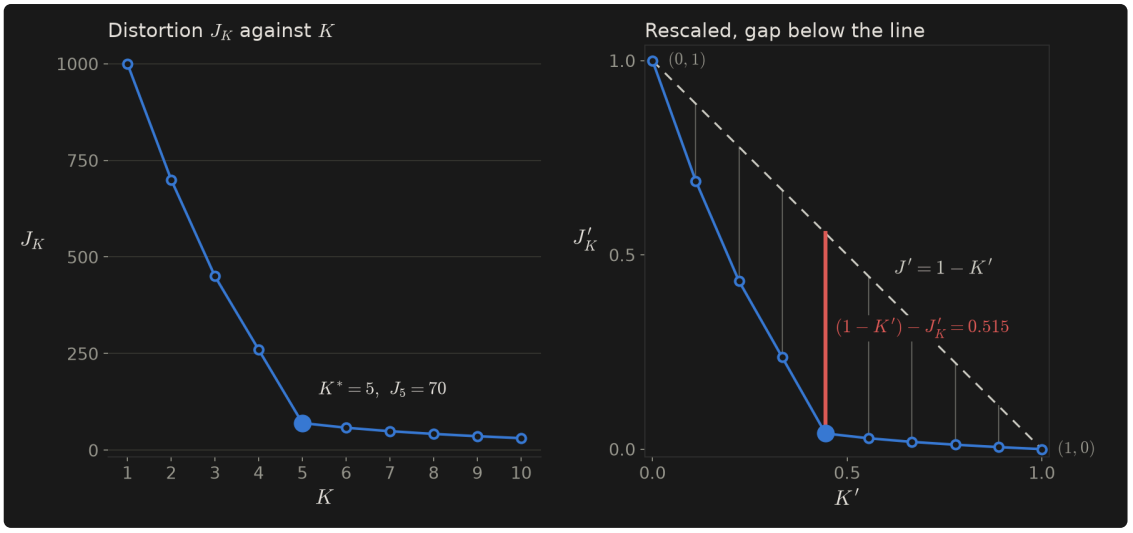
To pick K^* without eyeballing the plot, we take the point that lies farthest below the straight line joining the first point $(1, J_1)$ and the last point $\left(K_{\max}, J_{K_{\max}}\right)$. It is convenient to rescale both axes to lie between 0 and 1:

$$K' = \frac{K-1}{K_{\max}-1}, \quad J'_K = \frac{J_K - J_{K_{\max}}}{J_1 - J_{K_{\max}}}.$$

After rescaling, the first point is $(0, 1)$, the last point is $(1, 0)$, and the line joining them is $J' = 1 - K'$. The gap below this line at K is $(1 - K') - J'_K$, and K^* is the K with the largest gap:

$$K^* = \arg \max_{1 \le K \le K_{\max}} (1 - K') - J'_K.$$

The rescaling does not change which K wins, since the raw gap equals $\left(J_1 - J_{K_{\max}}\right)$ times the rescaled gap. It only makes the gap's size unit-free.



The same rule tunes ε in DBSCAN, where the number of clusters is a result of the algorithm rather than a hyperparameter.

---

3. K-Means++

K-Means++ is an improvement of K-Means on the initialization, and that is its only difference from K-Means.

K-Means++ chooses the initial K centroids as follows.

1. Randomly pick the first centroid.
2. For each subsequent centroid, first compute the distance from each non-centroid point to its nearest centroid. Then pick the next centroid with probability proportional to the squared computed distance.