# Detecting Projected Outliers in High-dimensional Data Streams

Ji Zhang[1], Qigang Gao[2], Hai Wang[3], Qing Liu[1], Kai Xu[1]

[1]CSIRO ICT Center, Hobart, TAS, Australia
{*ji.zhang, q.liu, kai.xu*}*@csiro.au*
[2]Dalhousie University, Halifax, NS, Canada
*qggao@cs.dal.ca*
[3]Saint Mary's University, Halifax, NS, Canada
*hwang@smu.ca*

**Abstract.** In this paper, we study the problem of projected outlier detection in high dimensional data streams and propose a new technique, called **S**tream **P**rojected **O**uliter de**T**ector (SPOT), to identify outliers embedded in subspaces. Sparse Subspace Template (SST), a set of subspaces obtained by unsupervised and/or supervised learning processes, is constructed in SPOT to detect projected outliers effectively. Multi-Objective Genetic Algorithm (MOGA) is employed as an effective search method for finding outlying subspaces from training data to construct SST. SST is able to carry out online self-evolution in the detection stage to cope with dynamics of data streams. The experimental results demonstrate the efficiency and effectiveness of SPOT in detecting outliers in high-dimensional data streams.

## 1 Introduction

Outlier detection is an important research problem in data mining that aims to find objects that are considerably dissimilar, exceptional and inconsistent with respect to the majority data in an input database [10]. In recent years, we have witnessed a tremendous research interest sparked by the explosion of data collected and transferred in the format of streams. Outlier detection in data streams can be useful in many fields such as analysis and monitoring of network traffic data, web log, sensor networks and financial transactions, etc.

A key observation that motivates our work is that outliers existing in high-dimensional data streams are embedded in some lower-dimensional subspaces. Here, a subspace refers to as the data space consisting of a subset of attributes. These outliers are termed *projected outliers* in the high-dimensional space. The existence of projected outliers is due to the fact that, as the dimensionality of data goes up, data tend to become equally distant from each other [1]. As a result, the difference of data points' outlier-ness will become increasingly weak and thus undistinguishable. Only in moderate or low dimensional subspaces can significant outlier-ness of data be observed.

The problem of detecting projected outliers from high-dimensional data streams can be formulated as follows: given a $\varphi$-dimensional data stream $\mathcal{D}$, for each data point $p_i = \{p_{i1}, p_{i2}, \ldots, p_{i\varphi}\}$ in $\mathcal{D}$, the projected outlier detection method performs a mapping as $f : p_i \rightarrow (b, S_i, Score_i)$. $b$ is a Boolean variable indicating whether or not $p_i$ is a projected outlier. $p_i$ is a project outlier (*i.e.*, $b = true$) if there is one or more subspaces where $p_i$ is an outlier. These subspaces are called the outlying subspaces of $p_i$. In this case, $S_i$ is the set of outlying subspaces of $p_i$ and $Score_i$ is the corresponding outlier-ness score of $p_i$ in each subspace of $S_i$. The users have the discretion to pick up the top $k$ projected outliers that have the highest outlier-ness. In contrast, the traditional definition of outliers does not explicitly present outlying subspaces of outliers in the final result as outliers are detected in the full or a pre-specified data space that is known to users before outliers are detected.

Detecting projected outliers in high-dimensional data streams is a nontrivial research problem. There are two major challenges for us to tackle this problem. First, finding the right outlying subspaces for projected outliers is crucial to detection performance of the algorithm. However, the number of possible subspaces increases dramatically when the data

dimensionality goes up. Thus, finding the outlying subspaces of the data through an exhaustive search is infeasible for high-dimensional data streams. Second, data stream outlier detection algorithms are restricted to one pass over the data stream and thus need to process data in an incremental, online and real-time paradigm.

In this paper, we present a new technique, called Stream Projected Outlier deTector (SPOT), to approach the problem of outlier detection in high-dimensional data streams. The major technical contributions of this paper can be summarized as follows:

- First, this paper formally formulates the problem of projected outlier detection for high-dimensional data streams. To the best of our knowledge, there has been little research work in literature that targets this research problem;
- Second, SPOT constructs the novel Sparse Subspace Template (SST) to detect projected outliers. SST consists of a number of mutually supplemented subspace groups that contribute collectively to an effective detection of projected outliers. SPOT is able to perform supervised and/or unsupervised learning to construct SST, providing a maximum level of flexibility to users. The strategy of self-evolution of SST has also been incorporated into SPOT to greatly enhance its adaptability to dynamics of data streams;
- Third, unlike most other outlier detection methods that measure outlier-ness of data points based on a single criterion, SPOT adopts a more flexible framework allowing for the use of multiple measures for outlier detection. Employing multiple measures is generally more effective than a single measure. SPOT utilizes the Multi-Objective Genetic algorithm (MOGA) as an effective search method to find subspaces that are able to optimize these criteria for constructing SST;
- We show that SPOT is efficient and outperforms the existing methods in terms of effectiveness through experiments on both synthetic and real-life data sets.

The rest of this paper is organized as follows. The basic concepts and definitions used in SPOT will be presented in Section 2. In Section 3, we dwell on algorithms of SPOT, with an emphasis on the learning and detection stages of SPOT. Experimental results of SPOT are reported in Section 4. The final section concludes this paper.

## 2 Concepts and Definitions

### 2.1 Data Space Partitioning

To facilitate the quantification of data synopsis for outlier detection, a hypercube is superimposed and equi-width partition of domain space is performed. Each attribute of the data is partitioned into a few number of non-overlapping equal-length intervals. The cells in hypercube can be classified into two categories, *i.e.*, the base and projected cells. A *base cell* is a cell in the full data space (with the finest granularity in hypercube). The dimensionality (*i.e.*, number of attributes) of a base cell is equal to $\varphi$, where $\varphi$ is the dimension of the data stream. A *projected cell* is a cell that exists in a particular subspace. The dimensionality of a projected cell is smaller than $\varphi$. For example, if each dimension of a 3-dimensional data space is divided into 3 intervals, then there are 27 base cells, but only 9 projected cells in all 2-dimensional subspaces and 3 projected cells in all 1-dimensional subspaces.

Using the equi-width partition method is advantageous because it is not influenced by the distribution of data and it is unnecessary to update the space partition in the whole data processing process. In contrast, the equi-depth and V-optimal partition methods are not suitable in processing data streams because they need to update the partition frequently when data continuously arrive to ensure their partition criteria are met and, in the worst, such update is rather difficult due to a lack of sufficient data summaries required in the space re-partition.

### 2.2 Data Synopsis

Based on the hypercube structure, we employ two data synopsis, called *Base Cell Summary* (BCS) and *Projected Cell Summary* (PCS), to capture the major underlying characteristics of the data stream for detecting projected outliers. They are defined as follows.

**Definition 1**. **Base Cell Summary (BCS)**: The Base Cell Summary of a base cell $c$ in the hypercube is a triplet defined as $BCS(c) = \{D_c, \vec{LS}_c, \vec{SS}_c)$, where $D_c$, $\vec{LS}_c$ and $\vec{SS}_c$ denote the number of points in $c$, the sum and squared sum of data values in each dimension of points in $c$, respectively, *i.e.*, $\vec{LS}_c = \sum \vec{p}_i$ and $\vec{SS}_c = \sum \vec{p}_i^2$, for $p_i$ located in $c$, $1 \leq i \leq \varphi$.

BCS features two desirable properties, *i.e.*, additivity and incremental maintainability [22], that can be used to compute data synopsis for projected cells in subspaces.

**Definition 2**. **Projected Cell Summary (PCS)**: The Projected Cell Summary of a cell $c$ in a subspace $s$ is a triplet defined as $PCS(c, s) = (RD, IRSD, IkRD)$, where $RD$, $IRSD$ and $IkRD$ are the Relative Density, Inverse Relative Standard Deviation and Inverse k-Relative Distance of data points in $c$ of $s$, respectively.

$RD$, $IRSD$ and $IkRD$ are three effective measures to represent the overall data sparsity of each projected cell from different perspectives. They are used together in SPOT to achieve a good measurement of data outlier-ness. They are all defined as ratio-type measures in order to achieve statistical significance in measurement. The outlier-ness thresholds defined based on ratio-type measures are also intuitive and easy to specify.

**Definition 3**. **Relative Density (RD):** Relative Density of a cell $c$ in subspace $s$ measures the relative density of $c$ w.r.t the expected level of density of non-empty cells in $s$. If the density of $c$ is significantly lower than the average level of cell density in the same subspace, then the data in $c$ can be labeled as outliers. RD is calculated as $RD(c, s) = \frac{D_c}{E(D_s)}$, where $D_c$ and $E(D_s)$ represent the density (*i.e.*, number of points) in $c$ and the expected density of all the cells in $s$. Since $E(D_s) = \frac{N}{\delta^{|s|}}$, where $N$ corresponds to the effective stream length (the decayed total number of data points at a certain time), thus, $RD(c, s) = \frac{D_c \cdot \delta^{|s|}}{N}, p \in c$.

**Definition 4**. **Inverse Relative Standard Deviation (IRSD):** Inverse Relative Standard Deviation of a cell $c$ in subspace $s$ is defined as inverse of the ratio of standard deviation of $c$ in $s$ against the expected level of standard deviation of non-empty cells in $s$. Under a fixed density, if the data in a cell features a remarkably high standard deviation, then the data are distributed more sparsely in the cell and the overall outlier-ness of data in this cell is high. $IRSD(c, s)$ is computed as $IRSD(c, s) = \left[\frac{\sigma_c}{E(\sigma_s)}\right]^{-1}$, where $\sigma_c$ denotes the standard deviation of $c$ and $E(\sigma_s)$ denotes the expected standard deviation of cells in subspace $s$. Since $\sigma_c$ is larger than 0 but does not exceed the length of the longest diagonal of the cell in subspace $s$, which is $\sqrt{|s|}l$, where $|s|$ is the dimensionality of $s$ and $l$ is the side length of each interval, thus $E(\sigma_s)$ can be estimated as $E(\sigma_s) = \frac{0+\sqrt{|s|}l}{2} = \frac{\sqrt{|s|}l}{2}$.

**Definition 5**. **Inverse *k*-Relative Distance (IkRD):** Inverse *k*-Relative Distance for a cell $c$ in a subspace $s$ is the inverse of ratio of the distance between the centroid of $c$ and its nearest representative points in $s$ against the average level of such distance in $s$ for all the non-empty cells. A high IkRD value of $c$ indicates that $c$ is noticeably far from the dense regions of the data in $s$, thus the outlier-ness of data in $c$ is high. The IkRD is defined as $IkRD(c, s, k) = \left[\frac{k\_dist(c,s)}{average\_k\_dist(c_i,s)}\right]^{-1}$, $k\_dist(c, s)$ is the sum of distances between the centroid of $c$ and its $k$ nearest representative points in $s$ and $average\_k\_dist(c_i, s)$ is the average level of $k\_dist(c_i, s)$ for all the non-empty cells in $s$.

**Decaying Function.** In response to the possible concept drift of data streams, we use *exponential decaying function* in calculating data synopsis at different time. Suppose that the latest snapshot of data synopsis $\mathcal{M}$ of a cell $c$ ($c$ can either be a base cell or a projected cell) is created at time $T$, then we can update $\mathcal{M}$ of $c$ when a new data $p$ arrives at $c$ at the time $T'$ ($T' \geq T$) as follows: $\mathcal{M}(c)^{T'} = df(T' - T, t)\mathcal{M}(c)^T + \mathcal{M}(p)$, where $df()$ is the decayed function that is defined as $df(T' - T, t) = e^{-\frac{\alpha(T'-T)}{t}}$, $\alpha$ is the decaying coefficient that is used to adjust the speed of weighted decay, and $t$ is the basic time unit used for scaling elapsed time.

# 3 Stream Projected Outlier Detector (SPOT)

## 3.1 An Overview of SPOT

An overview of SPOT is presented in Figure 1. SPOT can be broadly divided into two stages: the learning stage and the detection stage. SPOT can further support two types of learning, namely offline learning and online learning. In the offline learning, Sparse Subspace Template (SST) is constructed using either the unlabeled training data (*e.g.*, some available historic data) and/or the labeled outlier examples provided by domain experts. SST is a set of subspaces that features a higher data sparsity/outlier-ness than others. It casts light on where projected outliers are likely to be found in the high-dimensional space. SST consists of three groups of subspaces, *i.e.*, Fixed SST Subspaces ($\mathcal{FS}$), Unsupervised SST Subspaces ($\mathcal{US}$) and Supervised SST Subspaces ($\mathcal{SS}$), where $\mathcal{FS}$ is a compulsory component of SST while $\mathcal{US}$ and $\mathcal{SS}$ are optional components. SST is mainly constructed in an unsupervised manner where no labeled examples are required. However, it is possible to use the labeled outlier exemplars to further improve SST. As such, SPOT is very flexible and is able to cater for different practical applications that may or may not have available labeled exemplars. Multiobjective Genetic Algorithm (MOGA) is used for outlying subspace search in constructing $\mathcal{US}$ and $\mathcal{SS}$.

When SST is constructed, SPOT can start to screen projected outliers from constantly arriving data in the detection stage. The arriving data will be first used to update the data summaries (*i.e.*, PCSs) of the cell it belongs to in each subspace of SST. This data will then be labeled as a projected outlier if PCS values of the cell where it belongs to are lower than some pre-specified thresholds. The detected outliers are archived in the so-called Outlier Repository. Finally, all or only a specified number of the top outliers in Outlier Repository will be returned to users when the detection stage is finished.

During the detection stage, SPOT can perform online learning periodically. The online learning involves updating SST with new sparse subspaces that SPOT finds based on the current data characteristics and the newly detected outliers. Online learning improves SPOT's adaptability to dynamic of data streams.

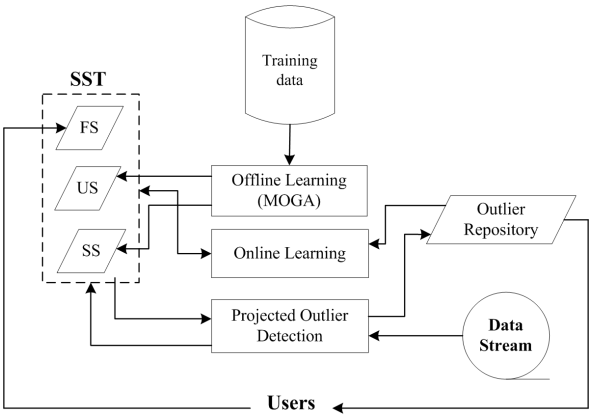


**Fig. 1.** An overview of SPOT

## 3.2 Learning Stage of SPOT

Since the number of subspaces grows exponentially with regard to data dimensionality, evaluating each streaming data in each possible subspace becomes prohibitively expensive. As such, we only evaluate each point in SST alternatively, in an effort to render this problem tractable. The central task of the offline learning stage is to construct SST.

**Construction of SST.** It is desired that SST can contain one or more outlying subspaces for as many projected outliers in the streams as possible. To do this, SST is designed to contain a few groups of subspaces that are generated by different underlying rationales. Different subspace groups supplement each other towards capturing the right subspaces where projected outliers are hidden. This helps enable SPOT to detect projected outliers more effectively. Specifically, SST contains the following three subspace groups, *Fixed SST Subspaces ($\mathcal{FS}$)*, *Unsupervised SST Subspaces ($\mathcal{US}$)* and *Supervised SST Subspaces ($\mathcal{SS}$)*, respectively.

• *Fixed SST Subspaces ($\mathcal{FS}$)*

Fixed SST Subspaces ($\mathcal{FS}$) contains all the subspaces in the full lattice whose maximum dimension is $MaxDimension$, where $MaxDimension$ is a user-specified parameter. In other words, $\mathcal{FS}$ contains all the subspaces with dimensions of 1, 2, $\cdots$, $MaxDimension$. $\mathcal{FS}$ satisfies that $\forall s,$ we have $|s| \leq MaxDimension$ if and only if $s \in \mathcal{FS}$. Construction of $\mathcal{FS}$ does not require any learning process.

• *Unsupervised SST Subspaces ($\mathcal{US}$)*

Unsupervised SST Subspaces ($\mathcal{US}$) are constructed through an unsupervised offline learning process. We assume that a set of historical data is available for this unsupervised learning. All the training data are scanned and assigned into one (and only one) cell in the hypercube. The BCS of each occupied cell in $H$ are maintained during this data assignment process. Multi-Objective Genetic Algorithm (MOGA) is then applied on the whole training data to find the subspaces that feature a higher number of outliers. These subspaces will be added to $\mathcal{US}$.

Once we have obtained the initial $\mathcal{US}$, we can further procure more useful subspaces for $\mathcal{US}$. We can find the outlying subspaces of the top training data that have the highest overall outlying degree. The selected training data are more likely to be considered as outliers and they can be potentially used to detect more similar outliers in the stream. The overall outlying degree of the training data is computed in an unsupervised manner by employing clustering analysis.

As the distance between two data points may vary significantly in different subspaces, we therefore expect the distance metric used in the clustering to be able to well reflect the overall distance of data points in difference subspaces, especially those where outliers are likely to be detected. To achieve this, we employ a novel distance metric, called *Outlying Distance(OD)*, for clustering training data. It is defined as the average distance between two points in top sparse subspaces of the whole training data obtained by MOGA, *i.e.*, $OD(p_1, p_2) = \frac{\sum_{i=1}^{m} dist(p_1, p_2, s_i)}{m}$, where $m$ is the number of top sparse subspaces returned by MOGA and $s_i$ is the $i^{th}$ subspaces in this set.

We utilize *lead clustering method*, also called the fixed-width clustering, to cluster the whole training data into a few clusters. Lead clustering method is a highly efficient clustering method that adopts an incremental paradigm to cluster data. For each data $p$ in the data set that has yet been clustered, it will be assigned to the cluster $c'$ such that $OD(p, c') < d_c$ and $\forall c_i \neq c', OD(p, c') \leq OD(p, c_i)$. The centriod of $c'$ will be updated upon the cluster assignment of $p$. If $\forall c_i$, we have $OD(p, c_i) \geq d_c$, then a new cluster is initiated and $p$ becomes the centroid of this new cluster. These steps will be repeated until all the data in the data set have been clustered.

Due to its incremental nature, lead clustering method features a promising linear scalability with regard to number and dimensions of training data. However, its result is sensitive to the order in which the training data are clustered. To solve this problem, we perform lead clustering in multiple runs under different data orders. Even though an outlier may be assigned into different clusters in different runs, the chance that it is assigned to a small cluster is relatively high. Hence, the outlying degree of a training data can be measured by the average size of clusters it belongs to in different runs.

• *Supervised SST Subspaces ($\mathcal{SS}$)*

In some applications, a small number of outlier exemplars may be provided by domain experts or are available from some earlier detection process. These outlier exemplars can be considered as carriers of domain knowledge that are potentially useful to improve SST for better a detection effectiveness. MOGA is applied on each of these outlier exemplars to

**Algorithm: SPOT_Unsupervised_Learning ($D_T, d_c, N_{runs}$)**

**Input:** Training data $D_T$, clustering distance threshold $d_c$, number of clustering runs $N_{runs}$;
**Output:** Unsupervised SST subspaces;
1. $US \leftarrow \emptyset$;
2. **MOGA**($D_T$);
3. $US \leftarrow US \cup$ top sparse subspaces of $D_T$;
4. FOR $i$=1 to $N_{runs}$ DO
5. **Cluster**($D_T$, $US$, $d_c$);
6. **ComputeOF**($D_T$);
7. FOR each top training data $p$ DO {
8. **MOGA**($p$);
9. $US \leftarrow US \cup$ top sparse subspaces of $p$; }
10. $SST \leftarrow US$;
11. **Return**($SST$);

**Fig. 2.** Unsupervised learning algorithm of SPOT

find their top sparse subspaces. There subspaces are called Supervised SST Subspaces ($\mathcal{SS}$). Based on $\mathcal{SS}$, example-based outlier detection [23] can be performed that effectively detects more outliers that are similar to these outlier examples.

**Remarks:** $\mathcal{FS}$, $\mathcal{US}$ and $\mathcal{SS}$ differ in their respective role they play in projected outlier detection. $\mathcal{FS}$ is deterministic and it tries to establish an effectiveness *bottomline* for projected outlier detection without performing any learning process. Definitely, $\mathcal{FS}$ is able to detect the projected outliers as long as one of their respective outlying subspaces falls into the $MaxDimension$-dimensional space lattice. Due to an exponential growth in size of this space lattice, $MaxDimension$ is normally set quite small, say 3 or 4. As such, the detecting capability of $\mathcal{FS}$ is limited. It cannot detect those projected outliers if the dimensionality of all their outlying subspaces exceed $MaxDimension$. In contrast, $\mathcal{US}$ and $\mathcal{SS}$ are randomized by nature and do not subject to the dimensionality constraint. $\mathcal{US}$ and $\mathcal{SS}$ can help detect more subsequent outliers that share similar characteristics of the top outlying training data or the outlier exemplars provided by human users. They can supplement $\mathcal{FS}$ to detect projected outliers whose outlying subspace are not located in the $MaxDimension-$dimensional space lattice.■

**Computing PCS of a Projected Cell.** In MOGA, PCS of projected cells need to be computed. The central components required for quantifying RD, IRSD, IkRD of PCS for a projected cell $c$ are the density ($D_c$), mean ($\mu_c$) and standard deviation ($\sigma_c$) of data points in $c$ and the representative points in the subspace where $c$ exists. Fortunately, as shown in [22], $D_c$, $\mu_c$ and $\sigma_c$ can be efficiently obtained using the information maintained in BCS. Thus, we only show how representative points can be obtained for computing IkRD here. The representative points of a subspace $s$ are the centroids of a selected set of non-empty cells in $s$. This selected set of cells are termed *coverage cells*. A rule-of-thumb in selecting coverage cells in a subspace $s$ is to select a specified number of the most populated cells in $s$ such that they cover a majority of data in $s$, such as 90%. It is noted that the initial representative data for different subspaces are obtained offline before outlier detection is conducted, thus it is not subject to the one-pass scan and time-criticality requirements of data stream applications.

**Learning Algorithms.** Figure 2 presents the unsupervised learning algorithm of SPOT. Steps 4-5 perform lead clustering of the training data $D_T$. The distance metric used in the clustering, Outlying Distance (OD), is based on the top sparse subspaces of the whole training data obtained by MOGA in Step 2 and 3. Step 6 quantifies the Outlying Factor of each training data. Step 7-9 try to find the sparse subspaces for top training data that have highest Outlying Factor. MOGA is applied on each of top training data to find their respective top sparse subspaces. $\mathcal{US}$ is added to SST and returned to users in Step 10-11. The algorithm for the supervised learning, which is simpler than that of the unsupervised learning, is presented in Figure 3. Step 2-4 find the sparse subspaces for each outlier exemplar using MOGA. $\mathcal{SS}$ is added to SST and returned to users in Step 5-6.

**Algorithm: SPOT_Supervised_Learning ($OE$)**

**Input:** Set of outlier examplars $OE$;
**Output:** Supervised SST subspaces;
1. $SS \leftarrow \emptyset$;
2. FOR each outlier examplar $o$ in $OE$ DO {
3. **MOGA**($o$);
4. $SS \leftarrow SS \cup$ top sparse subspaces of $o$; }
5. $SST \leftarrow SS$;
6. **Return**($SST$);

**Fig. 3.** Supervised learning algorithm of SPOT

### 3.3 Multi-Objective Genetic Algorithm (MOGA)

In the offline learning stage, we employ Multi-Objective Genetic Algorithm (MOGA) to search for subspaces whose RD, IRSD and IkRD objectives can be minimized in construction of SST. MOGA conducts search of good subspaces through a number of generations each of which has a population containing a specific number of individuals (*i.e.*, subspaces). The subspaces in the first generation are typically generated randomly, while the subspaces in the subsequent generations are generated by applying search operators such as crossover and mutation on those subspaces in their preceding generation. In a multi-objective minimization problem, subspaces in the population of each generation can be positioned on different trade-off surfaces in the objective function space. The subspaces located on a surface closer to the origin is better than the one far from the origin. The superiority (or inferiority) of the subspaces on the same surface are not distinguishable. The surface where the optimal subspaces are located is called *Pareto Front*. Figure 4 shows an example of two surfaces when there are only two objective functions. The goal of MOGA is to gradually produce an increasing number of optimal subspaces, located in the Pareto Front, from non-optimal subspaces as evolution proceeds. MOGA provides a good general framework for dealing with multi-objective search problems. However, we still need to perform ad-hoc design of MOGA in SPOT for outlying subspace search, including individual representation, objective functions, fitness function, selection scheme and elitism.

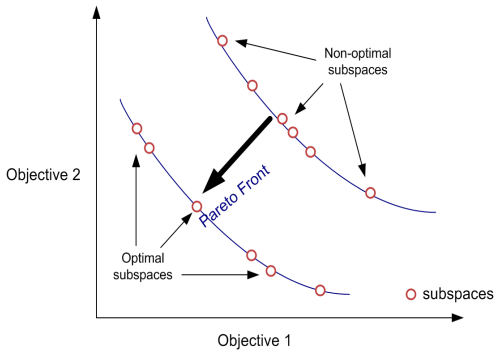


**Fig. 4.** Optimal and non-optimal subspaces in MOGA

**Individual representation**. In SPOT, we adopt the straightforward yet effective binary representation scheme for subspaces; all individuals are represented by binary strings with fixed and equal length $\varphi$, where $\varphi$ is the number of dimensions of the dataset. Each bit in the individual will take on the value of "0" and "1", respectively, indicating whether or not its corresponding dimension is selected for a particular subspace.

**Objective Functions**. We need to define objective functions for subspaces w.r.t two types of data. The first type of data is *a single data point*. This applies to each top training data

and each outlier exemplar. For a single data point, we have $f(p,s) = f(c,s)$, meaning that the objective function of subspace $s$ w.r.t data point $p$ is the data sparsity (measured by RD, IRSD, IkRD) of the cell $c$ in $s$ where $p$ belongs to. The second type of data is *the whole training data* $D_t$. The objective function of $s$ for $D_t$ is defined as the percentage of data points in $D_t$ with low PCS in $s$. Its calculation involves summing up the densities of projected cells in $s$ with lower PCS. Note that the objective functions for both types of data are 3-dimensional function vectors.

**Fitness function**. The fitness of a subspace $s$ in SPOT is defined based upon its Pareto-count in the whole population. The Pareto-count of $s$ is the number of subspaces that are inferior to $s$ in the population, *i.e.*, $fitness(s) = |\{s_i : s \succ s_i\}|$. $s_i$ is inferior to $s$, denoted as $s \succ s_i$, if none of the objective function values of $s_i$ is better than or as good as $s$.

**Selection scheme**. *Pareto-based selection scheme* is used to select fitter solutions in each step of evolution. It is a stochastic selection method where the selection probability of a subspace is proportional to its fitness value, *i.e.*, $Pr(s) = \frac{fitness(s)}{\sum_{i=1}^{P} fitness(s_i)}$, where $P$ is the population size.

**Elitism**. Elitism is the effort to address the problem of losing those potentially good solutions during the optimization process because of the randomness of MOGA. If no special measures are taken, MOGA cannot guarantee the individuals obtained in a newer generation always outperform those in the older one. In SPOT, we use the *elitism* method that directly copies the best or a few best subspaces in the population of one generation to the next one, in an attempt to achieve a constantly improving population of subspaces.

### 3.4 Detection Stage of SPOT

The detection stage of SPOT performs outlier detection for arriving streaming data. As streaming data arrive continuously, the data synopsis PCS of the projected cell where the streaming data belongs to in each subspace of SST are first updated in order to capture new information of the arriving data. Hash function is employed here to quickly map a data into the cell it is located in any subspace. Then, the data is labeled as a projected outlier on the fly if RD, IRSD or IkRD of the cell it belongs to in one or more SST subspaces falls under the pre-specified outlier-ness thresholds. These subspaces are the outlying subspaces of this outlier. All the outliers, together with their outlying subspaces and PCS of the cell they belong to in these outlying subspaces, are output to the Outlier Repository. All or a specified number of the top outliers in Outlier Repository are retuned to users in the end.

**Updating PCS of a Projected Cell.** The detection stage of SPOT involves the update of PCS of projected cells as data in the stream are being processed. The naive approach for updating PCS is to first update BCS of base cells when new data arrives and then project base cells from full domain space into the appropriate subspace in order to update PCSs. This approach, however, will be expensive because as high as $\delta^{\varphi-|s|}$ aggregations may be involved. Moreover, PCS of projected cells has to be updated in this way each time when new data arrives at this cell, thus frequent aggregation operations is required, which makes this naive method rather costly. In this subsection, we will demonstrate the incremental property of PCS, based on which PCS can be self-maintainable without using BCS.

• *Update RD of PCS*. Suppose that the latest snapshot of PCS of $c$ is created at time $T$ with an effective stream length $N$, then we can update RD in PCS of $c$ incrementally when a new data point arrives at $c$ at the time $T'$ ($T \leq T'$) with an effective stream length $N'$ as follows:

$$RD(c,s)^{T'} = \frac{[df(T'-T)\frac{RD(c,s)^{T} \cdot N}{\delta^{|s|}} + 1] \cdot \delta^{|s|}}{N'} \tag{1}$$

where $\frac{RD(c,s)^{T} \cdot N}{\delta^{|s|}}$ is the density of $c$ at time $T$ and $df(T'-T)\frac{RD(c,s)^{T} \cdot N}{\delta^{|s|}} + 1$ is thus the new density of $c$ at time $T'$. After simplifying Eq. (1), we can get

$$RD(c,s)^{T'} = df(T'-T)\frac{N}{N'}RD(c,s)^{T} + \frac{\delta^{|s|}}{N'} \tag{2}$$

From Eq. (2) we can see that, for incremental maintenance of RD, we need to only additionally maintain, for each non-empty cell in the subspace, the effective stream length $N$ when PCS of this cell is updated last time.

• *Update IRSD of PCS*. Suppose the density of cell $c$ is $m$ and let $IRSD(c,s)^T$ be the IRSD of cell $c$ in subspace $s$ at time $T$, which can be computed as follows based on the definition of $IRSD(c,s)^T$:

$$IRSD(c,s)^T = \frac{\sqrt{|s|}l}{2\sigma(c)} = \frac{\sqrt{|s|}l}{2}\sqrt{\frac{m-1}{\sum_{i=1}^{m} Dist(p_i,\mu_c)^2}} \tag{3}$$

where $p_i$ is located in $c$. Based on Eq. (3), we can get

$$\sum_{i=1}^{m} Dist(p_i,\mu_c)^2 = \frac{|s|l^2(m-1)}{4(IRSD(c,s)^T)^2} \tag{4}$$

The $IRSD(c,s)$ after the $(m+1)^{th}$ point is assigned into $c$ at time $T'$ ($T \leq T'$) is computed as

$$IRSD(c,s)^{T'} = \frac{\sqrt{|s|}l}{2}\sqrt{\frac{df(T'-T)(m-1)+1}{df(T'-T)\sum_{i=1}^{m} Dist(pi,\mu_c')^2 + Dist(p_{m+1},\mu_c')^2}} \tag{5}$$

where $\mu_c'$ denotes the *new* mean of points in $c$ when the $(m+1)^{th}$ point is inserted into $c$.

• *Update Representative Points in a Subspace*. To ensure a quick computation of IkRD for a new data in the stream, we need to obtain the current set of coverage cells efficiently. A native way would be re-sorting all the populated cells and pick up a specific number of high-density cells as the new coverage cells. This method is, however, expensive given the fact that the sorting has been performed in each subspace for each incoming data and the number of populated cells in many subspaces may be large. To solve this problem, we devise the following heuristics to minimize the number of re-sorting of projected cells:

1. If a new data falls into one of the coverage cells in a subspace, then there is no need to update the current set of coverage cells in this subspace;
2. Both the total coverage and the minimum density of the current set of coverage cells in each subspace $s$ of SST are maintained, denoted by $Cov$ and $Den_{min}$, respectively. If a new data falls into a non-coverage cell $c'$ in $s$, then there is no need to update the current set of coverage cells in $s$ either if we have $Cov' > q$ and $den(c') \leq Den'_{min}$, where $Cov'$ and $Den'_{min}$ correspond respectively to the decayed $Cov$ and $Den_{min}$ after the new data is processed, and $q$ denotes the coverage ratio required for the coverage cells. Both $Cov'$ and $Den'_{min}$ can be updated efficiently.

The reason that the current set of coverage cells does not need to update in the above two cases is that, after arrival of the new data, the current coverage cells still satisfy the coverage requirement and are still the cells with the highest densities in the subspace. These two heuristics can contribute to a significant efficiency improvement as majority of the data are expected to fall into coverage cells. For example, if the coverage cells cover 95% of the data in a subspace, then the possibility that the coverage cells need to be updated when processing a new data is much lower than 5% in practice by using these two heuristics.

**Deal with dynamics of data streams.** An essential issue to the effectiveness of SPOT is to cope with dynamics of data streams and respond to the possible concept drift. Besides using decaying functions on data synopsis, additional efforts have been taken in SPOT to deal with this issue, which are summarized as follows.

First, both BCSs of populated base cells and PCSs of populated projected cells in subspaces of SST will be efficiently maintained as data flow in. This ensures SST to be able to capture the latest data characteristics of the stream and response quickly to data changes;

**Algorithm: SPOT_Detection** (SST, $t$, $N_{can}$, $top\_k$)

**Input:** SST, self-evolution time period $t$ and number of new subspaces generated $N_{can}$ in self-evolution of SST;
**Output:** Outlier_Repository where outliers detected are stored.

```
1. SST_Candidate← ∅;
2. WHILE NOT (end of stream) DO {
3. IF a new data p in the stream arrives THEN {
4.   Update_BCS(p);
5.   Update_PCS(p, SST, SST_Candidate);
6.   IF (Outlier_Detection(p, SST)=True) THEN
7.     Insert(Outlier_Repository, p); }
8. IF ((Curent_time–Start_time) mod t=0) THEN {
9.   SST← SST_Evolution(SST, SST_Candidate);
10.  SST_Candidate ← Generate_SST_Candidate(SST, N_can);
11.  For each new outliers o added to Outlier_Repository DO{
12.    MOGA(o);
13.    SST← SST ∪ top sparse subspaces of o; } } }
14. Return(top_k_outliers(Outlier_Repository, top_k));
```

**Fig. 5.** Detection algorithm of SPOT

Second, any outlier, whenever it is detected, will not be discarded but rather be stored in Outlier Repository. Their top sparse subspaces produced by MOGA will be added into $\mathcal{SS}$ of SST to detect outliers from streaming data arriving later. As a consequence, the detecting ability of SST will be enhanced constantly as an increasing number of outliers are detected along the detection process;

Finally, SST is equipped with an unique feature of *online self-evolution*. As the detection stage proceeds, a number of new subspaces are periodically generated online by crossovering and mutating the top subspaces in the current SST. These newly generated subspaces represent the evolution of SST. Once the new subspaces join SST, the whole SST, including the new subspaces, will be re-ranked and the new top sparse subspaces will be chosen to create a new SST. As a trigger of the self-evolution, a concept drift monitoring mechanism is also incorporated to detect the sudden change of data in the streams. The data distribution can be measured at any time using PCSs of cells in the hypercube and a significantly different PCS (i.e., higher than some pre-defined threshold) is indicative of a noticeable change of data distribution, representing a possible concept drift. When this happens, the self-evolution will be activated to re-shuffle the SST.

**Detection algorithm.** The detection algorithm of SPOT is presented in Figure 5. The detection algorithm is executed as long as the end of stream has not been reached. The set called *SST_Candidate* is used to store the new subspaces generated periodically from SST to represent its self-evolution. Upon arrival of a new data, three substeps are performed (Step 3-7), that are, 1) BCS of the base cell in the hypercube to which the point belong are updated; 2) PCS of the projected cell in each SST subspace and candidate SST subspace to which the point belongs are updated and 3) this point is checked in each of subspaces in SST to decide whether or not it is a projected outlier. If this point is found to be a projected outlier, it will be added to Outlier_Repository where all the detected projected outliers are to be stored. Every time when a time period of $t$ is elapsed, the self-evolution is SST is carried out. It creates a new SST by using the current SST and SST_Candidate (Step 9). Then, it generates new subspaces for self-evolution in the next cycle. In Step 10, a number of new candidate SST subspaces are generated. For those newly detected outliers in the latest cycle, MOGA is applied in Step 11 and 12 to find their respective top sparse subspaces in order to update SST. Finally, the $top\_k$ projected outliers detected in the detection stage are returned (Step 14).

## 4 Experimental Results

We use both synthetic and real-life datasets for performance evaluation. All the experimental evaluations are carried out on a Pentium 4 PC with 512MB RAM.

**Synthetic data sets.** The synthetic data sets are generated by two high-dimensional data generators. The first generator SD1 is able to produce data sets that generally exhibit remarkably different data characteristics in different subspaces. The number, location, size and distribution of the data in different subspaces are generated randomly. This generator has been used to generate high dimensional data sets for outlying subspace detection [21][19][20]. The second synthetic data generator SD2 is specially designed for comparative study of SPOT and the existing methods. Two data ranges are defined as $R_1 = (a, b)$ and $R_2 = (c, d)$, where $b + l < c$, $l$ is the length of a partitioning interval in each dimension. This ensures that the data points in $R_1$ and $R_2$ will not fall into the same interval for each dimension. In SD2, we first generate a large set of normal data points $D$, each of which will fall into $R_1$ in $\varphi - 1$ dimensions and into $R_2$ in only one dimension. We then generate a small set of projected outliers $O$. Each projected outlier will be placed into $R_2$ for all the $\varphi$ dimensions. Given the large size of $D$ relative to $O$, no projected outliers will exist in $D$. An important characteristic of SD2 is that the projected outliers appear perfectly normal in all 1-dimensional subspaces. To tailor them to fit data stream applications, time dimension are added into SD1 and SD2. The outliers generated in SD2 are assumed to arrive at $|O|$ randomly distributed time points $t_i \in \{t_1, \cdots, t_{|O|}\}$ of the whole data stream.

**Real data sets.** We also use 4 real-life multi- and high-dimensional datasets in our experiments. The first two data sets come from the UCI machine learning repository. They are called *Letter Image* (RD1, 17-dimensions) and *Musk* (RD2, 168-dimensions), respectively. RD1 and RD2 does not have timestamps. However, the decayed weight functions for the data in these data sets can be computed based on their sequential index, instead of the timestamps, in the data sets. The third real-life data set is the *KDD-CUP'99 Network Intrusion Detection stream data set* (RD3, 42 dimensions). RD3 has been used to evaluate the clustering quality for several stream clustering algorithms [4][5]. The fourth real-life data set is the MIT wireless LAN (WLAN) data stream (RD4, 15 dimensions), which can be downloaded from *http://nms.lcs.mit.edu/ mbalazin/wireless/*.

### 4.1 Scalability Study

The scalability study investigates the scalability of SPOT (both learning and detection processes) w.r.t length $N$ and dimensionality $\varphi$ of data streams. The learning process we study here refers only to the unsupervised learning that generates $\mathcal{US}$ of SST. Due to its generality, SD1 with different $N$ and $\varphi$ is used in all scalability experiments.

**Scalability of learning process w.r.t $N$.** Figure 6 shows the scalability of unsupervised learning process w.r.t number of training data $N$. The major tasks involved in the unsupervised learning process are multi-run clustering of training data, selection of top training data that have highest outlying degree and application of MOGA on each of them to generate $\mathcal{US}$ of SST. The lead clustering method we use requires only a single scan of the training data, and the number of top training data we choose is usually linearly depended on $N$. Therefore, the overall complexity of unsupervised learning process scales linearly w.r.t $N$.

**Scalability of learning process w.r.t $\varphi$.** Since the construction of $\mathcal{FS}$ in SST does not need any leaning process, thus the dimension of training data $\varphi$ will only affect the complexity of learning process in a linear manner, *provided that a fixed search workload is specified for MOGA*. As confirmed in Figure 7, we witness an approximately linear growth of execution time of learning process when $\varphi$ is increased from 20 to 100 under a fixed search workload in MOGA.

**Scalability of detection process w.r.t $N$.** In Figure 8, we present the scalability result of detection process w.r.t stream length $N$. In this experiment, the stream length is set much larger than the number of training data in order to study the performance of SPOT in coping

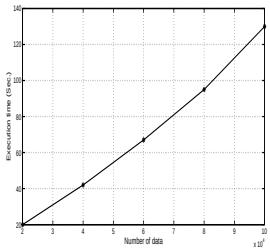


**Fig. 6.** Scalability of learning process w.r.t data number

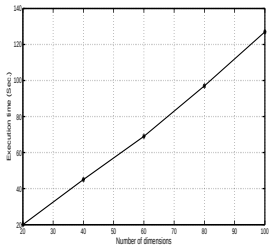


**Fig. 7.** Scalability of learning process w.r.t data dimension

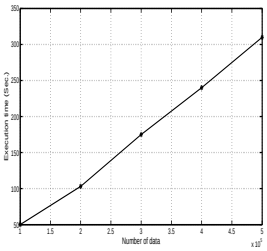


**Fig. 8.** Scalability of detection process w.r.t data number

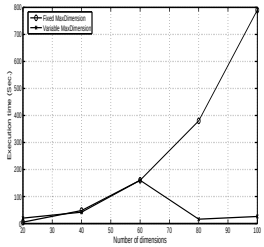


**Fig. 9.** Scalability of detection process w.r.t data dimension

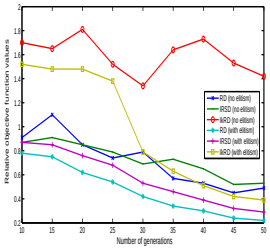


**Fig. 10.** Convergence study of MOGA

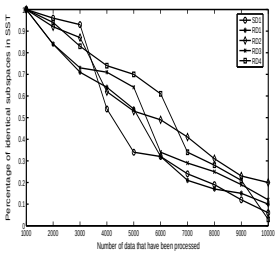


**Fig. 11.** Evolution of SST

with large data streams. Figure 8 shows a promising linear behavior of detection process when handing an increasing amount of streaming data. This is because that the detection process needs only one scan of the arriving streaming data. In addition, since BCS and PCS are both incrementally maintainable, detection process of SPOT thus becomes very efficient. This leads to a high throughput of SPOT and enables it to deal with fast data streams.

**Scalability of detection process w.r.t $\varphi$.** $\varphi$ affects the size of $\mathcal{FS}$ that is used in detection process. When $MaxDimension$ is fixed, the size of $\mathcal{FS}$ is in an exponential order of $\varphi$, which is usually much larger than that of $\mathcal{US}$ and OS. This causes $\mathcal{FS}$ to dominate the whole SST. As such, the execution time of detection process is expected to grow exponentially w.r.t $\varphi$. We typically set lower $MaxDimension$ values for data streams with higher dimensionality to prevent an explosion of $\mathcal{FS}$. We first use $MaxDimension = 3$ for data streams of different dimensions and we can see an exponential behavior of the detection process. Then, we use variable values for $MaxDimension$ to adjust the size of $\mathcal{FS}$. We set $MaxDimension = 4$ for data sets with dimension of 20 and 40, set $MaxDimension = 3$ for data sets with dimension of 60 and finally set $MaxDimension = 2$ for data sets with dimension of 80 and 100. If this strategy is used, we will witness an irregularly-shaped, rather than an exponential, curve of the detection process. The results are presented in Figure 9.

### 4.2 Effectiveness Study

**Convergence study of MOGA.** We first study the convergence of MOGA in terms of optimizing RD, IRSD and IkRD. Convergence of MOGA is crucial to outlying subspaces search in SPOT. We investigate the average of RD, IRSD and IkRD of the top 10 subspaces in the population of each generation of MOGA. This experiment is conducted on SD1, RD1, RD2, RD3 and RD4. Only the results of RD3 (KDD-CUP'99 data stream) are presented (see Figure 10). Similar results are obtained for other datasets. Generally speaking, the criteria are being improved (minimized) as more generations are performed in MOGA. This indicates a good convergence behavior of MOGA in searching outlying subspaces. However,

there are some abrupt increase of optimizing criteria values in the search process. The underlying reason is that, when elitism is not used, there is a higher chance of occurrence of *degrading generations* in the evolution. This is due to the randomized nature of MOGA that likely renders good solutions in one generation to be lost in the next one by crossover or mutation operations. When elitism is employed, we can achieve a better optimization of RD, IRSD and IkRD. Because the best solutions are copied into the next generation, we do not see any vibrations of optimizing criteria values in MOGA and the optimizing criteria can be constantly improved as evolution proceeds.

**Detection performance of SST.** The three sub-groups of SST work collectively to detect outliers from data streams. Their respective contribution in outlier detection is not identical though. This experiment aims to study the percentage of outliers that can be detected by using each of the sub-groups of SST alone. The experimental results indicate that by using $\mathcal{FS}$ alone, which covers the full low-dimensional space lattice, we can detect as many as 80-90% of the total outliers that exist, while $\mathcal{US}$ and $\mathcal{SS}$ can further help the system detect another 10-20% of the outliers.

**Evolution of SST.** One of the important features of SPOT is its capability of self-evolution. This is a very useful feature of SPOT to cope with the fast-changing data streams. In this experiment, we investigate the evolution of SST as an informative indicator of concept drift of data streams. The setup is as follows. We keep the initial version of SST (*i.e.*, the SST obtained after 1000 data points are processed) and record the versions of SST when every 1000 data (up to 10,000) are processed afterwards. Self-evolution is activated at every 1000-data interval. We compare different SST versions with the initial one and calculate the percentage of identical subspaces. SD1, RD1, RD2, RD3 and RD4 are used for this experiment. The results are shown in Figure 11. We find that an increasing number of subspaces in the initial SST have disappeared in the later SST versions as more data points are processed. We use the same seeds in MOGA, ruling out the randomness in individual generation for different self-evolution sessions. Therefore, the change of SST is due to the varying characteristics of the data stream and outliers we detect in different stages.

**Comparative study.** Since there is little research conducted in projected outlier detection for high-dimensional data streams, we cannot find the techniques tackling exactly the same problem as SPOT does for comparison. However, there are some related existing approaches for detecting outlier detection from data streams that we can compare SPOT with. They can be broadly categorized as methods using *histogram*, *Kernel density function*, *distance-based function* and *clustering analysis*, respectively.

Histogram and Kernel density function are the most commonly used statistical techniques in outlier detection. Histogram technique creates bins for each dimension of the data stream. The density of each bin of the histogram are recorded. The sparsity of a bin in the histogram can be quantified by computing the ratio of the density of this bin against the average density of all the bins in the histogram for this dimension. A data point is considered as outlying in a dimension if it falls into an excessively sparse bin. Kernel density function models local data distribution in a single or multiple dimensions of space. A point is detected as an outlier if the number of values that have fallen into its neighborhood is less than an application-specific threshold. The number of values in the neighborhood can be computed by the kernel density function. Distance-based function draws on some distance metrics to measure the local density of data in order to find outliers. A major recent distance-based method for data stream is called *Incremental LOF* [16]. Clustering analysis can also be used to detect outliers from those data that are located far apart from data clusters. *HPStream* [5] is the representative method for finding subspace clusters in high-dimensional data streams.

Amongst the 4 competitive methods, histogram method mainly deals with single attribute and HPStream has mechanism to explore subspaces for optimizing clusters. However, the Kernel density function method and Incremental LOF can handle multiple attributes but lacks the ability to explore subspaces. Their detection performance is thus heavily dependent on the selection of subspaces for outlier detection. In this experiment, we study three different ways for deciding subspaces where these two method will be applied, namely randomly selecting a single subspace, randomly selecting multiple subspaces and using SST

| Methods | SD2 | | RD3 | | RD4 | |
|---|---|---|---|---|---|---|
| | DR | FPR | DR | FPR | DR | FPR |
| Histogram | 0% | 32.1% | 43.3% | 24.7% | 48.9% | 19.2% |
| Kernel density function (random single subspace) | 3.2% | 1.1% | 7.3% | 0% | 2.8% | 0.3% |
| Kernel density function (random multi subspaces) | 87.1% | 1.5% | 79.3% | 8.6% | 83.8% | 7.4% |
| Kernel density function (SST) | 100% | 0% | 84.3% | 4.6% | 91.9% | 7.5% |
| Incremental LOF (random single subspace) | 6.2% | 0% | 8.9% | 0.1% | 4.5% | 0% |
| Incremental LOF (random multi subspaces) | 85.2% | 2.5% | 79.2% | 3.2% | 81.7% | 4.9% |
| Incremental LOF (SST) | 95% | 0.7% | 91.4% | 7.9% | 90.7% | 8.9% |
| HPStream | 8.3% | 1.2% | 7.3% | 1.1% | 3.1% | 0.8% |
| SPOT | 100% | 0% | 89.5% | 5.3% | 92.9% | 6.5% |

**Fig. 12.** Comparative study results

obtained by SPOT. We use SD2, RD3 and RD4 to carry out the comparative experiments. As RD4 is an unlabeled data stream, an offline detection is thus performed to established the ground truth results prior to the comparative study.

The performance of all the methods are measured by *detection rate (DR)* and *false positive rate (FPR)*. The results are summarized in Figure 12. To facilitate result analysis, two selection rules are devised based on DR and FPR. They are $R_1 : DR > 90\%$ and $FPR < 5\%$, $R_2 : DR > 85\%$ and $FPR < 10\%$. Obviously, $R_1$ is more desirable than $R_2$ in terms of detection performance. We highlight the cells in the table that satisfy $R_1$ using red colour and those that satisfy $R_2$ using purple colour, respectively. After colouring, it becomes much easier for us to see, from a macro scale, that SPOT and Incremental LOF (SST) achieve the best performance overall for the three data sets, followed by Kernel density function (SST), Kernel density function (random multi subspaces) and Incremental LOF (random multi subspaces). Histogram, Kernel density function (random single subspace), Incremental LOF (random single subspace) and HPStream bottom the list, which does not contain any colored (DR, FPR) pairs. Compared with other competitive methods, SPOT is advantageous that it is equipped with subspace exploration capability, which contributes to a good detection rate. Moreover, using multiple criteria enables SPOT to deliver much more accurate detection which helps SPOT to reduce its false positive rate.

## 5 Related Work

There have been abundant research in outlier detection in the past decade. Most of the conventional outlier detection techniques are only applicable to relatively low dimensional static data [8][11][12][14][18][15]. Because they use the full set of attributes for outlier detection, thus they are not able to detect projected outliers. They cannot handle data streams either. Recently, there are some emerging work in dealing with outlier detection either in high-dimensional static data or data streams. However, there have not been any reported concrete research work so far for exploring the intersection of these two active research directions. For those methods in projected outlier detection in high-dimensional space [3][23][21][19][20], they can detect projected outliers that are embedded in subspaces. However, their measurements used for evaluating points' outlier-ness are not incrementally updatable and many of the methods involve multiple scans of data, making them incapable of handling data streams. For instance, [3][23] use the Sparsity Coefficient to measure data sparsity. Sparsity Coefficient is based on an equi-depth data partition that has to be updated frequently in data stream. This will be expensive and such updates will require multiple scan of data. [21][19][20] use data sparsity metrics that are based on distance involving the concept of $k$NN. This is not suitable for data streams either as one scan of data is not sufficient for retaining $k$NN information of data points. One the other hand, the techniques for tackling outlier detection in data streams [13][2][17] rely on full data space to detect outliers and thus projected outliers cannot be discovered by these techniques.

## 6 Conclusions

In this paper, we propose SPOT, a novel technique to deal with the problem of projected outlier detection in high-dimensional data streams. SPOT is equipped with incrementally updatable data synapses (BCS and PCS) and is able to deal with fast high-dimensional streams. It is flexible in allowing for both supervised and unsupervised learning in generating the detecting template SST. It is also capable of handling dynamics of date streams. The experimental results demonstrate the efficiency and effectiveness of SPOT in detecting outliers in high-dimensional data streams.

## References


1. C. C. Aggarwal and P. S. Yu. An effective and efficient algorithm for high-dimensional outlier detection. *VLDB Journal*, 14: 211-221, 2005.
2. C. C. Aggarwal. On Abnormality Detection in Spuriously Populated Data Streams. *SDM'05*, Newport Beach, CA, 2005.
3. C. C. Aggarwal and P. S. Yu. Outlier Detection in High Dimensional Data. *SIGMOD'01*, 37-46, Santa Barbara, California, USA, 2001.
4. C. C. Aggarwal, J. Han, J. Wang, P. S. Yu. A Framework for Clustering Evolving Data Streams. *VLDB'03*, 81-92, Berlin, Germany, 2003.
5. C. C. Aggarwal, J. Han, J. Wang, P. S. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. *VLDB'04*, 852-863, Toronto, Canada, 2004.
6. F. Angiulli and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. *PKDD'02*, Helsinki, Finland, 15-26, 2002.
7. D. Barbara. Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, Volume 3, Issue 2, 23 - 27, ACM Press, 2002.
8. M. Breuning, H-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. *SIGMOD'00*, Dallas, Texas, 93-104, 2000.
9. A. Guttman. A. Guttman. R-trees: a Dynamic Index Structure for Spatial Searching, *SIGMOD'84*, 47-57, Boston, Massachusetts, 1984.
10. J. Han and M Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2000.
11. E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-based Outliers in Large Dataset. *VLDB'98*, New York, NY, 392-403, 1998.
12. E. M. Knorr and R. T. Ng. Finding Intentional Knowledge of Distance-based Outliers. *VLDB'99*, Edinburgh, Scotland, 211-222 1999.
13. T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Record* 32(4): 77-82, 2003.
14. S. Ramaswamy, R. Rastogi, and S. Kyuseok. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD'00*, Dallas Texas, 427-438, 2000.
15. S. Papadimitriou, H. Kitagawa, P. B. Gibbons, C. Faloutsos. LOCI: Fast Outlier Detection Using the Local Correlation Integral. *ICDE'03*, 315 Bangalore, India, 2003.
16. D. Pokrajac, A. Lazarevic, L. Latecki. Incremental Local Outlier Detection for Data Streams, *CIDM'07*, 504-515, Honolulu, Hawaii, USA, 2007.
17. S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. *VLDB'06*, 187-198, Seoul, Korea, 2006.
18. J. Tang, Z. Chen, A. Fu, and D. W. Cheung. Enhancing Effectiveness of Outlier Detections for Low Density Patterns. *PAKDD'02*, Taipei, Taiwan, 2002.
19. J. Zhang, M. Lou, T. W. Ling and H. Wang. HOS-Miner: A System for Detecting Outlying Subspaces of High-dimensional Data. *VLDB'04*, 1265-1268, Toronto, Canada, 2004.
20. J. Zhang, Q. Gao and H. Wang. A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm. *ICDM'06*, 731-740, Hong Kong, China, 2006.
21. J. Zhang and H. Wang. Detecting Outlying Subspaces for High-dimensional Data: the New Task, Algorithms and Performance. *Knowledge and Information Systems (KAIS)*, 33-355, 2006.
22. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD'96*, 103-114, Montreal, Canada, 1996.
23. C. Zhu, H. Kitagawa and C. Faloutsos. Example-Based Robust Outlier Detection in High Dimensional Datasets. *ICDM'05*, 829-832, Houston, Texas, 2005.