**An Analysis of Initialization Methods for Lloyd’s K-Means Clustering Algorithm Using Unsupervised Learning of Wine Classes**

Julia Fairbank and Caroline Cutter

Department of Computer Science, Middlebury College

May 23, 2022

CSCI 311: Artificial Intelligence

Professor Ananya Das Christman

**ABSTRACT**

Clustering is an unsupervised machine learning technique that aims to classify unlabeled data into groups using its attributes. One of the most common clustering algorithms used is K-Means clustering with Lloyd’s algorithm. The advantages of Lloyd’s algorithm include its speed, easy implementation, and accuracy with a known K (number of classification groups) value. However, Lloyd’s algorithm is highly sensitive to the outliers’ initial centers chosen in the algorithm because of the possibility of the centers getting trapped in a local minimum. However, many different initialization techniques have been developed that can improve the accuracy and efficiency of Lloyd’s algorithm. In this project, we aim to run a four of these initialization techniques, MacQueen, Forgy, K-Means++ and Naïve Sharding, on a new set of data and use these results to develop an algorithm, the Cutter-Fairbank, to produce the most accurate results in the most efficient time. The dataset we will use is a result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars, which we will treat as the different clusters (K = 3). We coded these algorithms using Python and various libraries to help standardize and compute the data. We collected accuracy, iterations, and time (seconds) of each initialization method over 100 runs. The results showed that Naïve Sharding produced the most accurate results, while the Cutter-Fairbank method produces the most efficient results. Although no definite conclusions can be drawn, these results help recommend the Naïve Sharding if accuracy needed and the Cutter-Fairbank method if efficiency is needed for a dataset with relatively small attributes and clusters. This research could be extended by testing the Cutter-Fairbank method on more extensive datasets to see how the accuracy and efficiency changes in terms of number of clusters, number of attributes and number of instances.

**INTRODUCTION**

Clustering is a technique that is used to classify data using unsupervised learning technique. There are many forms of clustering algorithms, one of the most well-known is K-Means clustering with Lloyd’s Algorithm. The advantages to Lloyd’s algorithm are that it is easy to implement, it can be easily modified for enhancements, it is efficient with a linear time and space complexity and that it is guaranteed to converge. However, this algorithm also suffers from disadvantages such as a high sensitivity to outliers, some knowledge of K is required for the clusters, and that convergence can occur on local minimum, so the output is dependent on the initial centers chose (Celebi, 2013). Different initialization methods for choosing the starting centers in Lloyd’s algorithm have been proposed to reduce the likelihood of landing on a local minimum and to reduce sensitivity of outliers in the data. The problem that we are addressing in this project is optimizing Lloyd’s algorithm using different initialization methods due to its high sensitivity to initial centers. The motivation behind this problem is that K-Means clustering is one of the primary machine learning techniques used to classify objects when prior information about the data is not available and it is important to have an efficient algorithm to obtain accurate results (Jain, 1999). In this project, we attempt to improve this issue by classifying wines based on their chemical composition into three different origins using five different initialization methods of Lloyds algorithm. The aims of this project are to 1) Apply previously described initialization methods on a new dataset, 2) Analyze and compare the efficiency and accuracy of different initialization methods of Lloyds algorithm and 3) Propose a new initialization method to optimize the classifications. We will use the data of accuracy and efficiency on initialization methods that have already been described to form a new initialization method to optimize the results.

**RELATED WORK**

Several studies have been conducted that test the efficiency of various initialization methods of Lloyd’s algorithm such as Peña et al (1999), He et al (2004) and Celebi et al (2013). Peña et al compared four different methods of initialization, including MacQueen, Random, Forgy and the Kaufman approach. This paper focuses on convergence speed of these algorithms on three different commonly used datasets including the Iris database, the Ruspini database, and the Glass database. What the researchers concluded was that the random and Kaufman approaches had a higher accuracy and efficiency compared to the MacQueen and Forgy approaches. The study done by Celebi et al (2004) was very similar, however it was much more extensive and tested 8 different initialization methods on 32 different datasets. To test these methods, the coded these methods in C, ran the data and collected information about initial and final SSE, Normalized Rand, number of iterations and CPU time of computation. The main conclusions from their results were that 1) non-deterministic (i.e. randomized) algorithms performed better than deterministic ones, 2) the Bradly Fayyad method of initialization performed the best and 3) that the K-Means++ and MacQueen’s method performed generally well. The shortcomings of these experiments is that different researchers often obtain different results based on the data they are looking at.

Our project differs from these studies because we 1) use four previously described initialization methods on a new data 2) use this data to analyze which techniques work best to develop a new initialization algorithm and 3) test this newly developed initialization algorithm to compare its efficiency and accuracy from the previously implemented algorithms. We also aim to explore a newer method of initialization called Naïve Sharding which has not been heavily tested in published experiments.

**PROBLEM STATEMENT**

*The Problem.* Lloyd’s algorithm for K-Means clustering has many important applications in the world of machine learning, such as data mining, image recognition, fraud detection and recommendation systems. However, three of the main disadvantages to Lloyd’s algorithm is that some knowledge of K (number of clusters) is required, that it is easily changed by outliers, and that it is highly sensitive to center initialization due to local minima. Due to Lloyd’s practicality in the real-world, we aim to improve these issues by testing different center initialization methods on data with a known K value. Although this project will not examine the issue of choosing a K value, we do aim to study which initialization method will help reduce sensitivity to outliers and result in the best overall accuracy and time complexity with a given dataset.

*Lloyd’s Algorithm.* Lloyd’s algorithm is a K-Means clustering algorithm that takes in K (the number of clusters) and a vector of instances with M features. The algorithm starts by choosing K arbitrary centers, then each instance is assigned to the nearest center. The new centers become the means of the assigned clusters, and these two steps are repeated until convergence, resulting in no reassignment of instances to clusters. In this project we will be targeting Step 1, the center initialization step, with five different algorithms described below (Compeau, 2018). With standard initialization, Lloyd’s algorithm has a linear runtime of O(K\*D\*N) for a K clusters of N points with D attributes. The Lloyd’s algorithm and initialization method pseudocodes can be found in Appendix A below. These initialization techniques were chosen due to their popularity in the field of machine learning, feasibility to implement and base on their expected efficiency and accuracy.

*MacQueen’s Initialization.* This algorithm takes the first K points in the dataset as the centers. The advantage is that it is computationally efficient with a constant runtime, however a disadvantage is that it is sensitive to the data order, so the results will be the same for each run if the data is not shuffled (Celebi, 2013).

*Forgy Initialization.* Forgy is one of the most common initialization techniques used for Lloyds, which selects K random instances in the dataset as centers. The reasoning behind this algorithm is that points are likely to be selected from denser areas of points and that it is efficient with a constant runtime, however the major disadvantage is that there is no filtering for choosing outliers as centers and since the algorithm is random, there is a possibility of different results each time (convergence on local minimum can occur) (Thakur, 2020).

*K-Means++ Initialization.* This algorithm is a well-known optimization of Lloyd’s algorithm, which choses a random center, then chooses the next centers from a probability distribution that is proportional to the squared distance from the points to its closest center. The idea behind that a point having a large distance from its closest center is more likely to be chosen, so the centers will be distributed across the feature space. The advantage of this algorithm is that it is efficient with a logarithmic time complexity of O(log(k)) and improves accuracy over random initialization, however a major disadvantage is that it is more computationally complex than random initialization and that since it is randomized, it can produce different initial centers and results each time it is ran (Arthur, 2007).

*Naïve Sharding Initialization.* This initialization algorithm is relatively unknown however through recent studies, it has shown to reduce time complexity and improve accuracy over the previously mentioned methods. Naïve Sharding uses the calculation of summation values for all the attributes of an instance and sorts the values based off the summed values. Once sorted, the data is split into K “shards”, the attributes of the shards are summed, the mean is computed and then these means of the attributes become the initial centers (Mayo, 2017). Since this approach is not randomized, it will result in the same initial centers and clustering each time it is ran.

*Cutter-Fairbank Initialization.* After studying many methods of initialization and analyzing the results of the methods above on the wine classification data, we created an initialization technique that we expected to increase efficiency and accuracy of Lloyd’s. This algorithm is a combination of Naïve Sharding and Al-Doud’s variance-based model (Al-Doud, 2007). This algorithm takes the variance of each column of attributes like the Al-Doud method, then sorts the columns based off the attribute with the maximum variance. Once sorted, the data is split into K “shards” and the mean of each attribute of the shard becomes the new center, like the Naïve Sharding approach (Mayo, 2017). Although experimental, we believe this will perform best with this data set because it uses Naïve Sharding which is one of the most efficient algorithms, but implements it with variances, so that the initial centers will be closer to the denser clusters.

*Standardization and PCA Transformation.* In K-Means clustering, Euclidean distance is used to measure the distance between points in the feature space. If the attributes are measured in different units, it is recommended to standardize the data so attributes with larger ranges don’t dominate the distance calculations (Celebi, 2013). In this project, we use a variance-based z-score standardization of the data. With high-dimensional (more attribute) K-Means clustering, it is common to transform the data use Principal Component Analysis to reduce the dimensions and speed up analysis. Our project does not use high dimensional data but will use Principal Component Analysis (PCA) to transform the resulting centers to a 2D space for data visualization. Since this is performed after the methods are run, it will not affect the results (Galarnyk, 2022).

**METHODS**

*Data Set.* The dataset is a result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars, which we will treat as the different classes. The dataset was acquired from the UCI Machine Learning Repository (Wine, 1991). There are 178 instances (59 class 1, 71 class 2, 48 class 3) with 13 decimal attributes including malic acid, ash, alkalinity of ash, magnesium, total phenols, flavonoids, nonflavonoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines and proline. The dataset was read as a CSV file using Python’s *csv* library. This dataset was chosen due to its relatively small number of attributes and because it has a set number of classification groups (K=3).

*Standardization.* The attributes of the dataset were first standardized using unit variance z-scores with the Python *sklearn.StandardScalar()* function. The data was standardized before running the Lloyd’s and the Initialization methods to improve accuracy and reduce the runtime of the algorithm (Galarnyk, 2022). Figure 2 in Appendix B show the results of running Lloyd’s on the unstandardized data, which was ~20% less accurate than when the data was standardized.

*Lloyd’s & Initialization Methods.* Lloyd’s algorithm and the five initialization methods were all implemented using Python language in the Thonny IDE. Pseudocode for each implementation can be found in Appendix A below. Many of these implementations relied on Python’s *NumPy* library for matrix operations and representing arrays of attributes. The Euclidean distance between points in M-dimensional space was measured using Python’s *scipy.spatial.distance.euclidean()* function.

*Performance Criteria.* To analyze the efficiency of the initialization methods, we measured the number of iterations that the algorithm required until it reached convergence.

​

We also measured efficiency by recording the runtime from the start of the initialization process to when the centers of converged clusters are returned, measured in seconds using Python’s *Time* library. The accuracy of the algorithm was measured by computing the number of correctly classified wines divided by the total wines. The number of correctly classified wines was calculated by pairwise comparison of the original data’s wine group versus the Lloyd’s classification of the wine groups. The data passed into Lloyd’s was stripped of the labels before computation to avoid the label being treated as an attribute. The Lloyd’s labeled data’s groupings were set to the majority label of the original groups to keep a cohesive “Group” label and to be able to accurately compute pairwise comparisons between the two datasets.

*Data Collection and Graphing.* The program was run for 100 times for each of the five initialization methods. For each run, the initialization method used, number of iterations, the accuracy of classifications (in percentage) and the time (in seconds) was recorded in a CSV datasheet. We used R Studio to analyze the results by computing averages for each initialization technique, as well as creating bar and box plots to visualize the data. To visualize the clustering’s, we chose to use Python’s *matplotlib* library and used a principal component analysis (PCA) transformation after the clusters had been found to reduce the 13 attribute wines into a two-dimensional space. The tool used to perform the PCA transformation was the Python *sklearn.PCA()* function with Python’s *Panda* library. This procedure was learned and implemented from Galarnyk, 2022. Graphs were created to show the initial centers chosen for each initialization method, the original groups of wine, and the groups of wines computed by Lloyd’s algorithm. All figures and graphs can be seen in Appendix B below.

**RESULTS**

*I. Overall Clustering***.** Please refer to Appendix B Section I for the corresponding figures. Figure 1 shows the graph of the original cultivars of wine (Groups 1 – 3) using a PCA transform to graph the attributes in a 2D space. Figure 2 shows the clustering derived from Lloyd’s algorithm using the z-score standardization of wine attributes with a 97.8% accuracy of classification. Figure 3 shows the clustering derived from Lloyd’s algorithm using non-standardized standardization of wine attributes with a 70.2% accuracy of classification. These figures and accuracy results help us understand why standardizing the values for a given dataset can significantly improve accuracy, as well as let us visualize the data in a two-dimensional space.

*II. Initial Centers.*Please refer to Appendix B Section II for the corresponding figures. Figures 4 – 8 show the initial centers resulting from each of the initialization methods. The randomized algorithms (Forgy and K-Means++) show 5 different starting configurations for centers, while the non-randomized algorithms (MacQueen, Naïve Sharding, Cutter-Fairbank) show the starting configuration that happens each time the algorithm is ran for this dataset. These graphs help us visualize how these initialization methods are working to choose the centers, and why some may produce better results in a faster time. For example, MacQueen’s initialization chooses three centers in proximity, while Naïve Sharding chooses uniformly distributes centers that are far apart and MacQueen’s typically takes more iterations to converge.

**Table 1.** Average Accuracy, Iterations and Time Across Five Initialization Methods

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization Method** | **Average Accuracy (%)** | **Average Iterations** | **Average Time (Seconds)** |
| **MacQueen** | 96.61 | 8 | 0.0839 |
| **Forgy** | 95.747 | 7.41 | 0.0771 |
| **K-Means++** | 92.878 | 7.75 | 0.09 |
| **Naïve Sharding** | 97.18 | 6 | 0.748 |
| **Cutter-Fairbank** | 95.48 | 5 | 0.0657 |

*III. Accuracy***.** Please refer to Appendix B Section III for the corresponding figures. As shown in Table 1, the average accuracy from greatest to least for different initialization methods was Naïve Sharding, MacQueen, Forgy, Cutter-Fairbank and then K-Means++. From the boxplot in Figure 10, it is evident that Naïve Sharding always produced the maximum accuracy. It is important to note that although sometimes Forgy and K-Means ++ produced the maximum accuracy on some runs, they also had outliers in the ~60-70% accuracy range, so these methods are not as consistent as the non-randomized methods.

*IV. Iterations.*Please refer to Appendix B Section IV for the corresponding figures. As shown in Table 1, the average iterations from least (more efficient) to greatest (least efficient) for different initialization methods was Cutter-Fairbank, Naïve Sharding, Forgy, K-Means++, and then MacQueen. Although Cutter-Fairbank had the lowest average iteration value, according to Figure 12, both Forgy and K-Means++ had a few runs with the minimum iterations. However, they also both experienced the majority of runs with much higher-than-average iterations.

*V. Time.*Please refer to Appendix B Section V for the corresponding figures. As shown in Table 1, the average time from most efficient to least efficient for different initialization methods was Cutter-Fairbank, Naïve Sharding, Forgy, MacQueen and then K-Means++. According to the boxplot in Figure 14, MacQueen and Naïve Sharding experienced the most consistent runtime, however neither of these runtimes compared to the Cutter-Fairbank Method. It is also important to note that the Forgy and K-Means++ both experienced the fastest runtimes on a few of their runs, but the Cutter-Fairbank experience the highest average and was quite consistent.

It is important to note that although sometimes the randomized approaches produce the best results for some runs, they are not always consistent, so we look at averages across 100 runs. The Naïve Sharding and Cutter Fairbank approaches reduce the chances of “bad partitions” that can lead to less accurate and more time-consuming results.

**DISCUSSION**

*Summary of Results.* Although there is variability between the 100 runs, according to the data, Naïve Sharding produced the highest accuracy, and the Cutter-Fairbank method produced the highest efficiency in terms of run time in seconds and iterations. We originally hypothesized that the Cutter-Fairbank method would produce the highest accuracy and run the fastest, however the Cutter-Fairbank method on average was ~2% less accurate than the Naïve Sharding method. Although we cannot make any definitive conclusions, based on our results we would recommend the Naïve Sharding method for datasets with a relatively small K and number of attributes because it obtained the highest accuracy and second highest efficiency. If the problem required to be done quickly with some sacrifice to accuracy, we would recommend the Cutter-Fairbank Method. If the problem required the most accuracy without focus on the time of computation, the recommended method would be Naïve Sharding. The results from the K-Means++ were a bit surprising, because much of our research led us to believe that it would perform the best in accuracy and time, but it performed the worst out of these five different initialization methods for this dataset.

*Pitfalls.* One major weakness of this experiment is that these initialization methods were tested on only one dataset. Since K was known for the wines, we did not examine the effect of different K’s, number of attributes or number of instances on accuracy or efficiency. This could be improved by adding wine instances or more chemical attributions to each of the wines and compare the resulting efficiencies and accuracies to the current data. Due to our limited data used in this project, we can only recommend which algorithm to use for small attribute and K datasets, but further testing with more data could strengthen these recommendations. Another pitfall of this experiment is that the MacQueen initialization method produces the same results each time, even though it is sensitive to data ordering. One way to improve this result is by randomly shuffling the dataset each time it is run through the algorithm and average the results.

*Future work.* In the future, this study could be extended by trying multiple data sets with different sizes of K, attributes, and instances to compare how these may affect runtime and accuracy with these initialization methods. Also, further testing of the Cutter-Fairbank method could prove useful in unsupervised machine learning situations that need to classify data quickly, with some flexibility in accuracy such as anomaly detection situations such as fraud detection or predictive maintenance of machines.

**WEBSITE**

URL: <https://jfairbank1.wixsite.com/cutter-fairbank>

**REFERENCES**

Al-Daoud, Moth'd Belal. “A New Algorithm for Cluster Initialization.” *Zenodo*, 21 Apr. 2007, https://doi.org/10.5281/zenodo.1334075.

Arthur, David, et al. “K-Means++: The Advantages of Careful Seedings.” *ACM Conferences*, 1 Jan. 2007, https://dl.acm.org/doi/10.5555/1283383.1283494.

Celebi, M. Emre, et al. “A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm.” *Expert Systems with Applications*, vol. 40, no. 1, 2013, pp. 200–210., https://doi.org/10.1016/j.eswa.2012.07.021.

Compeau, Phillip, and Pavel Pevzner. *Bioinformatics Algorithms: An Active Learning Approach*. Active Learning Publishers, 2018

Galarnyk, Michael. “PCA Using Python (Scikit-Learn).” *Medium*, Towards Data Science, 27 Apr. 2022, https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60.

Jain, A. K., et al. “Data Clustering.” *ACM Computing Surveys*, vol. 31, no. 3, 1999, pp. 264–323., https://doi.org/10.1145/331499.331504.

He, Ji, et al. “Initialization of Cluster Refinement Algorithms: A Review and Comparative Study.” *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, 2004, https://doi.org/10.1109/ijcnn.2004.1379917.

Mayo, Matthew. “Toward Increased K-Means Clustering Efficiency with the Naive Sharding Centroid Initialization Method.” *KDnuggets*, 2017, https://www.kdnuggets.com/2017/03/naive-sharding-centroid-initialization-method.html.

Peña, J.M, et al. “An Empirical Comparison of Four Initialization Methods for the K-Means Algorithm.” *Pattern Recognition Letters*, vol. 20, no. 10, 1999, pp. 1027–1040., https://doi.org/10.1016/s0167-8655(99)00069-0.

Thakur, Nitish Kumar. “Comparison of Initialization Strategies for K-Means.” *Medium*, Analytics Vidhya, 19 Apr. 2020, https://medium.com/analytics-vidhya/comparison-of-initialization-strategies-for-k-means-d5ddd8b0350e.

Wine. (1991). UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/wine.

**APPENDIX A:**

**Pseudocodes for Algorithms**

1. **Lloyd’s Algorithm.** Each initialization method will be run as step 1 above. This algorithm returns K cluster centers as points.

Text

Description automatically generated

1. **MacQueen’s Initialization**

Text

Description automatically generated

1. **Forgy’s Initialization**

Text

Description automatically generated

1. **K-Means++ Initialization**

Text, letter

Description automatically generated

1. **Naïve Sharding Initialization**

Text

Description automatically generated

1. **Cutter-Fairbank Initialization**

Text

Description automatically generated

**APPENDIX B:**

**Figures and Graphs**

1. Chart, scatter chart

   Description automatically generated**Clustering Graphs**

Chart, scatter chart

Description automatically generated

**Figure 2.** *Lloyd’s Algorithm Classification of Wines using Standardized Values.* This plot shows the labeling of the most accurate groupings of Wine which was 97.8% correctly labeled.

**Figure 1.** *Original Groups of Wines.*

Chart, scatter chart

Description automatically generated

**Figure 3.** *Lloyd’s Algorithm Classification of Wines using Non-Standardized Values.* This plot shows the labeling of the most accurate groupings of wine using non-standardized data which was only ~70.2% accurate. Therefore, we use standardized data for our experiment.

1. **Center Initialization Graphs**

Chart, scatter chart

Description automatically generated

**Figure 4.** *MacQueen’s Method Initial Centers.* The red X’s mark the original centers chosen by MacQueen’s algorithm. Since it is not randomized, each run results in the same centers.

Scatter chart

Description automatically generated

**Figure 5.** *Forgy’s Method Initial Centers.* The red X’s mark the original centers chosen by Forgy’s algorithm over 5 different initializations. Since it is randomized, each run results in a different starting arrangement.

Chart, scatter chart

Description automatically generated

**Figure 5.** *K-Mean++ Method Initial Centers.* The red X’s mark the original centers chosen by the K-Mean++ algorithm over 5 different initializations. Since it is randomized, each run results in a different starting arrangement.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

**Figure 7.** *Naive Sharding Method Initial Centers.* The red X’s mark the original centers chosen by the Naïve Sharding algorithm. Since it is not randomized, each run results in the same centers.

**Chart, scatter chart

Description automatically generated**

**Figure 8.** *Cutter-Fairbank Method Initial Centers.* The red X’s mark the original centers chosen by the Cutter-Fairbank algorithm. Since it is not randomized, each run results in the same centers.

1. **Accuracy**

**Chart, bar chart

Description automatically generated**

**Figure 9.** *Bar Graph of Average Accuracy of Classification.* The average accuracy of each initialization method across 100 runs.

**Chart

Description automatically generated**

**Figure 10.** *Box Plot of Accuracy of Classification.* A box plot of accuracy of different initialization methods across 100 runs. Outliers are represented as empty circles.

1. **Iterations**

Chart, bar chart

Description automatically generated

**Figure 11.** *Bar Graph of Average Iterations of Classification.* The average iterations of each initialization method across 100 runs.

**Chart, box and whisker chart

Description automatically generated**

**Figure 12.** *Box Plot of Iterations of Classification.* A box plot of iterations of different initialization methods across 100 runs. Outliers are represented as empty circles.

1. **Time**

Chart, bar chart

Description automatically generated

**Figure 13.** *Bar Graph of Average Time of Classification.* The average time of each initialization method across 100 runs.

Chart, box and whisker chart

Description automatically generated

**Figure 14.** *Box Plot of Time of Classification.* A box plot of time of different initialization methods across 100 runs. Outliers are represented as empty circles.