Detecting Outlying Subspaces for High-Dimensional Data: A Heuristic Search Approach

Ji Zhang
Department of Computer Science,
University of Toronto, Canada
jzhang@cs.toronto.edu

Abstract

In this paper, we identify a new task for studying the outlying degree of high-dimensional data, i.e. finding the subspaces (subset of features) in which given points are outliers, and propose a novel detection algorithm, called High-D Outlying subspace Detection (HighDOD). We measure the outlying degree of the point using the sum of distances between this point and its k nearest neighbors. Heuristic pruning strategies are proposed to realize fast pruning in the subspace search and an efficient dynamic subspace search method with a sample-based learning process has been implemented. Experimental results show that HighDOD is efficient and outperforms other searching alternatives such as the naive top-down, bottom-up and random search methods.

Keywords: Outlying Subspaces, High-dimensional Data, Heuristic Search, Sample-based Learning.

1 Introduction

Outlier detection is a classic problem in data mining that enjoys a wide range of applications such as the detection of credit card frauds, criminal activities and exceptional patterns in databases. Outlier detection problem can be formulated as follows: Given a set of data points or objects, find a specific number of objects that are considerably dissimilar, exceptional and inconsistent with respect to the remaining data [5].

Numerous research works in outlier detection have been proposed to deal with the outlier detection problem defined above. They can broadly be divided into distance-based methods [7], [8], [11] and local density-based methods [4], [6], [10]. However, many of these outlier detection algorithms are unable to deal with high-dimensional datasets efficiently as many of them only consider outliers in the entire space. This implies that they will miss out the important information about the subspaces in which these outliers exist.

A recent trend in high-dimensional outlier detection is to use the evolutionary search method [2] where outliers are detected by searching for sparse subspaces. Points in these sparse subspaces are assumed to be the outliers. While knowing which data points are the outliers can be useful, in many applications, it is more important to identify the subspaces in which a given point is an outlier, which motivates the proposal of a new technique in this paper to handle this new task.

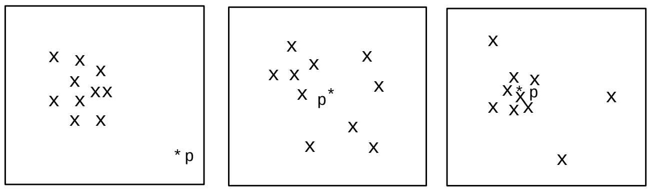

Figure 1: 2-dimensional views of the high-dimensional data

To better demonstrate the motivation of exploring outlying subspace detection, let us consider the example in Figure 1, in which three 2-dimensional views of the high-dimensional data are presented. Note that point p exhibits different outlying degrees in these three views. In the leftmost view, p is clearly an outlier. However, this is not so in the other two views. Finding the correct subspaces so that outliers can be detected is informative and useful in many practical applications. For example, in the case of designing a training program for an athlete, it is critical to identify the specific subspace(s) in which an athlete deviates from his or her teammates in the daily training performances. Knowing the specific weakness (subspace) allows a more targeted training program to be designed. In a medical system, it is useful for the Doctors to identify from voluminous medical data the subspaces in which a particular patient is found abnormal and therefore a corresponding medical treatment can be provided in a timely manner.

The major contribution of this paper is the proposal of *a dynamic subspace search algorithm, called HighDOD, that utilizes a sample-based learning process to* *efficiently identify the subspaces in which a given point is an outlier.* Note that, instead of detecting outliers in specific subspaces, our method searches from the space lattice for the associated subspaces whereby the given data points exhibit abnormal deviations. To our best knowledge, this is the first such work in the literature so far. The main features of HighDOD include:

1. The outlying measure, OD, is based on the sum of distances between a data and its k nearest neighbors [1]. This measure is simple and independent of any underlying statistical and distribution characteristics of the data points;
2. Heuristic pruning strategies are proposed to aid in the search for outlying subspaces;
3. A fast dynamic subspace search algorithm with a sample-based learning process is proposed;
4. The heuristic on the minimum sample size based on the hypothesis testing method is also presented.

The reminder of this paper is organized as follows. Section 2 discusses the basic notions and problem to be solved. In Section 3, we present our outlying subspace detection technique, called HighDOD, for high-dimensional data. Experimental results are reported in Section 4. Section 5 concludes this paper.

2 Outlying Degree Measure and Problem Formulation

Before we formally discuss our outlying subspace detection technique, we start with introduction of the outlying degree measure that will be used in this paper and formulation of the new problem of outlying subspace detection we identify.

2.1 Outlying Degree OD. For each point, we define the degree to which the point differs from the majority of the other points in the same space, termed the *Outlying Degree* (*OD* in short). OD is defined as the sum of the distances between a point and its k nearest neighbors in a data space [1]. Mathematically speaking, the OD of a point p in space s is computed as:

$$OD_s(p) = \sum_{i=1}^{k} Dist(p, p_i) | p_i \in KNNSet(p, s)$$

where $KNNSet(p, s)$ denotes the set composed by the k nearest neighbors of p in s. Note that the outlying degree measure is applicable to both numeric and nominal data: for numeric data we use Euclidean distance while for nominal data we use the simple match method. Mathematically, the Euclidean distance between two numeric points p_1 and p_2 is defined as $Dist(p_1, p_2) = [\sum((p_{1i} - p_{2i})/(Max_i - Min_i))^2]^{1/2}$, where Max_i and Min_i denote the maximum and minimum data value of the i^{th} dimension. The simple match method measures the distance between two nominal points p_1 and p_2 as $Dist(p_1, p_2) = \sum |p_{1i} - p_{2i}|/t$, where $|p_{1i} - p_{2i}|$ is 0 if p_{1i} equals to p_{2i} and is 1 otherwise. t is the total number of attributes.

2.2 Problem Formulation. We now formulate the new problem of outlying subspace detection for high-dimensional data as follows: given a data point or object, find the subspaces in which this data is considerably dissimilar, exceptional or inconsistent with respect to the remaining points or objects. These points under study are called *query points*, which are usually the data that users are interested in or concerned with.

A distance threshold T is utilized to decide whether or not a data point deviates significantly from its neighboring points. We call a subspace s is an outlying subspace of data point p if $OD_s(p) \geq T$.

2.3 Applicability of Existing High-dimensional Outlier Detection Techniques. The existing high-dimensional outlier detection techniques, i.e. find outliers in given subspaces, are theoretically applicable to solve the new problem identified in this paper. To do this, we have to detect outliers in all subspaces and a searching in all these subspaces is needed to find the set of outlying subspaces of p, which are those subspaces in which p is in their respective set of outliers. Obviously, the computational and space costs are both in an exponential order of d, where d is the number of dimensions of the data point. Such an exhaustive space searching is rather expensive in high-dimensional scenario. In addition, they usually only return the top-k outliers in a given subspace, thus it is impossible to check whether or not p is an outlier in this subspace if p is not in this top-k list. This analysis provides an insight into the inherent difficulty of using the existing high-dimensional outlier detection techniques to solve the new outlying subspace detection problem.

3 HighDOD

In this section, we present an overview of our High-Dimension Outlying subspace Detection (HighDOD) method (shown in Figure 2). It mainly consists of three modules. The X-tree Indexing module performs X-tree [3] indexing of the high-dimensional dataset to facilitate kNN search in every subspace. Sample-based Learning module randomly samples the dataset and performs dynamic subspace search to estimate the downward and

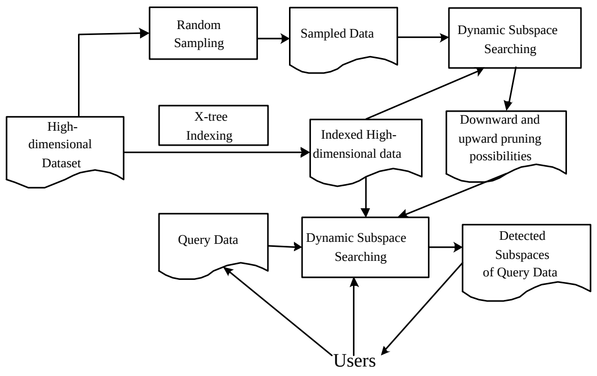
Random Sampling
Sampled Data
Dynamic Subspace Searching
High-dimensional Dataset
X-tree Indexing
Indexed High-dimensional data
Downward and upward pruning possibilities
Query Data
Dynamic Subspace Searching
Detected Subspaces of Query Data
Users

Figure 2: The overview of HighDoD

upward pruning probabilities of subspaces from 1 to d dimensions. Outlying Subspace Detection module uses the probabilities obtained in the Learning module to carry out a dynamic subspace search to find the outlying subspaces of the given query data point.

3.1 Subspace Pruning. To find the outlying subspaces of a query point, we make use of the heuristics we devise to quickly detect the subspaces in which the point is not an outlier or the subspaces in which the point is an outlier. All these subspaces can be removed from further consideration in the later stage of the search process.

In our work, we utilize a distance threshold T is used for delimiting outlying and non-outlying subspaces in the space lattice for a query data point.

OD maintains two interesting monotonic properties that allow the design of an efficient outlying subspace search algorithm.

Property 1: If a point p is not an outlier in a subspace s, then it cannot be an outlier in any subspace that is a subset of s.

Property 2: If a point p is an outlier in a subspace s, then it will be an outlier in any subspace that is a superset of s.

The above properties are based on the fact that the OD value of a point in a subspace cannot be less than that in its subset spaces. Mathematically, we have $OD_{s_1}(p) \geq OD_{s_2}(p)$ if $s_1 \supseteq s_2$.

Proof: Let a_k and b_k be the k^{th} nearest neighbors of p in the an m-dimensional subspace s_1 and n-dimensional subspaces s_2, respectively ($1 \leq n \leq m \leq d$ and $s_1 \supseteq s_2$). $MaxDist_{s2}(p)$ is the maximum distance between p and a_i, $1 \leq i \leq k$, in the subspace s_2.

We have $Dist_{s1}(p, a_k) \geq Dist_{s1}(p, a_i)|_{1 \leq i \leq k}$. Since s_1 is a superset of s_2, we thus know $Dist_{s1}(p, a_i) \geq Dist_{s2}(p, a_i)|_{1 \leq i \leq k}$. This implies $Dist_{s1}(p, a_k) \geq Dist_{s2}(p, a_i)|_{1 \leq i \leq k}$. By definition of $MaxDist_{s2}$, we have $Dist_{s_1}(p, a_k) \geq MaxDist_{s2}(p) \geq Dist_{s2}(p, b_k)$. In other words, $Dist_{s1}(p, a_k) \geq Dist_{s2}(p, b_k)$. Likewise, it is hold that $Dist_{s1}(p, a_i) \geq Dist_{s2}(p, b_i)|_{1 \leq i \leq k}$, Since $OD_{s1}(p) = \sum_1^k Dist_{s1}(p, a_i)$ and $OD_{s2}(p) = \sum_1^k Dist_{s2}(p, b_i)$. We therefore conclude: $OD_{s1}(p) \geq OD_{s2}(p)$. ■

We make use of Property 1 of OD to quickly prune away those subspaces in which the point cannot be an outlier. This is because if $OD_{s1}(p) < T$, then $OD_{s2}(p) < T$, where $s_1 \supseteq s_2$ and T is the distance threshold. In the upward pruning strategy, Property 2 of OD is utilized to detect those subspaces in which the point is definitely an outlier. The reason is that if $OD_{s2}(p) \geq T$, then $OD_{s1}(p) \geq T$.

The distance threshold T is specified as follows:

$$T = C\sqrt{\sum_{i=1}^{d} \overline{OD_{s_i}}^2}, \text{where}\, dim(s_i) = 1$$

where $\overline{OD_{s_i}}$ denotes the averaged OD value of points in the 1-dimensional subspace s_i and C is a constant factor ($C > 1$). This specification stipulates that, in any subspace, only those points whose OD values are significantly larger than the average level in the full space are regarded as outliers. The average OD level in the full space is approximated by $\sqrt{\sum_{i=1}^{d} \overline{OD_{s_i}}^2}$ and the significance of deviation is specified by the constant factor C, normally we set C=2 or 3.

3.2 Saving Factors of Subspaces Pruning. Now, we will compute the savings obtained by applying the pruning strategies during the search process quantitatively. Before that, let us first give three definitions.

Definition 1: *Downward Saving Factor (DSF) of a Subspace*

The Downward Saving Factor of a m-dimensional subspace s is defined as the savings obtained by pruning all the subspaces that are subsets of s. In other words, the Downward Saving Factor of s, denoted as DSF(s), is computed as $DSF(s) = \sum_{i=1}^{m-1} C_m^i * i$, where C_m^i denotes the combinatorial number of choosing i items out of a total of m items.

Definition 2: *Upward Saving Factor (USF) of a Subspace*

The Upward Saving Factor of an m-dimensional subspace s, denoted as USF(s), is defined as the savings obtained by pruning all the subspaces that are supersets of s. It is computed as $USF(s) = \sum_{i=1}^{d-m} [C_{d-m}^i * (m+i)]$.

Definition 3: *Total Saving Factor (TSF) of a Subspace*

The Total Saving Factor of a m-dimensional subspace, in terms of a query point p, denoted as TSF(m, p), is defined as the combined savings obtained by applying the two pruning strategies during the search process. It is computed as follows:

$TSF(m,p) = pr_{up}(m,p) * f_{up}(m) * USF(m)$, when $m = 1$;
$TSF(m,p) = pr_{down}(m,p) * f_{down}(m) * DSF(m)$
$+ pr_{up}(m,p) * f_{up}(m) * USF(m)$, when $1 < m < d$;
$TSF(m,p) = pr_{down}(m,p) * f_{down}(m) * DSF(m)$, when $m = d$.

where

(1) $f_{down}(m)$ and $f_{up}(m)$ are the percentages of the remaining subspaces to be searched. specifically, $f_{down}(m) = C_{down\_left}(m)/C_{down}(m)$ and $f_{up}(m) = C_{up\_left}(m)/C_{up}(m)$

Let $dim(s)$ denote the number of dimensions for subspace s. $C_{down\_left}(m)$ and $C_{up\_left}(m)$ are computed as: $C_{down\_left}(m) = \sum dim(s)$, where s is an unpruned or unevaluated subspace and $dim(s) < m$. $C_{up\_left}(m) = \sum dim(s)$, where s is an unpruned or unevaluated subspace and $dim(s) > m$.

$C_{down}(m)$ and $C_{up}(m)$ are the total subspace search workload in the subspaces whose dimensions are lower and higher than m, respectively. Intuitively, $f_{down}(m)$ and $f_{up}(m)$ approximate the fraction of DSF and USF of an m-dimensional subspace that are potentially achievable in each step of the search process.

(2) $pr_{up}(m,p)$ and $pr_{down}(m,p)$ are the probabilities that upward and downward pruning can be performed in the m-dimensional subspace, respectively. In other words, for a m-dimensional subspace s, $pr_{up}(m,p) = Pr(OD_s(p) \geq T)$ and $pr_{down}(m,p) = Pr(OD_s(p) < T)$. A difficulty in computing the two prior probabilities, i.e. $pr_{up}(m,p)$ and $pr_{down}(m,p)$, is that their values are unknown if there lacks any prior knowledge of the dataset. To overcome this difficulty, we first perform a sample-based learning process to obtain some knowledge about the dataset and then apply this knowledge in the later subspace search for each query point.

3.3 Sampling-based Learning. We adopt a sample-based learning process to obtain some knowledge about the dataset before subspace search of the query points are performed. This is desirable when the dataset is large so that learning the whole dataset becomes prohibitive. The task of performing this sampling-based learning is two-fold: first, we will have to estimate $\overline{OD_{s_i}}$ which will be used in specifying the distance threshold. Secondly, we will have to compute the two priors $pr_{up}(m,p)$ and $pr_{down}(m,p)$. In this learning process, a small number of points are randomly sampled from the dataset.

At first, the subspace searches are performed in the d 1-dimensional subspaces s_i on all the sampling data and $\overline{OD_{s_i}}$ is computed as the average OD values of all sampling points in subspace s_i, i.e.

$$\overline{OD_{s_i}} = \frac{1}{S}\sum_{j=1}^{S} OD_{s_i}(sp_j)$$

where S is the number of sampling points and sp_j denotes the i^{th} sampling point.

Secondly, the subspace searches are performed in the lattice of data space on the sampling data. For each sampling point sp, we have the following initial specifications regarding the two priors $pr_{up}(m,p)$ and $pr_{down}(m,p)$:

$$pr_{up}(m,sp) = pr_{down}(m,sp) = 0.5, 1 < m < d$$
$$pr_{up}(m,sp) = 1 \text{ and} pr_{down}(m,sp) = 0, m = 1$$
$$pr_{up}(m,sp) = 0 \text{ and } pr_{down}(m,sp) = 1, m = d$$

This initialization implies that we assume equal probabilities for upward and downward pruning in the subspaces of any dimension, except 1 and d, for each sampling point at the beginning. After all the m dimensional subspaces have been evaluated for sp, the $pr_{up}(m,sp)$ and $pr_{down}(m,sp)$ are computed as the percentages of m-dimensional subspaces s in which $OD_s(sp) \geq T$ and $OD_s(sp) < T$, respectively. The average pr_{up} and pr_{down} values of subspaces from 1 to d dimensions can be obtained as follows:

$$\overline{pr_{up}(m)} = \tfrac{1}{S}\textstyle\sum_{i=1}^{S} pr_{up}(m,sp_i)$$
$$\overline{pr_{down}(m)} = \tfrac{1}{S}\textstyle\sum_{i=1}^{S} pr_{down}(m,sp_i)$$

where we have $\overline{pr_{down}(1)} = \overline{pr_{up}(d)} = 0$.

For each query point p, we set $pr_{up}(m,p) = \overline{pr_{up}(m)}$ and $pr_{down}(m,p) = \overline{pr_{down}(m)}$ in the computation of TSF(m, p) of the query point p.

Remarks: There might be a misunderstanding that the sampling technique will fail here because the outliers are rare in the dataset. Recall that we are trying to detect outlying subspaces of query points, not outliers. Every point can become query point and every query point will have its outlying subspaces, if its set of outlying subspaces is not empty. Hence, the outlying subspaces can be regarded as a global property for all the points and a sample of sufficient size will make sense in the learning process.

3.4 Dynamic Subspace Search. In HighDOD, we use a dynamic subspace search method to find the subspaces in which the sampling points and the query points are outliers. The basic idea of the dynamic subspace search method is to commence search on those subspaces with the same dimension that has the highest TSF value. As the search proceeds, the TSF of subspaces with different dimensions will be updated and the set of subspaces with the highest TSF values are selected for exploration in each subsequent step. The search process terminates when all the subspaces have been evaluated or pruned. Note that the only difference between the dynamic subspace search method used on the sample points and query points lies in the decision of values of $pr_{up}(m,p)$ and $pr_{down}(m,p)$: *For sample points, we assume an equal probability of upward and downward pruning while for query points we use the averaged probabilities obtained in the learning process.*

3.5 Minimum Sampling Size for Training Dataset. Recall that the sampling method is utilized to obtain a training dataset that can be used to pre-compute the prior probabilities of upward and downward pruning, namely $\overline{pr_{up}(m)}$ and $\overline{pr_{down}(m)}$ $(1 \leq m \leq d)$. As such, samples of different sizes will only affect the pruning efficiency of the algorithm. They will not change the number of subspaces found.

With this in mind, we now wish to determine the minimum sample size to accurately predict $\overline{pr_{up}(m)}$ and $\overline{pr_{down}(m)}$ with certain degree of confidence. We denote X as the sample point that can be expressed as an S-dimensional vector as $X = [x_1, x_2, \ldots, x_S]$ where S is the size of the sample. Each data in the sample is a d-dimensional vector as $x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]^T$ where $x_{i,j}$ denote the value of j^{th} dimension of i^{th} data in the sample. Applying dynamic subspace searching on sampling points, for each dimension m, we obtain

$$Y_{down}(m) = [pr_{down}(m, sp_1), pr_{down}(m, sp_2), \ldots,$$
$$pr_{down}(m, sp_S)] \quad (1 \leq m \leq d)$$

We use the S measurements, $pr_{down}(m, sp_i)(1 \leq i \leq S)$ as the training data to estimate the mean of $pr_{down}(m)$. We estimate the sample size by constructing the confidence interval of the mean of $pr_{down}(m)$. Specifically, to obtain a $(1-\alpha)$-confidence interval, the minimum size of a random sample is given as follows [9]:

$$S_{min}(m) = [\frac{t_{\alpha/2} * \sigma_m^{'}}{\delta^*}]^2$$

where $\sigma_m^{'}$ denotes the estimated standard deviation of pr_{down} in the m^{th} dimension using the training points that is defined as:

$$\sigma_m^{'} = \sqrt{\sum_{i=1}^{S}(pr_{down}(m, sp_i) - \overline{pr_{down}(m, sp)})^2/(S-1)}$$

δ^* denotes the half-width of the confidence interval.

Note that the value of $\sigma_m^{'}$ varies for different m. Let $\sigma_{max}^{'} = max(\sigma_m^{'})(1 \leq m \leq d)$, the minimum sample size S_{min} that satisfies respective minimum sample size requirement of each dimension is computed as:

$$S_{min} = [\frac{t_{\alpha/2} * \sigma_{max}^{'}}{\delta^*}]^2$$

Similarly reasoning applies to $\overline{pr_{up}(m)}$ since $\overline{pr_{up}(m)}$= 1- $\overline{pr_{down}(m)}$.

4 Experimental Results

In this section, we will carry out extensive experiments to test the efficiency of outlying subspace detection and the effectiveness of outlying subspace compression in HighDOD. Synthetic datasets are generated using a high-dimensional dataset generator and four real-life high-dimensional datasets from the UCI machine learning repository, which have been used in [2] for performance evaluation of their high-dimensional outlier detection technique, are also used.

Since the existing high-dimensional outlier detection techniques fail to handle the new outlying subspace detection problem, we thus choose to compare the efficiency of several subspace search methods, i.e. top-down, bottom-up, random and dynamic subspace search, instead.

These searching methods aim to find the outlying subspaces of the given query data using various searching strategies. The top-down search method only employs a downward pruning strategy while the bottom-up search method only uses an upward pruning strategy. The random search method, the "headless chicken" search alternative, randomly selects the layer in the lattice for search without replacement in each step. The dynamic search method, a hybrid of upward and downward search, computes the TSF of all subspaces of different dimensions and selects the best layer of subspaces for search. To evaluate the efficiency of the sample-based learning process , we run the dynamic search algorithm with and without incorporating the sample-based learning process. Note that the execution times shown in this section are the average time spent in processing each point in the learning and query process.

Effect of Dimensionality. First, we investigate the effect of dimensions on the average execution time of HighDOD (see Figure 3) . We can see that the execution time of all the five methods increase at an exponential rate since the number of subspaces increases exponentially as the number of dimension goes up, regardless of which searching and pruning strategy is utilized. On a closer examination, we see that (1) The execution

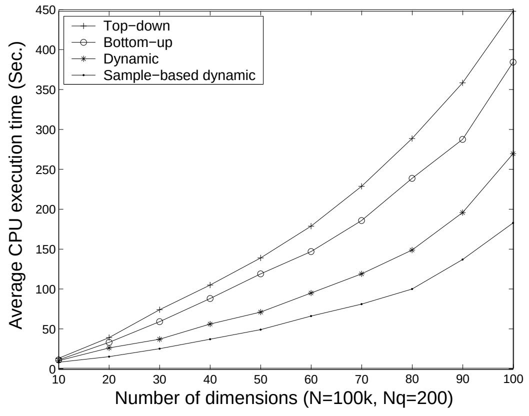

Figure 3: Execution time when varying dimension of data

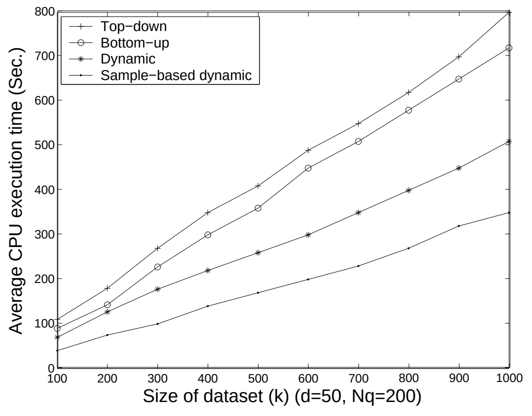

Figure 4: Execution time when varying size of dataset

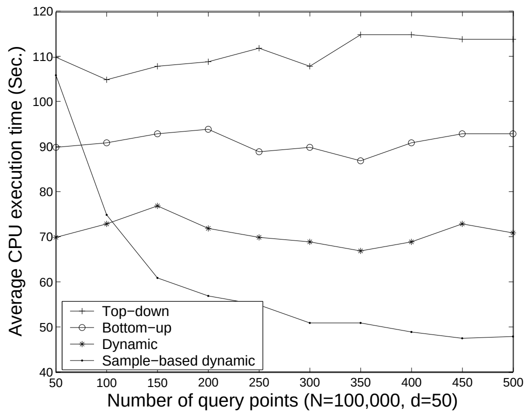

Figure 5: Execution time when varying the number of query points

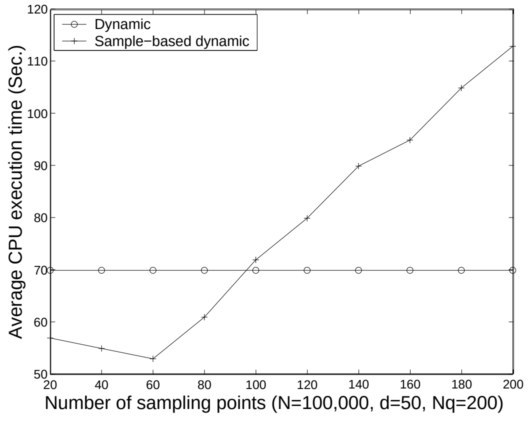

Figure 6: Execution time when varying the size of sample

time of top-down and bottom-up search methods increase much faster than the dynamic search method; (2) When using the sample-based learning process, the dynamic search method performs better than the method without using the sample-based learning process.

Effect of Dataset Size. Second, we fix the number of dimensions at 50 and vary the size of datasets from $100k$ to 1,000k. Figure 4 shows that the average execution times using the five methods to process each query point are approximately linear with respect to the size of the dataset. Similar to results of the first experiment, the dynamic search method with sample-based learning process gives the best performance.

Effect of Number of Query Points. Next, we vary the number of query points N_q. Figure 5 shows the results of the five searching methods. It is interesting to note that when N_q is large, dynamic search method with sample-based learning process gives the best performance. However, when N_q is small, it is better to use dynamic search without sample-based learning. The reason is because when the number of query points is small, the saving in computation by using the learning process is not sufficient to justify the cost of the learning process itself.

Effect of Sample Size. We also investigate the effect of the number of sampling points, S, used in the learning process. A large S gives a more accurate estimation of the possibilities of upward and downward pruning in subspaces, which in turn, helps to speedup the search process. However, a large S also implies an increase in the computation during the learning process, which may increase the average time spent in the whole detection process. As shown in Figure 6, the execution time is first decreased when the number of sampling points is small, this is because the prediction of possibility is not accurate enough, which cannot

Datasets(dimensions)	Top-down	Bottom-up	Random	Dynamic	Sample+ Dynamic
Machine(8)	56	49	58	41	32
Breast Cancer (14)	165	176	150	121	110
Segmentation (19)	251	237	256	222	197
Ionosphere (34)	472	477	456	414	387
Musk (160)	5203	4860	5002	4389	3904

Table 1: Results of running five methods on real-life datasets (average CPU time in seconds for each query point)

greatly speedup the later searching process. When the sample size increases, the prediction of the possibilities are sufficiently accurate, therefore any larger size of sample will no longer contribute to the speedup of the search process, but only increase the execution time as a whole. The horizontal dot-line in Figure 6 indicates the execution time when dynamic subspace search without sample-based learning is employed.

Results on Real-life Datasets. Finally like [2], we evaluate the practical relevance of HighDOD by running experiments on five real-life high-dimensional datasets in the UCL machine learning repository. The datasets range from 8 to 160 dimensions. Table 1 shows the results of the five search methods. It is obvious that dynamic search with sampling-based learning process works best in all the real-life datasets. Furthermore, using dynamic subspace search alone is faster than top-down bottom-up or random search methods by approximately 20% while incorporating sample-based learning process into dynamic subspace search further reduces the execution time by about 30%.

5 Conclusions

In this paper, we propose a novel algorithm, called HighDOD, to address the new problem of detecting outlying subspaces for high-dimensional data. In HighDOD, heuristics for fast pruning in the subspace search and a dynamic subspace search method with a sample-based learning process are used. Experimental results justify the efficiency of outlying subspace searching in HighDOD. We believe that HighDOD is useful in revealing interesting and new knowledge in outlying analysis of high-dimensional data and can be potentially used in many practical applications.

References

[1] F. Angiulli and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. Proc. *PKDD'02*,Helsinki, Finland, 2002.

[2] C. C Aggarwal and P.S. Yu. Outlier Detection in High Dimensional Data. Proc. *ACM SIGMOD'00*, Santa Barbara, California, 2001.

[3] S. Berchtold, D. A. Keim and H. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. Proc. *VLDB'96*, Mumbai, India, 1996.

[4] M. Breuning, H-P, Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. Proc. *ACM SIGMOD'00*, Dallas, Texas, 2000.

[5] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2000.

[6] W. Jin, A. K. H. Tung, J. Han. Finding Top n Local Outliers in Large Database. Proc. *SIGKDD'01*, San Francisco, CA, August, 2001.

[7] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-based Outliers in Large Dataset. Proc. *VLDB'98*, pages 392-403, New York, NY, August 1998.

[8] E. M. Knorr and R. T. Ng. Finding Intentional Knowledge of Distance-based Outliers. Proc. *VLDB'99*, pages 211-222, Edinburgh, Scotland, 1999.

[9] A. E. Mace. *Sample-size Determination*. Reinhold Publishing Corporation, New York, 1964.

[10] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos: LOCI: Fast Outlier Detection Using the Local Correlation Integral. Proc. *ICDE'03*, pages 315, Bangalore, India, 2003.

[11] S. Ramaswamy, R. Rastogi, and S. Kyuseok. Efficient Algorithms for Mining Outliers from Large Data Sets. Proc. *ACM SIGMOD'00*, Dallas, Texas, 2000.