Towards Subspace Clustering on Dynamic Data: An Incremental Version of PreDeCon

Hans-Peter Kriegel Peer Kröger Irene Ntoutsi Arthur Zimek

Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538, München, Germany
http://www.dbs.ifi.lmu.de/
{kriegel, kroegerp, ntoutsi, zimek}@dbs.ifi.lmu.de

ABSTRACT

Todays data are high dimensional and dynamic, thus clustering over such kind of data is rather complicated. To deal with the high dimensionality problem, the subspace clustering research area has lately emerged that aims at finding clusters in subspaces of the original feature space. So far, the subspace clustering methods are mainly static and thus, cannot address the dynamic nature of modern data. In this paper, we propose an incremental version of the density based projected clustering algorithm PreDeCon, called incPreDeCon. The proposed algorithm efficiently updates only those subspace clusters that might be affected due to the population update.

1. INTRODUCTION

Clustering is the unsupervised classification of data into natural groups/ clusters so that data points within a cluster are more similar to each other than to data points in other clusters. Cluster analysis has been studied extensively in many contexts and disciplines including Data Mining, and as a result a large number of clustering algorithms exists in the literature [15], [16].

However, modern data impose new challenges and requirements for the clustering algorithms due to their special characteristics. First of all, a *huge amount* of data is collected nowadays as a result of the wide spread usage of computer devices. Another new characteristic is the *high dimensionality* of modern data; an object might be described by a large number of attributes (and each attribute might take values within a large domain, as well) and there might be correlations or overlaps between these attributes. In addition to their quantity and multidimensionality, modern data are also characterized by a high degree of variability. Indeed, most of today's data is *dynamic*; new data records might be inserted and existing data records might be deleted, as time goes by. A special, particularly interesting category of dynamic data is *stream* data that continuously flow in and out of systems at high-speed (e.g. Internet traffic data, sensor data, position tracking data etc.).

Lately, a lot of work has been carried out on adapting traditional clustering algorithms in order to meet the requirements of modern systems or on proposing new algorithms that are specialized on handling data with the above features. In particular, several methods have been proposed for clustering of large amounts of data, e.g., BIRCH [20], DBSCAN [10], OPTICS [5], for clustering over data streams e.g., STREAM [14], CluStream [2] and for change detection and monitoring over evolving data e.g., [1], [19].

To deal with the high dimensionality of data, several dimensionality reduction techniques have been proposed such as feature transformation and feature selection. One of the latest directions in this category, is *subspace clustering* which deals with the problem of high dimensionality by detecting clusters in different subspaces of the original feature space. The main challenges in the subspace clustering problem are to find the correct subspace for each cluster and the correct cluster in each relevant subspace. Several methods have been proposed which can be classified into the following categories: axis-parallel subspace clustering methods (that search for axis-parallel subspace clusters), pattern based clustering methods (that rely on patterns in the data matrix), arbitrarily oriented subspace clustering methods (here clusters appear as hyperplanes of arbitrary dimensionality and orientation in the data space). Several algorithms have been proposed for each category; see a recent survey for further information [18]. So far however, the subspace clustering methods deal mainly with static datasets.

In this work, we propose an incremental version of the density based subspace preference clustering algorithm PreDeCon [6]. PreDeCon belongs to the so called "projected clustering" algorithms since it seeks to assign each point to a unique cluster (thus, clusters are not overlapping), whereas clusters might exist in different (axis-parallel, though) subspaces. We choose the algorithm PreDeCon because it relies on a density-based clustering model such that updates usually do not affect the entire clustering structure but rather cause only limited local changes.

The rest of the paper is organized as follows: In Section 2, we discuss the related work. In Section 3, we present the basic notions of PreDeCon which are necessary for the understanding of the incremental method. In Section 4, we

present the incremental version of PreDeCon, incPreDeCon. Initial experimental results are presented in Section 5, whereas a discussion on the results and on the ongoing work is presented in Section 6. Section 7 concludes this paper and discusses future work.

2. RELATED WORK

So far, the subspace clustering methods (See [18] for an overview) emphasize on the extraction of meaningful clusters from different subspaces of the original feature space and pay little attention to the application of such methods in dynamic data like data warehouses and data streams. In the traditional clustering settings however, there exists a lot of work on both incremental clustering methods and stream methods; we overview below some representative approaches.

Traditional incremental clustering methods rely on the existing clustering at timepoint $t-1$ (based on dataset $\mathcal{D}_{t-1}$) and on the update operations (insertions, deletions) at t in order to derive the new clustering at t. In this category belong methods like incDBSCAN [9] which comprises the incremental version of the density based algorithm DBSCAN [10] and incOPTICS [17] which comprises the incremental version of the density based hierarchical clustering algorithm OPTICS [5]. Both incDBSCAN and incOPTICS exploit the fact that, due to the density based nature of the corresponding static algorithms (DBSCAN and OPTICS, respectively), an update operation affects only some part of the old clustering instead of the whole clustering. The update process works directly upon raw data (although access is restricted to only a subset of these data). Both these methods produce the same results with the corresponding static methods when the latest are applied over the accumulative dataset $\mathcal{D}_t$.

Charikar et al.[7] present an incremental K–Means method which maintains a collection of k clusters as the dataset evolves. When a new point is presented, either it is assigned to one of the current k clusters, or it starts a new cluster while two existing clusters are merged into one, so as the total number of clusters does not exceed the threshold k. Chen et al. [8] propose the incremental hierarchical clustering algorithm GRIN which is based on gravity theory in physics. In the *first phase*, GRIN constructs the initial clustering dendrogram, which is then flattened and its bottom levels are pruned in order to derive the so called tentative dendrogram. For each cluster, the tentative histogram keeps the centroid, the radius and the mass of the cluster. In the *second phase*, new data instances are inserted one by one and it is decided whether they belong to leaf nodes of the tentative dendrogram or are outliers. If the tentative outlier buffer exceeds some threshold, a new tentative dendrogram is reconstructed. Both [7] and [8] are approximate methods, by means that the resulting clustering after the update is not assured to be identical to the one we would obtain if we applied from scratch the static versions of the algorithms over the accumulative dataset $\mathcal{D}_t$. This is due to the fact, that the update process works upon cluster summaries rather than upon raw data; the new data at t are actually "mapped" to the closer cluster of the existing clustering (from timepoint $t-1$).

Data streams impose new challenges for the clustering problem since "it is usually impossible to store an entire data stream or to scan it multiple times due to its tremendous volume" [13]. As a result, several methods have been proposed that first summarize the data through some summary structure and then apply clustering over these summaries instead of the original raw data. With respect to the clustering quality, these summaries might be either lossy (that is, they comprise some approximation of the raw data) or lossless (that, is they exactly maintain the information contained in the original raw data).

Agrawal et al. [2] propose the CluStream framework for clustering of evolving data streams. The clustering process is split into an online and an offline part: The online component periodically stores summary statistics (the so called, micro–clusters), whereas the offline component uses these micro–clusters for the formation of the actual clusters (the so called, macro–clusters) over a user–defined time horizon. No access to raw data is required in this method, since the clustering takes place over the microclusters, which comprise a lossy representation of the original data. The incremental part in this case is the online component which updates the micro–clusters, whereas the clustering process is applied from scratch over these updated summaries.

In the context of their DEMON framework, Ganti et al. [11] present BIRCH+, an incremental extension of BIRCH [20]. The original BIRCH [20] first summarizes the data into subclusters and then it clusters those subclusters using some traditional clustering algorithm. The subclusters are represented very concisely through cluster features. In BIRCH+, the cluster features are maintained incrementally as updates occur, and then the clustering step takes place as in BIRCH over those (now updated) summaries. So, the incremental part is that of summary structure update, whereas clustering is then applied from scratch over the updated summary structure. The incremental version produces the same results as the static version when applied on the accumulative dataset $\mathcal{D}_t$.

Gao et al. [12], propose DUCStream, an incremental data stream clustering algorithm that applies the idea of dense units introduced in CLIQUE [4] to stream data. As in CLIQUE [4], the data space is split into units and a cluster is defined as a maximal set of connected dense units. Their method relies on incrementally updating, according to the update operation, the density of these units and on detecting units that change from dense to non-dense and the inverse. After the grid update phase, they identify the clusters using the original procedure of CLIQUE. DUCStream does not require access to the raw data of the past time points, but only over the summary grid structure. The incremental version produces the same results as the static version when applied to the accumulative dataset $\mathcal{D}_t$. Note that although CLIQUE is a subspace clustering algorithm, the proposed method [12] updates incrementally only the grid summary structure, whereas the clusters are discovered from scratch over the (now updated) grid. This is a clear difference to our work, where the goal is to incrementally update the existing clustering (at $t-1$) based on the dataset updates at t, so as to finally derive the new clustering at t.

Agrawal et al. [3] extend the idea of CluStream [2] to high dimensional data streams by proposing HPStream, a method for projected data stream clustering. A summary structure, the so called fading cluster structure, is proposed which comprises a condensed representation of the statistics of the points inside a cluster and can be updated effectively as the data stream proceeds. The input to the algorithm includes the current cluster structure and the relevant set of dimensions associated with each cluster. When a new point arrives, it is assigned to the closest cluster structure or if this violates the limiting radius criteria, a new cluster is created and thus some old cluster should be deleted in order for the total number of clusters to not exceed the maximum number k. In each case, the cluster structure and the relevant dimensions for each cluster are dynamically updated. Although HPStream is a subspace clustering method and we propose an incremental subspace clustering method in this work, there are core differences between the two approaches and their scopes. In particular, HPStream is targeted to stream data and thus works upon summaries and provides an approximation solution to the clustering problem. On the other hand, our incPreDeCon method works upon dynamic data, requires access to raw data (although this access is restricted to only a subset of the original dataset) and provides exact solution to the clustering problem (i.e., we obtain the same results with those obtained by applying the static PreDeCon on the acumulated dataset $\mathcal{D}_t$).

3. THE ALGORITHM PREDECON

PreDeCon adapts the concept of density based clusters, introduced in DBSCAN [10], to the context of subspace clustering. To this end, the notion of *subspace preferences* for each point is introduced that defines which dimensions the point prefers. Roughly speaking, a point prefers a dimension if along this dimension the point "builds" a neighborhood of small-variance. Intuitively, a *subspace preference cluster* is a density connected set of points associated with a certain subspace preference vector.

In this section, we present a brief description of PreDeCon including the definitions that are required for understanding the incremental version as well. Let $\mathcal{D}$ be a database of d-dimensional points ($\mathcal{D} \subseteq R^d$), where the set of attributes is denoted by $A = \{A_1, A_2, \ldots, A_d\}$. Let $dist : R^d \times R^d \to R$ be a metric distance function between points in $\mathcal{D}$. Let $\mathcal{N}_\varepsilon(p)$ be the ε-neighborhood of $p \in \mathcal{D}$, i.e., $\mathcal{N}_\varepsilon(p)$ contains all points $q \in \mathcal{D}$: $dist(p,q) \leq \varepsilon$.

First the notion of variance along an attribute/dimension A_i is defined for the neighborhood of a point p, in order to derive preferred dimensions for p and then, the preference weighted similarity between two points is defined. Next, the notions of core member property, direct density reachability, density reachability, density connectivity of DBSCAN are adapted to the concept of preferred dimensions.

Definition 1 (Variance along an attribute). *Let $p \in \mathcal{D}$ and $\varepsilon \in \mathbb{R}$. The* variance of $\mathcal{N}_\varepsilon(p)$ along an attribute $A_i \in \mathcal{A}$, *denoted by* $\mathrm{VAR}_{A_i}(\mathcal{N}_\varepsilon(p))$, *is defined as follows:*

$$\mathrm{VAR}_{A_i}(\mathcal{N}_\varepsilon(p)) = \frac{\sum_{q \in \mathcal{N}_\varepsilon(p)} (dist(\pi_{A_i}(p), \pi_{A_i}(q)))^2}{|\mathcal{N}_\varepsilon(p)|}$$

A_i is considered a *preferable dimension* for p, if p builds a dense area in A_i, i.e., the variance with respect to A_i in its neighborhood is smaller than threshold δ.

Definition 2 (Subspace preference dimensionality). *Let $p \in \mathcal{D}$ and $\delta \in \mathbb{R}$. The number of attributes A_i with $\mathrm{VAR}_{A_i} \leq \delta$ is called the* subspace preference dimensionality *of $\mathcal{N}_\varepsilon(p)$, denoted by* $\mathrm{PDIM}(\mathcal{N}_\varepsilon(p))$.

Definition 3 (Subspace preference vector). *Let $p \in \mathcal{D}$, $\delta \in \mathbb{R}$ and $\kappa \in \mathbb{R}$ be a constant with $\kappa \gg 1$. Then, the* subspace preference vector *of p is defined as follows:*

$$\bar{\mathbf{w}}_p = (w_1, w_2, \ldots w_d)$$

where w_i is:

$$w_i = \begin{cases} 1 & \text{if} \quad \mathrm{VAR}_{A_i}(\mathcal{N}_\varepsilon(p)) > \delta \\ \kappa & \text{if} \quad \mathrm{VAR}_{A_i}(\mathcal{N}_\varepsilon(p)) \leq \delta \end{cases}$$

This vector distinguishes preferable dimensions from non preferable ones.

Definition 4 (Preference weighted similarity). *The* preference weighted similarity measure *associated with a point p is defined as follows:*

$$dist_p(p,q) = \sqrt{\sum_{i=1}^{d} w_i \cdot (\pi_{A_i}(p) - \pi_{A_i}(q))^2}$$

where w_i is the i-th component of $\bar{\mathbf{w}}_p$.

The preference weighted similarity measure overweights the original data points distance with a constant parameter $k >> 1$ if the attribute variance is small. Thus preferable attributes are weighted considerable lower comparing to non–preferable attributes.

Since the above distance is not symmetric, a symmetric version of it is defined as follows:

Definition 5 (General preference similarity). *The* general preference weighted similarity *of two arbitrary points $p, q \in \mathcal{D}$, denoted by $dist_{pref}(p,q)$, is defined as the maximum of the corresponding preference weighted similarity measures of p ($dist_p$) and q ($dist_q$), formally:*

$$dist_{pref}(p,q) = \max\{dist_p(p,q), dist_q(q,p)\}.$$

The preference weighted $\varepsilon-$neighborhood of a point p is now defined, containing all those points that are within preference weighted distance ε from p.

Definition 6 (preference ε-neighborhood). *Let $\varepsilon \in \mathbb{R}$. The* preference weighted ε-neighborhood *of a point $o \in \mathcal{D}$, denoted by $\mathcal{N}_\varepsilon^{\bar{\mathbf{w}}_o}(o)$, is defined by:*

$$\mathcal{N}_\varepsilon^{\bar{\mathbf{w}}_o}(o) = \{x \in \mathcal{D} \,|\, dist_{pref}(o,x) \leq \varepsilon\}.$$

The preference weighted core points are now defined as follows:

DEFINITION 7 (PREFERENCE WEIGHTED CORE POINTS). *Let $\varepsilon, \delta \in \mathbb{R}$ and $\mu, \lambda \in \mathbb{N}$. A point $o \in \mathcal{D}$ is called* preference weighted core point *w.r.t. ε, μ, δ, and λ (denoted by $\text{CORE}^{\text{pref}}_{\text{den}}(o)$), if the preference dimensionality of its ε-neighborhood is at most λ and its preference weighted ε-neighborhood contains at least μ points, formally:*

$$\text{CORE}^{\text{pref}}_{\text{den}}(o) \Leftrightarrow \text{PDIM}(\mathcal{N}_\varepsilon(o)) \leq \lambda \wedge |\mathcal{N}^{\bar{\mathbf{w}}_o}_\varepsilon(o)| \geq \mu.$$

The notions of direct reachability, reachability and connectivity in DBSCAN [9] are now extended in order to take into consideration the notion of preferences.

DEFINITION 8 (DIRECT PREFERENCE REACHABILITY). *Let $\varepsilon, \delta \in \mathbb{R}$ and $\mu, \lambda \in \mathbb{N}$. A point $p \in \mathcal{D}$ is* directly preference weighted reachable *from a point $q \in \mathcal{D}$ w.r.t. ε, μ, δ, and λ (denoted by $\text{DIRREACH}^{\text{pref}}_{\text{den}}(q,p)$), if q is a preference weighted core point, the subspace preference dimensionality of $\mathcal{N}_\varepsilon(p)$ is at most λ, and $p \in \mathcal{N}^{\bar{\mathbf{w}}_q}_\varepsilon(q)$, formally:*

$\text{DIRREACH}^{\text{pref}}_{\text{den}}(q,p) \Leftrightarrow$

(1) $\text{CORE}^{\text{pref}}_{\text{den}}(q)$

(2) $\text{PDIM}(\mathcal{N}_\varepsilon(p)) \leq \lambda$

(3) $p \in \mathcal{N}^{\bar{\mathbf{w}}_q}_\varepsilon(q)$.

DEFINITION 9 (PREFERENCE REACHABILITY). *Let $\varepsilon, \delta \in \mathbb{R}$ and $\mu, \lambda \in \mathbb{N}$. A point $p \in \mathcal{D}$ is* preference weighted reachable *from a point $q \in \mathcal{D}$ w.r.t. ε, μ, δ, and λ (denoted by $\text{REACH}^{\text{pref}}_{\text{den}}(q,p)$), if there is a chain of points $p_1, \ldots, p_n$ such that $p_1 = q, p_n = p$ and p_{i+1} is directly preference weighted reachable from p_i, formally:*

$$\begin{aligned}&\text{REACH}^{\text{pref}}_{\text{den}}(q,p) \Leftrightarrow \\ &\quad \exists p_1, \ldots, p_n \in \mathcal{D} : p_1 = q \wedge p_n = p \wedge \\ &\quad \forall i \in \{1, \ldots, n-1\} : \text{DIRREACH}^{\text{pref}}_{\text{den}}(p_i, p_{i+1}).\end{aligned}$$

DEFINITION 10 (PREFERENCE CONNECTIVITY). *Let $\varepsilon, \delta \in \mathbb{R}$ and $\mu, \lambda \in \mathbb{N}$. A point $p \in \mathcal{D}$ is* preference weighted connected *to a point $q \in \mathcal{D}$, if there is a point $o \in \mathcal{D}$ such that both p and q are preference weighted reachable from o, formally:*

$$\begin{aligned}&\text{CONNECT}^{\text{pref}}_{\text{den}}(q,p) \Leftrightarrow \\ &\exists o \in \mathcal{D} : \text{REACH}^{\text{pref}}_{\text{den}}(o,q) \wedge \text{REACH}^{\text{pref}}_{\text{den}}(o,p).\end{aligned}$$

DEFINITION 11 (SUBSPACE PREFERENCE CLUSTER). *Let $\varepsilon, \delta \in \mathbb{R}$ and $\mu, \lambda \in \mathbb{N}$. A non-empty subset $\mathcal{C} \subseteq \mathcal{D}$ is called a* subspace preference cluster *w.r.t. ε, μ, δ, and λ, if all points in $\mathcal{C}$ are preference weighted connected and $\mathcal{C}$ is maximal w.r.t. preference weighted reachability, formally:*

$\text{CONSET}^{\text{pref}}_{\text{den}}(\mathcal{C}) \Leftrightarrow$
Connectivity: $\forall o, q \in \mathcal{C} : \text{CONNECT}^{\text{pref}}_{\text{den}}(o,q)$
Maximality: $\forall p, q \in \mathcal{D} : q \in \mathcal{C} \wedge \text{REACH}^{\text{pref}}_{\text{den}}(q,p) \Rightarrow p \in \mathcal{C}$.

As in DBSCAN, in PREDECON a cluster is uniquely determined by any of its preference weighted core points. As far as such a point is detected, we can discover the associated cluster by detecting all points that are preference weighted reachable from it. The pseudocode of the algorithm PREDECON is depicted in Figure 1.

```
algorithm PreDeCon(D, ε, μ, λ, δ)
  for each o ∈ D do
    compute the subspace preference vector w̄_o;
  for each unclassified o ∈ D do
    expandCluster(D,o, ε, μ, λ);

function expandCluster(D, o, ε, μ, λ)
    if CORE^pref_den(o) then      // expand a new cluster
      generate new clusterID;
      insert all x ∈ N^{w̄_o}_ε(o) into queue Φ;
      while Φ ≠ ∅ do
        q = first point in Φ;
        compute R = {x ∈ D | DIRREACH^pref_den(q, x)};
        for each x ∈ R do
          if x is unclassified then
            insert x into Φ;
          if x is unclassified or noise then
            assign current clusterID to x
        remove q from Φ;
    else      // o is noise
      mark o as noise;
end.
```

Figure 1: Pseudo code of the PreDeCon algorithm.

4. INCREMENTAL PREDECON

The goal of incremental PREDECON is to update the so far built clustering model (at timepoint $t-1$) as new data arrive at t, and thus to derive a valid clustering model at t.

The key observation is that the preference weighted core member property[1] of some object might change due to the update and as a result, the existing clustering model might also change, e.g., new clusters might arise, old clusters might be abolished etc. The challenge is to exploit the old clustering model at $t-1$ (both clusters and subspaces where these clusters exist) and to adjust only that part of it which is affected by the update at t. Due to the density based nature of the algorithm, such an adjustment is expected (although not ensured in general) to be restricted to some part of the clustering instead of the whole clustering.

We consider a dynamic environment where data are coming sequentially and the underlying clustering model is updated so as to be compatible with the data. Our methodology is lossless, that is the incrementally updated model at t (which is based on the clustering model at $t-1$ and on the data updates at t) is the same as the one we would obtain if we applied from scratch the traditional PREDECON algorithm over the accumulated dataset $\mathcal{D}_t$.

Due to the density based nature of PREDECON, a preference weighted cluster is determined uniquely by one of its preference weighted core points. So, the key idea for the

[1]For simplicity, we omit from now on the term preference weighted and refer to a preference weighted core point simply as a core point

incremental version of PreDeCon is to check whether the update operation affects the preference weighted core member property of some point. If a non-core point becomes core, new density connections might be established. On the other hand, if a core point becomes non-core, some density connections might be abolished. There is also another case in PreDeCon, when a core point remains core but under different preferences. Such a change might cause either the establishment of new connections or the abolishment of existing ones.

We first present the effect of the update on the core member property (Section 4.1) and on the cluster membership in general (Section 4.2). The reorganization of the old clustering starts with seed objects defined in Section 4.3. The actual reorganization is described in Section 4.4.

4.1 Effect on the core member property

The insertion of a point p *directly affects* the points that are in the ε-neighborhood of p, i.e., all those points $q \in \mathcal{D} : dist(p,q) \leq \varepsilon$. In particular, the neighborhood of q, $\mathcal{N}_\varepsilon(q)$, might be affected, since the newly inserted object p is now a member of this neighborhood. Since $\mathcal{N}_\varepsilon(q)$ might change, the variance of $\mathcal{N}_\varepsilon(q)$ along some dimension $A_i \in A$ might also change causing A_i to turn into a preferable/ non-preferable dimension. This might cause changes in the subspace preference dimensionality of q, $\text{PDim}(\mathcal{N}_\varepsilon(q))$. Also, the subspace preference vector of q, $\bar{\mathbf{w}}_q$, might change; this in turn, might result in changes in the preference weighted ε-neighborhood of q, $\mathcal{N}_\varepsilon^{\bar{\mathbf{w}}}(q)_q$.

As a result, the core member property of q might be affected. According to Definition 7, two conditions should be fulfilled in order for a point q to be core: In terms of condition i), the preference dimensionality of q must contain at most λ dimensions (i.e., $\text{PDim}(\mathcal{N}_\varepsilon(q)) \leq \lambda$). In terms of condition ii), the preference weighted ε-neighborhood of q should contain at least μ points (i.e., $|\mathcal{N}_\varepsilon^{\bar{\mathbf{w}}_q}(q)\,| \geq \mu$). So, we have to check how the update operation affects both these conditions for object q. Since the definition of the preference weighted ε-neighborhood of q (condition ii) relies on the subspace preference dimensionality of q (condition i), it is reasonable to start with the examination of the first condition.

Let p be the new point, and let $\mathcal{D}^* = \mathcal{D} \cup \{p\}$ be the new dataset after the insertion of p. As already stated, the addition of p, might affect the core member property of any object $q \in \mathcal{N}_\varepsilon(p)$. In particular, since $\mathcal{N}_\varepsilon(q)$ changes, the variance along some attribute $A_i \in A$, i.e., $\text{Var}_{A_i}(\mathcal{N}_\varepsilon(q))$ might also change.

- If A_i was a non-preferable dimension (that is, $\text{Var}_{A_i}(\mathcal{N}_\varepsilon(q)) > \delta$), it might either remain non-preferable (if still $\text{Var}_{A_i}(\mathcal{N}_\varepsilon(q)) > \delta$) or it might turn into preferable (if now $\text{Var}_{A_i}(\mathcal{N}_\varepsilon(q)) \leq \delta$).
- If A_i was a preferable dimension, it might either remain preferable (if still $\text{Var}_{A_i}(\mathcal{N}_\varepsilon(q)) \leq \delta$) or it might turn into non-preferable (if now $\text{Var}_{A_i}(\mathcal{N}_\varepsilon(q)) > \delta$).

A change in the preference of A_i might result in changes in the subspace preference vector of q, $\bar{\mathbf{w}}_q$, since some dimension might swap from preferable to non preferable and vice-versa. Thus, we can have more or less preferable dimensions comparing to the previous state (*quantitative differences*) or we can have the same dimensionality but under different preferred dimensions (*qualitative differences*). A change in $\bar{\mathbf{w}}_q$, might cause changes in both the subspace preference dimensionality of q, $\text{PDim}(\mathcal{N}_\varepsilon(q))$, and in the preferred neighborhood of q, $\mathcal{N}_\varepsilon^{\bar{\mathbf{w}}_q}(q)$.

If the subspace preference dimensionality of q, $\text{PDim}(\mathcal{N}_\varepsilon(q))$, changes the first condition of Definition 7 might be violated. In particular, if $|\text{PDim}(\mathcal{N}_\varepsilon(q))| > \lambda$, the point q cannot be core. So, if q was a core point, it now looses this property (*core* $\rightarrow$ *noncore*), whereas if it was a non-core it still remains non-core. This is the first condition to be checked, and it is quantitative since it is based on the number of preferred dimensions (whether they exceed δ or not).

If after the insertion of p, this condition holds (that is, $|\text{PDim}(\mathcal{N}_\varepsilon(q))| <= \lambda$) then we can proceed to the evaluation of the second condition of Definition 7 to check whether q is core after the update.

- If q was a core point, and now $|\mathcal{N}_\varepsilon^{\bar{\mathbf{w}}}(q)_q| < \mu$, then q looses its core member property (*core* $\rightarrow$ *noncore*). Otherwise, it remains core.
- If q was not a core point, and now $|\mathcal{N}_\varepsilon^{\bar{\mathbf{w}}}(q)_q| \geq \mu$ then q turns into core (*noncore* $\rightarrow$ *core*). Otherwise, it remains non core.
- There is also another case of change for q, where it still remains core (*core* $\rightarrow$ *core*) but under different preferences (this might happen e.g., when there are qualitative changes in $\bar{\mathbf{w}}_q$). Note that, although q might remain core its neighborhood might change due to different preferred dimensions.

To summarize, the possible effects of an insert operation in the core member property of an object are the following:

- core $\rightarrow$ noncore
- noncore $\rightarrow$ core
- core $\rightarrow$ core but under different preferences

Note that, as already presented, the objects with a changed core member property are all located in $\mathcal{N}_\varepsilon(p)$, since such a change is due to the insertion of p.

4.2 Affected objects

So far, we referred to the objects in $\mathcal{N}_\varepsilon(p)$ that are *directly affected* by the insertion of p and we presented when and how their core member property might change.

Note however, that a change in the core member property of an object q might cause changes in the objects that are preference weighted reachable from q. For example, if q was a core point before the insertion and it becomes non-core after the insertion, then any density connectivity that relied on q is destroyed. On the other hand, if q was a non-core

point before the insertion and it turns into core after the insertion, then some new density connectivity based on q might arise.

We denote by $\text{AFFECTED}_{\mathcal{D}}(p)$ the set of points in $\mathcal{D}$ that might be affected after the insertion of p. This set contains both directly affected points (those located in $\mathcal{N}_\varepsilon(p)$, which might change their core member property after the update) and indirectly affected objects (those that are density reachable by some point in $\mathcal{N}_\varepsilon(p)$, which might change their cluster membership after the update).

DEFINITION 12 (AFFECTED OBJECTS). *Let $\mathcal{D}$ be a dataset and let $\mathcal{D}^* = \mathcal{D} \cup \{p\}$ be the new dataset after the insertion of object p. We define the set of objects in $\mathcal{D}$ affected by the insertion of p as follows:*

$$\text{AFFECTED}_{\mathcal{D}}(p) = \mathcal{N}_\varepsilon(p) \quad \cup \quad \{q | \exists o \in \mathcal{N}_\varepsilon(p) : REACH^{pref}_{den}(o,q) \text{ in } \mathcal{D}^*\}$$

Any other object in the database is not affected by the specific update.

4.3 Seed objects for the update

The update of p might cause changes in the cluster membership of only some object $q \in \text{AFFECTED}_{\mathcal{D}}(p)$. A naive solution would be to reapply the static PREDECON over this set in order to obtain the new clustering model for the set of affected data. This way however, although we would restrict reclustering over only this subset of the data, we actually ignore any old clustering information for this set and build it from scratch.

Our solution is based on the observation that any changes in $\text{AFFECTED}_{\mathcal{D}}(p)$, are exclusively initiated by objects that change their core member property, i.e., those in $\mathcal{N}_\varepsilon(p)$. So, instead of examining all objects in $\text{AFFECTED}_{\mathcal{D}}(p)$, we can start searching from objects in $\mathcal{N}_\varepsilon(p)$ and "discover" the rest of the affected object on the road (those objects would belong to $\text{AFFECTED}_{\mathcal{D}}(p)$ though). Note also that there is no need to examine each $q \in \mathcal{N}_\varepsilon(p)$ since some objects might have not change their core member property so related density connections from the previous clustering would be still valid. So, we need to examine only those objects in $\mathcal{N}_\varepsilon(p)$ that change their core member property after the insertion of p, instead of all objects in $\mathcal{N}_\varepsilon(p)$, so as to avoid rediscovering density connections. As already described, a possible change in the core member property of an object after the insertion of p falls into one of the following cases: i) core $\rightarrow$ noncore, ii) noncore $\rightarrow$ core and, iii) core $\rightarrow$ core but under different preferences.

When the core member property of a point $q \in \mathcal{N}_\varepsilon(p)$ changes as described above, we should consider as seed points for the update any core point $q' \in \mathcal{N}_\varepsilon(q)$. That is, the update process starts from core points in the neighborhood of the objects with changed core member property (which, in turn are all located in $\mathcal{N}_\varepsilon(p)$).

DEFINITION 13 (SEED OBJECTS FOR THE UPDATE). *Let $\mathcal{D}$ be a dataset and let $\mathcal{D}^* = \mathcal{D} \cup \{p\}$ be the new dataset after the insertion of p. We define the seed objects for the update as:*

$$\text{UPDSEED} = \{q \mid q \text{ is core in } \mathcal{D}^*, \exists q' : q \in \mathcal{N}_\varepsilon(q') \text{ and } q' \text{ changes his core member property in } \mathcal{D}^*\}$$

```
algorithm incPreDeCon(D, INSERTS, ε, μ, λ, δ)
  for each p ∈ INSERTS do
    D = D ∪ p;
    compute the subspace preference vector w̄_p;
    // update preferred dimensionality and
    // check changes in the core member property in N_ε(p)
    for each q ∈ N_ε(p) do
        update w̄_q;
        check changes in the core member property of q;
        if change exists, add q to AFFECTEDCORE;
    compute UPDSEED based on AFFECTEDCORE
    for each q ∈ UPDSEED do
        expandCluster(D,q, ε, μ, λ);
  end;
```

Figure 2: Pseudo code of the incPreDeCon algorithm.

4.4 Updating the model

After the insertion of a new object p new density connections might be established whereas existing connections might be abolished or change. We can detect these changes starting with the seed objects in UPDSEED. The *expandCluster*() procedure of PREDECON is then invoked starting from objects in UPDSEED and considering the results of the so far built clustering. The pseudocode of the algorithm is displayed in Figure 2. After the insertion of a point p, its subspace preference vector is computed and the subspace preference vectors of objects in its neighborhood $\mathcal{N}_\varepsilon(p)$ are updated. For each of these neighborhood points, we examine whether some change in the core member property has occurred as a result of the insertion of p. Next, we derive the seed objects for the update and start the re–organization of the old clustering, which involves some call to the *expandCluster*() function of PREDECON.

This is a generic solution that works on every effect caused by the update of p, i.e., objects turning into core or/and the opposite. Of course, there are simpler cases where we can deal with the update without invoking the *expandCluster*() procedure of PREDECON. For example, if the update of p does not affect the core member property of its neighborhood and its neighborhood belongs to exactly one cluster before the update, then p is also added to this cluster (absorption). However there are many such special cases, since as already stated the update of p might cause both destroy of old density connections and creation of new density connections depending on the changes in the core member property of its neighborhood. So, we opt for the generic solution.

5. EXPERIMENTS

In this section, we present early-stage results to evaluate the efficiency of INCPREDECON versus PREDECON. Note that the incremental version produces the same results as the static version when applied over the accumulated dataset, so there is no need for quality evaluation. So, we compare the performance of INCPREDECON versus the performance

of PreDeCon in terms of the range queries required by each version, since these are the only operations requiring page accesses.

PreDeCon: The preferred dimensionality vector for each object has to be computed; such a computation is based on the result of a range query and thus, totally $|\mathcal{D}|$ range queries are required for this preprocessing phase. Also, checking the core member property and expanding a preference weighted cluster requires for each object the evaluation of a range query, thus totally $|\mathcal{D}|$ range queries are required for clustering. So, the static version of the algorithm requires $2 \times |\mathcal{D}|$ range queries in total, where $|\mathcal{D}|$ is the size of the database.

incPreDeCon: The initialization of the model, which takes place once requires $2 \times |\mathcal{D}|$ range queries as described above. When a new point p is inserted, the preferred dimensionality vector of p should be computed and also the preferred dimensionality vectors of any point $q \in \mathcal{N}_\varepsilon(p)$ should be updated, this requires $1 + |\mathcal{N}_\varepsilon(p)|$ range queries. Next, objects in $\mathcal{N}_\varepsilon(p)$ should be examined for any change in their core member property, this requires $\mathcal{N}_\varepsilon(p)|$ range queries. Finally, the reorganization of the old clustering starting from the objects in UpdSeed requires range queries, which in the worst case equal to the number of $\text{Affected}_{\mathcal{D}}(p)$. So, the maximal number of range queries after each insert is

$$1 + 2 \times |\mathcal{N}_\varepsilon(p)| + |\text{Affected}_{\mathcal{D}}(p)|.$$

We evaluated incPreDeCon using 3 synthetic datasets consisting of $|\mathcal{D}_1| = 1.000$, $|\mathcal{D}_2| = 5.000$, $|\mathcal{D}_3| = 10.000$ points. For each dataset, we performed 100 random inserts and compared the number of range queries required by PreDeCon and incPreDeCon. The results for $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_2$ are depicted in Figures 3, 4, 5 respectively. As it seems from these figures incPreDeCon outperforms PreDeCon in all datasets. The speed up factor achieved by incPreDeCon after all the insertions is about 22, 29 and 36 for datasets respectively.

From the above results, it seems that incPreDeCon outperforms PreDeCon in terms of the required range queries. As the dataset size increases we see that the speed up factor increases, which is reasonable since PreDeCon has to do reclustering from scratch over the new bigger database, whereas incPreDeCon exploits the so far build clustering model and only reorganizes the affected part of the old clustering instead of the whole clustering.

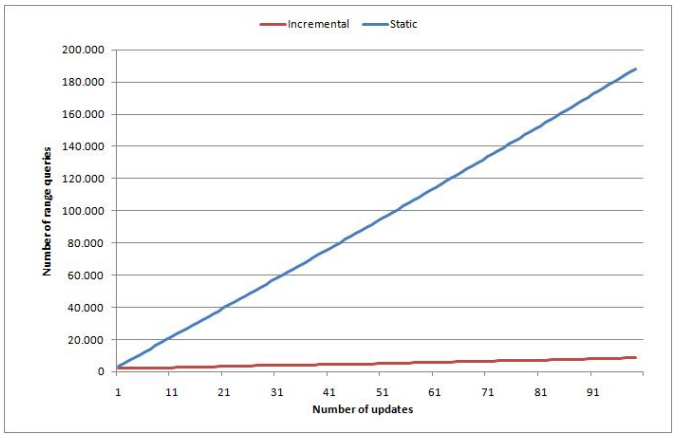

Figure 3: Comparing range queries for $\mathcal{D}_1$

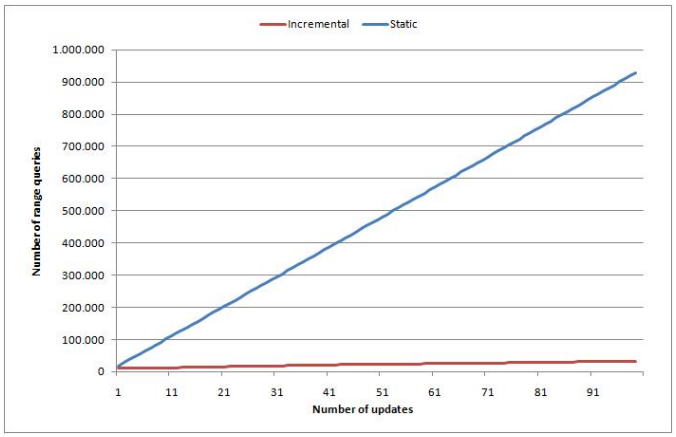

Figure 4: Comparing range queries for $\mathcal{D}_2$

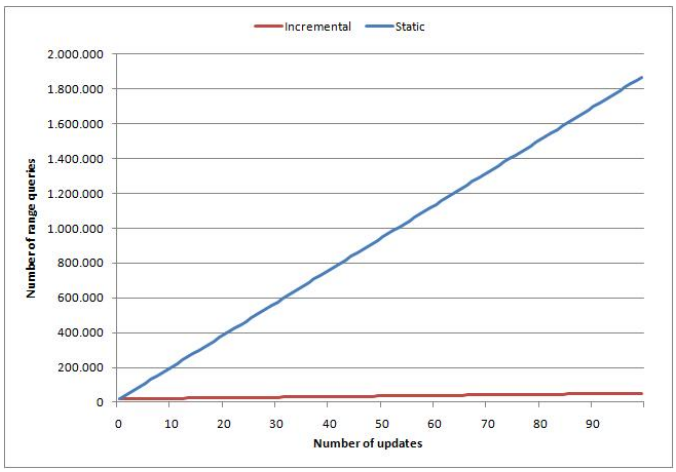

Figure 5: Comparing range queries for $\mathcal{D}_3$

6. DISCUSSION

Although the first experimental results are promising, further experimentation should be done considering larger datasets and real world datasets as well. Also, we should experiment with the number of dimensions and the number of generated clusters and see how the performance of both incPreDeCon and PreDeCon is affected. Anyway, this is an early-stage work that is also intended to draw the attention of the research community to the problem of clustering high dimensional data streams and initiate discussions for research challenges and novel solutions.

So far, we presented the effect of an insertion on the resulting clustering model. A deletion might also occur, which as

with the insertion case might result in both the creation of new density based connections and the abolishment of existing ones. This is due to the fact that the deletion might change the preferred dimensions of a point and thus the conditions of Definition 7 might now be fulfilled or not. What we described for the insertion case in Section 4 also holds for the deletion case. In particular, the deletion of an object might affect the core member property in its neighborhood as described in Section 4.1, whereas the general affected objects are those described in Section 4.2. The reorganization of the clusters should be initiated by seed objects defined in Section 4.3, whereas the reorganization process is as described in Section 4.4. However, we should run experiments with deletes also so as to evaluate the performance of the proposed method.

Furthermore, in this work we considered only single update operations. We are investigating the case of batch updates; the idea is to treat the effects of all these updates together instead of treating each update independently. The rationale is that the batch might contain updates that are related to each other (e.g., one update belongs to the neighborhood of the other). Consider for example, news data: when a story arises usually within a small time interval there exists a "burst" of news articles all referring to this story. In case of batch updates, both PreDeCon would perform since reclustering would take place once the batch is processed and also incPreDeCon would perform better since the update of the preference weighted vectors of the affected points, changes in the core member property and cluster re-organization would take place once considering the accumulative effect of all updates in the batch. Again, experiments are required in order to evaluate the performance improvement of our method.

7. CONCLUSIONS

In this paper, we presented the first incremental subspace clustering algorithm, based on the algorithm PreDeCon. The update strategy, exploits the density based nature of clusters managing to restructure only that part of the old clustering that is affected by the update. Our initial experimental results demonstrate the efficiency of the proposed method against static PreDeCon. Further evaluation experiments are already described in Section 6 and those comprise part of our ongoing work.

The incPreDeCon method described here is suitable for slowly changing environments, where we also have access to the raw data. Data Warehouses are such an application where the data and models are maintained at different levels of granularity and are updated according to changes in the data on a regular basis, e.g., every night. As a next step, we are going to examine dynamic subspace clustering over fast changing dynamic environments and data streams. Since, having access to raw data is not the case for such kinds of environments, one has to consider some condensed structure for the summarization of the so far built subspace clustering model and use this summary as the basis for model update.

Also, except for the model update problem, which we investigate in this work, it is also important to track changes with respect to the old clustering (e.g., some new cluster might arise, two or more old clusters might merge into a new single cluster or a cluster might change orientation). Finding such kind of changes is challenging in these settings, since a cluster is now defined not only in terms of its members but also in terms of its preferred dimensions. Defining and efficiently detecting subspace cluster changes comprises part of our future work.

Finally, there are several methods for subspace clustering in correspondence to the traditional clustering methods. In this work, we deal with a specific density based subspace clustering algorithm. It will be interesting to investigate other methods as well, and find some unified framework for turning these methods into dynamic methods.

8. REFERENCES

[1] C. C. Aggarwal. On change diagnosis in evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):587–600, 2005.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for projected clustering of high dimensional data streams. In *VLDB*, pages 852–863, 2004.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD Record*, 27(2):94–105, 1998.

[5] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *SIGMOD*, pages 49–60, 1999.

[6] C. Bohm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *ICDM*, pages 27–34, 2004.

[7] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

[8] C.-Y. Chen, S.-C. Hwang, and Y.-J. Oyang. An incremental hierarchical data clustering algorithm based on gravity theory. In *PAKDD*, pages 237–250, 2002.

[9] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, pages 323–333, 1998.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[11] V. Ganti, J. Gehrke, and R. Ramakrishnan. Demon: Mining and monitoring evolving data. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):50–63, 2001.

[12] J. Gao, J. Li, Z. Zhang, and P.-N. Tan. An incremental data stream clustering algorithm based on dense units detection. In *PAKDD*, pages 420–425, 2005.

[13] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: you only get one look a tutorial. In *SIGMOD*, pages 635–635, 2002.

[14] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data*

Engineering, 15(3):515–528, 2003.

[15] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann Publishers Inc., 2000.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computer Surveys*, 31(3):264–323, 1999.

[17] H.-P. Kriegel, P. Kröger, and I. Gotlibovich. Incremental optics: Efficient computation of updates in a hierarchical cluster ordering. In *DaWaK*, pages 224–233, 2003.

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

[19] H. Yang, S. Parthasarathy, and S. Mehta. A generalized framework for mining spatio-temporal patterns in scientific data. In *KDD*, pages 716–721, 2005.

[20] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.