Information Systems 117 (2023) 102212

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Kernel-based clustering via Isolation Distributional Kernel

Ye Zhu [a], Kai Ming Ting [b,*]

[a] School of Information Technology, Deakin University, Victoria, 3125, Australia
[b] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China

ARTICLE INFO

Article history:
Received 29 January 2023
Received in revised form 16 March 2023
Accepted 11 April 2023
Available online 23 April 2023
Recommended by Dennis sasha

Dataset link: http://cs.joensuu.fi/sipu/datasets/, https://unsplash.com/photos/9FvTOCfvt_w, https://github.com/Yunfan-Li/Contrastive-Clustering

Keywords:
Kernel-based clustering
Expectation-and-Maximization
Isolation kernel

ABSTRACT

Clustering has become one of the widely used automatic data-labeling techniques applied in a variety of disciplines. Kernel-based clustering is a technique designed to identify non-linearly separable clusters with irregular shapes. However, existing kernel-based clustering algorithms usually produce weaker clustering outcomes than density-based clustering, and they have high computational cost which limits their applications to small datasets only. In this paper, we contend that these limitations are mainly due to the use of (a) the Expectation-and-Maximization algorithm as an optimization procedure, and (b) a non-adaptive kernel. In addition, to address the limitations of current kernel-based algorithms, we propose the first clustering algorithm that employs an adaptive distributional kernel without any optimization, while achieving a similar optimization objective function. We demonstrate its superior performance of identifying complex clusters on massive datasets under different real-world application scenarios.

1. Introduction

We live in the age of information explosion and collect more data than ever before to make data-driven decisions. However, it is impossible for humans to manually label and understand massive datasets. Clustering, as a crucial unsupervised knowledge discovery technique, has become one of the most useful automatic data-labeling techniques, widely studied in a large variety of disciplines, such as economics [1], health [2] and engineering [3]. The applications cover image and speech processing [4], gene pattern extraction [5], community detection [6] and market segmentation [7]. Clustering aims to automatically partition a group of data points into a set of homogeneous groups based on their similarity in order to discover "natural" hidden patterns and data structures, with little or no prior information [8].

Clustering algorithms have been developed to detect various cluster structures, based on different purposes or views [9]. Kernel-based clustering is a technique designed to identify non-linearly separable clusters with irregular shapes [10].

Existing kernel-based clustering algorithms differ in aims and motivations. Kernel K-means [10–12] is intended to improve the clustering capability of K-means [13], as K-means is limited to producing globular-shaped clusters. Mean-shift [14] is a mode-seeking algorithm that attempts to find the maxima of a density function estimated via a kernel density estimator. To produce exactly k clusters, K-Modes clustering [15] is developed from mean-shift. Laplacian K-Modes [16,17] is the latest version of K-Modes, targeted at finding k modes with a graph Laplacian regularizer to improve the clustering outcomes of K-Modes. The common denominator of the above-mentioned clustering methods is the use of the Expectation-and-Maximization (EM) procedure [18][1] to optimize a clustering objective function.

The frequently stated problems with these algorithms include poor clustering outcomes and high computational costs. For example, kernel K-means (e.g.,[10,12]), though ameliorates the clustering outcome of K-means (a non-kernel-based clustering), is still overshadowed by density-based clustering (e.g., [20]). In addition, kernel K-means runs significantly slower than K-means on large datasets.

How to improve the clustering performance at a low computational cost with a kernel-based method has raised our concern. Therefore, this paper seeks to answer the following questions:

(I) What are the root causes of poor clustering performance or/and high computational cost in existing kernel-based clustering algorithms?
(II) Is there a better way than using an EM procedure to achieve the same objective function?

While investigating the first question, we discover that existing kernel-based clustering algorithms limit the kind of clusters

* Corresponding author.
E-mail addresses: ye.zhu@ieee.org (Y. Zhu), tingkm@nju.edu.cn (K.M. Ting).

[1] Mean-shift has been shown to be an EM-like procedure. See [19] for details.

https://doi.org/10.1016/j.is.2023.102212
0306-4379/

that can be detected. Then we identify that the EM procedure and the employed point-to-point kernel are the root causes of poor clustering results and/or high computational cost. EM has two main steps: The E-step determines the average point representative of all points in each cluster, decided in the previous iteration; and the M-step reassigns every point in the given dataset to the closest average point representative of a cluster. The set of k average points is critical to the clusters formation in parallel in the M-step of the iteration process; and they limit the type of clusters which can be discovered to those in the Voronoi diagram in the input space. This is the root cause of poor clustering outcome of an EM-based clustering. In addition, the type of kernel used has a direct impact on a clustering algorithm's ability to find clusters of varied densities, even on density-based clustering which has the ability to discover clusters of arbitrary shapes [21]. Further details can be found in Section 3.4.

Aiming to eliminate the root causes, we introduce a new kernel-based clustering algorithm called IDKC that is based on Isolation Distributional Kernel (IDK) [22]. It is motivated to discover clusters of arbitrary shapes, varied densities and sizes. To avoid using the set of representative points as a whole for clusters formation, IDKC uses a distribution to represent each cluster, and each cluster is grown independently of each other by recruiting neighboring points which are most similar to a distribution representing a cluster. As a result, the clusters discovered can be clusters of any shapes, sizes and densities, represented as distributions. The distributions can be estimated efficiently using a distributional kernel without density estimation or mixture modeling (which are one of the root causes of high computational cost in existing clustering algorithms).

IDKC is the first kernel-based clustering algorithm that grows k clusters from k cluster seeds, based on a distributional kernel. It achieves the same objective function of existing K-Modes clustering algorithms without using the EM procedure, i.e., without relying on the iterative two-steps: k centroids determination and reassigning all points in the dataset based on the k centroids. Unlike existing kernel-based clustering algorithms that detect the limited kind of clusters with Voronoi diagram only (or its relaxation), IDKC is able to discover clusters of arbitrary shapes and sizes. Moreover, IDKC does not rely on random initialization and is not sensitive to outliers, because both the cluster seeds selection and growing a cluster are based on a distributional kernel. We also demonstrate that IDKC is the only kernel-based clustering algorithm that can deal with large-scale datasets having millions of data points.

Among the existing kernel-based clustering algorithms that use the EM optimization, we find that Laplacian K-Modes [16] produces the best clustering outcomes. With the use of a graph Laplacian regularizer, Laplacian K-Modes achieves better clustering outcomes than K-Modes. Also, the graph Laplacian regularizer enables Laplacian K-Modes to discover clusters that are mutants of Voronoi diagram.[2] However, the identified cluster types are still constrained due to the use of EM. Here we show that IDKC yields better clustering outcomes than Laplacian K-Modes; and its superiority has two key contributing factors: the proposed clustering algorithm and the kernel employed, i.e., IDK, has data dependent property. We also show that using a data independent distributional kernel significantly weakens the clustering outcomes of the proposed clustering algorithm.

IDKC has four key advantages over existing kernel-based clustering algorithms that employ the EM optimization because it

(i) Achieves the same objective function of existing K-Modes clustering algorithms without using the EM procedure;
(ii) Is able to discover clusters of arbitrary shapes and sizes;
(iii) Does not rely on random initialization and is not sensitive to outliers;
(iv) Has higher efficiency without sacrificing effectiveness.

The remaining part of the paper proceeds as follows. Section 2 describes the related work. Section 3 presents the methodology and the proposed IDKC algorithm. Section 4 empirically evaluates the performance of IDKC under different real-world application scenarios. Conclusion is provided in the last section.

2. Related work

2.1. EM-based clustering algorithms

The EM procedure used in the above-mentioned kernel-based clustering algorithms has two key steps, i.e., (i) update k centroids based on the points assigned to each of the k clusters, and (ii) reassign all points based on the k centroids. It has the following known issues:

(i) Random initialization is the cause of clustering outcomes that are counter-intuitive and noticeably different/unstable from one another, even when the number of clusters is specified correctly.
(ii) The clusters it can discover are limited to cells in a Voronoi diagram in the feature space of the kernel employed. Though these clusters may be non-globular shapes in the input space, they are not arbitrary shapes. The only exception is Laplacian K-Modes.
(iii) The use of centroids makes the clustering outcomes sensitive to outliers in the dataset.
(iv) Most algorithms of kernel-based clustering have high computational cost; and the efficient versions often trade off accuracy for less runtime.

Although extensive research on how to remove cluster structure limitation has been carried out, (e.g., using kNN kernel [10], using modes instead means [15,16], and using power K-means [24]), the issue of high-computational cost is not sufficiently addressed. Kernel functional approximation [12] and relaxation of the objection function [17] were introduced to lower the computational time. However, they demand a special hardware for parallelization to lower the cost. Yet, they rarely resolve the time complexity issue and parallelization has a diminishing payoff as the data size increases.

The EM-based kernel-based clustering has two main categories, i.e., kernel K-means and K-Modes.

Kernel K-means is pretty well established where a number of variants have been created, e.g., (a) the one that focuses on kernel design [10]; (b) spectral clustering (SC) which is motivated by graph Laplacian [25], and SC is closely related to kernel K-means as a form of weighted K-means [11]; (c) a connection between SC and kernel K-means via regularization [26]; and (d) scalable kernel K-means that exploits massive parallelization [12] via a kernel functional approximation called the Nyström method [27].

However, K-Modes that requires mean-shift is not sufficiently studied. Laplacian K-Modes [16], which incorporates the graph Laplacian regularization, achieves notable improvement over K-Modes. This problem formulation is challenging because of integer constraints and a non-linear or non-differentiable component. Laplacian K-Modes has two components that have quadratic time complexity, i.e., the affinity matrix (due to the use of graph Laplacian) and the mean-shift algorithm used to find the modes.[3]

[2] This is a similar effect of the graph Laplacian used in spectral clustering [23] to improve the clustering outcomes of K-means.

[3] Spectral clustering has the same requirement of affinity matrix, and a more computationally expensive matrix factorization process.

Laplacian K-Modes, as well as other existing kernel-based clustering algorithms, cannot deal with large-scale datasets because of their high time complexity.

Scalable Laplacian K-Modes [17] has been proposed to solve a concave-convex relaxation of the objective function of Laplacian K-Modes in order to improve its efficiency. It eliminates the need to store the affinity matrix and enables parallelization of the point reassignment step. However, our experiment shows that this relaxation weakens the clustering outcomes (on a larger set of datasets used by [17]).

2.2. Distributional kernel

Let X and Y be two datasets generated independently from the two distributions $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively. Kernel mean embedding (KME) [28] defines a distribution kernel to measure the similarity between two distributions $\mathcal{P}_X$ and $\mathcal{P}_Y$ as follows:

$$\mathcal{K}_G(\mathcal{P}_X, \mathcal{P}_Y) = \frac{1}{|X||Y|} \sum_{\mathbf{x}\in X, \mathbf{y}\in Y} \mathbb{k}(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $\mathbb{k}$ is typically a Gaussian kernel in the KME framework [28].

A recent work [22] has shown that using Isolation Kernel [29] is a better option than Gaussian kernel when applying KME for point anomaly detection. The result is called Isolation Distributional Kernel (IDK) [22]. As Isolation Kernel (IK) has a finite-dimensional feature map derived directly from a given dataset D, Eq. (1) can be re-expressed as:

$$\begin{aligned}\mathcal{K}_I(\mathcal{P}_X, \mathcal{P}_Y|D) &= \frac{1}{|X||Y|} \sum_{\mathbf{x}\in X, \mathbf{y}\in Y} \mathbb{k}_I(\mathbf{x}, \mathbf{y}|D)\\ &= \frac{1}{t|X||Y|} \sum_{\mathbf{x}\in X, \mathbf{y}\in Y} \langle \phi(\mathbf{x}|D), \phi(\mathbf{y}|D)\rangle\\ &= \frac{1}{t} \left\langle \Phi(\mathcal{P}_X|D), \Phi(\mathcal{P}_Y|D)\right\rangle, \end{aligned} \tag{2}$$

where $\mathbb{k}_I(\mathbf{x}, \mathbf{y}|D) = \frac{1}{t}\langle \phi(\mathbf{x}|D), \phi(\mathbf{y}|D)\rangle$ is Isolation Kernel; $\phi(\cdot|D)$ is its feature map; and the kernel mean map of $\mathcal{K}_I$ is $\Phi(\mathcal{P}_X|D) = \frac{1}{|X|}\sum_{\mathbf{x}\in X} \phi(\mathbf{x}|D)$.

IK is a data-dependent kernel, i.e., two points are more similar to each other when measured by IK derived from a sparse region than that from a dense region [21,22,29]. Note that IK has no closed-form expression, unlike a data-independent kernel such as Gaussian kernel.

We use the implementation of IK as described in [22] to produce IDK. Like in the case of point anomaly detection [22], $\mathcal{K}_I$ is used to measure the similarity between a Dirac measure $\delta(\mathbf{x})$ and $\mathcal{P}_X$ for any $\mathbf{x} \in \mathbb{R}^d$ as follows:

$$\mathcal{K}_I(\delta(\mathbf{x}), \mathcal{P}_X|D) = \frac{1}{t} \left\langle \Phi(\delta(\mathbf{x})|D), \Phi(\mathcal{P}_X|D)\right\rangle. \tag{3}$$

The above measurement can be interpreted as the similarity between point $\mathbf{x}$ and a distribution $\mathcal{P}_X$ that generates the dataset X. In other words, the similarity between a point x and a set X is the dot product of two vectors in the kernel feature space, i.e., one vector represents the x and the other vector is the kernel mean map of X. Thus, the point-set similarity calculation has linear time complexity.

In the next section, we present the problem setting and the first kernel-based clustering algorithm IDKC, based on distributional kernel $\mathcal{K}$, described above.

3. Problem setting and idkc algorithm

3.1. Problem setting

Definition 1. Discovering k clusters of a dataset with a distributional kernel $\mathcal{K}$. Given a dataset $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathbf{x} \in \mathbb{R}^d$, the aim is to find clusters $C_j, j = 1, \ldots, k$ such that the following objective function is optimized:

$$\underset{C_1,\ldots,C_k}{\operatorname{argmax}} \sum_{j=1}^{k} \sum_{\mathbf{x}\in C_j} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_j}). \tag{4}$$

where $\delta(\mathbf{x})$ is a Dirac measure which converts point $\mathbf{x}$ into a distribution.

It is interesting to note that, independent of the algorithm used to maximize the objective function, the similarity distribution derived from the kernel $\mathcal{K}$ can be used to describe the data distribution of a dataset as follows:

Definition 2. Given cluster $C_j, j = 1 \ldots, k$ in a dataset D and the distributional kernel $\mathcal{K}$ derived from D, the $\widetilde{\mathcal{K}}$ similarity distribution is defined as:

$$\widetilde{\mathcal{K}}(\mathbf{x}|D) = \max_j \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_j}), \forall \mathbf{x} \in \mathbb{R}^d.$$

In the proposed clustering described in the next section, while growing cluster C, we recruit points $\mathbf{x}$ in the dataset D excluding C which have high similarity wrt $\mathcal{P}_C$. The similarity is measured based on $\mathcal{K}_I(\delta(\mathbf{x}), \mathcal{P}_C|D)$. The high similarity points are at the edges of C (and low similarity points are further away from C.) Only high similarity points are recruited at each iteration.

3.2. IDKC algorithm

Given a dataset N, the key steps of the proposed Isolation Distributional Kernel Clustering (IDKC) are:

(I) Identify k high local-contrast points ($\mathbf{c}_j$), based on IDK $\mathcal{K}$, which are dissimilar from each other. Each becomes the seed member of cluster $C_j = \{\mathbf{c}_j\}, j = 1, \ldots, k$.

(II) Grow cluster $C_{I(\mathbf{x})}$ if $\mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_{I(\mathbf{x})}}) > \tau_i$ with $\mathbf{x} \in N$ at iteration i until some stopping criterion is met, where $I(\mathbf{x}) = \underset{l\in[1,k]}{\operatorname{argmax}} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_l})$; and $\tau_i = \varrho \times \tau_{i-1}$ and ϱ is a user-defined growth rate. $|N|$ decreases accordingly as all clusters are grown at the end of each iteration, i.e., $N = N \setminus \cup_j C_j$.

(III) Refine clusters to optimize the objection function shown in Eq. (4).

The IDKC procedure is shown in Algorithm 1. Note that the core of the algorithm is step II, i.e., growing k clusters iteratively. Step III is a tweak at the edges, and any improvement is minor. In addition, only one parameter ψ of Algorithm 1 needs to be tuned; and the rest of the parameters can be set to some default values.

Note that the last step in IDKC is not a variant of the EM procedure, which is commonly used in kernel-based clustering. The last step in IDKC only tweaks at the edges of the clusters after the core part of each cluster has been determined in the previous step. In other words, the main clustering procedure of IDKC is not EM. Indeed, the clusters produced in the cluster growing step have good cluster quality already, as they are close to achieving the objective function[4] shown in Eq. (4) at the end of this step. The cluster refinement step does provide a minor tweak, if there is room for improvement, to the final clustering outcome.

[4] See the parameter settings used in Table A.1, and the example effect of the cluster refinement step in Table A.2 in Appendix A. A comparison of our seed selection method with the two related methods (previously used in the density peaks clustering [20,30] to select density peaks) is provided in Appendix B.

Algorithm 1: IDKC(D, ψ, t, k, q, s, ϱ)

Input : D: dataset, ψ: subsample size for IK, t: number of subsamples, k: number of clusters, q: number of nearest neighbors, s: sample size for seed selection, ϱ: growth rate
Output: Clusters $C_j^*, j = 1, \ldots, k$

$\mathcal{K}(\cdot,\cdot) \leftarrow$ Create $IDK(D, \psi, t)$;
$D' \leftarrow$ Randomly select s data points from D;
$C^0 \leftarrow$ SeedsSelection($\mathcal{K}$, D', k, q) ▷ (I) Cluster seeds selection;
Initialize $N \leftarrow D \setminus \cup_{j\in[1,k]} C_j^0$;
$\tau \leftarrow \max_{\mathbf{x}\in N, j\in[1,k]} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_j^0})$;
for $(i = 1; \tau > 10^{-5}; i++)$ **do**
  $\tau \leftarrow \varrho \times \tau$ ▷ (II) Grow k clusters based on $\mathcal{K}$;
  $C_j^i \leftarrow C_j^{i-1} \cup \{\mathbf{x} \in N \mid [\operatorname{argmax}_{l\in[1,k]} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_l^{i-1}}) = j] \wedge [\mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_j^{i-1}}) > \tau]\}, \forall_{j\in[1,k]}$;
  $N \leftarrow D \setminus \cup_{j\in[1,k]} C_j^i$;
end
$C_j^* \leftarrow C_j^{i-1} \cup \{\mathbf{x} \in N \mid \operatorname{argmax}_{l\in[1,k]} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_l^{i-1}}) = j\}, \forall_{j\in[1,k]}$ ▷ Assign all unassigned points in N;
$C^* \leftarrow$ Refinement($\mathcal{K}$, D, C^*, k) ▷ (III) Clusters Refinement;
return $C_j^*, j = 1, \ldots, k$;

Algorithm 2: SeedsSelection($\mathcal{K}$, D', k, q)

Input : $\mathcal{K}$: IDK, D': dataset, k: number of clusters, q: number of nearest neighbors used in $\mathcal{KLC}$
Output: Cluster seeds $C_j, j = 1, \ldots, k$

Select the top k points $\mathbf{c}_j, j = 1, \ldots, k$ with the largest $\mathcal{KLC}(\mathbf{x}|D', q) \times \Delta_{\mathcal{KLC}}(\mathbf{x}|D', q)$ ▷ Details in Appendix B;
$C_j \leftarrow \{\mathbf{c}_j\}, j = 1, \ldots, k$;
return $C_j, j = 1, \ldots, k$;

Algorithm 3: Refinement($\mathcal{K}$, D, C, k)

Input : $\mathcal{K}$: IDK, D: dataset, $C_j, j = 1, \ldots, k$: given clusters
Output: Refined Clusters $C_j, j = 1, \ldots, k$

Initialize $t \leftarrow 0$ and $NumChanged \leftarrow |D|$;
for *each* $\mathbf{x} \in D$ **do**
  if $\mathbf{x} \in C_k$ then $I(\mathbf{x}) \leftarrow k$;
end
while $(t < 100$ & $NumChanged > 0.01 * |D|)$ **do**
  $NumChanged \leftarrow 0$;
  for *each* $\mathbf{x} \in D$ **do**
    $J(\mathbf{x}) \leftarrow \operatorname{argmax}_{l\in[1,k]} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C_l})$;
    if $J(\mathbf{x}) \neq I(\mathbf{x})$ **then**
      $C_{I(\mathbf{x})} \leftarrow C_{I(\mathbf{x})} \setminus \{\mathbf{x}\}$;
      $C_{J(\mathbf{x})} \leftarrow C_{J(\mathbf{x})} \cup \{\mathbf{x}\}$;
      $I(\mathbf{x}) \leftarrow J(\mathbf{x})$;
      $NumChanged$++;
    end
  end
  t++ ;
end
return $C_j, j = 1, \ldots, k$;

Table 1
Time complexities of five main processes in IDKC. t and ψ are parameters of Isolation Kernel. d, k and b are numbers of dimensions, clusters and iterations, respectively.

1.	Build Isolation Kernel (IK)	$\mathcal{O}(dt\psi)$
2.	Map dataset D of n points using the feature map of IK	$\mathcal{O}(ndt\psi)$
3.	Cluster seeds selection	$\mathcal{O}(s^2t\psi)$
4.	Growing k clusters	$\mathcal{O}(bknt\psi)$
5.	Clusters Refinement	$\mathcal{O}(knt\psi)$
Total time cost for IDKC		$\mathcal{O}((s^2 + n(bk + d))t\psi)$

3.3. IDKC produces $\mathcal{K}_\varrho$-expanded clusters

IDKC discovers well-defined $\mathcal{K}_\varrho$-expanded clusters as follows:

Definition 3. A $\mathcal{K}_\varrho$-expanded cluster C grows from a seed $\mathbf{c}$ selected from D, using $\mathcal{K}(\cdot,\cdot|D)$ with growth rate $\varrho \in (0, 1)$, is defined recursively as:

$$C^i = \{\mathbf{x} \in D \mid \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C^{i-1}}) > \tau_i\},$$

where $C^0 = \{\mathbf{c}\}$; $\tau_0 = \max_{\mathbf{x}\in D\setminus C^0} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C^0})$; $\tau_i = \varrho \times \tau_{i-1}$.

All k clusters are grown in parallel at each iteration in Algorithm 1 until the last iteration b where either there are no more unassigned points in D or all unassigned points $\mathbf{x}$ have $\mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_{C^{b-1}}) \leq \tau_b$.

Once all clusters are obtained, the representative point or 'peak' of each cluster C is:

$$\hat{\mathbf{c}} = \operatorname{argmax}_{\mathbf{x}\in C} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_C).$$

$\mathcal{K}_\varrho$-expanded clusters have arbitrary shapes, varied densities and sizes. This is because each cluster is expanded in all directions in the local neighborhood of the growing cluster C^{i-1} at iteration i until threshold τ_i is reached. The expansion rate at each direction depends on the local data distribution, moderated through the distributional kernel $\mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_C)$. In other words, no structural constraints are imposed on the type of clusters that can be discovered by IDKC.

Note that Algorithm 1 is encapsulated in the above definition, and the algorithm is deterministic, given the kernel $\mathcal{K}$ and other the parameter settings of the algorithm (i.e., k and ϱ).

Table 1 shows the time complexities of the five main processes in IDKC. Building Isolation kernel and mapping n points using IK's feature map cost $\mathcal{O}(ndt\psi)$, where $\psi \ll n$. The cluster seeds selection is based on a data subset of fixed size $s \ll n$ for a large dataset. Processes 4 and 5 (based on the distributional kernel) and the total time complexity of IDKC are linear to the dataset size n since all other parameters are constant.

3.4. Conceptual comparison with K-modes and Laplacian K-modes

The related algorithms K-Modes [15] and Laplacian K-Modes [16] were introduced as a way of extending K-means via mean-shift [14]. Table 2 shows a summary of the comparison of IDKC, K-Modes and Laplacian K-Modes.

Some key points are given below:

(i) IDKC is not a representatives-based method like existing kernel-based clustering algorithms that employ EM, where all k representatives influence the point reassignments at each iteration. In fact, there is no point reassignment once a point is assigned to a cluster in the main process, i.e., step II in the IDKC algorithm. Yet, it is able to produce a typical representative as the cluster peak which has the highest similarity, corresponds to the mode of each cluster in K-Modes and Laplacian K-Modes.

Table 2
Differences among IDKC, K-Modes and Laplacian K-Modes (L-KM). $\mathfrak{f}$ or $\mathfrak{f}'$ or $\mathfrak{f}''$ is the maximum number of iterations in each algorithm.

	IDKC	K-Modes	Laplacian K-Modes
Objective function	Eq. (4)	Eq. (C.1)≡ Eq. (4)	Eq. (C.1) + regularizer
Optimization algorithm	Not optimization	EM	EM
Point assignment	Hard	Hard	Soft
Clusters formation	Grow iteratively via $\mathcal{K}$	Parallel formation via means & mean-shift	Parallel formation via means & mean-shift
Kernel mean map (key role)	Growing each cluster	Not applicable	Not applicable
Representatives	Highest $\mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_C)$ points	Mode points	Mode points
Means in input space	No	Yes	Yes
- critical to cluster formation	No	Yes	Yes
- sensitive to initial means	No	Yes	Yes
Cluster definition	Definition 3 (Section 3.3)	Definition 4 Appendix C.1	Undefined
Cluster type	Arbitrary shapes, varied densities and sizes	Voronoi diagram	Mutant Voronoi diagram
Time complexity	$\mathcal{O}(\mathfrak{f}(\tau,\varrho)nkdt\psi)$	$\mathcal{O}(\mathfrak{f}'(d,\beta)nkd)$	$\mathcal{O}(\mathfrak{f}''(d,\beta,\epsilon_1,\epsilon_2)nk+n^2)$

(ii) IDKC is not an optimization method; yet it is arguably an algorithm better than EM-like methods which are often used to optimize objection function (C.1) which is equivalent to objective function (4) used by IDKC. The superiority of IDKC is exemplified by its simplicity, fast runtime, no constraint on cluster structure, and no random initialization. IDKC is simpler because the main operation has only one step in each iteration, i.e., growing a cluster. In contrast, an EM procedure has two steps (the E-step and M-step) in each iteration.

(iii) Kernel mean map plays a key role in growing a cluster in IDKC, and it can be regarded as the mean (in the feature space) of a growing cluster. But it has no influence on the formation of other clusters. In contrast, K-Modes and its variants (and also kernel K-means) do not explicitly use the kernel mean map in the computation[5] and often employ a data-independent kernel that has a feature map with intractable dimensionality. Using Isolation Kernel that has a finite dimensional feature map, IDKC has linear runtime because each evaluation of $\mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_C)$ takes constant time.

(iv) The time complexities of the three algorithms have the same form, i.e., $\mathfrak{f}(\cdot)nk$, where $\mathfrak{f}(\cdot)$ is the maximum number of iterations. IDKC has $\mathfrak{f}$ independent of data size, i.e., $\mathfrak{f}(\tau,\varrho) = \left\lfloor \frac{\log \tau}{\log(1-\varrho)} \right\rfloor$. L-KM has $\mathfrak{f}''(d,\beta,\epsilon_1,\epsilon_2) = d\log 1/\epsilon_1 + \beta/\sqrt{\epsilon_2}$ [16], where ϵ_1 and ϵ_2 are the optimization precision of the two steps, and β is the neighborhood size in constructing the graph Laplacian. The key computational cost of L-KM is building the graph Laplacian which costs n^2. The scalable version SL-KM has a total cost that must include n^2 (though not stated in [12]) because of the need to compute kNN search to calculate the affinity matrix for the graph Laplacian (see the scaleup test in the next section).

The details of K-Modes and Laplacian K-Modes are provided in Appendix C.

4. Experimental results

4.1. Experimental design

The experiments are designed to answer the following two questions:

(i) Is IDK (Isolation Distributional Kernel) a better kernel to grow a cluster than GDK (Gaussian Distributional Kernel)?

(ii) Does the new clustering algorithm produce better clustering outcomes than existing kernel-based clustering algorithms that employ the EM procedure?

To answer the first question, we use exactly the same algorithm as IDKC, except that IDK is replaced with GDK; and it is called GDKC.

To answer the second question, existing kernel-based clustering algorithms used in the evaluation include K-Modes (KM) [15], Laplacian K-Modes (L-KM) [16], Scalable Laplacian K-Modes (SL-KM) [12], Spectral Clustering (SC) [31], Structured Graph Learning (SGL) [32], and SP-DP [33] and LC-DP [30].[6] Note that SGL is a K-means-based clustering that does not employ a kernel. It is included in the comparison because it is a recent method that can deal with large-scale datasets. We also include IK-KM which is KM using Isolation Kernel instead of Gaussian kernel. All algorithms were implemented by their original authors in Matlab, except that SL-KM was implemented in Python. We systematically searched the parameters for all comparison algorithms and reported their best performance. The parameter search ranges of these clustering algorithms for the following evaluations are provided in Table A.1 in Appendix A.

We conduct 6 sets of experiments in order to evaluate the performance of the proposed IDKC in different aspects. They are reported in the following 6 subsections. The clustering performance is measured in terms of NMI (Normalised Mutual Information) [37].

4.2. Comparison with Gaussian distributional kernel

Fig. 1 illustrates the cluster expanding process of IDKC as the number of iterations increases. The two-dimensional dataset has two Gaussian clusters of different densities, as shown in Fig. 1(a).

Notice that IDKC, which is adaptive to local data distribution by using Isolation Kernel, creates all cluster peaks with high similarities in both dense and sparse regions, as shown in Fig. 1(i). Also, the cluster in the sparse region expands at a faster rate than that in the dense region: IDKC has the outermost contour line of the sparse cluster that covers a much larger area than that of the dense cluster, as shown in Figs. 1 (f)–(h). In contrast, GDKC (which is exactly the same algorithm as IDKC, except GDK [28] is used instead of IDK)[7] has very low cluster peak in the sparse region.

[5] Conceptually, the implicit kernel mean maps in kernel K-means can be used to denote the means in the feature space.

[6] DP [20] is a density-based clustering method that can effectively identify clusters with varied densities and usually perform better than the classic density-based clustering method DBSCAN [34], as shown in recent literature [33,35,36]. SP-DP and LC-DP denote the shortest path-based version and local contrast version of density peak, respectively. Both SP-DP and LC-DP show better clustering performance than DP.

[7] We utilized a large-scale Nyström approximation [38] to generate the feature map for Gaussian kernel.

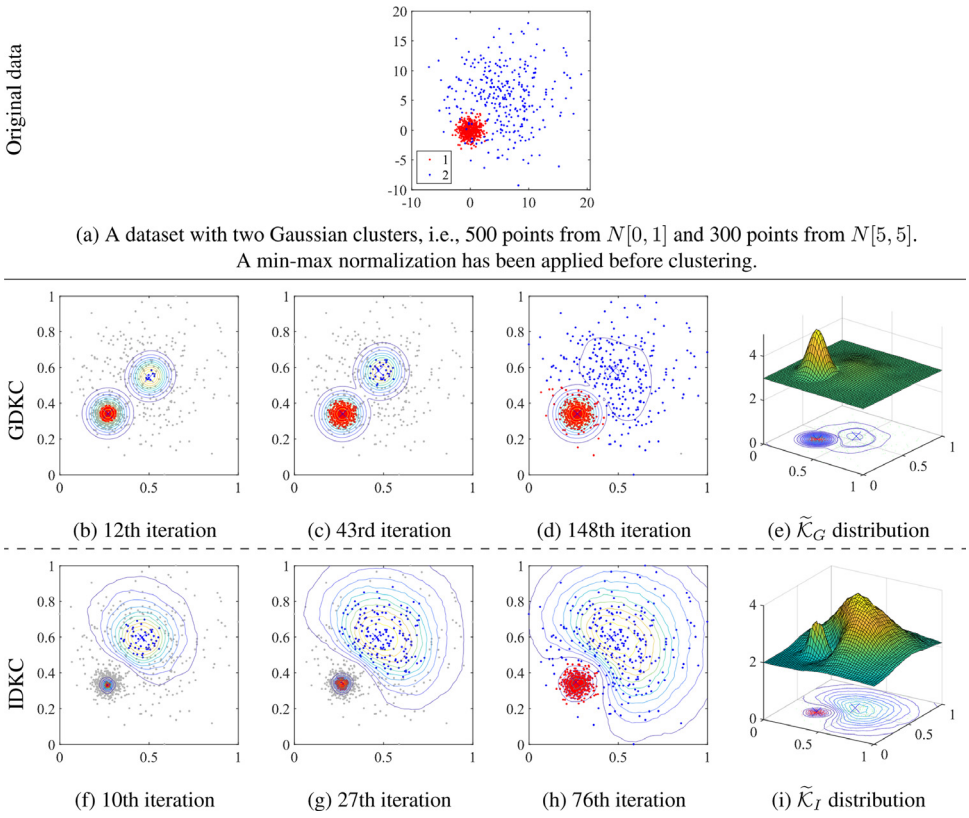


Fig. 1. Cluster expanding processes of GDKC (in subfigures (b-d)) and IDKC (in subfigures (f-h)) at different iterations, given the same initial cluster seeds. The contour depicts the similarity distribution as stated in Definition 2. Gray points are points that are yet to be clustered. Each cross indicates the cluster peak of a cluster at the end of the iteration. GDKC and IDKC produced the NMI scores of 0.78 and 0.86 for the final clustering outcomes, respectively, as shown in the third column. The last column shows the $\widetilde{\mathcal{K}}$ distributions based on the true clusters, as given in Definition 2.

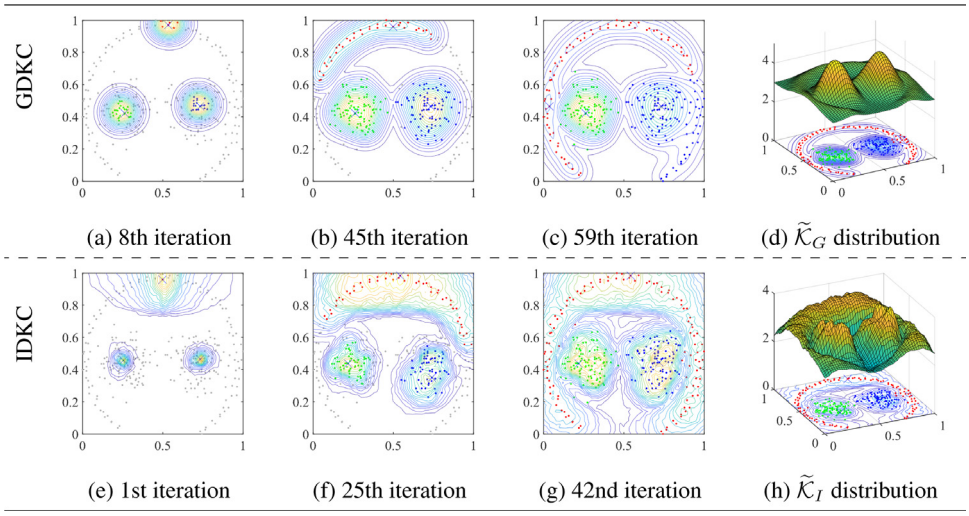


Fig. 2. Cluster expanding processes of GDKC (in subfigures (a-c)) and IDKC (in subfigures (e-g)) at different iterations, given the same initial cluster seeds, on the Pathbased dataset. The contour depicts the similarity distribution as stated in Definition 2. Gray points are points that are yet to be clustered. Each cross indicates the cluster peak of a cluster at the end of the iteration. GDKC and IDKC produced the NMI scores of 0.78 and 0.97 for the final clustering outcomes, respectively, as shown in the third column. The last column shows the $\widetilde{\mathcal{K}}$ distributions based on the true clusters, as given in Definition 2.

Compare Fig. 1(d) and Fig. 1(h), GDKC's outermost contour line of the dense cluster covers a much larger area than that of the dense cluster produced by IDKC; and the reverse is true for the sparse clusters. Note that GDKC has significantly fewer contour lines in the sparse region because of low density.

Fig. 2 shows the same effect on another two-dimensional dataset (Pathbased) with three clusters of different densities [39]. IDKC obtained a nearly perfect clustering result NMI= 0.97. However, GDKC got NMI= 0.78 only, since the right blue dense (Gaussian) cluster merges many points from the sparse circle cluster.

The negative impact of a data-independent kernel such as Gaussian kernel, which has no adaptive capability, is again demonstrated in the last column in Fig. 2. Using Gaussian kernel, the dense cluster peaks always have similarities significantly higher than that of the sparse cluster. Notice that the valley between

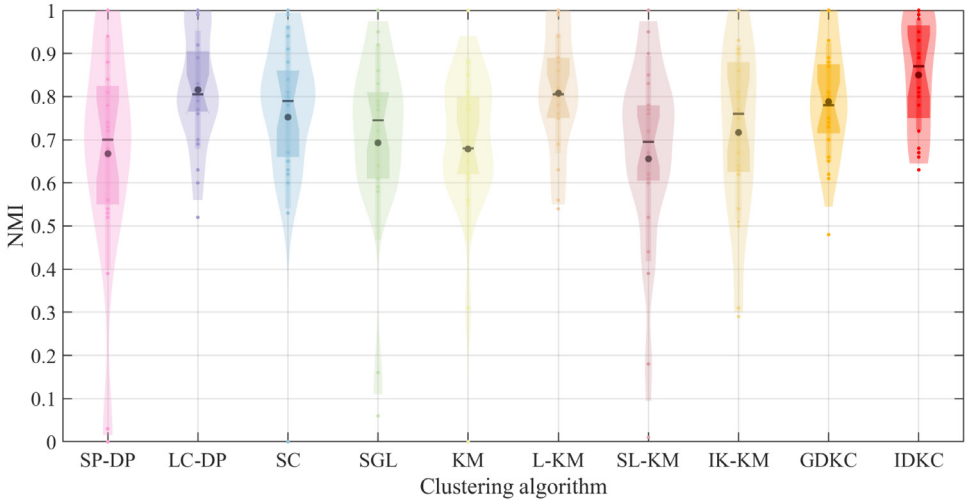


Fig. 3. The clustering results of 10 clustering algorithms in terms of NMI on 22 datasets, presented in a violin plot. The black bar and black dot denote the median and mean values, respectively.

Table 3
Best clustering results in NMI on 22 datasets. We report the best result over 10 trials on each dataset for the clustering algorithm with randomization. SP-DP and SC ran out of memory on the MNIST dataset using a machine with 256GB RAM.

	#Points	d	k	SP-DP	LC-DP	SC	SGL	KM	L-KM	SL-KM	IK-KM	GDKC	IDKC
Pathbased	300	2	3	0.73	0.76	0.79	0.58	0.55	0.56	0.78	0.31	0.78	**0.98**
Spiral	312	2	3	**1.00**	**1.00**	**1.00**	0.16	0.00	**1.00**	0.01	0.29	**1.00**	**1.00**
Aggregation	788	2	7	0.88	**0.99**	0.96	0.86	0.85	**0.99**	0.76	0.90	0.89	**0.99**
AC	1004	2	2	**1.00**	**1.00**	0.63	0.61	0.62	**1.00**	0.39	**1.00**	**1.00**	**1.00**
S3	5000	2	15	0.78	**0.80**	**0.80**	0.77	**0.80**	**0.80**	0.44	**0.80**	**0.80**	**0.80**
Iris	150	4	3	0.66	0.89	0.78	0.83	0.77	0.89	0.62	0.90	0.88	**0.93**
Wine	178	13	3	0.54	0.83	0.91	**0.95**	0.88	0.89	0.78	0.91	0.87	0.90
Seeds	210	7	3	0.67	0.77	**0.79**	0.77	0.68	0.78	0.72	0.75	0.74	0.78
Ecoli	336	7	8	0.52	0.69	0.65	0.69	0.65	**0.75**	0.52	0.63	0.66	0.67
Dermatology	358	34	6	0.67	0.89	0.88	0.92	0.88	0.94	**0.95**	0.93	0.87	**0.95**
Libras	360	90	15	0.56	**0.70**	0.62	0.64	0.62	0.69	0.67	0.64	0.65	0.68
ForestType	523	27	4	0.39	0.63	0.53	0.58	0.62	0.63	0.60	0.63	0.61	**0.66**
Control	600	60	6	0.72	0.81	0.81	0.78	0.78	0.86	0.62	0.86	0.78	**0.93**
LandCover	675	147	9	0.53	0.60	0.60	0.59	0.61	**0.63**	0.61	0.62	0.62	**0.63**
Mice	1080	83	8	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Banknote	1372	4	2	0.94	**0.99**	0.94	0.06	0.31	0.88	0.18	0.51	0.93	**0.99**
Coil20	1440	1024	20	0.84	**0.92**	0.84	0.86	0.80	0.80	0.90	0.81	0.81	0.89
Isolet	1560	617	26	0.68	0.77	0.80	0.78	0.80	0.82	**0.85**	0.80	0.78	0.81
Segment	2310	19	7	0.74	**0.77**	0.67	0.69	0.65	0.75	0.67	0.67	0.70	0.72
LabelMe	2688	4096	8	0.03	0.52	0.78	0.79	0.81	0.81	**0.83**	0.54	0.73	0.72
Pendig	10,992	16	10	0.81	0.83	0.79	0.72	0.68	0.76	0.76	0.77	0.75	**0.85**
MNIST	70,000	784	10	–	0.79	–	0.61	0.56	0.54	0.77	0.50	0.48	**0.82**
			Average	0.70	0.82	0.79	0.69	0.68	0.81	0.66	0.72	0.79	0.85
			#Best Performer	3	9	3	2	2	7	4	3	4	14

the sparse cluster and dense clusters are not distinct enough to clear separation between them. Note that no bandwidth setting of Gaussian kernel is able to accentuate the sparse cluster.

In contrast, all cluster peaks produced by the Isolation Kernel have high similarities in Fig. 2(h), independent of their densities; and most importantly, it produces a deep valley in-between clusters—making a clear distinction between clusters in both sparse and dense regions. The implementation details of Isolation Kernel and its data-dependent property are provided in Appendix D.

4.3. Comparison with all benchmark algorithms on numerical datasets

Fig. 3 shows the overall clustering results of the 10 clustering algorithms on 5 synthetic[8] and 17 real-world datasets in terms of NMI, presented in a violin plot.[9] It shows that IDKC has the highest mean and median values among all clustering algorithms.

The full comparison results of the 22 datasets are shown in Table 3. IDKC is the best performer on 14 datasets; followed by LC-DP and L-KM on 9 and 7 datasets, respectively. It also performs the best on all of the 5 synthetic datasets. Fig. 4 demonstrates the best clustering results of 5 clustering algorithms on 3 synthetic datasets. It shows that IDKC has the ability to detect clusters of arbitrary shapes and varied densities.

SC uses a feature transformation procedure before running a K-means clustering method, and it has been shown to be equivalent to a weighted kernel K-means [40]. However, it still has difficulty identifying clusters with arbitrary shapes, such as the Pathbased and AC datasets as shown in Fig. 4.

[8] We select those benchmark synthetic datasets to evaluate the effectiveness of different clustering algorithms for clustering clusters of varied densities, sizes and shapes. Pathbased, Sprial, Aggregation and S3 are from http://cs.joensuu.fi/sipu/datasets/. The AC dataset was first used in [10].

[9] A violin plot is similar to a box plot, having an additional rotated plot of density distribution of the data on each side.

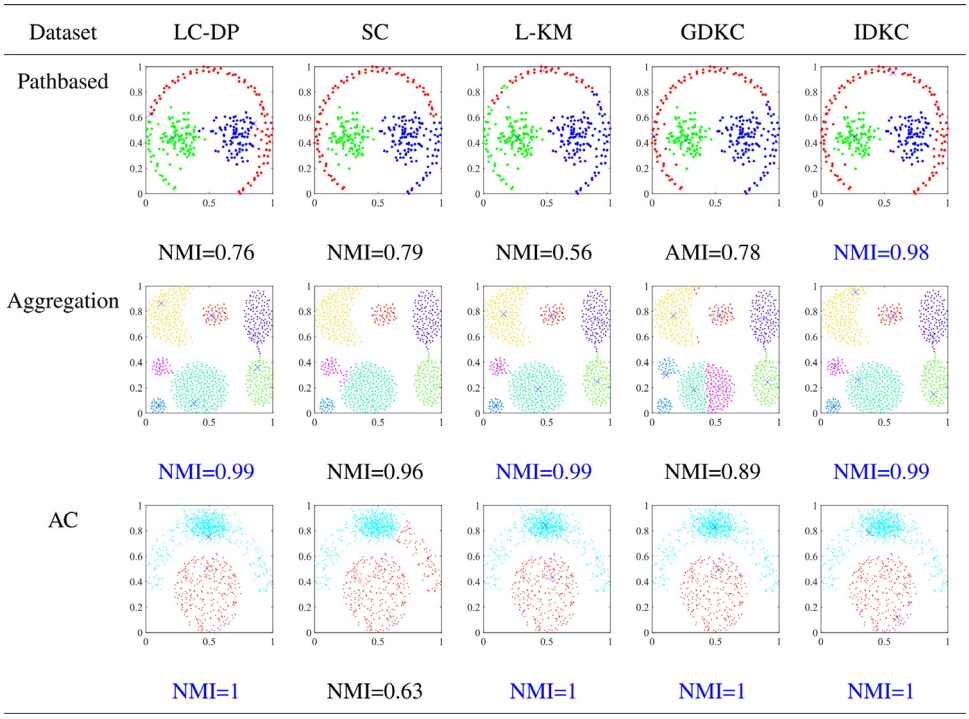


Fig. 4. The clustering results of 5 clustering algorithms on 3 synthetic datasets.

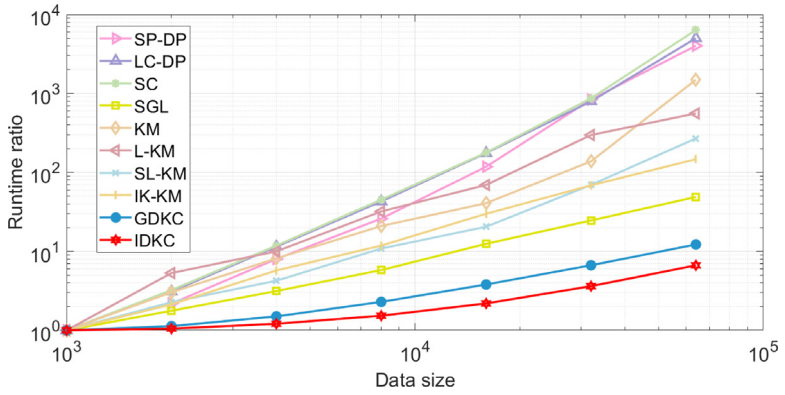


Fig. 5. Scaleup test results reported in runtime ratio with respect to 1000 points.

It is interesting to see that although both IDKC and IK-KM are based on the same data-dependent Isolation kernel, IK-KM performed significantly worse than IDKC due to the use the EM procedure. The key difference is: IK-KM uses the set of representative points as a whole for clusters formation, while IDKC uses a distribution to represent each cluster.

A comparison between GDKC and IDKC in Table 3 shows that IDK is a better kernel to grow clusters than GDK in all datasets, because Gaussian kernel used in GDK is a *data-independent* kernel; and Isolation Kernel used IDK is a *data dependent* kernel.[10] The only exceptions are (a) LabelMe where the difference in NMI is very small; and (b) equal NMI in four datasets. Compared with the best EM-based L-KM, IDKC performed better in all but five datasets in which the difference in NMI is small (and equal NMI in seven datasets.) Still, IDKC has a huge NMI advantage over L-KM on the Pathbased, Banknote and MNIST datasets. When Gaussian kernel is used, GDKC performed worse than L-KM in all but three datasets (and equal NMI in four datasets).

These comparisons show that IDK is the key to obtaining superior clustering outcomes. IDKC also performed better than its closest contender LC-DP in all but four datasets with small difference in NMI (and equal NMI in six datasets). IDKC has a huge advantage over LC-DP on the Pathbased, Control and LabelME datasets.

4.4. Scaleup test

We conduct a scaleup test on the subsets of MNIST between 1000 and 64,000 data points. Both GDKC and IDKC use a set of 1000 random points to determine the initial cluster seeds. SL-KM utilizes a fast approximation [42] for the kNN search. Since these clustering algorithms were implemented in different platforms with different packages, it is unfair to directly compare their runtimes. The runtime ratio is a fair and better indicator of the time complexity of an algorithm. It is a ratio of the runtime on a dataset of n points and the runtime on a subset of 1000 points. Fig. 5 shows the result of a scaleup test for 9 algorithms.

The result of a scaleup test in Fig. 5 shows that IDKC and GDKC have the best scaleup capability, followed by SGL, IK-KM and

[10] The influence of Isolation kernel on SVM classification, density-based clustering and t-SNE visualization has been given in previous studies [21,29,41].

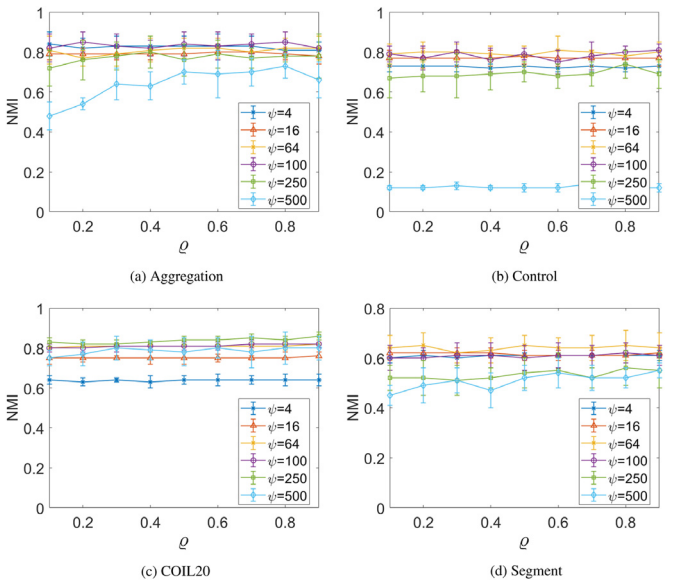


Fig. 6. The effects of different ψ and ϱ values on the clustering performance of IDKC, measured in NMI scores.

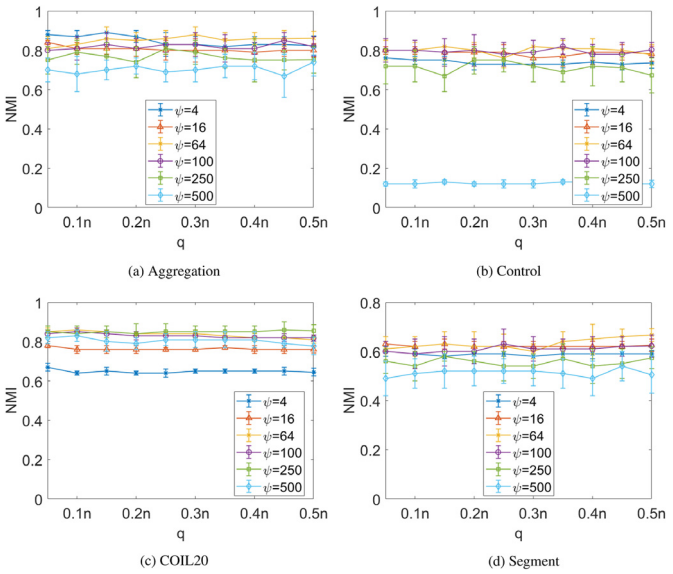


Fig. 7. The effects of different ψ and q values on the clustering performance of IDKC, measured in NMI scores.

SL-KM. In other words, all current K-Modes algorithms, i.e., KM, L-KM and SL-KM have problems dealing with large-scale datasets. This result is consistent with the time complexities (shown in Tables 1 and 2). It is worth mentioning that IDKC only took 12 s to cluster 64,000 data points, while LC-DP, SC and SL-KM took 825, 630 and 252 s, respectively.

4.5. Sensitivity analysis

Although IDKC has five parameters as shown in Algorithm 1, most parameters can be set to default values in practice. The only parameter that needs to be tuned is the sharpness parameter ψ, and its role is similar to the bandwidth parameter σ in GDK.

The default settings of sample size s and growth rate ϱ are 10000 and 0.9, respectively; the number of nearest neighbors $q = 0.4 \times n$; and the ensemble size parameter $t = 100$.

Here we examine the effects of different parameters in IDKC. We reported the average NMI score (with a standard deviation) over ten independent trials for every parameter setting. Fig. 6 reports the NMI results on four representative datasets using different ψ and ϱ values, with $q = 0.4n$, $t = 100$ and k is set to the true number of clusters. In addition, Fig. 7 reports the NMI results using different ψ and q values, with $\varrho = 0.9$, while t and k were kept the same as in Fig. 6.

Both results show that IDKC often maintains a stable clustering quality when ψ is set to 64 and 100. The parameters q and ϱ have significantly less impact on the clustering performance of IDKC than ψ.

4.6. Image segmentation via clustering

In this section, we compare the three scalable algorithms, IDKC, GDKC and SGL, in an image segmentation task, where the image has more than one million pixels.[11] Other EM algorithms take too long to run on this image.

Fig. 8 shows the segmentation results on the original image shown in the first row. Each clustering was performed on the pixels in the CIELAB space [43]. IDKC produced a 'clean' separation of two clusters in the CIELAB space shown in Fig. 8 (in the last row), where the two clusters of fish and water are almost perfectly separated by IDKC. Notice that, for both SGL and GDKC (which employ Euclidean distance and Gaussian kernel, respectively), the water cluster includes some spots from the fish cluster.

The last column in Fig. 8 shows the pixel distribution in the CIELAB space. We can see that the fish cluster (denoted with green pixels) has a sparse region; whereas the water cluster (denoted with blue pixels) is entirely in a dense region. The observations are:

- SGL links some pixels around the fish eye to the water cluster.
- GDKC links the yellow spots on the fish to the water cluster. This 'far-end' connection in the CIELAB space could be due to the estimation error in the Nyström approximation [27] used in GDKC. Because of its high time complexity, we are unable to perform GDKC clustering on this dataset using Gaussian kernel without using the Nyström approximation.
- IDKC is able to produce an almost perfect segmentation on this million-data-points dataset because of its data dependency and a finite-dimensional feature map.

4.7. Deep-learning based clustering on a dataset of images

Clustering unstructured data objects such as images in a dataset is challenging because the given representation is at a low level. Using neural networks to perform representation learning from images have shown promising results in deep-learning-based clustering. Here we compare with a recent unsupervised end-to-end deep-learning method called Contrastive Clustering (CC) [44]. CC has two components: the instance-level contrastive head (ICH) and cluster-level contrastive head (CCH), which connect to the same backbone. ICH together with the backbone is effectively responsible for feature extraction; and CCH performs clustering.

To examine CC's clustering capability in comparison with other clustering methods having no representation learning, the representation learning component is separated from its clustering component. The learned feature representation, extracted from the neural network, is trained using the instance-level

[11] We also evaluated the commonly used K-means clustering (imsegkmeans in Matlab Image Processing Toolbox), but found that the results are poorer than that of SGL on this task.

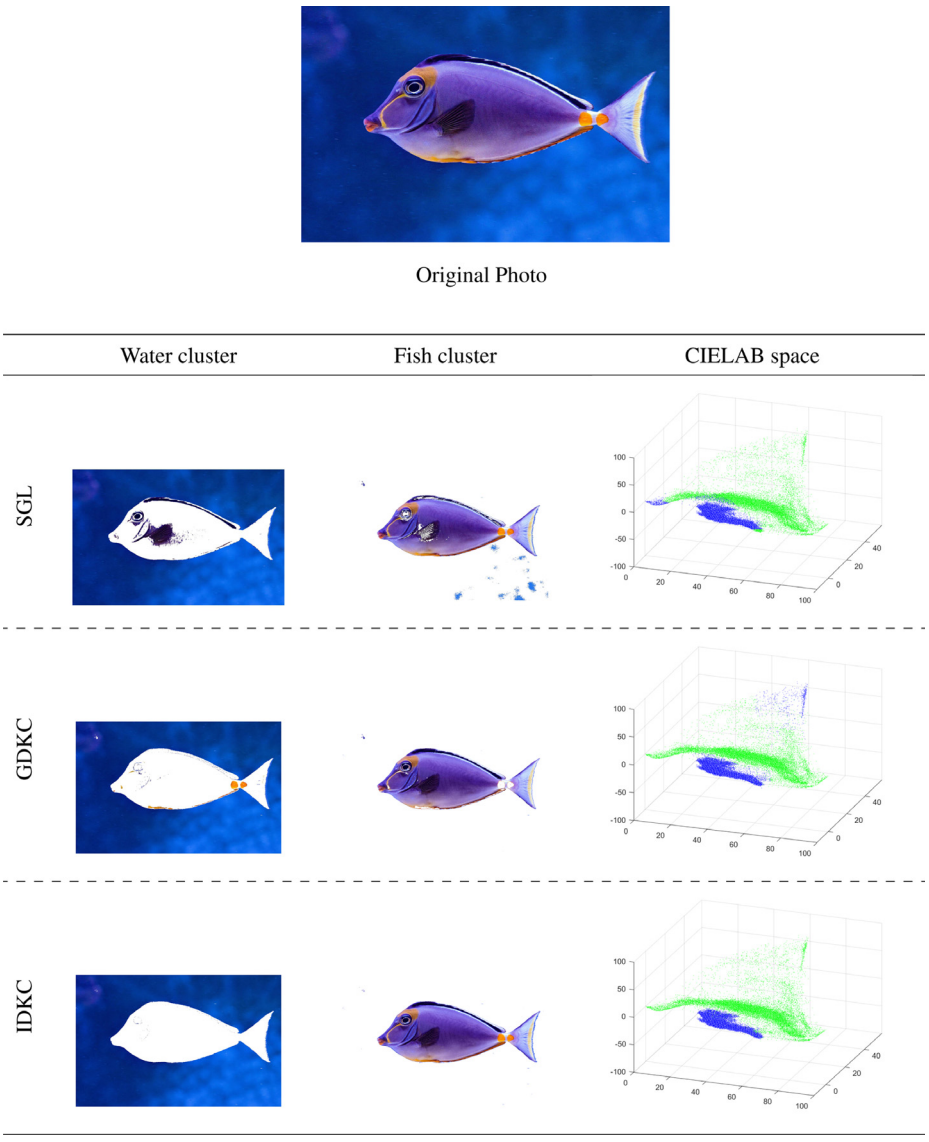


Fig. 8. Image segmentation results on the original image with 1 million pixels shown on the first row. The last column shows the pixel distribution of the fish image in the CIELAB space. Blue and green pixels represent the water cluster and fish cluster identified by a clustering algorithm, respectively. For better readability, the CIELAB space plots use 0.1 million points only.

Table 4
Best clustering results of 7 algorithms in terms of NMI on 6 image datasets. All methods use the representation learned from the ICH component of Contrastive Clustering (CC) [44], without the clustering component CCH. SC ran out of 256GB memory on Tiny-ImageNet.

	#Points	d	k	CC	K-means	LC-DP	SC	SGL	L-KM	IDKC
STL-10	13,000	128	10	0.63	0.66	0.67	0.68	0.69	0.68	**0.70**
CIFAR-10	60,000	128	10	0.71	0.70	0.69	**0.74**	0.73	0.73	**0.74**
CIFAR-100	60,000	128	20	0.43	0.43	0.46	**0.48**	0.46	0.46	0.46
ImageNet-10	13,000	128	10	0.86	0.84	0.87	0.83	0.88	0.87	**0.89**
ImageNet-Dogs	19,500	128	15	0.45	0.49	0.51	0.50	0.53	**0.54**	**0.54**
Tiny-ImageNet	100,000	128	200	0.35	**0.39**	0.38	–	0.35	0.38	**0.39**

contrastive head (ICH) of CC with the backbone only in this experiment.[12] All methods without representation learning are given with datasets in this learned representation. In this way, we can compare head-to-head on the clustering procedures using the same feature representation provided by the deep network.

Here We evaluated 7 algorithms on the same 6 image datasets used in the study of CC [44]. The data sizes range from 13,000 to 100,000. The detailed data properties and performance of each algorithm are provided in Table 4. It can be seen from the table that the clustering capability of the deep-learning CC is not better than these non-deep-learning methods.[13] For example, CC performs worse than L-KM and IDKC on all the six datasets; and

[12] We used the source code with the best parameter settings provided by the author [44], trained and tested on the complete datasets (as conducted by previous work [45–47]).

[13] The original clustering results of CC are very close to the results in Table 4, except the STL-10 was conducted on a larger datasets in the CC paper [44].

worse than LC-DP and SGL in five out of six datasets. Even when compared with the simplest K-means, CC has more losses than wins. Overall, IDKC has the best NMI in five out of the six datasets; and the closest contender L-KM has the best NMI in one dataset only (ImageNet-Dogs).

In a nutshell, the clustering performance of the end-to-end deep-learning method CC is mainly derived from representation learning. Using the same feature representation, we show that the current clustering capability of the end-to-end deep clustering is weaker than many existing non-deep-learning methods. Consistent with the results in the previous sections, IDKC has the best clustering capability among non-deep clustering methods, and it is also better than the deep-learning method CC, when they are all compared on the same learned representation.

5. Conclusion

This paper introduces the first algorithm to achieve the same objective function as that of existing K-Modes clustering, without using the EM optimization procedure. The algorithm and its use of Isolation Distributional Kernel (IDK) are the keys to its success. It is a deterministic algorithm that employs IDK to (a) find the seeds which have high similarity with respect to the data distribution but are not similar to each other; (b) grow each current cluster C by recruiting points that are most similar to the distribution of C iteratively; and (c) define the clustering objective function such that the clusters obtained are further refined to optimize the objective function.

The existing kernel-based clustering algorithms have two main issues, i.e., the EM procedure and the use of a non-adaptive kernel. The EM procedure limits the type of clusters to the Voronoi diagram; even the attempt to relax this constraint through a soft assignment in L-KM also produces a mutant Voronoi diagram. While this relaxation enables L-KM to perform better than KM, the clusters discovered could not be defined formally. Furthermore, the issues of instability and sensitivity to initial seeds and outliers of the EM procedure remain unresolved.

We present a variant of the proposed clustering algorithm that uses Gaussian Distributional Kernel, i.e., GDKC, and it performed significantly worse than IDKC in terms of clustering outcomes because the non-adaptive GDK does not accentuate clusters in the sparse region, often resulting in dense clusters to encroach the sparse region—leading to poor clustering outcomes.

The new clustering algorithm IDKC that employs IDK has none of the limitations of current kernel-based clustering algorithms that employ the EM procedure and a non-adaptive kernel because IDKC employs an adaptive distributional kernel without any optimization. As a result, it often produces better clustering outcomes; and it is the only kernel-based clustering that can deal with large-scale datasets with millions of data points without approximation or sampling.

In addition, we show that a recent end-to-end deep-learning clustering method derives its good clustering performance on an image dataset mainly from the learned representation. In an ablation study, we discover that the clustering capability of the end-to-end deep-learning clustering method is weaker than many existing non-deep-learning clustering methods, including IDKC.

CRediT authorship contribution statement

Ye Zhu: Investigation, Methodology, Data curation, Software, Validation. **Kai Ming Ting:** Supervision, Conceptualization, Writing – Original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

In Table 3, Pathbased, Sprial, Aggregation and S3 are from http://cs.joensuu.fi/sipu/datasets/, and LabelMe is from [17]. The AC dataset was first used in [10]. All other real-world datasets are from the UCI Machine Learning Repository [48]. We used a min–max normalization [49] on each attribute to ensure its values in [0,1]. Fig. A1 in Appendix A shows the scatter plots of the 5 synthetic datasets used in the evaluation. The fish image used in Fig. 8 is from https://unsplash.com/photos/9FvTOCfvt_w. The image datasets used in Table 4 are obtained following https://github.com/Yunfan-Li/Contrastive-Clustering.

Acknowledgments

Funding

Kai Ming Ting is supported by the National Natural Science Foundation of China (Grant No. 62076120).

Code availability

The source codes of GDKC and IDKC are available at https://github.com/zhuye88/IDKC.

Appendix A. Empirical evaluation

Table A.1 shows the parameter search ranges of different clustering algorithms for empirical evaluation.

Fig. A1 shows the scatter plots of the 5 synthetic datasets used in the evaluation.

Table A.2 demonstrates the effect of cluster refinement in IDKC.

Appendix B. Three ways to find the peaks of a data distribution

The two existing methods, i.e., DP [20] and LC-DP [30], attempt to find the peaks in a density distribution. In contrast, IDKC attempts to find the peaks in a $\mathcal{K}$ similarity distribution.

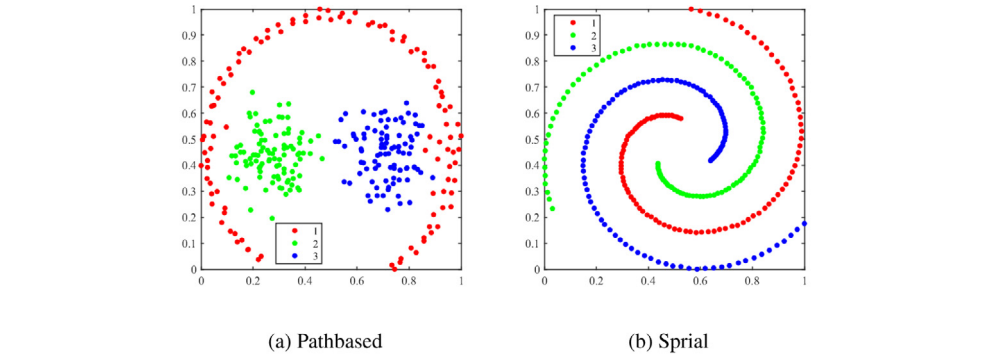
Instead of computing density for each point based on ϵ-neighborhood density estimator in DP [20] and LC-DP [30], we measure the similarity of a point $\mathbf{x}$ with respect to the distribution that generates the given dataset D. For every point $\mathbf{x} \in D$, IDKC first estimates the similarity $g(\mathbf{x}|D) = \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_D)$ and counts the number of times that $\mathbf{x}$ has a similarity higher than those of its q nearest neighbors, i.e., $\mathcal{KLC}(\mathbf{x}) = \sum_{\mathbf{y} \in N_q(\mathbf{x})} I_{\{g(\mathbf{x}|D) > g(\mathbf{y}|D)\}}$, where $N_q(\mathbf{x})$ is the set of the q nearest neighbors of $\mathbf{x}$.

Then it calculates the minimum dissimilarity between $\mathbf{x}$ and another point $\mathbf{z}$ with a higher $\mathcal{KLC}(\mathbf{z})$ as $\Delta_{\mathcal{KLC}}(\mathbf{x}) = \min_{\mathcal{KLC}(\mathbf{z}) > \mathcal{KLC}(\mathbf{x}), \mathbf{z} \in D} 1 - \mathcal{K}(\delta(\mathbf{x}), \delta(\mathbf{z}))$; and $\Delta_{\mathcal{KLC}}(\mathbf{x}_m) = 1$ if $\mathbf{x}_m = \operatorname{argmax}_{\mathbf{x} \in D} \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_D)$. Finally, the k points with the highest $\mathcal{KLC}(\mathbf{x}) \times \Delta_{\mathcal{KLC}}(\mathbf{x})$ are selected as the cluster seeds for k clusters. We apply the rank transform to each of $\mathcal{KLC}(\mathbf{x})$ and $\Delta_{\mathcal{KLC}}(\mathbf{x})$ before the product.

Fig. A2 compares the cluster centers (peaks) identified by three methods on the 2-dimensional dataset shown in Fig. 1(a). Note that DP cannot identify the two cluster peaks because the

Table A.1
Parameter search range for each clustering algorithm. The k in every algorithm is set to the true number of clusters of a dataset. k is the number of nearest neighbors used in the kNN search.

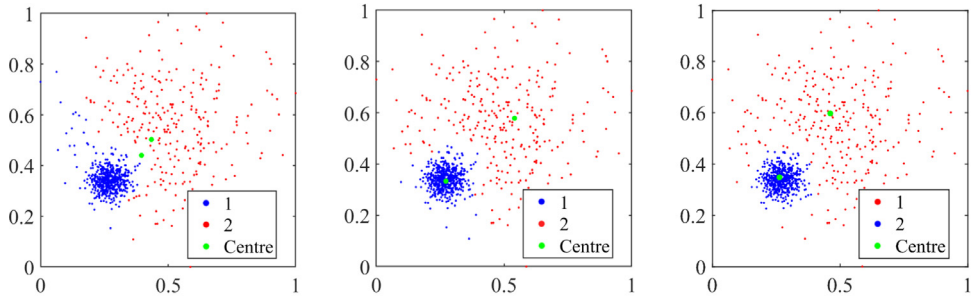
Algorithm	Parameter search range
DP-SP	$DP_{METHOD} = topmost_{otsu}$; $TH_{PRUNING} \in \{0, 100, \ldots, 1000\}$
LC-DP	$k \in \{2, 3, \ldots, 50\}$; $\epsilon \in \{0.001, 0.002, \ldots, 0.999\}$; $q \in \{0.05n, 0.1n, \ldots, 0.45n, 0.5n\}$
SC	$\sigma = d \times \mu$; $\mu \in \{2^{-5}, 2^{-4}, \ldots, 2^5\}$
SGL	$\#anchor \in \{30, 40, 50\}$; $\alpha \in \{0.1, 1, 10\}$; $\beta \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$
KM	$\sigma \in \{0.10, 0.11, \ldots, 10\}$
L-KM	$k \in \{2, 3, \ldots, 50\}$; $\sigma \in \{0.1, 0.2, \ldots, 1\}$; $\lambda \in \{0.1, 0.2, \ldots, 10\}$
SL-KM	$k \in \{2, 3, \ldots, 50\}$; $\sigma \in \{0.1, 0.2, \ldots, 1\}$; $\lambda \in \{0.1, 0.2, \ldots, 10\}$
IK-KM	$\psi \in \{2, 4, 6, 8, 16, 24, 32, 48, 64, 80, 100, 200, 250, 500, 750, 1000, 2000, 2500\}$; $t = 100$; $\sigma \in \{0.10, 0.11, \ldots, 10\}$
GDKC	$\sigma = d \times \mu$; $\mu \in \{2^{-5}, 2^{-4}, \ldots, 2^5\}$; $t = 100$; $q \in \{0.05n, 0.1n, \ldots, 0.45n, 0.5n\}$; $s = 10,000$; $\varrho = 0.9$
IDKC	$\psi \in \{2, 4, 6, 8, 16, 24, 32, 48, 64, 80, 100, 200, 250, 500, 750, 1000, 2000, 2500\}$; $t = 100$; $q \in \{0.05n, 0.1n, \ldots, 0.45n, 0.5n\}$; $s = 10,000$; $\varrho = 0.9$



(a) Pathbased (b) Sprial

(c) Aggregation (d) AC (e) S3

Fig. A1. The scatter plots of 5 synthetic datasets.



(a) DP, NMI=0.74 (b) LC-DP, NMI=0.83 (c) IDKC, NMI=0.87

Fig. A2. Cluster centers (peaks) identified by three methods, DP, LC-DP and IDKC, on the 2-dimensional data shown in Fig. 1. (a), (b) and (c) show their best clustering results.

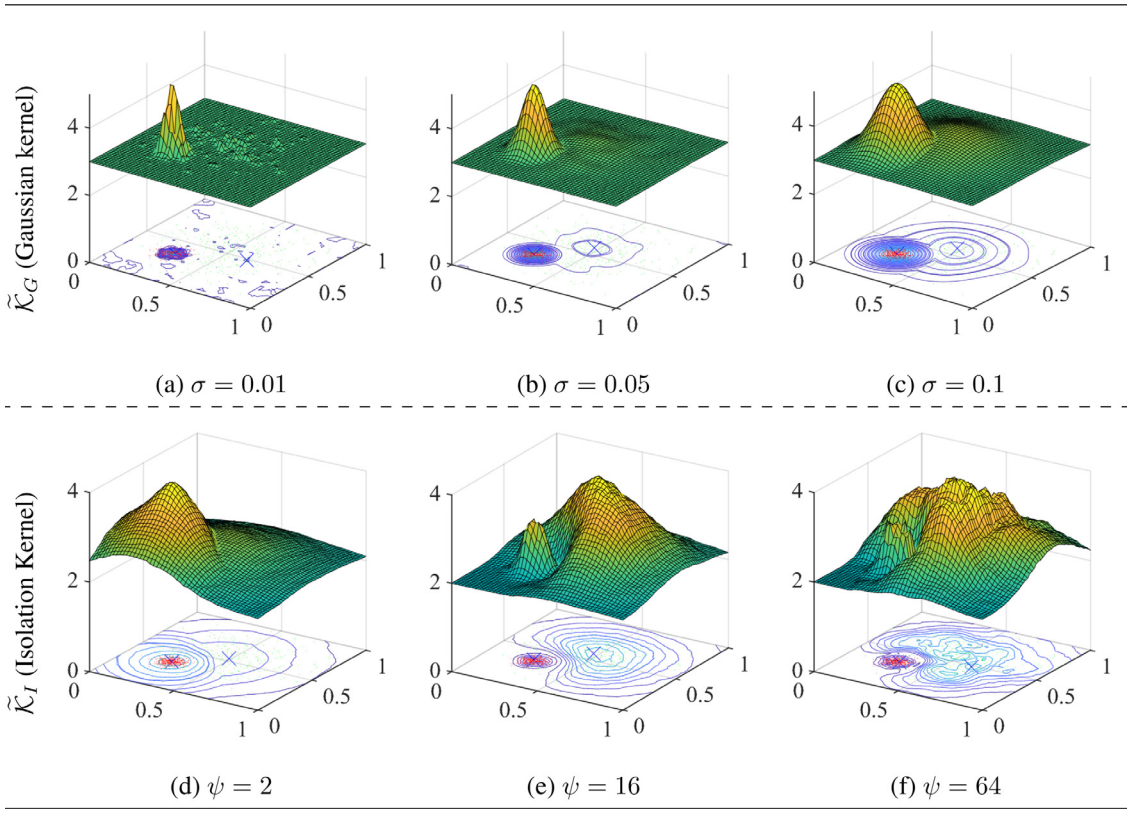


Fig. A3. Example $\widetilde{\mathcal{K}}_I$ and $\widetilde{\mathcal{K}}_G$ distributions based on the true clusters as shown in Fig. 1(a), with different parameter settings. The distribution $\widetilde{\mathcal{K}}$ is as given in Definition 2.

Table A.2
The effect of the clusters refinement in IDKC.

	Before refinement	After refinement		
	NMI	NMI	#points re-assigned	Time (%)
Pathbased	0.98	0.98	0	2.4%
AC	1.00	1.00	0	1.3%
S3	0.80	0.80	0	0.6%
Dermatology	0.95	0.95	0	2.4%
Libras	0.69	0.68	4	5.3%
LandCover	0.63	0.63	11	2.0%
Banknote	0.88	0.99	78	1.6%
Segment	0.72	0.72	40	0.3%
MNIST	0.82	0.82	0	0.1%

two clusters have hugely different densities, i.e., a smaller bandwidth found two local peaks in the dense cluster, while a large bandwidth found two peaks in the sparse cluster, as shown in Fig. A2(a), which produced its best NMI result. Yet, both LC-DP and IDKC correctly locate the two cluster peaks based on the local contrast calculations.

Table A.3 compares the proposed local contrast method with the two methods previously used in the density peaks clustering [20,30] to select density peaks.

Appendix C. K-modes and Laplacian K-modes

C.1. K-modes

K-Modes [15] was introduced as a way of extending K-means via mean-shift [14], i.e., instead of computing means in the M-step (in the EM procedure), weighted means are derived successively via a Gaussian kernel density estimator in the mean-shift algorithm to find the mode for each point. Then, the point is assigned to the mean which is closest to the mode of the point.

The objective function of K-Modes [15] is given as follows:

$$\underset{C_1,\dots,C_k}{\operatorname{argmax}} \sum_{j=1}^{k} \sum_{\mathbf{x}\in C_j} \Bbbk(\mathbf{x}, \mathbf{c}_j). \tag{C.1}$$

where each $\mathbf{c}_j = \frac{1}{|C_j|} \sum_{\mathbf{y}\in C_j} \mathbf{y}$ is the mean vector of cluster C_j; and $\Bbbk$ is Gaussian kernel.

Notice that the objective function is defined using means rather than modes; and the weighted means are used in the intermediate process (in mean-shift) [15]. The only hint that it is related to density is in the optimization method that employs mean-shift. In other words, this is an indirect way to find k modes.

One may view $\Bbbk(\mathbf{x}, \mathbf{c})$ as a degenerated version of Eq. (1):

$$\mathcal{K}_G(\delta(\mathbf{x}), \mathcal{P}_C) = \frac{1}{|C|} \sum_{\mathbf{y}\in C} \Bbbk(\mathbf{x}, \mathbf{y}).$$

Therefore, the objective function of K-Modes is exactly the same as used in IDKC, shown in Eq. (4).

C.2. Cluster definition of K-modes

Definition 4. Clusters $C_j, j = 1, \dots, k$ produced by K-Modes constitutes a Voronoi diagram in the input space, where each Voronoi cell has a centroid $\mathbf{c}_j$ and the boundary between cells is defined for $\mathbf{x} \in \mathbb{R}^d$ such that:

$$\Bbbk(\mathbf{x}, \mathbf{c}_j) = \Bbbk(\mathbf{x}, \mathbf{c}_j)\ \forall j \neq j.$$

Table A.3
Functions used to select the seeds for k clusters in DP, LC-DP and $\mathcal{K}LC$ (employed in the proposed IDKC clustering). Note that two key changes are made: (i) IDK is used instead of density; and (ii) $\Delta_{\mathcal{K}LC}(\mathbf{x})$ is used instead of $\Delta_f(\mathbf{x})$ or $\Delta_{LC}(\mathbf{x})$. Therefore, k points that have the largest $\mathcal{K}LC(\mathbf{x}) \times \Delta_{\mathcal{K}LC}$ are selected, instead of the largest in either $f(\mathbf{x}) \times \Delta_f(\mathbf{x})$ or $LC(\mathbf{x}) \times \Delta_{LC}(\mathbf{x})$. $N_q(\mathbf{x})$ is the set of the q nearest neighbors of $\mathbf{x}$. Note that we omitted the parameter q in the input of both LC and $\mathcal{K}LC$ for brevity. $\|\mathbf{x}-\mathbf{y}\|$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$.

DP	LC	$\mathcal{K}LC$
density $f(\mathbf{x}\|D)$	density $f(\mathbf{x}\|D)$ & $LC(\mathbf{x}) = \sum_{\mathbf{y}\in N_q(\mathbf{x})} I_{\{f(\mathbf{x}\|D)>f(\mathbf{y}\|D)\}}$	similarity $g(\mathbf{x}\|D) = \mathcal{K}(\delta(\mathbf{x}), \mathcal{P}_D)$ & $\mathcal{K}LC(\mathbf{x}) = \sum_{\mathbf{y}\in N_q(\mathbf{x})} I_{\{g(\mathbf{x}\|D)>g(\mathbf{y}\|D)\}}$
$\Delta_f(\mathbf{x}) = \min_{\mathbf{z}\in D,\ f(\mathbf{z})>f(\mathbf{x})} \|\mathbf{x}-\mathbf{z}\|$	$\Delta_{LC}(\mathbf{x}) = \min_{\mathbf{z}\in D,\ LC(\mathbf{z})>LC(\mathbf{x})} \|\mathbf{x}-\mathbf{z}\|$	$\Delta_{\mathcal{K}LC}(\mathbf{x}) = \min_{\mathbf{z}\in D,\ \mathcal{K}LC(\mathbf{z})>\mathcal{K}LC(\mathbf{x})} 1-\mathcal{K}(\delta(\mathbf{x}), \delta(\mathbf{z}))$
$f(\mathbf{x}) \times \Delta_f(\mathbf{x})$	$LC(\mathbf{x}) \times \Delta_{LC}(\mathbf{x})$	$\mathcal{K}LC(\mathbf{x}) \times \Delta_{\mathcal{K}LC}(\mathbf{x})$

Definition 4 states that the kind of clusters which can be discovered by kernel K-Modes is restricted to the structure of a Voronoi diagram in the input space.

K-Modes was motivated to resolve the fundamental problem of mean-shift [14,50] which associates each mode with a cluster. As the densities estimated by any density estimator can be bumpy, many local modes can be estimated in a true cluster. K-Modes resolves this problem "by separating the roles of cluster assignment and density. Each cluster has its own KDE, which can be multimodal, and the homotopy algorithm tends to select an important mode among these within each cluster" [15]. This solution is still in the same paradigm of using K-means-like algorithms, as many of its limitations still exist (e.g., random initialization, sensitive to outliers), and most importantly the cluster structure that can be discovered is limited to Voronoi diagram, as stated in Definition 4.

C.3. Laplacian K-modes and its scalable version

The objective function of Laplacian K-Modes (L-KM) [16] is:

$$\min_{\mathbf{Z},\mathbf{C}} \left\{ -\sum_{i=1}^{n}\sum_{j=1}^{k} z_{i,j}\mathbb{k}(\mathbf{x}_i, \mathbf{c}_j) + \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{\imath=1}^{n} w_{i,j} \parallel \mathbf{z}_i - \mathbf{z}_\imath \parallel^2 \right\} \quad \text{s.t.} \ \sum_{j=1}^{k} z_{i,j} = 1 \ \forall i, z_{i,j} \in [0,1], \tag{C.2}$$

where $\mathbf{Z}$ is the assignment $n \times k$ matrix having entries $z_{i,j}$; $\mathbf{C}$ is a matrix containing weighted means $\mathbf{c}_j,\ j = 1, \ldots, k$ (as computed in mean-shift); $w_{i,j}$ are entries in the affinity matrix $\mathbf{W}$ which is used to derive the graph Laplacian; and $\parallel \mathbf{x} - \mathbf{y} \parallel$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$.

Note that, ignoring the regularization (second) term, this objective function is equivalent to the objection function (C.1).

L-KM removes the structural constraints imposed by K-Modes (and kernel K-means) by using (a) soft assignment instead of hard assignment; and (b) $\mathbf{c}_j$ is a weighted mean instead of a mean. These relax Definition 4 to clusters that are a mutant of the Voronoi diagram.

However, it is unclear that how clusters discovered by L-KM can be defined with the Laplacian regularization.[14]

The key computational cost of L-KM is the need to store the affinity matrix for graph Laplacian. Scalable Laplacian K-Modes (SL-KM) [17] solves a concave-convex relaxation of the objective function (C.2) instead, by using an iterative bound optimization.

The SL-KM algorithm has two steps: (i) $\mathbf{z}$-updates reassign points to modes with the modes fixed, and (ii) mode-updates recompute modes with assignment variable $\mathbf{Z}$ fixed [17]. Rather than using mean-shift to update modes, the authors advocate an efficient alternative that updates modes as byproducts of the $\mathbf{z}$-updates.

SL-KM, like the original K-Modes and mean-shift, is a clustering method via density mode estimation. Unlike the original K-Modes or Laplacian K-Modes, it can scale up to large-scale datasets [17]. This is because SL-KM eliminates the need to store the kNN graph affinity matrix and each update of $\mathbf{z}$ corresponds to each point can be done independently in parallel.

SL-KM implementation details. SL-KM requires an affinity matrix to be sparse for efficient computation; thus kNN affinity is used. Then, Gaussian kernel is used to compute the density via KDE. In addition, SL-KM needs to run kMeans++ [51] to obtain the initial centers. There are three parameter settings: k is the number of clusters; $\mathtt{k}$ is used in kNN affinity, and λ is the regularization parameter. The squared bandwidth of Gaussian kernel is estimated using the average squared distance in kNN in each cluster.

However, contrary to the previous result presented by [17], we have found that SL-KM often performs worse than L-KM. For example, as shown in Table 3, (i) on each of the five artificial datasets, L-KM is able to identify the true clusters, but SL-KM fails to do so; and (ii) SL-KM is significantly worse than L-KM in terms of NMI, average over 22 datasets.

SL-KM performed better than L-KM on the few high-dimensional datasets used in [17]. This is consistent with our results in Table 3. But on many other low-dimensional datasets shown in Table 3, the reverse is true.

Appendix D. Isolation Kernel: implementation and data-dependent property

Isolation Kernel uses a space partitioning strategy to subdivide the data space into different partitions. The similarity between any two points, as measured by Isolation Kernel, is the expectation that the two points fall into the same partition [21,29]. To produce the data-dependent Isolation Kernel, the partitioning mechanism must produce large partitions in a sparse region and small partitions in a dense region [21,22,29]. This data-dependent property is the key that enables a distance/density-based clustering algorithm to detect clusters with varied densities [21].

Let $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a dataset sampled from an unknown probability density function $\mathbf{x}_i \sim F$. Moreover, let $\mathbb{H}_\psi(D)$ denote the set of all partitionings H admissible for the given dataset D, where each H covers the entire space of $\mathbb{R}^d$; and each of the ψ isolating partitions $\theta[\mathbf{z}] \in H$ isolates one data point $\mathbf{z}$ from the rest of the points in a random subset $\mathcal{D} \subset D$, and $|\mathcal{D}| = \psi$.

Definition 5. For any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Isolation Kernel of $\mathbf{x}$ and $\mathbf{y}$ wrt D is defined to be the expectation taken over the probability distribution on all partitionings $H \in \mathbb{H}_\psi(D)$ that both $\mathbf{x}$ and $\mathbf{y}$ fall into the same isolating partition $\theta[\mathbf{z}] \in H, \mathbf{z} \in \mathcal{D}$:

$$\mathbb{k}_I(\mathbf{x}, \mathbf{y}|D) = \mathbb{E}_{\mathbb{H}_\psi(D)}[\mathbb{1}(\mathbf{x}, \mathbf{y} \in \theta[\mathbf{z}] \mid \theta[\mathbf{z}] \in H)], \tag{D.1}$$

where $\mathbb{1}(\cdot)$ is an indicator function.

[14] This is true for all clustering algorithms which employ graph Laplacian, including spectral clustering, irrespective of hard or soft assignment.

In practice, the expectation is estimated via a finite number of partitionings $H_i, i = 1, \ldots, t$, where each H_i is created using a random sample $\mathcal{D}_i \subset D$:

$$\begin{aligned}\mathbb{k}_I(\mathbf{x}, \mathbf{y}|D) &= \frac{1}{t}\sum_{i=1}^{t} \mathbb{1}(\mathbf{x}, \mathbf{y} \in \theta \mid \theta \in H_i) \\ &= \frac{1}{t}\sum_{i=1}^{t}\sum_{\theta \in H_i} \mathbb{1}(\mathbf{x} \in \theta)\mathbb{1}(\mathbf{y} \in \theta), \end{aligned} \tag{D.2}$$

where θ is a shorthand for $\theta[\mathbf{z}]$.

ψ is the sharpness parameter; and it is the only parameter of the Isolation Kernel that needs to be tuned. The larger ψ is, the sharper the kernel distribution is. This corresponds to σ in the Gaussian kernel, i.e., the smaller σ is, the narrower the kernel distribution is.

Note that the number of partitionings t can be fixed to a large value to ensure the stability of the estimation; and t can usually be set to a default value.

Isolation Kernel admits different partitioning mechanisms. We use hyperspheres to partition the data space [22] in this paper as follows: given a subsample $\mathcal{D}$ of $\psi \geq 2$ points, each point $\mathbf{z} \in \mathcal{D}$ is the center of a hypersphere $\theta[\mathbf{z}]$ with the radius determined by the distance between $\mathbf{z}$ and its nearest neighbor in $\mathcal{D} \setminus \mathbf{z}$. Given a subsample $\mathcal{D} \subset D$, it produces ψ hyperspheres in the space. Given a test point $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}$ falls into either (a) inside a hypersphere, or (b) outside all hyperspheres, or (c) an overlapping region between two hyperspheres, where $\mathbf{x}$ is considered to be in the hypersphere whose center is the closest to $\mathbf{x}$.

The feature map of Isolation Kernel based on the above hyperspheres is given as follows:

Definition 6. Given $\mathcal{D}_i, i = 1, \ldots, t$, the feature mapping $\phi :\rightarrow \{0, 1\}^{t\times\psi}$ of $\mathbb{k}_I$ is a binary vector that represents the $|\mathcal{D}_i| = \psi$ partitions/hyperspheres in each partitioning $H_i, i = 1, \ldots, t$, such that the feature representing the hypersphere into which a test point falls has value 1 and other features have 0 values.

With the feature map, Eq. (D.2) can then be re-expressed as $\mathbb{k}_I(\mathbf{x}, \mathbf{y}|D) = \frac{1}{t}\langle\phi(\mathbf{x}|D), \phi(\mathbf{y}|D)\rangle$.

The data-dependent property of Isolation Kernel: two points in a sparse region are more similar than two points of equal inter-point distance in a dense region [21,22,29], is given in the following theorem:

Let $\mathcal{X}_S$ and $\mathcal{X}_T$ be sparse and dense regions of $\mathbb{R}^d$, respectively; and $\| \mathbf{x} - \mathbf{y} \|$ be the distance between $\mathbf{x}$ and $\mathbf{y}$.

Theorem 1. *$\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}_S$ and $\forall \mathbf{x}', \mathbf{y}' \in \mathcal{X}_T$ such that $\| \mathbf{x} - \mathbf{y} \| = \| \mathbf{x}' - \mathbf{y}' \|$, $\mathbb{k}_I$ satisfies the following condition:*

$$\mathbb{k}_I(\mathbf{x}, \mathbf{y}|D) > \mathbb{k}_I(\mathbf{x}', \mathbf{y}'|D). \tag{D.3}$$

Fig. A3 compares the $\widetilde{\mathcal{K}}$ distributions due to Isolation Kernel and Gaussian kernel, with different parameter settings. This shows that there exists some ψ of Isolation Kernel which enables $\widetilde{\mathcal{K}}_I$ to accentuate the sparse cluster. However, no σ setting can do the same for $\widetilde{\mathcal{K}}_G$. In fact, the sparse cluster becomes a part of the dense cluster in $\widetilde{\mathcal{K}}_G$, shown in Fig. A3(c).

References

[1] X. Wang, Application of weighted fuzzy clustering algorithm in urban economics development, in: International Conference on Big Data Analytics for Cyber-Physical-Systems, Springer, 2020, pp. 1698–1702.

[2] G. Gilam, E.M. Cramer, K.A. Webber, M.S. Ziadni, M.-C. Kao, S.C. Mackey, Classifying chronic pain using multidimensional pain-agnostic symptom assessments and clustering analysis, Sci. Adv. 7 (37) (2021).

[3] S. Saitta, P. Kripakaran, B. Raphael, I.F. Smith, Improving system identification using clustering, J. Comput. Civ. Eng. 22 (5) (2008) 292–302.

[4] N. Zeghidour, D. Grangier, Wavesplit: End-to-end speech separation by speaker clustering, IEEE/ACM Trans. Audio Speech Lang. Process. 29 (2021) 2840–2849.

[5] Y. He, X. Tang, J. Huang, J. Ren, H. Zhou, K. Chen, A. Liu, H. Shi, Z. Lin, Q. Li, A. Aditham, J. Ounadjela, E.I. Grody, J. Shu, J. Liu, X. Wang, ClusterMap for multi-scale clustering analysis of spatial gene expression, Nature Commun. 12 (1) (2021) http://dx.doi.org/10.1038/s41467-021-26044-x.

[6] F.D. Malliaros, M. Vazirgiannis, Clustering and community detection in directed networks: A survey, Phys. Rep. 533 (4) (2013) 95–142.

[7] D. Arunachalam, N. Kumar, Benefit-based consumer segmentation and performance evaluation of clustering approaches: An evidence of data-driven decision-making, Expert Syst. Appl. 111 (2018) 11–34.

[8] C.C. Aggarwal, C.K. Reddy, Data clustering, in: Algorithms and Applications, Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC, Londra, 2014.

[9] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, First Ed., Addison-Wesley Longman Publishing, Boston, MA, USA, 2005.

[10] D. Marin, M. Tang, I.B. Ayed, Y. Boykov, Kernel clustering: Density biases and solutions, IEEE Trans. Pattern Anal. Mach. Intell. 41 (1) (2019) 136–147.

[11] I.S. Dhillon, Y. Guan, B. Kulis, Kernel K-means: Spectral clustering and normalized cuts, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 551–556.

[12] S. Wang, A. Gittens, M.W. Mahoney, Scalable kernel K-means clustering with Nyström approximation: Relative-error bounds, J. Mach. Learn. Res. 20 (12) (2019) 1–49.

[13] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A K-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Stat.) 28 (1) (1979) 100–108.

[14] K. Fukunaga, L.D. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, IEEE Trans. Inform. Theory 21 (1) (1975) 32–40.

[15] M.Á. Carreira-Perpiñán, W. Wang, The K-modes algorithm for clustering, 2013, CoRR abs/1304.6478.

[16] W. Wang, M.Á. Carreira-Perpiñán, The Laplacian K-modes algorithm for clustering, 2014, CoRR abs/1406.3895.

[17] I. Ziko, E. Granger, I. Ben Ayed, Scalable Laplacian K-modes, in: Advances in Neural Information Processing Systems, Vol. 31, 2018.

[18] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1) (1977) 1–38.

[19] M.A. Carreira-Perpinan, Gaussian mean-shift is an EM algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 29 (5) (2007) 767–776.

[20] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

[21] X. Qin, K.M. Ting, Y. Zhu, V.C.S. Lee, Nearest-neighbour-induced isolation similarity and its impact on density-based clustering, in: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, 2019 pp. 4755–4762.

[22] K.M. Ting, B.-C. Xu, T. Washio, Z.-H. Zhou, Isolation distributional kernel: A new tool for kernel based anomaly detection, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 198–206.

[23] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, 2001, pp. 849–856.

[24] D. Paul, S. Chakraborty, S. Das, J. Xu, Kernel k-means, by all means: Algorithms and strong consistency, 2020, CoRR abs/2011.06461.

[25] F.R. Bach, M.I. Jordan, Learning spectral clustering, in: S. Thrun, L.K. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems,Vol. 16, 2004, pp. 305–312.

[26] M. Tang, D. Marin, I. Ben Ayed, Y. Boykov, Kernel cuts: Kernel and spectral clustering meet regularization, Int. J. Comput. Vis. 127 (5) (2019) 477–511.

[27] C.K.I. Williams, M. Seeger, Using the nyström method to speed up kernel machines, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems, Vol. 13, 2001, pp. 682–688.

[28] K. Muandet, K. Fukumizu, B. Sriperumbudur, B. Schölkopf, Kernel mean embedding of distributions: A review and beyond, Found. Trends Mach. Learn. 10 (1–2) (2017) 1–141.

[29] K.M. Ting, Y. Zhu, Z.-H. Zhou, Isolation kernel and its effect on SVM, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 2329–2337.

[30] B. Chen, K.M. Ting, T. Washio, Y. Zhu, Local contrast as an effective means to robust clustering against varying densities, Mach. Learn. 107 (8) (2018) 1621–1645.

[31] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E.Y. Chang, Parallel spectral clustering in distributed systems, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2010) 568–586.

[32] Z. Kang, X.Z. Zhiping Lin, W. Xu, Structured graph learning for scalable subspace clustering: From single view to multiview, IEEE Trans. Cybern. (2021).

[33] D.U. Pizzagalli, S.F. Gonzalez, R. Krause, A trainable clustering algorithm based on shortest paths from density peaks, Sci. Adv. 5 (10) (2019) eaax3770.
[34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.
[35] L. Bai, X. Cheng, J. Liang, H. Shen, Y. Guo, Fast density clustering strategies based on the k-means algorithm, Pattern Recognit. 71 (2017) 375–386.
[36] Y. Zhu, K.M. Ting, Y. Jin, M. Angelova, Hierarchical clustering that takes advantage of both density-peak and density-connectivity, Inf. Syst. 103 (2022) 101871.
[37] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (Dec) (2002) 583–617.
[38] M. Li, J.T.-Y. Kwok, B. Lü, Making large-scale nyström approximation possible, in: Proceedings of the 27th International Conference on Machine Learning, 2010, p. 631.
[39] H. Chang, D.-Y. Yeung, Robust path-based spectral clustering, Pattern Recognit. 41 (1) (2008) 191–203.
[40] I.S. Dhillon, Y. Guan, B. Kulis, Kernel k-means: Spectral clustering and normalized cuts, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 551–556.
[41] Y. Zhu, K.M. Ting, Improving the effectiveness and efficiency of stochastic neighbour embedding with isolation kernel, J. Artificial Intelligence Res. 71 (2021) 667–695.
[42] M. Muja, D.G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: Proceedings of the Fourth International Conference on Computer Vision Theory and Applications, 2009 pp. 331–340.
[43] R. Szeliski, Computer Vision: Algorithms and Applications, Springer Science & Business Media, 2010.
[44] Y. Li, P. Hu, Z. Liu, D. Peng, J.T. Zhou, X. Peng, Contrastive clustering, in: Proceeginds of the Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI, 2021.
[45] J. Chang, L. Wang, G. Meng, S. Xiang, C. Pan, Deep adaptive image clustering, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5879–5887.
[46] J. Wu, K. Long, F. Wang, C. Qian, C. Li, Z. Lin, H. Zha, Deep comprehensive correlation mining for image clustering, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8150–8159.
[47] Y. Tao, K. Takagi, K. Nakata, Clustering-friendly representation learning via instance discrimination and feature decorrelation, 2021, arXiv preprint arXiv:2106.00131.
[48] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017, URL http://archive.ics.uci.edu/ml.
[49] S. Aksoy, R.M. Haralick, Feature normalization and likelihood-based similarity measures for image retrieval, Pattern Recognit. Lett. 22 (5) (2001) 563–582.
[50] M.Á. Carreira-Perpiñán, A review of mean-shift algorithms for clustering, 2015, CoRR abs/1503.00687.
[51] S. Vassilvitskii, D. Arthur, K-means++: The advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 1027–1035.