A Scalable and Efficient Outlier Detection Strategy for Categorical Data

A. Koufakou[1] E.G. Ortiz[1] M. Georgiopoulos[1] G.C. Anagnostopoulos[2] K.M. Reynolds[3]
[1]*University of Central Florida, School of EECS, Orlando, FL*
Emails: akoufako@mail.ucf.edu, eortiz@cs.ucf.edu, michaelg@mail.ucf.edu
[2]*Florida Institute of Technology, Dept of ECE, Melbourne, FL - Email: georgio@fit.edu*
[3]*University of Central Florida, Dept of Criminal Justice, Orlando, FL-Email: kreynold@mail.ucf.edu*

Abstract

Outlier detection has received significant attention in many applications, such as detecting credit card fraud or network intrusions. Most existing research focuses on numerical datasets, and cannot directly apply to categorical sets where there is little sense in calculating distances among data points. Furthermore, a number of outlier detection methods require quadratic time with respect to the dataset size and usually multiple dataset scans. These characteristics are undesirable for large datasets, potentially scattered over multiple distributed sites. In this paper, we introduce Attribute Value Frequency (AVF), a fast and scalable outlier detection strategy for categorical data. AVF scales linearly with the number of data points and attributes, and relies on a single data scan. AVF is compared with a list of representative outlier detection approaches that have not been contrasted against each other. Our proposed solution is experimentally shown to be significantly faster, and as effective in discovering outliers.

1. Introduction

Mining for outliers in data is an important research field with many applications in credit card fraud detection, discovery of criminal activities in electronic commerce, and network intrusion detection. Outlier detection approaches focus on discovering patterns that occur infrequently in the data, as opposed to traditional data mining techniques that attempt to find patterns that occur frequently in the data. One of the most widely accepted definitions of an outlier pattern is provided by Hawkins [1]: "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism".

Outliers are frequently treated as noise that needs to be removed from a dataset in order for a specific model or algorithm to succeed (e.g. points not belonging in clusters in a clustering algorithm). Alternatively, outlier detection techniques can lead to the discovery of important information in the data ("one person's noise is another person's signal") [2]. Finally, outlier detection strategies can also be used for data cleaning before any traditional mining algorithm is applied to the data. Examples of data where the discovery of outliers is useful include irregular credit card transactions, indicating potential credit card fraud [3], or patients who exhibit abnormal symptoms due to their suffering from a specific disease or ailment [4].

Most of the research efforts in outlier detection strategies have focused on datasets that are comprised of numerical attributes, or ordinal attributes that can be directly mapped into numerical values. Quite often, when we have data with categorical attributes it is assumed that the categorical attributes could be easily mapped into numerical values. However, there are cases of categorical attributes, where mapping to numerical attributes is not a straightforward process, and the results greatly depend on the mapping that is used, e.g., the mapping of a marital status attribute (married or single) or a person's profession (engineer, financial analyst, etc.) to a numerical attribute.

Recently there has been some focus on data with categorical or mixed attributes (e.g. He et al. [5],[6],[7], and Otey et al. [8]). However these efforts have not been contrasted to each other using the same data. In this paper, we explore these methods and evaluate them using the same datasets with regard to their efficiency (speed), scalability, and effectiveness (accuracy) in detecting outliers in categorical data.

Another issue that has only recently become the focus in the literature is related to the large and distributed nature of the datasets available today. With the explosion of technology, the size of data for a particular application has grown and will continue to grow. Also, most of the data is distributed among different sites belonging to the same or even different organizations. Transferring data to a central location and then detecting outliers is typically impractical due to data size and expense of moving it, as well as data ownership and control issues. Hence, successful outlier detection strategies must scale well as the size and dimensionality of the dataset grow.

Furthermore, in order to deal with the large and possible distributed nature of the data, the dataset scans should be minimal as well.

In this paper, we introduce an outlier detection strategy for categorical data, called *Attribute Value Frequency* (AVF). We compare AVF with existing outlier detection methods ([5],[6],[7],[8]) with respect to outlier detection speed and accuracy. AVF is experimentally shown to have a significant performance advantage, and to scale linearly as the data increases in points and dimensions. AVF does not depend on extra user parameters, and performs only one dataset scan, thus would be efficient for geographically distributed data.

The organization of this paper is as follows: In Section 2, we provide a thorough literature review. In Section 3, we describe our proposed algorithm, AVF, and the outlier detection algorithms mentioned above. Section 4 contains our experiments and results. Finally, in Section 5, we summarize our work and provide concluding remarks.

2. Previous Work

The earliest approaches to detect outliers, ***statistical-model based*** methods, assume that a parametric model describes the distribution of the data (e.g., normal distribution), and are mostly single-dimensional or univariate [9]. The limitations of these approaches include the difficulty to find the right model for each dataset and application, as well as the fact that their efficiency declines as the data dimensionality increases [8]. Another issue with high dimensional datasets is that the dataset become less dense, which makes the convex hull harder to determine ("Curse of Dimensionality")[10]. There are methods to help alleviate this problem, e.g. Principal Component Analysis. Another idea to handle higher dimensionality data is to organize the data points in layers, based on the idea that shallow layers tend to contain outliers more often than the deep layers (e.g. [10]), these ideas however are impractical for more than 2 dimensions.

Distance-based approaches do not make assumptions for the distribution of the data since they essentially compute distances among points. These approaches become impractical for large datasets (e.g. nearest neighbor method has quadratic complexity with respect to the dataset size). There have been improvements on the original distance-based algorithms, e.g. Knorr et al. [2], where an outlier is defined as an object O in a dataset T that has at least p% of the objects in T further than distance D from it. The complexity of their approach is still exponential in the number of nearest neighbors. Bay and Schwabacher [12] propose a distance-based method and claim its complexity is close to linear in practice.

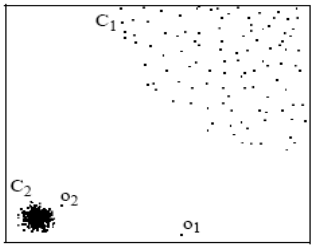


Figure 1. Distance-based methods miss outlier o2 [13]

Clustering techniques can be used with the idea that the points that do not belong in the formed clusters are designated as outliers. However, clustering-based methods are focused on optimizing clustering measures of goodness, and not on finding the outliers in the data [2].

Density-based methods estimate the density distribution of the data and identify outliers as those lying in low-density regions. Breunig et al. [13] assign a local outlier factor (*LOF*) to each point based on the local density of its neighborhood, which is determined by a user-given minimum number of points (*MinPts*). Papadimitriou et al. [14] present *LOCI* (Local Correlation Integral) which uses statistical values based on the data itself to tackle the issue of choosing values for *MinPts*. All density-based techniques have the advantage that they can detect outliers that would be missed by techniques with a single, global criterion, as shown in Figure 1. However, data is usually sparse in high-dimensional spaces rendering density-based methods problematic [15].

Other outlier detection efforts include Support Vector methods (e.g. [16]), using Replicator Neural Networks (RNNs) [17], and using a relative degree of density with respect only to a few fixed reference points [18].

All the aforementioned methods are geared towards numerical data and thus are more applicable to numerical datasets or ordinal data that can be easily transformed to suitable numerical values [10]. In the case of categorical datasets, there is little sense of ordering the data values, then mapping them to numerical values and computing distances (e.g., distance between two values such as TCP Protocol and UDP Protocol [8]). Consequently, methods such as those based on distance or density are unsuitable. Also many previous methods are quadratic in complexity with respect to the data size, which would be unacceptable for large datasets. If we assume a distributed setting, techniques relying on pair-wise distance computations become infeasible as the different sites would have to either exchange every local data point in order to calculate distances or replicate all data points globally.

In this paper, we implement and experiment with three current outlier detection approaches directed towards categorical data. The first is proposed by He et al. [7] and is based on the idea of Entropy. The second technique by Otey et al. [8] focuses on datasets with mixed attributes (both categorical and numerical). The third technique in

[6] is also by He et al. Both methods by He [6] and by Otey [8] compute an outlier score for each data point using the concept of frequent itemsets (see [19]). Wei et al. [15] also deal with outliers in categorical datasets and use frequent itemsets, as well: they use hyperedges, which simply store frequent itemsets along with the data points that contain these frequent itemsets; since their method is based on a premise similar to that in [6], it was not considered in our experiments. Finally, Xu et al. [20] use mutual reinforcement to discover outliers in a mixed attribute space; however their method focuses on a different outlier detection problem than the one described in this paper ("instead of detecting local outliers as noise, [they] identify local outliers in the center, where they are similar to some clusters of objects on one hand, and are unique on the other."[20]).

Table 1. Terminology

Term	*Description*
k	Target input number of outliers
n	Number of data points
m	Number of attributes for each data point
q	Number of Distinct values per attribute
l	A specific attribute (ranging from 1 to m)
x_i	The i-th point in the dataset
x_{il}	The l-th value of the i-th point in the dataset
D	Dataset
I	Itemset
F	Frequent Itemset
FIS	The Set of all Frequent Itemsets
Minsup	Minimum support for the frequent itemsets
support(I)	Support of itemset I

3. Algorithms

3.1. Greedy Algorithm

The algorithms in [5],[7] are based on the premise that outliers are likely the points that, once removed from the data, the dataset as a whole has less "uncertainty" or "disorder". In particular, they propose the idea of finding a small subset (of size k) of the data points that contribute to the elevated "disorder" of the dataset the most. The idea of Entropy, or uncertainty, of a random variable is attributed to Shannon [21]. Formally, if X is a random variable, $S(X)$ is the set of values that X can take, and $p(x)$ is the probability function of X, then the entropy, $E(X)$, can be defined as follows:

$$E(X) = -\sum_{x \in S(X)} p(x) \log_2 p(x) \qquad (1)$$

Given a multivariate vector, $\boldsymbol{X} = \{X_1, X_2, \ldots, X_m\}$ (or a multidimensional dataset containing m attributes) and the assumption that the attributes of $\boldsymbol{X}$ are independent, the Entropy of $\boldsymbol{X}$, $E(\boldsymbol{X})$, is equal to the sum of the entropies of each one of the m attributes, and is defined as follows:

$$E(\mathbf{X}) = E(X_1) + E(X_2) + \cdots + E(X_m) \qquad (2)$$

He et al. introduce a Local-Search heuristic-based Algorithm (LSA) in [5], and a Greedy Algorithm in [7], both relying on the entropy idea mentioned above. Since the Greedy algorithm [7] is superior to LSA [5], we will only discuss the Greedy algorithm. The Greedy algorithm [7] takes as input the desired number of outliers (k). All points in the set are initially designated as non-outliers. At the beginning, the frequencies of all attribute values are computed, as well as the initial entropy of the dataset. Then, Greedy conducts k scans over the data to determine the top k outliers. During each scan, every non-outlier is temporarily removed from the dataset and the total entropy is recalculated. The non-outlier point that results in the maximum decrease for the entropy of the entire dataset is the outlier data-point removed by the algorithm in each scan.

The complexity of Greedy is $O(k*n*m*q)$, where k is the target number of outlier points, n designates the number of points in the dataset, m is the number of attributes, and q is the number of distinct attribute values, per attribute. If the number of attribute values per attribute, q, is small, the complexity of Greedy becomes $O(k*n*m)$. Pseudocode for Greedy is provided in Figure 3.

3.2. Frequent Itemset Mining (FIM)-based Algorithms

In this section, we describe two algorithms based on the concept of Frequent Itemset Mining. *Frequent Itemset Mining (FIM)* has received considerable attention since the seminal paper on the related subject of *Association Rule Mining* by Agrawal and Srikant [19]. Given a dataset D and a set of r literals, $S = \{i_1, i_2, \ldots, i_r\}$ that are found in D, we can define an itemset I as a non-empty subset of S. For example, items in a supermarket could be "bread", "milk", etc; then a possible itemset I could be {"bread", "milk"}. Given a user-defined threshold called minimum support, *minsup*, a frequent itemset F is one that appears in the dataset at least *minsup* times. The set of all frequent itemsets given *minsup* is denoted by *FIS*. The support of an itemset I, designated as *support*(I), is the fraction of points in D that contain itemset I.

He et al. in [6] observe that, since frequent itemsets are "common patterns" that are found in many of the points of the dataset, outliers are likely to be the points that contain very few common patterns or subsets. They define a *Frequent Pattern Outlier Factor (FPOF)* for every data point, x, based on the support of the frequent itemsets contained in x. In addition, they use a *contradictness* score to better explain the outliers and *not*

to detect the outliers, so it is omitted from our discussion. The *FPOF* outlier score is calculated as follows:

$$FPOF \quad Score(x) = \frac{\sum \text{support}(F)}{\|FIS\|} \qquad (3)$$

The above score is the sum of the support of all the frequent subsets F contained in point x, over the number of all frequent sets in dataset D, *FIS*. Data points with lower FPOF scores are likely to be outliers since they contain fewer common subsets. The *FPOF* algorithm runs the Apriori algorithm [19] first, to identify all frequent itemsets in dataset D given *minsup* (*FIS*), then calculates the outlier score in (3) for each data point, to identify the top k outliers (see pseudocode in Figure 4).

The method by Otey et al. in [8] is also based on frequent itemsets. They assign to each point an anomaly score inversely proportionate to its infrequent itemsets. They also maintain a covariance matrix for each itemset to handle continuous attributes; we omit this part since our focus is on categorical data. Specifically, they calculate the following score for each data point x:

$$Otey's \quad Score(x) = \sum_{I \subseteq x, \sup(I) \le minsup} \frac{1}{|I|} \qquad (4)$$

which is explained as follows: given subsets I of x which are infrequent, i.e. support of I is less than *minsup*, the anomaly score of x will be the sum of the inverse of the length of the infrequent itemsets. If a point has few frequent itemsets, its outlier factor will be high, so the outliers are the k points with the maximum outlier score in (4). This algorithm also first mines the data for the frequent itemsets, then calculates an outlier score for each point (see pseudocode in Figure 5). The authors state that the execution time is linear to the number of data points, but exponential to the number of categorical attributes.

3.3. Attribute Value Frequency (AVF) Algorithm

The algorithms discussed thus far scale linearly with respect to the number of data points, n. However, Greedy (section 3.1) still needs k scans over the dataset to find k outliers, a disadvantage for the case of very large datasets that are potentially distributed among different sites. On the other hand, the Frequent Itemset Mining (FIM)-based approaches (section 3.2) need to create a potentially large space of subsets or itemsets, and then search for these sets in each and every data point. These techniques can become extremely slow for low values of the *minsup* threshold, as more and more candidate itemsets need to be examined. In this section we present a simpler and faster approach to detect outliers that minimizes the scans over the data and does not need to create or search through different combinations of attribute values or itemsets. We call this outlier detection algorithm *Attribute Value Frequency* (AVF) algorithm.

It is intuitive that outliers are those points which are infrequent in the dataset. Additionally, an 'ideal' outlier point in a categorical dataset is one whose each and every attribute value is extremely irregular (or infrequent). The *infrequent-ness* of an attribute value can be measured by computing the number of times this value is assumed by the corresponding attribute in the dataset.

Let's assume that the dataset contains n data points, x_i, $i = 1 \ldots n$. If each data point has m attributes, we can write $x_i = [x_{i1},\ldots,x_{il},\ldots,x_{im}]$, where x_{il} is the value of the l-th attribute of x_i. Following the reasoning given above, a good indicator or score to decide if point x_i is an outlier can be defined as the AVF Score below:

$$AVF \quad Score(x_i) = \frac{1}{m}\sum_{l=1}^{m} f(x_{il}) \qquad (5)$$

where, $f(x_{il})$ is the number of times the l-th attribute value of x_i appears in the dataset. A lower AVF score means that it is more likely that the point is an outlier. The choice for the AVF score can be justified by noting the following. Since in (5) we essentially have a sum of m positive numbers, the AVF score is minimized when each of the sum's terms are individually minimized. Thus for an 'ideal' outlier as defined above, the AVF score will be the minimum. Figure 2 contains an example with 12 points taken from UCI [22] Breast Cancer dataset. The outliers are in bold (points 7 and 12). As this example clearly illustrates, the outliers have infrequent values for all or most of the 9 attributes, compared to the rest of the data points.

Data	**Attributes**								
point	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	1	1	1	1	2	10	3	1	1
2	2	1	1	1	2	1	2	1	1
3	1	1	1	1	2	3	3	1	1
4	4	1	1	1	2	1	2	1	1
5	4	1	1	1	2	1	3	1	1
6	6	1	1	1	2	1	3	1	1
7	**7**	**3**	**2**	**10**	**5**	**10**	**5**	**4**	**4**
8	3	1	1	1	2	1	2	1	1
9	1	1	1	1	2	1	3	1	1
10	3	2	1	1	1	1	2	1	1
11	5	1	1	1	2	1	2	1	1
12	**2**	**5**	**3**	**3**	**6**	**7**	**7**	**5**	**1**

Figure 2. Example from UCI Breast Cancer Dataset

Once we calculate the AVF score of all the points, we designate the k points with the smallest AVF scores as the k outliers (see Figure 6 for AVF pseudocode). The complexity of AVF is $O(n*m)$ compared to Greedy's complexity, $O(k*n*m)$, since AVF detects outliers after only one scan of the dataset, instead of the k scans needed by Greedy.

```
Input:  Dataset – D
        Target number of outliers – k
Output: k detected outliers
Label all data points as non-outliers
calculate frequency of each attribute value
calculate initial entropy of dataset
while ( not k scans) do
  while (not end of database) do
    read next record x and label x as outlier
    calculate decrease in entropy by removing x
  end while
  if (maximal decrease achieved by record x)
    add x to set of outliers
  end if
end while
```

Figure 3. Greedy Algorithm Pseudocode

```
Input:  Dataset – D
        Minimum support – minsup
        Target number of outliers – k
Output: k detected outliers
FIS = Mine for frequent item sets (D, minsup)
foreach point x in D
  foreach frequent pattern F in FIS
    if x contains F
      FPOFoutlierScore(x) += support(F)/ length(FIS)
    end if
  end for
end for
return top k outliers that minimize FPOFoutlierScore
```

Figure 4. FPOF Algorithm Pseudocode

```
Input:  Dataset – D
        Minimum support – minsup
        Target number of outliers – k
Output: k detected outliers
FIS = Mine for frequent itemsets(D, minsup)
foreach point x in D
  foreach itemset I ∈ x
    if FIS does not contain I
      Otey's outlierScore(x) +=  1/ length(I)
    end if
end for
return k outliers that maximize outlierScore
```

Figure 5. Otey's Algorithm Pseudocode

```
Input:  Dataset – D
        Target number of outliers – k
Output: k detected outliers
Label all data points as non-outliers
calculate frequency of each attribute value
foreach point x
  AVFscore = Sum(frequency each attrib. value ∈ x)/m
end foreach
return top k outliers with minimum AVFscore
```

Figure 6. AVF Algorithm Pseudocode

4. Experiments

4.1. Experimental Setup

We conducted all our experiments on a workstation with a Pentium 4 2.6 GHz processor and 1.5 GB of RAM. We implemented all algorithms in C++ and ran our own implementation of Apriori [19] for mining frequent itemsets for FPOF and Otey's algorithms (*minsup*=10%). We experimented with 5 real datasets from the UCI Machine Learning repository [22], as well as artificially generated datasets. An advantage of using artificially generated data is the capability to work with datasets of various sizes and dimensionalities.

A. Real Datasets (UCI):

- *Wisconsin Breast Cancer:* This dataset has 699 points and 9 attributes. Each record is labeled as either benign or malignant. Following the method by Harkins et al. [17], we only kept every sixth malignant record, resulting in 39 outliers (8%) and 444 non-outliers (92%).

– *Lymphography*: This dataset contains 148 instances and 19 attributes including the class attribute. There are 4 classes where classes 1 and 4 comprise 4% of the data, so they are considered as the outliers.

– *Post-operative*: This dataset is used to determine where patients should go to after a postoperative unit (to Intensive Care Unit, home, or general hospital floor). It contains 90 instances and 9 attributes, including the class. We regard class 1 and 2 as outliers (26 points total).

– *Pageblocks:* It contains 5,473 instances with 10 attributes. There are 5 classes, where one class is about 90% of dataset, so the rest of the data can be thought of as outliers. We discretized the continuous attributes using equal-frequency discretization, and removed half of the outliers so that we have a more imbalanced dataset (i.e., 280 outliers).

– *Adult*: This dataset has 48,842 points and 14 attributes. We discretized the numerical attributes using equal width discretization, and regarded as outliers the points with income more than $50K/year (about 24% of the dataset). We only kept every sixth point to make the dataset more imbalanced.

B. Artificial Datasets: The experiments conducted with the simulated data were used to showcase the speed and associated scalability of the algorithms that we are evaluating, and not their detection rates/capabilities (effectiveness). The simulated dataset used was created using available software by Cristofor [23]. The idea behind these experiments is to see the change of each algorithm's performance as parameters change (e.g., an important parameter is the size of the dataset, *n*). Another parameter with which we experimented is *m*, the dataset dimensionality. For example, Greedy calculates the Entropy for each dimension of each data point. Finally, it is worth mentioning that Greedy algorithm also depends

on the input number of outliers, k, while the other algorithms do not. For the first experiment we used a dataset with 10 attributes and input number of outliers, k=30, and 1K to 800K data points. For the second experiment, the dataset has 100K points and 10 attributes, and the outlier target number k varies from 1 to 1K. For the last experiment, the outlier number k is set to 30, the dataset contains 100K points, and the number of attributes is varied from 2 to 40.

4.2. Results and Discussion

Table 2 and Figure 7 depict the outliers detected by each algorithm using the real datasets (Figure 7 contains only the results for 3 sets due to space). As we see in the experiments with the real datasets, the outlier detection accuracy of AVF is the same as, or very close to Greedy. For example, in Figure 7(a), all algorithms converge to the 39 outliers for k equal to 56. While we notice similar accuracy in detecting outliers for all algorithms for the breast cancer, lymphography, and post-operative patients datasets (Figure 7(a)-(c)), for the other two datasets (Table 2(d)-pageblocks and Table 2(e)-adult), AVF has lower accuracy than Greedy, while FPOF and Otey's have much lower accuracy (e.g. in Table 2(d) for k=900 Greedy detects 242 outliers, AVF detects 223, while FPOF and Otey's detect only 116).

Table 3 contains the runtime performance of all algorithms using the first simulated dataset, with varying number of data points, n (see also Figure 8(a)). For example, in Table 3, for n=700K, Greedy finishes execution at around 185 seconds, Otey's methods at around 3127 seconds, FPOF at 564 seconds, while AVF has a running time of 1.33 seconds. Further experiments (not shown due to space limitations) with larger dimensionalities than in Figure 8(c) showed similar results for AVF: e.g. for a dataset 150 attributes (k=30, n=100K), AVF had a running time of about 2 seconds.

As we see from these results with real and artificially generated data, AVF approximates very well the outlier detection accuracy of Greedy, while it outperforms Greedy for larger values of the data size, n, data dimensionality, m, and the target number of outliers, k. E.g., Greedy becomes exceedingly slow for larger n values (Figure 8(a)), due to the increasingly more expensive entropy calculations. AVF's performance does not change notably for larger k values, and it runs significantly faster than Greedy with respect to both n and m, as AVF relies on simple calculations for each point. The FIM-based methods also become increasingly slower for larger datasets (see Figure 8(a)), as they need to search through all possible subsets of each and every point. Higher dimensionalities of the datasets slow down these two algorithms as well, because of the increasingly larger itemsets that are created as the number of attributes grows (see Figure 8(c)). The authors in [6] and [8] discussed using a parameter for the maximum length of itemsets; for example, if the user enters 5 as the max length, the itemsets created contain 5 items or less. However, experiments in [8] show that this negatively effects their algorithmic accuracy.

Table 2. Results on the Real Datasets

(a) Breast Cancer

k	Greedy	AVF	FPOF	Otey's
4	4	4	3	3
8	8	7	7	7
16	15	14	14	15
24	22	21	21	21
32	29	28	27	28
40	33	32	31	33
48	37	36	35	37
56	39	39	39	39

(b) Lymphography

k	Greedy	AVF	FPOF	Otey's
2	2	2	2	2
4	4	4	4	4
6	5	4	4	4
8	6	5	5	5
12 (13)	6	6	5	5 (6)
15	6	6	6	6

(c) Post-Operative

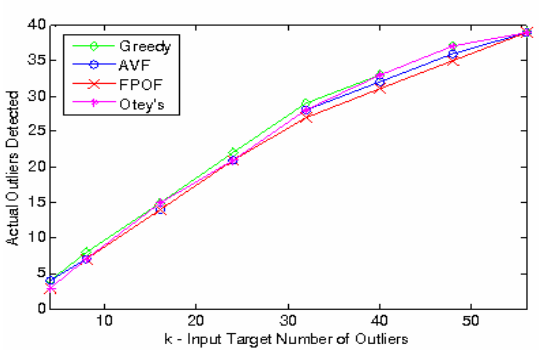
k	Greedy	AVF	FPOF	Otey's
10	4	3	3	1
20	7	7	7	7
30	8	10	9	9
40	12	11	10	10
50	13	12	12	13
60	20	16	17	18
70	21	21	21	21
80	24	24	24	24

(d) Pageblocks

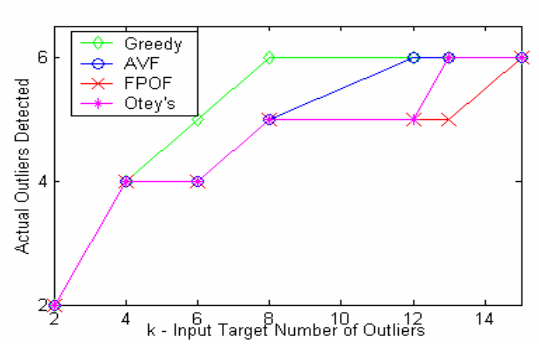
k	Greedy	AVF	FPOF	Otey's
100	45	40	19	19
200	81	84	42	42
300	130	120	63	63
400	157	168	74	74
500	177	189	80	80
600	183	201	94	94
700	213	206	96	96
800	237	214	110	110
900	242	223	116	116
1000	242	233	121	121

(e) Adult

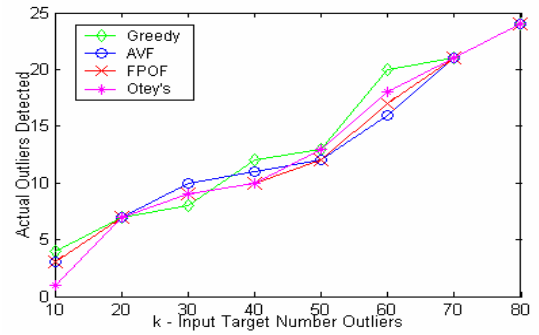
k	Greedy	AVF	FPOF	Otey's
100	24	27	21	16
200	41	45	42	37
300	69	71	64	54
400	87	88	82	66
500	107	104	89	77
600	124	116	114	96
700	147	139	137	112
800	171	164	148	139



(a) Breast Cancer

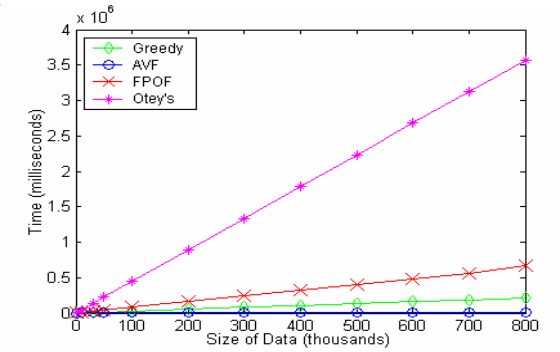


(b) Lymphography

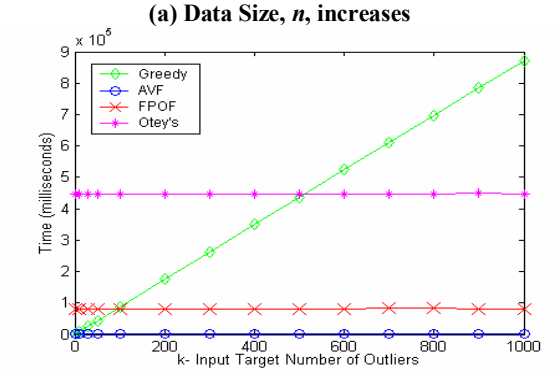


(c) Post-Op

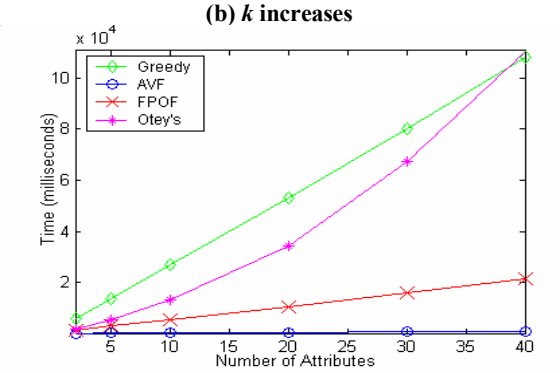
Figure 7. Actual Outliers Detected vs. Input Number of Target Outliers, *k*, on real datasets



(a) Data Size, *n*, increases



(b) *k* increases



(c) Number of Attributes, *m*, increases

Figure 8. Performance for simulated data as various parameters increase (*milliseconds*)

Table 3. Runtime in seconds for the simulated datasets with varying data size, *n*, from 1K to 800K data points

Data Size *(thousands)*	Greedy	AVF	FPOF	Otey's
1	0.27	0.00	0.81	4.58
10	2.72	0.03	8.13	44.72
30	8.53	0.06	24.02	134.30
50	14.31	0.09	40.19	222.88
100	26.42	0.19	81.06	445.39
200	52.75	0.39	165.08	891.28
300	79.39	0.58	241.61	1337.06
400	106.14	0.80	323.97	1781.78
500	131.75	0.94	404.45	2233.74
600	158.70	1.16	484.00	2678.73
700	184.94	1.33	564.80	3127.22
800	212.08	1.56	667.55	3568.55

In addition, the algorithms based on Frequent Itemsets, specifically FPOF and Otey's, depend on the user-entered minimum support threshold. In fact, different values of *minsup* lead to improved accuracy of FPOF and Otey's. For example, using the pageblocks data and *k*=200 (see Table 2(d)), Otey's method with *minsup*=0.04 detected 120 outliers, while for *minsup*=0.03, it detects 51 outliers. This is because lower *minsup* values will create more frequent sets, while higher *minsup* values will make more subsets infrequent. Also, lower *minsup* values make the algorithms increasingly slower as more frequent itemsets need to be created. This illustrates the challenge of selecting an appropriate minimum support threshold.

The advantages of AVF are that it does not create itemsets and that it entails only a single pass over the

entire dataset. Also, as shown from the experiments with the simulated datasets, the runtime of AVF increases particularly slowly with respect to *n* and *m*, in comparison to the other three algorithms. Moreover, AVF eliminates the need for difficult choices for any user-given parameters such as minimum support or maximum itemset length. However, experimentation with some small datasets showed that the single scan of the data may cause AVF to miss a few outliers that Greedy finds. This effect was negligible in the real datasets with which we experimented in this paper. We are examining a number of avenues to increase the outlier detection accuracy of AVF, but at the expense of declining performance.

5. Conclusions

We have compared and experimented with a representative list of currently available techniques for outlier detection in categorical datasets, using both real and artificially generated data. These methods have not been compared against each other before using the same datasets. We have proposed a scalable and effective outlier detection technique, called AVF (Attribute Value Frequency). AVF lends itself to the nature of datasets today, as its performance does not deteriorate given large datasets or datasets of higher dimensionality. Specifically, AVF has the following advantages:

- A computational complexity of $O(n*m)$, where n is the number of points, and m is the data dimensionality,
- AVF's performance does not rely on user-specified parameters such as minimum support,
- AVF scans the dataset only once to detect the desired outliers, i.e. the number of data scans needed by AVF is not influenced by the input number of target outliers, k,
- AVF is significantly faster than existing representative techniques against which we compared it, while maintaining virtually equal detection accuracy.

One of the limitations of AVF, mentioned at the end of the previous section, and how to overcome it, is the topic of some of our future work.

Acknowledgments: This work was supported in part by the NSF grants: CRCD: 0203446, CCLI: 0341601, DUE: 05254209, IIS-REU: 0647120, and IIS-REU: 0647018.

References

[1] Hawkins, S., He, H., Williams, G., and Baxter, R., "Outlier Detection Using Replicator Neural Networks", *Proc. of the Fifth Int'l Conference Data Warehousing and Knowledge Discovery*, pp.170-180, 2002.

[2] Knorr, E., Ng, R., and Tucakov, V., "Distance-based outliers: Algorithms and applications", *VLDB Journal*, 2000.

[3] Bolton, R.J., Hand, D.J., "Statistical fraud detection: A review", *Statistical Science*, 17, pp. 235–255, 2002.

[4] Penny, K.I., Jolliffe, I.T., "A comparison of multivariate outlier detection methods for clinical laboratory safety data", *The Statistician, Journal of the Royal Statistical Society*, 50, pp. 295–308, 2001.

[5] He, Z., Deng, S., Xu, X., "An Optimization Model for Outlier Detection in Categorical Data", *Proc. of 2005 International Conference on Intelligent Computing (ICIC'05)*, pp.400-409, 2005.

[6] He, Z., Xu, X., Huang, J., Deng, J., "FP-Outlier: Frequent Pattern Based Outlier Detection", *Computer Science and Information System*, pp. 103-118, 2005.

[7] He, Z., Deng, S., Xu, X., "A Fast Greedy algorithm for outlier mining", *Proc. of PAKDD*, 2006.

[8] Otey, M.E., Ghoting, A., Parthasarathy, A., "Fast Distributed Outlier Detection in Mixed-Attribute Data Sets", *Data Mining and Knowledge Discovery*, 2006.

[9] Barnett, V., Lewis, T. *Outliers in Statistical Data*. John Wiley, 1994.

[10] Hodge, V., Austin, J., "A Survey of Outlier Detection Methodologies", *Artificial Intelligence Review*, pp. 85, 2004.

[11] Ruts, I., Rousseeuw, P., "Computing depth contours of bivariate point clouds", *Comput. Stat Data Anal*, pp. 153, 1996.

[12] Bay, S.D. Schwabacher, M., "Mining distance-based outliers in near linear time with randomization and a simple pruning rule", *Proc. of ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2003.

[13] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J., "LOF: Identifying density-based local outliers", *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, 2000.

[14] Papadimitriou, S., Kitawaga, H., Gibbons, P., Faloutsos, C., "LOCI: Fast outlier detection using the local correlation integral", *Proc. of the Int'l Conf. on Data Engineering*, 2003.

[15] Wei, L., Qian, W., Zhou, A., Jin, W., "HOT: Hypergraph-based Outlier Test for Categorical Data", *Proc. of 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD*, pp. 399-410, 2003.

[16] Tax, D., Duin, R., "Support Vector Data Description", *Machine Learning*, pp. 45–66, 2004.

[17] Harkins, S., He, H., Williams, G., Baster, R., "Outlier Detection Using Replicator Neural Networks", *DaWaK'02*, pp. 170-180, 2002.

[18] Pei, Y., Zaiane, O., Gao, Y., "An Efficient Reference-based Approach to Outlier Detection in Large Dataset", *IEEE Int'l Conference on Data Mining*, 2006.

[19] Agrawal, R., Srikant, R., "Fast algorithms for mining association rules", *Proc. of the Int'l Conference on Very Large Data Bases VLDB*, pp. 487–499, 1994.

[20] Xu, Y.J., Qian, W., Lu, H., Zhou, A., "Finding centric local outliers in categorical/numerical spaces", *Knowledge Information Systems*, 9, pp. 309–338, 2006.

[21] Shannon, C.E., "A mathematical theory of communication", *Bell System Technical Journal*, pp. 379, 1948.

[22] Blake, C., Merz, C. *UCI machine learning repository:* www.ics.uci.edu/~mlearn/MLRepository.html.

[23] Cristofor, D., Simovici, D., "Finding Median Partitions Using Information-Theoretical Algorithms", *Journal of Universal Computer Science*, 8, pp. 153-172 (software at http://www.cs.umb.edu/~dana/GAClust/index.html).