Seeded Center Initialization In `k-means` Clustering

Masum Billal BILLALMASUM93@GMAIL.COM
Farhad Naeem EMAIL
Nobel Khandaker EMAIL
Walid Khan EMAIL
Data Science Department
Shohoz
Bangladesh

Editor: Not assigned

Abstract

`k-means` is a popular clustering algorithm that aims to reduce the sum of minimum squared distances from data points to the centers. Now a days most of the times seeded centers are used instead of choosing all uniformly at random. Initially, our objective was to improve the most popular seeding technique of `k-means` known as `k-means++`. Fortunately, we discovered a lot more than that and we talk about all those findings in this paper. We take a second look at theoretical and experimental results presented in the `k-means++` paper. Experimental results will be shown various type of seeding methods. Therefore, this paper also serves as an updated survey of `k-means` initial center seeding methods. Our conclusion is that, contrary to popular belief, usually other initialization methods than `k-means++` perform better. We also prove a theorem on upper bound of inertia for any `k-means` method.

1. Introduction

Widely regarded as the most popular clustering techniques, `k-means` remains a humble interesting topic in machine learning as well as computational geometry. Roughly the problem is: given a set of points $\mathbf{X}$ in $\mathbb{R}^d$. Find a set of centers $\mathcal{C}$ such that the function *inertia*

$$\mathcal{I} = \sum_{\mathbf{x}\in\mathbf{X}} \min_{\mathbf{c}\in\mathcal{C}} (\|\mathbf{c} - \mathbf{x}\|^2)$$

is minimum where $\|\cdot\|$ is the L_2 norm[1].

Default `k-means` algorithm starts with random centers and then converge based on minimum distances of the centers from the data points. New centers are calculated based on the centroid. This is known as Lloyd's algorithm (Lloyd, 1982). We repeat this process until no more change is possible. Ostrovsky et al. (2006) and Arthur et al. (2007) take it one step further by choosing the initial centers with a probability. And today, `k-means++` is without a doubt the most popular clustering technique. We intend to introduce other ways

1. $\|c - x\|$ or L_2 norm of $\mathbf{c} - \mathbf{x}$ is the distance between the center $\mathbf{c}$ and point $\mathbf{x}$ or the magnitude of the vector $\mathbf{c} - \mathbf{x}$.

of initialization and compare their performances in terms of inertia, convergence speed and CPU time taken.

First, we will discuss a sensitive issue regarding `k-means++` algorithm. Then we discuss the initialization methods used to compare the performance of clustering. We also prove a theorem that provides an upper bound for the inertia when `k-means` is done using seeded centers. Finally, we show experimental results for both our argument regarding `k-means++` and performance of initialization methods. For experimental results, we assumed that an algorithm converges only when the centers do not change anymore, that is, the algorithm absolutely stops changing centers altogether. There are some available techniques of stopping `k-means` algorithms but we chose not to use any of them as they might produce unreliable results.

2. Related Work

There has been multiple surveys on `k-means` in the literature. Probably the most relevant work in this regard is done by Celebi et al. (2013). However, most of the algorithms used in that paper are not used practically very much. It was also noted by the authors themselves that `k-means++` and its greedy version work better than most. They also mention that probabilistic algorithms perform better than deterministic ones. Moreover, the not so popular algorithm (Ostrovsky et al., 2006) (`ORSS`) was not considered in their experiments. There is no mention of Ostrovsky's algorithm in their paper whatsoever. Experimental results do not agree with their decision of not including `ORSS` in their survey. Of course there are other algorithms such as Forgy (1965), etc but as we mentioned, most of them are not used practically as much. Therefore, we focused our attention only on the ones that matter the most.

We would like to point out that to our knowledge no surveys were done after removing linear dependency prior to running the experiments. This is a very important step if we are to get a meaningful clustering out of `k-means` algorithm. We can achieve this using some known algorithm such as **principle component analysis** or **singular value decomposition**. We opted for PCA in this paper. PCA decomposes the existing data points into orthogonal[2] ones. If we use PCA (principal component analysis) before running a clustering algorithm, we can redefine the variables into linearly independent ones. For our experiment, we have used PCA on every data set before running cluster algorithm. However, we did not reduce dimensions in order to preserve the originality of the data set. We only used PCA for removing linear dependency among variables. We strongly believe this strengthens our result over other available results.

3. How Good Is `k-means++`?

Given how simple `k-means++` is, it seems very justified that it is the most popular algorithm for `k-means`. Even in the very well known scientific python package `scikit-learn` (Pedregosa et al., 2011), `k-means++` initialization is supported by default. And it is no surprise considering the theoretical and experimental results are presented in (Arthur et al., 2007). For example, Arthur et al. (2007) claim `k-means++` is $\log k$ competitive which

2. It is well known that orthogonal vectors are linearly independent.

is a very lucrative result without any doubt. Moreover, they show experimental results that k-means++ performs tenfold or even better in most data sets. These are all properties we look for in an ideal algorithm. However, there are some concerns we raise in the next section.

3.1 k-means++ vs ORSS

Surprisingly, there is no mention of Ostrovsky's algorithm in k-means++ paper either even though it was published in 2007 whereas Ostrovsky et al. (2006) was published in 2006. In fact, after going through ORSS paper again, we argue that k-means++ algorithm is actually a special case of ORSS algorithm. In k-means++, the first center is chosen at random. Then every center is chosen based on the minimum distance of the existing centers from the point in consideration, which Arthur et al. (2007) call D^2 *weighting*. Now, ORSS chooses two centers x, y with probability proportional to $\|x-y\|^2$, which seems very simple at first sight but there is more to it than meets the eye. After choosing those two centers, a new center is added to the set of centers in each iteration until there are k centers, which is done exactly as in k-means++. So, the center adding step is same for both k-means++ and ORSS. Considering that ORSS was published before k-means++, we want to give credit to Ostrovsky et al. (2006) more than Arthur et al. (2007) for this novel approach.

Next, we talk about the first step where k-means++ chooses first center at random and ORSS chooses two centers with probability proportional to $\|x-y\|^2$. At this point, we would like to say that our proposed method of improving k-means++ is not novel at all. In fact, this was already discussed in (Ostrovsky et al., 2006). However, we did not know that at the time because we did not read the full paper back then. It was only after we discovered that k-means++ results could be wrong, we went through ORSS paper again. This time we paid more attention to it and we consider ourselves lucky to have discovered such a gem. We would really like to commend Ostrovsky et al. (2006) for such an impressive work and it is indeed very unlucky of them not getting the credit they deserve. When we read their paper for the second time, we found out that they had already mentioned a result (Ostrovsky et al., 2006, Page 4, section 3, Paragraph: Running Time) similar to ours in their paper. They say that choosing two centers x, y as in ORSS is the same as choosing the first center x with probability proportional to $\sum_{y \in X} \|x-y\|^2$ and the second center with probability proportional to $\|y-c_1\|^2$. As we will see in section 5 that the first step was exactly our idea of improving k-means++. Despite being a duplicate result, we have decided to discuss our reasoning in this paper because we believe our motivation was different than theirs. As for the second step, again, notice that this is the same as second step in k-means++. In k-means++, after the first center has been chosen randomly, a point x actually gets chosen with probability $\|x-c_1\|^2$. The reason is, there is only one center so the minimum distance is the only distance from c_1 to x.

Moreover, including ORSS makes the results very interesting. ORSS is like a dark horse. There are cases where ORSS is better than both k-means++ and our proposed initialization methods. Therefore, we conclude that ORSS is actually an improved version of k-means++.

3.2 `k-means++` Results Are Wrong?

We will now discuss the other issue at hand. Regarding the accuracy of `k-means++` results, we have two concerns. One is that the theoretical results are not correct. And the other is that experimental results might not be correct either.

Since we tried to improve `k-means++`, obviously we made our own implementations of the algorithm and checked the results against the ones shown in the paper. Our results conflict with some results shown in `k-means++` paper Arthur et al. (2007). For this reason, we ran clustering on the same data set (cloud data set) using `k-means` package from `scikit-learn` library (Pedregosa et al., 2011). We found out that our implementations achieve similar inertia to the one in `scikit-learn`. On the other hand, results shown in `k-means++` paper differ from both our and `scikit-learn` results. At first, we thought this might be due to normalization or standardization. However, even after using both `min max` and `standard` normalization (also known as z-score), we found out that `k-means++` results still do not match with neither of the results. We tried collecting every data set that is used in `k-means++` paper but we managed to get only the cloud data set. We could not find the other real world data set *Intrusion* data set. There were some data sets named intrusion detection data set but they did matched neither the number of data points (494019) nor the dimension (35). Therefore, cloud data set was our primary source of comparison.

4. Proposed Initialization Methods

For a set of points S and a point x, we use $\min(\|x-S\|)$ to denote the minimum of distances from x to the points of S that is $\min(\|x-S\|) = \min_{a\in S}(\|x-a\|)$. Set $D(\mathbf{x}) = \min(\|\mathbf{x}-\mathcal{C}\|)$ for a point x and a set of centers $\mathcal{C}$. Let us denote the centroid of S by μ_S that is $\mu_S = \frac{1}{|S|}\sum_{x\in S} x$.

4.1 First center for `k-means++`

In `k-means++` algorithm, the first center is chosen uniformly at random. However, not all points have the same contribution to inertia. We choose x as a first center in a way that is equivalent to the variance explained by x.

i Choose $\mathbf{x}$ with probability $\dfrac{\|\mathbf{x}-\mu_{\mathbf{X}}\|^2}{\sum_{\mathbf{x}\in\mathbf{X}}\|\mathbf{x}-\mu_{\mathbf{X}}\|^2}$. Set $\mathcal{C}_1 = \{\mathbf{x}\}$.

ii Repeat the remaining steps in `k-means++` (Arthur et al., 2007, Section 2.2, Page 3).

4.2 Centroid of Centers Based Seeding

We want to choose x with probability proportional to squared distance from centroid of the cluster centers. Our motivation for doing so is the following. In `k-means++`, probability is considered proportional to squared minimum distance from the centers. Therefore, the larger this minimum squared distance is, the higher the probability is for x to be chosen as a center. So, in a sense this can be thought of maximizing the minimum squared distance from the centers to the point in consideration. We intend to check the case where we choose the probability proportional to the total sum of squared distances rather than just

the minimum one. As we will show later, sum of all squared distances from centers to the point in discussion is dependent on the distance from centroid of those cluster centers to that point. Note that this opens up the discussion where we need to check other ways of taking this probability. For example, minimizing variance of distances from centers, etc. However, those ideas are out of our scope for this paper.

i Choose a point $\mathbf{x}$ as stated in step (i) of section (4.1). Set $\mathcal{C}_1 = \{\mathbf{x}\}$.

ii For an already existing set of i centers $\mathcal{C}_i = \{c_1, \cdots, c_i\}$, choose a new center $\mathbf{x} \in \mathbf{X}$ with probability proportional to $\|\mathbf{x} - \mu_{\mathcal{C}_i}\|^2$.

iii Repeat step (ii) until $i = k$.

iv For each $1 \leq i \leq k$, set $\mathcal{C}_i = \{\mathbf{x} \in \mathbf{X} : \|\mathbf{x} - c_i\| = \min(\mathbf{x} - \mathcal{C})\}$.

v Set $c_i = \mu_{\mathcal{C}_i}$.

vi Repeat (iv) and (v) until convergence is reached.

4.3 Variance Based Seeding

In this method, we have tried making the clusters as balanced as possible in terms of radius. That is, for a set of centers $\mathcal{C}$ and a point $\mathbf{x} \in \mathbf{X}$, we want to minimize the variance of the squared distances $\{\|\mathbf{x} - \mathbf{c}_1\|^2, \cdots, \|\mathbf{x} - \mathbf{c}_k\|^2\}$ as much as possible. Denote this variance using $\nu_{\mathbf{x}}(\mathcal{C})$.

i Choose two points $\mathbf{x}, \mathbf{y}$ as stated in (Ostrovsky et al., 2006, Section 4.1, subsection 4.1.1).

ii For an already existing set of i centers $\mathcal{C}_i = \{c_1, \cdots, c_i\}$, choose $\mathbf{x} \in \mathbf{X}$ with probability $1 - \dfrac{\nu_{\mathbf{x}}(\mathcal{C})}{\sum_{\mathbf{y} \in \mathbf{X}} \nu_y(\mathcal{C})}$.

iii Repeat step (ii) until $i = k$.

iv For each $1 \leq i \leq k$, set $\mathcal{C}_i = \{\mathbf{x} \in \mathbf{X} \| \mathbf{x} - c_i\| = \min(\mathbf{x} - \mathcal{C})\}$.

v Set $c_i = \mu_{\mathcal{C}_i}$.

vi Repeat (iv) and (v) until centers change no longer.

5. Motivation

In this section, we discuss the motivation behind our algorithm and why they make sense mathematically. Consider a set of n points $\mathbf{X}$ and that the probability of $x \in \mathbf{X}$ being chosen as a center as $p(x)$. Following the definition of variance, for a set of points S, we define

$$\sigma^2(S) = \frac{\sum_{x \in S} \|x - \mu_S\|^2}{|S|}$$

$$\sum_{x \in S} \|x - \mu_S\|^2 = |S|\sigma^2 \tag{1}$$

For any arbitrary point a and μ as the centroid of $\mathbf{X}$,

$$\begin{aligned}
\sum_{\mathbf{x}\in\mathbf{X}} \|\mathbf{x}-a\|^2 &= \sum_{\mathbf{x}\in\mathbf{X}} \|\mathbf{x}-\mu+\mu-a\|^2 \\
&= \sum_{\mathbf{x}\in\mathbf{X}} \left(\|\mathbf{x}-\mu\|^2 + 2\langle \mathbf{x}-\mu, \mu-a\rangle + \|\mu-a\|^2\right) \\
&= \sum_{\mathbf{x}\in\mathbf{X}} \|\mathbf{x}-\mu\|^2 + 2\left\langle \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}-n\mu, \mu-a \right\rangle + n\|\mu-a\|^2 \\
&= n\sigma^2 + 2\langle n\mu - n\mu, \mu-a\rangle + n\|\mu-a\|^2 \\
&= n(\sigma^2 + \|\mu-a\|^2) \qquad (2)
\end{aligned}$$

Here $\langle a, b\rangle$ is the dot product of vectors a and b. Using equation (2), we have the following.

$$\begin{aligned}
\sum_{\mathbf{x}\in\mathbf{X}}\sum_{\mathbf{y}\in\mathbf{X}} \|\mathbf{x}-\mathbf{y}\|^2 &= \sum_{\mathbf{x}\in\mathbf{X}} n(\sigma^2 + \|\mu-\mathbf{x}\|^2) \\
&= n(n\sigma^2 + \sum_{\mathbf{x}\in\mathbf{X}} \|\mu-\mathbf{x}\|^2) \\
&= n(n\sigma^2 + n\sigma^2) \\
&= 2n^2\sigma^2 \qquad (3)
\end{aligned}$$

Using equation (3), the probability becomes

$$\begin{aligned}
p(x) &= \frac{\sum_{\mathbf{y}\in\mathbf{X}} \|\mathbf{x}-\mathbf{y}\|^2}{\sum_{\mathbf{y}\in\mathbf{X}}\sum_{\mathbf{x}'\in\mathbf{X}} \|\mathbf{x}'-\mathbf{y}\|^2} \\
&= \frac{n(\sigma^2 + \|\mu-\mathbf{x}\|^2)}{2n^2\sigma^2} \\
&= \frac{\sigma^2 + \|\mu-\mathbf{x}\|^2}{2n\sigma^2} \\
&= \frac{1}{2n} + \frac{\|\mu-\mathbf{x}\|^2}{2n\sigma^2}
\end{aligned}$$

However, to make things smoother, one can also choose to use the following as the probability of x being chosen as the first center.

$$p(x) = \frac{\|\mu-\mathbf{x}\|^2}{n\sigma^2}$$

Notice that the variance of S is in the denominator. We can consider this as the amount of variance explained by x. Also, notice that computationally this version of $p(x)$ is much cheaper than using $\sum_{\mathbf{y}\in\mathbf{X}} \|\mathbf{x}-\mathbf{y}\|^2$. Therefore, we considered $p(x)$ proportional to $\|x-\mu\|^2$ for choosing x as the first center in our experiments.

6. The Inertia Theorem

For the minimum distance from $D(x)$ from x to the set of points $\mathcal{C}$,

$$\begin{aligned} D(x)^2 &\leq \frac{1}{k}\sum_{c\in\mathcal{C}} \|x-c\|^2 \\ &= \frac{1}{k}\left(k\|x-\mu_{\mathcal{C}}\|^2 + \sum_{c\in\mathcal{C}}\|\mu_{\mathcal{C}}-c\|^2\right) \\ &= \|x-\mu_{\mathcal{C}}\|^2 + \frac{1}{k}\sum_{c\in\mathcal{C}}\|\mu_{\mathcal{C}}-c\|^2 \end{aligned}$$

Thus, we can write inertia as

$$\begin{aligned} I &= \sum_{x\in S}\min_{c\in C}\|x-c\|^2 \\ D(x)^2 &= \min_{c\in C}\|x-c\|^2 \\ D(x)^2 &\leq \|x-\mu_C\|^2 + \frac{1}{k}\sum_{c\in C}\|\mu_C-c\|^2 \\ I &\leq \sum_{x\in S}\left(\|x-\mu_C\|^2 + \frac{1}{k}I_C\right) \\ &= \sum_{x\in S}\|x-\mu_C\|^2 + \frac{n}{k}I_C \\ &= \sum_{x\in S}\|x-\mu\|^2 + n\|\mu-\mu_C\|^2 + \frac{n}{k}I_C \end{aligned}$$

This holds for any seeding technique.

Theorem 1 *For a set of points S and a set of centers C, we have*

$$\mathcal{I}(S) \leq n(\sigma^2 + \|\mu_S-\mu_C\|^2) + \frac{n}{k}\mathcal{I}_{opt}(C)$$

regardless of what seeding method is used.

7. Experiment Setup

We ensure that all algorithms are run under the same conditions. All of them share the same environment and no special optimizations were made for any particular algorithm. Only CPU was used to determine the values we are interested in and no parallelism mechanism was in place for speeding up the process. This way, we can get an idea about the raw performances of the algorithms involved.

Python is used as the programming language to write necessary codes. Some common auxiliary packages such as *scipy, scikit-learn, numpy* etc are used to help with the code. The algorithms are simply different methods of the same class, so they share the same fitting and prediction function. Only the initialization differs for different algorithm. It

should be mentioned that even though some packages have native support for `k-means` implementation, we did not use them to run the experiments. Not all algorithms we want to test are available in those packages. Therefore, in order to ensure same environment and optimizations for every algorithm, we wrote them all from scratch so that we could be sure they are tested under the same settings.

The data sets used for the experiment are some of the popular ones.

1. Boston housing data set
2. Wine quality testing data set
3. Mall customers data set
4. Airlines cluster data set
5. Cloud data set

Miligan et al. (1988) shows that using z-score to standardize the data is not favorable for clustering because it loses between-cluster variation. Therefore, we did not use any sort of standardization or normalization lest it should lose variance or become prone to bias. Instead, we have used PCA to remove linear dependency among variables.

While experimenting on such algorithms, it is of utmost importance to run the same experiment more than once under the same parameters and conditions. For example, assume that we want to compare `k-means++` and Forgy's algorithm (Forgy, 1965) for $k = 5$ clusters. We should run this experiment at least m times where $m > 1$ in order to eliminate bias and account for randomness. We run each experiment a total of 20 times and take the average and minimum values of inertia.

As we will see in section (8), usually every method can reach the minimum inertia at least once. Therefore, the method with lower average performs more consistently without any doubt. However, there are also cases where the difference between default `k-means` and seeded `k-means` becomes obvious.

8. Results

We present the results on the initialization procedures mentioned before. We have shown the results for $k = 10$ clusters. The ceiling of average number of iterations is taken since it may not be an integer. The time registered here is the average CPU time taken by each algorithm for those 20 iterations. As expected, default `k-means` is usually the fastest algorithm but also the worst in terms of optimizing inertia.

The comparison of inertia and number of iterations required for convergence are shown in tables 1, 3, 4.

9. Conclusion

We proposed methods to improve current state of `k-means` algorithms and showed their comparison in terms of convergence and inertia. `k-means++` is usually worse than both our proposed algorithm and `ORSS`.

Algorithm	Average $\mathcal{I}$	Minimum $\mathcal{I}$	Iterations	Time
`k-means`	2576156.1	1467415.54	9	0.09s
`k-means++`	1685296.83	1442170.41	7	0.1s
`ORSS`	1612137.72	1442170.41	7	0.1s
CoC	1604805.21	1443125.04	9	0.09s

Table 1: Results on Boston housing data set, 5 clusters

Algorithm	Average $\mathcal{I}$	Minimum $\mathcal{I}$	Iterations	Time
`k-means`	2586435.45	1442170.41		2.68s
`k-means++`	1638992.86	1442170.41		2.66s
`ORSS`	1620320.77	1442170.41		2.69s
CoC	1562697.24	1442170.41		2.66s

Table 2: Results on Cloud data set, 10 clusters

Algorithm	Average $\mathcal{I}$	Minimum $\mathcal{I}$	Iterations	Time
`k-means`	1011171.27	916424.19		0.93s
`ORSS`	1020273.2	916424.19		0.94s
`k-means++`	1011171.27	916424.19		0.94s
CoC	994075.55	916424.19		0.94s

Table 3: Results on Wine data set, 5 clusters

Algorithm	Average $\mathcal{I}$	Minimum $\mathcal{I}$	Iterations	Time
`k-means`	2672710216978.34	2633315933118.1	75	9.33s
`ORSS`	2684480210031.3	2622281801735.52	40	5.52s
`k-means++`	2681719187656.43	2626435706106.56	40	5.41s
CoC	2667560530212.89	2627599615029.58	73	9.13s

Table 4: Results on Airlines data set, 10 clusters

References

D. Arthur and S. Vassilvitskii. `k-means++`: The Advantages of Careful Seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035.

E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. Biometrics, $21, 768 - 780, 1965$.

Stuart P. Lloyd. Least squares quantization in pcm. IEEE Transactions on Information Theory, $28(2) : 129–136, 1982$.

M. E. Dyer. A simple heuristic for the p-center problem. Operations Research Letters, Volume 3, February 1985, pp. $285 - 288$.

G. Milligan, M. C. Coope. A Study of Standardization of Variables in Cluster Analysis, Journal of Classification 5(2) (1988) 181–204.

R. Ostrovsky, Y. Rabani, Leonard J. Schulman, C. Swamy. The Effectiveness of Lloyd-Type Methods for the k-Means Problem. Proceedings of the 47th Annual Symposium on Foundations of Computer Science. 2006.

M. Emre Celebi, Hassan A. Kingravi, Patricio A. Vela. A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm. Expert Systems with Applications, $40(1) : 200–210, 2013$.

Tom M. Apostol. Introduction to Analytic Number Theory. Springer, $10.1007/978 - 3 - 662 - 28579 - 4$, 1976.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. Scikit-learn: Machine Learning in Python, , JMLR 12, pp. 2825-2830, 2011.