Journal of Artificial Intelligence Research 57 (2016) 593-620 Submitted 06/16; published 12/16

ZERO++: Harnessing the Power of Zero Appearances to Detect Anomalies in Large-Scale Data Sets

Guansong Pang Guansong.Pang@student.uts.edu.au
Advanced Analytics Institute
University of Technology Sydney
Ultimo NSW 2007, Australia

Kai Ming Ting Kaiming.Ting@federation.edu.au
School of Engineering and Information Technology
Federation University
Churchill VIC 3842, Australia

David Albrecht David.Albrecht@monash.edu
Clayton School of Information Technology
Monash University
Clayton VIC 3800, Australia

Huidong Jin Warren.Jin@csiro.au
Data61, CSIRO, GPO Box 664
Canberra ACT 2601, Australia

Abstract

This paper introduces a new unsupervised anomaly detector called ZERO++ which employs the number of zero appearances in subspaces to detect anomalies in categorical data. It is unique in that it works in regions of subspaces that are not occupied by data; whereas existing methods work in regions occupied by data. ZERO++ examines only a small number of low dimensional subspaces to successfully identify anomalies. Unlike existing frequency-based algorithms, ZERO++ does not involve subspace pattern searching. We show that ZERO++ is better than or comparable with the state-of-the-art anomaly detection methods over a wide range of real-world categorical and numeric data sets; and it is efficient with linear time complexity and constant space complexity which make it a suitable candidate for large-scale data sets.

1. Introduction

Anomalies are data instances that are rare and exceptional compared to the majority of data. Anomaly detection generally refers to the process of finding anomalies. It is regarded as one of the most important tasks in data mining due to its wide application in various domains, such as intrusion detection, event detection, disease and adverse reaction detection (Chandola, Banerjee, & Kumar, 2009; Jin, Chen, He, Kelman, McAullay, & O'Keefe, 2010; Xu & Shelton, 2010).

In a categorical data set, anomalies are rare instances, i.e., those instances which have combinations of values that are rare. Furthermore, in a random *subsample*, the probability of having no instances in the subsample with the same values as a given test instance, on any attribute subset (i.e., *subspace*), increases monotonically with a decrease in the frequencies of the values in the full data set. Therefore, anomalies are likely to have *zero appearances*

in small subsamples, and also have a higher probability of having zero appearances than normal instances in subsamples of any size (see Definitions 1 and 2 in Section 3.2 for the formal definitions of subspaces and zero appearances).

Based on this observation, we introduce a new unsupervised anomaly detector called ZERO++ which employs the number of zero appearances in subspaces over a set of subsamples to identify anomalies. The challenge to harness the power of zero appearances is that the number of subspaces is exponential to data dimensionality. As a result, examining zero appearances in all the subspaces is computationally intractable for high dimensional data sets. ZERO++ is an efficient and effective method because it uses a small set of low dimensional subspaces to harness the power of zero appearances.

ZERO++ is unique in that it works in regions of subspaces that are not occupied by data; whereas existing methods (Akoglu, Tong, Vreeken, & Faloutsos, 2012; Breunig, Kriegel, Ng, & Sander, 2000; He, Xu, Huang, & Deng, 2005; Liu, Ting, & Zhou, 2012) identify anomalies based on the assumption that anomalies lie in regions of low density/frequency, i.e., in regions occupied by data.

Being the first to harness the power of zero appearances to detect anomalies, our contributions are to:

- Provide analyses which underlie the power of zero appearances in subspaces and subsamples.

- Propose a new unsupervised anomaly detection method called ZERO++ which works on categorical data. ZERO++ is an efficient algorithm with linear time complexity w.r.t. data size and data dimensionality, and constant space complexity. This makes ZERO++ a suitable candidate for large-scale data sets.

- Conduct a series of experiments to evaluate the effectiveness and efficiency of ZERO++. It is shown to detect anomalies more effectively than the state-of-the-art anomaly detectors in real-world categorical and numeric data.

ZERO++ is a novel categorical anomaly detector in the following aspects:

1. The anomaly score used by ZERO++ is simpler than the existing frequency-based scores; and it is a score which is sufficient to detect anomalies.

2. Frequency-based algorithms (Akoglu et al., 2012; He et al., 2005; Smets & Vreeken, 2011) need to conduct a subspace pattern searching which have time and space complexities that are at least quadratic in terms of the data dimensionality. ZERO++ involves no searching; thus it runs significantly faster.

3. Frequency-based algorithms employ the anti-monotone property to reduce the search space. ZERO++ conducts a fixed number of examinations without search.

The rest of this paper is organised as follows. Related work is discussed in Section 2. We introduce the proposed method ZERO++ in Section 3. After an introduction to the experimental setup in Section 4, Section 5 provides a series of our empirical results. The conclusions and future work are provided in Section 6.

2. Related Work

Anomaly detection methods can be generally categorised into supervised methods, semi-supervised methods and unsupervised methods. Supervised methods employ labelled instances of both the normal class and the anomalous class to train detection models. Some examples of these methods are Support Vector Machines (SVMs) and Neural Networks (Erfani, Baktashmotlagh, Rajasegarar, Karunasekera, & Leckie, 2015). Semi-supervised methods require labelled instances of the normal class only, in order to train their detection models, e.g., one-class classifiers (Görnitz, Kloft, Rieck, & Brefeld, 2013). Compared to supervised methods and semi-supervised methods, unsupervised methods, which do not require labelled instances, are more widely used in industry, because obtaining accurate labelled data for anomaly detection often has a very high cost (Chandola et al., 2009). This paper focuses on unsupervised methods. We refer readers to a recent survey (Chandola et al., 2009) for related work in the other two methods.

2.1 Methods for Numeric Data

Many unsupervised anomaly detection methods are numeric data oriented, and they assume that anomalies lie in regions of low density. Examples are distance based methods (e.g., kNN, see Ramaswamy, Rastogi, & Shim, 2000; LDOF, see Zhang, Hutter, & Jin, 2009), density based methods (e.g., LOF, see Breunig et al., 2000), clustering based methods (e.g., CBOF, see Duan, Xu, Liu, & Lee, 2009), and isolation based methods (e.g., iForest, see Liu et al., 2012; MassAD, see Ting, Zhou, Liu, & Tan, 2013). These methods rely on the key characteristic of numeric data, i.e., the notion of ordering. Categorical attributes are required to be transformed into numeric attributes in order to make these methods applicable for categorical data, but it is difficult to find a universally effective transformation method for different data sets (Boriah, Chandola, & Kumar, 2008). Also, many traditional detectors, such as kNN and LOF, require $O(d^2)$ time complexity where d is the total number of instances. Though it can be reduced to $O(d \log d)$ if an indexing scheme such as R^*-tree (Beckmann, Kriegel, Schneider, & Seeger, 1990) is employed, most indexing methods break down with high dimensionality or categorical data.

`ZERO++` uses a similar algorithmic framework as iForest, i.e., it constructs models over subspaces and subsamples, and has similar time and space complexities; but they have significant differences in terms of motivation, working principles and anomaly scores. `ZERO++` is motivated by and operated on zero appearances; whereas iForest aims to isolate every instance from the rest of the instances. As a result, they employ different anomaly scores. `ZERO++` is designed based on categorical data, while iForest is based on numeric data.

Some research (Wu & Jermaine, 2006; Sugiyama & Borgwardt, 2013) proposes to perform kNN search in *one* small subsample, in which the *kth*-NN distance is used as an anomaly score. As anomalies normally have larger *kth*-NN distance and they are less likely to be selected in a small subsample than normal instances, anomalies have high probability of having larger *kth*-NN distance than normal instances in a small subsample. Contrasting to these work which aims to improve the scalability in traditional kNN methods via subsampling, `ZERO++` utilises the probability of zero appearances to introduce a new anomaly scoring measure. Also, they operate in a single subsample only and use the full dimensionality to compute the *kth*-NN distance, which can lead to unstable detection performance

and have the *curse of dimensionality* problem in high-dimensional data (Zimek, Campello, & Sander, 2013a), while ZERO++ builds a set of models in a set of subsamples and subspaces.

2.2 Methods for Categorical Data

Compared to methods for numeric data, significantly less research has been conducted for categorical data. Most existing categorical data oriented methods are based on a general assumption that anomalies lie in regions of low frequency (Akoglu et al., 2012; Ghoting, Otey, & Parthasarathy, 2004; He et al., 2005; Koufakou, Ortiz, Georgiopoulos, Anagnostopoulos, & Reynolds, 2007; Koufakou & Georgiopoulos, 2010; Smets & Vreeken, 2011; He, Deng, Xu, & Huang, 2006). Typical examples are frequent patterns based methods FPOF (He et al., 2005) and infrequent patterns based methods LOADED (Ghoting et al., 2004). FPOF and LOADED build a single model on the entire training set, and identify anomalies based on frequent patterns and infrequent patterns, respectively. KRIMP (Smets & Vreeken, 2011) and COMPREX (Akoglu et al., 2012) also build a single model on the entire training set using pattern-based compression techniques. KRIMP generates the patterns based on frequent itemsets, while COMPREX employs the Minimum Description Length (Barron, Rissanen, & Yu, 1998) principle to automatically generate patterns from attribute groups (subspaces) with high information gain and avoid the costly frequent itemset search. In contrast, ZERO++ records the number of appearances on a set of subspaces and subsamples, and employs the number of zero appearances in subspaces and subsamples to identify anomalies.

ZERO++ has much lower time and space complexities than FPOF, LOADED, KRIMP and COMPREX. Those four methods need to search for abnormal or normal patterns to detect anomalies, which have time and space complexities at least quadratic to data dimensionality. In contrast, ZERO++ detects anomalies in a small set of randomised two-dimensional subspaces, which has time complexity linear to data dimensionality and data size. The anti-monotone property is applied in FPOF, LOADED and KRIMP to reduce the search space, whereas ZERO++ examines a fixed number of low dimensional subspaces only and no search is required in ZERO++.

Some computationally expensive algorithms (Ghoting et al., 2004; Koufakou & Georgiopoulos, 2010) require to use parallelism in order to reduce their runtime. But parallelism does not reduce the time complexity of the base algorithm. ZERO++ can easily be applied to parallel processors to achieve further speedup.

2.3 Exact Methods versus Probabilistic Methods

An exact method produces exactly the same outcome that satisfies the given criteria. In contrast, probabilistic methods do not have such a guarantee.

Due to the attractiveness of exact outcomes and relatively easy implementation, most traditional methods for numeric data and categorical data are exact methods, e.g., kNN, LOF, FPOF and COMPREX. In practice, probabilistic methods are more desirable than exact methods due to several reasons (see the next paragraph); that is why they have become popular, especially the sampling based methods (Lazarevic & Kumar, 2005; Wu & Jermaine, 2006; Liu et al., 2012; Ting et al., 2013; Zimek, Gaudet, Campello, & Sander, 2013b; Sugiyama & Borgwardt, 2013; Pang, Ting, & Albrecht, 2015).

First, many exact methods, such as kNN and LOF, suffer from the effects of *masking* (i.e., some anomalies are overlooked due to the presence of other anomalies) and/or *swamping* (i.e., some normal instances are incorrectly reported as anomalies due to the presence of other anomalies) because they are required to work on the entire dataset; while probabilistic methods working on data subsets are more robust with respect to these two effects (Liu et al., 2012; Zimek et al., 2013b; Pang et al., 2015). Second, exact methods that work on full attribute sets can be more sensitive to irrelevant and noisy features than probabilistic methods that work on attribute subsets (Lazarevic & Kumar, 2005; Liu et al., 2012). Third, as discussed in the above two subsections, probabilistic methods normally have much lower time complexity, especially for large data sets, because distance computation and/or subspace searching are not required. Most importantly, the anomaly detection accuracy of probabilistic methods is as good as, if not better than, that produced by exact methods (Liu et al., 2012; Sugiyama & Borgwardt, 2013; Pang et al., 2015).

Motivated by the above reasons, `ZERO++` takes advantage of probabilistic methods by building models on data subsets as well as attribute subsets. Although `ZERO++` works on low-dimensional subspaces only, our empirical results show that it works well in high-dimensional data, which is consistent with the previous results (Lazarevic & Kumar, 2005; Liu et al., 2012; Ting et al., 2013). Also, by working on a sufficiently large ensemble size, `ZERO++` works stably even using small subsamples only, as will be shown in Section 5.2.4.

3. Zero Appearances as a Means to Detect Anomalies

We provide our intuition of harnessing zero appearances to detect anomalies in Section 3.1, followed by formal definitions in Section 3.2. The anomaly score is defined in Section 3.3, and an efficient method for computing the score is provided in Section 3.4. We provide an analysis for understanding the reason why our method can work well with small subsampling size and ensemble size in Section 3.5, and present the algorithmic procedure in Section 3.6.

3.1 Intuition

Here we provide two intuitive examples that zero appearances can be used to detect anomalies. The first example shows the fact that anomalies are more likely to have zero appearances in a subsample than normal instances. The second example intuits that, using zero appearances, only subspaces need to be examined to detect anomalies.

Example 1: Given an univariate categorical data set with 1,000 instances, and there exists anomalies with at most 10 appearances and normal instances with at least 100 appearances. In a subset of 8 instances, randomly sampled without replacement from the data set, the expected probability of any anomaly having zero appearances in this subset is at least $\binom{1000-10}{8}/\binom{1000}{8} \approx 0.9225$; whereas that of any normal instance is at most $\binom{1000-100}{8}/\binom{1000}{8} \approx 0.4291$. As a result, having sampled multiple such subsets, the anomaly is likely to have zero appearances in significantly more subsets than normal instances.

Example 2: In a multivariate data set, the rarity characteristic of anomalies is usually manifested as zero appearances in subspaces. In other words, test instances, which appear in regions of subspaces that are not occupied by training instances, are likely to be anomalies.

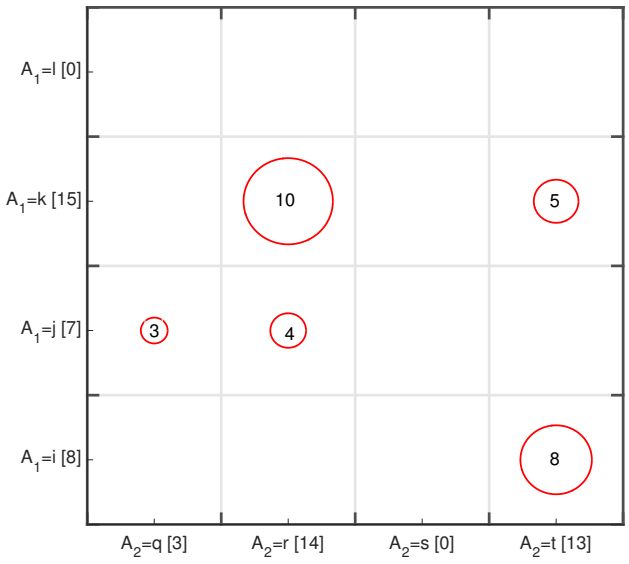

Figure 1: A two-dimensional data set with 30 instances, where the size of the circle indicates the number of instances in a region; and the number in [ ] indicates the number of instances in one-dimensional subspace. The categorical values in A_1 are $\{i, j, k, l\}$ and the values in A_2 are $\{q, r, s, t\}$.

An example of the zero appearances in subspaces for a simple two-dimensional subspace is shown in Figure 1. Given the data distribution shown in Figure 1, regions of zero appearances in one-dimensional subspaces are $S_{A_1 l}$ and $S_{A_2 s}$ only, where $S_{A_x y}$ denotes region $A_x = y$ in the one-dimensional subspace of attribute A_x. Any test instance having either $A_1 = l$ or $A_2 = s$ is likely to be an anomaly. Since instances must have zero appearances in higher-dimensional subspaces of either $A_1 = l$ or $A_2 = s$, we only need to examine regions in these one-dimensional subspaces.

We harness the power of zero appearances in both subspaces and subsamples, described above, to detect anomalies.

3.2 Zero Appearances in Subspaces and Subsamples

Let $\mathcal{D}$ be a set of d instances $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_d}\}$ with q categorical attributes, and $A = A_1 \times A_2 \times \ldots \times A_q$ be a product set of the attributes; and $\boldsymbol{x} = (x_1, \ldots, x_q)^T$.

Definition 1 (Subspace)**.** *A subspace is a product set of m attributes*

$$S = A_{l_1} \times A_{l_2} \times \ldots \times A_{l_m}, \tag{1}$$

where $1 \leq l_1 < l_2 \ldots < l_m \leq q$.

Let $\mathcal{N}$, with n randomly selected instances (by sampling without replacement), be a random subset of $\mathcal{D}$; and $I_S(\boldsymbol{x} = \boldsymbol{x}')$ be an indicator function, which is 1 if an instance $\boldsymbol{x}$ is identical to another instance $\boldsymbol{x}'$ in S, and 0 otherwise.

Definition 2 (Zero Appearance)**.** *An instance $\boldsymbol{x}$ has zero appearances in S given $\mathcal{N}$, if $\forall \boldsymbol{x}' \in \mathcal{N}$, $I_S(\boldsymbol{x} = \boldsymbol{x}') = 0$.*

For example, assume A_i and A_j are two attributes of $\mathcal{D}$, and A_i has three values a_1, a_2, a_3 and A_j has two values b_1, b_2, then $S = A_i \times A_j = \{(a, b) : a \in A_i \,\&\, b \in A_j\} = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2), (a_3, b_1), (a_3, b_2)\}$. In a subsample $\mathcal{N}$ of $\mathcal{D}$, suppose only $\{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$ occur, then (a_3, b_1) and (a_3, b_2) have zero appearances in S given $\mathcal{N}$.

Definition 3 (Zero Appearance Indicator Function)**.** *An zero appearance indicator function $\eta : \mathcal{D} \mapsto \{1, 0\}$ is defined as: $\eta_S(\boldsymbol{x}) = 1$ if $\boldsymbol{x}$ has zero appearances in S given $\mathcal{N}$, and $\eta_S(\boldsymbol{x}) = 0$ otherwise.*

Proposition 1 (Zero Appearance Expectation)**.** *The probability of $\eta_S(\boldsymbol{x})$ has an expected value $E(\eta_S(\boldsymbol{x})) = \frac{\binom{d-\tau_S(\boldsymbol{x})}{n}}{\binom{d}{n}}$, where $\tau_S(\boldsymbol{x}) = |\{\boldsymbol{x}' \in \mathcal{D} : I_S(\boldsymbol{x} = \boldsymbol{x}') = 1\}|$.*

Proof. There are $\binom{d-\tau_S(\boldsymbol{x})}{n}$ choices for sampling n instances from d instances while excluding instances that are identical to $\boldsymbol{x}$ in S.

On the other hand, simply sampling n instances from d instances has $\binom{d}{n}$ choices. Therefore, $E(\eta_S(\boldsymbol{x})) = \frac{\binom{d-\tau_S(\boldsymbol{x})}{n}}{\binom{d}{n}}$. ■

According to Proposition 1, the probability of instances having zero appearances in a random subsample is inversely proportional to its frequency in the full data set. In *anomalous subspaces* (i.e., subspaces where anomalies have significantly lower frequency than normal instances), anomalies have substantially higher probability of having zero appearances in the subsamples compared to normal instances. On the other hand, in non-anomalous subspaces, anomalies and normal instances often have similar occurrence frequencies, leading to similar probabilities of having zero appearances. Therefore, anomalies are likely to have a larger number of zero appearances in subspaces than normal instances.

3.3 Anomaly Score

Let $P_S(\boldsymbol{x}|\mathcal{N}) = \frac{\sum_{\boldsymbol{x}' \in \mathcal{N}} I_S(\boldsymbol{x}=\boldsymbol{x}')}{|\mathcal{N}|}$ be the probability of instance $\boldsymbol{x}$ in subspace S given subsample $\mathcal{N}$, and $\mathcal{R}$ be the set of all subspaces in A, i.e., $|\mathcal{R}| = \sum_{m=1}^{q} \binom{q}{m} = 2^q - 1$. Based on Proposition 1, we define a new anomaly score as follows:

Definition 4 (Anomaly Score)**.** *The anomaly score for $\boldsymbol{x}$ is defined as the number of zero appearances in subspaces in $\mathcal{R}$ over t subsamples $\mathcal{N}_i$, $i = 1, 2, \cdots, t$:*

$$zero\_score(\boldsymbol{x}) = \sum_{i=1}^{t} \sum_{S \in \mathcal{R}} \omega(S) I(P_S(\boldsymbol{x}|\mathcal{N}_i) = 0) \tag{2}$$

where $\omega(S)$ is a subspace weighting function and $I(\cdot)$ is an indicator function.

The anomaly score is bounded by $[0, t|\mathcal{R}|]$ [1]. Anomalies are instances having large anomaly scores, i.e., instances having zero appearances in a large number of (highly weighted) subspaces over a set of subsamples.

The number of subspaces in $\mathcal{R}$ is exponential to the data dimensionality. Therefore, anomaly detection algorithms based on $\mathcal{R}$ is computationally intractable in many real-world data sets which have hundreds of dimensions.

In addition, the function $\omega(\cdot)$ is introduced because the number of subspaces of different sizes (i.e., subspaces spanned by different number of attributes) varies substantially and so the importance of zero appearances in subspaces of different sizes are actually not the same. Therefore, a proper subspace weighting scheme is required in the instantiation of the anomaly score.

We will provide a method to solve these two issues in the next section.

3.4 Using a Smaller Number of Low Dimensional Subspaces only

Here we present a closure property of zero appearances, and then use it to simplify the anomaly score function in Equation (2) to significantly reduce the number of subspaces that needs to be examined while preserving its power in anomaly detection.

Let S' be a higher-dimensional subspace containing S, denoted by $S' \supset S$, where $S = A_{l_1} \times A_{l_2} \times \ldots \times A_{l_m}$ and $S' = A_{l_1} \times A_{l_2} \times \ldots \times A_{l_g}$, and $g > m$.

Property (Closure Property)**.** *If* $P_S(\boldsymbol{x}|\mathcal{N}) = 0$*, then* $P_{S'}(\boldsymbol{x}|\mathcal{N}) = 0$*,* $\forall S' \supset S$*.*

Based on the closure property, if a test instance has zero appearances in a subspace, it must also have zero appearances in the higher dimensional subspaces of this subspace (This property was exemplified by the second example in Section 3.1). Thus, examining zero appearances in low dimensional subspaces is sufficient to distinguish anomalies from normal instances, if the anomalous subspaces in low dimensions. This is often the case even in high dimensional data, as most attributes of this data are irrelevant.

This property motivates us to approximate our anomaly score by using $\mathcal{R}_m$ with a small m only, where $\mathcal{R}_m$ is the set of all subspaces having exactly m attributes only.

However, the time and space complexities of examining zero appearances in subspaces in $\mathcal{R}_m$ with $m \geq 2$ are at least quadratic to data dimensionality, and they are still prohibitive for high dimensional data sets.

In this paper, `ZERO++` employs the zero appearances in subspaces in $\mathcal{R}'_2$, as the default setting, to identify anomalies. $\mathcal{R}'_2$ is a subset of $\mathcal{R}_2$ having $|\mathcal{R}'_2| = q$ such that every attribute appears exactly twice in $\mathcal{R}'_2$. Since we work on subspaces of the same size, the weights for the subspaces can be assigned with the same value, e.g., $\omega(S) = 1$. Thus, we have the following anomaly scoring function.

$$zero\_score(\boldsymbol{x}) = \sum_{i=1}^{t} \sum_{S \in \mathcal{R}'_2} I(P_S(\boldsymbol{x}|\mathcal{N}_i) = 0) \tag{3}$$

1. Note that the anomaly score for $\boldsymbol{x}$ can also be defined as the difference between the total number of subspaces and the number of subspaces in $\mathcal{R}$ having at least one appearances over t subsamples, i.e., $\neg zero\_score(\boldsymbol{x}) = t|\mathcal{R}| - \sum_{i=1}^{t} \sum_{S \in \mathcal{R}} \omega(S) I(P_S(\boldsymbol{x}|\mathcal{N}_i) > 0)$. Both *zero_score* and $\neg$*zero_score* produce the same anomaly ranking.

Specifically, $\mathcal{R}_2'$ is generated randomly as follows [2]: After a random order of q attributes $A_{i_1}, A_{i_2}, \ldots, A_{i_q}$ is generated, q attribute-pairs are formed by chaining the consecutive pair of attributes circularly until each attribute appears exactly twice, yielding $\mathcal{R}_2' = \{S_{i_1 i_2}, S_{i_2 i_3}, \ldots, S_{i_{(q-1)} i_q}, S_{i_q i_1}\}$, where $S_{ij} = A_i \times A_j$. Note that to produce an anomaly score for $\boldsymbol{x}$ as shown in Equation (3), $\mathcal{R}_2'$ is generated randomly t times for t subsamples.

The use of $\mathcal{R}_2'$ brings about three advantages. First, using $\mathcal{R}_2'$ significantly reduces its time complexity from $d2^q$ (using $\mathcal{R}$) or dq^2 (using $\mathcal{R}_2$) to dq ; Second, using $\mathcal{R}_2'$ in an ensemble of t models provides a good approximation to $\mathcal{R}_2$. Third, $\mathcal{R}_2'$ covers all the attributes and every attribute has an equal chance to be considered in $\mathcal{R}_2'$, which avoids bias in the subspaces chosen.

These advantages can be observed in Figures 2 and 3 which show the detection accuracy of ZERO++ using $\mathcal{R}$, $\mathcal{R}_2$ and $\mathcal{R}_2'$ in *Chess* and *Mushroom*[3] with increasing sampling size n. Note that $\mathcal{R}$ is too large to be used for $q > 20$. The results show that ZERO++ using $\mathcal{R}_2'$ obtains comparable detection accuracy to that using $\mathcal{R}_2$ or $\mathcal{R}$. The results further show that it is most expensive to run ZERO++ using $\mathcal{R}$, and there is almost no gain in AUC by using $\mathcal{R}_2$.

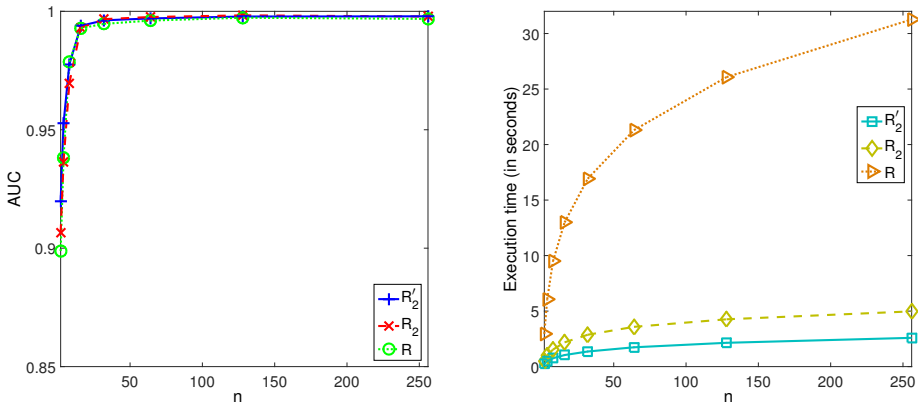

Figure 2: AUC performance and runtime comparison between using $\mathcal{R}_2'$, $\mathcal{R}_2$ and $\mathcal{R}$ on *Chess* with $q = 6$. $t = 50$ is used to produce this result.

However, note that the reverse of the closure property is not necessarily true, i.e., when $P_{\mathcal{S}'}(\boldsymbol{x}|\mathcal{N}) = 0$, there may exist some $S \subset S'$ that $P_S(\boldsymbol{x}|\mathcal{N}) \neq 0$. As a result, using $\mathcal{R}_2'$ fails to work in data sets having high-order feature dependence, where anomalies are detectable in higher-dimensional subspace S' only since they may have similar behaviours as normal instances in all the lower-dimensional subspaces of S'. For example, in Figure 1, we would overlook the zero appearance in the intersecting region of $S_{A_1 i}$ and $S_{A_2 r}$ in two-dimensional subspaces if we focus on the zero appearances in one-dimensional subspaces. Similarly, we

2. We have attempted a method alternative to $\mathcal{R}_2'$ which generates a number of (randomly selected) subspaces by random sampling with replacement. However, having a number of subspaces larger than $|\mathcal{R}_2'|$ through this method does not seem to produce a significantly better AUC result than that from $\mathcal{R}_2'$.
3. These data sets are from the UCI repository (Bache & Lichman, 2013). Section 4.2 shows their characteristics.

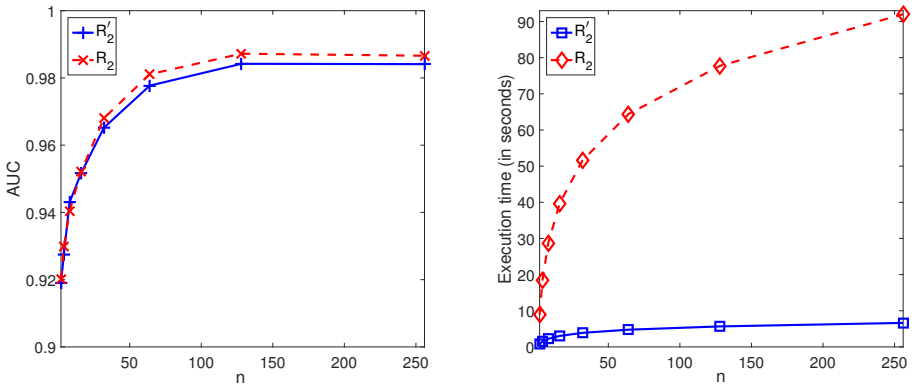

Figure 3: AUC performance and runtime comparison between using $\mathcal{R}'_2$ and $\mathcal{R}_2$ on *Mushroom* with $q = 22$. $t = 50$ is used to produced this result. The runtime is prohibitive using $\mathcal{R}$.

may fail to detect zero appearances in subspaces of size no less than three if we only examine appearances in subspaces of sizes one or two.

To detect anomalies which appear in anomalous subspaces more than two dimensions, we suggest to use a parameterised version of ZERO++, where m needs to be tuned to yield the best detection result, as follows.

Using $\mathcal{R}'_m$ [4] with $3 \leq m < q$, which has the same advantages as the use of $\mathcal{R}'_2$, can handle those data sets. The 'right' m to use mainly depends on the order of the feature dependence in a data set. The higher the order dependence, the larger m shall be used. Example real-world data sets with this characteristic will be given in Section 5.2.2.

In practice, m is searched in the range $[2, q]$ in order to yield the best detection result. This is analogous to the search of an appropriate neighbourhood size in nearest neighbours-based algorithms.

The default $\mathcal{R}'_2$ is used as a trade-off between $\mathcal{R}_1$ and $\mathcal{R}'_m$ with $3 \leq m < q$ in all our experiments, unless stated otherwise.

3.5 Why Does ZERO++ Work?

This section provides an analysis to explain the reasons why ZERO++ works well with small subsampling size and a small number of subsamples. Most data sets often have significantly different frequency distributions in different subspaces or different random subsamples, and thus it is difficult to quantify the contribution of all the subspaces together. We simplify the analysis by examining the contribution of *single* subspace.

3.5.1 Statistical Support for Zero Appearances in Small Subsamples

In data sets with different data sizes, ZERO++ works consistently well with a small subsample size. To explain this, let $P_S(\boldsymbol{x}) = \frac{\tau_S(\boldsymbol{x})}{d}$ be the probability of $\boldsymbol{x}$ occurring in a subspace S

4. The same process of generating $\mathcal{R}'_2$ can also be used to generate $\mathcal{R}'_m$, but it should be noted that $\mathcal{R}'_m$ with $m = 1$ or $m = q$ has a different property to that of $\mathcal{R}'_m$ with $1 < m < q$, i.e., every attribute appears once only in $\mathcal{R}'_1$ and $\mathcal{R}'_d$.

given $\mathcal{D}$. $E(\eta_S(\boldsymbol{x}))$ can also be computed as follows:

$$E(\eta_S(\boldsymbol{x})) = (1 - P_S(\boldsymbol{x})) \times \frac{d(1 - P_S(\boldsymbol{x})) - 1}{d - 1} \times \cdots \times \frac{d(1 - P_S(\boldsymbol{x})) - (n - 1)}{d - (n - 1)}$$

For a large d, it can be simply approximated as follows:

$$E(\eta_S(\boldsymbol{x})) \approx (1 - P_S(\boldsymbol{x}))^n \tag{4}$$

Based on Equation (4), given a small subsample $\mathcal{N}$, if $\boldsymbol{x}$ is a rare instance in $\mathcal{D}$, i.e., $P_S(\boldsymbol{x})$ is very small, then the probability of $\boldsymbol{x}$ having zero appearances in $\mathcal{N}$ is very high.

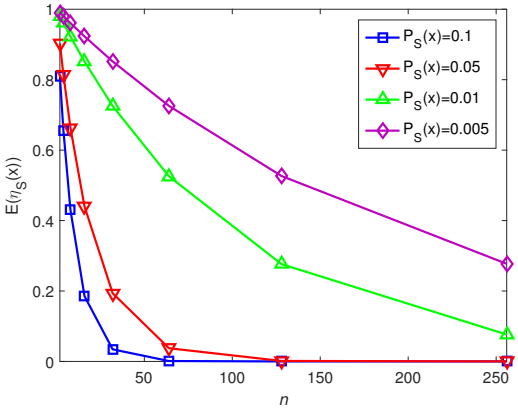

Figure 4: Probability of having zero appearances in subsamples w.r.t. different subsample sizes and different $P_S(\boldsymbol{x})$.

Figure 4 presents $E(\eta_S(\boldsymbol{x}))$ w.r.t. different n, i.e., 2, 4, 8, 16, 32, 64, 128, 256, given different $P_S(\boldsymbol{x})$, i.e., 0.1, 0.05, 0.01, 0.005. It shows that a small subsample size, e.g., $n \leqslant 64$, can generally ensure rare instances having high probability (> 0.5) of having zero appearances in subsamples. Particularly, for $P_S(\boldsymbol{x}) \leqslant 0.05$, $E(\eta_S(\boldsymbol{x})) > 0.65$ if $n \leqslant 8$. For a relatively large $P_S(\boldsymbol{x})$, e.g., $P_S(\boldsymbol{x}) = 0.1$, a smaller n, e.g., 2 or 4, should be taken in order to ensure $E(\eta_S(\boldsymbol{x}))$ within the range $(0.5, 1.0]$. Considering the percentage of anomalies is normally less than 5%, a small subsample size, e.g., $n \leqslant 64$, is preferred in order to ensure that anomalies have zero appearances in a sufficiently large number of subspaces. We use $n = 8$ as default in our experiments.

3.5.2 The Number of Subsamples Required

We now use statistics to estimate a reasonable number of subsamples (t) required in ZERO++. Since we focus on single subspace, Equation (3) is transformed to $zero\_score(\boldsymbol{x})_S = \sum_{i=1}^{t} I\left(P_S(\boldsymbol{x}|\mathcal{N}_i) = 0\right)$, which can be viewed as the total number of successes (i.e., $\boldsymbol{x}$ has zero appearance in S) out of t subsamples, each drawn with replacement from the population. Thus, $zero\_score(\boldsymbol{x})_S$ could be modelled by the binomial distribution with a success probability p_S, i.e., the expected probability of zero appearances of $\boldsymbol{x}$ in a subsample. That is, from the observation $zero\_score(\boldsymbol{x})_S$, the expected probability of zero appearances

is estimated as $\hat{p}_S = \frac{zero\_score(\boldsymbol{x})_S}{t}$, and its confidence interval is $\hat{p}_S \pm z_{\alpha/2}\sqrt{\frac{\hat{p}_S(1-\hat{p}_S)}{t}}$ for significance level α, where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution, according to the simple Wald method (Wallis, 2013).

To successfully detect anomalies, ZERO++ requires the subsample size t large enough to distinguish anomalies from normal instances. For a given significance level α, t needs to be large enough to ensure that the confidence intervals for anomalies do not cover $p_S = 0$ (which is the conservative expected probability of zero appearances for normal instances). In other words, the lower bound of p_S for an anomaly (Ω) is greater than 0.

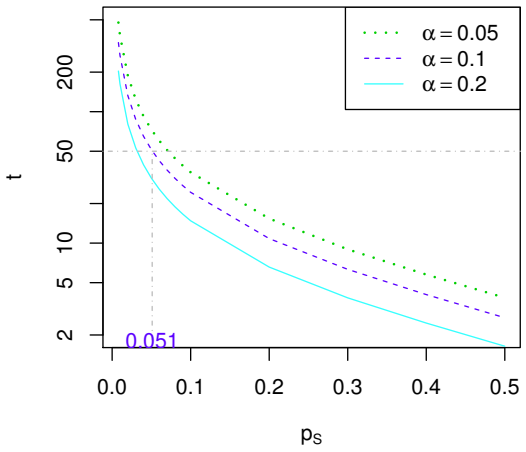

Figure 5: Reasonable number of subsamples t required to identify anomalies from normal instances for three significance levels of α.

Figure 5 illustrates the required values for t when $\Omega = 0$ for three significance levels of α. When $t \geq 50$, we can expect ZERO++ to distinguish anomalies from normal instances with small expected zero appearances (e.g., 0.1 or 0.05). For example, given $t = 50$, we have 95% ($= 1 - \frac{\alpha}{2}$ as t-distribution is symmetric) confidence that the estimated probability $\hat{p}_S > 0$ when the 'true' expected probability p_S is 0.051 or larger. We use $t = 50$ as the default setting in our experiments.

3.6 ZERO++: The Algorithm

Given a data set with categorical attributes, ZERO++ builds a set of models in the training stage, and these models can then be used to score every instance in the testing stage. The procedures of these two stages are given below.

Training. In the training stage, t subsamples are generated by random sampling without replacement, and ZERO++ builds a probability table from each subsample based on $\mathcal{R}'_2$ for each model. Each table h has $q \times 2$ columns. Every two columns store the index of the product set of a subspace and their corresponding probabilities. Note that we are interested in entries of the subspaces having zero probabilities only, and the non-zero entries in the

Algorithm 1 *CreateProbabilityTables($\mathcal{D}$, t, n)*

Input: $\mathcal{D}$ - input data, t - the number of subsamples, n - subsample size
Output: $\mathcal{H}$ - a set of probability tables

```
Initialise H as an empty set
for i = 1 to t do
    N_i ← Randomly select n instances without replacement from D
    Generate a randomised subspace set R'_2
    Initialise a probability table h_i with q × 2 columns, where each row stores the index
    of a subspace S and its probability P_S(·|N_i).
    Update h_i with P_S(·|N_i), ∀S ∈ R'_2
end for
H = ⋃_{i=1}^{t} h_i
return H
```

probability table are only useful in so far as to identify zero entries. The procedure to generate the probability tables is presented in Algorithm 1.

Testing. To score a test instance $\boldsymbol{x}$, ZERO++ computes the number of zero appearances in the subspaces in t probability tables as the anomaly score for $\boldsymbol{x}$ (as shown in Equation 3). The higher the score is, the more likely $\boldsymbol{x}$ is an anomaly. This procedure is presented in Algorithm 2.

Algorithm 2 *zero_score($\boldsymbol{x}$)*

Input: $\boldsymbol{x}$ - a test instance
Output: *score* - the total number of zero appearances in subspaces of $\boldsymbol{x}$

```
score ← 0
for i = 1 to t do
    r ← Σ_{S∈R'_2} I(P_S(x|N_i) = 0), by examining h_i
    score ← score + r
end for
return score
```

Complexity analysis. In the training stage, ZERO++ builds t q-sized probability tables, each using a subsample of n instances. Thus, ZERO++ has time complexity $O(ntq)$ since each table only needs to be updated a maximum of n entries after initialisation. During the testing stage, for each test instance, ZERO++ needs to look up t probability tables, where each table look up takes $O(q)$. To score d instances in a data set, ZERO++ has time complexity $O(dtq)$. Since d is far larger than n, the total time complexity of ZERO++ is $O(dtq)$. Note that, in order to calculate the probabilities, we do not need to obtain the frequency of all possible combinations of categorical values. We count the frequency of the values in the subspaces in $\mathcal{R}'_2$ in an incremental way, so the time complexity of ZERO++ is independent of the number of values in each attribute.

In terms of space complexity, ZERO++ needs to store t q-sized probability tables. Let ℓ be the average number of values per categorical attribute. For each probability table, $O(q\ell^2)$ is required to store values in its subspaces. Therefore, ZERO++ has space complexity $O(tq\ell^2)$.

Note that ZERO++ using $\mathcal{R}'_m$ with $3 \leq m < d$ for handling data sets having high-order feature dependence has the same time and space complexities as that using $\mathcal{R}'_2$, because $\mathcal{R}'_m$ is generated using the same process.

A comparison of time and space complexities between ZERO++, FPOF (He et al., 2005), COMPREX (Akoglu et al., 2012), iForest (Liu et al., 2012) and LOF (Breunig et al., 2000) is provided in Table 1. Both ZERO++ and iForest have linear time complexity w.r.t. both data size and dimensionality, and constant space complexity w.r.t. data size. FPOF and LOF have much higher time and space complexities than ZERO++ and iForest. The time complexity of FPOF is linear w.r.t. data size but exponential w.r.t. dimensionality, and it is affected by the length of the itemsets considered and the minimum *support* threshold. The time complexity of COMPREX is linear w.r.t. data size but quadratic w.r.t. data dimensionality, though it becomes near-linear w.r.t. the dimensionality when many attributes are correlated. Though the time complexity of LOF can be reduced to $O(d \log(d)\ q)$ from $O(d^2 q)$ when using some indexing scheme such as R^*-tree (Beckmann et al., 1990), most indexing schemes do not work on high-dimensional numeric data or data with categorical attributes.

Table 1: Time and space complexities of ZERO++, FPOF, COMPREX, iForest and LOF.

Method	Time Complexity	Space Complexity
ZERO++	$O(dtq)$	$O(tq\ell^2)$
FPOF	$O(d2^q)$	$O(2^q)$
COMPREX	$O(dq^2)$	$O(q^2)$
iForest	$O(dt)$	$O(tn)$
LOF	$O(d^2 q)$	$O(dq)$

4. Experimental Setup

Section 4.1 presents the parameter settings of ZERO++ and its contenders. It is followed by an introduction to the evaluation method and data sets in Section 4.2. We then introduce the significance test for comparing multiple detectors in the end.

4.1 Contenders and Their Parameter Settings

We compared ZERO++ with FPOF (He et al., 2005), COMPREX (Akoglu et al., 2012), iForest (Liu et al., 2012) and LOF (Breunig et al., 2000). FPOF is a state-of-the-art frequency-based method for categorical data. FPOF was selected as the frequency-based contender over another representative method LOADED (Ghoting et al., 2004) because, as reported in (Koufakou et al., 2007; Wu & Wang, 2013), FPOF performs more effectively than LOADED in a range of real-world data sets, and it also has lower time complexity than LOADED.

COMPREX was selected as a contender because it is related to ZERO++ in that both of them operate in a set of subspaces and it is a recently proposed state-of-the-art anomaly detector for categorical data.

LOF and iForest are state-of-the-art anomaly detection methods for numeric data. LOF has been recognised as a state-of-the-art method in handling numeric data sets, and it is

widely used as a performance baseline in the literature (Chandola et al., 2009). iForest is a recently proposed ensemble method for anomaly detection, and it is closely related to ZERO++, as discussed in Section 2.

ZERO++ and its four contenders used default settings in the experiments. Both ZERO++ and iForest employed $t = 50$ as default. The subsampling size n was 8 by default in ZERO++. As recommended in (Liu et al., 2012), iForest employed $n = 256$ as the default setting.

Following (He et al., 2005), FPOF employed the minimum *support* threshold $\delta = 0.1$ and the maximum length of itemsets is set to 5 by default. COMPREX is a parameter-free method. LOF often employed a small k value, e.g., $k = 10$ or $k = 20$, while we found LOF with $k = 150$ worked better in most of our data sets, as many data sets in our experiments have a large number of instances. Thus, LOF employed $k = 150$ as the default setting.

We implemented ZERO++ [5] in WEKA (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009). FPOF was implemented using the *Apriori* algorithm in WEKA. iForest is already in the WEKA platform, and LOF in the ELKI platform (Achtert, Kriegel, Schubert, & Zimek, 2013) was used in the experiments. R^*-tree indexing (Beckmann et al., 1990) was used by default in LOF. COMPREX was implemented by (Akoglu et al., 2012) in MATLAB. All the experiments were performed as a single-thread job at 2.27 GHz in a Linux cluster with 40GB memory.

Note that iForest and LOF are numeric data oriented methods. To run these methods on categorical data sets, the attributes were first converted into binary attributes using the 1-of-ℓ transformation method (Hall et al., 2009), i.e., an attribute with ℓ values is converted into ℓ binary attributes [6]. The binary attributes were then regarded as numeric attributes to be further processed by these two methods.

4.2 Detection Performance Measure and Data Sets

All the anomaly detectors produce a ranking based on their anomaly scores, i.e., top ranked instances are the most likely anomalies. The area under ROC curve (AUC) was derived based on the ranking (Hand & Till, 2001). Higher AUC indicates better detection accuracy. We also recorded the runtime to compare their efficiency. The AUC and runtime results were averaged over 10 runs for all randomised methods. The method that produces the best AUC performance for each data set is underlined in all relevant tables.

Experiments were conducted on 17 real-world data sets from the UCI repository (Bache & Lichman, 2013) and one synthetic data set having one normal cluster with two anomaly clusters, which was generated by the Mulcross data generator (Rocke & Woodruff, 1996). Following (He et al., 2005; Liu et al., 2012; Akoglu et al., 2012), the smallest class(es) or a small random subset of the smallest class was used as the anomaly class against the largest class(es) in a data set. A summary of the data sets is given in Table 2. These data

5. The code of ZERO++ is available at https://sites.google.com/site/gspangsite/sourcecode/zeroplusplus.
6. Alternative methods for the conversion includes occurrence frequency based methods and inverse occurrence frequency based methods (Boriah et al., 2008), but these conversion methods are designed for detectors requiring a distance function. If a natural ordering of categorical values is available, it is possible to convert each value to an integer which obeys the order. However, most categorical attributes have no such natural ordering. iForest does not work with these methods because it requires an ordering of all the instances in order to build the isolation trees. The 1-of-ℓ transformation method was used because it can be easily used by different types of detectors.

sets are from different application domains (e.g., health care, network security and image recognition) and widely used in the literature (Liu et al., 2012; Akoglu et al., 2012; Wu & Wang, 2013; Ting et al., 2013). The first six data sets with mixed-type attributes were used as categorical attributes only or numeric attributes only in our experiments.

We employed a commonly used performance evaluation method for unsupervised anomaly detection techniques (Liu et al., 2012). Specifically, we trained and evaluated detection models on the same data set, but it is assumed that class labels are unavailable in the training stage. The class labels are only used to compute the AUC in the evaluation stage.

Table 2: A summary of data sets used. #num and #cate denote the number of numeric and categorical attributes, respectively. The *Anomaly Class* column presents the anomaly class selected and its percentage in each data set. #binary is the total number of binary attributes converted by the 1-of-ℓ transformation from categorical attributes.

Data Set	d	q	#num	#cate	Anomaly Class	#binary
Linkage	5,749,132	7	2	5	match(0.36%)	5
Census	299,285	40	7	33	50K+(6.20%)	493
CoverType	286,048	54	10	44	cottonwood (0.96%)	44
Probe	64,759	41	34	7	attack(6.43%)	83
U2R	60,821	41	34	7	attack(0.37%)	83
Arrhythmia	452	279	206	73	arrhythmia(14.60%)	73
Nursery	4,648	8	0	8	very_recom (7.06%)	26
Chess	4,580	6	0	6	zero(0.59%)	39
Mushroom	4,429	22	0	22	poisonous(5.00%)	121
SolarFlare	1,066	10	0	10	flare X(0.47%)	29
Http	567,497	3	3	0	attack(0.39%)	0
Mulcross	262,144	4	4	0	two clusters(10.00%)	0
Smtp	95,156	3	3	0	attack(0.03%)	0
Shuttle	49,097	9	9	0	classes 2,3,5,6,7 (7.15%)	0
Mammo	11,183	6	6	0	class 1(2.32%)	0
Satimage	6,435	36	36	0	crop(10.92%)	0
Isolet	730	617	617	0	class Y (1.37%)	0
Mfeat	410	649	649	0	digit 0 (2.44%)	0

4.3 Statistical Significance Tests

For our empirical results, we also report the statistical significance test results at two levels: single and multiple data sets. For each data set, significance tests are based the average AUC over 10 runs and its two standard errors, i.e., if the lower bound of one detector is higher than the upper bound of another detector, then the former has outperformed the latter at 5% significance level.

The Friedman test with a post-hoc Nemenyi test is used to conduct significance tests over multiple data sets. The Friedman test is first used to rank all the detectors based on their average ranks, which are defined as $AR = \frac{1}{u}\sum_i r_i$ where u is the number of data sets used and r_i is the rank of a given detector on the i-th data set. The Nemenyi test is employed as a post-hoc test to check whether the performance of every pair of detectors,

represented by average ranks, is significantly different. The difference between two average ranks is deemed to be statistically significant if it differs by at least one *critical difference*, $CD = q_\alpha \sqrt{\frac{v(v+1)}{6u}}$, where q_α is derived from the Studentised range statistic and v denotes the number of the detectors. $\alpha = 0.05$ is used in our experiments.

5. Empirical Results

The AUC performance on categorical data and numeric data is summarised using the Friedman-Nemenyi test in Section 5.1, followed by the detailed performance results in the next two sections and a summary in the last section.

5.1 AUC Performance Summary

This section presents the summary of the empirical results conducted in the next two sections in terms of the Friedman-Nemenyi test.

The Friedman-Nemenyi test result of comparing the detectors over 10 categorical data sets is presented in Figure 6. It shows that ZERO++ and COMPREX are the top ranked detectors and outperforms iForest significantly, and they also outperforms FPOF. There is no significant difference among ZERO++, COMPREX and FPOF.

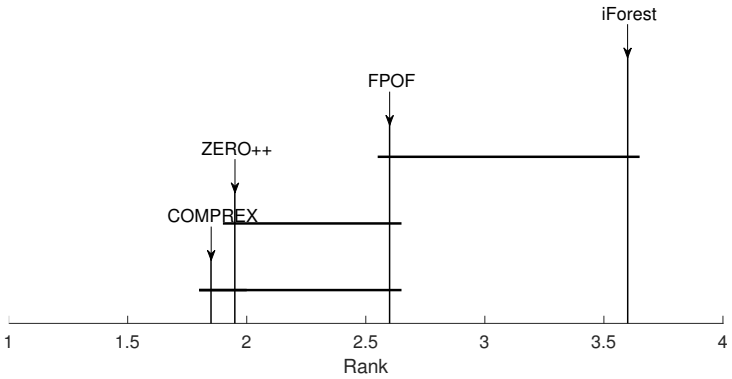

Figure 6: The Friedman-Nemenyi test result based on AUC performance of the four detectors on the 10 categorical data sets. The horizontal line indicates the critical difference (CD) between the average ranks. A detector outperforms another detector significantly if there is no connection between the two detectors by a CD line; otherwise there is no significant difference. LOF is excluded in this test because there is too many missing values in its AUC results. For detectors which have a few missing values in its AUC results, we replace the missing values with lowest AUC in the rows before conducting the tests.

Figure 7 presents the Friedman test result on numeric data sets. It shows that ZERO++(MS) is the top-ranked anomaly detector and performs significantly better than COMPREX,

FPOF and LOF [7]. iForest ranked second, and there is no significant difference between iForest and ZERO++.

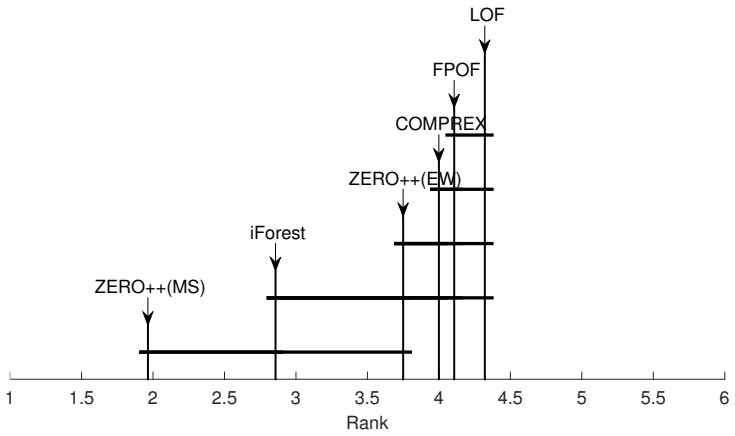

Figure 7: The Friedman-Nemenyi test result based on AUC performance of the five detectors on the 14 numeric data sets. MS and EW are two preprocessing methods which transform numeric attributes to categorical attributes. See Section 5.3.1 for the details.

The above results show that ZERO++ is the only anomaly detector which works well in both types of datasets. A numeric data based anomaly detector such as iForest works well in numeric data but works poorly in categorical data. Similarly, categorical data based COMPREX and FPOF work well in categorical data but work poorly in numeric data.

5.2 Results on Categorical Data

A series of experiments was conducted to evaluate ZERO++ in terms of effectiveness, the effect of varying ZERO++'s parameter m, scalability and sensitivity on categorical data. These are described in the following four subsections. We also tested ZERO++ on data sets without ground truth in the fifth subsection.

5.2.1 AUC Results

Table 3 shows the detection performance of ZERO++, FPOF, COMPREX, iForest and LOF on categorical data. ZERO++ is the best performer in four data sets, with three close to the best (having the difference in AUC less than 0.01). ZERO++ performs comparably to or significantly better than FPOF and COMPREX in eight data sets. Note that FPOF cannot work on the high dimensional data set, *Arrhythmia*, due to an out-of-memory exception error resulting from its high space complexity, even though the data set has no more than

7. Removing all data sets in which some algorithms could not complete their runs (see Tables 3 and 4 for details) yielded the same Friedman test results as presented in Figure 6 and Figure 7. The only exception is that there is no significant difference between ZERO++ and COMPREX in numeric data sets.

500 instances. ZERO++ outperforms iForest significantly in most data sets. ZERO++ has two wins, one draw and two losses against LOF. Note that LOF was unable to process large categorical data sets because the indexing method did not work in the data sets: *Linkage*, *Census*, *CoverType*, *Probe* and *U2R* [8]. ZERO++ performs less effective than other methods in some data sets such as *Mushroom*, because some anomalies in these data sets are only detectable in subspaces of size larger than two. A detailed analysis of this issue is presented in Section 5.2.2.

Table 3: AUC performance of ZERO++, FPOF, iForest and LOF with the default settings on categorical data. The best AUC in each data set is underlined.

	ZERO++	FPOF	COMPREX	iForest	LOF
Linkage	0.9973±0.0001	0.9972	0.9973	0.9790±0.0041	na
Census	0.6420±0.0056	0.6148	0.6352	0.5449±0.0172	na
CoverType	0.9946±0.0020	0.9965	0.9936	0.9773±0.0043	na
Probe	0.9802±0.0020	0.9867	0.9790	0.9776±0.0022	na
U2R	0.9891±0.0009	0.9156	0.9893	0.9729±0.0073	na
Arrhythmia	0.6588±0.0093	na	0.6848	0.6878±0.0024	0.6295
Nursery	1.0000±0.0000	1.0000	1.0000	0.9986±0.0010	1.0000
Chess	0.9774±0.0101	0.9122	0.9943	0.8606±0.0566	0.9945
Mushroom	0.9430±0.0047	0.9218	0.9359	0.9182±0.0117	0.9770
SolarFlare	0.9750±0.0052	0.9791	0.9793	0.9325±0.0070	0.9229
#na	0	1	0	0	5
ZERO++ vs. (#wins/draws/losses)		4/4/1	2/6/2	8/1/1	2/1/2

In addition to the above experiment using the default settings, we have also conducted a search for the best parameter setting for each algorithm (except COMPREX). It is interesting to note that the best AUC result for each algorithm from this experiment is not much different from that shown in Table 3. Significant differences only occur in a few data sets. The detailed results are shown in Appendix A.

5.2.2 The Effect of Varying Subspace Size m

Here we show that varying the default $m = 2$ can further improve the detection performance of ZERO++ in some situations. The 'right' value for m mainly depends on the attribute dependency in a particular data set. If the detection of anomalies relies on multiple attributes, then a large m shall be used; if it is depended on single attributes, then $m = 1$ is sufficient. Two such examples are provided in the following two paragraphs.

In *Mushroom*, many anomalies exhibit abnormal behaviours based on two or more attributes. Figure 8 shows the AUC results using $1 \leq m \leq 22$. It shows that the best results are obtained using $7 \leq m \leq 10$, and higher m values do not degrade the detection performance significantly. But $m < 6$ produce significantly worse results.

In *Nursery*, all anomalies can be detected by examining behaviours in the *Health* attribute. In this type of data, $m = 1$ produces the best result. This is because data subspace

8. For LOF, R^*-tree or other indexing methods in ELKI do not work on large categorical data because there are too many identical values in the attributes. Also, since data is pre-indexed by default in ELKI, LOF still cannot work on those data sets even though they do not use R^*-tree or other indexing methods.

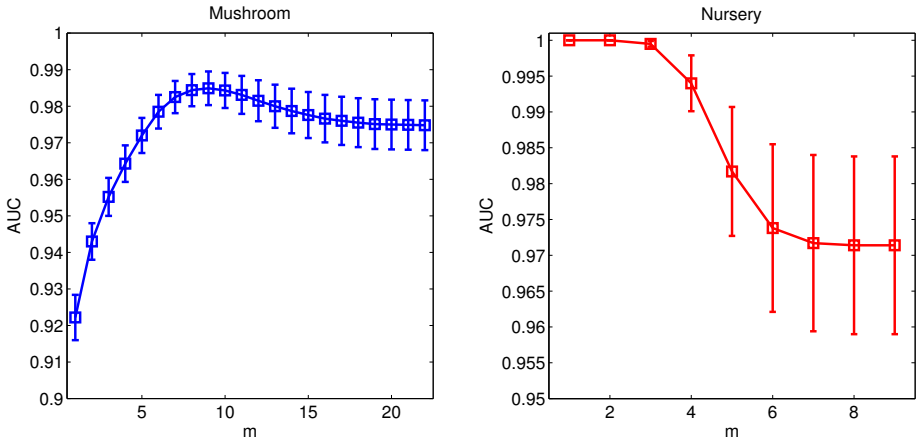

Figure 8: AUC results using $\mathcal{R}'_m$ with different m in *Mushroom* and *Nursery*. ZERO++ uses $t = 50$ and $n = 8$. The results are average over 10 runs and two standard errors.

becomes sparser with increasing number of dimensions, and normal instances also become rare instances in subspaces in $\mathcal{R}'_m$ with $m > 2$. As a result, normal instances can be incorrectly reported as anomalies. Figure 8 shows that the AUC performance decreases quickly with increasing m, and the results for $m > 3$ are significantly worse than those for $m \leq 3$.

Different data sets have different attribute dependencies of abnormal behaviours. Thus, a search of m is required to produce the best detection result for a particular data set. This is the same for most algorithms, e.g., LOF requires a search for a suitable k to yield the best detection result, for the k-nearest neighbour density estimator used in the algorithm.

5.2.3 Scalability Tests

The scalability tests employ data sets with binary attributes only to ensure that both categorical data oriented methods and numeric data oriented methods work on data sets with the same dimensionality.

We examine the scalability of ZERO++ w.r.t. data size using seven subsets of the largest data set *Linkage*. The smallest data subset contains 16,000 instances, and subsequent subsets are increased by a factor of four, until the largest subset which contains 4,096,000 instances. The results reported in Figure 9(a). shows that ZERO++, FPOF, COMPREX and iForest have runtime linear to the data size, and ZERO++ runs at least two orders of magnitude faster than LOF. LOF failed to work in data sets having 64,000 or more instances.

We examine the scalability of ZERO++ w.r.t. dimensionality using synthetic data sets. The data sets contain the same number of instances, i.e., 10,000 instances. The data set with the lowest dimensionality contains 10 attributes, and subsequent data sets are increased by a factor of two, until the data set with the highest dimensionality contains 320 attributes. The results are shown in Figure 9(b). The results show that both ZERO++ and LOF have runtime linear to the data dimensionality, and run more than four orders of magnitude faster than FPOF. Note that the space complexity of FPOF increases quickly with increasing dimensions, and FPOF runs out-of-memory when the dimension reaches 80. The runtime

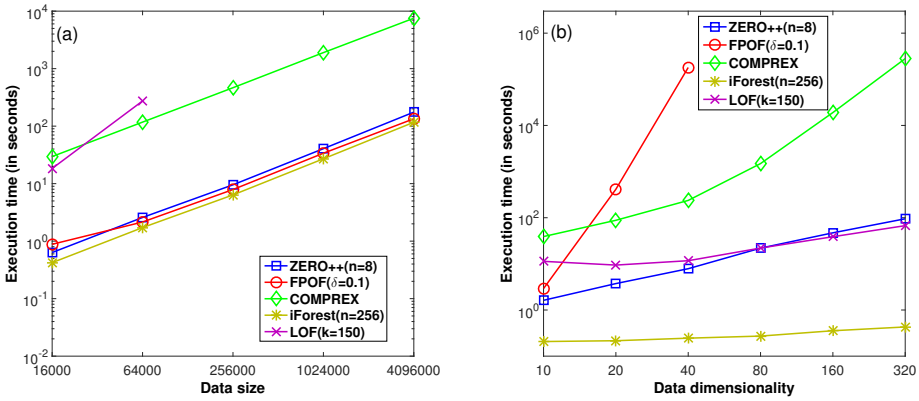

Figure 9: Scaleup tests of ZERO++ using FPOF, COMPREX, iForest and LOF as baselines. Figures (a) and (b) are the scaleup test results w.r.t. data size and dimensionality, respectively. The runtime is the average over 10 runs for the randomised methods.

of COMPREX increases by a factor of more than 7,000 when the dimensionality increases by a factor of 32; while that of ZERO++ increases by less than 60. Therefore, though ZERO++ and COMPREX were implemented in different programming languages, the increase in runtime ratio indicates that ZERO++ runs significantly faster than COMPREX by a factor of more than 110. The time complexity of iForest is constant w.r.t. data dimensionality since it works on a few randomly selected attributes only.

5.2.4 Sensitivity Tests

We investigated the sensitivity of ZERO++ w.r.t. n and t in all the data sets. We used the default setting for t when conducting the sensitivity test w.r.t. n, and vice versa.

Figure 10 reports the AUC mean values and two standard error bars over 10 runs of ZERO++ w.r.t. n in four selected data sets. The results show that although the detection performance of ZERO++ may vary with increasing subsample size, ZERO++ normally achieves the best performance using small subsample sizes, e.g., $\leqslant 64$. In data sets with complicated model complexity (e.g., distributions with a large number of modes), it needs to use a large subsampling size to do well. Examples are *Chess* and *Mushroom*. It should be noted that ZERO++ performs stably as the two standard errors are very small, e.g., they are often smaller than 0.01. Similar results can also be found in other data sets.

Figure 11 presents the sensitivity test results of ZERO++ w.r.t. t in the four selected data sets. The results show that the AUC performance of ZERO++ converges very quickly w.r.t. t. ZERO++ normally obtains the best performance and works stably using $t \geqslant 50$. Similar results can also be found in other data sets.

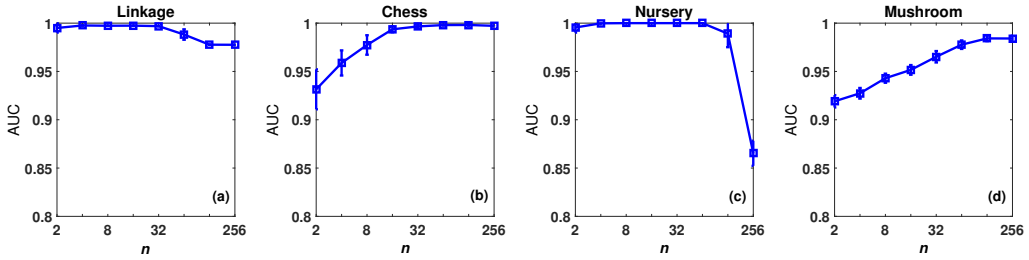

Figure 10: Sensitivity tests of ZERO++ ($t = 50$) w.r.t. n.

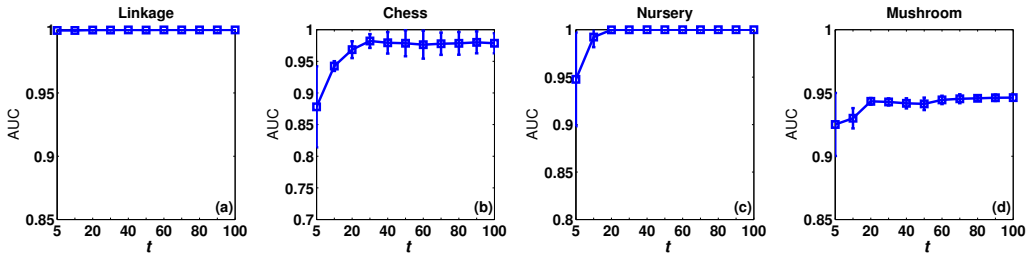

Figure 11: Sensitivity tests of ZERO++ ($n = 8$) w.r.t. t.

5.2.5 Application on Data Sets without Ground Truth

In this section, we investigated the application of ZERO++ on two UCI categorical data sets without ground truth, i.e., *Zoo* and *Internet Usage*. ZERO++ used $t = 50$ and $n = 8$ in the following experiments.

Zoo: *Zoo* has 17 attributes and 101 instances from 7 species of animals, including 8 instances of insect, 10 instances of invertebrate, 20 instances of bird, 41 instances of mammal, 13 instances of fish, 4 instances of amphibian and 5 instances of reptile. We ran ZERO++ on this data set and obtained the following top three anomalies:

- Honeybee is a top ranked anomaly because it is the only animal in the data set that is venomous and domestic.
- Scorpion is an unusual invertebrate animal because it has a breathing system and a tail, contrasting to all the other invertebrate animals.
- Octopus is an extreme case in the data set because none of the other animals has eight legs and is cat sized.

Internet usage: This data set comes from a survey about Internet usage in 1997. It consists of 10,104 instances with 71 categorical attributes plus an ID attribute. Each instance contains general demographic information about an Internet user. The top two ranked anomalies detected by ZERO++ are:

- User 99179 is a five-year old nurse in Denmark with 1-3 years Internet usage.
- User 91839 is a nine-year old Virginia male with a college degree and has a networking occupation but has less than six months on Internet.

5.3 Results on Numeric Data

Although ZERO++ is based on categorical attributes, here we show that it can be applied to numeric data by first converting numeric attributes to categorical attributes using two simple methods. The first subsection describes the discretisation methods used and the second subsection reports the empirical results.

5.3.1 Simple Numeric to Categorical Conversion Methods

ZERO++, FPOF and COMPREX are based on categorical data. Two discretisation methods, i.e., the equal-width method (Liu, Hussain, Tan, & Dash, 2002) and the $\overline{x} \pm 3s$ rule method, were used to enable them to work in numeric data. The Equal-Width (EW) method divides instances into a fixed number of bins of equal width in each attribute. The $\overline{x} \pm 3s$ rule (MS) method is a simple subsample-based method which converts a numeric attribute into a categorical attribute with two bins as follows. For each subsample, we compute the mean $\overline{x}$ and the standard deviation s for each attribute. If a numeric value falls within the range $[\overline{x} - 3s, \overline{x} + 3s]$, it is assigned the categorical value 'y'; otherwise the value 'n' is assigned. ZERO++ with the EW method and the MS method are denoted by ZERO++(EW) and ZERO++(MS), respectively. For the EW discretisation method, numeric attributes are discretised into 10 bins by default.

We have also tested FPOF and COMPREX with both EW and MS. Note that MS used in ZERO++ is based on $\overline{x}$ and s derived from each subsample. MS was adapted to FPOF and COMPREX by using $\overline{x}$ and s derived from the full data set. Our results showed that the detection performance of FPOF and COMPREX with EW was much better than that using MS. This is because the EW method discretised data in a much smaller granularity than the MS method. Therefore, FPOF and COMPREX with EW obtain significantly more patterns than those using MS, which help to capture the normal behaviours more effectively. Also, the full data set based MS discretisation can perform poorly, since $\overline{x}$ and s derived from the full data set are sensitive to anomalies. Thus, we report the results of FPOF and COMPREX with EW.

5.3.2 AUC Results

Table 4 shows the AUC performance of ZERO++(MS), ZERO++(EW), FPOF, COMPREX, iForest and LOF on the 14 numeric data sets. ZERO++(MS) is the best performer, which obtains the best performance in 8 out of 14 data sets. It outperforms FPOF and LOF significantly in eight data sets, and outperforms both COMPREX and iForest significantly in seven data sets.

ZERO++(EW) works less effectively than ZERO++(MS). It performs comparably to or significantly better than FPOF and COMPREX in six data sets, while it has five and six losses against FPOF and COMPREX, respectively. It performs significantly better than LOF in seven data sets, while has nine losses compared to iForest.

Note that we cannot obtain the results of FPOF in high dimensional data sets, including *HAR*, *Isolet*, *Mfeat* and *Arrhythmia*, due to out-of-memory exception errors. For COMPREX, we also cannot obtain its results in *HAR*, *Isolet* and *Mfeat* within two weeks. Also, the runtime for LOF is prohibitive in the two largest data sets (i.e., *Http* and *Linkage*) and we cannot obtain the results in two weeks.

Table 4: AUC performance of ZERO++, FPOF, COMPREX, iForest and LOF with the default settings on numeric data. COMP is for COMPREX. The best AUC in each data set is underlined.

	ZERO++(MS)	ZERO++(EW)	FPOF	COMP	iForest	LOF
Linkage	0.8772±0.0243	0.6390±0.0233	0.5469	0.5480	0.9974±0.0000	na
Census	0.7890±0.0171	0.7049±0.0101	0.7531	0.7462	0.6659±0.0072	0.6560
CoverType	0.9198±0.0058	0.6882±0.0075	0.5793	0.8491	0.8726±0.0173	0.6287
Probe	0.9900±0.0003	0.9924±0.0003	0.9943	0.9917	0.9652±0.0110	0.5537
U2R	0.9863±0.0005	0.9774±0.0012	0.9795	0.9801	0.9860±0.0015	0.5037
Arrhythmia	0.8137±0.0026	0.8102±0.0032	na	0.8147	0.7962±0.0104	0.8287
Http	0.9981±0.0012	0.9975±0.0001	0.9973	0.9978	0.9997±0.0001	na
Mulcross	0.9980±0.0009	0.6517±0.0632	0.9494	0.8997	0.9533±0.0085	0.5913
Smtp	0.8879±0.0082	0.5892±0.0000	0.5892	0.4977	0.8834±0.0058	0.8051
Shuttle	0.9984±0.0001	0.9862±0.0006	0.9751	0.9843	0.9957±0.0008	0.5295
Mammo	0.8386±0.0080	0.8554±0.0028	0.8537	0.8532	0.8484±0.0069	0.8645
Satimage	0.9856±0.0012	0.9677±0.0061	0.9756	0.9593	0.9804±0.0028	0.5896
Isolet	1.0000±0.0000	0.9987±0.0011	na	na	0.9997±0.0003	1.0000
Mfeat	0.9499±0.0060	0.9190±0.0146	na	na	0.9401±0.0124	0.9670
#na	0	0	3	2	0	2
ZERO++(MS) vs. (#wins/draws/losses)			8/1/2	7/3/2	7/5/2	8/1/3
ZERO++(EW) vs. (#wins/draws/losses)			4/2/5	5/1/6	2/3/9	7/0/5

It is interesting to see that ZERO++ performs better than state-of-the-art numeric data based detectors on numeric data sets, i.e., LOF and iForest. As ZERO++ is based on categorical data, a suitable discretisation method is important to enable ZERO++ to work well in numeric data. The MS discretisation method used in ZERO++ is simple but very effective for most data sets. This is due to the following two main reasons. First, compared to general discretisation methods such as EW which does not take into account the application context [9], MS is specially designed for anomaly detection and attempts to discretise normal instances and anomalies into different bins. Second, many real-world data sets follow *Gaussian distribution*, which is implicitly assumed in MS.

5.4 Section Summary

We summarise the above empirical results as follows.

- ZERO++ performs stably and is the only detector that performs well in both categorical data and numeric data using the default settings. It performs less effectively than its contenders in a few data sets (e.g., *Mushroom* and *Mammo*) mainly due to two reasons (i) some anomalies are detectable only when considering subspaces of size more than two; and (ii) a large subsampling size is required to capture anomalous behaviours in data sets with multi-modal distributions. With a search for either m or n, ZERO++ can achieve comparable or better results compared to its contenders in those data.

9. As the characteristics of one data set often differ substantially from the other, to get the best performance for a data set, the number of bins shall be tuned. In addition, more advanced discretisation methods are required in order to successfully apply FPOF and COMPREX to numeric data.

- As expected, the runtime of ZERO++ is linear w.r.t. both data size and data dimensionality. It runs two to four orders of magnitude faster than state-of-the-art categorical data based detectors FPOF and COMPREX.

- Among categorical data based methods, COMPREX is the most competitive contender to ZERO++ in terms of performance accuracy, but it spends a significant amount of time in searching informative subspaces. For FPOF, it is difficult to capture all the normal patterns in a data set, especially in large data sets; so a small δ is normally needed in FPOF in order to obtain a sufficient number of normal patterns.

- For numeric data based methods, LOF's detection performance is mainly dependent on the size of its neighbourhood set, which is strongly related to the data size. In general, they require a large neighbourhood size to perform well in large data sets while a small neighbourhood size is needed in small data sets. For iForest, it normally requires a larger subsample size ($n = 256$) than ZERO++ in order to perform well in large data sets mainly because it considers fewer subspaces than ZERO++.

6. Conclusions and Future Work

This is the first paper to show that the power of zero appearances can be harnessed to detect anomalies. ZERO++ is the only anomaly detector based on zero appearances, as far as we know. It is a simpler and more efficient non-search based anomaly detector than existing frequency-based anomaly detectors which rely on search. As a result, ZERO++ is a suitable candidate for large-scale data sets.

Our empirical evaluation shows that ZERO++ is competitive to or better than four state-of-the-art anomaly detectors in terms of detection accuracy, and it can scale up easily to large and high dimensional data sets, with orders of magnitude better than most existing anomaly detectors in terms of time and space complexities. Our experiments also demonstrate that ZERO++ discovers interesting anomalies on two real-world data sets.

Although ZERO++ is based on categorical data only, we have also shown that, with a simple method to convert numeric attributes to categorical attributes, ZERO++ also performs competitively to state-of-the-art numeric oriented anomaly detectors on numeric data.

ZERO++ is not suitable for data with complex feature dependency. For example, in data where the abnormal behaviours occur in subspaces of varying sizes, ZERO++ using subspaces with a single fixed subspace size cannot capture all zero appearances in subspaces of different sizes according to the closure property. One possible solution is to employ a search for different sizes of subspaces in conjunction with ZERO++ and learn appropriate weights for the subspaces. Also, a better discretisation method is required for ZERO++ to handle numeric data with multi-modal distribution or non-Gaussian distribution. We are interested in addressing these two issues in future work.

Acknowledgments

This material is based upon work partially supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-13-1-4043, awarded to Kai Ming Ting.

Appendix A. Best Detection Performance Results on Categorical Data

ZERO++ and its three contenders were examined with tuned parameters to obtain their best AUC for each data set. Both ZERO++ and iForest employed $t = 50$ as the default settings. For both methods, the best n was searched over the range 2, 4, 8, 16, 32, 64, 128 and 256.

Following (He et al., 2005), FPOF employed 5 as the maximum length of itemsets by default, and we searched the minimum *support* threshold δ over the range 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9, and report the best results.

For LOF, we searched k over the range 10, 20, 40, 60, 80, 150, 250, 300, 500, 1000, 2000, 3000 and 4000, and report the best results.

The best performance of ZERO++, FPOF, iForest and LOF is shown in Table 5. ZERO++ is significantly better than FPOF in three data sets, draws in five and loses in one. ZERO++ outperforms iForest significantly in 7 out of 10 data sets, with three draws and no loss. ZERO++ outperforms LOF significantly in three data sets, with two draws and no loss.

Table 5: AUC performance comparison between ZERO++, FPOF, iForest and LOF with the best parameter in categorical data.

	ZERO++	FPOF	iForest	LOF
	best n	best δ	best n	best k
Linkage	0.9976±0.0002	0.9978	0.9790±0.0041	na
Census	0.6465±0.0053	0.6148	0.5544±0.0286	na
CoverType	0.9954±0.0021	0.9965	0.9773±0.0043	na
Probe	0.9820±0.0025	0.9867	0.9776±0.0022	na
U2R	0.9910±0.0010	0.9203	0.9729±0.0073	na
Arrhythmia	0.6905±0.0012	na	0.6878±0.0024	0.6295
Nursery	1.0000±0.0000	1.0000	0.9995±0.0008	1.0000
Chess	0.9981±0.0010	0.9290	0.8606±0.0566	0.9948
Mushroom	0.9842±0.0023	0.9400	0.9182±0.0117	0.9770
SolarFlare	0.9784±0.0011	0.9791	0.9641±0.0057	0.9778
#na	0	1	0	5
#wins/draws/losses		3/5/1	7/3/0	3/2/0

References

Achtert, E., Kriegel, H., Schubert, E., & Zimek, A. (2013). Interactive data mining with 3d-parallel-coordinate-trees. In *SIGMOD*, pp. 1009–1012.

Akoglu, L., Tong, H., Vreeken, J., & Faloutsos, C. (2012). Fast and reliable anomaly detection in categorical data. In *CIKM*, pp. 415–424. ACM.

Bache, K., & Lichman, M. (2013). UCI machine learning repository. *URL http://archive.ics.uci.edu/ml*.

Barron, A., Rissanen, J., & Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, *44*(6), 2743–2760.

Beckmann, N., Kriegel, H.-P., Schneider, R., & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*, pp. 322–331. ACM.

Boriah, S., Chandola, V., & Kumar, V. (2008). Similarity measures for categorical data: A comparative evaluation. In *SDM*, pp. 243–254. SIAM.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. *ACM SIGMOD Record*, *29*(2), 93–104.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 15.

Duan, L., Xu, L., Liu, Y., & Lee, J. (2009). Cluster-based outlier detection. *Annals of Operations Research*, *168*(1), 151–168.

Erfani, S., Baktashmotlagh, M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2015). R1SVM: A randomised nonlinear approach to large-scale anomaly detection. In *AAAI*, pp. 432–438.

Ghoting, A., Otey, M. E., & Parthasarathy, S. (2004). LOADED: Link-based outlier and anomaly detection in evolving data sets. In *ICDM*, pp. 387–390. IEEE.

Görnitz, N., Kloft, M., Rieck, K., & Brefeld, U. (2013). Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, *46*, 235–262.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18.

Hand, D. J., & Till, R. J. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, *45*(2), 171–186.

He, Z., Deng, S., Xu, X., & Huang, J. Z. (2006). A fast greedy algorithm for outlier mining. In *PAKDD*, pp. 567–576. Springer.

He, Z., Xu, X., Huang, Z. J., & Deng, S. (2005). FP-outlier: Frequent pattern based outlier detection. *Computer Science and Information Systems/ComSIS*, *2*(1), 103–118.

Jin, H., Chen, J., He, H., Kelman, C., McAullay, D., & O'Keefe, C. M. (2010). Signaling potential adverse drug reactions from administrative health databases. *IEEE Transactions on Knowledge and Data Engineering*, *22*(6), 839–853.

Koufakou, A., & Georgiopoulos, M. (2010). A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes. *Data Mining and Knowledge Discovery*, *20*(2), 259–289.

Koufakou, A., Ortiz, E. G., Georgiopoulos, M., Anagnostopoulos, G. C., & Reynolds, K. M. (2007). A scalable and efficient outlier detection strategy for categorical data. In *ICTAI*, pp. 210–217. IEEE.

Lazarevic, A., & Kumar, V. (2005). Feature bagging for outlier detection. In *SIGKDD*, pp. 157–166. ACM.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, *6*(1), 3:1–3:39.

Liu, H., Hussain, F., Tan, C. L., & Dash, M. (2002). Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, *6*(4), 393–423.

Pang, G., Ting, K. M., & Albrecht, D. (2015). Lesinn: Detecting anomalies by identifying least similar nearest neighbours. In *ICDMW*, pp. 623–630. IEEE.

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record*, *29*(2), 427–438.

Rocke, D. M., & Woodruff, D. L. (1996). Identification of outliers in multivariate data. *Journal of the American Statistical Association*, *91*(435), 1047–1061.

Smets, K., & Vreeken, J. (2011). The odd one out: Identifying and characterising anomalies. In *SDM*, pp. 109–148. SIAM.

Sugiyama, M., & Borgwardt, K. (2013). Rapid distance-based outlier detection via sampling. In *NIPS*, pp. 467–475.

Ting, K. M., Zhou, G.-T., Liu, F. T., & Tan, S. C. (2013). Mass estimation. *Machine Learning*, *90*(1), 127–160.

Wallis, S. (2013). Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods. *Journal of Quantitative Linguistics*, *20*(3), 178–208.

Wu, M., & Jermaine, C. (2006). Outlier detection by sampling with accuracy guarantees. In *SIGKDD*, pp. 767–772. ACM.

Wu, S., & Wang, S. (2013). Information-theoretic outlier detection for large-scale categorical data. *IEEE Transactions on Knowledge and Data Engineering*, *25*(3), 589–602.

Xu, J., & Shelton, C. R. (2010). Intrusion detection using continuous time bayesian networks. *Journal of Artificial Intelligence Research*, *39*, 745–774.

Zhang, K., Hutter, M., & Jin, H. (2009). A new local distance-based outlier detection approach for scattered real-world data. In *PAKDD*, pp. 813–822. Springer.

Zimek, A., Campello, R. J., & Sander, J. (2013a). Ensembles for unsupervised outlier detection: Challenges and research questions. *ACM SIGKDD Explorations Newsletter*, *15*(1), 11–22.

Zimek, A., Gaudet, M., Campello, R. J., & Sander, J. (2013b). Subsampling for efficient and effective unsupervised outlier detection ensembles. In *SIGKDD*, pp. 428–436. ACM.