LOG-Means: Efficiently Estimating the Number of Clusters in Large Datasets

Manuel Fritz Michael Behringer Holger Schwarz

University of Stuttgart, Germany
{firstname.lastname}@ipvs.uni-stuttgart.de

ABSTRACT

Clustering is a fundamental primitive in manifold applications. In order to achieve valuable results, parameters of the clustering algorithm, e.g., the number of clusters, have to be set appropriately, which is a tremendous pitfall. To this end, analysts rely on their domain knowledge in order to define parameter search spaces. While experienced analysts may be able to define a small search space, especially novice analysts often define rather large search spaces due to the lack of in-depth domain knowledge. These search spaces can be explored in different ways by estimation methods for the number of clusters. In the worst case, estimation methods perform an exhaustive search in the given search space, which leads to infeasible runtimes for large datasets and large search spaces. We propose LOG-Means, which is able to overcome these issues of existing methods. We show that LOG-Means provides estimates in sublinear time regarding the defined search space, thus being a strong fit for large datasets and large search spaces. In our comprehensive evaluation on an Apache Spark cluster, we compare LOG-Means to 13 existing estimation methods. The evaluation shows that LOG-Means significantly outperforms these methods in terms of runtime and accuracy. To the best of our knowledge, this is the most systematic comparison on large datasets and search spaces as of today.

PVLDB Reference Format:
Manuel Fritz, Michael Behringer, Holger Schwarz. LOG-Means: Efficiently Estimating the Number of Clusters in Large Datasets. *PVLDB*, 13(11): 2118-2131, 2020.
DOI: https://doi.org/10.14778/3407790.3407813

1. INTRODUCTION

Clustering is a fundamental primitive for exploratory tasks. Manifold application domains rely on clustering techniques: In computer vision, image segmentation tasks can be formulated as a clustering problem [20, 39]. Documents may be clustered to support faster information access and retrieval [9, 27]. For business purposes, clustering may be used for grouping customers, for workforce management or for planning tasks [25, 35]. In biology, clustering is applied to study genome data amongst others [8].

Jain identified three main general purposes of clustering throughout these and many more application domains, which emphasize the exploratory power of clustering analyses [28]: (1) Assessing the structure of the data. Here, the goal is to exploit clustering to gain a better understanding of data, to generate hypotheses or to detect anomalies. (2) Grouping entities. Clustering aims to group similar entities into the same cluster. Thus, previously unseen entities can be assigned to a specific cluster. (3) Compressing data, i.e., to use the clusters and their information as summary of the data for further steps.

Due to their appealing runtime behavior in practice, k-center clustering algorithms [12], such as k-Means [31, 32], k-Medians [10, 29], or k-Mode [26] are commonly used [44]. However, the expected number of clusters k has to be provided prior to the execution of these algorithms. Especially for arbitrary, previously unknown datasets, estimating this number is a tremendous pitfall and requires particular caution. Wrong values for k lead to bad results regarding the above-mentioned purposes, i.e., wrong structurings, groupings or compressions are performed.

Several methods have been proposed to estimate the number of clusters in arbitrary datasets [11, 14, 15, 17, 24, 34, 36, 41, 43]. These estimation methods perform a clustering algorithm with varying values for k within a given search space $\mathcal{R}$ and subsequently evaluate the results. This search space $\mathcal{R}$ is defined by analysts. Here, we have to distinguish between experienced and novice analysts: Experienced analysts may use their strong domain knowledge to reduce $\mathcal{R}$ to a manageable size. In contrast, novice analysts typically lack in-depth domain knowledge and thus often define larger search spaces, because of the underlying uncertainty. Especially in today's Big Data era, where data characteristics are hard to grasp, this uncertainty becomes even more severe.

Existing estimation methods commonly use two strategies to explore the search space: (1) An exhaustive search in $\mathcal{R}$ is often conducted. A well-known estimation method following this search strategy is the elbow method [42]. (2) $\mathcal{R}$ can be explored in a non-exhaustive manner, i.e., stopping the search as soon as the clustering results of adjacent values for $k \in \mathcal{R}$ differ only marginally. Common methods following this approach are G-Means [24] or X-Means [34].

Especially for large search spaces, as they are often defined by novice analysts, these two search strategies lead to long runtimes for two reasons: (a) The runtime complexity

Proceedings of the VLDB Endowment, Vol. 13, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3407790.3407813

regarding the search space $\mathcal{R}$ lies in $\mathcal{O}(|\mathcal{R}|)$, i.e., in the worst case, a clustering algorithm needs to be executed for each value of $k \in \mathcal{R}$. (b) Each single execution is costly, since for example k-Means as concrete instantiation of a k-center clustering algorithm has a super-polynomial worst case runtime in the input size [5]. Because of these long runtimes, existing estimation methods are not suitable for analyzing large datasets or large search spaces. Hence, there is currently no feasible support for novice analysts in order to achieve appropriate clustering results in a reasonable time.

A common approach for similar estimation problems is to "throw more machines at the problem" and thus reducing the calculation time. Yet, this approach comes with the price of high costs for additional or more powerful hardware. A better approach is to engineer thorough solutions, which can work on existing hardware. Such an approach for finding well-performing parameters without an exhaustive search arose in the research area of hyperparameter optimization [18, 30, 40]. For supervised learning problems, these concepts were successfully applied [19]. However, it is not well-studied how to transfer this approach to unsupervised clustering problems as ground-truth labels are missing for unsupervised learning.

In this work, we propose LOG-Means, which is able to overcome the pitfalls of existing estimation methods. Similarly to existing methods, LOG-Means draws on individual clustering results from $\mathcal{R}$, yet aims for significantly fewer executions of a clustering algorithm until an estimation can be made. Thus, it is of particular interest for large datasets or large search spaces, as they might be defined by novice analysts.

Our contributions include:

- We analyze characteristics and procedures of existing estimation methods for the number of clusters in datasets, e.g., the elbow method, and discuss their advantages and pitfalls.
- We propose our new estimation method LOG-Means and discuss it in comparison to existing estimation methods. In particular, we illustrate how LOG-Means proceeds in a greedy manner to efficiently estimate the number of clusters in datasets.
- We analyze LOG-Means and demonstrate that it provides better estimates and scales sublinearly with the search space, thus being a strong fit for large datasets and large search spaces.
- Our comprehensive experimental evaluation shows that LOG-Means outperforms existing estimation methods regarding runtime and accuracy. To the best of our knowledge, this is the most systematic comparison, since we address more estimation methods, larger datasets and larger search spaces than related work.

The remainder of this paper is structured as follows: In Section 2, we present related work, i.e., existing estimation methods. We detail on the elbow method in Section 3 and unveil its most important characteristics. We reveal our approach LOG-Means in Section 4 and analyze it in detail. In Section 5, we discuss the results of our experimental evaluation of LOG-Means in contrast to existing estimation methods. Finally, we conclude this work in Section 6.

2. RELATED WORK

Estimation methods require a prior definition of the search space $\mathcal{R}$ for the expected number of clusters. $\mathcal{R}$ is a discrete range of values for $k \in \mathbb{N}$ where the actual number of clusters is expected to be in. Since clustering groups similar entities together, this search space is in the worst case $\mathcal{R} = [2; n-1]$, where n is the number of entities in the dataset. An experienced analyst may be able to significantly reduce the search space based on prior domain knowledge. However, especially for inexperienced users, advanced estimation methods are of paramount interest to efficiently estimate the number of clusters. Such estimation methods follow a common procedure of three steps: (1) Identify which parameter in $\mathcal{R}$ to execute next, (2) execute the clustering algorithm with the determined parameter, and subsequently (3) evaluate the result. Typically, k-Means is used in the second step due to its appealing runtime behavior in practice.

In general, estimation methods can be divided into different categories: They are either exhaustive, meaning they perform an exhaustive search and execute the clustering algorithm for each $k \in \mathcal{R}$, or they are non-exhaustive in the sense that they do not perform an exhaustive search. On the other hand, these methods work in an automated or in a semi-automatic manner, i.e., with user interaction. In the following, we present related work for these categories.

Exhaustive Estimation Methods. These methods execute a clustering algorithm for each $k \in \mathcal{R}$ and subsequently evaluate each result, e.g., according to a clustering validity measure [11, 14, 15, 17, 36, 41]. Finally, the best result is selected as an estimation for k. Estimation methods in this category mainly differ in the validity measures they use to evaluate the quality of a single clustering result.

Existing clustering validity measures focus on the compactness of the clusters, their separation or combinations thereof [21]. A common measure for the compactness of a cluster is the sum of squared errors (SSE). This measure denotes the sum of the variances of the resulting clusters. Hence, the smaller the SSE is, the more compact are the clusters. The silhouette coefficient [36] is a prominent example for a validity measure that focuses on the separation of the clustering results. It aims to measure for each entity the distance to the neighboring clusters and can thereby state whether the entity is correctly assigned or should be re-assigned. Combinations of the compactness and the separation are for example used by the Calinski-Harabasz Index [11], the Coggins-Jain Index [14], the Davies-Bouldin Index [15] and the Dunn Index [17]. They have in common to use quotients of the measured compactness and the separation of a clustering result in different variations.

Another area of validity measures arose from information theory. The Akaike Information Criterion (AIC) [1] and the Bayesian Information Criterion (BIC) [37] are commonly used information criteria. They consist of two terms: The first term measures the fitness of the model, whereas the second term is a penalty regarding the number of parameters in a model. The goal of the latter is to avoid overfitting (in terms of a too high value for k). This penalty term is larger for BIC than for AIC. Sugar and James proposed the so-called jump method [41]. This method proceeds by calculating the distortion of the resulting clusters. Subsequently, a rate distortion function is applied, similarly as proposed by Shannon [38]. Finally, this distortion allows to compare clustering results, analog to clustering validity measures.

Non-Exhaustive Estimation Methods. Non-exhaustive methods perform an ascending search in $\mathcal{R}$ and stop as soon as subsequent clustering results barely differ according to a certain evaluation criterion.

The idea of the gap statistic [43] is to compare the graph of $log(SSE)$ of the dataset with the graphs of so-called reference distributions. To provide these reference distributions, several datasets are created in a way that each entity lies between the minimum and maximum value across all features of the original dataset. The estimated number of clusters are in the area, where these two graphs have the largest difference. In order to limit the search space $\mathcal{R}$, the gap statistic relies on the notion of a standard error. The search stops as soon as the gap between two subsequent values for k changes less than this standard error or until a predefined upper bound is met (e.g., $max(\mathcal{R})$).

The X-Means [34] method starts by clustering the dataset into parent clusters with $k = min(\mathcal{R})$. Subsequently, each parent cluster is clustered with $k = 2$. This result is compared to the parent cluster according to a scoring criterion, such as AIC or BIC (see above). The split is accepted if the score of the child cluster is better. The resulting clusters are again clustered with $k = 2$ and evaluated according to the scoring criterion. These steps are repeated until a predefined upper bound for k is met or until the scoring criterion rates the child clusters worse than the parent clusters.

Similarly to that, G-Means [24] checks in each iteration for a Gaussian distribution of each child cluster with the Anderson-Darling test [3]. If the test is not successful, i.e., a child cluster does not follow a Gaussian distribution, k is increased by one per test. In contrast to X-Means, after each iteration, k-Means is executed over the whole dataset with the new value for k to refine the overall result. The iterations stop as soon as all clusters follow a Gaussian distribution or if an upper bound, such as $max(\mathcal{R})$, is met.

Semi-Automatic Methods. While all the estimation methods presented above are purely automated, there is also the group of semi-automated methods. Undoubtedly, the elbow method [42] is the most commonly used method of this group. Since it is an important basis for our new estimation method, we explain it in more detail in the following section.

As this overview shows, existing estimation methods rely on the excessive execution of a clustering algorithm. Exhaustive methods execute such an algorithm for each element in $\mathcal{R}$. Even non-exhaustive methods typically draw on an excessive execution of a clustering algorithm until they stop. Hence, these approaches are not feasible for large search spaces, since they are costly to perform. Our novel estimation method LOG-Means addresses this problem of exploring large search spaces by drawing on a more elaborated search strategy.

As of today, there are some examinations of existing estimation methods [4, 13, 16, 33]. However, they all focus solely on a few thousand entities, a small number of dimensions (around 10), and clusters (less than 10). Furthermore, they mostly consider small search spaces. Similar observations can be made for related work on existing estimation methods [24, 41, 43]. To the best of our knowledge, no comprehensive evaluations of estimation methods considering more voluminous datasets across several characteristics and larger search spaces are available. Hence, drawing conclusions about their performance on these data characteristics remains an open challenge.

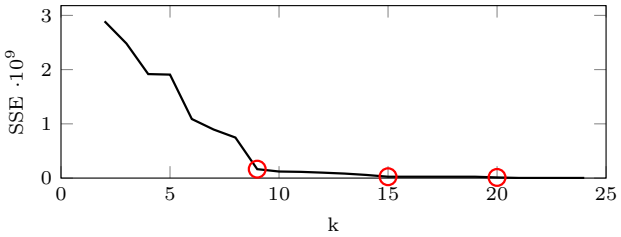


Figure 1: Elbow graph. Red circles depict possible bends that may be selected by analysts.

3. ELBOW METHOD

Before detailing on the elbow method, we briefly summarize the basics of k-center clustering algorithms. Let $\mathcal{X}$ be a dataset with n entities and d dimensions, i.e., $\mathcal{X} \subset \mathbb{R}^d$. The goal of k-center clustering algorithms is to group $\mathcal{X}$ into k disjoint clusters, such that each entity is assigned to the closest centroid $c \in \mathcal{C}$. As this problem is NP-hard [2, 23], several heuristics exist which aim to approximate the solution. One of these heuristics is the k-Means algorithm [31, 32]. The goal of k-Means is to find the set $\mathcal{C}$ of k centroids which minimizes the objective function in Equation 1.

$$\phi_{\mathcal{X}}(\mathcal{C}) = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} \|x - c\|^2 \quad (1)$$

Here, the Euclidean distance from an entity $x \in \mathcal{X}$ to the closest centroid $c \in \mathcal{C}$ is calculated. $\phi_{\mathcal{X}}(\mathcal{C})$ denotes the sum of these distances over all entities in $\mathcal{X}$. This sum is also called the sum of squared errors (SSE). Algorithms like k-Means move these k centroids to a better position in each iteration, until their position converges, i.e., until no more changes occur. In order to measure the quality of a clustering result, the SSE can be used. This validity measure denotes the variance of the resulting clusters, i.e., the smaller the SSE, the more compact the clusters.

3.1 Procedure

The elbow method was first discussed by Thorndike [42]. It comprises four steps: (1) Execute k-Means for each $k \in \mathcal{R}$, (2) calculate the SSE for each clustering result, (3) plot the results in a graph, and (4) select the bend in the graph. As the first two steps are straight-forward and do not differ from existing exhaustive estimation methods (cf. Section 2), we focus on the latter two steps. Figure 1 shows an example of an elbow graph as it is created in the third step.

Here, $k \in \mathcal{R}$ is depicted on the x-axis and the corresponding SSE values are shown on the y-axis. The intuition of this graph is to visually show after which $k \in \mathcal{R}$ the reduction of the SSE becomes negligible with an increasing value for k. This point can be seen as a "bend" in the graph, similarly to the bend of the human's elbow. That's why practitioners coined the name elbow method for this estimation method. It is the task of the analyst to select this bend in the fourth step (depicted in red in Figure 1). By doing so, the analyst prevents an overfitting of the clustering to the data in terms of a too high value for k, where no crucial additional benefits are achieved. In conclusion, the elbow method consists of automated parts (steps 1-3) and parts that require human interaction (step 4) in order to estimate the number of clusters in a dataset.

3.2 Discussion

One important characteristic of the elbow method is the usage of SSE as the clustering validity measure, since it can directly be derived from the goal of the objective function $\phi_{\mathcal{X}}(\mathcal{C})$ of the k-Means algorithm (cf. Equation 1). Other existing clustering validity measures mainly focus on combinations of compactness and separation of the clusters. Undoubtedly, these are important characteristics of clustering results, yet they require additional computations. Remember, that k-Means aims to reduce the variance (= sum of squared errors) of the resulting clusters. A similar objective function can be denoted for other k-center clustering algorithms as well. Hence, we argue that the SSE is generally more applicable than other validity measures, because it measures how well the objective function has performed.

Furthermore, the elbow method itself can easily be used by analysts with different domain knowledge background. Since solely the fourth step of selecting the bend in the elbow graph is performed by the analyst, the complexity is clearly manageable. The elbow method is also generally applicable, since finding the bend in the graph requires no profound technical knowledge.

Albeit these appealing advantages of the elbow method, there are also striking downsides. In the first place, it is an exhaustive method, i.e., it requires an exhaustive execution and subsequent calculation of the SSE for each $k \in \mathcal{R}$. As a consequence, the overall time until an estimation can be made is very large, in particular if $|\mathcal{R}|$ is large. Secondly, the manual part of the elbow method must be treated with caution. Several problems arise, if for example no clear bend can be seen or if multiple bends exist. The selection of a bend in a graph is the personal decision of an analyst, which potentially makes several selections possible (cf. circles in Figure 1). Hence, the estimated value for k depends on the personal and subjective analyst's selection. These pitfalls are also confirmed by our user study (cf. Section 5.4).

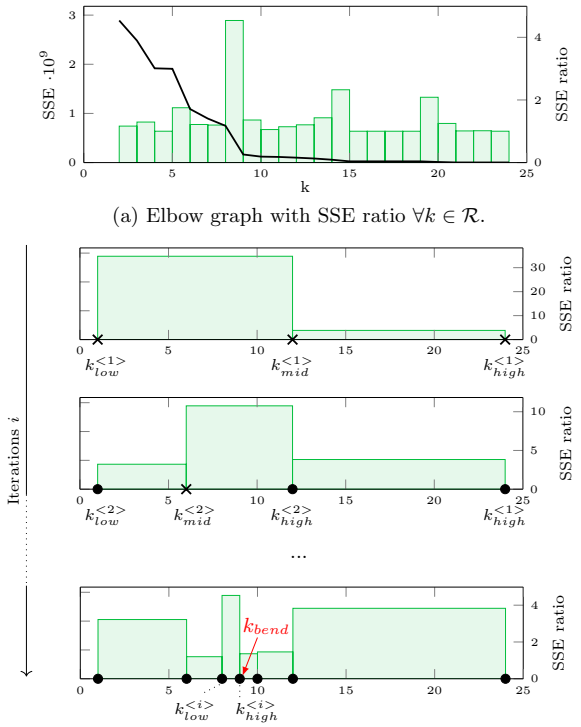
4. LOG-Means

As discussed in the previous section, the elbow method has severe pitfalls as well as promising advantages over existing estimation methods. We propose LOG-Means as a new estimation method. It aims to overcome the pitfalls of the elbow method, while exploiting its advantages.

4.1 Intuition

The elbow graph provides valuable properties, which can be exploited by specialized search strategies. The intuition of LOG-Means relies in particular on two specific properties of this graph. Since these properties are valid independent of datasets and the size of search spaces, LOG-Means preserves generality by exploiting these properties.

Property 1: In general, the sum of squared errors (SSE) follows a decreasing trend with an increasing value for k. This can be shown based on the objective function in Equation 1, assuming that a global optimum can be found by the clustering algorithm. We proof this property by induction, where the base case is $|\mathcal{C}| = |\mathcal{X}|$, i.e., we cluster with as many clusters as entities in $\mathcal{X}$. The goal of k-center clustering algorithms is to assign entities $x \in \mathcal{X}$ to centroids $c \in \mathcal{C}$, which are closer to c_i than to any other c_j with $c_i \neq c_j \in \mathcal{C}$. Hence, for the base case of the induction, each entity x is its own centroid c, thus having no errors regarding the objective function, i.e., $\phi_{\mathcal{X}}(\mathcal{C}) = 0$. Proving the induction step



(a) Elbow graph with SSE ratio $\forall k \in \mathcal{R}$.

(b) Procedure of LOG-Means for $i = 1, 2$ and the last iteration.

SSE SSE ratio Executing Already executed

Figure 2: Relation between SSE and SSE ratios is shown. LOG-Means iteratively halves areas with the highest ratios.

is trivial based on Equation 1: We argue that if we remove one centroid from an arbitrary $\mathcal{C}'$ with $1 < |\mathcal{C}'| \leq |\mathcal{X}|$, the SSE increases, since the distance between entities and their closest centroids increases. Turning these observations the other way around, we conclude that the SSE decreases with an increasing value for k. A more detailed discussion, which also addresses possible local optima of the clustering algorithm, is deferred to Section 4.3.

Property 2: The bend describes a significant change in the elbow graph. Thorndike describes the bend as the sudden drop of the SSE between two adjacent values for k [42], yet left a clear definition for the terms "sudden" and "drop" open. Following his statement on the visual representation of the bend, we formalize the decrease of the SSE as $SSEratio_k = SSE_{k-1}/SSE_k$. This ratio can be exploited to investigate the SSE throughout $\mathcal{R}$. The *most significant* bend is denoted by $max(SSEratio_k)$. Figure 2a shows the SSE ratio for all $k \in \mathcal{R}$ for an exemplary search space $\mathcal{R}$. Here, the highest SSE ratio is between $k = 8$ and $k = 9$, i.e., the most significant bend is at $k = 9$.

Putting both properties together, we aim to avoid an exhaustive search by calculating the SSE ratio (cf. property 2) for areas of non-directly adjacent values of $k \in \mathcal{R}$. Due to the decreasing character of the elbow graph (cf. property 1), these areas provide a meaningful insight of the SSE ratio. Hence, the SSE ratio can be used to iteratively shrink these areas in a greedy manner and subsequently find the bend efficiently without an exhaustive search in $\mathcal{R}$.

The general idea of LOG-Means is depicted in Figure 2b. For each iteration i, crosses show for which values $k \in \mathcal{R}$ a clustering algorithm is executed, whereas dots indicate where the SSE is available from previous iterations. After each execution of the clustering algorithm, the corresponding SSE value is calculated. Since the SSE ratio is a relative measure between two clustering results, it is updated whenever new adjacent SSE values for $k \in \mathcal{R}$ become available.

For $i = 1$, $k_{low}^{<1>} = min(\mathcal{R}) - 1$ and $k_{high}^{<1>} = max(\mathcal{R})$ and the clustering algorithm is executed for these two values. Note, that $min(\mathcal{R})$ must be reduced by one in order to keep the possibility to predict $min(\mathcal{R})$ as the value for k due to the definition of the SSE ratio. Next, the middle element $k_{mid}^{<1>}$ halves the area between $k_{low}^{<1>}$ and $k_{high}^{<1>}$. Subsequently, k-Means with $k = k_{mid}^{<1>}$ is executed and the SSE ratios of adjacent values of $k_{mid}^{<1>}$ are calculated, i.e., the ratios for $(k_{low}^{<1>}, k_{mid}^{<1>})$ and $(k_{mid}^{<1>}, k_{high}^{<1>})$. This allows to identify the next area with the highest SSE ratio. In Figure 2b, the highest ratio is found in the area of $(k_{low}^{<1>}, k_{mid}^{<1>})$. Hence, for $i = 2$, the ratios $(k_{low}^{<2>}, k_{mid}^{<2>})$, $(k_{mid}^{<2>}, k_{high}^{<2>})$ and $(k_{high}^{<2>}, k_{high}^{<1>})$ are calculated, where the latter is equal to $(k_{mid}^{<1>}, k_{high}^{<1>})$ (note the different y-axes). Note, that previously calculated ratios are kept, if they are not adjacent to $k_{mid}^{<i>}$. Subsequently, the same procedure is iteratively applied to the area with the highest ratio.

The search stops as soon as the low and high elements are directly adjacent. In this area, the SSE ratio in the elbow graph is expected to be the highest. We denote the value for k with the highest SSE ratio of k_{low} and k_{high} of the last iteration as k_{bend}, since we expect the *bend* here.

It can be seen that no exhaustive search in the search space is conducted. However, the idea is to approach the area of the bend from the left- and the right-hand side of the elbow graph by following principles from logarithmic search. To this end, the search space is efficiently narrowed down around the highest SSE ratio and the algorithm is executed with only few selected values from $\mathcal{R}$.

As our evaluation in Section 5 unveils, an optional additional step may further increase the accuracy. The reason for this is that k_{bend} may be only a local optimum. Yet, the global optimum, i.e., $max(SSERatio)$, is typically within a small ε environment. To this end, the ε environment around k_{bend} can be optionally evaluated in an exhaustive manner.

The general procedure of LOG-Means can also be applied to other k-center clustering algorithms, since they also minimize their notion of variance. Yet, we use k-Means, since it is the most commonly used algorithm of this family [44].

4.2 Algorithm

Algorithm 1 outlines the pseudo code for LOG-Means. The algorithm is separated into 5 parts:

In the first part, key-value data structures are defined (lines 2 and 3). These data structures keep track of already evaluated values within LOG-Means. $\mathcal{K}$ stores tuples of executed values for k and the corresponding SSE value. $\mathcal{M}$ stores tuples of k and the corresponding SSE ratio between k and the left adjacent value.

In the second part, i.e., from lines 4 to 7, k-Means is executed for k_{low} and k_{high}. The clustering results are evaluated according to the SSE measure and stored in $\mathcal{K}$.

The third part ranges from lines 8 to 20 and narrows down the search space around the estimated bend in the elbow graph. Here, the middle element is defined, k-Means is executed, the SSE is calculated and stored in $\mathcal{K}$ (lines 9-11). Subsequently, the SSE ratios are calculated in lines 12 and 13. These calculated SSE ratios are stored in $\mathcal{M}$ (lines 14 and 15). Since the area with the highest SSE ratio is halved in each iteration, the corresponding values in $\mathcal{M}$ are either stored or updated, if calculated previously. At the end of each iteration, new values for k_{low} and k_{high} are set in such a way that the area between these values localize the highest SSE ratio (lines 16 and 17). Subsequently, the SSE values for k_{low} and k_{high} are retrieved from $\mathcal{K}$ for the calculation of the SSE ratios in the next iteration (lines 18 and 19). The loop stops as soon as k_{low} and k_{high} are directly adjacent.

The fourth part ranges from lines 21 to 35 and is optional, if an $\varepsilon > 0$ environment is given. To this end, the highest SSE ratio between k_{low} and k_{high} is retrieved from $\mathcal{M}$ in line 22. As the bend is expected to be here, this point is called k_{bend}. Subsequently, the ε environment around k_{bend} is defined (lines 23 and 24). Within this ε environment, the SSE values are determined for each value of k. If available in $\mathcal{K}$, the corresponding SSE is retrieved, otherwise it will be calculated and stored in $\mathcal{K}$ (lines 27-32). Subsequently, the SSE ratio is calculated and stored in $\mathcal{M}$ (lines 33-34).

Finally, the algorithm provides an estimate in line 36 by selecting the value for k with the highest SSE ratio in $\mathcal{M}$. Note, that with $\varepsilon = 0$, this would be the same result as when ignoring the optional fourth step from lines 21 to 35.

Algorithm 1: LOG-Means

Input: $\mathcal{X}$ - dataset, k_{low} - minimum number of desired clusters, k_{high} - maximum number of desired clusters, ε - number of neighbors to evaluate
Output: k_{est} - estimated number of clusters for $\mathcal{X}$

```
1  k_low ← k_low − 1;
2  K ← ∅;
3  M ← ∅;
4  SSE_low ← SSE from k-Means with k_low;
5  K ← K ∪ {(k_low, SSE_low)};
6  SSE_high ← SSE from k-Means with k_high;
7  K ← K ∪ {(k_high, SSE_high)};
8  while ( k_low and k_high are not directly adjacent ) {
9      k_mid ← ⌊(k_high + k_low)/2⌋;
10     SSE_mid ← SSE from k-Means with k_mid;
11     K ← K ∪ {(k_mid, SSE_mid)};
12     ratio_left ← SSE_low/SSE_mid;
13     ratio_right ← SSE_mid/SSE_high;
14     M ← store or update {(k_mid, ratio_left)};
15     M ← store or update {(k_high, ratio_right)};
16     k_high ← k with highest ratio from M;
17     k_low ← left adjacent value of k_high from K;
18     SSE_high ← SSE for k_high from K;
19     SSE_low ← SSE for k_low from K;
20 }
21 if ε > 0 then
22     k_bend ← k ∈ [k_low, k_high] with highest ratio in M;
23     k_low ← k_bend − ⌊ε/2⌋;
24     k_high ← k_bend + ⌊ε/2⌋;
25     for ( ∀k ∈ [k_low; k_high] ) {
26         SSE_k_prev ← SSE of k_prev from K;
27         if k ∈ K then
28             SSE_k ← SSE for k from K;
29         else
30             SSE_k ← SSE from k-Means with k;
31             K ← K ∪ {(k, SSE_k)};
32         end
33         ratio_k ← SSE_k_prev/SSE_k;
34         M ← store or update {(k, ratio_k)};
35     }
36 k_est ← k ∈ [k_low, k_high] with highest ratio in M;
37 return k_est;
```

4.3 Analysis

Property 1 states that the elbow graph follows a decreasing trend for increasing values of $k \in \mathcal{R}$. However, as k-Means is solely a heuristic to the NP-hard k-center clustering problem, the objective function in Equation 1 may comprise local optima, i.e., the centroids are not at the globally best position. Hence, the SSE does not necessarily decrease monotonously. Yet, it has been proven that enhanced initialization algorithms, such as k-Means++ [6] or k-Means‖ [7], provide an $\mathcal{O}(log\ k)$-approximation to the optimal clustering result w.r.t. the error (SSE) from the objective function in Equation 1 independent of dataset characteristics.

Despite these local optima, the SSE ratio (property 2) still provides crucial insights into how the elbow graph changes between two values for $k \in \mathcal{R}$. The $\mathcal{O}(log\ k)$-approximation becomes more noticeable the closer the values for $k \in \mathcal{R}$ are evaluated: For far-distant values for k, the approximation error has only slight impact on the SSE ratio, thus predicting large areas with high SSE ratios mostly correct. Hence, LOG-Means can efficiently narrow down large search spaces around areas with high SSE ratios.

For closer distant values for k, especially for direct neighbors of $k \in \mathcal{R}$, the SSE and therefore the SSE ratio can be more strongly influenced by local optima of the clustering algorithm. This can be seen for example in Figure 2a at $k \approx 5$, where the SSE does not tend to decrease between two subsequent values for k, before it decreases rather strong at $k = 6$. Since LOG-Means keeps all areas and compares them in each iteration regarding their SSE ratio, we argue that using these "false" elbows for an estimation is rather unlikely. Yet, since the areas become smaller in each iteration, LOG-Means becomes most sensitive towards the $\mathcal{O}(log\ k)$-approximation in the last iteration, resulting solely in a minor deviation from the optimum number of clusters. We argue and show in the evaluation in the next section that with the $\mathcal{O}(log\ k)$-approximation of state-of-the-art approaches of k-Means, LOG-Means is able to provide reasonable accurate estimates, yet no perfect estimates in every scenario. By analyzing an optional ε environment around the expected bend, the effect of the local optima of the clustering algorithm can be further reduced (cf. Section 5.5).

Complexity Analysis

Estimation methods typically proceed in three steps: (1) Identifying which parameter to consider next, (2) executing k-Means with the determined parameter, and subsequently (3) evaluating the result. As discussed in Section 2, existing estimation methods typically perform an ascending or even worse an exhaustive search in the search space $\mathcal{R}$. Hence, the complexity of existing estimation methods lies in $\mathcal{O}(|\mathcal{R}|)$. Furthermore, the evaluation step can be costly due to a complex metric, which may lead to an even worse complexity class. Hence, these strategies require a huge overall runtime until an estimation can be made. On the contrary, LOG-Means promises a better runtime behavior regarding $\mathcal{R}$. To analyze this complexity, we focus on LOG-Means with $\varepsilon = 0$ and address the above mentioned three steps.

(1) Identifying which parameter to execute next can be done in $\mathcal{O}(1)$ when exploiting matching data structures. In Algorithm 1, these observations apply to lines 16 and 17, where the highest SSE ratio is identified. The middle element in this area is calculated in line 9, which can also be done in $\mathcal{O}(1)$, since it is only an arithmetical division.

(2) Executing k-Means. Due to the principle of logarithmic search, only $\mathcal{O}(log|\mathcal{R}|)$ executions of the clustering algorithm are required. However, as we do not eliminate areas with lower SSE ratios, LOG-Means could proceed with logarithmic search in eliminated and non-eliminated areas within each iteration. That is, $\mathcal{O}(log|\mathcal{R}| + log|\mathcal{R}|) = \mathcal{O}(2 * log|\mathcal{R}|)$ executions of k-Means are performed at highest, which can be reduced to $\mathcal{O}(log|\mathcal{R}|)$ again.

(3) Evaluating a single clustering result via the SSE metric can be done in linear time for a single clustering result with $k = |\mathcal{C}|$ clusters, because the SSE depends linearly on the number of clusters in the dataset (cf. Equation 1), i.e., the complexity lies in $\mathcal{O}(k)$. However, since solely $\mathcal{O}(log|\mathcal{R}|)$ clustering results are evaluated, the complexity of evaluating the results also lies in $\mathcal{O}(log|\mathcal{R}|)$.

Concluding, the overall complexity of LOG-Means is $\mathcal{O}(1 + log|\mathcal{R}| + log|\mathcal{R}|)$, which can be reduced to $\mathcal{O}(log|\mathcal{R}|)$.

5. EVALUATION

The purpose of our comprehensive evaluation is to systematically compare LOG-Means with existing methods to estimate the number of clusters in datasets regarding their runtime and accuracy. As estimation methods are divided into exhaustive and non-exhaustive methods (cf. Section 2), we compare LOG-Means to commonly used approaches of both categories. We also conducted a user study to evaluate the semi-automatic elbow method, since we exploit its properties for LOG-Means.

This section is structured as follows: Firstly, we discuss the setup. Secondly, we investigate the runtime as well as the accuracy of various estimation methods on synthetic datasets. Thirdly, based on the results of our user study, we compare LOG-Means to the elbow method in more detail. Fourthly, we discuss the impact of different ε environments for LOG-Means. Finally, we present results on several real-world datasets from various domains.

5.1 Experimental Setup

In this section, we present the hardware and software setup for our experiments. Furthermore, we detail the characteristics of the synthetic datasets, before we discuss the implementation and details of the experiments. Finally, we explain the details of the user study that we conducted to evaluate the performance of the elbow method.

Hardware and Software

We conducted all of our experiments on a distributed Apache Spark cluster. This cluster consists of one master node and six worker nodes. The master node has a 12-core CPU with 2.10 GHz each and 192 GB RAM. Each worker has a 12-core CPU with 2.10 GHz each and 160 GB RAM. Each node in this cluster operates on Ubuntu 18.04. We installed OpenJDK 8u191, Scala 2.11.12 and used Apache Hadoop 3.2.0 as well as Apache Spark 2.4.0.

Synthetic Datasets

Existing works on methods to estimate the number of clusters in datasets focus on rather small datasets [11, 14, 15, 17, 34, 36, 41, 43]. They rely on synthetic datasets with different numbers of entities (up to 36,000), dimensions (up to 10), and clusters in the dataset (up to 150, however for small datasets). Furthermore, the distribution of the used datasets varies: some use a Gaussian distribution for each

Table 1: Characteristics of the used 27 synthetic datasets.

Dataset	n	d	c
I - III	10,000	10	{10; 50; 100}
IV - VI	10,000	50	{10; 50; 100}
VII - IX	10,000	100	{10; 50; 100}
X - XII	100,000	10	{10; 50; 100}
XIII - XV	100,000	50	{10; 50; 100}
XVI - XVIII	100,000	100	{10; 50; 100}
XIX - XXI	1,000,000	10	{10; 50; 100}
XXII - XXIV	1,000,000	50	{10; 50; 100}
XXV - XXVII	1,000,000	100	{10; 50; 100}

cluster, others create a 2-dimensional dataset and create the clusters manually by placing the entities close to each other.

For our evaluation, we conduct a comprehensive comparison across many existing estimation methods that considers more voluminous datasets and is more systematic with respect to varying dataset characteristics. For this purpose, we implemented a synthetic dataset generator. This tool generates datasets based on the following input parameters: n as the number of entities, d as the number of dimensions, and c as the number of clusters, where each cluster contains n/c entities. Our tool generates datasets with values that lie within the range $[-10; 10]$ for each dimension. Each cluster has a Gaussian distribution with the mean at the center and a standard deviation of 0.5. The c centers are chosen randomly and the clusters are non-overlapping. Table 1 depicts the characteristics of the 27 datasets used for the evaluation.

Implementation

Besides LOG-Means, we implemented several estimation methods on Apache Spark. This includes very commonly used exhaustive and non-exhaustive estimation methods as described in Section 2. Table 2 summarizes the 14 methods that we used throughout the evaluation as well as their abbreviations that we use for referring to these methods. Since some estimation methods draw on parameters, we used recommendations provided by the authors of the respective estimation method, where available. The names of the parameters cling to the definition of the corresponding authors and can be found in Table 2. For BIC and X-Means, we used an existing implementation[1]. Since multiple scoring criteria can be used for X-Means, we used the Akaike Information Criterion (XAI) and the Bayesian Information Criterion (XBI) separately. We used Spark's variation of the Silhouette coefficient[2]. Regarding LOG-Means, we set $\varepsilon = 0$, i.e., only logarithmic search is performed.

For each run of k-Means, we used the default Apache Spark implementation. That is, we used k-Means‖ [7] for the initialization due to its proven $\mathcal{O}(log\ k)$-approximation and performed at most 20 iterations. Further improvements of the clustering algorithm in an exploratory setting [21, 22] are out of scope. If an estimation method failed to provide an estimation within a predefined time budget of 30 minutes, we stopped the execution and mark the corresponding estimation as failed. This time budget solely includes the runtime for identifying which parameter to execute and to evaluate the clustering result, and not the runtime for the repetitive execution of k-Means. We performed all runs three times and present median values in the results.

[1] `https://git.io/Je1sm` [2] `https://git.io/JfnPa`

Table 2: Overview of estimation methods for the evaluation.

Abbr.	Name	Parameters	Characteristic
ELM	Elbow Method [42]		exhaustive, semi-automatic
AIC	Akaike Information Criterion [1]		exhaustive, automatic
BIC	Bayesian Information Criterion [37]		
CHI	Calinski-Harabasz Index [11]		
CJI	Coggins-Jain Index [14]		
DBI	Davies-Bouldin Index [15]		
DUI	Dunn Index [17]		
JUM	Jump Method [41]	$Y = r/2$	
SIL	Silhouette coefficient [36]		
GAP	Gap Statistic [43]	$b = 5$	non-exhaustive, ascending, automatic
GME	G-Means [24]	$\alpha = 0.0001$	
XAI	X-Means (AIC) [34]		
XBI	X-Means (BIC) [34]		
LOG	LOG-Means	$\varepsilon = 0$	non-exhaustive, logarithmic, automatic

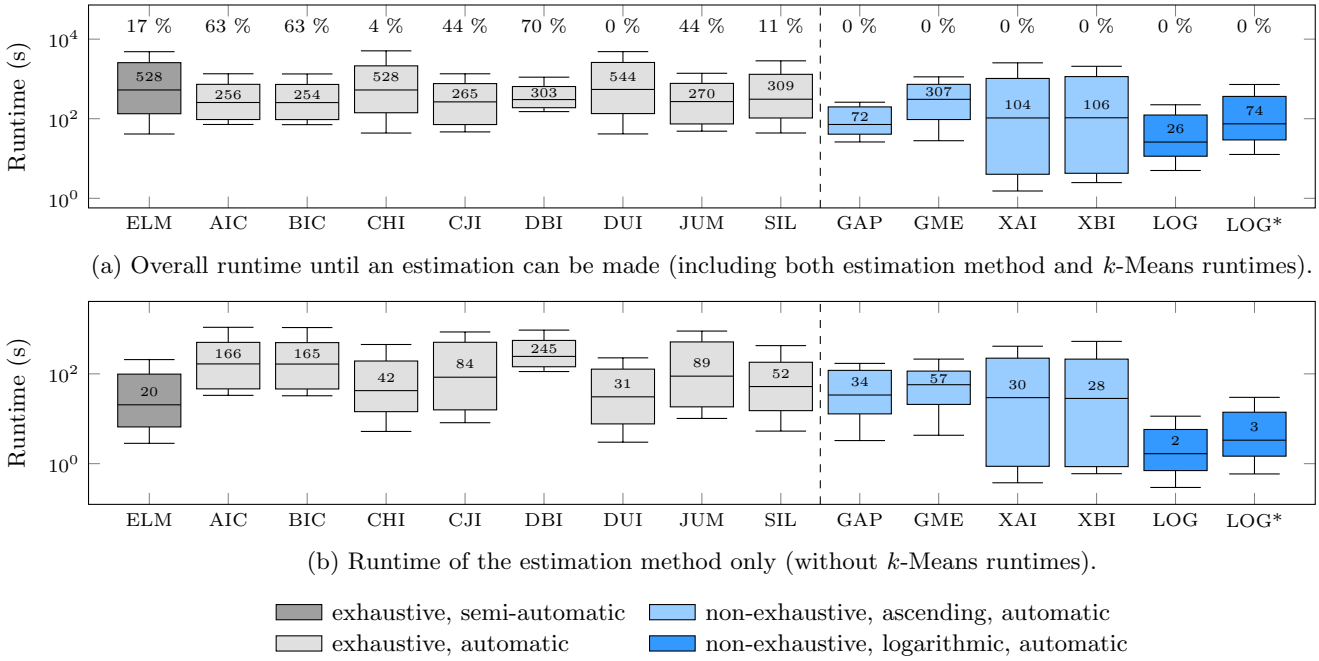
Regarding the search space, we conducted two different experiments. For each experiment, we will present the runtime measurements as well as the accuracy results.

Experiment 1: The goal of this experiment is to simulate rather strong domain knowledge of the analyst. Based on this domain knowledge, the analyst is able to drastically reduce the search space. We simulate this by setting the search space $\mathcal{R}$ to $[2; 2c]$ for all estimation methods, where c denotes the actual number of clusters in a dataset. In total, we performed more than 1,100 runs (= 27 datasets × 14 estimation methods × 3 repetitions).

Experiment 2: Here, we simulate less prior knowledge of the analyst, as it is typical for novice analysts. The goal is to demonstrate the benefits of LOG-Means for rather inexperienced analysts, who have only little domain knowledge and can therefore limit the search space only very vaguely. Therefore, we set the search space $\mathcal{R}$ to $[2; 10c]$ and perform solely LOG-Means with $\varepsilon = 0$. In total, 81 runs (= 27 datasets × 1 estimation method × 3 repetitions) are performed. The results for experiment 2 are marked with an asterisk (*) in the presentation of the results.

User Study on the Elbow Method

As described in Section 3, the elbow method consists of two parts: an automatic part and a manual part. We conducted the automatic part of the elbow method and presented the elbow graph to participants of a user study. Since we performed each estimation method three times, we also performed the automatic steps 3 times. Subsequently, we presented 81 graphs (27 datasets × 3 repetitions) to the participants. Their task was to select the most significant bend in the elbow graph. We captured the selected bend and measured the time until the selection was completed. If the participants did not see a significant bend, they could skip and proceed with the next elbow graph. The graphs appeared in a random order for each participant. For this user study, we had 15 participants, which mostly had a background in computer science. As their task was simply to select a bend in the elbow graph, we considered also non-technical participants as suitable for this user study.



(a) Overall runtime until an estimation can be made (including both estimation method and k-Means runtimes).

(b) Runtime of the estimation method only (without k-Means runtimes).

Figure 3: Runtime of successful estimations across all synthetic datasets. Median runtimes are depicted in box plots. Note the different logarithmic y-axes. Percentages in (a) depict the relative amount of failed estimates within the given time budget. The dashed line depicts the separation of exhaustive estimation methods (left) and non-exhaustive estimation methods (right).

5.2 Runtime

Figure 3 shows the runtime for each estimation method over all datasets. Figure 3a shows the overall runtime until an estimation was made. This includes the necessary executions of k-Means. The runtime of the core steps of the estimation method is presented in Figure 3b. This comprises solely the runtime for evaluating the clustering result for a specific parameter as well as the runtime for determining the next parameter to be used for executing k-Means. Details of both steps depend on the respective estimation method.

For the elbow method ELM, we included the median runtimes over all participants of the user study. Regarding the estimation runtime of ELM in Figure 3b, we aggregate the runtime of the calculation of the SSE as well as the selection of the bend via the participants and present the median values. We make the following five observations:

1) Exhaustive estimation methods require in general more runtime until an estimation can be made than non-exhaustive estimation methods (cf. Figure 3a). The median runtime of exhaustive methods range from 254 seconds (BIC, without failed estimations) to 544 seconds (DUI). On the other hand, non-exhaustive estimation methods range from 26 seconds (LOG) to 307 seconds (GME). These runtime savings were expected, since non-exhaustive estimation methods do not perform k-Means for each $k \in \mathcal{R}$.

2) Regarding the runtime of the core steps of an estimation method, we make a similar observation. The general trend is that exhaustive estimation methods require a longer runtime than non-exhaustive methods. However, the differences are not as drastic as for the first observation. That is, for exhaustive methods, the runtime of the estimation method itself ranges from 20 seconds (ELM) to 245 seconds (DBI). Non-exhaustive methods are faster, i.e., they range from 2 seconds (LOG) to 57 seconds (GME). The reason is that non-exhaustive methods do not need to evaluate the result for each clustering result in the search space. Yet, they can draw on more complex evaluation metrics or employ complex strategies on how to select the next value for $k \in \mathcal{R}$, which can be time consuming.

3) The relative amount of failed estimations due to the given time budget is depicted at the top of Figure 3a. These numbers clearly show that only exhaustive estimation methods were regularly not able provide an estimation within this budget. The reason for this is that exhaustive methods - in contrast to non-exhaustive methods - evaluate each result in $\mathcal{R}$, which requires long runtimes. Figure 4 details this observation. This figure shows the overall runtime until an estimation was made for several datasets. It can be seen that for more entities n and larger search spaces (due to the higher number of clusters c in the dataset), an increasing number of exhaustive estimation methods fail to provide an estimate in the given time frame. Especially for the largest dataset XXVII in Figure 4h, only 7 out of 14 methods are able to provide an estimate within the given time frame.

4) Our new estimation method LOG-Means is the fastest method in regard to the median values for the overall runtime (cf. Figure 3a) as well as for the runtime of the core steps of the estimation method (cf. Figure 3b). Regarding the latter, LOG-Means benefits from the SSE validity measure that can be calculated very fast. Furthermore, we argue that the computational overhead of the method itself with 2 seconds in median is negligible compared to the overhead of other estimation methods (cf. Figure 3b). Figure 4 shows that for a few specific datasets LOG-Means is not the fastest method with respect to the overall runtime. This is for example true for datasets III, IX, XXI, and XXVII. For these datasets, the X-Means variants XAI and XBI or

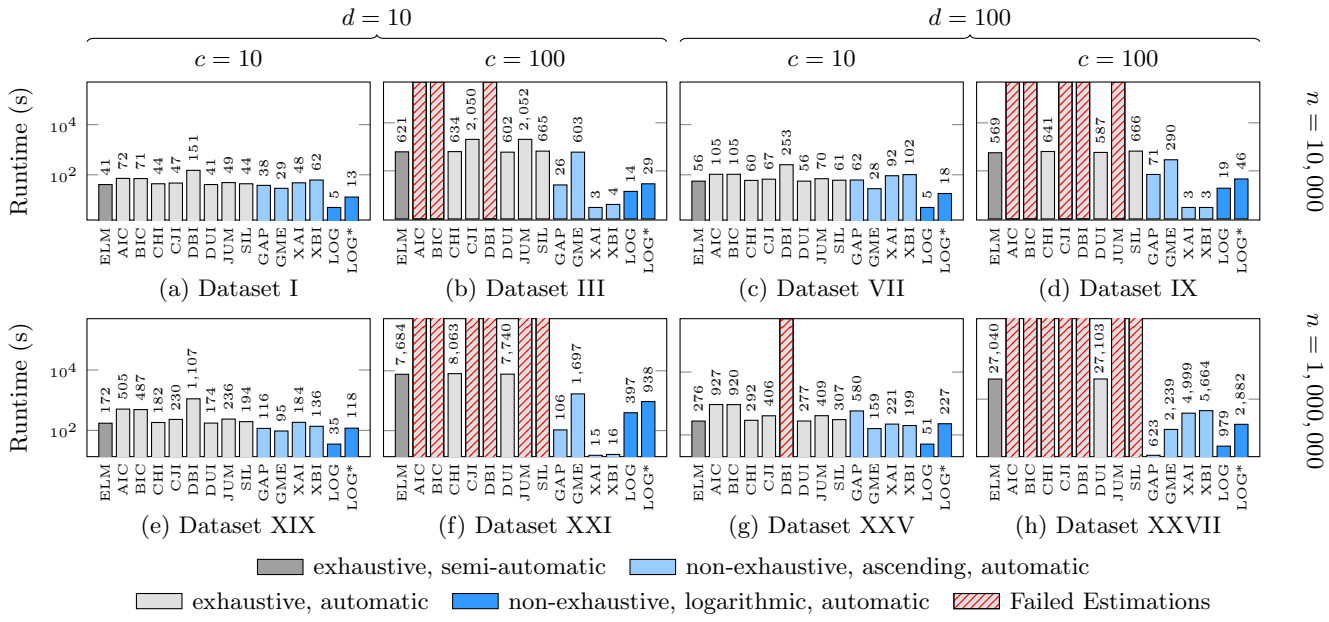


Figure 4: Overall runtime until an estimation was made on selected synthetic datasets with varying values for n, d and c.

GAP achieved faster estimates. However, as the later evaluation of the accuracy of the methods will reveal, the faster estimates of XAI, XBI and GAP are very inaccurate.

5) Comparing the results of LOG-Means from experiment 1 and experiment 2, we observe that the runtime roughly triples (see results for LOG and LOG* in Figure 3a), whereas the search space is increased by a factor of 5. This emphasizes the benefits of the logarithmic complexity of LOG-Means. Hence, we conclude that LOG-Means can also be employed in large search spaces, such as they might be defined by novice analysts.

5.3 Accuracy

Since we created synthetic datasets, the actual number of clusters c for each dataset is known. For our evaluation, we exploit this and define the accuracy as relative error $\delta k = (k_{est} - c)/c * 100$, where k_{est} denotes the estimation provided by the respective estimation method. This notion of the relative error allows us to identify to what extent certain estimation methods tend to under- or overestimate the number of clusters. Note, that δk solely addresses successful estimations, i.e., estimations within the given time budget.

Figure 5 summarizes the results across all investigated estimation methods and datasets. It is evident that some estimation methods regularly fail to provide an estimate within the given time frame. These methods are AIC, BIC, CJI, DBI and JUM. While some of them were able to provide actually reasonable estimates in the given time frame for a small search space, these methods fail for larger search spaces. Therefore, we argue that these methods are not feasible for large search spaces.

Other methods lead to imprecise results, such as GAP, GME or both X-Means variants. The gap statistic tends to underestimate the number of clusters in datasets. Since its authors did not provide details about the choice of the parameter b, other values for b may lead to better results. However, this solely transfers the problem of finding solid parameters of k-center clustering algorithms to finding parameters for the estimation method instead. G-Means (GME) on the other hand tends to overestimate the number of clusters. The reason is that it repeatedly searches for Gaussian distributions and stops, if all clusters follow a Gaussian distribution or a specified upper-bound is met. For the investigated datasets, GME regularly hits the upper-bound for high-dimensional datasets, i.e., where $d = 50$ and $d = 100$. Both X-Means variants XAI and XBI regularly provide a solid estimate. However, for some datasets, the estimation was very imprecise. For these datasets X-Means performed an ascending search and stopped at the first element $k_{est} = 2$, i.e., splitting into more clusters did not improve the AIC or BIC scoring criterion respectively. Hence, X-Means was the fastest estimation method for these datasets (cf. Figure 4), but at the same time suffers from an imprecise estimation.

The remaining methods provide acceptable estimates. Figure 5n shows that the estimates of LOG-Means are very accurate and only exhibit minor deviations. The highest deviation is $\delta k = -8$ %, which is still significantly lower than the deviation for other estimation methods (up to 100 % for GME, XAI, XBI). Very similar observations are made for the results of experiment 2 in Figure 5o. It should be highlighted that, similarly to ELM, LOG-Means provides accurate estimates independent of data characteristics. The reason for this can be found in the similarity of these methods, since both rely on the SSE. The SSE provides appealing characteristics, such that it remains reliable regardless of the voluminosity and the search space.

Figure 5 also depicts the average relative error $\bar{\delta}k$ across all datasets for the investigated estimation methods. Solely CHI, SIL and LOG achieve an average relative error below 2 %, hence provide very accurate estimates in general. However, as shown in Figure 3, the runtimes of CHI and SIL are significantly higher than for LOG. Interestingly, $\bar{\delta}k$ for LOG* is also below 2 %, i.e., LOG-Means is able to provide very accurate estimates even for large search spaces.

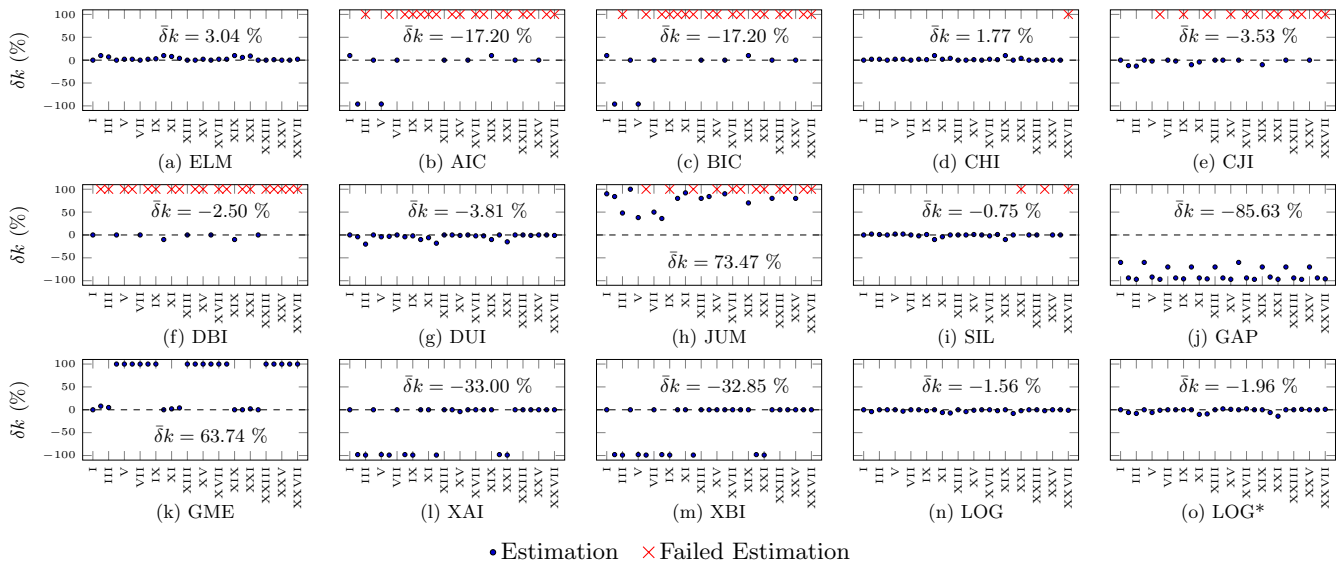


Figure 5: Comparison of the accuracy of estimation methods. $\bar{\delta}k$ shows the average relative error across all synthetic datasets.

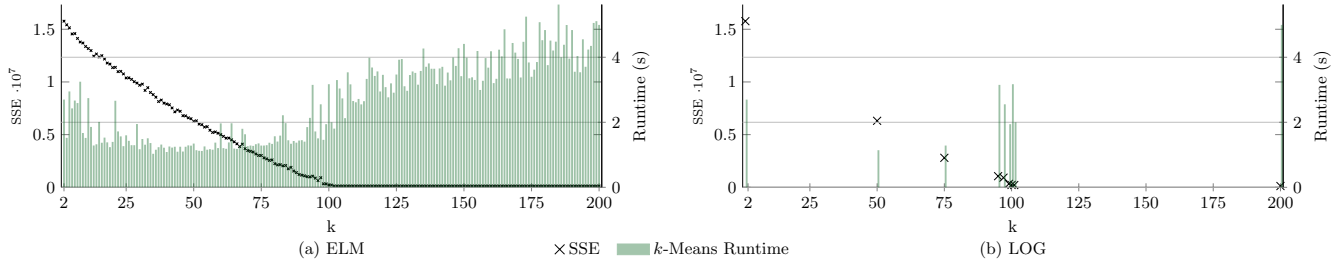


Figure 6: Comparison between the elbow method (ELM) and LOG-Means (LOG) regarding k-Means executions.

5.4 Comparison with the Elbow Method

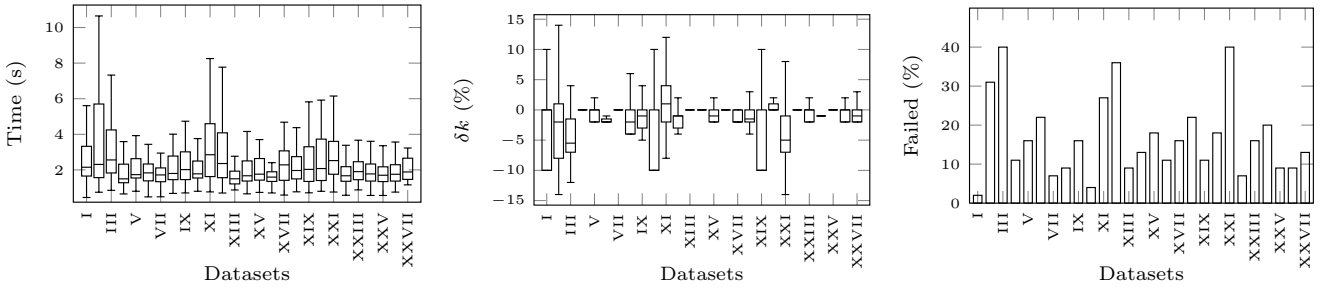
Since LOG-Means exploits the advantages of the elbow method, we compare both methods in this section. Figure 6 depicts their typical behavior. We focus on a single dataset, however the results for the other datasets are very similar. The values for the SSE validity measure are shown in black and the corresponding runtimes for k-Means are depicted in green. Figure 6a shows the results for ELM, where k-Means is executed for all values of k and the SSE is calculated for each of the clustering results. As the measurements show, the runtime of k-Means increases with growing values for k. This is the typical behavior as the larger k is, the more distance calculations are necessary. The consequence for ELM is that large parts of the overall runtime are caused by executions of k-Means after the bend at $k \approx 100$, where the SSE barely changes, i.e., no crucial additional information is gained. However, these executions are very costly in terms of the overall runtime. Even before the bend, some executions can be saved, since the SSE is decreasing very strongly and does not tend to flatten.

As Figure 6b clearly demonstrates, LOG-Means proceeds much more efficiently and tremendously narrows down the search space around the bend. This way, many executions of the clustering algorithm can be saved, which is in particular true for the most expensive executions of k-Means, i.e., those with large values for k. Despite these runtime savings, the accuracy of the estimation is not affected in a negative way by this search strategy, as the experimental results in Figure 5n clearly demonstrate.

Results of the User Study

As the analyst has to select the bend in the elbow graph (cf. Section 3), the time until such an selection is made is also of paramount interest for this estimation method. Figure 7a depicts the time the participants of the user study needed until a decision was made for all investigated datasets. These results show that the median time to select the bend is around 2 seconds. Here, it is of particular interest, that data characteristics apparently have no impact on this time. We assume that the reason for this can be found in the simplicity of the elbow graph: As the SSE denotes the variance of the resulting clusters, it appears that the elbow graph is a reasonable representation and allows to easily select the estimated number of clusters independent of the dataset characteristics. However, in contrast to the overwhelming runtime of the repeated execution of the clustering algorithm (cf. ELM in Figure 3a), the time to select the bend with 2 seconds is rather negligible.

Figure 7b shows the variance of δk for each dataset for the elbow method. The deviations are rather small, as they are below 15 %. Strong deviations appear for a few datasets, e.g., II, III, XI, XIX and XXI. The sole similarity between these datasets is the low dimensionality of $d = 10$ and the



(a) Time until a decision was made. (b) δk of estimates. (c) Failed estimations.

Figure 7: Results of the user study of the elbow method.

rather high number of actual clusters in the dataset. Our observation of the elbow graphs reveals that they flatten slowly without a clear significant bend. Hence, we assume that the participants were insecure about their decision, which makes their selection not as accurate as for the other elbow graphs with clearly significant bends.

Figure 7c outlines the failed estimations within the user study. We marked an estimation as failed, if the participant couldn't find a bend in the elbow graph. Most of the time a selection could be made, but with an increasing number of clusters in the dataset (and therefore an increasing search space in our experiments), the participants more frequently couldn't provide an estimate. This observation can be seen for example when regarding datasets I-III. Here, the data characteristics remain unchanged, except for the number of clusters. This implies that for a larger number of clusters or a larger search space, participants struggle to provide a clear estimate. We assume that the problem mainly arose due to the larger search space. Here, the participants required more time, because more possible choices for the bend were available. Furthermore, only for few datasets (cf. datasets II, III, XI, XII, XXI), the participants couldn't find a bend in the graph more regularly than for the others. Interestingly, these observations correlate with a longer time until a decision was made (cf. Figure 7a) as well as with a lower accuracy of successful estimates (cf. Figure 7b).

Concluding the observations of the user study of the elbow method, we unveiled three main problems: 1) As an exhaustive estimation method, the elbow method requires a huge overall runtime until an estimation can be made. The main reason can be found in the repetitive executions of k-Means, especially for large values of k. The time until a decision was made by an analyst requires approximately 2 seconds, which is negligible in contrast to the huge overall runtime. 2) For datasets, where the corresponding elbow graph did not provide a significant bend or the search space was large, the participants struggled with an estimation. If, however, an estimation was made for these datasets, the estimations across all participants differ. Especially for dataset XXI in Figure 7b δk lies between -14 % and 8 %, which is a remarkable difference in the estimations. 3) For some datasets, the participants were not able to provide an estimate in up to 40 % of all cases (cf. Figure 7c). This correlates with the observation of the datasets from problem 2.

Benefits of LOG-Means

LOG-Means is able to overcome the aforementioned problems. As shown in Figure 6b, LOG-Means executes the clustering algorithm only for few values $k \in \mathcal{R}$. Hence, tremendous runtime savings can be achieved in contrast to existing estimation methods due to fewer executions of k-Means.

Furthermore, as LOG-Means is an automatic method, it requires no human interaction. To this end, two promising advantages can be achieved: The first advantage is that LOG-Means provides an objective estimate in contrast to the elbow method. Due to the formal definition of LOG-Means (cf. Section 4.2), it proceeds in a clear sequence of steps. On the other hand, the elbow method suffers from the perception and the subjectivity of the analyst, which can lead to inaccurate estimates, when for example the elbow graph provides multiple bends or no clear bend at all.

The estimates of LOG-Means reflect a high SSE ratio. Hence, it provides accurate estimates even for datasets with an ambiguous elbow graph. That is, LOG-Means achieved accurate results in terms of δk for datasets II (-4 %), III (0 %), and XXI (-2 %), whereas the participants of the user study struggled to provide accurate estimates.

5.5 Analysis of ε Environments

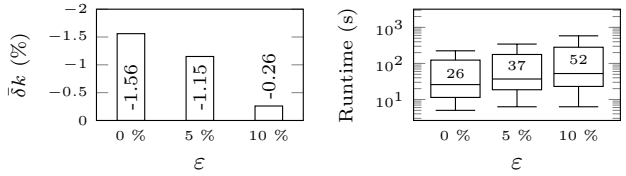


Figure 8: Results for different ε environments. Left: Relative error $\bar{\delta k}$. Right: Overall runtime until an estimation was made. Median values over all datasets are depicted.

As LOG-Means with $\varepsilon = 0$, i.e., without an ε environment, already provides very accurate results with a relative error below 2 % (cf. Figure 5n), there is only little space for improvement. Yet, we repeated experiment 1 on synthetic data, where $\mathcal{R} = [2; 2c]$, with two additional ε environments. We selected these ε environments as relative proportions of the size of the search space $\mathcal{R}$. That is, if $\mathcal{R} = [2; 100]$, then $|\mathcal{R}| = 99$. Hence, if we set $\varepsilon = 5$ %, then $\lfloor 0.05 \times 99 \rfloor = \lfloor 4.95 \rfloor = 4$ additional elements next to the expected bend are executed. In our experiments, we set ε to 5 % and 10 % respectively. As the results in Figure 8 show, the average relative error $\bar{\delta k}$ can be indeed further reduced, yet with the trade-off of a higher runtime. To this end, LOG-Means is able to provide estimates with an average relative error of $\bar{\delta k} = -0.26$ % for $\varepsilon = 10$ % and is therefore

the most accurate estimation method in our experiments (cf. Figure 5). The reason for this is, that with $\varepsilon = 0$, the area where the bend is expected to be can be efficiently narrowed down. Yet, there might be higher SSE ratios in a ε nearby area, which can only be found when focusing on this area. When analyzing this ε environment, we observe a rise of the overall runtime. That is, with an increasing ε environment, k-Means is executed multiple times with varying values for k until the ε environment is analyzed. Hence, the median overall runtime until an estimation can be made by LOG-Means with $\varepsilon = 10$ % rises to 52 seconds. However, this is still faster than existing estimation methods (cf. Figure 3a).

Note, that the question remains how to choose a promising value for ε. A thorough analysis of our results unveiled that even more accurate estimates can be achieved by analyzing the ε environment for elbow graphs with several bends, i.e., high values for the SSE ratio, in a nearby location. This can be traced back to the $\mathcal{O}(log\ k)$-approximation of the clustering algorithm falling into a local optima, thus resulting in several bends for closer distant values for $k \in \mathcal{R}$ (cf. Section 4.3). Hence, increasing ε can reduce the effect of these local optima on the SSE ratios, thus providing more accurate estimates. Yet, prior to execution, it is typically unclear whether the clustering algorithm provides a local optimum. Furthermore, the choice of ε also depends on the underlying goals: If the analyst is interested in rather fast and sufficiently accurate estimates, which are still more accurate than most estimates from existing estimation methods (cf. Section 5.3), setting $\varepsilon = 0$ is a good choice. On the other hand, if one wants even more accurate estimates, increasing ε a little may increase the accuracy by reducing the effect of local optima of the clustering algorithm around the bend, yet with the cost of a higher runtime. Hence, we argue that there is no "best" ε for all scenarios, since we regard ε as a parameter on a continuum trading off runtime with even more accuracy, which depends on the analysts' goals.

5.6 Evaluation on Real-World Data

While all previous experiments were performed on synthetic datasets, we also investigated the effects of LOG-Means on real-world datasets. We used 5 real-world classification datasets mostly from the UCI machine learning library[3], which are regularly used to benchmark new algorithms. In order to use these datasets for clustering, we removed any non-numeric and symbolic values, IDs, timestamps, class labels and empty values. Table 3 summarizes the datasets' characteristics. Note, that these datasets exhibit similar or even larger characteristics as the synthetic datasets from prior experiments regarding n and d.

For this evaluation, we set $\mathcal{R} = [0.5c; 2c]$, since we assume prior knowledge, i.e., that the datasets comprise multi-class problems instead of binary classification problems. Furthermore, we set $\varepsilon = 0$, i.e., only apply the logarithmic search alike part of LOG-Means.

Table 3: Real-world datasets and their characteristics, with n as #entities, d as #dimensions, and c as #classes.

Abbr.	Dataset	n	d	c
A	Avila	10,430	10	12
B	Dataset for Sensorless Drive Diagnosis	58,509	48	11
C	MNIST	60,000	784	10
D	KDD Cup 1999 Data	4,898,431	34	23
E	KITSUNE	21,017,597	115	10

[3] `https://archive.ics.uci.edu/ml/datasets.php`

Table 4: Median values of relative error δk (left) and overall runtime (right) over 3 runs. Bold values indicate best results per dataset, "-" indicates an exceedance of the time budget.

Est. Method	δk (%)					Runtime (s)				
	A	B	C	D	E	A	B	C	D	E
AIC	58	100	-50	52	-	70	115	981	1,667	-
BIC	-8	100	-50	52	-	68	114	977	1,660	-
CHI	-50	18	-50	52	100	32	51	401	310	1,626
CJI	100	-45	90	-52	-50	38	63	559	660	2,214
DBI	58	27	-	-	-	167	306	-	-	-
DUI	50	-45	90	-9	-50	31	49	386	292	1,557
JUM	-50	-55	-50	-52	-50	38	64	562	674	2,303
SIL	-42	-45	**-20**	-52	-50	32	52	412	380	1,766
GAP	-33	-45	-	-39	-	28	60	-	571	-
GME	100	100	100	100	100	26	54	101	221	616
XAI	100	100	-50	100	100	43	51	**49**	226	1,460
XBI	100	100	-50	100	100	95	46	50	282	1,462
LOG	**0**	**-9**	-40	**0**	**-25**	**7**	**13**	115	**75**	**538**

The results in Table 4 unveil that LOG-Means is in most of the cases the fastest and the most accurate method. Solely for dataset C, LOG-Means was the second most accurate method. Furthermore, the more accurate method for this dataset required a multiple of runtime compared to LOG-Means. Also, LOG-Means was in almost all cases the fastest estimation method. For dataset C, GME and both X-Means variants XAI and XBI were faster, yet heavily over- and underestimated the number of clusters.

Concluding, our experiments unveil that LOG-Means is also able to achieve accurate and fast estimates on real-world datasets as well, thus regularly outperforming existing estimation methods for the number of clusters in datasets.

6. CONCLUSION

In this paper, we propose LOG-Means, which is based on formal properties of elbow graphs and scales in sublinear time regarding the parameter search space. Thus, it is a strong fit for large datasets and large search spaces. Our comprehensive evaluation compares LOG-Means with 13 commonly used estimation methods on large datasets and large search spaces. To the best of our knowledge, this is the most systematic comparison as of today. The results unveil that LOG-Means significantly outperforms existing estimation methods in terms of runtime and accuracy. Furthermore, LOG-Means provides accurate estimates even in large search spaces and is therefore of paramount interest for novice analysts.

Since multiple executions of a k-center clustering algorithm are at the core of each estimation method, future research will address how to exploit previous results in subsequent executions to achieve additional speed-ups.

7. ACKNOWLEDGMENTS

This research was partially funded by the Ministry of Science of Baden-Württemberg, Germany, for the Doctoral Program "Services Computing". Some work presented in this paper was performed in the project "INTERACT" as part of the Software Campus program, which is funded by the German Federal Ministry of Education and Research (BMBF) under Grant No.: 01IS17051. Finally, we would like to thank Tim Niederhausen, Dennis Tschechlov and Julius Voggesberger for their crucial implementation work.

8. REFERENCES

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactionson Automatic Control*, 19(6):716–723, 1974.

[2] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 5 2009.

[3] T. W. Anderson and D. A. Darling. Asymptotic Theory of Certain "Goodness of Fit" Criteria Based on Stochastic Processes. *The Annals of Mathematical Statistics*, 23(2):193–212, 6 1952.

[4] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 1 2013.

[5] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *Proceedings of the Annual Symposium on Computational Geometry*, volume 2006, pages 144–153, 2006.

[6] D. Arthur and S. Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1025, 2007.

[7] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable K-Means++. *PVLDB*, 5(7):622–633, 2012.

[8] P. Baldi and G. W. Hatfield. *DNA microarrays and gene expression: from experiments to data analysis and modeling*. Cambridge University Press, 2002.

[9] S. K. Bhatia and J. S. Deogun. Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(3):427–436, 6 1998.

[10] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In *Advances in Neural Information Processing Systems*, pages 368–374, 1997.

[11] T. Caliñski and J. Harabasz. A Dendrite Method For Cluster Analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[12] M. Ceccarello, A. Pietracaprina, and G. Pucci. Solving k-center clustering (with outliers) in MapReduce and streaming, almost as accurately as sequentially. *PVLDB*, 12(7):766–778, 2019.

[13] M. M. T. Chiang and B. Mirkin. Intelligent choice of the number of clusters in k-means clustering: An experimental study with different cluster spreads. *Journal of Classification*, 27(1):3–40, 3 2010.

[14] J. M. Coggins and A. K. Jain. A spatial filtering approach to texture analysis. *Pattern Recognition Letters*, 3(3):195–203, 5 1985.

[15] D. L. Davies and D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 4 1979.

[16] E. Dimitriadou, S. Dolničar, and A. Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1):137–160, 2002.

[17] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1 1974.

[18] S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *35th International Conference on Machine Learning, ICML 2018*, volume 4, pages 2323–2341, 2018.

[19] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and Robust Automated Machine Learning. *Advances in Neural Information Processing Systems*, 2015.

[20] H. Frigui and R. Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):450–465, 1999.

[21] M. Fritz, M. Behringer, and H. Schwarz. Quality-driven early stopping for explorative cluster analysis for big data. *SICS Software-Intensive Cyber-Physical Systems*, 34(2-3):129–140, 6 2019.

[22] M. Fritz and H. Schwarz. Initializing k-means efficiently: Benefits for exploratory cluster analysis. In *Proceedings of OnTheMove Federated Conferences and Workshops (OTM), 27th International Conference on Cooperative Information Systems (CoopIS 2019)*, volume 11877 LNCS, pages 146–163. Springer, Cham, 2019.

[23] M. R. Garey, D. S. Johnson, and H. S. Witsenhausen. The Complexity of the Generalized Lloyd-Max Problem. *IEEE Transactions on Information Theory*, 28(2):255–256, 1982.

[24] G. Hamerly and C. Elkan. Learning the k in kmeans. *Advances in Neural Information Processing Systems*, 17:1–8, 2004.

[25] J. Hu, B. K. Ray, and M. Singh. Statistical methods for automated generation of service engagement staffing plans. *IBM Journal of Research and Development*, 51(3-4):281–293, 2007.

[26] Z. Huang. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. *Research Issues on Data Mining and Knowledge Discovery*, page 1–8, 1997.

[27] M. Iwayama and T. Tokunaga. Cluster-based text categorization: A comparison of category search strategies. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 273–280. Association for Computing Machinery (ACM), 1995.

[28] A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 6 2010.

[29] A. K. Jain and R. C. Dubes. *Algorithms for data clustering*. Prentice Hall, 1988.

[30] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:1–52, 2018.

[31] S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 3 1982.

[32] J. B. Macqueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[33] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 6 1985.

[34] D. Pelleg and A. Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. *In Proceedings of the 17th International Conference on Machine Learning*, pages 727–734, 2000.

[35] G. Punj and D. W. Stewart. Cluster Analysis in Marketing Research: Review and Suggestions for Application. *Journal of Marketing Research*, 20(2):134–148, 5 1983.

[36] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(C):53–65, 11 1987.

[37] G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6:461–464, 1978.

[38] C. E. Shannon. Coding Theorems for a Discrete Source With a Fidelity Criterion. *Institute of Radio Engineers, International Convention Record, Vol 7*, pages 142–163, 1959.

[39] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[40] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 4, pages 2951–2959, 2012.

[41] C. A. Sugar and G. M. James. Finding the Number of Clusters in a Dataset: An Information-Theoretic Approach. *Journal of the American Statistical Association*, 98(463):750–763, 2003.

[42] R. L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 12 1953.

[43] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 63(2):411–423, 2001.

[44] X. Wu, V. Kumar, Q. J. Ross, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.