K-Means: Variance Based Initialization And A Comparative Study

Masum Billal

November 18, 2019

Abstract

k-Means is a popular clustering algorithm that reduces sum of minimum distances from data points to the centers. In this paper, we introduce a new way of initializing the centers. We also show comparison of different initialization for k-Means centers in terms of inertia, convergence speed and variance. Experiments show that our variance based algorithm performs better than k-Means, Ostrovsky-Rabani-Schulman-Swamy (ORSS) and k-Means++ algorithm in terms of minimizing sum of squared distances from each point to their respective clusters.

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 $\mathbb{X}$ in $\mathbb{R}^d$. Find a set of centers $\mathcal{C}$ such that the function *inertia*

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

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[2]. This is done until no more change is possible. k-Means++ takes it one step further by choosing the initial centers carefully. Only the first center is chosen at random. Then the rest of the $k-1$ centers are chosen using D^2 *weighting* as Arthur and Vassilvitskii[1] call it. At first it seems very surprising that no one really wants to work on improving on the centers. However, that is easily explained with the following.

$$\begin{aligned}
E &= \sum_{i=1}^{n} ||\mathbf{x}_i-\mathbf{x}||^2 \\
\implies E &= \sum_{i=1}^{n} (\mathbf{x}_i-\mathbf{x}) I (\mathbf{x}_i-\mathbf{x})^T \\
\implies \frac{\partial E}{\partial x} &= -2\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{x}) \\
\frac{\partial E}{\partial x} &= 0 \\
\iff \mathbf{x} &= \frac{\sum_{i=1}^{n}\mathbf{x}_i}{n}
\end{aligned}$$

where I is the identity matrix of the same rank as $\mathbf{x}$. This pretty much shows why taking the average of all the coordinates in a cluster minimizes the sum of squared distances.

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, Kingravi and Vela [5]. However, most of the algorithms used for comparison are practically

[1]Simply put, it is the distance between the center $\mathbf{c}$ and point $\mathbf{x}$ or the magnitude of the vector $\mathbf{c}-\mathbf{x}$

not used very much, as also noted by the authors themselves that `k-means++` and its greedy version work better than most. They also mention that probabilistic algorithms work better than deterministic ones. Moreover, it seems the first of the most influential center initialization algorithm [4] was not considered for their experiments. There is no mention of Ostrovsky's algorithm in their paper whatsoever. Therefore, we should compare the probabilistic algorithms and so we take a look at exactly how good `k-means++` algorithm is. We propose another way of initialization. Experiments show that this performs better than Lloyd random initialization but

The other reason why we need to take a second look at the results in [1] is that, experiments suggest that some standard procedures were not performed for comparison in [1]. They were not normalized or standardized as their high inertia values indicate. Moreover, even without following those procedures, our results seem to differ a little bit from theirs. This may not be any technical error in coding, most likely this is just different ways of implementation coupled with the fact that these are probabilistic algorithms.

We would also 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. Specially, if we use PCA (principal component analysis) before running a clustering algorithm, this helps us redefine the variables into linearly independent variables since PCA decomposes the existing variables into orthogonal[2] ones. For our experiment, we have used PCA on every dataset before running cluster algorithm[3].

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 [6] 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. It is commendable that [5] follows the literature in using $m = 100$ as mentioned in the paper. In this paper, we use $m = 100$ as well.

To summarize, we will be mostly looking at these variations of initial seeding of centers: Lloyd, Ostrovsky, `k-means++` and a variance based seeding. Then we will be comparing inertia for a comparative study among these algorithms.

3 Variance Based Initialization

We will describe k-Means, Ostrovsky's and k-Means++ algorithm. Then we describe our proposed method of center initialization.

3.1 k-Means Algorithm

k-Means is the simplest of all algorithms.

i Choose k centers $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$ randomly.

ii For each i, C_i is the set of points $\mathbf{x} \in \mathbf{X}$ such that $||\mathbf{x} - c_i|| \leq ||\mathbf{x} - c_j||$ for all $1 \leq j \leq n$.

iii Set $c_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$.

iv Repeat step 2 and 3.

The initial centers are chosen at random uniformly.

3.2 ORSS Algorithm

ORSS algorithm does not choose initial centers uniformly.

i Choose two points $\mathbf{x}, \mathbf{y}$ with probability proportional to $||x - y||^2$.

ii For already i existing centers, choose a random point $\mathbf{x} \in \mathbf{X}$ with probability proportional to $\min(||x - c_j||^2)$ for $1 \leq j \leq j$. Set $c_{i+1} = x$.

iii Repeat previous step until $i = k$ while recalculating centers as in k-Means.

[2]It is easy to prove that orthogonal vectors are linearly independent.

[3]Number of components was set to 2 for simpler visualization and verification.

3.3 k-Means++ Algorithm

k-Means++ is supposed to be an improvement on ORSS algorithm. Although it is fair to say that k-Means++ is an inspired modification of ORSS.

i Choose a point $\mathbf{x} \in \mathbf{X}$ at random uniformly. This is the first center.

ii Choose a new center $\mathbf{x} \in \mathbf{X}$ with probability $\frac{D(x)}{\sum_{\mathbf{x}\in\mathbf{X}} D(x)}$ where $D(x)$ is the minimum distance of $\mathbf{x}$ from the centers $c_1, \cdots, c_i$.

iii Repeat previous step until k centers are chosen and update them as in k-Means.

3.4 Var-based Algorithm

We do not provide any mathematical analysis of our algorithm right now. However, we want to mention the idea behind this initialization. Roughly, we want to minimize the variance of distances from the new center to existing ones. The reason why this is so important and also works better experiment-wise is that it tries to keep the clusters as balanced as possible, hence possibly eliminating some anomalies within a cluster and reducing inertia. However, since the new centers are taken based on variances, we need two initial centers. We chose to initialize the first two centers as in ORSS algorithm. Assume that for a new point $\mathbf{x}$ and a set of centers $\mathcal{C}_k = \{c_1, \cdots, c_k\}$, the variance of the squared distances $\{||c_1 - \mathbf{x}||^2, ||c_2 - \mathbf{x}||^2 \cdots, ||c_k - \mathbf{x}||^2\}$ is $\nu_{\mathbf{x}}(\mathcal{C}_k)$.

i Choose two centers $\mathbf{x}, \mathbf{y}$ with probability proportional to $||\mathbf{x} - \mathbf{y}||^2$.

ii For 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 $\frac{\nu_{\mathbf{x}}(\mathcal{C}_i)}{\sum_{\mathbf{x}\in\mathbf{X}} \nu_{\mathbf{x}}(\mathcal{C}_i)}$.

iii Repeat previous step and recalculate centers as in k-Means.

4 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 performance metrics 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. All 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. The reason is simple. Not all the algorithms we want to test are available in those packages. Therefore, in order to ensure same environment and optimizations for every algorithm, we wrote every algorithm from scratch so that we could be sure they are done in the same setting. In order to check performance bias, we have used both normalization and standardization. Some datasets were normalized and others were standardized instead.

The datasets used are some popular ones: Iris dataset, Mall customers dataset, Airline clustering dataset, Wine testing dataset. We did not duel too much on using too many datasets. The number of datasets does not really mean very much for k-Means. It is the quality of clustering we are interested in. And a good clustering algorithm should be able to handle all type of datasets. For a fixed number of clusters k, we choose a dataset we want to run the experiment on. After loading and cleaning the data, we use PCA with number of components set to either 2 or 3. Then we run the experiment on the modified dataset. For every dataset, we repeat the experiment a total of 100 times as mentioned before. Finally, we use the average and minimum values of inertia, and variance.

5 Results

We present the results on those four algorithms based on inertia and time required to runt them. These are average values of inertia after 100 iterations on each dataset, as mentioned before. After seeing the results, one may argue that the difference is negligible. We beg to differ. We standardized the datasets and used PCA before running the cluster algorithms like we mentioned. Therefore, these very small differences actually signify a lot higher value if converted to original values. Moreover, instead of checking how many iterations it take for the centers to converge, we have let each algorithm run exactly the same number of times (500) which is more than enough for convergence. The time registered here is the time taken by each algorithm for those 100 iterations. As expected, default k-Means is the fastest algorithm. However, if we were to define convergence, that leads to a separate problem which we think other implementations suffer from. We also mentioned something similar before. This might also be one of the reasons why our results of k-Means++ differ from the authors implementation. Our argument is also complemented by the experiments. All algorithms seem to reach the same minimum inertia. Therefore, it is the average of inertia that we should focus on.

Here are the results for $k = 5$ clusters.

Algorithm	Average	Minimum	Time
k-Means	4.582897752536098	4.075902609386082	1.7800957500000003s
ORSS	4.3710321716046785	4.075902609386082	1.7859569949999994s
k-Means++	4.284250240984546	4.075902609386082	1.786941793000002s
Var-based	4.265397551315433	4.075902609386082	1.8023865810000048s

Table 1: Results on Mall customers dataset

Algorithm	Average	Minimum	Time
k-Means	2.1264993964151784	1.4774401153809424	0.12384454899999969s
ORSS	1.8621443053327922	1.4774401153809424	0.13034087000000005s
k-Means++	1.8327558137236732	1.4774401153809424	0.13094220199999967s
Var-based	1.7742336610964762	1.4774401153809424	0.13951454999999968s

Table 2: Results on IRIS dataset

Algorithm	Average	Minimum	Time
k-Means	1388.694358670742	1387.3919240019864	5.6806866900000115s
ORSS	1392.3920454250817	1387.3919240019864	5.641342363999997s
k-Means++	1392.0420776007209	1387.3919240019864	5.672814887999994s
Var-based	1387.6963754678748	1387.3919240019864	5.730628591999996s

Table 3: Results on Cloud dataset

References

[1] David Arthur and Sergei 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.

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

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

Algorithm	Average	Minimum	Time
k-Means	151.77628080921232	145.4648931505064	1.445791673000001s
ORSS	150.59855263211452	145.4648931505064	1.4534875189999985s
k-Means++	148.90907392065532	145.4648931505064	1.4551421590000013s
Var-based	148.86819005795846	145.4648931505064	1.4661181270000034s

Table 4: Results on Wine dataset

[4] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, Chaitanya Swamy. The Effectiveness of Lloyd-Type Methods for the k-Means Problem. Proceedings of the 47th Annual Symposium on Foundations of Computer Science. 2006.

[5] 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, $40(1) : 200–210, 2013$.

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