Jacob Huss
Dr. ChengXiang Zhai
CS 410: Text Information Systems
14 November 2020

Technology Review: Overview and Comparison of K-Means, Kernel K-Means, K-Medians, and K-Medoids Approaches to Clustering Text Data

In this technology review, I will be comparing four related and well-known algorithms for clustering text data: K-Means, Kernel K-Means, K-Medians, and K-Medoids. I will begin by giving an overview of the goal of clustering text data. I will then explain how the K-Means algorithm goes about accomplishing this. Afterwards, I will delve into each of the other three algorithms. For each algorithm, I will explain how it differs from K-Means and describe the benefits and detriments of using it. Finally, I will give an example of a recent paper where the author used both K-Means and another variation on K-Means to cluster emails into various categories of spam.

K-Means, Kernel K-Means, K-Medians, and K-Medoids are all clustering algorithms. The goal of each of these algorithms is to create K clusters, with each cluster containing similar, related data points. In terms of text mining, each data point represents one term or document, represented by a vector of values. The goal of clustering is not to group data points under pre-defined labels determined by the user. Rather, it is often used for initial exploration of a data set. In these cases, the goal is to reveal interesting structures and patterns within the data that would have otherwise gone unnoticed. However, in order to let any of the above-mentioned algorithms find groups of similar data points, there must be a way of defining how similar two data points are. Thus, all of the algorithms will be able to use a user-defined similarity function to determine the 'distance' between any two data points according to the values in their vectors. Shorter distances between data points indicate that the data points are more similar (Kumar).

The first algorithm I must explain is the K-Means algorithm. This algorithm starts by generating K random vectors. Each of these vectors will be considered the 'centroid' for one cluster. Then, all of the data points are assigned to the cluster whose centroid they are closest to. After all data points are assigned to a cluster, each cluster's centroid vector is recalculated. In K-Means, each element of a cluster's recalculated centroid vector is set to be the mean of all the values of that element in the cluster. Then, the step of assigning data points to their closest centroid and the step of recalculating centroids is repeated. This repetition continues until the sum of the square distances between points in the same cluster converges to a local minimum (Kumar).

While the K-Means algorithm is fast and generally works well for clustering, it comes with some disadvantages. Its biggest weakness is that it is not robust against outliers (Whelan et al.). For example, imagine that there is an outlying data point that is far from any other data point or initially-chosen centroid. This outlier must be added to the cluster of whichever centroid is closest, even if that centroid is quite far away. Then, because means can be drastically affected by a single outlier, the newly-calculated centroid would be pulled away from the majority of its data points.

The K-Medians algorithm makes one change to the K-Means algorithm to try to solve the outlier problem. The changed step is the method by which centroids are recalculated. In K-Means, the mean of each element within the cluster is used to create the new centroid. K-Medians instead uses the median of each element within the cluster. This algorithm is much more robust against outliers because the median calculation itself is more robust against outliers. However, in data sets that do not have outliers, this algorithm may produce clusters that are somewhat worse than those produced by K-Means (Whelan et al.).

The K-Medoids algorithm is also quite similar to the K-Means algorithm. However, instead of initializing centroids by generating K completely random vectors, K-Medoids chooses the K initial centroids by randomly selecting data points from the data set. In other words, the initial centroids will always be data points directly included in the data set. The other difference between K-Medoids and K-Means is in how they choose new centroids in each iteration of the algorithm. In K-Medoids, the recalculated centroid of a cluster is the data point in the cluster that minimizes a loss function, such as the sum of square errors. One of the main advantages of this algorithm is that centroids are always one of the data points from the data set, making the results of the cluster more interpretable (Kumar). For example, one might use K-Medoids to cluster terms and find a cluster that includes the terms 'Baseball', 'Glove', 'Bat', 'Football', 'Field', 'Sports', 'Basketball', and 'Hoop'. If the data point for the word 'Sports' is the centroid of this cluster, the commonality between the terms and why they are clustered together becomes immediately obvious to a user. Another advantage of this algorithm is that, like K-Medians, K-Medoids is robust against outliers. However, an important disadvantage of this algorithm is that its processing cost grows quickly with the size of the data set. This is because for every iteration, every data point in every cluster has to calculate what the loss function would be if it were the centroid. Formally, we can say that K-Medoids has complexity $O(K*(n-K)^2)$, where K is the number of clusters and n is the total number of data points in the data set ("ML | K-Medoids Clustering with Solved Example."). This complexity means that while K-Medoids can work well on small data sets, it can become much too slow on larger data sets.

Kernel K-Means looks to solve a problem shared by the K-Means, K-Medians, and K-Medoids algorithms. All three of the algorithms are only capable of detecting convex clusters, even though the most valuable clusters for some datasets are non-convex. For example, Figure 1 shows a set of data points that intuitively belong in the two non-convex clusters shown in green and red. While these clusters can't be produced by any of the previously-discussed algorithms, they can be produced by the Kernel K-Means algorithm. Kernel K-Means differs from K-Means in one way. Before calculating distances between data points and centroids, Kernel K-Means uses a kernel function to map each point into a higher-dimensional space. Various kernel functions can be used for this, including the polynomial kernel and the Gaussian radial basis

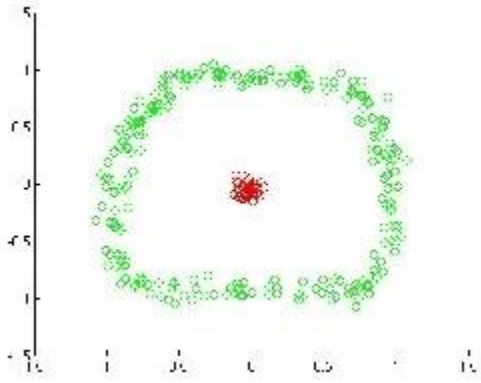

Figure 1 (from https://sites.google.com/site/dataclusteringalgorithms/kernel-k-means-clustering-algorithm)

function kernel. Each kernel maps the data points into an n by n kernel matrix. From this higher-dimensional space, non-convex clusters can be formed. The downside to this algorithm is that it takes much more processing time and memory to map into and make calculations from this higher dimensional space ("Data Clustering Algorithms - Kernel k-Means Clustering Algorithm.").

Finally, I would like to briefly explore the paper "An Effective Algorithm for Spam Filtering and Cluster Formation" by Kavitha Guda. In this paper, Guda is able to successfully use K-Means for a very practical purpose, clustering emails into various categories of spam. However, while the author was able to successfully create these email clusters using the normal version of K-Means, they also tried running a modified version of K-Means on the same dataset. The goal of this modified version was to minimize the algorithm's calculation time by avoiding recalculating the distance between data points and cluster centroids when possible. The modified algorithm did this by keeping two additional arrays of data containing each cluster's locations and smallest distances. The author found that this approach was successful in that it was able to create very similar clusters to the original K-Means algorithm in a much shorter execution time (Guda).

This paper has been a brief overview of the K-Means, Kernel K-Means, K-Medians, and K-Medoids algorithms and the benefits that each has. K-Means is a relatively fast algorithm that is good for creating convex clusters when there aren't any outliers in the data set. K-Medians is very similar to K-Means, but uses median values for centroid recalculation. This change makes K-Medians much more robust against outliers, but can produce results that aren't as good as K-Means when no outliers are present. K-Medoids differs from K-Means in that all centroids are points in the data set. This allows the algorithm to be more robust against outliers and improves the interpretability of the results. However, the processing time of this algorithm increases quickly with the size of the data set, making it less ideal for large data sets. Next, we explored Kernel K-Means, which involves mapping data points to higher dimensions. This allows the algorithm to create non-convex clusters, but comes at a cost of memory and processing time. Finally, we looked at a real-world application of using K-Means, as well as a unique variation on K-Means, to cluster emails together. Overall, K-Means is a very useful algorithm that has many different applications and variations. Each variation has benefits and detriments that are uniquely suited for different situations.

Works Cited

"Data Clustering Algorithms - Kernel k-Means Clustering Algorithm." *Sites.Google.com*, sites.google.com/site/dataclusteringalgorithms/kernel-k-means-clustering-algorithm. Accessed 16 Nov. 2020.

Guda, Kavitha. "An Effective Algorithm for Spam Filtering and Cluster Formation." *International Journal of Computer EngineeringIn Research Trends*, vol. 3, no. 12, Dec. 2016, pp. 659–666. *IJCERT Database*, ijcert.org/ems/ijcert_papers/V3I1210.pdf. Accessed 15 Nov. 2020.

Kumar, Satyam. "Understanding K-Means, K-Means++ and, K-Medoids Clustering Algorithms." *Medium*, 25 Aug. 2020, towardsdatascience.com/understanding-k-means-k-means-and-k-medoids-clustering-algorithms-ad9c9fbf47ca. Accessed 16 Nov. 2020.

"ML | K-Medoids Clustering with Solved Example." *GeeksforGeeks*, GeeksforGeeks, 8 July 2020, www.geeksforgeeks.org/ml-k-medoids-clustering-with-example/. Accessed 15 Nov. 2020.

Whelan, Christopher, et al. "Understanding the K-Medians Problem." *Int'l Conf. Scientific Computing*, 2015. Accessed 15 Nov. 2020.