Density and Non-Grid based Subspace Clustering via Kernel Density Estimation

Jing Zhang, Lantao Yu, Hanqi Zhu, and Gang Sun

Department of Information Management & Systems, Tianjin University,
Tianjin 300072, China
{zjing,yulantao1991,hqzhu,sungang}@tju.edu.cn

Abstract. Instead of finding clusters in full data space, subspace clustering has been an emergent task which aims at discovering clusters embedded in subspaces. Most of the previous works in the literature are grid and density based approaches, where the performance of clustering algorithms heavily depends on the partition of grids which may incur the serious loss of the real data distribution. In order to address the dilemma of grid partition, in this paper we propose a Density and Non-Grid based Subspace Clustering (DNGSC) algorithm via Kernel Density Estimation, which is able to discover arbitrarily shaped subspace clusters and insensitive to the only input parameter. The Kernel Density Estimation method with its firm mathematical basis has the ability to accurately uncover the underlying boundaries of data without any presumed canonical distribution. Besides, in order to deal with "the density divergence problem" which means the region densities vary in difference subspace cardinalities, different density thresholds for different subspaces are automatically determined to effectively discover arbitrarily shaped clusters in all subspaces, and FP-tree structure is employed to mine all dense subspace clusters with two efficient pruning strategies. To demonstrate the effectiveness and efficiency of DNGSC, we conduct extensive experiments on both synthetic and real data sets, and the performance comparison on accuracy and efficiency with DENCOS, MAFIA and CLIQUE shows the superiority of our new algorithm.

Keywords: Subspace Clustering, Density and Non-Grid, Kernel Density Estimation, FP-tree

1 Introduction

Clustering is one of the most important data mining tools for exploring the underlying structure of a data set. However, traditional clustering techniques do not work well for clustering high-dimensional data due to the curse of dimensionality [1, 4, 11], which refers to the phenomenon that as the number of dimensions increases, the distance of a given point x to its nearest point will be close to the distance of x to its farthest point. Thus, discovering meaningful and accurate clusters will be very challenging when facing the loss of distance metrics in high dimensions.

A common approach to address the curse of dimensionality problem is to reduce the dimensionality by using techniques of feature transformation and feature selection [17]. The feature transformation techniques, such as principle component analysis (PCA) or Karhunen-Loeve transformation [12] transform the original data space into a lower dimensional data space by forming dimensions that are linear combinations of the original attributes. However, the transformed dimensions have lost intuitive meanings and thus the discovered clusters are difficult to interpret and analyze. On the other hand, the feature selection methods [2, 9] reduce the data dimensionality by optimally selecting the most relevant attributes from the original data attributes. In such a scenario, these methods are not effective in identifying clusters that may exist in different subspaces of the original data space. Motivated by the challenges that different data points may be clustered in different subspaces, much work has been done on subspace clustering, which aims at discovering clusters embedded in subspaces of the original data space. And the effectiveness of subspace clustering has been demonstrated in various applications, including gene expression data analysis, customer recommendation systems, DNA microarray analysis and so forth [13, 16, 25].

Most of the previous subspace clustering works [7, 14, 21, 24] are grid and density based algorithms which aim at discovering subspace clusters by regarding the clusters as regions of higher densities than their surroundings in a subspace. These algorithms partition each dimension into a number of equal sized bins and data falling in a bin shares the same density with the bin. The number of bins in each dimension determines the computation requirements and the quality of the clustering results. In data sets with large amounts of dimensions and varying data distributions, a uniform grid size leads to enormous amount of computation for fine grids and very poor cluster quality due to the loss of real underlying data distribution. MAFIA [23] proposes an adaptive grid size method to reduce the computation and improve the clustering quality whereas it still suffers from the difficulties in determining the parameters of window size and adjacent windows merging threshold, which may have great influence on representing the underlying data distributions. Considering the dilemma of grid partition method, we note that a more effective way to cope with this dilemma is to calculate the density of each data point rather than the density of each bin. However, little work has been done from this perspective. Motivated by this idea, in this paper we introduce the nonparametric estimation method, i.e. Kernel Density Estimation (KDE) [3, 18, 19] to uncover the real underlying distribution of data sets in each dimension. To the best of our knowledge, it is the first attempt to utilize the KDE method to address the dilemma of grid partition. In KDE, the influence of each data point is modeled using an influence (or kernel) function, and the overall density of the data is calculated as the sum of the kernel functions of all data points. Utilizing the KDE method, the real distribution can be accurately discovered and regions whose densities exceed average density can be effectively identified, and no extra DFS procedure mostly used in other studies [7, 21, 24] is needed to find connected dense units, thus remarkably reducing the computation time and improving clustering quality.

In addition, DENCOS [24] raises a critical problem, called "the density divergence problem" which refers to the phenomenon that cluster densities vary in different subspace cardinalities. Most grid-based algorithms [7, 21, 23] who utilized an absolute unit density threshold failed to take this critical problem into account. As the number of dimensions increases, data points are spread out in a more sparse space, thus displaying the varying region densities in different subspace cardinalities. In order to address density divergence problem, DENCOS proposes that different density threshold in different subspace cardinalities should be determined to identify subspace clusters with different subspace cardinalities. In DENCOS, subspace clusters with the same subspace cardinality share the same density threshold with its corresponding subspace cardinality. More generally, our approach extends this idea into a more general case that the spans of dense regions in different dimensions generated by KDE can be directly utilized to calculate the density thresholds of different subspaces more than in different subspace cardinalities, thus showing the ability to accurately discover arbitrarily shaped clusters in all subspaces. After identifying dense regions in each dimension, the original data is transformed into the encoded numbers with their corresponding dense regions, on which the FP-tree [10] is constructed to mine all dense subspace clusters with two efficient pruning strategies. By conducting experiments on both synthetic and real data sets, the experimental results reveal that our algorithm has better performance in both effectiveness and efficiency than previous grid-based work.

In this paper, we make the following contributions:

– We introduce the Kernel Density Estimation method to accurately uncover the underlying boundary of data without any presumed canonical distribution, which is able to effectively address the dilemma of grid partition.
– Different density thresholds for different subspaces are automatically determined to identify clusters in all subspaces to effectively deal with "the density divergence problem".
– We employ the FP-tree structure rather than an Aprior like scheme to mine all dense subspace clusters and two efficient pruning strategies are further designed to accelerate the mining process.
– Our DNGSC algorithm can discover arbitrarily shaped subspace clusters and is insensitive to the only input parameter.

The remaining of the paper is organized as follows. In Section 2, some related works in subspace clustering are presented. In Section 3, we give the preliminary of Kernel Density Estimation and formal definitions of our proposed subspace clustering model. Then, the proposed DNGSC algorithm will be described in Section 4. In Section 5, we present the experimental results. Finally, Section 6 concludes our work.

2 Related Work

In general, the subspace clustering algorithms can be divided into two categories distinguished by their approach to searching for subspaces, i.e. bottom-up search and top-down search methods. The first category of approaches to subspace clustering are grid and density based algorithms by using a bottom-up search method, including [7, 21, 23, 24]. In CLIQUE [21], the data space is first partitioned into equal sized units. Then subspace clusters are discovered by finding connected dense units whose densities exceed a threshold τ. However, by using grid partitioning strategy, CLIQUE fails to consider the real data distribution in each dimension, thus degrading the quality of clustering results. Further, an absolute unit density threshold makes CLIQUE difficult to identify high quality clusters in subspaces in high dimensionalities.

Additionally, the algorithm ENCLUS [7] extends CLIQUE by introducing the concept of subspace entropy for selecting some interesting subspaces to discover clusters. The quality of a subspace is defined using three criteria: coverage, density and correlation. Entropy can be used to measure all these three criteria. However, in ENCLUS, the same clustering model in CLIQUE is employed in discovering the clusters such that the dilemma of grid partitioning and the density divergence problem will be inevitably faced. A more significant modification of CLIQUE is presented in MAFIA [23] which uses an adaptive grid based on the distribution of data to improve the efficiency and the cluster quality. MAFIA initially creates a histogram to determine the minimum number of bins for each dimension. Then adjacent cells of similar density are merged to form larger cells. In this manner, each dimension is partitioned based on the data distribution and the boundaries of the cells capture the cluster perimeter more accurately than fixed sized grid cells. As a result, MAFIA performs many times faster than CLIQUE and achieves higher quality of clustering results than CLIQUE. Nonetheless, MAFIA is rather sensitive to parameters of window size and merging adjacent threshold, which has great impact on discovering the real data distribution. Also, like CLIQUE and ENCLUS, MAFIA fails to take the density divergence problem into consideration, rendering it difficult to discover clusters with high qualities in all subspace cardinalities.

Instead of using an absolute unit density threshold to identify dense units, DENCOS [24] adaptively determines different density thresholds to discover clusters of relatively high densities in subspaces of different subspace cardinalities by introducing a density parameter α for users to specify their expected relative rate of densities of dense regions and the average regions density in a subspace. In DENCOS, the dense unit discovery is performed by utilizing a novel data structure DFP-tree (Density FP tree) which is constructed on the data set to store the complete information of the dense units. From the DFP tree, it computes the lower bounds and upper bounds of the unit counts for accelerating dense unit discovery and this information is utilized in a divide-and-conquer scheme to efficiently extract dense regions. Experimental results reveal that DENCOS has better performance in both clustering quality and efficiency than previous works. However, DENCOS still suffers from the dilemma of grid partition in which case data distributions are neglected, rendering it difficult to identify subspace clusters with high qualities in data sets with varying distributions.

The second category of approaches to subspace clustering are top-down search based methods, also called projected clustering [5, 6, 15]. The problem of projected clustering aims at finding $(k+1)$ partitions $\{C_1, C_2, \cdots, C_k, O\}$ of the data, where C_i representing the ith cluster is a set of objects that are closely correlated in a subspace and O contains the outliers. While projected clustering techniques also aim at discovering subspace clusters in various subspaces, their output is significantly different from the output of grid and density based subspace clustering, where projected clustering aims to partition the data into disjoint sets and grid and density based subspace clustering allows a data point belong to different subspace clusters. Therefore, projected clustering algorithms that do not allow overlapping clusters are not considered here.

3 Preliminaries

In this section, we first briefly introduce the basic idea of Kernel Density Estimation and formally define the necessary notations (for a more detailed discussion of KDE see [19]). Then, illustrations of identifying dense regions in each single dimension are displayed, and density threshold definitions for different subspaces are precisely formulated in the end of this section.

3.1 Kernel Density Estimation

Kernel density estimation is a non-parametric method of estimating the probability density function of a random variable, which can be used with arbitrary distributions and without the assumptions that the forms of underlying densities are known. KDE is based on the idea that the influence of each data point can be modeled formally using a mathematical function, called the kernel. The kernel function can be seen as a function which describes the influence of a data point within its neighborhood. The kernel function is applied to all data points. An estimate of overall density can be calculated as the sum of the influences of all data points. Given a one dimensional data set $D = \{x_1, \cdots, x_n\}$ drawn from some unknown distribution, our goal is to estimate its probability density function(p.d.f) $p(x)$ and the estimated density is denoted as $\hat{p}(x)$. Its kernel density estimator [19] is

$$\hat{p}_n(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}K(\frac{x - x_i}{h}) \tag{1}$$

Where $K(\cdot)$ is the kernel, a symmetric but not necessarily positive function that integrates to one and $h > 0$ is a smoothing parameter called the bandwidth. A kernel with subscript h is called the scaled kernel and defined as $K_h(x) = \frac{1}{h}K(\frac{x}{h})$. Due to its good mathematical properties, Gaussian kernel function [3] is most commonly used in practice:

$$K_{\text{Gauss}}(x) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}x^T x) \tag{2}$$

By substituting Gaussian kernel function into (1), we obtain the following estimate:

$$\hat{p}_n(x) = \frac{1}{\sqrt{2\pi}nh}\sum_{i=1}^{n}\exp(-\frac{(x - x_i)^T(x - x_i)}{2h^2}) \tag{3}$$

The bandwidth h controls how much the influence of a data point depends on the distance to its neighborhood points, thus exhibiting a strong influence on the resulting estimate. As a result, a considerable amount of work [3, 8, 22] has been done in determining the optimal bandwidth for a good estimate. In this paper, we choose the automatic bandwidth selection with normal kernels which is carried out by [26][1] because it is a reliable and extremely fast kernel density estimator for one-dimensional data. This KDE script implements a novel automatic bandwidth selector that does not rely on the commonly used Gaussian plug-in rule of thumb heuristic [3]. Unlike many other implementations, this KDE script is immune to problems caused by multi-modal densities with widely separated modes, and the estimation does not deteriorate for multi-modal densities because no parametric models are assumed for the data.

In general, the utility of original density function leads to a quadratic runtime behavior because the density of each point requires summing up of the kernel density functions of all data points. However, this KDE script shows a time complexity of $O(N \cdot \log(N))$ where N equals $2^{\lfloor \log 2(n) \rfloor}$ and n is the size of data set, because the script uses Fast Fourier Transform (FFT) whose performance is fastest when N is a power of two. By a simple deduction[2], we

[1] The KDE script is available at http://www.mathworks.com/matlabcentral/fileexchange/14034.

[2] Since $1 \leq \frac{N \cdot \log(N)}{n \cdot \log(n)} = \frac{N}{n} \cdot \frac{\log(N)}{\log(n)} < 2 \cdot 2 = 4$, hence (4) holds.

can obtain the complexity of KDE script in $O(n \cdot \log(n))$ which means

$$n \cdot \log(n) \leq N \cdot \log(N) \leq c \cdot n \cdot \log(n), \quad c \text{ is a constant} \tag{4}$$

For an intuitive image, the figure of (4) is illustrated in Fig.1.

3.2 Identification of dense regions in one-dimensional data

Let $D = \{x_1, \cdots, x_n\}$ be a one-dimensional data set and our goal is to identify the dense regions by utilizing KDE method. We consider a region to be statistically significant "dense" if its density exceeds the Uniform Distributed Density (Udd) where the data is uniformly scattered along the entire data space. To clearly illustrate the effectiveness of KDE to identify dense regions, we use a synthetic data sample D of 100 points drawn from the standard normal and 100 points from a normal distribution with mean 5 and variance 1. When KDE method is applied to D, two statistically significant dense regions are identified whose densities exceed Udd and its illustrative figure is explicitly demonstrated in Fig. 2.

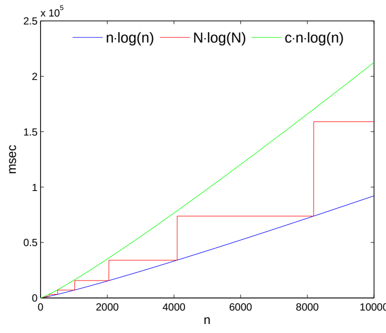


Fig. 1: Time complexity of KDE

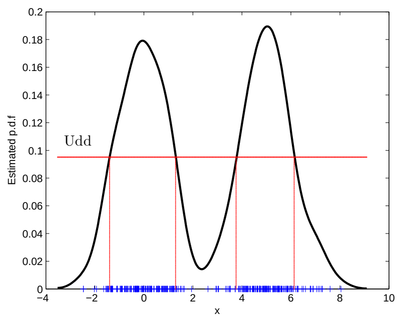


Fig. 2: Illustration of KDE on D

Figure 2 shows that two statistically significant dense regions identified by KDE truly reflect the real distribution boundaries, in which case it is quite difficult for most grid-based approaches to achieve such effects. Furthermore, in order to filter regions with low coverage (the number of data points falling in its region) when the density estimate has much statistical variability, we introduce a coverage threshold[3] φ, which is a user-specified parameter, to formally define a dense region when a statistically significant dense region has a larger coverage percentage than φ. Therefore, identifying one-dimensional dense regions using KDE method can be formulated into a two step procedure which is explicitly shown as follows:

Step 1. Identify statistically significant dense regions whose estimated density $\hat{p}_n \geq Udd$, where Udd is calculated with $Udd = \frac{1}{\max(x)-\min(x)}$. As a result, we can obtain the statistically significant dense regions set denoted as $R = \{[l_1, h_1], \cdots, [l_t, h_t]\}$ where $[l_i, h_i] (1 \leq i \leq t)$ denotes the ith statistically significant dense region and t is the number of statistically significant dense regions.

[3] In this paper, the coverage threshold φ means the percentage of overall data points rather than the number of data points.

Step 2. Filter R to identify dense regions[4] when $c_i \geq \varphi \cdot n$, $(1 \leq i \leq t)$, where c_i denotes the coverage of the ith statistically significant dense region. Accordingly, the dense regions set $R' = \{[l_1, h_1], \cdots, [l_{t'}, h_{t'}]\}$ where $t' \leq t$ can be ultimately discovered.

3.3 Density thresholds for different subspaces

Let $A = \{A_1, A_2, \cdots, A_d\}$ be the set of d attributes of the data set, and $S = A_1 \times A_2 \times \cdots \times A_d$ be the corresponding d dimensional data space. L_i denotes the span of the ith dimension where $i \leq d$. Any k dimensional subspace of S is the space with k dimensions drawn from the d attributes, where $k \leq d$. The cardinality of the subspace is defined as the number of dimensions forming this subspace. For any k dimensional subspace, its density threshold τ_k is defined as follows:

$$\tau_k = \prod_{i=1}^{k} \frac{h_{ij} - l_{ij}}{L_i} \tag{5}$$

Where the second level subscript j stands for jth dense region that the ith dimension of this k dimensional subspace resides in. We restrict $k \geq 2$ where one dimensional dense subspaces can be directly obtained from dense regions in each dimension. Particularly, when dense regions in all dimensions have the equal length (equal-sized grid), (5) is specialized to density threshold formulation proposed in DENCOS. In such scenario, (5) is the generalization of the method proposed in DENCOS such that it has the ability to discriminate more accurate subspaces in the same subspace cardinality. Consequently, a k dimensional subspace is considered to be dense when its coverage $C_k \geq \tau_k \cdot n$. Thus, k dimensional subspace clusters are discovered by identifying all k dense dimensional subspaces.
Problem Definition. Given a set of d dimensional data points and the input φ, find clusters in all subspaces of the original data space whose densities satisfy (5).

4 DNGSC Algorithm

In this paper, we devise the **DNGSC** algorithm, standing for *Density and Non-Grid based Subspace Clustering*, to discover the clusters with our proposed density thresholds. The DNGSC algorithm is divided into three steps: we first identify all the dense regions in each dimension and transform the original data into encoded number format, and the FP-tree is constructed to store the complete information of the transformed data set. Finally, dense subspace clusters are mined in FP-tree with two efficient pruning strategies to accelerate the mining process.

4.1 Data tranformation

In the data transformation phase, we first encode all the dense regions identified by KDE with consecutive numbers $\{1, 2, \cdots, i, \cdots, \sum_{j=1}^{d'} k_j\}$, where k_j

[4] When we say statistically significant dense regions, we refer to regions identified by **Step 1**;Otherwise, when we say dense regions we refer to filtered regions identified by **Step 2**.

is the number of dense regions in jth dimension and d' is the number of dimensions containing dense regions. The encoded numbers of dense regions can be mapped to the dimensions they belong to using a hash table. Then each data in every dimension is transformed into the encoded number of their corresponding dense region they reside in. Consider the example dataset DB with four dimensions shown in Table 1, which is part of our synthetic data (see Section 5.1). Zero in the transformed DB (TDB) means that its corresponding original data falls in the region which is not statistically significant dense or whose coverage percentage is lower than the coverage threshold φ. Particularly, the data in the fourth dimension is all transformed into zero, indicating no dense regions are identified in the fourth dimension in which case data is uniformly distributed in this dimension. Accordingly, the data in the first dimension is transformed into encode numbers 1 and 2, showing that two dense regions are identified in the first dimension, and the rest data in other dimensions is transformed using the same manner. Subsequently, TDB is sorted in the descending order according to their coverage of its corresponding dense region for further inserting into the FP-tree, where the Sorted TDB (STDB) is shown in the rightmost column of Table 1.

Table 1: Transformation of example DB

TID	DB	TDB	STDB
01	(-0.5, -0.2, 20.7, -8.5)	(1,3,6,0)	(1,6,3)
02	(0.3, -0.4, 8.8, -1.3)	(1,3,6,0)	(1,6,3)
03	(-0.2, 5.2, 8.1, 4.3)	(1,4,6,0)	(1,6,4)
04	(49.9, 15.7, 30.4, 5.3)	(2,5,7,0)	(7,5,2)
05	(50.7, 14, 31.7, 6.4)	(2,5,0,0)	(5,2)
06	(48.0, 18.2, 30.3, 6.3)	(2,0,7,0)	(7,2)

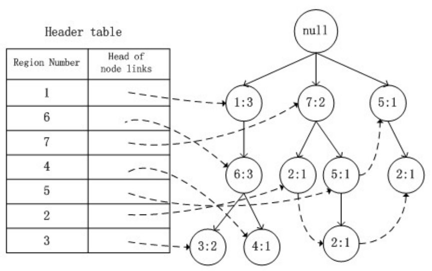


Fig. 3: FP-tree constructed on STDB in Table 1

4.2 Construction of FP-tree

We model the problem of subspace clustering to a similar problem of frequent itemset mining. We note that by regarding the dense regions in all dimensions as a set of unique items in the frequent itemset mining problem, any k-dimensional subspace can be regarded as a k-itemset. However, since different density thresholds are utilized to discover the dense subspaces for different subspaces, the frequent itemset mining techniques cannot be adopted here to discover clusters in which case the monotonicity property in Aprior [20] candidate generate-and-test scheme no longer exists. In addition, the FP-tree [10] based mining algorithm cannot be directly adopted to discover the clusters with multiple thresholds since the theorem[5] in [10] does not hold in such an environment [24]. For this challenge, we utilize the FP-tree structure but devise a different itemset mining algorithm with two efficient pruning strategies to accelerate the mining process.

After transforming the data set, the FP-tree is constructed to condense the transformed DB by inserting each sorted TDB data as a path in the FP-tree with nodes storing the one-dimensional dense regions. The paths with common prefix nodes will be merged and their node counts are accumulated. Fig. 3 shows the FP-tree constructed on the transformed DB in Table 1 (For

[5] Lemma 3.2 in [10].

a detailed construction process, see [10]). Each node contains two fields, i.e. encoded number of dense region and its node count. The encoded numbers of dense regions in the Header table are sorted in the descending order according to their corresponding coverage. It is important to note that the FP-tree in our algorithm costs less space than DFP proposed in DENCOS since in most cases the number of dense regions is dramatically smaller than the number of dense units, in which perspective connected dense units remain to be grouped to form dense regions. Thus, only a small number of dense-regions-based nodes rather than a large number of dense-units-based nodes need to be stored in the FP-tree.

4.3 Mining dense subspaces

In the mining phase, the FP-tree is employed to discover all the dense subspaces. The one-dimensional dense subspaces are directly discovered from the header table of FP-tree and the hash table which maps the dense regions to their corresponding dimensions. To discover higher dimensional dense subspaces from the FP-tree, a naive method would be to first enumerate the combinations of the nodes in each path to generate the candidate subspaces, and then examine the coverage of these subspaces with the corresponding density thresholds. In FP-tree, for any dense region θ_i the paths ending with θ_i can be obtained by following θ_i's node-links in the header table of FP-tree. From this perspective, without the brute-force generation of the candidate subspaces from each path, we exploit a bottom-up scheme to discover all the dense subspaces from the paths set of the header table whose member is paths ending with the same dense region θ_i, and this paths set contains all the paths information of FP-tree which has been demonstrated[6] in [10]. In order to accelerate the discovering process, two efficient pruning strategies are further introduced in the followings.

Lemma 1 (Single path for a dense region). *Suppose that there is only a single path p ending with a given dense region θ_i, indicating the node of θ_i occurs only once in the FP-tree. If p is not dense[7] according to the corresponding density threshold, any sub-path of p ending with θ_i is not dense, either.*

Proof. Let c_i be the coverage of dense region θ_i and k be the length of p. If p is not dense according to the corresponding density threshold, then $c_i < \tau_k \cdot n$ holds. Due to the monotonically decreasing property of our density thresholds definition, $\tau_k < \tau_{k'}$ where $k > k'$ and k' is the length of any sub-path of p ending with θ_i. Accordingly, $c_i < \tau_k \cdot n < \tau_{k'} \cdot n$ holds, showing that any sub-path of p ending with θ_i is not dense, either. Therefore, we have Lemma 1. $\square$

When the node of a given dense region θ_i occurs only once in the FP-tree and its path p is not dense, all its sub-paths no longer need to be tested and thus all the corresponding subspaces are pruned. Otherwise, Lemma 1 is utilized when discovering dense subspaces in sub-paths with length of $\{k-1, k-2, \cdots, 2\}$. However, when the node of θ_i occurs more than once in the FP-tree, we compute the upper bound of the subspace coverage to prune the lower dimensional subspaces whose idea is displayed in Lemma 2.

[6] Property 3.1 in [10].

[7] When we say a path p is dense, we refer to the subspace corresponding with p is dense.

Lemma 2 (Upper bound of subspace coverage). *Given θ_i whose corresponding code has an occurrence of $m (m > 1)$ in the FP-tree, let $P = \{p_1, \cdots, p_m\}$ be the set of m paths ending with θ_i. If the total counts of these m nodes are lower than the product of data size n and the minimum density threshold of subspace cardinality k, then any sub-path of $p_j (1 \leq j \leq m)$ whose length does not exceed k could not be dense, either.*

Proof. Since $\sum_{j=1}^{k} c_j < n \cdot \min(\tau_k)$ and $\tau_k < \tau_{k'}$, where $k > k'$ and c_j denotes the count of the jth node of θ_i occurring in FP-tree, we derive that $\sum_{j=1}^{k} c_j < n \cdot \min(\tau_k) < n \cdot \tau_{k'}$, indicating any combination of the sub-paths from these m paths could not be dense. Thus, we have the Lemma 2. □

In order to have a better understanding of Lemma 2, we give a detailed example to illustrate how Lemma 2 works when pruning candidate subspaces.

Example 1. Let $P = \{(2/5/7/10 : 3), (3/5/7/10 : 4), (3/5/8/10 : 4), (4/5/8/10 : 5)\}$ be the set of four paths ending with dense region whose encoded number is 10, the data size be 1000 and the minimum density threshold for 4-dimensional subspaces be 0.02. In this case, the total coverage of these four paths is 16 which is lower than 20(1000·0.02). Thus, there is no necessity to further check the sub-paths of P, showing excellent capabilities to prune candidate subspaces.

Based on the above Lemma 1 and 2, we have the following algorithm for mining dense subspaces using FP-tree.

Table 2: *DenseSubspaceMining* algorithm

```
Algorithm: DenseSubspaceMining
Input: FP-tree and dense regions identified by KDE.
Output: All dense subspaces.
1. Get paths set $S$ of head table by BFS of FP-tree;
2. For each member $P$ in $S$
3.     If $P$ has a single path $p$
4.         mine $P$ using Lemma 1;
5.     Else
6.         For each $k$ from 2 to maxLength($p_j$)
7.             If $\sum_{j=1}^{m} < n \cdot \min(\tau_k)$ continue;
8.             Else mine dense subspaces from all sub-paths with length of $k$;
9.         End For
10.    End Else
11.End For
```

Time complexity of DNGSC. Let n be the size of data set, d be the dimensionality of the data space and k be the maximal dimensionality of subspace clusters. The total time complexity of DNGSC is $O(d \cdot n \cdot \log(n) + c \cdot 2^k)$ where c is the number of dense regions identified by KDE method. $O(d \cdot n \cdot \log(n))$ indicates that the KDE method has been applied in all of d dimensions where the complexity of KDE is $O(n \cdot \log(n))$. $O(c \cdot 2^k)$ shows the theoretic complexity of mining procedure whereas in reality the performance of mining procedure is much faster than $O(c \cdot 2^k)$ for the utility of two efficient pruning strategies.

5 Experiments

We empirically evaluate DNGSC using synthetic as well as real datasets. On synthetic datasets, we report the results of scalability of the DNGSC algorithm in terms of data size, dimensionality of data and dimensionality of clusters. The performance comparison results of accuracy and efficiency on both synthetic and real datasets with CLIQUE, MAFIA and DENCOS are also evaluated in our experiments. All the experiments are conducted in Windows 7 Home Basic system with 2GB RAM and 2.27G i3CPU.

5.1 Synthetic data generation

We generate synthetic datasets using our own data generator which employs techniques similar to those mentioned in [21, 24]. They permit to control the size and the structure of the generated data sets through parameters such as dimensionality of the data and dimensionality of subspace clusters. We scale all the dimensions to lie in the range [0,100]. Then we generate the data set such that the average densities of the data points inside the clusters are much larger than their surrounding regions. The attribute values for a data point assigned to a cluster are generated as follows. For those attributes that define the subspace in which the cluster is embedded, the value is drawn independently at random from uniform distribution over within the range of the cluster in which it is embedded. For the remaining attributes, the value is drawn independently from the uniform distribution over the entire range of the attribute. After populating the user defined subspace clusters, an additional 10% of points are added as random noise.

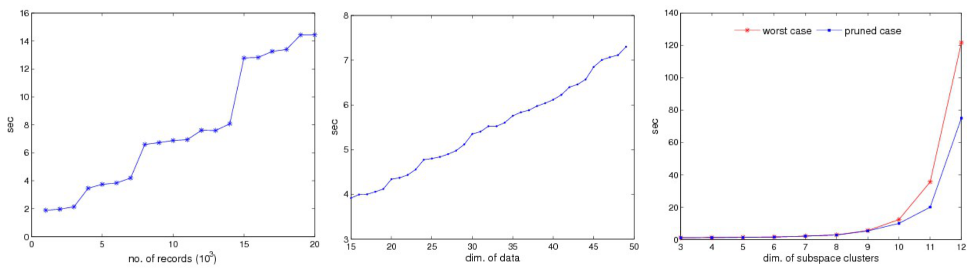
5.2 Synthetic data results

Scalability of DNGSC We evaluated the scalability of DNGSC against the size of data set, the dimensionality of data and the dimensionality of subspace clusters using several synthetic datasets. All experiments were run with $\varphi = 0.25$.

Scalability against data size. Figure 4a shows the scalability as the data size was increased from 1000 to 20000 records. The data space had 15 dimensions and there were 3 clusters with a 3-dimensional, 5-dimensional and 7-dimensional subspace, respectively. As expected, the curve exhibited a running time behavior of $O(n \cdot \log(n))$ which accorded with the trend displayed in Fig. 1, because the number of passes through the database did not change.

Scalability against dimensionality of data. Figure 4b shows the scalability as the dimensionality of the data varied from 15 to 49. The data set had 5000 records and there were 3 clusters with a 3-dimensional, 5-dimensional and 7-dimensional subspace respectively. We can see that DNGSC scaled linearly with the dimensionality of the data since the KDE method had been applied d(dimensionality of data) times to identify dense regions independently in each dimension.

Scalability against dimensionality of subspace clusters. Figure 4c shows the scalability as the highest dimensionality of the subspace clusters increased from 3 to 12 in a 15 dimensional space with 5000 records. In each case, one cluster was embedded in the relevant subspace of highest dimensionality. The running time reflected the time complexity of our algorithm, which was $O(c \cdot 2^k)$ where c is the number of dense regions and k is the maximal dimensionality of subspace clusters. However, in reality DNGSC algorithm performed much better than the worst case because many of the candidate subspaces were pruned during mining process, where the costs of worst case and pruned case were comparatively illustrated in Fig. 4c, showing a great improvement on the behavior of running time when our pruning strategies were utilized to mine high dimensionality of subspace clusters.



(a) Data size (b) Dim. of data (c) Dim. of subspace clusters

Fig. 4: Scalability of DNGSC

Performance comparison To evaluate the accuracy and efficiency of DNGSC we compared it with CLIQUE, MAFIA and DENCOS on the same synthetic data set with 5000 records. The data space had 15 dimensions and there were 3 clusters with a 3-dimensional, 5-dimensional and 7-dimensional subspace cluster respectively.

Accuracy. We evaluate the quality of discovered clusters by two criteria, precision and recall. For a cluster discovered, "precision" is defined as the percentage of the data points in this cluster that really belong to the known cluster and "recall" is defined as the percentage of the data points in a known cluster that are identified in this cluster. In DNGSC, we used $\varphi = 0.25$ to identify dense regions in each dimension. We studied a broad range of parameter settings in CLIQUE, MAFIA and DENCOS, and the parameters achieving the best results are chosen for fair comparisons. In such a scenario, we used $\delta = 10$ to partition the dimensions and $\tau = 0.02$ to select dense units in CLIQUE. In MAFIA, we set the windows size to 20, the merging adjacent threshold to 0.2 and $\tau = 0.02$ to select dense bins. In DENCOS, we employed $\delta = 10$ to , $\alpha = 3$ and $k_{\max} = 15$ to discover all the possible dense subspace clusters. The χ^2-test statistics on recall were computed to compare the accuracy performance of DNGSC with the other three algorithms.

Table 3 shows that all the four algorithms can discover the three subspace clusters embedded in a 15-dimensional data space. We noted that DNGSC, DENCOS, MAFIA and CLIQUE gained the same high precision for the three subspace clusters as the average densities of these three subspace clusters were significantly larger than their surrounding regions. However, DNGSC achieved better performance on recall than DENCOS, MAFIA and CLIQUE due to the utility of the KDE method to discover the real data distribution, where the χ^2-test value asterisked with "*" showed that DNGSC outperformed other methods significantly with a statistical confidence level of 95%. When the grid partition method was employed in DENCOS, MAFIA and CLIQUE, it was difficult for them to discover the underlying data distribution in which case the data points belonging to a specific cluster can be only partly identified, thus the accuracy of recall degraded.

Efficiency. We compared the execution time of DNGSC with DENCOS, MAFIA and CLIQUE shown in Table 4 using the same parameter settings as the above Section. We noted that DNGSC performanced much faster than DENCOS, MAFIA and CLIQUE due to two critical reasons. On one hand, the

Table 3: Accuracy comparison on synthetic data set

Algorithm	PR&Recall	Dim of subspace clusters			χ^2-test
		Dim 3	Dim 5	Dim 7	
DNGSC	Precision	99.8%	100%	100%	–
	Recall	99.6%	100%	100%	0.0067*
DENCOS	Precision	99.8%	100%	100%	–
	Recall	85%	92.1%	93.2%	20.11
MAFIA	Precision	99.8%	100%	100%	–
	Recall	79.8%	88.7%	87.7%	41.22
CLIQUE	Precision	99.8%	100%	100%	–
	Recall	79.8%	82.2%	75.3%	80.11

Table 4: Efficiency comparison on synthetic data set

Algorithm	DNGSC	DENCOS	MAFIA	CLIQUE
Exe.Time(s)	2.1	5.3	11.6	45.7

number of dense regions identified by KDE method was significantly smaller than the number of grid-based dense units employed in DENCOS, MAFIA and CLIQUE, thus reducing the computation time when generating candidate subspaces. On the other hand, the FP-tree in DNGSC and DFP-tree in DENCOS were much more efficient than Aprior-Like candidate generate-and-test scheme in MAFIA and CLIQUE when discovering the dense subspace clusters, rendering DNGSC and DENCOS with better time efficiency than MAFIA and CLIQUE.

5.3 Real datasets

Two real data sets from UCI machine learning repository[8] were also utilized to evaluate the clustering results of DNGSC. The two real data sets are yeast data set and image segmentation data set. The yeast data set contains 1484 sequences with seven numerical attributes from the original data set where we remove the seventh numerical attribute who almost shares the identical value through the entire data space, and the image segmentation data set contains 2310 instances drawn randomly from an image database with ten independently numerical attributes where we remove nine attributes (3-5, 14-19) from the original data set who have been calculated from other attributes.

Accuracy. The quality of the clustering results is evaluated in terms of *subspace density*, abbreviated as SD. For a subspace S', SD is the ratio of the coverage of S' to its corresponding occupied spatial volume, which is formulated as:

$$SD(S') = \frac{C}{\prod_{i=1}^{k}(h_{ij} - l_{ij})} \tag{6}$$

Where C is the coverage of S' and k is the cardinality of S'. The second level subscript j stands for jth dense region that ith dimension of S' in which it resides. A more intuitive interpretation of SD is the average number of data points per volume unit in its corresponding subspace. Since DNGSC, DENCOS, MAFIA and CLIQUE, all aim at extracting high density subspaces as clusters, higher SD values mean that the subspaces with higher densities can be better separated from their surrounding regions, and thus is a better quality result. In order to compare SD value in a comparative magnitude, data

[8] http://archive.ics.uci.edu/ml/datasets.html

in all dimensions is normalized into the scale [0,10]. In DNGSC, we still used $\varphi = 0.25$ to identify dense regions in each dimension for these two real data sets. As for DENCOS, MAFIA and CLIQUE, for each data set we ran with many parameter settings and chose the parameters with the best performance. More specifically, in DENCOS we employed $\delta = 10, \alpha = 3$ and $k_{\max} = d$ for both real data sets, where d denotes the dimensionality of each data set. In MAFIA, we set the windows size to 20, the merging adjacent threshold to 0.2 and $\tau = 0.02$ for yeast data set, and set windows size to 10, the merging adjacent threshold to 0.2 and $\tau = 0.033$ for image segmentation data set. In CLIQUE, we utilized $\delta = 10$ and $\tau = 0.02$ for yeast data set, and $\delta = 8$ and $\tau = 0.05$ for image segmentation data set.

Table 5 and Table 6 show the quality of results of DNGSC compared with DENCOS, MAFIA and CLIQUE on yeast and image segmentation data sets. When comparing the clustering results, we showed the SD gain for each subspace cardinality k. For a subspace cardinality k, the SD gain is the average SD value of the subspaces with cardinality k. As can be seen from Table 5 and Table 6, DNGSC resulted in much higher SD gain than other three algorithms in almost one magnitude for these two real data sets. The critical reason was that the KDE method has the capability to identify the boundary of data distribution more accurately than grid-based methods, in which case even the algorithms shared the same coverage with regards to a specific subspace, the more accurate boundary identification contributed to a smaller denominator calculated in (6), thus resulting in a higher SD gain than other algorithms. Besides, in MAFIA and CLIQUE, a strict density threshold setting can discover clusters with good qualities in lower subspaces but poor qualities in higher subspaces, whereas a looser density threshold setting can discover clusters in higher subspaces but will result in the decreased quality of clusters in lower subspaces. However, in DNGSC and DENCOS, different density thresholds were utilized to discover clusters in all subspaces. As a result, higher SD gains than MAFIA and CLIQUE were achieved. More specifically, DNGSC achieved a higher SD gain than DENCOS since different density thresholds were employed for different subspaces rather than different subspace cardinalities, in which scenario subspace clusters in an identical subspace cardinality can be more accurately identified, thus demonstrating the effectiveness of our proposed subspace clustering model.

Table 5: Yeast data set

Algorithm	SD			
	Dim 2	Dim 3	Dim 4	Dim 5
DNGSC	4010	1363	418	122
DENCOS	475	297	188	70
MAFIA	240	189	139	87
CLIQUE	111	53	38	33

Table 6: Image Segmentation data set

Algorithm	SD				
	Dim 2	Dim 3	Dim 4	Dim 5	Dim 6
DNGSC	2128	1182	574	224	74
DENCOS	511	266	147	84	49
MAFIA	255	153	101	75	51
CLIQUE	235	140	96	70	53

Table 7: Efficiency comparison on two real data sets

Algorithms		DNGSC	DENCOS	MAFIA	CLIQUE
Exe.Time(s)	Yeast	1.1	2.47	9.95	17.4
	Segmentation	1.5	2.5	23.4	57

Efficiency. Table 7 shows the execution time of algorithms DNGSC, DENCOS, MAFIA and CLIQUE on the two real data sets utilizing the same parameter settings in the above Section. As shown in Table 7, DNGSC had better time efficiency than DENCOS, MAFIA and CLIQUE for the primary reason that the number of dense regions generated by the KDE method was much smaller than the number of grid-based dense unit used in DENCOS, MAFIA and CLIQUE. Furthermore, DNGSC and DENCOS outperformed MAFIA and CLIQUE since in the real data set with high dimensionality, FP-tree in DNGSC and DFP-tree in DENCOS were much more efficient than the Aprior-Like candidate generate-and-test scheme in MAFIA and CLIQUE. Thus, the utilities of the KDE method and FP-tree demonstrated the high efficiency for real data sets.

5.4 Parameter discussion

As in most approaches, the quality of clustering results heavily depends on an appropriate choice of parameters. In DNGSC, we only have one input parameter, namely the coverage threshold φ which defines a region to be dense when its coverage percentage exceeds φ, and the density thresholds for different subspaces can be automatically determined with the spans of dense regions identified by KDE method. We varied φ from 0.1 to 0.3 where DNGSC achieved consistently good performance on both synthetic data sets and two real data sets in Section 5.2 and Section 5.3, respectively, showing that DNGSC was not sensitive to the setting of parameter φ. Particularly, DNGSC with $\varphi = 0.2$ could achieve the average good performance in most cases where we regarded a region covering more than 20% of the total data points in one dimensional data as dense. It is an appealing property because we do not need to tune the parameters painfully in the real applications.

6 Conclusion

In this paper, we propose a novel subspace clustering algorithm DNGSC via Kernel Density Estimation to discover the arbitrarily shaped subspace clusters. It is insensitive to parameter setting and does not presume any canonical data distribution. In order to address the dilemma suffered by most grid-based subspace clustering algorithms, a well-known theory KDE is utilized to accurately uncover the underlying boundary of data distribution. In addition, different density thresholds for different subspaces are automatically determined to effectively discover clusters in all subspaces, and the FP-tree is constructed to mine all the dense subspace clusters with two efficient pruning strategies to accelerate the mining process. The experimental results reveal that DNGSC can discover the clusters in all subspaces with high quality, and both the effectiveness and efficiency of DNGSC[9] significantly outperform previous grid-based algorithms, thus demonstrating its practical applicability for subspace clustering.

Acknowledgments. We would like to thank Zdravko Botev for online discussion on the time complexity of KDE implementation.

References

1. A. Hinneburg, C. C. Aggarwal, and D. A. Keim.: What is the nearest neighbor in high dimensional spaces? In: VLDB 2000, pp. 506-515. Morgan Kaufmann, San Francisco (2000)

[9] For the source code of DNGSC in Matlab, please contact zjing@tju.edu.cn

2. A. L. Blum and P. Langley.: Selection of relevant features and examples in machine learning. Artificial intelligence 97, 245-271(1997)
3. B. W. Silverman.: Density estimation for statistics and data analysis. Chapman & Hall/CRC, London (1986)
4. C. Aggarwal, A. Hinneburg, and D. Keim.: On the surprising behavior of distance metrics in high dimensional space. In: Proceedings of International Conference On Database Theory, ICDT 2001, pp.420-434. Springer-Verlag, London (2001)
5. C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park.: Fast algorithms for projected clustering. ACM SIGMOD Record 28, 61-72. ACM, New York (1999)
6. C. C. Aggarwal and P. S. Yu.: Finding generalized projected clusters in high dimensional spaces. ACM SIGMOD Record 29, Issue 2. ACM, New York (2000)
7. C. H. Cheng, A. W. Fu, and Y. Zhang.: Entropy-based subspace clustering for mining numerical data. In: KDD 1999, pp. 84-93. ACM, New York (1999)
8. G. Wahba.: Optimal convergence properties of variable knot, kernel, and orthogonal series methods for density estimation. The Annals of Statistics, 15-29 (1975)
9. H. Liu and H. Motoda.: Feature selection for knowledge discovery and data mining, Springer (1998)
10. J. Han, J. Pei, and Y. Yin.: Mining frequent patterns without candidate generation. In: SIGMOD 2000, pp. 1-12. ACM, New York (2000)
11. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft.: When is "nearest neighbor" meaningful? In: ICDT'99, pp.217-235. Springer-Verlag, London (1999)
12. K. Fukunaga.: Introduction to statistical pattern recognition. Academic Press (1990)
13. K. Kailing, H. P. Kriegel, and P. Kröger.: Density-connected subspace clustering for high-dimensional data. In: SDM 2004, pp. 246-257 (2004)
14. K. Sequeira and M. Zaki.: SCHISM: A new approach for interesting subspace mining. In: ICDM 2004, pp.186-193 (2004)
15. K. G. Woo, J. H. Lee, M. H. Kim, and Y. J. Lee.: FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. Information and Software Technology 46, 255-271 (2004)
16. L. Lu and R. Vidal.: Combined central and subspace clustering for computer vision applications. In: ICML 2006, pp. 593-600. ACM, New York (2006)
17. L. Parsons, E. Haque, and H. Liu.: Subspace clustering for high dimensional data: a review. ACM SIGKDD Explorations Newsletter 6, 90-105 (2004)
18. M. P. Wand and M.C. Jones.: Kernel smoothing. Chapman & Hall/CRC (1995)
19. O. D. Richard, E. H. Peter, and G. S. David: Pattern classification, 2nd Edition. A Wiley-Interscience Public (2001)
20. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo.: Fast discovery of association rules. Advances in knowledge discovery and data mining 12, 307-328 (1996)
21. R. A. J. Gehrke, D. Gunopulos, and P. Raghavan.: Automatic subspace clustering of high dimensional data for data mining applications. ACMSIMOD 72, pp.94-105. ACM, New York (1998)
22. R. Cao, A. Cuevas, and W. Gonzalez Manteiga.: A comparative study of several smoothing methods in density estimation. Computational Statistics & Data Analysis 17, 153-176 (1994)
23. S. Goil, H. Nagesh, and A. Choudhary.: MAFIA: Efficient and scalable subspace clustering for very large data sets. In: KDD 1999, pp. 443-452. ACM. New York (1999)
24. Y. H. Chu, J. W. Huang, K. T. Chuang, D. N. Yang, and M. S. Chen.: Density conscious subspace clustering for high-dimensional data. IEEE Transactions on Knowledge and Data Engineering 22, 16-30 (2010)
25. Y. Kim, J. Oh, and J. Gao.: Emerging pattern based subspace clustering of microarray gene expression data using mixture models. In: Proceedings of International Conference on Bioinformatics and Its Applications, ICBA 2004, pp.13-24 (2004)
26. Z. Botev, J. Grotowski, and D. Kroese.: Kernel density estimation via diffusion. The Annals of Statistics 38, 2916-2957 (2010)