Enhancements on Local Outlier Detection

Anny Lai-mei Chiu, Ada Wai-chee Fu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong
{lmchiu, adafu}@cse.cuhk.edu.hk

Abstract

Outliers, or commonly referred to as exceptional cases, exist in many real-world databases. Detection of such outliers is important for many applications. In this paper, we focus on the density-based notion that discovers local outliers by means of the Local Outlier Factor (LOF) formulation. Three enhancement schemes over LOF are introduced, namely LOF′, LOF″ and GridLOF. Thorough explanation and analysis is given to demonstrate the abilities of LOF′ in providing simpler and more intuitive meaning of local outlier-ness; LOF″ in handling cases where LOF fails to work appropriately; and GridLOF in improving the efficiency and accuracy.

Keywords: *outlier detection, outlier-ness, density*

1. Introduction

In contrast to most KDD tasks, such as clustering and classification, outlier detection aims to find the small portion of data which are deviating from common patterns in the database. Studying the extraordinary behavior of outliers helps uncovering the valuable knowledge hidden behind them. The hidden knowledge obtained can be useful in the detection of criminal activities in E-commerce, telecom and credit card frauds, video surveillance, pharmaceutical research, loan approval and intrusion detection.

A well-quoted definition of outliers is the Hawkin-Outlier which first appeared in [10]. This definition states that *an outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism*. Hawkin-Outlier is defined in an intuitive manner.

With increasing awareness on outlier detection in both statistical and database literatures, more concrete meanings of outliers are defined for solving problems in specific domains. Nonetheless, each of these definitions follows the spirit of the Hawkin-Outlier. In this paper, we emphasize the scheme of density-based outliers and the corresponding Local Outlier Factor (LOF) formulation, which is used to indicate the local outlier-ness of objects in databases. Strength and weakness of the LOF formulation will be considered.

Our contributions in this paper are three enhancement schemes which address the weaknesses of LOF accordingly, they are (1) LOF′, (2) LOF″ and (3) GridLOF. The first two schemes are variants of the original LOF formulation. LOF′ provides simpler and more intuitive meaning of local outlier-ness, while LOF″ can handle cases which LOF fails to work appropriately. The third enhancement, GridLOF, is an efficient and adaptive algorithm in calculating LOF value of each data objects in the databases, GridLOF can also increase accuracy since it avoids some false identifications that can occur with LOF.

1.1. Related Work in Outlier Detections

Early schemes that consider outlier detection as the primary objective are in the field of statistics [5]. The distribution-based approach works by fitting suitable statistical models on the data. Another approach used in statistics is based on a depth notion (e.g. [13]) but this method is unscalable with the dataset dimensionality.

In order to perform effective clustering, most clustering algorithms are able to handle noise in the datasets. We refer to the returned noisy data as clustering-based outliers. Example clustering algorithms which also handle outliers are BIRCH [19], CLARANS [15], DBSCAN [9], GDBSCAN [17], OPTICS [4] and PROCLUS [1]. However, the outliers are identified as by-products and are highly dependent on the algorithms used. In outlier analysis, we want to focus our efforts on outlier detections. In this case, finding outliers without the need of clustering operations is desirable.

A distance-based outlier is a data point having a far distance to the other data points in the data space [14] [16]. The density-based notion of local outliers overcomes the problem that distance-based approaches fail to handle clusters of different densities [8]. A degree of outlier-ness is given by the Local Outlier Factor (LOF) in [8]. Local outliers are points having considerable density difference from their neighboring points, they have high LOF values.

Despite the fact that the concept of LOF is a useful one, the computation of LOF value of each data object requires a lot of k-nearest neighbors queries. This makes each calculation of LOF a costly operation. Based on the assumption that most data objects are unlikely to be outliers and users are only interested in getting the information of the strongest n local outliers in a large database of size N, an algorithm is proposed in [12] which let users decide the number of strongest outliers they would like the algorithm to return. As such, a lot of the LOF computations are avoided (assuming $n << N$) which results in higher efficiency for the algorithm.

2. LOF Revisited

Based on the same theoretical foundation as DBSCAN [9] and OPTICS-OF [7], LOF [8], a method identifying density-based local outliers, computes the local outliers of a dataset, by assigning an outlier factor to each object, based on the outlying property relative to their surrounding space.

Definition 1 (k-$dist(p)$)
Given any positive integer k and dataset D, the k-distance of an object p, denoted as k-$dist(p)$, is defined as the distance $dist(p, o)$ between p and an object $o \in D$ satisfying:

1. *at least k objects $q \in D \setminus \{q\}$ having $dist(p, q) \leq dist(p, o)$, and*
2. *at most $(k-1)$ objects $q \in D \setminus \{q\}$ having $dist(p, q) < dist(p, o)$.*

Definition 2 ($N_{k\text{-}dist(p)}(p)$)
Given k-$dist(p)$, $N_{k\text{-}dist(p)}(p)$ denotes the k-distance neighborhood of p which is the set of objects q whose distance from p is at most k-$dist(p)$. More formally, $N_{k\text{-}dist(p)}(p) = \{q \mid q \in D \setminus \{p\}, dist(p, q) \leq k\text{-}dist(p)\}$ with q being the k-nearest neighbors of p.

Figure 1 is a example which shows the meaning of k-distance for $k = 5$. In this example, the k-distance of object o is the radius of the dashed circle, while the k-distance neighborhood of o are the five points inside this circle except for o.

Definition 3 ($reach\text{-}dist_k(p, o)$)
For a given positive integer k and an object p, reachability distance of p w.r.t. object o is defined as $reach\text{-}dist_k(p, o) = \max\{k\text{-}dist(o), dist(p, o)\}$.

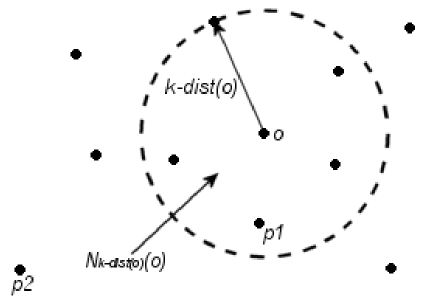


Figure 1. k-$dist(o)$ **and** $N_{k\text{-}dist(o)}(o)$ **for** $k = 5$

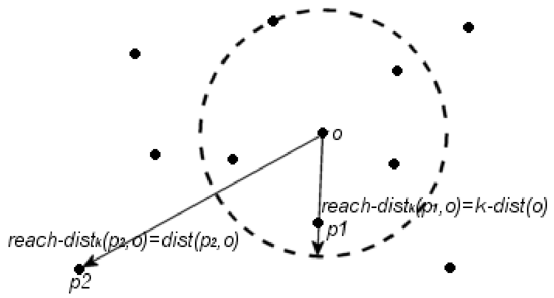


Figure 2. $reach\text{-}dist_k(p_1, o)$ **and** $reach\text{-}dist_k(p_2, o)$ **for** k**=5.**

Figure 2 is a example which demonstrates the concept of reachability distance when k=5. The k-distance of object o is the radius of the dashed circle. For object p_1, since its distance to object o is less than the k-distance of o, the reachability distance of it w.r.t. o equals to the d-distance of o. For object p_2, the distance between the object and object o is greater than o's k-distance, so the reachability of p_2 w.r.t. o is the distance between p_2 and o.

In order to detect density-based outliers, the density of the neighborhood of each object p is determined. A parameter $MinPts$, which is a positive integer, is kept to specify the minimum number of points that resides in p's neighborhood. Let $o \in N_{MinPts\text{-}dist(p)}(p)$, the reachability distance $reach\text{-}dist_{MinPts}(p, o)$ regarding this $MinPts$ is used as a measure of volume of p's neighborhood.

Definition 4 ($lrd_{MinPts}(p)$)
The local reachability density of object p is defined as

$$lrd_{MinPts}(p) = 1 \Bigg/ \frac{\sum_{o \in N_{MinPts\text{-}dist(p)}} reach\text{-}dist_{MinPts}(p, o)}{|N_{MinPts\text{-}dist(p)}(p)|}.$$

The local reachability density of object p is the inverse of the average reachability distance of the $MinPts$-distance neighborhood of p. Finally, the Local Outlier Factor (LOF) is defined as below.

Definition 5 ($LOF_{MinPts}(p)$)
The local outlier factor of object p *is defined as*

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts\text{-}dist(p)}} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts\text{-}dist(p)}(p)|}.$$

The LOF of object p is the average ratio of local reachability density of it and its $MinPts$-distance neighborhood.

2.1. LOF′: A Simpler Formula

We propose here a better formulation compared with LOF. Unlike the method of connectivity-based outlier factor (COF) in [18] which focuses on outlier detections for low density patterns, our enhancement scheme improves the efficiency and effectiveness of LOF for general datasets.

It can be seen that the notion of LOF is quite complex. Three components including $MinPts$-dist, reachability distance and local reachability density are to be understood before the understanding of the LOF formulation. Local reachability density is an indication of the density of the region around a data point. We argue that $MinPts$-dist already captures this notion: a large $MinPts$-dist corresponds to a sparse region, a small $MinPts$-dist corresponds to a dense region. In view of this, LOF′ is defined as a simpler formula for ease of understanding, and also simpler computation. This variant of LOF bears more intuitive meaning and exhibits similar properties as LOF.

Definition 6 (LOF′)

$$LOF'_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts\text{-}dist(p)}(p)} \frac{MinPts\text{-}dist(p)}{MinPts\text{-}dist(o)}}{|N_{MinPts-dist(p)}(p)|}$$

LOF′ defined here is the average ratio of $MinPts$-dist of an object and that of its neighbors within $MinPts$-dist. We reason that $MinPts$-dist is already an indicator of the local density of a data point. A large $MinPts$-dist means that the density is low since the distance to the nearest $MinPts$ neighbors is large. With this new definition, the components reachability distance and local reachability density needed in the LOF formula are not required anymore. LOF′ captures the degree of outlier-ness in a similar way as LOF but provides a clear and simple way of formulation.

Resembling the formula of LOF, LOF′ value increases as the degree of outlier-ness increases for an object. We can derive a lemma that is similar to the one exhibited in [8] for LOF to show the correctness of LOF' for clustered points. Similar to [8], we assume inside the cluster, the maximum distance between neighbors and minimum distance between neighbors are very close in values. Then objects deep inside clusters have LOF′ values approximately equal to 1:

Lemma 1
Let C *be a set of objects forming a cluster,*
$minDist = \min\{MinPts\text{-}dist(o) \mid o \in C\}$,
$maxDist = \max\{MinPts\text{-}dist(o) \mid o \in C\}$, *and*
$\varphi = \frac{maxDist}{minDist}$.
Assume φ *is close to 1. Let* $p \in C$ *be an object embedded inside the cluster. Then* $LOF'(p)$ *is approximately 1.*

Proof:
Within the specific cluster C, since $LOF'(p)$ is the average ratio of $MinPts\text{-}distance(p)$ to $MinPts\text{-}distance(q)$, for some q also in C, therefore $\frac{minDist}{maxDist} \leq LOF'(p) \leq \frac{maxDist}{minDist}$. Hence, $1/\varphi \leq LOF'(p) \leq \varphi$. If C is a tight cluster such that $maxDist$ is nearly the same as $minDist$, then φ is quite close to 1 and thus $LOF'(p)$ is approximately 1.

∎

Another advantage of the simplicity is that to compute LOF′ is more efficient than computing LOF since one pass over the data is saved by eliminating the reachability distance and local reachability density in the definition. For very large databases, each scan through the data is a costly operation, so saving a pass is a nice feature.

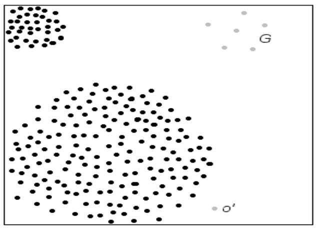
2.2. LOF′′ for Detecting Small Groups of Outliers

Sometimes outlying objects may be quite close to each other in the data space, forming small groups of outlying objects. An example illustrating this phenomenon is shown in Figure 3(a). Since $MinPts$ reveals the minimum number of points to be considered as a cluster, if the $MinPts$ is set too low, the groups of outlying objects will be wrongly identified as clusters. On the other hand, $MinPts$ is also used to compute the density of each point, so if $MinPts$ is set too high, some outliers near dense clusters may be misidentified as clustering points.

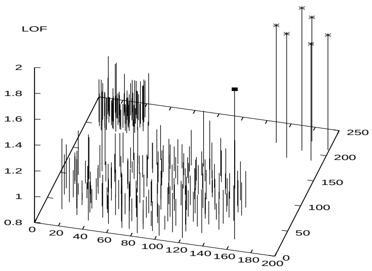
We notice there are in fact two different neighborhoods: (1) neighbors in computing the density and (2) neighbors in comparing the densities. In LOF, these two neighborhoods are identical. Here we suggest that they can be different, so we have two $MinPts$ values. For example, consider Figure 3(a). If we use a small neighborhood ($MinPts_1$) for computing the density, o' (see the labeled point at the lower right corner) in Figure 3(a) will be uncovered. If we compare the density of a point to a large neighborhood of points ($MinPts_2$), G (the group of points in the upper right corner of Figure 3(a))will be identified as outliers. The new notion of LOF′′ is given below:

Definition 7 (LOF′′)

$$LOF''_{MinPts_1,MinPts_2} = \frac{\sum_{o \in N_{MinPts_1\text{-}dist(p)}} \frac{lrd_{MinPts_2}(o)}{lrd_{MinPts_2}(p)}}{|N_{MinPts_1\text{-}dist(p)}(p)|}$$



(a)



(b)

Figure 3. (a) Sample dataset $DB2$. (b) LOF$''$ result of $DB2$.

One can put a relatively small value as $MinPts_2$ compared with $MinPts_1$. With this simple amendment, LOF$''$ is able to capture local outliers under different general circumstances. In the example in Figure 3(a), if we choose $MinPts_1$ = 10, and $MinPts_2 = 5$, we can identify both o' and G as shown in Figure 3(b). If we use only a single $MinPts$ value as in LOF, then we show in Figure 8 that no value of $MinPts$ can uncover all outliers exactly.

When $MinPts_2 = MinPts_1$, the formula of LOF$''$ is reduced to that of LOF. It can be said that LOF$''$ is a generalization of LOF. LOF$''$ exhibits the similar property as LOF and LOF$'$ that points deep inside a cluster have LOF$''$ values close to 1:

Lemma 2
Let C be a set of objects forming a cluster,
$minDist'' = \max\{reach\text{-}dist(a,b) | a, b \in C\}$,
$maxDist'' = \min\{reach\text{-}dist(a,b) | a, b \in C\}$, *and*
$\varphi'' = \frac{maxDist''}{minDist''}$.
Assume that φ'' is close to 1. Let $p \in C$ be an object embedded inside the cluster. Then $LOF''(p)$ is approximately 1 for p.

Proof:
Within the specific cluster C, since $LOF'(p)$ is the average ratio of $MinPts\text{-}distance(p)$ to $MinPts\text{-}distance(q)$, for some q in C, therefore $\frac{minDist}{maxDist} \leq LOF'(p) \leq \frac{maxDist}{minDist}$. Hence, $1/\varphi \leq LOF'(p) \leq \varphi$. If C is a tight cluster such that $maxDist$ is nearly the same as $minDist$, then φ is quite close to 1 and thus $LOF'(p)$ is approximately 1.

■

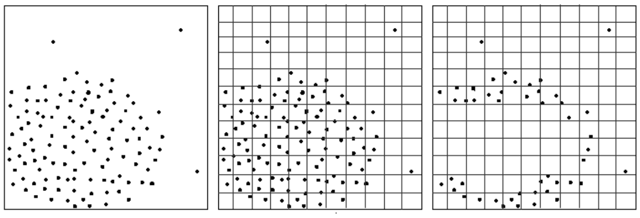


Figure 4. Example illustrating the idea of GridLOF algorithm.

2.3. GridLOF for Pruning Reasonable Portions from Datasets

In common situations, the number of outliers in any dataset is expected to be extremely small. It is highly inefficient for the LOF algorithm in [8] to compute the LOF values for all points inside a dataset. According to this observation, we introduce an adaptive algorithm called GridLOF (Grid-based LOF) algorithm which prunes away the portion of dataset known to be non-outliers, LOF of the remaining points are then calculated. Hence the overall cost for computing LOF can be reduced.

GridLOF utilizes a simple grid-based method as the pruning heuristic. At first, each dimension of the data space is quantized into equi-width intervals, resulting in a grid-based structure. Then for each non empty grid cell c, the neighboring grid cells are examined, c is labeled as a boundary cell once a neighboring grid cell with less than or equal to the pre-defined threshold (σ) number of points residing in it is found. σ is a relatively small number. In the extreme case, σ can be set to zero. (In our experiments, we found that $\sigma = 0$ gives pretty good results.) Finally, only the LOF values of points inside boundary cells are calculated. This heuristic works if the interval value used in partitioning the data space is appropriate. Figure 4 illustrates the idea of GridLOF algorithm.

Instead of keeping all the grids explicitly, GridLOF uses a method similar to the coding function for grid cells in [11]. To do this, $signature$ is defined (Definition 8) to play the role as the coding function and serves as the identity number of each grid cell.

Definition 8 (sig)
Given a dataset D with dimensionality l and the number of

intervals ω. sig *is an* l*-dimensional array for the grid cell* $signatures$. $sig = [s_1][s_2]\ldots[s_l]$ *such that* s_i *is the interval ID for dimension* i *ranging from any positive integer from 0 to* $\omega - 1$.

GridLOF scans the dataset once and based on the input parameter of the number of intervals ω to be partitioned in each dimension, it determines the grid cell that each point belongs to and determine the $signature$ of that grid cell. In this case, the data space is partitioned logically and GridLOF only remembers non empty grid cells which containing points. This method prevents the exponential growth of number of grid cells as dimensionality increases, since the number of distinct grid cells obtained is at most N (N is the size of the dataset), which is independent of the dimensionality when each point is residing in a different grid cell. The data structure used to store the set of unique grid cell signatures should guarantee efficient retrieval of the signatures, thus we choose hashing as the data structure for signatures storage. For each distinct grid cell signature in the hash table, GridLOF determines its *1-cell thick neighboring grid cells* $Nsig$ as in Definition 9. Once GridLOF finds that there is an empty cell in $Nsig$, the grid cell with the current signature can be identified as a **boundary grid cell**.

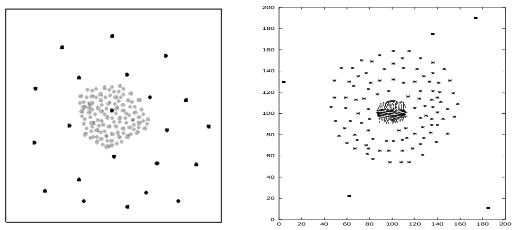
Definition 9 ($Nsig(sig_i)$)
For a given $signature$ *of a grid cell,* $sig_i = [s_1][s_2]\ldots[s_l]$, $Nsig(sig_i)$ *is a set of* $signatures$ *for the 1-cell thick neighboring grid cells of this grid cell.*

$$Nsig(sig_i) = \{[n_1][n_2]\ldots[n_l] \mid n_i = s_i \pm 1, 0 \leq n_i \leq \omega - 1\}$$

Up to this step, a preprocessing for LOF computation is done. The resulting set of points R residing in boundary grid cells is used in the later steps, where their LOF values are computed as for the original LOF. For typical situations, most points from the dataset D are pruned, so $|R| << |N|$. GridLOF scans through R and obtain the $MinPts$-dist and $MinPts$ nearest neighborhood of each point. Then the reachability distance and local reachability density are computed in a second pass over R. Finally a pass through R is needed to compute the LOF value for each of the points in R.

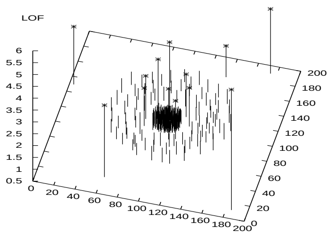
Aside from improving efficiency, GridLOF can handle datasets with overlapping clusters with different densities for which LOF algorithm fails to work appropriately. Two example datasets are shown in Figure 5. For these examples, since the LOF value of an object is the measure of the relative degree of isolation of that object with respect to its surrounding neighborhood, points of the less dense cluster that are close to the border points of the denser cluster will be wrongly regarded as local outliers.

GridLOF does not have this misidentification problem. By partitioning with reasonable equi-width intervals, the

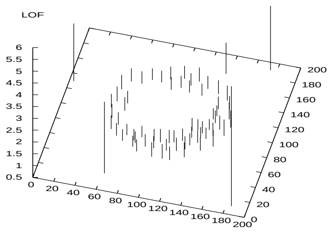


(a) (b)

Figure 5. Example datasets with overlapping clusters of different densities.



(a)



(b)

Figure 6. Example showing the ability of GridLOF to correctly identify outliers.

whole dense cluster and the layer of points in less dense cluster surrounding the dense one are pruned. As a consequence, only the outer boundary points of the less dense clusters are examined and this solves the problem.

Figure 6(a) is the LOF result obtained for the dataset in Figure 5(b). The original LOF algorithm finds the LOF values of every point in the dataset. The top 5% of points with the largest LOF values are indicated by crosses in Figure 6(a). The five outliers have high LOF values, however we find that some points of the less dense cluster which are near the denser cluster have even higher LOF values, and they are misidentified as outliers. This problem is solved by GridLOF algorithm. In Figure 6(b), the result obtained by using GridLOF is shown. Since points of the denser cluster are pruned and most of the points inside the less dense cluster are pruned too, it is easy to distinguish the five outliers from the unpruned points in the dataset and avoid the problem of misidentification of clustering points as outliers.

Selection of w: The correctness of GridLOF method depends on the choice of w. An error will occur if there is an outlier in a grid cell x, and all the neighboring grid cells are non-empty. This can happen if the grid size is large, or w is small.

When an outlier exists in a grid cell x where all neighboring grid cells are occupied, there are two possible cases when considering any two of such neighboring grid cells: either they belong to the same cluster, or they belong to two different clusters. In one possible scenario, a grid cell x with an outlier is inside a cluster, meaning there is a hollow or concave area of a cluster where the outlier is located. Suppose a cluster has a boundary surface. We consider hyper-rectangles defined by ranges of values on each dimension of the data space. Let us define a hollow hyper-rectangle inside a cluster as a hyper-rectangle which is within the boundary of the cluster, containing no cluster points, and with an edge length much greater than the average distances between neighboring points in the cluster. (That is, we do not want to consider any empty spaces in between cluster points as a hollow hyper-rectangle.) For any cluster C, suppose we are given a lower bound e on the edge length of any hollow hyper-rectangle. Then if we set the grid cell edge length to be at most 1/3 of e. an outlier that may exist inside such a hollow can be detected, or it will not be pruned. This can be easily shown by contradiction.

We may consider a second case where a grid cell x with an outlier is surrounded by neighbor grid cells containing points in different clusters. For two clusters A and B, there will be at least one dimension d where the closest points from the two clusters are the furthest apart. Let us call the distance between the closest points at such a dimension the cluster distance for A and B. If we also have a lower bound on the cluster distance for any two clusters, we can set the grid cell edge length to be smaller than one third of this lower bound. Then we shall have an empty neighboring grid cell for a cell containing an outlier even if the outlier is at the narrowest strait between two close clusters.

3. Time Complexity

Mining local outliers by LOF typically requires three passes over the data. A first pass for finding every objects' $MinPts$-dist and $MinPts$-nearest neighborhoods. Then a second pass to compute the reachability distance and local reachability density of each object. Finally LOF value of all objects in the database is calculated in the third pass. For LOF$'$ computation, the second pass described above is eliminated since the reachability distance and local reachability density are not needed. As a consequence, only two passes over the data is needed. Also note that the second pass that is saved is more complex than the first pass, since for each data point, it requires the collection of information for the neigborhood of a data point.

The algorithm for finding LOF$''$ is nearly the same as that of LOF except for the first pass. No extra pass is required since the $MinPts_2$-distance neighborhood can be obtained directly from $MinPts_1$-distance neighborhood.

For LOF, LOF$'$ and LOF$''$, suppose that objects in a database D of size N is being examined, totally there are N $MinPts$-nearest neighbors queries in the first pass. The complexity ranges from $O(N \log N)$ to $O(N^2)$ depending on the use of indexing structure and dimensionality of data.

Although the runtime complexity of GridLOF also depends on the number of $MinPts$-nearest neighbors queries, because most of the points residing deep inside clusters are pruned, the total number of $MinPts$-nearest neighbors queries is much less than N. It can be observed from analysis and from experiments that the runtime of the preprocessing step of partitioning and data pruning is being dominated by the querying step.

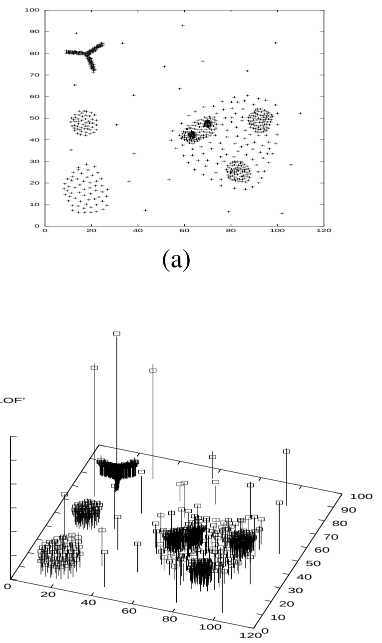
4. Experiments

Several programs were written in C++ language to calculate LOF, LOF$'$ and LOF$''$ by the original LOF algorithm as stated in [8]. In addition, the GridLOF algorithm is implemented in a C++ program. Experiments on these programs are made under the computing environment of a Sun Enterprise E4500 running Solaris 7 with 12 UltraSPARC-II 400MHz and 8 GB RAM.

For all the formulations, an X-tree [6] indexing structure is provided for speeding up the $MinPts$-nearest neighbors queries. X-tree is chosen because it is an index structure for efficient query processing of high-dimensional data and building time of the X-tree index structure is considerably small.

There are two types of data used in the experiments. The first type of data is a set of 2-dimensional datasets created especially to verify the correctness and to demonstrate the idea of our enhancement schemes. 2-dimensional datasets are also used for better visualization.

The second type is a set of data generated by the synthetic data generator which generates data following the synthetic data generation suggested in [3] with some modifications, so that clusters are associated with the whole data space, instead of associating with a subspace. The generated clusters have arbitrary orientation regarding the whole data space and data objects in each cluster follow the normal distribution with small variance. Variances used are randomly drawn from an exponential distribution. Outliers are generated by restricting the distances between outliers and each cluster to be greater than five standard deviation in all dimensions. The number of outliers generated is set to be 0.5 percent of the size of the datasets.



(a)

(b)

Figure 7. (a) Sample dataset $DB1$. (b) LOF′ and LOF results of $DB1$.

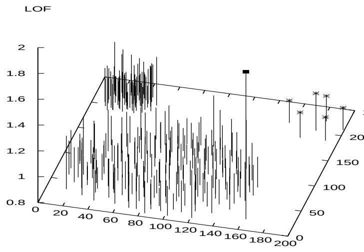
LOF′: In order to verify the correctness of the newly proposed LOF′ formulation, a sample 2-dimensional dataset $DB1$ is used for better visualization. $DB1$ is a 2-D dataset with 640 points within. The original datasets is illustrated in Figure 7(a). In Figure 7(b), the corresponding LOF′ and LOF values for $MinPts = 5$ are plotted in the same figure for ease of comparison. The LOF′ values are indicated by the impulse lines while the LOF values of the corresponding points are indicated by a square point on the impulse lines. By investigating the plotted graph in detail, it can be observed that the proposed LOF′ formulation captures the same degree of local outlier-ness as the original LOF formulation does. For points whose LOF values are high, they also possess high LOF′ values, and vice versa. For different $MinPts$ values used, similar experimental results are obtained. To further investigate the accuracy of our LOF′ formulation, the LOF′ and LOF values for the different sets of synthetic data are studied, with different dimensionality, dataset size, and the value of k. It is found that the results are always very close and in most cases, they are identical.

Based on the fact that LOF′ requires one pass over the data less than LOF, it is computationally more efficient when compared to LOF. In Table 1, experimental results in counting the total number of page access required for different datasets in traversing all data points in the index (we use X-tree) once is shown. Since we have saved a pass of the data that requires more than a simple scan of data, the actual page access improvements would be more than what is shown in the table.

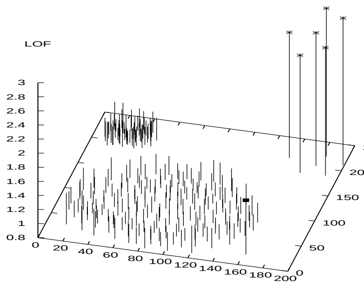
LOF′′: In order to verify the ability of LOF′′ in capturing small groups of outliers, a set of 2-dimensional data called $DB2$ is generated to illustrate the scenario as in Figure 3(a). $DB2$ is a 2-D dataset with 250 points, there is a local outlier o' at the bottom right hand corner and a small group of outliers G at the top right hand corner of the graph. Original LOF plots with different $MinPts$ values used are shown in Figure 8. From the plots, it can be seen that LOF is inadequate to capture the set of outliers G and the local outlier o' at the same time. In Figure 8(a), $MinPts$ is set to 5. In this case LOF successfully point out the local outlier-ness of o', however all points in the outlier group G have LOF values approximately equal to 1, that means G is determined as a small cluster instead of outliers.

By increasing the $MinPts$ value used as in Figure 8(b), Figure 8(c) and Figure 8(d), LOF is capable to uncover G as outliers. However, this lowers the degree of outlier-ness of some local outliers, which are relatively close to some cluster. This problem can be solved by using two $MinPts$ values as stated if LOF''. In the dataset $DB2$, $MinPts_2$ and $MinPts_1$ values used are 5 and 10 respectively. The corresponding LOF′′ results are plotted in Figure 3(b) which correctly identifies all the outliers.

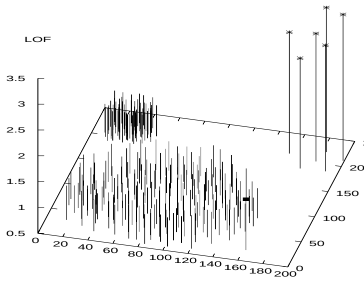
GridLOF: The GridLOF algorithm is developed to perform pruning upon the dataset. Two datasets $DB1$ and $DB3$ are used to illustrate the pruning performed by GridLOF. Two sample 2-dimensional datasets are used for ease of visualization and understanding. $DB1$ is the aforementioned dataset used in the LOF′ experiment, and is illustrated in Figure 7(a). $DB3$ is a more complex dataset with overlap-



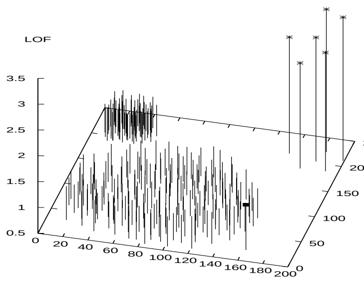
(a) $MinPts = 5$



(b) $MinPts = 10$



(c) $MinPts = 15$

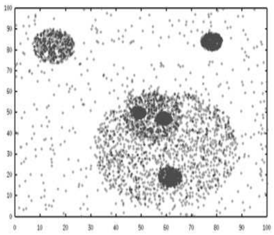


(d) $MinPts = 20$

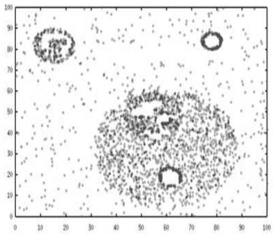
Figure 8. LOF plot of $DB2$ with different $MinPts$.

Size of database (10^3)	Number of page access (10^3) for		
	1 pass	LOF'	LOF
60	66.6	188.0	254.6
80	88.9	262.2	351.1
100	111.0	338	449.0
120	133.2	416.2	549.4
140	155.4	496.0	651.4
200	222.0	742.9	964.9
500	554.9	2077.7	2632.6

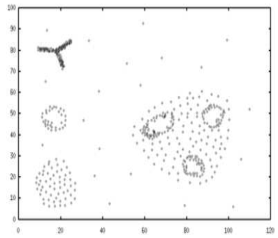
Table 1. Page access of indexes for databases with different size.



(a) Sample dataset $DB3$



(b) $DB3$ after pruning

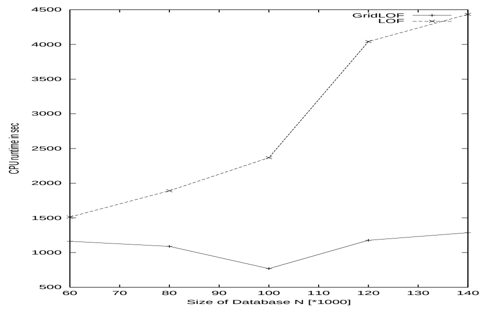


(c) $DB1$ after pruning

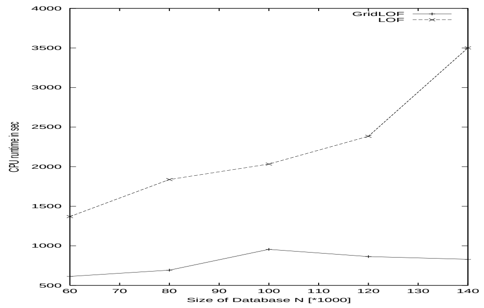
Figure 9. Datasets after the pruning step in GridLOF.

ping clusters of different densities showing a hierarchical structure. $DB3$ is shown in Figure 9(a) which is a dataset with 5000 points. Figure 9(b) is the resulting $DB3$ after being pruned by GridLOF. The LOF values of this set of remaining points are to be computed in the later steps in GridLOF. From this figure, it can be seen that GridLOF succeeds in pruning noisy dataset with highly complex structure. In $DB3$, the range of points are $0 < 100$ for both dimensions. By partitioning each dimension into 100 equi-width intervals and applying the pruning, 2079 points are pruned as stated in Figure 9(b).

The resulting $DB1$ after the pruning step in GridLOF is given in Figure 9(c). The range of points are $0 < 100$ for both dimensions in $DB1$ and each dimension is partitioned into 50 equi-width intervals. This yields a set of 483 points



(a) number of dimensions=2



(b) number of dimensions=3

Figure 10. CPU runtime of GridLOF and LOF for datasest.

in Figure 9(c) for further LOF computation.

Experiments were performed to examine the runtime complexity difference between GridLOF and LOF on a set of synthetic data with size (N) ranging from 60000 to 140000. Each CPU runtime obtained is the average time required for five experimental runs on the datasets. With different setting of dimensionalities, number of clusters and $MinPts$ values, we find that GridLOF can reduce the computation time in all cases. It works best for dimensionality less than 6 and is very good for 2 dimensional and 3 dimensional datasets. In Figure 10, the CPU runtimes for 2 dimensional and 3 dimensional datasets with 8 clusters of varying dataset size (N) are shown.

5. Conclusion

Recently, the topic of outlier detection in data mining arouses attention because of their potential usage in many applications. In this paper, the LOF formulation and algorithm for grading the degree of outlier-ness for local outlier detection is examined. Three enhancements aiming to address different problems of LOF are introduced, with two new definitions for the degree of outlier-ness, LOF$'$ and LOF$''$, and an algorithm GridLOF which adds a pruning step before the original LOF algorithm. By formal analysis and experimental results, the three enhancement techniques are shown to work effectively with advantages over LOF in different aspects.

Our future work can include the following. In this paper, we have separately considered three proposed methods and compared with original LOF method, this will let us identify their individual effects. In the future, we can combine the techniques and hopefully that would combine their different strengths.

In our GridLOF algorithm, a parameter indicating the number of intervals to be partitioned in each dimension is needed. In our further work, we hope to include in our GridLOF algorithm the ability to self determine the appropriate interval values. For example, GridLOF can starts to use a small interval number to partition each dimension and further fine partition some intervals based on certain judgement. This yields an uneven partitioning on the data space which can achieve more effective pruning and can be more adaptive to datasets with great variations in density.

Further work can also be done on outlier detection of datasets with extremely high dimensionality and datasets with clusters in subspaces. In [2], ideas similar to the projected clustering are used. Through studying the behavior of projections from the dataset, outliers are identified. Combining the projection technique with the use of evolutionary algorithm, density-based outliers are found with the corresponding subspace dimensions that these outliers showing their most deviating behavior. We can examine similar issues in the future for the outlier detection problem.

ACKNOWLEDGEMENT The authors thanks Mr. M. M. Breunig of the Database Group of the University of Munich in providing two datasets ($DB1$ and $DB3$) for experimental use. This research is supported by the Hong Kong RGC Earmarked Grant UGC REF.CUHK 4179/01E. This research is also supported by the Chinese University of Hong Kong RGC Research Grant Direct Allocation, Proj ID 2050279.

References

[1] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 61–72, Philadephia, Pennsylvania, U.S.A., 1999.

[2] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 37–46, Santa Barbara, California, U.S.A., 2001.

[3] C. C. Aggarwal and P. S. Yu. Redefining clustering for high-dimensional applications. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):210–255, 2002.

[4] M. Ankerst, M. M. Breunig, H.-P.Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In

Proceedings of ACM SIGMOD International Conference on Management of Data, pages 49–60, Philadephia, Pennsylvania, U.S.A., 1999.

[5] V. Barnet and T.Lewis. *Outliers in Statistical Data*. John Wiley, 1994.

[6] S. Berchtold, D. A. Keim, and H.-P. Kreigel. The X-tree: An index structure for high-dimensional data. In *Proceedings of 22nd International Conference on Very Large Data Bases*, pages 28–39, Bombay, India, 1996.

[7] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. OPTICS-OF: Identifying local outliers. In *Proceedings of 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, pages 262–270, Prague, Czech Republic, 1999.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 93–104, Dallas, Texas, U.S.A., 2000.

[9] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Portland, Oregon, 1996.

[10] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[11] A. Hinneburg and D. A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of 25th International Conference on Very Large Data Bases*, pages 506–517, Edinburgh, Scotland, U.K., 1999.

[12] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proceedings of 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 293–298, San Francisco, California, U.S.A., 2001.

[13] T. Johnson, I. Kwok, and R. T. Ng. Fast computation of 2-dimensional depth contours. In *Proceedings of 4th International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 224–228, New York City, New York, U.S.A., 1998.

[14] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of 24th International Conference on Very Large Data Bases*, pages 392–403, New York City, New York, U.S.A., 1998.

[15] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of 20th International Conference on Very Large Data Bases*, pages 144–155, Santiago de Chile, Chile, 1994.

[16] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 427–438, Dallas, Texas, U.S.A., 2000.

[17] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[18] J. Tang, Z. Chen, A. Fu, and D. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Proceedings of Advances in Knowledge Discovery and Data Mining 6th Pacific-Asia Conference (PAKDD 2002)*, pages 535–548, Taipei, Taiwan, 2002.

[19] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Quebec, Canada, 1996.