Nonlinear Dimensionality Reduction for Clustering

Sotiris Tasoulis[a,*], Nicos G. Pavlidis[b], Teemu Roos[c]

[a] *Department of Computer Science and Biomedical Informatics, University of Thessaly, Greece.*
[b] *Department of Management Science, Lancaster University, UK.*
[c] *Department of Computer Science, University of Helsinki, Finland.*

Abstract

We introduce an approach to divisive hierarchical clustering that is capable of identifying clusters in nonlinear manifolds. This approach uses the isometric mapping (Isomap) to recursively embed (subsets of) the data in one dimension, and then performs a binary partition designed to avoid the splitting of clusters. We provide a theoretical analysis of the conditions under which contiguous and high-density clusters in the original space are guaranteed to be separable in the one-dimensional embedding. To the best of our knowledge there is little prior work that studies this problem. Extensive experiments on simulated and real data sets show that hierarchical divisive clustering algorithms derived from this approach are effective.

Keywords: Nonlinearity, Dimensionality Reduction, Divisive Hierarchical Clustering, Manifold Clustering

1. Introduction

Dimensionality reduction is a central component in clustering high dimensional data. Well established methods like Principal Component Analysis (PCA) and metric Multi-Dimensional Scaling (MDS) [1] have been shown to be effective in a plethora of applications [2], despite the fact that neither method is guaranteed to preserve the cluster structure in the data. More recently, a number of

*Corresponding author
Email addresses: stasoulis@uth.gr (Sotiris Tasoulis), n.pavlidis@lancaster.ac.uk (Nicos G. Pavlidis), teemu.roos@cs.helsinki.fi (Teemu Roos)

linear projection pursuit methods have been proposed that explicitly optimize a clustering objective to identify appropriate subspaces for clustering. Such works have considered optimizing objectives related to k-means clustering [3, 4], density clustering [5], and spectral connectivity [6, 7, 8], Even if a low dimensional subspace in which all clusters can be identified exists, establishing the dimensionality of this subspace remains an open problem (for clustering algorithms other than k-means, for known k). Furthermore, in a number of applications such as computer vision, and image processing, clusters are defined in multiple subspaces [9]. In this case there might not even exist a single subspace that is appropriate to identify all clusters. To overcome this issue hierarchical divisive clustering algorithms have been proposed that determine the binary partition at each level of the hierarchy from a one-dimensional projection [5, 4, 7]. This approach is equivalent to splitting each cluster in the hierarchy through a hyperplane. However, hyperplane separators are inherently inappropriate when the clusters are not linearly separable.

Nonlinear dimensionality reduction techniques have been explicitly designed to handle high dimensional data that lie in or close to a manifold of intrinsically low dimension. Widely used manifold learning methods include isometric mapping (Isomap) [10], Kernel PCA [11], Locally Linear Embedding (LLE) [12], and its variants such as Laplacian Eigenmaps (LE) [13]. These methods share a common framework: First the neighbors of each observation are determined, and are used to determine pairwise distances. Then the nonlinear transformation between the original points and their low-dimensional embedding is obtained through the eigenvectors of an appropriately defined matrix [14]. Nonlinear dimensionality reduction methods can be categorised into *global* and *local*. Global methods, such as Isomap and Kernel PCA, attempt to preserve global spatial relationships in the data. In contrast, local methods such as LLE prioritise the preservation of distances within small neighborhoods of points. Although, visualization in two and three dimensional space can be significantly improved using local methods, there is no clear evidence that these also enhance clustering performance.

Manifold learning techniques have been widely used in clustering but to the best of our knowledge there is very little work that addresses the question of whether and under which conditions clusters (of any type) are preserved after applying nonlinear dimensionality reduction. In this work we investigate conditions under which two types of clusters (contiguous and density clusters) are preserved after a one-dimensional embedding through Isomap. This motivates the development of divisive hierarchical clustering algorithms that rely on one-dimensional embeddings and can handle nonlinearly separable clusters when the number of dimensions is high relative to the number of samples.

Related Work: Density based clustering algorithms like the established DBSCAN [15] and the more recent pdfCluster [16] and densityPeaks [17], are by design capable of identifying nonlinear clusters. However, they are rarely effective when the number of dimensions increases even in the scale of a few tens. The recently proposed cut-edge spatial clustering (CutESC) [18] can identify clusters of different densities and arbitrary shapes, and has proven competitive against other density based methods. CutESC identifies clusters by partitioning an appropriate constricted proximity graph. However, this algorithm has been mainly tested on relatively low dimensional datasets. The authors in [19] present a novel approach for identifying local high-density samples utilizing the inherent properties of the nearest neighbor graph. After using the density estimator to filter noise samples, the proposed algorithm performs a DBSCAN-like clustering process. This algorithm has also been mainly assessed on relatively low dimensional datasets. In [20] a method with promising results for high dimensional nonlinear clustering problems is proposed. The algorithm first generates several tight and small subclusters and then merges these by exploiting the connectivity among them. To select appropriate values for the parameters of the algorithm the authors propose an internal validity index whose relation to the actual clustering structure is not known.

Kernel k-means [21] is among the most popular methods for nonlinear clustering. Kernel methods project the data into a high dimensional feature space in which clusters are separable [11]. Kernel k-means applies the well known

k-means algorithm on the feature space. The main limitation of this approach is the difficulty of specifying an appropriate kernel for a particular dataset. An alternative and very recent approach to identify clusters defined in Riemannian submanifolds, is the diffusion k-means algorithm [22]. This is achieved by constructing a random walk on an appropriate similarity graph.

More generally, a number of nonlinear clustering algorithms have been proposed under the generic term *spectral clustering* (SC) [23, 24]. These methods share the basic idea of applying a nonlinear dimension reduction step in order to aid the extraction of cluster structures, and are closely related to LE [13]. In [25] the authors propose a modification to the normalised cut criterion, and develop an iterative method with proved convergence to obtain the optimal solution that does not involve eigendecomposition. A spectral clustering approach for multiview data which performs simultaneously graph fusion and spectral clustering has been recently proposed [26]. As in the case of kernel k-means a critical choice in SC is the specification of the kernel and its parameters which are used to construct the Laplacian matrix [24].

The authors in [27] applied k-means after embedding the data through Isomap. They observed that the resulting method failed even in simple artificial examples. They then proposed a modified definition of the geodesic distances but concluded that this was also unsatisfactory in real-world datasets where the data is noisy, or the clusters are highly nonlinear.

In this paper we develop methods that explicitly leverage theoretical properties of the cluster structures and their low-dimensional (in fact, one-dimensional) embeddings. In the experimental section, we demonstrate that in many cases this has a significant positive effect on the resulting clustering.

The rest of the paper is organized as follows. In Section 2 we give a brief description of the Isomap algorithm and its variant used in this work. In Section 3 we develop the theoretical background and propose new clustering algorithms. Section 4 is devoted to experimental evaluation of the proposed approaches. The paper ends with concluding remarks in Section 5.

2. The Embedding Algorithm

Let $\mathcal{D}$ denote the set of d-dimensional observations, $\mathcal{D} = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, and $X \in R^{n \times d}$ the associated data matrix. The first step of the Isomap algorithm is to create an undirected graph that represents the connectivity between observations. There are two methods to determine the *neighbors* of an observation. According to the first two points are neighbors if one of them is amongst the k-nearest neighbors of the other, for a pre-specified value of k, and choice of distance metric $d_X(x_i, x_j)$. The default choice of distance metric is the Euclidean distance, $d_X(x_i, x_j) = \|x_i - x_j\|_2$. In the second method, two points are considered neighbors if $d_X(x_i, x_j) \leqslant \epsilon$, for a pre-specified $\epsilon > 0$. The resulting undirected graph can be equivalently represented in terms of the adjacency matrix, $W \in \mathbb{R}^{n \times n}$, where $W_{i,j} > 0$ if and only if x_i and x_j are connected, and for all pairs of connected observations $W_{i,j} = d_X(x_i, x_j)$. From this graph the *geodesic distance* between two points, $G_{i,j} = d_G(x_i, x_j)$, is defined as the length of the shortest path connecting x_i to x_j on the graph. Isomap applies MDS to the matrix of geodesic distances G to compute an optimal embedding into a low-dimensional space where the pairwise distances match the geodesic distances as well as possible.

A shown in [28] both approaches to build the neighbor graph are subject to instability. More precisely, selecting a large value for k or ϵ can lead to the so called "short-cut edges", which are misleading connections between different folds of the manifold, that render the embedding inaccurate. On the other hand, too small values for k or ϵ can create disconnected subgraphs in which case the Isomap algorithm is not directly applicable. The latter problem pertains especially to clustering tasks where different clusters may be far from each other. To deal with this problem, Orsenigo and Vercellis [14] propose a variant of Isomap called dbt-Isomap. In dbt-Isomap if the original neighborhood graph is disconnected, the separate subgraphs are then connected by creating a minimum spanning tree among the centroids of the unconnected subgraphs. A visual example of this method is given in Figure 1 where we employ an artificial two

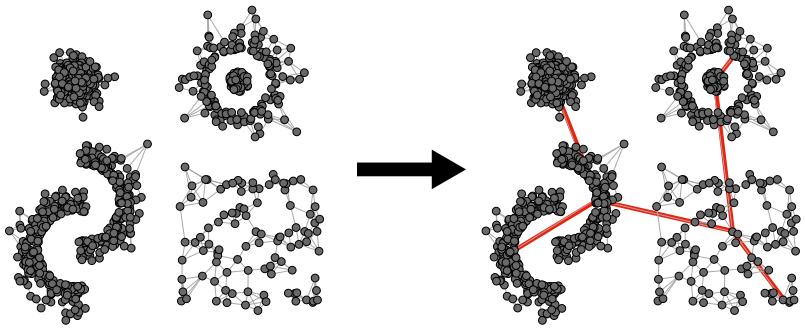

Figure 1: The initial disconnected k-NN graph (left) and the resulting connected graph (right) using the dbt-Isomap method. The edges added by dbt-Isomap are indicated with thicker red lines on the graph on the right.

dimensional dataset specially designed in a way so that building the k-nearest neighbors graph for a relative small k value ($k = 5$ for this particular example) will result in several unconnected components (left part of Figure 1). At the right part of the Figure 1 we can see the added undirected edges in red color within the resulting fully connected graph.

3. Proposed Framework

In this section, we provide theoretical results characterizing the relationship between true clusters in the full-dimensional space and their one-dimensional embeddings. Based on specific assumptions about the properties of clusters, we propose appropriate methodologies to identify them. Throughout we assume that using as input a finite set of points $\mathcal{D}$, the dbt-Isomap algorithm produces the undirected weighted graph $G(\mathcal{D}) = (\mathcal{D}, \mathcal{E})$, which we call the *neighborhood graph*. The vertices of $G(\mathcal{D})$ correspond to the set of points. The edge (x_i, x_j) exists, only if x_i and x_j are connected through the Isomap algorithm. The weight of every edge in $G(\mathcal{D})$ is equal to the Euclidean distance between the corresponding points, $W_{ij} = \|x_i - x_j\|_2$ if $(x_i, x_j) \in \mathcal{E}$. The geodesic distance between any pair of observations, $d_G(x_k, x_l)$, is defined as the length of the shortest path connecting x_k to x_l on $G(\mathcal{D})$. Therefore, $d_G(x_k, x_l) \geqslant \|x_k -$

$x_l\|_2$ and since $G(\mathcal{D})$ is typically very sparse for the majority of pairs of points $d_G(x_k, x_l) > \|x_k - x_l\|_2$.

3.1. Contiguous Clusterability

Before investigating the theoretical properties of a clustering algorithm we need to formally define a cluster with respect to the neighborhood graph $G(\mathcal{D})$. In the following for every non-empty set $\mathcal{C} \subset \mathcal{D}$, we define $G(\mathcal{C}) = (\mathcal{C}, \mathcal{E}_{\mathcal{C}})$ as the subgraph obtained by removing from $G(\mathcal{D})$ the vertices in $\mathcal{D} \setminus \mathcal{C}$ and all the edges with an end-point in $\mathcal{D} \setminus \mathcal{C}$,

$$\mathcal{E}_{\mathcal{C}} = \{(x_i, x_j) \,|\, x_i, x_j \in \mathcal{C}, (x_i, x_j) \in \mathcal{E}\}\,.$$

We first consider *contiguous* clusters [29]. To define this type of cluster we first need to define the *coherence* of a non-empty subset of $\mathcal{D}$, and the coherence of a partition of $\mathcal{D}$ into pairwise disjoint subsets.

Definition 1. (**Set Coherence**): Let $\mathcal{C} \subset \mathcal{D}$ be a non-empty subset of $\mathcal{D}$, and $G(\mathcal{D})$ the associated neighborhood graph. The coherence of $\mathcal{C}$, $\mathrm{coh}(\mathcal{C})$, is defined as,

$$\mathrm{coh}(\mathcal{C}) = \begin{cases} \max\{\|x_i - x_j\|_2 \,|\, (x_i, x_j) \in \mathcal{E}_{\mathcal{C}}\}\,, & \text{if } G(\mathcal{C}) \text{ is connected,} \\ \infty, & \text{otherwise.} \end{cases} \tag{1}$$

Definition 2. (**Partition Coherence and Separability**): Let $\mathcal{D}$ be a set of points and $G(\mathcal{D})$ the associated neighborhood graph. The coherence of a partition of $\mathcal{D}$, into k non-empty and pairwise disjoint subsets, $\Pi = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$, is the maximum coherence of any of the elements of the partition,

$$\mathrm{coh}(\Pi) = \max_{m=1,\ldots,k} \mathrm{coh}(\mathcal{C}_m),$$

while the separability of Π is defined as,

$$\mathrm{sep}(\Pi) = \min_{m=1,\ldots,k} \min_{\substack{x_i \in \mathcal{C}_m, \\ x_j \in \mathcal{D} \setminus \mathcal{C}_m}} d_G(x_i, x_j). \tag{2}$$

Definition 3. (**Contiguous Set**): Let $\mathcal{D}$ be a set of points and $G(\mathcal{D})$ the associated neighborhood graph. A non-empty set $\mathcal{C} \subset \mathcal{D}$, is a contiguous set if for all $x_i \in \mathcal{D} \setminus \mathcal{C}$,

$$\text{coh}(\mathcal{C}) < \text{coh}(\mathcal{C} \cup \{x_i\}). \tag{3}$$

We are now able to define a *k-contiguous clusterable set*, and a *k-contiguous clustering*.

Definition 4. (***k*-Contiguous Clustering**): Let $\mathcal{D}$ be a set of points and $G(\mathcal{D})$ the associated neighborhood graph. If there exists a partition $\Pi = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ into k non-empty and pairwise disjoint subsets of $\mathcal{D}$ such that $\text{coh}(\Pi) < \infty$ and $\text{sep}(\Pi) > \text{coh}(\Pi)$, then $\mathcal{D}$ is *k-contiguous clusterable*, and Π is a *k-contiguous clustering*.

A k-contiguous clustering with maximum separability,

$$\begin{aligned} \Pi^\star = & \underset{\Pi=\{\mathcal{C}_1,\ldots,\mathcal{C}_k\}}{\arg\max} \ \text{sep}(\Pi), \\ \text{s.t.} \quad & \mathcal{D} = \cup_{m=1}^{k} \mathcal{C}_m, \\ & \mathcal{C}_m \cap \mathcal{C}_l = \emptyset \ \text{ for all } \ m \neq l, \end{aligned} \tag{4}$$

is called an *optimal k-contiguous clustering*.

Next we state and derive the main result of this section which provides conditions under which contiguous clusters in the full dimensional space are recovered by a simple splitting criterion that involves only the one-dimensional embeddings.

Theorem 1. *Let $\mathcal{D} = \{x_i\}_{i=1}^n$ be a k-contiguous clusterable set for some $k > 1$, and $\Pi^\star = \{\mathcal{C}_1^\star, \ldots, \mathcal{C}_k^\star\}$ the optimal k-contiguous clustering of $\mathcal{D}$. Denote as $\mathcal{P} = \{p_i\}_{i=1}^n$ the set of one-dimensional embeddings of the points in $\mathcal{D}$.*

If there exists $\varepsilon > 0$ such that for all $x_i, x_j \in \mathcal{D}$ with $(x_i, x_j) \in \mathcal{E}$,

$$\|x_i - x_j\|_2 \leqslant \text{coh}(\Pi^\star) \Rightarrow |p_i - p_j| \leqslant \varepsilon, \tag{5}$$

and for any $x_i, x_j \in \mathcal{D}$,

$$d_G(x_i, x_j) > \text{coh}(\Pi^\star) \Rightarrow |p_i - p_j| > \varepsilon, \tag{6}$$

then, partitioning the data at the largest gap between consecutive points in $\mathcal{P}$ induces a binary partition which is guaranteed not to split any contiguous cluster.

Proof. Consider a pair of consecutive points that belong to different clusters, $x_i \in \mathcal{C}_m$ and $x_j \notin \mathcal{C}_m$. By the definition of a k-contiguous clusterable set $d_G(x_i, x_j) > \text{coh}(\Pi^\star)$, and the condition in (6) ensures that $|p_i - p_j| > \varepsilon$.

Consider now the case of two consecutive points from the same cluster, $x_i, x_j \in \mathcal{C}_m$. If x_i is connected to x_j in $G(\mathcal{D})$, then $\|x_i - x_j\|_2 \leqslant \text{coh}(\Pi^\star)$ and the condition in (5) ensures that $|p_i - p_j| \leqslant \varepsilon$. Assume instead that $(x_i, x_j) \notin \mathcal{E}$. Since $\text{coh}(\mathcal{C}_m) < \infty$ the subgraph $G(\mathcal{C}_m)$ is connected. Therefore there exists at least one path in $G(\mathcal{C}_m)$ that connects x_i to x_j. Any such path on $G(\mathcal{C}_m)$ contains an edge (x_k, x_l) (where it is possible that either $x_k = x_i$, or $x_l = x_j$) that satisfies $\|x_k - x_l\|_2 \leqslant \text{coh}(\Pi^\star)$. Therefore,

$$|p_i - p_j| \leqslant |p_k - p_l| \leqslant \varepsilon.$$

The first inequality follows from the fact that p_i and p_j are consecutive, but not directly connected in $G(\mathcal{D})$, while the second is due to the condition in (5). □

The above theorem states that if $\mathcal{D}$ is k-contiguous clusterable, for any $k > 1$ and the two conditions in (5) and (6) hold, then splitting at the largest gap between two consecutive one-dimensional embeddings is guaranteed to avoid splitting any contiguous cluster. Figure 2 provides a visualization of this result. The condition in (5) requires that pairs of points from the same cluster that are connected by an edge in $G(\mathcal{D})$ are embedded within a distance of at most ε of each other. In Figure 2 $|p_1 - p_2|$ is the maximum distance between the embeddings of any pair of points that are connected with an edge in $G(\mathcal{D})$ and belong to the same contiguous cluster. Therefore, in this example any $x_i, x_j \in \mathcal{C}_m$, where $m = \{1, 2\}$, such that $(x_i, x_j) \in \mathcal{E}$, $|p_i - p_j| \leqslant |p_1 - p_2|$. The condition in (6) has two implications. Let x_i, x_j be a pair of points in different clusters, and p_i, p_j their one-dimensional embeddings. If x_i, x_j are connected by an edge in $G(\mathcal{D})$ then $d_G(x_i, x_j) = \|x_i - x_j\|_2 > \text{coh}(\Pi^\star)$. (The last inequality follows from the definition of a k-contiguous clusterable set.) According to (6) the embeddings of

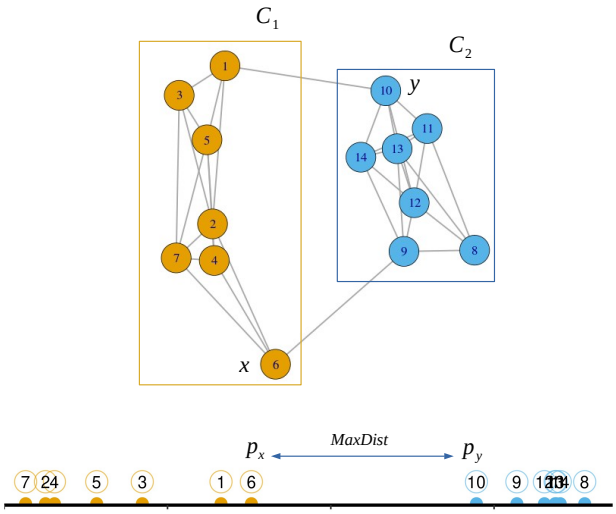

Figure 2: The top figure depicts the neighbor graph created by dbt-Isomap, using $k = 2$. Points from the two contiguous clusters are indicated with different color. At the bottom the one-dimensional embeddings are illustrated. Splitting at the largest gap between consecutive embeddings avoids splitting a contiguous cluster.

points from different clusters that are connected in $G(\mathcal{D})$ are more than ε apart. In the example of Figure 2 there are two edges that connect points from different clusters, (x_1, x_{10}) and (x_6, x_9). The embeddings of the corresponding point satisfy $|p_{10} - p_1| > |p_1 - p_2|$ and $|p_6 - p_9| > |p_1 - p_2|$. Assume instead that x_i, x_j are not connected in $G(\mathcal{D})$. In this case every path in $G(\mathcal{D})$ that connects x_i to x_j has at least one edge (x_k, x_l) such that $d_G(x_k, x_l) = \|x_k - x_l\|_2 > \text{coh}(\Pi^\star)$. The definition of geodesic distance ensures that $d_G(x_i, x_j) > \|x_k - x_l\|_2$, and for such points the condition in (6) requires that $|p_i - p_j| > \varepsilon$. In Figure 2 the distance between the embeddings of any pair of points from different clusters and not connected by an edge satisfies $|p_i - p_j| > |p_1 - p_2|$. Therefore, for the example in this figure $\varepsilon = |p_1 - p_2| > 0$ satisfies the conditions in (5) and (6).

Clearly a user that wants to estimate a clustering of $\mathcal{D}$ does not know the optimal k-contiguous clustering, $\Pi^\star$. Therefore it is not feasible to assess in advance whether the conditions in (5) and (6) are satisfied. It is however possible

to verify these conditions after observing the partition, $\hat{\Pi} = \{\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_m\}$, produced by the proposed clustering algorithm (described in Section 3.3). First it is straightforward to verify whether each cluster is a contiguous set and whether $\text{sep}(\hat{\Pi}) > \text{coh}(\hat{\Pi})$, using Eqs. (3) and (2), respectively. This allows the user to assess whether $\hat{\Pi}$ is a k-contiguous clustering. If $\hat{\Pi}$ is a k-contiguous clustering then it is also straightforward to verify whether an $\varepsilon > 0$ that satisfies conditions (5) and (6) exists. What is not feasible is to determine whether $\hat{\Pi}$ is the optimal k-contiguous clustering since to estimate $\Pi^\star$ requires solving the combinatorial optimization problem in Eq. (4).

3.2. Density Clustering

The contiguous cluster definition in the previous subsection is appropriate when clusters are well separated. In the presence of noise and outliers this is not the case. To accommodate such datasets we consider *high-density clusters (at level ρ)* as defined by Hartigan [30, Ch. 11]. For brevity we refer to such clusters as *ρ-dense clusters*. For continuous data, ρ-dense clusters are defined as follows.

Definition 5. [30, Ch. 11] (**ρ-dense cluster):** Assume that $\mathcal{D}$ is an i.i.d. sample of a random variable on $\mathbb{R}^d$ with unknown probability density function $f : \mathbb{R}^d \to \mathbb{R}_+$. For $\rho \geqslant 0$ and a choice of density estimator, $\hat{f}$, a ρ-dense cluster is a maximally connected component of the superlevel set of $\hat{f}$,

$$L(\rho; \hat{f}) = \left\{ x \in \mathbb{R}^d \,\middle|\, \hat{f}(x) > \rho \right\}.$$

This cluster definition underlies the influential DBSCAN algorithm [15], which uses a uniform kernel density estimator as $\hat{f}$. By definition, ρ-dense clusters are separated from each other by contiguous regions of low density, $\hat{f}(x) < \rho$. It is therefore possible (depending on the choice of ρ) that only a strict subset of $\mathcal{D}$ is allocated to ρ-dense clusters, that is $L(\rho; \hat{f}) \subsetneq \mathcal{D}$. The remaining points, for which $\hat{f}(x_i) < \rho$, are typically considered to be *noise points*. Density clustering algorithms either do not assign these observations to a cluster

(effectively defining a separate "noise" cluster) [15], or use a heuristic to assign each of these to the "closest" cluster [16].

To define ρ-dense clusters on a neighborhood graph we first define the *geodesic probability mass function*.

Definition 6. (**Geodesic Probability Mass Function**): Given a set of points $\mathcal{D} \subset \mathbb{R}^d$ and the associated neighborhood graph $G(\mathcal{D})$, the probability mass at $x_i \in \mathcal{D}$ is,

$$\hat{f}_G(x_i; \mathcal{D}, h) = \frac{1}{c_0} \sum_{j=1}^{n} K(d_G(x_i, x_j), h),$$

where K is a symmetric kernel, $h > 0$ is the scaling (or bandwidth) parameter of this kernel, and c_0 is a normalizing constant which ensures that $\sum_{x_i \in \mathcal{D}} \hat{f}_G(x_i; \mathcal{D}, h) = 1$.

Several choices of K are available, but typically, the quality of the approximation depends less on the shape of the kernel and more on the value of the bandwidth, h. In this paper we use the uniform kernel,

$$\hat{f}_G(x_i; \mathcal{D}, h) = \frac{1}{c_0} \sum_{j=1}^{n} \mathbf{1}_{[0,1]} \left(d_G(x_i, x_j)/h \right), \quad x_i \in \mathcal{D} \tag{7}$$

$$c_0 = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}_{[0,1]} \left(d_G(x_i, x_j)/h \right), \tag{8}$$

where $\mathbf{1}_{\mathcal{A}}(x)$ is the indicator function which takes the value one if $x \in \mathcal{A}$, and zero otherwise. The central property of the uniform kernel is that $\hat{f}_G(x_i; \mathcal{D}, h)$ is proportional to the number of points in $\mathcal{D}$ that are within a geodesic distance of h from x_i. The normalizing constant, defined in Eq. (8), is equal to the total number of neighbors within a geodesic distance of h over all the points in $\mathcal{D}$. We can thus define the superlevel set of $\hat{f}_G$ as,

$$L_G(\rho\,; \hat{f}_G(\cdot; \mathcal{D}, h)) = \left\{ x \in \mathcal{D} \,|\, \hat{f}_G(x; \mathcal{D}, h) > \rho \right\}. \tag{9}$$

We are now in a position to define a (k, ρ)-dense clusterable set.

Definition 7. ((k, ρ)**-Dense Clusterable Set**): A set $\mathcal{D}$ with associated neighborhood graph $G(\mathcal{D})$ and geodesic probability mass function, $\hat{f}_G(\cdot; \mathcal{D}, h)$, is

(k, ρ)-dense clusterable if the ρ-superlevel set of $\hat{f}_G$ can be decomposed into k non-empty pairwise disjoint sets, $L(\rho; \hat{f}_G) = \bigcup_{i=1}^{k} \mathcal{C}_i$, and $G(\mathcal{C}_i)$ is connected for all $i = 1, \ldots, k$.

After $\mathcal{D}$ is embedded in the one-dimensional Euclidean space through dbt-Isomap, the density of any point in the real line can be estimated using a kernel density estimator. We will also use the uniform kernel for this purpose,

$$\hat{f}_{1D}(y; \mathcal{P}, h_1) = \frac{1}{2nh} \sum_{i=1}^{n} \mathbf{1}_{[0,1]} \left(|y - p_i| / h_1 \right), \tag{10}$$

where $\mathcal{P} = \{p_i\}_{i=1}^{n}$ denotes the set of one-dimensional embeddings of the points in $\mathcal{D}$. Note that the bandwidth parameter of $\hat{f}_{1D}$ differs from that used in the geodesic probability mass function, $\hat{f}_G$. Using the same bandwidth would be warranted only if pairwise geodesic distances were approximately preserved after the one-dimensional embedding. Unfortunately this assumption is overly restrictive. Instead we propose to use the following mapping between the bandwidth in the geodesic probability mass function, and that used in one-dimensional density estimator,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}_{[0,1]} \left(|p_i - p_j| / h_1 \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}_{[0,1]} \left(d_G(x_i, x_j) / h \right). \tag{11}$$

The above condition effectively requires that the average number of neighbors within a geodesic distance of h for the points in $\mathcal{D}$, is equal to the average number of points within a Euclidean distance of h_1 for the one-dimensional embeddings $\mathcal{P}$. We are now in a position to express the main result of this section.

Theorem 2. *Let $\mathcal{D} \subset \mathbb{R}^d$ be a set of points, and $G(\mathcal{D})$ the associated neighborhood graph. Denote as $\mathcal{P} = \{p_i\}_{i=1}^{n}$ the set of one-dimensional embeddings of the points in $\mathcal{D}$, and without loss of generality assume that $p_1 \leqslant p_2 \cdots \leqslant p_n$. Assume that $\mathcal{D}$ is (k, ρ)-dense clusterable with respect to the geodesic probability mass function $\hat{f}_G(\cdot; \mathcal{D}, h)$ using the uniform kernel as defined in Eq. (7). Let $\mathcal{C}_1, \ldots, \mathcal{C}_k$ denote the ρ-dense clusters, and for each cluster let $l_m = \min_{p_i \in \mathcal{C}_m} p_i$*

and $u_m = \max_{p_i \in \mathcal{C}_m} p_i$*. Let*

$$\rho_1 = \min_{y \in \cup_{m=1}^{k} [l_m, u_m]} \hat{f}_{1D}(y; \mathcal{P}, h_1), \tag{12}$$

where $\hat{f}_{1D}$ *is the kernel density estimator defined in Eq.* (10)*, and the bandwidth parameter,* h_1*, satisfies Eq.* (11)*. If,*

$$\rho_1 > \min_{y \in \text{conv}(\cup_{m=1}^{k} [l_m, u_m])} \hat{f}_{1D}(y; \mathcal{P}, h_1), \tag{13}$$

where conv *denotes the convex hull, then splitting at the lowest minimum of* $\hat{f}_{1D}(y; \mathcal{P}, h_1)$ *in the interval* $[p_1, p_n]$ *is guaranteed to not split a* ρ*-dense cluster.*

Proof. The proof is straightforward. The definition of ρ_1 in Eq. (12) ensures that $f(y; \mathcal{P}, h_1) \geqslant \rho_1$ for all $y \in [l_m, u_m]$. In other words every interval $[l_m, u_m]$ is a subset of $L(\rho_1; \hat{f}_{1D}(\cdot; \mathcal{P}, h_1))$, the ρ_1-superlevel set $\hat{f}_{1D}$. Moreover,

$$\min_{y \in [p_1, p_n]} \hat{f}_{1D}(y; \mathcal{P}, h_1) \leqslant \min_{y \in [l, u]} \hat{f}_{1D}(y; \mathcal{P}, h_1) < \rho_1.$$

The first inequality holds because $[l, u] \subset [p_1, p_n]$ and the second inequality is due to Eq. (13). Therefore splitting at the minimizer of $\hat{f}_{1D}$ in the interval $[p_1, p_n]$ is guaranteed not to split a ρ-dense cluster.

□

This theorem is analogous to Theorem 1 in the case of dense clusterable sets. Figure 3 provides a visualization of this result. The dataset and neighborhood graph used to obtain the one-dimensional embeddings is the same as in Figure 2. The kernel density estimator, $\hat{f}_{1D}$, in Figure 3 uses the uniform kernel as specified in Eq. (10). In the figure points from the first cluster, $\mathcal{C}_1$, are embedded in the interval $[l_1, u_1]$, while points from the second cluster, $\mathcal{C}_2$, are embedded in $[l_2, u_2]$. The value of ρ_1, defined in Eq. (13), is the minimum of the density in $[l_1, u_1] \cup [l_2, u_2]$. In this example the minimum of $\hat{f}_{1D}$ occurs in $[l_1, u_1]$ and is indicated with a dashed horizontal line. The interval containing the embedded points from all clusters, $[l, u]$, is in this case equal to $[l_1, u_2]$. The minimum value of $\hat{f}_{1D}$ in $[l, u]$ (indicated with a red dashed line in Figure 3) is smaller than ρ_1. Therefore splitting at the global minimum of $\hat{f}_{1D}$ in the

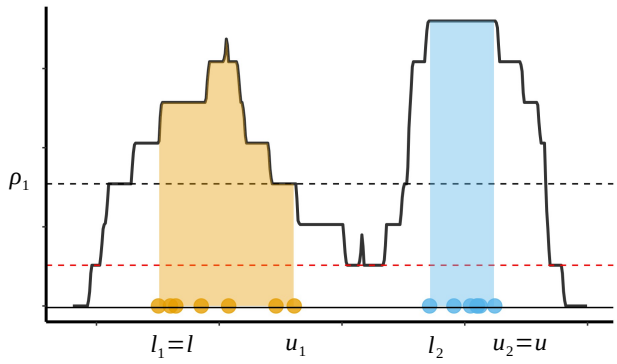

Figure 3: Kernel density estimator (using the uniform kernel) constructed from one-dimensional embeddings. Splitting at the global minimizer of the density in the interval defined by the points with smallest and largest values avoids splitting a dense cluster.

interval containing all the embedded points (which in this case this is equal to $[l, u]$) is guaranteed to not split a dense cluster.

The mapping of the bandwidth parameter in Eq (11) allows the user to verify whether and to what extend the assumptions of Theorem 2 hold. The user can specify the bandwidth parameter for the geodesic probability mass function, $\hat{f}_G$ and then the ρ-dense clusterable sets can be identified directly from the graph $G(\mathcal{D})$. By embedding the points in one-dimension and computing an appropriate value for h_1 through Eq. (11) the user can then verify the extent to which the conditions of Theorem 2, namely Eqs. (12) and (13) hold. In the contrary case, if the user has no prior information about an appropriate value of h for the geodesic probability mass function, they can follow the reverse procedure, which is how our clustering algorithm operates. First, embed the data in one dimension through the dbt-Isomap algorithm and select an appropriate bandwidth for the univariate dataset. Identify the splitting point as the minimizer of $\hat{f}_{1D}$ in the interval between the first and last modes of this function. One can then use Eq. (11) to obtain an appropriate bandwidth for $\hat{f}_G$ and then verify whether the graph partition induced by the one-dimensional splitting procedure preserves

Algorithm 1 Function Cluster ($\mathcal{D}$)

Calculate the undirected graph $W \in \mathbb{R}^{n \times n}$
Calculate the geodesic distances $\mathcal{G}$ from W
Set $\Pi = \{\mathcal{G}\}$
repeat
    Apply dimension reduction for each set $\mathcal{C} \in \Pi$
    Select a set $\mathcal{C} \in \Pi$, using either of the criteria
    Split $\mathcal{C}$ into two sub-sets $\mathcal{C}_1$ and $\mathcal{C}_2$
    Replace $\mathcal{C}$ in Π by $\mathcal{C}_1, \mathcal{C}_2$
until Stopping criterion is satisfied
Return Π, the partition of $\mathcal{D}$ into $|\Pi|$ clusters

dense clusters.

3.3. Summarizing Algorithms

The above theoretical results suggest the construction of a clustering methodology by splitting the data points either based on the maximum distance, *MaxDist*, between consecutive one dimensional embeddings, or based on the global minimum of the corresponding density estimator, *MinLocal*. The algorithm starts with a single cluster $C_0 = \mathcal{D}$ and iteratively splits cluster C_i that corresponds to the maximum *MaxDist* or minimum *MinLocal*, depending on the choice of splitting criterion. The selected cluster is replaced by two subclusters that constist of the points whose embeddings lie at the left and right of the splitpoint. The procedure terminates when the given number of clusters has been retrieved. Ideally when using the density based approach we could introduce an automatic cluster number determination procedure to estimate the actual number of clusters. For instance in [29] it is suggested to terminate the algorithm when there are no local minima of $\hat{f}_{1D}$ in any of the remaining clusters. However the choice of bandwidth parameter greatly affects the outcome of such rules, and suggesting an optimal bandwidth value for correctly estimating the number of clusters is beyond the scope of the current work. Algorithm 1

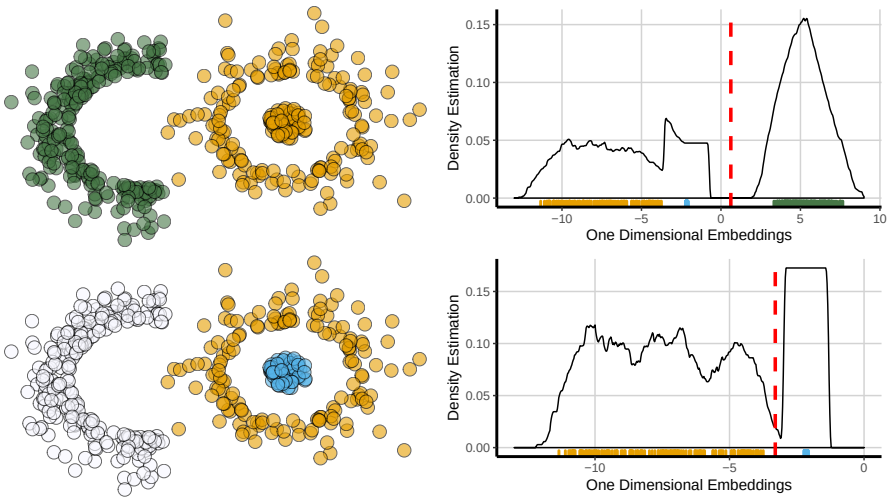

Figure 4: A toy example showing the hierarchical procedure of i-DivClu. First step at the top row of the Figure where we see a scatter diagram and the corresponding kernel density estimation of the one dimensional embedding. At the bottom row the cluster separated at the previous step has been grayed out and the cluster split step is applied to the remaining dataset.

provides an outline of the proposed iterative procedure. Note that the graph construction and the computation of geodesic distances takes place once at the start of the clustering procedure, while the one-dimensional embedding through MDS is executed at each binary split in the divisive process.

A simple illustrative example of the algorithmic procedure is presented in Figure 4, where we employ a two dimensional dataset that contains clusters that are not linearly separable. A scatter diagram is presented in top left while at the top right we see the one dimensional embedding along with the corresponding kernel density estimation and the splitting hyperplane defined by *MinLocal*. Next at the bottom we see the next step of the procedure where the algorithm chooses to split the next cluster (corresponding to the coloured points in the bottom left scatter plot) recalculating an appropriate splitting criterion. In what follows, we will refer to the proposed algorithm using the acronym i-DivClu corresponding to the description "Isometric mapping for Divisive Clustering".

3.4. Computational Complexity

The cost of the nearest neighbor search for k nearest neighbors of n points in r dimensions, is approximately $O(r \log(k) n \log(n))$. This can be significantly improved especially for high dimensional data by employing approximate nearest neighbor searches [31]. Using Dijkstra's algorithm to compute the shortest paths with the implementation of a priority queue with a Fibonacci heap, the time complexity is $O(|\mathcal{E}| + n \log n)$ for each vertex, where $|\mathcal{E}|$ is the number of edges in the neighborhood graph (cardinality of $\mathcal{E}$). Since the number of edges in a k-NN graph is bounded by kn the complexity of evaluating the shortest paths for all vertices is $O(kn^2 + n^2 \log n)$. If k is considered a fixed parameter the complexity is bounded by $O(n^2 \log n)$. Now, given the inexpensive step of calculating the shortest path amongst the centroids connected by a spanning tree, with dbt-Isomap the computational costs may be kept even lower since the shortest paths are computed only within each connected component, allowing us to bound the computational complexity for the evaluation of the shortest paths by $O(m^2 \log m)$ where m is the number of points in the largest connected component. Finally the complexity of MDS, based on the partial eigenvalue decomposition is $O(r_d n^2)$, where r_d is the output dimension. For the proposed clustering algorithms we require only one-dimensional embeddings, thus the complexity of MDS is $O(n^2)$. Several methods for the fast approximation of MDS have been proposed, that significantly reduce its computational cost [32]. Finally note that the computational cost of evaluating the uniform kernel is much smaller than that of other commonly used kernels such as the Gaussian.

4. Experimental Results

In this section we perform a comparative evaluation of the proposed i-DivClu methodology. We consider both the density and the maximum distance variants, which we refer to as i-DivClu-D and i-DivClu-M respectively. We compare against ten well-established and state-of-the-art clustering algorithms. Kernel k-means [21] and (normalized) spectral clustering (SC) [33] are the two most well-

established and widely used algorithms for nonlinear clustering. Both methods still receive considerable attention from the research community. We use the default implementation from the `kernlab` `R` package for both kernel k-means and SC. We also consider a two-step approach in which we first apply Kernel PCA (KPCA) to calculate the projections of the high dimensional feature vectors onto a number of kernel principal components and then perform Minimum Density Divisive Clustering (MDDC) [5, 34]. By pre-processing the data through KPCA a method like MDDC, that uses hyperplanes to recursively bi-partition the data can identify nonlinearly separable clusters, as long as the classes are separable in the feature space. We will refer the aforementioned approach as "Kernel MDDC". We use the implementation of MDDC with default options from the `PPCI` `R`. The use the default implementation from the As baseline comparisons we also consider hierarchical agglomerative clustering using single linkage and average linkage.

In our experiments we compare against three density-based algorithms. DBSCAN [15] (implemented in the `R` package `dbscan`) is considered the baseline density clustering algorithm and has been shown to be particularly effective in low dimensions. We also consider the more recent pdfCluster [16] and densityPeaks [17], implemented in the `R` packages `pdfCluster` and `densityClust` respectively. Unlike DBSCAN, pdfCluster and densityPeaks can detect clusters defined at different density thresholds. From the family of model-based clustering methods we consider the High Dimensional Data Clustering (HDDC) algorithm [35] and implemented in the `R` package `HDClassif`. HDDC handles high dimensional data through a parametrisation of the Gaussian mixture model that combines the idea of dimension reduction and model constraints on the covariance matrices. Finally we consider the recent cut-edge spatial clustering (CutESC) algorithm [18] [1]. CutESC formulates the clustering problem as a graph partitioning problem. It is designed to handle clusters of complicated shapes and different densities, and can accommodate outliers.

[1] For a Python implementation see `https://https://github.com/alperaksac/cutESC`

The implementation of i-DivClu as well as that of all the competing algorithms (except CutESC) is in basic `R` language. This renders computational time comparisons meaningful. For consistency all experiments were performed on a PC with Intel(R) Core(TM) i9-7920X CPU with 32 Gigabytes of RAM and Ubuntu 18.04 Linux operating system.

For i-DivClu we need to define the number of nearest neighbors k used to construct the k-NN graph. All the reported experimental results are obtained using $k = 5$. In Section 4.3 we perform a sensitivity analysis of the performance of i-DivClu with respect to the choice of k. To avoid outliers affecting the number of clusters retrieved by i-DivClu we prohibit binary partitions that result in clusters with less than $\max\{5, n/(4L)\}$ points, where n is the number of samples and L the user-defined number of clusters. For i-DivClu-D we also need to specify the bandwidth parameter, h, used in the univariate kernel density estimation. In all experiments h is equal to half the value suggested by the rule proposed by Sheather and Jones [36]. For algorithms that require the number of clusters to be specified by the user, the actual number of clusters is provided as input. DBSCAN, densityPeaks, pdfCluster, and CutESC estimate the number of clusters during their execution, and their implementations do not allow the user to determine this parameter. For densityPeaks there is no automatic way to define the parameters "rho" and "delta" that affect the number of retrieved clusters. The authors provide a graphical tool with which the user can manually set the respective values through a visual investigation of a scatter plot. However, even through this approach this task is not straightforward for high dimensional datasets. In our experiments we set these as the average values across the parameters calculated for all samples. For pdfCluster we set the `graphtype` parameter to "pairs" when evaluated on datasets with more than 10 dimensions as suggested by the authors. For all other methods we used the default parameters provided by their respective implementations.

To assess clustering quality we use *Normalized Mutual Information (NMI)* [37] and the *Adjusted Rand index (ARI)* [38], Both NMI and ARI are external cluster-validity measures that quantify the degree of agreement between a clus-

ter assignment and the true classes to which the data belong. We selected these evaluation measures because they are appropriate for comparing clustering assignments in which the number of clusters differ. NMI is an information theoretic measure that takes values in $[0, 1]$, with higher values indicating better performance. The Adjusted Rand index is the corrected-for-chance version of the Rand index and takes values in $[-1, 1]$. Again higher values indicate better performance, while a value of zero corresponds to a clustering for which the Rand index is equal to its expected value (under the generalized hypergeometric distribution assumption for randomness).

To enable the reproducibility of our experimental analysis we provide all the datasets and the source code for the empirical evaluation in a GitHub repository[2].

4.1. Artificial Example

We first assess i-DivClu on two-dimensional artificial datasets that contain both linearly and nonlinearly separable clusters of various shapes, sizes, and densities. This allows us to visualize the cluster assignment and thus obtain insights into the behavior of different algorithms. Although it is typical to assess clustering algorithms on a number of artificial datasets, each exhibiting a single characteristic of interest[3] we prefer to consider two datasets each containing numerous challenging features. We created the first artificial dataset, Dataset1, to comply with this requirement[4], while the second dataset, Dataset2, is provided in [18][5].

Figures 5 and 6 provide a visualization of the cluster assignments produced by all the considered algorithms on Dataset1 and Dataset2, respectively. Points allocated to different clusters are indicated by points of different color and shape.

[2] see `https://github.com/usersotiris/nonlinearclustering`

[3] See for instance `http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html`

[4] Available at `https://github.com/usersotiris/nonlinearclustering`

[5] Available at `https://https://github.com/alperaksac/cutESC`.

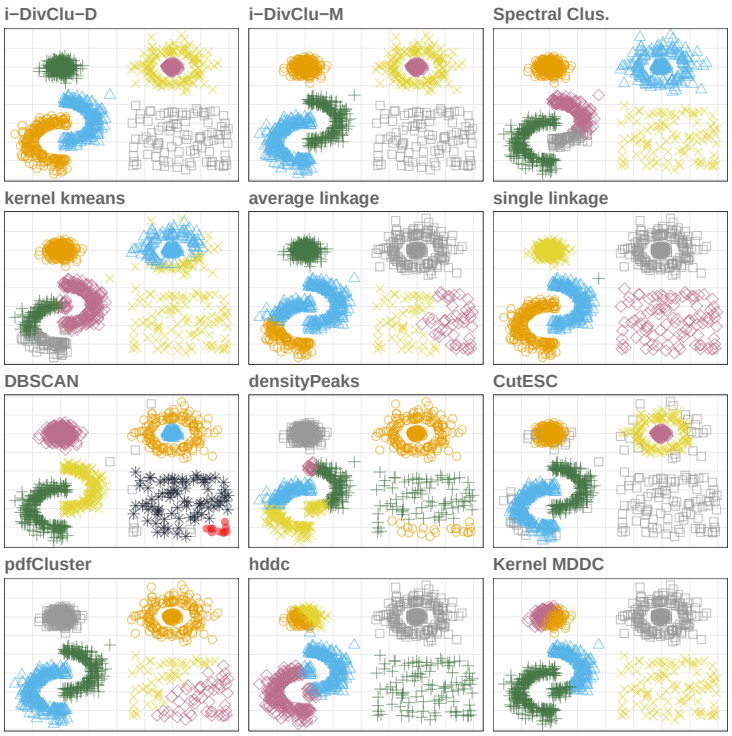

Figure 5: Clustering result of various algorithms for the artificial two dimensional dataset Dataset1.

The output of kernel k-means and SC is not deterministic due to the randomization in the initialization of k-means. In the figures we depict the outcome of an indicative run for each of these methods. SC performs reasonably well but in both datasets fails to identify the dense cluster contained within the circular cluster. Kernel k-means also fails to detect this cluster, and performs poorly on Dataset1. Hierarchical agglomerative clustering using single linkage can detect nonlinear clusters but is also heavily affected by outliers. This is most evident on Dataset2 where a few outliers are assigned to separate clusters. Since for this algorithm the number of clusters is predefined this causes the clustering procedure to terminate prior to detecting all the actual clusters. Single linkage performs much better on Dataset1 but again the dense cluster contained within the circular cluster is not detected. Using average linkage increases the

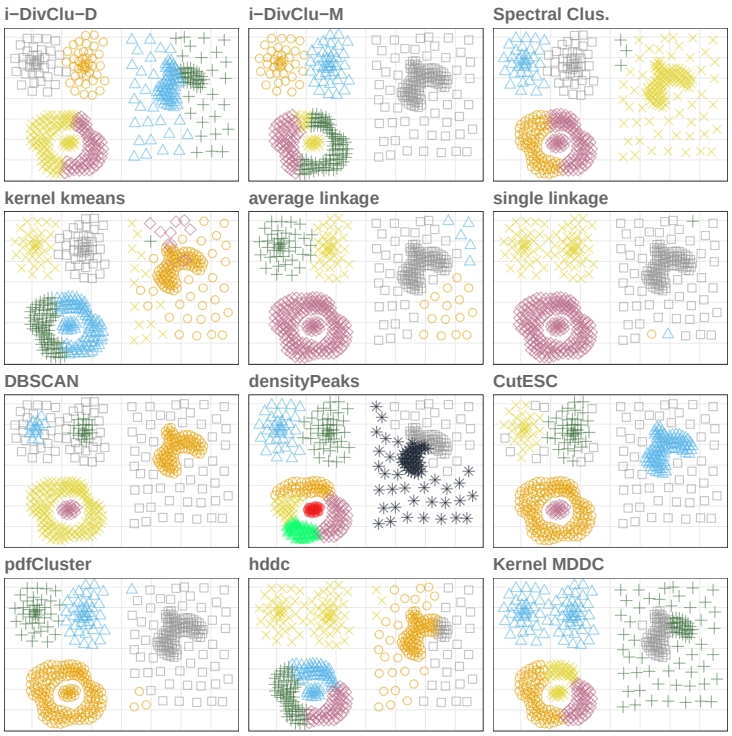

Figure 6: Clustering result of various algorithms for the artificial two dimensional dataset refereed as Dataset2.

robustness to outliers however this approach produces poor clusterings on both artificial datasets. After performing several experiments with DBSCAN to select an appropriate neighborhood size parameter we visualize the most accurate result obtained. As expected an appropriate choice of neighborhood size enables DBSCAN to discover all the clusters. However on both datasets many points are incorrectly characterized as outliers. This highlights the limitations of using a single neighborhood size parameter in the presence of clusters with different densities. pdfCluster fails to detect the cluster contained in the center of the circular cluster on both datasets. On Dataset1 this algorithm also splits the sparse cluster in the bottom right corner into two, while on Dataset2 the points in the sparse region are incorrectly assigned. densityPeaks performs poorly on Dataset1, and slightly better on Dataset2, where it substantially overestimates

the number of clusters. On Dataset1 the performance of HDDC is similar to that of pdfCluster. On Dataset2 HDDC incorrectly partitions the circular cluster and the dense cluster in its center. It also partitions the dense cluster in the right and the surrounding sparse region of points. Kernel MDDC produces very similar clustering to that of HDDC for both datasets. CutESC achieves a nearly perfect clustering on Dataset2 but it considers the sparse cluster at the bottom right of Dataset1 as outliers. The two i-DivClu algorithms achieve an optimal result for Dataset1 while i-DivClu-M appears to perform better than i-DivClu-D on Dataset2.

4.2. Experiments on Real Datasets

In this section we assess the relative performance of the proposed methodology on real-world datasets. We selected a wide range of datasets with different characteristics but specifically focused on high dimensional datasets that are likely to contain data drawn from multiple nonlinear manifolds. Image datasets containing different poses of similar objects, such as those used for facial recognition, are known to have this property [9].

COIL An image dataset provided by Columbia University Computer Vision Laboratory. In the original dataset there are images of 20 objects in 72 different angles. The resulting dimensionality is 16384[6].

USPS A portion of the original USPS dataset containing images of handwritten digits. Each image is of size 16×16. Here we use a subset containing the digits 2, 4, 6 and 8 forming a 4575×256 data matrix[7].

Olivetti This dataset provided by AT&T Laboratories Cambridge contains a set of face images. There are ten different 92×112 images of each of 40 distinct subjects forming the 400×10304 data matrix[8].

[6] `www.cs.columbia.edu/CAVE/software/softlib/coil-20.php`
[7] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps`
[8] `http://www.uk.research.att.com/`

ISOLET A widely used UCI dataset that was generated by subjects that pronounced the name of each letter of the alphabet twice. It consists of 1560 observations of dimensionality 617[9].

SEEDS A dataset from the UCI repository based on images of kernels belonging to three different varieties of wheat. There are 70 elements from each category characterized by 7 variables[10].

MOVE A video capturing different movements is analysed to generate 90 features. The dataset contains 15 classes of 24 instances each, where each class refers to a type of hand movement[11].

CMU 640 black and white face images of people taken with varying pose (straight, left, right, up), expression (neutral, happy, sad, angry), eyes (wearing sunglasses or not), and size. There are 32 images for each person capturing every combination of features while the image resolution used here is 128×120[12].

Fashion MNIST A random sample of the Fashion-MNIST dataset consisting of a 70000 examples in total. Each example is a 28×28 grayscale image, associated with a label from 10 classes. The resulting data matrix is of size 3000×784 representing all available classes [13].

OML DNA microarray expression profiles for 383 samples with 54675 genes characterized by 9 classes, including data for medical diagnosis and treatment by the RSCTC'2010 Discovery Challenge [39]. Data were gathered by the OpenML open science platform.

PANCAC Microarray expression profiles for 801 samples and 20531 genes. Samples are categorized on five tumor types profiled by the TCGA (Cancer

[9] `https://archive.ics.uci.edu/ml/datasets/isolet`
[10] `https://archive.ics.uci.edu/ml/datasets/seeds`
[11] `https://archive.ics.uci.edu/ml/datasets/Libras+Movement`
[12] `http://archive.ics.uci.edu/ml/datasets/cmu+face+images`
[13] `https://github.com/zalandoresearch/fashion-mnists`

Genome Atlas), a landmark cancer genomics program [40].

MNIST The MNIST database of handwritten digits which have been size-normalized and centered in a fixed-size image. The sample used here is of size 3000×784 representing all available classes [14].

We take representative samples of the Fashion-MNIST and MNIST datasets to ensure that the majority of the considered algorithms can process these datasets with reasonable computational requirements.

Table 1 reports the performance of each algorithm with respect to NMI, ARI and computational time in seconds. For non-deterministic algorithms the results are averages over 20 replications. In the table we only present the performance of algorithms that could handle the size and dimensionality of all the real datasets with reasonable memory and computational time requirements. In particular, CutESC is only applicable when the number of samples exceeds the number of dimensions. Even then CutESC and pdfCluster required more than 60 GB of RAM for most of the datasets considered. These algorithms are therefore excluded from the analysis of real-word datasets. For DBSCAN we tried a range of values for the neighborhood size parameter for each dataset. We found that the appropriate value for this parameter differs considerably across datasets, and even for the best parameter choice DBSCAN was not competitive. Given this and due to space considerations we do not report the performance of this algorithm in Table 1. To economize on space we only report the performance of the single linkage algorithm as it performed significantly better on average that the average linkage strategy for agglomerative clustering.

The results reported in Table 1 indicate that overall i-DivClu-D outperforms the competing methods. The table shows that on real-world datasets i-DivClu-M outperforms i-DivClu-D only once. We therefore always recommend using the density variant of i-DivClu. In terms of NMI i-DivClu-D is the best algorithm in seven out of the eleven datasets. In terms of ARI it outperforms all other meth-

[14] `http://yann.lecun.com/exdb/mnist/`

Table 1: Clustering accuracy with respect to NMI, ARI and the corresponding computational time in seconds for the real datasets.

	Algorithms							
	i-Div Clu-D	i-Div Clu-M	Spectral Clust.	Kernel k-means	Single Link.	Kernel. MDDC	density Peaks	HDDC
Datasets	NMI/ARI							
CMU	**0.79**/0.51	0.76/0.43	0.8/**0.52**	0.65/0.37	0.71/0.36	0.23/0.05	0.78/0.47	0.75/0.5
COIL	**0.89**/**0.73**	0.82/0.39	0.82/0.55	0.68/0.43	0.64/0.32	0.10/0.02	0.77/0.54	0.78/0.57
FASH.	**0.57**/**0.39**	0.54/0.21	0.54/0.37	0.42/0.28	0.4/0.24	0.01/0.00	0.51/0.28	0.54/0.36
ISOL.	**0.77**/0.49	0.6/0.11	0.74/0.51	0.62/0.32	0.73/0.42	0.36/0.14	0.71/0.42	0.76/**0.52**
MNIST	0.54/0.32	0.42/0.16	**0.63**/0.45	0.39/0.29	0.4/0.21	0.00/0.00	0.54/0.28	0.61/**0.46**
MOVE	0.6/**0.32**	0.6/0.25	**0.62**/0.33	0.53/0.26	0.54/0.23	0.57/0.27	0.61/0.29	0.62/**0.34**
OLIV.	**0.88**/**0.66**	0.86/0.6	0.84/0.53	0.65/0.26	0.84/0.52	0.45/0.03	0.86/0.56	0.86/0.57
OML.	**0.52**/**0.31**	0.45/0.23	0.45/0.24	0.37/0.19	0.38/0.29	0.06/0.01	0.5/0.08	0.48/0.27
PANC.	0.97/0.98	**0.98**/**0.99**	0.9/0.88	0.82/0.75	0.86/0.85	0.01/0.01	0.67/0.54	0.92/0.89
SEEDS	0.63/0.57	0.44/0.27	0.36/0.32	**0.65**/**0.68**	0.62/0.55	0.64/0.58	0.53/0.5	0.3/0.29
USPS	**0.86**/**0.9**	0.83/0.86	0.76/0.65	0.71/0.68	0.44/0.29	0.49/0.44	0/0	0.6/0.5
Aver.	**0.73**/**0.56**	0.66/0.41	0.68/0.49	0.59/0.41	0.59/0.39	0.26/0.14	0.59/0.36	0.66/0.48
	Time in seconds							
CMU	8.78	8.11	3.32	23.26	22.37	9.42	22.33	68.63
COIL	30.14	22.8	25.32	113.04	154.81	70.53	153.78	302.02
FASH.	35.34	25.56	179.53	19.81	12.89	358.38	12.98	14.3
ISOL.	6.6	9.45	26.33	4.05	1.25	126.28	1.34	26.84
MNIST	36.32	69.98	147.64	29.68	12.8	340.15	12.84	17.06
MOVE	0.36	0.15	0.58	0.17	0.01	5.39	0.02	0.5
OLIV.	2.12	1.87	1.24	12.68	5.14	9.21	5.16	28.18
OML.	7.85	7.72	3.03	44.88	28.05	3.40	28.07	56.92
PANC.	8.85	8.56	6.13	58.89	54.09	6.76	54.12	29.08
SEEDS	0.23	0.04	0.22	0.04	0	1.65	0.01	0.22
USPS	74.68	75.76	659.49	30.57	5.15	552.96	5.22	5.94
Aver.	19.21	**14.44**	82.36	30.31	26.03	134.92	26.06	49.02

ods in six datasets. Moreover i-DivClu-D is among the best performing methods in all cases. This is also reflected in the fact that it achieves the highest average NMI and ARI performance. The second best average performance with respect to both NMI and ARI is achieved by SC, followed by HDDC. HDDC achieves the highest ARI score in four datasets while the Kernel k-means achieves the

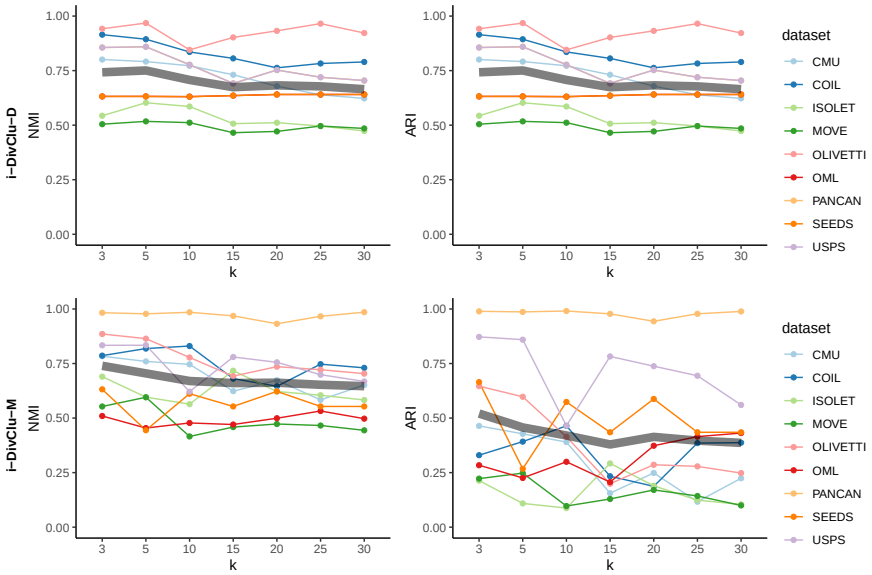

Figure 7: Clustering performance in terms of NMI and ARI for both proposed methods with respect to the parameter k used to build the k-NN graph. Average values over k are illustrated through the gray shaded line.

best performance on the SEEDS dataset. Kernel MDDC perform poorly in most cases probably due to inappropriate representations by the default KPCA parameters while also being the most computationally intensive method.

The computational cost of both i-DivClu variants is on average lower, but in the same order of magnitude, as that of competing algorithms. Computational time clearly depends on the characteristics of each dataset however i-DivClu offers a compromise between algorithms like SC whose running time is mainly determined by the number of samples, and those like HDDC and densityPeaks whose running time is mainly affected by the dimensionality of the dataset.

4.3. Parameter Analysis

We conclude our empirical evaluation by studying the sensitivity of i-DivClu to the choice k in the k-NN graph employed by Isomap. For many manifold learning methods this parameter is critical as it can greatly affect the structure of the similarity graph. Using a large value for k can lead to connections being

established between previously unconnected components of the graph, while a low value may cause dense areas of points to be split to many small disconnected components [14]. Our hypothesis is that the dbt-Isomap approach of ensuring the k-NN graph is connected, jointly with the proposed splitting criteria for binary partitioning, render i-DivClu relatively robust to variations of this parameter. Figure 7 illustrates the performance of i-DivClu-D and i-DivClu-M on the real-word datasets for different values of k. Overall a better performance is attained using relatively small values for k (specifically in the range between 3 and 5). In most datasets increasing k causes a gradual performance degradation, but this trend is not monotonic.

5. Concluding Remarks

In this paper we extended earlier clustering approaches to deal with clusters that are not linearly separable. The central idea is to discuss conditions under which that a suitably defined one-dimensional representation is sufficient to partition the data without splitting any of the clusters. Applying this procedure recursively provides a theoretically justified and efficient nonlinear clustering methodology. The theoretical results show that the two clustering techniques we propose can deal with both linear and nonlinear cases under assumptions on the cluster structure. Our experimental results indicate that the proposed methods perform well on challenging simulated and real data sets compared to well-established and state-of-the-art clustering algorithms.

In future work, we intend to explore the use of alternative dimension reduction methods, including the automatic determination of their parameters. We are currently also working on applications to very large datasets and streaming data, which require development of iterative and approximate variants.

Acknowledgment

This project has received funding from the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology

(GSRT), under grant agreement No 1901.

[1] M. A. Cox, T. F. Cox, Multidimensional scaling, in: Handbook of Data Visualization, Springer Handbooks Comp.Statistics, Springer Berlin Heidelberg, 2008, pp. 315–347.

[2] H.-P. Kriegel, P. Kröger, A. Zimek, Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering, ACM Transactions on Knowledge Discovery from Data 3 (1) (2009) 1–58.

[3] C. Ding, T. Li, Adaptive dimension reduction using discriminant analysis and k-means clustering, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 521–528.

[4] D. P. Hofmeyr, N. G. Pavlidis, Maximum clusterability divisive clustering, in: IEEE Symposium Series on Computational Intelligence, 2015, pp. 780–786.

[5] N. G. Pavlidis, D. P. Hofmeyr, S. K. Tasoulis, Minimum density hyperplanes, Journal of Machine Learning Research 17 (156) (2016) 1–33.

[6] D. Niu, J. G. Dy, M. I. Jordan, Dimensionality reduction for spectral clustering, in: International Conference on Artificial Intelligence and Statistics, 2011, pp. 552–560.

[7] D. P. Hofmeyr, Clustering by minimum cut hyperplanes, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (8) (2017) 1547–1560.

[8] D. P. Hofmeyr, N. G. Pavlidis, I. A. Eckley, Minimum spectral connectivity projection pursuit, Statistics and Computing 29 (2) (2019) 391–414.

[9] R. Vidal, Subspace clustering, IEEE Signal Processing Magazine, vol. 28, issue 2, pp. 52-68 28 (2011) 52–68.

[10] J. B. Tenenbaum, V. Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.

[11] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319.

[12] S. T. Roweis, L. K. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science 290 (5500) (2000) 2323–2326.

[13] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15 (6) (2003) 1373–1396.

[14] C. Orsenigo, C. Vercellis, An effective double-bounded tree-connected isomap algorithm for microarray data classification, Pattern Recognition Letters 33 (1) (2012) 9 – 16.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise., in: Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining, Vol. 96, 1996, pp. 226–231.

[16] G. Menardi, A. Azzalini, An advancement in clustering via nonparametric density estimation, Statistics and Computing 24 (5) (2014) 753–767.

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

[18] A. Aksac, T. Özyer, R. Alhajj, CutESC: Cutting edge spatial clustering technique based on proximity graphs, Pattern Recognition 96 (2019) 106948.

[19] H. Li, X. Liu, T. Li, R. Gan, A novel density-based clustering algorithm using nearest neighbor graph, Pattern Recognition 102 (2020) 107206.

[20] Y. Qin, Z. L. Yu, C.-D. Wang, Z. Gu, Y. Li, A novel clustering method based on hybrid k-nearest-neighbor graph, Pattern Recognition 74 (2018) 1 – 14.

[21] I. 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.

[22] X. Chen, Y. Yang, Diffusion k-means clustering on manifolds: provable exact recovery via semidefinite relaxations, ArXiv abs/1903.04416.

[23] A. Ng, M. Jordan, Y. Weiss, On Spectral Clustering: Analysis and an algorithm, in: T. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, MIT Press, 2001, pp. 849–856.

[24] U. von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (4) (2007) 395–416.

[25] X. Chen, W. Hong, F. Nie, J. Z. Huang, L. Shen, Enhanced balanced min cut, International Journal of Computer Visiondoi:10.1007/s11263-020-01320-3.
URL https://doi.org/10.1007/s11263-020-01320-3

[26] Z. Kang, G. Shi, S. Huang, W. Chen, X. Pu, J. T. Zhou, Z. Xu, Multi-graph fusion for multi-view spectral clustering, Knowledge-Based Systems 189 (2020) 105102. doi:https://doi.org/10.1016/j.knosys.2019.105102.
URL http://www.sciencedirect.com/science/article/pii/S0950705119304770

[27] H. Yu, X. Zhang, Y. Yang, X. Zhao, L. Cai, An extended isomap by enhancing similarity for clustering, in: H. Jiang, W. Ding, M. Ali, X. Wu (Eds.), Advanced Research in Applied Artificial Intelligence, Vol. 7345 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 808–815.

[28] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, J. C. Langford, The isomap algorithm and topological stability, Science 295 (5552) (2002) 7a.

[29] S. Tasoulis, D. Tasoulis, V. Plagianakos, Enhancing principal direction divisive clustering, Pattern Recognition 43 (2010) 3391–3411.

[30] J. A. Hartigan, Clustering Algorithms, Wiley Series in Probability and Mathematical Statistics, Wiley, New York, 1975.

[31] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Communications of the ACM 51 (1) (2008) 117–122.

[32] T. Yang, J. Liu, L. Mcmillan, W. Wang, A fast approximation to multidimensional scaling, in: Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision (CIMCV, 2006.

[33] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: T. G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14, MIT Press, 2002, pp. 849–856.

[34] K. R. Yates, N. G. Pavlidis, Minimum density hyperplanes in the feature space, in: IEEE International Conference on Big Data (Big Data), 2016, pp. 3613–3618.

[35] C. Bouveyron, G. Girard, C. Schmid, High dimensional data clustering, Computational Statistics & Data Analysis 52 (1) (2007) 502–519.

[36] S. J. Sheather, M. C. Jones, A reliable data-based bandwidth selection method for kernel density estimation, Journal of the Royal Statistical Society. Series B (Methodological) 53 (3) (1991) 683–690.

[37] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, Journal of Machine Learning Research 3 (2002) 583–617.

[38] L. Hubert, P. Arabie, Comparing partitions, Journal of Classification 2 (1) (1985) 193–218.

[39] M. Wojnarski, A. Janusz, H. S. Nguyen, J. Bazan, C. Luo, Z. Chen, F. Hu, G. Wang, L. Guan, H. Luo, et al., Rsctc?2010 discovery challenge: Mining dna microarray data for medical diagnosis and treatment, in: International Conference on Rough Sets and Current Trends in Computing, Springer, 2010, pp. 4–19.

[40] J. N. Weinstein, E. A. Collisson, G. B. Mills, K. R. M. Shaw, B. A. Ozenberger, K. Ellrott, I. Shmulevich, C. Sander, J. M. Stuart, C. G. A. R. Network, et al., The cancer genome atlas pan-cancer analysis project, Nature genetics 45 (10) (2013) 1113.