A Near-linear Time Approximation Algorithm for Angle-based Outlier Detection in High-dimensional Data

Ninh Pham
IT University of Copenhagen
Copenhagen, Denmark
ndap@itu.dk

Rasmus Pagh
IT University of Copenhagen
Copenhagen, Denmark
pagh@itu.dk

ABSTRACT

Outlier mining in d-dimensional point sets is a fundamental and well studied data mining task due to its variety of applications. Most such applications arise in high-dimensional domains. A bottleneck of existing approaches is that implicit or explicit assessments on concepts of distance or nearest neighbor are deteriorated in high-dimensional data. Following up on the work of Kriegel et al. (KDD '08), we investigate the use of *angle-based outlier factor* in mining high-dimensional outliers. While their algorithm runs in cubic time (with a quadratic time heuristic), we propose a novel random projection-based technique that is able to estimate the angle-based outlier factor for all data points in time near-linear in the size of the data. Also, our approach is suitable to be performed in parallel environment to achieve a parallel speedup. We introduce a theoretical analysis of the quality of approximation to guarantee the reliability of our estimation algorithm. The empirical experiments on synthetic and real world data sets demonstrate that our approach is efficient and scalable to very large high-dimensional data sets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms

Algorithms

Keywords

outlier detection, high-dimensional, angle-based, random projection, AMS Sketch

1. INTRODUCTION

Outlier mining is a fundamental and well studied data mining task due to the variety of domain applications, such as fraud detection for credit cards, intrusion detection in network traffic, and anomaly motion detection in surveillance video, etc. Detecting outliers is to identify the objects that considerably deviate from the general distribution of the data. Such the objects may be seen as suspicious objects due to the different mechanism of generation. For example, consider the problem of fraud detection for credit cards and the data set containing the card owners' transactions. The transaction records consist of usage profiles of each customer corresponding the purchasing behavior. The purchasing behavior of customer usually changes when the credit card is stolen. The abnormal purchasing patterns may be reflected in transaction records that contain high payments, high rate of purchase or the orders comprising large numbers of duplicate items, etc.

Most such applications arise in very high-dimensional domains. For instance, the credit card data set contains transaction records described by over 100 attributes [21]. To detect anomalous motion trajectories in surveillance videos, we have to deal with very high representational dimensionality of pixel features of sequential video frames [16]. Because of the notorious "curse of dimensionality", most proposed approaches so far which are explicitly or implicitly based on the assessment of differences in Euclidean distance metric between objects in full-dimensional space do not work efficiently. Traditional algorithms to detect distance-based outliers [13, 19] or density-based outliers [6, 18] suffer from the high computational complexity for high-dimensional nearest neighbor search. In addition, the higher the dimensionality is, the poorer the discrimination between the nearest and the farthest neighbor becomes [1]. That leads to a situation where most of the objects in the data set appear likely to be outliers based on the evaluation on their neighborhood using concepts like distance or nearest neighbor in high-dimensional space.

In KDD 2008, Kriegel et al. [14] proposed a novel outlier ranking approach based on the variance of the angles between an object and all other pairs of objects. This approach, named Angle-based Outlier Detection (ABOD), evaluates the degree of outlierness of each object on the assessment of the broadness of its angle spectrum. The smaller the angle spectrum of a object to other pairs of objects is, the more likely it is an outlier. Because "*angles are more stable than distances in high-dimensional space*" [15], this approach does not substantially deteriorate in high-dimensional data. In spite of many advantages of alleviating the effects of the "curse of dimensionality" and being a parameter-free measure, the time complexity taken to compute ABOD is significant with $O(dn^3)$ for a data set of n objects in d-dimensional

KDD'12, August 12–16, 2012, Beijing, China.

space. To avoid the cubic time complexity, the authors also proposed heuristic approximation variants of ABOD for efficient computations. These approximations, however, still rely on nearest neighbors and require high computational complexity with $O(dn^2)$ used in sequential search for neighbors. Moreover, there is no analysis to guarantee the accuracy of these approximations.

In this paper, we develop a near-linear time algorithm to approximate the variance of angles for each data object. Our proposed approach works in $O(n \log n(d + \log n))$ time for a data set of size n in d-dimensional space, and outputs an unbiased estimator of variance of angles for each object. The main technical insight is the combination between random hyperplane projections [11, 7] and AMS Sketch on product domains [12, 5], which enables us to reduce the computational complexity from cubic time complexity in the naïve approach to near-linear time complexity in the approximation solution. Another advantage of our algorithm is the suitability for parallel processing. In fact, we can achieve a nearly linear (in the number of processors used) parallel speedup of running time. We give a theoretical analysis of the quality of approximation to guarantee the reliability of our estimation algorithm. The empirical experiments on real world and synthetic data sets demonstrate that our approach is efficient and scalable to very large high-dimensional data.

The organization of the paper is as follows. In Section 2, we briefly review related work. The algorithm description including preliminaries and the proposed approach is presented in Section 3. Section 4 introduces the analysis of the accuracy of our approach. In Section 5, we show experimental evaluations of our proposed approach with synthetic and real world data sets. Finally, we make some conclusions about our work in Section 6.

2. RELATED WORK

A good outlier measure is the key aspect for achieving effectiveness and efficiency when managing the outlier mining tasks. A great number of outlier measures have been proposed, including global and local outlier models. Global outlier models typically take the complete database into account while local outlier models only consider a restricted surrounding neighborhood of each data object.

Knorr and Ng [13] proposed a simple and intuitive distance-based definition of outlier as an earliest global outlier model in the context of databases. The outliers with respect to parameter k and λ are the objects that have less than k neighbors within distance λ. A variant of the distance-based notion is proposed in [19]. This approach takes the distance of a object to its k^{th} nearest neighbor as its outlier score and retrieve the top m objects having the highest outlier scores as the top m outliers. The distance-based approaches are based on the assumption, that the lower density region that the data object is in, the more likely it is an outlier. The basic algorithm to detect such distance-based outliers is the nested loop algorithm [19] that simply computes the distance between each object and its k^{th} nearest neighbor and retrieve top m objects with the maximum k^{th} nearest neighbor distances. To avoid the quadratic worst case complexity of nested loop algorithm, several key optimizations are proposed in the literature. Such optimizations can be classified based on the different pruning strategies, such as the approximate nearest neighbor search [19], data partitioning strategies [19] and data ranking strategies [4, 10, 20]. Although these optimizations may improve performance, they scale poorly and are therefore inefficient as the dimensionality or the data size increases, and objects become increasingly sparse [2].

While global models take the complete database into account and detect outliers based on the distances to their neighbors, local density-based models evaluate the degree of outlierness of each object based on the local density of its neighborhood. In many applications, local outlier models give many advantages such as the ability to detect both global and local outliers with different densities and providing the boundary between normal and abnormal behaviors [6]. The approaches in this category assign to each object a local outlier factor as the outlierness degree based on the local density of its k-nearest neighbors [6] or the multi-granularity deviation of its ϵ-neighborhood [18]. In fact, these approaches implicitly rely on finding nearest neighbors for every object and typically use indexing data structures to improve the performance. Therefore, they are unsuitable for the requirements in mining high-dimensional outliers.

Due to the fact that the measures like distance or nearest neighbor may not be qualitatively meaningful in high-dimensional space, recent approaches focus on subspace projections for outlier ranking [2, 17]. In other words, these approaches take a subset of attributes of objects as subspaces into account. However, these approaches suffer from the difficulty of choosing meaningful subspaces [2] or the exponential time complexity in the data dimensionality [17]. As mentioned above, Kriegel at al. [14] proposed a robust angle-based measure to detect high-dimensional outliers. This approach evaluates the degree of outlierness of each data object on the assessment of the variance of angles between itself and other pairs of objects. The smaller the variance of angles between a object to the residual objects is, the more likely it is outlier. Because the angle spectrum between objects is more stable than distances as the dimensionality increases [15], this measure does not substantially deteriorate in high-dimensional data. However, the naïve and approximation approaches suffer from the high computational complexity with cubic time and quadratic time, respectively.

3. ALGORITHM DESCRIPTION

3.1 Angle-based outlier detection (ABOD)

As elaborated above, using concepts like distance or nearest neighbor for mining outlier patterns in high-dimensional data is unsuitable. A novel approach based on the variance of angles between pairs of data points is proposed to alleviate the effects of "curse of dimensionality" [14]. Figure 1 shows the variance of angles for the three kinds of points. Notice that the border and inner points of the cluster have very large variance of angles whereas this value is much smaller for the outliers. In other words, the smaller the angle variance of a point to the residual points is, the more likely it is an outlier. This is because the points inside the cluster are surrounded by other points in all possible directions while the points outside the cluster are positioned in particular directions. Therefore, we use the variance of angles (VOA) as the outlier factor to evaluate the degree of outlierness of each point of the data set. The proposed approaches in [14] do not deal directly with the variance of angles but variance of cosine of angles weighted by the corresponding distances of the points instead. We argue that the weighting factors are

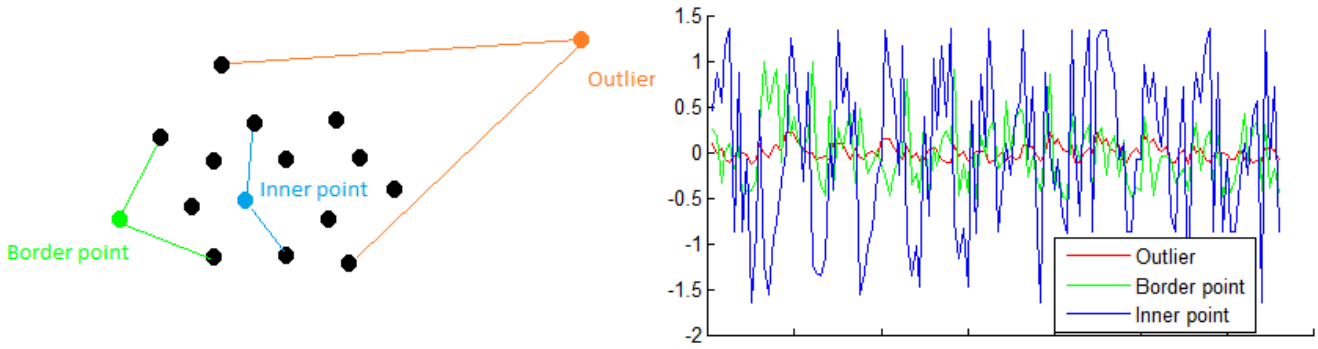

Figure 1: The variance of angles for different kinds of points.

less and less meaningful in high-dimensional data due to the "curse of dimensionality". We expect the outlier rankings based on the variance of cosine spectrum with or without weighting factors and the variance of angle spectrum are likely similar in high-dimensional data. We therefore formulate the angle-based outlier factor using the variance of angles as follows:

DEFINITION 1. *Given a point set $S \subseteq \mathbf{R}^d$, $|S| = n$ and a point $p \in S$. For a random pair of different points $a, b \in S \setminus \{p\}$, let Θ_{apb} denote the angle between the difference vectors $a - p$ and $b - p$. The angle-based outlier factor $VOA(p)$ is the variance of Θ_{apb}:*

$$VOA(p) = \mathbf{Var}[\Theta_{apb}] = MOA_2(p) - (MOA_1(p))^2$$

where MOA_2 and MOA_1 are defined as follows:

$$MOA_2(p) = \frac{\sum_{\substack{a,b \in S\setminus\{p\} \\ a \neq b}} \Theta_{apb}^2}{\frac{1}{2}(n-1)(n-2)}; MOA_1(p) = \frac{\sum_{\substack{a,b \in S\setminus\{p\} \\ a \neq b}} \Theta_{apb}}{\frac{1}{2}(n-1)(n-2)}$$

It is obvious that the VOA measure is entirely free of parameters and therefore is suitable for unsupervised outlier detection methods. The naïve ABOD algorithm computes the VOA for each point of the data set and return the top m points having the smallest VOA as outliers. However, the time complexity of the naïve algorithm is in $O(dn^3)$. The cubic computational complexity means that it will be very difficult to mine outliers in very large data sets.

3.2 Technical Overview

The general idea of our approach is to efficiently compute an unbiased estimator of the variance of the angles for each point of the data set. In other words, the expected value of our estimate is equal to the variance of angles and we show that it is concentrated around its expected value. These estimated values are then used to rank the points. The top m points having the smallest variances of angles are retrieved as top m outliers of the data set.

In order to estimate the variance of angles between a point and all other pairs of points, we first project the data set on the hyperplanes orthogonal to random vectors whose coordinates are independently chosen from the standard normal distribution $\mathbf{N}(0, 1)$. Based on the partitions of the data set after projection, we are able to estimate the unbiased mean of angles for each point. We then approximate the second moment and derive its variance by using the AMS Sketches to summarize the frequency moments of the points projected on the random hyperplanes. The combination between random hyperplane projections and AMS Sketches on product domains enables us to reduce the computational complexity to $O(n \log n(d + \log n))$ time. In the following we start with some basic notions of random hyperplane projection and AMS Sketch, then propose our approach to estimate the variance of angles for each point of the data set.

3.3 Preliminaries

3.3.1 Random Hyperplane Projection

Following Charikar [7], we take random vectors r_1,..., r_t $\in \mathbf{R}^d$ such that each coordinate is chosen independently from the standard normal distribution $\mathbf{N}(0, 1)$.

For $i = 1, \ldots, t$ and points $a, b, p \in S$ consider the independent random variables

$$X_{apb}^{(i)} = \begin{cases} 1 & \text{if } a \cdot r_i < p \cdot r_i < b \cdot r_i \\ 0 & \text{otherwise} \end{cases}$$

For a vector r_i we see that $X_{apb}^{(i)} = 1$ only if the vectors $a - p$ and $b - p$ are on different sides of the hyperplane orthogonal to r_i, and in addition $(a-p) \cdot r_i < 0$. The probability that this happens is proportional to Θ_{apb}, as exploited in the seminal papers of Goemans and Williamson [11] and Charikar [7]. More precisely we have:

LEMMA 2. *For all a, b, p, i, $\Pr[X_{apb}^{(i)} = 1] = \Theta_{apb}/(2\pi)$.*

Note that we also have $\Pr[X_{bpa}^{(i)} = 1] = \Theta_{apb}/(2\pi)$ due to symmetry [11]. By using t random vectors r_i, we are able to boost the accuracy of the estimator of Θ_{apb}. In particular, we have $\Theta_{apb} = \frac{2\pi}{t} \sum_{i=1}^{t} X_{apb}^{(i)}$. The analysis of accuracy for random projections will be presented in the Section 4.

3.3.2 AMS Sketch

Alon at al. [3] described and analyzed a sketching approach, called AMS Sketch, to estimate the second frequency moment of a high-dimensional vector.

LEMMA 3. *Given a high-dimensional vector $w_1, \cdots, w_q$, take a 4-wise independent vector $s \in \{\pm 1\}^q$. The AMS Sketch is the value $\mathbf{Z} = \sum_{i=1}^{q} s_i w_i$. Define $\mathbf{Y} = \mathbf{Z}^2$ then $\mathbf{E}[\mathbf{Y}] = \sum_{i=1}^{q} w_i^2$ and $\mathbf{Var}[\mathbf{Y}] \leq 2\,(\mathbf{E}[\mathbf{Y}])^2$.*

Recently, Indyk and McGregor [12], and Braverman et al. [5] have considered AMS Sketches with two different 4-wise independent vectors for outer product. In this case, we view the matrix as vector of matrix elements.

LEMMA 4. *Given two different 4-wise independent vectors* $s^1, s^2 \in \{\pm 1\}^q$. *The AMS Sketch of an outer product* (uv), *where by definition* $(uv)_{ij} = u_i v_j$, *is:*

$$\mathbf{Z} = \sum_{(i,j)\in[q]\times[q]} s_i^1 s_j^2 (uv)_{ij} = \left(\sum_{i=1}^{q} s_i^1 u_i\right)\left(\sum_{j=1}^{q} s_j^2 v_j\right)$$

Define $\mathbf{Y} = \mathbf{Z}^2$ *then* $\mathbf{E}[\mathbf{Y}] = \sum_{ij}(u_i v_j)^2$ *or squared Frobenius norm of the outer product* (uv) *and* $\mathbf{Var}[\mathbf{Y}] \le 8(\mathbf{E}[\mathbf{Y}])^2$.

That is, the AMS sketch of the outer product is simply the product of the AMS sketches of the two vectors (using different 4-wise independent random vectors).

3.4 Approximate ABOD

To avoid the cubic time complexity, we propose a near-linear time algorithm to estimate the variance of angles for each data point based on random hyperplane projections.

3.4.1 First Moment Estimator

Given a random vector r_i and a point $p \in S$, we estimate $MOA_1(p)$ using Lemma 2 as follows:

$$\begin{aligned} F_1(p) &= \tfrac{2}{(n-1)(n-2)}\left(2\pi \sum_{\substack{a,b\in S\setminus\{p\}\\ a\neq b}} \mathbf{E}[X_{apb}^{(i)}]\right) \\ &= \tfrac{2\pi}{(n-1)(n-2)} \sum_{\substack{a,b\in S\setminus\{p\}\\ a\neq b}} \left(\mathbf{E}[X_{apb}^{(i)}] + \mathbf{E}[X_{bpa}^{(i)}]\right) \\ &= \tfrac{2\pi}{(n-1)(n-2)} |\mathbf{L}_p^{(i)}||\mathbf{R}_p^{(i)}| \end{aligned}$$

where the sets $\mathbf{L}_p^{(i)} = \{x \in S\setminus\{p\} \mid x \cdot r_i < p \cdot r_i\}$ and $\mathbf{R}_p^{(i)} = \{x \in S\setminus\{p\} \mid x \cdot r_i > p \cdot r_i\}$ consist of the points on each side of p under the random projection.

Note that this value is an unbiased estimator of mean of angles between the point p and the other pairs of points. We boost the accuracy of the estimation by using t random projections. We therefore have the more accurate unbiased estimator of $MOA_1(p)$:

$$F_1(p) = \tfrac{2\pi}{t(n-1)(n-2)} \sum_{i=1}^{t} |\mathbf{L}_p^{(i)}||\mathbf{R}_p^{(i)}| \qquad (1)$$

3.4.2 Second Moment Estimator

Since estimation of the second moment is more complicated, we first present the general idea by considering a less efficient approach and then propose an efficient algorithm to compute the unbiased second moment estimator. Focus on a single point p, suppose that we fix an arbitrary ordering of the set $S\setminus\{p\}$ as $x_1, x_2, \cdots, x_{n-1}$. For each projection using the random vector r_i, we take the two vectors $u_i, v_i \in \{0,1\}^{n-1}$ such that their k^{th} coordinate corresponds to the k^{th} point of the set $S\setminus\{p\}$. The k^{th} coordinate of u_i (or v_i) is 1 if the k^{th} point of the set locates on the left (or right) partition, and 0, otherwise. We consider the matrix $\mathbf{P} = \sum_{i=1}^{t}(u_i v_i)$ where $(u_i v_i)$ is the outer product of u_i and v_i. Note that all diagonal elements of $\mathbf{P}$ are 0. Consider any pair of points $a, b \in S\setminus\{p\}$ where $a = x_i$ and $b = x_j$, it is clear that $\mathbf{P}_{ij}$ is the number of times that a locates on the left side and b locates on the right side after t projections. We can therefore estimate Θ_{apb}^2, the squared angle between p and a, b based on the element $\mathbf{P}_{ij}$ of the matrix $\mathbf{P}$.

$$\begin{aligned} \mathbf{P}_{ij}^2 &= \left(\sum_{i=1}^{t} X_{apb}^{(i)}\right)^2 = \sum_{i=1}^{t}\left(X_{apb}^{(i)}\right)^2 + 2\sum_{\substack{i,j=1\\ i\neq j}}^{t} X_{apb}^{(i)} X_{apb}^{(j)} \\ \mathbf{E}[\mathbf{P}_{ij}^2] &= \sum_{i=1}^{t} \mathbf{E}[(X_{apb}^{(i)})^2] + 2\sum_{\substack{i,j=1\\ i\neq j}}^{t} \mathbf{E}[X_{apb}^{(i)}]\mathbf{E}[X_{apb}^{(j)}] \\ &= t\tfrac{\Theta_{apb}}{2\pi} + t(t-1)\left(\tfrac{\Theta_{apb}}{2\pi}\right)^2 \end{aligned}$$

So we have the unbiased estimator:

$$\Theta_{apb}^2 = \tfrac{(2\pi)^2}{t(t-1)}\left(\mathbf{E}[\mathbf{P}_{ij}^2] - \tfrac{t}{2\pi}\Theta_{apb}\right)$$

Therefore, we can compute $MOA_2(p)$ based on all elements of $\mathbf{P}$ as follows:

$$\begin{aligned} MOA_2(p) &= \tfrac{2}{(n-1)(n-2)} \sum_{\substack{a,b\in S\setminus\{p\}\\ a\neq b}} \Theta_{apb}^2 \\ &= \tfrac{1}{(n-1)(n-2)} \sum_{\substack{a,b\in S\setminus\{p\}\\ a\neq b}} \left(\Theta_{apb}^2 + \Theta_{bpa}^2\right) \\ &= \tfrac{4\pi^2}{t(t-1)(n-1)(n-2)}\left(\sum_{i=1}^{n-1}\sum_{j=1}^{n-1} \mathbf{E}[\mathbf{P}_{ij}^2] - \tfrac{t}{\pi} \sum_{\substack{a,b\in S\setminus\{p\}\\ a\neq b}} \Theta_{apb}\right) \\ &= \tfrac{4\pi^2}{t(t-1)(n-1)(n-2)}\left(\mathbf{E}[\|\mathbf{P}\|_F^2] - \tfrac{t(n-1)(n-2)}{2\pi} MOA_1(p)\right) \\ &= \tfrac{4\pi^2}{t(t-1)(n-1)(n-2)}\mathbf{E}[\|\mathbf{P}\|_F^2] - \tfrac{2\pi}{t-1} MOA_1(p) \end{aligned}$$

From the equation above, we can estimate $MOA_2(p)$:

$$F_2'(p) = \tfrac{4\pi^2}{t(t-1)(n-1)(n-2)} \|\mathbf{P}\|_F^2 - \tfrac{2\pi}{t-1} F_1(p) \qquad (2)$$

However, the squared Frobenius norm $\|\mathbf{P}\|_F^2$ will not be computed exactly, since we do not know how to achieve this in less than quadratic time. Instead, it will be estimated using AMS Sketches on product domains. Let $AMS(\mathbf{L}_p^{(i)})$ and $AMS(\mathbf{R}_p^{(i)})$ be the AMS Sketches of the vectors u_i and v_i (using different 4-wise independent random vectors), respectively. Due to linearity the sketch of sum of distributions is equal to the sum of sketches of the distributions, so:

$$\|\mathbf{P}\|_F^2 = \left(\sum_{i=1}^{t} AMS(\mathbf{L}_p^{(i)}) AMS(\mathbf{R}_p^{(i)})\right)^2$$

We therefore derive the second moment estimator $F_2(p)$:

$$F_2(p) = \tfrac{4\pi^2\left(\sum_{i=1}^{t} AMS(\mathbf{L}_p^{(i)}) AMS(\mathbf{R}_p^{(i)})\right)^2}{t(t-1)(n-1)(n-2)} - \tfrac{2\pi F_1(p)}{t-1} \qquad (3)$$

3.4.3 Algorithm

Based on the estimators of $MOA_1(p)$, and $MOA_2(p)$ for any point p described above, we introduce FastVOA, a near-linear time algorithm to estimate the variance of angles for all points of the data set. The pseudo-code in Algorithm 1 shows how FastVOA works.

At first, we project the data set S on the hyperplanes orthogonal to random projection vectors (Algorithm 2). *RandomProjection()* returns a data structure $\mathcal{L}$ containing the

Algorithm 1 FastVOA(S, t, s_1, s_2)

Ensure: Return the variance estimator for all points

1: $\mathcal{L} \leftarrow RandomProjection(S, t)$
2: $F1 \leftarrow FirstMomentEstimator(\mathcal{L}, t, n)$
3: **for** $i = 1 \rightarrow s_2$ **do**
4: $\quad \mathbf{Y}_i \leftarrow \sum_{j=1}^{s_1} \left(FrobeniusNorm(\mathcal{L}, t, n)\right)^2 / s_1$
5: **end for**
6: $F2 \leftarrow median\left\{\mathbf{Y}_1, \cdots, \mathbf{Y}_{s_2}\right\}$
7: $\mathbf{Var} \leftarrow [0]^n$
8: **for** $j = 1 \rightarrow n$ **do**
9: $\quad F2[j] = \frac{4\pi^2}{t(t-1)(n-1)(n-2)} F2[j] - \frac{2\pi F1[j]}{t-1}$
10: $\quad \mathbf{Var}[j] = F2[j] - (F1[j])^2$
11: **end for**
12: **return** $\mathbf{Var}$

Algorithm 2 RandomProjection(S, t)

Ensure: Return a list $\mathcal{L} = L_1 L_2 \cdots L_t$ where L_i is a list of points ordered by their dot product with r_i

1: $\mathcal{L} \leftarrow \emptyset$
2: **for** $i = 1 \rightarrow t$ **do**
3: $\quad$ Generate a random vector r_i whose coordinates are independently chosen from $\mathbf{N}(0, 1)$
4: $\quad$ Let L_i be an empty list containing pairs of point ID and its dot product with r_i
5: $\quad$ **for** $j = 1 \rightarrow n$ **do**
6: $\quad\quad$ Insert $(x_j, x_j \cdot r_i)$ into the list L_i
7: $\quad$ **end for**
8: $\quad$ Sort L_i based on the dot product order
9: $\quad$ Insert L_i into $\mathcal{L}$
10: **end for**
11: **return** $\mathcal{L}$

information of the partitions of S under t random projections. Using $\mathcal{L}$, we are able to efficiently identify the values $|L_p^{(i)}|$ and $|R_p^{(i)}|$ corresponding to each point p and r_i. The pseudo-code in Algorithm 3 computes the first moment estimator for each point. Similarly, we also make use of $\mathcal{L}$ to compute the Frobenius norm $\|\mathbf{P}\|_F$ for each point p in Algorithm 4. To boost the accuracy of the AMS Sketches, we have to repeat the computation of *FrobeniusNorm()* $s_1 s_2$ times, and output $F2$ as the median of s_2 random variables $\mathbf{Y}_1, \cdots, \mathbf{Y}_{s_2}$, each being the average of s_1 values (lines 3 - 6). After that, the second moment estimator and variance value for each point are computed in lines 9 - 10.

3.4.4 Computational Complexity and Parallelization

It is clear that the computational complexity of FastVOA depends on Algorithms 2 - 4. We note that Algorithm 2 takes $O(tn(d + \log n))$ time in computing dot products and sorting for all points while both Algorithm 3 and 4 run in $O(tn)$ time. Since we have to repeat the Algorithm 4 in $s_1 s_2$ times, the total running time is $O(tn(d + \log n + s_1 s_2))$. To guarantee the accuracy of FastVOA, we have to choose $t = O(\log n)$ and $s_1 s_2$ sufficiently large to boost the accuracy of estimation as analyzed later in Section 4. Therefore, the running time is dominated by the AMS Sketch computational time. That means FastVOA runs in $O(s_1 s_2 n \log n)$ time.

It is worth noting that Algorithms 2 - 4 use the **for** loop with t random vectors that performs the same independent operations for each random vector. Therefore, we can simply parallelize this loop in these three algorithms to achieve a nearly linear (in the number of processors used) speedup.

Algorithm 3 FirstMomentEstimator($\mathcal{L}, t, n$)

Ensure: Return the first moment estimator for all points

1: $F1 \leftarrow [0]^n$
2: **for** $i = 1 \rightarrow t$ **do**
3: $\quad C_l \leftarrow [0]^n, C_r \leftarrow [0]^n$
4: $\quad L_i \leftarrow \mathcal{L}[i]$
5: $\quad$ **for** $j = 1 \rightarrow n$ **do**
6: $\quad\quad idx = L_i[j].pointID$
7: $\quad\quad C_l[idx] = j - 1$
8: $\quad\quad C_r[idx] = n - 1 - C_l[idx]$
9: $\quad$ **end for**
10: $\quad$ **for** $j = 1 \rightarrow n$ **do**
11: $\quad\quad F1[j] = F1[j] + C_l[j] C_r[j]$
12: $\quad$ **end for**
13: **end for**
14: **return** $\frac{2\pi}{t(n-1)(n-2)} F1$

Algorithm 4 FrobeniusNorm($\mathcal{L}, t, n$)

Ensure: Return $\|\mathbf{P}\|_F$ for each point p

1: $F2 \leftarrow [0]^n$
2: Initialize 4-wise independent vectors $S_l[n]$, $S_r[n]$ whose entries are in $\{\pm 1\}$ with equal probability
3: **for** $i = 1 \rightarrow t$ **do**
4: $\quad AMS_l \leftarrow [0]^n, AMS_r \leftarrow [0]^n$
5: $\quad L_i \leftarrow \mathcal{L}[i]$
6: $\quad$ **for** $j = 2 \rightarrow n$ **do**
7: $\quad\quad idx1 = L_i[j].pointID$
8: $\quad\quad idx2 = L_i[j-1].pointID$
9: $\quad\quad AMS_l[idx1] = AMS_l[idx2] + S_l[idx2]$
10: $\quad$ **end for**
11: $\quad$ **for** $j = n - 1 \rightarrow 1$ **do**
12: $\quad\quad idx1 = L_i[j].pointID$
13: $\quad\quad idx2 = L_i[j+1].pointID$
14: $\quad\quad AMS_r[idx1] = AMS_r[idx2] + S_r[idx2]$
15: $\quad$ **end for**
16: $\quad$ **for** $j = 1 \rightarrow n$ **do**
17: $\quad\quad F2[j] = F2[j] + AMS_l[j] AMS_r[j]$
18: $\quad$ **end for**
19: **end for**
20: **return** $F2$

4. ERROR ANALYSIS

It has already been argued that our estimators are unbiased, i.e., produce the right first and second moments in expectation: $\mathbf{E}[F_1(p)] = MOA_1(p)$ and $\mathbf{E}[F_2(p)] = MOA_2(p)$.

In this section we analyze the precision, showing bounds on the number of random projections and AMS sketches needed to achieve a given precision ε. This will imply that the variance is estimated within an additive error of $O(\varepsilon)$. For $MOA_1(p)$ we get this directly with high probability, whereas for $MOA_2(p)$ the basic success probability of the estimator $F_2(p)$ is only 3/4. However, by repeating the second moment estimation procedure $s_2 = O(\log(1/\delta))$ times and taking the median outlier score for each point, the success probability can be magnified to $1 - \delta$, for any $\delta > 0$ as argued in [3].

We will use the following version of the Chernoff bound from [8, Theorem 1.1]:

LEMMA 5. *Let $Y = \sum_{i=1}^{t} Y^{(i)}$ be a sum of independent random variables with values in $[0;1]$. For any $\Delta > 0$,*

$$\Pr[|Y - \mathbf{E}[Y]| > \Delta] \leq 2e^{-2\Delta^2/t} \ .$$

First moment estimator.

Consider the probability (over choice of vectors r_1,...,r_t) that $F_1(p)$ deviates from $MOA_1(p)$ by more than ε. Splitting the sum $F_1(p)t/\pi$ into t terms we can apply Lemma 5. Then we get that its deviation from the mean exceeds $\varepsilon t/\pi$ with probability at most $2e^{-2(\varepsilon t/\pi)^2/t}$. If we choose $t > \varepsilon^{-2}\pi^2 \ln(n)$ this probability is at most $2/n^2$. So the probability that all of n first moment estimators have error at most ε is $1 - O(1/n)$.

Second moment estimator.

Next, consider the probability (again over choice of vectors $r_1, \ldots, r_t$) that the first version of the second moment estimator, $F_2'(p)$, deviates from its expectation by more than ε, given that $F_1(p)$ deviates by at most ε from its expectation. Looking at (2) we see that this happens when $||\mathbf{P}||_F^2$ deviates from its expectation by at least $\binom{n-1}{2}\binom{t}{2}\varepsilon/\pi^2$. Thus, it suffices to show that each squared entry $\mathbf{P}_{ij}^2$ deviates by at most $\frac{1}{4}\varepsilon t^2/\pi^2$ from its expectation with high probability. Recall that $\mathbf{P}_{ij} = \sum_{k=1}^{t} X_{x_i,x_j}^{(k)}$, a sum of independent indicator random variables, which means that Lemma 5 applies. For $t > 16\varepsilon^{-2}\pi^4 \ln(n)$ we get that $\mathbf{P}_{ij}$ deviates from its expectation by at most $\frac{1}{4}\varepsilon t/\pi^2$ with probability $1 - O(1/n^2)$. Since $\mathbf{P}_{ij} \leq t$ this implies that $\mathbf{P}_{ij}^2$ deviates by at most $\frac{1}{4}\varepsilon t^2/\pi^2$, as desired. The total error for $F_2'(p)$, accounting for all $2\binom{n-1}{2}$ entries of $\mathbf{P}$, is therefore bounded by ε with probability $1 - O(1/n^2)$.

Finally, we should account for the error caused by the use of AMS sketches in the final estimator $F_2(p)$, equation (3). By Lemma 4 the variance of the estimator is bounded by $8MOA_2(p)^2$. Taking the average of s_1 sketches the variance is reduced to at most $\frac{8MOA_2(p)^2}{s_1}$. By applying Chebychev's inequality, the probability that $F_2(p)$ deviates by $\frac{\varepsilon}{\pi^2}MOA_2(p)$ from its expectation of $MOA_2(p)$ is at most:

$$\frac{8MOA_2(p)^2/s_1}{(MOA_2(p)\varepsilon/(\pi^2))^2} = \frac{8\pi^4}{s_1\varepsilon^2} \ .$$

For $s_1 > 32\pi^4/\varepsilon^2$ this is less than $1/4$. It is simple to verify that this deviation corresponds to a deviation for $F_2(p)$ of 2ε. As stated above, the failure probability is reduced exponentially by repeating the estimation s_2 times.

5. EXPERIMENTS

We implemented all algorithms in C++ and conducted experiments in a 2.67 GHz core i7 Windows platform with 3GB of RAM on both synthetic and real world data sets.

5.1 Data sets

For the sake of fair comparison, we made use of the same synthetic data generation process as ABOD approaches [14]. We generated a Gaussian mixture including 5 equally weighted clusters having random means and variances as normal points and employed a uniform distribution in full-dimensional space as the outliers. For each synthetic data set, we generated 10 outliers which are independent on the Gaussian mixture. We evaluated the performance of all algorithms on synthetic data sets with varying sizes and dimensions.

For the real world high-dimensional data sets, we picked three data sets (Isolet, Multiple Features and Optical Digits) designed for classification and machine learning tasks from UCI machine learning repository [9]. Isolet contains the pronunciation data of 26 letters of the alphabet while Multiple Features and Optical Digits consist of the data of handwritten numerals ('0' - '9'). For each data set, we picked all data points from some classes having common behaviors as normal points and 10 data points from another class as outliers. For instance, we picked points of classes C, D, and E of Isolet that share the "e" sound as normal points and 10 points from class Y as outliers. Similarly, we picked points of classes 6 and 9 of Multiple Features, classes 3 and 9 of Optical Digits as normal points because of the similar shapes and 10 points of class 0 as outliers. It is worth noting that there are some outliers that probably locate on the region covered by inliers. Therefore, we are not able to isolate exactly all outliers. Instead, we expect our algorithms to rank all outliers into sufficiently high positions.

5.2 Accuracy of Estimation

This subsection presents the accuracy experiments to evaluate the reliability of our estimation algorithm. As analysis in the Section 4, the estimators $F_1(p)$, $F_2'(p)$, and $F_2(p)$ for any point p of the data set can deviate from their expectations by more than ϵ with probability at most δ by using a sufficiently large number of random projections $t = O(\log n)$ and AMS Sketches $s_1 s_2$. Note that $F_2(p)$ is the second moment estimator using AMS Sketch while $F_2'(p)$ is based on only random projections. At first, we carried out experiments to measure the accuracy of estimators based on only random projections. We measured the deviation error ϵ of $F_1(p)$ and $F_2'(p)$ from their expectations with error probability $\delta = 0.1$. We took t in ranges [100, 1,000] and conducted experiments on 2 synthetic data sets having 1,000 points on 50 and 100 dimensions, namely Syn50 and Syn100, as well as the three real world data sets, namely Isolet, Mfeat and Digit. Figures 2.a and 2.b display the deviation error (ϵ) from expectation of the estimators $F_1(p)$ and $F_2'(p)$ with error probability $\delta = 0.1$. Using these two estimators, we derived the variance estimator and measured its deviation from expectation with $\delta = 0.1$, as shown in Figure 2.c. Although the theoretical analysis requires a sufficiently large number of random projections t to achieve the small ϵ, the results on 5 data sets surprisingly show that with a rather small t, we are able to estimate exactly the variance of angles for all points. With $t = 600$, 90% number of points of 5 data sets have the first moment, the second moment and the derived variance estimators deviating from their expectations at most 0.035, 0.08 and 0.015 respectively. When t increases to 1,000, 90% of points of 5 data sets have the variance estimator deviate from its expectation by at most 0.01. Therefore, for such data sets having large difference between VOA of outliers and VOA of border points, the use of random projections to estimate VOA can achieve good performance on detecting outliers.

To quantify the error due to the AMS Sketches, we measured the error probability δ of the variance estimator us-

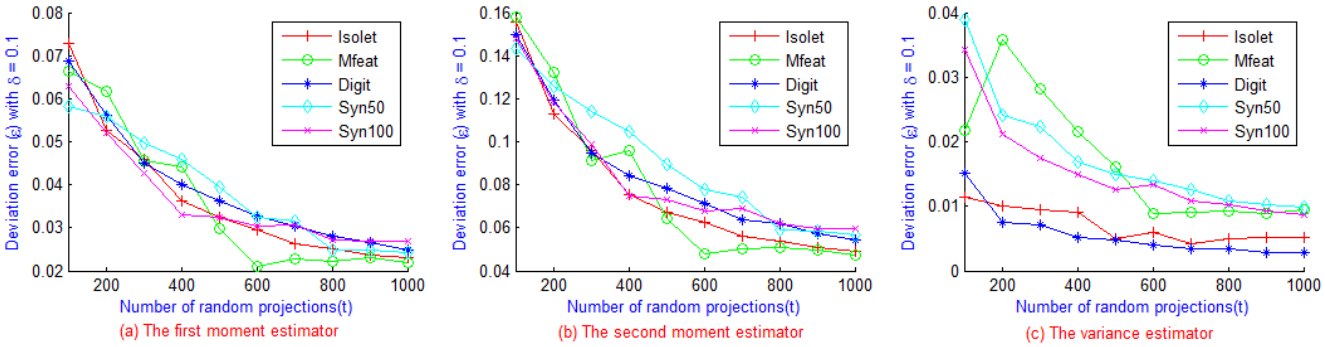

Figure 2: Deviation error of random projection estimators on 5 data sets.

ing AMS Sketches and fixing parameters $t = 1,000, s_1 = 7,200, s_2 = 50, \epsilon = 0.1$ on all data sets. Concretely, we computed the number of points p of the data set such that its variance estimator by using AMS Sketch deviates by more than $\epsilon VOA(p)$ from its expectation $VOA(p)$. Table 1 presents the error probability of variance estimators on 5 data sets.

Table 1: Error probability of variance estimator using AMS Sketch on 5 data sets

Isolet	Mfeat	Digit	Syn50	Syn100
0.75	0.19	0.35	0.04	0.03

It is clear that the two synthetic data sets obtain very small errors while the real world data sets take rather large errors, especially on Isolet. This is because the variance estimator of all points of the data set may be underestimated or overestimated by using AMS Sketch. To guarantee the capability of our approximation approach on detecting outliers, we analyzed the accuracy of outlier ranking between the brute force algorithm called SimpleVOA and the approximate algorithm FastVOA. The accuracy of outlier ranking is defined as $\frac{|A \cap B|}{m}$ where A and B are the top m positions retrieved by SimpleVOA and FastVOA algorithms, respectively. Figure 3 shows the accuracy of outlier ranking between SimpleVOA and FastVOA where m is in ranges 10 - 100.

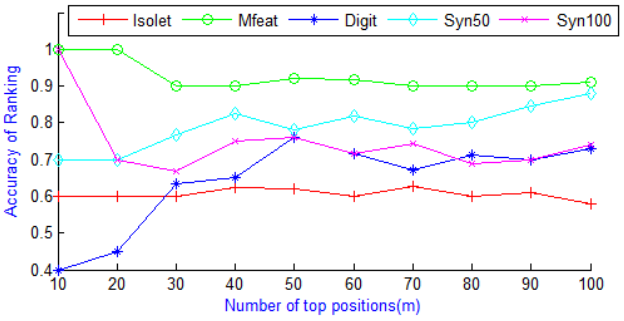

Figure 3: The accuracy of outlier ranking between SimpleVOA and FastVOA on 5 data sets.

The results of outlier ranking indicate that FastVOA provided a rather high accurate ranking on all data sets. While 2 synthetic data sets and Multiple Feature show a highly accurate ranking for all ranges of top positions, the other data sets offered a medium accurate ranking when $m < 30$ but more accurate when $m > 40$. Although the use of AMS Sketch may lead to underestimate or overestimate of the variance estimator, FastVOA still introduces good performance on ranking data points based on VOA.

5.3 Effectiveness

It is obvious that our approaches are dealing directly with the variance of angles (VOA) while the approaches in [14] compute the variance of cosine of angles weighted by distances (ABOF). This subsection demonstrates experiments to measure the effectiveness of both measures on detecting outliers. For each measure, we compared the quality of outlier ranking provided by brute force (SimpleVOA and ABOD) and approximation algorithms (FastVOA and FastABOD). For the sake of fair comparison, we used the precision-recall graph to evaluate the capability of each algorithm to retrieve the most likely outliers. The precision is the number of retrieved points that are indeed outliers. For each precision level, we measured the recall as the percentage of the number of outliers in the retrieved set.

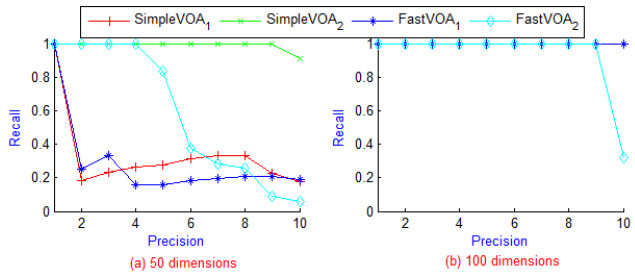

Figure 4: Precision-Recall Graph for 4 synthetic data sets. Each graph describes the behavior on 1,000 and 5,000 points.

For synthetic data sets, we generated 4 data sets with varying sizes of 1,000 and 5,000 points and dimensions of 50 and 100. We observed that the differences of VOA between outliers and border points on synthetic data sets become large when the size increases. Therefore, we adjusted the parameter settings for FastVOA on synthetic data sets of size 5,000 points to reduce the time complexity. In particular, we determined $t = 100$, $s_1 = 1600$, $s_2 = 10$. We

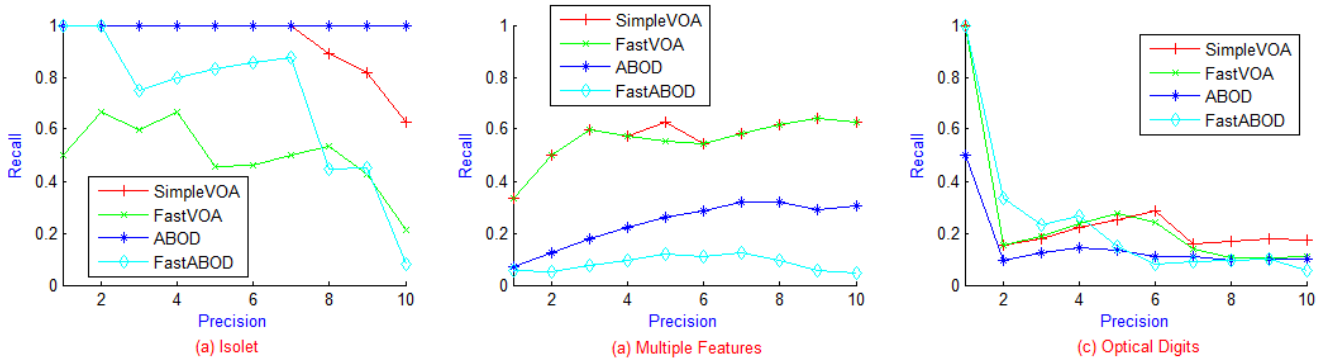

Figure 5: Precision-Recall Graph for 3 real world data sets.

kept the same parameter setting as Section 5.2 for the other data sets. The sample size of FastABOD is chosen as $0.1n$ as [14]. Let us note that both ABOD and FastABOD offered perfect results on 4 synthetic data sets. That means all 10 outliers were ranked into the top 10 positions. Therefore, we did not show the results of ABOD and FastABOD on synthetic data sets. Figure 4 depicts the precision-recall graph for synthetic data sets. Figure 4.a shows the results of brute force (SimpleVOA_1 and SimpleVOA_2) and approximation algorithms (FastVOA_1 and FastVOA_2) on the 2 data sets of 50 dimensions and varying sizes of 1,000 and 5,000 points. In the medium dimensionality of 50, VOA did not work well in the small data set size but achieved almost perfect performance in the large data set by ranking all 10 outliers between top 11 retrieved points. It is clear that the better performance of SimpleVOA leads to the better performance of FastVOA. Results of 2 synthetic data sets with 100 dimensions are displayed in Figure 4.b. Since the effect of weighting factors in ABOF is not meaningful in high-dimensional data, SimpleVOA and FastVOA show competitive results with ABOD and FastABOD with almost perfect performance.

Figure 5 shows the observed precision-recall graphs for 3 real world data sets. On Isolet, SimpleVOA and ABOD offered almost perfect performance by ranking 10 outliers in top 10 and top 16 positions, respectively. FastABOD provided better outlier ranking than FastVOA on detecting 7 outliers in top 10 positions. However, on ranking all 10 outliers, both of them did not work well for large recall levels. Both SimpleVOA and FastVOA performed rather well on Multiple Features by ranking all outliers on the top 16 positions while both ABOD and FastABOD performed very badly. All approaches had difficulties to detect outliers on Optical Digits. However, the VOA based approaches clearly offered better results than the ABOF based ones.

5.4 Efficiency

This section compares the running time of 3 algorithms, namely FastVOA, LB_ABOD and FastABOD on large high-dimensional data sets. In fact, there are very few large real world data sets where the outliers are identified exactly in advance. Therefore, we decided to evaluate the efficiency of these 3 approaches on synthetic data sets. We carried out experiments measuring the CPU time of each approach on data sets with varying both size and dimensions in ranges 10,000 - 100,000 points and 100 - 1,000 respectively.

It is clear that both LB_ABOD and FastABOD run in $O(dn^2)$ time while the running time of FastVOA depends on the parameters t, s_1, s_2. As mentioned in Section 5.3, we can use rather small parameter settings for FastVOA in very large high-dimensional synthetic data sets without reducing the accuracy. Therefore, we set $t = 100$, $s_1 = 1600$, $s_2 = 10$ for FastVOA and the sample size of FastABOF chosen as $0.1n$. Let us note that the value $0.1n$ becomes rather large when the data set size increases. In contrast, FastVOA only needs rather small number of random projections and the AMS Sketch sizes. As analysis in the Section 3.4.4, the total running time of FastVOA is $O(tn(d + \log n + s_1 s_2))$. With the choice of parameters above, the total running time of FastVOA is still dominated by the computation time of AMS Sketches with $O(ts_1 s_2 n)$ time.

Figure 6.a shows the CPU time in (ms) of FastVOA, LB_ABOD and FastABOD for data sets having 100 dimensions and sizes of 10,000 - 100,000 points while Figure 6.b displays the CPU time in (ms) for data sets having size of 20,000 points and dimensions in 100 - 1,000. It is clear that the running time of FastVOA is linear time in the size of data set and independent on number of dimensions. In contrast, both LB_ABOD and FastABOD run in quadratic time in the size of data set and linear time in number of dimensions.

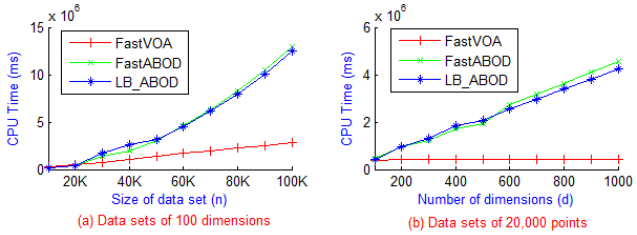

Figure 6: Comparison of CPU time of FastVOA, FastABOD and LB_ABOD.

We conclude the efficiency evaluation of FastVOA by illustrating its suitability for parallel processing. We made use of Open Multi-Processing API (OpenMP) supporting multi-platform shared memory multiprocessing programming in C++ to parallelize the **for** loop of random projection vectors in Algorithms 2 - 4 of Section 3.4.3. We measured the parallel speedup when running on 4 processors of Core i7 machine. Table 2 illustrates a nearly linear parallel speedup

of FastVOA (in the number of processors used) on synthetic data sets with size of 10,000 points on 100 dimensions.

Table 2: Parallel speedup of FastVOA

Number of processors	1	2	4
Speedup	1	2.3	3.7

6. CONCLUSION

In this paper, we introduced a random projection-based algorithm to approximate the variance of angles between pairs of points of the data set, a robust outlier score to detect high-dimensional outlier patterns. By combining random projections and AMS Sketches on product domains, our approximation algorithm runs in near-linear time in the size of data set and is suited for parallel processing. We presented a theoretical analysis of the quality of approximation to guarantee the reliability of our estimation algorithm. The empirical experiments on synthetic and real world data sets demonstrate the scalability, effectiveness and efficiency of our approach on detecting outliers in very large high-dimensional data sets.

7. ACKNOWLEDGMENTS

We deeply thank Dr. M. Schubert for his released synthetic data generator and useful comments about ABOD. We also thank T. L. Hoang for useful discussion in the early stage of this work.

8. REFERENCES

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of ICDT'01*, pages 420–434, 2001.

[2] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of SIGMOD'01*, pages 37–46, 2001.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[4] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of KDD'03*, pages 29–38, 2003.

[5] V. Braverman, K.-M. Chung, Z. Liu, M. Mitzenmacher, and R. Ostrovsky. Ams without 4-wise independence on product domains. In *Proceedings of STACS'10*, pages 119–130, 2008.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proceedings of SIGMOD'00*, pages 93–104, 2000.

[7] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC'02*, pages 380–388, 2002.

[8] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[9] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[10] A. Ghoting, S. Parthasarathy, and M. E. Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16(3):349–364, 2008.

[11] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[12] P. Indyk and A. McGregor. Declaring independence via the sketching of sketches. In *Proceedings of SODA'08*, pages 737–745, 2008.

[13] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of VLDB'98*, pages 392–403, 1998.

[14] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings KDD'08*, pages 444–452, 2008.

[15] H.-P. Kriegel, M. Schubert, and A. Zimek. Outlier detection techniques. In *Tutorial at KDD'10*, 2010.

[16] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *Proceedings of CVPR'10*, pages 1975–1981, 2010.

[17] E. Müller, M. Schiffer, and T. Seidl. Statistical selection of relevant subspace projections for outlier ranking. In *Proceedings of ICDE'11*, pages 434–445, 2011.

[18] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings of ICDE'03*, pages 315–326, 2003.

[19] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of SIGMOD'00*, pages 427–438, 2000.

[20] Y. Wang, S. Parthasarathy, and S. Tatikonda. Locality sensitive outlier detection: A ranking driven approach. In *Proceedings of ICDE'11*, pages 410–421, 2011.

[21] R. Wheeler and J. S. Aitken. Multiple algorithms for fraud detection. *Knowledge Based Systems*, 13(2-3):93–99, 2000.