K-Means Clustering from Scratch in Python

Shaojun Yu

[1] *Emory University*

Abstract. K-Means Clustering is an unsupervised learning algorithm that aims to group the observations in a given dataset into clusters. The number of clusters is provided as an input. It forms the clusters by minimizing the sum of the distance of points from their respective cluster centroids. Here I implemented the K-Means Clustering from Scratch in Python. For source code see Github Repo.

1 Basic Overview

Clustering is a type of unsupervised learning which is used to split unlabeled data into different groups. It is generally used in Data Analysis to get to know about the different groups that may exist in datasets, so that the data points in the same group have similar characteristics than the data points in different groups.

K-Means follows an iterative process in which it tries to minimize the distance of the data points from the centroid points. It's used to group the data points into k number of clusters based on their similarity. Euclidean distance is used to calculate the similarity.[1]

2 Key Steps in K-Means[2]

2.1 Initialize centroids

The algorithms starts with initial estimates for the K centroids, which can either be randomly generated or randomly selected from the data set. We randomly pick K cluster centers(centroids). Let's assume these are $c_1, c_2, ..., c_K$ and we can say that:

$$C = \{c_1, c_2, ..., c_K\} \tag{1}$$

where C is the set of all centroids.

2.2 Assign Clusters

Each centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance. More formally, if c_i is the collection of centroids in set C, then each data point x is assigned to a cluster based on

$$\arg\min_{c_i \in C} dist(c_i, x)^2 \tag{2}$$

where $dist()$ is the standard (L2) Euclidean distance. Let the set of data point assignments for each ith cluster centroid be S_i. Note that the distance function in the cluster assignment step can be chosen specifically for different problems.

2.3 Re-compute the centroids

In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i \tag{3}$$

where S_i is the set of all points assigned to the ith cluster.

The algorithm iterates between steps two and three until a stopping criteria is met (i.e., no data points change clusters, the sum of the distances is minimized, or some maximum number of iterations is reached). The best number of clusters K leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function.

3 Test Case on Metagenomics Data

The test dataset used is a metagenomics dataset, containing composition of microbial communities from metagenomic shotgun sequencing data. Each variable is a microbial species, and the table data describes the relative abundance of species for 33 samples. Code and data for this section can also be found in the Github Repo.

3.1 Data Preprocessing

The original data has 103 features, which makes it hard to visualize and unsuitable for directly K-Means clustering, so PCA was applied to reduce the feature dimensions to 2. PCA plot is shown as Figure Figure 1.

3.2 Clustering with K-Means

Default k is 2, so it will cluster data into 2 groups. As shown in Figure 2, different groups are represented by different colors.

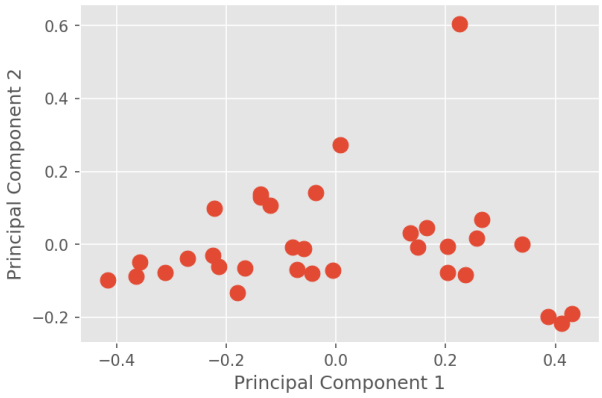

Figure 1. PCA Plot of original data

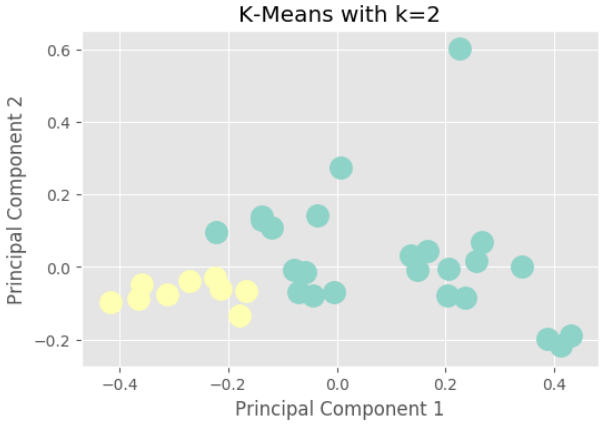

Figure 2. K-Means clustering with k=2

3.3 Choosing the Optimal Value of K

Determining the right number of clusters in a data set is important, because the appropriate number of clusters controls the proper level of cluster analysis. This can be done by iterating it through a number of k values and then finding the optimal k value. Plot loss values vs the k value and find the point where the graph is flattening, this point is considered as the optimal k value. The loss values is

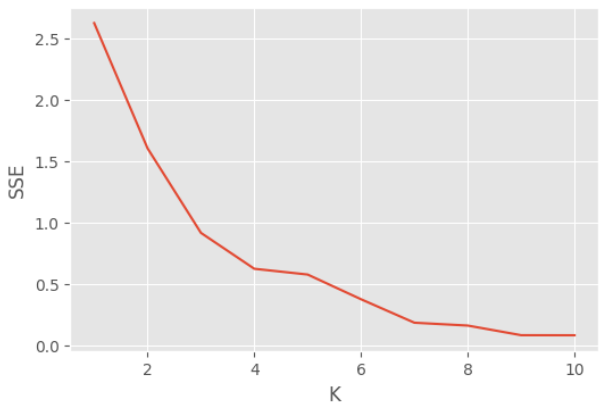

Figure 3. SSE vs K

defined as SSE(sum of squared errors):

$$SSE = \sum_{x_j \in S_i} (x_j - c_i)^2 \tag{4}$$

where S_i is the set of all points assigned to the ith cluster, c_i is the centroid of the ith cluster.

As shown in Figure 3, the graph starts to flatten after reaching 5, which means that even if we increase the no of clusters after that point, there is no significant change in the loss value. So we can take the optimal value to be 5 which can be confirmed by visualizing the scatter plot.

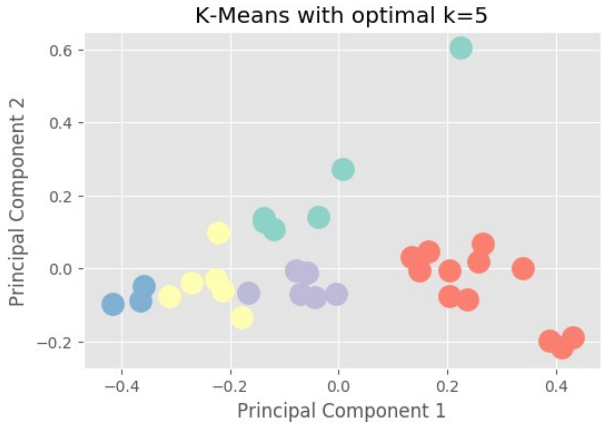

Figure 4. K-Means clustering with k=4

4 Discussion

Here I implemented the K-Means algorithm from scratch in Python and explored a metagenomics dataset with this algorithm. Although we could find some meanful results in the results, there are still some constraints of the algorithm. The standard K-means algorithm isn't directly applicable to categorical data and only numerical data can be used, for various reasons. The sample space for categorical data is discrete, and doesn't have a natural origin. A Euclidean distance function on such a space is not really meaningful. However, the clustering algorithm is free to choose any distance metric / similarity score. Euclidean is the most popular.

K-Means does not behave very well when the clusters have varying sizes, different densities, or non-spherical shapes. In that case, one can use Mixture models or EM algorithm. Because K-means assumes the variance of the distribution of each attribute (variable) is spherical; all variables have the same variance; the prior probability for all K clusters is the same, i.e., each cluster has roughly equal number of observations. If any one of these 3 assumptions are violated, then K-means will fail.

References

[1] V.A. D, *sklearn.cluster.kmeans*, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[2] V.A. D, *K-means clustering algorithm from scratch*, https://www.machinelearningplus.com/predictive-modeling/k-means-clustering/ April 26, 2020