Angle-Based Outlier Detection in High-dimensional Data

Hans-Peter Kriegel Matthias Schubert Arthur Zimek
Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München, Germany
http://www.dbs.ifi.lmu.de
{kriegel,schubert,zimek}@dbs.ifi.lmu.de

ABSTRACT

Detecting outliers in a large set of data objects is a major data mining task aiming at finding different mechanisms responsible for different groups of objects in a data set. All existing approaches, however, are based on an assessment of distances (sometimes indirectly by assuming certain distributions) in the full-dimensional Euclidean data space. In high-dimensional data, these approaches are bound to deteriorate due to the notorious "curse of dimensionality". In this paper, we propose a novel approach named ABOD (Angle-Based Outlier Detection) and some variants assessing the variance in the angles between the difference vectors of a point to the other points. This way, the effects of the "curse of dimensionality" are alleviated compared to purely distance-based approaches. A main advantage of our new approach is that our method does not rely on any parameter selection influencing the quality of the achieved ranking. In a thorough experimental evaluation, we compare ABOD to the well-established distance-based method LOF for various artificial and a real world data set and show ABOD to perform especially well on high-dimensional data.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications—*Data mining*

General Terms: Algorithms

Keywords: outlier detection, high-dimensional, angle-based

1. INTRODUCTION

The general idea of outlier detection is to identify data objects that do not fit well in the general data distributions. This is a major data mining task and an important application in many fields such as detection of credit card abuse in financial transactions data, or the identification of measurement errors in scientific data. The reasoning is that data objects (observations) are generated by certain mechanisms or statistical processes. Distinct deviations from the main distributions then are supposed to originate from a different mechanism. Such a different mechanism may be a fraud, a disturbance impairing the sensors, or simply incorrect reading of the measurement equipment. But it could also be an unexpected and therefore interesting behavior requiring an adaptation of the theory underlying to the experiment in question. This ambivalence in the meaning of outliers is expressed in the frequently cited sentence "one person's noise is another person's signal". Thus, a well known characterization of an outlier is given by Hawkins as being "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [12]. This general idea has been addressed by very diverse approaches pursuing very different intuitions and sometimes also different notions of what exactly constitutes an outlier. We will discuss different approaches in more detail in Section 2. Some approaches are, due to their computational complexity, simply not applicable to high-dimensional data. However, all known methods that are, at least theoretically, applicable to high-dimensional data are based on the evaluation of ε-range queries or k-nearest neighborhoods for local methods, or, in general, assessments of differences in distances between objects (e.g. in computing data distributions). This makes all approaches known so far more or less unsuitable for high-dimensional data due to the notorious "curse of dimensionality". One of the most thoroughly discussed effects of this malediction of mining high-dimensional data is that concepts like proximity, distance, or nearest neighbor become less meaningful with increasing dimensionality of data sets [7, 13, 1]. Roughly, the results in these studies state that the relative contrast of the farthest point and the nearest point converges to 0 for increasing dimensionality d:

$$\lim_{d\to\infty} \frac{dist_{\max} - dist_{\min}}{dist_{\min}} \to 0$$

This means, the discrimination between the nearest and the farthest neighbor becomes rather poor in high dimensional space. These observations are valid for a broad range of data distributions and occur simply based on the mere number of dimensions even if all attributes are relevant. Independently, the problem worsens with irrelevant attributes which are likely to emerge in high-dimensional data. Such attributes are related to as "noise". However, global feature reduction methods may be inadequate to get rid of noise attributes because, often, there is no global noise, but certain attributes are noisy only w.r.t. certain sets of objects. All these effects are far more fundamental problems then mere complexity issues and trigger the exploration of data mining methods that are less dependent on the mere distances between objects. In this paper, we propose a new method

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.

of outlier detection that still takes distances into account, but only as a secondary measure to normalize the results. The main contribution to detecting outliers is in considering the variances of the angles between the difference vectors of data objects. This measure appears to be far less sensitive to an increasing dimensionality of a data set than distance based criteria.

In the remainder of this paper, we will first discuss different approaches to outlier detection in more detail in Section 2. In Section 3, we introduce our new approach and discuss its properties. We evaluate the proposed method in Section 4. In Section 5, we conclude the paper.

2. RELATED WORK

The general problem of identifying outliers has been addressed by very different approaches that can be roughly classified as global versus local outlier models. A global outlier model leads to a binary decision of whether or not a given object is an outlier. A local outlier approach rather assigns a degree of outlierness to each object. Such an "outlier factor" is a value characterizing each object in "how much" this object is an outlier. In those applications where it is interesting to rank the outliers in the database and to retrieve the top-n outliers, a local outlier approach is obviously preferable. A different classification of outlier approaches discerns between supervised and unsupervised approaches. A supervised approach is based on a set of observations where the status of being an outlier or not is known and the differences between those different types of observations is learned. An example for this type of approaches is [33]. Usually, supervised approaches are also global approaches and can be considered as very unbalanced classification problems (since the class of outliers has inherently relatively few members only). However, in most cases outlier detection is encountered as an unsupervised problem since one does not have enough previous knowledge for supervised learning. Statistical approaches to the identification of outliers are based on presumed distributions of objects. The classical textbook of Barnett and Lewis [5] discusses numerous tests for different distributions. The tests are optimized for each distribution dependent on the specific parameters of the corresponding distribution, the number of expected outliers, and the space where to expect an outlier. Problems of these classical approaches are obviously the required assumption of a specific distribution in order to apply a specific test. Furthermore, all tests are univariate and examine a single attribute to determine an outlier. Related approaches combine given models and supervised learning methods but still assume the distribution of objects to be known in advance [31, 32]. Sometimes, the data are assumed to consist of k Gaussian distributions and the means and standard deviations are computed data driven. However, these methods are not really robust, since mean and standard deviation are rather sensitive to outliers and the potential outliers are still considered for the computation step. In [25], a more robust estimation of the mean and the standard deviation is proposed in order to tackle this problem. Depth based approaches organize data objects in convex hull layers expecting outliers from data objects with shallow depth values only [30, 26, 16]. These approaches from computer graphics are infeasible for data spaces of high dimensionality due to the inherent exponential complexity of computing convex hulls. Deviation-based outlier detection groups objects and considers those objects as outliers that deviate considerably from the general characteristics of the groups. This approach has been pursued e.g. in [4, 27]. The forming of groups at random is rather arbitrary and so are the results depending on the selected groups. Forming groups at random, however, is inevitable in order to avoid exponential complexity. The distance based notion of outliers unifies distribution based approaches [17, 18]. An object $x \in \mathcal{D}$ is an outlier if at least a fraction p of all data objects in $\mathcal{D}$ has a distance above D from x. Variants of the distance based notion of outliers are [24], [20], and [6]. In [24], the distances to the k nearest neighbors are used and the objects are ranked according to their distances to their k-th nearest neighbor. A partition-based algorithm is then used to efficiently mine top-n outliers. An approximation solution to enable scalability with increasing data dimensionality is proposed in [3]. However, as adaptation to high-dimensional data, only the time-complexity issue is tackled. The inherent problems of high-dimensional data are not addressed by this or any other approach. On the contrary, the problems are even aggravated since the approximation is based on space filling curves. Another approximation based on reference points is proposed in [23]. This approximation, too, is only on low-dimensional data shown to be valuable. The idea of using the k nearest neighbors already resembles density based approaches that consider ratios between the local density around an object and the local density around its neighboring objects. These approaches therefore introduce the notion of local outliers. The basic idea is to assign a density-based local outlier factor (LOF) to each object of the database denoting a degree of outlierness [8]. The LOF compares the density of each object o of a database $\mathcal{D}$ with the density of the k nearest neighbors of o. A LOF value of approximately 1 indicates that the corresponding object is located within a region of homogeneous density (i.e. a cluster). If the difference between the density in the local neighborhood of o and the density around the k nearest neighbors of o is higher, o gets assigned a higher LOF value. The higher the LOF value of an object o is, the more distinctly is o considered an outlier. Several extensions and refinements of the basic LOF model have been proposed, e.g. a connectivity-based outlier factor (COF) [29] or a spatial local outlier measure (SLOM) [28]. Using the concept of micro-clusters to efficiently mine the top-n density-based local outliers in large databases (i.e., those n objects having the highest LOF value) is proposed in [14]. A similar algorithm is presented in [15] for an extension of the LOF model using also the reverse nearest neighbors additionally to the nearest neighbors and considering a symmetric relationship between both values as a measure of outlierness. In [22], the authors propose another local outlier detection schema named Local Outlier Integral (LOCI) based on the concept of a multi-granularity deviation factor (MDEF). The main difference between the LOF and the LOCI outlier model is that the MDEF of LOCI uses ε-neighborhoods rather than k nearest neighbors. The authors propose an approximative algorithm computing the LOCI values of each database object for any ε value. The results are displayed as a rather intuitive outlier plot. This way, the approach becomes much less sensitive to input parameters. Furthermore, an exact algorithm is introduced for outlier detection based on the LOCI model. The resolution-based outlier factor (ROF) [9] is a mix of the local and the global outlier paradigm. The

outlier schema is based on the idea of a change of resolution. Roughly, the "resolution" specifies the number of objects considered to be neighbors of a given data object and, thus, is a data driven concept based on distances rather than on concepts like the k nearest neighbors or an ε-neighborhood that rely on user-specified parametrization. An approach claimed to be suitable for high dimensional data is proposed in [2]. The idea resembles a grid-based subspace clustering approach where not dense but sparse grid cells are sought to report objects within sparse grid cells as outliers. Since this is exponential in the data dimensionality, an evolutionary algorithm is proposed to search heuristically for sparse cells.As an extension of the distance based outlier detection, some algorithms for finding an explanation for the outlierness of a point are proposed in [19]. The idea is to navigate through the lattice of combinations of attributes and to find the most significant combination of attributes where the point is an outlier. This is an interesting feature because an explicit and concise explanation why a certain point is considered to be an outlier (so that a user could conveniently gain some insights in the nature of the data) has not been provided by any other outlier detection model so far. In summary, we find all outlier models proposed so far inherently unsuitable for the requirements met in mining high-dimensional data since they rely implicitly or explicitly on distances. Aiming to explain why a point is an outlier, we found only one other approach proposed in the literature deriving subsets of attributes where an object is an outlier most significantly, based on a global outlier model. In the classification of outlier models, our new approach is unsupervised and can be regarded as a local approach. Generally, local outlier detection models have shown better accuracy than global outlier detection models. Therefore, as one of the most prominent local methods, LOF will be used as competitor in comparison to our new approach.

3. ANGLE-BASED OUTLIER DETECTION

3.1 General Idea

As elaborated above (see Section 1), comparing distances becomes more and more meaningless with increasing data dimensionality. Thus, mining high-dimensional data requires different approaches to the quest for patterns. Here, we propose not only to use the distance between points in a vector space but primarily the directions of distance vectors. Comparing the angles between pairs of distance vectors to other points helps to discern between points similar to other points and outliers. This idea is motivated by the following intuition. Consider a simple data set as illustrated in Figure 1. For a point within a cluster, the angles between difference vectors to pairs of other points differ widely. The variance of the angles will become smaller for points at the border of a cluster. However, even here the variance is still relatively high compared to the variance of angles for real outliers. Here, the angles to most pairs of points will be small since most points are clustered in some directions. The corresponding spectra for these three types of points are illustrated for a sample data set in Figure 2. As the graph shows, the spectrum of angles to pairs of points remains rather small for an outlier whereas the variance of angles is higher for border points of a cluster and very high for inner points of a cluster. As a result of these considerations, an angle-based outlier factor (ABOF) can describe the

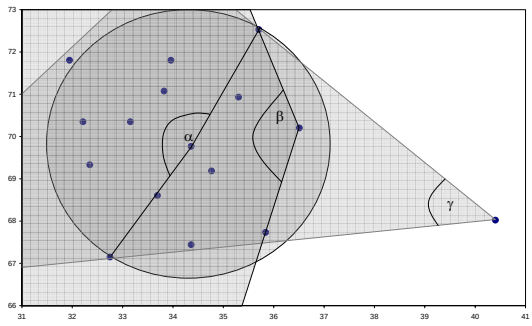


Figure 1: Intuition of angle-based outlier detection.

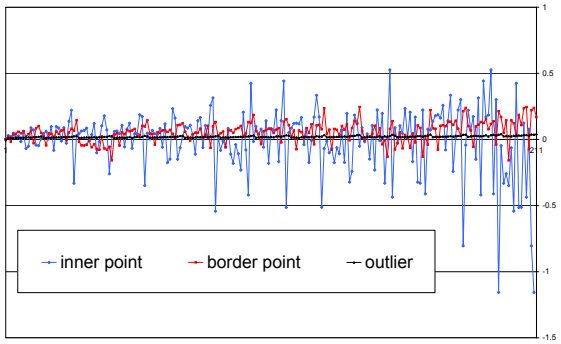


Figure 2: Spectra of angles for different types of points.

divergence in directions of objects relatively to one another. If the spectrum of observed angles for a point is broad, the point will be surrounded by other points in all possible directions meaning the point is positioned inside a cluster. If the spectrum of observed angles for a point is rather small, other points will be positioned only in certain directions. This means, the point is positioned outside of some sets of points that are grouped together. Thus, rather small angles for a point $\vec{P}$ that are rather similar to one another imply that $\vec{P}$ is an outlier.

3.2 Angle-based Outlier Detection (ABOD)

As an approach to assign the ABOF value to any object in the database $\mathcal{D}$, we compute the scalar product of the difference vectors of any triple of points (i.e. a query point $\vec{A} \in \mathcal{D}$ and all pairs $(\vec{B}, \vec{C})$ of all remaining points in $\mathcal{D} \setminus \{\vec{A}\}$) normalized by the quadratic product of the length of the difference vectors, i.e. the angle is weighted less if the corresponding points are far from the query point. By this weighting factor, the distance influences the value after all, but only to a minor part. Nevertheless, this weighting of the variance is important since the angle to a pair of points varies naturally stronger for a bigger distance. The variance of this value over all pairs for the query point $\vec{A}$ constitutes the angle-based outlier factor (ABOF) of $\vec{A}$. Formally:

DEFINITION 1 (ABOF).
Given a database $\mathcal{D} \subseteq \mathbb{R}^d$, a point $\vec{A} \in \mathcal{D}$, and a norm $\|.\| : \mathbb{R}^d \to \mathbb{R}_0^+$. The scalar product is denoted by $\langle ., . \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. For two points $\vec{B}, \vec{C} \in \mathcal{D}$, $\overline{BC}$ denotes the difference vector $\vec{C} - \vec{B}$.

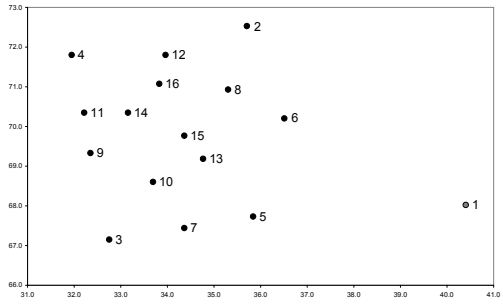


Figure 3: Ranking of points in the sample data set according to ABOF.

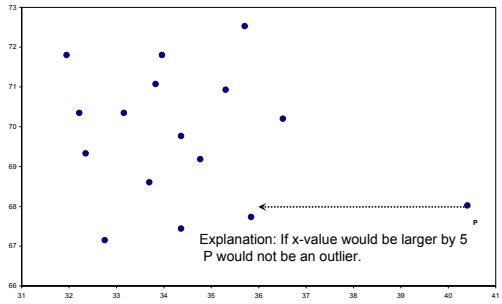


Figure 4: Explanation for an outlier as provided by ABOD.

The angle-based outlier factor $ABOF(\vec{A})$ *is the variance over the angles between the difference vectors of* $\vec{A}$ *to all pairs of points in* $\mathcal{D}$ *weighted by the distance of the points:*

$$\begin{aligned} ABOF(\vec{A}) &= VAR_{\vec{B},\vec{C}\in\mathcal{D}} \left(\frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2} \right) \\ &= \frac{\sum_{\vec{B}\in\mathcal{D}} \sum_{\vec{C}\in\mathcal{D}} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|} \cdot \left(\frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2} \right)^2}{\sum_{\vec{B}\in\mathcal{D}} \sum_{\vec{C}\in\mathcal{D}} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|}} \\ &\quad - \left(\frac{\sum_{\vec{B}\in\mathcal{D}} \sum_{\vec{C}\in\mathcal{D}} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|} \cdot \frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2}}{\sum_{\vec{B}\in\mathcal{D}} \sum_{\vec{C}\in\mathcal{D}} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|}} \right)^2 \end{aligned}$$

Note that for each triple of points in question, $(\vec{A}, \vec{B}, \vec{C})$, the three points are mutual different. This means, instead of $\vec{B} \in \mathcal{D}$ and $\vec{C} \in \mathcal{D}$, the definition more exactly reads as $\vec{B} \in \mathcal{D} \setminus \{\vec{A}\}$ and $\vec{C} \in \mathcal{D} \setminus \{\vec{A}, \vec{B}\}$, respectively. We spared this in favor of readability in this definition as well as in the following ones.

The algorithm ABOD assigns the angle-based outlier factor ABOF to each point in the database and returns as a result the list of points sorted according to their ABOF. Consider again the sample data set in Figure 1. The ranking of these points as provided by ABOD is denoted in Figure 3. In this toy-example, the top-ranked point (rank 1) is clearly the utmost outlier. The next ranks are taken by border points of the cluster. The lowest ranks are assigned to the inner points of the cluster. Since the distance is accounted for only as a weight for the main criterion, the variance of angles, ABOD is able to concisely detect outliers even in high-dimensional data where LOF and other purely distance-based approaches deteriorate in accuracy. Furthermore, as illustrated above, ABOD allows also a different ranking of border points versus inner points of a cluster. This is not possible for most of the other outlier models.

Most outlier detection models require the user to specify parameters that are crucial to the outcome of the approach. For unsupervised approaches, such requirements are always a drawback. Thus, a big advantage of ABOD is being completely free of parameters. On the fly, ABOD retrieves an explanation why the point is considered to be an outlier. The difference vector to the most similar object in the nearest group of points provides the divergence quantitatively for each attribute and, thus, explains why (i.e., in which attributes by how much) the point is an outlier. For the running example, the explanation for the top-ranked outlier is that it deviates from the nearest point of the nearest cluster by the difference vector as illustrated in Figure 4.

3.3 Speed-up by Approximation (FastABOD)

A problem of the basic approach ABOD is obvious: since for each point all pairs of points must be considered, the time-complexity is in $O(n^3)$ which is not attractive compared e.g. to LOF which is in $O(n^2 \cdot k)$. In this section, we therefore discuss also an approximation algorithm. This approximation algorithm, FastABOD, approximates ABOF based on a sample of the database. We propose to use the pairs of points with the strongest weight in the variance, e.g. pairs between the k nearest neighbors. Let us note that a random set of k arbitrary data points could be used as well for this approximation. However, the nearest neighbors have the largest weights in the ABOF. Thus, employing the nearest neighbors might result in a better approximation, especially in data sets of low dimensionality where the distance is more meaningful.

The ABOF relaxes to an approximate ABOF as follows:

Definition 2 (Approximate ABOF).
Given a database $\mathcal{D} \subseteq \mathbb{R}^d$*, a point* $\vec{A} \in \mathcal{D}$*, and a norm* $\|.\| : \mathbb{R}^d \to \mathbb{R}_0^+$*. The scalar product is denoted by* $\langle ., . \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$*. For two points* $\vec{B}, \vec{C} \in \mathcal{D}$*,* $\overline{BC}$ *denotes the difference vector* $\vec{C} - \vec{B}$*.* $\mathcal{N}_k(\vec{A}) \subseteq \mathcal{D}$ *denotes the set of the* k *nearest neighbors of* $\vec{A}$*. The* approximate angle-based outlier factor $approxABOF_k(\vec{A})$ *is the variance over the angles between the difference vectors of* $\vec{A}$ *to all pairs of points in* $\mathcal{N}_k(\vec{A})$ *weighted by the distance of the points:*

$$\begin{aligned} approxABOF_k(\vec{A}) &= VAR_{\vec{B},\vec{C}\in\mathcal{N}_k(\vec{A})} \left(\frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2} \right) \\ &= \frac{\sum_{\vec{B}\in\mathcal{N}_k(\vec{A})} \sum_{\vec{C}\in\mathcal{N}_k(\vec{A})} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|} \cdot \left(\frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2} \right)^2}{\sum_{\vec{B}\in\mathcal{N}_k(\vec{A})} \sum_{\vec{C}\in\mathcal{N}_k(\vec{A})} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|}} \\ &\quad - \left(\frac{\sum_{\vec{B}\in\mathcal{N}_k(\vec{A})} \sum_{\vec{C}\in\mathcal{N}_k(\vec{A})} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|} \cdot \frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2}}{\sum_{\vec{B}\in\mathcal{N}_k(\vec{A})} \sum_{\vec{C}\in\mathcal{N}_k(\vec{A})} \frac{1}{\|\overline{AB}\|\cdot\|\overline{AC}\|}} \right)^2 \end{aligned}$$

This approximation results in an acceleration of one order of magnitude. The resulting algorithm FastABOD is in $O(n^2 + n \cdot k^2)$. This makes FastABOD suitable for data sets

consisting of many points. However, the quality of the approximation depends on the number k of nearest neighbors and the quality of the selection of nearest neighbors. This quality usually deteriorates with increasing data dimensionality, as discussed above. Indeed, our experiments show that the quality of the approximation and therefore the ranking of outliers becomes worse for higher dimensional data (see Section 4).

3.4 Approximation as Filter-Refinement Approach (LB-ABOD)

We have seen that the approximation FastABOD is not suitable directly for high-dimensional data. Nevertheless, the outliers are usually still on a high rank, albeit occasionally not at the top rank. This observation motivates finally another approximation which is also suitable as a lower bound for ABOF. Having a lower bound approximation for the exact ABOF allows us to retrieve the best outliers more efficiently. In other words, we select candidates for the top l outliers w.r.t. the lower-bound and afterwards refine the candidates until none of the remaining candidates can have a lower ABOF the largest of the best already examined data objects.

To gain a lower bound based on the FastABOD approach, we estimate approxABOF conservatively as follows:

DEFINITION 3 (LB-ABOF).
Given a database $\mathcal{D} \subseteq \mathbb{R}^d$, a point $\vec{A} \in \mathcal{D}$, and a norm $\|.\| : \mathbb{R}^d \to \mathbb{R}_0^+$. The scalar product is denoted by $\langle .,. \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. For two points $\vec{B}, \vec{C} \in \mathcal{D}$, $\overline{BC}$ denotes the difference vector $\vec{C} - \vec{B}$. $\mathcal{N}_k(\vec{A}) \subseteq \mathcal{D}$ denotes the set of the k nearest neighbors of $\vec{A}$.

The lower-bound for the angle-based outlier factor *$LB\text{-}ABOF_k(\vec{A})$ is the conservatively approximated variance over the angles between the difference vectors of $\vec{A}$ to all pairs of points in $\mathcal{D}$ weighted by the distance of the points:*

$$LB\text{-}ABOF_k(\vec{A}) = \frac{\sum_{\vec{B} \in \mathcal{N}_k(\vec{A})} \sum_{\vec{C} \in \mathcal{N}_k(\vec{A})} \frac{1}{\|\overline{AB}\| \cdot \|\overline{AC}\|} \cdot \left(\frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2} \right)^2 + R_1}{\sum_{\vec{B} \in \mathcal{D}} \sum_{\vec{C} \in \mathcal{D}} \frac{1}{\|\overline{AB}\| \cdot \|\overline{AC}\|}} - \left(\frac{\sum_{\vec{B} \in \mathcal{N}_k(\vec{A})} \sum_{\vec{C} \in \mathcal{N}_k(\vec{A})} \frac{1}{\|\overline{AB}\| \cdot \|\overline{AC}\|} \cdot \frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2} + R_2}{\sum_{\vec{B} \in \mathcal{D}} \sum_{\vec{C} \in \mathcal{D}} \frac{1}{\|\overline{AB}\| \cdot \|\overline{AC}\|}} \right)^2$$

In Definition 3, the remainders R_1 and R_2 are responsible for the difference between the approximation based on the sample of the k nearest neighbors and the complete ABOF. Note that this approximation is normalized by the sum of the inverse norms of the difference vectors of all pairs of the complete database, not only all pairs of the set of k nearest neighbors as in the approximate ABOF. However, the sum and squared sum necessary for calculating the variance are approximated only over the k nearest neighbors. Thus, a difference remains to the complete ABOF. We have to make sure that $ABOF(\vec{A}) - LB\text{-}ABOF_k(\vec{A}) \geq 0$. Furthermore, this conservative approximation must be computable much more efficiently than calculating the ABOF. This way, LB-ABOF can serve as a lower bound for a filter-refinement approach.

Since the normalization factor is built by summing up the weights for each angle being observed at $\vec{A}$, a straight forward calculation would have a quadratic complexity. Thus, a first step to efficiently calculating LB-ABOF is to find a linear method to compute the normalization factor. This is possible because of the following observation:

$$\sum_{\vec{B} \in \mathcal{D}} \sum_{\vec{C} \in D} \frac{1}{\|\overline{AB}\| \cdot \|\overline{AC}\|} = \sum_{\vec{B} \in \mathcal{D}} \frac{1}{\|\overline{AB}\|} \cdot \sum_{\vec{C} \in \mathcal{D}} \frac{1}{\|\overline{AC}\|}$$

Thus, we can calculate the sum of the inverse distances over all objects first, and afterwards, we add up the products of the inverse distance vectors of each object $\vec{B}$ with this sum.

To guarantee that LB-ABOF is a conservative approximation of ABOF, we have to find conservative estimates for the remainders R_1 and R_2. R_1 is the remainder of the squared sum. This means, the larger R_1 is the larger is the complete variance. Thus, R_1 has to be selected as small as possible. Since R_1 is a sum of weighted angles, R_1 has to be approximated by 0 which would be the case if all of the observed missing angles would be orthogonal. The second remainder R_2 increases the weighted sum over all angles being squared and subtracted from the square sum. Thus, in order to find a conservative approximation of the ABOF, R_2 has to be as large as possible. To find the largest possible value of R_2, we start by assuming the maximum value of the angle, which is 1, for each addend. However, we have a weighted sum and thus, R_2 is given by the sum of weights for all unknown angles. To efficiently compute R_2, we calculate the complete sum over all possible pairs of objects which can be done in linear time, as shown above. By subtracting the already observed factors from this sum, we can find a maximum approximation of R_2:

$$\sum_{\vec{B} \in \mathcal{D}} \sum_{\vec{C} \in \mathcal{D}} \frac{1}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2} - \sum_{\vec{B} \in \mathcal{N}_k(\vec{A})} \sum_{\vec{C} \in \mathcal{N}_k(\vec{A})} \frac{1}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2}.$$

Based on the conservative approximation LB-ABOF, we propose the following procedure LB-ABOD as efficient algorithm to find the top l outliers w.r.t. to the ABOF :

1. For each point $\vec{A} \in \mathcal{D}$, derive the k points of highest impact (i.e., the k nearest neighbors).
2. Compute LB-ABOF for each point $\vec{A} \in \mathcal{D}$.
3. Organize the database objects in a candidate list ordered ascendingly w.r.t. their assigned LB-ABOF.
4. Determine the exact ABOF for the first l objects in the candidate list, remove them from the candidate list and insert them into the current result list.
5. Remove and examine the next best candidate $\vec{C}$ from the candidate list and determine the exact ABOF. If the ABOF of $\vec{C}$ is smaller than the largest ABOF of an object $\vec{A}$ in the result list, remove $\vec{A}$ from the result list and insert $\vec{C}$ into the result list.
6. If the largest ABOF in the result list is smaller than the smallest approximated ABOF in the candidate list, terminate. Else, proceed with step 5.

This procedure combines the gains of FastABOD in terms of scalability with the size of the data set with the robustness of ABOD w.r.t. the dimensionality of the data set. The time

complexity of the filter step is in $O(n^2 + n \cdot k^2)$ (equal to FastABOD). The refinement for r objects to be refined is in $O(r \cdot n^2)$. Thus, the eventual acceleration w.r.t. ABOD depends on the quality of the lower bound and the resulting number r of objects to refine. We show in Section 4 that the lower bound indeed allows for a considerable speed-up.

3.5 Generalization for Arbitrary Data Types: Kernel-based ABOF

An advantage of distance based approaches to outlier detection is that they are often not restricted to vector spaces. As long as there is a suitable distance function for comparing two objects, it is usually possible to apply these methods. For ABOD, on the other hand, it is necessary to provide a scalar product for comparing the data objects which seems to be more restricting than providing a distance measure. However, due to the recent interest in maximum margin learners and kernel methods in machine learning, a wide variety of meaningful kernel functions for varying types of data has been proposed [11]. Since a kernel function is basically a scalar product in a kernel space, it is possible to find outliers in a wide variety of applications whenever a suitable kernel function is available.

4. EVALUATION

We implemented our new algorithms in Java 1.5. As the most established example for distance based local outlier ranking, we additionally implemented LOF [8] into our framework. All experiments were run on a Dell Precision 690 workstation with 2 XEON 3.0 Ghz CPUs and 16Gb main memory.

4.1 Artificial Data

A large problem when evaluating outlier detection methods is that there are very few real world data sets where it is exactly known which objects are really behaving differently due to belonging to a different mechanism. Though there exist multiple case studies on outlier detection, the question whether an object is an outlier or not is often depending on the point of view. Another problem is that the list of possible outliers is often incomplete making it hard to evaluate whether the algorithm ranked all outliers in the database properly. Therefore, we decided to evaluate our methods on artificially generated data. Thus, we can generate outliers and ordinary data points with respect to the initial definition, i.e. an outlier is a point being generated by a different mechanism than the majority of data objects. To exactly evaluate the behavior of our new method for different dimensionalities and database sizes, we generated multiple data sets having 25, 50 and 100 dimensions. As database sizes (*dbsize*) we selected 500, 1,000, 5,000 and 10,000 data objects.

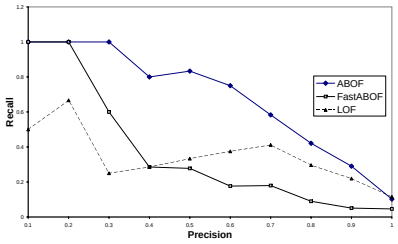
In order to find data sets having well-defined but not obvious outliers, we proceeded as follows. First of all, we randomly generated a Gaussian mixture model consisting of five equally weighted processes having random mean and variance values. This mixture model now describes the ordinary data points, i.e. the none-outlier data points. To build the outliers corresponding to another mechanism that does not assign all the outliers to an additional cluster, we employed a uniform distribution on the complete data space. This way we generated 10 outliers for each data set which are totally independent on the mixture model describing the general data distribution. Let us note that it is possible that some outliers might be generated in an area being populated by none-outlier objects drawn from the Gaussian mixture model. Thus, even if an outlier detection mechanism works well, it does not necessarily have to rank all outliers into top positions.

4.2 Effectiveness

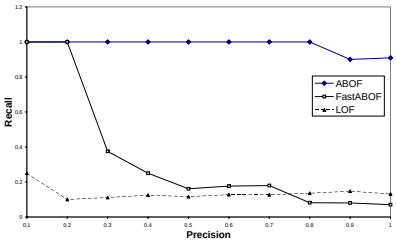
In this set of experiments, we compared the quality of the ranking provided by our new algorithms to each other and to LOF. To measure the capability of each algorithm to retrieve the most likely outliers first, we used precision and recall graphs. In other words, we successively retrieve the most likely outlier until all ten outliers are retrieved. For each result set, we measured precision and recall. Since the recall is the percentage of all outliers in the data set which were already retrieved, we can observe a new recall level for each additional outlier being found. For each of these recall levels, we now measure the precision, i.e. how many objects in the current result set are indeed outliers. In our experiments, we compared FastABOD, ABOD and LOF for multiple database sizes and dimensionalities. For LOF, we varied the parameter $MinPts$ from 10 to 25. The sample size of FastABOD was determined by $0.1 \cdot dbsize$.

Figure 5 displays the observed precision recall graphs for two different database sizes, 1000 and 5000 data points. For each size, we compare three different dimensionalities: 25, 50 and 100. Starting with the comparably low dimensionality of 25 (cf. Figure 5(a) and Figure 5(d)), it can be observed that all methods had difficulties in both data sets with detecting all ten outliers. In Figure 5(a), ABOD clearly offered the best ranking while both other methods FastABOD and LOF did not perform very well for the larger recall levels. In Figure 5(d), all three methods offered comparable results. To conclude the advantage of angle based outlier detection are not very evident on this comparably low dimensional data set. The next two data sets contain 50 dimensional feature points (cf. Figure 5(b) and Figure 5(e)). In this medium dimensionality, ABOD starts to display its advantages for high-dimensional data. While LOF performs very bad on both data sets, FastABOD still can compete with ABOD for small recall levels. However, while ABOD performs almost perfect on the smaller data set (cf. Figure 5(b)) by ranking all ten outliers between the top 11 objects, FastABOD only retrieves two outliers before retrieving a none-outlier object. In the larger data set (cf. Figure 5(e)), ABOD achieves a perfect performance for the first five recall levels. However, for the remaining 5 objects the ranking of ABOD contains some none-outliers before retrieving all outliers. Nevertheless, ABOD provides the best outlier ranking for the 50 dimensional data sets.

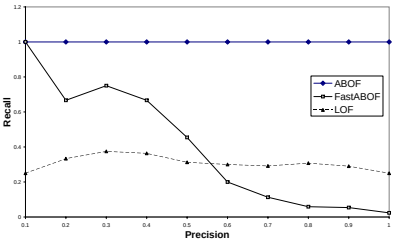
Finally, we tested the methods on a high dimensional data set of about 100 dimensions. In both data sets, ABOD ranked all ten outliers first, i.e. ABOD achieved the maximum possible performance. For the smaller data set, FastABOD performed rather bad, being even overtaken by LOF for the larger recall values. For the large data set, FastABOD performed almost perfect with only one non-outlier object at the tenth position of the ranking before retrieving all ten outliers. As expected LOF performed significantly worse on the high dimensional data. Let us note that it is not feasible to compare the performance of the methods between the different plots because the test data set were generated



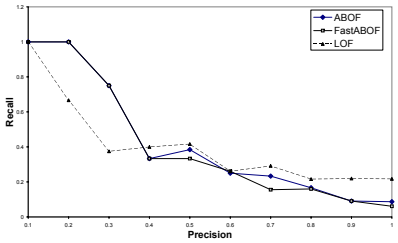
(a) 25 dimensions and 1000 data points.



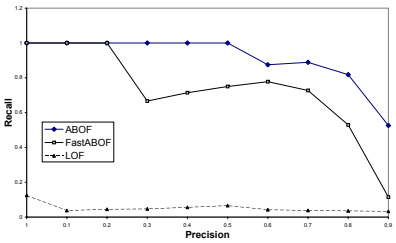
(b) 50 dimensions and 1000 data points.



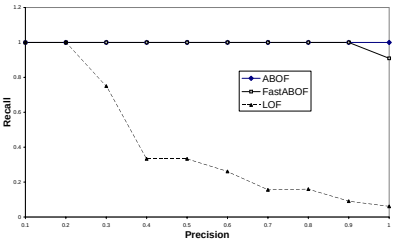
(c) 100 dimensions and 1000 data points.



(d) 25 dimensions and 5000 data points.



(e) 50 dimensions and 5000 data points.



(f) 100 dimensions and 5000 data points.

Figure 5: Precision-Recall graphs on artificial data sets for 25, 50 and 100 dimensions. The upper row describes the behavior on 1000 data points and the lower row the behavior on 5000 data points

independently. Thus, it is possible that LOF performs better on the larger 100 dimensional data set compared to some lower dimensional data sets. The reason for this effect is that the generated outliers sometimes might be more difficult to detect. In other words, the difficulty of the problem is randomly determined for each data set and cannot easily be adjusted to be comparable between data sets. The strongly varying performance of FastABOD can be explained by the fact that FastABOD is strongly dependent on a suitable selection of the sample size which will be discussed more deeply in a later experiment. In contrast, ABOD is not dependent on finding a suitable parametrization. Therefore, it is more suitable for applications where it is not obvious whether the given object is an outlier or not. To conclude, ABOD provided a better ranking w.r.t. the precision of the top ranked objects. As can be seen in our experiments the performance of ABOD is constantly good even for large dimensionalities where LOF and the partly distance-based FastABOD approach do not provide top quality outlier rankings.

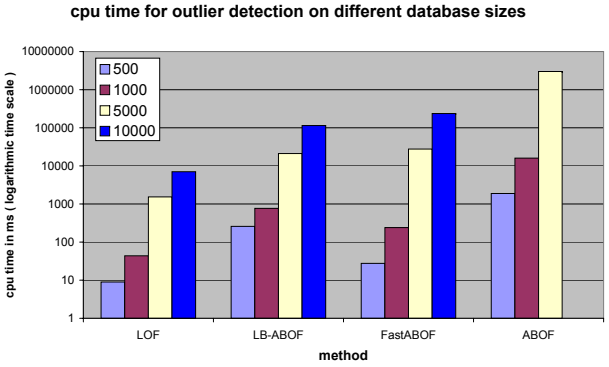


Figure 6: Comparison of CPU times of LOF, LB-ABOD, FastABOD and ABOD for 4 sizes of databases.

4.3 Efficiency

In this section, we compare the cpu time of each algorithm for selecting the n best outliers. Therefore, we perform experiments on four different database sizes: 500, 1000, 5000 and 10000. For this experiment, we compared ABOD, with and without the filter refinement approach LB-ABOD, to FastABOD and LOF on a data set of 25 dimensions. The sample size for FastABOD as well as the sample size for LB-ABOD were selected to be $0.1 \cdot dbsize$.

The results are illustrated in Figure 6. Let us note that we used a logarithmic time scale in this figure because even the fastest method in the experiment, LOF, has a quadratic and therefore, super linear time complexity. As can be seen LOF ran significantly faster on all four data sets as expected. However, we can also observe that both accelerated angle based methods, FastABOD and LB-ABOD, have a comparable increase in runtime between the data sets. This underlines that both methods have also a quadratic runtime complexity. Due to the refinement step in LB-ABOD and the usually larger sample set in FastABOD, both methods need additional time to find a ranking.

Considering the runtime of an naive application of ABOD, we observe that the cubic runtime is a large restriction when directly applying ABOD to larger data sets. For example, the straight forward calculation of ABOD on a data set of 5000 data objects takes more than eight hours. Let us note that the reason for the missing value for ABOD on the data set containing 10,000 objects is that the algorithm did not finish its computations after 24 hours and, thus, we cannot name a valid runtime. Therefore, to use ABOD with data sets having larger cardinalities, it is mandatory to employ

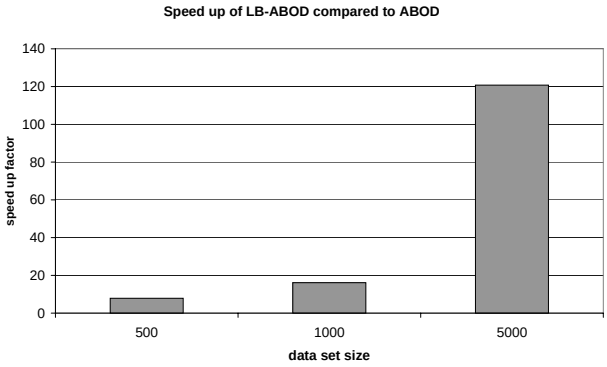


Figure 7: Speed-up factor of LB-ABOD compared to straight forward computation (ABOD).

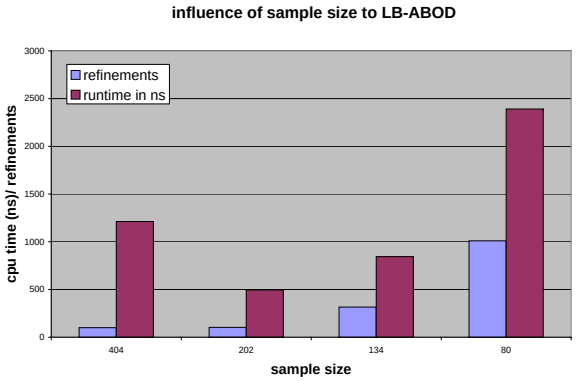


Figure 8: Influence of the sample size to runtime and number of refinements for LB-ABOD.

LB-ABOD. To further underline the increased efficiency in calculating the ABOF using LB-ABOD, Figure 7 displays the speed-up factor of LB-ABOD compared to a straight forward computation. For the data set containing 5000 data points, LB-ABOD computed the ranking of the top 100 outliers up to 120 times faster than ABOD. Let us note again that the only difference in the result is that LB-ABOD only provides the l top ranked results instead of ordering the complete data set with respect to the ABOF.

Our next experiment examines the influence of the sample size being considered in the filter step of LB-ABOD. Therefore, we compared the runtime and the number of refined candidates for 4 different sample sizes on the 50 dimensional data set containing 1000 data points. The results can be seen in Figure 8 displaying the complete cpu time for finding the top 100 outliers. Additionally, the number of refined candidates is shown for each sample size. The results indicate that if the size of the sample is selected too small (in this case 80) the number of refined candidates comprises the complete data set leading to a cubic runtime. On the other hand, selecting the size of the sample set too large might cause only a small reduction of candidates but also increases the computation time of the filter step. Having a properly selected sample size, the filter step efficiently ranks candidates and the refinement step has to examine only a small portion of the data set. In the majority of our experiments, we usually selected the sample size to be $0.1 \cdot dbsize$

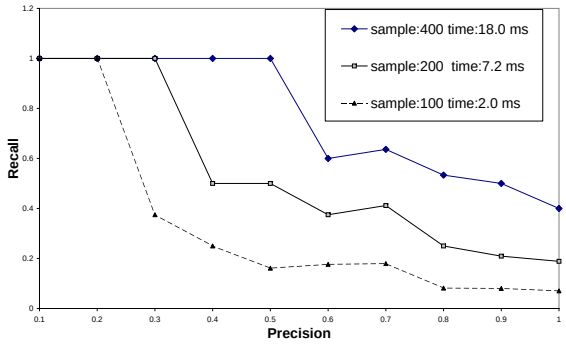


Figure 9: Influence of the sample size to runtime, precision and recall of FastABOD.

which led to the already displayed speed-up factors. Let us note that for larger data sets smaller sample sizes often offered a comparable performance.

In a final experiment on artificial data sets, we demonstrate the effect of the sample size on the simple approximation method FastABOD. We ran FastABOD with 3 different sample sizes, 100, 200 and 400 data points, on the 50 dimensional data set containing 1000 data points. Figure 9 displays 3 curves. Each curve is labeled by the used sample size and the cpu time in ms that was measured for FastABOD in this experiment. As expected there is a clear dependency between the sample size and the quality of the results; the larger the sample is the better is usually the ranking. Furthermore, we can additionally see that the size of the sample significantly increases the runtime. Thus, FastABOD runs 9 times faster, i.e. 2 ms instead of 18 ms, when having only 100 sample points instead of 400. To conclude, for high dimensional data FastABOD seems to be rather unsuitable due to its dependency on the distance based concept of nearest neighbors. Additionally, the quality of the outlier ranking provided by FastABOD is strongly dependent on a large enough sample size because unlike LB-ABOD the method does not correct its approximative ranking in a refinement step. Furthermore, a too large sample size leads to a strong increase of the computation time. However, for smaller dimensionalities FastABOD offers a simple way to efficient angle based outlier detection.

Let us finally remark that the effect of the dimensionality to the runtime of all four algorithms is negligible because all of the compared methods need to store distances or scalar product values, respectively, in order to assure a fast computation.

4.4 Real World Data

In this section, we provide experiments on real world data in order to demonstrate that there is some semantic meaning behind the proposed outlier ranking. In our first experiment, we employ the caltech 101 data set [10]. The data set consists of 101 classes comprising 8674 images. For each image the object of interest is annotated by a polygon comprising its borders. Based on these outline, we built a 200 dimensional 2D shape descriptor describing the border of each object. Thus, in our experiments, we want to extract the 10 most uncommon 2D shapes in the data set. We again compare the result of ABOD to LOF to have a reference outlier ranking. Figure 10 contains the top 5 ranked out-

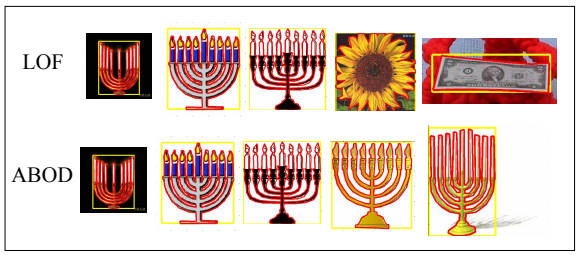


Figure 10: Top 5 ranked outliers by LOF and ABOD on the Caltech 101 image data set and a 2D shape representation.

liers by each method. Both methods decided that the top 3 outlier shapes in the data set belong to the same images of menorahs. However, while ABOD consistently ranked further menorah images as outliers with respect to their very special 2D shape, LOF started to rank much more common forms of dollar bills or sun flowers before ranking other menorahs.

In a final experiment, we tested the explanation component of ABOD on the zoo data set from the UCI machine learning repository [21] and received the following outliers for which we derived the following explanations by building the difference vector to the most similar other data object. RANK1: Scorpion. Its most similar animal in the data set is the termite. Thus, the scorpion is an outlier because it has 8 instead of 6 legs, it is venomous and does have a tail. RANK2: Octopus. The most similar animal is the cancer. However, the octopus has 2 more legs and is cat sized.

5. CONCLUSION

In this paper, we introduced a novel, parameter-free approach to outlier detection based on the variance of angles between pairs of data points. This idea alleviates the effects of the "curse of dimensionality" on mining high-dimensional data where distance-based approaches often fail to offer high quality results. In addition to the basic approach ABOD, we proposed two variants: FastABOD as an acceleration suitable for low-dimensional but big data sets, and LB-ABOD, a filter-refinement approach as an acceleration suitable also for high-dimensional data. In a thorough evaluation, we demonstrate the ability of our new approach to rank the best candidates for being an outlier with high precision and recall. Furthermore, the evaluation discusses efficiency issues and explains the influence of the sample size to the runtime of the introduced methods.

6. REFERENCES

[1] C. C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proc. ICDT*, 2001.

[2] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proc. SIGMOD*, 2001.

[3] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *Proc. PKDD*, 2002.

[4] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proc. KDD*, 1996.

[5] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley&Sons, 3rd edition, 1994.

[6] S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proc. KDD*, 2003.

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proc. ICDT*, 1999.

[8] M. M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proc. SIGMOD*, 2000.

[9] H. Fan, O. R. Zaïane, A. Foss, and J. Wu. A nonparametric outlier detection for efficiently discovering top-N outliers from engineering data. In *Proc. PAKDD*, 2006.

[10] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *IEEE. CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.

[11] T. G. Gärtner. A survey of kernels for structured data. *SIGKDD Explor. Newsl.*, 5(1):49–58, 2003.

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

[13] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proc. VLDB*, 2000.

[14] W. Jin, A. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proc. KDD*, 2001.

[15] W. Jin, A. K. H. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. In *Proc. PAKDD*, 2006.

[16] T. Johnson, I. Kwok, and R. Ng. Fast computation of 2-dimensional depth contours. In *Proc. KDD*, 1998.

[17] E. M. Knorr and R. T. Ng. A unified approach for mining outliers. In *Proc. CASCON*, 1997.

[18] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. VLDB*, 1998.

[19] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *Proc. VLDB*, 1999.

[20] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchthold. Efficient biased sampling for approximate clustering and outlier detection in large datasets. *IEEE TKDE*, 15(5):1170–1187, 2003.

[21] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.

[22] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proc. ICDE*, 2003.

[23] Y. Pei, O. Zaïane, and Y. Gao. An efficient reference-based approach to outlier detection in large datasets. In *Proc. ICDM*, 2006.

[24] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. SIGMOD*, 2000.

[25] P. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 1999.

[26] I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

[27] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proc. EDBT*, 1998.

[28] P. Sun and S. Chawla. On local spatial outliers. In *Proc. ICDM*, 2004.

[29] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Proc. PAKDD*, 2002.

[30] J. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[31] G. Williams, K. Yamanishi, and J. Takeuchi. Online unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proc. KDD*, 2000.

[32] K. Yamanishi and J. Takeuchi. Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner. In *Proc. KDD*, 2001.

[33] C. Zhu, H. Kitagawa, and C. Faloutsos. Example-based robust outlier detection in high dimensional datasets. In *Proc. ICDM*, 2005.