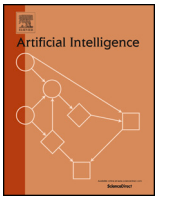
Artificial Intelligence 350 (2026) 104440

Contents lists available at ScienceDirect

Artificial Intelligence

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



Kernel-bounded clustering: Achieving the objective of spectral clustering without eigendecomposition

Hang Zhang [a,b,1], Kai Ming Ting [a,b,*,1], Ye Zhu [c]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, 210023, China
[b] School of Artificial Intelligence, Nanjing University, Nanjing, 210023, China
[c] School of IT, Deakin University, Geelong, 3125, Australia

ARTICLE INFO

Keywords:
Spectral clustering
Distributional kernel
Eigendecomposition

ABSTRACT

The research on spectral clustering (SC) has thus far been pursued on the same track using the same tool of eigendecomposition of a matrix since the idea was first introduced in 1973. Despite its successes, SC has been identified to have fundamental limitations that prevent SC from discovering certain types of clusters, and SC has slow runtime. We offer an alternative path that does not involve the eigendecomposition, and, more broadly, it uses no optimization. The proposed new Kernel-Bounded Clustering (KBC) is a complete metamorphosis in 50 years of research in SC in view of the fact that KBC achieves the same objective of SC without eigendecomposition or optimization. We evaluated KBC on the datasets that have been used to demonstrate the fundamental limitations of SC, genome-wide expression data, large image datasets and many commonly used real-world benchmark datasets. KBC produced better quality clusters than various variants of SC, and it ran six orders of magnitude faster than the traditional SC on a set of 5 million data points.

1. Introduction

Clustering is a fundamental problem in data mining and machine learning, whose purpose is to partition a given dataset into several disjoint groups. One of the most commonly used algorithms is k-means clustering [1]. The well-understood disadvantage of k-means is that it does not work on clusters of non-globular shape, varying sizes and densities. Two existing solutions are kernel k-means and spectral clustering [2] which still employ k-means to perform the clustering.

Spectral clustering (SC) is motivated to perform graph clustering via minimum cut using the tool of eigendecomposition of a matrix [3,4]. Since the introduction of the theory of eigendecomposition in 1973 [4], eigendecomposition has been the unquestionable tool in SC. Even when the fundamental limitations of SC were identified [5], the ability of eigendecomposition to produce a good clustering outcome was never contested. Instead, the efforts have been devoted to finding a better objective criterion or/and similarity measure for the eigendecomposition [5–11].

SC, that centers around the eigendecomposition approach, has the following issues in achieving a given objective of the minimum cut problem:

* Corresponding author at: State Key Laboratory for Novel Software Technology, Nanjing University, 210023, China.
E-mail addresses: zhanghang@lamda.nju.edu.cn (H. Zhang), tingkm@nju.edu.cn (K.M. Ting), ye.zhu@ieee.org (Y. Zhu).
[1] Both authors contributed equally to the paper.

https://doi.org/10.1016/j.artint.2025.104440
Received 5 September 2024; Received in revised form 30 September 2025; Accepted 10 October 2025

Available online 15 October 2025

- SC approximates the original normalized minimum or ratio cut criterion by relaxing the discrete optimization problem to a continuous optimization one [12]. This approximation method raises two important questions. First, is this a good approximation of the original problem? Second, can the quality of such an approximate solution be guaranteed with respect to the exact solution? No answers to these questions are available thus far. In fact, it has been shown that the difference in quality can be arbitrarily large in some scenarios [13]. The approximation method is popular not because it produces fine solutions, but rather because it translates the complex problem into an easy-to-solve linear algebra problem [12].
- While the solution to the original problem offers to perform clustering, the approximate solution (of the eigendecomposition) requires a reverse translation from the continuous solution eigenvectors to discrete ones in order to produce a clustering outcome. Simple heuristics to do this translation often do not work well [12]. As a convenient means, most SC algorithms opt to use the eigenvectors as the transformed features of the original dataset, and then employ k-means to perform the actual clustering. This effectively repurposes the original aim of the eigendecomposition from clustering to feature transformation only. In other words, SC conducts feature transformation only, and the actual clustering is performed by an existing algorithm such as k-means.

Recent research on spectral clustering focuses on its efficient improvement so that it is applicable to large-scale data. The landmark-based method first obtains p landmark points from the given dataset of n points to construct a similarity matrix between these p points and n points. Then, it performs eigendecomposition on this $n \times p$ bipartite graph to accelerate spectral clustering [10,14]. GBSC [15] constructs a similarity matrix for the p landmark points only. This greatly reduces the runtime of the eigendecomposition process, but its clustering outcome is poor. Macgregor [16] employs a more efficient power method, instead of the spectral decomposition, to approximate the eigenvectors. All these are approximation methods that trade off accuracy for efficiency.

From a broader standpoint, SC has been investigated from four perspectives, i.e., graph cut [13], random walks [17], perturbation theory [18], and stochastic block model [19]. Here, we offer the fifth perspective: clusters as samples generated from different distributions, and the clustering process is based on the similarity between a point and a cluster/distribution, measured using a distributional kernel [20,21].

SC following the eigendecomposition approach has three major shortcomings. First, although there are many variants of SC proposed in the last five decades, they still suffer from the fundamental limitations that prevent them from finding certain types of clusters [5]. Second, they have high computational costs. Third, the type of clusters produced from SC (with any of the four existing perspectives) is unclear as no cluster definition has been provided in the literature thus far. In this paper, we establish that eigendecomposition is the root cause of the above three shortcomings.

The above findings prompted us to establish the fifth perspective of SC that leads to a new kernel-based approach with four significant advances over the eigendecomposition approach. First and foremost, we show that the objective of SC can be achieved, without eigendecomposition of a matrix, via a distributional kernel. Second, we are the first to formally define a cluster based on the new self-similarity criterion expressed in terms of a distributional kernel, independent of the means to find the clusters. Third, the cluster definition inspires a new Kernel-Bounded Clustering (KBC) algorithm. For the first time, KBC enables any existing criterion to be achieved in *linear time without an optimization algorithm*. Yet, its clustering outcome is still better than many variants of the optimization-based quadratic-time SC. The cluster-definition-inspired clustering algorithm is simpler and more direct than SC because it does not need eigendecomposition and an existing clustering algorithm such as k-means. Fourth, KBC has no fundamental limitations of SC.

In a nutshell, SC is method-driven where the must-use method is eigendecomposition, and the kind of clusters, that SC discovers, is unclear. In contrast, KBC is inspired by a cluster definition, where the algorithm is created to find well-defined clusters.

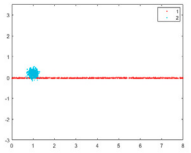
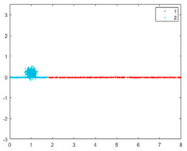
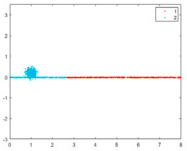
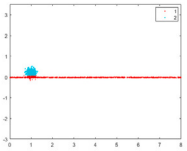
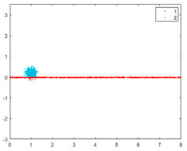
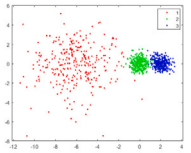
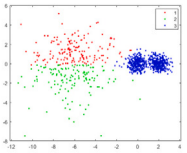
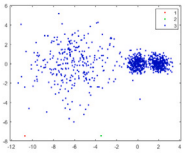
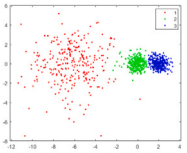
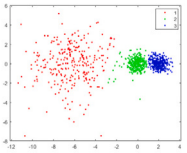
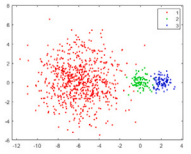
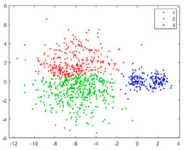
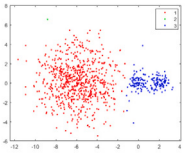
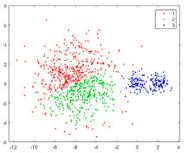
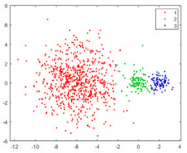
Significance: The proposed Kernel-Bounded Clustering is the first algorithm to achieve the same objective of Spectral Clustering with the following three unique characteristics. First, it uses no eigendecomposition, which has been the unquestionable method for Spectral Clustering since the theory of eigendecomposition was introduced 50 years ago. Second, it produces better clustering outcomes because it does not have the fundamental limitations of Spectral Clustering. Third, it is a linear-time algorithm that runs orders of magnitude faster than Spectral Clustering.

The rest of the paper is organized as follows. Two fundamental limitations of SC are described in the next section. Section 3 presents a new criterion of SC and a new expression of all three criteria of SC in terms of the distributional kernel. Section 4 provides a cluster definition based on the distributional kernel, and Section 5 introduces the cluster-definition-inspired clustering algorithm called Kernel Bounded Clustering (KBC). Section 6 shows that the proposed KBC has no fundamental limitations of SC. We have conducted a comprehensive assessment which consists of three sets of empirical evaluation, and they are reported in the following three sections. A discussion of various issues and the conclusions are provided in the last two sections.

2. Fundamental limitations of spectral clustering

We use the same three examples as used in [5] to illustrate the two limitations of SC. The first fundamental limitation, as attributed by [5], is that using local information provided by a similarity measure in the optimization objective criterion is not sufficient to produce a good global clustering outcome. An example is provided on the G-Strip dataset shown in Table 1, where SC using any of the commonly used criteria called normalized cut ($NCut$) [7,8] and ratio cut ($RCut$) [9] chose to have a cut along the thin strip over a cut along the overlap between the Gaussian ball and the rectangular strip. This is because the latter has a wider cut than the former (the width of the strip). These criteria, which rely on local information provided the similarity matrix, are thus said to be unsuitable for general data clustering [5].

Table 1
Examples of fundamental limitations of SC first presented by [5] are shown in the first column, where SC fails to detect the appropriate clusters using either $NCut$ or $RCut$ (shown in the second the third columns). The G-Strip dataset has one Gaussian ball and a rectangular-strip cluster; the Same-Size dataset has three clusters with the same number of data points but different densities; and the High-Low dataset has one large cluster with 80% of the data points and two small clusters, each having 10% data points. The clustering outcome in terms of NMI (normalized mutual information [22]) is shown at the bottom of each plot.

Dataset	Ground Truth	$NCut$-SC	$RCut$-SC	NSS-SC	$NCut$-KBC
G-Strip		 0.57	 0.44	 0.71	 0.86
Same-Size		 0.63	 0.02	 0.91	 0.89
High-Low		 0.58	 0.80	 0.59	 0.90

The second fundamental limitation is the inability of SC to detect clusters of varying sizes and densities. This is the same limitation of k-means. This was previously identified to be due to the use of a data-independent measure such as Gaussian kernel, and an adaptive Gaussian kernel was suggested [6]. However, Nadler and Galun [5] have used the High-Low dataset shown in Table 1 to demonstrate that the adaptive Gaussian kernel is not good enough in general. Note that SC using either adaptive Gaussian kernel or Gaussian kernel produces similar results as shown in the $NCut$-SC column in Table 1.

Although many spectral clustering methods have been proposed recently, they can be broadly categorized into three categories. The first category derives a sparse similarity matrix ($n \times n$) from the similarity matrix [23,10,24–26]. The second extracts a smaller similarity matrix ($a \times b$, where $a, b \leq n$) from the $n \times n$ similarity matrix [15,27,28]. Both of these facilitate to more efficiently embed the data into the spectral space. The third category employs a means different from eigendecomposition, such as the power methods, to embed data into a low-dimensional space [16]. They all focus on accelerating the original spectral clustering algorithm without addressing the two above-mentioned fundamental limitations.

3. Existing and new criteria of spectral clustering

Given a dataset D of n points $\mathbf{x} \in \mathbb{R}^d$, the aim of a clustering task is to produce a set of k clusters, i.e., $\mathbb{C} = \{C_1, \ldots, C_k\}$, which satisfies some criterion.

Treating D as the set of vertices in an undirected graph $\mathcal{G}$ having edges with weights defined by the similarity between all pairs of points $\mathbf{x}$ and $\mathbf{y}$ in D, one can solve the problem with a graph clustering method such as SC which aims to minimize the total weight of the edges eliminated due to the cuts on the graph in order to produce k subgraphs.

The minimum cut criterion [29], on which SC is based, minimizes the sum of the similarity between cluster C and $D \setminus C$ (i.e., the complement of C) over all clusters:

$$Cut(\mathcal{G}) := \sum_{C \in \mathbb{C}} W(C, D \setminus C),$$

where $W(A, B) := \sum_{\mathbf{x} \in A, \mathbf{y} \in B} \kappa(\mathbf{x}, \mathbf{y})$ is the sum of the weights/similarities between two sets of points A and B; and $\kappa(\mathbf{x}, \mathbf{y})$ is a kernel measuring the similarity between $\mathbf{x}$ and $\mathbf{y}$.

Here we show that the cut criterion can be re-expressed as:

$$\sum_{C \in \mathbb{C}} W(C, D \setminus C) = \sum_{C \in \mathbb{C}} W(C, D) - \sum_{C \in \mathbb{C}} W(C, C) = W(D, D) - \sum_{C \in \mathbb{C}} W(C, C)$$

Because $W(D, D)$ is a constant for the given dataset D, we have:

Table 2
Cut-associated criteria vs the new SS-associated criteria, and traditional expressions using W vs the new expressions using K.

Criterion	Expression using W	New expression using K
$\min_{\mathbb{C}} Cut(\mathcal{G})$	$\min_{\mathbb{C}} \sum_{C\in\mathbb{C}} W(C, D\setminus C)$	$\max_{\mathbb{C}} \sum_{C\in\mathbb{C}} K(\mathcal{P}_C,\mathcal{P}_C)\times\lvert C\rvert^2 - K(\mathcal{P}_C,\mathcal{P}_D)\times\lvert C\rvert\times\lvert D\rvert$
$\min_{\mathbb{C}} NCut(\mathcal{G})$	$\min_{\mathbb{C}} \sum_{C\in\mathbb{C}} \frac{W(C, D\setminus C)}{W(C,D)}$	$\max_{\mathbb{C}} \sum_{C\in\mathbb{C}} \frac{K(\mathcal{P}_C,\mathcal{P}_C)\times\lvert C\rvert}{K(\mathcal{P}_C,\mathcal{P}_D)\times\lvert D\rvert}$
$\min_{\mathbb{C}} RCut(\mathcal{G})$	$\min_{\mathbb{C}} \sum_{C\in\mathbb{C}} \frac{W(C, D\setminus C)}{\lvert C\rvert}$	$\max_{\mathbb{C}} \sum_{C\in\mathbb{C}} K(\mathcal{P}_C,\mathcal{P}_C)\times\lvert C\rvert - K(\mathcal{P}_C,\mathcal{P}_D)\times\lvert D\rvert$
$\max_{\mathbb{C}} SS(\mathcal{G})$	$\max_{\mathbb{C}} \sum_{C\in\mathbb{C}} W(C,C)$	$\max_{\mathbb{C}} \sum_{C\in\mathbb{C}} K(\mathcal{P}_C,\mathcal{P}_C)\times\lvert C\rvert^2$
$\max_{\mathbb{C}} NSS(\mathcal{G})$	$\max_{\mathbb{C}} \sum_{C\in\mathbb{C}} \frac{W(C,C)}{\lvert C\rvert}$	$\max_{\mathbb{C}} \sum_{C\in\mathbb{C}} K(\mathcal{P}_C,\mathcal{P}_C)\times\lvert C\rvert$

$$\min_{\mathbb{C}} Cut(\mathcal{G}) \Leftrightarrow \max_{\mathbb{C}} SS(\mathcal{G}) := \max_{\mathbb{C}} \sum_{C\in\mathbb{C}} W(C,C)$$

In other words, minimizing the cuts of all pairs of C and $D\setminus C$ to produce the clusters in $\mathbb{C}$ is equivalent to maximizing the total self-similarity (denoted as SS) of all clusters in $\mathbb{C}$. Conceptually, the latter is a much simpler proposition as one needs to consider individual clusters only while maximizing the total self-similarity.

3.1. Express in terms of distributional kernel

Let S and T be two nonempty datasets drawn from probability distributions $\mathcal{P}_S$ and $\mathcal{P}_T$, respectively. The empirical estimation of a distributional kernel K via kernel mean embedding [30,20] on $\mathcal{P}_S$ and $\mathcal{P}_T$ is given as:

$\hat{K}(\mathcal{P}_S,\mathcal{P}_T) = \frac{1}{|S||T|}\sum_{\mathbf{x}\in S,\mathbf{y}\in T}\kappa(\mathbf{x},\mathbf{y})$, which can be re-expressed as $\hat{K}(\mathcal{P}_S,\mathcal{P}_T) = \frac{1}{|S||T|}W(S,T)$. For brevity, we have used $K(\cdot,\cdot)$ to denote $\hat{K}(\cdot,\cdot)$ in the rest of this paper.

Table 2 provides a comparison between two dimensions: *Cut*-associated versus SS-associated criteria, and W versus K expressions. It includes the two commonly used normalized criteria, i.e., normalized cut ($NCut$) [7,8] and ratio cut ($RCut$) [9]; and a new normalized self-similarity (NSS).

The SS-associated criteria are 'cleaner' and simpler than the *Cut*-associated criteria using either the W or K expression.

3.2. New interpretations of the three normalized criteria in terms of K expressions

In terms of the K expressions shown in Table 2, a comparison among the three normalized criteria is instructive.

The NSS criterion, $\max\sum_{C\in\mathbb{C}} K(\mathcal{P}_C,\mathcal{P}_C)|C|$, demands clusters to have high self similarities as well as large cluster sizes.

Yet, both the $NCut$ and $RCut$ criteria require an auxiliary condition in addition to the NSS criterion: both require that each cluster to be less similar to (or further away from) the center of the entire dataset, though the means to achieve this requirement is different—via either division or subtraction.

4. Similarity score and cluster definition based on distributional kernel K

Here we first define a new similarity score as a means to estimate the data distribution of a dataset. Then we provide a cluster definition based on this distribution.

The similarity score based on distributional kernel K w.r.t. a given set of clusters is defined as follows:

Definition 1. Given a dataset D, clusters $C_i, i = 1\dots,k$, identified by a clustering algorithm and the distributional kernel K, the similarity score $\widetilde{K}(\mathbf{x})$ w.r.t. the k clusters for all $\mathbf{x}\in\mathbb{R}^d$ is defined as:

$$\widetilde{K}(\mathbf{x}|C_i,\ i=1,\dots,k) = \max_i K(\delta(\mathbf{x}),\mathcal{P}_{C_i}).$$

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

The similarity scores for all $\mathbf{x}\in\mathbb{R}^d$ can be used to compare the clustering outcomes produced by different clustering algorithms (see an example in Fig. 2 in Section 6).

A new cluster definition based on distributional kernel K is given as follows:

Definition 2. The set of clusters $\mathbb{C} = \{C_1,\dots,C_k\}$ constitutes a Voronoi diagram in the feature space $\widehat{\phi}$ of K, and the boundary between any two of the k Voronoi cells, representing the k clusters, is defined for $\mathbf{x}\in\mathbb{R}^d$ as follows:

$$K(\delta(\mathbf{x}),\mathcal{P}_{C_i}) = K(\delta(\mathbf{x}),\mathcal{P}_{C_j}) \Leftrightarrow \left\langle \phi(\mathbf{x}),\widehat{\phi}(\mathcal{P}_{C_i})\right\rangle = \left\langle \phi(\mathbf{x}),\widehat{\phi}(\mathcal{P}_{C_j})\right\rangle,\ \forall i\neq j$$

where the kernel mean map of K is $\widehat{\phi}(\mathcal{P}_C) = \frac{1}{|C|}\sum_{\mathbf{y}\in C}\phi(\mathbf{y})$; $\phi(\cdot)$ is the feature map of a point-to-point kernel κ; and $\widehat{\phi}(\delta(\mathbf{x})) = \phi(\mathbf{x})$.

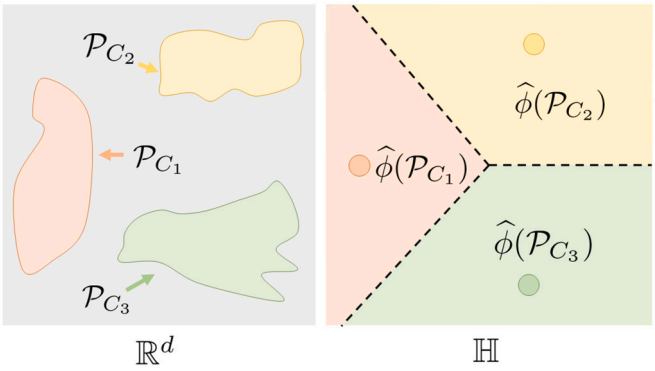


Fig. 1. An illustration of clusters in the input space $\mathbb{R}^d$ and the reproducing kernel Hilbert space (RKHS) $\mathbb{H}$ of K, given in Definition 2. Each cluster, as a distribution, in the input space is represented as a point in RKHS.

An illustration of clusters in the input space $\mathbb{R}^d$ and the reproducing kernel Hilbert space $\mathbb{H}$ induced by the feature map $\widehat{\phi}$ of K is shown in Fig. 1.

The above cluster definition is interesting in two aspects. First, it is independent of the algorithm to produce the clusters. Second, it enables a means to produce a set of clusters $\mathbb{C}$ which satisfies the following criterion of optimization:

$$\sum_{C\in\mathbb{C}}\sum_{\mathbf{x}\in C} K(\delta(\mathbf{x}),\mathcal{P}_C) = \sum_{C\in\mathbb{C}} K(\mathcal{P}_C,\mathcal{P}_C) * |C|. \tag{1}$$

Note that the above criterion is the normalized self-similarity (NSS) criterion, revealed in the last section.

We show in the next section that the above criterion of optimization can be achieved without using an optimization procedure.

5. Cluster-definition-inspired clustering algorithm: kernel bounded clustering

Rather than typically relying on an optimization procedure, the idea inspired by Definition 2 is to achieve an objective of SC in two key steps, exploiting the fact that the clusters form a Voronoi diagram in the feature space of K.

The first step is to identify the k initial cluster cores G_i of the Voronoi diagram from D.

The second step is to assign every point in D to its most similar initial cluster core, according to the Voronoi diagram:

$$\text{For } j = 1,\dots,k : C_j = \{\mathbf{x}\in D \mid \operatorname*{argmax}_{i\in[1,k]} K(\delta(\mathbf{x}),\mathcal{P}_{G_i}) = j\}.$$

Note that the point assignment is completed in one iteration, i.e., once a point is assigned to a cluster, it does not need to be reassigned. The ability to complete the point assignment in one iteration relies on a set of fitting initial cluster cores G that encapsulates the 'core' of each cluster, defined based on a kernel κ, as follows:

$$G = \{\mathbf{x},\mathbf{y}\in D \mid \text{there exists a chain: } \mathbf{z}_1,\mathbf{z}_2,\cdots,\mathbf{z}_q;\mathbf{z}_1=\mathbf{x},\mathbf{z}_q=\mathbf{y},\forall i\ \kappa(\mathbf{z}_i,\mathbf{z}_{i+1})>\tau\} \tag{2}$$

G is equivalent to a connected component after removing those edges whose weights are less than τ, and we use the standard connected-component function called 'conncomp' function in MATLAB to find it. Note that each core can be determined pretty reliably[2] using a random subset $D_s \subset D$ because the points $\mathbf{x}$ and $\mathbf{y}$ within a core of a cluster have the highest similarity, i.e., $\kappa(\mathbf{x},\mathbf{y}) > \tau$. The threshold parameter τ plays a critical role in cluster formation. When τ is set too low, it results in fewer connected components than the desired k clusters, necessitating an increase in τ to promote additional cluster formation. Conversely, when τ is set too high, it generates an excessive number of small clusters, requiring a decrease in τ to facilitate cluster merging. While the algorithm aims to identify k primary clusters, the total number of connected components may exceed k, as noise points and outliers often form separate, smaller clusters. Finally, the k largest clusters are selected based on the point count, effectively filtering out these noise-induced clusters. The probability of a tie is very low and we have never encountered one during our experiments.

The algorithm is called Kernel Bounded Clustering or KBC because the above bound on kernel κ is used and the boundaries between clusters are determined by the distributional kernel K (derived from κ). Algorithm 1 shows the full procedure.

5.1. Conceptual comparison of SC and KBC

Table 3 provides a summary of the two types of spectral clustering (SC): (i) stochastic block model [19] is a generative model approach, and (ii) graph cut aims to find the minimum cut of a given graph [13].

[2] This can be achieved as long as the sample is a representative subset of the entire dataset. See the details in Section 6.

Algorithm 1 Kernel Bounded Clustering.

Input: D - dataset, k - number of clusters, s - sample size, τ - similarity threshold
Output: $\mathbb{C} = \{C_1, \dots, C_k\}$
Step 1: Find the k initial clusters
From $D_s \subset D$, find the largest k initial clusters G_j via kernel κ as follows:
$G_j = \{\mathbf{x}, \mathbf{y} \in D \mid$ there exists a chain: $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_q; \mathbf{z}_1 = \mathbf{x}, \mathbf{z}_q = \mathbf{y}, \forall i\ \kappa(\mathbf{z}_i, \mathbf{z}_{i+1}) > \tau\}, \forall j \in [1, k]$
if the number of clusters in $\{G_1, G_2, \dots, G_j\} < k$ **then**
Assert 'Parameter τ is set too small !' and Exit
end if
Step 2: Assignment of data points to clusters
$C_j = \{\mathbf{x} \in D \mid \underset{i \in [1,k]}{\operatorname{argmax}} K(\delta(\mathbf{x}), \mathcal{P}_{G_i}) = j\}, \forall_{j \in [1,k]}$
Step 3: Refine $\mathbb{C} = \{C_1, \dots, C_k\}$ to improve the objective: $\max_{\mathbb{C}} \sum_{C \in \mathbb{C}} \sum_{\mathbf{x} \in C} K(\delta(\mathbf{x}), \mathcal{P}_C)$.
return $\mathbb{C}$

Table 3
Two types of spectral clustering (via eigendecomposition) versus the proposed kernel-bounded clustering.

Methodology	Spectral Clustering (eigendecomposition)		Kernel-bounded Clustering
Motivation	Stochastic Block model	$MinCut$	Self-Similarity (SS)
Matrix type	Adjacency Matrix	Laplacian Matrix	kernel (aka Adjacency Matrix)
Normalized criterion	—	$NCut$ & $RCut$	NSS & $NCut$ & $RCut$
	Normalized eigenvectors	—	—
Relaxed Optimization	yes	yes	not applicable
Clustering Algorithm	Gaussian mixture model	k-means	not required

The methodology employed in these two types of SC is the same, i.e., eigendecomposition of a matrix related to similarity between points. The key difference is the type of matrix used: the former uses adjacency matrix; and the latter uses Laplacian matrix. In order to produce a balanced cut, two possible normalized criteria for graph cut are the normalized cut ($NCut$) and the ratio cut ($RCut$). Note that there is no explicit normalized criterion in the stochastic block model; instead, it normalizes the eigenvectors after the eigendecomposition process. Either type of spectral clustering, which employs eigendecomposition, does not produce a clustering outcome. It is effectively a feature transformation method and requires a clustering algorithm to perform the final clustering.

The proposed kernel-bounded clustering (KBC), shown in the last column in Table 3, does not employ eigendecomposition at all, and it is not an optimization algorithm. It is motivated by the self-similarity (SS) criterion which is equivalent to $MinCut$, as stated in Section 3. Yet, it relies on a distributional kernel, and at its core, it measures similarity between points via a kernel (equivalent to the use of an adjacency matrix). KBC uses a normalized criterion called NSS, as stated in Section 4. Unlike the two types of spectral clustering that employ k-means to perform the actual clustering, KBC produces a clustering outcome satisfying a selected criterion directly, without optimization.

Note that although KBC was motivated by the SS criterion and its cluster definition, KBC is not restricted to the SS or NSS criterion only. KBC can employ the two existing criteria $NCut$ and $RCut$ used in SC, without using the Laplacian matrix, but via the K expressions shown in Table 2 (see Appendix A). Similarly, the NSS criterion can be achieved via eigendecomposition if required. In other words, either SC or KBC can apply any of the three normalized criteria specified in Table 2.

5.2. Time complexities

SC has $O(n^2)$ time complexity, mainly because of the computationally expensive process of eigendecomposition [12].

KBC has two steps which have a total cost of $O(s^2 + nkt\psi)$, where t is the number of partitionings and ψ is the number of partitions in each partitioning used to create an Isolation Kernel[3] [31]. As shown in Algorithm 1, s is the data size of the subset used in step 1, n is the given dataset size, and k is the number of clusters. The first term can be ignored for a large dataset because $s \ll n$.

6. The root cause of the fundamental limitations of SC, and why KBC has no such limitations

Recall that SC has two fundamental limitations [5], as illustrated in Section 2. Although different criteria of optimization have been investigated, the fundamental limitations persist.

Here we use one limitation, i.e., the inability of SC to detect clusters of varying sizes and densities, to highlight its root cause.

[3] Isolation Kernel is a data-dependent kernel which is derived from a given dataset and has no closed-form expression [31]. It employs a space partitioning mechanism to divide the entire data space into ψ non-overlapping partitions using ψ randomly sampled points from a given dataset. The partitioning is adaptive to local density, i.e., it creates large partitions in sparse regions and small partitions in dense regions. The kernel measures the similarity between two points as the expectation that the two points fall into the same partition. For instance, consider a partitioning mechanism that creates ψ non-overlapping partitions, and the expectation is estimated from 100 independent partitionings. If two points fall into the same partition in 20 out of 100 partitionings, then their similarity is estimated as 0.20. Isolation Kernel exhibits two distinctive characteristics: (1) *Two points in a sparse region have higher similarity than two equidistant points in a dense region*, enabling better separation of clusters with varying densities than that using a data-independent kernel such as Gaussian kernel. (2) It has *a finite-dimensional feature map*, facilitating efficient similarity calculations of distributional kernel $K(\mathcal{P}_{G_i}, \mathcal{P}_{G_j})$ with linear time complexity. In contrast, Gaussian kernel has a feature map with infinite dimensionality.

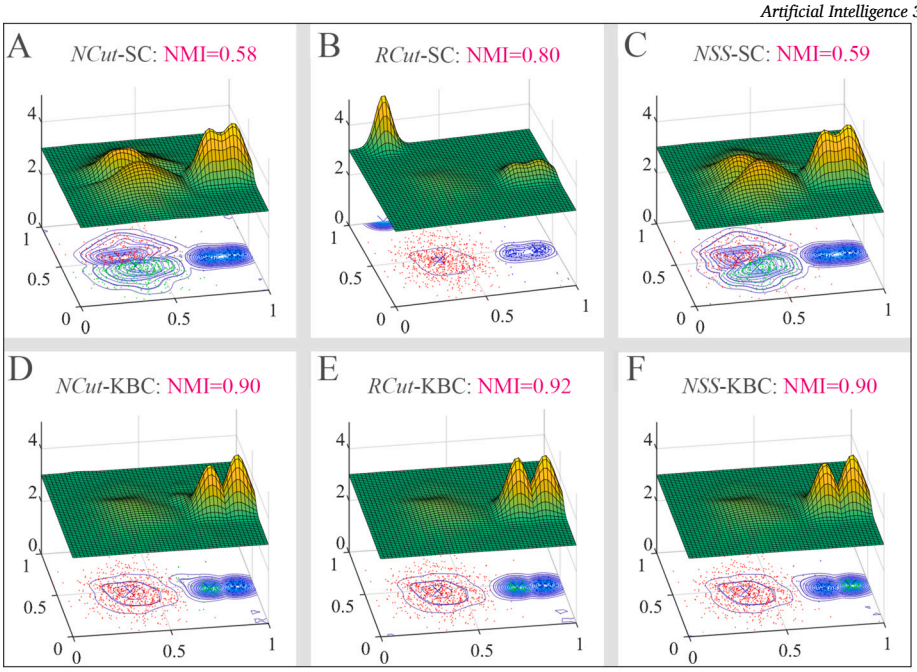


Fig. 2. The distributions of $\widetilde{K}$ similarity scores for three versions of SC in subfigures A, B & C; and three versions of KBC in subfigures D, E & F on the High-Low dataset which was used by [5] to demonstrate a fundamental limitation of SC. Gaussian kernel is used in KBC and SC throughout the paper, unless stated otherwise. The quality of each clustering outcome, measured in terms of NMI (normalized mutual information [22]), is also provided. The $\widetilde{K}$ similarity score is stated in Definition 1.

Figs. 2 (A to C) show that the clustering outcomes of SC have serious shortcomings: they all merge the two small dense clusters into one, and then either split the large sparse cluster into two (when the $NCut$ and NSS criteria are used) or impose a cut at the edge of the data space that creates a cluster of a single point (when $RCut$ is used).

Although $RCut$ produces better NMI than those produced by the other two criteria on this dataset (see [12] for details), their clustering outcomes are far from ideal.

In contrast, KBC using any one of the three criteria is able to produce a clustering outcome that corresponds to the desired clustering outcome, shown in Figs. 2 (D to F).

The key to KBC's superior clustering outcomes lies in its initial cluster cores—they are the cores of the final clusters that provide the essential information necessary to produce a good clustering outcome. SC, which aims to minimize a chosen criterion based on local information of point-to-point similarity only, could produce ill-matched clusters—exactly because the essential information of clusters is not used to guide the clustering process in every example shown in Figs. 2 (A to C).

In a nutshell, initial cluster cores, which correspond to the desired clustering outcome, are crucial. We argue that the fundamental limitations of SC, identified previously [5], are a direct result of insisting on using the 'ill-matched' tool, i.e., eigendecomposition. In addition, SC pays no attention to the thing that really matters, i.e., the initial cluster cores. The specific objective/criterion of the eigendecomposition fails to address the fundamental limitations.

7. Analyses of two conditions of step 1 in KBC

Here we analyze two different conditions of step 1 in KBC that influence its clustering outcomes, where the cluster cores are obtained via sampling from known clusters and unknown clusters in a given dataset. They are described in the first two subsections. In the third subsection, we show the impact of KBC's cluster cores on k-means clustering.

7.1. What are the optimal cluster cores in order to produce an optimal clustering outcome?

To produce an optimal clustering outcome that discovers all clusters C in a given dataset D using KBC, an optimal cluster core for every C must be used to perform the point assignment in step 2 of KBC.

Definition 3. A subset D_s with sufficient sample size s is a representative sample of the distribution of D such that $\mathcal{P}_{D_s} \approx \mathcal{P}_D$.

Definition 4. A fitting initial cluster core of cluster C is a representative sample $G \subset C$ of the distribution of C such that $\mathcal{P}_G \approx \mathcal{P}_C$.

Given C, the 'quality' of a cluster core G can be examined by comparing the clustering outcomes of step 2 of KBC using cluster cores of different 'qualities'.

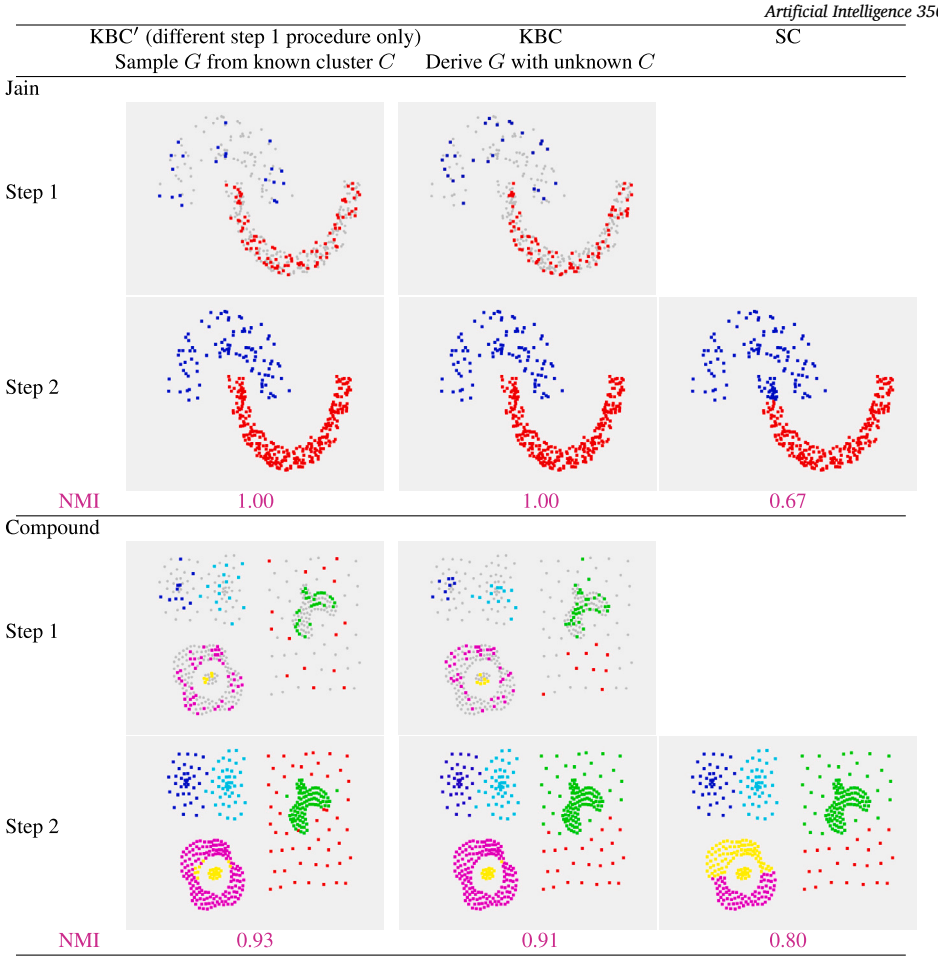


Fig. 3. The clustering outcomes (also in terms of NMI) of KBC′, KBC and SC on the Jain (top) and Compound (bottom) datasets. On each dataset, the cluster cores G sampled/found are shown in the 'Step 1' row; the final clustering outcomes are shown in the 'Step 2' row.

The second column in Fig. 3 shows the clustering outcome derived from cluster cores via random sampling from known clusters, which replaces step 1 in KBC. With a representative sample $\mathcal{P}_{D_s} \approx \mathcal{P}_D$, the cluster cores are found to be optimal (NMI = 1) on the Jain dataset and close to the optimal on the Compound dataset.

7.2. Finding fitting initial cluster cores from a given dataset with unknown clusters

In practice, we do not know the clusters in a given dataset D. In KBC, the first task is to identify cluster cores $G \subset D$ for all clusters without the knowledge of clusters in D.

We formulate a theorem as follows:

Theorem 1. *A necessary and sufficient condition for KBC to produce an optimal clustering result is to identify fitting cluster cores* $G = \{G_1, ..., G_k\}$ *with* k *clusters such that* $\forall_{i \in 1,...,k}, \mathcal{P}_{G_i} \approx \mathcal{P}_{C_i}$.

Proof. We start by establishing the sufficient component of the condition that if the probability distributions $\mathcal{P}_{G_i}$ and $\mathcal{P}_{C_i}$ are approximately equal, the expected clustering error is guaranteed to be very small. Then, we show that achieving a high-quality clustering result necessitates that $\mathcal{P}_{G_i}$ be approximately equal to $\mathcal{P}_{C_i}$. This is done by providing an example that, fails to meet this condition, could result in a large clustering error rate.

Sufficient Component: The probability that point $\mathbf{x}$ is correctly clustered to C_i is:

$$P(\mathbf{x} \in C_i) = P(K(\delta(\mathbf{x}), \mathcal{P}_{G_i}) \geq K(\delta(\mathbf{x}), \mathcal{P}_{G_j}), \forall j \neq i).$$

As $\forall_{i \in 1,...,k}, \mathcal{P}_{G_i} \approx \mathcal{P}_{C_i}$,

$$P(\mathbf{x} \in C_i) = P(K(\delta(\mathbf{x}), \mathcal{P}_{C_i}) + \epsilon_{\mathbf{x},i} \geq K(\delta(\mathbf{x}), \mathcal{P}_{C_j}) + \epsilon_{\mathbf{x},j}, \forall j \neq i),$$

where $\epsilon_{\mathbf{x},i} = K(\delta(\mathbf{x}), \mathcal{P}_{G_i}) - K(\delta(\mathbf{x}), \mathcal{P}_{C_i})$ is the error introduced by using $\mathcal{P}_{G_i}$ as an estimate of $\mathcal{P}_{C_i}$.

For the convenience of analysis, we rewrite the above probability as follows:

$P(\mathbf{x} \in C_i) = P(A_i(\mathbf{x}))$, where $A_i(\mathbf{x}) := K(\delta(\mathbf{x}), \mathcal{P}_{C_i}) + \epsilon_{\mathbf{x},i} \geq K(\delta(\mathbf{x}), \mathcal{P}_{C_j}) + \epsilon_{\mathbf{x},j}, \forall j \neq i$.

Let

$$Y_{i,j}(\mathbf{x}) = K(\delta(\mathbf{x}), \mathcal{P}_{C_j}) + \epsilon_{\mathbf{x},j} - K(\delta(\mathbf{x}), \mathcal{P}_{C_i}) - \epsilon_{\mathbf{x},i},$$
$$Z_i(\mathbf{x}) = \sum_j Y_{i,j}(\mathbf{x}), \ Z_i \in [-1, 1]$$

Hence, $A_i(\mathbf{x}) := Z_i(\mathbf{x}) \leq 0$. Let $\mu = \mathbb{E}_i[Z_i(\mathbf{x})] < 0$.

According to Hoeffding's inequality (see Appendix B for details), we have:

$$P\left(Z_i(\mathbf{x}) \leq 0\right) = 1 - P\left(Z_i(\mathbf{x}) \geq 0\right) \geq 1 - \exp\left(-\frac{\mu^2}{2(k-1)}\right)$$
$$1 - \exp\left(-\frac{\mu^2}{2(k-1)}\right) \geq 1 - \delta \Longleftarrow \mu \leq -\sqrt{-2(k-1)\ln\delta}$$

Therefore, when $\mu \leq -\sqrt{-2(k-1)\ln\delta}$, $P(\mathbf{x} \in C_i) = P(A_i(\mathbf{x})) \geq 1 - \delta$, where k is the number of clusters and $\delta > 0$ is a small constant.

Considering the entire data set, the clustering error rate is:

$$Err = 1 - \mathbb{E}_{\mathbf{x}}(A_i(\mathbf{x})) \leq \delta.$$

We need:

$$\mu \leq -\sqrt{-2(k-1)\ln\delta}$$

since $\mu_0 - 2\epsilon \leq \mu \leq \mu_0 + 2\epsilon$, where $\mu_0 = \mathbb{E}_i[Z_i(\mathbf{x})] < 0$ with $\epsilon_{\mathbf{x},i} = 0$, and $|\epsilon_{\mathbf{x},i}| \leq \epsilon$.

So we need $\mu_0 + 2\epsilon \leq -\sqrt{-2(k-1)\ln\delta}$,

$$\epsilon \leq -\frac{1}{2}\left(\sqrt{-2(k-1)\ln\delta} + \mu_0\right) = \epsilon_{max};$$

This shows that if $\forall_{i\in 1,\ldots,k}, \mathcal{P}_{G_i} \approx \mathcal{P}_{C_i}$ (i.e., the error $|\epsilon_{\mathbf{x},i}|$ is smaller than ϵ_{max}), then the expected clustering error is smaller than δ.

Necessary Component: An example, in which the condition $\forall_{i\in 1,\ldots,k}, \mathcal{P}_{G_i} \approx \mathcal{P}_{C_i}$ does not hold, is given as follows: If $\exists_{i\in 1,\ldots,k}, \mathcal{P}_{G_i} \not\approx \mathcal{P}_{C_i}$, and cluster cores identified G_i and G_j are samples of the same cluster C_i, i.e., $G_i \cap G_j = \emptyset$ and $(G_i \cup G_j) \subset C_i$, then the clustering outcome T obtained by KBC is: $\mathbb{E}[Err(T)] \geq \Delta_j$, where $\Delta_j > 0$ is the error rate due to cluster C_j in which many of its members are misclassified into other clusters.

This shows that if the condition $\forall_{i\in 1,\ldots,k}, \mathcal{P}_{G_i} \approx \mathcal{P}_{C_i}$ does not hold, then the expected clustering error rate could be very large. □

Step 1 in KBC ensures that the initial cluster cores obtained are 'good' (though not necessarily optimal as stated in Theorem 1) for step 2 in KBC under the following conditions:

- Each core G is the highest similarity connected component of the distribution of cluster C which has pair-wise similarities higher than τ, where $G \subset C$, and $\mathcal{P}_G$ represents the core part of the distribution $\mathcal{P}_G \approx \mathcal{P}_C$.
- A cluster of arbitrary shape has at least one prominent core $\mathcal{P}_G$ for an appropriate τ if
  - the representative subset $D_s \subset D$ contains a representative subset $G \subset C$ of every (unknown) cluster C in D, where all clusters $C \subset D$ are to be discovered by KBC; and
  - each (unknown) cluster C has either a density distribution that is a monotonically decreasing function from its peak or a uniform density distribution.

The clustering outcomes of the steps in KBC are shown in Fig. 3 (for each dataset). These outcomes show that step 1 of KBC (with an appropriate τ) identifies fitting cluster cores automatically and finds all optimal clusters on both datasets, except for a few mistakes in the sparsest cluster of the Compound dataset.

In contrast, SC's best clustering outcomes (shown in the last column in Fig. 3) are still significantly poorer than those of KBC on both datasets. Note that SC has difficulty identifying some clusters on both datasets because of the existence of sparse and dense clusters. For example on the Compound dataset, SC fails to identify correctly the two dense clusters on the bottom left and the sparsest cluster.

7.3. What if random seeds in k-means clustering are replaced with KBC's initial cluster cores?

The quality of KBC's initial cluster cores are further examined using two k-means based clustering, i.e., (A) kernel k-means clustering, and (B) k-means clustering on a dataset transformed using SC's eigenvectors.

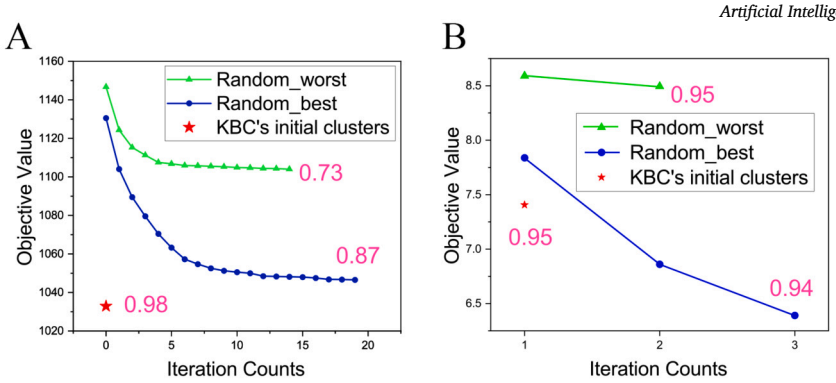


Fig. 4. (A) Kernel k-means with different initial seeds. (B) k-means using SC's eigenvectors: different initial seeds. The objective values of kernel k-means and k-means using SC's eigenvectors with (i) random initial seeds (the best and worst trials in terms of the final objective values out of 20 trials), and (ii) initial cluster cores as determined in step 1 of KBC. The COIL20 dataset is used here. The quality of each clustering outcome, measured in terms of NMI, is also provided. k-means already converged after the first iteration when k-means is initialized by KBC. We search for τ from {0.05, 0.1,..., 0.95} and select the τ with the best clustering outcome. 'Best' refers to the best average clustering outcome in terms of NMI after running 20 times with the same parameter (e.g., parameter k for k-means), rather than the result corresponding to the best parameter setting in a single run.

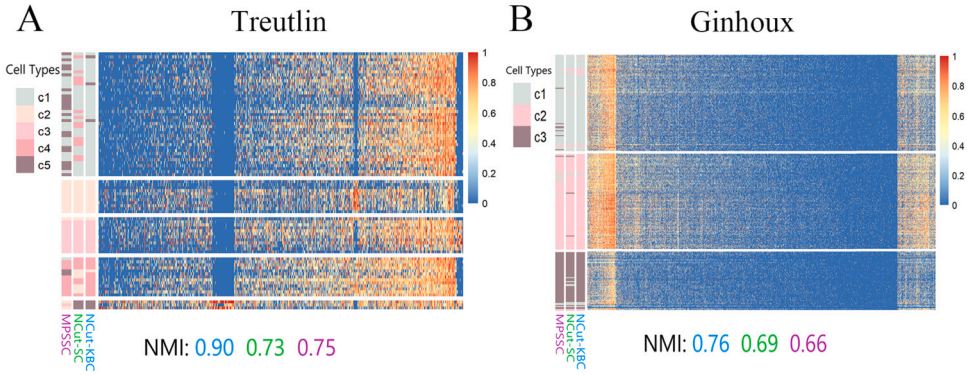


Fig. 5. The clustering results of MPSSC, $NCut$-SC and $NCut$-KBC on the Treutlein and Ginhoux datasets. The heatmap of gene expression is arranged according to the ground-truth cluster labels. Their NMI results are printed with their corresponding colors.

Here we compare the optimization outcomes of each of the two k-means clustering algorithms using two different initializations, i.e., random initial seeds versus KBC's initial cluster cores G_i.

Fig. 4A shows the progression of the optimization process in terms of the objective values. Even with the best result out of the 20 trials, kernel k-means using random initial seeds produces a worse outcome than that using KBC's initial cluster cores, completed in one iteration only.

A similar experiment using SC's eigenvectors as the starting point to find initial seeds and then perform k-means clustering is shown in Fig. 4B. This result shows the benefit of the eigenvectors produced by SC, i.e., the random seeds have high quality that enables k-means to converge in a few iterations. Even in this case, KBC's initial cluster cores have similar or better quality than the random seeds as k-means completes in one iteration with equal or better NMI.

In a nutshell, KBC produces the final clusters, implicitly maximizing the optimization (NSS) criterion, shown in Eq (1), without an optimization procedure. KBC can be used to produce clusters satisfying either the $NCut$ or $RCut$ criterion. KBC's initial cluster cores are the key to achieving this optimized clustering outcome without an optimization procedure.

8. Application to two domains in biomedical studies

8.1. Gene expression data

Fig. 5 shows the clustering outcomes of a tailored-made SC called MPSSC [32] for the gene-expression Treutlein and Ginhoux datasets in comparison with the typical SC and KBC, both of which employ the same $NCut$ criterion. KBC performs better than MPSSC and SC on these datasets, and the performance gaps in NMI are large on the Treutlein dataset. The c1 cluster in this dataset posts a real challenge for both versions of SC, where MPSSC merged the entire c5 cluster into the c1 cluster; and SC mixed c1 substantially with c3 & c4. The detailed results are provided in Appendix D.1.

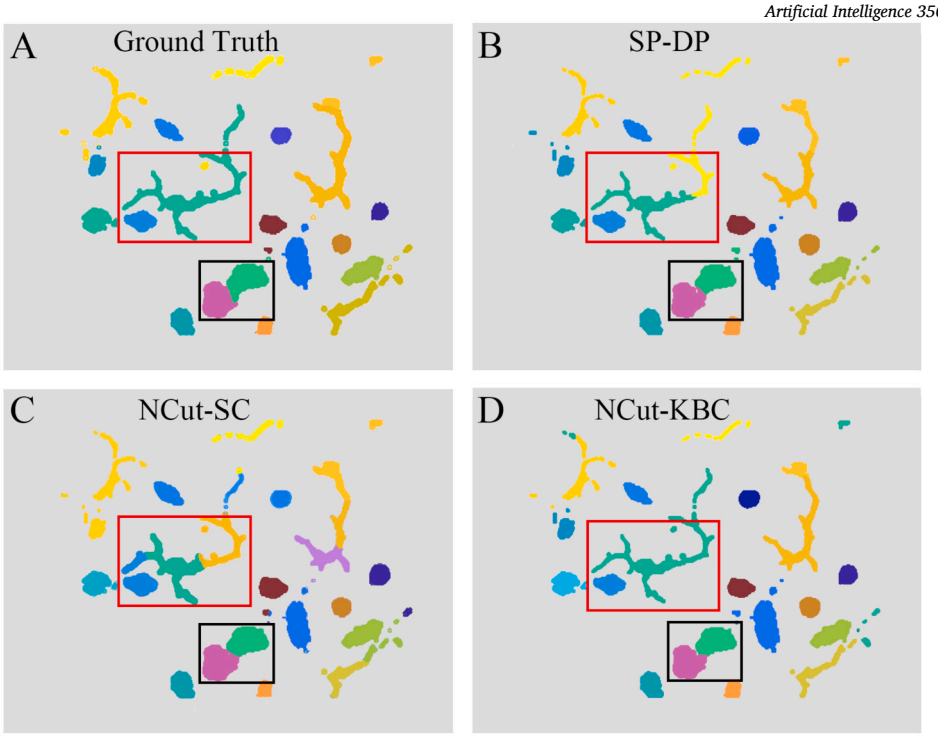


Fig. 6. Segmentation of immune cells (as used in [33]) by SP-DP, $NCut$-SC & $NCut$-KBC. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

8.2. Segmentation of immune cells in confocal microscopy

Cell segmentation aims to identify cells in a microscopic image, where each cell is a segmented region in the image. Correctly identified cells in a microscope image capture biologically relevant morphological information which indicates the physiological states of the cells. Cell segmentation is considered to be an important step in many image-based cellular studies.

Here, we apply our clustering algorithm to perform cell segmentation on a data set as used in DP-SP [33]. The comparison result is shown in Fig. 6. Like in [33], we focus on two regions, indicated in black and red boxes in Fig. 6. In the black-box region, all the three algorithms, SP-DP, $NCut$-SC & $NCut$-KBC, have correctly separated the two cells, even though the cells appear to be connected.[4] In the red-box region, only $NCut$-KBC correctly connects all dendrites of the dendritic cell (i.e., the irregular-shaped cell), even the tail of the cell outside the box. Yet, $NCut$-SC separates the cell into three segments; and SP-DP [33] splits it into two segments.

The above results of the two applications are manifestations of the fundamental limitations of SC mentioned in Sections 2 and 6.

9. Empirical evaluation using 26 datasets

Here we conduct three sets of experiments, each using the same 26 datasets which include artificial datasets, high-dimensional datasets, and some benchmark datasets. The details of the experimental settings are provided in Appendix D.2.

The first compares SC with KBC, each using one of the three criteria. The second compares kernel k-means with KBC. The third assesses the utility of Gaussian Kernel versus Isolation Kernel in KBC. They are described in the following three subsections.

9.1. SC with KBC

Fig. 7 shows a summary of the comparison between SC and KBC, each pair with one of the three criteria: $NCut$, $RCut$ and NSS. Without exception, KBC improves the clustering performance of SC in terms of NMI for every criterion. The improvements are statistically significant for both the NSS and $RCut$ criteria, as shown in the significance test result in Fig. 7(right). The detailed results of the individual 26 datasets are given in Table A4 in Appendix D.4.

[4] Some clustering algorithm such as DBSCAN [34] fails to separate the two cells (see Fig 5 in [33]).

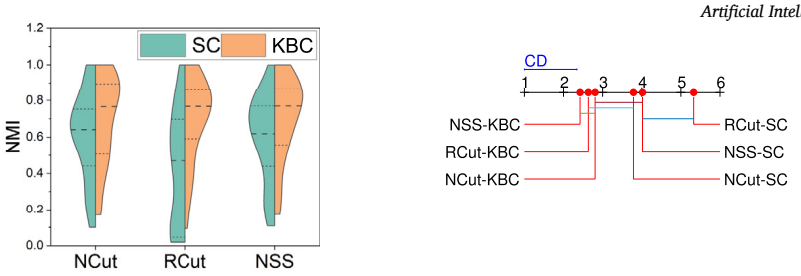


Fig. 7. Violin plots and the Nemenyi significance test results at 0.1 significance level of the comparison over 26 datasets in terms of NMI for SC vs KBC.

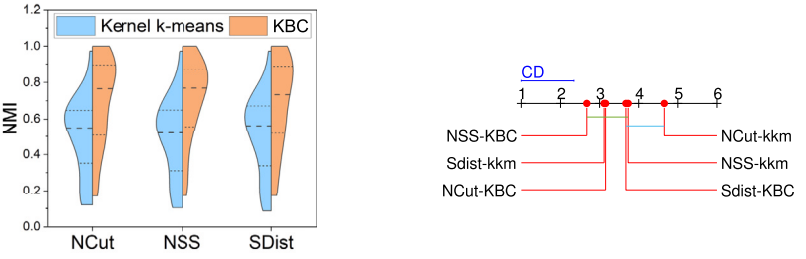


Fig. 8. Violin plots and the Nemenyi significance test results at 0.1 significance level of the comparison over 26 datasets in terms of NMI for Kernel k-means (kkm) vs KBC.

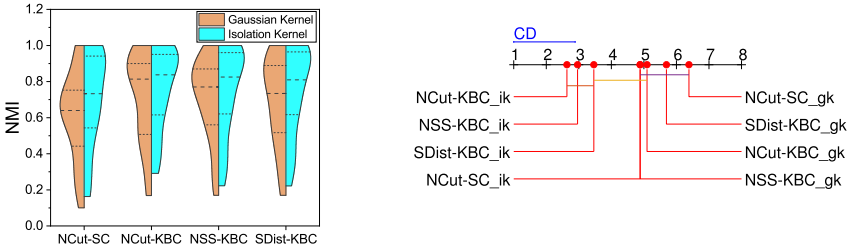


Fig. 9. Violin plots and the Nemenyi significance test results at 0.1 significance level of the comparison over 26 datasets in terms of NMI for Gaussian Kernel (gk) vs Isolation Kernel (ik).

9.2. Kernel k-means vs KBC

Fig. 8 shows a summary of the comparison between Kernel k-means and KBC, each pair with one of the three criteria: $NCut$, NSS and $SDist$ (which is the squared Euclidean distance criterion commonly used in k-means). Overall, KBC performs better than Kernel k-means in terms of NMI for every criterion without exception. The improvements are statistically significant for both the NSS and $RCut$ criteria, as shown in the significance test result in Fig. 8(right). Table A5 in Appendix D.4 shows that the detailed results of individual datasets.

9.3. Gaussian kernel vs isolation kernel in KBC

Kernel-based methods always use Gaussian kernel (gk) by default (so as all the experiments we have conducted thus far). A recent Isolation Kernel (ik) [31,35,21] has been shown to produce better task-specific performance in SVM classification, density-based clustering and anomaly detection than those using the Gaussian kernel because the former is data-dependent and the latter is not.

Fig. 9 shows the clustering results of using either Gaussian Kernel (gk) or Isolation Kernel (ik) in SC and KBC. We have the following observations. First, every ik version of KBC performs better than the corresponding gk version; so as SC. The difference is significant for KBC between the ik and gk versions. See the significance test result in Fig. 9(right).

Second, the best performer among the gk versions is NSS-KBC; and $NCut$-SC is the worst performer. The use of ik lifted the clustering performance of the worst performer $NCut$-SC_gk to a level similar to the best performer NSS-KBC_gk of all gk versions.

Third, the use of ik can further improve the clustering performance of NSS-KBC that employs GK.

The detailed results of individual datasets are given in Table A6 in Appendix D.4.

Our results here are consistent with the previous results comparing ik and gk in SVM classification, density-based clustering and anomaly detection [31,35,21].

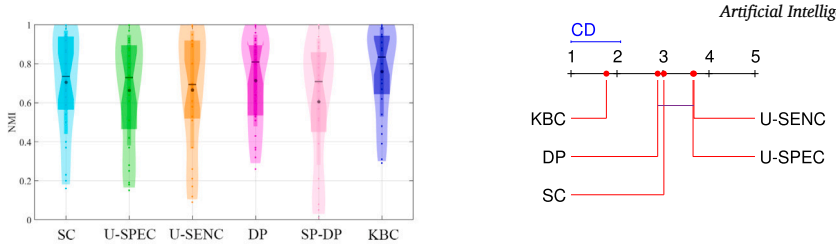


Fig. 10. Violin plot of the comparison on the 26 datasets among U-SPEC & U-SENC, SC and KBC, and DP & SP-DP; and the Nemenyi significance test results at 0.1 significance level.

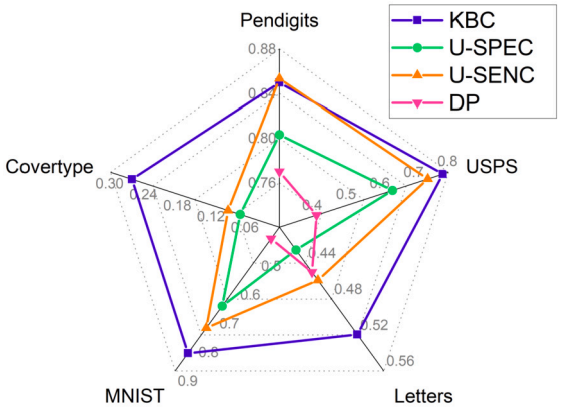


Fig. 11. Radar chart on five large image datasets (in terms of NMI).

10. Compare with recent spectral clustering and density-based clustering algorithms

A recent method called U-SPEC [10] has used a bipartite graph to speed up the runtime of SC, and its ensemble version is called U-SENC [10]. They are compared with the ik versions of SC & KBC. All of these four algorithms use the same $NCut$ criterion in this section. Also included in the comparison are two density-based clustering algorithms DP [36] and SP-DP [33]. We use two sets of datasets here: the 26 datasets (we have used in the last section) and five image datasets, where the images datasets have significantly larger data size than the any of the 26 datasets. The largest of the 26 datasets contains 6,500 data points, while the five image datasets have between 11,000 and 581,000 data points. Their comparison results on the 26 small datasets and 5 large image datasets are summarized and presented in Fig. 10 and Fig. 11, respectively.

We have the following observations:

- KBC is the best performer in terms of the clustering outcomes on both sets of small and large datasets.
- On the small 26 datasets, the closest contender to KBC is DP, but DP has the worst performance on the large image datasets among the four algorithms shown in Fig. 11. In addition, DP performs worse than KBC on each of these small and large datasets, some with large margins. SP-DP performs worse than DP on all these datasets (the results of SP-DP are omitted in Fig. 11 to improve readability).
- Like SC, U-SPEC and U-SENC are not competitive with KBC on most datasets. Note that SC and U-SPEC are comparable since they both use the eigencomposition on the same $NCut$ criterion. U-SENC, which makes use of an ensemble of U-SPEC, does not perform better than U-SPEC in general on the 26 datasets. However, on large datasets as shown in Figs. 11 and 12, U-SENC performs better than U-SPEC. These latter results are consistent with those presented previously [10].

The detailed results of individual datasets are given in Table A6 in Appendix D.4.

The Nemenyi significance test shown in Fig. 10(right) reveals that KBC is significantly better than both U-SPEC[5] and U-SENC, as well as DP and SC.

[5] U-SPEC [10] achieves linear time complexity by solving the eigendecomposition of a bipartite graph of n nodes wrt p representatives rather than a general graph (represented as a $n \times n$ matrix). The p representatives are obtained via running k-means by setting $k = p$ on a data subset of a given dataset of n points. The weight of each edge of the bipartite graph is obtained using a version of Gaussian kernel. U-SPEC performs eigendecomposition based on the $NCut$ criterion on the bipartite graph before running k-means again to produce the final clustering outcome. In other words, U-SPEC runs k-means twice. U-SENC [10] performs U-SPEC $m = 20$ times by default, each with a randomly selected $k_i, i = 1, \ldots, m$. Then, an $n \times k_c$ ($k_c = \sum_i k_i$ is the total number of clusters out of the 20 U-SPEC runs) bipartite graph is constructed, before the eigendecomposition and k-means are applied again to get the final clustering result.

Table 4
Comparison (in terms of NMI) between NSS-KBC ($s = 10000$) and NSS-KBC$_s$ ($s = 1000$).

datasets	#points	#dimensions	#clusters	NSS-SC	NSS-KBC	NSS-KBC$_s$
s3	5000	2	15	.79	.78	.80
RingG	1536	2	4	.80	.78	.78
complex9	3031	2	9	.61	.81	.80
unbalance	6500	2	8	1.00	1.00	1.00
cure-t2-4k	4200	2	7	.86	.74	.75
COIL20	1440	1024	20	.77	.84	.83
banknote	1372	4	2	.25	.87	.89
landsat	2000	36	6	.60	.64	.64
segment	2310	18	7	.67	.70	.69
waveform3	5000	21	3	.37	.40	.49
G-Strip	1400	2	2	.71	.87	.86
			Average NMI	.68	.77	.78

Table 5
Comparison (in terms of NMI) on three large datasets. N/A indicates out-of-memory error.

datasets	#points	#dimensions	#clusters	NSS-SC	U-SPEC	NSS-KBC
SF2M	2,000,000	2	4	N/A	.83	.94
CG10M	10,000,000	2	11	N/A	.85	.95
Flower20M	20,000,000	2	13	N/A	.89	.99

Table 6
Clustering scores on the three high-dimensional datasets.

datasets	metric	#points	#dimensions	k	NSS-SC	U-SPEC	NSS-KBC
MNIST8M	(NMI)(↑)	8100000	784	10	N/A	0.60	**0.74**
GLOVE-300	(DB)(↓)	400000	300	10	N/A	8.16	**3.50**
MJ5	(DB)(↓)	1475969	768	5	N/A	4.92	**4.42**
GLOVE-300	(CH)(↑)	400000	300	10	N/A	2310	**4009**
MJ5	(CH)(↑)	1475969	768	5	N/A	**29483**	26678

We decrease s from 10000 (NSS-KBC) to 1000 (NSS-KBC$_s$) and select the datasets with more than 1400 points for this experiment. The results are shown in Table 4. There is almost no difference between the two settings.

Due to time and memory limitations, SC cannot be extended to large datasets. We compared USPEC and NSS-KBC on the three large datasets (ranging from 2 million to 20 million) used in the U-SPEC paper [10]. The results are shown in Table 5. NSS-KBC performs better than U-SPEC on all three datasets.

In addition to the limitation of the number of points, the dimensionality of the data also limits the scalability of spectral clustering. Therefore, we conduct experiments on three large-scale datasets with high dimensionality to demonstrate the scalability of our proposed method: 768-D CLIP embeddings (MJ5), 300-D GloVe (GLOVE-300), and 784-D MNIST-full (MNIST8M). MNIST8M contains ten clusters. Since MJ5 and GLOVE-300 do not have ground truths, we determine the number of clusters k before the experiment (see the details in the Appendix D.3). We then use Davies-Bouldin (DB) [37] and CalinskiHarabasz (CH) [38] metrics to evaluate the clustering performance of each clustering algorithm. The clustering results are shown in Table 6. KBC outperforms U-SPEC on all three datasets except MJ5 in terms of the CH score only.

The main limitation of spectral clustering is that it is difficult to scale to large datasets. We use $s = 10000$ as the default setting mainly considering the application to large-scale data. We conducted an experiment to evaluate the running time of our algorithm, KBC, as the dataset size varied. The dataset sizes tested ranged from 1,000 to 5,000,000 instances, with the default s setting.

The scaleup test result in Fig. 12A shows that KBC consistently produces high NMI regardless of data sizes. U-SENC performs poorly when the data size is small. Its NMI increases as the data size goes up while U-SPEC's does not. The superior clustering outcome of U-SENC over U-SPEC comes at a price of running one order of magnitude slower.

Fig. 12B shows the result of a larger scaleup test comparing SC, U-SPEC and KBC. The actual runtime (CPU seconds) is given in Table 7. Note that SC cannot handle datasets larger than 500,000. The runtime of KBC is closed to the approximate method U-SPEC. As expected, SC has time complexity in-between $O(n^2)$ and $O(n^3)$; U-SPEC and KBC have sublinear runtime. Note that KBC and U-SPEC took 43 & 34 seconds, respectively, to complete a dataset of 5 million points, and they were still faster than SC which took 285 seconds to complete the run on a dataset of 0.05 million points. SC is projected to take more than 6 years to complete the run on the set of 5 million data points!

The parameter sensitivity test results of U-SPEC and KBC are provided in Appendix D.5.

11. Discussion

11.1. KBC is a more direct approach than SC that uses eigendecomposition

The proposed algorithm KBC takes a completely different path that offers a more direct approach than eigendecomposition. KBC is more direct in three aspects. First, it solves the equivalence of the original normalized graph cut problem, not an approximate one. As a result, the conversion from discrete to continuous and the reverse conversion, as required by SC that uses eigendecomposition

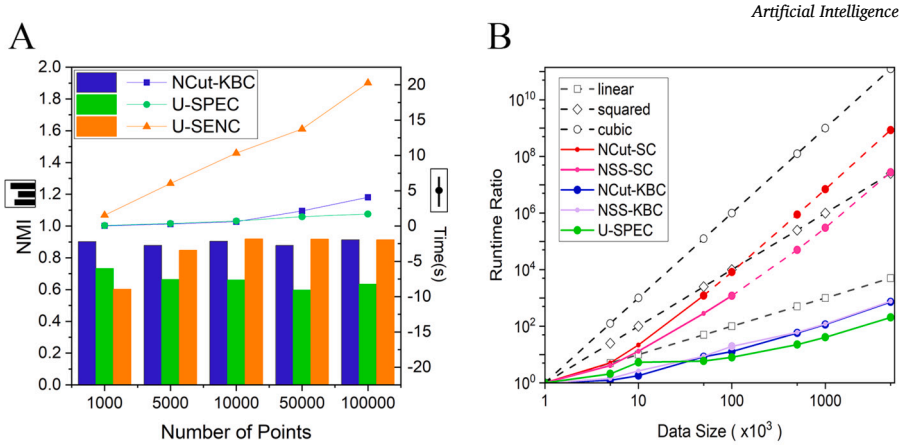


Fig. 12. (A) U-SPEC and U-SENC versus KBC in terms of NMI (bar chart) and runtimes (line chart) as data size increases on the High-Low dataset. (B) A larger scaleup test for SC, KBC and U-SPEC. The other criteria are omitted to avoid too many overlapping lines because they have similar results. SC took too long to run on large datasets; the dashed lines of $NCut$-SC and NSS-SC are projections from their last two runs that took less than runtime ratio = 10^4. U-SPEC and U-SENC have the same runtime ratio since U-SENC runs U-SPEC multiple times. 'Runtime ratio' refers to the ratio of the runtime on a dataset size to the runtime on the base dataset size of $n = 1000$.

Table 7
Actual runtime (in CPU seconds) for the scaleup test shown in Fig. 12B. N/A indicates the out-of-memory error.

Data size	U-SPEC	$NCut$-SC	NSS-SC	$NCut$-KBC	NSS-KBC
1000	0.17	0.23	0.17	0.06	0.06
5000	0.35	1.19	0.72	0.07	0.08
10000	0.89	5.10	2.29	0.11	0.15
50000	0.99	284.75	49.65	0.50	0.50
100000	1.34	N/A	209.79	0.76	1.11
500000	3.78	N/A	N/A	3.47	3.45
1000000	6.84	N/A	N/A	6.92	6.87
5000000	34.45	N/A	N/A	43.28	43.38

(described in Section 1), are no longer required. Second, KBC performs clustering directly, unlike SC which must perform eigendecomposition (as feature transformation) and then clustering using an existing clustering algorithm. Third, the ability to find fitting initial cluster cores (as exemplified in Figs. 3 and 4), which enables the point assignment step to be performed without an iterative process, is a key factor that facilitates this direct approach.

In terms of the use of heuristics, SC in its implementation uses heuristics that are approximations during and after the eigendecomposition. KBC in its implementation determines the initial cluster cores via a heuristic.

In addition to the two commonly used criteria, i.e., $NCut$ and $RCut$, there are other less commonly used criteria studied in the literature, e.g., average cut [39] and min/max cut [40]. They are variants of the base minimum cut criterion, unlike the SS and NSS criteria, stated in Section 3. However, none of these criteria address the two fundamental limitations of SC we have mentioned in Section 2.

11.2. Relation to Jarvis-Patrick algorithm and density-based clustering

It is common knowledge that the Jarvis-Patrick (JP) algorithm [41] and DBSCAN [34] have no objective function; thus, they could not have achieved the same objective function of SC. The reason why KBC can achieve the same objective of SC is due to KBC's use of distributional kernel to perform clustering. It is also clear that JP and DBSCAN do not use a distributional kernel.

The linking procedure used to find κ_τ-bounded clusters in the first step of KBC is equivalent to that used in JP[6] and DBSCAN[7], except that (a) JP employs a Shared-Nearest-Neighbor (SNN) similarity measure, DBSCAN uses Euclidean distance, and step 1 of KBC

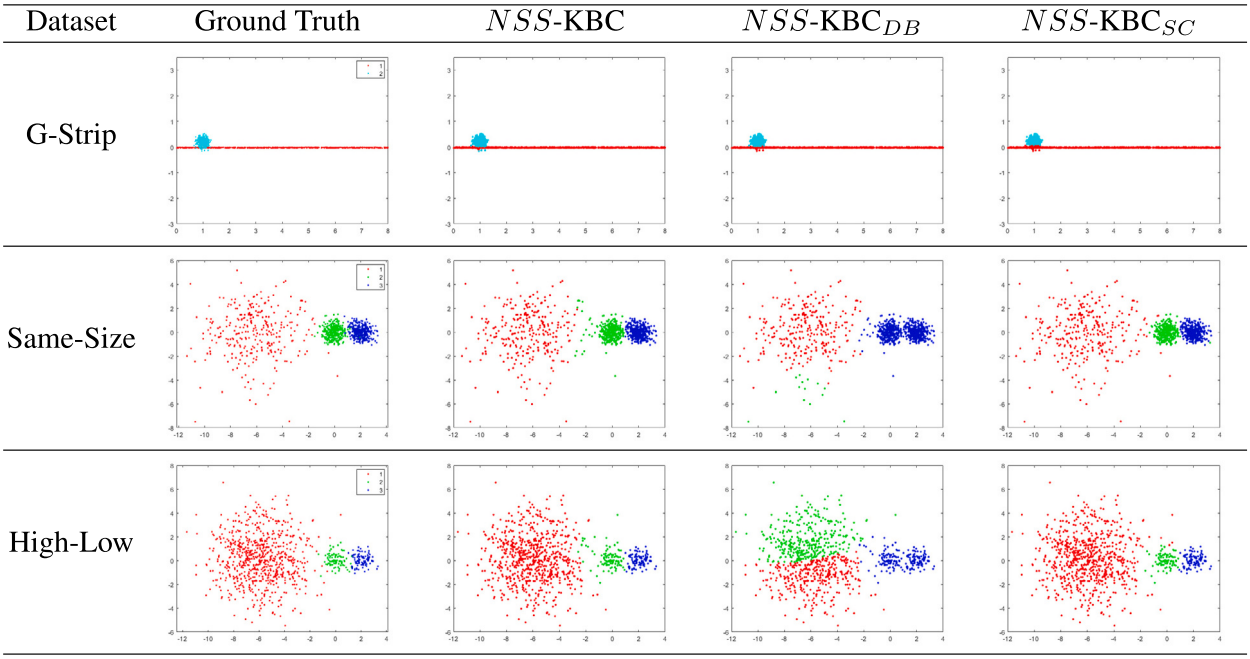
[6] JP first calculates the set of k nearest neighbors for each point and then links two points together if they are in each other's k-nearest neighbor list and they have at least k_{min} of their k nearest neighbors in common. The points that are either directly or indirectly linked belong to the same cluster. The JP algorithm itself does not produce a good clustering outcome since it is very brittle such that each true cluster is often split into multiple clusters. To overcome this limitation, SNN was used to replace Euclidean distance in DBSCAN [34] to produce a more robust clustering outcome [42].

[7] DBSCAN estimates the density $pdf(x)$ of a point x by counting points within a small ϵ-neighborhood. It uses a global density threshold to identify core points (with densities higher than the threshold) and links each point to a neighboring core point together if they are in each other's ϵ-neighborhood to form clusters. Similar to JP, the connected points belong to the same cluster. It is interesting to mention that when setting Minpts to 1 for DBSCAN, all points are core points. Then DBSCAN, employing the same similarity function used in JP and set $\epsilon = k_{min}$, produces the same clusters as the JP algorithm, provided that they use the same point linking process.

Table 8
Comparison (in terms of NMI) of different methods to find cluster cores in step 1 of KBC ($s = 1000$). Two search increments 0.05 and 0.01 for τ are used because DBSCAN is very sensitive to its parameter setting. This is a widely recognized limitation of DBSCAN [43].

datasets	$\tau \in [0.1, 0.95]$ with increments of 0.05			$\tau \in [0.02, 0.99]$ with increments of 0.01		
	NSS-KBC	NSS-KBC$_{DB}$	NSS-KBC$_{SC}$	NSS-KBC	NSS-KBC$_{DB}$	NSS-KBC$_{SC}$
s3	0.80	0.00	0.80	0.80	0.79	0.80
RingG	0.78	0.44	0.81	0.80	0.74	0.83
complex9	0.80	0.00	0.82	0.86	0.81	0.85
unbalance	1.00	1.00	1.00	1.00	1.00	1.00
cure-t2-4k	0.75	0.75	0.77	0.79	0.77	0.89
COIL20	0.83	0.88	0.76	0.82	0.88	0.78
banknote	0.89	0.74	0.59	0.86	0.83	0.59
landsat	0.64	0.63	0.65	0.65	0.65	0.65
segment	0.69	0.66	0.71	0.70	0.68	0.72
waveform3	0.49	0.41	0.41	0.44	0.43	0.45
Average Score	0.77	0.55	0.73	0.77	0.76	0.75

Table 9
Comparison on the three datasets used to assess the fundamental limitations of SC.

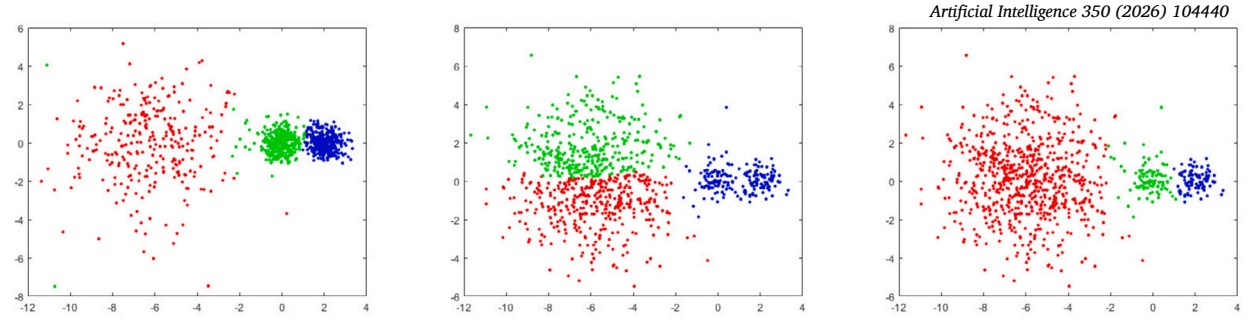


employs a kernel; and (b) JP and DBSCAN perform the clustering from the entire dataset, but step 1 of KBC identifies the cluster cores from a data subset only.

We show that step 1 in KBC can use an existing clustering algorithm to find the cluster cores from a data subset. We compare NSS-KBC with NSS-KBC$_{DB}$ and NSS-KBC$_{SC}$, which employ DBSCAN and Spectral Clustering, respectively, to find cluster cores in step 1 in Algorithm 1.

The comparison results, showed in Table 8, reveal that when we fine-tune the parameter τ with 0.05 increments, NSS-KBC$_{DB}$ yields no clusters on the s3 and complex9 datasets. Only when we reduce the increment size to 0.01, do we obtain more satisfactory outcomes. NSS-KBC consistently outperforms NSS-KBC$_{DB}$, with the only exception on COIL20 and unbalance. Similarly, NSS-KBC$_{SC}$ demonstrates superior clustering performance compared to NSS-KBC$_{DB}$, except on COIL20 and banknote. Note that NSS-KBC and NSS-KBC$_{SC}$ exhibit comparable clustering performance on all datasets, except COIL20 and banknote, where NSS-KBC performs better.

In Table 9, we show the comparison results of the three algorithms on the three datasets used to assess the fundamental limitations of spectral clustering. Both NSS-KBC and NSS-KBC$_{SC}$ have good performance on all three datasets, while NSS-KBC$_{DB}$ achieves good clustering on G-Strip only, but it performs very poorly on the Same-Size and High-Low datasets. Only by reducing the step size of the search increment to 0.01 can we achieve better results for NSS-KBC$_{DB}$ on the Same-Size dataset as shown in Fig. 13. Further, to obtain good clustering results on the High-Low dataset in Fig. 13, the increment size must be decreased to 0.001 for NSS-KBC$_{DB}$. These results, due to the sensitivity of parameter setting in DBSCAN, are consistent with the results shown in Table 8.



(a) Same-Size, increment:0.01 (b) High-Low, increment:0.01 (c) High-Low, increment:0.001

Fig. 13. Clustering outcomes of NSS-KBC$_{DB}$ with different step size.

In summary, we identify two good candidates to find cluster cores in step 1 of KBC, i.e., the linking procedure (commonly used in DBSCAN and JP) that employs a kernel and SC. The use of a kernel reduces the brittleness issue of the linking procedure. This is because only the largest k clusters are required, and they often contain the peaks and are therefore stable. It is so despite the fact that (i) each true cluster may be split into multiple cluster cores (and often only the largest one is selected by KBC); and (ii) finding the cluster cores from a subsample does not alter its final clustering outcome as long as the subsample is a representative sample of the data distribution, as analyzed in Section 7 and demonstrated in Fig. 3.

The use of SC in step 1 of KBC is interesting. This suggests that SC's clustering outcome can be improved by performing the clustering using a distributional kernel. Note that the key procedure of SC is the use of eigendecomposition to perform the feature transformation, not clustering (as stated in Section 1).

In essence, the main clustering process of KBC is step 2, while step 1 (though important) can employ an existing algorithm to find the cluster cores from a data subset, rather than the entire dataset. This is a distinctive characteristic of KBC to significantly ease the burden of final clustering. None of the existing density-based clustering algorithms[8] (including the more recent DP [36] and SP-DP [33]) make use of this idea.

KBC has linear complexity for two reasons: (a) KBC enables step 1 to be conducted using a data subset, and assigns all points in the entire dataset using a distributional kernel in the last step. This is impossible with DBSCAN, DENCLUE or DP because the density of each point in the dataset must be estimated in their procedures. (b) With cluster cores, the point assignment step in KBC can be completed in one iteration, and no revision is required. The key efficiency advantage comes from the use of a distributional kernel having a finite-dimensional feature map to perform the point assignment. This enables the last step to be completed in linear time. In contrast, density-based clustering [34,44,36,33] has quadratic time complexity in each of their density estimation and linking processes.

In terms of the quality of the clustering outcomes, *KBC discovers clusters of arbitrary shapes and varied densities but others do not.* Compared with density-based clustering algorithms, such as DBSCAN and DP [36], KBC achieves better clustering outcomes (see the details in Table A6 in Appendix D.4); and DBSCAN generally performs poorer than DP on datasets with clusters of varied densities [45].

The key reason of KBC's superiority is the use of a distributional kernel to perform the final clustering, enabling arbitrary-shaped and varied densities clusters to be discovered. All density-based clustering can discover arbitrary-shaped clusters if the densities of the clusters are approximately the same. DP has relaxed this condition to find clusters of varied densities; but the cure creates a new weakness.[9] See [45] for details.

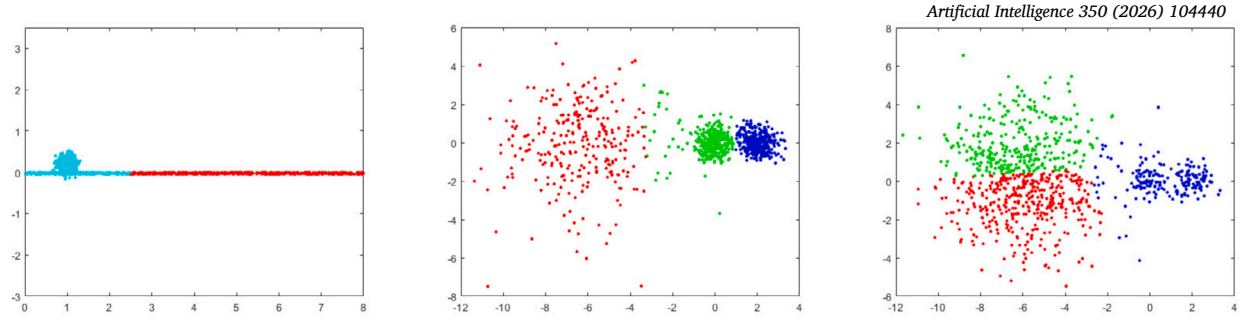
11.3. Why can KBC produce a good clustering outcome with one iteration of clustering?

On the surface, the clustering process of KBC in step 2 is similar to the maximization step in k-means clustering which assigns every point to its most similar cluster. Why can KBC produce a good clustering outcome in one iteration, whereas k-means requires multiple iterations?

KBC differs from k-means in two key aspects:

[8] DBSCAN and DENCLUE identify core individual points which have densities higher than a threshold among all points in the given dataset. No clusters are formed in this density estimation process. Step 1 of KBC identifies cluster cores from a data subset only, rather than the entire given dataset, where any two points within each cluster core have pair-wise similarities more than τ.

[9] DP uses a tailored-made measure to identify a peak of each cluster (at the end of the density estimation process). This effectively eliminates the shortcomings of early density-based clustering algorithms, dealing with varying density clusters, which rely on a fixed density threshold in the linking process. However, this creates a new weakness, i.e., the wrongly identified peaks (under some conditions) impact the final clustering outcomes severely. To overcome this weakness of DP, SP-DP [33] proposes to find each cluster based on the shortest paths from its density peak to all members of the cluster. However, SP-DP uses the same peak-finding procedure as DP. Therefore, it has exactly the same weakness of DP as described above. The condition this occurs has been identified [45].



(a) G-Strip (b) Same-Size (c) High-Low

Fig. 14. Clustering outcomes of the point-oriented approach which employs k-means in step 2 of KBC.

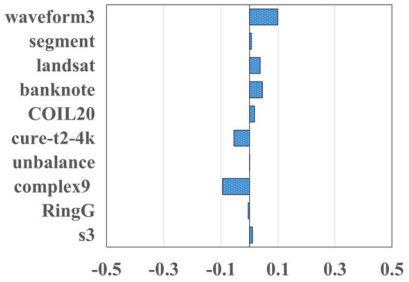


Fig. 15. Improvement (in terms of NMI) due to post-processing in terms of NMI.

- KBC identifies members of each cluster by treating each cluster as a distribution. That requires a distributional kernel to accomplish the clustering task. In contrast, k-means clusters points in the input space via a distance function, and it has no concept of a distribution.
- KBC requires only one iteration to complete the clustering task because cluster cores have been identified in step 1, and both the cluster cores and final clusters are treated as distributions in the clustering process. In contrast, k-means is essentially an Expectation-Maximization optimization algorithm, designed by thinking in terms of points only: Start with some random centers and optimize to find the good centers of clusters. Like all optimization procedures, multiple iterations is part of the search process required in order to achieve its objective function.

A follow-up question is: Is there another way to perform clustering in step 2 of KBC that can also be completed in one iteration? We are not aware of one, to the best of our knowledge. While there are other alternative distributional measures such as Wasserstein distance [46] and KL-Divergence [47], they are not applicable because the clustering requires to compute the similarity between a point w.r.t. a distribution, not between two distributions. In addition, they have high computational cost.

An alternative is to stick with the point-oriented paradigm by using k-means. That is, once the cluster cores are obtained, compute the average center from each cluster core, and then use them as the starting centers for k-means.

The average NMI of the point-oriented approach (using k-means in step 2 of KBC) on the 10 datasets shown in Table 7 is only 0.64, much lower than the 0.77 of the distribution-oriented approach of KBC. And the point-oriented approach still has fundamental limitations as shown in Fig. 14.

The above result shows that the distributional treatment is critical to the ability to complete the clustering task with a good outcome in one iteration, and a distributional kernel is the only means to achieve it, as far as we know.

11.4. Effect of post-processing

In all of the above experiments, we have employed a post-processing (step 3 in Algorithm 1) to refine $\mathbb{C} = \{C_1, \dots, C_k\}$ to improve the objective: $\max_{\mathbb{C}} \sum_{C \in \mathbb{C}} \sum_{\mathbf{x} \in C} K(\delta(\mathbf{x}), \mathcal{P}_C)$.

The effect of the post-processing of NSS-KBC is shown in Fig. 15. Nine out of the ten datasets have less than 0.05 or no improvement of NMI due to the post-processing. These are the datasets in which representative samples can be obtained in step 1. The post-processing has exerted an significant improvement on the waveform3 dataset because the samples obtained in step 1 are not representative enough. This is due to its peculiar data distribution. Note that the results of all the algorithms on this dataset are lower than 0.5 in terms of NMI.

12. Conclusions

This work is motivated by the two fundamental limitations of Spectral Clustering, stated in Section 2. We identify that the unquestionable tool of eigendecomposition has been the key obstacle in addressing these fundamental limitations.

We show here that the alternative tool of distributional kernel has provided a better means to achieve the same objective without the fundamental limitations. The resultant simpler and more direct Kernel-Bounded Clustering (KBC) algorithm is better than Spectral Clustering in three key aspects: It has better clustering outcomes, runs orders of magnitude faster, and a clearer cluster definition. It is a complete metamorphosis in the 50 years of research in Spectral Clustering in view of the fact that KBC achieves the same objective of Spectral Clustering without eigendecomposition or optimization.

CRediT authorship contribution statement

Hang Zhang: Writing – review & editing, Visualization, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Kai Ming Ting:** Writing – review & editing, Writing – original draft, Project administration, Funding acquisition, Conceptualization. **Ye Zhu:** Writing – review & editing, Writing – original draft, Visualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Kai Ming Ting reports financial support was provided by National Natural Science Foundation of China. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Kai Ming Ting is supported by the National Natural Science Foundation of China (Grant No. W2531050 & 92470116). This project is supported by the State Key Laboratory for Novel Software Technology at Nanjing University (Grant No. KFKT2024A01).

Appendix A. Use either $NCut$ or $RCut$ in KBC

To use either $NCut$ or $RCut$ in KBC, one only needs to modify the second step of the KBC algorithm.
For $NCut$, the second step is changed to:

$$C_j = \{\mathbf{x} \in D \mid \underset{i\in[1,k]}{\operatorname{argmax}} \frac{K(\delta(\mathbf{x}), \mathcal{P}_{G_i})}{K(\mathcal{P}_{G_i}, \mathcal{P}_D)} = j\}, \forall_{j\in[1,k]}.$$

For $RCut$, the second step is changed to:

$$C_j = \{\mathbf{x} \in D \mid \underset{i\in[1,k]}{\operatorname{argmax}} K(\delta(\mathbf{x}), \mathcal{P}_{G_i}) - K(\mathcal{P}_{G_i}, \mathcal{P}_D) * |D| = j\}, \forall_{j\in[1,k]}.$$

Appendix B. Hoeffding's inequality

Let $X_1, ..., X_n$ be independent random variables such that $a_i \le X_i \le b_i$ almost surely. Consider the sum $S_n = X_1 + \cdots + X_n$. Then Hoeffding's theorem states that, for all $t > 0$.

$$P(S_n - \mathbb{E}[S_n] \ge t) \le exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

So we have:

$$P(S_n \ge t) = P(S_n - \mathbb{E}[S_n] \ge t - \mathbb{E}[S_n]) \le exp\left(-\frac{2(t - \mathbb{E}[S_n])^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Let $S_n = Z_i(x), t = 0, \mathbb{E}[S_n] = \mu, n = k-1, b_i = 1, a_i = -1$, we have:

$$P(Z_i(x) \ge 0) \le exp\left(-\frac{\mu^2}{2(k-1)}\right).$$

Appendix C. Pseudocode for post-processing

The pseudocode of post-processing is shown in Algorithm 2.

Algorithm 2 Post-processing.

Input: D - dataset, K - kernel function, $\mathbb{C}$ - given clusters
Output: Refined clusters $\mathbb{C} = \{C_1, \dots, C_k\}$
  Initialize $\mathbb{C}_0 \leftarrow \mathbb{C}$ ▷ Initial cluster assignments
  $h \leftarrow \lceil 0.01 \times |D| \rceil$ ▷ Threshold for convergence
  $\mathbb{C}_1 \leftarrow \mathbb{C}_0$ ▷ Copy of initial clusters
  for $iter = 1$ to 100 **do**
    $C_j = \{\mathbf{x} \in D \mid \underset{i \in [1,k] (G_i \in \mathbb{C}_0)}{\operatorname{argmax}} K(\delta(\mathbf{x}), P_{G_i}) = j\}, \forall_{j \in [1,k]}$
    $\mathbb{C}_1 = \{C_1, \dots, C_k\}$
    if sum$(\mathbb{C}_1 \neq \mathbb{C}_0) < h$ **then**
      break for ▷ Convergence check
    end if
    $\mathbb{C}_0 \leftarrow \mathbb{C}_1$ ▷ Update cluster assignments
  end for
  return $\mathbb{C} \leftarrow \mathbb{C}_0$ ▷ Refined cluster assignments

Table A1
A comparison (in terms of NMI) with MPSSC [32] on gene-expression datasets.

datasets	#cells(pts)	#genes(dimen)	#cell types(c)	SC			KBC			MPSSC
				NCut	*RCut*	*NSS*	*NCut*	*RCut*	*NSS*	
Ting	114	14405	5	.98	1.00	1.00	.98	.90	.98	.98
Treutlein	80	959	5	.74	.61	.64	.87	.74	.85	.75
Deng	135	12548	7	.71	.70	.74	.77	.69	.78	.76
Ginhoux	251	11834	3	.68	.59	.51	.74	.50	.76	.64
Pollen	249	14805	11	.93	.93	.93	.93	.88	.93	.94
Buettner	182	8989	3	.87	.84	.81	.90	.64	.89	.83
Average NMI				.82	.78	.77	.86	.72	.87	.81
Average RANK				4.25	4.33	4.33	2.58	6.50	2.58	3.42

Appendix D. Experimental settings and detailed results

Like many existing evaluations (e.g., [10,14–16]), we have employed the widely used NMI (normalized mutual information [22] to assess the clustering performance of clustering algorithms.

D.1. Comparison using gene-expression datasets

In recent years, there has been a rapid increase in the use of single-cell sequencing (scRNA-seq) approaches in the field of biology. At the single-cell level, scRNA-seq technology produces genome-wide expression data. One key goal of scRNA-seq analysis is to cluster cells based on gene expression patterns, yielding each cluster consisting of cells that have the similar gene expression patterns.

A recently tailored made framework that employs spectral clustering is called MPSSC [32]; and it has two main steps. First, it produces a learned similarity matrix P from W (the similarity matrix derived directly from Gaussian kernel and the gene expression data). This is deem necessary in this domain in order to deal with high-level of noise while measuring the genome-wide expression [32]. Second, using P as input, MPSSC utilizes sparse spectral clustering [48] and k-means to produce the final clusters. We denote this second step as SSC.

To have a fair comparison with MPSSC, we replace SSC with either SC or KBC in the MPSSC framework; that is, all the three clustering algorithms use the same learned similarity matrix P obtained at the first step of MPSSC. The same six datasets[10] used in [32] are employed in the experiments.

The experimental results are given in Table A1, and we have the following observations. First, the best performers are KBCs that employ the $NCut$ and NSS criteria, and they have very close NMIs in all datasets. They have better NMI than SSC in four out of the five datasets. This result is consistent with the results, run using the 26 datasets, i.e., KBCs that employ the $NCut$ and NSS criteria are better than existing SC algorithms, now including SSC used in MPSSC.

Second, SSC and SC with $NCut$ are the second-best performers; and only two datasets (Deng & Ginhoux) have large differences, where one algorithm is better than the other in one dataset only. This means that SSC is not necessarily better than the standard SC with $NCut$.

D.2. Experimental settings

The parameter search ranges used for each algorithm in the experiments are provided in Table A2.

A guideline to set the threshold parameter τ of KBC is provided here. While the best τ is specific to a dataset, there are some indications to increase/decrease the first setting. For example, if the first setting $\tau = 0.3$ produces the number of initial clusters to be less than k, then this means that many points are connected together. So we need to increase τ. If $\tau = 0.5$ produces the largest k clusters to have too few points, then this means that τ is too large. Thus we need to decrease τ.

[10] There are additional three large datasets [32]. Because they took too long to complete in the first step of MPSSC, we have omitted the comparison with them here.

Table A2
Parameter search ranges. d: #dimensions; n: data size.

Algorithm	Parameter search ranges
SC	$\sigma \in \{2^q \mid q = -5, \dots, 5\} \cup \{d2^q \mid q = -5, \dots, 5\}$
KBC	$\tau \in \{0.05, 0.1, \dots, 0.95\}$; $s = \min(n, 10000)$
Isolation Kernel (ik)	$\psi \in \{2^q \mid q = 1, \dots, 10\}$; $t = 400$.
Gaussian kernel (gk)	σ is the same as SC;
U-SPEC	$k \in \{2, 3, 5, 7, 10, 15, 20, 30, 45, 60, 80\}$; $s = \min(n, 1000)$
U-SENC	$m = 20$; plus U-SPEC settings
DP	$k \in \{2, 3, ..., 20\}$; $\epsilon \in [0.001, 0.002, ..., 0.5]$

Table A3
Data characteristics of the five large image datasets used in Fig. 11.

dataset	#points	#dimension	#clusters
Pendigits	10992	16	10
USPS	11000	256	10
Letters	20000	16	26
MNIST	70000	784	10
Covertype	581012	54	7

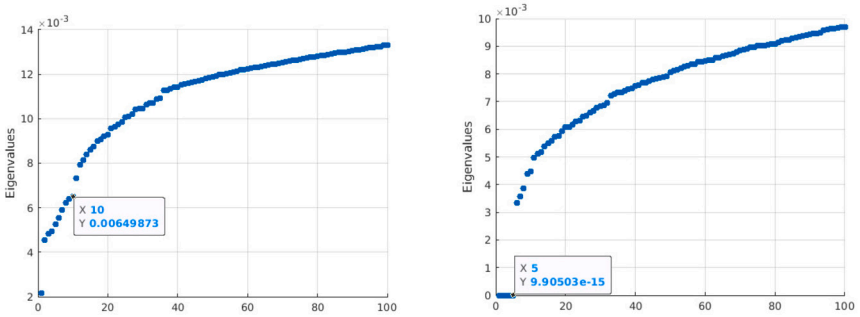


Fig. A.1. An example of the eigenvalues of GLOVE-300 (left) and MJ5 (right). x-axis: index of the eigenvalues. y-axis: eigenvalue.

The G-Strip. Same-Size and High-Low datasets (created according to the description in [5]) and the KBC source code are available in https://github.com/IsolationKernel/Kernel-bounded-clustering-versus-spectral-clustering.

The data characteristics of the five large image datasets used in Fig. 11 are given in Table A3.

D.3. The number of clusters for GLOVE-300 and MJ5

For each of MJ5 and GLOVE-300, we first sample 10,000 points to perform the eigendecomposition. From the first 100 smallest eigenvalues, we select the index i corresponding to the eigenvalue with the large gap as the number of clusters $k >= 5$. Let v_i be the i-th smallest eigenvalue, where $i \in [5, 100]$. The number of clusters is set to $k = i$ when the first $v_{i+1} - v_i$ is relatively large [12]. This is repeated 10 times, and the median is set as the final number of clusters k. An example of the eigenvalues of GLOVE-300 and MJ5 are shown in Fig. A.1. Finally, we choose $k = 10$ for GLOVE-300 and $k = 5$ for MJ5.

D.4. Empirical evaluation using 26 datasets

Tables A4, A5 and A6 show the detailed results of the individual 26 datasets for the summary results shown in Figs. 7, 8 and 9, respectively.

D.5. Stability and parameter sensitivity tests

IKBC and U-SENC rely on subsamples, and thus their results could be unstable. The ik used in IKBC has parameter t which has been used to reduce its variance [21]. U-SENC has parameter m (the number of U-SPEC runs) which may be used to reduce its variance.

Fig. A2 shows two example results on two datasets. It shows that IKBC improves its stability by increasing the t parameter setting. The same applies to U-SENC by increasing m. However, U-SENC has $O(m^2)$ time complexity because of the use of eigendecomposition; while IKBC has $O(t)$ time complexity. This means that U-SENC needs to pay more for improving stability.

Fig. A3 reports the results of the parameter sensitivity test of IKBC, examining the sensitivity of ψ and τ. They show that IKBC is relatively insensitive to either ψ (when searched over a range of τ values) or τ (when searched over a range of ψ values).

Table A4
SC versus KBC for each of the three criteria: $NCut$, $RCut$ and NSS. Each NMI result is an average over 5 trials.

datasets	#points	#dimension	#clusters	SC			KBC		
				$NCut$	$RCut$	NSS	$NCut$	$RCut$	NSS
4C	1000	2	4	.75	.75	.75	1.00	1.00	1.00
AC	1004	2	2	1.00	1.00	.59	1.00	1.00	1.00
aggregation	788	2	7	.98	.99	.75	.94	.88	.90
s3	5000	2	15	.80	.52	.79	.69	.78	.78
RingG	1536	2	4	.66	.66	.80	.80	.80	.78
complex9	3031	2	9	.71	.74	.61	.96	.81	.81
unbalance	6500	2	8	1.00	.70	1.00	1.00	1.00	1.00
cure-t2-4k	4200	2	7	.75	.82	.86	.83	.78	.74
ALLAML	72	7129	2	.10	.02	.11	.17	.09	.17
CLL_SUB_111	111	11340	3	.33	.05	.20	.47	.45	.55
COIL20	1440	1024	20	.92	.55	.77	.85	.83	.84
GLI_85	85	22283	2	.21	.05	.18	.17	.28	.23
GLIOMA	50	4434	4	.52	.47	.48	.51	.59	.60
lung	203	3312	5	.62	.51	.57	.46	.60	.56
TOX_171	171	5748	4	.28	.05	.26	.32	.30	.32
banknote	1372	4	2	.18	.03	.25	.90	.74	.87
ecoli	336	7	8	.65	.09	.63	.60	.70	.70
iris	150	4	3	.77	.59	.79	.83	.81	.85
landsat	2000	36	6	.61	.15	.60	.60	.65	.64
seeds	210	7	3	.75	.47	.76	.73	.76	.76
segment	2310	18	7	.68	.21	.67	.70	.69	.70
vote	435	16	2	.44	.03	.44	.53	.46	.52
waveform3	5000	21	3	.37	.31	.37	.41	.40	.40
G-Strip	1400	2	2	.57	.44	.71	.86	.86	.87
Same-Size	900	2	3	.63	.02	.91	.89	.93	.90
High-Low	900	2	3	.58	.80	.59	.90	.92	.90
Average NMI				.61	.42	.59	.70	.70	.71
Average RANK				3.77	5.33	4.00	2.83	2.65	2.42

Table A5
Kernel k-means (kkm) versus KBC with the $NCut$, NSS and $SDist$ criteria in terms of NMI.

datasets	#points	#dimension	#clusters	$NCut$		NSS		$SDist$	
				kkm	KBC	kkm	KBC	kkm	KBC
4C	1000	2	4	.27	1.00	.29	1.00	.46	1.00
AC	1004	2	2	.59	1.00	.43	1.00	.46	1.00
aggregation	788	2	7	.75	.94	.78	.90	.76	.89
s3	5000	2	15	.71	.69	.73	.78	.72	.64
RingG	1536	2	4	.56	.80	.50	.78	.70	.81
complex9	3031	2	9	.60	.96	.60	.81	.60	.98
unbalance	6500	2	8	.97	1.00	.97	1.00	.97	1.00
cure-t2-4k	4200	2	7	.65	.83	.68	.74	.67	.82
ALLAML	72	7129	2	.12	.17	.11	.17	.09	.17
CLL_SUB_111	111	11340	3	.13	.47	.24	.55	.21	.38
COIL20	1440	1024	20	.66	.85	.66	.84	.67	.85
GLI_85	85	22283	2	.13	.17	.10	.23	.12	.19
GLIOMA	50	4434	4	.48	.51	.49	.60	.49	.52
lung	203	3312	5	.35	.46	.50	.56	.41	.58
TOX_171	171	5748	4	.16	.32	.17	.32	.19	.31
banknote	1372	4	2	.23	.90	.15	.87	.17	.86
ecoli	336	7	8	.58	.60	.58	.70	.58	.55
iris	150	4	3	.65	.83	.72	.85	.73	.74
landsat	2000	36	6	.54	.60	.54	.64	.55	.60
seeds	210	7	3	.64	.73	.63	.76	.62	.73
segment	2310	18	7	.54	.70	.60	.70	.61	.72
vote	435	16	2	.41	.53	.33	.52	.34	.49
waveform3	5000	21	3	.37	.41	.31	.40	.32	.40
G-Strip	1400	2	2	.64	.86	.52	.87	.57	.86
Same-Size	900	2	3	.71	.89	.65	.90	.72	.95
High-Low	900	2	3	.46	.90	.52	.90	.55	.91
Average NMI				.50	.70	.49	.71	.51	.69
Average RANK				5.00	2.15	5.04	1.79	4.58	2.44

Data availability

The datasets are sourced from:
(a) https://archive.ics.uci.edu/ml/datasets.php;
(b) https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets;

Table A6
Gaussian Kernel versus Isolation Kernel in $NCut$-SC, NSS-KBC and $SDist$-KBC; and a comparison with U-SPEC, U-SENC & DP in terms of NMI.

datasets	#points	#dimension	#clusters	$NCut$-SC		$NCut$-KBC		NSS-KBC		$SDist$-KBC		U-SPEC	U-SENC	DP
				gk	ik	gk	ik	gk	ik	gk	ik			
4C	1000	2	4	.75	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.85
AC	1004	2	2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
aggregation	788	2	7	.98	.95	.94	.91	.90	.96	.89	.98	.97	.99	.99
s3	5000	2	15	.80	.80	.94	.80	.78	.80	.64	.80	.80	.80	.80
RingG	1536	2	4	.66	.98	.80	.99	.78	.98	.81	.98	.78	.99	.82
complex9	3031	2	9	.71	.98	.96	1.00	.81	1.00	.98	1.00	.92	.92	.86
unbalance	6500	2	8	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
cure-t2-4k	4200	2	7	.75	.92	.83	.95	.74	.95	.82	.95	.88	.89	.90
ALLAML	72	7129	2	.10	.16	.17	.31	.17	.22	.17	.22	.15	.09	.32
CLL_SUB_111	111	11340	3	.33	.20	.47	.44	.55	.42	.38	.42	.18	.17	.43
COIL20	1440	1024	20	.92	.94	.85	.98	.84	.98	.85	.96	.93	.95	.89
GLI_85	85	22283	2	.21	.23	.17	.29	.23	.29	.19	.34	.25	.12	.36
GLIOMA	50	4434	4	.52	.54	.51	.62	.60	.62	.52	.62	.42	.26	.51
lung	203	3312	5	.62	.69	.46	.72	.56	.68	.58	.74	.74	.58	.64
TOX_171	171	5748	4	.28	.40	.32	.39	.32	.33	.31	.32	.19	.21	.26
banknote	1372	4	2	.18	.75	.90	.87	.87	.89	.86	.84	.61	.61	.96
ecoli	336	7	8	.65	.59	.60	.70	.70	.63	.55	.65	.64	.53	.61
iris	150	4	3	.77	.86	.83	.88	.85	.85	.74	.82	.84	.69	.87
landsat	2000	36	6	.61	.66	.60	.67	.64	.64	.60	.62	.64	.66	.61
seeds	210	7	3	.75	.72	.73	.76	.76	.76	.73	.76	.72	.70	.72
segment	2310	18	7	.68	.67	.70	.70	.70	.75	.72	.67	.28	.68	.53
vote	435	16	2	.44	.50	.53	.54	.52	.55	.49	.52	.51	.51	.54
waveform3	5000	21	3	.37	.37	.41	.48	.40	.40	.40	.40	.37	.37	.37
G-Strip	1400	2	2	.57	.87	.86	.88	.87	.89	.86	.89	.87	.80	.89
Same-Size	900	2	3	.63	.94	.89	.94	.90	.94	.95	.94	.91	.92	.92
High-Low	900	2	3	.58	.63	.90	.91	.90	.93	.91	.95	.68	.90	.93
Average NMI				.61	.71	.71	.76	.71	.75	.69	.75	.66	.67	.71
Average RANK				8.27	6.06	6.52	3.23	6.33	3.67	7.17	4.21	7.25	7.65	5.63

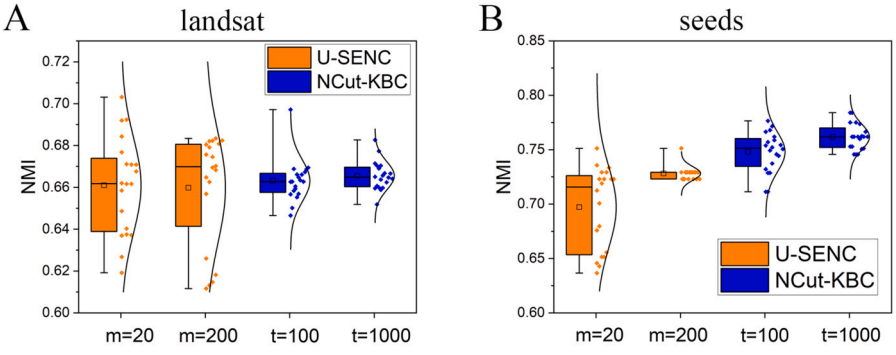


Fig. A2. Stability test for $NCut$-KBC and U-SENC. Each box is produced from 20 trials.

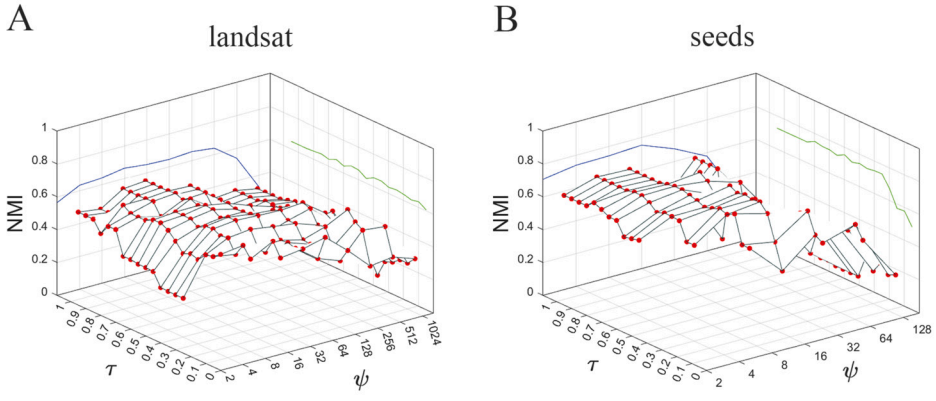


Fig. A3. Parameter sensitivity analysis of $NCut$-KBC. The blue and green lines are the best clustering results in terms of NMI for a ψ setting (i.e., $\max_{\tau}(\text{KBC}(D,\psi))$) and that for a τ setting (i.e., $\max_{\psi}(\text{KBC}(D,\tau))$), respectively.

(c) https://github.com/milaan9/Clustering-Datasets;

References

[1] J.A. Hartigan, M.A. Wong, Algorithm as 136: a k-means clustering algorithm, J. R. Stat. Soc., Ser. C, Appl. Stat. 28 (1979) 100–108.
[2] 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.
[3] W. Donath, A. Hoffman, Algorithms for partitioning graphs and computer logic based on eigenvectors of connection matrices, IBM Tech. Dis. Bull. 15 (1972) 938–944.
[4] M. Fiedler, Algebraic connectivity of graphs, Czechoslov. Math. J. 23 (1973) 298–305.
[5] B. Nadler, M. Galun, Fundamental limitations of spectral clustering, in: Proceedings of the 19th International Conference on Neural Information Processing Systems, 2006, pp. 1017–1024.
[6] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: Advances in Neural Information Processing Systems, 2005, pp. 1601–1608.
[7] 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.
[8] Jianbo Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 888–905.
[9] L. Hagen, A.B. Kahng, New spectral methods for ratio cut partitioning and clustering, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 11 (1992) 1074–1085.
[10] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, C.-K. Kwoh, Ultra-scalable spectral clustering and ensemble clustering, IEEE Trans. Knowl. Data Eng. 32 (2020) 1212–1226.
[11] L. Bai, M. Qi, J. Liang, Spectral clustering with robust self-learning constraints, Artif. Intell. 320 (2023) 103924.
[12] U. von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (2007) 395–416.
[13] S. Guattery, G.L. Miller, On the quality of spectral separators, SIAM J. Matrix Anal. Appl. 19 (1998) 701–719.
[14] H. Li, X. Ye, A. Imakura, T. Sakurai, Divide-and-conquer based large-scale spectral clustering, Neurocomputing 501 (2022) 664–678.
[15] J. Xie, W. Kong, S. Xia, G. Wang, X. Gao, An efficient spectral clustering algorithm based on granular-ball, IEEE Trans. Knowl. Data Eng. 35 (2023) 9743–9753.
[16] P. Macgregor, Fast and simple spectral clustering in theory and practice, Adv. Neural Inf. Process. Syst. 36 (2023) 34410–34425.
[17] L. Lovász, Random walks on graphs: a survey, in: Combinatorics, Paul Erdös Is Eighty, János Bolyai Math. Soc., Budapest, 1993, pp. 353–397.
[18] G.W. Stewart, J. Sun, Matrix Perturbation Theory, Academic, New York, 1990.
[19] P.W. Holland, K.B. Laskey, S. Leinhardt, Stochastic blockmodels: first steps, Soc. Netw. 5 (1983) 109–137.
[20] 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.
[21] 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.
[22] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance, J. Mach. Learn. Res. 11 (2010) 2837–2854.
[23] J. Liu, C. Wang, M. Danilevsky, J. Han, Large-scale spectral clustering on graphs, in: The Twenty-Third International Joint Conference on Artificial Intelligence, 2013.
[24] X. Yang, M. Zhu, Y. Cai, Z. Wang, F. Nie, Fast spectral clustering with self-adapted bipartite graph learning, Inf. Sci. 644 (2023) 118810.
[25] G. Yang, S. Deng, X. Chen, C. Chen, Y. Yang, Z. Gong, Z. Hao, RESKM: a general framework to accelerate large-scale spectral clustering, Pattern Recognit. 137 (2023) 109275.
[26] C. Gao, W. Chen, F. Nie, W. Yu, Z. Wang, Spectral clustering with linear embedding: a discrete clustering method for large-scale data, Pattern Recognit. 151 (2024) 110396.
[27] 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 (2010) 568–586.
[28] E. Hohma, C.M. Frey, A. Beer, T. Seidl, Scar: spectral clustering accelerated and robustified, Proc. VLDB Endow. 15 (2022) 3031–3044.
[29] L.R. Ford, D.R. Fulkerson, Maximal flow through a network, Can. J. Math. 8 (1956) 399–404, https://doi.org/10.4153/CJM-1956-045-5.
[30] A. Smola, A. Gretton, L. Song, B. Schölkopf, A Hilbert space embedding for distributions, in: M. Hutter, R.A. Servedio, E. Takimoto (Eds.), Algorithmic Learning Theory, Springer Berlin Heidelberg, 2007, pp. 13–31.
[31] K.M. Ting, Y. Zhu, Z.-H. Zhou, Isolation kernel and its effect on SVM, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 2329–2337.
[32] S. Park, H. Zhao, Spectral clustering based on learning similarity matrix, Bioinformatics 34 (2018) 2069–2076.
[33] D.U. Pizzagalli, S.F. Gonzalez, R. Krause, A trainable clustering algorithm based on shortest paths from density peaks, Sci. Adv. 5 (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 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, vol. 96, 1996, pp. 226–231.
[35] 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.
[36] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492–1496.
[37] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. (2009) 224–227.
[38] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat., Theory Methods 3 (1974) 1–27.
[39] S. Sarkar, P. Soundararajan, Supervised learning of large perceptual organization: graph spectral partitioning and learning automata, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 504–525.
[40] C.H.Q. Ding, X. He, H. Zha, M. Gu, H.D. Simon, A min-max cut algorithm for graph partitioning and data clustering, in: Proceedings of the IEEE International Conference on Data Mining, 2001, pp. 107–114.
[41] R. Jarvis, E. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Trans. Comput. C-22 (1973) 1025–1034.
[42] L. Ertöz, M. Steinbach, V. Kumar, Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data, in: Proceedings of SIAM Data Mining Conference, 2003, pp. 47–58.
[43] J. Han, J. Pei, H. Tong, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2022.
[44] A. Hinneburg, D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1998, pp. 58–65.
[45] 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.
[46] V.M. Panaretos, Y. Zemel, Statistical aspects of Wasserstein distances, Annu. Rev. Stat. Appl. 6 (2019) 405–431.
[47] T. Van Erven, P. Harremos, Rényi divergence and Kullback-Leibler divergence, IEEE Trans. Inf. Theory 60 (2014) 3797–3820.
[48] C. Lu, S. Yan, Z. Lin, Convex sparse spectral clustering: single-view to multi-view, IEEE Trans. Image Process. 25 (2016) 2833–2843.