

HOT: Hypergraph-based Outlier Test for Categorical Data ^{*}

Aoying Zhou, Li Wei, Weining Qian[†]
Dept. of Computer Science, Fudan Univ.
{ayzhou, lwei, wnjian}@fudan.edu.cn

Wen Jin
Dept. of Computer Science, Simon Fraser Univ.
wjn@cs.sfu.ca

ABSTRACT

As a widely used data mining technique, outlier detection is a process which aims to find anomalies while providing good explanations. Most existing detection methods are basically designed for numeric data, however, real-life data such as web pages, business transactions and bioinformatics records always contain categorical data. So it causes difficulty to find reasonable exceptions in the real world applications.

In this paper, we introduce a novel outlier mining method based on hypergraph model for categorical data. Since hypergraphs precisely capture the distribution characteristics in data subspaces, this method is effective in identifying anomalies in dense subspaces and presents good interpretations for the local outlierness. By selecting the most relevant subspaces, the problem of "curse of dimensionality" in very large databases can also be ameliorated. Furthermore, the connectivity property is used to replace the distance metrics, so that the distance-based computation is not needed anymore, which enhances the robustness for handling missing-value data. The fact that connectivity computation facilitates the aggregation operations supported by most SQL-compatible database systems, makes the mining process much efficient. Finally, we give experiments and analysis which show that our method can find outliers in categorical data with good performance and quality.

Keywords

Outlier, Hypergraph, High-dimensional data

1. INTRODUCTION

^{*}The work is partially supported by the National Grand Fundamental Research 973 Program of China under Grant No. G1998030414 and the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 99038

[†]The author is partially supported by Microsoft Research Fellowship.

Outlier detection is one of the major technologies in data mining, whose task is to find small groups of data objects that are exceptional when compared with rest large amount of data. Outlier mining has strong application background in telecommunication, financial fraud detection, and data cleaning, since the patterns lying behind the outliers are usually interesting for helping the decision makers to make profit or improve the service quality.

A descriptive definition of outliers is given by Hawkins like this: "an outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [12]. Although some different definitions have been adopted by researchers, they may meet problems when being applied to real-life data. In real applications, data are usually mix-typed, which means they contain both numeric and categorical data. Since most current definitions are based on distance, such as distance-based outlier [18, 19], local outlier [9], or density in cells, such as high-dimensional outlier [1], they cannot handle categorical data effectively. The following example shows the difficulties of processing categorical data.

Example 1. Consider a ten-record, five-dimensional customer information dataset with dimensions *RID*, *Name*, *Age-range*, *Car-type*, and *Salary-level*. as shown in Table 1. Obviously, it is nonsensical to test dimensions record-id and customer name. We are interested in the other three dimensions, that is *Age-range*, *Car-type*, and *Salary-level*, which may be useful for analyzing the latent behavior of the customers. Each of the three attributes has two possible values, as presented in the table. By calculating, we can see that occurrences of the combinations of the attribute values are close to each other - there are two instances of ('Middle', 'Sedan', 'Low'), three of ('Young', 'Sports', 'Low'), one of ('Young', 'Sedan', 'High'), one of ('Young', 'Sports', 'High'), and three of ('Middle', 'Sedan', 'High'). So it is hard to figure out the outliers: based on the occurrence, no record is outlier, or all are outliers. Along with the dimensionality and the number of possible values of each attribute increase, this problem may become even more severe, for the well-known *curse of dimensionality* problem. Therefore, finding outliers in global space is meaningless. However, even finding outliers in subspaces is impossible for the existence of the problem *combinational explosion*, which will lead to inefficient search and inexplicable result.

Example 1 illustrates the problems of *curse of dimension-*

Table 1: Some Simple Customer Data

RID	Name	Age-range	Car-type	Salary-level
1	Mike	Middle	Sedan	Low
2	Jack	Middle	Sedan	High
3	Mary	Young	Sedan	High
4	Alice	Middle	Sedan	Low
5	Frank	Young	Sports	High
6	Linda	Young	Sports	Low
7	Bob	Middle	Sedan	High
8	Sam	Young	Sports	Low
9	Helen	Middle	Sedan	High
10	Gary	Young	Sports	Low

ality and *combinational explosion* met by data containing categorical attributes. Unfortunately, they are not the only difficulties we are confronting. In some applications, data may contain missing values, which means that for some objects, the value of certain attribute is unknown. Obviously, distance- or density-in-cell- based definitions cannot process this kind of data. However, these research works do provide valuable ideas for observing outliers. In [9], the authors emphasize that outlying is a relative concept, which should be studied in local area. In [19] and [1], the outliers are mined in subspaces, where only partial attributes are considered, so that the *curse of dimensionality* is partially overcome.

How to define the local outliers in mix-typed, high-dimensional data? Is there an approach to find the outliers efficiently? This paper introduces a possible solution to these two questions.

1.1 Our Contributions

The major contributions of this paper are as follows:

- We propose a definition for outliers in high-dimensional categorical data, which not only considers the locality of the whole data space, but also of the dimensions, so that the outliers can be easily explained;
- We propose an efficient algorithm for finding this kind of outliers, which is robust for missing values;
- The techniques for handling real life data are discussed, which include the preprocessing for numeric data, handling missing values, postprocessing for pruning banal outliers, and explanation and management of outliers and the corresponding knowledge, so that the method can be applied in real applications;
- We introduce a quantified method for measuring the outliers, which can be used to analyze the quality of outlier detection results.

1.2 Paper Organization

The rest of the paper is organized as follows. Section 2 provides the hypergraph model and the definitions for outliers formally. In section 3, the algorithm for mining outliers is presented with the enhancement for handling real-life data. The empirical study of the proposed method is given in section 4. After a brief introduction to related work in section 5, section 6 is for conclusion remarks.

Table 2: Hypergraph modeling

HyperedgeID	Frequent itemsets	Vertices
1	('Middle', *, *)	1, 2, 4, 7, 9
2	('Young', *, *)	3, 5, 6, 8, 10
3	(* , 'Sedan', *)	1, 2, 3, 4, 7, 9
4	(* , *, 'Low')	1, 4, 6, 8, 10
5	(* , *, 'High')	2, 3, 5, 7, 9
6	('Middle', 'Sedan', *)	1, 2, 4, 7, 9

2. PROBLEM STATEMENT

In the above, we have briefly examined the problems existing in the outlier detection of real-life data, especially those with categorical attributes or missing attribute values. To address the problems, we propose a hypergraph-based outlier detection method for categorical data. Here, we will first describe the hypergraph model and give our own outlier definition based on the model.

Definition 1. A *hypergraph* $H = (V, HE)$ is a generalized graph, where V is a set of vertices and HE is a set of hyperedges. Each hyperedge is a set that contains more than two vertices.

In our model, each vertex $v \in V$ corresponds to a data object in the dataset, and each hyperedge $he \in HE$ denotes a group of objects that all contain a frequent itemset. In other words, each hyperedge corresponds to a frequent itemset, and is the set of vertices corresponding to the objects containing the itemset.

With this model, the original dataset in the high-dimensional space is mapped into a hypergraph. In the following examples, we will show that hypergraph can provide outliers with both appropriate viewpoints and reasonable explanations.

Example 2. To model the dataset in Example 1 using a hypergraph, the ten records in the dataset are mapped to the vertices in the hypergraph. Assume that the minimum support count is set to five, we can get the hyperedges shown in Table 2. The frequent itemsets in the table are presented in trinarities, of which the elements denote the values in *Age - range*, *Car - type*, and *Salary - level* respectively. Furthermore, the '*' denotes that any value of the corresponding attribute does not appear in the itemset. The items in the vertices column denote the *RIDs* of the objects that appear in each hyperedge.

It is obvious that the objects falling in the same hyperedge have common attribute values that form the hyperedge. Therefore, if we observe the dataset from this angle, these objects are similar. They fall in a local area determined by part of the dimensions.

Before giving our definition for outliers, we list the notions to be used in Table 3.

Definition 2. For data object o in a hyperedge he and attribute A , the **deviation of the data object o on at-**

Table 3: Notions

Notion	Meaning
N	The number of objects in database DB .
$\ DS\ $	The number of elements in set DS .
A, A_i	Each denotes an attribute.
B, C	Each denotes a set of attributes.
v_o^i	The value of attribute A_i in object o .
A^{DS}	The set of values of A appearing in dataset DS . Then A^{he} and A^{DB} denotes the A 's values appear in hyperedge he and whole database respectively.
$S_A^{DS}(x)$	Given $x \in A$ and dataset DS , it is the number of objects in DS having value x in A . Similar to A^{DS} , S_A^{he} and S_A^{DB} are defined respectively.

Table 4: The deviation value of some data records in example 1

RID	Name	Age-range	Car-type	Salary-level	$Dev^{he}(o, Car - type)$
3	Mary	Young	Sedan	High	-0.708
5	Frank	Young	Sports	High	0.708
6	Linda	Young	Sports	Low	0.708
8	Sam	Young	Sports	Low	0.708
10	Gary	Young	Sports	Low	0.708

tribute A w.r.t. he is defined as $Dev^{he}(o, A) = \frac{S_A^{he}(x_o) - \mu_{S_A^{he}}}{\sigma_{S_A^{he}}}$, where $\mu_{S_A^{he}} = \frac{1}{\|A^{he}\|} \sum_{x \in A} S_A^{he}(x)$ is the average value of $S_A^{he}(x)$ for all $x \in A^{he}$, and $\sigma_{S_A^{he}} = \sqrt{\frac{1}{\|A^{he}\|} \sum (S_A^{he}(x) - \mu_{S_A^{he}})^2}$ is the standard deviation of $S_A^{he}(x)$ for all $x \in A^{he}$.

Definition 3. Given a hyperedge he , a data object o in it is defined as an **outlier with common attributes C and outlying attribute A** , in which C is the set of attributes that have values appear in the frequent itemset corresponding to he , if $Dev^{he}(o, A) < \theta$.

The threshold of deviation θ determines how abnormal the outliers will be. Usually, θ is set to a negative value.

Example 3. Let us revisit the dataset in Example 1. When analyzing the second hyperedge, which contains objects 3, 5, 6, 8, and 10, we calculate the deviation of them on attribute $Car - type$. The result is shown in Table 4. According to the outlier definition above, object 3 is discerned as an outlier, for the deviation value of it is -0.708, while that of other data records in the hyperedge is 0.708. We can also give reasonable explanation to the phenomenon, that is, data records with $Age - range = 'Young'$ usually have the attribute value $Car - type = 'Sports'$, but the third object is different with the attribute value $Car - type = 'Sedan'$, so it is an outlier in the hyperedge. Although characteristics of these objects are similar when observed from all three attributes, the third one is totally different with other four objects when we study them from the $Age - range$ angle.

From this example, we find that although the objects are always sparse in the whole space, and seem to have no special characteristics from that point of view, some of them are anomalies when observed from certain viewpoint. We argue that the hyperedges are the appropriate viewpoints for observing outliers. Firstly, since the itemsets forming hyperedges are frequent, the objects falling in each of them construct a quite large group. Note that outliers are anomalies according to *common* objects. The size of the group, which equals to the support of the corresponding itemset, guarantees that the objects in it are *common* from certain view. Secondly, a hyperedge determines not only the locality of objects but also the dimensions. In other words, it determines a dense subspace in the whole data space. Current research has proved that subspace-based approach is useful for high-dimensional data. At last, since only part of the dimensions are considered in each hyperedge, the objects with missing-values can also be examined in the hyperedges that are not related to the attribute its missing values belong to. The hyperedge is robust to incomplete data.

Furthermore, the example shows that discriminating the common attributes and outlying attributes when finding outliers is important for searching anomalous objects. Therefore, in this paper, we study the problem of *given the minimum support threshold min_sup and the deviation threshold θ , finding all outliers according to definition 3.*

3. ALGORITHM

In this section, we will introduce a bottom-up algorithm using multi-dimensional array.

3.1 Basic Algorithm

The main algorithm for mining outliers is shown in Figure 1. The process for mining outliers can be roughly divided into three steps for each outlier, as we will introduce one by one in the follows.

STEP 1. Building the hierarchy of the hyperedges

The line 1 and 2 of main algorithm find the frequent itemsets, in which items are attribute values, and build the hierarchy of them. For one k -frequent-itemset I_k and one $(k+1)$ -frequent-itemset I_{k+1} , if $I_k \subset I_{k+1}$, then I_k is I_{k+1} 's ancestor in the hierarchy. And, i -frequent-itemset is in i th level. We employ Apriori [5] for finding the frequent itemsets. Note that Apriori tests all subsets of I_{k+1} , including I_k , when finding I_{k+1} [5]. Our algorithm just records the subset relationships, so that the two steps are integrated together.

STEP 2. Constructing multidimensional array

For each frequent itemset $I = \{A_1 = a_1, A_2 = a_2, \dots, A_p = a_p\}$, we construct a multi-dimensional array M , whose dimensions are attributes other than A_1, A_2, \dots, A_p , and coordinates are the identities of values of corresponding attributes. Each entry in the array is the count of objects falling in the hyperedge whose attribute values are equal to

```

PROCEDURE HypergraphBasedOutlierTest
INPUT:  $DB, min\_sup, \theta$ 
OUTPUT: Ouliers:  $(o, C, A)$ 
//  $C$  is the set of common attributes
//  $A$  is the outlying attribute

1. Mine_frequent_itemsets( $DB, min\_sup$ );
2. Build hypergraph and construct the hierarchy;
3. FOR EACH node  $i$  in level 1
4.   Construct multi-dim. array  $M_i$ ;
5. FOR level  $l = 2$  TO  $n$ 
6.   FOR EACH node  $i$  in level  $l$  BEGIN
7.     Choose one of  $i$ 's ancestor  $j$ ;
8.     Construct multi-dim. array  $M_i$  from  $M_j$ ;
9.     FindOutlier( $he_i, M_i, \theta$ );
10.  ENDFOR
ENDPROCEDURE

```

Figure 1: HOT algorithm

the coordinates respectively. More formally, the entry of the array, named as *amount* in the rest of paper, according to frequent itemset I above with coordinates $(a_{p+1}, a_{p+2}, \dots, a_k)$ is $\|\{o | o.A_i = a_i, i = 1, 2, \dots, k\}\|$, in which $A_i, i = p + 1, \dots, k$ are the attributes that have no value appear in I .

Assume that i and j are two nodes in the hierarchy, and j is one of i 's ancestor, which means $j \subset i$, and $\|i\| = \|j\| + 1$. M_i and M_j are their multi-dimensional arrays respectively. M_j is stored in a table $(A'_1, A'_2, \dots, A'_{k-p}, amount)$, in which $i - j = \{A'_{k-p} = a\}$, and M_i will be stored in table $(A'_1, A'_2, \dots, A'_{k-p-1}, amount)$. Then, we get M_i from M_j like this:

```

SELECT  $A'_1, A'_2, \dots, A'_{k-p-1}, sum(amount)$ 
INTO  $M_i$ 
FROM  $M_j$ 
WHERE  $A'_{k-p} = a$ 

```

STEP 3. *Finding outliers in the array*

Given a multi-dimensional array, the process to traverse the array to find outliers is shown in Figure 2. For each dimension, it calculates the occurrence of each value (line 3). Then, the deviation of each value is tested, so that outliers are found.

HEURISTIC 1. *When choosing ancestors to generate multidimensional array (line 7 of Figure 1), we choose the ancestor with minimum records.*

Although any ancestor of a node can be chosen for computing the multi-dimensional array, using the smallest one is

```

PROCEDURE FindOutlier
INPUT:  $he, M, \theta$ 
OUTPUT:  $(o, C, A)$ 

1. Set  $C$  as the set of attributes forming  $he$ ;
2. FOR EACH dimension  $A_i$  in  $M$  BEGIN
3.   Calculate  $S_{A_i}^{he}(v)$  for each value  $v$  in  $A_i^{he}$ ;
4.   Calculate  $\mu_{S_{A_i}^{he}}$ ;
5.   FOR EACH value  $v$  in  $A_i^{he}$ 
6.     IF  $Dev^{he}(v, A_i) < \theta$ 
7.       FOR EACH object  $o \in he$  with  $v$  in  $A_i$ 
8.         Output  $(o, C, A_i)$ ;
9.   ENDFOR
ENDPROCEDURE

```

Figure 2: Finding outliers in array

the most efficient choice. This is because that it only need linear scan on the multi-dimensional array of the ancestor node to get the new one. Therefore, choosing the smallest one can reduce the entries to be examined, and minimize the I/O cost.

HEURISTIC 2. *If both i and j are nodes in the hierarchy, and j is an ancestor of i (so, i has one more item $A = a$ than j), then, when finding outliers in the hyperedge corresponding to j , we don't execute the test of line 6 in Figure 2.*

Since any object o in i has quite a lot of similar objects on attribute A in j , for i is frequent, they may not be outliers with outlying attribute A . In all of our experiments, whether using this heuristic doesn't affect the outliers found.

3.2 Analysis

The performance analysis for frequent itemset mining using Apriori is given in [5]. To construct a multi-dimensional array for node i from that of node j , the complexity is $O(n_j)$, in which n_j is the number of entries in multi-dimensional array of j .

The time complexity of each step in the outer iteration of Figure 2 is shown in Table 5. The outer iteration will be executed at most $(k - p)$ times, where k is the dimensionality of the database, and p is the number of itemsets corresponding to the hyperedge. Note that in each hyperedge, n_i is always larger than any $\|A^{he}\|$, and smaller than $\|he\|$, the total time complexity is $O(\|HE\| \cdot k \cdot \max\{\|he\|\})$, where $\|HE\|$ is the number of hyperedges found by Apriori, after the hyperedges are found. The algorithm needs to store the multi-dimensional array for each hyperedge. Therefore, the space complexity is $O(\|HE\| \cdot k \cdot \max\{\|he\|\})$.

Table 5: Time complexity for finding outliers in multi-dimensional array according to certain he

Process	Time complexity	Notes
Calculate $S_{A_i}^{he}(v)$	$O(n_i)$	n_i denotes the number of entries in multi-dimensional array of i , $n_i < \ he\ $
Calculate $\mu_{S_{A_i}^{he}}$	$O(\ A_i^{he}\)$	
Test outliers	$O(\ A_i^{he}\)$	

3.3 Enhancement for Real Applications

The datasets in real-life applications are usually complex. They have not only categorical data but also numeric data. Sometimes, they are *incomplete*, which means some data values are missing. In most cases, the datasets are very large, for example, containing 1,000,000 objects, even one percent outliers will have 10,000 objects! Therefore, explanation and postprocessing are important. Furthermore, in real applications, expert knowledge is usually a valuable resource. How to utilize it should also be considered. In this section, we discuss the techniques for handling data with these characteristics in HOT.

3.3.1 Handling numeric data

To process numeric data, we apply the widely used binning technique [15], since it is the only available popular technique for discretizing numeric data in unsupervised condition. Our purpose is to find outliers, so we choose equal-width method, although it is not preferred in some environments for its sensitiveness to outliers. Furthermore, we apply another heuristic to make sure the bins are enough for discriminating the different characteristics of different objects.

HEURISTIC 3. *The number of bins is set to the maximum cardinality of all categorical attributes.*

Although, other preprocessing technique can also be combined with HOT for handling numeric data. Expert knowledge may be helpful as well. Since our paper focus on handling categorical data, we omit the details here.

3.3.2 Handling missing-value data

HOT does not need additional techniques for handling missing-value. It is robust to incomplete data, since the relationships between two objects are tested on attributes one by one, instead of on distance between two objects, like in most other outlier mining algorithms. An incomplete object will not be considered when the attribute containing the missing-value is tested. However, it may still be found as outliers with other outlying attributes. Meanwhile, this object is also considered in hyperedges it falls in, so that it still contributes in finding other outliers that have common attribute values with it.

3.3.3 Existing knowledge integration

HOT can take advantage of two kinds of expert knowledge: *horizontal* and *vertical*. Horizontal knowledge means the information of grouping of objects. The groups of objects can be added into the hypergraph as new hyperedges or even just use these groups as hyperedges. Therefore, similar objects are defined by existing knowledge. However, at that time, common attributes do not exist anymore. But the outlying attributes are still available. Vertical knowledge means the information of interested attributes. The values in attributes that are interested by experts or users can be viewed as *class labels*, and these attributes as *class attributes*. Then, the HOT changes from an *unsupervised* algorithm to a *supervised* one. Only the class attributes values are tested in the algorithm.

3.3.4 Pruning banal outliers

Outliers found by HOT may be not of equal interest. Some outliers are consistent with commonsense - the "already-known" exceptions. Others are out of the expectation of the experts and are more valuable because they may give us some novel knowledge. Following example demonstrates the condition.

Example 4. Suppose that in a 1000-record customer information dataset, only five customers have Rolls-Royce motor cars. Since the occurrence is low, this kind of data records will be identified as outliers in most hyperedges. They are trivial outliers because they are accord with the general knowledge, that is, owners of Rolls-Royce are sure very exceptional, and they form a small cluster in the whole dataset. Now assume there are 500 customers having Sedan cars in the same dataset. It's not a small cluster and we would not expect them to be outliers. However, in a specific 500-record hyperedge, the number of data records with $Car - type = 'Sedan'$ is only five, and are discerned as outliers. They are interesting since they deviate greatly from our expectation.

To distinguish the two kinds of outliers, we give the concept of *degree of interest* first.

Definition 4. Given a hyperedge he and the outlier o in it with common attributes C and outlying attribute A , $\frac{\|he\| \cdot S_A^{he}(x_o)}{N \cdot S_A^{DB}(x_o)}$ is called the **degree of interest of outlier o with common attributes C and outlying attribute A** , denoted as $Doi_{he}(o, A)$.

Then, the formal definition of *banal outliers* and *interesting outliers*, for distinguishing trivial and unexpected outliers, is given as follows.

Definition 5. Given an outlier o in hyperedge he with outlying attribute A , if $Doi_{he}(o, A) \geq \delta$, where δ is the threshold of interest, in he o is an **interesting outlier according to A** , otherwise it is a **banal outlier according to A** .

Obviously, $\frac{\|he\|}{N \cdot S_A^{DB}(x_o)}$ is the reciprocal of the expectation of data records having $A = x_o$ in hyperedge he , according to

the theory of probability. After multiplying with $S_A^{he}(x_o)$, which is the actual number of data records having $A = x_o$ in hyperedge he , the value can be regarded as a measurement of the difference between reality and estimation. The larger the value is, the more surprising the outlier is. According to users' information request, a pruning process can be integrated easily into HOT algorithm to filter *banal outliers*, as postprocessing.

3.3.5 Mining result explanation

HOT provides sufficient information for explaining an outlier: common attributes, outlying attribute and deviation. Therefore, the objects similar to the outlier in the common attributes are easy to be retrieved for further study. Meanwhile, the values in outlying attributes can be provided to users. This kind of knowledge is useful in applications such as data cleaning, since users may be interested in knowing not only the anomalous object but also the outlying attribute values, so that *dirty* attribute values can be found. Furthermore, the deviation values can help users to find possible correct values.

4. EXPERIMENTAL RESULT

4.1 Experimental Setup

The experiments are tested on a Pentium 4 PC workstation with 256MB RAM running Microsoft Windows 2000.

To test the ability of our method for finding outliers, some experiments are done based on the Mushroom and Flag datasets obtained from the UCI repository [20], which includes data sets designed for classification and machine learning applications. We choose these two datasets for three reasons. First, they both contain categorical data, which is our main aim to handle. Each attribute may contain two to several possible values, on which no measurement can be defined except equal. Second, they are both high-dimensional data, especially when considering the dimensionality vs. the number of objects in the dataset. Last but not the least, mushroom dataset contains unknown attribute values, which is a restrict condition for other outlier mining algorithms' applying on. The first two conditions are related to *curse of dimensionality* and *combinational explosion*. The last one is the well known *incomplete data* problem. All these three problems make the problem of mining outliers in such kind of datasets difficult. We will first present some interesting outliers found by our method, then use quantified measurements to show why the outliers found are interesting.

4.2 Experiments on Real-life Data

We test the effectiveness of our algorithm on the Mushroom and Flag datasets and detect numerous outliers. Actually, all outliers found are rather deviant in some respect, but it is impractical to exhibit them one by one here. So only some outlier examples will be presented.

Mushroom Dataset Mushroom dataset is an 8124-record dataset, which includes descriptions of hypothetical samples of gilled mushrooms in the Agaricus and Lepiota Family. Each mushroom is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. Besides the classification propriety, there are other

Table 6: The deviation value of data records in hyperedge he corresponding to frequent itemset $\{Veil - type = 'p', Ring - type = 'p'\}$ in mushroom dataset

Cap-surface	Number of occurrence	$Dev^{he}(o, Cap - surface)$
'f'	1248	0.22
'g'	4	-0.85
's'	1212	0.19
'y'	1504	0.44

Table 7: The deviation value of data records in hyperedge he corresponding to $\{Bruises = 'f', Gill - attachment = 'f', Gill - spacing = 'c', Veil - type = 'p'\}$ in mushroom dataset

Edibility	Number of Occurrence	$Dev^{he}(o, Edibility)$
'p'	3170	0.71
'e'	160	-0.71

22 attributes for each mushroom, such as cap-shape, odor, gill-attachment, stalk-shape etc.. All attributes are nominally valued and there are some missing values among them, which are denoted by "?".

When finding frequent itemsets, the minimal support is set to 40 percent and 565 frequent itemsets are generated.

In one test, we set the threshold of deviation to -0.849 to discover outliers deviating greatly from other data. We find that among 3968 records satisfying the frequent itemset $\{Veil - type = 'p', Ring - type = 'p'\}$, only 4 records have the attribute value $Cap - surface = 'g'$, as shown in Table 6. That is to say, most mushrooms with partial veil and pendant ring will not have grooves-like cap surface. But there are 4 mushrooms that do have partial veil, pendant ring, and grooves-like cap surface. So they are detected as outliers.

When the threshold is set to -0.7, another interesting kind of outliers we find is as following. Totally, there are 3330 records comply the frequent itemset $\{Bruises = 'f', Gill - attachment = 'f', Gill - spacing = 'c', Veil - type = 'p'\}$, most of which are poisonous. But there are 160 mushrooms of this kind are edible and so are regarded as outliers. Table 7 illustrates the condition clearly. These outliers are not only interesting but also useful. When the knowledge is applied to practice, it will gain much benefit.

Flag Dataset Flag dataset contains details of 194 nations and their flags. There are overall 30 attributes, such as name, landmass, religion, color in the top-left corner of the flag etc., 10 of which are numeric-valued, others are either Boolean- or nominal-valued. In the experiment, we ignore numeric-valued attributes, since our purpose is to test the ability of the algorithm for processing categorical data.

We set the minimal support to 60 percent when finding frequent itemsets. 75 frequent itemsets are found, and accordingly, there are 75 hyperedges in the hypergraph.

When the threshold of deviation is set to -0.71, 13 data objects are detected as outliers, which belong to two types. One kind is that most countries (111 countries in the Flag dataset), whose flags have white color but have neither crescent moon symbol nor letters, are not located in South America. However, we find 11 countries having these attributes are South American countries. The other kind is that gold color appears in two countries' flags which have no black or orange colors, while it is not present in the flags of other 120 countries without black or orange colors.

4.3 Quantitative Empirical Study

In this subsection, we will show the quantized analysis of the outliers found by HOT. First of all, we will define two kinds of densities.

Definition 6. In a database DB , for an object $o(v_o^1, v_o^2, \dots, v_o^k)$, its **density in whole space** is defined as $density_o^{all} = \|\{p|p \in DB, v_p^i = v_o^i, i = 1, \dots, k\}\|/\|DB\|$. Given an attribute set $B = \{A_{i_1}, A_{i_2}, \dots, A_{i_l}\} \subseteq A$, o 's **subspace density over B** is defined as $density_o^B = \|\{p|p \in DB, v_p^{i_j} = v_o^{i_j}, j = 1, 2, \dots, l\}\|/\|DB\|$.

We will study an outlier o in hyperedge he with common attributes C and outlying attribute A_o from following four aspects:

- o 's density in whole space vs. minimum, maximum, average density in whole space for all $p \in DB$;
- o 's density in whole space vs. minimum, maximum, average density in whole space for all $p \in he$;
- o 's subspace density over $C \cup \{A_o\}$ vs. minimum, maximum, average subspace density over $C \cup \{A_o\}$ for all $p \in DB$;
- o 's subspace density over $C \cup \{A_o\}$ vs. minimum, maximum, average subspace density over $C \cup \{A_o\}$ for all $p \in he$;

The experiments still run on mushroom dataset, while the threshold of deviation is set to -0.84, and 32 outlier types are found as well as 76 outliers. In the test dataset, the density in whole space of any object is 1/8124. This result is consistent with our foresight. Since in high-dimensional data with categorical attributes, most data object is distinct. This also supports our conclusion that full-space outlier mining algorithm is ineffective for this kind of data, since every data object or no object is outlier according to those definitions.

Figure 3 shows the result for comparison of subspace density between outliers and all data. Remember that results we get by HOT algorithm are data records with outlying attribute in a certain hyperedge. They are actually a kind of data records and we call the characteristics shared by them *outlier class*. The x -axis denotes the 32 outlier classes found by HOT, the y -axis denotes the logarithmic subspace density. We can see that although outliers found by HOT always have very low subspace density relatively in the database,

they are not always the most *sparse* ones. There are twelve outlier classes (No. 1, 3, 4, 9, 10, 27, 28, 29, 30, 31, 32 in the result) have higher subspace density values than that of some other objects in the database over the same attributes. After examining the result, we found that this situation happens for two reasons:

1. Some objects with very low subspace density over some attributes even do not appear in any hyperedges. Therefore, they are not similar to large amount of other data. Note that we want to find relative outliers. These data can be treated as noises, since the values in any attribute of them are rare.
2. Some objects appear in certain hyperedges, but they do not have outlying attributes. This means that in those hyperedges, most distinct values in attributes other than common ones occur few times. Therefore, no object is special compared to other objects *similar* to it. Although they are *sparse* in the whole database over those attributes, this only means that they form a cluster over certain attributes (e.g. the common attributes).

Note that our purpose is to find objects having anomalous attribute value while be similar to a large group of other data in some other attributes. These two kinds of data objects should not be treated as outliers.

Figure 4 shows the result for comparison of subspace density between outliers and all data in the hyperedge the outlier falling in. Different with the result that compared to the data in whole database, outliers' subspace densities are always the lowest ones in the hyperedge. This property is guaranteed by our definition and algorithm. It ensures that the most anomalous data can be found in each local dense area.

In the above experiments, it is found that when the threshold of deviation is not very negative, large amount of outlier classes as well as outliers will be found. It is still hard for users to browse so many outliers and find information valuable to them. Actually, some of the outliers are not so interesting to users, which are banal outliers we defined in section 3.3.4. Fortunately, with the threshold of interest, we can prune banal outliers in advance and only keep interesting outliers in the result.

Experiments are done on Mushroom dataset with the threshold of interest set to 3.0, to filter those outliers with $Doi_{he}(o, A)$ smaller than 3.0. Table 8 shows the number of outlier classes and outliers after pruning compared to those without pruning, which indicates that our algorithm is efficient in finding and pruning banal outliers.

5. RELATED WORK

Outlier detection is firstly studied in the field of statistics [6, 12]. Many techniques have been developed to find outliers, which can be categorized into distribution-based [6, 12] and depth-based [22, 24]. However, recent research has proved that these methods are not suitable for data mining applications, since that they are either ineffective and inefficient

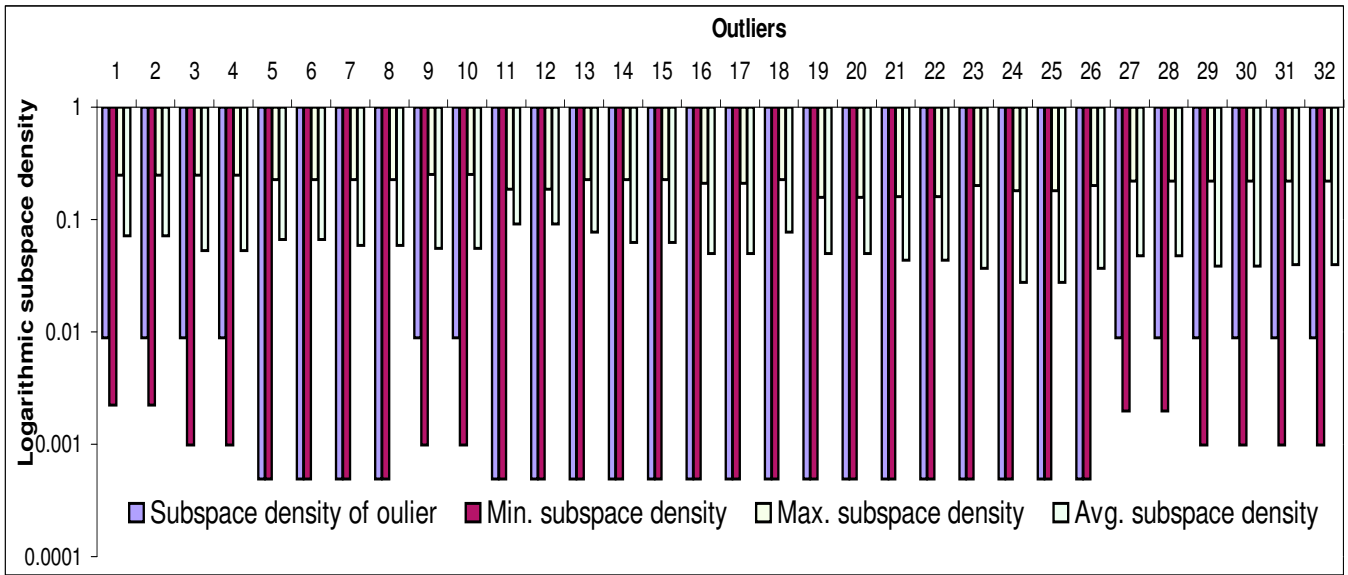


Figure 3: o 's subspace density over $A \cup \{A_o\}$ vs. minimum, maximum, average subspace density over $A \cup \{A_o\}$ for all $p \in DB$

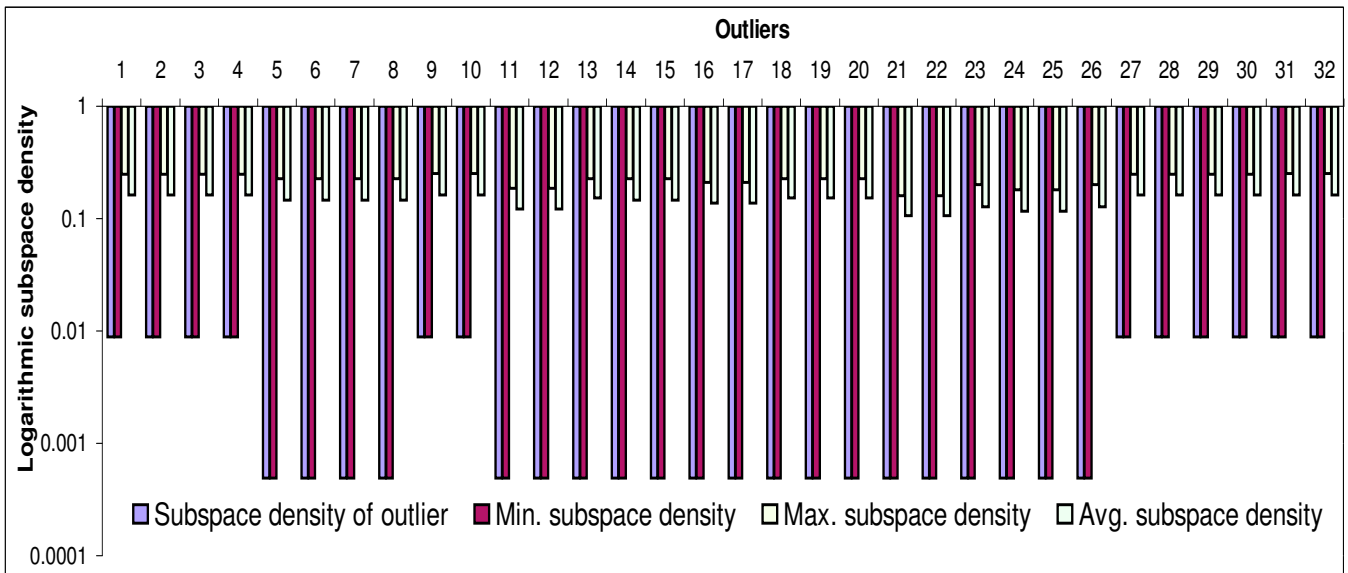


Figure 4: o 's subspace density over $A \cup \{A_o\}$ vs. minimum, maximum, average subspace density over $A \cup \{A_o\}$ for all $p \in he$

Table 8: Banal outliers pruning ratio

θ	Without pruning		With pruning		Pruning ratio
	Num of outlier classes	Num of outliers	Num of outlier classes	Num of outliers	
-0.75	637	5856	130	676	88.46%
-0.77	569	5200	126	668	87.15%
-0.79	415	4756	108	620	86.96%
-0.81	269	2484	90	572	76.97%

for multidimensional data, or need a priori knowledge about the distribution underlying the data [18].

Traditional clustering algorithms focus on minimizing the affect of outliers on cluster-finding [21, 10, 28, 27, 11, 25, 13, 17]. Outliers as well as noises are eliminated without further analyzing in these methods, so that they are only by-products which are not cared about.

In recent years, outliers themselves draw much attention, and outlier detection is studied intensively by the data mining community [18, 19, 8, 23, 9, 16, 26, 1]. **Distance-based outlier** detection is to find data that are sparse within the hyper-sphere having given radius [18]. Researchers also developed efficient algorithms for mining distance-based outliers, such as cell-based algorithm for disk-resident data [18], and partition-based algorithm [23]. However, since these algorithms are based on distance computation, they may fall down when processing categorical data or datasets with missing values. In fact, the heuristics used in these algorithms are inapplicable for high-dimensional datasets even all attributes are numerical and contain no missing value, that is caused by *curse of dimensionality* [7].

Graph-based spatial outlier detection is to find outliers in spatial graphs based on statistical test [26]. However, both the attributes for locating a spatial object and the attribute value along with each spatial object are assumed to be numeric, which restricts the application of the efficient algorithm on datasets with categorical attributes.

In [8] and [9], the authors argue that the outlying characteristics should be studied in local area, in which data points usually share similar distribution property, namely density. This kind of methods is called **local outlier** detection. Both efficient algorithms [8, 16] and theoretical background [9] have been researched for local outliers. However, it should be noted that in high-dimensional space, data are almost always sparse, so that density-based methods may suffer the problems that all data points are outliers or none of them is outlier. Similar condition holds in categorical datasets. Furthermore, the density definition employed also bases on distance computation. As the result, it is inapplicable for the condition datasets having missing values, which is usually true in real data mining applications.

Multi- and high-dimensional data make the outlier mining problem more complex because of the impact of *curse of dimensionality* on algorithms' both performance and effectiveness. In [19], Knorr and Ng tried to find the smallest attributes to explain why an object is exceptional, and is it

dominated by other outliers. These information are called *intensional knowledge* of the outliers. Different strategies to scan the value lattice are analyzed and evaluated so that efficient algorithm is developed. However, Aggarwal and Yu argued that this approach may be expensive for high-dimensional data [1]. Therefore, they proposed a definition for outliers in low-dimensional projections and developed an evolutionary algorithm for finding outliers in projected subspaces. Both of these two methods consider the outlier in global. The sparsity or deviation property is studied in the whole dataset, so that they cannot find outliers relatively exceptional according to the objects near it. Moreover, as other existing outlier detection methods, they are both designed for numeric data, and cannot handle dataset with missing values.

Analyzing properties of high-dimensional data in subspaces is a general approach for overcoming *curse of dimensionality* that has been widely applied in clustering [4, 2, 14, 3]. However, these works focus on finding patterns for large amount of data. As other clustering methods, they eliminate or ignore outliers with noises to achieve robustness. However, the success of these algorithms proves that different sets of data may have different grouping property in different subsets of dimensions. This conclusion supports our choice of the approach to find outliers.

6. CONCLUSION AND FUTURE WORK

In this paper, we present a novel definition for outliers that captures the local property of objects in partial dimensions. This definition has the advantages that 1) it can process categorical data effectively, since it overcomes the *curse of dimensionality* and *combinational explosion* problems; 2) it is robust to incomplete data, for its independence to traditional distance definition; 3) the knowledge, which includes common attributes, outlying attribute, and deviation, is provided along with the outliers, so that the mining result is easy for explanation. Therefore, it is suitable for modeling anomalies in real applications, such as fraud detection or data cleaning for commercial data. Both the algorithm for mining such kind of outliers and the techniques for applying it in real-life dataset are introduced. Furthermore, a method for analyzing outlier-mining results in subspaces is developed. By using this analyzing method, our experimental result shows that HOT can find interesting, although may not be most sparse, objects in subspaces. Both qualitative and quantitative empirical studies support the conclusion that our definition of outliers can capture the anomalous properties in categorical and high-dimensional data finely. To the best of our knowledge, this is the first trial to find outliers in categorical data.

Current work uses pre- and post-processing for handling numeric data and finding interesting outliers. Our future work includes integrating these two processes into the algorithm, so that the algorithm can be more efficient.

7. REFERENCES

- [1] C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 37–47. ACM Press, 2001.

- [2] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 61–72. ACM Press, 1999.
- [3] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 70–81. ACM Press, 2000.
- [4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 94–105. ACM Press, 1998.
- [5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of 20th Int'l Conf. on Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.
- [6] V. Barnett and T. Lewis. *Outliers In Statistical Data*. John Wiley, Reading, New York, 1994.
- [7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful? In *Proc. of 7th Int'l Conf. on Data Theory*, pages 217–235. Springer, 1999.
- [8] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. Optics-of: Identifying local outliers. In *Proc. of 3rd European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 262–270. Springer, 1999.
- [9] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 93–104. ACM Press, 2000.
- [10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.
- [11] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 73–84. ACM Press, 1998.
- [12] D. Hawkins. *Identification of Outliers*. Chapman and Hall, Reading, London, 1980.
- [13] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. of 4th Int'l Conf. on Knowledge Discovery and Data Mining*, pages 58–65. AAAI Press, 1998.
- [14] A. Hinneburg and D. A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. of 25th Int'l Conf. on Very Large Data Bases*, pages 506–517. Morgan Kaufmann, 1999.
- [15] F. Hussain, H. Liu, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Technical Report TRC6/99, National University of Singapore, School of Computing*, 1999.
- [16] W. Jin, A. K. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proc. of ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 293–298. ACM Press, 2001.
- [17] G. Karypis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computing*, 32(8):68–75, 1999.
- [18] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. of 24th Int'l Conf. on Very Large Data Bases*, pages 392–403. Morgan Kaufmann, 1998.
- [19] E. Knorr and R. Ng. Finding intensional knowledge of distance-based outliers. In *Proc. of 25th Int'l Conf. on Very Large Data Bases*, pages 211–222. Morgan Kaufmann, 1999.
- [20] G. Merz and P. Murphy. Uci repository of machine learning databases. *Technical Report, University of California, Department of Information and Computer Science*: <http://www.ics.uci.edu/mllearn/MLRepository.html>, 1996.
- [21] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of 20th Int'l Conf. on Very Large Data Bases*, pages 144–155. Morgan Kaufmann, 1994.
- [22] F. Preparata and M. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, Reading, New York, 1988.
- [23] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 427–438. ACM Press, 2000.
- [24] I. Ruts and P. Rousseeuw. Computing depth contours of bivariate point clouds. *Journal of Computational Statistics and data Analysis*, 23:153–168, 1996.
- [25] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proc. of 24th Int'l Conf. on Very Large Data Bases*, pages 428–439. Morgan Kaufmann, 1998.
- [26] S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers: Algorithms and applications (a summary of results). In *Proc. of ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*. ACM Press, 2001.
- [27] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proc. of 23rd Int'l Conf. on Very Large Data Bases*, pages 186–195. Morgan Kaufmann, 1997.
- [28] T. Zhang, R. Ramakrishnan, and M. Linvy. Birch: An efficient data clustering method for very large databases. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 103–114. ACM Press, 1996.