Khatri-Rao Clustering for Data Summarization

Martino Ciaperoni
Scuola Normale Superiore
Pisa, Italy
martino.ciaperoni@sns.it

Collin Leiber
Aalto University
Espoo, Finland
collin.leiber@aalto.fi

Aristides Gionis
KTH Royal Institute of Technology
Digital Futures
Stockholm, Sweden
argioni@kth.se

Heikki Mannila
Aalto University
Espoo, Finland
heikki.mannila@aalto.fi

Abstract

As datasets continue to grow in size and complexity, finding succinct yet accurate data summaries poses a key challenge. Centroid-based clustering, a widely adopted approach to address this challenge, finds informative summaries of datasets in terms of few prototypes, each representing a cluster in the data. Despite their wide adoption, the resulting data summaries often contain redundancies, limiting their effectiveness particularly in datasets characterized by a large number of underlying clusters. To overcome this limitation, we introduce the Khatri-Rao clustering paradigm that extends traditional centroid-based clustering to produce more succinct but equally accurate data summaries by postulating that centroids arise from the interaction of two or more succinct sets of protocentroids.

We study two approaches to centroid-based clustering, the well-established k-Means algorithm and the increasingly popular deep clustering, under the lens of the Khatri-Rao paradigm. To this end, we introduce the Khatri-Rao-k-Means algorithm and the Khatri-Rao deep clustering framework. Extensive experiments show that Khatri-Rao-k-Means can strike a more favorable trade-off between succinctness and accuracy in data summarization than standard k-Means. Leveraging representation learning, the Khatri-Rao deep clustering framework offers even greater benefits, reducing even more the size of data summaries given by deep clustering while preserving their accuracy.

Keywords

Data summarization, centroid-based clustering, deep clustering

1 Introduction

Distilling datasets into succinct and meaningful summaries represents an essential and ubiquitous task. Clustering has emerged as a core methodology for tackling this problem. In particular, centroid-based clustering holds a central position when looking for effective data-summarization strategies. Given a notion of similarity, centroid-based clustering algorithms summarize data by assigning each data point to the closest element in a succinct set of representative data points called *centroids*. The data summaries provided by centroid-based clustering identify meaningful clusters and reveal patterns useful for applications such as exploratory data analysis, segmentation, anomaly detection, and document organization [58].

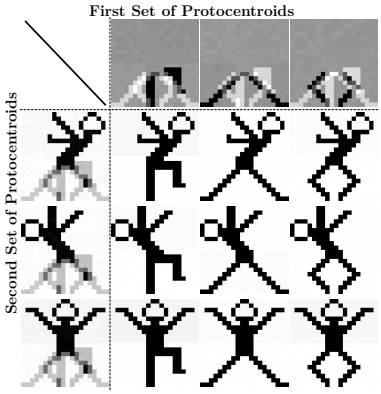

Figure 1: STICKFIGURES dataset. Example of two sets of 3 protocentroids interacting additively to generate 9 centroids.

In many cases, the interest lies in extracting a data summary which is as succinct and accurate as possible. Clustering has consistently shown success in summarizing data across diverse domains, for example, in image [41], document [14], tabular [22], and spatio-temporal [56] data processing, as well as for compressing deep neural networks [52, 53].

Pushing the boundaries of centroid-based clustering for data summarization through Khatri-Rao operators. Many modern applications demand the summarization of datasets of unprecedented scale [13]. As the size of a dataset increases, the underlying cluster structure may evolve either by enlarging clusters or by introducing additional clusters. Thus, modern datasets can exhibit a massive number of underlying clusters [4, 30]. For instance, studies on protein structures analyze millions of clusters [3]. Similarly, a large number of clusters can be incurred in topic modeling for documents [50]. In entity resolution, including record linkage and de-duplication, the number of clusters grows with dataset size [5], and in real-world networks the number of clusters tends to follow a power-law distribution that also increases with network size [10]. Centroid-based clustering struggles to yield data summaries that are both accurate and succinct in applications characterized by a large number of clusters.

In spite of this, no existing work challenges the long-standing centroid-based clustering paradigm toward more succinct data summarization. To address this gap, we adopt a data-compression perspective on centroid-based clustering, and we investigate the following research question: *do standard centroid-based clustering algorithms produce data summaries that carry some degree of redundancy, suggesting the potential for further compression?*

EDBT '26, Tampere (Finland)

10.48786/edbt.2026.32

To shed light on this question, we relax a crucial assumption of centroid-based clustering, namely that centroids are independent entities. Often, richer representations emerge from the interaction of simpler building blocks, as illustrated, e.g., by multi-head attention [54]. In this spirit, we postulate that centroids admit a more succinct representation in terms of a smaller set of building blocks. More specifically, we introduce the Khatri-Rao clustering paradigm, in which centroids arise from the interaction of succinct sets of *protocentroids* through so-called Khatri-Rao operators. These operators aggregate each protocentroid in each set with all protocentroids of the other sets via elementwise operations such as sums or products.

A simple example of this formulation is given in Figure 1. It shows that the nine clusters in the STICKFIGURES dataset [19] can be represented by two sets of three protocentroids. Three protocentroids explain the upper part of the stick figures and three explain the lower part. The final centroids can then be generated by additively combining each protocentroid in one set with every protocentroid in the other set. In this case, centroids are said to exhibit a Khatri-Rao structure, and more precisely they correspond to the Khatri-Rao sum of the two sets of protocentroids. The stick-figure example shows that the dataset can be summarized by just 6 images, whereas standard centroid-based clustering algorithms require 9 images. Extending beyond the example, p sets of $h_1, h_2, \ldots$ and h_p protocentroids can represent up to $\prod_{i=1}^{p} h_i$ centroids.

A naïve approach to obtain such a succinct clustering-based description of a dataset would be to start from a set of centroids and subsequently extract an approximation of the centroids satisfying the Khatri-Rao structure. However, multiple centroid-based summaries may achieve nearly the same succinctness and accuracy, some of which may (approximately or exactly) admit an even more succinct representation using Khatri–Rao operators, while others may not. Thus, the Khatri-Rao clustering paradigm we introduce does not start by finding centroids through standard approaches, but extends standard approaches to directly target data summaries satisfying the Khatri-Rao structure.

As a first concrete instantiation of the Khatri-Rao clustering paradigm, we design KHATRI-RAO-k-MEANS, which builds on the most popular algorithm for centroid-based clustering, i.e., the k-MEANS algorithm [38], to find centroids that can be succinctly expressed as the Khatri-Rao sum or product of two or more sets of protocentroids. Although KHATRI-RAO-k-MEANS can result in considerably more succinct and yet equally accurate data summaries than standard k-MEANS, it is more prone to converge to undesirable local minima. Finding high-quality solutions may require exploring many different initializations. Motivated by this limitation, we introduce the Khatri-Rao deep clustering framework, which starts from KHATRI-RAO-k-MEANS and extends the Khatri-Rao paradigm to deep clustering. Deep clustering effectively handles large-scale and high-dimensional data by incorporating deep-learning-based representation learning. By aligning the learned representations with the Khatri–Rao structure, the Khatri-Rao deep clustering framework overcomes the limitations of KHATRI-RAO-k-MEANS, and usually provides data summaries that have a comparable accuracy to unconstrained baselines but are consistently more succinct.

Figure 2 anticipates the benefits of Khatri-Rao clustering by comparing k-MEANS and the deep clustering algorithms IMPROVED DEEP EMBEDDED CLUSTERING (IDEC) [20] and DEEP-K-MEANS

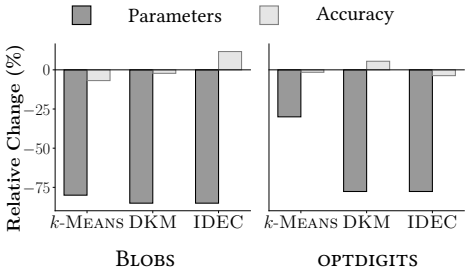

Figure 2: For a synthetic (BLOBS) and a real (OPTDIGITS) dataset, we show relative percentage changes in unsupervised clustering accuracy and parameter count for clustering solutions from algorithms based on the Khatri-Rao paradigm relative to the baseline algorithms k-MEANS, DEEP-K-MEANS (DKM) and IMPROVED DEEP EMBEDDED CLUSTERING (IDEC).

(DKM) [15] against their extensions based on the Khatri-Rao paradigm. The Khatri-Rao clustering algorithms maintain comparable accuracy (here measured by *unsupervised clustering accuracy* [59]) and simultaneously achieve a high level of compression.

Contributions. Our contributions can be summarized as follows.

- We formalize the Khatri-Rao clustering paradigm, outlining its general principles and formulating the Khatri-Rao k-Means and Khatri-Rao deep clustering problems as concrete instantiations of the paradigm.
- We introduce the KHATRI-RAO-k-MEANS clustering algorithm and the Khatri-Rao deep clustering framework to effectively address the formulated problems.
- Through extensive experiments, we show that KHATRI-RAO-k-MEANS can improve the trade-off between data summary size and accuracy over standard k-MEANS. Even more remarkably, Khatri–Rao deep clustering can compress the data summaries produced by deep clustering algorithms by up to 85% with negligible loss in accuracy.

Roadmap. The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the background. Section 4 formalizes the Khatri-Rao clustering paradigm, and introduces the Khatri-Rao k-Means and Khatri-Rao deep clustering problem statements. Section 5 discusses a naïve approach to Khatri-Rao clustering. Sections 6 and 7 present the KHATRI-RAO-k-MEANS algorithm and the Khatri-Rao deep clustering framework, respectively. Section 8 guides design choices in Khatri-Rao clustering, and Section 9 contains the experimental evaluation. Finally, conclusions are drawn in Section 10. An appendix and our implementations are available online.[1]

2 Related Work

In this section, we review work relevant to the present study.

Clustering, data summarization and compression. Obtaining data summaries that are as succinct and as accurate as possible has established itself as a central problem in such domains as database systems under various names. It is useful to distinguish between data summarization and compression. The goal of data summarization is to craft a summary of a dataset that captures its essential patterns. Centroid-based clustering is a popular approach for data summarization, although alternative approaches exist (e.g., aggregation, dimensionality reduction, or sampling)

[1] https://github.com/maciap/KhatriRaoClustering

and may better suit particular applications [1]. A related field to data summarization is data compression, which emphasizes reducing data size for storage or transmission, either with or without information loss, over capturing the main patterns in the data [27]. The Khatri-Rao clustering paradigm effectively compresses the data summaries given by centroid-based clustering. Thus, our work pertains to both data summarization and compression. We henceforth use "compression" to refer specifically to the reduction of the size of a data summary.

Research has leveraged information-theoretic principles to establish a theoretical foundation for understanding clustering as a compression problem, explicitly considering the trade-off between compression, efficiency and clustering quality [6, 45, 51]. However, there is a lack of practical methods designed to enhance the compression capabilities of clustering, a gap we aim to address in this work. We concentrate on two central approaches to centroid-based clustering, k-Means and deep clustering.

k-Means clustering. k-Means is a seminal problem in clustering [38], and the classic k-Means algorithm has established itself as the cornerstone of many clustering algorithms [26]. More details on the k-Means problem and on the classic k-Means algorithm are given in Section 3. Today, k-Means clustering is still an active topic of research [25]. Nonetheless, prior to our work which extends k-Means to the Khatri-Rao paradigm, improving the compression power of k-Means has been largely overlooked.

Deep clustering. Unlike k-Means, which performs clustering directly in the input space, recent deep clustering algorithms combine clustering with neural networks to learn representations more amenable to clustering [35, 46, 60].

Deep clustering algorithms can use various types of neural networks. We focus on autoencoder-based algorithms as they are agnostic to data type and easily extendable [36]. Centroid-based deep clustering algorithms are useful for summarization purposes and have demonstrated state-of-the-art performance [40]. A well-known representative is Deep-k-Means (DKM) [15]. DKM is similar to k-Means. However, it introduces soft cluster assignments based on the softmax function. Deep Embedded Clustering (DEC) [57] and its successor Improved Deep Embedded Clustering (IDEC) [20] utilize the Kullback-Leibler divergence [31] to learn an embedding that aligns a *Student's t* model of the data distribution with a target distribution. Our work extends the DKM and IDEC algorithms to the Khatri-Rao paradigm. Further details on deep clustering, DKM and IDEC are given in Section 3.

Matrix decomposition. Clustering is closely related to matrix decomposition, which expresses a matrix as a product of two simpler matrices. It is known that, given a data matrix, the model imposed by a centroid-based clustering algorithm like k-Means can be seen as a special case of a matrix-decomposition model where one of the factor matrices is constrained to indicate cluster assignments and the other stores centroids [12]. Similarly, Khatri-Rao clustering hinges on known operators like Hadamard (i.e., elementwise) and Khatri-Rao products [28] and is closely connected to the Hadamard decomposition, which models a data matrix as the Hadamard product of matrix decompositions [9]. In particular, the model imposed by Khatri-Rao clustering with protocentroids aggregated via elementwise product on a data matrix is equivalent to a Hadamard decomposition where one matrix in each decomposition is constrained to indicate the assignment to protocentroids. In general, algorithms for matrix decomposition can be useful for clustering [37] and can also be potentially extended to the Khatri-Rao clustering setting.

3 Preliminaries

Before we introduce the main ideas behind the proposed methods, we present the notation and preliminary notions that are used throughout the paper. Additionally, we review the essential clustering background required to understand our contributions.

Notation and basic definitions. Vectors are denoted by lowercase bold letters, and matrices by upper-case bold letters. Similarly, sets are denoted by upper-case letters and scalars by lowercase letters. Greek letters are reserved for data-summary parameters. The product of two scalars a and b is denoted by ab, while $\mathbf{AB}$ and $\mathbf{A} \odot \mathbf{B}$ denote the standard and Hadamard matrix products, respectively. The Euclidean norm of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$.

We consider datasets $\mathcal{D} \subset \mathbb{R}^m$ composed of a set of data points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i = (x_1, x_2 \ldots, x_m)$. The goal of clustering is to divide the dataset into k clusters $\{C_1, C_2, \ldots, C_k\}$ so that $\mathcal{D} = \cup_{i=1}^{k} C_i$ and $C_i \cap_{i \neq j} C_j = \emptyset$. Furthermore, we denote the set of cluster centroids by $M_{\boldsymbol{\mu}} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\} \subset \mathbb{R}^m$. In the Khatri-Rao clustering paradigm, we define the centroids in $M_{\boldsymbol{\mu}}$ more succinctly by combining protocentroids. Unless stated otherwise, we assume that there are k centroids and p sets of protocentroids, where the i-th set has cardinality h_i. The protocentroids in different sets are combined using a function referred to as *aggregator* and denoted by $\oplus$. While the Khatri-Rao clustering paradigm is general and could in principle accommodate any aggregator function, in this work, we focus on the sum and product, i.e., $\oplus \in \{+, \times\}$. When $\oplus$ is left unspecified, it is understood to represent an arbitrary choice of aggregator. For notational convenience, $\oplus$ is applied to scalars, vectors, matrices and sets. When applied to vectors and matrices, it is an elementwise operator, corresponding to the standard sum for $\oplus = +$ and to the Hadamard product for $\oplus = \times$. When applied to sets (e.g., sets of protocentroids), it is a so-called Khatri-Rao operator. We define the Khatri-Rao $\oplus$ operator as an operator that, given p sets of vectors, produces the set of all vectors obtained by elementwise application of $\oplus$ to all possible combinations with one vector from each set. The name "Khatri-Rao" operator is chosen since, if protocentroids in each set are arranged as rows of matrices and $\oplus = \times$, the operator reduces to the Khatri-Rao product [28].

Clustering for data summarization. From the perspective of this work, clustering algorithms are functions $f_C : \mathcal{D} \rightarrow \Theta$ mapping a dataset $\mathcal{D}$ to a succinct representation Θ. The function f_C is chosen to minimize the size of Θ while optimizing a measure of quality $Q_C(\mathcal{D}, \Theta)$. All clustering algorithms discussed in this work follow from particular constraints imposed on Θ as well as different choices of $Q_C(\mathcal{D}, \Theta)$.

While the Khatri-Rao paradigm can be applied to any centroid-based clustering algorithm, we focus on two popular approaches to clustering: the seminal k-Means clustering and the emerging centroid-based deep clustering. In the remainder of this section, we review the basic principles underlying both approaches.

k-Means clustering. In k-Means, the dataset is summarized in terms of a set of centroids $\Theta = \{\boldsymbol{\mu}_i\}_{i=1}^{k}$, and the measure of quality is the total squared Euclidean distance of each point to the closest centroid, henceforth called *inertia*, as is standard in the literature [43].

Let $\boldsymbol{\mu}_i$ be the centroid of cluster C_i. The k-Means clustering problem asks to partition a dataset $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ in $\mathbb{R}^m$ into k clusters $\{C_1, C_2, \ldots, C_k\}$ such that the inertia

$$Q_C(\mathcal{D}, \Theta) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{1}$$

is minimized.

To address this problem, the standard k-MEANS algorithm starts by initializing k cluster centroids. Centroids can be initialized by sampling data points uniformly at random. Alternatively, the popular k-MEANS++ [2] strategy chooses data points that are *sufficiently* far apart from each other as initial centroids, which gives theoretical approximation guarantees and often results in performance improvements. After initialization, the k-MEANS algorithm iteratively assigns each data point to its nearest (in Euclidean distance) centroid and updates the cluster centroids by computing the mean of the points assigned to each cluster. The algorithm terminates and outputs the current centroids when either the centroids converge to a stable position or a maximum number of iterations is reached.

Deep clustering. Deep clustering combines clustering with deep neural networks to perform some kind of representation learning. Unlike more traditional methods that rely on fixed features, it jointly optimizes feature extraction and cluster assignment in an end-to-end manner. In this study, we concentrate on autoencoder-based, centroid-based deep clustering approaches. In this setting, the input dataset $\mathcal{D}$ is summarized as $\Theta = \Theta_{\boldsymbol{\alpha}} \cup \Theta_{\boldsymbol{\mu}}$, where $\Theta_{\boldsymbol{\alpha}}$ and $\Theta_{\boldsymbol{\mu}}$ denote the autoencoder and centroid parameters, respectively, and the quality function Q_C to optimize captures the trade-off between clustering quality in the latent space and quality of reconstruction of $\mathcal{D}$ from its latent-space representation.

Formally, given a dataset $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, let $f^e_{\Theta_{\boldsymbol{\alpha}}} : \mathcal{D} \to Z$ be a parametric mapping (encoder) to latent representations $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\} \subset \mathbb{R}^{m'}$ and $f^d_{\Theta_{\boldsymbol{\alpha}}} : Z \to \hat{\mathcal{D}}$ a mapping (decoder) back to the original feature space. Further, let $\mathcal{L}_{\text{rec}}(\mathcal{D}, f^d_{\Theta_{\boldsymbol{\alpha}}}(Z))$ denote a reconstruction loss and $\mathcal{L}_{\text{cluster}}$ a loss capturing clustering quality in the latent space. Then, the deep clustering problem can be framed as the problem of minimizing

$$Q_C(\mathcal{D}, \Theta) = \mathcal{L}_{\text{cluster}}(Z, \Theta_{\boldsymbol{\mu}}) + w_{rec}\mathcal{L}_{\text{rec}}(\mathcal{D}, f^d_{\Theta_{\boldsymbol{\alpha}}}(Z)), \quad (2)$$

where $\Theta_{\boldsymbol{\mu}} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$ denotes the cluster centroids in the latent space and $w_{rec} \geq 0$ balances clustering and representation learning. In our study, we consider two clustering loss functions for $\mathcal{L}_{\text{cluster}}$; the ones proposed for the DKM [15] and IDEC [20] algorithms. Thus, $\mathcal{L}_{\text{cluster}} \in \{\mathcal{L}_{\text{DKM}}, \mathcal{L}_{\text{IDEC}}\}$, where

$$\mathcal{L}_{\text{DKM}}(Z, \Theta_{\boldsymbol{\mu}}) = \frac{1}{n} \sum_{z \in Z} \sum_{i=1}^{k} ||\mathbf{z} - \boldsymbol{\mu}_i||^2 \frac{e^{-a||\mathbf{z}-\boldsymbol{\mu}_i||^2}}{\sum_{j=1}^{k} e^{-a||\mathbf{z}-\boldsymbol{\mu}_j||^2}} \quad (3)$$

and

$$\mathcal{L}_{\text{IDEC}}(Z, \Theta_{\boldsymbol{\mu}}) = \frac{1}{n} \sum_{l=1}^{n} \sum_{i=1}^{k} p_{l,i} \log(\frac{p_{l,i}}{q_{l,i}}). \quad (4)$$

Here,

$$q_{l,i} = \frac{(1 + ||\mathbf{z}_l - \boldsymbol{\mu}_i||_2^2)^{-\frac{a+1}{2}}}{\sum_{j=1}^{k}(1 + ||\mathbf{z}_l - \boldsymbol{\mu}_j||_2^2)^{-\frac{a+1}{2}}} \text{ and } p_{l,i} = \frac{q_{l,i}^2 / \sum_{t=1}^{n} q_{t,i}}{\sum_{j=1}^{k}(q_{l,j}^2 / \sum_{t=1}^{n} q_{t,j})},$$

with $q_{l,i}$ and $p_{l,i}$ representing the data and target distributions, respectively, and a is an algorithm-specific parameter that is usually set to 1 for IDEC and 1000 for DKM. To obtain a succinct data summary in terms of autoencoder and centroid parameters, the objective in Equation (2) is typically optimized via batch-wise backpropagation, using automatic differentiation [35].

4 Problem Formulation

Given any clustering algorithm $f_C : \mathcal{D} \to \Theta$, we introduce its Khatri-Rao extension by adopting its quality function $Q_C(\mathcal{D}, \Theta)$ and introducing particular constraints on the parameters Θ that define the summary of $\mathcal{D}$ associated with f_C.

As the Khatri-Rao clustering paradigm is designed for centroid-based clustering, all algorithms that can be framed within the Khatri-Rao clustering framework use a set of parameters to represent centroids. Thus, in all cases, given a user-specified integer p, Khatri-Rao clustering assumes that each centroid satisfies

$$\boldsymbol{\mu}_i = \boldsymbol{\theta}_1^{j_1} \oplus \boldsymbol{\theta}_2^{j_2} \ldots \oplus \boldsymbol{\theta}_p^{j_p} \quad \forall i \in [1, k], \quad (5)$$

where $\boldsymbol{\theta}_q^{j_l}$ denotes the j_l-th protocentroid vector in the q-th set of protocentroids. Accordingly, each centroid is associated with a cluster C_i and is uniquely identified by a tuple of p indices, one for each set of protocentroids, so that $C_i = C_{j_1, j_2, \ldots j_p}$, and similarly $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{j_1, j_2, \ldots j_p}$. Figure 3 provides a schematic visualization describing how sets of protocentroids are combined to create cluster centroids. Additionally, Figure 4 shows examples in synthetic data.[2]

All algorithms based on the Khatri-Rao clustering paradigm yield succinct representations of centroid parameters by enforcing the constraint from Equation (5). In some cases, as in the deep clustering setting, there are additional parameters (the autoencoder parameters) that Khatri-Rao clustering seeks to reduce.

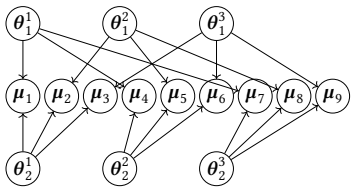

Figure 3: Diagram showing the interactions of two sets of protocentroids to generate cluster centroids.

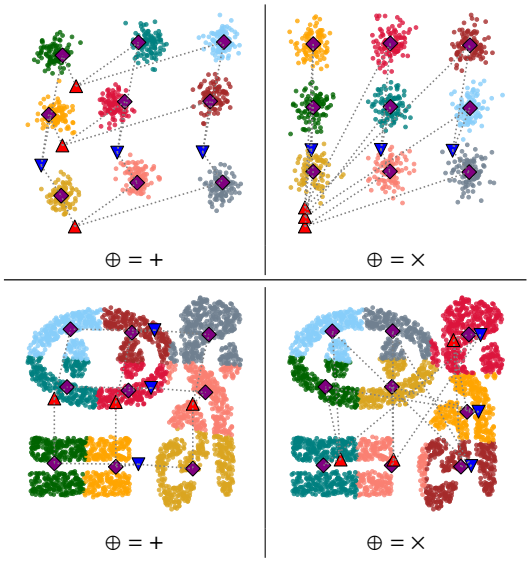

Figure 4: Khatri-Rao-based (top) and arbitrarily-structured (bottom) synthetic data. Combining additively (⊕ = +) or multiplicatively (⊕ = ×) the first (red triangles) and second (blue triangles) sets of protocentroids yields the cluster centroids (purple diamonds). Gray lines indicate which protocentroid affects each cluster centroid.

[2]Bottom dataset available at https://github.com/milaan9/Clustering-Datasets

Although Khatri-Rao clustering defines a general paradigm for centroid-based clustering, we focus on k-Means and two deep clustering algorithms. Khatri-Rao extensions of other centroid-based clustering algorithms, e.g., based on matrix decomposition [37] or gradient descent [49], are possible but require method-specific adjustments. Investigating whether our ideas can also be applied to clustering algorithms that do not rely on centroids, such as hierarchical clustering [18], is left to future research.

4.1 Khatri-Rao k-Means Clustering

In k-Means clustering (introduced in Section 3), the dataset is summarized using a set of prototype vectors, i.e., the centroids $\Theta = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$, and the objective function Q_C is the inertia in Equation (1).

Khatri-Rao k-Means also seeks to minimize the inertia. However, it restricts all centroids Θ to arise from Khatri-Rao aggregations of protocentroids, so that $\Theta = \{\boldsymbol{\theta}_l^{j_i} \mid j_i \in [1, h_l] \text{ and } l \in [1, p]\}$. Formally, the Khatri-Rao k-Means problem is defined as follows.

Problem 1 (Khatri-Rao k-Means). *Given a dataset and an input $p \in \mathbb{N}_+$, partition $\mathcal{D}$ into $\prod_{l=1}^{p} h_l$ clusters so as to minimize*

$$Q_C(\mathcal{D}, \Theta) = \sum_{j_1=1}^{h_1} \sum_{j_2=1}^{h_2} \cdots \sum_{j_p=1}^{h_p} \sum_{\mathbf{x} \in C_{j_1, j_2, \ldots j_p}} \|\mathbf{x} - \boldsymbol{\theta}_1^{j_1} \oplus \boldsymbol{\theta}_2^{j_2} \ldots \oplus \boldsymbol{\theta}_p^{j_p}\|^2.$$

The classic k-Means problem is known to be **NP**-Hard [11]. Any instance of k-Means can be mapped to an instance of Khatri-Rao k-Means simply by setting $p = 1$ and $h_1 = k$. Thus, Problem 1 is also **NP**-hard.

In view of this hardness, as it is customary for clustering methods, in Section 6, we devise effective algorithms f_C that do not seek a globally optimal solution. Figure 5 (**a**) provides a schematic representation of the Khatri-Rao k-Means problem.

4.2 Khatri-Rao Deep Clustering

As explained in Section 3, centroid-based deep clustering algorithms f_C map datasets $\mathcal{D}$ to summaries in the form $\Theta = \Theta_{\boldsymbol{\alpha}} \cup \Theta_{\boldsymbol{\mu}}$, where $\Theta_{\boldsymbol{\alpha}}$ and $\Theta_{\boldsymbol{\mu}}$ are autoencoder and centroid parameters, respectively.

The Khatri-Rao extension of the deep clustering problem adopts the same objective function Q_C, while enforcing constraints on Θ to craft a succinct representation. As in Khatri-Rao k-Means, (latent-space) centroids are conjectured to follow the Khatri-Rao structure, i.e.: $\Theta_\mu = \{\boldsymbol{\theta}_l^{j_i} \mid j_i \in [1, h_l] \text{ and } l \in [1, p]\}$. In addition, we also enforce a succinct representation of the autoencoder parameters.

In the autoencoder-based deep clustering setting, autoencoders consist of an *encoder* $f^e_{\Theta_{\boldsymbol{\alpha}}} : \mathcal{D} \to Z$ mapping data to latent representations $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, and a *decoder* $f^d_{\Theta_{\boldsymbol{\alpha}}} : Z \to \hat{\mathcal{D}}$, reconstructing data from their latent representations. We consider an autoencoder with n_l layers. The l-th layer is parameterized by a matrix $\mathbf{W}_l \in \mathbb{R}^{d_l \times m_l}$, which could be relatively large. To obtain a succinct representation of the output of deep clustering algorithms, it is thus not sufficient to compress the centroid parameters, but it is also necessary to compress the autoencoder parameters $\Theta_{\boldsymbol{\alpha}} = \{\mathbf{W}_l\}_{l=1}^{n_l}$. To this aim, we draw on the connection between Khatri-Rao clustering and Hadamard decomposition [9] discussed in Section 2, and we reparametrize matrices $\mathbf{W}_l$ as Hadamard products of q factors, i.e.:

$$\mathbf{W}_l = (\mathbf{A}_1^l \mathbf{B}_1^l) \odot (\mathbf{A}_2^l \mathbf{B}_2^l) \ldots (\mathbf{A}_q^l \mathbf{B}_q^l) \quad \forall l \in [1, n_l], \tag{6}$$

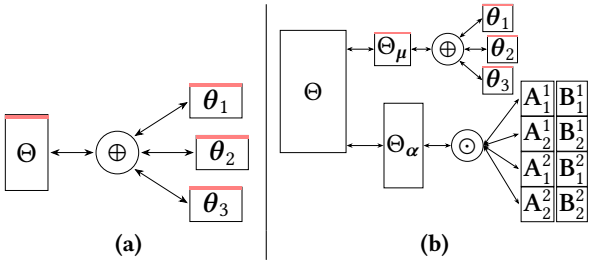

Figure 5: Diagram summarizing the Khatri-Rao clustering paradigm in the k-Means (a) and deep clustering (b) settings; in this example, $p = 3$, $n_l = 2$ and $q = 2$. The centroid in red is obtained by aggregating the protocentroids in red.

where $\mathbf{A}_i^l \in \mathbb{R}^{d_l \times r_i}$ and $\mathbf{B}_i^l \in \mathbb{R}^{r_i \times m_l}$. The key intuition behind such reparameterization mirrors that of Khatri-Rao clustering. The Hadamard product of q matrices of ranks $r_1, r_2, \ldots$ and r_q can represent matrices of rank up to $\prod_{l=1}^{q} r_l$ while using only $2\sum_{l=1}^{q} r_l$ vectors. In practice, this provides an effective compression mechanism, as also demonstrated in the context of computer vision in federated learning environments [24], where a similar reparameterization is considered.

Having illustrated the constraints that Khatri-Rao deep clustering poses on $\Theta_{\boldsymbol{\mu}}$ and $\Theta_{\boldsymbol{\alpha}}$, we can now formally state the Khatri-Rao deep clustering problem.

Problem 2. *Given a dataset $\mathcal{D}$, a parametric mapping $f^e_{\Theta_{\boldsymbol{\alpha}}} : \mathcal{D} \to Z$ to latent representations $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, a mapping $f^d_{\Theta_{\boldsymbol{\alpha}}} : Z \to \hat{\mathcal{D}}$ back to the original feature space and input $p \in \mathbb{N}_+$, partition $\mathcal{D}$ into $h_1 h_2 \ldots h_p$ clusters so as to minimize*

$$Q_C(\mathcal{D}, \Theta) = \mathcal{L}_{\text{cluster}}(Z, \Theta_{\boldsymbol{\mu}}) + w_{rec} \mathcal{L}_{\text{rec}}(\mathcal{D}, f^d_{\Theta_{\boldsymbol{\alpha}}}(Z)),$$

where $\Theta_\mu = \{\boldsymbol{\theta}_l^{j_i} \mid j_i \in [1, h_l]$ and $l \in [1, p]\}$ and $\Theta_{\boldsymbol{\alpha}} = \{\mathbf{A}_j^i, \mathbf{B}_j^i \mid i \in [1, n_l]$ and $j \in [1, q]\}$.

Considering the loss $\mathcal{L}_{cluster}$ in Equation (3) optimized by DKM, it can be defined as follows for Khatri-Rao DKM:

$$\mathcal{L}_{\text{KR-DKM}}(\mathcal{D}, \Theta_{\boldsymbol{\mu}}) = \frac{1}{n} \sum_{z \in Z} \sum_{j_1=1}^{h_1} \cdots \sum_{j_p=1}^{h_p} ||\mathbf{z} - \boldsymbol{\theta}_1^{j_1} \ldots \oplus \boldsymbol{\theta}_p^{j_p}||^2 \cdot \frac{e^{-a||\mathbf{z} - \boldsymbol{\theta}_1^{j_1} \ldots \oplus \boldsymbol{\theta}_p^{j_p}||^2}}{\sum_{l_1=1}^{h_1} \cdots \sum_{l_p=1}^{h_p} e^{-a||\mathbf{z} - \boldsymbol{\theta}_1^{l_1} \ldots \oplus \boldsymbol{\theta}_p^{l_p}||^2}}.$$

Adjusting the loss $\mathcal{L}_{cluster}$ in Equation (4) for the IDEC algorithm to the Khatri-Rao paradigm follows the same logic. Figure 5 (**b**) summarizes the Khatri-Rao deep clustering problem. Our solution to the problem is presented in Section 7.

5 Naïve Khatri-Rao Clustering

As anticipated in Section 1, a naïve approach to Khatri-Rao clustering first applies a standard clustering algorithm (e.g., k-Means) to obtain an initial set of centroids, and then post-processes these centroids to impose the succinct Khatri-Rao structure.

In this section, we describe such a naïve solution to Khatri-Rao clustering. As for the rest of the algorithms we discuss, for clarity of exposition, we consider the simple case of two sets of protocentroids, i.e., $p = 2$. The extension to the general formulation is straightforward but more cumbersome in notation.

Let $M_{\boldsymbol{\mu}}$ denote the set of centroids output by a centroid-based clustering algorithm like k-Means. Given $M_{\boldsymbol{\mu}}$, we aim to find the

closest sets of protocentroids that yield M_{μ} upon aggregation, which translates into the following optimization problem:

$$\min_{\{\boldsymbol{\theta}_1^i\}_{i=1}^{h_1},\{\boldsymbol{\theta}_2^j\}_{j=1}^{h_2}} \sum_{i,j} \|\boldsymbol{\mu}_{i,j} - \boldsymbol{\theta}_1^i \oplus \boldsymbol{\theta}_2^j\|^2.$$

This problem formulation suggests that protocentroids could potentially be estimated via an alternating gradient-descent procedure where, at each iteration, protocentroids from one set are updated by taking a step in the opposite direction of the gradient.

If the aggregator function $\oplus$ is the sum or the product, gradients can be computed in closed form. For example, assume that $\oplus$ is the product. Then, at any given iteration, the gradient with respect to, e.g., the i-th protocentroid in the first set $\boldsymbol{\theta}_1^i$ would be

$$2\sum_{j=1}^{h_2}\left(\boldsymbol{\mu}_{i,j} - \boldsymbol{\theta}_1^i \odot \boldsymbol{\theta}_2^j\right) \odot \boldsymbol{\theta}_2^j, \tag{7}$$

where we have used the fact that only h_2 centroids are affected by $\boldsymbol{\theta}_1^i$ (i.e., those corresponding to the combination of $\boldsymbol{\theta}_1^i$ with all protocentroids in the second set), and hence contribute to the gradient. The gradient in Equation (7) implies that, at any given iteration, the update rule for the i-th protocentroid in the first set $\boldsymbol{\theta}_1^i$, obtained by equating the corresponding gradient to $\mathbf{0}$, is

$$\boldsymbol{\theta}_1^i \leftarrow \frac{\sum_{j=1}^{h_2} \boldsymbol{\mu}_{i,j} \odot \boldsymbol{\theta}_2^j}{\sum_{j=1}^{h_2} \boldsymbol{\theta}_2^j \odot \boldsymbol{\theta}_2^j}. \tag{8}$$

Update rules for the sum aggregator and alternative configurations of protocentroid sets can be obtained analogously.

Limitations. While the gradient-descent algorithm described above can be effective for estimating protocentroids associated with a given set of centroids, the performance of the two-phase naïve approach to Khatri-Rao clustering is often far from satisfactory. This is because the centroids obtained in the first step may accurately describe the dataset, yet be arbitrarily far from a Khatri-Rao structure. As a consequence, the accuracy of the data summary found in the first phase can be destroyed in the second phase.

Rather than using a two-phase approach, we need algorithms that simultaneously optimize clustering and enforce the Khatri-Rao structure. The next section illustrates one such algorithm.

6 The Khatri-Rao-k-Means Algorithm

In this section, we present the Khatri-Rao-k-Means algorithm, an extension of the standard k-Means algorithm that addresses Problem 1 while overcoming the limitations of the naïve approach. The key steps of Khatri-Rao-k-Means are similar, in spirit, to those of standard k-Means outlined in Section 3. However, Khatri-Rao-k-Means leverages the Khatri-Rao structure to achieve a more succinct representation of the centroids, which requires modifying the steps of standard k-Means.

A detailed description of the steps of Khatri-Rao-k-Means follows, and Algorithm 1 summarizes them. For clarity, we again focus on the scenario where we have two sets of protocentroids.

Initialization. Instead of centroids, Khatri-Rao-k-Means starts by initializing protocentroids. As for standard k-Means, a simple initialization strategy involves choosing initial protocentroids by sampling data points uniformly at random. Alternatively, we can adapt the more effective initialization of k-Means++ to be compatible with Khatri-Rao-k-Means. To achieve this, we sample $h_1 + h_2$ centroids $\boldsymbol{\mu}_{j_1,j_2}$ such that they are far from each other, and for each sampled centroid, we generate two protocentroids such that $\boldsymbol{\mu}_{j_1,j_2} = \boldsymbol{\theta}_1^{j_1} \oplus \boldsymbol{\theta}_2^{j_2}$. Because of the constraints imposed in Khatri-Rao clustering, the remaining initial $h_1h_2 - (h_1 + h_2)$ centroids are determined by the choice of the first $h_1 + h_2$. For simplicity, in Algorithm 1, random sampling is used at the initialization stage.

Algorithm 1 Khatri-Rao-k-Means
(for two sets of protocentroids, i.e., $p = 2$)

Input: dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$, protocentroid sets cardinalities h_1, h_2, maximum number of iterations n_{iter}, tolerance ϵ.
Output: assignments $C_{\cdot,\cdot}$, protocentroids $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$.
$\boldsymbol{\theta}_1 \leftarrow$ SampleRandomly$(\mathcal{D}, h_1)$ ▷ sample h_1 points from $\mathcal{D}$
$\boldsymbol{\theta}_2 \leftarrow$ SampleRandomly$(\mathcal{D}, h_2)$ ▷ sample h_2 points from $\mathcal{D}$
$\boldsymbol{\theta}_1^{\text{old}} \leftarrow \boldsymbol{\theta}_1,\ \boldsymbol{\theta}_2^{\text{old}} \leftarrow \boldsymbol{\theta}_2$
for $it = 1$ **to** n_{iter} **do**
  $\mathbf{a}_1 \leftarrow \mathbf{0},\ \mathbf{a}_2 \leftarrow \mathbf{0},\ \mathbf{d}_{\min} \leftarrow \infty$
  for $i = 1$ **to** h_1 **do**
    for $j = 1$ **to** h_2 **do**
      $\boldsymbol{\mu}_{ij} \leftarrow \boldsymbol{\theta}_1^i \oplus \boldsymbol{\theta}_2^j$ ▷ centroid computed on the fly
      $\mathbf{d}_{ij} \leftarrow \|\mathcal{D} - \boldsymbol{\mu}_{ij}\|^2$ ▷ compute distances
      $\mathbf{F} \leftarrow (\mathbf{d}_{ij} < \mathbf{d}_{\min})$ ▷ flag relevant samples
      $\mathbf{a}_1[\mathbf{F}] \leftarrow i,\ \mathbf{a}_2[\mathbf{F}] \leftarrow j$ ▷ update assignments
      $\mathbf{d}_{\min}[\mathbf{F}] \leftarrow \mathbf{d}_{ij}[\mathbf{F}]$
  $C_{i,j} \leftarrow \{\mathbf{x}_t \mid t \in \{1, 2 \dots n\} \wedge i = \mathbf{a}_{1_t} \wedge j = \mathbf{a}_{2_t}\}$
  for $i = 1$ **to** h_1 **do** ▷ update protocentroids
    $\boldsymbol{\theta}_1^i \leftarrow \arg\min_{\boldsymbol{\theta}_1} \sum_{l=1}^{h_2} \sum_{\mathbf{x} \in C_{i,l}} (\mathbf{x} - \boldsymbol{\theta}_1 \oplus \boldsymbol{\theta}_2^l)^2$
  for $j = 1$ **to** h_2 **do**
    $\boldsymbol{\theta}_2^j \leftarrow \arg\min_{\boldsymbol{\theta}_2} \sum_{l=1}^{h_1} \sum_{\mathbf{x} \in C_{l,j}} (\mathbf{x} - \boldsymbol{\theta}_1^l \oplus \boldsymbol{\theta}_2)^2$
  $\Delta \leftarrow \sum_{i=1}^{h_1} \sum_{j=1}^{h_2} \left\| \boldsymbol{\theta}_1^i \oplus \boldsymbol{\theta}_2^j - \boldsymbol{\theta}_1^{\text{old},i} \oplus \boldsymbol{\theta}_2^{\text{old},j} \right\|^2$
  if $\Delta < \epsilon$ **then** ▷ check stopping condition
    break
  $\boldsymbol{\theta}_1^{\text{old}} \leftarrow \boldsymbol{\theta}_1,\ \boldsymbol{\theta}_2^{\text{old}} \leftarrow \boldsymbol{\theta}_2$
Return: assignments $C_{\cdot,\cdot}$ and protocentroids $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$.

Centroid computation. The centroids are obtained on the fly at each iteration by simply aggregating protocentroids according to the chosen aggregator function.

Assignment update. Following the update of the centroids, at each iteration, just like k-Means, Khatri-Rao-k-Means updates cluster assignments based on the latest set of centroids. This step is accomplished by assigning each observation to the centroid it is closest to according to the Euclidean distance.

Since each centroid index is uniquely associated with a tuple of protocentroid indices, the assignments of data points to protocentroids readily follow.

Protocentroid update. The standard k-Means algorithm updates cluster centroids by computing cluster means since the cluster means minimize the sum of squared distances to the centroid within each cluster (i.e., the cluster-specific contribution to the total inertia) based on the latest cluster assignments. Likewise, Khatri-Rao-k-Means updates protocentroids by considering the same optimization. Nevertheless, as a consequence of the constraints imposed on the centroids, the optimal updates for Khatri-Rao-k-Means are not obtained by merely averaging, as stated in Proposition 6.1 (proof in Appendix C) for the sum and product aggregators.

Proposition 6.1. *The optimal updates of the j-th protocentroid in the first and second set of protocentroids at any iteration of*

Khatri-Rao-k-Means are given by

$$\boldsymbol{\theta}_1^j = \frac{\sum_{l=1}^{h_2} \sum_{\mathbf{x} \in C_{j,l}} \mathbf{x} \odot \boldsymbol{\theta}_2^l}{\sum_{l=1}^{h_2} |C_{j,l}| \boldsymbol{\theta}_2^l \odot \boldsymbol{\theta}_2^l} \text{ and } \boldsymbol{\theta}_2^j = \frac{\sum_{l=1}^{h_1} \sum_{\mathbf{x} \in C_{l,j}} \mathbf{x} \odot \boldsymbol{\theta}_1^l}{\sum_{l=1}^{h_1} |C_{l,j}| \boldsymbol{\theta}_1^l \odot \boldsymbol{\theta}_1^l},$$

if $\oplus = \times$*, or if* $\oplus = +$*:*

$$\boldsymbol{\theta}_1^j = \frac{\sum_{l=1}^{h_2} \sum_{\mathbf{x} \in C_{j,l}} (\mathbf{x} - \boldsymbol{\theta}_2^l)}{\sum_{l=1}^{h_2} |C_{j,l}|} \text{ and } \boldsymbol{\theta}_2^j = \frac{\sum_{l=1}^{h_1} \sum_{\mathbf{x} \in C_{l,j}} (\mathbf{x} - \boldsymbol{\theta}_1^l)}{\sum_{l=1}^{h_1} |C_{l,j}|}.$$

Termination. Like k-Means, Khatri-Rao-k-Means stops when either the protocentroids move less than a tolerance threshold ϵ or a maximum number of iterations n_{iter} is reached.

Complexity. The updates of the protocentroids, i.e., Line 17 and Line 19 in Algorithm 1, can be efficiently computed in closed form as per Proposition 6.1. In practice, the computations are sped up by keeping track of the assignments of each data point for both sets of protocentroids and only considering points that are assigned to the protocentroid to update.

Like for the standard k-Means algorithm, the computational bottleneck is the computation of the distances for the assignment in Line 11. Khatri-Rao-k-Means with $h_1 + h_2$ protocentroids has the same time complexity as k-Means with $h_1 h_2$ centroids, namely $O(nmh_1h_2)$ per iteration. Considering dataset storage, Khatri-Rao-k-Means requires $O\big((n + h_1 + h_2)m\big)$ space, which is less than the $O((n + h_1h_2)m)$ required by k-Means, as long as h_1 and h_2 are larger than 2. Hence, Khatri-Rao-k-Means can be more space-efficient than k-Means in applications with a large number of clusters. If memory is not a concern, one can opt for a time-efficient implementation of Khatri-Rao-k-Means which stores the full set of h_1h_2 centroids (implementation details in Appendix B). Section 9.3 presents an empirical scalability analysis that corroborates the discussion of time and space complexity.

Limitations. As demonstrated empirically in Section 9, Khatri-Rao-k-Means can provide a more accurate summary of the data than standard k-Means which uses the same number of parameters. However, Khatri-Rao-k-Means has a significant limitation. As it can also be seen from Figures 3 and 4, given two sets of h_1 and h_2 protocentroids, each update of a protocentroid in the first set (Line 17) affects the position of h_2 protocentroids in the second set (Line 19), and vice versa. In the standard k-Means algorithm, instead, centroids for a given assignment are updated independently of each other. The additional rigidity of Khatri-Rao-k-Means makes it more likely to converge to poor local minima compared to standard k-Means.

To overcome the lack of flexibility of Khatri-Rao-k-Means, we start from Khatri-Rao-k-Means and turn to deep clustering, relying on representation learning to perform clustering in a latent space which exhibits both a strong cluster structure and the Khatri-Rao structure, as discussed in the following section.

7 Khatri-Rao Deep Clustering Algorithms

The Khatri-Rao deep clustering problem (Problem 2) requires optimizing a deep-clustering loss function Q_C while compressing the centroid parameters $\Theta_{\boldsymbol{\mu}}$ and the autoencoder parameters $\Theta_{\boldsymbol{\alpha}}$ of a deep clustering algorithm. Unlike for Khatri-Rao-k-Means, addressing the Khatri-Rao deep clustering problem does not require the introduction of a completely novel machinery. The Khatri-Rao deep clustering framework extends standard deep clustering without requiring major adjustments.

Initialization. The initialization of the centroids in the latent space plays an important role in driving the performance of deep clustering algorithms. To find a high-quality set of initial centroids, many deep clustering algorithms, including DKM and IDEC, rely on algorithms like k-Means. Therefore, for the same goal, it is natural for algorithms based on the Khatri-Rao deep clustering framework to use Khatri-Rao-k-Means.

Reparameterization. From an optimization standpoint, the trainable parameters in deep clustering are the autoencoder and centroid parameters. To ensure that the constraints on those parameters placed by the Khatri-Rao deep clustering problem are met, the Khatri-Rao deep clustering framework reparameterizes standard deep clustering imposing the Hadamard-decomposition structure on the autoencoder parameters and the Khatri-Rao structure on the centroid parameters. As in standard deep clustering, all parameters are optimized via batch-wise backpropagation.

From a computational standpoint, Khatri-Rao deep clustering reduces trainable parameters at the cost of few additional operations.

Our experiments reveal that the Khatri-Rao deep clustering framework reduces the size of data summaries found by standard deep clustering algorithms by more than 50% on average in the datasets we consider, at little or no cost in accuracy.

8 Design Choices in Khatri-Rao Clustering

In standard clustering, the number of clusters and centroids is either specified based on domain knowledge, determined by practical constraints (e.g., the available memory for storing centroids), or estimated through data-driven procedures. Given a desired number of centroids to be represented, Khatri-Rao clustering requires choosing the cardinality of each set of protocentroids, the number of sets and the aggregator function. Khatri-Rao deep clustering also requires choosing the number of Hadamard factors and the rank of each factor. In this section, we address these choices, and we also explain how Khatri-Rao clustering can integrate existing techniques to estimate the number of clusters.

Choosing the cardinality of sets of protocentroids. To maximize the number of centroids that can potentially be represented, it is convenient to consider protocentroid sets that are as balanced as possible. In general, p sets of $h_1, h_2, \dots h_p$ protocentroids can represent $\prod_{i=1}^{p} h_i$ centroids. Given a budget b of ph vectors, allocating them in p sets can represent h^p centroids. Any other allocation results in the potential representation of $\prod_{i=1}^{p} h_i < h^p$ centroids.

Possible advantages over standard clustering arise whenever $\prod_{i=1}^{p} h_i > \sum_{i=1}^{p} h_i$. For instance, two sets of two protocentroids can represent four centroids, yielding no advantage.

Choosing the number of sets of protocentroids. In all Khatri-Rao clustering algorithms, the number p of protocentroids plays an important role. In this work, we primarily focus on the case of $p = 2$, which we recommend as the default choice. Increasing the value of p renders the optimization problem more challenging and hinders interpretability. However, larger compression gains may in principle be obtained by considering $p > 2$. For example, given a budget of 12 vectors to represent centroids, allocating them in 2 and 3 sets of equal size can represent 36 and 64 centroids, respectively. As explained in the previous paragraph, sets of equal size maximize the number of centroids that can be represented. In addition, Proposition 8.1, proved in Appendix C, characterizes the value of p maximizing the number of centroids that can be represented.

Proposition 8.1. *Given a fixed budget of vectors b to represent centroids, among possible divisors of b, the number p^{max} of protocentroid sets of equal size that maximizes the number of centroids*

that can be represented is one of the two divisors of b that are closest to $\frac{b}{e}$, where e is the natural logarithm base.

It is also possible to bound the number of sets that are guaranteed to represent k centroids, as formalized in Proposition 8.2.

Proposition 8.2. *Let h_{min} be the minimum number of protocentroids in each set. The number p^* of sets of protocentroids that are guaranteed to represent k centroids satisfies*

$$\log_{h_{min}} k \leq p^* \leq \lceil \frac{k}{h_{min} - 1} \rceil.$$

The proof is given in Appendix C.

Choosing the number of centroids. If the budget b of vectors is not fixed and one wants to identify the number of clusters that are appropriate for the data, Khatri-Rao clustering can be combined with established techniques such as X-Means [44] or G-Means [21]. Here, the number of centroids is successively increased and the current parameterization is evaluated, e.g., by using the Bayesian Information Criterion [48] or by testing if certain distributional conditions are fulfilled. In Khatri-Rao clustering, increasing the number of clusters is equivalent to either increasing the cardinality of one set of protocentroids or the number of sets of protocentroids. Moreover, for deep clustering, a Khatri-Rao variant of such algorithms as DipDECK [34] or DeepDPM [47], which also optimize the number of underlying clusters, can be designed.

Choosing the aggregator function. In this work, we focus on the sum and product aggregator functions. Unfortunately, deciding between the sum and product aggregators prior to running our algorithms is difficult, particularly when moving beyond the naïve two-phase procedure described in Section 5, which relies on an initial set of unconstrained centroids. More specifically, given an initial set of unconstrained centroids, in the additive model, centroid differences $\boldsymbol{\mu}_{i,j} - \boldsymbol{\mu}_{i',j} = (\boldsymbol{\theta}_i + \boldsymbol{\theta}_j) - (\boldsymbol{\theta}_{i'} + \boldsymbol{\theta}_j)$ remain nearly constant across j, and a similar invariance holds for the multiplicative model after taking logarithms. This can provide a simple heuristic for deciding between the two models. Without an initial set of centroids, similar heuristics cannot be considered. In our empirical evaluation, both the sum and product aggregators achieve competitive performance in k-Means clustering. In contrast, in Khatri-Rao deep clustering, the sum aggregator is generally preferable, as it results in an easier optimization.

The choice of aggregator function does not need to be limited to sum or product. Conceptually, the Khatri-Rao clustering paradigm could accommodate arbitrary aggregator functions. However, there exists a tension between expressivity and optimization difficulty. As a consequence, increasing aggregator complexity not only compromises interpretability, but can also hurt performance.

Choosing the number of Hadamard factors and their ranks. In Khatri-Rao deep clustering, by default, we reparameterize the autoencoder weights using a two-factor Hadamard decomposition, i.e., we set $q = 2$ in Equation (6), which ensures stable gradient-based optimization. Using more Hadamard factors can improve compression, but reduces stability. The ranks of both factors are set to be equal (to maximize the rank that can be captured) and large enough so that the compressed autoencoder incurs comparable reconstruction quality to the uncompressed one.

Table 1: Characteristics of the datasets used in the experiments. For each dataset, we report the number of data points, the number of features, the number of ground-truth clusters (# Labels) and the imbalance ratio (IR), defined as the ratio of the smallest to the largest cluster size.

Dataset	# Data points	# Features	# Labels	IR
MNIST	25000	784	10	1.00
Double MNIST	10000	1568	100	1.00
HAR	10299	561	6	0.72
Olivetti Faces	400	4096	40	1.00
CMU Faces	624	960	20	0.88
Symbols	1020	398	6	0.90
stickfigures	900	400	9	1.00
optdigits	5620	64	10	0.97
Classification	5000	10	100	0.91
Chameleon	10000	2	10	0.10
Soybean Large	562	35	15	0.22
Blobs	5000	2	100	1.00
R15	600	2	15	1.00

9 Experiments

In this section, we evaluate the performance of Khatri-Rao clustering against baselines considering both synthetic and real benchmark datasets. The experiments primarily aim to show that Khatri-Rao clustering achieves the goal of reducing the size of a centroid-based data summary without compromising its accuracy. We also showcase the practical benefits of Khatri-Rao clustering through case studies.

9.1 Settings

In the following, we present the experimental setup.

Datasets. We consider two synthetic dataset generators (Blobs and Classification) from the scikit-learn Python library [43], which allow exploring the behavior of different algorithms as a function of the number of data points, features and underlying clusters. The preliminary experiments and the scalability analysis focus on those datasets.

In addition, we consider seven real-world benchmark datasets, available through the ClustPy [35] Python library and two synthetic datasets available through the clustbench [16] Python library. Finally, we consider the Double MNIST dataset, which is created by concatenating pairs of digits from MNIST [33], yielding 100 clusters representing the numbers 0 to 99.

The salient characteristics of the datasets considered in the experimental evaluation are provided in Table 1. A detailed description of each dataset is provided in Appendix A.

Parameter settings. For the purposes of our evaluation, we assume that the underlying ground-truth number of clusters in a dataset is known, and is given as input to all algorithms. Unless stated otherwise, for all experiments regarding Khatri-Rao clustering, we consider two sets of protocentroids (i.e., $p = 2$). To maximize the number of centroids that can be represented, the cardinality of each set of protocentroids is determined by selecting the two factors of the total number of clusters that are closest in value so that $h_1 h_2 = k$ (e.g., $h_1 = 8$ and $h_2 = 5$ for $k = 40$). In all experiments, we run each method 20 times with random initialization and select the solution giving the smallest inertia, i.e., the smallest squared Euclidean distance between samples and their assigned cluster centroids.

Khatri-Rao-k-Means terminates when either $n_{iter} = 200$ iterations are reached, or the centroid movement falls below $\epsilon = 10^{-4}$. We consider Khatri-Rao-k-Means with sum and product aggregator functions. Instead, in the deep clustering experiments

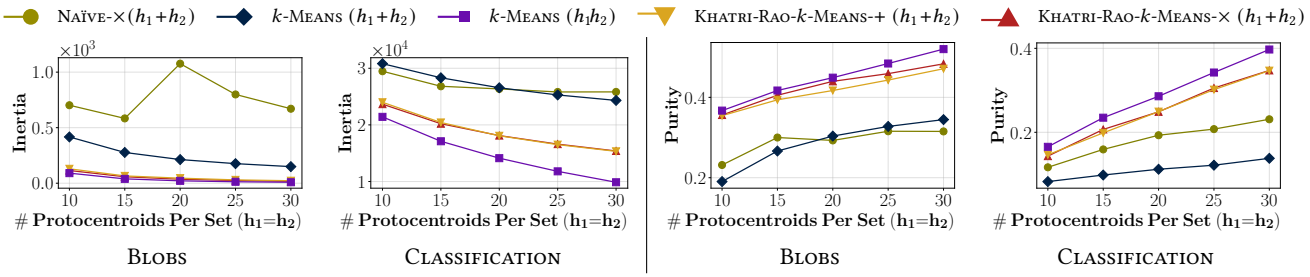

Figure 6: Experiments using the Blobs and Classification datasets with 100 ground truth clusters. Inertia (left) and purity (right) as a function of the cardinality $h_1 = h_2$ of two sets of protocentroids. k-Means (h_1h_2) uses h_1h_2 vectors to represent centroids, while all other algorithms use $h_1 + h_2$ vectors.

we focus on the sum since it generally yields superior results, as explained in Section 8.

In all deep clustering experiments, the parameter w_{rec}, which balances the clustering and reconstruction losses as per Equation (2), is set to 1, the batch size for batch-wise gradient-based optimization is set to 512, and we use the ADAM optimizer [29] using a learning rate of 10^{-3} for pre-training the autoencoder and 10^{-4} for clustering.

Further, a preliminary step in the deep clustering experiments is required to learn two autoencoders, a fully-connected autoencoder for standard deep clustering algorithms and a compressed autoencoder for their Khatri-Rao variants. Autoencoders have an encoder with 5 layers of dimensions m−1024−512−256−10 and the decoder is a mirrored version of the encoder. To ensure reliable performance of the Khatri-Rao deep clustering algorithms, it is crucial that the initial compressed autoencoder achieves a similar reconstruction quality as the uncompressed one. Given a matrix of weights $\mathbf{W}_l \in \mathbb{R}^{d_l \times m_l}$, we initially set the rank of each of two Hadamard-decomposition factors to $\max\{10, \min\{n_l, m_l\}\}$, and we leave the input and output layers uncompressed, which improves performance. If the loss of the initial compressed autoencoder is higher than that of the full autoencoder, we iteratively multiply the rank by $2, 3, \ldots$ until the loss of the compressed autoencoder falls under that of the full autoencoder. As shown in Section 9.2, this strategy guarantees remarkable parameter reductions. The number of epochs used for pre-training the standard autoencoders and for the actual clustering procedure is set to 150. For pre-training the compressed autoencoder, we set the number of epochs to 1000 and 500 additional epochs are added whenever the rank is increased as explained above.

Metrics. In our experiments, the accuracy of a data summary is measured by the quality of clustering results. We consider the following standard metrics to evaluate clustering results that take advantage of ground truth labels: adjusted Rand index (ARI) [23], unsupervised clustering accuracy (ACC) [59] and normalized mutual information (NMI) [32]. These metrics have a maximum value of 1, which reveals a perfect match between predicted and ground-truth labels. In the case of k-Means clustering, we also monitor inertia, which is the objective function optimized for by standard k-Means and Khatri-Rao-k-Means. In addition, in the preliminary experiments, we monitor clustering purity, which measures the proportion of correctly assigned data points after assigning each data point to the majority ground-truth label of its cluster [39].

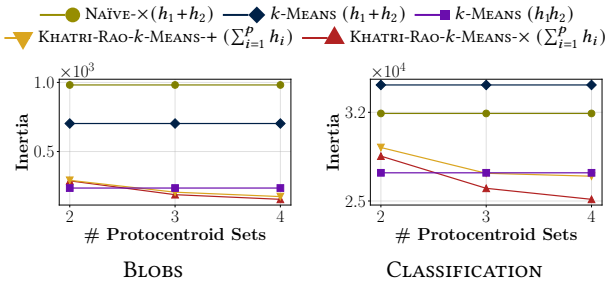

Figure 7: Experiments using the Blobs and Classification datasets with 100 ground truth clusters. Inertia as a function of the number of sets of protocentroids. Here, Khatri-Rao-k-Means always uses 12 total vectors to represent centroids. All baselines assume $h_1 = h_2 = 6$.

To quantify the amount of compression, we monitor the number of parameters that a given algorithm uses to obtain a succinct summary of a dataset.

Clustering algorithms. In the case of k-Means clustering, we compare Khatri-Rao-k-Means against the standard k-Means algorithm (implementation from scikit-learn [43]). Additionally, we consider the two-phase naïve approach described in Section 5 with product aggregator. When not clear from context, we specify the aggregator function and the budget of vectors used to represent centroids (e.g., Khatri-Rao-k-Means-+ ($h_1 + h_2$) denotes Khatri-Rao-k-Means with sum aggregator and $h_1 + h_2$ protocentroids).

In the case of deep clustering, we consider DKM and IDEC (implementations from ClustPy [35]) against their Khatri-Rao variants, Khatri-Rao DKM and Khatri-Rao IDEC. Appendix B provides implementation details for all algorithms.

Computing environment. Experiments use a machine with a 13th Gen Intel Core i9-13900H processor (14 cores, 20 threads), an NVIDIA GeForce RTX 4060 GPU and 32 GB of RAM.

9.2 Quantitative Results

In this section, we present the results of the experiments, first for the k-Means setting and second for the deep clustering setting.

Preliminary results. We begin by presenting preliminary experiments on the Blobs and Classification datasets. Figure 6 shows inertia and purity as a function of the cardinality ($h_1 = h_2$) of both sets of protocentroids. The inertia incurred by Khatri-Rao-k-Means is at most 31% and 81% of that incurred by any of the baselines using the same amount of parameters in the Blobs

Table 2: Experiments on synthetic and real-world datasets comparing Khatri-Rao-k-Means using the product (Khatri-Rao-k-Means-×) and sum (Khatri-Rao-k-Means-+) aggregator functions with two sets of h_1 and h_2 protocentroids against k-Means ($h_1 + h_2$) and k-Means (h_1h_2). For each dataset and algorithm, we report unsupervised clustering accuracy (ACC), adjusted Rand index (ARI), normalized mutual information (NMI) and inertia (normalized by dividing by the inertia of k-Means (h_1h_2)). The last column reports the ratio of the number of parameters used compared to k-Means (h_1h_2).

Dataset	Khatri-Rao-k-Means-+ ($h_1 + h_2$)				Khatri-Rao-k-Means-× ($h_1 + h_2$)				k-Means ($h_1 + h_2$)				k-Means (h_1h_2)				Params
	ARI	ACC	NMI	Inertia	ARI	ACC	NMI	Inertia	ARI	ACC	NMI	Inertia	ARI	ACC	NMI	Inertia	
MNIST	0.298	0.436	0.428	1.05	0.309	0.435	0.441	1.07	0.357	0.515	0.467	1.05	0.366	0.515	0.498	1.00	0.70
Double MNIST	0.109	0.211	0.467	1.16	0.068	0.119	0.396	1.07	0.049	0.081	0.329	1.20	0.075	0.143	0.428	1.00	0.20
HAR	0.320	0.515	0.486	1.06	0.285	0.456	0.451	1.04	0.291	0.442	0.461	1.03	0.420	0.539	0.559	1.00	0.83
Olivetti Faces	0.248	0.398	0.664	1.34	0.239	0.393	0.643	1.41	0.181	0.287	0.565	1.46	0.456	0.600	0.790	1.00	0.33
CMU Faces	0.414	0.503	0.760	1.52	0.463	0.561	0.788	1.52	0.408	0.452	0.767	1.46	0.754	0.772	0.902	1.00	0.45
Symbols	0.459	0.605	0.682	1.52	0.693	0.768	0.815	1.35	0.682	0.758	0.810	1.22	0.666	0.689	0.794	1.00	0.83
stickfigures	1.000	1.000	1.000	1.00	0.885	0.889	0.964	4.79	0.708	0.667	0.882	11.57	1.000	1.000	1.000	1.00	0.67
optdigits	0.477	0.625	0.622	1.12	0.457	0.596	0.589	1.07	0.465	0.612	0.622	1.11	0.491	0.640	0.648	1.00	0.70
Classification	0.041	0.134	0.362	1.12	0.044	0.137	0.368	1.10	0.035	0.083	0.245	1.44	0.052	0.159	0.387	1.00	0.20
Chameleon	0.318	0.435	0.551	1.07	0.307	0.419	0.545	1.06	0.373	0.452	0.583	1.52	0.296	0.439	0.538	1.00	0.70
Soybean Large	0.331	0.484	0.647	1.49	0.368	0.534	0.654	1.29	0.329	0.482	0.637	1.43	0.406	0.589	0.674	1.00	0.53
Blobs	0.236	0.326	0.655	1.45	0.242	0.341	0.656	1.34	0.207	0.190	0.655	4.61	0.238	0.350	0.658	1.00	0.20
R15	0.787	0.815	0.910	3.44	0.919	0.928	0.970	1.68	0.264	0.533	0.743	11.77	0.993	0.997	0.994	1.00	0.53

and Classification datasets, respectively. Similarly, the purity achieved by any such baselines is at most 76% and 81% of that achieved by Khatri-Rao-k-Means. Standard k-Means can attain lower inertia and higher purity than Khatri-Rao-k-Means, but using h_1h_2 vectors to represent centroids rather than $h_1 + h_2$.

The performance of Khatri-Rao-k-Means could be improved further by increasing the number p of sets of protocentroids. While we leave a thorough investigation to future work, Figure 7 compares the inertia incurred by Khatri-Rao-k-Means as a total of 12 vectors that are split between 2, 3 and 4 sets of protocentroids against that incurred by baselines with $h_1 = h_2 = 6$. The results indicate that Khatri-Rao-k-Means with 12 vectors to represent centroids can incur lower inertia than standard k-Means with $h_1h_2 = 36$ vectors as p increases. As discussed in Section 8, increasing p can improve performance but makes optimization more challenging. As a consequence, inertia decreases monotonically, but with diminishing reductions. In the remaining experiments, we focus on two sets of protocentroids. Furthermore, the naïve approach is not competitive with Khatri-Rao-k-Means. Thus, in the remaining experiments, we focus on comparing Khatri-Rao-k-Means against k-Means.

Results for k-Means. Table 2 summarizes the results of the experiments comparing Khatri-Rao-k-Means with two sets of h_1 and h_2 protocentroids for sum and product aggregators against standard k-Means with $h_1 + h_2$ and h_1h_2 centroids. The table reports the adopted measures of clustering quality as well as the ratio of the inertia and the amount of parameters used for summarization relative to k-Means with h_1h_2 centroids. The results indicate that Khatri-Rao-k-Means often but not always outperforms standard k-Means with the same number of parameters.

The medians (means) of the inertia ratios reported in the table across datasets are 1.16 (1.41), 1.29 (1.52) and 1.44 (3.14), for Khatri-Rao-k-Means-+, Khatri-Rao-k-Means-× and k-Means respectively, with all algorithms using $h_1 + h_2$ vectors to represent centroids. Nonetheless, standard k-Means with h_1h_2 centroids generally provides superior performance to Khatri-Rao-k-Means, suggesting that Khatri-Rao-k-Means struggles to achieve its ideal performance also due to its lack of flexibility discussed in Section 6.

Results for deep clustering. The experimental results comparing the deep clustering approaches DKM against Khatri-Rao DKM and IDEC against Khatri-Rao IDEC are shown in Table 3. DKM and IDEC directly return h_1h_2 clusters, while Khatri-Rao DKM and Khatri-Rao IDEC are working with two sets of h_1 and h_2 protocentroids leading to the same number of clusters. The results demonstrate that the Khatri-Rao deep clustering framework can reduce the number of parameters by at least more than 10% and as much as 85% with a minor impact on performance. In some datasets, the performance of Khatri-Rao deep clustering is even superior to that of standard deep clustering. It is conceivable that Khatri-Rao deep clustering provides an implicit form of regularization.

In summary, while Khatri-Rao-k-Means with $h_1 + h_2$ protocentroids may not attain the same clustering quality as standard k-Means with h_1h_2 centroids, this is not the case for deep clustering. The performance of deep Khatri-Rao clustering algorithms is close to their optimistic bound.

In our experiments, only the stickfigures and Double MNIST datasets are known to exhibit a Khatri-Rao structure. However, Khatri-Rao clustering can deliver advantages over traditional clustering, irrespective of whether its underlying model holds.

9.3 Scalability

Our quantitative experiments suggest that Khatri-Rao clustering can yield more succinct clustering-based summaries than standard clustering methods while maintaining a comparable accuracy. However, it is also important to ensure that the provided advantages are not undermined by significantly worse scalability. According to the discussion in Section 6, Khatri-Rao-k-Means has the same asymptotic time complexity as k-Means. In addition, Khatri-Rao-k-Means can reduce the memory usage of k-Means when the number of clusters and hence centroids to be represented grows. To complement the analysis of time and space complexity, we conduct an empirical scalability analysis.

The results of the scalability analysis are summarized in Figure 8. This figure shows runtime (in seconds) and peak memory usage (in Mebibytes) for a single execution of an algorithm when increasing the number of data points, features and centroids using the Blobs dataset. In particular, we vary the number of data points in $\{4000, 8000, 12000, 16000\}$ (with 100 clusters and 100 features), features in $\{10000, 15000, 20000, 25000\}$ (with 1000 data points and 100 clusters), and clusters in $\{2500, 5000, 7500, 10000\}$ (with 20000 data points and 100 features).

The results suggest that Khatri-Rao-k-Means clustering introduces a runtime overhead compared to standard k-Means.

Table 3: Experiments on synthetic and real-world datasets comparing the deep-clustering algorithms DKM and IDEC to their Khatri-Rao counterparts using the sum aggregator function. For each dataset and algorithm, we report unsupervised clustering accuracy (ACC), adjusted Rand index (ARI) and normalized mutual information (NMI). The last column reports the ratio of the number of parameters used by Khatri-Rao DKM and Khatri-Rao IDEC compared to DKM and IDEC.

Dataset	IDEC			Khatri-Rao IDEC			DKM			Khatri-Rao DKM			Params Ratio
	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	
MNIST	0.616	0.771	0.682	0.630	0.746	0.712	0.574	0.718	0.693	0.645	0.788	0.719	0.74
Double MNIST	0.174	0.285	0.532	0.180	0.292	0.536	0.178	0.295	0.537	0.201	0.330	0.553	0.79
HAR	0.581	0.692	0.660	0.621	0.703	0.701	0.539	0.649	0.634	0.619	0.676	0.714	0.62
Olivetti Faces	0.399	0.517	0.746	0.371	0.510	0.734	0.437	0.570	0.769	0.387	0.542	0.736	0.88
CMU Faces	0.564	0.646	0.782	0.648	0.707	0.823	0.690	0.761	0.852	0.525	0.631	0.743	0.66
Symbols	0.655	0.684	0.779	0.689	0.722	0.798	0.662	0.693	0.786	0.711	0.746	0.813	0.47
stickfigures	1	1	1	1	1	1	1	1	1	1	1	1	0.47
optdigits	0.688	0.809	0.751	0.645	0.775	0.721	0.634	0.733	0.742	0.681	0.774	0.766	0.22
Classification	0.032	0.124	0.336	0.007	0.070	0.245	0.044	0.144	0.369	0.040	0.133	0.356	0.16
Chameleon	0.321	0.466	0.564	0.312	0.415	0.550	0.329	0.474	0.589	0.319	0.476	0.540	0.15
Soybean Large	0.391	0.562	0.657	0.451	0.617	0.696	0.373	0.546	0.661	0.445	0.635	0.693	0.19
Blobs	0.182	0.214	0.597	0.188	0.239	0.609	0.226	0.314	0.653	0.215	0.307	0.650	0.15
R15	0.986	0.993	0.988	0.989	0.995	0.991	0.830	0.850	0.926	0.993	0.997	0.994	0.15

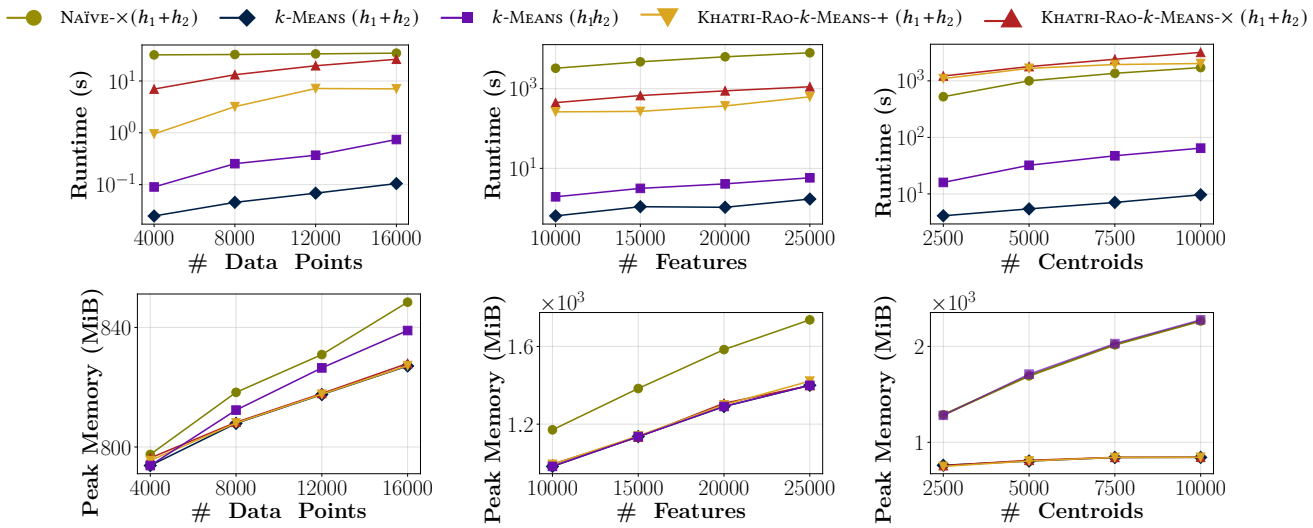

Figure 8: Experiments using the Blobs dataset. Runtime in seconds (top) and peak memory usage in Mebibytes (bottom) by number of data points n, number of features m and number of centroids k. Here, k-Means (h_1h_2) uses h_1h_2 vectors to represent centroids, while all other algorithms use $h_1 + h_2$ vectors. The y-axis is on a logarithmic scale.

However, the overhead remains nearly constant as the number of data points, features and centroids increases, in line with the time-complexity discussion in Section 6. The two-phase naïve approach to Khatri-Rao k-Means is usually the slowest algorithm, except when the number of centroids becomes large, in which case Khatri-Rao-k-Means can become slower.

Considering memory requirements, Khatri-Rao-k-Means often exhibits a similar behavior as standard k-Means using $h_1 + h_2$ vectors to represent centroids. Standard k-Means with h_1h_2 centroids, on the other hand, can be more memory demanding, particularly when the number of centroids grows, using up to 2.7 times more memory than Khatri-Rao-k-Means. Furthermore, the gap grows with the number of centroids, in agreement with the space-complexity discussion in Section 6.

9.4 Case Studies

We conclude our experimental evaluation by presenting two simple case studies demonstrating the benefits of our Khatri-Rao clustering paradigm when used for color quantization and in a federated-learning environment. The focus of the case studies is not to claim state-of-the-art performance on the two tasks, but to highlight the practical advantages that our paradigm can offer in real-world applications. Thus, for the purposes of the case studies, we restrict our attention to Khatri-Rao-k-Means using the product aggregator.

Obtaining more succinct codebooks for color quantization. Color quantization in computer graphics enables efficient compression of certain types of images by reducing the number of colors. It allows displaying images with many colors on hardware-constrained devices that can only display a limited number of colors, usually due to memory limitations [7]. A common approach casts color quantization as clustering in a three-dimensional RGB space, where pixels are points. The centroids obtained by k-Means form a codebook of representative colors, and each pixel is mapped to its closest centroid for color quantization.

Khatri-Rao-k-Means has the potential to extract more succinct and yet equally accurate codebooks compared to standard k-Means. To investigate this potential, we consider the image

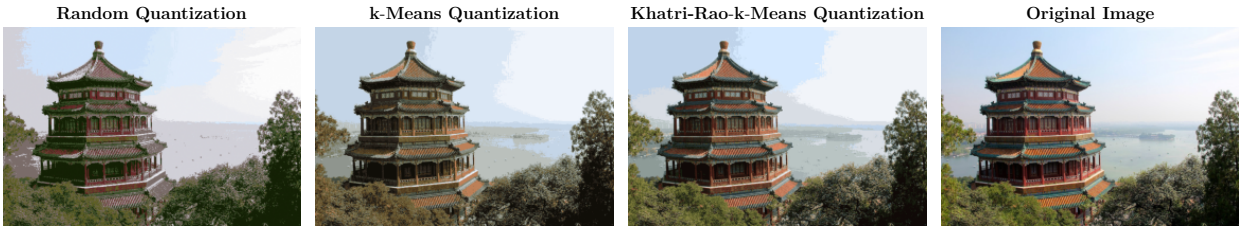

Figure 9: Comparison of Khatri-Rao-k-Means with product aggregator using two sets of $h_1 = h_2 = 6$ protocentroids with different approaches to color quantization using the same number of parameters.

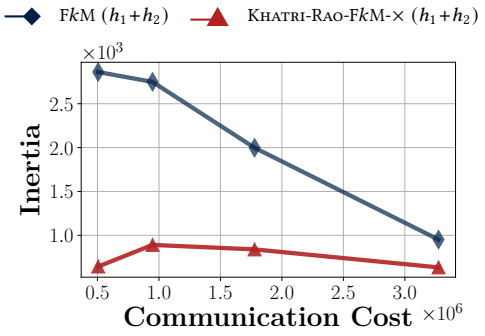

Figure 10: FEMNIST dataset. Inertia against communication costs from the server to the clients (in Bytes) in a simulated federated learning environment for FkM and Khatri-Rao-FkM using the product aggregator.

from the scikit-learn "Color Quantization using K-Means" example.[3] We run k-Means and Khatri-Rao-k-Means with the same number of centroid parameters on a subset of the image (1000 pixels), and compare the obtained color quantizations. In particular, k-Means uses 12 centroids, and Khatri-Rao-k-Means two sets of 6 protocentroids. For reference, we also include a color-quantization strategy which picks 12 pixels uniformly at random to form the codebook.

The results are given in Figure 9, and verify that Khatri-Rao-k-Means can outperform standard k-Means in the task of color quantization as it better preserves the colors of the original image. This is particularly evident in the improved representation of red tones. This visual assessment is confirmed numerically as the values of the inertia incurred by random quantization, k-Means and Khatri-Rao-k-Means are 4686, 2009 and 1144, respectively.

Reducing communication costs for clustering in federated learning environments. In recent years, federated learning, which allows for collaborative training of machine learning models across distributed entities, has emerged as a central area of research [55]. The Khatri-Rao clustering paradigm can benefit federated learning by reducing communication costs between entities, an important goal in federated learning research [42].

Recently, Garst and Reinders [17] have introduced an effective implementation of k-Means in a federated setting (henceforth called FkM). As FkM requires communicating centroids between several clients and a server, Khatri-Rao-FkM can potentially reduce the costs of communication by considering a lightweight set of protocentroids instead of regular centroids. To extend FkM to Khatri-Rao clustering, it suffices to replace each invocation of k-Means in FkM with Khatri-Rao-k-Means. The resulting algorithm is referred to as Khatri-Rao-FkM. To demonstrate the advantages given by Khatri-Rao-FkM with respect to communication costs, we simulate a federated learning environment with 10 clients and one server. In this experiment, we consider the benchmark FEMNIST dataset [8], and we measure the inertia achieved by FkM and Khatri-Rao-FkM as a function of the communication costs from the server to the clients. The results of this experiment are displayed in Figure 10 and indicate that Khatri-Rao-FkM consistently reduces inertia relative to FkM at parity communication cost. For the smallest communication cost, the inertia incurred by FkM is about five times larger than that incurred by Khatri-Rao-FkM. This suggests that Khatri-Rao-FkM can communicate clustering results of a given quality much more efficiently than FkM.

[3] https://scikit-learn.org/0.19/auto_examples/cluster/plot_color_quantization.html

10 Conclusion

We have introduced the Khatri-Rao clustering paradigm which extends centroid-based clustering to find more succinct data summaries without significantly compromising their accuracy.

First, we have applied the paradigm in the context of the popular k-Means clustering by introducing the Khatri-Rao-k-Means algorithm. Extensive experiments and case studies show that Khatri-Rao-k-Means can find more succinct but equally accurate data summaries than the long-standing k-Means algorithm.

We have also introduced the Khatri-Rao deep clustering framework which hinges on Khatri-Rao-k-Means but fruitfully leverages representation learning to address its lack of flexibility. Khatri-Rao deep clustering often performs on par with standard deep clustering, while using considerably fewer parameters.

There are many possibilities for future work. For example, it would be interesting to study potential extensions of the Khatri-Rao clustering paradigm, e.g., by considering a wider range of baseline clustering approaches. However, the most pressing challenge is to understand the underlying structure of the clusters before applying Khatri–Rao clustering algorithms. In particular, how can data exhibiting additive or multiplicative Khatri–Rao structures be effectively characterized?

Acknowledgments

Aristides Gionis is supported by the ERC Advanced Grant REBOUND (834862), the Swedish Research Council project ExCLUS (2024-05603), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Collin Leiber is supported by the Research Council of Finland (decision 368654) and Heikki Mannila by the Technology Industries of Finland Centennial Foundation.

Artifacts

Our Python implementations of Khatri-Rao k-Means, Khatri-Rao DKM and Khatri-Rao IDEC are available in an online repository,[4] along with code to load datasets and reproduce the experiments. The repository also contains the Appendix.[5]

References

[1] Mohiuddin Ahmed. 2019. Data summarization: a survey. *Knowledge and Information Systems* 58, 2 (2019), 249–273.

[2] David Arthur and Sergei Vassilvitskii. 2007. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 1027–1035.

[3] Inigo Barrio-Hernandez, Jingi Yeo, Jürgen Jänes, Milot Mirdita, Cameron LM Gilchrist, Tanita Wein, Mihaly Varadi, Sameer Velankar, Pedro Beltrao, and Martin Steinegger. 2023. Clustering predicted structures at the scale of the known protein universe. *Nature* 622, 7983 (2023), 637–645.

[4] MohammadHossein Bateni, Hossein Esfandiari, Manuela Fischer, and Vahab Mirrokni. 2021. Extreme k-center clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 3941–3949.

[5] Brenda Betancourt, Giacomo Zanella, Jeffrey W Miller, Hanna Wallach, Abbas Zaidi, and Rebecca C Steorts. 2016. Flexible models for microclustering with application to entity resolution. *Advances in neural information processing systems* 29 (2016).

[6] Horst Bischof, Ales Leonardis, and Alexander Selb. 1999. MDL Principle for Robust Vector Quantisation. *Pattern Anal. Appl.* 2, 1 (1999), 59–72. doi:10.1007/S100440050015

[7] Luc Brun and Alain Trémeau. 2017. Color quantization. In *Digital color imaging handbook*. CRC press, 589–637.

[8] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[9] Martino Ciaperoni, Aristides Gionis, and Heikki Mannila. 2024. The Hadamard decomposition problem. *Data Mining and Knowledge Discovery* (2024), 1–42.

[10] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 70, 6 (2004), 066111.

[11] Sanjoy Dasgupta. 2008. The hardness of k-means clustering. (2008).

[12] Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 606–610.

[13] Andrew Draganov, David Saulpic, and Chris Schwiegelshohn. 2024. Settling time vs. accuracy tradeoffs for clustering big data. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–25.

[14] Jirí Dvorskỳ, Jan Martinovic, Jan Platoš, and Václav Snášel. 2010. Document Compression Improvements Based on Data Clustering. *Web Intelligence and Intelligent Agents* (2010), 133.

[15] Maziar Moradi Fard, Thibaut Thonet, and Éric Gaussier. 2020. Deep k-Means: Jointly clustering with k-Means and learning representations. *Pattern Recognit. Lett.* 138 (2020), 185–192. doi:10.1016/J.PATREC.2020.07.028

[16] Marek Gagolewski. 2022. A framework for benchmarking clustering algorithms. *SoftwareX* 20 (2022), 101270.

[17] Swier Garst and Marcel Reinders. 2024. Federated k-means clustering. In *International Conference on Pattern Recognition*. Springer, 107–122.

[18] Attri Ghosal, Arunima Nandy, Amit Kumar Das, Saptarsi Goswami, and Mrityunjoy Panday. 2020. A short review on different clustering techniques and their applications. *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018* (2020), 69–83.

[19] Stephan Günnemann, Ines Färber, Matthias Sebastian Rüdiger, and Thomas Seidl. 2014. SMVC: semi-supervised multi-view clustering in subspace projections. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. ACM, 253–262. doi:10.1145/2623330.2623734

[20] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved Deep Embedded Clustering with Local Structure Preservation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI Organization, 1753–1759. doi:10.24963/ijcai.2017/243

[21] Greg Hamerly and Charles Elkan. 2003. Learning the k in k-means. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*. MIT Press, 281–288. https://proceedings.neurips.cc/paper/2003/hash/234833147b97bb6aed53a8f4f1c7a7d8-Abstract.html

[22] Bo Han and Bolang Li. 2016. Lossless Compression of Data Tables in Mobile Devices by Using Co-clustering. *International Journal of Computers Communications & Control* 11, 6 (2016), 776–788.

[23] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2 (1985), 193–218.

[24] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. 2021. FedPara: Low-rank Hadamard Product for Communication-Efficient Federated Learning. In *International Conference on Learning Representations*.

[25] Abiodun M Ikotun, Absalom E Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. 2023. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences* 622 (2023), 178–210.

[26] Anil K. Jain. 2008. Data Clustering: 50 Years Beyond K-means. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 5211)*. Springer, 3–4. doi:10.1007/978-3-540-87479-9_3

[27] Uthayakumar Jayasankar, Vengattaraman Thirumal, and Dhavachelvan Ponnurangam. 2021. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University-Computer and Information Sciences* 33, 2 (2021), 119–140.

[28] CG Khatri and C Radhakrishna Rao. 1968. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhyā: The Indian Journal of Statistics, Series A* (1968), 167–180.

[29] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980

[30] Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. 2017. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 255–264.

[31] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[32] Tarald O Kvalseth. 1987. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics* 17, 3 (1987), 517–519.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[34] Collin Leiber, Lena G. M. Bauer, Benjamin Schelling, Christian Böhm, and Claudia Plant. 2021. Dip-based Deep Embedded Clustering with k-Estimation. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 903–913. doi:10.1145/3447548.3467316

[35] Collin Leiber, Lukas Miklautz, Claudia Plant, and Christian Böhm. 2023. Benchmarking deep clustering algorithms with clustpy. In *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 625–632.

[36] Collin Leiber, Lukas Miklautz, Claudia Plant, and Christian Böhm. 2025. An Introductory Survey to Autoencoder-based Deep Clustering - Sandboxes for Combining Clustering with Deep Learning. *CoRR* abs/2504.02087 (2025). doi:10.48550/ARXIV.2504.02087 arXiv:2504.02087

[37] Tao Li and Chris HQ Ding. 2013. Nonnegative Matrix Factorizations for Clustering: A Survey. *Data clustering: algorithms and applications* (2013), 149–176.

[38] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[39] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press. doi:10.1017/CBO9780511809071

[40] Lukas Miklautz, Timo Klein, Kevin Sidak, Collin Leiber, Thomas Lang, Andrii Shkabrii, Sebastian Tschiatschek, and Claudia Plant. 2025. Breaking the Reclustering Barrier in Centroid-based Deep Clustering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=r01fcKhzT5

[41] Mohammed Omari, Mohammed Kaddi, Khouloud Salameh, and Ali Alnoman. 2025. Advancing Image Compression Through Clustering Techniques: A Comprehensive Analysis. *Technologies* 13, 3 (2025), 123.

[42] Giovanni Paragliola and Antonio Coronato. 2022. Definition of a novel federated learning approach to reduce communication costs. *Expert Systems with Applications* 189 (2022), 116109.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[44] Dan Pelleg and Andrew W. Moore. 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Morgan Kaufmann, 727–734.

[45] Anil Raj and Chris H Wiggins. 2009. An information-theoretic derivation of min-cut-based clustering. *IEEE transactions on pattern analysis and machine intelligence* 32, 6 (2009), 988–995.

[46] Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S Yu, and Lifang He. 2024. Deep clustering: A comprehensive survey. *IEEE transactions on neural networks and learning systems* 36, 4 (2024), 5858–5878.

[47] Meitar Ronen, Shahaf E. Finder, and Oren Freifeld. 2022. DeepDPM: Deep Clustering With an Unknown Number of Clusters. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 9851–9860. doi:10.1109/CVPR52688.2022.00963

[48] Gideon Schwarz. 1978. Estimating the dimension of a model. *The annals of statistics* (1978), 461–464.

[4] https://github.com/maciap/KhatriRaoClustering

[5] https://github.com/maciap/KhatriRaoClustering/blob/main/Appendix.pdf

[49] D. Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti (Eds.). ACM, 1177–1178. doi:10.1145/1772690.1772862

[50] Kashi Sethia, Madhur Saxena, Mukul Goyal, and RK Yadav. 2022. Framework for topic modeling using BERT, LDA and K-means. In *2022 2nd International conference on advance computing and innovative technologies in engineering (ICACITE)*. IEEE, 2204–2208.

[51] Noam Slonim, Gurinder Singh Atwal, Gašper Tkačik, and William Bialek. 2005. Information-based clustering. *Proceedings of the National Academy of Sciences* 102, 51 (2005), 18297–18302.

[52] K Somasundaram and M Mary Shanthi Rani. 2011. Novel K-means algorithm for compressing images. *International Journal of Computer Applications* 18, 8 (2011), 9–13.

[53] Sanghyun Son, Seungjun Nah, and Kyoung Mu Lee. 2018. Clustering convolutional kernels to compress deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*. 216–232.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[55] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. 2023. A survey on federated learning: challenges and applications. *International journal of machine learning and cybernetics* 14, 2 (2023), 513–535.

[56] Michael Whelan, Nhien An Le Khac, and M-Tahar Kechadi. 2010. Data reduction in very large spatio-temporal datasets. In *2010 19th IEEE international workshops on enabling technologies: infrastructures for collaborative enterprises*. IEEE, 104–109.

[57] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings, Vol. 48)*. JMLR.org, 478–487. http://proceedings.mlr.press/v48/xieb16.html

[58] Dongkuan Xu and Yingjie Tian. 2015. A comprehensive survey of clustering algorithms. *Annals of data science* 2 (2015), 165–193.

[59] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. 2010. Image Clustering Using Local Discriminant Models and Global Integration. *IEEE Trans. Image Process.* 19, 10 (2010), 2761–2773. doi:10.1109/TIP.2010.2049235

[60] Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. 2025. A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions. *ACM Comput. Surv.* 57, 3 (2025), 69:1–69:38. doi:10.1145/3689036