IE 529 Project Report
K Means Seeding

Prerna Rathi
Rahul Mahesh
Shreekant Gokhale
Sushama Chamarajanagar Shankar

November 2021

1 Introduction

1.1 What is the problem?

The ability to effectively group together (cluster) data points that share similar features has been one of the most sought-after answers in optimization. The main objective is to find k clusters in a set of n data points to optimize a given cost criterion. The cost criterion is generally given by norm-based distance function.

The cost equation ϕ in consideration is computed as the objective criteria to minimize. The goal of k-means is to find k-centers that can minimize the cost function based on the distance parameter. Lloyd's algorithm is a heuristic solution proposed to solve the clustering problem. It is an iterative refinement technique. The centroids initialized are uniformly randomly selected points from the dataset. Pairwise comparison of the cost function through the iterations result in a local minimum; this acts as the solution. Clearly, Lloyd's algorithm is extremely sensitive to proper initialization [2] and can cause noticeably separate clusters to merge. Additionally, this method offers no guarantee on the accuracy [1] and convergence to a local minimum can give poor results.
The problem that is being addressed in this research paper pertains to increasing the accuracy and speed of the k-means Lloyd algorithm. A D^2-seeding technique is proposed, which is O(log k) competitive with respect to optimal clustering [1].

1.2 Why do we care about the problem?

Clustering proves useful in finding hidden patterns in data sets, which can be vastly applied in fields of mathematics, statistics, machine learning, etc. K-means clustering (Lloyd's algorithm) is the most frequently utilized approach to perform clustering. However, there is a trade-off between the simplicity of execution and accuracy. Maintaining a delicate balance in this case can significantly improve the efficiency. In addition, the existence of upper bound with respect to the running time and the cost is not certain in the K-means algorithm[3]. The randomly selected initial points highly affect the results of this algorithm. Also, the result could converge to a local minimum, which could be very far from the optimal result and it is significantly sensitive to outliers [4]. These concerns, paired with the search for optimal solution makes this problem highly interesting.

1.3 What are the main assumptions made to solve the problem?

1. Each attribute is of similar importance for each cluster, resulting in spherical variance of distribution.
2. A certain number of clusters are always assumed to exist for the dataset.
3. Typically, n $\gg$ k.
4. The centers of the clusters are always assumed to belong to the dataset.

2 Technical Background

It is important to define a few key concepts and functions to understand the k-means++ algorithm for a better clustering of data points with a given cost function.

2.1 Functions and Notations

Φ – a potential function indicating the squared 2-norm of distances from
datapoints $\in$ X to their nearest centroids.
X – set of all points from the dataset.
d – The dimensionality of points in X.
C – set of k-centers for the clustering problem.
D(x) – distance from a point x to its nearest centroid.
C_{OPT} – The optimal set of k-centroids (objective minimizing centres).
Φ_{OPT} – The optimal objective function value (smallest distance from all points to their closest centres).
A – an arbitrary cluster with a centroid in C_{OPT}.
We now introduce a set of essential functions that keep track of, and capture key quantities/observations necessary for us to form conclusions about the results of the algorithm.

$$\Phi = \sum_{x \in X} min_{c \in C} ||x - c||^2$$

The objective function capturing the squared total deviation of all points in the model about their nearest centres. Here distance is measures as a 2-norm of points from X and C which are d-dimensional. This expression can also be generalized for any subset S within X:

$$\Phi = \sum_{x \in X} min_{s \in S} ||x - s||^2$$

Since the k-means++ problem is an extension of the k-means clustering model, it is necessary to cover ideas from the latter first before diving into all concepts in the k-means++ approach. Both algorithms are similar with the exception of the initialization step: while the k-means model relies on an arbitrary selection of k-centres from X, the k-means++ algorithm relies on a structured heuristic approach while attempting to optimally select the k-centers for set C. This can be achieved through approaches such as the Greedy K-Centres model for selecting the most spread out set of k-centres or probabilistic modeling for selection of the k-centres as a function of D(x). Consequently, we can describe the following:

$$c_i = \frac{1}{|C_i|} \sum_{x \in C i} x$$

Whenever a set of points in X is assigned to the cluster Ci in the k-means portion of the k-means++ approach, we must recompute the centroid of the ith cluster of data with the formula used above. Note that this process must be repeated d-times for recomputing the mean of each dimension of this cluster.

$$D(x) = min_{c \in C} ||x - c||^2$$

The distance of a point x $\in$ X from its nearest center is evaluated as the minimum distance among a set of distances from x to all centers in C.

Furthermore, each point is given a weightage or score for selection as a candidate to enter C based on its probability of fulfilling this criterion. It can be computed as: $p(x) = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}$, the probability of choosing a point to be in the set C is related to its relative distance to nearest center among all data points. Generally, we would expect a higher p-value to imply that this distance should be of larger magnitude and consequently, is a better candidate for entering the C because we want all centers to be far apart (for a better clustering). This squared-distance weighting on points allows us to capture Euclidean distances of higher dimensions as well, but other metrics can also be used.

Since clustering is an NP-Hard problem, we must deploy heuristics to solve problems in this set. This way, we can be guaranteed an approximate solution in time that can be verifiably polynomial.

2.2 Other Facts/Concepts/Definitions

The algorithm also relies on several other lemmas and formulations for better interpretability and robustness. In order to achieve a thorough understanding of the approaches in the paper, these must be explored further.
Firstly, for a given clustering (set of points) in a dataset, the two-norm of distances from the set S to an arbitrary point l can be expressed in terms of the normed distance from the center of this set to l. Intuitively, such a construction can be fairly convenient and easy to use mathematically while completing several derivations and proofs.

$$\Sigma_{x \in S} ||x - z||^2 - \Sigma_{x \in S} ||x - c(S)||^2 = |S|.||c(S) - z||^2$$

Secondly, for a random cluster R chosen from C$_{OPT}$, we can show that the expected cost function (objective) for R with respect to any arbitrary clustering in the entire dataset (with its center coming from a point in R) is exactly twice the optimal cost for R. This expected value is computed as the average over all possible such arbitrary clustering with centers coming from R. At most, every point in the set can be a centroidal candidate. This can be proved as follows:

$$E[\phi(R)] = \frac{\sum_{x_0 \in R} \sum_{x \in R} ||x - x_0||^2}{|R|} = \frac{\sum_{x_0 \in R} ||x - c(R)||^2 + |R| ||x_0 - c(R)||^2}{|R|} \text{ - - - - - - -(1)}$$

The second term inside the inner summation when summed over all x$_0$ in R is equivalent to the first, thus we obtain twice the first expression inside the inner summation. Simplifying further, we get that:

$$E[\phi(R)] = 2 \sum_{x \in R} ||x_0 - c(R)||^2$$

The above result holds true for a single randomly selected R with center in C$_{OPT}$. Generalizing the above expression for any single optimal cluster and aggregating the final results, we can conclude that:

$$E[\phi(R)] = 2\phi_{OPT} \text{ - - - - - - (2)}$$

Finally, we can represent an expected function as a weighted probabilistic sum over all possible values that it can take. Translating this idea to our problem, we can represent the expected potential function for any R as the sum of squared 2-norms of the minimum value among the distances from any point in R to either it's nearest center, or the selected center in R for an arbitrary clustering in X. This would be weighted by the probability that x0 is the chosen candidate center from R for the arbitrary clustering mentioned above.

$$E[\phi(R)] = p(x_0) \sum_{x \in R} [min(D(x), ||x_0 - x||^2)] \text{ - - - - - - (3)}$$

We can further reduce this expression by applying a triangle inequality for an exponent of 2 to this problem where x and x0 are the points of reference. Doing so, allows us to simplify the above formula into a more readable one. Triangle-Inequality –

$$D(x) = p(x_0) \sum_{x \in R} min(D(x), ||x_0 - x||^2)$$

We can rewrite and simplify (3) with the triangle inequality above to get that

$$E[\phi(R)] \leq \frac{4 \sum_{x_0 \in R} \sum_{x \in R} ||x_0 - x||^2}{|R|}$$

Recall from (1) that the above expression can be re-written as twice the optimal objective value for the set R with center in C_{OPT}. Consequently:

$$E[\phi(R)] \leq 2 * 4 \sum_{x \in R} ||x_0 - c(R)||^2 = 8\phi_{OPT}(R) \text{ - - - - - - (4)}$$

Generalizing for all clusterings with optimal centers in C_{OPT}, we obtain an upper bound on the cost of the clustering problem which is a function of its optimal cost. The final step of the problem would be to call an inductive argument to account for the generalization error of our model. This way, we increase its robustness and obtain a more accurate sense of the true results it can produce. For the log(k) competitiveness of this model, the argument claims that the model error can be best approximated by a factor of log(k). Integrating the above results from equations (2), (4) and the inductive argument, we get that:

$$E[\phi(R)] \leq (2 + 8log(k))\phi_{OPT} \sum_{x \in R} ||x_0 - c(R)||^2 = 8\phi_{OPT}(R)$$

3 Summary of main results

3.1 How is the problem solved?

K-means is a hard partitional clustering problem.
Given n data points, we assign each x_i to one of the k clusters according to
Objective

$$\text{min f(C,P)} = \sum_{i=1}^{M} \sum_{j=1}^{N} p_{ij} ||x_j - c_i||^2$$

subject to

$$\sum_{i=1}^{M} p_{ij} = 1, j = 1, 2, ..., N$$

$$p_{ij} \in \{0, 1\} i = 1, 2, ..., M; j = 1, 2, ...N$$

where

$$c_i = \frac{\sum_{j=1}^{N} p_{ij} x_j}{\sum_{j=1}^{N} p_{ij}} i = 1, 2, ..., M$$

The k-means (Lloyd's) algorithm is solved using the following steps:
Step 1: Initialization: Choose k random centers $c_i \in X_i$.
Step 2: Assignment: Assign each x_i's to the nearest center, given by the Euclidean distance ($\sqrt{x_i^2 - c_j^2}$)
Step 3: Update mean: The centroids are re-calculated by using the xis assigned to it using:

$$c_j^k = \frac{1}{|C_j|} \sum_{i=1}^{N} x_i p_{ij}$$

Step 4: Repeat Steps 2 and 3 are repeated until convergence.

This approach works well with a combination of normally distributed data points. However, it fails to give good results while considering some other types of data (Example - data with varying density across the distribution, non-convex data sets, etc.). The dependency on proper initialization is quite evident. Thus arises the need to come up with improved

seeding techniques. One such way is proposed in the paper under review [1].

This algorithm is termed k-means++. Here, unlike the k-means algorithm, which selects k random centers, only one random center is chosen during initialization. The other k-1 centers are chosen by the following process:
Step 1: Initialization: Choose one random $c_i \in X_i$. Assign this as the first center.
Step 2: Seeding using probability function: The next center is chosen based on the probability function $\frac{D(x)^2}{\Sigma_{x \in X} D(x)^2}$
Step 3: Iteration: This process is repeated until k centers are obtained.
Step 4: k-means calculation: After the completion of these steps, the usual Lloyd's algorithm is implemented.

This seeding technique greatly improves the approximation factor of the cost criterion. Even with the performance of extra steps in the beginning, k-means++ converges in less number of iterations, thus reducing the computation time.
The expected value of the cost criterion for k-means++ is upper bounded as shown:

$$E[\phi] \leq 8(lnk + 2)\phi_{OPT}$$

3.2 Keys steps in theoretical designs

K-means++ algorithm works in the following manner:
1. Choose one center (c_1) randomly from the dataset.
2. For each data point x_i, calculate the distance from c_1 and then further calculate the ratio of the squared distance over total squared distance. This gives the probability of a particular x_i being chosen in the next iteration.
3. Each data point is associated to it's closest center. Then, perform step 2 from it's closest center.
Note: The above 2 steps can be considered a max-min formulation.
4. The steps 2 and 3 are repeated until all k centers are computed.

Since the first step does not guarantee selection of a center from C_{opt}, the cost can be worsened by 2 times the optimum value. That is:

$$E[\phi(A)] = 2\phi_{OPT}(A)$$

An elaboration of the above equation shows:

$$\begin{aligned} E[\phi(A)] &= \tfrac{1}{|A|}\Sigma_{a_0 \in A}\Sigma_{a \in A}||a_0 - a||^2 \\ &= 2\ \Sigma_{a \in A}||a - c(A)||^2 \end{aligned}$$

This follows:

$$\Sigma_{x \in S}||x - z||^2 - \Sigma_{x \in S}||x - c(S)||^2 = |S|.||c(S) - z||^2$$

On similar lines, the subsequent centers selection worsens the cost by a factor of 8, which is given by

$$E[\phi(A)] = \Sigma_{a_0 \in A} \tfrac{D(x)^2}{\Sigma_{x \in X} D(x)^2} \Sigma_{a \in A} min(D(a), ||a - a_0||)^2$$

Using the power-mean inequality, summing over a and performing some algebraic manipulations, the final equation is obtained as follows:

$$E[\phi(A)] \leq \tfrac{4}{|A|}\Sigma_{a_0 \in A}\Sigma_{a \in A}||a - a_0||^2$$

3.3 Flowchart/Algorithm proposed in the paper

K-means Flowchart and K-means++ flowcharts are shown below:

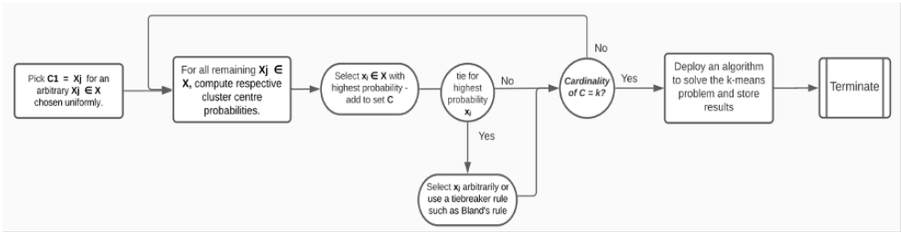

Figure 1: K-means Algorithm

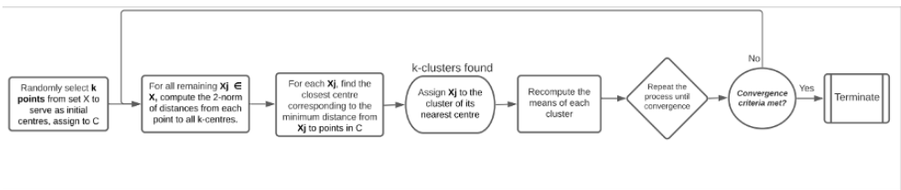

Figure 2: K-means++ Algorithm

4 Application Example

4.1 Code

The code that has been developed has the flow as described below:

- **euclid_dist(df1, df2):** Input: All the data points (actual data set) and chosen centres as dataframes This function is used to calculate the euclidean distance of each data point from the chosen centres. After calculating these distances, distance of the data point from the closest centre is given as the output of the function.
- **Dsq_init(df,k)** : Input: df is the dataframe of the actual data, k= No. of clusters This function calculates the distance of closest centre from each data point by calling Euclid_dist function and then for each datapoint weightage is assigned by calculating the ratio of square of minimun distance to the cost. After assigning the weightage, k No. of points are chosen from the data arbitrarily with the discrete probability distribution given by weightage assigned. Output of the function gives the Initial centres for k-means++.
- **K_means_fit(X, centroids, n)**: Input: X is actual dataset as a framework, Centroids are the chosen centres , k = No. of clusters This function chooses the nearest centroid from each data point and cluster number is assigned accordingly (cluster index is same as position of the centre in centroid dataframe. Copy of the dataframe of actual data is modified by rearranging the datapoints according to cluster numbers. Centroid for these clusters are calculated and saved as centroids_new dataframe. Later centroids dataframe is replaced by centroids_new. Convergence is checked at the end of each iteration and the loop runs until convergence. This function gives an output of modified dataframe, new centroids as dataframe and no. of iterations until convergence.

Following other functions are defined in the code: **k_means(data, k):** Carries out random centre initialization and completes the k-means clustering till convergence. It gives cost after convergence and number of iterations required as outputs. This function calls k_means_fit function described above to cluster using Lloyd's algorithm. **k_means_Dsq(data,k):** Carries out D2 seeding as initialization and completes the k-means++ clustering till convergence. It gives cost after convergence and number of iterations required as outputs. The functions euclid_dist,Dsq_init and k_means_fit are called to obtain the clustering result using the k-means++ algorithm. **k_means_plot(data,k)** and **k_means_Dsq_plot(data,k**) are corresponding functions where final clustering result is given as plot as well.

4.2 Application and data used

The code was run for two datasets. The execution of each is as follows:
Dataset 1: Iris data

- Considering the assumption that the variance in every feature is given equal weightage, PCA was computed for the iris data

- The first 2 Principal components were considered. The clusters in Iris data set are comparatively easily visible and hence, getting the result of correct cluster formation was possible for this data.
- A heat-map for the correctness of the cluster formation was plotted to gauge the accuracy of the algorithm while forming the clusters.

Dataset 2: NORM data

- Another dataset used is synthetically generated by choosing random datapoints from the laplace distribution whose means are 10 fixed points in the 2-D real space.
- The main objective of working on such a custom generated dataset is that we are able to compare the results, since we already have some idea about the optimal cost of the problem as the means for each laplace distribution is known.
- In order to replicate real-world conditions as much as possible, the cardinality for the each cluster was chosen arbitrarily.
- Using this data set, the costs and the number of iterations for different cluster sizes were calculated.

4.3 Statistics of the application results and the data

The minimum and average costs, and average number of iterations required to converge the algorithm were compared across both models (k-means and k-means++). Thereafter, the results were recorded and analyzed for conclusions.

Iris Data:

Comparison of the results from k means and k means++ on Iris data: Both the algorithms were run 100 times and minimum and average costs are calculated. It can be seen that k-means++ gives better costs and on average requires fewer number of iterations for the convergence. Consequently we are able to utilize a more efficient algorithm to accomplish the same task and we end up with better results.

	k_meansAvg	k_means ++ Avg	k_means Min	k_means ++ Min	k_meansAvgI	k_means++AvgI
0	118.439176	116.468868	112.3569	112.3569	5.62	5.24

Figure 3: Comparison of cost and number of iterations for the Iris Dataset

For the same runs mentioned above, clustering of both k-means and k-means++ is noted:

Only k-means++ chooses correct clustering: 22	Both choose correct clustering: 60
Both choose wrong clustering: 5	Only k-means chooses correct clustering: 13

Figure 4: Comparison of k-means and k-means++ clustering

D^2 seeding in k-means++ ensures that the initial centres chosen are well spread in the data so that the resulting clustering can converge in lesser number of steps.

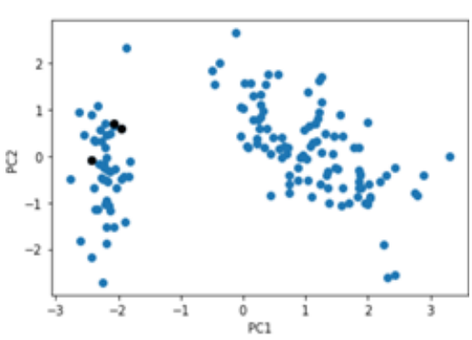

Figure 5: Random centroid initialization – kmeans

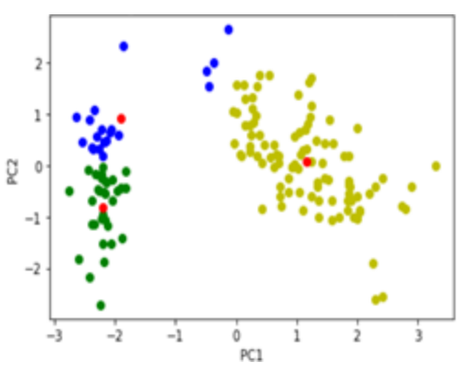

Figure 6: Final clustering result – k means

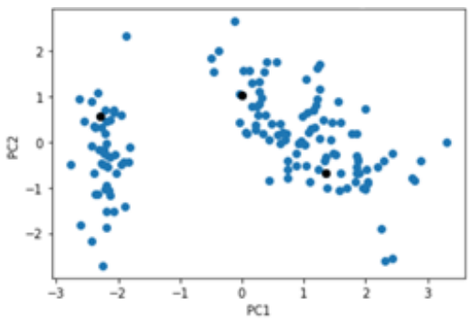

Figure 7: D^2 seeding – k means++

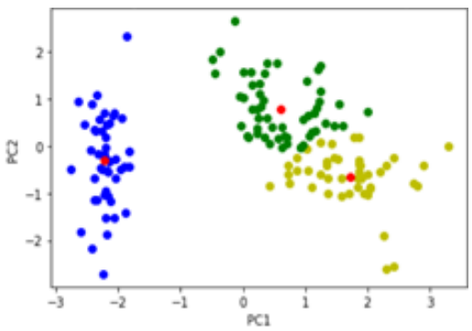

Figure 8: Final clustering result - k means++

Randomly synthesized dataset from several Laplacian Distributions:

Average and minimum costs after the convergence are calculated for 10 runs of both algorithms, each with different set of k values ranging from 2 to 18.
We note that the optimal costs for the k-means++ algorithm are consistently better (smaller) than those of the k-means algorithm. Furthermore, the k-means++ algorithm also frequently converged in fewer iterations as compared to its counterpart.

	Cluster No.	k_meansAvg	k_means ++ Avg	k_means Min	k_means ++ Min	k_meansAvgI	k_means++AvgI
0	2	33170.163288	33005.881940	32896.361041	32896.361041	11.7	10.1
1	6	14568.953915	14569.264588	14457.591694	14457.591694	16.9	14.6
2	10	10725.573261	10581.781382	10346.827602	10356.140905	24.8	28.6
3	14	9536.751927	9587.749706	9393.472089	9426.229204	30.3	25.2
4	18	8744.945718	8738.561355	8627.693476	8627.718730	24.0	27.8

Figure 9: Comparison of cost and number of iterations for the Synthetic Dataset

For the synthetic dataset, as we know the means for each clusters, we can calculate the optimum cost value. We get 2.61% average deviation for the final clustering costs from the optimality when we calculate the costs for k-means++ at each iteration. This follows from the fact that the k-means++ heuristic gives us a bounded cost function and terminates after finite number of iterations.

4.4 Relevance of the coding language

Python was used for the purposes of implementation and analysis of the algorithm. The versatility of python allows us to frame our code in the most lucrative manner. The ease of the language makes this a feasible language for consideration. We were able to accomplish several tedious tasks (such as importing the data, formatting it for analysis, and recursive clustering function implementation) efficiently and with fewer lines of code as compared to any other programming language. Furthermore, we utilized some convenient built-in packages such as numpy, pandas and matplotlib to conduct statistical analysis of the data.

4.5 Complexity analysis of the algorithm

The k-means++ formulation can run in O(nkdi) steps:

k - centres
d – dimensionality of each point in X
n – cardinality of set X
i – Number of iterations Note that i is proportional to log(k)

The algorithm starts by selecting k-centers from X to be the cluster centroids. In total, this step would be repeated k-times (once for each centroid selection)

The algorithm computes n-distances for each of the n datapoints in X to their nearest centers (distance from each centroid to itself would be zero).

The algorithm computes 2-normed distance by performing a computation on each point

and summing the results. However, for a single point, we would need to repeat the distance calculations a total of d times in order to account for variations from the nearest centroid in all dimensions.

Since it takes i iterations to find the initial k-means in the k-means++ algorithm, we account for this added layer of complexity by repeating the computations i times.

Finally, since the k-means++ algorithm always guarantees an upper bound on the potential function value which is log(k) competitive, it is clear that i (number of iterations to find the k-centroids) is proportional to log(k).

5 Summary

As stated before, clustering is a versatile Machine Learning technique which can be used for several tasks such as data analysis and pattern recognition. Some of these methods are critical to solving problems in quantitative and qualitative fields which require a large amount of data analysis for drawing conclusions and results on a particular subject/topic of interest. Consequently, it is imperative that we come up with reliable approaches and algorithms to tackle such problems with ease and confidence. For these reasons, Lloyd's algorithm and an alternative formulation in k-means++ was studied further in this report. While Lloyd's heuristic approach to the k-means problem produces a fairly accurate clustering approximation in a small amount of time, the k-means++ algorithm guarantees that a finite solution to the problem with an upper bound on the maximum order or run-time is possible. The implications are that we are able to solve a subset of NP hard clustering problems with certainty and reliability. It is comforting to know that several topics covered in the course can be thought of as valuable analytical tools to solve challenging problems spanning areas such as predictive analytics, statistical inference, and machine learning. Even though this paper bolstered our understanding of clustering algorithms covered in the course, the ideas and concepts discussed in class helped shape our perspective and critical thinking ability over problems in unsupervised learning.

5.1 Appendix

Note: all relevant codes and datasets can be accessed through the Github link below

https://github.com/ShreekantGokhale/D-square-seeding-in-k-means-clustering.git

6 References

[1] Arthur, D. A., Vassilvitskii, S. V. (2007). k-means++: the advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, New Orleans, Louisiana.

[2] Arthur, D., Vassilvitskii, S. (2006). How slow is the k-means method? Proceedings of the Twenty-Second Annual Symposium on Computational Geometry - SCG '06. https://doi.org/10.1145/1137856.1137880

[3] Berkhin, P. (2006). A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data.

[4] Ortega, J., Rocío, M., Rojas, B., García, M. (01 2009). Research issues on K-means Algorithm: An Experimental Trial Using Matlab. CEUR Workshop Proceedings, 534.