An Efficient Unsupervised AdaptiveAntihub Technique for Outlier Detection in High Dimensional Data

1Mrs.R.Lakshmi Devi, 2Dr.R.Amalraj

1ResearchScholar, Mother Teresa Women’s University, Kodaikanal, India
2Associate Professor, Dept. of Computer Science Sri Vasavi Arts College, Erode, India.

ABSTRACT

Identification of unsupervised outliers in a high dimensional data becomes an emerging technique in today’s research in the area of data mining. Increase of dimensionality leads to various challenges. Hubness especially Antihubs (points that infrequently occur in k nearest neighbor lists) is the recently known concept for the increase of dimensionality pertaining to nearest neighbors. Outlier detection using AntiHub method is refined as Antihub2 to reevaluate the outlier scores of a point produced by the AntiHub method. However, it leads to the increase of computation time of an algorithm with large number of computations. This paper establishes an approach called AdaptiveAntihub, which embeds an adaptive technique in Antihub2 for unsupervised outlier detection mainly to reduce the number of computations and computation time of an algorithm and compares the results produced by Antihub2 with AdaptiveAntihub. The experimental results illustrate that AdaptiveAntihub outperforms well and it also proves that there is a significant reduction in the number of computations and computation time when this proposed technique is applied to detect unsupervised outliers in high dimensional data.

Keywords: AdaptiveAntihub, Antihub, Antihub2, Outliers, Unsupervised

I. INTRODUCTION

An outlier is an observation which appears to be inconsistent with the remainder of that set of data. In data mining, detection of outliers is an important research area. Most of the applications which apply outlier detection are high dimensional. Increase of dimensionality leads to sparsity of data. Sparse data is difficult to handle. The sparsity of high dimensional data signifies that every point is an almost equally good outlier [1]. Outlier (anomaly) detection refers to the process of finding patterns that do not conform to standard behavior.

Outliers can be of three types such as point, contextual or collective outliers. Outlier detection techniques can be classified into three different categories such as supervised, semi supervised and unsupervised based on the existence of the labels for outliers. The unsupervised outlier detection is more applicable, where only a single data set without labels is given. The other techniques both require appropriate training set which is an expensive, time consuming and burdensome task [2]. In this paper, the proposed technique is applied to an unsupervised outlier detection. Hubness or k-hubness of an object x: NH(x) is the number of times a point x is counted as one of the k nearest neighbors of any other point in a data set [3].

The concept of hubness has recently become as an essential aspect of the increase of dimensionality related to nearest neighbors [4] and can be used in a standard methods used for detecting outliers. [5] Explores an impact of hubness on a various machine-learning tasks that utilizes distances between points, belonging to supervised, semi-supervised, and unsupervised learning families and by examining the origin of hubness, authors show that it is an essential property of data distributions in high-dimensional data.

The rest of the paper is organized as follows: Section 2 specifies an existing methods related to unsupervised outlier detection and Section 3 explains the proposed approach and its implications. Finally, Section 4 describes the experimental results with real datasets, and the conclusion is in Section 5.

II. RELATED WORK

In Recent research, various papers explored the influence of hubness in high-dimensional data on different data mining outlier detection tasks. Reverse nearest neighbors count is recognized in unsupervised distance-based outlier detection [4]. Outlier scoring based on NH counts used in the ODIN method was reformulated and introduced here as an antihub which defines the outlier score of point x from data set D as a function of NH(x) and explores the interplay of hubness and data sparsity. Outlier detection methods are
implemented centered on the properties of antihubs. The relationship between dimensionality, neighborhood size, and reverse neighbors are taken into account for the effectiveness.

Unsupervised outlier detection is the process of detecting outliers in a given dataset without the need of labels in the training set. The paper [6] proposes method for finding distance based outliers based upon the k nearest neighbor points. To calculate the number of data points falling, two algorithms such as nested loop join and index join algorithms are used. Also partition-based algorithm is used. This algorithm divides the data set into different subsets and then cuts entire partitions rapidly as it is determined that they cannot contain outliers.

Distance based method to deal with the problem of finding outliers for k dimensional data sets where k >=5 is focused in paper [7]. Applying three algorithms such as index based, nested loop based, and cell based, authors come to the conclusion that cell based is for k<=4 and nested loop is the choice for k>=5 and also finds that there is no limit on the size of the dimensions.

A mostly used density based method is the local outlier factor (LOF) [8], which influenced many variations, e.g. LDOF (Local Distance-based Outlier Factor) approach [9], and LoOP (Local Outlier Probability) [10]. In many unsupervised outlier detection algorithms proposed, nearest-neighbor based algorithms appears to be the mostly used methods today [11, 12]. In this context, outliers are determined by their distances to their nearest neighbors.

The approach used in [13] is based on the relationship between k- Nearest Neighbor and Reverse Nearest Neighbor which involves two phases. First phase is dealing with the problem of finding a query point using kNN. The second step introduces Boolean Range Query which checks the existence of a point in a given region can be used for RNN queries problems.

Reverse nearest neighbor search [14] is used on high dimensional and multimedia data. [15] explores an important feature of the curse concerning to the distribution of k-occurrences (the number of times a point appears among the k nearest neighbors of additional points in a data set) and shows that, as dimensionality increases, this distribution of data is skewed and hub points arise (points with very high k-occurrences).

The concept of hubness (Antihub\(^2\)) presented in [4] motivated to implement the proposed approach AdaptiveAntihub to reduce the computation time by reducing computations.

III. PROPOSED APPROACH

I. Methodology

1.1 Antihubs

Hubness is derived from the idea of k occurrences. Different data points occur in k neighbor sets with increasingly unequal frequencies. When some points occur in many kNN sets, it is referred to as hubs, while others occur either very rarely or not at all, then they are referred to as antihubs. More specifically, hubness refers to an increasing skewness in the k occurrence distribution in high-dimensional data [16].

The concept of Antihub has recently become as an essential aspect of the increase of dimensionality related to nearest neighbors. In summary, the emergence of antihubs is closely interrelated with outliers in high-dimensional data suggests that antihubs can be used as an alternative to standard outlier-detection methods.

The basic structure of Antihub [4] is given below:

\[ \text{AntiHub}_{\text{data}} (D, k) \]

**Input:**
- Distance measure dist.
- Ordered data set \( D = (x_1, x_2, \ldots x_n) \), where \( x_i \in \mathbb{R}^d \), for \( i \in \{1, 2, \ldots n\} \)
- No. of neighbors \( k \in \{1, 2, \ldots \} \)

**Output:**
- Vector \( s = (s_1, s_2, \ldots s_n) \in \mathbb{R}^n \), where \( s_i \) is the outlier score of \( x_i \) for \( i \in \{1, 2, \ldots n\} \)

**Temporary variables:**
- \( t \in \mathbb{R} \)

**Steps:**
1) For each \( i \in \{1, 2, \ldots n\} \)
2) \( t := N_i (x_i) \) computed w.r.t. dist and data set \( D|x_i \)
3) \( s_i := f (t) \), where \( f : \mathbb{R} \rightarrow \mathbb{R} \) is a monotone function.

Antihub method takes high dimensional data set as an input. For each point \( x \), it finds out the \( N_i(x) \) value (reverse k-nearest neighbor count of \( x \)) with respect to Euclidian distance. Outlier score is obtained based on the function \( f \) where it is \( 1/ (N_i(x) + 1) \), which assumes that the higher the score, the more the point is considered an outlier, and maps the scores to the \( (0,1] \) range. If the outlier score crosses the limit of user defined threshold, then it is treated as an outlier.
1.2 Proposed Algorithm

An AdaptiveAntihub technique extending Antihub\(^2\) is constructed to reevaluate the outlier score of a point by considering \(N_k\) scores of the neighbors of \(x\) in addition to \(N_k(x)\) itself. This technique is explored in this paper especially to improve the efficiency and to reduce the computational time with reduced number of computations. The basic architecture of the proposed approach is given in Fig 1.

![Fig 1. Basic architecture of the proposed approach.](image)

This paper establishes an approach called AdaptiveAntihub which embeds an adaptive technique in Antihub\(^2\) algorithm which is a refinement of Antihub where scores of \(N_k(x)\) and \(N_k\) scores of the neighbors of \(x\), are considered for unsupervised outlier detection. In high dimensional data, for each point \(x\), AntiHub\(^2\) adds \((1 - \alpha) \cdot N_k(x)\) to \(\alpha\) times the sum of \(N_k\) scores of the \(k\) nearest neighbors of \(x\), where \(\alpha \in [0, 1]\). Since each \(\alpha\) is applied for each point \(x\) and \(\alpha\) is also measured by the step size where \(\text{step} \in (0, 1)\), it may lead Antihub\(^2\) to attain longer period of time mainly because of large number of iterations. In contrast to this concept, an AdaptiveAntihub is introduced where, it divides the \(\alpha\) set into four sections. Instead of applying all \(\alpha\) values for each point \(x\), the proposed technique applies only the limited number of \(\alpha\). It omits moving into last section based on the concept that \(x\) depends its neighbors only on certain percentage. By comparing the corresponding cdisc values of first three sections it moves into the corresponding direction quickly so that it will reduce the number of computations and computation time of an algorithm.

The basic structure of the proposed algorithm AdaptiveAntihub is as follows:

AdaptiveAntiHub\(_{\text{num}}\) (\(x, k, p, \text{step}\))

**Input:**
- Distance measure dist
- Ordered data set \(D = (x_1, x_2, \ldots x_n)\), where \(x_i \in \mathbb{R}^d\), for \(i \in \{1, 2 \ldots n\}\)
- No. of neighbors \(k \in \{1, 2 \ldots \}\)
- Ratio of outliers to maximize discrimination \(p \in (0, 1]\)
- Search parameter \(\text{step} \in (0, 1]\)

**Output:**
- Vector \(s = (s_1, s_2, \ldots s_n) \in \mathbb{R}^n\), where \(s_i\) is the outlier score of \(x_i\), for \(i \in \{1, 2 \ldots n\}\)

**Temporary variables:**
- AntiHub scores \(a \in \mathbb{R}^n\)
- Sums of nearest neighbors’ AntiHub scores \(\text{ann} \in \mathbb{R}^n\)
- Proportion \(\alpha \in [0, 1]\)
- (Current) discrimination score \(\text{cdisc}, \text{disc} \in \mathbb{R}\)
- (Current) raw outlier scores \(\text{ct}, t \in \mathbb{R}^n\)
Local functions:

discScore(y, p): for \( y \in \mathbb{R}^n \) and \( p \in (0, 1] \) outputs the number of unique items among \( \lceil np \rceil \) smallest members of \( y \), divided by \( \lceil np \rceil \).

1. \( a_i = \text{AntiHub}_{\text{dist}}(D, k) \)
2. For each \( i \in (1, 2, \ldots n) \)
3. \( a_{nn_i} = \sum \text{NN}_{\text{dist}}(k, i) a_i \), where \( \text{NN}_{\text{dist}}(k, i) \) is the set of indices of \( k \) nearest neighbors of \( x_i \)
4. disc: = 0
5. For each \( \alpha \in (0, \text{step}, 2 \cdot \text{step}, \ldots 1) \)
6. sbeg = 1; send = no. of \( \alpha \) values; divv = send/4;
7. lmid = round (sbeg + divv);
8. rmid = round (send − divv);
9. while lmid <= rmid & conc==0 && lmid >= sbeg {
10. Find \( \alpha_l \) and \( \alpha_r \) as alpha_{\text{lmid}} and alpha_{\text{rmid}} respectively
11. Find lct and rct values based on alpha_{\text{lmid}} and alpha_{\text{rmid}} respectively where\( ct_i = (1 - \alpha) \cdot a_i + \alpha \cdot an_{nn_i} \)
12. If lmc = lmdisc
13. t = lct; disc = lmc; conc=1; break;
else if lmc < lmdisc
14. t = rct; disc = rmdisc; lmid = round (lmid + divv);
15. else if lmc >= rmdisc
16. t = lct; disc = lmdisc; rmid = lmid; mid = round (lmdiv); }
17. For each \( i \in (1, 2, \ldots n) \)
18. \( s_i = f(t_i) \), where \( f: \mathbb{R} \rightarrow \mathbb{R} \) is a monotone function

IV. EXPERIMENTAL EVALUATION

An adaptive technique is applied in Antihub\textsuperscript{2} algorithm for the evaluation of computation complexity and statistical measures of accuracy is used for performance evaluation. In this section, the effectiveness and behavior of the proposed approach is examined in terms of number of computations, computation time and accuracy by applying it on three different real data sets obtained (wilt, aloi, and churn) against that algorithm. Wilt data set consists of image segments, generated by segmenting the pansharpened image with totally 4339 image segments. It involves 6 attributes. ALOI (Amsterdam Library of Object Images) dataset is a color image collection of 1, 00,000 small objects with 64 attributes. churn is a dataset with 1667 objects and 21 attributes. This section describes those experiments and their results.

Accuracy is the proportion of true results, either true positive or true negative, in a population. It measures the degree of veracity of a test on a condition. The terms that are used along with the description of accuracy are true positive (TP), true negative (TN), false negative (FN), and false positive (FP). Accuracy can be described as

\[
\text{Accuracy} = \frac{(TN + TP)}{(TN+TP+FN+FP)} = \frac{\text{No. of correct assessments}}{\text{No of all assessments}}
\]

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>NUMBER OF COMPUTATIONS ENCOUNTERED FOR ANTIHUB\textsuperscript{2} AND ADAPTIVEANTIHUB WHEN K=120 FOR ALOI, WILT AND CHURN DATA SETS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Antihub\textsuperscript{2}</td>
</tr>
<tr>
<td>ALOI</td>
<td>48279</td>
</tr>
<tr>
<td>WILT</td>
<td>91119</td>
</tr>
<tr>
<td>CHURN</td>
<td>35007</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>58135</td>
</tr>
</tbody>
</table>

Table I shows the number of computations occurred for both the algorithms for all the three datasets when \( k=120 \). From this table, it is well understood that AdaptiveAntihub has a significant reduction (in an
average of 90.47% of reduction) in number of computations, occurs for all the three datasets when it is compared with the existing Antihub\(^2\). It proves that AdaptiveAntihub outperforms well than the other.

![Graph](image)

**Fig 2** The number of computations encountered for Antihub\(^2\) and AdaptiveAntihub for ALOI, WILT and CHURN data sets

Fig 2 demonstrates that, when we compare the number of computations for the Antihub\(^2\) and AdaptiveAntihub for all the three datasets, AdaptiveAntihub has a significant reduction in number of computations and also proves that it outperforms well with the data set dimensionality than Antihub\(^2\).

**TABLE II**

<table>
<thead>
<tr>
<th></th>
<th>Antihub(^2) (secs)</th>
<th>AdaptiveAntihub (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALOI</td>
<td>3.6426</td>
<td>3.1532</td>
</tr>
<tr>
<td>WILT</td>
<td>1.8552</td>
<td>1.376</td>
</tr>
<tr>
<td>CHURN</td>
<td>9.3879</td>
<td>7.1802</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>4.9619</td>
<td>3.9031</td>
</tr>
</tbody>
</table>

Table II shows the computation time taken by Antihub\(^2\) and AdaptiveAntihub for all the three datasets when k=120. From this table, it is well understood that the proposed AdaptiveAntihub has 21.33% of reduction in an average in computation time for all the three datasets when it is compared with the existing Antihub\(^2\). It proves that AdaptiveAntihub outperforms well than the other.

![Graph](image)

**Fig 3** The computation time of Antihub\(^2\)and AdaptiveAntihub for ALOI, WILT and CHURN data sets
Fig 3 illustrates that, when we compare the computation time of Antihub$^2$ with AdaptiveAntihub for all the three datasets, AdaptiveAntihub has a significant reduction in computation time and also proves that it outperforms well with the data set dimensionality than Antihub$^2$.

**TABLE III**

**THE PERFORMANCE ACCURACY OF ANTIHUB$^2$ AND ADAPTIVEANTIHUB FOR ALOI, WILT AND CHURN DATA SETS WHEN K=10, 50, 100 AND 120**

<table>
<thead>
<tr>
<th>Value</th>
<th>Antihub$^2$</th>
<th>AdaptiveAntihub</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALOI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.7703</td>
<td>0.7612</td>
</tr>
<tr>
<td>50</td>
<td>0.7577</td>
<td>0.7603</td>
</tr>
<tr>
<td>100</td>
<td>0.7586</td>
<td>0.759</td>
</tr>
<tr>
<td>120</td>
<td>0.7595</td>
<td>0.7595</td>
</tr>
<tr>
<td>Average</td>
<td>0.761525</td>
<td>0.76</td>
</tr>
<tr>
<td>WILT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.9813</td>
<td>0.9825</td>
</tr>
<tr>
<td>50</td>
<td>0.9829</td>
<td>0.9829</td>
</tr>
<tr>
<td>100</td>
<td>0.9827</td>
<td>0.9827</td>
</tr>
<tr>
<td>120</td>
<td>0.9829</td>
<td>0.9829</td>
</tr>
<tr>
<td>Average</td>
<td>0.98245</td>
<td>0.98275</td>
</tr>
<tr>
<td>CHURN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.9904</td>
<td>0.9994</td>
</tr>
<tr>
<td>50</td>
<td>0.9988</td>
<td>0.9868</td>
</tr>
<tr>
<td>100</td>
<td>0.9418</td>
<td>0.9418</td>
</tr>
<tr>
<td>120</td>
<td>0.9328</td>
<td>0.9304</td>
</tr>
<tr>
<td>Average</td>
<td>0.96595</td>
<td>0.9646</td>
</tr>
</tbody>
</table>

Table III shows the Performance Accuracy of Antihub$^2$ and AdaptiveAntihub, for ALOI, WILT and CHURN data sets when k=10, 50, 100 and 120. It also illustrates that accuracy in an average are at the same level for both the algorithms for all the three data sets even when the k value changes.

![Fig 4](image-url)

**Fig 4 The performance accuracy of Antihub$^2$ and AdaptiveAntihub for ALOI dataset**

Fig 4 shows that the performance accuracy of Antihub$^2$ and AdaptiveAntihub when using the dataset ALOI are almost at the same level when k=10, 50, 100, 120. Therefore while comparing the number of computations and computation time between the two algorithms in Table I as well as Table II, it is well understood that AdaptiveAntihub has obtained 90.47% of reduction in number of computations and 13.43% of reduction in computation time than Antihub$^2$ with almost, the same level of accuracy for both the algorithms in ALOI dataset.
Fig 5 shows that the performance accuracy of Antihub\textsuperscript{2} and AdaptiveAntihub when using the dataset WILT are almost at the same level when \( k = 10, 50, 100, 120 \). Therefore while comparing the number of computations and computation time between the two algorithms in Table I as well as Table II, it is well understood that AdaptiveAntihub has obtained 90.47\% of reduction in number of computations and 25.83\% of reduction in computation time than Antihub\textsuperscript{2} with almost, the same level of accuracy for both the algorithms in WILT dataset.

Fig 6 shows that the performance accuracy of Antihub\textsuperscript{2} and AdaptiveAntihub when using the dataset CHURN are almost at the same level when \( k = 10, 50, 100, 120 \). Therefore while comparing the number of computations and computation time between the two algorithms in Table I as well as Table II, it is well understood that AdaptiveAntihub has obtained 90.47\% of reduction in number of computations and 23.51\% of reduction in computation time than Antihub\textsuperscript{2} with almost, the same level of accuracy for both the algorithms in CHURN dataset.

V. CONCLUSION

This paper, presents an approach AdaptiveAntihub where an adaptive technique is applied to hubness, especially Antihub\textsuperscript{2} for unsupervised outlier detection to reduce the number of computations and computation time. The performance of Antihub\textsuperscript{2} is empirically compared with AdaptiveAntihub in terms of number of computations, computation time and accuracy by applying it into three different data sets and found from the experimental results that AdaptiveAntihub provides a significant reduction in number of computations and computational time than Antihub\textsuperscript{2} with same level of accuracy for both the algorithms for all the three datasets. From this analysis, it is well understood that when an AdaptiveAntihub technique is applied, it outperforms well with the data set dimensionality than Antihub\textsuperscript{2} algorithm on identifying meaningful and interesting outliers.
REFERENCES