Pattern Recognition 43 (2010) 3391–3411

Contents lists available at ScienceDirect

Pattern Recognition

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

Enhancing principal direction divisive clustering

S.K. Tasoulis [a,*], D.K. Tasoulis [b], V.P. Plagianakos [a]

[a] Department of Computer Science and Biomedical Informatics, University of Central Greece, 2–4 Papassiopoulou Str., Lamia 35100, Greece
[b] Mathematics Department, Imperial College London, 180 Queen's Gate, SW7 2AZ, UK

ARTICLE INFO

Article history:
Received 4 November 2009
Received in revised form
16 May 2010
Accepted 19 May 2010

Keywords:
Clustering
Principal component analysis
Kernel density estimation

ABSTRACT

While data clustering has a long history and a large amount of research has been devoted to the development of numerous clustering techniques, significant challenges still remain. One of the most important of them is associated with high data dimensionality. A particular class of clustering algorithms has been very successful in dealing with such datasets, utilising information driven by the principal component analysis. In this work, we try to deepen our understanding on what can be achieved by this kind of approaches. We attempt to theoretically discover the relationship between true clusters in the data and the distribution of their projection onto the principal components. Based on such findings, we propose appropriate criteria for the various steps involved in hierarchical divisive clustering and develop compilations of them into new algorithms. The proposed algorithms require minimal user-defined parameters and have the desirable feature of being able to provide approximations for the number of clusters present in the data. The experimental results indicate that the proposed techniques are effective in simulated as well as real data scenarios.

1. Introduction

Data clustering is a central component of the knowledge discovery process. Formally it can be defined as "*the process of partitioning a set of data vectors into disjoint groups* (*clusters*), *so that objects of the same cluster are more similar to each other than objects in different clusters*". The modern roots of data clustering date back to 1939 [1], but there are references to it from antiquity.

There exist many categorisations of clustering algorithms, but broadly we could divide them into three main categories: hierarchical, partitioning, and distance-based. Hierarchical clustering algorithms construct hierarchies of clusters in a top-down (agglomerative) or bottom-up (divisive) fashion. The former, start from n clusters, where n stands for the number of data points, each containing a single data point and iteratively merge the clusters satisfying certain measures of closeness. Divisive algorithms follow a reverse approach; starting with a single cluster containing all the data points and iteratively split existing clusters to subsets. Hierarchical clustering algorithms have been shown to result in high quality partitions especially for applications involving clustering text collections. Nonetheless, their high computational requirements, usually prevents their usage in real-life applications, where the number of samples and their dimensionality is expected to be high (the computational cost is quadratic to the number of samples).

Partitioning clustering algorithms [2], start from an initial clustering (that may be formed at random) and subsequently create flat partitionings by iteratively adjusting the clusters based on the distance of the data points from a representative member of each cluster. The most popular of partitioning clustering algorithms is k-means [3], which starting from k centres iteratively assigns each data point to the cluster whose centroid minimises the Euclidean distance from the data point. Spherical k-means [4] is a recently proposed modification of the algorithm that reduces k-means to the partitioning of the unit hypersphere by normalising the data points. Algorithms that belong to the same class as the k-means, can give adequate clustering results at low cost, since their running time is proportional to $k \cdot n$. However, their results depend heavily on their initialisation. Another similar approach is Gaussian mixture models (GMM) [5], where k multivariate normal density components are combined, by assuming that each component represents a cluster. Like k-means, an iterative algorithm is used (typically expectation maximization) to fit the parameters of each density to the data. Then the posterior probabilities for each data point to each component of the model is indicative of the probability of the point belonging to each cluster. GMM may be more appropriate than the k-means clustering algorithm, when clusters have different sizes and there exists correlated variables, but more control parameters need to be estimated. This makes their application in high dimensions almost prohibitive.

Finally, distance-based clustering algorithms create flat partitioning by considering neighbours of data points. DBSCAN [6] is a distance-based clustering algorithm that has been proved quite effective for spatial databases. Clusters are considered as high

* Corresponding author. Tel.: +30 22310 66717; fax: +30 22310 66907.
E-mail addresses: stas@ucg.gr (S.K. Tasoulis), d.tasoulis@imperial.ac.uk (D.K. Tasoulis), vpp@ucg.gr (V.P. Plagianakos).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.05.025

density neighbourhoods of data points. Although the density parameter is critical for DBSCAN's success, recently proposed heuristics are able to give high quality results.

While clustering has a long history and a large number of clustering techniques have been developed in statistics, pattern recognition, data mining, and other fields, significant challenges still remain. One of the most important challenges encountered in clustering is associated with high data dimensionality ("curse of dimensionality") [7]. In general terms, these problems result from the fact that a fixed number of data points become increasingly "sparse" as the dimensionality increases [8]. For clustering purposes, the most relevant aspect of the curse of dimensionality concerns the effect that it has on distance or similarity. In [9], for certain data distributions, it is shown that the relative difference of the distances of the closest and farthest data points of an independently selected point goes to 0 as the dimensionality increases. Thus, it is often said that "*in high dimensional spaces, distances between points become relatively uniform*" [8].

However, things are sometimes not as bad as they might seem, since it is often possible to reduce the dimensionality of the data without significant loss of information. This can be accomplished by the feature selection procedure, i.e. discarding features showing little variation or having high correlation to other features. Feature selection is a complicated subject in its own and is out of the scope of the present study.

Another clustering approach is to project points to a suitable low dimensional space. Typically this type of dimensionality reduction is accomplished by applying techniques from linear algebra or statistics such as principal component analysis (PCA) [10] or singular value decomposition (SVD) [11]. This way, a powerful class of clustering algorithms has emerged that provides the opportunity for the deployment of these computational technologies from numerical linear algebra, an area that has seen enormous expansion in recent decades. In this class of algorithms, the principal direction divisive partitioning (PDDP) algorithm is of particular value [12]. Compared to other similar techniques (like latent semantic indexing [13] and linear least square fit [14]), PDDP has the advantage of very low computational complexity.

PDDP is a "divisive" hierarchical clustering algorithm. As already described, this kind of techniques produce a nested sequence of partitions, with a single, all-inclusive cluster at the top. Starting from this all-inclusive cluster the nested sequence of partitions is constructed by iteratively splitting clusters, until a termination criterion is satisfied. Any divisive clustering algorithm can be characterised by the way it chooses to provide answers to the following three questions:

Q_1: Which cluster to split further?
Q_2: How to split the selected cluster?
Q_3: When should the iteration terminate?

The PDDP based algorithms in particular, use information from the PCA of the corresponding data matrix to provide answers to the questions above, in a computationally efficient manner. This is achieved by incorporating information from only the first singular vector and not a full rank decomposition of the data matrix. Note also that certain answers to one of these questions may obsolete one of the others. However, this is not always the case.

In this paper, we separately investigate each of the above questions involved in a divisive hierarchical clustering procedure. First in Section 2, we give the background and the related work of similar approaches. Next, Section 3 is devoted to the presentation of the proposed approach. We analyse both theoretically and intuitively what can be achieved when we are faced with the problem of splitting a cluster based on a particular projection. Subsequently, based on that analysis we propose splitting, selection and stopping criteria that have the potential to provide promising partitioning results. All these techniques are combined into algorithms in Section 4. Section 5 is devoted to the investigation of the efficiency of the proposed techniques in various simulated settings. Next Sections 6 and 7 discuss the selection of appropriate values for the algorithm's parameters. In Section 8 we present an experimental analysis of the running time for different algorithms. Subsequently, Sections 9–11 illustrate the applicability and examine the performance of the proposed techniques in publicly available datasets, as well as in real data scenarios. The paper ends with concluding remarks and some directions of interesting future research.

2. Background

To formally describe the manner in which principal direction based divisive clustering algorithms operate, let us assume the data is represented by an $n \times a$ matrix D, whose each row represents a data sample d_i, for $i=1,\ldots,n$. Also define the vector b and matrix Σ to represent the mean vector and the covariance of the data respectively:

$$b = \frac{1}{n}\sum_{i=1}^{n} d_i, \quad \Sigma = \frac{1}{n}(D - b \cdot e)^{\top} \cdot (D - b \cdot e),$$

where e is a column vector of ones. The covariance matrix Σ is symmetric and positive semi-definite, so all its eigenvalues are real and non-negative. The eigenvectors u_j, $j=1,\ldots,k$ corresponding to the k largest eigenvalues are called the principal components or principal directions. The PDDP algorithm and all similar techniques use the projections p_i:

$$p_i = u_1(d_i - b), \quad i = 1,\ldots,n,$$

onto the first principal component u_1 to initially separate the entire data set into two partitions P_1 and P_2. To generate this partitioning, the original PDDP algorithm uses the sign of the projection of each data point (division point 0). This reveals the text mining origins of the algorithm, since based on a term-document data specification [12], data points (documents) with positive projections should be more similar to each other than data points (documents) with negative ones. However, very often, the sign of the projection can lead to undesirable cluster splits [15–18]. More formally we can define this splitting criterion as follows:

- (Splitting criterion SPC_1): $\forall d_i \in \mathcal{D}$, if $p_i \geq 0$, then the i-th data point belongs to the first partition $P_1 = P_1 \cup d_i$; otherwise, it belongs to the second partition $P_2 = P_2 \cup d_i$.

In [19], the utilisation of the second, the third, etc. component (instead of the principal one) was proposed. A suitable projection was calculated based on a measure of coherence (scatter value, defined below) of the generated clusters. However, this criterion greatly increases the computational complexity of the algorithm. In the same theme, in [16] it is suggested to extend this in a $2l$-way partitioning strategy, by splitting simultaneously in $2l$ clusters using the sign of the projections of the $l \geq 1$ singular vectors. Although, both these approaches have been shown to lead to improved cluster quality, the problem is not solved. Using alternative principal components by no way guarantees improved partitioning, as the problem of where to split irrespective of which projection is used, still exists.

Another approach to determine the splitting point is to use a k-means steering procedure [17,18,20], with $k=2$. In [20] it is suggested to extensively search every splitting point and select

the one with the best k-means objective function (for $k=2$). The drawback of this approach lies in the fact that even if that best such splitting point is found, it does not guarantee a good splitting decision, as the k-means objective function is flawed in many cases (e.g. unbalanced clusters, more than two true clusters in the data, etc.) [10].

To answer the cluster selection question Q_2, the PDDP algorithm as well as all its variations [16–20], select the cluster with maximum scatter value SV, defined as:

$$SV = \|D - be\|_F, \tag{1}$$

where the vector b represents the mean vector of D and $\|\cdot\|_F$ is the Frobenious norm. This quantity can be a measure of coherence and it is the same as the one used by the k-means steered PDDP algorithm [20].

Finally, most of the PDDP variants terminate the clustering procedure when a user defined number of clusters has been retrieved. Little effort has been given to develop a method for the automatic cluster determination, which fits with the workings of the PDDP algorithm. This concept is extensively discussed in Section 3.3.

3. The proposed framework

In this work, in abstract terms, we aim to understand what can be achieved by partitioning the data based on their projection on a particular direction. To this end, we attempt to theoretically discover the relationship between true clusters in the data and the distribution of their projection onto the principal components.

Obviously, this requires assumptions of what true clusters actually mean, which are called "inductive bias" [22]. Here, out of the large variety of such assumptions, we focus on two particular ones: the "compact" and the "peak" based clusters, respectively. Based on these assumptions, we propose appropriate criteria for the various steps involved in hierarchical divisive clustering. In the sections that follow we are going to examine each of the three basic questions involved in the hierarchical divisive clustering procedure individually.

3.1. How to split the selected cluster?

This question is the most central one in a divisive clustering procedure. The reason for this lies in the fact that if a bad decision is made in the first steps on the procedure, little can be done in the subsequent steps to remedy the clustering result. This is the reason why a substantial amount of research has been devoted to find an optimal way to answer this question.

To illustrate the drawbacks of the approaches already in the literature, we can visually inspect the result of each of them in a simple example. In Fig. 1, we illustrate the clustering result of partitioning in two clusters a dataset composed of a mixture of 30 Gaussian distributions. Fig. 1 (top left) corresponds to the first step of an algorithm that splits based on the sign of the projections onto the first principal component (criterion SPC_1). Fig. 1 (top right) shows the equivalent result when the partitioning is performed based on the sign of the projections onto the second principal component, as proposed in [19]. Finally, Fig. 1 (bottom) depicts the result of the partitioning using k-means steering, where the best splitting point is calculated exhaustively as proposed in [20]. As shown, splitting utilising the sign results in sub-optimal decisions, since a large number of compact cluster are split in half in both cases. Also using k-means steering exhibits a marginal improvement.

3.1.1. Contiguous clusters

Recently in [15], a new splitting criterion was proposed based on density or histogram arguments. In that study, it was proposed to compute the most sparse region of the projected data, by first sorting and then finding the largest distance between two consecutive projections. It was shown through experimental analysis that this approach is capable to produce high quality partitions. In the present paper, we study this approach in more detail by theoretically showing when such a decision can be optimal. We also explore its shortcomings and propose an alternative criterion that avoids them.

Trying to investigate the situations in which a criterion can be optimal, we first need to formally define a cluster. Different cluster definitions exist in the literature [8]. The one employed

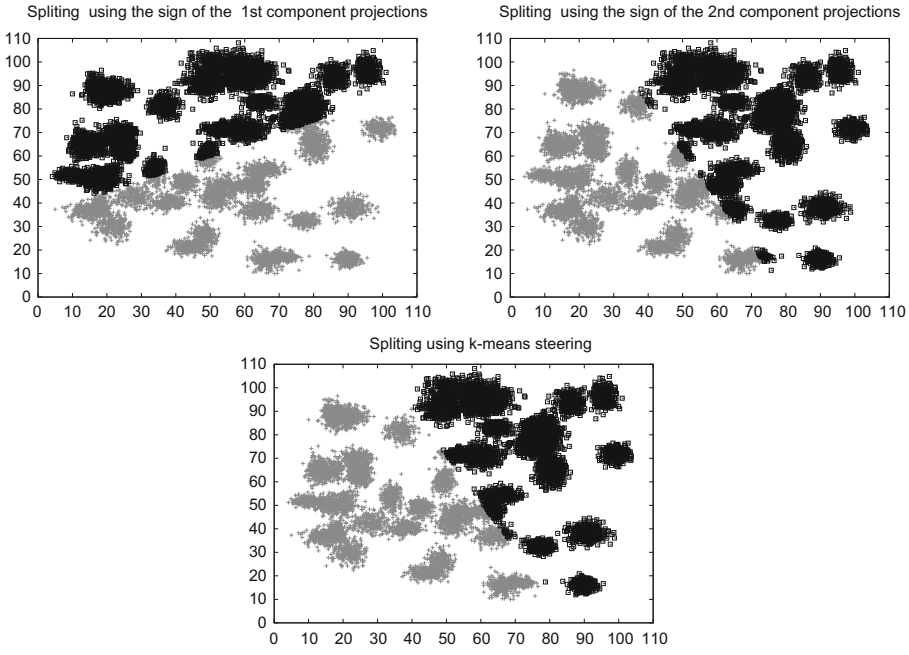

Fig. 1. The partitioning result of different splitting criteria.

here is used for *contiguous* clusters, where any point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster. Formally we define first the coherence of a set as follows:

Definition 1 (*Set coherence*). Let $D \in R^{n\times a}$ be a data matrix corresponding to a set of points $\mathcal{D}$. Then the coherence of the $\mathcal{D}$ is defined as $\mathrm{COH}(\mathcal{D}) = max\{\|d_i - d_{i*}\| : \forall i = 1,\ldots,n\}$, where d_i are the row vectors of D and $i^* = \arg\min_j\{\|d_i - d_j\| : \forall j = 1,\ldots,n$, and $j \neq i\}$.

Now, we can formally define a cluster as a contiguous set:

Definition 2 (*Contiguous set*). Let $\mathcal{D}$ set of points d_i, for $i=1,\ldots,n$, a subset $\mathcal{C} \subset \mathcal{D}$ is a Contiguous set, when $\mathrm{COH}(\mathcal{C}) < \mathrm{COH}(\mathcal{C} \cup \{d_i\})$, $\forall d_i \in \mathcal{D}/\mathcal{C}$.

In the above definition, COH of a set $\mathcal{C}$ is the coherence of the set and $\mathcal{D}/\mathcal{C} = \{d \in \mathcal{D} : d \notin \mathcal{C}\}$. Using such definitions we can now define the clustering problem.

Definition 3 (*Contiguous clusterable set*). A set $\mathcal{D}$ of points is k *contiguous clusterable* if there exists a partition Π of $\mathcal{D}$ into k subsets $\mathcal{C}_1,\ldots,\mathcal{C}_k$, such that any $\mathcal{C}_i$ is a contiguous set.

Now, we can prove the following Lemma, stating that if in a data projection the distance between two consecutive projections is at least M, then the points to the left of the interspace cannot lie in the same cluster with the points in the right, where M is the maximum of the cluster coherencies. As such, splitting on any point in that space is optimal in terms of the partition Π. In Fig. 2 such an example is illustrated. The distance L between the projections p_{t*} and p_{l*} is larger than both the cluster coherencies, so splitting the data on either p_{t*} or p_{l*} guarantees that each partition will contain points from different true clusters.

Lemma 1. *Let a k-contiguous-clusterable set $\mathcal{D}$ of points d_i, for $i=1,\ldots,n$ and Π its partition into k Contiguous subsets $\mathcal{C}_1,\ldots,\mathcal{C}_k$. Let $u \in \mathbb{R}^a$, with $\|u\| = 1$. Let $\mathcal{P}$ be the set of projections p_i of the vectors d_i on u. Also, let $M = \max\{\mathrm{COH}(\mathcal{C}) : \mathcal{C} \in \Pi\}$. If there is a $p_{l*} \in \mathcal{P}$ such that $p_{t*} - p_{l*} > M$, where $p_{t*} = \min\{p : p > p_{l*}, p \in \mathcal{P}\}$, then all d_t, for which $p_t > p_{l*}$, and all d_l, for which $p_l \leq p_{l*}$, belong to different sets of Π.*

Proof. First note that for any d_l and d_t, with $l \neq t$, it holds that

$$\|d_l - d_t\| \geq |p_l - p_t|. \tag{2}$$

Since $p_{t*} - p_{l*} \geq M$, for any d_l with $p_l \leq p_{l*}$ and for any d_t with $p_t > p_{l*}$ it holds that

$$p_t - p_l \geq p_{t*} - p_{l*} > M.$$

Thus from Eq. (2) we have:

$$\|d_l - d_t\| \geq |p_l - p_t| > M. \tag{3}$$

Since $\mathcal{D}$ is k-contiguous-clusterable, $\mathrm{COH}(\mathcal{C}) \leq M$, for all $\mathcal{C} \in \Pi$. If there was a $\mathcal{C} \in \Pi$, such that $d_l, d_t \in \mathcal{C}$, then from Definition 1, it would have to hold $\|d_l - d_t\| \leq M$. This contradicts Eq. (3), thus d_l and d_t need to be in different clusters. Thus, the lemma is proved. □

In real life scenarios however cluster coherencies cannot be known in advance. Also even if they were known, it could be difficult to find a large enough empty space in a particular projection. It might be possible to design an algorithm that finds such a projection, however this is outside the scope of this paper and is left as a promising future direction.

Even if we do not know the real cluster coherencies, we can still claim that by splitting on the largest distance between two consecutive projections is a good choice, given the particular projection, as it does not split any contiguous cluster. In the lemma that follow we show in more detail what can be achieved by this splitting criterion.

Lemma 2. *Let $\mathcal{D}$ be a set of n points $d_i \in \mathbb{R}^a$, for $i=1,\ldots,n$ and $u \in \mathbb{R}^a$, with $\|u\| = 1$. Let $\mathcal{P}$ be the set of projections p_i, for $i=1,\ldots,n$, of the vectors $d_i \in \mathcal{D}$ on u. Also, let for each $p_i \in \mathcal{P}$, $\mathrm{ne}(i) = \arg\min_j\{p_j : p_j > p_i, p_j \in \mathcal{P}\}$ and define i_{sp} as follows:*

$$i_{sp} = \arg\max_i\{p_{\mathrm{ne}(i)} - p_i, \forall p_i \in \mathcal{P}\}.$$

Then, if $\mathcal{D}_{i_{sp}} = \{d_i \in \mathcal{D} : p_i \leq p_{i_{sp}}\}$, $M = p_{i_{sp}} - p_{j*}$, where $p_{j*} = \min\{p_j \in \mathcal{P} : p_j > p_{i_{sp}}\}$, then for any subset $\mathcal{D}'_{i_{sp}} \subset \mathcal{D}_{i_{sp}}$ for which $\mathrm{COH}(\mathcal{D}'_{i_{sp}}) \leq M$, it holds that

$$\mathrm{COH}(\mathcal{D}'_{i_{sp}}) \leq \mathrm{COH}(\mathcal{D}'_{i_{sp}} \cup \{d_j\})$$

for any $d_j \in \mathcal{D}$ for which $p_j > p_{i_{sp}}$.

Proof. First note that for $d_i \in \mathcal{D}'_{i_{sp}}$, it holds that $p_i \leq p_{i_{sp}}$. This way, for any $d_j \in \mathcal{D}$ such that $p_j \geq p_{j*}$ it holds that $p_i - p_j \geq M$. Subsequently, it also holds $\|d_i - d_j\| \geq M$. As such, from Definition 1, $\mathrm{COH}(\mathcal{D}'_{i_{sp}} \cup \{d_j\}) \geq M$. Since $\mathrm{COH}(\mathcal{D}'_{i_{sp}}) \leq M$, the Lemma is proved. □

An example of the application of this Lemma is illustrated in Fig. 3. In this example $\mathrm{COH}(\mathcal{D}_{i_{sp}})$, is larger than M, but the coherence of the subsets $\mathrm{COH}(\mathcal{D}_{i_{sp,1}})$, $\mathrm{COH}(\mathcal{D}_{i_{sp,2}})$, and $\mathrm{COH}(\mathcal{D}_{i_{sp,3}})$, is smaller than M. Adding any point d_j with $d_j \in \mathcal{D}$ for which $p_j > p_{i_{sp}}$, would only increase the coherence of any of those subsets, although the coherence of $\mathcal{D}_{i_{sp}}$ will decrease if we add p_{j*}, since it holds that $L_1 < L_2, L_3$, and $L_1 = \mathrm{COH}(\mathcal{D}_{i_{sp}})$.

The above lemmas point out that partitioning data based on the largest distance M between consecutive projections is guaranteed not to split a cluster, if the maximum coherence of each cluster is less than M. In the case that such knowledge is not available, splitting in such a manner guarantees that no cluster with less than M coherence, is split. In real life data sets, we do not know how dense the real clusters are. Thus, one would like to

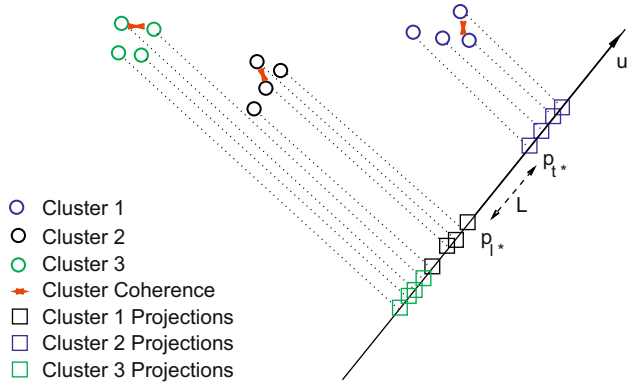

Fig. 2. An example application of Lemma 1.

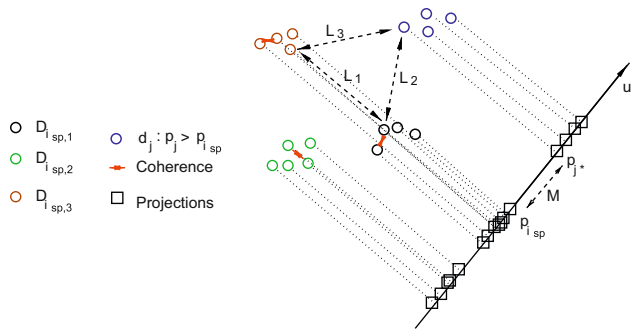

Fig. 3. An example application of Lemma 2.

discover clusters with coherence as large as possible. Given a particular projection, obviously the best strategy is to split on the largest distance between two consecutive projections. As proposed in [15], let us formally define the following splitting criterion:

- (Splitting criterion SPC_2): let $\forall p_i \in \mathcal{P}$, $\mathrm{ne}(i) = \arg\min_j\{p_j : p_j > p_i, \forall p_j \in \mathcal{P}\}$ and define i_{sp} as follows $i_{sp} = \arg\max_i\{ p_{\mathrm{ne}(i)} - p_i, \forall p_i \in \mathcal{P}\}$. Then, $P_1 = \{d_i \in \mathcal{D} : p_i \le p_{i_{sp}}\}$ and $P_2 = \{d_i \in \mathcal{D} : p_i > p_{i_{sp}}\}$.

Fig. 4 illustrates the result of this criterion on the exemplary dataset used previously, where all alternative criteria in the literature produce sub-optimal partitions. As shown, no cluster is split and only one of the points of one cluster is attributed to a wrong partition. Note also, that the result is heavily unbalanced in terms of number of points in each partition, where the alternative criteria produce somewhat balanced partitions.

3.1.2. Dense convex clusters

Although the SPC_2 criterion might seem appropriate in terms of cluster coherence, its main drawback is that it will operate sub-optimally in cases where there are outlying points in the data. In such situations all the distances between successive projections can be uniform and it could be impossible to make an informative splitting decision. However, this is not a problem of the particular criterion per se, but rather of the cluster definition that has been used. Nonetheless, we would like to be able to deal with such situations. In what follows, we will describe an alternative formulation of the clustering problem that allows us to design a splitting methodology able to deal with this situation. So, we are redefining clusters as convex subsets of the data space that are more dense than the regions around them.

Definition 4 (*Dense convex set*). If we assume $f : \mathbb{R}^a \to \mathbb{R}$ to be the probability density function of $x \in \mathbb{R}^a$, then a set $\mathcal{C} \subset \mathbb{R}^a$ is dense and convex if there is a set $\mathcal{C}' \subset \mathbb{R}^a$ such that $f(x) > f(y)$, $\forall x \in \mathcal{C}$, and $\forall y \in \mathcal{C}'$ and $\forall x,y \in C$ then $(1-t)x+ty \in C$, $\forall t \in [0,1]$.

The above definition determines clusters not in terms of the points they include, but their probability density function f. This way we can assume the data set $\mathcal{D}$, as a finite sample drawn from that distribution. Similarly, we can redefine the clustering problem as follows:

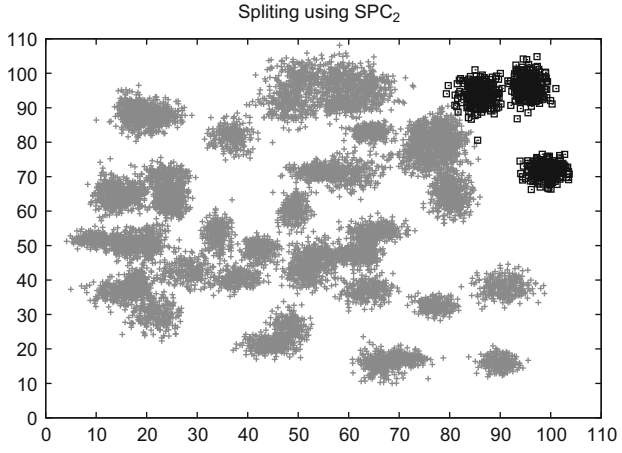

Fig. 4. The partitioning result of SPC_2 splitting criterion.

Definition 5 (*Dense convex clusterable set*). A set $\mathcal{D}$ of points is *dense convex k-clusterable* if there exist k disjoint subsets $\mathcal{C}_1,\ldots,\mathcal{C}_k$, such that the convex hull $H(\mathcal{C}_i)$ of any $\mathcal{C}_i$ is a dense convex set.

In practice, it is not possible to know in advance the real density of the data f, but we can construct estimates of it given the data set $\mathcal{D}$. There are approaches for such a task and among them *kernel density estimation* [23] is a well studied non-parametric estimation that does not assume any particular form for the real density f. In the most simple forms for a multivariate kernel density estimator is

$$\hat{f}(x;h) = n^{-1}h^{-a}\sum_{i=1}^{n} K((x-d_i)/h),$$

where $h \in \mathbb{R}^+$ is a real positive number called the bandwidth and $K : \mathbb{R}^a \to \mathbb{R}$ is a *kernel* function which needs to satisfy: $\int K(x)\,dx = 1$. A usual choice for the kernel is the standard multi-variate normal density:

$$K(x) = (2\pi)^{-a/2}e^{-0.5\|x\|}.$$

For the estimation properties of such a method refer to [23]. For any projection direction $u \in \mathbb{R}^a$, with $\|u\| = 1$, we can also estimate the univariate density of the projections $ux \in \mathbb{R}$, for $x \in \mathbb{R}^a$

$$\hat{f}'(ux;h) = n^{-1}h^{-1}\sum_{i=1}^{n} K((ux-p_i)/h).$$

Now, the kernel function used is:

$$K'(ux) = (2\pi)^{-1/2}e^{-0.5|ux|}.$$

Note that $K'(ux) \ge K(x)$, since $\|x\| \ge |ux|$, and the normal density is a monotonically decreasing function. Moreover

$$\hat{f}(x;h) \le \hat{f}'(ux;h'), \tag{4}$$

as long as $h = h' > 1$ or $h' \le h^a$. Using this estimation methodology, the following lemma can be proved:

Lemma 3. *Let a dense convex k-clusterable set $\mathcal{D}$ of points d_i, for $i=1,\ldots,n$ and Π its partition into k dense convex subsets $\mathcal{C}_1,\ldots,\mathcal{C}_k$. Let $u \in R^a$, with $\|u\| = 1$. Let $\mathcal{P}$ be the set of projections p_i of the vectors d_i on u. Also, let $M = \min\{\hat{f}(x;h) : x \in \mathcal{C}, \mathcal{C} \in \Pi\}$. If there exists $x \in \mathbb{R}$, such that the univariate estimate density $\hat{f}'(x;h') < M$, with $h = h' > 1$ or $h' \le h^a$, then all d_t for which $p_t > x$, and d_l for which $p_l \le x$, belong to different sets of Π.*

Proof. Let us suppose that there exist $d_t, d_l \in \mathcal{D}$, $\mathcal{C} \in P$ such that $d_t, d_l \in \mathcal{C}$, and $p_t = u\,d_t$ and $p_l = u\,d_l$ with $p_l \le x$ and $p_t > x$. Obviously, there exists $\lambda \in [0,1]$, such that $x = \lambda p_l + (1-\lambda)p_t$. Also for the vector $z \in \mathbb{R}^d$ with $z = \lambda d_l + (1-\lambda)d_t$, it holds that $z \in \mathcal{C}$ since $\mathcal{C}$ is convex (note that $uz = x$).

As such $\hat{f}'(uz;h') = \hat{f}'(x;h') < M$. However, from Eq. (4), we have that $\hat{f}(z;h) \le \hat{f}'(uz;h') = \hat{f}'(x;h')$ and since $z \in \mathcal{C}$, $\hat{f}(z;h) > M$, this is a contradiction. Thus, the Lemma is proved. □

This lemma extends the result of Lemma 1 in the case of dense convex clusterable sets. Now, all we need to find is a point on the projection line with low enough density. If we split based on that point we are guaranteed not to split any clusters. Of course the above Lemma assumes that the minimum density of the points in each cluster is known. This assumption could be unattainable in real life scenarios. Nevertheless, given a particular projection we could find the global minimiser x^* of the projections density $\hat{f}'(x;h')$ and split based on that point. That helps not to split any

clusters of density greater or equal to $\hat{f}'(x;h')$. However, x^* must be formally defined:

Definition 6 (*Global minimiser*). A global minimiser $x^* \in \mathbb{R}$, is the point that $\hat{f}'(x^*;h') \leq \hat{f}'(x';h')$, where $x' \in \mathcal{X} = \{x'' \in R : \exists \delta > 0, \hat{f}'(x'';h') < \hat{f}'(x;h'), \forall x \text{ for which } |x''-x| < \delta\}$.

Based on that definition the following splitting criterion is proposed:

- (Splitting criterion SPC_3): Let $\hat{f}'(x;h')$ be the kernel density estimation of the density of the projections $p_i \in \mathcal{P}$ and x^* its global minimiser given by Definition 6. Then construct $P_1 = \{d_i \in \mathcal{D} : p_i \leq x^*\}$ and $P_2 = \{d_i \in \mathcal{D} : p_i > x^*\}$.

Note, that finding such a minimiser might seem more difficult than it actually is. The reason for that lies in the fact that valid splitting points lie between consecutive data projections p_i. Also, between any two consecutive projections p_i and p_{i+1} the estimated density $\hat{f}'(x;h')$, for $p_i \leq x \leq p_{i+1}$, is monotonous, due to the manner that this is calculated. As such we can constrain the search for the global minimiser on the finite set $\mathcal{P}$ and set δ for each p_i, as the minimum of $|p_{i+1}-p_i|$ and $|p_{i-1}-p_i|$, where p_{i-1}, and p_{i+1} are the two nearest projections of p_i from either side.

Let us examine the result of this criterion on the exemplary dataset used above. Fig. 5 (left) illustrates the result that is similar to the one obtained by SPC_2. The difference lies in the fact that the outlying point that was misplaced under SPC_2, does not affect the SPC_3 criterion and it is assigned to the correct sub-partition. It is interesting to inspect the estimated density shown in Fig. 5 (right). The estimate density exhibits several minima. However, especially one of those minima has a very low density value (very close to 0) that provides a quite good indication that no cluster will be split.

This splitting criterion suggests that the minimiser of the estimated density is the best we can do to avoid splitting clusters. However, there always exists the problem of not being able to find such a minimiser. This could happen in two cases; either the data do not contain any density based convex clusters, or the selected bandwidth is very large [23]. The first case can act as an indication when to stop the recursive splitting of the data, under the assumption that the density estimation is accurate enough. The second case, is a well studied problem, especially in the particular one dimensional situation [24]. The way that the bandwidth selection influences the estimation can be explained by examining the asymptotic mean squared error (AMISE). Small values of h increase the (asymptotic) variance and thus the resulting estimate $\hat{f}(x;h)$ seems "wiggly" with many spurious features if graphically checked. On the other hand, big values of h reduce the (asymptotic) variance of $\hat{f}(x;h)$, but also increase the (asymptotic) bias, probably "smoothing away" the features of the true density f.

Numerous methods have been proposed to automatically select the bandwidth. For more details refer to [24], where various techniques to automatically tune the bandwidth are discussed and their particular characteristics are explained. A cross-validation selection method was proposed in [25], while [26] proposed a plug-in estimate which minimises an estimate of the mean weighted integrated squared error, using the density function as a weight function. In [27] it was proposed to choose the bandwidth depending on the roughness of the first derivative of the density, which is straightforward to estimate. In this work, we just follow the "normal reference rule", which suggests to use a bandwidth h_{opt} that minimises the Mean Integrated Squared Error (MISE). This is given by

$$h_{opt} = \sigma\left(\frac{4}{3n}\right)^{1/5}, \tag{5}$$

where σ is standard deviation of the data. In Section 7, we further investigate how this can affect the performance of the clustering procedure.

3.2. Which cluster to split?

The second question that any divisive algorithm needs to face is which cluster to select from the pool of already retrieved partitions, to forego with the clustering procedure. It is obvious that this step of the procedure is quite important as well. Being able to split effectively has no merit, when the selected cluster to split corresponds to a coherent group. Imposing a clustering structure on a dataset, when such a structure does not exist, is a central issue in cluster analysis. In general, determining the presence or the absence of a clustering structure is called the "cluster tendency" problem. A series of specialised tests have been developed to judge the existence of a clustering structure, before the application of a clustering algorithm, like the scan test, quadrant analysis, moment structure and inter-point distances [28]. However, it still remains an open issue.

As already discussed in Section 2, the PDDP algorithm, as well as all its variations [16–20], select the cluster with maximum scatter value SV (Eq. (1)). Another explanation of this selection method is that the cluster having in effect the widest (largest variance) projection on the first principal component is chosen ("shoot the biggest animal").

This can be very easily deemed unsuccessful through a simple example as shown in [15], which for completeness purposes is included here as well. Fig. 6 illustrates a case where the dataset has already been split into two clusters shown with different colours. In this case, the green large cluster has a SV value of 0.25414, which is 0.17945 larger than the SV value of the red cluster. So, in this case selecting the cluster with the larger SV value for further splitting would have no chance of producing a correct clustering.

The splitting criteria proposed in the previous section could also provide guidance for the cluster selection step. In the pool of already retrieved clusters, we expect the one with the largest

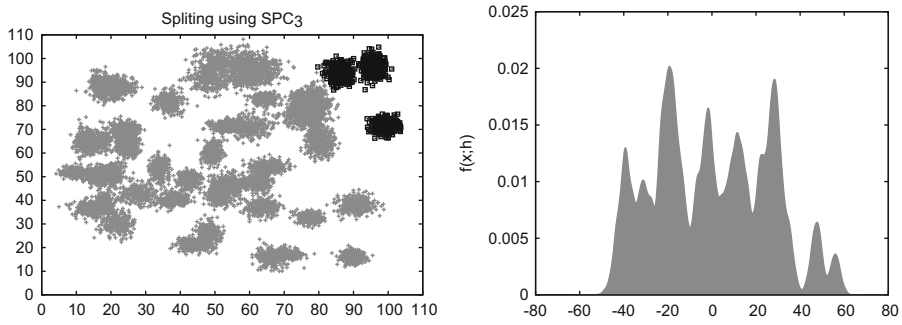

Fig. 5. The partitioning result of SPC_3 splitting criterion (left) and the density estimation of the projections (right).

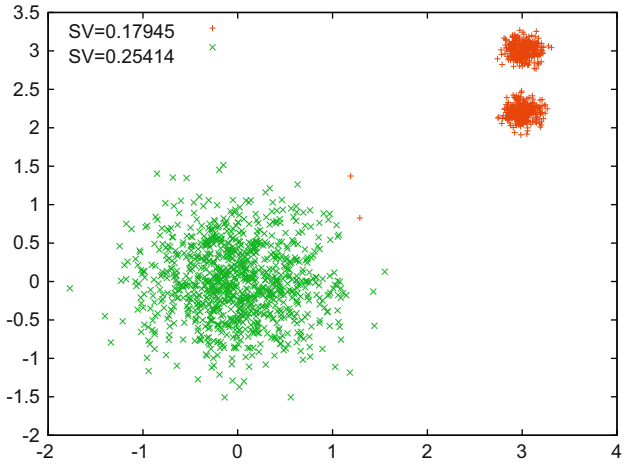

Fig. 6. A dataset of unbalanced clusters and the corresponding scatter values.

distance among consecutive projections to probably contain more than one actual clusters.

The same can be said for the minimum estimated density criterion. A minimiser with very small density should be a good indicator of multi-modality of the density function and consequently it lessens the chance to split an actual cluster. Thus we propose the following two selection criteria:

- (Cluster selection criterion CS_1): Let Π a partition of the data set $\mathcal{D}$ into k sets. Let $\mathcal{M}=\{M_i : i=1,\ldots,k\}$ be the set of the largest distances M_i among consecutive projections, for each $C_i \in \Pi$, $i=1,\ldots,k$. The next set to split is C_j, with $j=\arg\max_i\{M_i : M_i \in \mathcal{M}\}$.
- (Cluster selection criterion CS_2): Let Π a partition of the data set $\mathcal{D}$ into k sets. Let $\mathcal{F}$ be the set of the density estimates $f_i=\hat{f}(x_i^*;h)$ of the minimisers x_i^* for the projection of the data of each $C_i \in \Pi$, $i=1,\ldots,k$. The next set to split is C_j, with $j=\arg\max_i\{f_i : f_i \in \mathcal{F}\}$.

3.3. When should the iteration terminate?

Irrespectively of the method used, a fundamental issue in cluster analysis is the determination of the number of clusters present in a dataset. This issue remains an open problem in cluster analysis. For instance, well-known and widely used iterative techniques, such as the k-means algorithm [3], require from the user a priori designation of the number of clusters present in the dataset. There also exist approaches that adjust the number of clusters during training. The ISODATA technique [29] is based on the same key idea as the k-means algorithm; starting with a typical number of initial clusters, it iteratively merges and splits existing clusters according to "within-group variability" and "closeness" thresholds. Another popular approach is to employ Akaike information criterion (AIC) and Bayesian information criterion (BIC) to choose among partitions with different number of clusters. In the same theme the integrated completed likelihood (ICL) criterion has been proposed as more appropriate for clustering purposes [30]. In this category of algorithms PG-means [31] aims to learn the number of components of a Gaussian mixture modelling approach, using statistical hypothesis tests on one-dimensional projections of the data. The computationally efficient x-means algorithm, proposed in [32], is another popular approach from the class of partitioning algorithms that has the ability to approximate the number of clusters in the data.

One of the most promising approaches from the density based category of algorithms is DBSCAN [6]. DBSCAN is a density-based clustering algorithm that tries to recover clusters from spatial databases and automatically decides the number of clusters. Clusters are defined by means of neighbourhoods of objects. The density of each such neighbourhood has to exceed some threshold. The value of the threshold is critical for the execution of the algorithm and heuristics have been proposed to determine it. Finally, there exist some recent agglomerative hierarchical clustering algorithms that have been shown to be able to achieve high quality clustering results, such as BIRCH [33], CHAMELEON [34] and CURE [35]. However, these type of algorithms require higher user intervention to provide accurate estimations for the cluster number.

There also exist a few grid-based algorithms [36,37], that have been shown to be able to produce good clustering results. One of the most notable of these is CLIQUE [37]. Their biggest drawback is that they have a running time that is exponential to the data dimensionality. To be more precise, they are exponential not to the actual data dimensionality, but to the dimensionality of the subspaces where the clusters reside and is possibly much smaller than the full data dimensionality. So they are more fitted to operate on cases where specific clusters lie on few dimensions and medium data dimensions (e.g. 20, 40), and not in cases where dimensions lie in the range of thousands.

Little has been done however to develop an efficient technique for PDDP based approaches. The crudest approach would be to stop the execution when all the discovered clusters have a scatter value that is smaller than a predefined value, but the tuning of this parameter can be difficult from a user perspective. The criteria used in other algorithms could also be employed in the PDDP case. For example in [38] it is proposed to use BIC to determine if a further split would improve the clustering result or not. Additionally, we could use nearest neighbour statistics like the ones used in cluster tendency [28,39].

In [15], a termination criterion based on the maximum distance between consecutive projections was proposed. More specifically, we propose to have a maximum number of allowed clusters k_{max} as an upper bound and subsequently continue splitting as long as there exists clusters with more than *MinPts* points, where *MinPts* is a user defined parameter to describe the minimum number of points that are allowed to constitute a valid cluster. Notice that this is not an uncommon procedure for algorithms that are designed to deal with noisy datasets [6]. In this case, it is indirectly assumed that the distances between two outliers are larger than any two points of a cluster. Formally the termination criterion is the following based on the two user defined parameters k_{max} and *MinPts*:

- (Stopping criterion ST_1): Iteratively split the data set $\mathcal{D}$ into k_{max} subsets. Report as clusters the ones with more than *MinPts* points. Designate the remaining points as outliers.

For the density based approach presented here, we could allow the existence of a minimiser to guide the termination of the procedure. We can stop the iteration as long as no minimiser exists for any of the retrieved clusters. This stopping criterion makes the assumption that all the retrieved clusters are uniformly dense, so they cannot be further split. Note however, that this depends on the bandwidth selection and automated bandwidth selection techniques could be employed to remove user intervention. Formally:

- (Stopping criterion ST_2): Let Π a partition of the data set $\mathcal{D}$ into k sets. Let $\mathcal{X}$ be the set of minimisers x_i^* of the density estimates $\hat{f}(x_i^*;h)$ of the projection of the data of each $C_i \in \Pi$, $i=1,\ldots,k$. Stop the procedure when the set $\mathcal{X}$ is empty.

4. Algorithms

In this section we present algorithmic constructions based on the different proposed criteria in the earlier sections. These particular constructions are just employed to investigate (through experimental analysis) the effectiveness of these criteria. In principle, more algorithms having different characteristics can be constructed using any combination of criteria. The particular implementations do not try to mix different approaches, but are pure in the sense that they employ criteria from a particular methodology. Thus, we have two algorithmic implementations. The first implementation (iPDDP) is based on the idea of splitting based on the largest distance between any two consecutive projections. The second implementation (dePDDP) is a compilation of the criteria that are based on the minimiser of the estimated density of the projections.

Table 1
The iPDDP algorithm summary.

Function iPDDP ($\mathcal{D}$,k_{max},*MinPts*) {
1. Set $\Pi=\{\mathcal{D}\}$
2. While $|\Pi|<k_{max}$, do
3. Select a set $\mathcal{C}\in\Pi$, using cluster selection criterion CS_1
4. Split $\mathcal{C}$ into two sub-sets $\mathcal{C}_1$ and $\mathcal{C}_2$, using Splitting Criterion SPC_2
5. Remove $\mathcal{C}$ from Π and set $\Pi\rightarrow\Pi\cup\{\mathcal{C}_1,\mathcal{C}_2\}$
6. Set $\mathcal{O}=\emptyset$
7. For any $\mathcal{C}$ in Π with $|\mathcal{C}|<MinPts$, do
8. Remove $\mathcal{C}$ from Π
9. Set $\mathcal{C}$ with $|\mathcal{C}|<MinPts$ and set $\mathcal{O}\rightarrow\mathcal{O}\cup\mathcal{C}$,
10. Return Π the partition of $\mathcal{D}$ into $|\Pi|$ clusters and $\mathcal{O}$ the set of outliers.
}

Table 2
The dePDDP algorithm summary.

Function dePDDP ($\mathcal{D}$) {
1. Set $\Pi=\{\mathcal{D}\}$
2. Do
3. Select a set $\mathcal{C}\in\Pi$, using cluster selection criterion CS_2
3. Split $\mathcal{C}$ into two sub-sets $\mathcal{C}_1$ and $\mathcal{C}_2$ using Splitting Criterion SPC_3
4. Remove $\mathcal{C}$ from Π and set $\Pi\rightarrow\Pi\cup\{\mathcal{C}_1,\mathcal{C}_2\}$
5. While Stopping Criterion ST_2 is not satisfied
6. Return Π the partition of $\mathcal{D}$ into $|\Pi|$ clusters
}

The iPDDP implementation is shown in Table 1. This algorithmic schema is build around the stopping criterion ST_1, the cluster selection criterion CS_1, and the spitting criterion SPC_2. This has the drawback that the user needs to specify the maximum number of clusters k_{max} and the *MinPts* parameter to describe the minimum number of points that are allowed to constitute a valid cluster. Although the selection of *MinPts* parameter is easy, the selection of k_{max} is not that straightforward. If k_{max} is selected to obtain a much larger value than the actual number of clusters the algorithm will be forced to iteratively strip points from the clusters. In some extreme cases, a cluster could be totally decomposed into smaller sets of less than *MinPts* points, eventually ending up in the outlier set. The effect of these parameters is further investigated in Section 6.

The computational complexity of the iPDDP implementation is mostly influenced by the computation of the principal vectors as in the original PDDP algorithm. To compute this, the singular value decomposition of the data matrix D is employed. This introduces a total worst case complexity of $O(k_{max}(2+k_{SVD})s_{nz}\,n\,a)$, where k_{SVD} are the iterations needed by the Lanczos SVD computation algorithm and s_{nz} is the fraction of non-zero entries in D (for more details refer to [12]). In the iPDDP case, the additional

Table 3
Mean purity and V-measure (with the observed standard deviation in parenthesis), for $DSET_{\text{Gaussian}}$ generated data of different algorithms, over 100 experiments.

	15 Clusters		25 Clusters		50 Clusters	
	Pur.	V-meas.	Pur.	V-meas.	Pur.	V-meas.
Dimension 2						
PDDP	0.88(0.04)	0.89(0.03)	0.83(0.03)	0.86(0.02)	0.79(0.02)	0.86(0.01)
iPDDP	0.89(0.03)	0.88(0.06)	0.89(0.02)	0.77(0.15)	0.93(0.04)	0.38(0.24)
dePDDP	0.95(0.03)	0.96(0.02)	0.93(0.03)	0.93(0.02)	0.86(0.02)	0.91(0.01)
KM-PDDP	0.91(0.04)	0.93(0.03)	0.88(0.03)	0.79(0.22)	0.86(0.03)	0.71(0.06)
GMM	0.93(0.03)	0.94(0.03)	0.87(0.02)	0.91(0.02)	0.80(0.01)	0.90(0.00)
k-means	0.92(0.04)	0.93(0.03)	0.90(0.03)	0.92(0.02)	0.84(0.00)	0.90(0.00)
Dimension 5						
PDDP	0.94(0.03)	0.95(0.02)	0.92(0.03)	0.94(0.02)	0.89(0.02)	0.92(0.02)
iPDDP	1.00(0.01)	1.00(0.01)	0.98(0.02)	0.98(0.03)	0.97(0.01)	0.66(0.25)
dePDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.99(0.00)
KM-PDDP	0.98(0.02)	0.99(0.01)	0.96(0.03)	0.97(0.02)	0.91(0.05)	0.76(0.33)
GMM	0.96(0.02)	0.97(0.01)	0.92(0.02)	0.96(0.01)	0.90(0.02)	0.96(0.01)
k-means	0.93(0.02)	0.94(0.02)	0.93(0.02)	0.95(0.01)	0.93(0.01)	0.96(0.01)
Dimension 20						
PDDP	0.97(0.02)	0.97(0.02)	0.96(0.02)	0.95(0.04)	0.96(0.01)	0.94(0.03)
iPDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.99(0.01)	0.92(0.20)
dePDDP	1.00(0.00)	0.99(0.01)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.99(0.00)
KM-PDDP	0.98(0.03)	0.99(0.02)	0.96(0.03)	0.98(0.01)	0.90(0.03)	0.92(0.12)
GMM	0.93(0.03)	0.94(0.02)	0.90(0.02)	0.94(0.01)	0.90(0.01)	0.95(0.01)
k-means	0.94(0.02)	0.96(0.02)	0.93(0.02)	0.95(0.01)	0.92(0.01)	0.95(0.01)
Dimension 50						
PDDP	0.97(0.02)	0.97(0.03)	0.97(0.02)	0.97(0.02)	0.97(0.01)	0.94(0.02)
iPDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.01)	1.00(0.01)
dePDDP	1.00(0.00)	0.99(0.01)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.99(0.00)
KM-PDDP	0.95(0.03)	0.97(0.02)	0.93(0.03)	0.96(0.01)	0.90(0.02)	0.96(0.01)
GMM	0.89(0.03)	0.89(0.03)	0.88(0.02)	0.91(0.02)	0.87(0.02)	0.91(0.01)
k-means	0.94(0.02)	0.95(0.01)	0.93(0.02)	0.95(0.01)	0.92(0.01)	0.95(0.01)

computation steps that are required, increase the complexity to $O(k_{max}(2+k_{SVD})(s_{nz}na+n\log(n)))$, which although increased is still on par with the most clustering algorithms. Notice that the additional cost is not influenced by the data dimensionality. Thus, the ability of the algorithms to deal with ultra high dimensional data is maintained.

The proposed dePDDP implementation is shown in Table 2. This is just a compilation of the criteria SPC_3 for the cluster splitting, CS_2 for the cluster selection, and ST_2 for the termination of the algorithm. The computational complexity of this approach, using a brute force technique, would be quadratic in the number of samples. However, it has been shown [40,41] that using techniques like the fast Gauss transform we achieve linear running time for the kernel density estimation, especially for the one dimensional case at hand. To find the minimiser of that we only need to evaluate the density at n positions, in between the projected data points, since those are the only places we can have valid splitting points. Thus, the total complexity of the algorithm remains $O(k_{max}(2+k_{SVD})(s_{nz}na))$.

5. Experimental analysis

This part of the paper is devoted to the experimental evaluation of the algorithmic implementations proposed in the previous section. To this end, we firstly employ a series of simulated datasets. This procedure gives the opportunity to predesign (and hence know beforehand) the structure of the data that the clustering procedure aims to recover. This kind of artificial cluster construction method is typically used in similar empirical evaluations [19,38,42].

There are different way to assess the quality of a data partition [10,43]. First, we could formulate quality as a function of the given data (internal criteria). In this case the clustering problem actually becomes an optimisation problem. A better method would be to use additional external information not available to the algorithm, such as class labels. Then, we can measure the degree of correspondence between the resulting clusters and the classes assigned a priori to each object. For a dataset $\mathcal{D}$, let $\mathcal{L}$ be a set of labels $l_i \in \mathcal{L}$, for each point $d_i \in \mathcal{D}$, $i=1,\ldots,n$, with l_i taking values in $\{1,\ldots,L\}$. Let a k-cluster partitioning $\Pi=\{\mathcal{C}_1,\ldots,\ldots,\mathcal{C}_k\}$. The purity of Π is defined as

$$P(\Pi)=\frac{\sum_{j=1}^{k}\max\{|\{p_i\in\mathcal{C}_j : l_i=1\}|,\ldots,|\{p_i\in\mathcal{C} : l_i=L\}|\}}{n}, \tag{6}$$

so that $0\le P(\Pi)\le 1$. High values indicate that the majority of vectors in each cluster come from the same class, so in essence the partitioning is "pure" with respect to class labels.

However, cluster purity does not address the question of whether all members of a given class are included in a single cluster and therefore is expected to increase monotonically with the number of clusters in the result. For this reason, criteria like the V-measure [44] have been proposed. The V-measure tries to capture cluster homogeneity and completeness, which summarises a clustering solution's success in including every point of a single class and no others. For details on how these are calculated, the interested reader should refer to [44].

In the first set of experiments, we construct datasets by drawing points from a finite mixture of k Gaussian distributions that represent the actual clusters in the data. The mean of each Gaussian is randomly placed in $[100,200]^a$ and the covariance matrix is also randomly generated by an appropriate procedure, so as to ensure that it is symmetric and positive definite. Next, from each distribution 100 points are drawn, so the total number of points in each dataset is $k\times 100$. We will refer to this kind of data generation mechanism $DSET_{\text{Gaussian}}$.

Table 4
Mean purity and V-measure (with the observed standard deviation in parenthesis), for $DSET_{\text{Beta}}$ generated data of different algorithms, over 100 experiments.

	15 Clusters		25 Clusters		50 Clusters	
	Pur.	V-meas.	Pur.	V-meas.	Pur.	V-meas.
Dimension 2						
PDDP	0.87(0.04)	0.88(0.03)	0.83(0.03)	0.86(0.02)	0.76(0.02)	0.84(0.01)
iPDDP	0.90(0.04)	0.89(0.05)	0.89(0.03)	0.80(0.13)	0.91(0.04)	0.58(0.18)
dePDDP	0.95(0.02)	0.94(0.02)	0.91(0.04)	0.92(0.02)	0.84(0.03)	0.88(0.01)
KM-PDDP	0.89(0.03)	0.89(0.04)	0.86(0.04)	0.76(0.23)	0.86(0.04)	0.63(0.19)
GMM	0.92(0.02)	0.92(0.02)	0.87(0.03)	0.91(0.02)	0.23(0.10)	0.61(0.13)
k-means	0.91(0.03)	0.92(0.02)	0.88(0.02)	0.91(0.02)	0.81(0.04)	0.88(0.01)
Dimension 5						
PDDP	0.94(0.03)	0.94(0.03)	0.91(0.03)	0.93(0.02)	0.88(0.02)	0.91(0.02)
iPDDP	1.00(0.01)	1.00(0.01)	0.98(0.02)	0.98(0.03)	0.98(0.01)	0.50(0.24)
dePDDP	1.00(0.00)	0.99(0.01)	1.00(0.00)	0.99(0.01)	1.00(0.00)	0.99(0.00)
KM-PDDP	0.94(0.04)	0.96(0.03)	0.93(0.02)	0.96(0.02)	0.90(0.04)	0.68(0.33)
GMM	0.95(0.03)	0.96(0.01)	0.93(0.02)	0.96(0.01)	0.91(0.01)	0.96(0.01)
k-means	0.94(0.03)	0.95(0.02)	0.93(0.02)	0.95(0.01)	0.92(0.01)	0.96(0.01)
Dimension 20						
PDDP	0.96(0.02)	0.96(0.03)	0.96(0.02)	0.96(0.02)	0.95(0.01)	0.92(0.02)
iPDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.98(0.01)	0.80(0.27)
dePDDP	1.00(0.00)	0.99(0.01)	1.00(0.00)	0.99(0.01)	1.00(0.00)	0.99(0.00)
KM-PDDP	0.92(0.04)	0.95(0.03)	0.92(0.03)	0.96(0.01)	0.87(0.03)	0.92(0.06)
GMM	0.94(0.03)	0.96(0.01)	0.92(0.02)	0.95(0.01)	0.91(0.02)	0.96(0.01)
k-means	0.94(0.02)	0.95(0.02)	0.93(0.02)	0.95(0.01)	0.93(0.01)	0.95(0.01)
Dimension 50						
PDDP	0.97(0.02)	0.97(0.02)	0.97(0.01)	0.95(0.03)	0.97(0.01)	0.94(0.02)
iPDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.99(0.01)	0.97(0.07)
dePDDP	1.00(0.00)	0.99(0.01)	1.00(0.00)	0.99(0.00)	1.00(0.00)	0.99(0.00)
KM-PDDP	0.88(0.03)	0.93(0.02)	0.89(0.03)	0.95(0.02)	0.88(0.02)	0.95(0.02)
GMM	0.88(0.03)	0.87(0.03)	0.88(0.03)	0.91(0.02)	0.87(0.02)	0.92(0.01)
k-means	0.94(0.02)	0.95(0.02)	0.93(0.02)	0.95(0.01)	0.92(0.01)	0.95(0.01)

Table 3 reports the purity and V-measure of the PDDP, iPDDP, dePDDP, KM-PDDP [20], Gaussian mixture model (GMM), and k-means algorithms in 100 randomly generated datasets, using the previously described $DSET_{\text{Gaussian}}$. Each entry of the table is the mean observed value of the corresponding measure obtained over the 100 different datasets and the number in parentheses is the observed standard deviation. A standard deviation of 0 corresponds to a observed variance of less than 10^{-2}.

As shown, the performance of PDDP, KM-PDDP, and GMM algorithms deteriorates as the number of clusters increases. Regarding the V-measure, the same is true only for that low dimensional cases. The reason for this lies in the fact that in low dimensions a kind of "chaining" effect [6] can appear between clusters. On the other hand, the dePDDP algorithm is marginally affected and produces high quality results in all cases. Note that the *MinPts* parameters for iPDDP was set to 5 and k_{max} was set to the actual number of clusters across all experiments. The actual cluster number was also given as input to all algorithms except dePDDP. The bandwidth parameter of dePDDP was set to twice the optimal value given by Eq. (5). Further explanation for this value is given in Section 7. Notice that this does not guarantee that the algorithm will stop to the actual cluster number, but this issue will be further explored below (see Section 5.2).

To extend the generality of these results we have also constructed in a similar manner datasets using Beta distributions. In detail, the actual clusters are composed by independent univariate Beta distributions, one for each dimension, of which the shape parameters are drawn at random uniformly in the interval [1,6]. After drawing 100 points from each cluster, the data for each one is rescaled by a random factor (which is drawn uniformly in [10,20]) and subsequently repositioned again randomly in [100,200]. This data generation mechanism

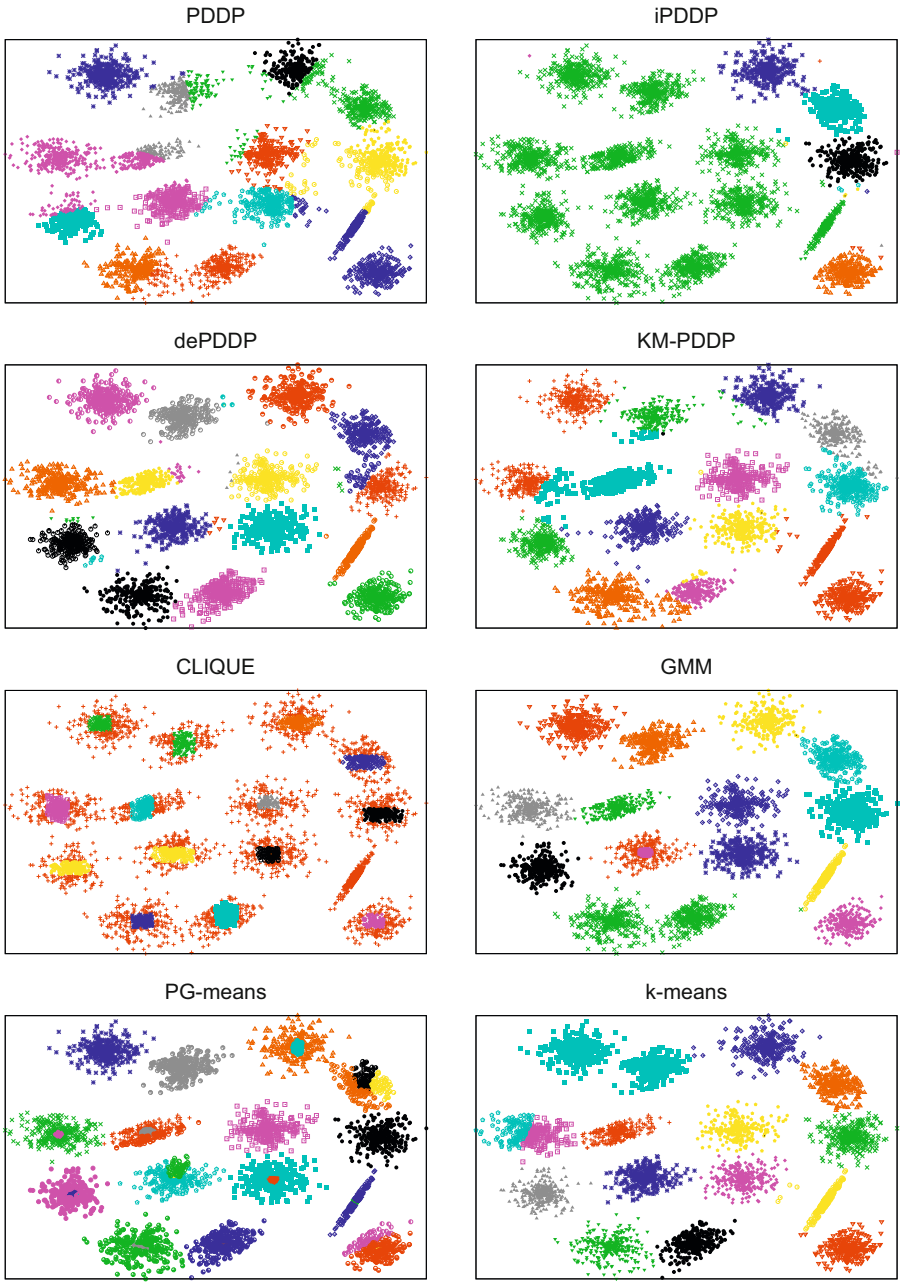

Fig. 7. Visual comparison of different algorithms on the S1 example dataset. Different colours and point types correspond to different clusters. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

generates clusters of random shapes depending on the values of the parameters of the Beta distributions. We will refer to this data generation mechanism as $DSET_{\text{Beta}}$.

The results for the purity and V-measure are tabulated in Table 4. Again, the reported values are the mean observed measure values in 100 different experiments and the number in parenthesis is the standard deviation. The results are similar to the $DSET_{\text{Gaussian}}$ case. The exception is GMM as it shows worse results, especially in the low dimensional high cluster number case. This is expected since the assumption of a Gaussian mixture on which GMM operates does not hold, so cluster recovery is hindered. However, all the PDDP variants show similar results, with dePDDP to produce stinkingly accurate clusterings. We have also followed the same procedure for the t and log-normal distributions but the results are very similar so they are not included for brevity.

To facilitate a more direct understanding of the results of each algorithm, we will use two two-dimensional datasets that are publicly available at http://cs.joensuu.fi/sipu/datasets/ and will resort to visual inspection of the results. These datasets S1 and S4, contain 5000 data points in 15 Gaussian clusters with different degree of cluster overlapping. In this case, we will test two additional algorithms. The first one is the CLIQUE algorithm [37], that were unable to use it in the previous experiments, as we only had a two-dimensional implementation. The second algorithm is the PG-means [31], which is considered a more sophisticated GMM approach.

The results are illustrated in Figs. 7 and 8, for S1 and S4, respectively. In the S1 case iPDDP retrieves only 4 of the clusters and due to the chaining effect fails to retrieve any more. In the S4 case it does not manage to produce sensible clustering, for the same reason. The high degree of cluster overlapping also hinders

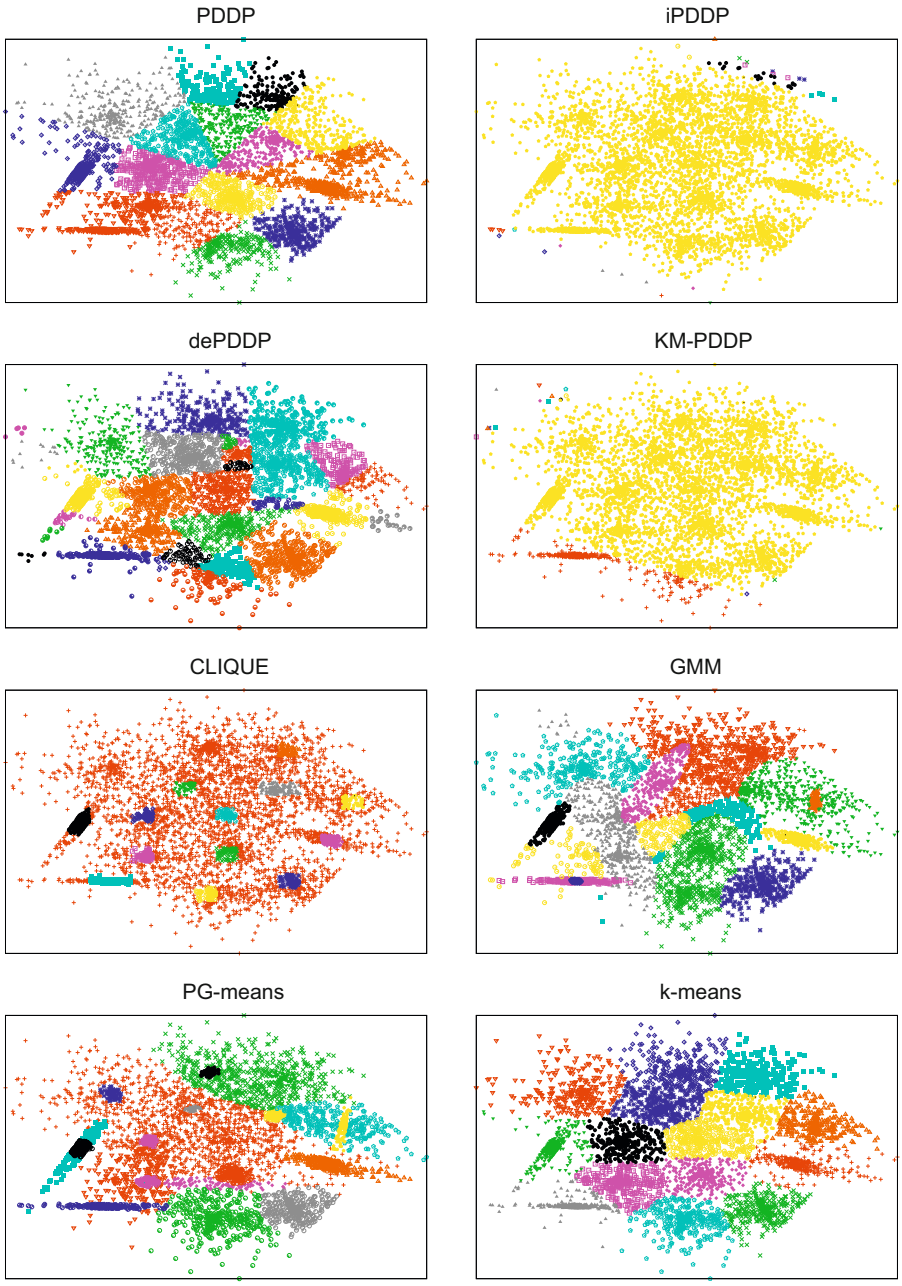

Fig. 8. Visual comparison of different algorithms on the S4 example dataset. Different colours and point types correspond to different clusters. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the KM-PDDP and GMM algorithms, both of them producing bad results in the case of S4 dataset. PG-means seems to overestimate the number of clusters, splitting most clusters in 2 or 3 parts. However, it can clearly identify the dense core of each cluster, but quite often this happens by more than two overlapping components which make the partitions quite confusing. Although this is a GMM based approach it shows much better results in the S4 case, substantially outperforming the standard GMM approach, but still there seem to exist one or two overlapping components. On the other hand, the high degree of cluster overlapping seems to help the *k*-means algorithm, resulting in a very good result for the S4 dataset, improving on the result of S1 dataset where one of the clusters is split and two others are merged. CLIQUE in both cases manages to retrieve the core of the clusters, but one or two clusters are totally missed. This probably happens because CLIQUE is very sensitive to user intervention. It is very difficult to choose parameters that retrieve all the clusters without merging any of them (especially in the S4 case). Finally, dePDDP produces very good results in both S1 and S4. In the S4 case, dePDDP retrieves 2 or 3 more clusters than the 15 actual ones, but this is due to the large amount of noise points.

5.1. Datasets with noise

In real life situations we are expecting to find cases where the data is contaminated with noise. To examine the performance of the different algorithms in such cases we are going to construct appropriate simulation settings. For this reason, we first employ the $DSET_{\text{Gaussian}}$ data generation mechanism and in each dataset we include a number of uniform randomly drawn points in the data space to represent noise. First, we are going to examine the case of 1000 noise points.

The results are shown in Table 5 with respect to cluster purity and V-measure, respectively. The reported results are again the mean values over 100 experiments, with the observed standard deviation shown in parenthesis. In this case, in low dimensions all the algorithms perform poorly. It seems that the noise contamination is so high that it makes it impossible for the algorithms to retrieve the cluster structure (especially the iPDDP and the KM-PDDP algorithms). Even in this case, the dePDDP algorithm produces the best results. However, as the dimensionality of the data increases, the clustering structure becomes more prominent and the algorithms manage to successfully retrieve it. The dePDDP algorithm stands out in this case, as it results in very pure clusterings and is the least affected by noise, irrespective of the number of clusters in the data. The iPDDP algorithm, as expected, manages to provide pure partitions when either the dimensionality is very large or the number noise points is small with respect to the total number of points in the data (cases with large number of clusters). However, in those cases PDDP and KM-PDDP perform similarly well. We have also performed the same set of experiments for the case of 2000 noise points, but the results are similar and for briefness are not included.

To better illustrate the effect of noise contamination, in Fig. 9 we plot the purity and the V-measure of different algorithms for an increasing percentage of noisy points in the data set. These plots refer to a particular case of 5 dimensions and 15 classes. The *x*-axis of this plot corresponds to the percentage of the noisy points in the dataset. For each algorithm the mean obtained

Table 5
Mean purity and V-measure (with the observed standard deviation in parenthesis), for $DSET_{\text{Gaussian}}$ generated data contaminated with 1000 noise points of different algorithms, over 100 experiments.

	15 Clusters		25 Clusters		50 Clusters	
	Pur.	V-meas.	Pur.	V-meas.	Pur.	V-meas.
Dimension 2						
PDDP	0.70(0.05)	0.80(0.04)	0.66(0.05)	0.79(0.03)	0.63(0.04)	0.78(0.02)
iPDDP	0.11(0.14)	0.01(0.04)	0.21(0.23)	0.01(0.03)	0.67(0.29)	0.05(0.04)
dePDDP	0.87(0.08)	0.89(0.12)	0.88(0.06)	0.92(0.03)	0.84(0.03)	0.88(0.02)
KM-PDDP	0.29(0.17)	0.32(0.23)	0.38(0.22)	0.44(0.24)	0.71(0.25)	0.38(0.21)
GMM	0.71(0.04)	0.84(0.02)	0.70(0.04)	0.84(0.02)	0.64(0.03)	0.83(0.02)
k-means	0.79(0.05)	0.90(0.03)	0.79(0.05)	0.91(0.02)	0.80(0.02)	0.90(0.01)
Dimension 5						
PDDP	0.78(0.06)	0.84(0.05)	0.77(0.06)	0.85(0.04)	0.78(0.04)	0.87(0.01)
iPDDP	0.22(0.23)	0.05(0.08)	0.56(0.27)	0.15(0.09)	0.85(0.11)	0.16(0.09)
dePDDP	0.98(0.02)	0.99(0.01)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.99(0.00)
KM-PDDP	0.65(0.17)	0.65(0.28)	0.62(0.22)	0.48(0.32)	0.87(0.09)	0.25(0.25)
GMM	0.74(0.05)	0.88(0.01)	0.75(0.04)	0.90(0.01)	0.75(0.03)	0.92(0.01)
k-means	0.87(0.04)	0.94(0.01)	0.90(0.03)	0.96(0.01)	0.88(0.02)	0.96(0.01)
Dimension 20						
PDDP	0.92(0.03)	0.88(0.11)	0.92(0.03)	0.79(0.10)	0.92(0.04)	0.68(0.07)
iPDDP	0.80(0.04)	0.47(0.12)	0.87(0.04)	0.37(0.10)	0.95(0.02)	0.29(0.13)
dePDDP	0.95(0.20)	0.95(0.22)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)
KM-PDDP	0.87(0.04)	0.93(0.03)	0.84(0.04)	0.91(0.03)	0.87(0.06)	0.55(0.32)
GMM	0.77(0.06)	0.90(0.02)	0.75(0.03)	0.91(0.01)	0.80(0.03)	0.93(0.01)
k-means	0.86(0.04)	0.93(0.02)	0.88(0.03)	0.95(0.01)	0.88(0.02)	0.96(0.00)
Dimension 50						
PDDP	0.89(0.03)	0.82(0.06)	0.93(0.01)	0.74(0.07)	0.92(0.04)	0.65(0.07)
iPDDP	0.93(0.03)	0.88(0.11)	0.95(0.01)	0.82(0.10)	0.97(0.01)	0.57(0.21)
dePDDP	1.00(0.00)	1.00(0.01)	1.00(0.00)	1.00(0.01)	1.00(0.00)	1.00(0.00)
KM-PDDP	0.89(0.05)	0.95(0.02)	0.86(0.05)	0.94(0.02)	0.87(0.03)	0.86(0.11)
GMM	0.64(0.08)	0.82(0.04)	0.70(0.06)	0.87(0.02)	0.77(0.03)	0.91(0.01)
k-means	0.87(0.04)	0.93(0.02)	0.86(0.03)	0.94(0.01)	0.88(0.02)	0.95(0.01)

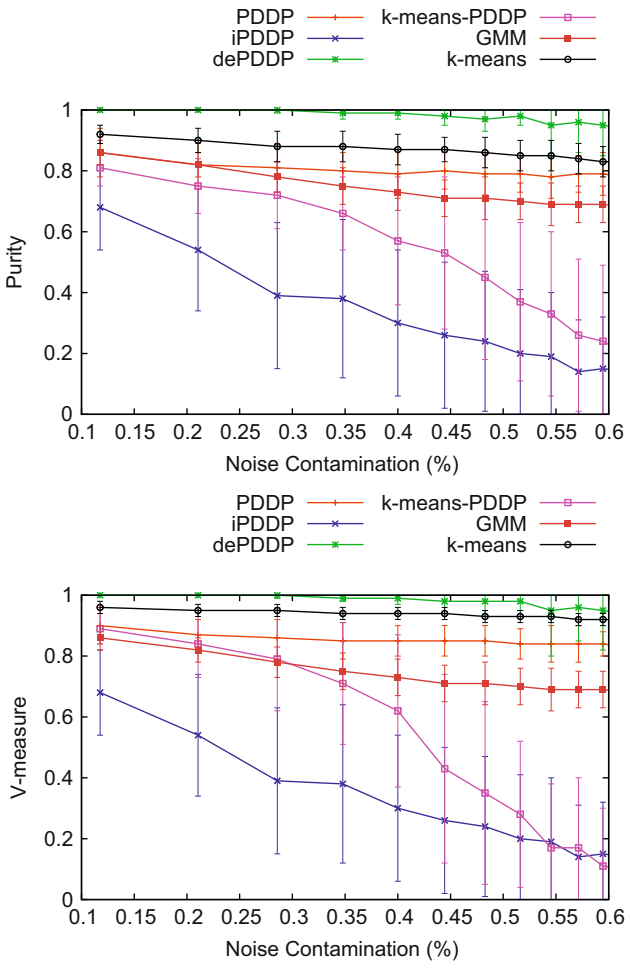

Fig. 9. The performance of the algorithms, when the noise contamination increases with respect to cluster purity (top) and the V-measure (bottom).

purity over 100 experiments is plotted. The vertical lines designate the standard deviation in each experiment. It is evident that the performance of both iPDDP and KM-PDDP heavily deteriorates with the increase of noise contamination, while the rest algorithms are only gradually affected. dePDDP is the least affected algorithm and only when more than 50% of the points in the data correspond to noise.

5.2. Automatic cluster number determination

As described above, the iPDDP and dePDDP algorithms can automatically determine the number of clusters in a data set. To measure the efficiency of the algorithms, we resort to a similar experimental procedure to the one previously described. 100 datasets are artificially constructed using the $DSET_{\text{Gaussian}}$ data generation mechanism.

In this case, the use of PDDP and KM-PDDP algorithms is not possible, since these algorithms do not have an automated stopping criterion. On the other hand, we additionally include two very popular algorithms: the DBSCAN algorithm [6] as a representative of the density-based class and the *x*-means algorithm [32] as a representative of the partitioning class. Regarding the GMM algorithm, we executed the algorithm for a varying number of clusters (up to a maximum k_{max}) and then selected the best amongst them using three model selection criteria, namely BIC, AIC and ICL [30]. Moreover, we also tested the PG-means algorithm [31], as a more sophisticated GMM approach.

Similarly to the previous experimental settings, a number of different dimensions and number of clusters were tested. The *MinPts* parameter for iPDDP was set to 5 and k_{max} was set twice the actual number of clusters across all experiments. The bandwidth parameter of dePDDP was set to the optimal one given by Eq. (5). For DBSCAN algorithm, the *MinPts* was set to 5 and *Eps* to 5*a*, where *a* is the data dimensionality.

The results from this experimental setting are exhibited in Table 6, with respect to cluster purity and V-measure, respectively. Furthermore, Table 7 reports the number of retrieved clusters for each algorithm.

The iPDDP algorithm in this case seems very efficient, especially as the dimensionality grows and the clusters are more clearly separated. It produces very good results with respect to both the purity and the V-measure, and gives almost perfect estimations for the number of clusters in high dimensional cases. Similarly, dePDDP performs very well in all dimensions examined and always produces accurate estimations for the number of clusters. DBSCAN manages to correctly retrieve the number of clusters, with very pure partitions. However, note that DBSCAN identifies many points as outliers including them in one big cluster and that this cluster is removed for the calculation of the purity and the V-measure (i.e. not shown in the tables). GMM-BIC and GMM-ICL perform similarly, producing good partitions in low dimensions, but their performance degrades as the dimensionality increases. The same holds for the number of retrieved clusters. As dimensionality increases the number of clusters is underestimated. GMM-AIC performs better with respect to both cluster purity and V-measure. In low dimensions it retrieves almost accurate cluster numbers, but in high dimensions it greatly over-estimates the number of clusters. Out of all the mixture modelling approaches PG-means managed to produce the best results, very close to the ones achieved by dePDDP and iPDDP. The problem with this approach is that it cannot handle the 50 dimensional cases, resulting in the majority of runs in trivial partitions of one cluster. For this reason we have excluded it from the particular part of Tables 6 and 7. Finally, the *x*-means algorithm exhibits very poor performance across all experiments.

Next, we repeat the same experimental procedure, but with noise contaminated data as before, by including 1000 noise points. The results are shown in Table 8 with respect to cluster purity and V-measure. Similarly, Table 9 reports the number of retrieved clusters for each algorithm. In this case, the results of the iPDDP algorithm are heavily affected by the inclusion of noise, except for the high dimensional cases. In low dimensions, the number of retrieved clusters is very small due the chaining effect. As expected, dePDDP is not affected by the inclusion of noise and again produces high quality results with respect to the cluster purity and V-measure, while simultaneously producing accurate estimations for the number of clusters. The same is true for the DBSCAN algorithm. However, the DBSCAN algorithm is also influenced by the chaining effect in the two-dimensional case. The GMM-BIC and GMM-ICL are also affected by the noise points and they produce bad results in all cases (especially in high dimensions). On the other hand, GMM-AIC can retrieve good results only in high dimensions. It seems that the inclusion of noise improved the performance of GMM-AIC in high dimensions with respect to the number of retrieved clusters. The PG-means algorithm in this case finds it more difficult to approximate the number of clusters, underestimating their number significantly. This translates to lower values for both cluster purity and V-measure but it still outperforms all the GMM alternatives. The results of the *x*-means algorithm remain pretty bad.

Table 6
Mean purity and V-measure (with the observed standard deviation in parenthesis), for $DSET_{\text{Gaussian}}$ generated data of different algorithms, with automated cluster determination over 100 experiments.

	15 Clusters		25 Clusters		50 Clusters	
	Pur.	V-meas.	Pur.	V-meas.	Pur.	V-meas.
Dimension 2						
iPDDP	0.88(0.05)	0.92(0.04)	0.84(0.04)	0.86(0.10)	0.75(0.05)	0.65(0.15)
dePDDP	0.94(0.04)	0.95(0.02)	0.92(0.03)	0.93(0.02)	0.84(0.04)	0.89(0.02)
GMM-BIC	0.88(0.04)	0.93(0.02)	0.72(0.13)	0.87(0.06)	0.18(0.09)	0.58(0.15)
GMM-AIC	0.89(0.07)	0.93(0.03)	0.68(0.22)	0.85(0.10)	0.24(0.10)	0.63(0.13)
GMM-ICL	0.85(0.10)	0.92(0.04)	0.72(0.15)	0.87(0.07)	0.25(0.13)	0.64(0.12)
PG-means	0.94(0.02)	0.95(0.01)	0.92(0.02)	0.93(0.02)	0.82(0.03)	0.90(0.01)
x-means	0.20(0.10)	0.44(0.09)	0.10(0.04)	0.35(0.06)	0.04(0.01)	0.29(0.03)
DBSCAN	0.80(0.07)	0.88(0.05)	0.73(0.06)	0.84(0.04)	0.56(0.09)	0.63(0.07)
Dimension 5						
iPDDP	1.00(0.01)	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.95(0.05)	0.79(0.27)
dePDDP	1.00(0.00)	0.99(0.01)	1.00(0.00)	0.99(0.00)	1.00(0.00)	0.99(0.00)
GMM-BIC	0.99(0.01)	0.98(0.01)	0.98(0.01)	0.98(0.00)	0.94(0.06)	0.97(0.01)
GMM-AIC	0.97(0.03)	0.95(0.01)	0.97(0.01)	0.96(0.01)	0.89(0.21)	0.92(0.18)
GMM-ICL	0.98(0.02)	0.98(0.01)	0.96(0.07)	0.98(0.02)	0.95(0.02)	0.97(0.00)
PG-means	1.00(0.00)	1.00(0.01)	1.00(0.01)	1.00(0.01)	0.95(0.02)	0.98(0.01)
x-means	0.28(0.15)	0.44(0.09)	0.10(0.04)	0.36(0.05)	0.04(0.00)	0.29(0.01)
DBSCAN	1.00(0.00)	1.00(0.00)	1.00(0.01)	1.00(0.00)	1.00(0.00)	1.00(0.00)
Dimension 20						
iPDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)
dePDDP	1.00(0.00)	1.00(0.01)	1.00(0.00)	0.99(0.00)	0.99(0.00)	0.99(0.00)
GMM-BIC	0.80(0.07)	0.92(0.02)	0.74(0.09)	0.92(0.02)	0.73(0.00)	0.93(0.00)
GMM-AIC	0.98(0.01)	0.92(0.01)	0.98(0.01)	0.93(0.01)	0.96(0.00)	0.96(0.00)
GMM-ICL	0.83(0.07)	0.93(0.02)	0.77(0.09)	0.93(0.02)	0.66(0.00)	0.92(0.00)
PG-means	0.99(0.02)	0.99(0.01)	0.97(0.02)	0.99(0.01)	0.98(0.01)	0.99(0.00)
x-means	0.32(0.07)	0.61(0.09)	0.16(0.05)	0.44(0.09)	0.05(0.02)	0.32(0.05)
DBSCAN	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)
Dimension 50						
iPDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)
dePDDP	1.00(0.00)	1.00(0.01)	1.00(0.00)	0.99(0.00)	1.00(0.00)	1.00(0.00)
GMM-BIC	0.31(0.06)	0.69(0.03)	0.28(0.03)	0.74(0.02)	0.27(0.02)	0.78(0.01)
GMM-AIC	0.98(0.02)	0.88(0.02)	0.98(0.01)	0.90(0.01)	0.96(0.01)	0.91(0.01)
GMM-ICL	0.31(0.04)	0.70(0.04)	0.29(0.03)	0.74(0.02)	0.30(0.03)	0.80(0.01)
x-means	0.34(0.09)	0.64(0.04)	0.20(0.07)	0.57(0.03)	0.09(0.03)	0.46(0.08)
DBSCAN	1.00(0.00)	0.98(0.01)	1.00(0.00)	0.98(0.01)	1.00(0.00)	0.99(0.00)

6. Selection of the k_{max} and *MinPts* parameters of the iPDDP algorithm

As we have seen, the iPDDP algorithm can be used to automatically determine the number of clusters in the dataset. To achieve this we need to force the algorithm to find more than the actual clusters, by selecting an appropriate value for the k_{max} parameter. However, in real life problems, the number of the actual clusters is not know and in effect the selection of the k_{max} parameter is not straightforward.

To examine the effect of this parameter on the performance of the proposed iPDDP algorithm, we employ datasets constructed as in Section 5, using $DSET_{\text{Gaussian}}$ with and without noise. In both cases, the datasets contain 25 clusters in 5 dimensions, while the noisy datasets contain additionally 1000 uniformly distributed random points to represent noise.

The results with respect to the number of retrieved clusters, as well as the purity and V-measure are presented in Fig. 10 (left) and (right), respectively. In the noiseless case, the algorithm consistently results in high values for both the purity and the V-measure. However, when k_{max} obtains very high values (higher than 300) the number of clusters are gradually underestimated. This occurs because when the value of k_{max} is too high the algorithm splits even the actual clusters in very small sets, incorrectly identifying them as outliers. So, in general, as along as we do not set extremely high values to k_{max}, the algorithm is expected to result in good partitions.

The presence of noise in the dataset makes the problem much more difficult. This is expected as the iPDDP algorithm has failed to result in good partitions in noisy cases (see results in Section 5.2). The algorithm manages to achieve high values for the purity and V-measure only when k_{max} is set high enough (over 500). But setting k_{max} to a very large value is not a solution. As shown, there exists a range of k_{max} values (i.e. 500–800) where the algorithm is able to retrieve the correct number of clusters. That interval is connected to the number of noisy points in the data, which of course in real life problems is not known beforehand. As such, it is obvious that in a noisy dataset the determination of appropriate k_{max} values can be problematic and, as shown in Section 5.2, the algorithm is expected to exhibit inferior performance.

To examine the effect of the *MinPts* parameter on the performance of the iPDDP algorithm we employ the same noiseless dataset used above. Now the k_{max} parameter is set to two times the actual number of clusters. In Fig. 11 (left) and (right), we present the results with respect to the number of retrieved clusters, and the purity and V-measure, respectively, for several values of the *MinPts* parameter. As shown, the performance of the algorithm is not affected, as long as, the *MinPts* parameter has value smaller than the size of the clusters. Thus, in the case that the user cannot make an informative

Table 7
Mean number of retrieved clusters (with the observed standard deviation in parenthesis), for $DSET_{\text{Gaussian}}$ generated data of different algorithms, over 100 experiments.

	15 Clusters	25 Clusters	50 Clusters
Dimension 2			
iPDDP	12.25(1.48)	16.55(2.91)	18.10 (3.63)
dePDDP	15.10(1.41)	25.45(2.84)	46.70 (5.26)
GMM-BIC	13.15(1.39)	15.65(3.80)	7.50 (3.64)
GMM-AIC	14.30(2.61)	15.40(5.53)	9.00 (3.49)
GMM-ICL	12.30(2.39)	15.95(4.42)	9.35 (4.36)
PG-means	14.00(1.10)	26.20(2.48)	37.80(4.07)
x-means	2.43(0.72)	2.13(0.34)	2.03(0.18)
DBSCAN	9.93(1.41)	13.03(1.72)	10.90(2.20)
Dimension 5			
iPDDP	14.95(0.22)	25.05(0.38)	34.75(16.15)
dePDDP	15.80(1.08)	26.65(1.01)	56.44(2.96)
GMM-BIC	18.05(1.56)	30.55(2.31)	61.25(10.19)
GMM-AIC	24.40(4.13)	45.30(4.12)	73.44(25.53)
GMM-ICL	17.95(2.01)	30.50(4.57)	68.00(6.50)
PG-means	15.60(0.80)	25.20(0.40)	45.00(1.79)
x-means	2.67(0.54)	2.17(0.45)	2.00(0.00)
DBSCAN	15.00(0.00)	24.83(0.37)	49.63(0.48)
Dimension 20			
iPDDP	15.05(0.22)	25.00(0.00)	50.00(0.00)
dePDDP	15.65(0.73)	26.80(1.50)	56.00(0.00)
GMM-BIC	10.90(1.14)	16.70(2.17)	33.00(0.00)
GMM-AIC	28.95(1.86)	48.15(5.12)	63.00(0.00)
GMM-ICL	11.35(1.28)	17.65(2.35)	31.00(0.00)
PG-means	14.80(0.40)	23.80(0.75)	48.20(0.75)
x-means	3.77(0.42)	2.97(0.66)	2.27(0.44)
DBSCAN	15.30(0.59)	25.30(0.46)	50.57(0.76)
Dimension 50			
iPDDP	15.15(0.36)	25.05(0.22)	50.00(0.00)
dePDDP	15.70(0.95)	26.60(1.16)	54.50(0.50)
GMM-BIC	4.30(0.56)	6.70(0.64)	12.50(0.50)
GMM-AIC	30.75(0.54)	50.60(0.66)	101.00(0.00)
GMM-ICL	4.50(0.59)	6.80(0.60)	14.00(1.00)
x-means	4.00(0.00)	4.00(0.00)	3.60(0.55)
DBSCAN	16.23(1.99)	28.33(3.67)	53.40(4.84)

selection of the value of *MinPts*, setting it to a small number is a sensible choice. The results for the noisy dataset are similar and are excluded for brevity.

7. Bandwidth selection for the dePDDP algorithm

As already discussed, the dePDDP algorithm relies on the density of the projected points to guide the clustering procedure. This density is approximated though a non-parametric kernel density estimation mechanism, a standard technique in exploratory data analysis, for which there is still a big dispute on how to assess the quality of the estimate and which choice of bandwidth is optimal [24]. Throughout the experiments in the paper we have set the bandwidth parameter to the value h_{opt} given by Eq. (5). In this section, we experimentally examine how different bandwidth selection schemes would affect the performance of dePDDP algorithm.

First we would like to investigate how an "improper" bandwidth would affect the performance of the algorithm. As it is not trivial to specify what an improper bandwidth is, since even that is data dependent, we are going to use a wide range of multiples of the h_{opt} value and examine the clustering performance of each one.

For this reason, we employ the $DSET_{\text{Gaussian}}$ data generation mechanism and for various values of the bandwidth multiplier we examine obtained purity and V-measures. Further, to examine the effect with respect to different dimensions and number of clusters, we use datasets of 2 and 15 dimensions, with 15 and 25 clusters. The results are illustrated in Fig. 12. It is evident that there exists a wide range of multiplier values between 1.5 and 3 that the dePDDP algorithm exhibits very good results with respect both to purity and V-measure. However, for small values of the multiplier, the results show high purity, the V-measure is much smaller as it quite possible that many actual clusters are split. For large values of the multiplier both the purity and V-measure obtain small values, indicative of bad partitions. Note also that the results are similar for all of the combinations of dimensions and clusters examined, shows that the bandwidth selection is independent of the data dimensionality or the number of clusters in the data. In conclusion, we can see that any value of the multiplier less than 2 will give good results. Large bandwidths should be avoided, since the algorithm is expected to perform poorly. On the other hand, small bandwidths are not such a big concern, since the algorithm is expected to return an increased number of (pure) clusters.

8. Running time analysis

In this section we study the running time of the proposed methods. We compare them against the running time of the original PDDP algorithm and the time required by iterative algorithms. The iterative algorithms used are the standard *k*-means (where the correct number of clusters is given a priori to the algorithm) and the GMM with the combination of the AIC to automatically estimate the number of clusters. Note that GMM is executed for a varying number of clusters up to a maximum k_{max}, where k_{max} was set twice the actual number of clusters and that the AIC is used to select amongst them. For the dePDDP we are using the fast Gauss transform [40] to estimate the density of the projected points.

In detail, we are going to test the running time of the different algorithms initially for an increasing number of points and next for an increasing dimensionality. In the first case we use datasets generated from a mixture of 10 Gaussian 10 dimensional distributions that represent the actual clusters in the data and iteratively increase the data set size by sampling an increasing number of points from each Gaussian distribution. The results are illustrated in Fig. 13 (left) (notice the log-scale of the *y* axis). For the second experiment, we again use a mixture of 10 Gaussian distributions, where the dimensionality of the distributions is iteratively increased. The results are illustrated in Fig. 13 (right). The running time of each algorithm is measured in seconds and the CPU used is an Intel(R) Xeon(R) E5405, operating at 2.00 GHz.

As shown the *k*-means algorithm is the fastest algorithm with respect to the number of points in the data. PDDP, iPDDP and dePDDP have similar performance and they seem quite competitive to *k*-means. GMM-AIC on the other hand needs many orders of magnitude large running times. Note that all the algorithms show a similar increasing trend. The results for the increasing dimensionality case are reversed. In this case PDDP turns out to be the fastest algorithm, closely followed by iPDDP and dePDDP. The *k*-means algorithm in this case is heavily affected by the increasing dimensionality, resulting in very large running times. The running time of GMM was enormous and after 200 dimensions the algorithm exhibited numerical stability problems. Since it was unable to converge, it was not included in the plot. In general, we observe that the performance of the proposed methods are quite competitive even to the simplest of iterative methods and show very good performance when the dimensionality is increasing.

Table 8
Mean purity and V-measure (with the observed standard deviation in parenthesis), for $DSET_{Gaussian}$ generated data contaminated with 1000 noise points of different algorithms with automated number of clusters determination, over 100 experiments.

	15 Clusters		25 Clusters		50 Clusters	
	Pur.	V-meas.	Pur.	V-meas.	Pur.	V-meas.
Dimension 2						
iPDDP	0.19(0.20)	0.05(0.09)	0.28(0.23)	0.12(0.10)	0.55(0.21)	0.19(0.11)
dePDDP	0.89(0.07)	0.92(0.05)	0.90(0.05)	0.91(0.05)	0.84(0.03)	0.89(0.02)
GMM-BIC	0.42(0.10)	0.66(0.11)	0.32(0.12)	0.60(0.13)	0.15(0.06)	0.50(0.13)
GMM-AIC	0.41(0.15)	0.64(0.13)	0.27(0.08)	0.55(0.10)	0.13(0.05)	0.45(0.15)
GMM-ICL	0.34(0.13)	0.59(0.16)	0.26(0.11)	0.51(0.17)	0.12(0.06)	0.41(0.13)
PG-means	0.83(0.07)	0.91(0.03)	0.82(0.04)	0.90(0.01)	0.71(0.05)	0.87(0.02)
x-means	0.20(0.10)	0.44(0.09)	0.10(0.04)	0.35(0.06)	0.04(0.01)	0.29(0.03)
DBSCAN	0.80(0.07)	0.88(0.05)	0.73(0.06)	0.84(0.04)	0.56(0.09)	0.63(0.07)
Dimension 5						
iPDDP	0.54(0.15)	0.21(0.10)	0.67(0.11)	0.26(0.10)	0.70(0.03)	0.16(0.14)
dePDDP	0.99(0.02)	0.99(0.01)	1.00(0.00)	1.00(0.00)	1.00(0.01)	0.99(0.00)
GMM-BIC	0.63(0.10)	0.85(0.03)	0.55(0.09)	0.84(0.04)	0.34(0.20)	0.63(0.29)
GMM-AIC	0.67(0.06)	0.85(0.03)	0.58(0.07)	0.85(0.02)	0.54(0.05)	0.87(0.01)
GMM-ICL	0.66(0.11)	0.85(0.04)	0.54(0.07)	0.84(0.03)	0.42(0.04)	0.83(0.01)
PG-means	0.76(0.20)	0.89(0.09)	0.78(0.12)	0.91(0.04)	0.77(0.12)	0.92(0.05)
x-means	0.28(0.15)	0.44(0.09)	0.10(0.04)	0.36(0.05)	0.04(0.00)	0.29(0.01)
DBSCAN	1.00(0.00)	1.00(0.00)	1.00(0.01)	1.00(0.00)	1.00(0.00)	1.00(0.00)
Dimension 20						
iPDDP	0.87(0.04)	0.70(0.10)	0.89(0.04)	0.58(0.14)	0.89(0.05)	0.27(0.15)
dePDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)
GMM-BIC	0.63(0.12)	0.85(0.04)	0.63(0.08)	0.88(0.02)	0.57(0.16)	0.89(0.03)
GMM-AIC	0.82(0.12)	0.92(0.04)	0.84(0.12)	0.94(0.04)	0.95(0.02)	0.97(0.00)
GMM-ICL	0.64(0.08)	0.86(0.03)	0.65(0.07)	0.88(0.02)	0.67(0.05)	0.90(0.01)
PG-means	0.85(0.16)	0.73(0.29)	0.93(0.02)	0.83(0.10)	0.90(0.03)	0.76(0.04)
x-means	0.32(0.07)	0.61(0.09)	0.16(0.05)	0.44(0.09)	0.05(0.02)	0.32(0.05)
DBSCAN	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)	1.00(0.00)
Dimension 50						
iPDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.76(0.24)	0.98(0.02)	0.76(0.24)
dePDDP	1.00(0.00)	1.00(0.00)	1.00(0.00)	0.99(0.00)	1.00(0.00)	0.99(0.00)
GMM-BIC	0.26(0.06)	0.62(0.09)	0.26(0.06)	0.74(0.03)	0.24(0.04)	0.74(0.03)
GMM-AIC	0.74(0.14)	0.86(0.06)	0.74(0.14)	0.87(0.09)	0.69(0.25)	0.87(0.09)
GMM-ICL	0.29(0.06)	0.65(0.05)	0.29(0.06)	0.77(0.01)	0.31(0.01)	0.77(0.01)
x-means	0.34(0.09)	0.64(0.04)	0.20(0.07)	0.57(0.03)	0.09(0.03)	0.46(0.08)
DBSCAN	1.00(0.00)	0.98(0.01)	1.00(0.00)	0.98(0.01)	1.00(0.00)	0.99(0.00)

9. Benchmark datasets

In this section we test the performance of the algorithms on benchmark datasets from the UCI machine learning repository [45]. These datasets have been used for the evaluation of a number of similar algorithms [46–49], so easy comparison to other approaches is facilitated. In particular we use the following datasets:

- (VOTES): This data set includes votes for each of the US House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise made a position known (these three simplified to an unknown disposition). Thus, the data are described by 16 attributes. The total number of objects is 435; 267 of which are labelled as democrats, while the remaining 168 as republicans.
- (BREAST-CANCER): This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. There are 369 instances in this datasets, described though 10 features. Each instance has one of two possible classes: benign or malignant. There are 16 instances that contain a single missing (i.e. unavailable) attribute value that we arbitrary set to 0.
- (SYNTHETIC CONTROL): This data consists of 600 examples of control charts synthetically generated by the process in [50]. There are six different classes of control charts: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift.

The results in terms of cluster purity and V-measured are shown in Table 10. Here, the PDDP algorithm manages to achieve very good results for the BREAST-CANCER dataset, but its performance deteriorates in the other two datasets. On the other hand, the iPDDP algorithm seems able to provide high purity partitions, but the associated V-measures are very low, indicating that some of the actual classes in the data are not split at all. This translates to possible chaining effects, since this is the main reason for the iPDDP failure, as shown by the experimental analysis on simulated data. The KM-PDDP algorithm, in the case of the SYNTHETIC CONTROL datasets, fails to provide good results in terms of V-measure. Possible reasons for this is that this dataset has uneven classes that the algorithm fails to capture. The GMM algorithm seems to have difficulties to deal with the VOTES dataset, but manages to achieve good results in the rest

two cases. On the other hand, *k*-means provides very good results in all cases supporting its popularity in such low dimensional data. PG-means provides good results amongst all GMM based approaches and competitively in general. Finally, the dePDDP algorithm achieves highly pure partitions in all tested datasets. It is of equal importance that in the considered problems, the dePDDP algorithm never underestimates the number of clusters, providing approximations that in two out of the three cases are very close to the actual number of classes, as well. Notice that the number of clusters is given as input to the rest of the clustering algorithms.

Table 9
Mean number of retrieved clusters (with the observed standard deviation in parenthesis), for $DSET_{Gaussian}$ generated data contaminated with 1000 noise points of different algorithms, over 100 experiments.

	15 Clusters	25 Clusters	50 Clusters
Dimension 2			
iPDDP	1.30(0.46)	1.90(0.77)	4.12(1.45)
dePDDP	13.25(2.84)	23.90(3.74)	45.50(5.41)
GMM-BIC	5.95(1.88)	6.45(2.73)	5.38(1.93)
GMM-AIC	5.95(2.62)	5.20(1.81)	4.62(1.87)
GMM-ICL	5.00(2.37)	4.95(2.71)	4.00(1.87)
PG-means	14.60(2.87)	20.80(1.33)	32.40(2.58)
x-means	2.00(0.00)	2.00(0.00)	2.00(0.00)
DBSCAN	35.20(3.25)	28.60(3.76)	12.05(4.15)
Dimension 5			
iPDDP	2.30(0.56)	3.65(1.01)	4.33(1.89)
dePDDP	14.90(0.89)	25.85(0.96)	57.00(0.00)
GMM-BIC	8.90(1.55)	12.95(2.18)	15.00(9.63)
GMM-AIC	9.10(0.94)	13.75(1.73)	25.00(1.41)
GMM-ICL	9.00(1.38)	12.60(1.50)	20.00(2.16)
PG-means	11.00(2.90)	17.60(3.44)	33.60(7.81)
x-means	2.00(0.00)	2.00(0.00)	2.00(0.00)
DBSCAN	15.00(0.00)	24.95(0.22)	49.50(0.67)
Dimension 20			
iPDDP	7.50(1.53)	9.55(2.73)	9.00 (4.00)
dePDDP	15.20(0.40)	26.10(1.30)	55.50 (1.50)
GMM-BIC	8.10(1.45)	13.70(1.76)	25.50 (6.50)
GMM-AIC	11.70(2.12)	21.75(5.31)	63.50 (0.50)
GMM-ICL	8.30(0.90)	14.00(1.61)	28.50 (1.50)
PG-means	9.00(3.74)	16.20(3.54)	23.60(2.42)
x-means	2.65(0.57)	2.35(0.48)	2.00(0.00)
DBSCAN	15.20(0.51)	25.20(0.40)	50.45(0.74)
Dimension 50			
iPDDP	15.00(0.00)	24.85(0.91)	34.00 (5.00)
dePDDP	15.60(0.80)	26.65(1.31)	54.00 (1.00)
GMM-BIC	3.65(0.91)	5.75(1.04)	10.50 (1.50)
GMM-AIC	10.00(2.37)	21.60(1.91)	35.50 (1.50)
GMM-ICL	3.90(0.54)	6.20(1.29)	12.50 (0.50)
x-means	3.90(0.30)	3.50(0.59)	2.95(0.59)
DBSCAN	17.15(2.67)	27.85(3.48)	53.85(4.25)

10. Microarray experiments

The development of microarray technologies gives scientists the ability to examine, discover and monitor the mRNA transcript levels of thousands of genes in a single experiment. Discovering the patterns hidden in gene expression microarray data is a tremendous opportunity and challenge for functional genomics and proteomics. A promising approach to address this task is to utilise data mining techniques. Cluster analysis is a key step in understanding how the activity of genes varies during biological processes and is affected by disease states and cellular environments. In particular, clustering can be used either to identify sets of genes according to their expression in a set of samples, or to cluster samples into homogeneous groups that may correspond to particular macroscopic phenotypes. The latter is in general more difficult because of the high dimensionality that describes these datasets. For this reason they make a perfect candidate to explore the performance of the algorithms proposed in this paper. As such we employ the following microarray datasets:

- The COLON data set [51] consists of 40 tumour and 22 normal colon tissues. For each sample there exist 2000 gene expression level measurements. The data set is available at: http://microarray.princeton.edu/oncology.
- The LYMPHOMA dataset [52] that contains 62 samples of the 3 lymphoid malignancies samples types. The samples are measured over 4026 gene expression levels. This dataset is available at: http://genome-www.stanford.edu/.
- The PROSTATE data set [53] that contains 52 prostate tumour samples and 50 non-tumour prostate samples. For each sample there exist 6033 gene expression level measurements. It is available at: http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi.
- The ALL data set [54] that contains 248 samples of 6 subtypes of acute lymphoblastic leukemia. There exist 12 559 gene expression level measurements per sample. It is available at: http://www.stjuderesearch.org/data/ALL1.

In Table 11, the clustering results with respect to purity and V-measure of the algorithms for each dataset are illustrated. Note that the *k*-means and GMM algorithms were not applied here, due to the ultra high dimensionality of the data. Moreover, the actual number of clusters in the data is given as input to the PDDP, the iPDDP, and the KM-PDDP algorithms.

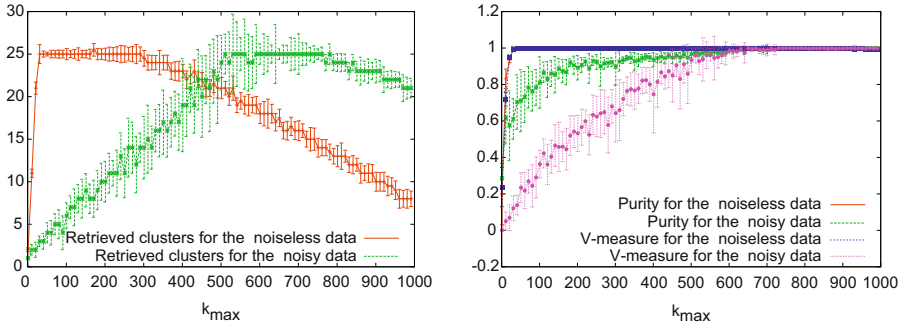

Fig. 10. The performance of the iPDDP algorithm with respect to the number of retrieved clusters (left) and the purity and V-measure values (right) for a range of values for the k_{max} parameter.

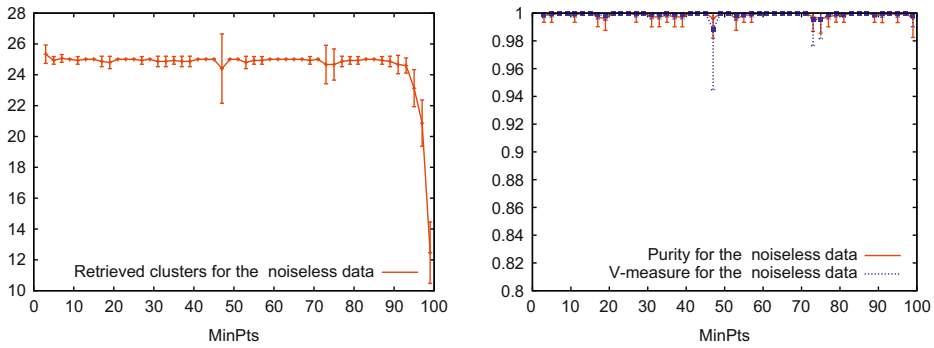

Fig. 11. The performance of the iPDDP algorithm with respect to the number of retrieved clusters (left) and the purity and V-measure values (right) for a range of values for the MinPts parameter.

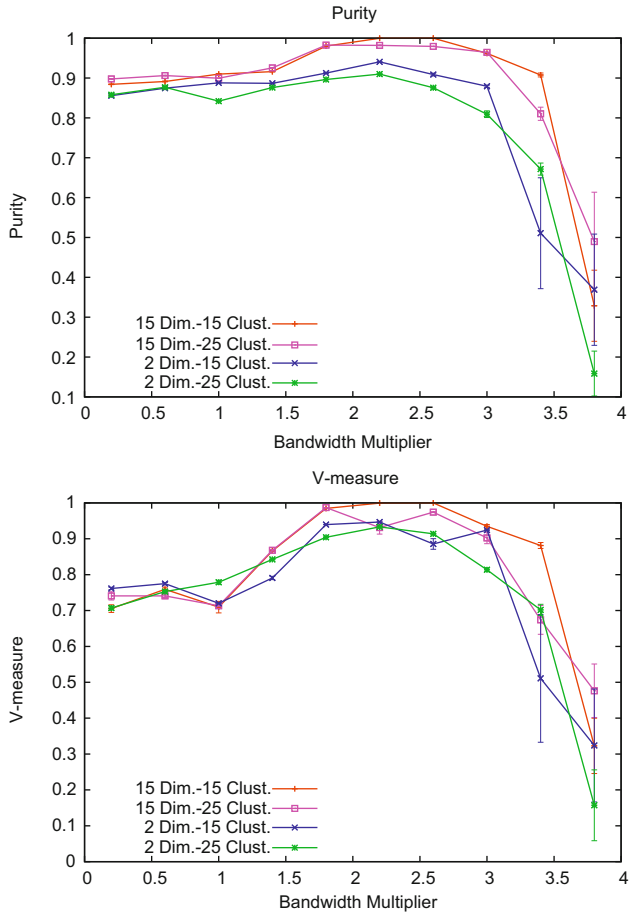

Fig. 12. The performance of the dePDDP algorithm with respect to cluster purity (top) and the V-measure (bottom), when the bandwidth h_{opt} is multiplied by a range of different values.

As shown by the values of the V-measures in Table 11, the PDDP, the iPDDP and the KM-PDDP algorithms manage to identify good partitions only in the case of the LYMPHOMA dataset. In particular iPDDP results in a V-measure of 1, which translates to clusters with direct correspondence to the class labels of the data. The dePDDP algorithm results in good partitions in all of the cases.

11. Text mining

Text mining was the initial motivation behind the development of the original PDDP algorithm. To this end, to examine the performance of the proposed approaches in text datasets we use a subset of the original TDT2 corpus. The TDT2 corpus (Nist Topic Detection and Tracking corpus) consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW and NYT), 2 radio programs (VOA and PRI) and 2 television programs (CNN and ABC). In total it consists of 11 201 on-topic documents, which are classified into 96 semantic categories. Here, we used 4 samples from the initial corpora, containing 2, 3, 4, and 5 categories, respectively. For each sample, 50 randomly generated sub-samples were used, provided in http://www.cs.uiuc.edu/homes/dengcai2/Data/TextData.html. The results are illustrated in Fig. 14. Each boxplot depicts the obtained values for the clustering purity, in each of the random sub-samples. The box depicts the interquantile range of the data and contains a bold line at the data median. The lines extending from each end of the box indicate the range covered by the remaining data.

As the plots indicate, the PDDP algorithm is quite effective in recognising the structure of the data, especially in the 2 and 3 class cases. In those particular cases, the efficiency the iPDDP and dePDDP is always on par and marginally improved. The 4 class and 5 class case is a bit different. Here, the PDDP accuracy has a median value around 70%, which points out to the fact that the data are quite complex so deriving the class structure is quite difficult. For these cases, the iPDDP algorithm produces results with reduced partitioning purity. The KM-PDDP algorithm provides the worst results on all cases.

12. Concluding remarks

Clustering is a subjective problem in its nature [22]. Although it sounds easy to find similar objects in a database, the similarity in itself is only in the eyes of the beholder. Additionally, the introduction of high dimensionality datasets posed new problems to the data analyst. Surprisingly however, data clustering algorithms work and they provide meaningful results for a variety of problems ranging from text mining to microarray data analysis [20,55].

To tackle the challenging new problems, a new class of algorithms has been developed. These algorithms operate on the projection of the data on a few principal components of the covariance matrix and have the ability to discover clusters effectively [43]. In this work, we try to deepen our understanding on what can be achieved by these approaches. We first make assumptions on the nature of the true clusters in the data ("inductive bias") and then attempt to theoretically discover the relationship between the true clusters and the distribution of their projection onto the principal components. Based on that, appropriate criteria for the various steps involved in hierarchical

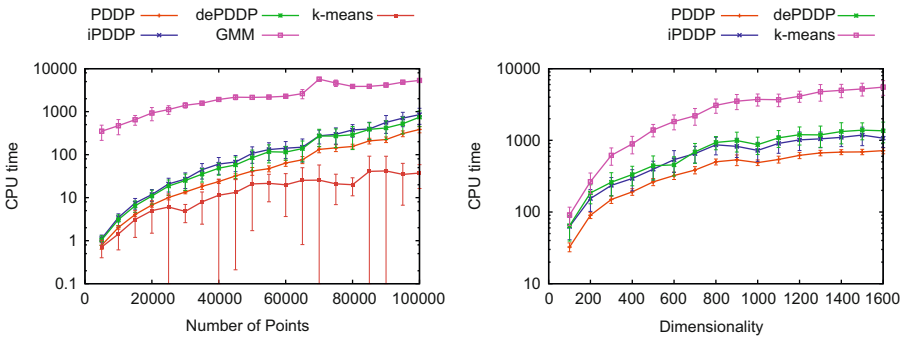

Fig. 13. The CPU time cost of the algorithms as the size of the dataset grows (left). The CPU time cost of the algorithms as the dimensionality grows (right).

Table 10
Results with respect to the clustering purity and V-measure for the three UCI repository data sets.

Dataset Classes		BREAST-CAN. 2	VOTES 2	SYNTH. CONT. 6
PDDP	Clusters	2	2	6
	Pur.	0.9639	0.8432	0.5766
	V-meas.	0.8103	0.4097	0.5780
iPDDP	Clusters	2	2	6
	Pur.	0.8338	0.8065	0.7726
	V-meas.	0.0547	0.0033	0.7517
dePDDP	Clusters	11	3	11
	Pur.	0.9174	0.7837	0.9076
	V-meas.	0.7634	0.3784	0.7600
KM-PDDP	Clusters	2	2	6
	Pur.	0.9617	0.8489	0.8613
	V-meas.	0.7634	0.3944	0.0163
GMM	Clusters	2	2	6
	Pur.	0.8663	0.6252	0.7479
	V-meas.	0.5553	0.0086	0.5219
GMM-BIC	Clusters	5	3	2
	Pur.	0.9463	0.6565	0.5888
	V-meas.	0.4544	0.0539	0.1203
GMM-AIC	Clusters	5	2	13
	Pur.	0.8670	0.6302	0.8745
	V-meas.	0.4882	0.0151	0.6597
GMM-ICL	Clusters	5	4	2
	Pur.	0.9466	0.7123	0.3931
	V-meas.	0.4467	0.0368	0.3511
PG-means	Clusters	6	6	5
	Pur.	0.9472	0.8140	0.8154
	V-meas.	0.5034	0.3402	0.2698
k-means	Clusters	2	2	6
	Pur.	0.9571	0.8434	0.7637
	V-meas.	0.7361	0.4037	0.7013

Table 11
Results with respect to the clustering purity and V-measure for the microarray datasets.

Dataset Classes		COLON 2	LYMPHOMA 3	ALL 6	PROSTATE 2
PDDP	Clusters	2	3	6	2
	Pur.	0.6544	0.9394	0.5560	0.5788
	V-meas.	0.0301	0.8776	0.0405	0.0177
iPDDP	Clusters	2	3	6	2
	Pur.	0.5750	1.0000	0.6742	0.6713
	V-meas.	0.0037	1.0000	0.0238	0.0535
dePDDP	Clusters	6	3	7	6
	Pur.	0.8366	1.0000	0.7947	0.6981
	V-meas.	0.1932	1.0000	0.5386	0.0696
KM-PDDP	Clusters	2	3	6	2
	Pur.	0.8197	0.8500	0.7678	0.6713
	V-meas.	0.0195	0.6475	0.0228	0.0535

divisive clustering are proposed. At a next step, these criteria are combined into new algorithms and their effectiveness is investigated.

The proposed algorithms require minimal user-defined parameters and have the desirable feature of being able to provide approximations for the number of clusters present in a dataset. This in itself is an open problem in cluster analysis and little has been done in the past for the high dimensional data case that we are mostly concerned with here.

The included experimental results indicate that the proposed techniques are effective both in terms of partitioning the data and determining the true number of clusters in the dataset. Their performance is very good, even compared against the popular density based methods. Finally, to stress test the proposed methods a series of experiments on real world microarray and test mining datasets are included. In this case, density based techniques cannot be directly applied and complicated feature selection methods are required to actually get results [55]. However, the proposed algorithms exhibited promising results even in these hard and extremely interesting problems.

Promising future directions stemming from the presented research are two-fold. Firstly, we would like to theoretically investigate the case of non convex clusters. In such a situation we cannot resort to any linear data projection, but techniques like kernel principal component Analysis need to be employed. It would be interesting to examine if the results in this paper can be also extended using such methods. Secondly, although PCA is optimal in the sense of maximum variance, by no means is optimal or even appropriate for a clustering procedure. To this end, devising an algorithm that finds a projection to facilitate the needs of partitioning the data seems a promising area of research.

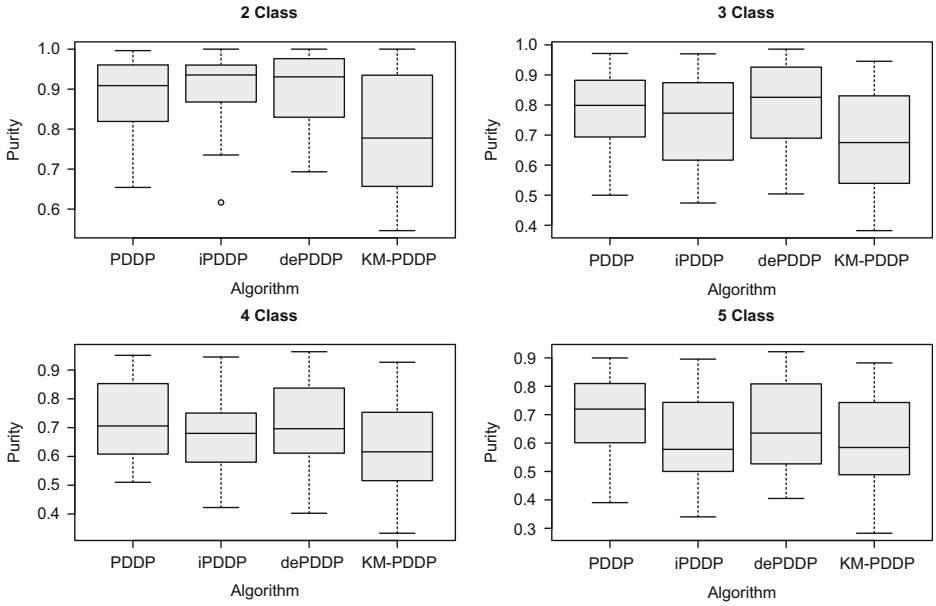

Fig. 14. The purity of clustering results for the TDT2 corpus subsets.

Acknowledgements

The authors would like to thank the Associate Editor and the two anonymous reviewers for their valuable comments and suggestions that helped to improve the content as well as the clarity of the manuscript. We also thank the authors of [31] for providing us the code of their method.

References

[1] C. Tryon, Cluster Analysis, Edward Brothers, Ann Arbor, MI, 1939.
[2] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, in: KDD Workshop on Text Mining, 2000.
[3] J. Hartigan, M. Wong, A *k*-means clustering algorithm, Applied Statistics 28 (1979) 100–108.
[4] I.S. Dhillon, D.S. Modha, Concept decompositions for large sparse text data using clustering, Machine Learning 42 (1) (2001) 143–175.
[5] G. McLachlan, K. Basford, Mixture Models: Inference and Applications to Clustering, Marcel Dekker, 1988.
[6] J. Sander, M. Ester, H.-P. Kriegel, X. Xu, Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications, Data Mining and Knowledge Discovery 2 (2) (1998) 169–194.
[7] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press, Princeton, NJ, 1961.
[8] M. Steinbach, L. Ertöz, V. Kumar, The challenges of clustering high dimensional data, New Vistas in Statistical Physics: Applications in Econophysics, Bioinformatics, and Pattern Recognition, 2003.
[9] K.S. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful, in: Seventh International Conference on Database Theory, 1999, pp. 217–235.
[10] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice Hall, New Jersey, 1988.
[11] P.D. Lax, Linear Algebra and its Applications, Wiley-Interscience, New York, USA, 2007.
[12] D. Boley, Principal direction divisive partitioning, Data Mining and Knowledge Discovery 2 (4) (1998) 325–344.
[13] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman, Indexing by latent semantic analysis, Journal of the American Society for Information Science 41 (6) (1990) 391–407.
[14] C. Chute, Y. Yang, An overview of statistical methods for the classification and retrieval of patient events, Methods of Information in Medicine 34 (1–2) (1995) 104–110.
[15] S. Tasoulis, D. Tasoulis, Improving principal direction divisive clustering, in: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), Workshop on Data Mining using Matrices and Tensors, Las Vegas, USA, 2008.
[16] D. Zeimpekis, E. Gallopoulos, PDDP(l): towards a flexing principal direction divisive partitioning clustering algorithms, in: D. Boley, I. Dhillon, J. Ghosh, J. Kogan (Eds.), Proceedings of the IEEE ICDM '03 Workshop on Clustering Large Data Sets, Melbourne, FL, 2003, pp. 26–35.
[17] I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge discovery and data mining, ACM New York, NY, USA, 2001, pp. 269–274.
[18] I. Dhillon, J. Kogan, C. Nicholas, Feature selection and document clustering, in: A Comprehensive Survey of Text Mining, 2003, pp. 73–100.
[19] M. Nilsson, Hierarchical clustering using non-greedy principal direction divisive partitioning, Information Retrieval 5 (4) (2002) 311–321.
[20] D. Zeimpekis, E. Gallopoulos, Principal direction divisive partitioning with kernels and k-means steering, in: Survey of Text Mining II: Clustering, Classification, and Retrieval, 2007, pp. 45–64.
[22] I. Guyon, U. von Luxburg, R. Williamson, Clustering: science or art? in: NIPS Workshop on Clustering Theory2009.
[23] M.P. Wand, M.C. Jones, Kernel Smoothing, Chapman & Hall/CRC, Florida, USA, 1995.
[24] B.A. Turlach, Bandwidth selection in kernel density estimation: a review, in: CORE and Institut de Statistique, 1993, pp. 23–493.
[25] P. Sarda, Smoothing parameter selection for smooth distribution functions, Journal of Statistical Planning and Inference 35 (1) (1993) 65–75.
[26] N. Altman, C. Léger, Bandwidth selection for kernel distribution function estimation, Journal of Statistical Planning and Inference 46 (2) (1995) 195–214.
[27] B.E. Hansen, Bandwidth selection for nonparametric distribution estimation, Ph.D. Thesis, University of Wisconsin, manuscript, 2004.
[28] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, San Diego, USA, 2006.
[29] G.H. Ball, D.J. Hall, A clustering technique for summarizing multivariate data, Behavioral Science 12 (1967) 153–155.
[30] C. Biernacki, G. Celeux, G. Govaert, Assessing a mixture model for clustering with the integrated completed likelihood, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (7) (2000) 719–725.
[31] Y. Feng, G. Hamerly, C. Elkan, PG-means: learning the number of clusters in data, Advances in Neural Information Processing Systems 19 (2007) 393.
[32] D. Pelleg, A. Moore, X-means: extending k-means with efficient estimation of the number of clusters, in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 727–734.
[33] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in: ACM SIGMOD International Conference on Management of Data, 1996, pp. 103–114.
[34] S. Guha, R. Rastogi, K. Shim, CURE: an efficient algorithm for clustering large databases, in: Proceedings of ACM-SIGMOD 1998 International Conference on Management of Data, Seattle, 1998, pp. 73–84.
[35] G. Karypis, E. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, IEEE Computer 32 (8) (1999) 68–75.
[36] A. Hinneburg, D. Keim, Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering, in: Proceedings of the 25th International Conference on Very Large Data Bases, 1999, pp. 506–517.
[37] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data, Data Mining and Knowledge Discovery 11 (1) (2005) 5–33.
[38] C. Kruengkrai, V. Sornlertlamvanich, H. Isahara, Refining a divisive partitioning algorithm for unsupervised clustering, in: Proceedings of the Third International Conference on Hybrid Intelligent Systems, 2003, pp. 535–542.

[39] D. Young, The linear nearest neighbour statistic, Biometrica 69 (2) (1982) 477–480.
[40] C. Yang, R. Duraiswami, N.A. Gumerov, L. Davis, Improved fast gauss transform and efficient kernel density estimation, in: Proceedings of Ninth IEEE International Conference on Computer Vision, 2003, pp. 664–671.
[41] L. Greengard, J. Strain, The fast gauss transform, SIAM Journal on Scientific and Statistical Computing 12 (1) (1991) 79–94.
[42] P. Berkhin, A survey of clustering data mining techniques, in: J. Kogan, C. Nicholas, M. Teboulle (Eds.), Grouping Multidimensional Data: Recent Advances in ClusteringSpringer, Berlin, 2006, pp. 25–72.
[43] J. Kogan, Introduction to Clustering Large and High-Dimensional Data, Cambridge University Press, New York, USA, 2007.
[44] A. Rosenberg, J. Hirschberg, V-measure: a conditional entropy-based external cluster evaluation measure, in: 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007, pp. 410–420.
[45] C. Blake, C. Merz, UCI repository of machine learning databases, University of California, School of Information and Computer Science, Irvine, CA, 1998.
[46] D.K. Tasoulis, M.N. Vrahatis, Generalizing the k-Windows clustering algorithm in metric spaces, Mathematical and Computer Modelling 46 (1–2) (2007) 268–277.
[47] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Mining and Knowledge Discovery 2 (3) (1998) 283–304.
[48] D.H. Fisher, Knowledge acquisition via incremental conceptual clustering, Machine Learning 2 (2) (1987) 139–172.
[49] R.S. Michalski, R.E. Stepp, Automated construction of classifications: conceptual clustering versus numerical taxonomy, IEEE Transactions on Pattern Analysis and Machine Intelligence 5 (4) (1983) 396–409.
[50] R.J. Alcock, Y. Manolopoulos, Time-series similarity queries employing a feature-based approach, in: Seventh Hellenic Conference on Informatics, Ioannina, Greece, 1999.
[51] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, A. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array, Proceedings of the National Academy of Sciences USA 96 (12) (1999) 6745–6750.
[52] A.A. Alizadeh, M.B. Eisen, R.E. Davis, C. Ma, I.S. Lossos, A. Rosenwald, J.C. Boldrick, H. Sabet, T. Tran, X. Yu, Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling, Nature 403 (2000) 503–511.
[53] D. Singh, P.G. Febbo, K. Ross, D.G. Jackson, J. Manola, C. Ladd, P. Tamayo, A.A. Renshaw, A.V. D'Amico, J.P. Richie, Gene expression correlates of clinical prostate cancer behavior, Cancer Cell 1 (2) (2002) 203–209.
[54] E.J. Yeoh, M.E. Ross, S.A. Shurtleff, W.K. Williams, D. Patel, R. Mahfouz, F.G. Behm, S.C. Raimondi, M.V. Relling, A. Patel, Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling, Cancer Cell 1 (2) (2002) 133–143.
[55] D. Tasoulis, V. Plagianakos, M. Vrahatis, Unsupervised clustering in mRNA expression profiles, Computers in Biology and Medicine 36 (2006) 1126–1142.

Sotirios K. Tasoulis received his B.Sc. from the Department of Mathematics, University of Patras, Greece and holds an M.Sc. Mathematics and Computer Science from the University of Patras Greece. He is currently a Ph.D. candidate in the Computer Science and Biomedical Informatics Department, at the University of Central Greece. His research interests are focused on data mining and pattern recognition.

Dimitris K. Tasoulis received his B.Sc., M.Sc., and Ph.D. degrees from the Department of Mathematics, University of Patras, Greece. He currently serves as a Lecturer at the Mathematics Department of Imperial College, London. His research is focused on various areas of data mining, machine learning, and pattern recognition. He has published more than 70 journal and conference papers, and he serves as a reviewer in numerous journals and conferences.

Vassilis P. Plagianakos received both his B.Sc. and Ph.D. degrees from the Department of Mathematics, University of Patras, Greece. He is currently an Assistant Professor at the Department of Computer Science and Biomedical Informatics, University of Central Greece. His research interests mainly focus on the areas of neural networks and machine learning, pattern recognition, intelligent decision making, evolutionary and genetic algorithms, clustering, parallel and distributed computations, intelligent optimization, bioinformatics, and real-world problem solving. He has published more than 70 journal and conference papers, and he serves as a reviewer in numerous journals and conferences.