Variance Based Initialization for k-Means Clustering

Masum Billal billalmasum93@gmail.com
Senior Data Scientist
Data Science Department
Shohoz
Bangladesh

Editor: To be submitted

Abstract

k-Means is a popular clustering algorithm that reduces sum of minimum 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. k-Means++ chooses the centers with a so called D^2 weighting. In this paper, we introduce a new way of initializing centers. We also describe a way to choose the first center with probability instead of choosing uniformly at random for k-Means++. Empirical evidence shows that our proposed algorithm performs better than available methods of center initialization and that our way of choosing first center for k-Means++ performs better than original k-means++.

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 algorithmLloyd (1982). 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 et al. (2007) call it. At first it seems very surprising that no one really wants to work on improving on the centers. However, that is easily

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 $\mathbf{c} - \mathbf{x}$.

explained with the following.

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

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 et al. (2013). However, most of the algorithms used for comparison are practically 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 perform better than deterministic ones. Moreover, another highly influential center initialization algorithm Ostrovsky et al. (2006) was not considered in their experiments. There is no mention of Ostrovsky's algorithm in their paper whatsoever.

We suspect that standard procedures were not followed for comparison in Arthur et al. (2007). While the authors provided theoretical limits, data sets were not normalized or standardized as their high inertia values indicate. Even without following those procedures, our results seem to differ quite a little bit from theirs. This may not be any technical error in implementation. Most likely this is due to different ways of implementations 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. More specifically, PCA decomposes the existing variables 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 dataset before running cluster algorithm.

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 $k = 10$ clusters. We should run this experiment at least m times where $m > 1$ in order to eliminate

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

bias and account for randomness. We run each experiment a total of 20 times and take the average and minimum values of inertia. The number of iterations was set to 300.

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

3. Proposed Initialization

First we describe k-Means, Ostrovsky's and k-Means++ algorithm. Then we describe our proposed method of center initialization. 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 $\bar{S}$ that is $\bar{S} = \dfrac{1}{|S|} \sum_{x \in S} x$.

3.1 Lloyd

Lloyd's algorithm is the simplest way of solving the k-Means problem.

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

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

iii Set $c_i = \bar{\mathcal{C}}_i$.

iv Repeat step 2 and 3 until convergence is reached or the number of iterations is reached.

3.2 ORSS

ORSS algorithm Ostrovsky et al. (2006) chooses two initial centers instead of one.

i Choose two points $\mathbf{x}, \mathbf{y} \in \mathbf{X}$ with probability proportional to $\|\mathbf{x} - \mathbf{y}\|^2$. Set $\mathcal{C}_2 = \{\mathbf{x}, \mathbf{y}\}$.

ii For a set of $i \geq 2$ existing centers $\mathcal{C}_i$, choose a random point $\mathbf{x} \in \mathbf{X}$ with probability $\dfrac{D(\mathbf{x})^2}{\sum_{\mathbf{y} \in \mathbf{X}} D(\mathbf{y})^2}$. Set $c_{i+1} = \mathbf{x}$ and $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{c_{i+1}\}$.

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

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

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

vi Repeat (iv) and (v) until convergence or number of iteration is reached.

3.3 k-Means++

k-Means++ Arthur et al. (2007) can be considered as an improvement on ORSS algorithm.

i Choose a point $\mathbf{x} \in \mathbf{X}$ at random uniformly and set $\mathcal{C}_1 = \{\mathbf{x}\}$.

ii For a set of $i \geq 1$ existing centers $\mathcal{C}_i$, choose a random point $\mathbf{x} \in \mathbf{X}$ with probability $\dfrac{D(\mathbf{x})^2}{\sum_{\mathbf{y}\in\mathbf{X}} D(\mathbf{y})^2}$. Set $c_{i+1} = \mathbf{x}$ and $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{c_{i+1}\}$.

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

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

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

vi Repeat (iv) and (v) until convergence or number of iteration is reached.

3.4 Variance Based

Roughly our idea is that we want to minimize the variance of distances from the new center to existing ones. The intuition behind this is to make 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 $\|\mathbf{x} - \mathbf{y}\|^2$. Set $\mathcal{C}_2 = \{\mathbf{x}, \mathbf{y}\}$.

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 $\rho(\mathbf{x}) = 1 - \dfrac{\nu_{\mathbf{x}}(\mathcal{C}_i)}{\sum_{\mathbf{y}\in\mathbf{X}} \nu_{\mathbf{y}}(\mathcal{C}_i)}$.

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

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

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

vi Repeat (iv) and (v) until convergence or number of iteration is reached.

3.5 First center for k-Means++

Our idea is to choose a point $\mathbf{x}$ with probability that has the most variance among $\mathbf{X}$.

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

ii Proceed as in k-Means++.

4. Analysis

Denote the sum of k-th powers of modulus of $\mathbf{x} \in \mathbf{X}$ as $\mathcal{S}_k(\mathbf{X})$ or in short $\mathcal{S}_k$ when the context is clear. That is,

$$\mathcal{S}_k(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} \|\mathbf{x}\|^k$$

We also write $\sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}$ as $\mathcal{S}$. Then we can express the probability $p(\mathbf{x})$ in terms of $\mathcal{S}$ and $\mathcal{S}_2$.

$$\begin{aligned}
f(\mathbf{x}) &= \sum_{\mathbf{y}\in\mathbf{X}} \|\mathbf{x}-\mathbf{y}\|^2 \\
&= \sum_{\mathbf{y}\in\mathbf{X}} \left(\|\mathbf{x}\|^2 - 2\langle\mathbf{x},\mathbf{y}\rangle + \|\mathbf{y}\|^2\right) \\
&= n\|\mathbf{x}\|^2 - 2\langle\mathbf{x}, \sum_{\mathbf{y}\in\mathbf{X}} \mathbf{y}\rangle + \sum_{\mathbf{y}\in\mathbf{X}} \|\mathbf{y}\|^2 \\
&= n\|\mathbf{x}\|^2 - 2\langle\mathbf{x},\mathcal{S}\rangle + \mathcal{S}_2(\mathbf{X}) \\
\sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}) &= \sum_{\mathbf{x}\in\mathbf{X}} \left(n\|\mathbf{x}\|^2 - 2\langle\mathbf{x},\mathcal{S}\rangle + \mathcal{S}_2\right) \\
&= n\sum_{\mathbf{x}\in\mathbf{X}} \|\mathbf{x}\|^2 - 2\langle\sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x},\mathcal{S}\rangle + \sum_{\mathbf{x}\in\mathbf{X}} \mathcal{S}_2 \\
&= n\mathcal{S}_2 - 2\langle\mathcal{S},\mathcal{S}\rangle + n\mathcal{S}_2 \\
&= 2(n\mathcal{S}_2 - \langle\mathcal{S},\mathcal{S}\rangle)
\end{aligned}$$

Notice that this looks similar to $2n^2\sigma^2(\mathbf{X})$, except we have $\langle\mathbf{x},\mathbf{x}\rangle = \|\mathbf{x}\|^2$ instead of $\mathbf{x}^2$ and $\left(\frac{\langle\mathcal{S},\mathcal{S}\rangle}{n}\right)^2$ instead of $\left(\frac{\mathcal{S}}{n}\right)^2$. Analogously, we can say that $p(\mathbf{x})$ is equivalent to the ratio of variance explained by $\mathbf{x}$.

For a point $\mathbf{x}\in\mathbf{X}$, we choose the probability to be the variance explained ratio by $\mathbf{x}$. That is,

$$\begin{aligned}
p(x) &= \frac{f(\mathbf{x})}{\sum_{\mathbf{y}\in\mathbf{X}} f(\mathbf{y})} \\
&= \frac{n\|\mathbf{x}\|^2 - 2\langle\mathbf{x},\mathcal{S}\rangle + \mathcal{S}_2}{2(n\mathcal{S}_2 - \langle\mathcal{S},\mathcal{S}\rangle)}
\end{aligned}$$

Let us get back to variance based algorithm. Since our algorithm is a probabilistic one, we are going to take a look at the expected value of inertia. For a set of points $\mathbf{X}$ and a set of centers $\mathcal{C}$, we denote the inertia by $\mathcal{I}_\mathcal{C}(\mathbf{X})$.

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

If the context is clear, we may omit $\mathcal{C}$ and $\mathbf{X}$. For a fixed set of points $\mathbf{X}$, if the probability of $\mathbf{x}\in\mathbf{X}$ being chosen to be a center is $p(x)$ with respect to a set of centers $\mathcal{C}$, then the expected value of inertia $E[\mathcal{I}(\mathbf{X})]$ is

$$E[\mathcal{I}(\mathbf{X})] = \sum_{\mathbf{x}\in\mathbf{X}} p(x) \sum_{\mathbf{y}\in\mathbf{X}} \min(D(x), \|y-x\|)^2$$

We will use the following easily proven lemma.

Lemma 1 *For a set of points S, its centroid* $\mathbf{c}$ *and an arbitrary point a,*

$$\sum_{x \in S} \|x - a\|^2 - \sum_{x \in S} \|x - c\|^2 = |S| \cdot \sum_{x \in S} \|c - a\|^2$$

Next lemma was proven in (Ostrovsky et al., 2006, Lemma 3.2).

Lemma 2 *The first two centers lie in different cores with probability $1 - O(p)$ where $\rho = \Omega(\epsilon^{\frac{2}{3}})$ for some real number ϵ.*

Theorem 3 *For a set of points X, consider the set of optimal cluster centers $\mathcal{C}_m$ and an arbitrary set of cluster centers $\mathcal{C}$. If A is one of the clusters and a random point is added to $\mathcal{C}$ uniformly, then*

Proof We will use Abel's summation formula (Apostol, 1976, Theorem 4.2). From triangle inequality, we have

$$\begin{aligned} D(x) &\leq D(y) + \|x - y\| \\ D(x)^2 &\leq 2D(y)^2 + 2\|x - y\|^2 \end{aligned}$$

We also have

$$\begin{aligned} \nu_x(\mathcal{C}) &= \frac{\sum_{\mathbf{c} \in \mathcal{C}} \|c - x\|^4}{k} - \left(\frac{\sum_{\mathbf{c} \in \mathcal{C}} \|c - x\|^2}{k} \right)^2 \\ &\leq (M(x) - \mu)(\mu - D(x)) \end{aligned}$$

We can now analyze the expected value of inertia. ■

5. 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. 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. In order to check performance bias, we have used both normalization and standardization.

Algorithm	Average	Minimum	Time
k-Means	1603.09	1577.05	9.27s
k-Means++	1563.71	1518.9	9.4s
ORSS	1569.41	1530.46	9.44s
Var-based	1552.37	1510.91	9.69s

Table 1: Results on Cloud data set

Algorithm	Average	Minimum	Time
k-Means	57.74	43.89	2.26s
ORSS	48.25	43.8	2.28s
k-Means++	49.19	43.83	2.28s
Var-based	47.05	44.79	2.32s

Table 2: Results on IRIS data set

The data sets we have used are some popular ones: Iris data set, Mall customers data set, Airline clustering data set, Wine testing dataset. We did not duel too much on using too many data sets. 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 data sets.

For a particular data set, we have first normalized the data using min max scalar. Miligan et al. (1988) shows that using z-score to standardize the data is not favorable for clustering because it loses between-cluster variation. Then we have used PCA to remove linear dependency among variables. For every data set, we run the experiment a total of 100 times as mentioned before.

6. 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 data set, as mentioned before. 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 500 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 $k = 10$ clusters.

For comparing k-Means++ against k-Means++ improvement, here are the results.

Algorithm	Average	Minimum	Time
k-Means	180.37	174.85	1.71s
ORSS	183.13	174.89	1.73s
k-Means++	181.23	174.93	1.72s
Var-based	179.76	174.85	1.74s

Table 3: Results on mall customers data set

Algorithm	Average	Minimum	Time
k-Means	1120.07	1015.19	1.49s
k-Means++	1103.17	1014.54	1.49s

Table 4: Results on Wine dataset

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.