Abstract

In machine learning or data mining in general, clustering is one of the most popular methods to extract valuable insights into the data. It becomes even more important when the data is high dimensional that can provide information regarding user behavior or underlying structure or affinity towards a certain direction. KMeans is a popular clustering algorithm that aims to solve the clustering problem by reducing the sum of minimum squared distances from data points to the centers. It is well established by now that seeding centers is better than choosing centers uniformly. In this paper, we first express our concerns about the results presented in k-means++ paper. Then we suggest an improvement, discuss various types of seeding methods and compare their performances. We also establish an upper bound on the inertia function associated with kmeans clustering. The author does not claim any originality regarding the improvement because the author found out later that another paper already considered the improvement suggested in this paper and that kmeans++ is essentially the same idea. Nonetheless, the paper also serves as an updated survey on the comparison of seeding methods.

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. Clustering is still an active field of research as there are a lot of recent developments in this area including some ideas that involve deep learning. This is so primarily because of how useful clustering is in both research and professional areas. For example, companies regularly use clustering to identify their most valuable users (or vice-versa). There have even been cases where some unstructured data have been first clustered and then the corresponding labels have been used to train machine learning models. Roughly the problem of clustering from the point of view of KMeans 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

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

based on the centroid. This is known as Lloyd's algorithm Lloyd [14]. We repeat this process until no more change is possible. Lloyd's method was published much later. Forgy [9] essentially publishes Llyod's method before Lloyd so sometimes we even call it Lloyd-Forgy algorithm. Bradley and Fayyad [3] and Ostrovsky et al. [17] and Arthur and Vassilvitskii [2] take it one step further by choosing the initial centers with a probability. We intend to introduce other ways of initialization and compare their performances in terms of inertia, convergence speed and CPU time taken.

First, we will discuss some benchmarks for most of the well established algorithms in clustering. Then we 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 A Brief Comparison of Synthetic Data

There has been multiple surveys and a lot of work on `k-means` in the literature. For our context, probably the most relevant work is done in Celebi, Kingravi, and Vela [5]. However, most of the algorithms used in that paper are practically not very useful. The same was also noted by the authors themselves. They concluded that `k-means++` and its greedy version work better than most. They also mention that probabilistic algorithms perform better than deterministic ones. Therefore, our primary focus has been on non-deterministic algorithms in this paper. We take this chance to introduce the not so popular algorithm Ostrovsky et al. [17] (`ORSS`). `ORSS` was not considered in their experiments. As we will show, `ORSS` is actually a very relevant topic in this regard. Note that we do not consider Gaussian Mixture Model for the experiment because it has been noted to be significantly slow for practical purposes, see Patel and Kushwaha [18] for a detailed discussion.

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** (PCA, see Pearson [19], Hotelling [12]) . We opted for PCA in this paper. PCA

decomposes the existing data points into orthogonal[2] ones. If we use PCA 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.

As a refresher we will mention some well known algorithms and compare their performances for practical purposes[3]. Affinity Propagation by Frey and Dueck [10] discusses a similarity based clustering method but as we will show later, it is very slow and does not scale. But more importantly, the data points do not necessarily always converge as noted in this experiment. There are also hierarchical algorithms such as DBSCAN by Ester et al. [8], HDBSCAN by Campello, Moulavi, and Sander [4], and improvements on DBSCAN and HDBSCAN by McInnes and Healy [15]. OPTICS by Ankerst et al. [1] is a generalization of DBSCAN which aims to build some sort of reachability graph to understand the inner structure of the data points. But for large datasets, similar results can be obtained using HDBSCAN so we do not use OPTICS. Comaniciu and Meer [6] is a robust technique that tries to show convergence of underlying density function and performs relatively well but not as good as KMeans. A comparatively efficient version of Yu and Shi [21] by Damle, Minden, and Ying [7] discusses a spectral clustering based on Eigen decomposition although it struggles to scale. There is also BIRCH by Zhang, Ramakrishnan, and Livny [22] which is based on a tree called cluster feature tree. However, it also struggles with high dimensional data, specially if the dimension is over 20.

The summary of the following results is that, considering all factors, KMeans is the only realistic solution clustering any kind of data. It also offers the extra benefit for us to choose the number of clusters and that is also sometimes important. The practical implication is that some methods e.g. Affine Propagation

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

[3]The code used for this experiment is based on scikit-learn library and the code is a modified form of `https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py`

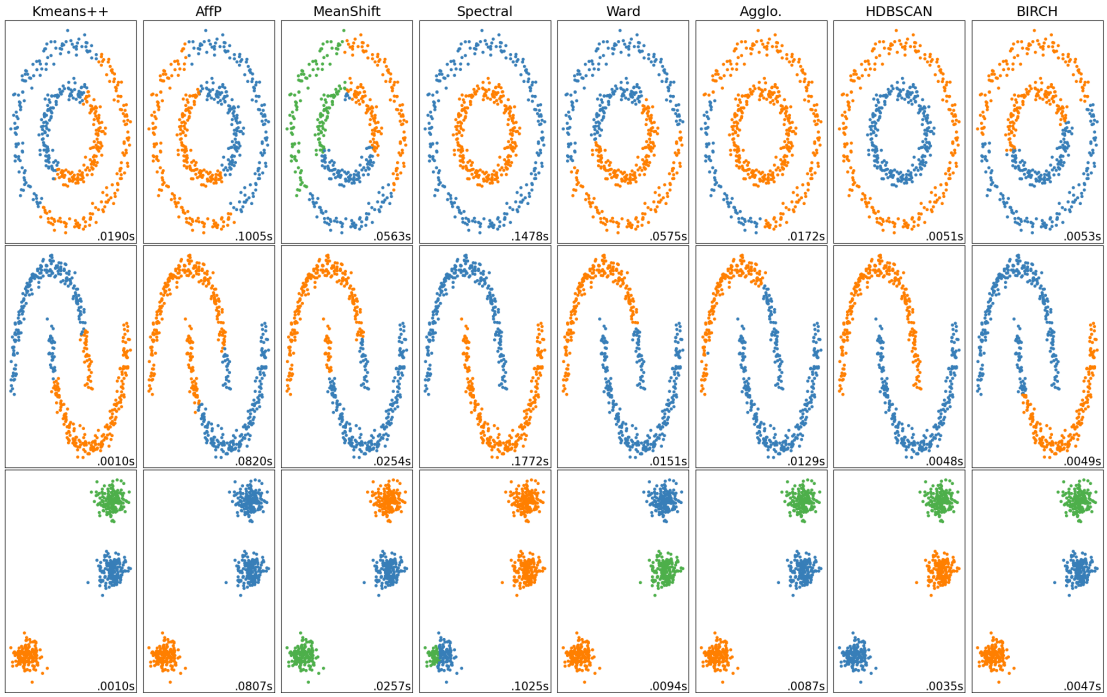

Figure 1: Time taken for 500 points

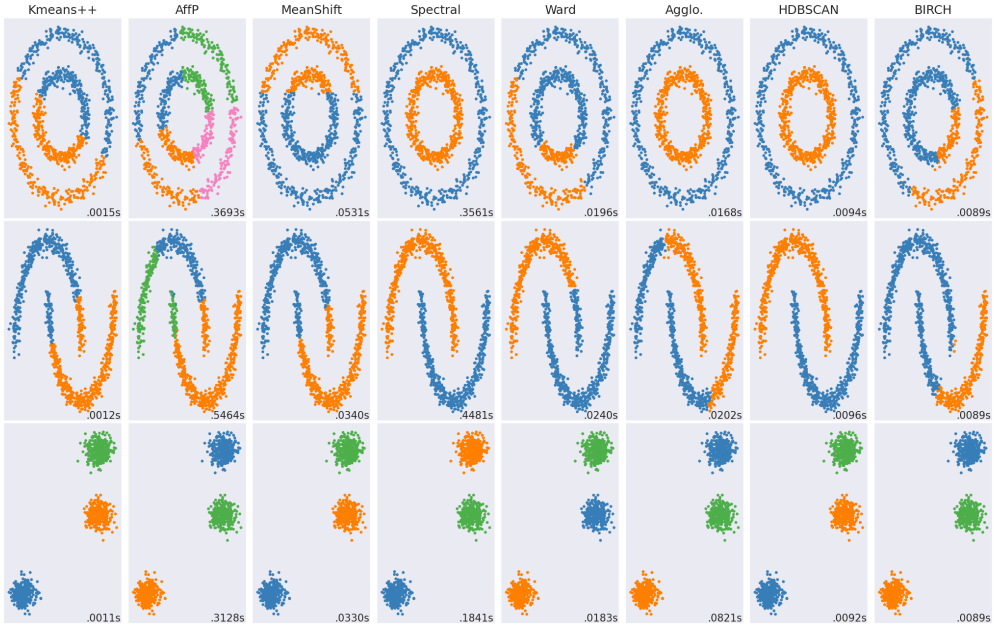

Figure 2: Time taken for 1000 points

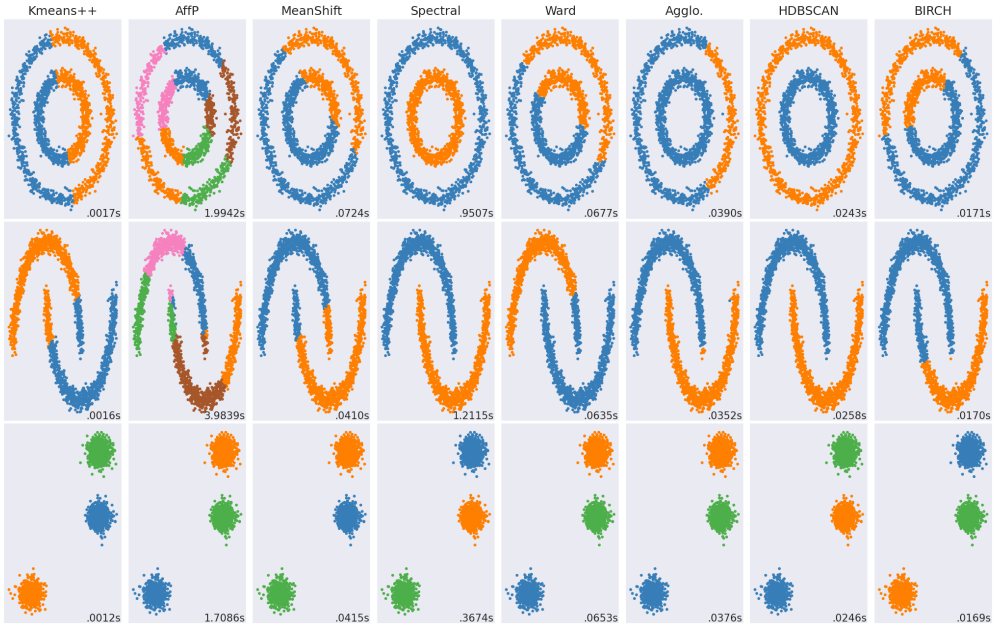

Figure 3: Time taken for 2000 points

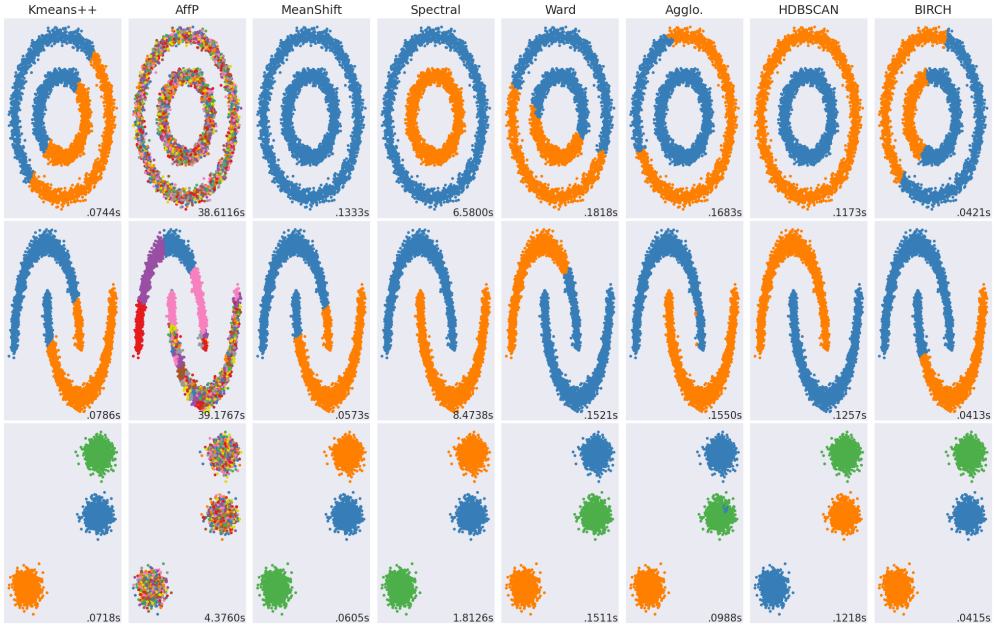

Figure 4: Time taken for 5000 points

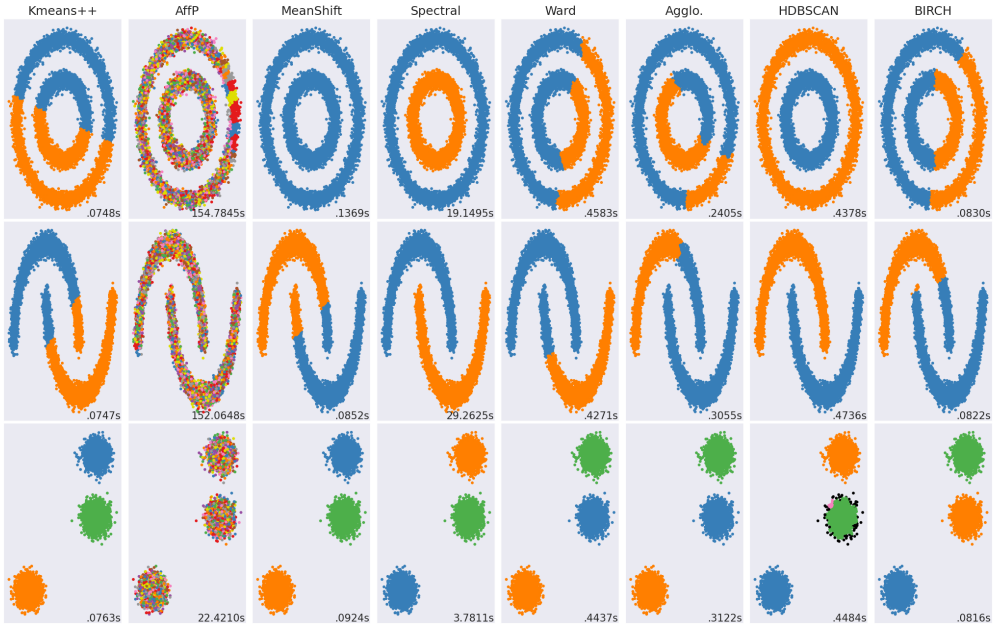

Figure 5: Time taken for 10000 points

3 Concerns Regarding `k-means++` Paper

Given how simple Lloyd's algorithm is, it seems very justified that it is the most popular algorithm for `k-means`. Ostrovsky et al. [17] is a great study on the unusual effectiveness of Lloyd's algorithm. Nowadays most of the times initial centers are chosen with a seeded probability rather than at random. This idea has gained much popularity in recent years. Even in the very well known scientific python package `scikit-learn` Pedregosa et al. [20], `k-means++` initialization is supported by default. And it is no surprise considering the theoretical and experimental results are presented in Arthur and Vassilvitskii [2]. For example, Arthur and Vassilvitskii [2] claim `k-means++` is $\log k$ competitive which 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, the validity of these results seems to be questionable.

3.1 `k-means++` and `ORSS`

The author remarks 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 and Vassilvitskii [2] call D^2 *weighting*. Now, `ORSS` chooses two centers x, y with probability proportional to $\|x - y\|^2$. 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`.

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 point out that our proposed method of improving `k-means++` is not actually a novel idea. This was already discussed in Ostrovsky et al. [17]. Regretfully, we did not know that at the time because we did not read the full paper back then. It was only after we found out that `k-means++` results could be wrong, we went through the `ORSS` paper again. When we read their paper for the second time, we found out that they had already mentioned a result Ostrovsky et al. [17, Page 4, section 3, Paragraph: Running Time] similar to ours in their paper. They say that choosing two centers x, y as in Ostrovsky et al. [17] is the same as choosing the first center c_1 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. Therefore, we argue that Ostrovsky et al. [17] is an improvement[4] over `k-means++`.

3.2 Accuracy of `k-means++` Results

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 may not be correct. And the other is that experimental results might not be correct either. This does not imply that `k-means++` is a bad algorithm. It still works really well, just that we do not think the results mentioned in the paper are correct.

First, let us discuss the theoretical concern. In Arthur and Vassilvitskii [2], $D(x)$ is assumed to be the minimum of squared distances from x to the centers $\mathcal{C}$. In Arthur and Vassilvitskii [2, Lemma 3.3, section 3], for two distinct point $a, a_0 \in A$, the

[4]To be more precise, this is exactly the improvement that we were trying to suggest

authors use *triangle inequality*[5] for expressing $D(a)^2$ in terms of $D(a_0)$ and $\|a - a_0\|$. Check the statement of Arthur and Vassilvitskii [2, Lemma 3.3].

Lemma 1. *Let A be an arbitrary cluster in $\mathcal{C}_{opt}$ and let $\mathcal{C}$ be an arbitrary clustering. If we add a random center to $\mathcal{C}$ from A, chosen with D^2 weighting, then $E[\phi(A)] \leq 8\phi_{opt}(A)$.*

The statement seems a bit vague to us for the reason mentioned below. Therefore, it is possible that we did not fully understand what they meant in this lemma. However, based on our interpretation, this lemma is wrong.

They claim in the proof that $D(a_0) \leq D(a) + \|a - a_0\|$. Our concern is whether this is necessarily true or not. First, they take an arbitrary cluster A from the optimal set of centers $\mathcal{C}_{opt}$. Then they choose a point a_0 from A with D^2 weighting to be added to $\mathcal{C}$, which is an arbitrary clustering. This is the part where our argument lies. Since a center is being added to $\mathcal{C}$ and not $\mathcal{C}_{opt}$, here $D(a)$ is the minimum distance from a to $\mathcal{C}$. If we consider $D(a) = \min(\|a - \mathcal{C}_{opt}\|)$, then the inequality $D(a_0) \leq D(a) + \|a - a_0\|$ holds true. However, in this case $D(a) = \min(\|a - \mathcal{C}\|)$ should be used instead since the center is being added to $\mathcal{C}$. By definition $D(x)$ is the minimum distance from x to the centers in $\mathcal{C}$ and not $\mathcal{C}_{opt}$. Therefore in the inequality, $D(a)$ and $D(a_0)$ are not distances from the same center and hence, it is not directly implied by triangle inequality. Although nitpicking, we would like to mention that the inequality $\dfrac{a^2 + b^2}{2} \geq \left(\dfrac{a+b}{2}\right)^2$ is not the *power mean inequality*. This inequality can be derived in many ways, including power mean inequality, but it itself is not the power mean inequality. For reference, power mean inequality (see Hardy, Littlewood, and G. [11]) states that for any $r \leq s$ and n real numbers $a_1, \cdots, a_n$,

$$\left(\frac{a_1^r + a_2^r + \cdots + a_n^r}{n}\right)^{\frac{1}{r}} \leq \left(\frac{a_1^s + a_2^s + \cdots + a_n^s}{n}\right)^{\frac{1}{s}}$$

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 and Vassilvitskii [2]. 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. [20]. We found out that our implementations achieve similar inertia to the one in `scikit-learn`.

[5]Triangle inequality states that for any triangle ABC, we have $\|AB\| + \|BC\| \geq \|AC\|$. This transforms into $|a| + |b| \geq |a + b|$ for real numbers, consequently $|a| + |x - a| \geq |x|$ holds as well.

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 do not match with ours. For cloud data set, both our and `scikit-learn` implementations have similar inertia values. The results are shown in tables 29 (min-max scalar used to process data), 27 (no pre-processing) and 28 (standard scalar pre-processing). But `k-means++` results Arthur and Vassilvitskii [2, Table 3] do not match with ours. We confirmed that our data set consisted of 1024 rows and had dimension 10. Moreover, we obtained the data set from UC-Irvine Machine Learning Repository which is the same source stated in Arthur and Vassilvitskii [2, Section 6.1].

We also tried collecting every data set that was 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* either. There were some similar data sets but they matched neither the number of data points (494019) nor the dimension (35). Therefore, cloud data set was our primary source of comparison. We would like to mention that in a footnote Arthur and Vassilvitskii [2, Section 6, page 8] mentioned a url that contained full test suite. However, the site seems to be no longer accessible even though the parent sites are. Therefore, we could neither retrieve the original data sets used in their paper nor check[6] their implementation.

4 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 $\mu_S = \dfrac{1}{|S|} \sum_{x \in S} x$.

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.

[6]The author contacted the authors of `kmeans++` in order to clarify whether it is a misunderstanding on the author's part or a calculation error on their end; however, did not receive any reply.

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

ii Repeat the remaining steps in `k-means++` [2, 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.

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.

5 `k-means++` Improvement

In this section, we discuss the motivation behind our idea of improving `k-means++`. 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 think of $p(x)$ as if it is the amount of variance explained by x. Also, this version of $p(x)$ is computationally much cheaper than $\sum_{\mathbf{y}\in\mathbf{X}} \|\mathbf{x}-\mathbf{y}\|^2$. Therefore, we considered $p(x)$ proportional to $\|x-\mu\|^2$ for choosing x as the first center in our experiments.

6 An Upper Bound on Inertia

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

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

This holds for any seeding technique. Thus, we get the following.

Theorem 1. *If S is a set of n points and C is a set of k centers, 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. Here is the list of data sets used.

i Boston housing data set

ii Wine quality testing data set

iii Mall customers data set

iv Airlines cluster data set

v Cloud data set

vi Two moons data set

vii Boston schools data set

viii Old faithful geyser data set

ix Iris data set

Milligan and Cooper [16] 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. We ran each experiment a total of 20 times. The average and minimum values of inertia, CPU time taken are reported in the result section.

8 Results

We have shown the results for $k \in \{5, 10, 25\}$ clusters in tables 1 to 26. As expected, default `k-means` is usually the fastest algorithm but also the worst in terms of optimizing inertia.

The comparison of inertia for `k-means++` seeding is showed in tables 27, 28, 29. We can check the results against Arthur and Vassilvitskii [2, Table 3]. See that even though we used both standard and min-max scaled data, their inertia values do not match with ours for any of the number of clusters in $\{10, 25, 50\}$. Since both our and `scikit-learn` implementations achieve similar inertia values, we believe our coding is correct. Also, since the cloud data set we used had exactly the number of dimension and rows as in Arthur and Vassilvitskii [2], we believe that this is also the correct data set. [7]

[7]The code and dataset are available at a Github repository which can be made available upon request.

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	5788610179951.84	5788610179951.84	1.68	1.35	34.15	28.0
`k-means++`	5895771708319.71	5724390573955.8	1.34	0.63	25.15	10.0
`orss`	5833118291028.73	5724390573955.8	1.29	0.68	23.9	11.0
`coc`	5788610179951.84	5788610179951.84	1.51	0.92	30.2	18.0

Table 1: Results on airlines data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	2812654.07	1475549.48	0.08	0.03	10.45	3.0
`k-means++`	1572564.77	1475549.48	0.07	0.05	7.25	3.0
`orss`	1547677.65	1475549.48	0.07	0.05	6.7	3.0
`coc`	1812654.49	1475612.32	0.07	0.04	8.15	4.0

Table 2: Results on boston data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	17740103.9	17700010.65	0.52	0.12	36.85	7.0
`k-means++`	17821421.19	17700010.65	0.32	0.11	20.05	4.0
`orss`	17980303.16	17699943.72	0.28	0.11	17.2	4.0
`coc`	17740103.9	17700010.65	0.51	0.2	35.75	13.0

Table 3: Results on cloud data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	59.34	50.33	0.02	0.01	8.25	4.0
`k-means++`	58.48	50.28	0.02	0.01	5.1	1.0
`orss`	56.69	50.28	0.02	0.01	4.65	2.0
`coc`	62.37	50.36	0.02	0.01	7.25	3.0

Table 4: Results on iris data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	84038.0	75412.6	0.02	0.01	8.0	3.0
`k-means++`	82360.19	75399.62	0.02	0.01	6.25	2.0
`orss`	82455.88	75399.62	0.03	0.02	6.7	3.0
`coc`	85772.29	75427.71	0.03	0.01	8.4	4.0

Table 5: Results on mall data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	22.8	18.89	0.01	0.01	5.5	3.0
`k-means++`	20.01	18.89	0.01	0.01	4.5	2.0
`orss`	19.78	18.89	0.01	0.01	4.35	2.0
`coc`	21.98	18.91	0.01	0.01	5.05	2.0

Table 6: Results on moons data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	2106.59	2028.44	0.03	0.01	6.85	2.0
`k-means++`	2162.48	2028.44	0.03	0.02	5.05	2.0
`orss`	2143.76	2036.83	0.03	0.02	4.9	2.0
`coc`	2164.2	2028.44	0.03	0.02	5.75	3.0

Table 7: Results on old data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	5911401430.11	5733432489.75	0.03	0.01	13.45	6.0
`k-means++`	6151075153.7	5733432489.75	0.02	0.01	9.75	3.0
`orss`	6037398253.1	5734713233.58	0.03	0.01	11.2	4.0
`coc`	6116023527.58	5733432489.75	0.02	0.01	9.6	3.0

Table 8: Results on schools data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	1005296.55	916424.19	0.03	0.01	10.5	4.0
`k-means++`	1005490.9	916424.19	0.02	0.02	6.25	3.0
`orss`	981833.05	916424.19	0.02	0.01	5.7	2.0
`coc`	1004998.14	916424.19	0.02	0.01	6.95	3.0

Table 9: Results on Wine data set, 5 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	5788610179951.84	5788610179951.84	1.91	1.3	35.25	23.0
`k-means++`	5841017195498.52	5724390573955.8	1.49	0.64	26.5	10.0
`orss`	5920823727383.38	5724390573955.8	1.36	0.68	22.1	9.0
`coc`	5788604697505.78	5788500531030.62	1.68	1.23	29.5	22.0

Table 10: Results on airlines data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	2717448.59	1500209.99	0.08	0.04	9.45	4.0
`k-means++`	1687335.04	1475549.48	0.08	0.05	7.2	3.0
`orss`	1782584.85	1475549.48	0.09	0.05	8.1	3.0
`coc`	1564391.58	1476556.31	0.09	0.05	9.7	5.0

Table 11: Results on boston data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	17700272.49	17700010.65	0.56	0.18	37.65	11.0
`k-means++`	17941909.09	17700010.65	0.34	0.13	19.5	5.0
`orss`	17941152.22	17700010.65	0.37	0.12	22.05	4.0
`coc`	17780988.12	17700010.65	0.47	0.12	30.4	6.0

Table 12: Results on cloud data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	58.41	50.33	0.02	0.01	6.05	2.0
`k-means++`	59.14	50.28	0.02	0.01	5.8	2.0
`orss`	56.36	50.28	0.02	0.01	4.6	2.0
`coc`	62.19	50.28	0.02	0.01	6.55	2.0

Table 13: Results on iris data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	84902.83	75399.62	0.04	0.02	7.9	4.0
`k-means++`	81352.0	75399.62	0.05	0.03	5.65	3.0
`orss`	81631.34	75399.62	0.05	0.03	8.65	3.0
`coc`	83759.84	75427.71	0.04	0.02	8.4	3.0

Table 14: Results on mall data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	23.62	18.91	0.02	0.01	5.7	4.0
`k-means++`	20.22	18.89	0.01	0.01	4.2	1.0
`orss`	20.46	18.89	0.01	0.01	4.7	2.0
`coc`	20.4	18.89	0.01	0.01	5.5	2.0

Table 15: Results on moons data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	2186.18	2028.44	0.04	0.01	7.1	2.0
`k-means++`	2222.88	2039.55	0.04	0.02	4.7	2.0
`orss`	2171.94	2028.44	0.03	0.02	3.5	1.0
`coc`	2188.5	2028.44	0.03	0.02	5.85	2.0

Table 16: Results on old data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	6036905058.56	5729423591.21	0.02	0.01	9.95	4.0
`k-means++`	6071879236.33	5728232615.59	0.02	0.01	7.95	2.0
`orss`	6152682962.47	5729423591.21	0.03	0.02	8.4	4.0
`coc`	6122042430.59	5739498515.77	0.03	0.01	11.85	4.0

Table 17: Results on schools data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	1010055.91	916424.19	0.03	0.01	9.2	3.0
`k-means++`	990859.67	916424.19	0.03	0.02	6.25	3.0
`orss`	1014711.02	916424.19	0.03	0.02	5.95	2.0
`coc`	1005327.62	916424.19	0.03	0.01	7.95	3.0

Table 18: Results on Wine data set, 10 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	1179535.38	726875.22	0.18	0.11	13.95	7.0
`k-means++`	796562.49	707943.36	0.17	0.12	8.55	4.0
`orss`	783434.35	708148.86	0.19	0.12	9.85	4.0
`coc`	836074.29	714810.01	0.22	0.12	15.65	7.0

Table 19: Results on boston data set, 25 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	7778602.59	6286432.16	1.13	0.39	46.5	15.0
`k-means++`	6175654.25	5754925.79	0.66	0.24	21.55	4.0
`orss`	6313890.31	5754925.79	0.79	0.34	27.85	7.0
`coc`	6684716.64	5754925.79	1.49	0.27	64.15	10.0

Table 20: Results on cloud data set, 25 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	31.57	27.59	0.03	0.02	6.85	4.0
`k-means++`	29.41	27.29	0.04	0.03	4.75	3.0
`orss`	30.38	26.84	0.04	0.03	5.8	3.0
`coc`	34.62	28.35	0.04	0.02	8.2	4.0

Table 21: Results on iris data set, 25 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	43066.3	37747.05	0.04	0.03	7.9	5.0
`k-means++`	40317.7	37581.02	0.05	0.05	6.1	4.0
`orss`	40096.13	37819.5	0.06	0.05	6.15	4.0
`coc`	42233.13	39118.08	0.05	0.03	8.15	5.0

Table 22: Results on mall data set, 25 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	10.26	7.6	0.02	0.01	7.05	4.0
`k-means++`	9.24	8.06	0.03	0.02	5.1	3.0
`orss`	8.78	7.59	0.02	0.02	4.25	2.0
`coc`	10.57	7.58	0.02	0.01	6.85	3.0

Table 23: Results on moons data set, 25 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	767.68	541.06	0.04	0.03	5.15	3.0
`k-means++`	599.9	545.47	0.06	0.05	3.65	2.0
`orss`	636.81	553.1	0.06	0.06	4.0	3.0
`coc`	747.26	556.26	0.05	0.03	6.05	2.0

Table 24: Results on old data set, 25 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	2780499617.96	2382345958.39	0.03	0.02	7.85	3.0
`k-means++`	2574515408.9	2346464028.7	0.04	0.02	6.1	2.0
`orss`	2564154859.1	2413558849.09	0.04	0.03	7.05	3.0
`coc`	2870371762.13	2436964942.43	0.03	0.02	8.75	4.0

Table 25: Results on schools data set, 25 clusters

Initialization	Avg. In.	Min. In.	Avg. T	Min. T	Avg. It.	Min. It.
`random`	341878.83	219710.39	0.05	0.02	10.3	4.0
`k-means++`	249461.66	224847.92	0.05	0.04	5.9	3.0
`orss`	242546.38	218112.55	0.05	0.04	7.05	4.0
`coc`	273645.86	234824.73	0.08	0.04	16.2	8.0

Table 26: Results on Wine data set, 25 clusters

Clusters	`sk-learn` Avg.	Our Avg.
5	17706689.573774982	17711385.61029025
10	5761674.929143367	6434641.098051261
25	2007444.7098438586	2255902.212313077
50	1099395.4420880969	1194727.658029051

Table 27: `k-mean++` results on cloud data set

Clusters	`sk-learn` Avg.	Our Avg.
5	2519.79170884024	2519.826060045223
10	1521.9262422548954	1510.9057079504264
25	817.4296830232086	833.2041864088023
50	519.1784696946825	543.6392985861094

Table 28: `k-means++` results on cloud data set (standardized)

Clusters	`sk-learn` Avg.	Our Avg.
5	57.096454766857384	57.302792493848216
10	32.929845052888986	32.93489472987519
25	17.941355048689374	18.50259622163847
50	11.362562342560503	11.7818373139353

Table 29: `k-means++` results on cloud data set (scaled)

9 Conclusion

We presented our arguments about `k-means++` results and discussed different seeding methods to improve `k-means` algorithms. Then we talked about different seeding techniques of center initialization and proved an upper bound of inertia. Finally, we showed comparison of seeding methods on 9 data sets for different number of clusters. Our observation is that `ORSS` is the most stable algorithm where both `k-means++` and `coc` often produce results on the extreme side. For a future improvement, a crucial idea could be to develop a deterministic method of seeding initial centers which would not only eliminate the uncertainty factor from KMeans algorithm but also provide better insights into the structure of the data.

Another very important consideration could be deep learning although it's still not concrete. Recently there have been some studies where deep learning has been used for clustering or clustering has been used as an extra embedding layer in deep learning (see Li et al. [13]).

Ethical Conduct As far as the author(s) is (are) concerned, this paper is not being considered for publication anywhere else nor does it break any ethical guidelines the author(s) can think of. No result is fabricated and code/data can be made available upon request.

References

[1] M. Ankerst et al. "OPTICS: ordering points to identify the clustering structure". In: *ACM SIGMOD Record* 28.2 (June 1999), pp. 49–60. DOI: 10.1145/304181.304187.

[2] D. Arthur and S. Vassilvitskii. "K-means++: the advantages of careful seeding". In: *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* 18 (2007), pp. 1027–1035.

[3] P. S. Bradley and U. M. Fayyad. "Refining Initial Points for K-Means Clustering". In: *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning* (July 1998), pp. 91–99. DOI: 10.5555/645527.657466.

[4] R. J. Campello, D. Moulavi, and J. Sander. "Density-based clustering based on hierarchical density estimates". In: *Advances in Knowledge Discovery and Data Mining* 7819 (2013), pp. 160–172. DOI: 10.1007/978-3-642-37456-2_14.

[5] M. E. Celebi, H. A. Kingravi, and P. A. Vela. "A comparative study of efficient initialization methods for the K-means clustering algorithm". In: *Expert Systems with Applications* 40.1 (2013), pp. 200–210. DOI: 10.1016/j.eswa.2012.07.021.

[6] D. Comaniciu and P. Meer. "Mean shift: A robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (May 2002), pp. 603–619. DOI: 10.1109/34.1000236.

[7] A. Damle, V. Minden, and L. Ying. "Simple, direct and efficient multi-way spectral clustering". In: *Information and Inference: A Journal of the IMA* 8.1 (June 2018), pp. 181–203. DOI: 10.1093/imaiai/iay008.

[8] M. Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Aug. 1996), pp. 226–231. DOI: 10.5555/3001460.3001507.

[9] E. Forgy. "Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classifications". In: *Biometrics* 21 (1965), pp. 768–780.

[10] B. J. Frey and D. Dueck. "Clustering by passing messages between Data Points". In: *Science* 315.5814 (2007), pp. 972–976. DOI: 10.1126/science.1136800.

[11] G. H. Hardy, J. E. Littlewood, and P. G. *Inequalities*. Cambridge University Press, 1934. ISBN: 0-521-35880-9.

[12] H. Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441. DOI: 10.1037/h0071325.

[13] B. Li et al. "DNC: A deep neural network-based clustering-oriented network embedding algorithm". In: *Journal of Network and Computer Applications* 173 (Sept. 2020). DOI: 10.1016/j.jnca.2020.102854.

[14] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/tit.1982.1056489.

[15] L. McInnes and J. Healy. "Accelerated hierarchical density based clustering". In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (Dec. 2017), pp. 33–42. DOI: 10.1109/icdmw.2017.12.

[16] G. W. Milligan and M. C. Cooper. "A study of standardization of variables in cluster analysis". In: *Journal of Classification* 5.2 (1988), pp. 181–204. DOI: 10.1007/bf01897163.

[17] R. Ostrovsky et al. “The effectiveness of Lloyd-type methods for the K-means problem”. In: *Journal of the ACM* 59.6 (2012), pp. 1–22. DOI: 10.1145/2395116.2395117.

[18] E. Patel and D. S. Kushwaha. “Clustering cloud workloads: K-means vs gaussian mixture model”. In: *Procedia Computer Science* 171 (2020), pp. 158–167. DOI: 10.1016/j.procs.2020.04.017.

[19] K. Pearson. “On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.

[20] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research*. 85th ser. 12 (2011), pp. 2825–2830.

[21] S. X. Yu and J. Shi. “Multiclass spectral clustering”. In: *Proceedings Ninth IEEE International Conference on Computer Vision* (2003). DOI: 10.1109/iccv.2003.1238361.

[22] T. Zhang, R. Ramakrishnan, and M. Livny. “BIRCH: an efficient data clustering method for very large databases”. In: *ACM SIGMOD Record* 25.2 (June 1996), pp. 103–114. DOI: 10.1145/235968.233324.