Isaiah Shaefer
Data Mining Foundations

K-means++ Optimal Clustering Initialization Algorithm

Executive Summary

After using the K-means Clustering algorithm several times, analysts noticed some inconsistencies and decided to search for a better, more consistent, clustering algorithm. It was decided not to find an entirely different algorithm, but to instead find a method that improves it. So, the K-means++ initialization algorithm was selected to be used and compared to the results of the regular random initialization of K-means clustering. Using three datasets, the Cluster Dataset (Aleshunas, 2013), the Iris Dataset (Fisher, 1936), and the Wine Dataset (Forina, 1988), each of those initialization methods for K-means would be run against each other to see whether or not one of them gives better results. In the end, it was decided that the K-means++ Clustering method should always be used instead of just the standard methods of K-means because it always gave equal or better results.

Problem Description

When viewing the K-Means Clustering algorithm, analysts noticed some inconsistencies in the results. The natural clusters could change greatly because of the random initialization the method uses. It was believed that a better and more consistent clustering method already existed. Therefore, it was decided that an improved technique should be found to give more consistent and, hopefully, better results. K-Means++ Clustering was the method selected to improve on the original algorithm.

Analysis Technique

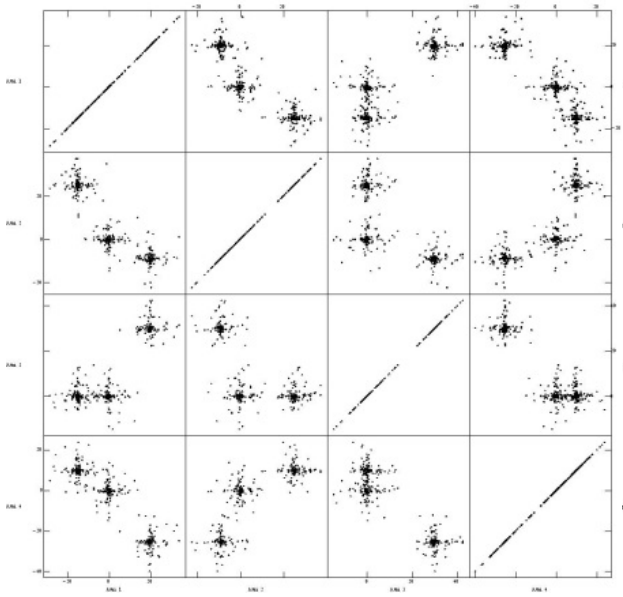
The K-Means Clustering Algorithm is a well-known naïve clustering method. The algorithm only requires one parameter, K, which designates the number of clusters to create from a dataset. Using that number, an equivalent number of arbitrary instances in the dataset are selected to become the starting point, or centroids, for each cluster. Then, each data instance in the dataset is systematically scanned and the Euclidean distance between them and each of the centroids is calculated. Each instance is assigned to the cluster whose centroid it is closest to. Once this process is complete, the mean of each of the resulting clusters is calculated and used as a new centroid and the entire dataset is scanned again using the same method to recreate clusters. This process is repeated until the clusters stop changing. Finally, the clustered results are displayed (MacKay, 2003).

The K-Means++ Clustering algorithm shares many of the main steps as K-means, but has a notably different method of selecting centroids. Just like K-means, the algorithm is given a parameter K for the number of clusters; however, only the first of those centroids is arbitrarily selects. Once the initial is found, K-mean++ search for another instance that is beyond a certain distance of the centroid, based on a weighted probability distribution, found within the dataset, and the desired number of clusters. Once an instance meeting the minimum distance requirement is found, it is selected as the second centroid. From then on, when searching for candidate centroids, the qualification is exactly the same, except the minimum separation must be met in relation to all of the existing centroids (Arthur & Vassilvitskii, 2007). When all of the centroids have been assigned, the algorithm functions identically to K-means. It still scans each remaining instance and assigns it to the cluster with the nearest centroid. It

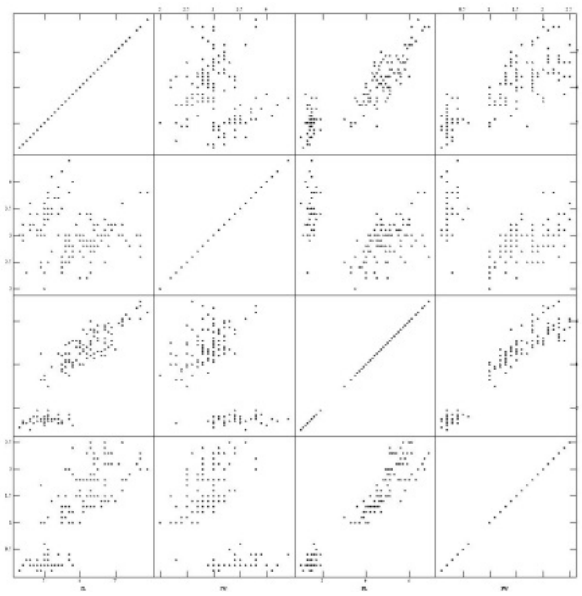
also still calculates the means of the resulting clusters and repeats the process until there is no change in the clusters. Now, it is important to note that K-means++ is actually defined as an initialization algorithm, not a clustering algorithm, because the initialization is the only difference between it and K-means (Alsabti, Ranka, & Singh, 1997).

In order to compare these two clustering methods, three datasets will be used. The first dataset is a modified version of the Cluster Dataset (Aleshunas, 2013). The original set contains 500 data instances with 4 unlabeled attributes and they are broken into 4 classes; however, in order to make it match the other two datasets, all of the instances labeled as class 4 were removed from the set, which resulted in 391 data instances instead (Shaefer, 2013). The second dataset is the Iris Dataset. It contains 150 instances of iris flowers with 4 labeled attributes and 3 classes; nothing about it has been modified (Fisher, 1936). The third dataset is the Wine dataset. It contains 153 instances of wine with 13 attributes and it is also broken into 3 classes, just like the other sets (Forina, 1988). These datasets were selected because they each have a varying degree of confusion regions, which allows the experiment to explore how the methods react in different situations. One way to understand these is to create pairwise scatterplots, which graphically compare two attributes of the dataset at a time. It gives an easy to understand visual representation of how all of the attributes contribute to class separation. Furthermore, respectively, the Cluster Set, Iris Set, and Wine set have a decreasing clarity that can easily be witnessed in the following pairwise scatterplots. Even with four attributes, the Cluster Set's natural clusters can easily be seen. The Iris Set's clusters become a little less clear, but they can still roughly be made out simply be looking at the scatterplots. But, the Wine Set has little visual distinction especially when collectively comparing all 13 of its attributes. They were also selected because they had the same number of classes, which made them more appropriate, and simple, for comparing results.

Cluster Dataset

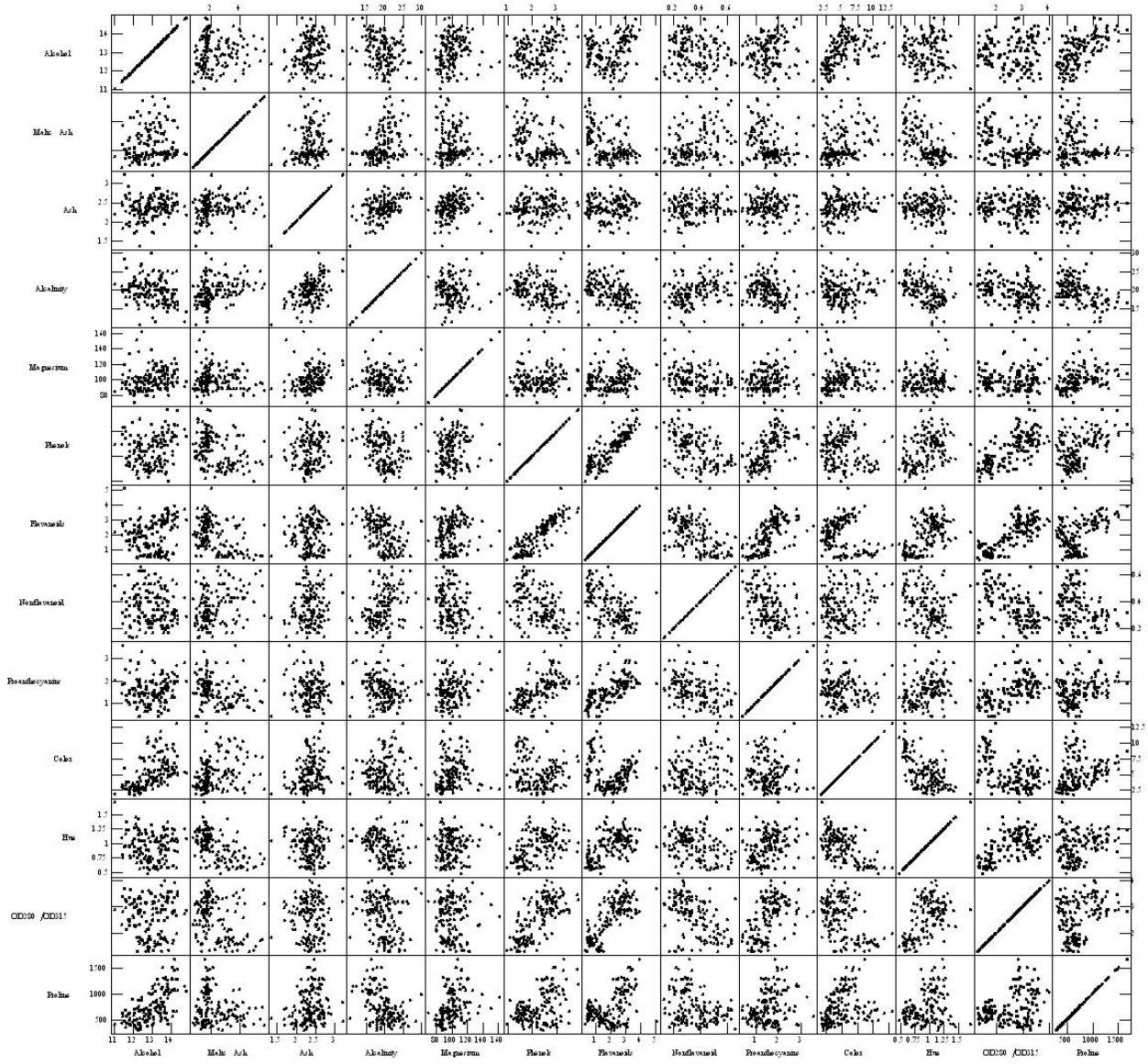


Iris Dataset



Isaiah Shaefer
Data Mining Foundations

Wine Dataset



Before the actual experiment is explained, it is important to understand some of the key components of the software being used for the clustering techniques. First of all, only one program will be used for both methods because the analysts were unable to find or create a separate program for the K-means++ method. Therefore, the K-means++ algorithm is only being simulated by attempting to follow the rules of the algorithm to produce the same selection method as a program would use. More importantly, the program being used simulates arbitrary selection of centroids simply by selecting the first instances found in each dataset until K amount of them have been chosen. This aspect of the software influences the experiment design, which will soon be explained. It allowed the analysts to engineer a terrible “arbitrary” selection and an “optimal” one to show how the results of the centroid selection affect the results, if at all.

In order to compare the K-means and K-means++ algorithms, the analysts developed an

experiment. The two clustering methods will be run against the same datasets using comparing the number of incorrectly clustered instances to discover if either method produces better results. The cluster quality is determined by first discovering which class the majority of instances in the cluster belong to. Then, any instances found in that cluster that do not belong to the majority class are labeled is incorrectly placed. The experiment will be run against the three dataset previously outlined, the Cluster Dataset, the Iris Dataset, and the Wine Dataset, in order to thoroughly test the results in different situations. Each of the listed datasets contains three known classes to help verify the quality of clusters found. Two versions of each dataset will be used: the "worst selection" where the first seven instances are all from the same class and another, the "optimal selection," where the first seven instances are a repeating sequence of the three classes. Furthermore, each clustering technique will be run against each dataset three times, per version, using a different number of clusters in each run. In each respective trial, K-means and K-means++ will create three, five, and seven clusters from the given datasets. When the trials are completed using both variations of each dataset, with each different number of clusters, it will total 36 trials, 18 with K-means and 18 with K-means++.

This experiment is designed to provide a clear understanding of how the subtle differences between each method can give significantly different results. At the beginning it was hypothesized that the results would be as follows:

- K-means Variation Two will be better than those of K-means Variation One
- K-means++ Variation One will be better than those of K-means Variation One and Two
- K-means++ Variation Two will be better than those of K-means Variation One and Two.
- K-means++ Variation One will be approximately equal to those of K-means++ Variation Two

Assumptions

The experiment explained was completed using the following assumptions:

- The classes in the datasets were correctly labeled and were related to the attributes given.
- K-means clustering is an appropriate algorithm for the given datasets.
- The software being used properly implements the K-means clustering algorithm.

Results

The results very clearly favored one algorithm over the other. This was not apparent in the Cluster Set because its confusion regions were nearly nonexistent. In fact, in every single variation, and different cluster size, the cluster set resulted in only one or less misplaced data instance, as seen in the Instance Misplacement Tables below, which just implies that algorithms work very well when the groupings of instances have little to no overlap. The Iris Dataset was the first to reveal the notable differences between the two algorithms' effectiveness. When three clusters were made, both algorithms gave about the same results in both the worst and optimal variations. When five clusters were made, the K-means had about 25% more incorrectly clustered instances they K-means++ in both the worst and optimal cases. When seven clusters were created, K-means++ blew K-means out of the water. In the optimal case K-means had three times more incorrectly placed instances then K-means++

and in the worst case it was closer to four times more than K-means++. The Wine Dataset still favored K-means++, but it wasn't quite is striking. In the three and seven cluster worst and optimal variations, K-means++ was only better by an instance or two. In the case of five clusters, though, K-means had about 50% more incorrect cluster placements in the optimal variation and about 70% more in the worst variation. As expected, after analyzing all of the results. The optimal variation gave better results than the worst variation when using regular K-means, but those variations made no noteworthy difference in the K-mean++ results. Additionally, as was just thoroughly explained, the k-means++ algorithm was proven to give noticeably better results those both variations of regular K-means. Therefore, the analysts concluded that K-means++ should always be used instead of K-means because it always provides clusters of equal or better quality.

Clustering – Instance Misplacement Tables

K-means

Cluster Dataset

	Optimal	Worst
3 Clusters	1	1
5 Clusters	0	1
7 Clusters	1	1

Iris Dataset

	Optimal	Worst
3 Clusters	17	18
5 Clusters	23	24
7 Clusters	10	17

Wine Dataset

	Optimal	Worst
3 Clusters	46	53
5 Clusters	39	44
7 Clusters	42	43

K-means++

Cluster Dataset

	Optimal	Worst
3 Clusters	1	1
5 Clusters	1	1
7 Clusters	1	1

Iris Dataset

	Optimal	Worst
3 Clusters	17	17
5 Clusters	19	19
7 Clusters	3	4

Wine Dataset

	Optimal	Worst
3 Clusters	45	45
5 Clusters	25	26
7 Clusters	42	42

Issues

The only issue that the analysts commented on was the lack of K-means++ clustering software. It is believed that much clearer results favoring K-means++ could have been revealed if software had be found or created and used to use the algorithm rather than trying to simulate it personally. The inconsistencies seen in some of the K-means++ algorithm are believed to be caused by human error in calculation, not flaws in the K-means++ initialization algorithm itself.

Appendices

Another large factor in comparing the quality of these algorithms is the time complexity. Like all other algorithms, the less iterations and or time it requires, the better it is because algorithms are meant to be efficient. K-means Clustering has a slow time complexity of $O(n^{dk+1} \log n)$, where k is the number of clusters to create, d is the number of attributes, and n is the number of number of data instances. That time complexity applies every time the algorithm repeats, which means it is more accurately $O(r(n^{dk+1} \log n))$, where r is the number of cycles of the algorithm it takes before the clusters stop changing. This is where K-means++ would step in to improve things. While it does not change the core complexity of the algorithm, it will minimize the value of r, which is a notable difference since each r doubles the original time (Inaba, Katoh, & Imai, 1994). Unfortunately, the analysts were unable to prove this in my experiment due to the limitation of time and resources, but it is an important component that should be known.

Works Cited

Aleshunas, J. (2013). Cluster Set.

Alsabti, K., Ranka, S., & Singh, V. (1997). *An effcient k-means clustering algorithm.*

Arthur, D., & Vassilvitskii, S. (2007). *K-means++: the advantages of careful seeding.* Philadelphia: Society for Industrial and Applied Mathematics Philadelphia.

Fisher, R. A. (1936). Iris Flower Data Set.

Forina, M. (1988). Wine Recognition Data. *PARVUS: An extendable package of programs for data exploration, classification and correlation*. Genoa, Italy: Institute of Pharmaceutical and Food Analysis and Technologies.

Inaba, M., Katoh, N., & Imai, H. (1994). Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. *SCG '94 Proceedings of the tenth annual symposium on Computational geometry* (pp. 332-339). New York: ACM.

MacKay, D. (2003). An Example Inference Task: Clustering. In D. MacKay, *Information Theory, Inference and Learning Algorithms* (pp. 284-292). Cambridge University Press.

Shaefer, I. (2013). Cluster Set Modified.