Bisecting K-means on wine data set
Project Report

Tímea Halmai - s1103349 Ágoston Czobor - s1103351

2 January 2023

Abstract

Clustering is a common way to analyze data and sort it into groups by defined rules. The K-means clustering algorithm defines groups in a way that in the groups, the objects are similar, while different objects of different groups are very different. And while it is a good way to observe data, it lacks in some cases. It has problems recognizing clusters of different shapes, sizes, and densities. That's where the expansion of k-means clustering comes in. Bisecting k-means has less of a problem when different shapes or sizes exist. It can create clusters that are more complex in size or shape, it doesn't just divide the data set into k clusters, it divides specific clusters.

Mathematical description

In this chapter, we explain the workings of K-means and the Bisecting K-means variant. The summary, the mathematical description, and the computational complexity of the algorithms will be explained.

K-means

Summary

The K-means algorithm is a clustering algorithm. It takes a data set, and sorts the objects into k clusters. The sorting is done so that the objects end up in the cluster in which they are closer to the centroid. The algorithm will create similarly shaped and sized clusters.

Description

The K-means algorithm takes a k number, which determines how many clusters we want to have. The choosing of the centroid is an important step, because based on the position we choose, the end result can change. The optimal positions of the centroids are for them to be the furthest away from each other. In the first step however we choose them randomly, and then throughout the algorithm, we try to optimize their positions. The centroids are the central elements of the soon to be clusters. During the algorithm, they are the elements of which the other objects will get measured to, and then be grouped into clusters if they are close enough to them. There are more options to see how close objects are to the centroids. We can either check by city block, euclidean, seuclidean or cosine distance. They each change the end result a little bit, so we need to decide which distance is best for us.

Mathematical description

Assume we have $x_1, x_2, x_3 \ldots x_n$ elements and k as the desired cluster number. In the beginning we pick k random objects from x to be the centroids

- We find the distance measure of each object to the centroids
- We can use one of the four distance measures described earlier

  The euclidean distance is measured:
  $d(p,q) = \sqrt{((q_1 - p_1)^2 + (q_2 - p_2)^2}$
  where $(p = (p1, p2)$ and $q = (q1, q2))$

  The secludean distance is measured:
  $\sqrt{\sum (x-y)^2 / V}$

  The city block is measured:
  $\sum |x - y|$

  The cosine distance is:
  $S_C(A,B) = (\sum A_i B_i)/(\sqrt{\sum (A_i)^2}) * \sqrt{\sum (B_i)^2})$

- We assign all of the objects to the clusters they are closest to.

  $arg \min dist(c_i, x)^2$

  where the $dist$ is the chosen distance measure, the c_i are the centroids are the clusters, x are the data points

- Then we find new centroids, at the average of all clusters.

  $c_i = 1/|S_i| * \sum x_i$

  where c_i will be the new centroid of the cluster and Si are all the points assigned to the cluster

And then we just have to repeat the steps in the list until the centroids no longer change.

Bisecting K-means

Summary

K-means is an effective clustering method, but it cannot recognize differently shaped/sized clusters. That's when Bisecting K-means comes in handy. Bisecting K-means is a combination of partitional and hierarchical clustering. It can create differently shaped and sized clusters. It first creates two equal clusters, chooses centroids, and then pick the less similar cluster and bisects it into two.

Description

In the algorithm we first initialize the k centroids, and then we bisect using K-means with $k = 2$. After each bisection, we choose one of the clusters, and do the process again. We split the clusters for as long as we need more of them, so we can and will have different-sized and shaped clusters. To choose whichever cluster needs to be bisected again, we need a measure of dissimilarity between the two.

Mathematical description

First, we need to set the data as one cluster, then use $k = 2$ to bisect it, as we've described earlier in the K-means section. After that we have to measure the distance in each cluster:

$\sum(X_i - \overline{X})^2$

Then we select the cluster with the largest distance, meaning the cluster that doesn't quite belong together yet, and we split that with $k = 2$. Then we just have to repeat this process until we have enough clusters.

Computational complexity

In the K-means algorithm, the computation includes every object from the data set and k centroids. Which, with a bigger data set or with more clusters, will take more time and resources. For Bisecting K-means the computational time is reduced because it only needs one of the two clusters' objects and the two centroids. We will explore the complexity with the code explanation.

The goals of the algorithm

Bisecting K-means is a clustering algorithm that uses K-means and hierarchical clustering. The algorithm is needed when it comes to bigger datasets, trying to achieve more clusters because it handles that better than K-means. Bisecting K-means also comes in handy when we want to have differently sized or shaped clusters, since K-means produces clusters of similar shapes, while Bisecting K-means can and will create different ones. Bisecting K-means also handles outliers in the data better than K-means.

Related previous works

K-means and Bisecting K-means is a useful algorithms in many cases. Some real-life examples of their usage are

- Image segmentation
- Chromosome segmentation
- News comments clustering
- Clustering animals
- Grouping inventory by sales activity

Bisecting K-means is used when we wish to have more clusters in the end.

Future directons

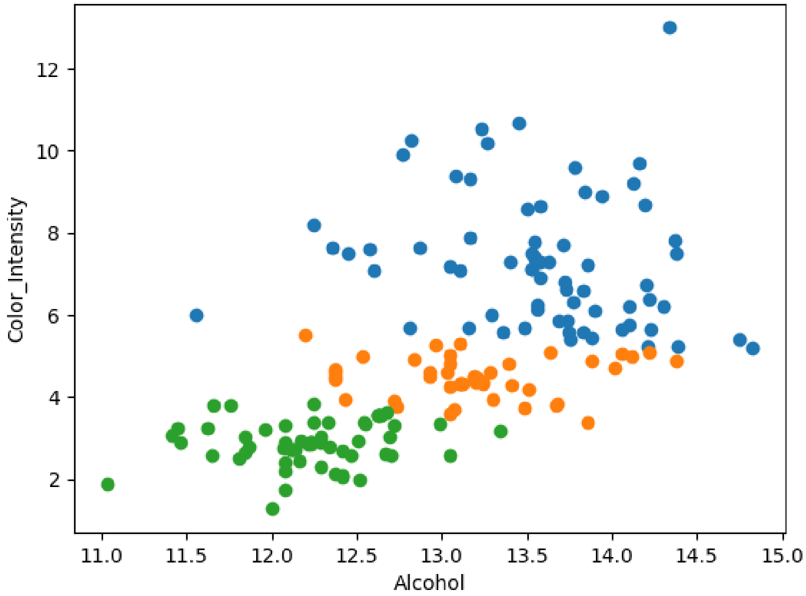
The determination of the centroids is a key part of the algorithm, as it changes the end result. A potential future change would be to determine centroids specifically. When we choose a centroid in the clusters, it also can potentially be an outlier, which would deter the end result. And secondly, if the K value is not selected properly, it could potentially cause a deviation between the end results and the ideal results. By using a specific centroid-choosing method, for example checking the density, we could avoid choosing outliers for centroids.

Real world illustration

We chose a wine data set[1] as our real world illustration. The set contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. In the program itself the data can be spliced according to what kind of clustering we want to achieve.

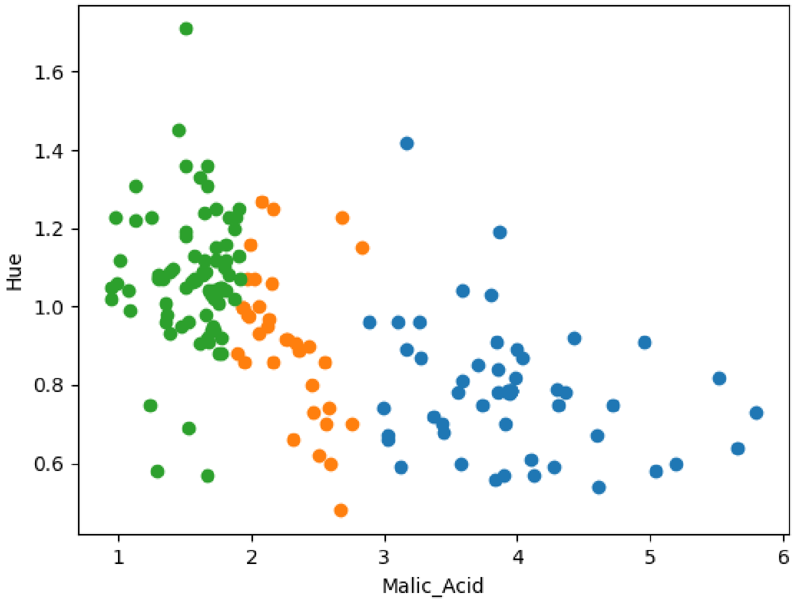
Some comparison examples:

0..1 Alcohol to Colour intensity



[1] https://www.kaggle.com/datasets/harrywang/wine-dataset-for-clustering

0..2 Alcohol to Colour intensity



Main result

With parameters $k = 3, epochs = 10, max_iter = 100$ the two algorithms ran almost equal times, they were within margin of error of 2%. With a high k value (= 12) the result got worse for the bisecting algorithm, the difference became obvious, with the normal K_Means algorithm working 30% faster on average. The situation did not change much with modifying the other values.

Analysis of result

Our report examined both the K-means algorithm by itself and Bisecting K-means, to see the effects of bisecting. Our data did not show the expected result. Bisecting can speed up the run time, but it's only beneficial when the data is bigger and we wish to have more clusters. Our data only needed 13 clusters, which wasn't enough to show the benefits of bisecting. On the other hand we saw improvements over the regular K-Means algorithm in the accuracy of the clusters. With some outliers in the data set we saw that the bisecting K-Means algorithm didn't get as skewed results because of them as the regular algorithm did. comparing the two sets.

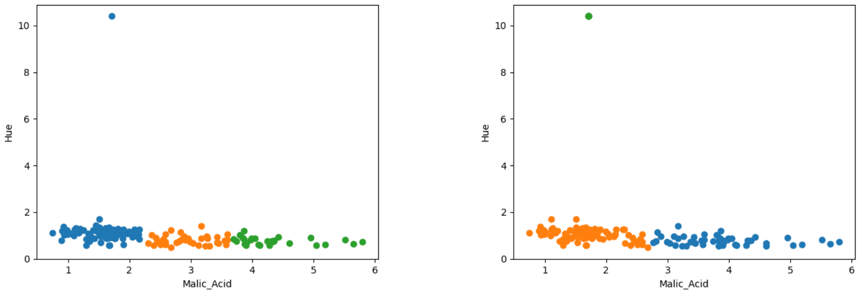


Figure 1: Regular and Bisecting K-Means Outlier test respectively

Insight to the code

We have our Bisecting K-Means algorithm in a script file called ”bisecting_kmeans.py” where there are 4 functions. Two basic helper functions and an implementation of a K-Means algorithm, and the implementation of our Bisecting K-Means algorithm with the use of our K-Means algorithm. We also have an accompanying ”main.py” script file which contains a main method to run the program, and a helping method to visualize the data.

convert_to_array

This method creates a 2D matrix represented by a numpy array from a list of lists, which contains the desired data of each subject. Complexity is $O(n^2)$ where n is the size of the list.

sum_squared_error

This method calculates the sum squared error of a cluster using the Euclidean distance. Complexity is $O(2n)$ where n is the size of the cluster.

kmeans

This method is a basic implementation of a kmeans algorithm. We are randomly selecting starting centroids, then doing cluster assignment, centroid update, sum squared error calculation for a set amount of times (max_iter), repeating this procedure *epochs* number of times, selecting the best result from these, and returning with a structure consisting of a list consisting of k number of arrays containing the values for the clusters. Complexity is $O(max\_iter * epochs * k * n)$ where n is the length of the data

bisecting_kmeans

This method is our implementation of the Bisecting K-Means algorithm. It uses the sum_squared_error method to calculate the point where the bisecting happens, and repeats the process while there are less clusters than desired. Complexity strongly depends on the size of k, as it impacts the size of the structure which need to be calculated at once, but worst case is $O(max\_iter * epochs * k * n)$ where n is the length of the data.