Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data, Dalles, TX, 2000

LOF: Identifying Density-Based Local Outliers

Markus M. Breunig†, Hans-Peter Kriegel†, Raymond T. Ng‡, Jörg Sander†

† Institute for Computer Science
University of Munich
Oettingenstr. 67, D-80538 Munich, Germany

{ breunig | kriegel | sander }
@dbs.informatik.uni-muenchen.de

Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada

rng@cs.ubc.ca

ABSTRACT

For many KDD applications, such as detecting criminal activities in E-commerce, finding the rare instances or the outliers, can be more interesting than finding the common patterns. Existing work in outlier detection regards being an outlier as a binary property. In this paper, we contend that for many scenarios, it is more meaningful to assign to each object a *degree* of being an outlier. This degree is called the *local outlier factor* (LOF) of an object. It is *local* in that the degree depends on how isolated the object is with respect to the surrounding neighborhood. We give a detailed formal analysis showing that LOF enjoys many desirable properties. Using real-world datasets, we demonstrate that LOF can be used to find outliers which appear to be meaningful, but can otherwise not be identified with existing approaches. Finally, a careful performance evaluation of our algorithm confirms we show that our approach of finding local outliers can be practical.

Keywords

Outlier Detection, Database Mining.

1. INTRODUCTION

Larger and larger amounts of data are collected and stored in databases, increasing the need for efficient and effective analysis methods to make use of the information contained implicitly in the data. *Knowledge discovery in databases* (KDD) has been defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable knowledge from the data [9].

Most studies in KDD focus on finding patterns applicable to a considerable portion of objects in a dataset. However, for applications such as detecting criminal activities of various kinds (e.g. in electronic commerce), rare events, deviations from the majority, or exceptional cases may be more interesting and useful than the common cases. Finding such exceptions and outliers, however, has not yet received as much attention in the KDD community as some other topics have, e.g. association rules.

Recently, a few studies have been conducted on outlier detection for large datasets (e.g. [18], [1], [13], [14]). While a more detailed discussion on these studies will be given in section 2, it suffices to point out here that most of these studies consider being an outlier as a binary property. That is, either an object in the dataset is an outlier or not. For many applications, the situation is more complex. And it becomes more meaningful to assign to each object a *degree* of being an outlier.

Also related to outlier detection is an extensive body of work on clustering algorithms. From the viewpoint of a clustering algorithm, outliers are objects not located in clusters of a dataset, usually called noise. The set of noise produced by a clustering algorithm, however, is highly dependent on the particular algorithm and on its clustering parameters. Only a few approaches are directly concerned with outlier detection. These algorithms, in general, consider outliers from a more global perspective, which also has some major drawbacks. These drawbacks are discussed in detail in section 2 and section 3. Furthermore, based on these clustering algorithms, the property of being an outlier is again binary.

In this paper, we introduce a new method for finding outliers in a multidimensional dataset. We introduce a local outlier (*LOF*) for each object in the dataset, indicating its degree of outlier-ness. This is, to the best of our knowledge, the first concept of an outlier which also quantifies how outlying an object is. The outlier factor is local in the sense that only a restricted neighborhood of each object is taken into account. Our approach is loosely related to density-based clustering. However, we do not require any explicit or implicit notion of clusters for our method. Specifically, our technical contributions in this paper are as follow:

- After introducing the concept of *LOF*, we analyze the formal properties of *LOF*. We show that for most objects in a cluster their *LOF* are approximately equal to 1. For any other object, we give a lower and upper bound on its *LOF*. These bounds highlight the local nature of *LOF*. Furthermore, we analyze when these bounds are tight. We identify classes of objects for which the bounds are tight. Finally, for those objects for which the bounds are not tight, we provide sharper bounds.
- The *LOF* of an object is based on the single parameter of *MinPts*, which is the number of nearest neighbors used in de-

fining the local neighborhood of the object. We study how this parameter affects the *LOF* value, and we present practical guidelines for choosing the *MinPts* values for finding local outliers.

- Last but not least, we present experimental results which show both the capability and the performance of finding local outliers. We conclude that finding local outliers using *LOF* is meaningful and efficient.

The paper is organized as follows. In section 2, we discuss related work on outlier detection and their drawbacks. In section 3 we discuss in detail the motivation of our notion of outliers, especially, the advantage of a local instead of a global view on outliers. In section 4 we introduce *LOF* and define other auxiliary notions. In section 5 we analyze thoroughly the formal properties of *LOF*. Since *LOF* requires the single parameter *MinPts*, in section 6 we analyze the impact of the parameter, and discuss ways to choose *MinPts* values for *LOF* computation. In section 7 we perform an extensive experimental evaluation.

2. RELATED WORK

Most of the previous studies on outlier detection were conducted in the field of statistics. These studies can be broadly classified into two categories. The first category is *distribution-based*, where a standard distribution (e.g. Normal, Poisson, etc.) is used to fit the data best. Outliers are defined based on the probability distribution. Over one hundred tests of this category, called discordancy tests, have been developed for different scenarios (see [5]). A key drawback of this category of tests is that most of the distributions used are univariate. There are some tests that are multivariate (e.g. multivariate normal outliers). But for many KDD applications, the underlying distribution is unknown. Fitting the data with standard distributions is costly, and may not produce satisfactory results.

The second category of outlier studies in statistics is *depth-based*. Each data object is represented as a point in a k-d space, and is assigned a depth. With respect to outlier detection, outliers are more likely to be data objects with smaller depths. There are many definitions of depth that have been proposed (e.g. [20], [16]). In theory, depth-based approaches could work for large values of k. However, in practice, while there exist efficient algorithms for $k = 2$ or 3 ([16], [18], [12]), depth-based approaches become inefficient for large datasets for $k \geq 4$. This is because depth-based approaches rely on the computation of k-d convex hulls which has a lower bound complexity of $\Omega(n^{k/2})$ for n objects.

Recently, Knorr and Ng proposed the notion of *distance-based* outliers [13], [14]. Their notion generalizes many notions from the distribution-based approaches, and enjoys better computational complexity than the depth-based approaches for larger values of k. Later in section 3, we will discuss in detail how their notion is different from the notion of local outliers proposed in this paper. In [17] the notion of distance based outliers is extended by using the distance to the k-nearest neighbor to rank the outliers. A very efficient algorithms to compute the top n outliers in this ranking is given, but their notion of an outlier is still distance-based.

Given the importance of the area, fraud detection has received more attention than the general area of outlier detection. Depending on the specifics of the application domains, elaborate fraud models and fraud detection algorithms have been developed (e.g. [8], [6]). In contrast to fraud detection, the kinds of outlier detection work discussed so far are more exploratory in nature. Outlier detection may indeed lead to the construction of fraud models.

Finally, most clustering algorithms, especially those developed in the context of KDD (e.g. CLARANS [15], DBSCAN [7], BIRCH [23], STING [22], WaveCluster [19], DenClue [11], CLIQUE [3]), are to some extent capable of handling exceptions. However, since the main objective of a clustering algorithm is to find clusters, they are developed to optimize clustering, and not to optimize outlier detection. The exceptions (called "noise" in the context of clustering) are typically just tolerated or ignored when producing the clustering result. Even if the outliers are not ignored, the notions of outliers are essentially binary, and there are no quantification as to how outlying an object is. Our notion of local outliers share a few fundamental concepts with density-based clustering approaches. However, our outlier detection method does not require any explicit or implicit notion of clusters.

3. PROBLEMS OF EXISTING (NON-LOCAL) APPROACHES

As we have seen in section 2, most of the existing work in outlier detection lies in the field of statistics. Intuitively, outliers can be defined as given by Hawkins [10].

Definition 1: (Hawkins-Outlier)

An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism.

This notion is formalized by Knorr and Ng [13] in the following definition of outliers. Throughout this paper, we use o, p, q to denote objects in a dataset. We use the notation $d(p, q)$ to denote the distance between objects p and q. For a set of objects, we use C (sometimes with the intuition that C forms a cluster). To simplify our notation, we use $d(p, C)$ to denote the minimum distance between p and object q in C, i.e. $d(p,C) = min\{\ d(p,q)\ \mid q \in C\ \}$.

Definition 2: (DB(*pct*, *dmin*)-Outlier)

An object p in a dataset D is a *DB*(*pct*, *dmin*)-outlier if at least percentage *pct* of the objects in D lies greater than distance *dmin* from p, i.e., the cardinality of the set $\{q \in D \mid d(p, q) \leq dmin\}$ is less than or equal to $(100 - pct)\%$ of the size of D.

The above definition captures only certain kinds of outliers. Because the definition takes a global view of the dataset, these outliers can be viewed as "global" outliers. However, for many interesting real-world datasets which exhibit a more complex structure, there is another kind of outliers. These can be objects that are outlying

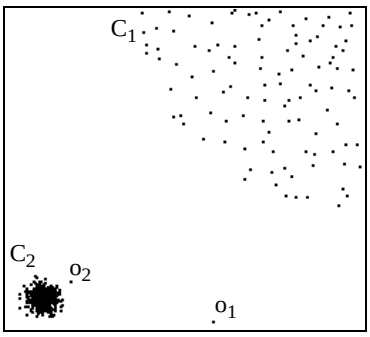


Figure 1: 2-*d* dataset DS1

relative to their local neighborhoods, particularly with respect to the densities of the neighborhoods. These outliers are regarded as "local" outliers.

To illustrate, consider the example given in Figure 1. This is a simple 2-dimensional dataset containing 502 objects. There are 400 objects in the first cluster C_1, 100 objects in the cluster C_2, and two additional objects o_1 and o_2. In this example, C_2 forms a denser cluster than C_1. According to Hawkins' definition, both o1 and o2 can be called outliers, whereas objects in C_1 and C_2 should not be. With our notion of a "local" outlier, we wish to label both o_1 and o_2 as outliers. In contrast, within the framework of distance-based outliers, only o_1 is a reasonable DB(*pct*,*dmin*)-outlier in the following sense. If for every object q in C_1, the distance between q and its nearest neighbor is greater than the distance between o_2 and C_2 (i.e., $d(o_2,C_2)$), we can in fact show that there is no appropriate value of *pct* and *dmin* such that o_2 is a DB(*pct*,*dmin*)-outlier but the the objects in C_1 are not.

The reason is as follows. If the *dmin* value is less than the distance $d(o_2,C_2)$, then all 501 objects (*pct* = 100*501/502) are further away from o_2 than *dmin*. But the same condition holds also for every object q in C_1. Thus, in this case, o_2 and all objects in C_1 are DB(*pct*, *dmin*)-outliers.

Otherwise, if the *dmin* value is greater than the distance $d(o_2, C_2)$, then it is easy to see that: o_2 is a DB(*pct*,*dmin*)-outlier implies that there are many objects q in C_1 such that q is also a DB(*pct*,*dmin*)-outlier. This is because the cardinality of the set $\{p \in D \mid d(p,o_2) \le dmin\}$ is always bigger than the cardinality of the set $\{p \in D \mid d(p,q) \le dmin\}$. Thus, in this case, if o_2 is a DB(*pct*,*dmin*)-outlier, so are many objects q in C_1. Worse still, there are values of *pct* and *dmin* such that while o_2 is not an outlier, some q in C_1 are.

4. FORMAL DEFINITION OF LOCAL OUTLIERS

The above example shows that the global view taken by DB(*pct*, *dmin*)-outliers is meaningful and adequate under certain conditions, but not satisfactory for the general case when clusters of different densities exist. In this section, we develop a formal definition of local outliers, which avoids the shortcomings presented in the previous section. The key difference between our notion and existing notions of outliers is that being outlying is *not a binary property*. Instead, we assign to each object an *outlier factor*, which is the degree the object is being outlying.

We begin with the notions of the *k-distance* of object *p*, and, correspondingly, the *k-distance neighborhood* of *p*.

Definition 3: (*k*-distance of an object *p*)

For any positive integer *k*, the *k*-distance of object *p*, denoted as *k-distance*(*p*), is defined as the distance $d(p,o)$ between p and an object $o \in D$ such that:

(i) for at least k objects $o' \in D \setminus \{p\}$ it holds that $d(p,o') \le d(p,o)$, and

(ii) for at most k-1 objects $o' \in D \setminus \{p\}$ it holds that $d(p,o') < d(p,o)$.

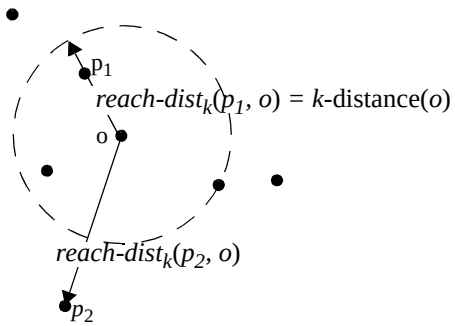


Figure 2: *reach-dist*(p_1,*o*) and *reach-dist*(p_2,*o*), for *k*=4

Definition 4: (*k*-distance neighborhood of an object *p*)

Given the *k*-distance of *p*, the *k-distance neighborhood of p* contains every object whose distance from *p* is not greater than the *k*-distance, i.e. $N_{k\text{-}distance(p)}(p) = \{ q \in D\setminus\{p\} \mid d(p, q) \le k\text{-distance}(p) \}$.

These objects q are called the *k*-nearest neighbors of *p*.

Whenever no confusion arises, we simplify our notation to use $N_k(p)$ as a shorthand for $N_{k\text{-}distance(p)}(p)$. Note that in definition 3, the *k*-distance(*p*) is well defined for any positive integer *k*, although the object *o* may not be unique. In this case, the cardinality of $N_k(p)$ is greater than *k*. For example, suppose that there are: (i) 1 object with distance 1 unit from *p*; (ii) 2 objects with distance 2 units from *p*; and (iii) 3 objects with distance 3 units from *p*. Then 2-distance(*p*) is identical to 3-distance(*p*). And there are 3 objects of 4-distance(*p*) from *p*. Thus, the cardinality of $N_4(p)$ can be greater than 4, in this case 6.

Definition 5: (reachability distance of an object *p* w.r.t. object *o*)

Let *k* be a natural number. The *reachability distance* of object *p* with respect to object *o* is defined as

$$reach\text{-}dist_k(p, o) = \max \{ k\text{-distance}(o), d(p, o) \}.$$

Figure 2 illustrates the idea of reachability distance with $k = 4$. Intuitively, if object *p* is far away from *o* (e.g. p_2 in the figure), then the reachability distance between the two is simply their actual distance. However, if they are "sufficiently" close (e.g., p_1 in the figure), the actual distance is replaced by the *k*-distance of *o*. The reason is that in so doing, the statistical fluctuations of $d(p,o)$ for all the *p*'s close to *o* can be significantly reduced. The strength of this smoothing effect can be controlled by the parameter *k*. The higher the value of *k*, the more similar the reachability distances for objects within the same neighborhood.

So far, we have defined *k*-distance(*p*) and $reach\text{-}dist_k(p)$ for any positive integer *k*. But for the purpose of defining outliers, we focus on a specific instantiation of *k* which links us back to density-based clustering. In a typical density-based clustering algorithm, such as [7], [3], [22], or [11], there are two parameters that define the notion of density: (i) a parameter *MinPts* specifying a minimum number of objects; (ii) a parameter specifying a volume. These two parameters determine a density *threshold* for the clustering algorithms to operate. That is, objects or regions are connected if their neighborhood densities exceed the given density threshold. To detect density-

based outliers, however, it is necessary to compare the densities of different sets of objects, which means that we have to determine the density of sets of objects dynamically. Therefore, we keep *MinPts* as the only parameter and use the values reach-dist$_{MinPts}(p, o)$, for $o \in N_{MinPts}(p)$, as a measure of the volume to determine the density in the neighborhood of an object *p*.

Definition 6: (local reachability density of an object *p*)

The *local reachability density* of *p* is defined as

$$lrd_{MinPts}(p) = 1 \Big/ \left(\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p, o)}{|N_{MinPts}(p)|} \right)$$

Intuitively, the local reachability density of an object *p* is the inverse of the average reachability distance based on the *MinPts*-nearest neighbors of *p*. Note that the local density can be ∞ if all the reachability distances in the summation are 0. This may occur for an object *p* if there are at least *MinPts* objects, different from *p*, but sharing the same spatial coordinates, i.e. if there are at least *MinPts* duplicates of *p* in the dataset. For simplicity, we will not handle this case explicitly but simply assume that there are no duplicates. (To deal with duplicates, we can base our notion of neighborhood on a *k*-distinct-distance, defined analogously to *k*-distance in definition 3, with the additional requirement that there be at least *k* objects with different spatial coordinates.)

Definition 7: ((local) outlier factor of an object *p*)

The (*local) outlier factor* of *p* is defined as

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

The outlier factor of object *p* captures the degree to which we call *p* an outlier. It is the average of the ratio of the local reachability density of *p* and those of *p*'s *MinPts*-nearest neighbors. It is easy to see that the lower *p*'s local reachability density is, and the higher the local reachability densities of *p*'s *MinPts*-nearest neighbors are, the higher is the *LOF* value of *p*. In the following section, the formal properties of *LOF* are made precise. To simplify notation, we drop the subscript *MinPts* from *reach-dist*, *lrd* and *LOF*, if no confusion arises.

5. PROPERTIES OF LOCAL OUTLIERS

In this section, we conduct a detailed analysis on the properties of *LOF*. The goal is to show that our definition of *LOF* captures the spirit of local outliers, and enjoys many desirable properties. Specifically, we show that for most objects *p* in a cluster, the *LOF* of *p* is approximately equal to 1. As for other objects, including those outside of a cluster, we give a general theorem giving a lower and upper bound on the *LOF*. Furthermore, we analyze the tightness of our bounds. We show that the bounds are tight for important classes of objects. However, for other classes of objects, the bounds may not be as tight. For the latter, we give another theorem specifying better bounds.

5.1 *LOF* for Objects Deep in a Cluster

In section 3, we motivate the notion of a local outlier using figure 1. In particular, we hope to label o_2 as outlying, but label all objects in the cluster C_1 as non-outlying. Below, we show that for most objects in C_1 its *LOF* is approximately 1, indicating that they cannot be labeled as outlying.

Lemma 1: Let *C* be a collection of objects. Let *reach-dist-min* denote the minimum reachability distance of objects in *C*, i.e., *reach-dist-min* = *min* {*reach-dist*(*p*, *q*) | *p*, *q* ∈ *C*}. Similarly, let *reach-dist-max* denote the maximum reachability distance of objects in *C*. Let ε be defined as (*reach-dist-max*/*reach-dist-min* – 1).
Then for all objects *p* ∈ *C*, such that:

(i) all the *MinPts*-nearest neighbors *q* of *p* are in *C*, and

(ii) all the *MinPts*-nearest neighbors *o* of *q* are also in *C*,

it holds that $1/(1 + \varepsilon) \le LOF(p) \le (1 + \varepsilon)$.

Proof (Sketch): For all *MinPts*-nearest neighbors *q* of *p*, *reach-dist*(*p*, *q*) ≥ *reach-dist-min*. Then the local reachability density of *p*, as per definition 6, is ≤ 1/*reach-dist-min*. On the other hand, *reach-dist*(*p*, *q*) ≤ *reach-dist-max*. Thus, the local reachability density of *p* is ≥ 1/*reach-dist-max*.
Let *q* be a *MinPts*-nearest neighbor of *p*. By an argument identical to the one for *p* above, the local reachability density of *q* is also between 1/*reach-dist-max* and 1/*reach-dist-min*.
Thus, by definition 7, we have *reach-dist-min*/*reach-dist-max* ≤ *LOF*(*p*) ≤ *reach-dist-max*/*reach-dist-min*. Hence, we establish $1/(1 + \varepsilon) \le LOF(p) \le (1 + \varepsilon)$. ■

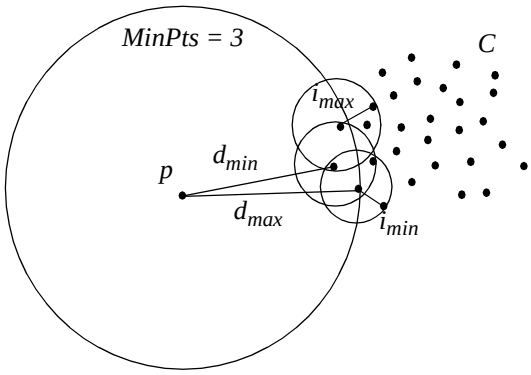
The interpretation of lemma 1 is as follows. Intuitively, *C* corresponds to a "cluster". Let us consider the objects *p* that are "deep" inside the cluster, which means that all the *MinPts*-nearest neighbors *q* of *p* are in *C*, and that, in turn, all the *MinPts*-nearest neighbors of *q* are also in *C*. For such deep objects *p*, the *LOF* of *p* is bounded. If *C* is a "tight" cluster, the ε value in lemma 1 can be quite small, thus forcing the *LOF* of *p* to be quite close to 1.

To return to the example in figure 1, we can apply lemma 1 to conclude that the *LOF*s of most objects in cluster C_1 are close to 1.

5.2 A General Upper and Lower Bound on *LOF*

Lemma 1 above shows a basic property of *LOF*, namely that for objects deep inside a cluster, their *LOF*s are close to 1, and should not be labeled as a local outlier. A few immediate questions come to mind. What about those objects that are near the periphery of the cluster? And what about those objects that are outside the cluster, such as o_2 in figure 1? Can we get an upper and lower bound on the *LOF* of these objects?

Theorem 1 below shows a general upper and lower bound on *LOF*(*p*) for *any* object *p*. As such, theorem 1 generalizes lemma 1 along two dimensions. First, theorem 1 applies to any object *p*, and is not restricted to objects deep inside a cluster. Second, even for objects deep inside a cluster, the bound given by theorem 1 can be tighter than the bound given by lemma 1, implying that the epsilon defined in lemma 1 can be made closer to zero.. This is because in lemma 1, the values of *reach-dist-min* and *reach-dist-max* are obtained based on a larger set of reachability distances. In contrast, in theorem 1, this minimum and maximum are based on just the *MinPts*-nearest neighborhoods of the objects under consideration,



$d_{min} = 4*i_{max}$ $\Rightarrow LOF_{MinPts}(p) \geq 4$

$d_{max} = 6*i_{min}$ $\Rightarrow LOF_{MinPts}(p) \leq 6$

Figure 3: Illustration of theorem 1

giving rise to tighter bounds. In section 5.3, we will analyze in greater details the tightness of the bounds given in theorem 1.

Before we present theorem 1, we define the following terms. For any object p, let $direct_{min}(p)$ denote the minimum reachability distance between p and a *MinPts*-nearest neighbor of p, i.e.,

$$direct_{min}(p) = \min \{ reach\text{-}dist(p, q) \mid q \in N_{MinPts}(p) \}.$$

Similarly, let $direct_max(p)$ denote the corresponding maximum, i.e. $direct_{max}(p) = \max \{ reach\text{-}dist(p, q) \mid q \in N_{MinPts}(p) \}$.

Furthermore, to generalize these definitions to the *MinPts*-nearest neighbor q of p, let $indirect_{min}(p)$ denote the minimum reachability distance between q and a *MinPts*-nearest neighbor of q, i.e.,

$$indirect_{min}(p) = \min \{ reach\text{-}dist(q, o) \mid q \in N_{MinPts}(p) \text{ and } o \in N_{MinPts}(q)\}.$$

Similarly, let $indirect_{max}(p)$ denote the corresponding maximum. In the sequel, we refer to p's *MinPts*-nearest neighborhood as p's *direct* neighborhood, and refer to q's *MinPts*-nearest neighbors as p's *indirect* neighbors, whenever q is a *MinPts*-nearest neighbor of p.

Figure 3 gives a simple example to illustrate these definitions. In this example, object p lies some distance away from a cluster of objects C. For ease of understanding, let *MinPts* = 3. The $direct_{min}(p)$ value is marked as d_{min} in the figure; the $direct_{max}(p)$ value is marked as d_{max}. Because p is relatively far away from C, the 3-distance of every object q in C is much smaller than the actual distance between p and q. Thus, from definition 5, the reachability distance of p w.r.t. q is given by the actual distance between p and q. Now among the 3-nearest neighbors of p, we in turn find their minimum and maximum reachability distances to their 3-nearest neighbors. In the figure, the $indirect_{min}(p)$ and $indirect_{max}(p)$ values are marked as i_{min} and i_{max} respectively.

Theorem 1: Let p be an object from the database D, and $1 \leq MinPts \leq |D|$.
Then, it is the case that

$$\frac{direct_{min}(p)}{indirect_{max}(p)} \leq LOF(p) \leq \frac{direct_{max}(p)}{indirect_{min}(p)}$$

Proof (Sketch): (a) $\frac{direct_{min}(p)}{indirect_{max}(p)} \leq LOF(p)$:

$\forall o \in N_{MinPts}(p)$: $reach\text{-}dist(p, o) \geq direct_{min}(p)$,

by definition of $direct_{min}(p)$.

$$\Rightarrow 1/\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist(p, o)}{|N_{MinPts}(p)|} \leq \frac{1}{direct_{min}(p)}, \text{ i.e.}$$

$$lrd(p) \leq \frac{1}{direct_{min}(p)}$$

$\forall q \in N_{MinPts}(o)$: $reach\text{-}dist(o, q) \leq indirect_{max}(p)$,

by definition of $indirect_{max}(p)$.

$$\Rightarrow 1/\frac{\sum_{q \in N_{MinPts}(o)} reach\text{-}dist(o, q)}{|N_{MinPts}(o)|} \geq \frac{1}{indirect_{max}(p)} \quad \text{,i.e.}$$

$$lrd(o) \geq \frac{1}{indirect_{max}(p)}$$

Thus, it follows that

$$LOF(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd(o)}{lrd(p)}}{|N_{MinPts}(p)|} \geq$$

$$\frac{\sum_{o \in N_{MinPts}(p)} \frac{\left(\frac{1}{indirect_{max}(p)}\right)}{\left(\frac{1}{direct_{min}(p)}\right)}}{|N_{MinPts}(p)|} = \frac{direct_{min}(p)}{indirect_{max}(p)}$$

(b) $LOF(p) \leq \frac{direct_{max}(p)}{indirect_{min}(p)}$: analogously. ■

To illustrate the theorem using the example in figure 3, suppose that d_{min} is 4 times that of i_{max}, and d_{max} is 6 times that of i_{min}. Then by theorem 1, the *LOF* of p is between 4 and 6. It should also be clear from theorem 1 that $LOF(p)$ has an easy-to-understand interpretation. It is simply a function of the reachability distances in p's direct neighborhood relative to those in p's indirect neighborhood.

5.3 The Tightness of the Bounds

As discussed before, theorem 1 is a general result with the specified upper and lower bounds for *LOF* applicable to any object p. An immediate question comes to mind. How good or tight are these bounds? In other words, if we use LOF_{max} to denote the upper bound $direct_{max}/indirect_{min}$, and use LOF_{min} to denote the lower bound $direct_{min}/indirect_{max}$, how large is the spread or difference between LOF_{max} and LOF_{min}? In the following we study this issue. A key part of the following analysis is to show that the spread LOF_{max}-LOF_{min} is dependent on the ratio of *direct/indirect*. It turns out that the spread is small under some conditions, but not so small under other conditions.

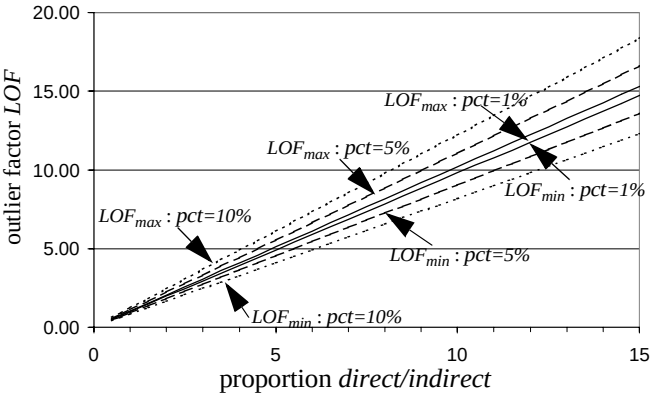


Figure 4: Upper and lower bound on *LOF* depending on *direct/indirect* for different values of *pct*

Given $direct_{min}(p)$ and $direct_{max}(p)$ as defined above, we use *direct*(*p*) to denote the mean value of $direct_{min}(p)$ and $direct_{max}(p)$. Similarly, we use *indirect*(*p*) to denote the mean value of $indirect_{min}(p)$ and $indirect_{max}(p)$. In the sequel, whenever no confusion arises, we drop the parameter *p*, e.g., *direct* as a shorthand of *direct*(*p*).

Now to make our following analysis easier to understand, we simplify our discussion by requiring that $(direct_{max} - direct_{min})/direct = (indirect_{max} - indirect_{min})/indirect$. That is, we assume that the reachability distances in the direct and indirect neighborhoods fluctuate by the same amount. Because of this simplification, we can use a single parameter *pct* in the sequel to control the fluctuation. More specifically, in figure 4, *pct* = *x*% corresponds to the situation where $direct_{max} = direct*(1+x\%)$, $direct_{min} = direct*(1-x\%)$, $indirect_{max} = indirect*(1+x\%)$ and $indirect_{min} = indirect*(1-x\%)$. Figure 4 shows the situations when *pct* is set to 1%, 5% and 10%. The spread between LOF_{max} and LOF_{min} increases as *pct* increases.

More importantly, figure 4 shows that, for a fixed percentage *pct*=*x*%, the spread between LOF_{max} and LOF_{min} grows linearly with respect to the ratio *direct*/*indirect*. This means that the relative span $(LOF_{max} - LOF_{min})/(direct/indirect)$ is constant. Stated differently, the relative fluctuation of the *LOF* depends only on the *ratios* of the underlying reachability distances and not on their *absolute* values. This highlights the spirit of local outliers.

To be more precise, in fact, the whole situation is best captured in the 3-dimensional space where the three dimensions are: $(LOF_{max} - LOF_{min})$, (*direct*/*indirect*), and *pct*. Figure 4 then represents a series of 2-D projections on the first two dimensions. But figure 4 does not show the strength of the dependency between the relative fluctuation of the *LOF* and the relative fluctuation of *pct*. For this purpose, figure 5 is useful. The *y*-axis of the figure shows the ratio between the two dimensions $(LOF_{max} - LOF_{min})$ and (*direct*/*indirect*) in the 3-dimensional space mentioned above, and the *x*-axis corresponds to the other dimension *pct*. To understand the shape of the curve in figure 5, we have to take a closer look at the ratio $(LOF_{max} - LOF_{min})/(direct/indirect)$:

$$\frac{LOF_{max} - LOF_{min}}{\frac{direct}{indirect}} = \frac{indirect}{direct} \cdot \left(\frac{direct + \frac{direct \cdot pct}{100}}{indirect - \frac{indirect \cdot pct}{100}} - \frac{direct - \frac{direct \cdot pct}{100}}{indirect + \frac{indirect \cdot pct}{100}} \right) =$$

$$= \left(\frac{1 + \frac{pct}{100}}{1 - \frac{pct}{100}} - \frac{1 - \frac{pct}{100}}{1 + \frac{pct}{100}} \right) = \frac{4 \times \frac{pct}{100}}{1 - \left(\frac{pct}{100}\right)^2}$$

Figure 5 shows that $(LOF_{max} - LOF_{min})/(direct/indirect)$ is only dependent on the percentage value *pct*. Its value approaches infinity if *pct* approaches 100, but it is very small for reasonable values of *pct*. This also verifies that the relative fluctuation of the *LOF* is constant for a fixed percentage *pct*, as we have seen in figure 4.

To summarize, if the fluctuation of the average reachability distances in the direct and indirect neighborhoods is small (i.e., *pct* is low), theorem 1 estimates the *LOF* very well, as the minimum and maximum *LOF* bounds are close to each other. There are two important cases for which this is true.

- The percentage *pct* is very low for an object *p*, if the fluctuation of the reachability distances is rather homogeneous, i.e., if the *MinPts*-nearest neighbors of *p* belong to the same cluster. In this case, the values $direct_{min}$, $direct_{max}$, $indirect_{min}$ and $indirect_{max}$ are almost identical, resulting in the *LOF* being close to 1. This is consistent with the result established in lemma 1.
- The argument above can be generalized to an object *p* which is not located deep inside a cluster, but whose *MinPts*-nearest neighbors all belong to the same cluster (as depicted in figure 3). In this case, even though *LOF* may not be close to 1, the bounds on *LOF* as predicted by theorem 1 are tight.

5.4 Bounds for Objects whose Direct Neighborhoods Overlap Multiple Clusters

So far we have analyzed the tightness of the bounds given in theorem 1, and have given two conditions under which the bounds are tight. An immediate question that comes to mind is: under what

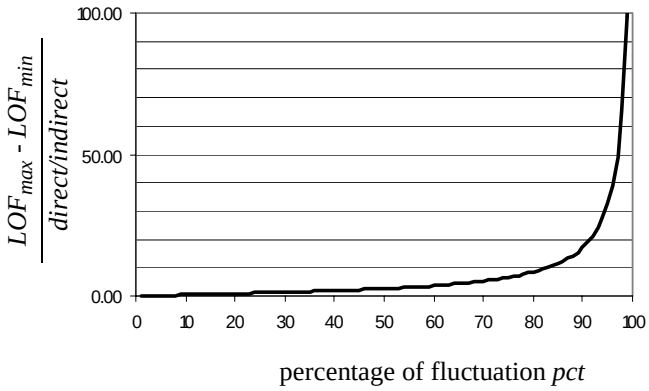


Figure 5: Relative span for *LOF* depending on percentage of fluctuation for *d* and *w*

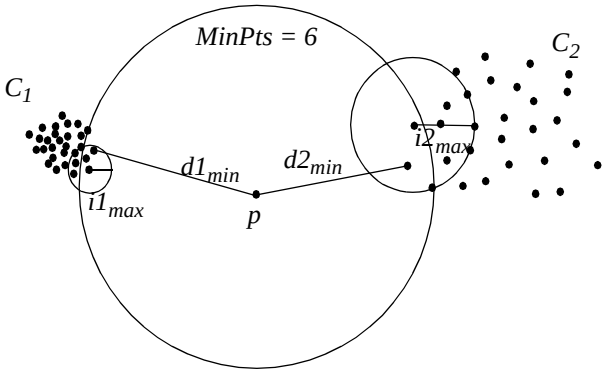


Figure 6: Illustration of theorem 2

condition are the bounds not tight? Based on figure 5, if the *MinPts*-nearest neighbors of an object *p* belong to different clusters having different densities, the value for *pct* may be very large. Then based on figure 5, the spread between LOF_{max} and LOF_{min} value can be large. In this case, the bounds given in theorem 1 do not work well.

As an example, let us consider the situation shown in figure 1 again. For object o_2, because all its *MinPts*-nearest neighbors come from the same cluster C_2, the bounds given by theorem 1 on the *LOF* of o_2 is expected to be tight. In contrast, the *MinPts*-nearest neighbors of o_1 come from both clusters C_1 and C_2. In this case, the given bounds on the *LOF* of o_1 may not be as good.

Theorem 2 below intends to give better bounds on the LOF of object *p* when *p*'s *MinPts*-nearest neighborhood overlaps with more than one cluster. The intuitive meaning of theorem 2 is that, when we partition the *MinPts*-nearest neighbors of *p* into several groups, each group contributes proportionally to the *LOF* of *p*.

An example is shown in figure 6 for *MinPts*=6. In this case, 3 of object *p*'s 6-nearest neighbors come from cluster C_1, and the other 3 come from cluster C_2. Then according to theorem 2, LOF_{min} is given by $(0.5*d1_{min} + 0.5*d2_{min})/(0.5/i1_{max} + 0.5/i2_{max})$, where $d1_{min}$ and $d2_{min}$ give the minimum reachability distances between *p* and the 6-nearest neighbors of *p* in C_1 and C_2 respectively, and $i1_{max}$ and $i2_{max}$ give the maximum reachability distances between *q* and *q*'s 6-nearest neighbors, where *q* is a 6-nearest neighbor of *p* from C_1 and C_2 respectively. For simplicity, figure 6 does not show the case for the upper bound LOF_{max}.

Theorem 2: Let *p* be an object from the database *D*, $1 \le MinPts \le |D|$, and $C_1, C_2, ..., C_n$ be a partition of $N_{MinPts}(p)$, i.e. $N_{MinPts}(p) = C_1 \cup C_2 \cup \ ... \cup C_n \cup \{p\}$ with $C_i \cap C_j = \emptyset$, $C_i \neq \emptyset$ for $1 \le i,j \le n$, $i \neq j$.

Furthermore, let $\xi_i = |C_i| / |N_{MinPts}(p)|$ be the percentage of objects in *p*'s neighborhood, which are also in C_i. Let the notions $direct^i_{min}(p)$, $direct^i_{max}(p)$, $indirect^i_{min}(p)$, and $indirect^i_{max}(p)$ be defined analogously to $direct_{min}(p)$, $direct_{max}(p)$, $indirect_{min}(p)$, and $indirect_{max}(p)$ but restricted to the set C_i (e.g., $direct^i_{min}(p)$ denotes the minimum reachability distance between *p* and a *MinPts*-nearest neighbor of *p* in the set C_i).

Then, it holds that (a)

$$LOF(p) \ge \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{min}(p) \right) \cdot \left(\sum_{i=1}^{n} \frac{\xi_i}{indirect^i_{max}(p)} \right)$$

and (b)

$$LOF(p) \le \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{max}(p) \right) \cdot \left(\sum_{i=1}^{n} \frac{\xi_i}{indirect^i_{min}(p)} \right)$$

■

We give a proof sketch of theorem 2 in the appendix. Theorem 2 generalizes theorem 1 in taking into consideration the ratios of the *MinPts*-nearest neighbors coming from multiple clusters. As such, there is the following corollary.

Corollary 1: If the number of partitions in theorem 2 is 1, then LOF_{min} and LOF_{max} given in theorem 2 are exactly the same corresponding bounds given in theorem 1.■

6. THE IMPACT OF THE PARAMETER *MINPTS*

In the previous section, we have analyzed the formal properties of *LOF*. For objects deep inside a cluster, we have shown that the *LOF* is approximately equal to 1. For other objects, we have established two sets of upper and lower bounds on the *LOF*, depending on whether the *MinPts*-nearest neighbors come from one or more clusters. It is important to note that all the previous results are based on a given *MinPts* value. In this section, we discuss how the *LOF* value is influenced by the choice of the *MinPts* value, and how to determine the right *MinPts* values for the *LOF* computation.

6.1 How *LOF* Varies according to Changing *MinPts* Values

Given the analytic results established in the previous section, several interesting questions come to mind. How does the *LOF* value change when the *MinPts* value is adjusted? Given an increasing sequence of *MinPts* values, is there a corresponding *monotonic* sequence of changes to *LOF*? That is, does *LOF* decrease or increase monotonically?

Unfortunately, the reality is that *LOF* neither decreases nor increases monotonically. Figure 7 shows a simple scenario where all the objects are distributed following a Gaussian distribution. For each *MinPts* value between 2 and 50, the minimum, maximum and mean *LOF* values, as well as the standard deviation, are shown.

Let us consider the maximum *LOF* as an example. Initially, when the *MinPts* value is set to 2, this reduces to using the actual inter-object distance $d(p,o)$ in definition 5. By increasing the *MinPts* value, the statistical fluctuations in reachability distances and in *LOF* are weakened. Thus, there is an initial drop on the maximum *LOF* value. However, as the *MinPts* value continues to increase, the maximum *LOF* value goes up and down, and eventually stabilizes to some value.

If the *LOF* value changes non-monotonically even for such a pure distribution like the Gaussian distribution, the *LOF* value changes more wildly for more complex situations. Figure 8 shows a two-dimensional dataset containing three clusters, where S_1 consists of 10

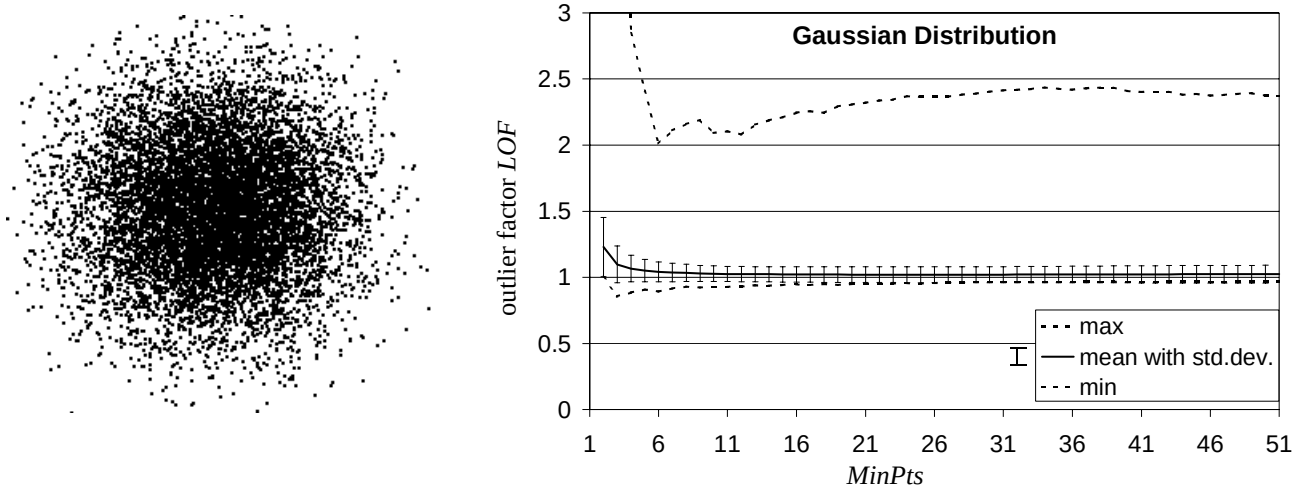


Figure 7: Fluctuation of the outlier-factors within a Gaussian cluster

objects, S_2 of 35 objects and S_3 of 500 objects. On the right side are representative plots for one object from each of these clusters. The plots show the *LOF* over *MinPts* for the range from 10 to 50. While the *LOF* of an object in S_3 is very stable around 1, the *LOF*s of the objects in S_1 and S_3 change more wildly.

6.2 Determining a Range of *MinPts* Values

Because the *LOF* value can go up and down, we propose as a heuristic that we use a range of *MinPts* values. In the following, we provide guidelines as to how this range can be picked. We use *MinPtsLB* and *MinPtsUB* to denote the "lower bound" and the "upper bound" of the range.

Let us first determine a reasonable value of *MinPtsLB*. Clearly, *MinPtsLB* can be as small as 2. However, as explained above and before definition 5, it is wise to remove unwanted statistical fluctuations due to *MinPts* being too small. As an example, for the Gaussian distribution shown in figure 7, the standard deviation of *LOF* only stabilizes when *MinPtsLB* is at least 10. As another extreme example, suppose we turn the Gaussian distribution in figure 7 to a uniform distribution. It turns out that for *MinPts* less than 10, there can be objects whose *LOF* are significant greater than 1. This is counter-intuitive because in a uniform distribution, no object should be labeled as outlying. Thus, the first guideline we provide for picking *MinPtsLB* is that it should be at least 10 to remove unwanted statistical fluctuations.

The second guideline we provide for picking *MinPtsLB* is based on a more subtle observation. Consider a simple situation of one object *p* and a set/cluster *C* of objects. If *C* contains fewer than *MinPtsLB* objects, then the set of *MinPts*-nearest neighbors of each object in *C* will include *p*, and vice versa. Thus, by applying theorem 1, the *LOF* of *p* and all the objects in *C* will be quite similar, thus making *p* indistinguishable from the objects in *C*.

If, on the other hand, *C* contains more than *MinPtsLB* objects, the *MinPts*-nearest neighborhoods of the objects deep in *C* will not contain *p*, but some objects of *C* will be included in *p*'s neighborhood. Thus, depending on the distance between *p* and *C* and the density of *C*, the *LOF* of *p* can be quite different from that of an object in *C*. The key observation here is that *MinPtsLB* can be regarded as the *minimum* number of objects a "cluster" (like *C* above) has to contain, so that other objects (like *p* above) can be local outliers relative to this cluster. This value could be application-dependent. For most of the datasets we experimented with, picking 10 to 20 appears to work well in general.

Next, we turn to the selection of a reasonable value of *MinPtsUB*, the upper bound value of the range of *MinPts* values. Like the lower bound *MinPtsLB*, the upper bound also has an associated meaning. Let *C* be a set/cluster of "close by" objects. Then *MinPtsUB* can be regarded as the *maximum* cardinality of *C* for all objects in *C* to potentially be local outliers. By "close by" we mean, that the $direct_{min}$, $direct_{max}$, $indirect_{min}$ and $indirect_{max}$ values are all very similar. In this case, for *MinPts* values exceeding *MinPtsUB*, theorem 1 requires that the *LOF* of all objects in *C* be close to 1. Hence, the guideline we provide for picking *MinPtsUB* is the maximum number of "close by" objects that can potentially be local outliers.

As an example, let us consider the situation shown in figure 8 again. Recall that S_1 consists of 10 objects, S_2 of 35 objects and S_3 of 500 objects. From the plots, it is clear that the objects in S_3 are never outliers, always having their *LOF* values close to 1. In contrast, the objects in S_1 are strong outliers for *MinPts* values between 10 and 35. The objects in S_2 are outliers starting at *MinPts* = 45. The reason for the last two effects is that, beginning at *MinPts* = 36, the *MinPts*-nearest neighborhoods of the objects in S_2 start to include some object(s) from S_1. From there on, the objects in S_1 and S_2 exhibit roughly the same behavior. Now at *MinPts* = 45, the members of this "combined" set of objects S_1 and S_2 start to include object(s) from S_3 in their neighborhoods, and thus starting to become outliers relative to S_3. Depending on the application domain, we may want to consider a group of 35 objects (like S_2) a cluster or a bunch of "close by" local outliers. To facilitate this, we can choose a *MinPtsUB* value accordingly, that is either smaller than 35 or larger than 35. A similar argument can be made for *MinPtsLB* with respect to the minimum number of objects relative to which other objects can be considered local outliers.

Having determined *MinPtsLB* and *MinPtsUB*, we can compute for each object its *LOF* values within this range. We propose the heuristic of ranking all objects with respect to the maximum *LOF* value within the specified range. That is, the ranking of an object *p* is based on: $\max\{LOF_{MinPts}(p) \mid MinPtsLB \le MinPts \le MinPtsUB\}$.

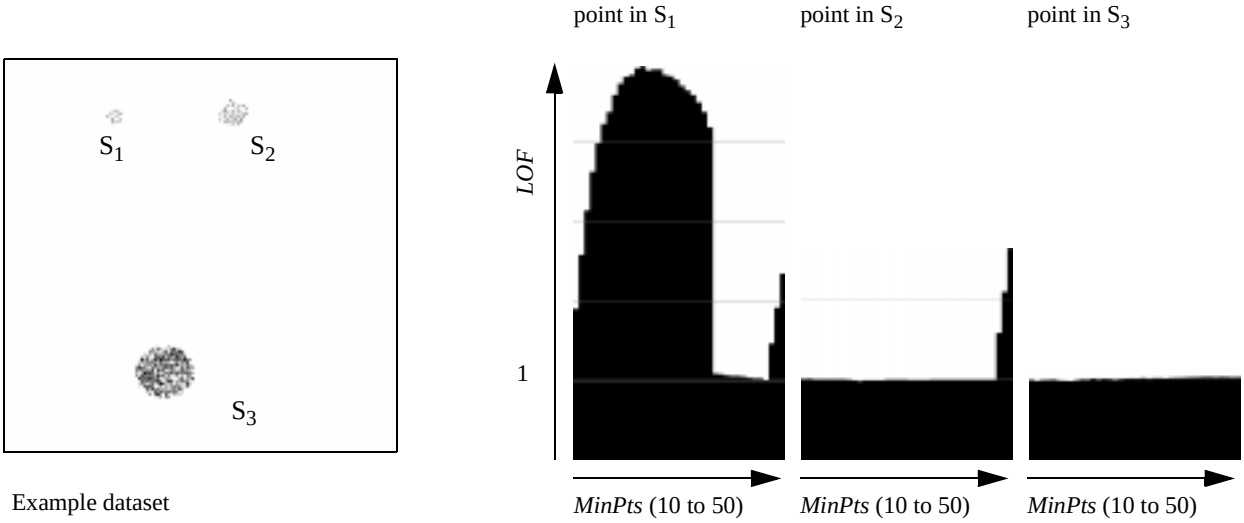


Figure 8: Ranges of *LOF* values for different objects in a sample dataset

Given all the *LOF* values within the range, instead of taking the maximum, we could take other aggregates, such as the minimum or the mean. The situation in figure 8 shows that taking the minimum could be inappropriate as the minimum may erase the outlying nature of an object completely. Taking the mean may also have the effect of diluting the outlying nature of the object. We propose to take the maximum to highlight the instance at which the object is the most outlying.

7. EXPERIMENTS

In this section, with the proposed heuristic of taking the maximum *LOF* value within the range, we show that our ideas can be used to successfully identify outliers which appear to be meaningful but cannot be identified by other methods. We start with a synthetical 2-dimensional dataset, for which we show the outlier factors for all objects, in order to give an intuitive notion of the *LOF* values computed. The second example uses the real-world dataset that has been used in [KN98] to evaluate the *DB*(*pct*, *dmin*) outliers. We repeat their experiments to validate our method. In the third example, we identify meaningful outliers in a database of german soccer players, for which we happen to have a "domain expert" handy, who confirmed the meaningfulness of the outliers found. The last subsection contains performance experiments showing the practicability of our approach even for large, high-dimensional datasets.

Additionally, we conducted experiments with a 64-dimensional dataset, to demonstrate that our definitions are reasonable in very high dimensional spaces. The feature vectors used are color histograms extracted from tv snapshots [2]. We indentified multiple clusters, e.g. a cluster of pictures from a tennis match, and reasonable local outliers with *LOF* values of up to 7.

7.1 A Synthetic Example

The left side of figure 9 shows a 2-dimensional dataset containing one low density Gaussian cluster of 200 objects and three large clusters of 500 objects each. Among these three, one is a dense Gaussian cluster and the other two are uniform clusters of different densities. Furthermore, it contains a couple of outliers. On the right side of figure 9 we plot the *LOF* of all the objects for *MinPts* = 40 as a third dimension. We see that the objects in the uniform clusters all have their *LOF* equal to 1. Most objects in the Gaussian clusters

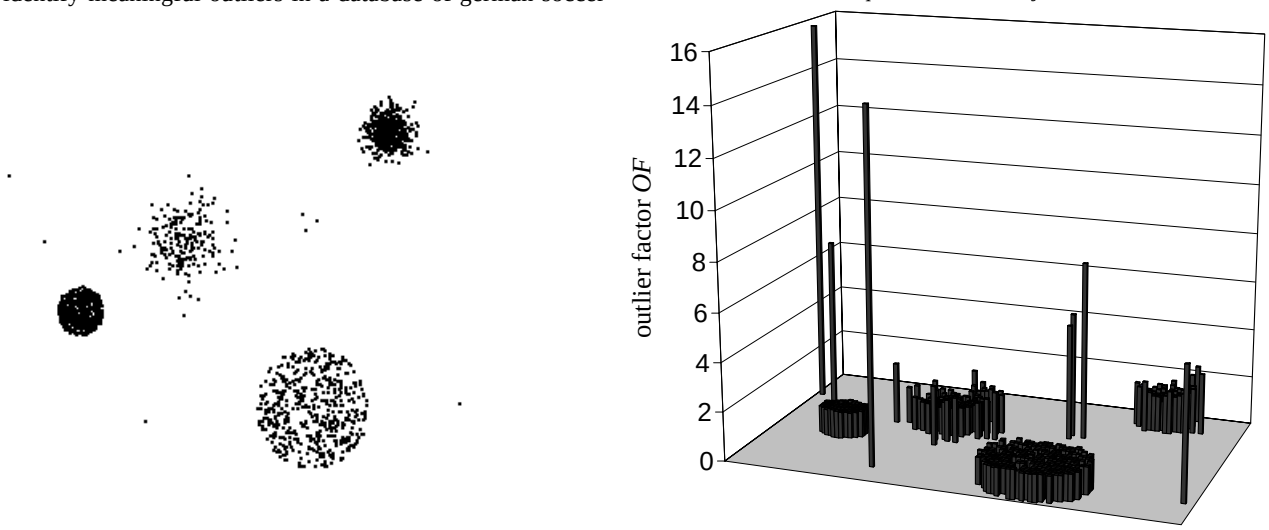


Figure 9: Outlier-factors for points in a sample dataset (*MinPts*=40)

also have 1 as their *LOF* values. Slightly outside the Gaussian clusters, there are several weak outliers, i.e., those with relatively low, but larger than 1, *LOF* values. The remaining seven objects all have significantly larger *LOF* values. Furthermore, it is clear from the figure that the value of the *LOF* for each of these outliers depends on the density of the cluster(s) relative to which the object is an outlier, and the distance of the outlier to the cluster(s).

7.2 Hockey Data

In [13], the authors conducted a number of experiments on historical NHL player data; see [13] for a more detailed explanation of the attributes used. We repeat their experiments on the NHL96 dataset, computing the maximum *LOF* in the *MinPts* range of 30 to 50.

For the first test, on the 3-dimensional subspace of points scored, plus-minus statistics and penalty-minutes, they identified Vladimir Konstantinov as the only *DB*(0.998, 26.3044) outlier. He was also our top outlier with the *LOF* value of 2.4. The second strongest local outlier, with the *LOF* of 2.0, is Matthew Barnaby. For most outliers found, we do not explain why they are outliers from a domain-expert standpoint here; the interested reader can find this information in [13]. The point here is that by ranking outliers with their maximum *LOF* value, we get almost identical results. In the next subsection, we show how this approach can identify some outliers that [13] cannot find.

In the second test, they identified the *DB*(0.997, 5) outliers in the 3-dimensional subspace of games played, goals scored and shooting percentage, finding Chris Osgood and Mario Lemieux as outliers. Again, they are our top outliers, Chris Osgood with the *LOF* of 6.0 and Mario Lemieux with the *LOF* of 2.8. On our ranked list based on *LOF*, Steve Poapst, ranked third with the *LOF* of 2.5, played only three games, scored once and had a shooting percentage of 50%.

7.3 Soccer Data

In the following experiment, we computed the local outliers for a database of soccer-player information from the "Fußball 1. Bundesliga" (the German national soccer league) for the season 1998/99. The database consists of 375 players, containing the name, the number of games played, the number of goals scored and the position of the player (goalie, defense, center, offense). From these we derived the average number of goals scored per game, and performed outlier detection on the three-dimensional subspace of number of games, average number of goals per game and position (coded as an integer). In general, this dataset can be partitioned into four clusters corresponding to the positions of the players. We computed the *LOF* values in the *MinPts* range of 30 to 50. Below we discuss all the local outliers with *LOF* > 1.5 (see table 3), and explain why they are exceptional.

The strongest outlier is Michael Preetz, who played the maximum number of games and also scored the maximum number of goals, which made him the top scorer in the league ("Torschützenkönig"). He was an outlier relative to the cluster of offensive players. The second strongest outlier is Michael Schjönberg. He played an average number of games, but he was an outlier because most other defense players had a much lower average number of goals scored per game. The reason for this is that he kicked the penalty shots ("Elfmeter") for his team. The player that was ranked third is Hans-Jörg Butt, a goalie who played the maximum number of games possible

Rank	Outlier Factor	Player Name	Games Played	Goals Scored	Position
1	1.87	Michael Preetz	34	23	Offense
2	1.70	Michael Schjönberg	15	6	Defense
3	1.67	Hans-Jörg Butt	34	7	Goalie
4	1.63	Ulf Kirsten	31	19	Offense
5	1.55	Giovane Elber	21	13	Offense
minimum			**0**	**0**	
median			**21**	**1**	
maximum			**34**	**23**	
mean			**18.0**	**1.9**	
standard deviation			**11.0**	**3.0**	

Table 3: Results of the soccer player dataset

and scored 7 goals. He was the only goalie to score *any* goal; he too kicked the penalty shots for his team. On rank positions four and five, we found Ulf Kirsten and Giovane Elber, two offensive players with very high scoring averages.

7.4 Performance

In this section, we evaluate the performance of the computation of *LOF*. The following experiments were conducted on an Pentium III-450 workstation with 256 MB main memory running Linux 2.2. All algorithms were implemented in Java and executed on the IBM JVM 1.1.8. The datasets used were generated randomly, containing different numbers of Gaussian clusters of different sizes and densities. All times are wall-clock times, i.e. include CPU-time and I/O.

To compute the *LOF* values within the range between *MinPtsLB* and *MinPtsUB*, for all the *n* objects in the database *D*, we implemented a two-step algorithm. In the first step, the *MinPtsUB*-nearest neighborhoods are found, and in the second step the *LOF*s are computed. Let us look at these two steps in detail.

In the first step, the *MinPtsUB*-nearest neighbors for every point *p* are materialized, together with their distances to *p*. The result of this step is a materialization database *M* of size *n*MinPtsUB* distances. Note that the size of this intermediate result is independent of the dimension of the original data. The runtime complexity of this step

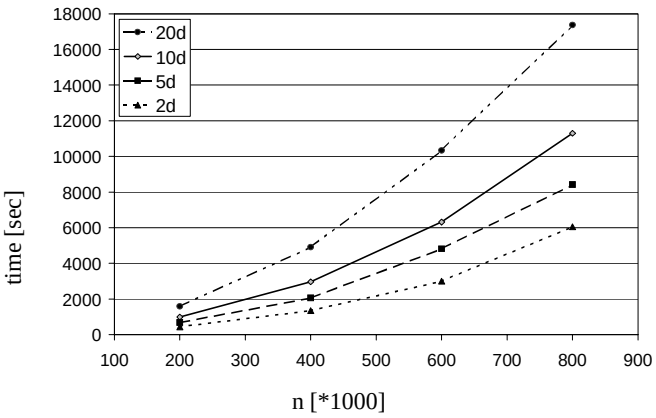


Figure 10: Runtime of the materialization of the 50-nn queries for different dataset sizes and different dimensions using an index

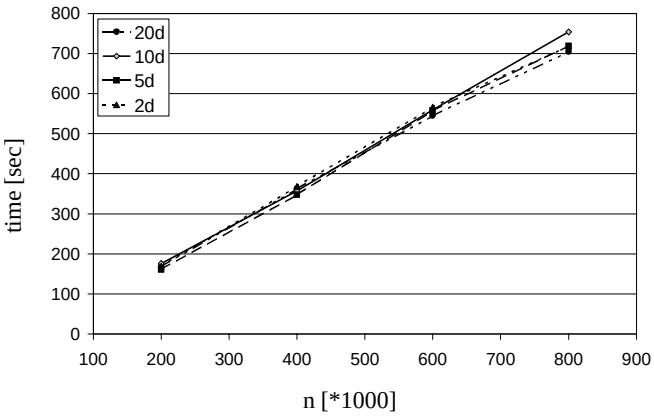


Figure 11: Runtime for the computation of the LOFs for different dataset sizes

is $O(n$*time for a *k-nn* query). For the *k-nn* queries, we have a choice among different methods. For low-dimensional data, we can use a grid based approach which can answer *k-nn* queries in constant time, leading to a complexity of $O(n)$ for the materialization step. For medium to medium high-dimensional data, we can use an index, which provides an average complexity of $O(\log n)$ for *k-nn* queries, leading to a complexity of $O(n \log n)$ for the materialization. For extremely high-dimensional data, we need to use a sequential scan or some variant of it, e.g. the VA-file ([21]), with a complexity of $O(n)$, leading to a complexity of $O(n^2)$ for the materialization step. In our experiments, we used a variant of the X-tree ([4]), leading to the complexity of $O(n \log n)$. Figure 10 shows performance experiments for different dimensional datasets and *MinPtsUB*=50. The times shown do include the time to build the index. Obviously, the index works very well for 2-dimensional and 5-dimensional dataset, leading to a near linear performance, but degenerates for the 10-dimensional and 20-dimensionsal dataset. It is a well known effect of index structures, that their effectivity decreases with increasing dimension.

In the second step, the *LOF* values are computed using the materialization database *M*. The original database *D* is not needed for this step, as *M* contains sufficient information to compute the *LOF*s. The database *M* is scanned twice for every value of *MinPts* between *MinPtsLB* and *MinPtsUB*. In the first scan, the local reachability densities of every object are computed. In the second step, the final *LOF* values are computed and written to a file. These values can then be used to rank the objects according to their maximum *LOF* value in the interval of *MinPtsLB* and *MinPtsUB*. The time complexity of this step is $O(n)$. This is confirmed by the graph shown in figure 11, where the *LOF* values for *MinPtsLB*=10 to *MinPtsUB*=50 were computed.

8. CONCLUSIONS

Finding outliers is an important task for many KDD applications. Existing proposals consider being an outlier as a binary property. In this paper, we show that for many situations, it is meaningful to consider being an outlier not as a binary property, but as the degree to which the object is isolated from its surrounding neighborhood. We introduce the notion of the local outlier factor *LOF*, which captures exactly this relative degree of isolation. We show that our definition of *LOF* enjoys many desirable properties. For objects deep inside a cluster, the *LOF* value is approximately 1. For other objects, we give tight lower and upper bounds on the *LOF* value, regardless of whether the *MinPts*-nearest neighbors come from one or more clusters. Furthermore, we analyze how the *LOF* value depends on the *MinPts* parameter. We give practical guidelines on how to select a range of *MinPts* values to use, and propose the heuristic of ranking objects by their maximum *LOF* value within the selected range. Experimental results demonstrate that our heuristic appears to be very promising in that it can identify meaningful local outliers that previous approaches cannot find. Last but not least, we show that our approach of finding local outliers is efficient for datasets where the nearest neighbor queries are supported by index structures and still practical for very large datasets.

There are two directions for ongoing work. The first one is on how to describe or explain why the identified local outliers are exceptional. This is particularly important for high-dimensional datasets, because a local outlier may be outlying only on some, but not on all, dimensions (cf. [14]). The second one is to further improve the performance of *LOF* computation. For both of these directions, it is interesting to investigate how *LOF* computation can "handshake" with a hierarchical clustering algorithm, like OPTICS [2]. On the one hand, such an algorithm may provide more detailed information about the local outliers, e.g., by analyzing the clusters relative to which they are outlying. On the other hand, computation may be shared between *LOF* processing and clustering. The shared computation may include *k-nn* queries and reachability distances.

References

[1] Arning, A., Agrawal R., Raghavan P.: "A Linear Method for Deviation Detection in Large Databases", Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, 1996, p. 164-169.

[2] Ankerst M., Breunig M. M., Kriegel H.-P., Sander J.: "*OPTICS: Ordering Points To Identify the Clustering Structure*", Proc. ACM SIGMOD Int. Conf. on Management of Data, Philadelphia, PA, 1999.

[3] Agrawal R., Gehrke J., Gunopulos D., Raghavan P.: "*Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications*", Proc. ACM SIGMOD Int. Conf. on Management of Data, Seattle, WA, 1998, pp. 94-105.

[4] Berchthold S., Keim D. A., Kriegel H.-P.: "*The X-Tree: An Index Structure for High-Dimensional Data*", 22nd Conf. on Very Large Data Bases, Bombay, India, 1996, pp. 28-39.

[5] Barnett V., Lewis T.: "*Outliers in statistical data*", John Wiley, 1994.

[6] DuMouchel W., Schonlau M.: "*A Fast Computer Intrusion Detection Algorithm based on Hypothesis Testing of Command Transition Probabilities*", Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, New York, NY, AAAI Press, 1998, pp. 189-193.

[7] Ester M., Kriegel H.-P., Sander J., Xu X.: *"A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise"*, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, 1996, pp. 226-231.

[8] Fawcett T., Provost F.: "*Adaptive Fraud Detection*", Data Mining and Knowledge Discovery Journal, Kluwer Academic Publishers, Vol. 1, No. 3, 1997, pp. 291-316.

[9] Fayyad U., Piatetsky-Shapiro G., Smyth P.: "*Knowledge*

Discovery and Data Mining: Towards a Unifying Framework", Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, 1996, pp. 82-88.

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

[11] Hinneburg A., Keim D. A.: *"An Efficient Approach to Clustering in Large Multimedia Databases with Noise"*, Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, New York City, NY, 1998,pp. 58-65.

[12] Johnson T., Kwok I., Ng R.: *"Fast Computation of 2-Dimensional Depth Contours*", Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, New York, NY, AAAI Press, 1998, pp. 224-228.

[13] Knorr E. M., Ng R. T.: *"Algorithms for Mining Distance-Based Outliers in Large Datasets*", Proc. 24th Int. Conf. on Very Large Data Bases, New York, NY, 1998, pp. 392-403.

[14] Knorr E. M., Ng R. T.: "Finding Intensional Knowledge of Distance-based Outliers", Proc. 25th Int. Conf. on Very Large Data Bases, Edinburgh, Scotland, 1999, pp. 211-222.

[15] Ng R. T., Han J.: *"Efficient and Effective Clustering Methods for Spatial Data Mining"*, Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, Morgan Kaufmann Publishers, San Francisco, CA, 1994, pp. 144-155.

[16] Preparata F., Shamos M.: *"Computational Geometry: an Introduction"*, Springer, 1988.

[17] Ramaswamy S., Rastogi R., Kyuseok S.: *"Efficient Algorithms for Mining Outliers from Large Data Sets"*, Proc. ACM SIDMOD Int. Conf. on Management of Data, 2000.

[18] Ruts I., Rousseeuw P.: "Computing Depth Contours of Bivariate Point Clouds, Journal of Computational Statistics and Data Analysis, 23, 1996, pp. 153-168.

[19] Sheikholeslami G., Chatterjee S., Zhang A.: *"WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases"*, Proc. Int. Conf. on Very Large Data Bases, New York, NY, 1998, pp. 428-439.

[20] Tukey J. W.: "*Exploratory Data Analysis*", Addison-Wesley, 1977.

[21] Weber R., Schek Hans-J., Blott S.: *"A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces*", Proc. Int. Conf. on Very Large Data Bases, New York, NY, 1998, pp. 194-205.

[22] Wang W., Yang J., Muntz R.: *"STING: A Statistical Information Grid Approach to Spatial Data Mining"*, Proc. 23th Int. Conf. on Very Large Data Bases, Athens, Greece, Morgan Kaufmann Publishers, San Francisco, CA, 1997, pp. 186-195.

[23] Zhang T., Ramakrishnan R., Linvy M.: *"BIRCH: An Efficient Data Clustering Method for Very Large Databases"*, Proc. ACM SIGMOD Int. Conf. on Management of Data, ACM Press, New York, 1996, pp.103-114.

Appendix

Proof of Theorem 2 (Sketch): Let p be an object from the database D, $1 \le MinPts \le |D|$, and $C_1, C_2, ..., C_n$ be a partition of $N_{MinPts}(p)$, i.e. $N_{MinPts}(p) = C_1 \cup C_2 \cup \ ... \cup C_n \cup \{p\}$ with $C_i \cap C_j = \emptyset$, $C_i \neq \emptyset$ for $1 \le i,j \le n$, $i \neq j$. Furthermore, let $\xi_i = |C_i| / |N_{MinPts}(p)|$ be the percentage of objects in p's neighborhood which are in the set C_i. Let the notions $direct^i_{min}(p)$, $direct^i_{max}(p)$, $indirect^i_{min}(p)$, and $indirect^i_{max}(p)$ be defined analogously to $direct_{min}(p)$, $direct_{max}(p)$, $indirect_{min}(p)$, and $indirect_{max}(p)$ but restricted to the set C_i.

(a)

$$LOF(p) \ge \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{min}(p) \right) \cdot \left(\sum_{i=1}^{n} \frac{\xi_i}{indirect^i_{max}(p)} \right)$$

$\forall o \in C_i$: $reach\text{-}dist(p, o) \ge direct^i_{min}(p)$, by definition of $direct^i_{min}(p)$. $\Rightarrow$

$$1 / \frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist(p, o)}{|N_{MinPts}(p)|} = \left(\sum_{i=1}^{n} \sum_{o \in C_i} \frac{reach\text{-}dist(p, o)}{|N_{MinPts}(p)|} \right)^{-1}$$

$$\le \left(\sum_{i=1}^{n} \sum_{o \in C_i} \frac{direct^i_{min}(p)}{|N_{MinPts}(p)|} \right)^{-1} =$$

$$= \left(\sum_{i=1}^{n} \frac{|C_i| \cdot direct^i_{min}(p)}{|N_{MinPts}(p)|} \right)^{-1} = \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{min}(p) \right)^{-1}$$

i.e. $lrd(p) \le \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{min}(p) \right)^{-1}$

$\forall q \in N_{MinPts}(o)$: $\quad reach\text{-}dist(o, q) \le indirect^i_{max}(p)$

$\Rightarrow lrd(o) \ge \dfrac{1}{indirect^i_{max}(p)}$ .Thus, it follows that

$$LOF(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd(o)}{lrd(p)}}{|N_{MinPts}(p)|} = \frac{1}{lrd(p)} \cdot \sum_{o \in N_{MinPts}(p)} \frac{lrd(o)}{|N_{MinPts}(p)|}$$

$$\ge \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{min}(p) \right) \cdot \left(\sum_{i=1}^{n} \sum_{o \in C_i} \frac{\frac{1}{indirect^i_{max}(p)}}{|N_{MinPts}(p)|} \right)$$

$$= \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{min}(p) \right) \cdot \left(\sum_{i=1}^{n} \frac{\xi_i}{indirect^i_{max}(p)} \right)$$

(b)

$$LOF(p) \le \left(\sum_{i=1}^{n} \xi_i \cdot direct^i_{max}(p) \right) \cdot \left(\sum_{i=1}^{n} \frac{\xi_i}{indirect^i_{min}(p)} \right)$$

: analogously. ■