Extending K-Means Clustering with Ptolemy's Inequality [2]

Max Pernklau [1], Nikita Averitchev[1], and Christian Beecks [1]

Abstract: Clustering is a fundamental data analytics operation in the field of unsupervised learning. Given a database of unknown structure, clustering aims to discover the inherent structure of the data objects according to similarity, such that similar data objects are grouped together, while dissimilar ones are separated in different groups or clusters. In this study, we focus on partitioning methods, specifically the *k-means clustering* approach, which minimizes the intra-cluster variance. As the standard algorithmic approach to k-means clustering, the Lloyd algorithm, is neither efficient nor scalable, various adaptations and modifications have been developed, resulting in the family of fast k-means clustering algorithms.

In this short paper, we extend the clustering algorithm proposed by Elkan with Ptolemy's inequality to prune superfluous distance calculations. It is not our intention to compete with the state-of-the-art algorithmic solutions for the k-means clustering problem; rather, we seek to investigate the potential of Ptolemy's inequality to further enhancements in the standard algorithm's performance, particularly in terms of reducing computations associated with distance evaluations.

Keywords: k-means clustering, Lloyd algorithm, Ptolemy's inequality

1 Introduction

Clustering is a fundamental data-analytics operation in the domain of data science. The basic task is to divide a set of data objects into different groups or clusters, such that objects within the same cluster have sufficient similarities; meanwhile objects in different clusters should have significant differences. The collection of clusters reflects the inherent structure of the underlying data-generating process and is denoted as a clustering.

The need for clustering arises in various scientific or economic application domains [Ez22, OT23, GMW20], ranging from archaeology [Tr24] and finance [CLKK16] to industry [LL21] and zoology [Sh21], to name just a few. Especially in the field of unsupervised learning, clustering is utilized to extract structures from unlabeled data [CG23].

Alongside the diverse applications of clustering, unique domain-specific requirements and challenges arise, which are addressed by leveraging different families of clustering approaches [XT15, HKP12]. These approaches, ranging from partitioning and hierarchical methods to density-based, grid-based and graph-based methods, are designed to accommodate varying data characteristics and domain-related requirements to facilitate efficient cluster analyses across diverse applications.

[1] FernUniversität in Hagen, Chair of Data Science, 58084 Hagen, Germany,
max.pernklau@fernuni-hagen.de, https://orcid.org/0009-0007-5520-4093; naveritchev@gmail.com;
christian.beecks@fernuni-hagen.de, https://orcid.org/0009-0000-9028-629X

[2] This work is partly funded by DFG grant No. 454630593: EPIX – Efficient Ptolemaic Indexing.

In this short paper, we focus on partitioning methods due to their simplicity in terms of interpretability and implementability [Be22]. In this field, the *k-means algorithm* [Bo07, HH08, Ja10, St06] has become one of the most influential clustering techniques [Wu08, ONM19]. Although the term *k-means algorithm* is explicitly credited to MacQueen [Ma67], its origins trace back to Steinhaus [St56] in 1956, with its first application to data clustering by Forgy [Fo65] in 1965. The widely recognized version in use today is the *Lloyd algorithm* [Ll82], introduced in 1982.

With the proliferation of complex data sources and the growing demand for more efficient clustering techniques, numerous *fast k-means algorithms* have been introduced [El03, Ha10, DH12, HD15, NF16, Di15, SLF21, LS23]. These methods are designed to achieve good clustering results with significantly reduced computational cost. Moreover, most of them do not require any form of precomputation, making them directly applicable across a wide range of scenarios. The major objective of fast k-means algorithms is to reduce the number of distance evaluations required when assigning data objects to cluster centers and to safely prune cluster centers from the assignment process. For this purpose, distances are approximated via lower and upper bounds.

The Elkan algorithm [El03] is a prominent representative of these algorithms and particularly suited for high-dimensional scenarios. It maintains one upper bound and multiple lower bounds for each data object in order to apply different pruning criteria for each combination of data object and cluster center. While the lower and upper bounds of Elkan's algorithm are directly derived from the triangle inequality, we propose to utilitze Ptolemy's inequality instead in order to increase the pruning performance. With this extension, we do not aim to compete with the state of the art in k-means clustering; rather, we intend to investigate the potential of Ptolemy's inequality to further enhancements in the standard algorithm's performance, specifically in terms of reducing computations associated with distance evaluations. We believe that our research findings offer a promising direction for future research and are beneficial for data scientists researchers and practitioners alike.

This short paper is structured as follows: Section 2 outlines the k-means problem and Elkan's approach of accelerating the algorithmic computation. In Section 3, we show how Ptolemy's inequality can be applied to the aforementioned algorithm. The results of our preliminary performance evaluation are detailed in Section 4, while Section 5 concludes this paper with a short outlook on future work.

2 Preliminaries

In this section, we first define the k-means clustering problem and the Lloyd algorithm as a straightforward, but optimizable solution to this problem in Section 2.1. We then continue with the introduction of means of acceleration trough lower and upper bounding in Section 2.2.

2.1 The k-means problem

In order to discuss the k-means problem, we first define the concept of a *clustering algorithm* in a formal and implementation-agnostic manner:

Definition 1 (clustering algorithm) *Given a target number of clusters $k \in \mathbb{N}$, a clustering algorithm is a function F that assigns each element from an input data set $D \subset \mathbb{R}^n$ a cluster index $i \in \{1, 2, \ldots, k\}$*

$$F : D \to \{1, 2, \ldots, k\} \; .$$

The intention is to assign similar objects the same index, thus clustering them together. Notably, we restrict our analysis to clustering problems that use the n-dimensional Euclidean space as the domain of the input data set. Likewise, we assume that the dissimilarity of the data points to be clustered is completely described by the Euclidean distance $d(x, y) = ||x - y||_2$. The rationale behind this assumption will be made clear shortly in Section 2.2.

The core of the k-means problem, then, is the assignment of clusters so that the variance in dissimilarity between all points in the same cluster should be reasonably small. More formally:

Definition 2 (k-means problem) *Let F be a clustering algorithm and C_i be the set of all elements assigned to cluster i:*

$$C_i = \{x \in D \mid F(x) = i\} \; .$$

Furthermore, let ϵ be the weighted sum of inter-cluster variances

$$\epsilon = \sum_{i=1}^{k} |C_i| \operatorname{Var}[C_i] = \sum_{i=1}^{k} \sum_{x_j \in C_i} d^2(x_j, c_i) \, ,$$

where the cluster's average element $c_i = \mathrm{E}[C_i]$ is called the center of the cluster C_i.

Then, the map F is called a solution to the k-means problem iff ϵ cannot be made smaller by changing the assignment of any one element to a different cluster.

Note that with this definition, common k-means algorithms provide solutions to this k-means problem, as we do not require that F achieves the smallest possible ϵ, just a local minimum of ϵ. This is consistent with the colloquial notion that solutions to the k-means problem "can be found", even when an optimal solution (an NP-hard problem) is not practically obtainable [HKP12].

The most prevalent algorithm employed to identify valid solutions to the k-means problem is the Lloyd algorithm [Ll82], which is reproduced in Algorithm 1; given its widespread use, we will only cover its most significant characteristics as they relate to our analysis.

Algorithm 1 k-Means Algorithm

```
Input: $k$: Number of clusters, $D$: Dataset containing $n$ objects
Output: Assignment of each $x_i \in D$ to a clusters
1: Initialize $k$ cluster centers $\{c_1, c_2, \ldots, c_k\}$ arbitrarily from $D$
2: repeat
3:     // Assign $x_i$ to the closest cluster
4:     for each object $x_i$ in $D$ do
5:         $C_j \leftarrow C_j \cup \{x_i\}$ where $j = \arg\min_j d^2(x_i, c_j)$
6:     // Update cluster center $c_j$
7:     for each cluster $C_j$ do
8:         $c_j \leftarrow \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$
9: until no change in cluster assignments
```

For each iteration of Lloyd's algorithm, the assignment step (line 4) needs to identify the closest cluster center c_j for each element x_i from the input dataset D. To do so, Lloyd's algorithm calculates the distances between each cluster and each element from D, requiring $k \cdot |D|$ total distance evaluations per iteration. These frequent distance evaluations can make up a significant share of the algorithms computational costs, especially for high-dimensional datasets.

2.2 Accelerating k-means

As we have seen, Lloyd's algorithm requires numerous evaluations of the distance function. However, the exact distances do not actually need to be calculated explicitly for every element, as it is sufficient to identify which cluster center is the closest to a given element. This is often possible from geometric considerations alone, when lower and upper distance bounds are available. Two properties of the Euclidean space are of particular use here: The triangle inequality

$$d(x, y) \leq d(x, z) + d(z, y) \tag{1}$$

and Ptolemy's inequality

$$d(x, y) \cdot d(v, u) \leq d(x, v) \cdot d(y, v) + d(x, u) \cdot d(y, v) \ . \tag{2}$$

Both inequalities, known since ancient times, provide lower and upper bounds on the distances between two points given two (respectively five) other distances.

Numerous algorithms have been developed that exploit the former inequality to avoid explicit distance evaluations. However, to the best of our knowledge, the latter has not been employed for the purposes of accelerating k-means clustering yet.

Algorithm 2 Elkan's Algorithm

```
Initialization: Initialize all cluster centers. For each point $x_i$ and each center $c_j$, set the lower bound $l(x_i, c_j)$ and the upper bound $u(x_i)$. Assign each $x_i$ to the nearest cluster $C_j$ such that $c(x_i) = \arg\min_j d(x_i, c_j)$, utilizing that $d(c_j, c_m) \geq 2d(x_i, c_j) \Rightarrow d(x_i, c_m) \geq d(x_i, c_j)$ to minimize distance calculations. Set $r(x_i)$ = true for all points.
repeat
    Step 1: Compute distances $d(c_i, c_j)$ between all centers, and calculate $s(c_i) = \frac{1}{2} \min_{c_j \neq c_i} d(c_i, c_j)$ for each center $c_i$.
    Step 2: Retain points $x_i$ in their current clusters if $u(x_i) \leq s(c(x_i))$.
    Step 3: For remaining points, consider $x_i$ for reassignment if:
        •  $c_j \neq c(x_i)$,
        •  $u(x_i) > l(x_i, c_j)$, and
        •  $u(x_i) > 0.5 \cdot d(c(x_i), c_j)$.
    Step 3a: If $r(x_i)$ is true, compute $d(x_i, c(x_i))$. Set $r(x_i)$ = false. Otherwise, $u(x_i) = d(x_i, c(x_i))$.
    Step 3b: If $d(x_i, c(x_i)) > l(x_i, c_j)$ or $d(x_i, c(x_i)) > \frac{1}{2} d(c(x_i), c_j)$, compute $d(x_i, c_j)$. Reassign $x_i$ to $C_j$ if $d(x_i, c_j) < d(x_i, c(x_i))$.
    Step 4: Compute the cluster centers as the centroids of the corresponding clusters $c'_j$.
    Step 5: Update lower bounds $l(x_i, c_j)$ for each $x_i$ and $c_j$ using Eq. 3.
    Step 6: Update upper bounds $u(x_i)$ for each $x_i$ using Eq. 4. Reset $r(x_i)$ = true
    Step 7: Replace each center $c_j$ with $c'_j$.
until convergence
```

To demonstrate the potential of the latter, we modify Elkan's algorithm [El03] that already employs the triangle inequality, to also make use of Ptolemy's inequality. We provide a short summary of Elkan's algorithm here and describe our extension in Section 3.

Elkan's algorithm maintains two kinds of distance bounds in each iteration of the algorithm:

i $u(x) \geq d(x, c_i) \mid i = F(x)$, upper bounds between data points and their currently assigned centers.

ii $l(x, c_i) \leq d(x, c_i) \forall i \neq F(x)$, lower bounds between points and all other centers.

Intuitively, when the upper bound is larger than all lower bounds $u(x) \geq l(x, c_i) \forall i \neq F(x)$, the cluster assignment of a data point has not changed. Explicit distances to other clusters are only required when this inequality does not hold, i.e. when the bounds are not tight enough or when the cluster assignment of the data point has indeed changed. It is immediately evident that the quality of the bounds has a significant impact on the achievable improvements in execution speed: The larger (smaller) the lower (upper) bound is, the more likely an explicit distance evaluation can be avoided.

While Elkan's algorithm also employs additional techniques to reduce the number of distance calculations, we want to focus on the calculation of the bounds here; a complete description of the algorithm is given in Algorithm 2, as reproduced from [El03].

The lower and upper bounds are calculated at the end of each iteration of Elkan's algorithm and are given by

$$l'(x_i, c_j) = \max\{l(x_i, c_j) - d(c'_j, c_j), 0\} \tag{3}$$

$$u'(x_i) = u(x_i) + d(c'(x_i), c(x_i)) , \tag{4}$$

where the prime symbol (c'_j, u', l') indicates quantities calculated in this iteration, while unprimed variables refer to quantities calculated in the preceding iteration. The given equations for the bounds follow directly from the triangle inequality 1.

The calculation of these bounds entails an overhead of $\mathrm{O}(k^2)$ distance calculations per iteration. However, this additional cost tends to be small compared to the number of distance calculations $\mathrm{O}(k \cdot |D|)$ that can potentially saved, as $k \ll |D|$ in most practical applications.

3 Applying Ptolemy's Inequality to the Elkan's Algorithm

As seen in the previous section, Elkan's algorithm can improve the execution speed of k-means clustering through the use of the triangle inequality. Explicit distance calculations are avoided when tight upper and lower bounds of the distance function are available.

In this section, we use Ptolemy's inequality to calculate an additional set of bounds and modify Elkan's algorithm accordingly. These Ptolemaic bounds often improve upon the triangular bounds, leading to increased execution speeds, as will be shown in Section 4.

3.1 Ptolemaic Bounds

Recall that the triangle inequality relates the distances between three points, while Ptolemy's inequality does so for four points. At the end of an iteration of Elkan's algorithm (Step 5 to 7 in Alg. 2), the distance bounds between the data points x_i and the new center positions c'_j are calculated. Therefore, the three points used in the calculation of the bounds are the aforementioned two points and c_j, the cluster center calculated in the previous iteration.

A natural extension to this procedure is then to use an even older cluster center $c^\diamond_j$ as a fourth point, i.e. the cluster center calculated in the penultimate iteration. To formalize, Ptolemy's inequality can be used to calculate the following upper and lower bounds:

Theorem 1 (Ptolemaic bounds for Elkan's algorithm) *During an iteration of Elkan's algorithm, let $x_i \in D \subset \mathbb{R}^n$ be a point from the dataset to be clustered, d the Euclidean distance function, c'_j a cluster center calculated in this iteration, c_j a cluster center* *calculated in the previous iteration, and* $c_j^\diamond$ *a cluster center calculated in the iteration before the previous iteration. A lower and upper bound on* $d(x_i, c_i')$ *is then given by*

$$d(x_i, c_j') \leq u_i' = \frac{1}{d(c_j, c_j^\diamond)} \cdot \Big(u_i \cdot d(c_j', c_j^\diamond) + u_i^\diamond \cdot d(c_j', c_j) \Big) \tag{5}$$

$$d(x_i, c_j') \geq l_{i,j}' = \frac{1}{d(c_j, c_j^\diamond)} \cdot \max \left\{ \begin{array}{l} l_{i,j}^\diamond \cdot d(c_j, c_j') - u_i \cdot d(c_j', c_j^\diamond) \\ l_{i,j} \cdot d(c_j', c_j^\diamond) - u_i^\diamond \cdot d(c_j, c_j') \end{array} \right\}, \tag{6}$$

where $u_i, u_i^\diamond$ *(*$l_{i,j}, l_{i,j}^\diamond$*) are the lower (upper) bounds calculated in the previous and the penultimate iteration, respectively*[3].

Proof: As all points are in the $\mathbb{R}^n$ and the distances function is also Euclidean, Ptolemy's inequality $d(x, y) \cdot d(v, u) \leq d(x, v) \cdot d(y, v) + d(x, u) \cdot d(y, v)$ holds. Eq. 5 can easily be shown by rearranging Ptolemy's inequality and noting that the upper bounds $u_i, u_i^\diamond$ used in place of exact distances can only increase the right-hand side of the inequality. Thus, $d(x_i, c_j') \leq u_i'$ holds.

The process is similar for Eq. 6; as the signs on the right-hand side are not equal in this case, there are two distinct ways to substitute into Ptolemy's inequality. This gives rise to two inequalities, of which the maximum is the stronger bound. Analogous to u_i', the right-hand side of Eq. 6 can only be made smaller by inserting lower bounds $l_{i,j}, l_{i,j}^\diamond$ in the minuend or upper bounds $u_i, u_i^\diamond$ in the subtrahend. Thus, $d(x_i, c_j') \geq l_{i,j}'$ also holds. □

3.2 Integration

To integrate the novel bounds into the existing framework of Elkan's algorithm, Steps 6 and 7 in Alg. 2 need to be updated to use the new bounds. It is possible to only replace either the upper or lower bound, which leads to a hybrid solution that is further explored in Section 4.

As the Ptolemaic bounds require information from two previous iterations, the first iteration of the algorithm is always conducted with Elkan's original bounds.

4 Preliminary Results

In order to demonstrate the efficacy of our proposed algorithm, we undertake a comparative analysis with the performance of Elkan's algorithm, which it extends. We measure the number of distance evaluations for different benchmark datasets and for different values of k, the number of clusters. Our implementation is available through GitHub [Av24].

[3] For readability, we suppress the bounds' arguments in favor of indices, e.g. u_i instead of $u(x_i)$.

Tab. 1: Datasets used in the experiments.

Group	Dataset	Cardinality	Dimensionality
fixed structure	Iris	150	4
	Wine	178	13
variable dimensionality	Gaussian Low Dimension	10 000	10
	Gaussian Medium Dimension	10 000	100
	Gaussian High Dimension	10 000	300
	Random Low Dimension	5000	10
	Random Medium Dimension	5000	100
	Random High Dimension	5000	300

4.1 Datasets

We employ two groups of datasets to compare the different algorithms, summarized in Table 1: The two real-world datasets include the Iris and Wine datasets [Pe11], which have fixed dimensions and cardinalities and represent different clustering structures. The two synthetic datasets include Gaussian and uniformly randomly generated points, which are used to test for the effects of changing dataset dimensionality.

4.2 Experiments

Our variant of the k-Means algorithm is tested by implementing the classical k-Means algorithm (Lloyd's algorithm), Elkan's algorithm, and its novel, Ptolemy-based extension in Python.

To provide a fair comparison of the algorithms, we measure the number of distance calculations instead of wall-clock time. As distance calculations account for the majority of computing time in k-Means algorithms, they offer a clear measure of efficiency regardless of implementational optimizations.

Additionally, two intermediate variants were implemented to assess the separate effects of the lower and upper bounds: The *Ptolemy Upper Bound* method uses Elkan's lower bound and Ptolemy's upper bound, while *Ptolemy Lower Bound* uses Ptolemy's lower bound and Elkan's upper bound.

Each algorithm is executed three times on each dataset, with the parameters $k = 3$, $k = 20$, and $k = 100$.

In Figure 1, we report our results in the form of speedups, i.e. as the number of distances calculated by our extension, divided by the number of distances calculated by Elkan's algorithm.

4.3 Discussion

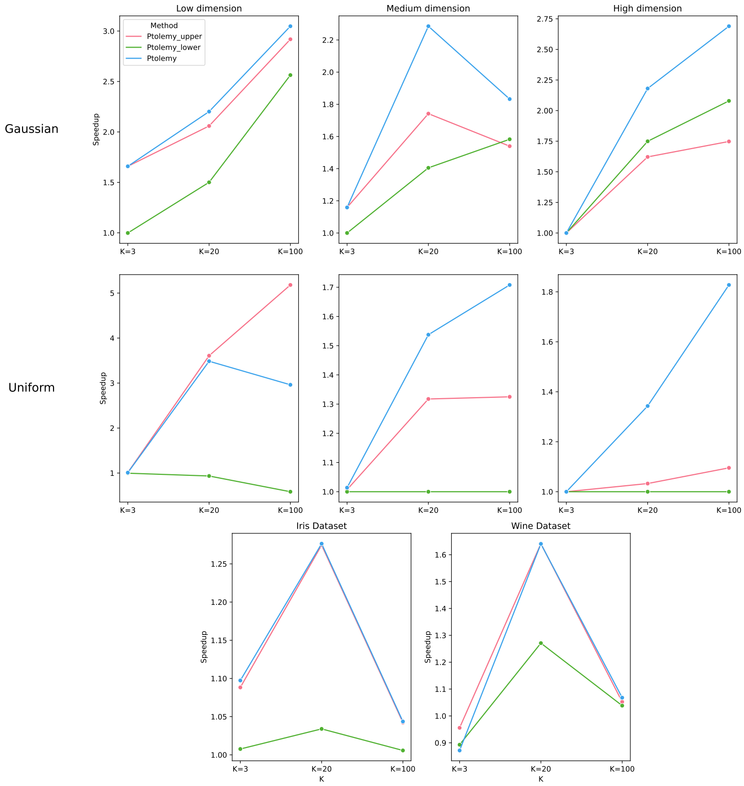

Fig. 1: Relative improvement in the number distance evaluations, compared to Elkan's algorithm. A speedup of > 1 indicates an improvement over Elkan's original algorithm. **Top and middle row:** Performance increases on synthetic data generated from Gaussian (top) and uniformly random (middle) distributions. The dimensionality of the dataset increases left-to-right. **Bottom row:** Performance increases on the real-world datasets Iris (left) and Wine (right).

The primary focus of this analysis is the full Ptolemaic algorithm as compared to Elkan's algorithm, since it is the most robust one of the three variants. In general, the full Ptolemaic extension shows substantial improvements in reducing the number of distance evaluations across most scenarios.

Number of Clusters. The full Ptolemaic algorithm consistently demonstrates significant improvements over Elkan's algorithm as the number of clusters k increases. For instance, in the Gaussian low-dimensional dataset, the Ptolemaic algorithm achieves larger than 2-fold speedups for $k = 20$ and $k = 100$, illustrating its ability to significantly reduce distance calculations in complex clustering tasks. For smaller cluster counts ($k = 3$), the advantage is less pronounced, with speedups around 1.1 in the Iris and Wine datasets. This suggests that Ptolemy's inequality offers significant improvements for larger values of k. However, lower-complexity clustering tasks may not benefit to the same extent, given the additional overhead imposed by the Ptolemaic bounds, which are computationally more complex to evaluate compared to the triangle bounds. The Ptolemaic Upper Bound variant performs comparably to the full Ptolemaic algorithm, especially for larger k values. However, the Lower Bound variant underperforms in some cases, particularly with smaller cluster counts or in higher-dimensional datasets.

Dimensionality. The full Ptolemaic algorithm performs well across varying dataset dimensionality. In low-dimensional datasets (10 dimensions), it achieves impressive speedups, particularly at higher cluster counts. As dimensionality increases, performance gains persist but become slightly less pronounced. Nonetheless, even in higher-dimensional datasets, the full Ptolemaic algorithm remains effective, consistently reducing distance evaluations. However, the Ptolemaic lower bound variant often struggles in these higher-dimensional cases, further emphasizing the robustness of the full Ptolemaic approach.

Limitations. The presented experiments are subject to three primary constraints: (i) While synthetic random datasets provide valuable best-case scenarios, more extensive real-world data with appropriate grouping structures is necessary for comprehensive evaluation. (ii) The significant reduction in distance calculations may not directly translate to improved execution times, as Ptolemy's bounds are computationally more expensive than the triangle inequality. (iii) Most critically, the limited sample size and fixed initial conditions, particularly the lack of variation in cluster initializations, severely restrict the generalizability of individual measurements.

Despite these limitations, we conclude that the observed overall trend remains compelling and warrants further investigation into the application of Ptolemy's inequality for clustering algorithms.

5 Conclusions and Future Work

This short paper has addressed the problem of efficient data clustering by means of the k-means algorithm. We have shown how to extend the more efficient clustering algorithm proposed by Elkan with novel distance approximations derived from Ptolemy's inequality. The results of our preliminary performance evaluation indicate that our proposal achieves significant improvements in performance compared to Elkan's algorithm.

Moreover, the use of the Ptolemy's inequality opens up new avenues for future work. Besides conducting more extensive experiments to verify our findings, it would be valuable to explore adaptations of this technique for other k-means algorithms and optimizations. In particular for algorithms which rely on the triangle inequality to prune distance calculations, Ptolemy's inequality can potentially be leveraged to achieve an enhancement in performance, making these clustering algorithms more practical for a broader range of applications.

Bibliography

[Av24] Averitchev, Nikita: kMeans-Extension-Ptolemy. `https://github.com/nikitaaveritchev/kmeans-Extension-Ptolemy`, 2024.

[Be22] Beecks, Christian; Berns, Fabian; Hüwel, Jan David; Linxen, Andrea; Schlake, Georg Stefan; Düsterhus, Tim: A Comparative Performance Analysis of Fast K-Means Clustering Algorithms. In (Pardede, Eric; Haghighi, Pari Delir; Khalil, Ismail; Kotsis, Gabriele, eds): Information Integration and Web Intelligence - 24th International Conference, iiWAS 2022, Virtual Event, November 28-30, 2022, Proceedings. volume 13635 of Lecture Notes in Computer Science. Springer, pp. 119–125, 2022.

[Bo07] Bock, Hans-Hermann: Clustering methods: a history of k-means algorithms. In: Selected contributions in data analysis and classification, pp. 161–172. Springer, 2007.

[CG23] Chander, Bhanu; Gopalakrishnan, Kumaravelan: Data clustering using unsupervised machine learning. In: Statistical Modeling in Machine Learning, pp. 179–204. Elsevier, 2023.

[CLKK16] Cai, Fan; Le-Khac, Nhien-An; Kechadi, Tahar: Clustering approaches for financial data analysis: a survey. arXiv preprint arXiv:1609.08520, 2016.

[DH12] Drake, Jonathan; Hamerly, Greg: Accelerated k-means with adaptive distance bounds. In: 5th NIPS workshop on optimization for machine learning. volume 8, 2012.

[Di15] Ding, Yufei; Zhao, Yue; Shen, Xipeng; Musuvathi, Madanlal; Mytkowicz, Todd: Yinyang K-Means: A Drop-In Replacement of the Classic K-Means with Consistent Speedup. In: ICML. volume 37 of JMLR Workshop and Conference Proceedings, pp. 579–587, 2015.

[El03] Elkan, Charles: Using the Triangle Inequality to Accelerate k-Means. In: ICML. AAAI Press, pp. 147–153, 2003.

[Ez22] Ezugwu, Absalom E; Ikotun, Abiodun M; Oyelade, Olaide O; Abualigah, Laith; Agushaka, Jeffery O; Eke, Christopher I; Akinyelu, Andronicus A: A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Engineering Applications of Artificial Intelligence, 110:104743, 2022.

[Fo65] Forgy, Edward W: Cluster analysis of multivariate data: efficiency versus interpretability of classifications. biometrics, 21:768–769, 1965.

[GMW20] Gan, Guojun; Ma, Chaoqun; Wu, Jianhong: Data clustering: theory, algorithms, and applications. SIAM, 2020.

[Ha10] Hamerly, Greg: Making k-means Even Faster. In: SDM. SIAM, pp. 130–140, 2010.

[HD15] Hamerly, Greg; Drake, Jonathan: Accelerating Lloyd's algorithm for k-means clustering. In: Partitional clustering algorithms, pp. 41–78. Springer, 2015.

[HH08] Hans-Hermann, BOCK: Origins and extensions of the k-means algorithm in cluster analysis. Journal Electronique d'Histoire des Probabilités et de la Statistique Electronic Journal for History of Probability and Statistics, 4(2), 2008.

[HKP12] Han, Jiawei; Kamber, Micheline; Pei, Jian: Data Mining: Concepts and Techniques. Morgan Kaufmann, third edition, 2012.

[Ja10] Jain, Anil K.: Data clustering: 50 years beyond K-means. Pattern Recognit. Lett., 31(8):651–666, 2010.

[Ll82] Lloyd, Stuart P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory, 28(2):129–136, 1982.

[LL21] Lee, Changhun; Lim, Chiehyeon: From technological development to social advance: A review of Industry 4.0 through machine learning. Technological Forecasting and Social Change, 167:120653, 2021.

[LS23] Lang, Andreas; Schubert, Erich: Accelerating k-Means Clustering with Cover Trees. In (Pedreira, Oscar; Estivill-Castro, Vladimir, eds): Similarity Search and Applications - 16th International Conference, SISAP 2023, A Coruña, Spain, October 9-11, 2023, Proceedings. volume 14289 of Lecture Notes in Computer Science. Springer, pp. 148–162, 2023.

[Ma67] MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics. University of California Press, pp. 281–297, 1967.

[NF16] Newling, James; Fleuret, François: Fast k-means with accurate bounds. In: ICML. volume 48 of JMLR Workshop and Conference Proceedings. JMLR.org, pp. 936–944, 2016.

[ONM19] Olukanmi, P; Nelwamondo, F; Marwala, T: Rethinking k-means clustering in the age of massive datasets: a constant-time approach. Neural Computing and Applications, pp. 1–23, 2019.

[OT23] Oyewole, Gbeminiyi John; Thopil, George Alex: Data clustering: application and trends. Artificial Intelligence Review, 56(7):6439–6475, 2023.

[Pe11] Pedregosa, Fabian; Varoquaux, Gaël; Gramfort, Alexandre; Michel, Vincent; Thirion, Bertrand; Grisel, Olivier; Blondel, Mathieu; Prettenhofer, Peter; Weiss, Ron; Dubourg, Vincent et al.: Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct):2825–2830, 2011.

[Sh21] Shen, Xiaocheng; Zhang, Shujie; Shen, Qi; Hu, Guilin; Lu, Jiqi: Multivariate similarity clustering analysis: A new method regarding biogeography and its application in global insects. Integrative Zoology, 16(3):390–403, 2021.

[SLF21] Schubert, Erich; Lang, Andreas; Feher, Gloria: Accelerating Spherical k-Means. In (Reyes, Nora; Connor, Richard; Kriege, Nils M.; Kazempour, Daniyal; Bartolini, Ilaria; Schubert, Erich; Chen, Jian-Jia, eds): Similarity Search and Applications - 14th International Conference, SISAP 2021, Dortmund, Germany, September 29 - October 1, 2021, Proceedings. volume 13058 of Lecture Notes in Computer Science. Springer, pp. 217–231, 2021.

[St56] Steinhaus, H: Sur la division des corps materiels en parties. Bull. Acad. Polon. Sci., C1. III vol IV: 801-804. 1956.

[St06] Steinley, Douglas: K-means clustering: a half-century synthesis. British Journal of Mathematical and Statistical Psychology, 59(1):1–34, 2006.

[Tr24] Troiano, Maurizio; Nobile, Eugenio; Grignaffini, Flavia; Mangini, Fabio; Mastrogiuseppe, Marco; Conati Barbaro, Cecilia; Frezza, Fabrizio: A Comparative Analysis of Machine Learning Algorithms for Identifying Cultural and Technological Groups in Archaeological Datasets through Clustering Analysis of Homogeneous Data. Electronics, 13(14):2752, 2024.

[Wu08] Wu, Xindong; Kumar, Vipin; Quinlan, J. Ross; Ghosh, Joydeep; Yang, Qiang; Motoda, Hiroshi; McLachlan, Geoffrey J.; Ng, Angus F. M.; Liu, Bing; Yu, Philip S.; Zhou, Zhi-Hua; Steinbach, Michael S.; Hand, David J.; Steinberg, Dan: Top 10 algorithms in data mining. Knowl. Inf. Syst., 14(1):1–37, 2008.

[XT15] Xu, Dongkuan; Tian, Yingjie: A comprehensive survey of clustering algorithms. Annals of Data Science, 2(2):165–193, 2015.