Ranking Outliers Using Symmetric Neighborhood Relationship

Wen Jin[1], Anthony K. H. Tung[2], Jiawei Han[3], and Wei Wang[4]

[1] School of Computing Science,Simon Fraser University
wjin@cs.sfu.ca
[2] Department of Computer Science, National University of Singapore
atung@comp.nus.edu.sg
[3] Department of Computer Science, Univ. of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu
[4] Department of Computer Science, Fudan University
weiwang1@fudan.edu.cn

Abstract. Mining outliers in database is to find exceptional objects that deviate from the rest of the data set. Besides classical outlier analysis algorithms, recent studies have focused on mining **local outliers**, i.e., the outliers that have density distribution significantly different from their neighborhood. The estimation of density distribution at the location of an object has so far been based on the density distribution of its k-nearest neighbors [2, 11]. However, when outliers are in the location where the density distributions in the neighborhood are significantly different, for example, in the case of objects from a sparse cluster close to a denser cluster, this may result in wrong estimation. To avoid this problem, here we propose a simple but effective measure on local outliers based on a symmetric neighborhood relationship. The proposed measure considers both neighbors and reverse neighbors of an object when estimating its density distribution. As a result, outliers so discovered are more meaningful. To compute such local outliers efficiently, several mining algorithms are developed that detects top-n outliers based on our definition. A comprehensive performance evaluation and analysis shows that our methods are not only efficient in the computation but also more effective in ranking outliers.

1 Introduction

From a knowledge discovery standpoint, outliers are often more interesting than the common ones since they contain useful information underlying the abnormal behavior. Basically, an outlier is defined as an exceptional object that deviates much from the rest of the dataset by some measure. Outlier detection has many important applications in fraud detection, intrusion discovery, video surveillance, pharmaceutical test and weather prediction. Various data mining algorithms [1–3, 8, 10–13, 15, 18, 17, 20, 21, 23–25] for outlier detection were proposed. The outlierness of an object typically appears to be more outstanding with respect to its local neighborhood. For example, a network intrusion might cause a significant spike in the number of network events within a low traffic period, but this spike might be insignificant when a period of high network traffic is also included in the comparison. In view of this, recent work on outlier detection has been focused on finding **local outliers**, which are essentially objects that have significantly lower density [5] than its local neighborhood [2]. As an objective measure, the degree of outlierness of an object p is defined to be **the ratio of its density and the average density of its neighboring objects** [2]. To quantify what are p's neighboring objects, users must specify a value k, and neighboring objects are defined as objects which are not further from p than p's k^{th} nearest objects [6]. As an example, let us look at Figure 1 in which k is given a value of 3. In this case, the three neighboring objects of p will have higher density than p and thus p will have a high degree of outlierness according to the definition in [2]. This is obviously correct based on our intuition.

Unfortunately, the same cannot hold in more complex situation. Let us look at the following example.

Example 1: We consider Figure 2 in which p is in fact part of a sparse cluster C_2 which is near the dense cluster C_1. Compared to objects q and r, p obviously displays less outlierness. However, if we use the measure proposed in [2], p could be mistakenly regarded to having stronger outlierness in the following two cases:

Case I: The densities of the nearest neighboring objects for both p and q are the same, but q is slightly closer to cluster C_1 than p. In this case, p will have a stronger outlierness measure than q, which is obviously wrong.

[5] The density of an object p is defined as $1/k_{dist}(p)$ where k is a user-supplied parameter and $k_{dist}(p)$ is the distance of the k^{th} nearest object to p.

[6] Note that p 's k^{th} nearest neighbor might not be unique and thus p could have more than k neighboring objects.

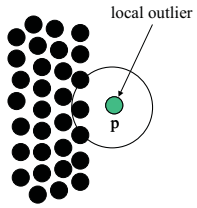

Fig. 1. A local outlier, p

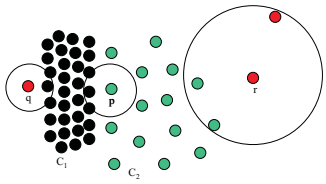

Fig. 2. Comparing the outlierness of p, q, r

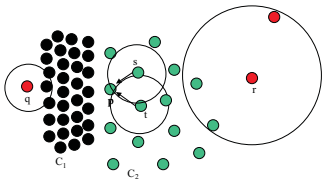

Fig. 3. Taking RNNs of p into account

Case II: Although the density of r is lower than p, the average density of its neighboring objects (consisting of 2 objects from C_2 and an outlier) is less than those of p. Thus, when the proposed measure is computed, p could turn out to have a stronger outlierness measure than r, which again is wrong.

Note that the two cases we described are not only applicable to p but also to the two objects above and below p. In general, any member of C_2 that is lying near the border between the two clusters could have been misclassified as showing stronger outlierness than q and r. □

From these examples, we can see that existing outlierness measure is not easily applicable to complex situation in which the dataset contains multiple clusters with very different density distribution. The reason for the above problem lies in the inaccurate estimation for the density distribution of an object's neighborhood. In Figure 2, although p belongs to cluster C_2, it is closer to cluster C_1, and thus the estimation of p's neighborhood density distribution is derived from C_1 instead of C_2.

To get a better estimation of the neighborhood's density distribution, we propose to take both the nearest neighbors **(NNs)** and reverse nearest neighbors **(RNNs)** [14] into account. The RNNs of an object p are essentially objects that have p as one of their k nearest neighbors. By considering the symmetric neighborhood relationship of both NN and RNN, the space of an object influenced by other objects is well determined, the densities of its neighborhood will be reasonably estimated, and thus the outliers found will be more meaningful. As a simple illustration in Figure 3 which depicts the same situation as Figure 2, we show that p has two RNNs: s and t. This distinguishes it from q which has no RNNs, and r which has only an outlier as its RNNs. Later on in this paper, we will show how such an observation can be incorporated to ensure that the outlierness measure for p will indicate that it is a weaker outlier than both q and r. We now summarize our contributions in this paper:

(1) We propose the mining of outliers based on a symmetric neighborhood relationship. The proposed method considers the *influenced space* considering both neighbors and reverse neighbors of an object when estimating its neighborhood density distribution. To the best of our knowledge, previous work of outlier detection has not considered the effect of RNN. Such a symmetric relationship between NNs and RNNs will make the outlierness measurement more robust and semantically correct comparing to the existing method.

(2) We assign each object of database the degree of being **INFLuenced Outlierness(INFLO)**. The higher **INFLO** is, the more likely that this object is an outlier. The lower **INFLO** is, the more likely that this object is a member of a cluster. Specifically, $INFLO \approx 1$ means the object locates in the core part of a cluster.

(3) We present several efficient algorithms to mining top-n outliers based on **INFLO**. To reduce the expensive cost incurred by a large number of KNN and RNN search, a two-way search method is developed by dynamically pruning those objects with value $INFLO \approx 1$ during the search process. Furthermore, we take advantage of the micro-cluster [11] technique to compress dataset for efficient symmetric queries, and use two-phase pruning method to prune out those objects which will never be among the top-n outliers.

(4) Last but not the least, we give a comprehensive performance evaluation and analysis on synthetic and real data sets. It shows that our method is not only efficient and scalable in performance, but also effective in ranking meaningful outliers.

The rest of this paper is organized as follows. In section 2, we formally define a new outlier measurement using symmetric neighborhood relationship and discuss some of its important properties. In section 3, we propose efficient methods for mining and ranking outliers in databases. In section 4, a comprehensive performance evaluation is made and the results are analyzed. Related work is discussed in section 5 and section 6 concludes the paper.

2 Influential Measure of Outlierness by Symmetric Relationship

In this section, we will introduce our new measure and related properties. The following notations will be used in the remaining of the paper. Let D be a database of size N, let p, q and o be some objects in D, and let k be a positive integer. We use $d(p, q)$ to denote the **Euclidean distance** between objects p and q.

Definition 1. (k-distance and nearest neighborhood of p) *The k-distance of p, denoted as $k_{dist}(p)$, is the distance $d(p, o)$ between p and o in D, such that: (1) at least for k objects $o' \in D$ it holds that $d(p, o') \leq d(p, o)$, and (2) at most for $(k-1)$ objects $o' \in D$ it holds that $d(p, o') < d(p, o)$. The k-nearest neighborhood of p, $NN_k(p)$ is a set of objects X in D with $d(p, X) \leq k_{dist}(p)$: $NN_k(p) = \{X \in D \backslash \{p\} |\ d(p, X) \leq k_{dist}(p))\}$.* □

Definition 2. (local density of p) *The density of p, denoted as $den(p)$, is the inverse of the k-distance of p, i.e., $den(p) = 1/k_{dist}(p)$.* □

Although the k-nearest neighbor of p may not be unique, $k_{dist}(p)$ is unique. Hence, the density of p is also unique. The nearest neighbor relation is not symmetric. For a given p, the nearest neighbors of p may not have p as one of their own nearest neighbors. As we discussed in Section 1, these neighbors should also be taken into account when the outlierness of p is computed. Therefore, we introduce the concept of reverse nearest neighbors [14] as follows.

Definition 3. (reverse nearest neighborhood of p) *The reverse k-nearest neighborhood RNN is an inverse relation which can be defined as: $RNN_k(p) = \{q | q \in D, p \in NN_k(q)\}$.* □

For any object $p \in D$, NN_k search always returns at least k results, while the RNN can be empty, or have one or more elements. *By combining $NN_k(p)$ and $RNN_k(p)$ together in a novel way, we form a local neighborhood space which will be used to estimate the density distribution around p.* We call this neighborhood space the **k-influence space for** p, denoted as $IS_k(\mathrm{p})$.

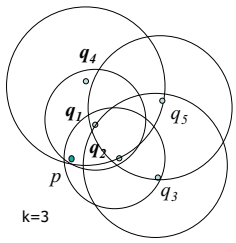

Fig. 4. RNN and Influence Space

Example 2: Figure 4 gives a simple description of how to obtain RNN in $\{p,\ q_1,\ q_2,\ q_3,\ q_4,\ q_5\}$ when $k = 3$. $NN_k(q_1) = \{p,\ q_2,\ q_4\}$, $NN_k(q_2) = \{p,\ q_1,\ q_3\}$, $NN_k(q_3) = \{q_1,\ q_2,\ q_5\}$, $NN_k(q_4) = \{p,\ q_1,\ q_2,\ q_5\}$, $NN_k(q_5) = \{q_1,\ q_2,\ q_3\}$. During the search of k-nearest neighbors of p, q_1, q_2, q_3, q_4 and q_5, $RNN_k(p) = \{q_1,\ q_2,\ q_4\}$ is incrementally built. Similarly, $RNN_k(q_1)$, $RNN_k(q_2)$, $RNN_k(q_3)$, $RNN_k(q_4)$ and $RNN_k(q_5)$ are found. Note that $NN_k(p) = \{q_1,\ q_2,\ q_4\} = RNN_k(p)$ (here $IS_3(p) = \{q_1,\ q_2,\ q_4\}$). If the value of k changes, $RNN_k(p)$ may not be equal to $NN_k(p)$, or totally different. □

Unlike the nearest neighborhood, the influence space for an object p contains influential objects affecting p, more precisely estimating density around p 's neighborhood w.r.t. these objects.

Definition 4. (influenced outlierness of p) *The influenced outlierness is defined as: $INFLO_k(p) = \frac{den_{avg}(IS_k(p))}{den(p)}$ where $den_{avg}(IS_k(p)) = \frac{\sum_{o \in IS_k(p)} den(o)}{|IS_k(p)|}$.* □

$INFLO$ is the ratio of the average density of objects in $IS_k(p)$ to p's local density. p's $INFLO$ will be very high if its density is much lower than those of its influence space objects. In this sense, p will be an outlier. We can assert *p is a local outlier if $INFLO_k(p) > t$ where $t \gg 1$.* On the other hand, objects with density very close to those in their influence space will have $INFLO \approx 1$. Without loss of generality, we assume that for any local outlier object q ($INFLO(q) > t$), we have $|RNN_k(q)| < j$(a value $< k$), and any non-local outlier p cannot belong to $RNN_k(q)$.

Lemma 1. *Given any object $p, q \in D$, if $\max_{p' \in IS_k(p)} k_{dist}(p') < \min_{q' \in IS_k(q)} k_{dist}(q')$ then $den_{avg}(IS_k(p)) > den_{avg}(IS_k(q))$.*

Proof. $den_{avg}(IS_k(p)) = \frac{\sum_{p' \in IS_k(p)} den(p')}{|IS_k(p)|} \geq \frac{|IS_k(p)| \cdot 1/\max_{p' \in IS_k(p)} k_{dist}(p')}{|IS_k(p)|} > \frac{|IS_k(p)| \cdot 1/\min_{q' \in IS_k(q)} k_{dist}(q')}{|IS_k(p)|}$
$= \frac{|IS_k(q)| \cdot 1/\min_{q' \in IS_k(q)} k_{dist}(q')}{|IS_k(q)|} \geq \frac{\sum_{q' \in IS_k(q)} den(q')}{|IS_k(q)|} = den_{avg}(IS_k(q))$ □

Lemma 2. *For $p \in D$, if $\frac{k_{dist}(p)}{max_{q' \in IS_k(p)} k_{dist}(q')} > t$, then p is a local outlier.*

Proof. $INFLO_k(p) = \frac{den_{avg}(IS_k(p))}{den(p)} = \frac{\sum_{p' \in IS_k(p)} den(p')}{|IS_k(p)| \cdot den(p)} \geq \frac{|IS_k(p)| \cdot 1 / \max_{p' \in IS_k(p)} k_{dist}(p')}{|IS_k(p)| \cdot den(p)} = \frac{k_{dist}(p)}{max_{q' \in IS_k(p)} k_{dist}(q')} > t.$ □

Lemma 3. *For $p \in D$, if there exists $r \in RNN_k(p)$ such that $k_{dist}(p) \leq k_{dist}(r) \leq k_{dist}(q)$ where $q \in NN_k(RNN_k(p))$, $r \neq q$ and $\frac{den_{avg}(IS_k(q))}{k_{dist}(p)} > t$, then p is a local outlier.*

Proof. Since $k_{dist}(p) \leq k_{dist}(q)$, so $q \in NN_k(p) \cap RNN_k(p)$, thus $\max_{p' \in IS_k(p)} k_{dist}(p') = \max_{p' \in NN_k(p) \cup RNN_k(p)} k_{dist}(p')$ $= \max_{p' \in RNN_k(p)} k_{dist}(p') \leq k_{dist}(r) = \min_{q' \in NN_k(q) \cup RNN_k(q)} k_{dist}(q') = \min_{q' \in IS_k(q)} k_{dist}(q')$. Based on Lemma 1, $den_{avg}(IS_k(p)) > den_{avg}(IS_k(q))$, so $INFLO_k(p) = \frac{den_{avg}(IS_k(p))}{den(p)} = den_{avg}(IS_k(p)) \cdot k_{dist}(p) > den_{avg}(IS_k(q)) \cdot k_{dist}(p) = \frac{den_{avg}(IS_k(q))}{k_{dist}(p)} > t$. So p is a local outlier. □

Lemma 4. *For $p \in D$, the value of $\frac{RNN_k(p) \cap NN_k(p)}{NN_k(p)}$ is proportional to the density value of p.*

Proof. Because the size of any cluster should be larger than k (usually $k = MinPts$ [2]), the higher the above ratio, the more influence for the local neighborhood to the object, and the higher density for this object. □

3 Mining Algorithms for Influence Outliers using Symmetric Relationship

Essentially, mining influenced outliers is based on the problem of finding the influence space of objects, which is in KNN and RNN. In this section, we provide several techniques for finding influenced outliers, including the naive index-based method, the two-way search method and the micro-cluster method.

3.1 A Naive Index-based method

Finding influence outliers requires the operations of KNN and RNN for each object in the database, so the search cost is huge. If we maintain all the points in a spatial index like R-tree, the cost of range queries can be greatly reduced by the state-of-the-art pruning technique [19]. Suppose that we have computed the temporary $k_{dist}(p)$ by checking a subset of the objects, the value that we have is clearly an upper bound for the actual $k_{dist}(p)$. If the minimum distance between p and the MBR [7] of a node in the R-tree (called MinDist(p, MBR)) is greater than the $k_{dist}(p)$ value that we currently have, none of the objects in the subtree rooted under the node will be among the k-nearest neighbors of p. This optimization can prune entire sub-tree containing points irrelevant to the KNN search for p. Along with the search of KNN, the RNN of each object can be dynamically maintained in R-tree [14]. After building the index of KNN and RNN, the outlier influence degree can be calculated and ranked. The following algorithm is to mining top-n $INFLO$ by building KNN and RNN index within R-tree.

Algorithm 1 Index-based method.
Input: k, D, n, the *root* of R-tree.
Output: Top-n $INFLO$ of D.
Method:

```
1. FOR each object p ∈ D DO
2.   MBRList = root; k_dist(p) = ∞; heap = 0;
3.   WHILE (MBRList) != empty DO
4.     Delete 1st MBR from MBRList;
5.     IF (1stMBR is a leaf) THEN
6.       FOR each object q in 1stMBR DO
7.         IF (d(p,q) < k_dist(p)) AND (heap.size < k) THEN
8.           heap.insert(q);
9.           k_dist(p) = d(p, heap.top);
10.    ELSE
11.      Append MBR's children to MBRList;
12.      Sort nodeList by MinDist;
```

[7] Minimum bounding rectangle

13. FOR each MBR in MBRList DO
14. IF ($k_{dist}(p) \leq MinDist(p, MBR)$) THEN
15. Remove Node from MBRList;
16. FOR each object q in heap DO
17. Add q into $NN_k(p)$, add p into $RNN_k(q)$;
18.FOR each object $p \in D$ DO
19. Ascending sort top-n INFLO from KNN and RNN;

Here MBRs are stored in ascending order based on MinDist(p, MBR), as lines 11-12. The algorithm searches KNN_p only in those MBRs with MinDist smaller than the temporary $k_{dist}(p)$, otherwise these MBRs are pruned (lines 13-15). If any nearer object is located (lines 6-7), it will be inserted into the heap and the current $k_{dist}(p)$ will be updated (lines 8-9). Whenever $NN_k(p)$ are found, they are stored as p's nearest neighbors. Meanwhile, it need store p as a reverse nearest neighbor (lines 16-17). Finally, $INFLO$ is calculated based on KNN and RNN index.

3.2 A Two-way search method

Two major factors hamper the efficiency of the previous algorithm. First, for any object p, RNN space cannot be determined unless all the other objects have finished nearest neighbor search. Second, large amount of extra storage is required on R-tree, where each object at least stores k pointers of its KNN, and stores m pointers (m varies from 0 to $o(k)$) for its RNN. The total space cost will be prohibitive. Therefore, we need reduce the computation cost for RNN and corresponding storage cost. By analyzing the characteristics of $INFLO$, it is clear that any object as a member of a cluster must have $INFLO \approx 1$ even without $INFLO$ calculation. So we can prune off these cluster objects, saving not only the computation cost but also the extra storage space.

Theorem 1. *For $p \in D$, if for each object $q \in NN_k(p)$, it always exists $p \in NN_k(q)$, then $INFLO_k(p) \approx 1$.*

Proof. Because for each $q \in NN_k(p)$, $p \in NN_k(q)$, p and its nearest neighbors are close to each other. They are actually in a mostly mutual-influenced neighborhood. Since k is potentially the number of objects forming a cluster, under this circumstance, p resides in core part of a cluster. □

To apply this theorem, we will first search p's k-nearest neighbor, then dynamically find the NN_k for each of these nearest neighbors. If $NN_k(NN_k(p))$ still contains p, which shows p is in a closely influenced space and is a core object of a cluster ($INFLO_k(p) \approx 1$), we can prune p immediately without searching corresponding RNN. Such a *early pruning* technique will improve the performance significantly. The two-way search algorithm is given as follows:

Algorithm 2 A Two-way search method.
Input: k, D, n, the *root* of R-tree, a threshold M.
Output: Top-n $INFLO$ of D
Method:

1. FOR each $p \in D$ DO
2. $count = |RNN_k(p)|$;
3. IF $unvisited(p)$ THEN
4. $S = getKNN(p)$; *//search k-nearest neighbors*
5. $unvisited(p) = FALSE$;
6. ELSE
7. $S = KNN(p)$; *//get nearest neighbors directly*
8. FOR each object $q \in S$ DO
9. IF $unvisited(q)$ THEN
10. $T = getKNN(q)$; $unvisited(q) = FALSE$;
11. IF $p \in T$ THEN
12. Add q into $RNN_k(p)$;
13. Add p into $RNN_k(q)$;
14. $count++$;
15. IF $count \geq |S| * M$ THEN *//M is a threshold*
16. Label p pruned mark;

17.FOR each object $p \in D'$ DO $//D'$ *is unpruned database*
18. Ascending sort top-n INFLO from KNN and RNN;

The algorithm aims to search and prune objects that are likely to have low $INFLO$, thus avoid unnecessary RNN search. The $|RNN_k(p)|$ is initialized to 0 for p. Search process is taken two directions, that is, from one object to its nearest neighbors, then to the new nearest neighbors (lines 8-14). If for p's nearest neighbors, their nearest neighbors' spaces contain p, or most of them contain p, p is a core object of a cluster and cannot be ranked as top-n outliers, and can be pruned (lines 15-16). Finally, top-n $INFLOs$ are calculated (lines 17-18).

3.3 A Micro-cluster-based method

In order to further reduce the cost of distance computation, we introduce micro-cluster to represent close objects [11] so that the number of k-nearest neighbor search will be greatly reduced. The upper and lower bound of k-distance for each micro-cluster can be estimated in influenced space. Under the guidance of the two-way search, those micro-clusters which actually are "core parts" of clusters can be pruned and top-n outliers are ranked in the remaining dataset.

Definition 5. (MicroCluster) *The MicroCluster C for a d-dimensional dataset X is defined as the $(3 \cdot d+2)$-tuple $(n, \overline{CF1(C)}, \overline{CF2(C)}, \overline{CF3(C)}, r)$, where $\overline{CF1}$ and $\overline{CF2}$ each corresponds to the linear sum and the sum of the squares of the data values for each dimension respectively. The number of data points $|C|$ is maintained in n, the centroid of $\overline{X_1} \ldots \overline{X_n}$ is $\overline{CF3(C)} = \frac{\overline{CF1(C)}}{n}$. The radius of the MicroCluster is $r = \max_{j=1}^{n} \sqrt{(\overline{X_j} - \overline{CF3(C)})^2}$.* □

[26] introduced an efficient clustering algorithm, BIRCH, with good linear scalability to the size of database, we borrow its basic idea to partitioning the database into micro-clusters. The detailed procedure can be referenced in [11]. The following theorem [11] can be used to estimate the lower and upper bound of k-distance of any object.

Theorem 2. *Let $p \in MC(n, c, r)$ and $MC_1(n_1, c_1, r_1), \ldots, MC_l(n_l, c_l, r_l)$ be a set of micro-clusters that could* **potentially** *contain the k-nearest neighbors of p. Each object o_i is treated as a micro-cluster $MC_i(1, o_i, 0)$. Thus we will now have $l + n - 1$ micro-clusters.*

1. *Let $\{d_{Min}(p, MC_1), \ldots, d_{Min}(p, MC_{l+n-1})\}$ be sorted in increasing order, then a lower bound on the k-distance of p, denoted as $\min k_{dist(p)}$ will be $d_{Min}(p, MC_i)$ such that $n_1 + \ldots + n_i \geq k$, and $n_1 + \ldots + n_{i-1} < k$*
2. *Let $\{d_{Max}(p, MC_1), \ldots, d_{Max}(p, MC_{l+n-1})\}$ be sorted in increasing order, then an upper bound on the k-distance of p, denoted as $\max k_{dist(p)}$ will be $d_{Max}(p, MC_i)$ such that $n_1 + \ldots + n_i \geq k$ and $n_1 + \ldots + n_{i-1} < k$.* □

The following is the micro-cluster based algorithm for mining top-n local outliers.

Algorithm 3 Micro-cluster method.
Input: A set of micro-clusters $MC_1, \ldots, MC_l$, M.
Output: Top-n $INFLO$ of D.
Method:

1. FOR each micro-cluster MC_i DO
2. FOR each $p \in MC_i$ Do
3. Get Max/Min of $k_{dist(p)}$; // *based on theorem 2*
4. IF Min $k_{dist}(p) <$ Min$k_{dist}(MC_i)$ THEN
5. Min $k_{dist}(MC_i) =$ Min$k_{dist}(p)$;
6. IF Max $k_{dist}(p) >$ Max$k_{dist}(MC_i)$ THEN
7. Max $k_{dist}(MC_i) =$ Max$k_{dist}(p)$;
8. FOR each micro-cluster MC_i DO
9. $count = |RNN_k(MC_i)|$;
10. IF $unvisited(MC_i)$ THEN
11. $S = getKNN(MC_i)$; //*search k-nearest micro-clusters*
12. $unvisited(MC_i) = FALSE$;
13. ELSE
14. $S = KNN(MC_i)$; //*get nearest micro-clusters directly*

15. FOR each micro-cluster $q \in S$ DO
16. IF $unvisited(q)$ THEN
17. $T = getKNN(q)$; $unvisited(q) = FALSE$;
18. IF Min $k_{dist}(q) \geq$ Max $k_{dist}(MC_i)$ THEN
19. Add q into $RNN_k(MC_i)$;
20. Add MC_i into $RNN_k(q)$;
21. $count++$;
22. IF $count \geq |S| * M$ THEN *//M is a threshold*
23. Label MC_i pruned mark;
24FOR each object $p \in$ unpruned micro-clusters MC' DO
25. Ascending sort top-n INFLO from KNN and RNN;

After building micro-clusters, the process of finding outliers is similar to the two-way search method. We simply treat each micro-cluster as a single object to search KNN. As the number of micro-clusters is much less than that of database objects, the computational cost will be saved a lot. The $|RNN_k(MC_i)|$ is initialized to 0 for each micro-cluster MC_i, and the lower/upper bound of k-distance of each MC_i is derived (lines 1-7) based on theorem 3.2. Then irrelevant objects in micro-clusters which cannot become top-n outliers are pruned if most of the k-nearest micro-clusters of a micro-cluster MC contain MC in their k-nearest micro-clusters as well, then MC will be located in the core part of clusters (lines 20-22) and could be removed. If the lower bound of k-distance for any MC's neighboring micro-cluster q is bigger than the upper bound of that for MC, then q belongs to MC's RNN (lines 18-21). By combining the two-way search and the micro-cluster technique, it achieves a significant improvement in performance.

4 Performance Evaluation

In this section, we will perform a comprehensive experimental evaluation on the efficiency and the effectiveness of our mining algorithm. We will compare our methods with the LOF method in [2] and show that our methods not only achieve a good performance but also identify more meaningful outliers than LOF. We perform tests on both real life data and synthetic data. Our real life dataset is the statistics archive of 2000-2002 National Hockey League (NHL), totally 22180 records with 12 dimensions[8]. Our synthetic datasets are generated based on multiple-gaussian distribution, where the cardinality varies from 1,000 to 1,000,000 tuples and the dimensionality varies from 2 to 18. The tests are run on 1.3GHZ AMD processor, with 512MB of main memory, under Windows 2000 advanced-server operating system. All algorithms are implemented by Microsoft Visual C++ 6.0.

Experiments on Effectiveness To achieve a comprehensive understanding on the effectiveness of the **INFLO** measure, it is necessary to test on a series of datasets with different sizes and dimensions. We generate our dataset with complex density distribution by a mixture of Gaussian distribution. Most outliers detected by our methods are meaningful with good explanations, and some of them cannot be found by LOF. For easily illustrating, we just pick up a portion of 2-dimensional dataset containing a low density cluster A and a high density cluster B in Figure 5. The top-6 outliers are listed by $INFLO$ and LOF respectively in Table 1.

Table 1. Outliers Ranking

Rank	Index	LOF	Index	INFLO
1	147	3.47	147	17.34
2	101	2.80	101	8.899
3	146	2.56	146	8.81
4	50	1.74	1	4.50
5	65	1.57	50	3.52
6	4	1.45	16	3.03

Table 2. Outliers Ranking(INFLO)

Rank	INFLO	Player	Games	Goals	Shoot %
1	25.95	Nurminen	2	1	100
2	12.66	Lemieux	43	35	20.5
3	7.60	Holmstrom	76	16	21.6
4	7.25	Blake	67	19	7.1
5	7.03	Maclnnis	59	12	5.5

Table 3. Outliers Ranking(LOF)

Rank	LOF	Player	Games	Goals	Shoot %
1	5.19	Nurminen	2	1	100
2	2.47	Jagr	81	52	16.4
3	2.61	Lemieux	43	35	20.5
4	2.31	McDonald	16	1	4.8
5	2.31	Skalde	19	1	4.2

Due to the limitation of space, we only show two instances. Table 1 lists the top 6 outliers based on the sample dataset in Figure 5, by both LOF and $INFLO$ measures. The most outstanding outliers can be recognized by

[8] http://www.usatoday.com/sports/hockey/stats/

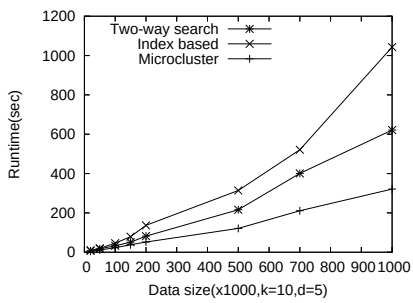

Fig. 7. Runtime vs datasize

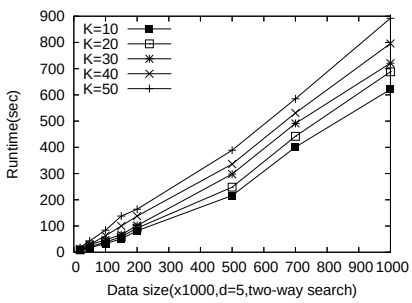

Fig. 8. Effects of k

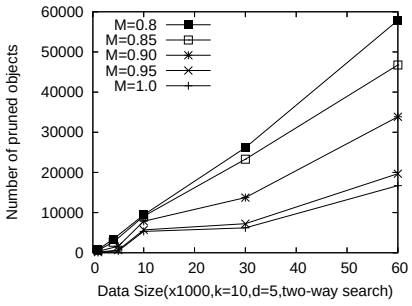

Fig. 9. Pruning results (1)

either measure. In this sample, 50 percentage of the top 6 outliers are the same points by both measures. When n is increased, $INFLO$ will find even more different top outliers from LOF. By visual comparison, the top 6 outliers found by $INFLO$ is more meaningful. Even for the same objects appeared in top-n lists of both measure, their position could be different and $INFLO$-based results are obviously more reasonable. In addition, $INFLO$ can detect outliers which can be overlooked by LOF. For instance, the 50^{th} object and the 4^{th} object have inversely ranking orders by different measure. LOF only considers nearest neighborhood as a density estimation space, and the NN of both the 1^{st} and the 50^{th} objects are in cluster A. Since the distance between the 50^{th} object and A is larger than that of the 1^{st} object and A, so the 50^{th} object with low density is ranked as a higher outlier than the 1^{st} with a high density. While $INFLO$ measure considers both NN and RNN, some objects of B will influence the 50^{th} object, and thus make it being less outlierness than 1^{st} object. It is clear that using $INFLO$ as outlierness measure preserves more semantics than using LOF. Another interesting phenomena in experiments is that $INFLO$ measure gives more rational indication for the outlier degree assignment. As an example, LOF value that are assigned to those bordering objects of a cluster has only a tiny difference with those in the core of a cluster. By $INFLO$, however, the bordering objects will have significantly larger $INFLO$ values than the core part of the same cluster while the value differences are smaller than objects in different cluster. Figure 6 presents such value differences curve by LOF and $INFLO$, in which the difference is evaluated by cluster bordering objects and cluster mean center.

In the following experiments, we run our proposed algorithms with NHL 2000-2002 playoff data (22180 tuples) to rank top-n exceptional players in NHL. The results are compared with those computed from LOF. We varied k from 10 to 50. Projection is done on dataset by randomly selecting dimensions, and the outlierness of hockey players is evaluated. For example, we focus on the statistics data in 3-dimensional subspace of Games played, Goals and Shooting percentage. Due to the limitation of space, we only list top-5 players in Table 2 and Table 3. Lots of interesting and useful information can be found in our examination. For example, there are two players who are listed in both tables as top-5 outliers. Nurminen is the strongest outlier. Although he only took two games and got one point, his 100% shooting percentage dominated other two statistics numbers in comparison. As it happens in the synthetic dataset, we can still find some surprising outliers which cannot be identified by LOF. For example, Rob Blake ranks 4th in our method but is only ranked as the 31^{th} outlier using LOF. Our reasoning for such surprising result is as follows. The variation of shooting percentage is usually small, since only a very few of players can become excellent shooter. Comparing to those players who have similar statistics number in Games Played and Goals dimensions, although Blake's shooting percentage is rather low, Blake is still not too far away from other player when viewed in term of distance. Thus based on LOF measure, Blake's could not be ranked in the top players. But the reason for him being a most exceptional player by $INFLO$ is that there is no such type of player whose Shooting Percentage is so low while having so many Goals. Actually, Blake is the only defence whose number of goals scored is over 12. He must have shot too many times in the games without getting goals.

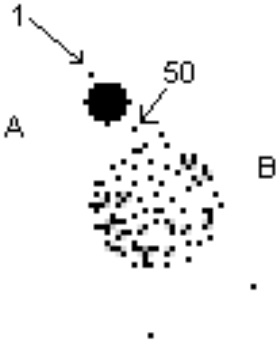

Fig. 5. A dataset

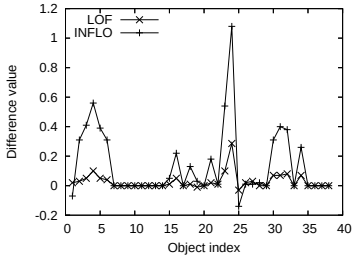

Fig. 6. LOF and INFLO

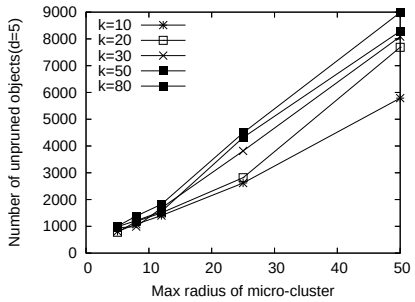

Fig. 10. Pruning results (2)

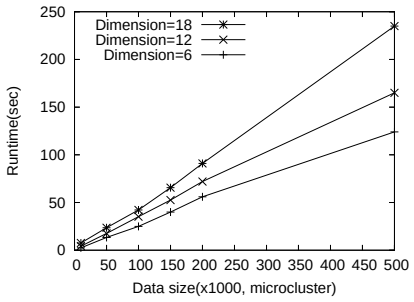

Fig. 11. Effects of dimensionality(1)

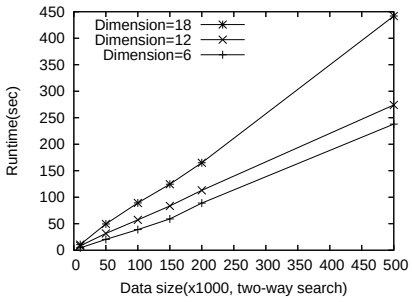

Fig. 12. Effects of dimensionality(2)

Another interesting example is Jaromir Jagr, who scores in the 3^{rd} position and ranks as the second outlier in LOF, but the 24^{th} in our measure. The reason is that even though Jagr has a strong goaling capability and a big fame, there are over twenty players who have higher statistics than him in Shooting Percentage and Games. So objectively, he is not ranked as the most exceptional player during 2000-2002 seasons. *Note that we treat all the hockey data equal in the analysis not like hockey fans who always weigh goals much higher than other factors.*

Efficiency Issues of Experiments We evaluate the efficiency of the proposed mining algorithms by varying the data size, dimension number, k and pruning parameter accordingly. Figure 7 shows the performance curves of different methods, along with the runtime (include CPU time and I/O time) corresponding to different size of dataset with 5 dimensions. It shows that the run time of three methods are similar when the number of tuple is less than 100k. When the data size increases to 200k or so, micro-cluster-based method is the best and the two-way search is better than index-based method. When the size of the load is near to 1000k, swapping operation between R-tree and disk will happen frequently. As such, the performance of index-based method starts to degrade. On the other hand, since the two-way search method does early pruning in the search process, it reduces the total computation cost greatly and saves much time. Micro-cluster method achieves best performance because it not only uses the similar pruning technique as the two-way search, but also reduces the huge number of the nearest neighbor search. So it takes the least time to finish mining outliers in each dataset and scales well to large databases. Unavoidably, this advantage in performance is done by sacrificing some precision in KNN approximation. However, if we adjust the micro-cluster to a suitable size, good quality mining results can still be obtained. Figure 9 shows the pruning results under the different values of threshold M (see the two-way search part in section 3). It can be seen that when M increases, more objects in the database remain unpruned, but the possibility of objects misses to be pruned will be reduced. If M decreases, more objects will be removed, and the cost of future computation will be reduced. It is particularly suitable for top-n case in which only a few objects can become the outlier candidates. Figure 10 shows the pruning results under the different radius of micro-cluster. We can see that when the radius increases, more objects will be inside the micro-clusters, and the difference between lower and upper bound of micro-clusters's k-distance will be larger. As a result, more micro-clusters will not be pruned. Figure 8 presents different performance results of the two-way search method when k varies from 10 to 50. If k is less than 30, the scalability is good with the support of R-tree. When k is over 30, the cost for the nearest search is rather expensive, more MBRs will be searched to compute the distance between the objects and the query object. Thus the running time would increase drastically with the increased number of distance computation.

We also studied the relationship between performance of our algorithm and the number of dimensions, and Figures 11 and 12 show the runtime of our algorithm with different dimensions and varying database with respect to the microcluster-based method and the two-way search method respectively. From the experiment results, we know that the algorithms on smaller dimensionality and data size always have shorter running time. Specifically, when dimensionality is larger than 12, the running time will be increased drastically, thus seriously hindering the efficiency of the algorithms.

5 Related Work

Knorr and Ng [12] initialized the concept of distance-based outlier, which defines an object o being an outlier, if at most p objects are within distance d of o. A cell-based outlier detection approach that partitions the dataset into cells is also presented. The time complexity of this cell-based algorithm is $O(N + c^k)$ where k is dimension

number, N is dataset size, c is a number inversely proportional to d. For very large databases, this method achieves better performance than depth-based method, but still exponential to the number of dimensions. Ramaswamy et al. extended the notion of distance-based outliers by using the distance to the k-nearest neighbor to rank the outliers. An efficient algorithm to compute the top-n global outliers is given, but their notion of an outlier is still distance-based [20].

Some clustering algorithms like CLARANS [16], DBSCAN [6], BIRCH [26], and CURE [7] consider outliers, but only to the point of ensuring that they do not interfere with the clustering process. Further, outliers are only by-products of clustering algorithms, and these algorithms **cannot rank the priority of outliers**.

The concept of local outlier, which assigns each data a local outlier factor **LOF** of being an outlier depending on their neighborhood, was introduced by Breunig et al. [2]. This outlier factor can be used to rank the objects regarding their outlierness. To compute LOF for all objects in a database, O(n*runtime of a KNN query) is needed. The outlier factors can be computed efficiently if OPTICS is used to analyze the clustering structure. A top-n based local outliers mining algorithm which uses distance bound of micro-cluster to estimate the density, was presented in [11].

There are several recent studies on local outlier detection. In [5], [4], three enhancement schemes over **LOF** are introduced, namely LOF' and LOF" and GridLOF, and [22] introduces a connectivity-based outlier factor (COF) scheme that improves the effectiveness of an existing local outlier factor **LOF** scheme when a pattern itself has similar neighborhood density as an outlier. They extensively study the reason of missed outliers by **LOF**, and focus on finding those outliers which are close to some non-outliers with similar densities. While our measure based on the symmetric relationship is not only compatible with their improved measures, but also identifies more meaningful outliers. LOCI [17] addresses the difficulty of choosing values for MinPts in the LOF technique by using statistical values derived from the data itself.

6 Conclusion

In this paper, we discuss the problem with existing local outlier measure and proposed a new measure $INFLO$ which is based on a symmetric neighborhood relationship. We proposed various methods for computing $INFLO$ including the naive-index based method, the two-way pruning method and the micro-cluster based method. Extensive experiments are conducted showing that our proposed methods are efficient and effective on both synthetic and real life datasets.

References

1. C. Aggarwal and P. Yu: Outlier Detection for High Dimensional Data. *SIGMOD* 2001
2. M. M. Breunig, H.P. Kriegel, R.T. Ng, and J.Sander: LOF: Identifying Density-based Local Outliers. *SIGMOD* 2000
3. D. Chakrabarti: AutoPart: Parameter-Free Graph Partitioning and Outlier Detection. *PKDD 2004*
4. Z. X. Chen, A. W. Fu, J. Tang: On Complementarity of Cluster and Outlier Detection Schemes. *DaWaK* 2003
5. A. L. Chiu, A. W. Fu: Enhancements on Local Outlier Detection. *IDEAS* 2003
6. M. Ester, H. P. Kriegel et al.: A Density-based Algorithm for Discovering Clusters in Large Spatial Databases. *KDD* 1996
7. S. Guha, R. Rastogi, and K.Shim: Cure: An Efficient Clustering Algorithm for Large Databases. *SIGMOD* 1998
8. V. Hautamki, I. Krkkinen and P. Frnti: Outlier Detection Using k-nearest Neighbour Graph, *ICPR* 2004.
9. J. W. Han, M. Kamber: Data Mining: Concepts and Techniques. In *Morgan Kaufmann* Publishers.
10. H. Jagadish, N. Koudas, and S. Muthukrishnan: Mining Deviants in a Time Series Database. *VLDB* 1999
11. W. Jin, K. H. Tung and J. W. Han: Mining Top-n Local Outliers in Large Databases. *KDD* 2001
12. E. Knorr, R. Ng: Algorithms for Mining Distance-Based Outliers in Large Datasets. *VLDB* 1998
13. E. Knorr and R. Ng: Finding Intensional Knowledge of Distance-Based Outliers. *VLDB* 1999
14. F. Korn and S. Muthukrishnan: Influence Sets Based on Reverse Nearest Neighbor Queries. *SIGMOD* 2000
15. S. Muthukrishnan, R. Shah, J. S. Vitter: Mining Deviants in Time Series Data Streams. *SSDBM* 2004
16. R. Ng and J. W. Han: Efficient and Effective Clustering Method for Spatial Data Mining. *VLDB* 1994
17. S. Papadimitriou, H. Kitagawa et al. LOCI: Fast Outlier Detection Using the Local Correlation Integral. *ICDE* 2003
18. S. Papadimitriou, C. Faloutsos: Cross-Outlier Detection. *SSTD* 2003
19. N. Roussopoulos, S. Kelley and F. Vincent: Nearest neighbor queries. *SIGMOD* 1995
20. S. Ramaswamy, R. Rastogi, K. Shim: Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD* 2000
21. S. Shekhar, C. T. Lu, P. S. Zhang: Detecting Graph-based Spatial Outliers. *KDD* 2001
22. J. Tang, Z. X. Chen et al.: Enhancing Effectiveness of Outlier Detections for Low Density Patterns. *PAKDD* 2002
23. W. K. Wong, A. W. Moore et al.: Rule-Based Anomaly Pattern Detection for Detecting Disease Outbreaks. *AAAI* 2002
24. M. L. Yiu, N. Mamoulis: Clustering Objects on a Spatial Network. *SIGMOD* 2004
25. M. L. Yiu et al.: Aggregate Nearest Neighbor Queries in Road Networks. *IEEE Trans. Knowl. Data Eng.* 17(6), 2005
26. T. Zhang et al.: BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD* 1996