Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.057693

ARTICLE

*k*ProtoClust: Towards Adaptive *k*-Prototype Clustering without Known *k*

Yuan Ping[1,2,*], Huina Li[1], Chun Guo[3] and Bin Hao[4]

[1]School of Information Engineering, Xuchang University, Xuchang, 461000, China
[2]Henan Province Engineering Technology Research Center of Big Data Security and Applications, Xuchang, 461000, China
[3]College of Computer Science and Technology, Guizhou University, Guiyang, 550025, China
[4]Here Data Technology, Shenzhen, 518000, China

*Corresponding Author: Yuan Ping. Email: pingyuan@xcu.edu.cn

Received: 25 August 2024; Accepted: 16 December 2024; Published: 06 March 2025

ABSTRACT: Towards optimal *k*-prototype discovery, *k*-means-like algorithms give us inspirations of central samples collection, yet the unstable seed samples selection, the hypothesis of a circle-like pattern, and the unknown *K* are still challenges, particularly for non-predetermined data patterns. We propose an adaptive *k*-prototype clustering method (*k*ProtoClust) which launches cluster exploration with a sketchy division of *K* clusters and finds evidence for splitting and merging. On behalf of a group of data samples, support vectors and outliers from the perspective of support vector data description are not the appropriate candidates for prototypes, while inner samples become the first candidates for instability reduction of seeds. Different from the representation of samples in traditional, we extend sample selection by encouraging fictitious samples to emphasize the representativeness of patterns. To get out of the circle-like pattern limitation, we introduce a convex decomposition-based strategy of one-cluster-multiple-prototypes in which convex hulls of varying sizes are prototypes, and accurate connection analysis makes the support of arbitrary cluster shapes possible. Inspired by geometry, the three presented strategies make *k*ProtoClust bypassing the *K* dependence well with the global and local position relationship analysis for data samples. Experimental results on twelve datasets of irregular cluster shape or high dimension suggest that *k*ProtoClust handles arbitrary cluster shapes with prominent accuracy even without the prior knowledge *K*.

KEYWORDS: Prototype finding; *k*-means++; convex hull; support vector data description; geometrical information

1 Introduction

Clustering is to discover natural data groups with maximal intra-cluster similarity yet minimal inter-cluster similarity. As one of the most well known algorithms, *k*-means adopts an iterative strategy with conditional termination, in which labeling each data sample to the nearest cluster centroid and recalculating the centroids are alternately conducted. Due to insufficient prior knowledge, discovering data groups is an exploration procedure toward the target data. First, a predefined *K* is frequently impractical before we judge the exact distribution pattern, e.g., customer segmentation, and anomaly detection [1,2]. Second, *k*-means ultimately does input space partition insensitive to the data distribution boundary. Therefore, a data set can be partitioned differently, and the random selection of initial cluster centroids can aggravate ineradicable instability. Third, under the *K* constraint, partitioning input space in terms of circle-like patterns requires further exploration to deal with arbitrary cluster shapes.

Considering the challenges above, many insightful works can be found in the literature. Without the prior knowledge of *K*, two representative methods are distance-based and vote-based analysis after running

k-means for a range of *K* values (e.g., 5–20). For the former, elbow curve and silhouette coefficient analysis [3] are two typical methods. For each *K*, elbow curve analysis calculates average distances to the centroid across all data samples and finds the point where the average distance from the centroid falls suddenly. Unlike elbow curve analysis, silhouette coefficient analysis measures how similar a data sample is within a cluster (cohesion) compared to other clusters (separation). An average score on each *K* will be obtained before making a judgment. Besides, authors in [4] introduce enhanced gap statistic (EGS) to measure the standardized difference between the log-transformed within-cluster sums of squares from a reference dataset of exponential distribution. A better *K* should reduce the variation. For the latter, Fred et al. [5] suggest evidence accumulation (EAC) to combine multiple results under a vote strategy. It works well in identifying arbitrarily shaped cluster, particularly on low-dimensional data with clear boundaries. Since the initialization phase strongly influences the obtained accuracy and computational load, a common sense of coping strategy is to restart *k*-means several times under one of seeding strategies and keep the final result with the lowest error. Besides the random seeding strategy employed by the classic *k*-means, the other popular strategies include Forgy's approach [6], probabilistic-based strategy [7,8], and the most recent split-merge approach [9]. For *k*-means and its variants, the circle-like pattern hypothesis is preserved. Therefore, split-merge [5,9], multiprototype [10], and multiple kernels-based strategy [11] are mainly adopted for imbalanced clusters. In addition, the outlier removal matching *k*-means is getting more attention for accuracy. Despite the above representative methods making improvements on accuracy, efficiency or adaptability for *k*-means, the requirement of *K* approximating the ground truth is almost kept. Furthermore, even if *K* is given, the intrinsic input space division strategy and the circle-like pattern hypothesis also can not match irregular cluster shapes [9], since they should be considered along with the exploration of unknown *K* clusters.

Despite challenges, *k*-means inspires us for its simplicity, intuition, and efficiency. In this paper, an adaptive *k*-prototype clustering namely *k*ProtoClust is designed to cast off the *K* dependence, optimize seeds extraction, and reduce the influence from circle-like pattern description. The discovered clusters can change in number and shape and become apparent along with the global and local position relationship analysis. The main contributions of this work lie in:

(1) Inspired by the concept of support vector data description (SVDD), convex hulls of variable sizes described by boundary patterns [12] are employed as prototypes. Discarding the circle-like pattern assumption, *k*ProtoClust is never an input space division method but a boundary discovery-based clustering method with a convex decomposition strategy that prefers one-cluster-multiple-prototypes. Thus, arbitrary-shaped clusters can be well discovered and described.

(2) *K* is no longer an objective for input space division but a starting point for cluster exploration. Based on a union analysis of the global and local position relationship for data samples, we find solid evidence for split and mergence. Begin with an initial *K* for a sketchy division of input space, *k*ProtoClust discovers optimal clusters in accord with data distribution. The prior knowledge *K* is bypassed due to giving no constraint on the final cluster number K_f.

(3) To reduce redundant computations, we propose a careful seeding strategy with position analysis (S^2PA) in which a lightweight data grouping algorithm and logical inner analysis are carefully introduced. It gets two critical yet intuitive features: *K* seeds selection must evade outliers and boundaries and should not be limited to existing samples in the target data. As an assistant of *k*ProtoClust in an early collection of cluster centers, *k*-means++ with S^2PA contributes to accurate prototype analysis in stability and efficiency.

(4) By integrating the aforementioned designs, *k*ProtoClust is proposed to achieve a better understanding of clusters from the basic component of convex hull. For efficiency, complete cluster exploration is conducted in input space. Furthermore, to reduce the impact of sparse data space on separability, a near maximin and random sampling (NMMRS) [13] is recommended for large and high dimensional dataset.

More exploration spirits with the descriptive ability can be found in *k*ProtoClust that motivates it to discover the accurate data distribution, not to divide the input data space under an inflexible rule. Compared with *k*-means-like algorithms [14], the cost of a slight efficiency drop is acceptable in practical data analysis.

The remainder of this paper proceeds as follows. Section 2 gives preliminaries of *k*-means++, SVDD, shrinkable edge pattern selection, and NMMRS. Section 3 describes the proposed *k*ProtoClust with its critical components, e.g., S^2PA, evidence of split and mergence, and convex decomposition based prototypes extraction. Section 4 evaluates the performance of *k*ProtoClust. In Section 5, we discuss related work. In Section 6, we conclude this work.

2 Preliminaries

2.1 k-means++

Consider a data set $\mathscr{X}$ with N samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, $C = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K\}$ is the set of expected K cluster centers where $\mathbf{x}_i, \mathbf{c}_v \in \mathbb{R}^d$ with $i = 1, \cdots, N; v = 1, \cdots, K$ and the integer d is the dimension. Let $Z = [z_{iv}]_{N\times K}$, where $z_{iv} \in \{0, 1\}$ indicates whether $\mathbf{x}_i$ belongs to the v-th cluster. Then, we formulate the objective function of *k*-means by

$$\min_{Z,C} \quad \sum_{i=1}^{N}\sum_{v=1}^{K} z_{iv}||\mathbf{x}_i - \mathbf{c}_v||^2. \tag{1}$$

Generally, *k*-means++ selects the first center $\mathbf{c}_1$ at random from $\mathscr{X}$, and repeatedly collects the next center $\mathbf{c}_i = \mathbf{x}' \in \mathscr{X}$ with probability $\frac{D(\mathbf{x}')^2}{\sum_{\mathbf{x}\in\mathscr{X}} D(\mathbf{x})^2}$ until a total of K centers are initialized. Here, $D(\mathbf{x})$ is the shortest distance from $\mathbf{x}$ to the closest center we have already chosen. Then, the solver of *k*-means++ iteratively updates the cluster centers and memberships formulated by the following equations:

$$\begin{aligned} \mathbf{c}_v &= \frac{\sum_{i=1}^{N} z_{iv}\mathbf{x}_{ij}}{\sum_{i=1}^{N} z_{iv}} \\ z_{iv} &= \begin{cases} 1 & \text{if } ||\mathbf{x}_i - \mathbf{c}_v||^2 = \min_{1\le v\le K} ||\mathbf{x}_i - \mathbf{c}_v||^2 \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

Here, $||\mathbf{x}_i - \mathbf{c}_v||$ is the Euclidean distance between $\mathbf{x}_i$ and the cluster center $\mathbf{c}_v$. Apparently, to conduct the iterative procedure, the initial K centers are pivotal and the root of instability. Meanwhile, K is also the prior knowledge because the iteration will end with K clusters.

2.2 Support Vector Data Description

Giving a nonlinear function $\Phi(\cdot)$, SVDD maps data samples into the feature space and obtains the minimum sphere which contains most of the data samples. Let $\boldsymbol{\alpha}$ and R denote its center and radius, respectively, the objective function is

$$\begin{aligned} \min_{R,\boldsymbol{\alpha},\xi_i} \quad & R^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & ||\Phi(\mathbf{x}_i) - \boldsymbol{\alpha}||^2 \le R^2 + \xi_i, \end{aligned} \tag{3}$$

where ξ_i is a slack variable, and C makes the trade-off between simplicity and error [15]. Its dual problem can be formulated by

$$\begin{aligned} \min_{\beta_j} \quad & \sum_{i,j} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & \sum_j \beta_j = 1, \quad 0 \le \beta_j \le C, \quad j = 1, \ldots, N \end{aligned} \tag{4}$$

with $\boldsymbol{\alpha} = \sum_j \beta_j \Phi(\mathbf{x}_j)$ if the Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-q\|\mathbf{x}_i - \mathbf{x}_j\|^2}$ with kernel width q is adopted. $\boldsymbol{\alpha}$ is a linear combination of data samples in feature space with weight factors β_j. Similar to k-means, only a part of data samples with $\beta_j > 0$ contribute to the center. Those with $0 < \beta_i < C$ located on the sphere's boundary are support vectors (SVs), which are essential for describing the sphere, clusters' shapes, and connectivity.

2.3 Shrinkable Edge Pattern Selection

As depicted by Fig. 1, in geometrical, a cluster's boundary consists of an edge and a border [16]. An edge belongs to one cluster, while a border appears in an overlapping region and is shared by two adjacent clusters which are generally considered as two components with the same cluster label. However, in unsupervised learning, it is difficult to find the border. Therefore, similar to SVDD, the edge contains the most informative samples that accurately describe the distribution structure. From this perspective, it can be considered a superset of SVs. In line with [12], the critical steps of shrinkable edge pattern selection (SEPS) for a given $\mathbf{x}_i$ with its k_e nearest neighbors $\mathbf{x}_j (j = 1, 2, \ldots, k_e)$ are described as follows:

- Setting two thresholds γ_l and γ_u $(0 < \gamma_l < \gamma_u \le 1)$ to respectively control the curvature and the shrinkage degree of the aforementioned surface.
- Generating the normal vector $\boldsymbol{n}_i = \sum_{j=1}^{k_e} \mathbf{u}_{ij}$, where $\mathbf{u}_{ij} = \mathbf{x}_j - \mathbf{x}_i$.
- Calculating $l_i = \frac{1}{k_e} \sum_{j=1}^{k_e} g(\boldsymbol{n}_i^T \cdot \mathbf{u}_{ij})$, where $(\cdot)$ means inner product and the function $g(x)$ returns 1 if $x \ge 0$, otherwise it returns 0.
- Cluster boundary identification. If $l_i \in [\gamma_l, \gamma_u]$, then $\mathbf{x}_i$ is considered as one of the boundary patterns.

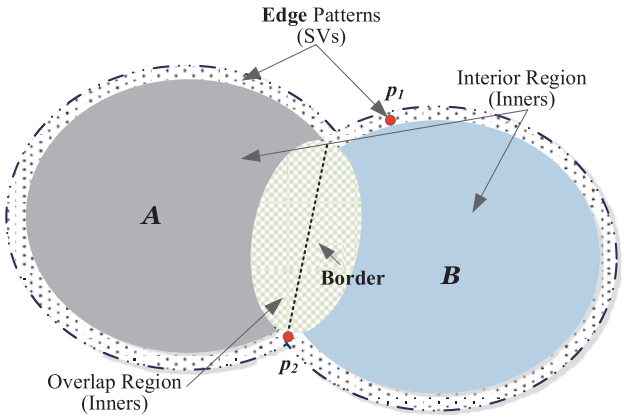


Figure 1: Edge patterns and border

2.4 Near Maximin & Random Sampling

Desired from the maximin sampling rule, NMMRS focuses on accurately portraying the distribution of $\mathscr{X}$ in the q-dimensional downspace by maximin and random sampling (MMRS) where $q \le d$. Let $\mathscr{X}_{ds} \subset \mathbb{R}^q$

be a downspace data of $\mathscr{X} \subset \mathbb{R}^d$ which is obtained by employing a random projection (RP). Following [13], NMMRS conducts three steps as follows:

- *Step 1: Collect k' maximin samples*. NMMRS finds the k' maximin samples in $\mathscr{X}_{\text{ds}}$, which are furthest from each other. Begin with a random data sample, it chooses the second maximin sample which is furthest from the initial one with respect to a chosen distance measure. The third one should have the maximum distance from the first two. This process continues until k' maximin samples are collected.
- *Step 2: Group each data sample with its nearest maximin sample*. By grouping each data sample in $\mathscr{X}_{\text{ds}}$ with its nearest maximin sample, we get k' groups of data $\{G_i\}_{i=1}^{k'}$ associating to k' maximin samples, respectively.
- *Step 3: Randomly select data near each maximin sample to obtain n samples*. The final data $\mathscr{X}_{\text{s}}$ of size $N_{\text{s}}(N_{\text{s}} \ll N)$ is built by selecting random samples from each group $G_i(i = 1, \ldots, k')$. The number n_i of samples collected from G_i is proportional to the number of data samples in G_i, i.e., $n_i = \lceil N_{\text{s}} \times |G_i|/N \rceil$.

3 The Proposed *k*ProtoClust

Towards discovering clusters without the ground truth, *k*ProtoClust adopts edge-based cluster analysis since edges describe clusters and contribute to the connectivity analysis. Therefore, split analysis with S^2PA given random seed K is designed to collect all the edge patterns forming convex hulls in local view. Mergence analysis tries to find the connectivity evidence of two neighboring convex hulls from a global view. In this section, we successively present the designs of edge collection, split and mergence analysis strategy, and finally give the algorithm description of *k*ProtoClust.

3.1 Edges in Global and Local View

In *k*ProtoClust, convex hulls with variable sizes are employed as the prototypes to support one-cluster-multiple-prototypes. From the perspective of SVDD following [12,17,18], the boundary is critical for convex hull extraction since the latter can be decomposed from the former. Even though a border is shared by two neighboring and connected convex hulls in the local view, it should be removed to accurately describe a cluster comprised of these two convex hulls in the global view. Therefore, although commonality exists, different perspectives frequently lead to very distinguished results. Without considering the border, the edge becomes the primary focus for *k*ProtoClust.

Towards accurately and visually describing the difference between edges discovered in the global and local view, we consider one cluster with two prototypes (convex hulls) denoted by $\boldsymbol{A}$ and $\boldsymbol{B}$ in Fig. 2. $\boldsymbol{A}$ and $\boldsymbol{B}$ are centers of the respective convex hulls. Generally, the global view is important to recognize edge patterns and make correct cluster judgments. As shown in Fig. 2a, we theoretically expect edge patterns in the global view to be a minimum set of data samples traveling along the cluster boundary. Due to the existence of non-convex turning points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, we have to accept the fact that there are two convex hulls. In the following context, we call these turning points split points since they are edge patterns in a cluster yet do not satisfy convexity [17] as the others. Apparently, $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ are inners, and SEPS can not extract them due to the balanced normal vectors radiating all around in their k_{e} nearest neighboring (KNN) regions.

Due to inaccurate data grouping or input space division, in the process of cluster exploration, inners like $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ may be chosen as edge patterns when $\boldsymbol{A}$ and $\boldsymbol{B}$ are split from a cluster. As depicted in Fig. 2b, in a local view, edge patterns $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ may have distinct convergence directions $\boldsymbol{n}_3$ and $\boldsymbol{n}_4$ such that they belong to different clusters A and B. This phenomenon is the same as what has been discussed in [18]. Even so, we can fortunately expect that those edge patterns collected in the global view (Fig. 2a) are still here and will not be missed (at least most of them) by SEPS in the local view with the same k_{e}. This is the commonality

of edge pattern collection from global and local perspectives. Naturally, the only difference is the occurrence of extra edges, which are the borders, if we finally get the evidence that $\boldsymbol{A}$ and $\boldsymbol{B}$ belong to the same cluster. Undoubtedly, sampled from the extra edges, $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ keep their correct roles of inners when we switch the edge analysis from their KNN regions back to the global view from the local view. That means, for the edge selection rule of $\ell \in [\gamma_l, \gamma_u]$, $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ are violating samples that exist in local analysis yet disappear from the perspective of global analysis.

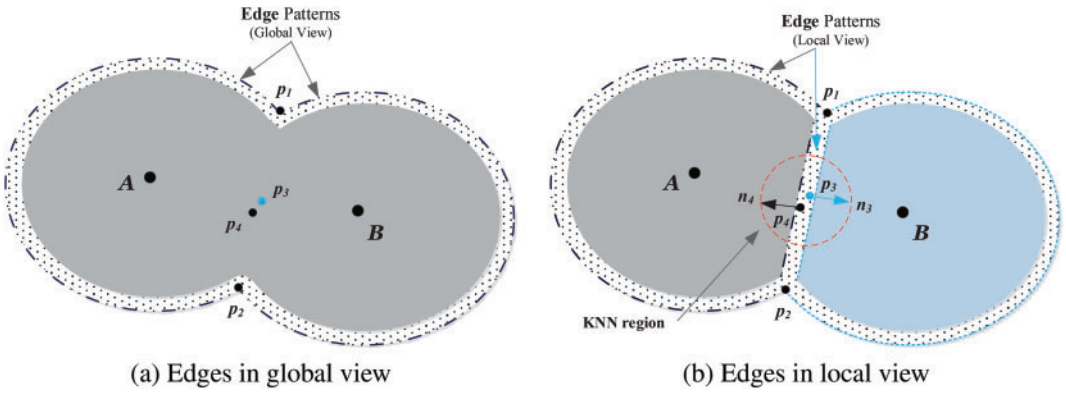


Figure 2: Edge analysis of two connected components in global and local view. In b), the two components are assigned with different labels after k-means++

Taking two irregularly shaped clusters Chameleon [19] as an example, edge patterns collected in a local view after k-means++ with $K = 5$ and in the global view ($K = 1$ equivalently) are shown in Fig. 3a,c, respectively. Obvious violating samples exist between adjacent clusters or convex hulls with connection relationships, e.g., cluster pairs $\{3, 5\}$, $\{5, 1\}$, and $\{1, 4\}$. A similar result with respect to the difference between Fig. 3c,d when we change K to 3 further confirms that the existence of violating samples is related to the connectivity determination of any two neighboring clusters or convex hulls.

3.2 Split Analysis with S^2PA

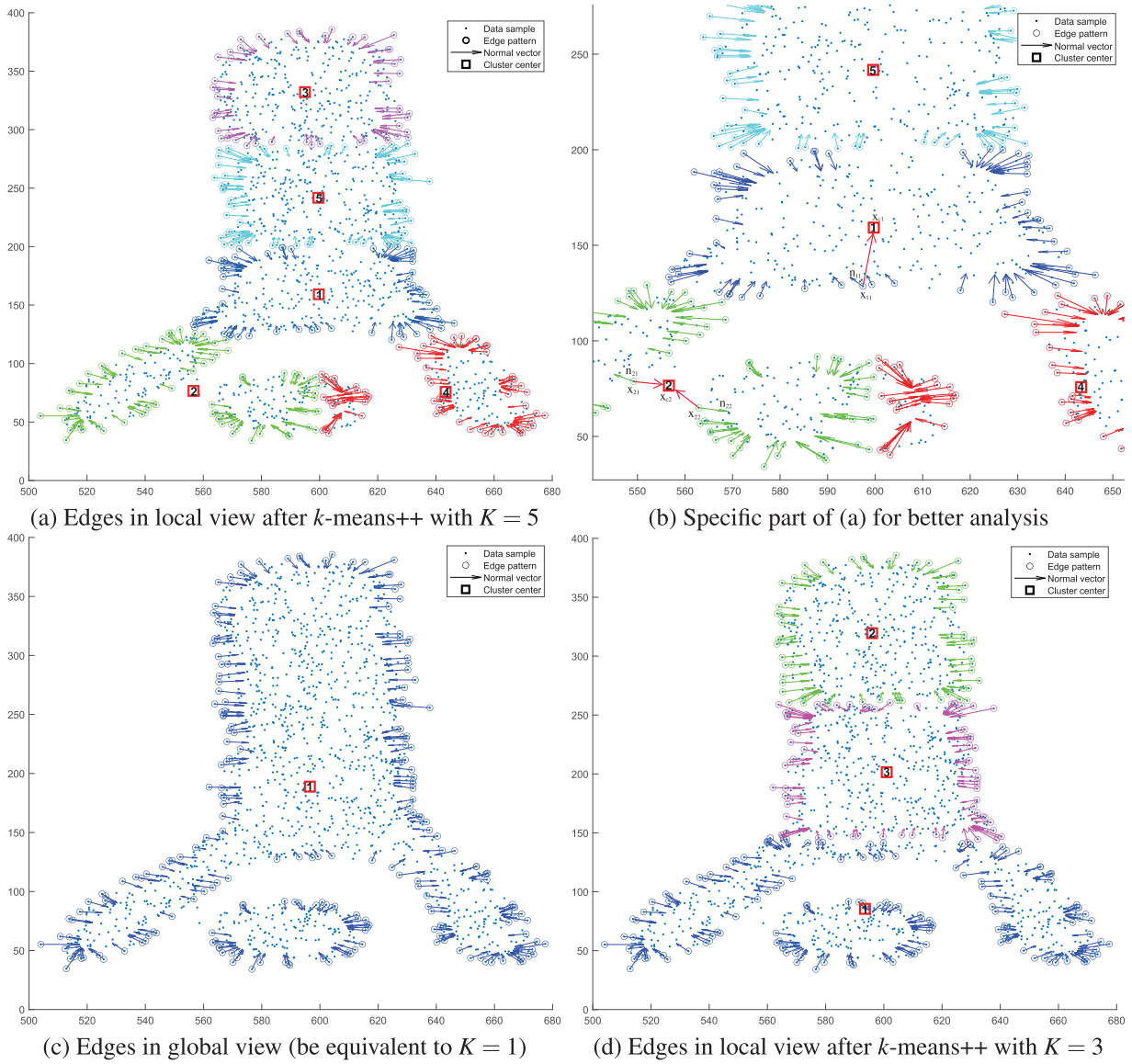
The input space division strategy of k-means+ + allows us to efficiently observe data distribution patterns in a local view. Besides the separation of components of one cluster, components from different clusters can also be mistakenly grouped due to the circle-like hypothesis. As shown in Fig. 3a, cluster 2 covers two components that should be assigned with different labels. A similar result occurs in cluster 4. Before discovering the disconnection evidence, we first retrospect the convex hull definition following [17,20].

Definition 1 (convex hull). *In Euclidean space, the convex hull of a data set $\mathscr{X}_h$ with N_h data samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{N_h}\}$ and $\mathbf{x}_i \in \mathbb{R}^d (i \in [1, N_h])$ is defined to contains all the line segments connecting each pair of $\mathscr{X}_h$. Let $CH(\mathscr{X}_h)$ be the convex hull, it can be formulated by*

$$CH(\mathscr{X}_h) = \{\mathbf{x} | \mathbf{x} = \sum_{i=1}^{N_h} \lambda_i \mathbf{x}_i, \sum_{i=1}^{N_h} \lambda_i = 1, 0 \le \lambda_i \le 1\}. \tag{5}$$

In $CH(\mathscr{X}_\mathrm{h})$, a vertex $\mathbf{x}$ is the data sample which does not satisfy $\mathbf{x} = \lambda \mathbf{x}_i + (1 - \lambda)\mathbf{x}_j$ with any two distinct samples $\mathbf{x}_i, \mathbf{x}_j \in \mathscr{X}_\mathrm{h}$ and $i, j = 1, \ldots, N_\mathrm{h}$ if $\lambda \neq \{0, 1\}$. Following the circle-like hypothesis, ideally, each cluster obtained by k-means++ should be a convex hull. Therefore, based on the Definition 1, the center of each circle-like cluster formulated by Eq. (2) must be located in the corresponding convex hull. That means each

center is an inner from the perspective of SVDD. However, the center of cluster 2 in Fig. 3a goes against this rule that motivates us to reconsider the cluster's reasonability. In other words, the current center of cluster 2 even should not be a good seed for data regrouping.



(a) Edges in local view after k-means++ with $K = 5$

(b) Specific part of (a) for better analysis

(c) Edges in global view (be equivalent to $K = 1$)

(d) Edges in local view after k-means++ with $K = 3$

Figure 3: Edge analysis on two classes of Chameleon in global and local view with $K_e = 30, \gamma_l = 0.85, \gamma_u = 1$. Each cluster is marked with the cluster index on its center

Proposition 1. For a convex hull $CH(\mathscr{X}_h)$ *with vertices* $\mathscr{X}_v$, *the included angle between the normal vector* $\boldsymbol{n}_v$ *of any vertex* $\boldsymbol{x}_v (\in \mathscr{X}_v)$ *and its direction pointing to the cluster center* $\boldsymbol{x}_c$ *is less than* 90°.

Proof. Based on Definition 1, there are N_h data samples in the convex hull $CH(\mathscr{X}_\mathrm{h})$ with center $\mathbf{x}_\mathrm{c}$. As an inner of the convex hull, $\mathbf{x}_\mathrm{c}$ can be formulated by $\mathbf{x}_\mathrm{c} = \sum_{i=1}^{N_\mathrm{h}} \lambda_i \mathbf{x}_i$ with $0 < \lambda_i < 1$. The vector between $\mathbf{x}_\mathrm{c}$ and $\mathbf{x}_v$ is $\sum_{i=1}^{N_\mathrm{h}} \lambda_i \mathbf{x}_i - \mathbf{x}_v$. Following Section 2.3, the normal vector $\mathbf{n}_v$ of any vertex $\mathbf{x}_v$ is $\sum_{i=1}^{N_\mathrm{h}} (\mathbf{x}_i - \mathbf{x}_v)$. We can formulate the cosine value of the included angle between the normal vector of $\mathbf{x}_v$ and its direction pointing to the cluster center $\mathbf{x}_\mathrm{c}$ as

$$\cos \angle \mathbf{x}_\mathrm{c}\mathbf{x}_v\mathbf{n}_v = \frac{\overrightarrow{\mathbf{x}_v\mathbf{x}_\mathrm{c}} \cdot \mathbf{n}_v}{\|\overrightarrow{\mathbf{x}_v\mathbf{x}_\mathrm{c}}\| \times \|\mathbf{n}_v\|}. \tag{6}$$

The numerator of Eq. (6) is

$$\begin{aligned}
\overrightarrow{\mathbf{x}_v\mathbf{x}_\mathrm{c}} \cdot \mathbf{n}_v &= \left(\sum_{i=1}^{N_\mathrm{h}} \lambda_i \mathbf{x}_i - \mathbf{x}_v\right) \cdot \left(\sum_{i=1}^{N_\mathrm{h}} (\mathbf{x}_i - \mathbf{x}_v)\right) \overset{\sum \lambda_i = 1}{=\!=} \left(\sum_{i=1}^{N_\mathrm{h}} \lambda_i \mathbf{x}_i - \sum_{i=1}^{N_\mathrm{h}} \lambda_i \mathbf{x}_v\right) \cdot \left(\sum_{i=1}^{N_\mathrm{h}} (\mathbf{x}_i - \mathbf{x}_v)\right) \\
&= \left(\sum_{i=1}^{N_\mathrm{h}} \lambda_i (\mathbf{x}_i - \mathbf{x}_v)\right) \cdot \left(\sum_{i=1}^{N_\mathrm{h}} (\mathbf{x}_i - \mathbf{x}_v)\right) \overset{\overline{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}_v}{=\!=} \left(\sum_{i=1}^{N_\mathrm{h}} \lambda_i \overline{\mathbf{x}}_i\right) \cdot \left(\sum_{i=1}^{N_\mathrm{h}} \overline{\mathbf{x}}_i\right) \\
&< \left(\sum_{i=1}^{N_\mathrm{h}} 1 \cdot \overline{\mathbf{x}}_i\right) \cdot \left(\sum_{i=1}^{N_\mathrm{h}} \overline{\mathbf{x}}_i\right) = \| \sum_{i=1}^{N_\mathrm{h}} \overline{\mathbf{x}}_i \|^2.
\end{aligned}$$

Meanwhile, due to $\exists \lambda_i > 0$ for $i = 1, \ldots, N_\mathrm{h}$, we have $\overrightarrow{\mathbf{x}_v\mathbf{x}_\mathrm{c}} \cdot \mathbf{n}_v > (\sum_{i=1}^{N_\mathrm{h}} 0 \cdot \overline{\mathbf{x}}_i) \cdot (\sum_{i=1}^{N_\mathrm{h}} \overline{\mathbf{x}}_i) = 0$. Thus, we get $\cos \angle \mathbf{x}_\mathrm{c}\mathbf{x}_v\mathbf{n}_v \in (0, 1)$ and the included angle has $\arccos \angle \mathbf{x}_\mathrm{c}\mathbf{x}_v\mathbf{n}_v \in (0, 90°)$. □

Based on the properties of convex hull and convex hull based decomposition [17], experimental result on Chameleon shown in Fig. 3b confirms the aforementioned two features.

- First, the center $\mathbf{x}_{c2}$ of cluster 2 violates the convex hull's property. The center should be an inner of a convex hull. A similar situation can be found in cluster 4. They are not appropriate centers or suitable for acting as seeds for k-means++. To reduce redundant computations, the proposed S^{2}PA strategy is quite simple and intuitive: to avoid non-inner samples becoming seeds for k-means++ based on the position analysis of SEPS.
- Second, $\mathbf{x}_{c2}$ is also a violating sample of the Proposition 1. $\mathbf{x}_{21}$ and $\mathbf{x}_{22}$ are two randomly selected data samples in cluster 2 judged by k-means++ with normal vectors $\mathbf{n}_{21}$ and $\mathbf{n}_{22}$, respectively. Obviously, both $\angle \mathbf{n}_{21}\mathbf{x}_{21}\mathbf{x}_{c2}$ and $\angle \mathbf{n}_{22}\mathbf{x}_{22}\mathbf{x}_{c2}$ are greater than 90°. It means that cluster 2 either contains at least two convex hulls from different clusters or has two prototypes of convex hulls due to irregular cluster shapes. Therefore, a cluster is suggested to be split if it violates the rule of the Proposition 1. On the contrary, $\mathbf{x}_{c1}$ follows the Proposition 1 well with a randomly chosen $\angle \mathbf{n}_{11}\mathbf{x}_{11}\mathbf{x}_{c1} < 90°$.

Based on the aforementioned S^{2}PA strategy and violating sample analysis, an essential cluster split for mistake correction can be briefly illustrated in Algorithm 1. Given a set of edge pattern groups $X_\mathrm{e} = \{X_{\mathrm{e}1}, X_{\mathrm{e}2}, \cdots, X_{\mathrm{e}K}\}$ with the corresponding normal vector set $N_\mathrm{e} = \{N_{\mathrm{e}1}, N_{\mathrm{e}2}, \cdots, N_{\mathrm{e}K}\}$, and the set of centroids $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_K\}$, the algorithm SplitAnalysis checks each cluster's reasonability, conducts iterative split analysis with irrefutable evidence, and outputs an updated set of edge pattern groups $X_\mathrm{e} = \{X_{\mathrm{e}1}, X_{\mathrm{e}2}, \cdots, X_{\mathrm{e}K_\mathrm{p}}\}$ with centroids $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_{K_\mathrm{p}}\}$ where $K_\mathrm{p} \geq K$. Notice that the working set $X_\mathrm{w} \subseteq X_{\mathrm{e}i}$ selection in line 2 is employed for efficiency while avoiding missing sufficient evidence of cluster split. To avert excessive split, we suggest an accumulated analysis by

$$p_d = \frac{1}{|X_\mathrm{w}|} \sum_{j=1}^{|X_\mathrm{w}|} \mathrm{sign}(-\cos \angle \boldsymbol{c}_i\mathbf{x}_j\mathbf{n}_j), \tag{7}$$

where $|X_w|$ is the size of X_w, $\mathbf{x}_j \in X_w$, and $\mathbf{n}_j \in N_{ei}$. The function sign(x) returns 1 if $x > 0$, otherwise it returns 0. The other function isInner ($\boldsymbol{c}_i, k_e$) is to check whether the data $\mathbf{x}$ is an inner (true) or not (false) by employing SEPS. Following Proposition 1, isInner ($\boldsymbol{c}_i, k_e$) can also be simplified by checking the included angle $\angle \boldsymbol{c}_i\mathbf{x}_i\mathbf{n}_i$ where $\mathbf{x}_i$ is the nearest neighbor of $\boldsymbol{c}_i$ in the same cluster. The replacement and update works in lines 6–8 are simple, e.g., line 6 does $X_e \leftarrow \{X_e \backslash \{X_{ei}\}\} \bigcup \{X_{ei}^1, X_{ei}^2\}$. Apparently, through the recursive call of SplitAnalysis in line 9, we transfer the flaw of the circle-like pattern hypothesis into its advantage of forming convex hulls with different sizes. These convex hulls are the cornerstone of multiple prototype support in cluster analysis. Meanwhile, following the previous discussion and [12,17] we can check the included angle declared in Proposition 1 for the evidence of cluster split or mergence in stead of direct constructing convex hulls for simplicity.

Algorithm 1: SplitAnalysis

Require: Set of edge groups X_e with its normal vector set N_e, the set of centroids C, and thresholds τ_d, k_e
Ensure: Updated sets of edge groups X_e, normal vectors N_e and centroids C

1. **for all** $\boldsymbol{c}_i$ in C **do**
2. Uniformly select a working set X_w from X_{ei}
3. $p_d = \frac{1}{|X_w|}\sum_{j=1}^{|X_w|}\text{sign}(-\cos\angle \boldsymbol{c}_i\mathbf{x}_j\mathbf{n}_j)$
4. **if** !isInner($\boldsymbol{c}_i, k_e$) $||p_d \geq \tau_d$ **then**
5. $\{X_{ei}^1, X_{ei}^2; \boldsymbol{c}_i^1, \boldsymbol{c}_i^2\} \leftarrow k$-means++($\mathscr{X}_{ei}$, 2)
6. replace X_{ei} by $\{X_{ei}^1, X_{ei}^2\}$
7. update N_{ei} by $\{N_{ei}^1, N_{ei}^2\}$ along with X_{ei}'s change
8. replace $\boldsymbol{c}_i$ by the set $C_i = \{\boldsymbol{c}_i^1, \boldsymbol{c}_i^2\}$
9. SplitAnalysis ($X_{ei}, N_{ei}, C_i, \tau_d$)
10. **end if**
11. **end for**

3.3 Mergence Analysis

According to the analysis in Section 3.1, violating samples exist in the overlapping region of two adjacent clusters or convex hulls. Therefore, as shown in Algorithm 2, the mergence analysis of convex hulls extracted by SplitAnalysis is quite intuitive: find out violating samples and merge the associated convex hulls to add prototypes for the corresponding cluster.

Algorithm 2: MergenceAnalysis

Require: Dataset $\mathscr{X}_s$, the set of edge groups X_e with its normal vector set N_e, the set of centroids C, thresholds $\gamma_l, \gamma_u, \tau_m, \tau_{im}$ and integer k_e
Ensure: Set of prototype groups P and their labels L_P

1. $\Delta X_e \leftarrow \varnothing, \Delta N_e \leftarrow \varnothing, \boldsymbol{A} = \boldsymbol{I}_K$ //Adjacent matrx $\boldsymbol{A}$
2. $\{X_{ge}, N_{ge}\} \leftarrow$EdgeSel($\mathscr{X}_s, k_e, \gamma_l, \gamma_u$) // global view
3. $\{\Delta X_e, \Delta N_e\} \leftarrow$DiffSet($X_e, X_{ge}, N_e, N_{ge}$)
4. **for** $i = 1, 2, \cdots, K_p$ **do**
5. **if** $|\Delta X_{ei}| == 0$ **then**
6. $\{\mathbf{x}_i, \mathbf{x}_j\} \leftarrow$ find the nearest point-pair from X_{e_i}, X_{e_j} with $\boldsymbol{A}_{ij} == 0$, respectively
7. **if** $\cos\angle \mathbf{n}_i\mathbf{n}_j > \tau_m$ **then**
8. $\boldsymbol{A}_{ij} \leftarrow 1, \boldsymbol{A}_{ji} \leftarrow 1$ //connected prototypes i, j

(Continued)

Algorithm 2 (continued)

```
 9.      end if
10.  else
11.      b = {0}^{1×K_P}, cnt = {0}^{1×K_P} //imbalance degree
12.      for v = 1, 2, ···, |ΔX_ei| do
13.          {n_v, n_j, L_j} ← KNNRegion(x_v, ΔX_e, ΔN_e, k_e)
14.          if A_{iL_j} == 1 then
15.              continue // skip, go to v + 1
16.          end if
17.          b_{L_j} ← b_{L_j} + (n_v + n_j) / (2×min{n_v, n_j})
18.          cnt_{L_j} ← cnt_{L_j} + 1
19.      end for
20.      b ← b./cnt // Division
21.      for j = 1, 2, ···, K_P do
22.          if b_j ≠ 0 and b_j ≤ τ_im then
23.              A_ij ← 1, A_ji ← 1 //connected prototypes i, j
24.          end if
25.      end for
26.  end if
27. end for
28. L_P ← find the connected prototypes using A
29. P ← get the set of prototype groups where P_i = ⋃_j c_j and label(c_j) = i
```

Algorithm 2 of MergenceAnalysis invokes EdgeSel(·) firstly, which implements SEPS to collect edge patterns X_{ge} in the global view. Then, DiffSet(·) gets the difference set ΔX_e between X_{ge} and X_e. Here, X_e are edge patterns collected in local view, and ΔN_e in line 3 is the set of normal vectors corresponding to ΔX_e. Based on ΔX_e and ΔN_e, lines 4–27 try to check the connectivities among k-prototype obtained by SplitAnalysis. Each prototype is represented by its center $\boldsymbol{c}_i \in C$ for simplicity.

If the current convex hull has no violating samples, lines 5–9 find two nearest edge patterns $\mathbf{x}_i$ and $\mathbf{x}_j$ separately from it and the remaining unconnected convex hulls. As a specific case extended from the KNN region, the normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$ keep similar directions, although they are mistakenly divided into different groups. Thus, lines 7–9 set the adjacent matrix with $\boldsymbol{A}_{ij} = 1$ and $\boldsymbol{A}_{ij} = 1$ to declare the connection while $\cos \angle \mathbf{n}_i \mathbf{n}_j > \tau_m$. Here, τ_m is the threshold for this judgment.

Line 10 starts a series of distinguished works with violating samples in the current convex hull. Each KNN Region around the violating point will be carefully checked to obtain an accumulated imbalance for each adjacent prototype pair. When KNNRegion(·) checks the nearest k_e neighbors of $\mathbf{x}_v$ from ΔX_e, we have introduced a specific constraint that every considered neighbor $\mathbf{x}_j$ should have different convergence direction with $\mathbf{x}_v$ unless it has the same label with $\mathbf{x}_v$. That means the included angle $\angle \mathbf{n}_v \mathbf{x}_v \mathbf{n}_j$ must be greater or equal to 90° if $L_j \neq i$; otherwise we skip this neighbor and let $k_e \leftarrow k_e - 1$ in this round. The constraint is to avoid excessive sampling beyond the overlapping area. Thus, KNNRegion(·) returns the number of neighbors n_v in the convex hull of $\mathbf{x}_v$, the number of neighbors n_j from the largest class with label L_j and $L_j \neq i$. The complete procedure is presented by Algorithm 3. To assist the connectivity analysis, we accumulate all the imbalance degrees of KNN Regions by considering all the data samples in the overlapping region between each adjacent prototypes-pair. Following [21], the imbalance degree of the KNN Region around $\mathbf{x}_v$ is formulated by

Algorithm 3: KNNRegion

Require: A data sample $\mathbf{x}_v$, a set of edge patterns ΔX_e and the corresponding normal vector set ΔN_e, and an integer k_e

Ensure: The number of neighbors n_v in the convex hull of $\mathbf{x}_v$ including itself, the number of neighbors n_j from the largest class with label L_j and $L_j \neq i$.

1. $n_v \leftarrow 1$, $n_j \leftarrow 0$, $S \leftarrow \varnothing$ // S stores <label, count>
2. $X_{k\mathrm{NN}} \leftarrow$ select the k_e nearest neighbors from $\Delta X_e \backslash \mathbf{x}_v$
3. **for all** $\mathbf{x}_j$ in $X_{k\mathrm{NN}}(j = 1, 2, \cdots, k_e)$ **do**
4. **if** label($\mathbf{x}_j$) == label($\mathbf{x}_v$) **then**
5. $n_v \leftarrow n_v + 1$ //neighbor from the same convex hull
6. **else if** $\cos \angle \mathbf{n}_v \mathbf{x}_v \mathbf{n}_j < 0$ **then**
7. $s \leftarrow <$ label$(\mathbf{x}_j), 1>$
8. **if** $\exists$ label = label$(\mathbf{x}_j) \in S$ **then**
9. increase 1 on count with label($\mathbf{x}_j$)
10. **else**
11. $S \leftarrow S \bigcup \{s\}$
12. **end if**
13. **end if**
14. **end for**
15. $< n_j, L_j >\leftarrow$ find the item with the largest `count` in S
16. **return** n_v, n_j, L_j

$$b = \frac{n_v + n_j}{2 \times \min\{n_v, n_j\}}, \tag{8}$$

where only two different labels $\{i, L_j\}$ are considered. The greater the gap between n_v and n_j is, the larger the imbalance is in the KNN Region. The imbalance degree is 1 for a balanced region when we have $n_v = n_j$. Consequently, as shown in lines 18–22 of Algorithm 2, an accumulated imbalance lower than or equal to τ_{im} suggests that the two adjacent prototypes are connected. Finally, we find all the standalone and connected prototypes based on $\boldsymbol{A}$. The former means a cluster with one prototype, whereas the latter suggests multiple prototypes in a cluster. The finally discovered cluster number K_f is the number of subsets in P.

3.4 Implementation of kProtoClust

Based on the split and mergence analysis, the proposed *k*ProtoClust is detailed by Algorithm 4.

Algorithm 4: kProtoClust

Require: Dataset $\mathscr{X}$, initial cluster number K, ratio ρ, thresholds $\gamma_l, \gamma_u, \tau_d, \tau_{\mathrm{m}}, \tau_{\mathrm{im}}$, integers q, k_e

Ensure: The set of prototypes P with labels L_{P} and L_{All}

1. $\mathscr{X}_{\mathrm{ds}} \leftarrow$ generate downspace dataset of $\mathscr{X}$ using RP
2. $\mathscr{X}_s \leftarrow$ select $N_s = \rho N$ samples from $\mathscr{X}_{\mathrm{ds}}$ using MMRS
3. $\{\mathscr{X}_s', C\} \leftarrow k$-means++ $(\mathscr{X}_s, K)$ // partition $\mathscr{X}_s$ into K clusters saved in $\mathscr{X}_s'$ with centroids in C
4. $\{X_e, N_e\} \leftarrow$EdgeSel$(\mathscr{X}_s', k_e, \gamma_l, \gamma_u)$ // select edge patterns from each cluster to set X_e
5. $\{X_e, N_e, C\} \leftarrow$ SplitAnalysis$(X_e, N_e, C, k_e, \tau_d)$ // decompose every divisible clusters into convex hulls
6. $\{P, L_{\mathrm{P}}\} \leftarrow$ MergeAnalysis$(\mathscr{X}_s, X_e, N_e, C, k_e, \gamma_l, \gamma_u, \tau_{\mathrm{m}}, \tau_{\mathrm{im}})$ //group all the connected convex hulls
7. $L_{\mathrm{All}} \leftarrow$LabelAssignment$(\mathscr{X}_{\mathrm{ds}}, P, L_{\mathrm{P}})$ //label each data sample with its nearest prototype in P

On the basis of keeping data distribution patterns, data sampling and dimension reduction are effective ways for efficiency. As data preparation works, however, they are frequently optional since only some of the data sets to be analyzed are large-scale with high dimensions. Therefore, Algorithm 4 prefers NMMRS [13] in lines 1–2 to make the following analysis done in the downspace if the target data is large-scale with high dimensions. Then, a standard k-means++ is employed to partition data into K clusters for further edge pattern selection in the local view. Since the difference of edge patterns collected in terms of local and global views mainly lies in the border patterns, which are also inners in the perspective of SVDD. K is not a decisive prior knowledge. How to init the K value in the first round of line 3 only depends on the efficiency requirement from EdgeSel(·) (using SEPS) in line 4. By introducing the global analysis, SplitAnalysis(·) in line 5 will try to find all the divisible ones from K clusters and decompose them into different numbers of convex hulls. In general, the number of decomposed convex hulls K_p is greater than or equal to K. As prototypes, all the convex hulls' connection relationships are determined by MergeAnalysis(·) in line 6, which groups all the convex hulls with the same label together. By now, each group of connected convex hulls $P_i \in P$ forms a cluster. So, the final cluster number discovered is not limited to K or K_p. The last phase is to assign all the data samples with appropriate labels based on their distance to prototypes. In this step, kProtoClust prefers each prototype to be represented by its center, i.e., the mean of the convex hull's vertices. For ease of reading, we summarize all the notations in Algorithm 4 in Table 1.

Table 1: Notations in kProtoClust

Notation	Description
$\mathscr{X}$	The original data set $\mathscr{X}$ with N data samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ and each one in $\mathbb{R}^d$.
$\mathscr{X}_\mathrm{ds}$	The downspace dataset of $\mathscr{X}$ using RP with each sample in $\mathbb{R}^q$ and $q \le d$.
$\mathcal{X}_\mathrm{s}$	Dataset with $N_\mathrm{s} = \rho N$ samples selected from $\mathscr{X}_\mathrm{ds}$ using MMRS.
$\mathscr{X}_\mathrm{s}', C$	Cluster set $\mathscr{X}_\mathrm{s}'$ has K clusters $\mathscr{X}_{\mathrm{s}1}', \mathscr{X}_{\mathrm{s}2}', \cdots, \mathscr{X}_{\mathrm{s}K}'$ and their centroids $\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K\}$ form C.
k_e	The k_e nearest neighbors is considered by EdgeSel(·) using SEPS.
γ_l, γ_u	Thresholds γ_l and $\gamma_u (0 < \gamma_l < \gamma_u \le 1)$ control the curvature and the shrinkage degree.
$X_\mathrm{e}, N_\mathrm{e}$	Set of edge groups X_e with its normal vector set N_e.
τ_d	A threshold to determine whether a cluster is divisible.
τ_m	A threshold to determine whether two adjacent data samples are in a cluster.
τ_im	A threshold to determine whether two adjacent prototypes (clusters) are connected.
P, L_P	Set of prototype groups P and their labels L_P.
L_All	Labels of all the N data samples in $\mathscr{X}$.

3.5 Time Complexity

As shown in Algorithm 4, kProtoClust comprises six critical tasks. NMMRS consists of the first two lines yet optional data preparation works, as discussed in [13], which are suggested to be invoked if $\mathscr{X} \subset \mathbb{R}^d$ is a large-scale and high dimensional dataset. RP is a simple computation work that consumes $O(dqN)$ where q is the final dimension reduced from d following the *Johnson-Lindenstrauss* Lemma [22]. With an appropriate ρ to keep data distribution pattern, MMRS requires $O(qk'N)$ to divide data into k' groups and extract a subset $\mathscr{X}_\mathrm{s} (\subset \mathscr{X}_\mathrm{ds})$ which has $N_\mathrm{s} (\ll N)$ samples for the following analysis and prototypes extraction.

Leaving NMMRS, k-means++ [7] costs $O(qKN_s)$ to divide $\mathscr{X}_s$ into K clusters. Then, EdgeSel(·) takes $O(N_s^2)$ to collect edge patterns in local view. The following two phases are critical for K rectification because kProtoClust does not consider K a prior knowledge for k-means++. First, SplitAnalysis(·) is a recursive algorithm. For each round, the worst situation is that k-means++($\mathscr{X}_{ei}$, 2) with $O(2qN_{ei})$ cannot be avoided, whereas the best situation is no further division required. Consider a K centers' traversal on average, invoking k-means++ K times means that all the edge patterns take part in the division, i.e., $N_e = \sum_{i=1}^{K} N_{ei}$. So the cost for each round can be approximated to $O(2qN_e)$. Assume that there are ℓ rounds of recursion, SplitAnalysis(·) may be finished in $O(2q\ell N_e)$. Second, MergeAnalysis(·) is conducted based on the K_p convex hulls discovered by SplitAnalysis(·). Due to $|\Delta X_e| = \sum_{i=1}^{K_p} |\Delta X_{ei}|$, lines 4–27 consume $O(|\Delta X_e|^2)$ if we have to invoke KNNRegion(·). It is the same as the requirement of line 28. However, we find that EdgeSel(·) in line 2 is the most time-consumption work for edge patterns extracted from $\mathscr{X}_s$ in global view since $|\Delta X_e| \ll N_s$. Therefore, MergeAnalysis(·) costs $O(N_s^2)$. Hereto, all the k-prototype unevenly distributed in different clusters are collected. So the last task LabelAssignment(·) can be finished in $O(K_pN)$.

The overall computational complexity of kProtoClust is $O(dqN + qk'N + qKN_s + 2N_s^2 + 2q\ell N_e + K_pN)$. Simply stated, it ranges from $O(N_s^2)$ to $O(N^2)$ corresponding to employing the sampling strategy or not.

4 Experimental Results

4.1 Datasets and Experimental Settings

Inspired by a union analysis in the local and global view, kProtoClust provides adaptive multiple prototypes support while guaranteeing the fundamental principle of k-means. We conduct the following four series of experiments on various datasets to achieve a complete performance analysis.

- Do parameter sensitivity analysis on clustering accuracy and the discovered cluster number with respect to parameters K, τ_d, τ_m and τ_{im}, even though the last three are suggested to be derived from the human's basic cognition of cluster discrimination. That means τ_d, τ_m and τ_{im} can be either fixed values for universality or may vary with each individual because different peoples can have distinct connectivity assertions of weak connection. The remaining parameters $\{\gamma_l, \gamma_u\}$ and k_e were respectively discussed by [12] and [16].
- Perform descriptive ability analysis for kProtoClust, in terms of accuracy, on classic datasets (type I in Table 2) with known cluster numbers. Since the foremost design of kProtoClust is to collect and well utilize the difference captured by local and global analysis, for fairness, we consider a compact version of kProtoClust including only lines 3–7 of Algorithm 4. Eight state-of-the-art algorithms are baselines: $k-$means, $k-$means++, Ball $k-$means [2], deep $k-$means with pretraining [23], coordinate descent based k-means (CDKM) [24], t-k-means, t-k-means++ [25], and a hybrid method of k-means with split-merge strategy (SMKM) [9]. Besides, k-median [26], k-medoid [27], a clustering method from the t-mixture model (TMM) [25], and EAC [5] are also introduced.
- Check the cluster discovery ability of the compact kProtoClust on datasets of type I in Table 2 without the prior knowledge of cluster number K_c. The selected baselines are those methods with relatively better performance in the second experiment.
- Explore some applicable suggestions by verifying the effectiveness of kProtoClust with NMMRS (lines 1–2 of Algorithm 4) denoted by kProtoClust (RS) on datasets of type II in Table 2 which have either large number of samples or high dimensionality.

Table 2: Description of the benchmark data sets

Data sets	Data set description			Data sets	Data set description		
(Type I)	Size	Dims	# of classes (K_c)	(Type II)	Size	Dims	# of classes (K_c)
Chameleon	7670	2	8	movement_libras	360	90	15
glass	214	9	7	Twonorm	7400	20	2
wisconsin	683	9	2	uspst	2007	256	10
abalone	4177	7	29	Ohsumed	13,929	23	23
WebKB	4199	4	4	shuttle	43,500	9	7
P2P traffic	9206	4	4	kddcup99	494,021	9	5

Table 2 lists the twelve employed datasets' statistical information. Type I denotes small data with very irregular cluster shapes, while type II means more samples or higher dimensionality. They will be considered separately in different experiments to achieve a clear analysis. glass, wisconsin, abalone, movement_libras, Twonorm, uspst, and shuttle are from UCI repository [28]. WebKB [29] and Ohsumed [30] are pre-processed versions from [12]. Chameleon is provided by [19], and P2P traffic has 9206 flows' features extracted by [31]. kddcup99 is a 9-dimensional data set extracted from KDD Cup 1999 Data [32] which is typical for intrusion detection. All the datasets are listed in the sequence of complexity NdK_c from lowest to highest, where K_c corresponds to "# of classes" in Table 2.

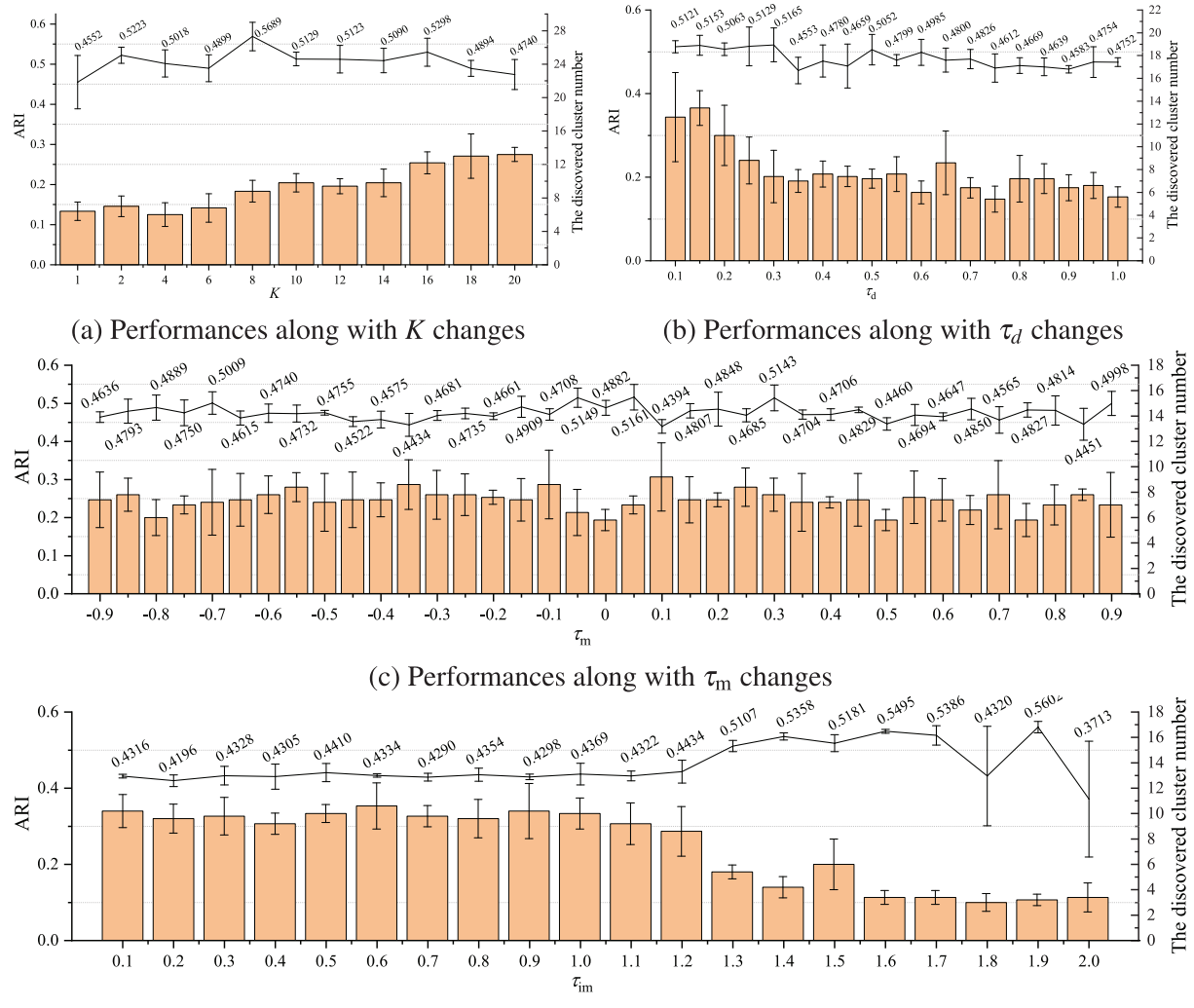
To evaluate the accuracy, adjusted rand index (ARI) [33], normalized mutual information (NMI) [34], and F-measure [35] are preferred. All the compared algorithms are implemented by MATLAB 2021b on a mobile workstation with Intel I9-9880H and 128 GB DRAM running Windows 10-X64. Since the main objectives are to verify the descriptive and cluster discovery abilities of the proposed *k*ProtoClust, we do not pursue the maximal efficiency and the run-time is for limited reference.

4.2 Parameter Sensitivity Analysis

4.2.1 Initial Cluster Number K

Generally, unsupervised learning faces the critical challenge of lacking sufficient prior knowledge, such as the cluster number K_c. For instance, along with various variants of intrusion behavior appearing, we can not simply consider K_c being equivalent to the number of categories. Different perspectives of the bounds for data grouping frequently lead to distinct clustering results. By fixing $\tau_d = 0.3$, $\tau_m = 0.1$ and $\tau_{im} = 1.2$, `Chameleon` [36] with eight irregular clusters is adopted to make an intuitive analysis of the influence from the initial cluster number K on accuracy and the final cluster number K_f. Fig. 4a depicts the results in which ARI's mean and standard variance and the discovered K_f are separately collected for each K. As K increases, *k*ProtoClust gets stable accuracies whose mean values fluctuate in a range $[-5.10\%, 8.86\%]$ around 0.4994. In contrast, accuracies achieved by *k*-means++ have a relatively large fluctuation from 0 to 0.4982. Particularly, the best accuracies obtained by *k*-means++ are 0, 0.1229, and 0.4025 when K is set to 1, 2, and 4, respectively. In theory, *k*ProtoClust may fail when K is set to 1 since edge patterns collected in terms of local view and global view are the same. However, lines 5–10 of Algorithm 2 bring a chance to

reconsider the connectivity of prototype-pairs split by Algorithm 1 under a human basic cognition parameter τ_{m}. Even so, $K = 1$ should be avoided to get the utmost out of union analysis. The mean value of the discovered K_{f} ranges from 6.0 to 13.2. Even though it is a relatively large fluctuation around the ground truth 8, the majority falls within $[6.8, 9.8]$, and the dramatic change begins with $K \geq 16$. This phenomenon gives us a notice about the choice of cluster number, i.e., the voting strategy also works even though we begin with different K. Therefore, *k*ProtoClust can benefit from further data analysis for reducing the constraint from the prior knowledge of cluster number.



(a) Performances along with K changes

(b) Performances along with τ_d changes

(c) Performances along with τ_{m} changes

(d) Performances along with τ_{im} changes

Figure 4: Sensitive analysis by means and standard variations of accuracy (ARI) and the cluster number (K_{f}) discovered by the compact version of *k*ProtoClust on Chameleon along with the change of K, τ_d, τ_{m} and τ_{im} while $\gamma_l = 0.85$, $\gamma_u = 1$ and $k_{\mathrm{e}} = 30$. These four parameters are fixed to $K = 8$ (the same as K_{c}), $\tau_d = 0.3$, $\tau_{\mathrm{m}} = 0.1$ and $\tau_{\mathrm{im}} = 1.2$ when we do sensitivity analysis for each of them

4.2.2 Hyperparameters τ_d, τ_m and τ_{im}

τ_d is defined in Algorithm 1. It is an upper bound for the allowed proportion of edge patterns in a cluster yet has a distinct convergence direction rather than pointing to the cluster center. The smaller the chosen value of τ_d is, the more confidence *k*ProtoClust emphasizes cluster splitting. As shown in Fig. 4b, given $K = 8$, the discovered K_f with $\tau_d < 0.3$ is obviously greater than that of found with $\tau_d \geq 0.3$. Due to imbalanced data distribution, we also reach greater ARIs with more clusters decomposed from the given data space in these cases. As τ_d increases, only a few changes in the discovered K_f are naturally determined by the data distribution structure. Therefore, in this study, we accept $\tau_d = 0.3$ as the reference line for checking whether a cluster is worthy of being split. Notice that one can fix τ_d to any other value to meet his personal decision although *k*ProtoClust has relatively stable performances with $\tau_d > 0.3$.

Due to imbalanced distribution, non-accurate parameter settings in SEPS may miss some border patterns even though the nearest neighboring convex hulls are split from one cluster. To deal with this case, τ_m is introduced in lines 5–10 of Algorithm 2. It should be considered a relatively extreme condition but not always exists. Results shown in Fig. 4c confirm that both of the accuracy and discovered K_f have relatively high stability as τ_m increases. Because, Chameleon without noise has a relatively clear boundary in which an edge pattern can hardly have its nearest neighbor from another cluster. Therefore, without loss of generality, based on Proposition 1, we prefer $\tau_\mathrm{m} = -0.3$ (i.e., 107.5°) not $\tau_\mathrm{m} = 0.0$ (i.e., 90.0°) or $\tau_\mathrm{m} = -0.2$ (i.e., 101.5°) to tolerate uncertainty of convergence direction although the latter two has smaller variance values. In another way, based on the observation, τ_m can be set flexibly if the target data has a clear boundary.

In KNN region, the more balanced the edge patterns from different clusters are, the cumulative imbalance degree is close to 1. Theoretically, the bound τ_im in line 22 of Algorithm 2 should be greater than or equal to 1. We add a range of $[0.1, 1)$ to τ_im to make the analysis more comprehensive. Apparently, the trends of the accuracies and the discovered K_f in Fig. 4d show sufficient evidence. Along with the increase of τ_im, many more extremely weak-connected convex hulls are considered connected, leading to continuous reduction of K_f. In this study, we take $\tau_\mathrm{im} = 1.2$ as the bound for connection.

4.3 Performance Contrast with the Prior Knowledge K_c

The prior knowledge of cluster number is critical for *k*-means and its variants. Although *k*ProtoClust is also derived from *k*-means, the design strategy of multiple prototypes support is to break through this constraint by emphasizing the descriptive ability. To verify the effectiveness, we compare it with twelve state-of-the-art methods on datasets of type I in terms of three accuracy metrics after thirty times of evaluations. The achieved mean and mean-square deviation (std) for each metric is illustrated in Table 3. Since descriptive ability is a common concern for all the methods considered from different perspectives, such as initialization strategy, data distribution description, or voting analysis, we omit efficiency comparisons for insignificant differences in low-complexity data despite having irregular clusters.

Table 3: Accuracy comparisons in terms of ARI, NMI and F1 on Datasets of Type I with Known K_c

Methods/Dataset	Chameleon			Glass		
	ARI	NMI	F1	ARI	NMI	F1
k-means	0.3953 ± 0.0252	0.6213 ± 0.0132	0.5767 ± 0.0110	0.2793 ± 0.0032	0.4323 ± 0.0099	0.5646 ± 0.0107

(Continued)

Table 3 (continued)

Methods/Dataset	Chameleon			Glass		
	ARI	**NMI**	**F1**	**ARI**	**NMI**	**F1**
k-means++	0.3996 ± 0.0298	*0.6315* ± *0.0182*	0.5848 ± 0.0244	**0.2913 ± 0.0036**	**0.4589 ± 0.0348**	**0.5935 ± 0.0187**
Ball *k*-means	0.3953 ± 0.0252	0.6213 ± 0.0132	0.5767 ± 0.0110	0.2774 ± 0.0041	0.4236 ± 0.0111	0.5619 ± 0.0127
Deep *k*-means	0.3324 ± 0.1671	0.4641 ± 0.2332	0.2956 ± 0.1311	0.0910 ± 0.0290	0.1662 ± 0.0371	0.3714 ± 0.0413
CDKM	0.3956 ± 0.0197	0.6005 ± 0.0176	0.5393 ± 0.0170	0.2799 ± 0.0007	0.4215 ± 0.0175	0.5537 ± 0.0047
SMKM	0.4423 ± 0.0064	0.6095 ± 0.0291	0.6048 ± 0.0197	0.2606 ± 0.0127	0.4059 ± 0.0353	0.5571 ± 0.0305
t-*k*-means	0.3915 ± 0.0000	0.5946 ± 0.0000	0.5444 ± 0.0000	0.2868 ± 0.0000	*0.4348* ± *0.0110*	0.5681 ± 0.0018
t-*k*-means++	0.3915 ± 0.0000	0.5946 ± 0.0000	0.5444 ± 0.0000	0.2820 ± 0.0065	0.4305 ± 0.0144	0.5667 ± 0.0038
k-median	0.4054 ± 0.0237	0.6003 ± 0.0174	0.5681 ± 0.0172	0.2016 ± 0.0350	0.3248 ± 0.0440	0.4992 ± 0.0427
k-medoid	0.4294 ± 0.0402	0.6257 ± 0.0378	0.5874 ± 0.0223	0.2666 ± 0.0129	0.3888 ± 0.0183	0.5561 ± 0.0124
TMM	*0.5576* ± *0.0692*	0.6147 ± 0.0426	*0.6600* ± *0.0518*	0.2429 ± 0.0150	0.4298 ± 0.0172	0.5546 ± 0.0088
EAC	0.5562 ± 0.0745	0.6811 ± 0.2395	0.5817 ± 0.2045	0.0106 ± 0.0000	0.0951 ± 0.0000	0.4139 ± 0.0000
*k*ProtoClust	**0.5689 ± 0.0352**	**0.7265 ± 0.0055**	**0.6771 ± 0.0339**	*0.2881* ± *0.0047*	0.4293 ± 0.0256	*0.5869* ± *0.0212*
Methods/Dataset	wisconsin			abalone		
	ARI	NMI	F1	ARI	NMI	F1
k-means	0.8487 ± 0.0030	0.7519 ± 0.0037	*0.9612* ± *0.0008*	0.0469 ± 0.0012	0.1813 ± 0.0009	0.1707 ± 0.0024
k-means++	0.8498 ± 0.0030	*0.7549* ± *0.0021*	**0.9644 ± 0.0005**	0.0525 ± 0.0012	**0.1981 ± 0.0031**	0.1934 ± 0.0065
Ball *k*-means	0.8487 ± 0.0030	0.7519 ± 0.0037	*0.9612* ± *0.0008*	0.0469 ± 0.0012	0.1813 ± 0.0009	0.1707 ± 0.0024
Deep *k*-means	0.8391 ± 0.0102	0.7421 ± 0.0130	0.9542 ± 0.0030	*0.0544* ± *0.0027*	0.1374 ± 0.0051	0.0646 ± 0.0073
CDKM	0.8546 ± 0.0000	0.7478 ± 0.0000	0.9603 ± 0.0000	0.0449 ± 0.0009	*0.1820* ± *0.0010*	0.1717 ± 0.0025
SMKM	*0.8587* ± *0.0108*	0.7722 ± 0.0199	0.9637 ± 0.0030	0.0549 ± 0.0014	0.1786 ± 0.0051	0.2002 ± 0.0063

(Continued)

Table 3 (continued)

Methods/Dataset	Chameleon			Glass		
	ARI	**NMI**	**F1**	**ARI**	**NMI**	**F1**
t-k-means	*0.8546* ±*0.0000*	0.7478 ±0.0000	0.9603 ±0.0000	0.0458 ± 0.0027	0.1775 ± 0.0013	0.1724 ± 0.0041
t-k-means++	*0.8546* ±*0.0000*	0.7478 ±0.0000	0.9603 ±0.0000	0.0460 ± 0.0013	0.1783 ±0.0008	0.1735 ± 0.0026
k-median	0.7566 ± 0.2670	0.6688 ± 0.2292	0.9190 ± 0.1259	0.0409 ± 0.0039	0.1729 ± 0.0048	0.1741 ± 0.0060
k-medoid	0.8410 ±0.0000	0.7412 ±0.0000	0.9588 ±0.0000	0.0446 ± 0.0017	0.1791 ± 0.0014	0.1778 ±0.0013
TMM	0.3120 ± 0.1852	0.3070 ± 0.1416	0.7684 ± 0.0776	0.0238 ± 0.0246	0.0862 ± 0.0821	*0.2029* ± *0.0209*
EAC	0.0072 ± 0.0214	0.0226 ± 0.0364	0.5542 ± 0.2921	0.0449 ± 0.0226	0.1577 ± 0.0573	0.2187 ± 0.0784
*k*ProtoClust	**0.8603 ± 0.0245**	**0.7747 ± 0.0413**	0.9580 ± 0.0122	**0.0665 ± 0.0038**	0.1766 ± 0.0090	**0.2243 ± 0.0119**
Methods/Dataset	WebKB			P2P Traffic		
	ARI	NMI	F1	ARI	NMI	F1
k-means	0.2766 ± 0.0427	0.4432 ± 0.0238	0.6802 ± 0.0350	0.3514 ± 0.0431	0.4925 ± 0.0503	0.7132 ± 0.0228
k-means++	*0.4581* ± *0.0313*	*0.4725* ± *0.0131*	0.7023 ± 0.0235	0.3605 ± 0.0549	0.4981 ± 0.0668	0.7290 ± 0.0645
Ball *k*-means	0.2766 ± 0.0427	0.4432 ± 0.0238	0.6802 ± 0.0350	0.3514 ± 0.0431	0.4925 ± 0.0503	0.7132 ± 0.0228
Deep *k*-means	0.0443 ± 0.0272	0.2235 ± 0.0891	0.3223 ± 0.0790	0.4260 ± 0.1211	0.5512 ± 0.0680	0.5683 ± 0.0670
CDKM	0.2816 ± 0.0245	0.4261 ± 0.0092	0.6532 ± 0.0167	0.4286 ± 0.0277	0.5796 ± 0.0510	0.7524 ± 0.0248
SMKM	**0.5616 ± 0.0854**	0.4496 ± 0.0436	*0.7177* ± *0.0207*	0.4355 ± 0.0181	0.5340 ± 0.0415	0.7668 ± 0.0271
t-k-means	0.2728 ±0.0000	0.4376 ±0.0000	0.6764 ±0.0000	0.4228 ± 0.0943	0.5640 ± 0.0651	0.7498 ± 0.0937
t-k-means++	0.2728 ±0.0000	0.4376 ±0.0000	0.6764 ±0.0000	0.4228 ± 0.0828	0.5640 ± 0.0644	0.7498 ± 0.0770
k-median	0.3705 ± 0.0815	0.4697 ± 0.0414	**0.7192 ± 0.0391**	0.3809 ± 0.1256	0.4862 ± 0.0550	0.7087 ± 0.0740
k-medoid	0.2643 ± 0.0245	0.4434 ± 0.0201	0.6673 ± 0.0018	0.4289 ±0.0000	0.5797 ±0.0000	0.7530 ±0.0000
TMM	0.2590 ± 0.0203	0.4150 ± 0.0150	0.6412 ± 0.0063	*0.7044* ± *0.0753*	*0.6486* ± *0.0894*	*0.8828* ± *0.0412*

(Continued)

Table 3 (continued)

Methods/Dataset	Chameleon			Glass		
	ARI	NMI	F1	ARI	NMI	F1
EAC	0.0001 ±0.0000	0.0058 ±0.0000	0.4387 ±0.0000	0.0598 ± 0.0597	0.1172 ± 0.0960	0.4875 ± 0.3365
*k*ProtoClust	0.3878 ± 0.0936	**0.4846 ± 0.0461**	0.6857 ± 0.0497	**0.7909 ± 0.0379**	**0.6702 ± 0.0322**	**0.9003 ± 0.0251**

Note: The first rank of each metric is highlighted by **boldface** and the second uses *italics* while the best std gets underline.

Consider data description in Table 2, we can get the following observations:

- Performances related to ARI, NMI, and F1 have similar trends for most cases. In terms of ARI, *k*ProtoClust reaches the first rank on Chameleon, wisconsin, abalone, and P2P Traffic while performing the second on glass and the third on WebKB. Similarly, *k*ProtoClust also reaches the best performance on four datasets in terms of NMI and on three data sets in terms of F1. Even though *k*ProtoClust is not ranked first, it frequently gets into the first three ranks or obtains comparable accuracies. Relatively, *k*ProtoClust has a significant advantage on datasets with small dimensions but more samples, e.g., Chameleon and P2P Traffic. Generally, in these cases, the more irregular the cluster shape is, the greater advantage it achieves.
- Besides *k*ProtoClust, SMKM and *k*-means++ perform better than the others. They separately have the best ARI on glass and WebKB. Compared with *k*-means, coincidentally, they both achieve improvement by introducing the initialization strategy of centroid selection. *k*-means++ selects K centroids far away from each other, whereas SMKM introduces a cheap split-merge step to re-start *k*-means after reaching a fixed point. Compared with *k*-means++, SMKM has advantages on all the other five datasets. What follows is CDKM, which achieves comparable accuracies with SMKM and *k*-means++ yet performs more stable with much lower mean-square deviations.
- Unfortunately, there are no obvious advantages for Deep *k*-means, *t*-*k*-means, *t*-*k*-means++, and TMM, even though they tend to build distribution models for these datasets with extremely irregular cluster shapes. Similar performance can be found in *k*-means, Ball *k*-means, *k*-median, and *k*-medoid. Like CDKM, *t*-*k*-means and *t*-*k*-means++ have significant stability advantages due to pursuing the global optimization of Eq. (1). However, limited by the circle-like pattern hypothesis, global optimization usually does not mean the best cluster description for irregular shapes and imbalance distribution. Among these methods, EAC outperforms most of the others on low-dimensional datasets with clear shapes, e.g., Chameleon. However, noises and higher dimensions frequently make it fail, e.g., glass, wisconsin, WebKB, and P2P Traffic.

To further confirm the effectiveness of *k*ProtoClust, we give results of pair comparisons in Table 4 following [37]. Taking *k*ProtoClust as the control method, a typical nonparametric statistical test of Friedman test is adopted to get the average ranks and unadjusted p values. By introducing the Bergmann-Hommel procedure [38], the adjusted p-value denoted by p_{Homm} corresponding to each comparison

is obtained. Obviously, *k*ProtoClust reaches the best performance. Since the Bergmann-Hommel procedure rejects those hypotheses with p-values ≤ 0.0167, together with the values of p_{Homm}, *k*ProtoClust performs better than SMKM, *k*-means++ and CDKM, and outperforms the others. The weakness is that several rounds of invoking *k*-means++ (i.e., lines 3 and 5 of Algorithm 4) may cause accumulative instability. Thus, *k*ProtoClust frequently has greater mean-square deviations than that achieved by *k*-means++ and CDKM.

Table 4: Comparison results under non-parametric statistical test

Methods	Average ranks	Unadjusted p	p_{Homm}
Control method: *k*ProtoClust, average rank = 1.5000			
k-means	7.7500	0.0054	0.0272
k-means++	5.0000	0.1195	0.2391
Ball *k*-means	7.9167	0.0043	0.0216
Deep *k*-means	9.3333	4.9422E-4	0.0054
CDKM	5.9167	0.0495	0.1485
SMKM	3.5000	0.3737	0.3737
t-*k*-means	6.9167	0.0160	0.0640
t-*k*-means++	7.7500	0.0054	0.0272
k-median	8.8333	0.0011	0.0102
k-medoid	7.6667	0.0061	0.0305
TMM	8.3333	0.0024	0.0142
EAC	10.5833	5.3495E-4	6.4194E-4

4.4 Exploration Contrast without the Prior Knowledge K_c

For many real-world problems, e.g., intrusion behavior analysis, data owners can not have the prior knowledge K_c. Due to polymorphism, the expected class number K_c (column 5 in Table 2) is sometimes not the same as the real cluster number, which generally corresponds to the actual distribution pattern. Therefore, we conduct experiments to check the cluster discovery ability of *k*ProtoClust on datasets of Type I without the prior knowledge K_c. Based on results in Tables 3 and 4, we take *k*-means, SMKM, *k*-means++, *t*-*k*-means++, CDKM, and *k*-median as baselines. On datasets of type I, the reserved methods either reach close accuracies with *k*ProtoClust or have significant stability advantages. Fig. 5 shows the accuracies achieved by all the compared methods and the obtained cluster numbers K_f with respect to different initialized K by *k*ProtoClust.

Intuitively, without enough attention to the actual distribution, *k*-means, *k*-means++, and *k*-median try to do data division under different initialization strategies or distance measures. Although CDKM researches instability reduction by perusing the global optimization, *t*-*k*-means++ pays more attention to data distribution by introducing *t*-mixture distribution model, and SMKM adopts split-merge strategy to adjust clusters, they have to ensure the input space is divided into K clusters. Therefore, they can not avoid a severe deviation from the truth cluster number if we get an incorrect assumption of K. In contrast, the objective of split and merge analysis in *k*ProtoClust is to dynamically generate an unfixed number of convex hulls of different sizes. Considering these convex hulls as prototypes, *k*ProtoClust can better profile clusters with arbitrary shapes. Even though the discovered cluster number may not be equal to K_c, it is located around the expected value in a relatively small error range on Chameleon, glass, and P2P Traffic. In general, data

separability reduces for high dimensional data with inadequate samples. Therefore, the discovered cluster numbers on wisconsin and abalone show increasing trends when K becomes much greater than the actual number. Even so, these deviations are still much smaller than those discovered by the other variants.

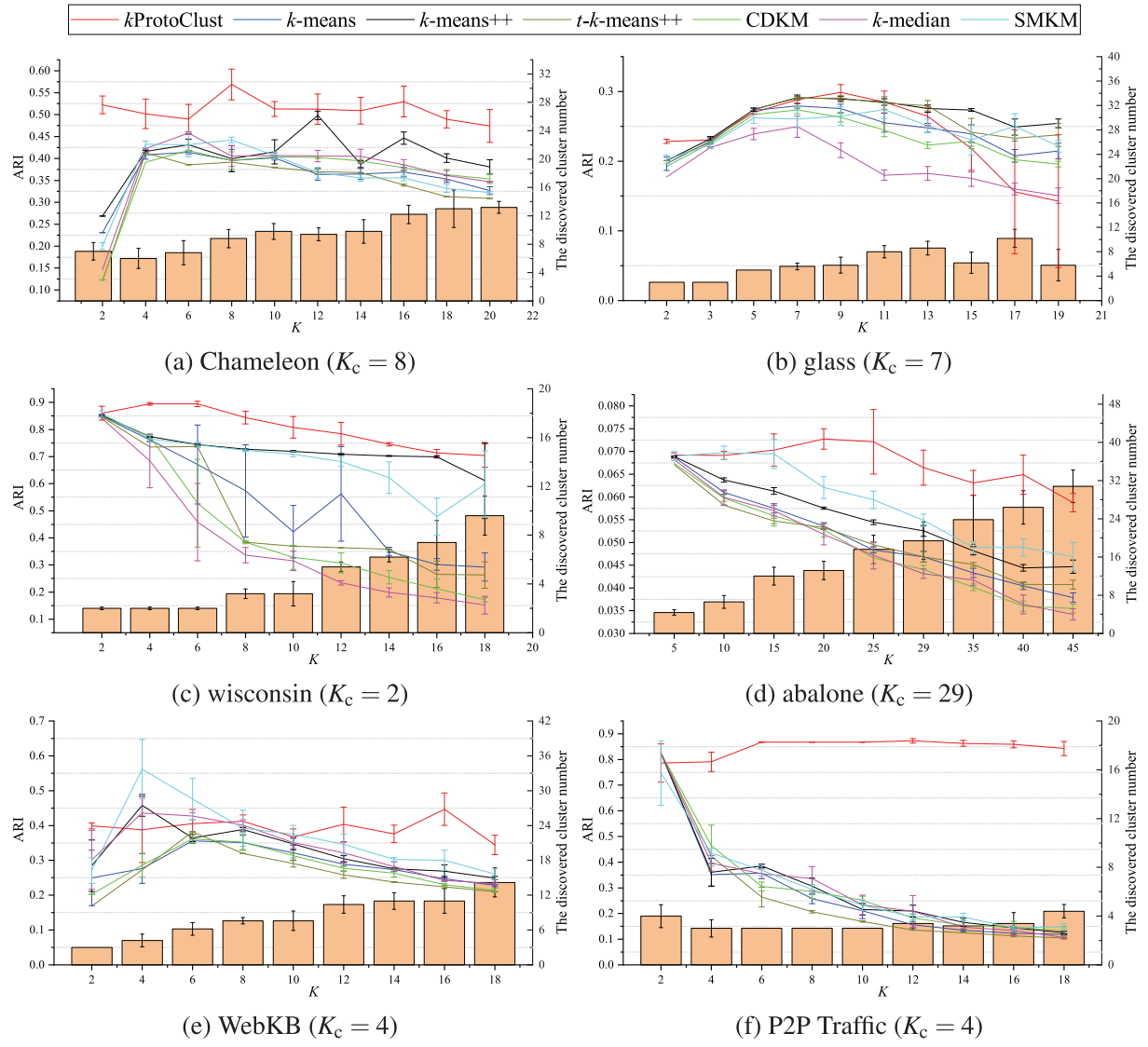


Figure 5: Accuracy (ARI) for all the compared methods and the cluster number found by kProtoClust without K_c

Regarding ARI, the obtained accuracies show evidence of imbalanced distribution and irregular cluster shapes. For instance, accuracies obtained on Chameleon by the baseline methods are not falling off a cliff along with the increase of K, and the best accuracy on P2P Traffic is achieved when $K = 2$, not $K = K_c$ (i.e., 4). Besides k-means++ reaches the best accuracy with $K = 12$ on Chameleon while $K_c = 8$, the baselines share a common characteristic of a downtrend as K increases, especially when K is far from the ground truth. Fortunately, we can find that kProtoClust has remarkable performances in stability which outperform the others for most cases, e.g., Chameleon, wisconsin, abalone, WebKB, and P2P Traffic. k-means++ and SMKM have comparable performances which rank only second to kProtoClust, whereas CDKM and t-k-means++

have similar performance. Although k-means does cluster analysis well, it frequently gets greater std than k-median. Therefore, we believe that kProtoClust outperforms the others when we can not have the actual cluster number as the prior knowledge.

Additionally, several other cues may be found from Fig. 5. First, for most cases, performances of baselines have specific decreases in accuracy when K is far from K_c. Similar trends happened to kProtoClust on glass and wisconsin. It means that a small cluster number assumed by k-means++ (line 3 of Algorithm 4) is unsuitable for convex decomposition towards prototype finding. Although the discovered cluster number by kProtoClust is frequently close to the ground truth. One can try several times and then avoid making K fall into a range where the discovered cluster numbers have large fluctuations. However, an appropriate assessment method for accurate cluster numbers may benefit from finding an optimal K for all the methods. Second, although the cluster discovery ability of kProtoClust on datasets of type I has been confirmed, too few samples in a dataset is also a challenge for kProtoClust, e.g., the 9-dimensional dataset glass with only 214 samples in 7 clusters. For a dataset like this particular case, CDKM, k-means, and k-means++ are recommended for simplicity if we get a K value close to the ground truth.

4.5 Necessity Analysis of NMMRS for kProtoClust

Based on the analysis in Sections 3.5 and 4.4, when we fix the data size N, a greater dimensionality d not only increases its complexity but also reduces the separability. However, if d is fixed, the increase of N increases the complexity but will not permanently change (either increase or reduce) the separability. Therefore, we introduce NMMRS (lines 1–2 of Algorithm 4) into kProtoClust and denote it by kProtoClust(SR) in the following necessity analysis.

As discussed in Section 3.5, EdgeSel consumes $O(N^2)$ which may be the most time-consumption phase in kProtoClust. By employing the "Run and Time" analysis provided by MATLAB, we separately list all the time-consumptions required by each phase of kProtoClust and kProtoClust(SR) on Chameleon in Table 5. Obviously, the first two significant time-consuming phases are MergeAnalysis and EdgeSel which occupy 54.95% and 30.27% of kProtoClust, respectively. Furthermore, EdgeSel is one of the main phases invoked by MergeAnalysis in line 2 of Algorithm 2. In kProtoClust(SR), their time-consumptions are separately reduced to 43.52% and 9.32%, and the total occupation is reduced to 52.84% from 85.22%. Relatively, LabelAssignment has been increased to 26.75% from 5.81%. Unlike experiments in Section 4.3, we restrict the max iteration of k-means++ to 10 for all the invocations (i.e., lines 3 and 5), and do not consider any parallel strategy for each phase.

kProtoClust(SR) is a variant of k-means++ with split and mergence strategy. Meanwhile, SMKM also prefers a split-merge strategy and performs closely to kProtoClust(SR) in terms of accuracy on datasets of type I. In this section, we take k-means++ and SMKM as the baselines for further comparative analysis with kProtoClust and kProtoClust(SR). Results of accuracies in terms of ARI are depicted in Fig. 6 in which run-time costs required by kProtoClust and kProtoClust(SR) are also given. Due to $dK_c > N$, NMMRS is employed by kProtoClust(SR) to make dimensionality reduction for movement_libras and uspst, while data reduction is preferred for the other four datasets since $dK_c \ll N$. As shown in Fig. 6, kProtoClust(SR) and kProtoClust consistently achieve the first two ranks of accuracy except for uspst. Obviously, the advantage usually rises along with N increases, e.g., results corresponding to kddcup99. On uspst, kProtoClust and k-means++ have comparable results while kProtoClust(SR) and SMKM perform similarly. Furthermore, a greater d frequently influences the std of accuracy for kProtoClust(SR) and SMKM, particularly for SMKM. It means that dimensionality reduction may not be the best choice if the dimensionality d of a dataset is high and $dK_c > N$. Because the ability of structure kept of NMMRS reduces significantly if we want to do distance-based analysis among data samples more than the nearest neighbors.

Table 5: Time-consumption analysis of algorithms in *k*ProtoClust

Algorithm	*k*ProtoClust		*k*ProtoClust(SR)	
	Run-time (s)	Ratio (%)	Run-time (s)	Ratio (%)
NMMRS (lines 1–2)	—	—	0.016	1.49%
k-means++ (line 3)	0.209	4.60%	0.161	15%
EdgeSel (line 4)	1.375	30.27%	0.100	9.32%
SplitAnalysis (line 5)	0.198	4.36%	0.042	3.91%
MergeAnalysis (line 6)	2.496	54.95%	0.467	43.52%
LabelAssignment (line 7)	0.264	5.81%	0.287	26.75%

Note: *k*-means++ restricts the max iterations to 10 for all the invocations.

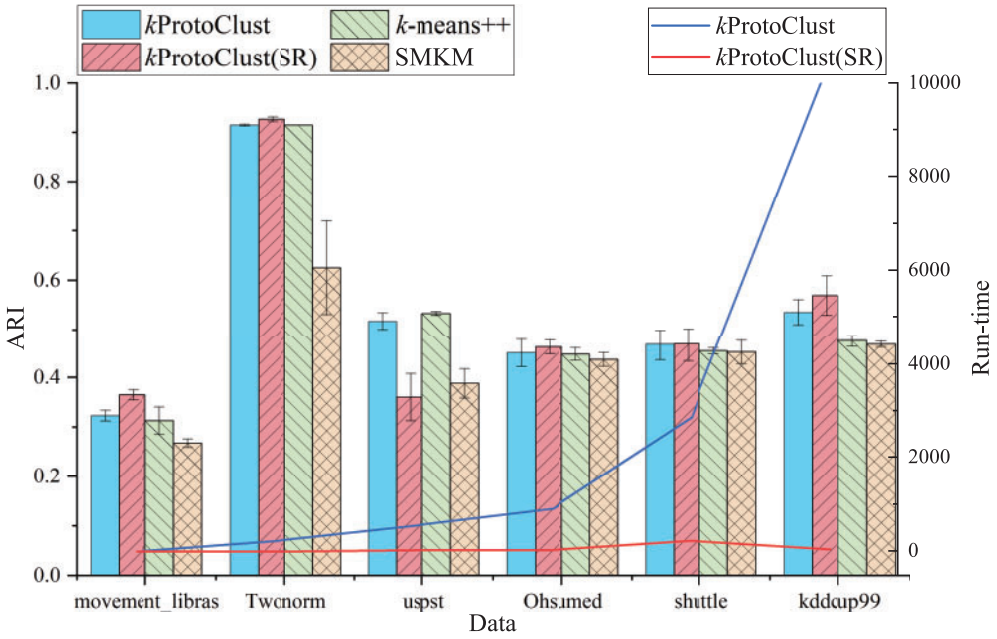


Figure 6: Performance analysis for *k*ProtoClust with or without sampling and dimensionality reduction by taking *k*-means++ and SMKM as baselines. $N_s = N$ and $q = 10 \ll d$ are set for movement_libras and uspst while $N_s = 5000 \ll N$ and $q = d$ are preferred for the other datasets

In terms of efficiency, *k*ProtoClust(SR) has a significant advantage over *k*ProtoClust due to the introduction of NMMRS. On Twonorm, Ohsumed, shuttle and kddcup99, *k*ProtoClust(SR) finish cluster analysis in 2.5, 40.7, 233.5 and 49.7 s, respectively. The run-time required by shuttle is greater than kddcup99 because the iterative analysis is more complex in SplitAnalysis (line 5 of Algorithm 4) due to extremely imbalanced distribution. Furthermore, results corresponding to all the datasets of type II confirm the contributions from NMMRS and suggest that the reduced dataset should have $N > dK_c$ or keep $N_s > qK_c$ to avoid affecting data separability.

5 Related Works

k-means is the inspiration source of *k*ProtoClust. Due to some intrinsic flaws inherited from the partition-based clustering strategy, the literature mainly focuses on further efficiency improvements, initialization strategy, stability optimization, and specific data types and patterns support.

The classical *k*-means selects *K* samples at random as the initial centroids and iteratively conducts the assignment and refinement until all the centroids do not change. For efficiency, besides hardware parallelization, fast convergence [7], result approximation [39], and distance computation reduction [14] are mainly focused. The last one receives the most attention for its generalization ability of computation optimization. Those representative works include avoiding unnecessary membership analysis by storing the different number of bounds for either centroids or data samples [40], and introducing group pruning and centroids regrouping strategy in the iteration [41]. For efficiency, Ryšavý et al. [42] makes a tighter centroid drift bound by employing the distance between the origin sample and the corresponding centroid, Bottesch et al. [43] adopts the Hölder's inequality and norms, while Broder et al. [44] introduce a pre-assignment search around each centroid which greatly reduces distance computations. Recently, to optimize cluster assignment, Ball *k*-means [2] utilizes centroid distance. Besides, *k*-median [26] and *k*-medoid [27] can also be considered variants of *k*-means since the former introduces the Manhattan distance while the latter chooses existing data samples as centroids.

Many valuable solutions [45] corresponding to the initialization strategy, the main works are to estimate an appropriate *K* for centroids initialization. For the former, the mainstream way is parameter tweaking or evaluating models with different *K* settings [46] before finding out the usable *K*. Meanwhile, combining multiple clustering results with some voting strategies like EAC [5] or introducing some extensions of criterion such as Elbow index [47] are also considered. For the latter, besides pre-analysis of a set of potential centroids, Peña et al. [48] suggest that dividing a data set into *K* clusters and iteratively collecting *K* representative instances are practical. Furthermore, *k*-means++ [7] is a representative adopting the latter strategy since it chooses *K* centroids far away from each other.

Stability optimization is strongly related to the initialization strategy because the random initialization increases the instability. Recently, Capó et al. [9] present a split-merge strategy to restart *k*-means after reaching a fixed point. Therefore, their variant of *k*-means, namely SMKM, improves stability. However, multiple rounds of invocations to *k*-means and the split-merge strategy make it more suitable for dealing with clusters with irregular shapes. Another typical work in the most recently is CDKM [24], which designs a coordinate descent method for *k*-means to reach the global optimization of Eq. (1).

Different data types frequently relate to different application domains and distinct data patterns. In the literature, most works prefer doing special designs of *k*-means for a specific application. For instance, privacy-preserving *k*-means is presented for encrypted data analysis [49], global *k*-means based neural network performs well on sound source localization [50], while *k*-means with bagging neural network is suitable for short-term wind power forecasting [51]. Consider those mixed data types, the major attentions prefer fuzzy method [52], distance calculation mechanism [45], and hybrid framework [14]. Towards challenges from arbitrary cluster shapes, Li et al. [25] present *t*-*k*-means and *t*-*k*-means++ by assuming data sampled from the *t*-mixture distribution. Besides, Khan et al. [53] consider an exponential distribution to standardize data and propose EGS to estimate the optimal number of clusters. Fard et al. [23] present a joint solution namely deep *k*-means by integrating *k*-means and deep learning, while Peng et al. [54] adopt deep learning to re-describe *k*-means. Given the cluster number *K*, another exciting work [55] introduces *k*-means into SVDD such that it supports *K* groups of submodel of SVDD to describe different clusters in a data set. Lacking the prior knowledge of cluster number, EAC [5] clusters data with *k*-means based evidence accumulation. It

conducts result combination with a specific vote strategy that well support low-dimensional data with clear and simple shape.

Despite many insightful works, they treat clusters fairly in size due to the limitation of the distance-based data partition. Few works change the initialized K in the clustering procedure. However, arbitrarily shaped clusters are very common in practice which motivates us to break the restriction of inaccurate prior-knowledge K and partition data into clusters with different sizes and connection relationships. Even though an optimal K is estimated by [53], utilizing K fixed clusters and K data samples as prototypes for label assignments can not match the human's cognition of discovering clusters from their shapes, sizes, and connection relationships well. Therefore, *k*ProtoClust takes convex hulls of varying sizes as prototypes following [17] and allows the change of K to support multiple prototypes in one cluster to improve cluster description ability.

6 Conclusion

With a specific distance measure, k-means find K data samples as prototypes to partition data space into clusters. However, not only the cluster number is unknown, but many practical datasets do not have a balanced and regular distribution as what k-means expected. To discover clusters following a general human cognition (corresponding to $\{\gamma_l, \gamma_u, \tau_d, \tau_{\mathrm{m}}, \tau_{\mathrm{im}}\}$), we propose *k*ProtoClust to replace prototypes with convex hulls of varying sizes. Given K selected at random, k-means++ divides the input space into K clusters. Regarding the local geometry view, edge patterns of each cluster are collected and split into subgroups if they can not construct a single convex hull. In terms of the global geometry view, all the violating samples existing in the overlapping region of two adjacent convex hulls are located. Since violating samples are not the actual edge patterns, removing violating samples means the mergence of the associated convex hulls increases the number of prototypes for the corresponding cluster. Finally, one-cluster-multiple-prototypes are supported by *k*ProtoClust in which the adaptive prototype size contributes to the ability of arbitrary cluster shape's description. As the composite indicators for a specific human cognition, the five additional thresholds are relatively insensitive to the clustering result. They can be preset fixedly following either the recommendations in Section 4.2 or user preferences.

Although *k*ProtoClust achieves accuracy improvement with fair efficiency on datasets of imbalanced distribution and irregular cluster shapes, some shortcomings still exist, e.g., a certain instability inherited from k-means++. Meanwhile, the ways of handling high dimensional data with low separability and making the cluster number assessment more accurate are worthy of further investigation, as well as further improving the generalizability and adaptability across different scenarios, including semi-supervised clustering and clustering data streams.

Acknowledgment: We thank all the authors for their research contributions.

Funding Statement: This work is supported by the National Natural Science Foundation of China under Grant No. 62162009, the Key Technologies R&D Program of He'nan Province under Grant No. 242102211065, the Scientific Research Innovation Team of Xuchang University under Grant No. 2022CXTD003, and Postgraduate Education Reform and Quality Improvement Project of Henan Province under Grant No. YJS2024JD38.

Author Contributions: Conceptualization, methodology, and draft manuscript preparation: Yuan Ping; Huina Li; data collection, visualization, and analysis: Yuan Ping; Chun Guo; Bin Hao. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available on request from the authors.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Kansal T, Bahuguna S, Singh V, Choudhury T. Customer segmentation using k-means clustering. In: 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS); 2018; Belgaum, India. p. 135–9.
2. Xia S, Peng D, Meng D, Zhang C, Wang G, Giem E, et al. Ball *k*-means: fast adaptive clustering with no bounds. IEEE Trans Pattern Anal Mach Intell. 2022;44(1):87–99. doi:10.1109/TPAMI.2020.3008694.
3. Banerji A. *k*-mean: getting the optimal number of clusters; 2022 [cited 2024 Oct 20]. Available from: https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/.
4. Khan IK, Daud HB, Zainuddin NB, Sokkalingam R, Abdussamad, Museeb A, et al. Addressing limitations of the k-means clustering algorithm: outliers, non-spherical data, and optimal cluster selection. AIMS Math. 2024;9(9):25070–97. doi:10.3934/math.20241222.
5. Fred ALN, Jain AK. Data clustering using evidence accumulation. In: Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02); 2002; Quebec City, QC, Canada. p. 276–80.
6. Celebi ME, Kingravi HA, Vela PA. A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst Appl. 2013;40(1):200–10. doi:10.1016/j.eswa.2012.07.021.
7. Arthur D, Vassilvitskii S. *k*-means++: The advantages of careful seeding. In: Proceedings of The Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07); 2007; New Orleans, LA, USA. p. 1027–35.
8. Fränti P, Sieranoja S. How much can *k*-means be improved by using better initialization and repeats? Pattern Recognit. 2019;93(9):95–112. doi:10.1016/j.patcog.2019.04.014.
9. Capó M, Pérez A, Lozano JA. An efficient split-merge re-start for the *k*-means algorithm. IEEE Trans Knowl Data Eng. 2022;34(4):1618–27. doi:10.1109/TKDE.2020.3002926.
10. Lu Y, Cheung Y-M, Tang YY. Self-adaptive multiprototype-based competitive learning approach: a *k*-means-type algorithm for imbalanced data clustering. IEEE Trans Cybern. 2021;51(3):1598–612. doi:10.1109/TCYB.2019.2916196.
11. Yao Y, Li Y, Jiang B, Chen H. Multiple kernel *k*-means clustering by selecting representative kernels. IEEE Trans Neural Netw Learn Syst. 2021;32(11):4983–96. doi:10.48550/arXiv.1811.00264.
12. Ping Y, Hao B, Li H, Lai Y, Guo C, Ma H, et al. Efficient training support vector clustering with appropriate boundary information. IEEE Access. 2019;7(10):146964–78. doi:10.1109/ACCESS.2019.2945926.
13. Rathore P, Kumar D, Bezdek JC, Rajasegarar S, Palaniswami M. A rapid hybrid clustering algorithm for large volumes of high dimensional data. IEEE Trans Knowl Data Eng. 2019;31(4):641–54. doi:10.1109/TKDE.2018.2842191.
14. Wang S, Sun Y, Bao Z. On the efficiency of *k*-means clustering: evaluation, optimization, and algorithm selection. Proceedings VLDB Endowment. 2020;14(2):163–75. doi:10.48550/arXiv.2010.06654.
15. Tax DMJ, Duin RPW. Support vector domain description. Pattern Recognit Lett. 1999;20:1191–9. doi:10.1023/B:MACH.0000008084.60811.49.
16. Li YH, Maguire L. Selecting critical patterns based on local geometrical and statistical information. IEEE Trans Pattern Anal Mach Intell. 2011;33(6):1189–201. doi:10.1109/TPAMI.2010.188.
17. Ping Y, Tian Y, Zhou Y, Yang Y. Convex decomposition based cluster labeling method for support vector clustering. J Comput Sci Technol. 2012;27(2):428–42. doi:10.1007/s11390-012-1232-1.
18. Li H, Ping Y, Hao B, Guo C, Liu Y. Improved boundary support vector clustering with self-adaption support. Electronics. 2022;11(12):1854. doi:10.3390/electronics11121854.
19. Karypis G, Han E-H, Kumar V. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. Computer. 1999;32(8):68–75. doi:10.1109/2.781637.
20. Liparulo L, Proietti A, Panella M. Fuzzy clustering using the convex hull as geometrical model. Adv Fuzzy Syst. 2015;2015(1):265135. doi:10.1155/2015/265135.
21. Pan C, Rana JSSPV, Milenkovic O. Machine unlearning of federated clusters. arXiv:2210.16424. 2022.

22. Johnson WB, Lindenstrauss J. Extensions of lipschitz mappings into a hilbert space. Contemp Math. 1984;26(1):189–206. doi:10.1090/conm/026/737400.
23. Fard MM, Thonet T, Gaussier E. Deep *k*-means: jointly clustering with *k*-means and learning representations. Pattern Recognit Lett. 2020;138(10):185–92. doi:10.1016/j.patrec.2020.07.028.
24. Nie D, Xue J, Wu D, Wang R, Li H, Li X. Coordinate descent method for *k*-means. IEEE Trans Pattern Anal Mach Intell. 2022;44(5):2371–85. doi:10.1109/TPAMI.2021.3085739.
25. Li Y, Zhang Y, Tang Q, Huang W, Jiang Y, Xia S-T. *t*-*k*-means: a robust and stable *k*-means variant. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2021; Toronto, ON, Canada. p. 3120–4.
26. Arora S, Raghavan P, Rao S. Approximation schemes for euclidean *k*-medians and related problems. In: Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC' 98); 1998; Dallas, TX, USA. p. 106–13.
27. Kaufman L, Rousseeuw P. Clustering by means of meoids. North-Holland: Faculty of Mathematics and Informatics; 1987.
28. Frank A, Asuncion A. UCI machine learning repository. Irvine: University of California, School of Information and Computer Sciences; 2010 [cited 2024 Oct 20]. Available from: http://archive.ics.uci.edu/ml.
29. Graven M, DiPasquo D, Freitag D, McCallum A, Mitchell T, Nigam K, et al. Learning to extract symbolic knowledge form the world wide web. In: Proceedings of 15th Nat'l Conference for Artificial Intelligence (AAAI'98); 1998; Madison, WI, USA. p. 509–16.
30. Hersh WR, Buckley C, Leone TJ, Hickam DH. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: Proceedings of 17th Annual ACM SIGIR Conference; 1994; Dublin, Ireland. p. 192–201.
31. Peng J, Zhou Y, Wang C, Yang Y, Ping Y. Early tcp traffic classification. J Appl Sci. 2011;29(1):73–7. doi:10.1016/j.dcan.2022.09.009.
32. Department of Information and Computer Science, University of California. KDD cup 99 intrusion detection dataset. Irvine: Department of Information and Computer Science, University of California; 1999 [cited 2024 Oct 20]. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
33. Xu R, Wunsch DC. Clustering. Hoboken, New Jersey: A John Wiley&Sons; 2008.
34. Strehl A, Ghosh J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J Mach Learn Res. 2002;3(12):583–617. doi:10.1162/153244303321897735.
35. Sasaki Y. The truth of the f-measure; 2007 [cited 2024 Oct 20]. Available from: https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf.
36. Ping Y, Zhou Y, Yang Y. A novel scheme for accelerating support vector clustering. Comput Inform. 2012;31(3):1001–26.
37. Garcia S, Herrera F. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J Mach Learn Res. 2008;9(Dec):2677–94.
38. Bergmann G, Hommel G. Improvements of general multiple test procedures for redundant systems of hypotheses. Mult Hypotheses Test. 1988;70:100–15.
39. Newling J, Fleuret F. Nested mini-batch *k*-means. In: Proceedings of the 13th Annual Conference on Neural Information ProcessingSystems (NIPS'16); 2016; Barcelona, Spain. p. 1352–60.
40. Newling J, Fleuret F. Fast *k*-means with accurate bounds. In: Proceedings of the 33rd International Conference on Machine Learning (ICML '16); 2016; New York City, NY, USA. p. 936–44.
41. Kwedlo W. Two modifications of yinyang *k*-means algorithm. In: Proceedings of 16th International Conference on Artificial Intelligence and Soft Computing (ICAISC '17); 2017; Zakopane, Poland. p. 94–103.
42. Ryšavý P, Hamerly G. Geometric methods to accelerate *k*-means algorithms. In: Proceedings of the 2016 SIAM International Conference on Data Mining (SDM '16); 2016; Miami, FL, USA. p. 324–32.
43. Bottesch T, Bühler T, Kächele M. Speeding up *k*-means by approximating euclidean distances via block vectors. In: Proceedings of the 33rd International Conference on Machine Learning (ICML '16); 2016. p. 2578–86.

44. Broder A, Garcia-Pueyo L, Josifovski V, Vassilvitskii S, Venkatesan S. Scalable *k*-means by ranked retrieval. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14); 2014; New York City, NewYork, USA. p. 233–42.
45. Ahmed M, Seraj R, Islam SMS. The *k*-means algorithm: a comprehensive survey and performance evaluation. Electronics. 2020;9(8):1295:1–12. doi:10.3390/electronics9081295.
46. Ikotun AM, Ezugwu AE, Abualigah L, Abuhaija B, Heming J. K-means clustering algorithms: a comprehensive review, variants analysis, and advances in the era of big data. Inf Sci. 2023;622:178–210. doi:10.1016/j.ins.2022.11.139.
47. Rykov A, De Amorim RC, Makarenkov V, Mirkin B. Inertia-based indices to determine the number of clusters in k-means: an experimental evaluation. IEEE Access. 2024;12:11761–73. doi:10.1109/ACCESS.2024.3350791.
48. Peña JM, Lozano JA, Larrañaga PL. An empirical comparison of four initialization methods for the k-means algorithm. Pattern Recognit Lett. 1999;20(10):1027–40. doi:10.1016/S0167-8655(99)00069-0.
49. Yuan J, Tian Y. Practical privacy-preserving mapreduce based *k*-means clustering over large-scale dataset. IEEE Trans Cloud Comput. 2019;7(2):568–79. doi:10.1109/TCC.2017.2656895.
50. Yang X, Li Y, Sun Y, Long T, Sarkar TK. Fast and robust RBF neural network based on global *k*-means clustering with adaptive selection radius for sound source angle estimation. IEEE Trans Antennas Propag. 2018;66(6):3097–107. doi:10.1109/TAP.2018.2823713.
51. Wu W, Peng M. A data mining approach combining *k*-means clustering with bagging neural network for short-term wind power forecasting. IEEE Internet Things J. 2017;4(4):979–86. doi:10.1109/JIOT.2017.2677578.
52. Nie F, Zhao X, Wang R, Li X, Li Z. Fuzzy *k*-means clustering with discriminative embedding. IEEE Trans Knowl Data Eng. 2022;34(3):1221–30. doi:10.1109/TKDE.2020.2995748.
53. Khan IK, Daud HB, Zainuddin NB, Sokkalingam R, Farooq M, Baig ME, et al. Determining the optimal number of clusters by enhanced gap statistic in k-mean algorithm. Egypt Inform J. 2024;27(1):100504. doi:10.1016/j.eij.2024.100504.
54. Peng X, Li Y, Tsang IW, Zhu H, Lv J, Zhou JT. Xai beyond classification: interpretable neural clustering. J Mach Learn Res. 2022;23(6):1–28. doi:10.48550/arXiv.1808.07292.
55. Gornitz N, Lima LA, Muller K-R, Kloft M, Nakajima S. Supprt vector data descriptions and k-means clustering: one class? IEEE Trans Neural Netw Learn Syst. 2018;29(9):3994–4006. doi:10.1109/TNNLS.2017.2737941.