Data Min Knowl Disc (2010) 20:259–289
DOI 10.1007/s10618-009-0148-z

A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes

Anna Koufakou · Michael Georgiopoulos

Received: 22 May 2008 / Accepted: 24 August 2009 / Published online: 11 November 2009
The Author(s) 2009

Abstract Outlier detection has attracted substantial attention in many applications and research areas; some of the most prominent applications are network intrusion detection or credit card fraud detection. Many of the existing approaches are based on calculating distances among the points in the dataset. These approaches cannot easily adapt to current datasets that usually contain a mix of categorical and continuous attributes, and may be distributed among different geographical locations. In addition, current datasets usually have a large number of dimensions. These datasets tend to be sparse, and traditional concepts such as Euclidean distance or nearest neighbor become unsuitable. We propose a fast distributed outlier detection strategy intended for datasets containing mixed attributes. The proposed method takes into consideration the sparseness of the dataset, and is experimentally shown to be highly scalable with the number of points and the number of attributes in the dataset. Experimental results show that the proposed outlier detection method compares very favorably with other state-of-the art outlier detection strategies proposed in the literature and that the speedup achieved by its distributed version is very close to linear.

Keywords Outlier detection · Anomaly detection · Data mining · Distributed data sets · Mixed attribute data sets · High-dimensional data sets

Responsible editor: Sanjay Chawla.

A. Koufakou (✉)
U.A. Whitaker School of Engineering, Florida Gulf Coast University, Fort Myers, FL 33965, USA
e-mail: akoufakou@fgcu.edu

A. Koufakou · M. Georgiopoulos
School of EECS, University of Central Florida, Orlando, FL 32816, USA

M. Georgiopoulos
e-mail: michaelg@mail.ucf.edu

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 (Dokas et al. 2002). Outlier detection approaches focus on discovering patterns that occur infrequently in the data, as opposed to many traditional data mining techniques, such as association analysis or frequent itemset mining, 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 (1980): "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" (Knorr and Ng 1998). Finally, outlier detection strategies can also be used for data cleaning as a step before any traditional mining algorithm is applied to the data. Examples of applications where the discovery of outliers is useful include detecting irregular credit card transactions, indicating potential credit card fraud (Bolton and Hand 2002), or identifying patients who exhibit abnormal symptoms due to their suffering from a specific disease or ailment (Penny and Jolliffe 2001).

Most of the research efforts in outlier detection have focused on datasets that consist of one type of attribute, i.e. only numerical attributes or ordinal attributes that can be directly mapped into numerical values, or only categorical attributes. Quite often, when we have data containing categorical attributes, for example, it is assumed that the categorical attributes could be easily mapped into numerical values. However, there are cases, where mapping categorical attributes 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 value (married or single) or a person's profession (engineer, financial analyst, etc.) to a numerical value.

In the case of continuous attributes, algorithms designed for categorical data often use discretization techniques to map intervals of continuous space into discrete values. Among unsupervised methods, i.e. those that do not use class information, popular algorithms include equal-width or equal-frequency binning. These methods might not produce good results when the distributions of the continuous attribute values are not uniform, and the resulting discretized ranges are significantly affected by outliers (Catlett 1991). In our case, for example, values that are associated with normal points and values associated with outliers might be combined in the same bin, which makes the outlier detection task more difficult.

A *second* issue that has only recently become the focus in the literature is related to the large and distributed nature of the datasets currently available. With the explosion of technology, the size of data for a particular application has grown and will continue to grow; terabyte-scale data is now common, e.g., Wal-Mart had 460 terabytes of data in 2004 (Hays 2004). Also, data may be distributed among different sites belonging to the same or different organizations. Transferring data to a central location and then detecting outliers is typically impractical due to the data size and expense of moving the data, as well as data ownership and control issues. Hence, successful outlier detection strategies must scale well as the number of data points and dataset dimensionality grow. Furthermore, in order to deal with the distributed nature of the data, communication overhead and synchronization between the different sites in which the data reside, as well as the data scans should be minimal.

A *third* issue is the high dimensionality of currently available datasets. Due to the large number of dimensions, the dataset becomes sparse, and in this type of setting, traditional concepts such as Euclidean distance between points, and nearest neighbor, become irrelevant (Ertoz et al. 2003). Employing dissimilarity measures that can handle sparse data becomes imperative. In addition, inspecting several, smaller *views* of the large, high-dimensional data can help uncover outliers, which would otherwise be *masked* by other outlier points if one were to look at the entire dataset at once.

In this paper, we propose an outlier detection approach to address all of the issues outlined above. Specifically, the main contributions of this paper are:

– We propose a method that relies on an intuitive anomaly score for categorical attributes and at the same time efficiently handles sparse continuous data; this method is fast, scalable, and uses two data passes;
– We extend this method for datasets that are distributed amongst different geographical sites; our method consists of two phases with low communication and synchronization overhead;
– We compare our method with the state-of-the-art distributed outlier detection method for mixed attributes (Otey et al. 2006); the results show that our method exhibits an overall higher detection rate, a lower false positive rate, and is orders of magnitude faster;
– We perform extensive experimentation in an effort to justify the reasoning behind the proposed outlierness scores, and the parameter values utilized.

The organization of this paper is as follows: Sect. 2 contains an overview of the previous research related to outlier detection. In Sect. 3, we present our outlier detection approach, called outlier detection for mixed attribute datasets (ODMAD), and its distributed version. Section 4 includes our experimental results and associated conclusions, while the overall summary of our work and high level conclusions are provided in Sect. 5.

2 Related work

Statistical outlier detection was one of the earliest approaches. Statistical-based methods assume that a parametric model (usually univariate) describes the distribution of the data (Barnett and Lewis 1978). Multivariate statistical approaches have been proposed, including use of robust (less affected by outliers) estimates of the multidimensional distribution parameters, e.g. minimum covariance determinant (MCD) and minimum volume ellipsoid (MVE) (Rousseeuw 1985; Rousseeuw and Leroy 1987). One inherent problem of statistical-based methods is finding the suitable model for each dataset and application; also, as data increases in dimensionality, it becomes increasingly more challenging to estimate the multidimensional distribution (Aggarwal and Yu 2001; Tan et al. 2005).

As the data increases in dimensionality, data is spread through a larger volume and become sparse. In addition to the slowing effect this has on performance, it also spreads the convex hull, thus distorting the data distribution [“Curse of Dimensionality” (Hodge and Austin 2004)]. This can be alleviated by preselecting the most significant features to work with (e.g. Aha and Bankert 1994), projecting to a lower-dimensional subspace (Aggarwal and Yu 2001), or applying Principal Component Analysis (PCA). Another approach to deal with higher dimensionalities is to organize data points in convex hull layers according to their peeling depth, based on the idea that outliers are data objects with a shallow depth value, e.g. (Preparata and Shamos 1985); these ideas quickly become computationally impractical.

Distance-based techniques do not make assumptions for the data since they basically compute the distance between each point. For example, Knorr and Ng (1998) proposed a k-NN approach where if m of the k nearest neighbors of a point are within a specific distance d then the point can be classified as normal. Knorr et al. (2000) define a point as an outlier if at least $p\%$ of the points in the dataset lie further than distance d from it. These methods exhibit high computational complexity (e.g. nearest neighbor based methods have quadratic complexity with respect to the dataset size) rendering them impractical for really large datasets. Several methods may be employed to make the k-NN queries faster (to achieve linear or logarithmic time), such as an indexing structure (e.g. KD-tree, or X-tree); however these structures have been shown to break down as the dimensionality grows (Bay and Schwabacher 2003): for example, Breunig et al. (2000) used an X-tree index and state that its performance degenerated for dimensionality ≥ 10. Bay and Schwabacher (2003) proposed a distance-based algorithm based on randomizing the data for efficient pruning of the search space, which in practice has complexity close to linear. This method was shown to have lower accuracy and slower performance for large data when compared to the method in Otey et al. (2006).

More recently, there have been efforts towards distance-based approaches in sub-quadratic time. Angiulli and Pizzuti (2005) used a Hilbert Space Filling Curve to assist with k-NN computations; however their method requires $d + 1$ scans of the data where d is the dataset dimensionality, which again might be impractical for distributed, high-dimensional data. Besides efficiency, a general limitation of distance-based methods is apparent in complex datasets that contain outliers which are only obvious when one looks locally at the neighborhood of each point. Distance-based methods using a global outlier criterion fail to identify these outliers. Several clustering methods can be used (e.g. k-means, k-medoids, etc.) to form clusters, based on the idea that outliers are points not included in formed clusters. In general, all clustering-based methods rely on the clusters to define outliers, thus focus in optimizing clustering, not outlier detection (Knorr et al. 2000).

Density-based methods estimate the density distribution of the input space and then identify outliers as those lying in regions of low density. Older examples include using Gaussian mixture models (GMMs), e.g. (Roberts and Tarassenko 1994). Breunig et al. (2000) assign a degree of outlierness to each data point, the local outlier factor (LOF), based on the local density of the area around the point and the local densities of its neighbors, relying on a user-given minimum number of points. Extensions include Local Correlation Integral (LOCI) which uses statistical values to avoid user entered parameters (Papadimitriou et al. 2003), and using kernel density functions (Latecki et al. 2007). These techniques are able to detect outliers that are missed by techniques with a single, global criterion such as distance-based techniques (Knorr et al. 2000). These methods are also based on distance computations which might be inappropriate for categorical data, and again not straightforward to implement in a distributed setting.

In addition, high-dimensional data is almost always sparse, which creates problems for density-based methods (Tan et al. 2005). In such data, the traditional Euclidean notion of density, i.e. number of points per unit volume, becomes meaningless. The increase in dimensionality leads to increase of volume, and density tends to zero as the number of dimensions grows much faster than the number of data points (Ertoz et al. 2003). The notion of a nearest neighbor or a similar data point does not hold as well because the distance between any two data points becomes almost identical (Beyer et al. 1999). To deal with sparse data, Ertoz et al. (2003) use the cosine function and shared nearest neighbors to cluster data with high dimensionality.

Other outlier detection efforts include *Support Vector* methods, e.g., (Tax and Duin 2004), and *Replicator Neural Networks* (Hawkins et al. 2002) among others. There has been much work related to intrusion detection, e.g. (Lazarevic et al. 2003). Many methods published in this area are particular to intrusion detection and cannot be generalized to other outlier detection areas.

In general, most of the existing research in outlier detection is geared towards datasets containing a specific attribute type, i.e. they assume that attributes are only numerical and/or ordinal (can be directly mapped into numerical values), or only categorical. In the case of categorical data, there is little sense in ordering categorical values, then mapping them to numerical values and computing distances, e.g., distance between two values such as TCP protocol and UDP protocol (Otey et al. 2006). Consequently, methods such as those based on distance are unsuitable.

On the other hand, methods that assume categorical attributes rely on discretization to map continuous data to categorical which can lead to loss of information, exacerbated by high dimensionality and sparseness of the datasets. There are many supervised discretization methods, however there has been little work in unsupervised discretization where class label information is not known beforehand, as in outlier detection applications. Commonly used methods such as binning are unreliable (Biba et al. 2007), and the situation becomes worse for large datasets. Recent unsupervised methods (Mehta et al. 2005; Biba et al. 2007) are more sophisticated, but do not scale linearly with the number of data points and/or the number of attributes, and cannot easily extend to sparse or distributed datasets.

There have been some outlier detection efforts geared towards categorical data such as He et al. (2006). Koufakou et al. (2007) experimented with categorical outlier detection approaches, and proposed AVF (attribute value frequency), a simple, fast, and scalable method for categorical sets. Koufakou et al. (2008a) proposed a parallel AVF method using the MapReduce paradigm of parallel programming (Dean and Ghemawat 2004), which ensures fault tolerance and load balancing. Yu et al. (2006) used mutual reinforcement to detect outliers in mixed attribute data, but they address a different outlier detection problem than the one described here.

Many previous methods quickly become very slow as the number of points increases. For distributed data, techniques based on pair-wise computations become infeasible as the different sites would have to exchange every local point or replicate all points globally in order to calculate distances. Branch et al. (2006) proposed a distributed outlier detection method, but their approach is designed for wireless sensor networks. Otey et al. (2006) presented a distributed outlier detection method for evolving datasets with mixed attributes. Their technique is based on the concept of frequent itemsets (Agrawal and Srikant 1994) to deal with categorical attributes, and the covariance for continuous attributes. Their method combines the categorical and continuous spaces as follows: for each possible categorical set (combination of categorical values), they isolate the data points that contain each set, then calculate the covariance matrix of the continuous values of these points. A point is likely to be an outlier if it contains infrequent categorical sets, or if its continuous values differ from the covariance. These quantities (e.g. frequencies, covariance) can be independently calculated on each site, and then combined to create a global model.

Although this approach has linear complexity with respect to the number of data points, it is exponential in the number of categorical attributes and quadratic in the number of continuous attributes. Maintaining a covariance matrix for each set of categorical values leads to prohibitive memory load and runtime, even after restricting the length of sets stored. As high-dimensional continuous data is sparse, calculating the covariance for each pair of continuous values might not be necessary. Maintaining a covariance matrix for frequent sets might also be superfluous, as some of these sets (e.g. those with frequency 90–99% of the data) correspond to more or less the same continuous values. Finally, we might encounter the following scenario: an outlier point, P_1, might be irregular only in its categorical values, while another outlier point, P_2, might have both anomalous categorical and continuous values; then P_1 will receive a lower outlier score than P_2, which may lead to missing point P_1 due to outlier P_2.

In this paper, we extend our work in Koufakou et al. (2007) and propose outlier detection for mixed attribute datasets (ODMAD), an outlier detection approach for categorical and continuous attributes, designed for high-dimensional distributed data. Preliminary results for the centralized version of ODMAD were presented in Koufakou et al. (2008b). ODMAD exhibits very good accuracy and performance, and it is highly scalable with the number of points and dimensions. We compare ODMAD to the state-of-the-art distributed outlier detection method for mixed attributes in Otey et al. (2006).

3 Algorithms

In this section we describe our approach, outlier detection for mixed attribute datasets (ODMAD), for mining outliers from data containing both categorical and continuous attributes. Table 1 contains the notation used in this paper.

Our approach first computes an anomaly score for each point taking into consideration the irregularity of the categorical values, the continuous values, and the relationship between the two spaces in the dataset. We present a centralized outlier detection strategy using this score, followed by a distributed outlier detection approach. This distributed method requires only one round of communication, as nodes need only

Table 1 Notation

Term	Definition
$\mathcal{D}$	Dataset
n	Number of data points in $\mathcal{D}$
m_c	Number of categorical attributes in $\mathcal{D}$
m_q	Number of continuous attributes in $\mathcal{D}$
v	Number of distinct values per categorical attribute
i	The index of a data point, $i = 1 \ldots n$
$\mathbf{x}_i$	The i-th data point in $\mathcal{D}$
$\mathbf{x}_i^q$	The continuous values of $\mathbf{x}_i$
$\mathbf{x}_i^c$	The categorical values of $\mathbf{x}_i$
l	Index of a categorical attribute; $l = 1 \ldots m_c$
j	Index of a continuous attribute; $j = 1 \ldots m_q$
r	Index of node in the distributed setting; $r = 1 \ldots R$
a	A categorical value
d	Set of categorical values (itemset)
$\|d\|$	Number of categorical values in set d; $1 \leq \|d\| \leq m_c$
$\mathcal{I}$	Set of all possible combinations of attributes and their values (distinct single items) in $\mathcal{D}$
σ	Minimum support
supp(d)	Support (frequency) of set d
MAXLEN	Maximum length of itemset d
μ_a	Mean vector of $\mathbf{x}_i^q$ for all $\mathbf{x}_i$ such that $\mathbf{x}_i^c$ contains categorical value a

exchange their local model once, in order to construct the global model. Moreover, this approach does not place heavy demands on the individual local nodes (light memory load). We note that Otey et al. (2006) also included an extension to their algorithm to handle dynamically changing datasets. ODMAD can be easily modified for dynamically changing data by a similar approach as in Otey et al. (2006) and it is not presented here.

3.1 Centralized algorithm

The outlier detection proposed in this paper, ODMAD, detects outliers assuming that, in the categorical space, outliers are points with highly irregular or infrequent values. Koufakou et al. (2007) showed how this idea could be used to effectively detect outliers in categorical data. The method in this paper goes a step further to explore outliers in the categorical and the continuous attribute space. In this mixed space, an outlier would be irregular or anomalous in either categorical space only (type a), or continuous space only (type b), or both (type c). The steps of ODMAD are presented below:

In the first step, we inspect the categorical space in order to detect data points with irregular categorical values. This enables us to detect outliers of type a and type c.

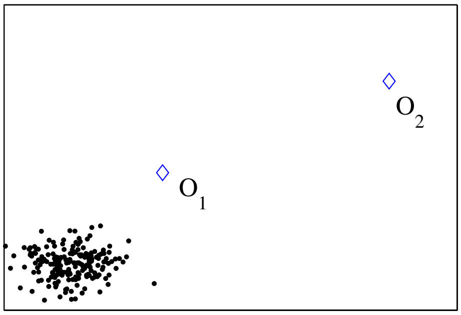

Fig. 1 Masking effect—Outlier point O_2 is more irregular than the normal points and outlier point O_1, therefore O_2 will likely mask O_1

In the second step, we 'set aside' the points found as being irregular or 'outlying' from the first step, and then focus on the remaining points, in an attempt to detect the rest of the outliers (type b). Based on the categorical values of the remaining points, we concentrate on subsets extracted from these data, and work only on these subsets, one at a time. These subsets are considered so that we can identify outliers that would have otherwise been missed (masked) by more irregular outliers.

To illustrate our point, consider the scenario in Fig. 1. This figure serves only to depict a scenario for masking, and not the high-dimensional issue we are addressing later in this paper. Outlier point O_2 is extremely irregular with respect to the rest of the data points, while the second outlier, point O_1, is closer to the normal data points. In this case, outlier point O_2 masks the other outlier point, O_1. One solution to this problem could be to sequentially remove outlier points from the dataset. This would imply several passes over the data, which is impractical in the case of large or distributed datasets. In Sect. 3.1.4, we discuss in more detail how we alleviate this masking issue by considering subsets of the data.

3.1.1 Categorical score

As shown in Koufakou et al. (2007), the 'ideal' outlier point in a categorical dataset is one for which each and every one of its values 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. Koufakou et al. (2007) assigned a score to each data point that reflects the frequency with which each value of the point occurs. In this paper, we extend this notion of 'outlierness' by stating that a point is more likely to be an outlier if it contains irregular values or irregular sets of values. This extension covers the likely scenario where none of the single values in an outlier point are infrequent, but the co-occurrence of two or more of its attribute values in the dataset is infrequent.

We consider a dataset $\mathcal{D}$ which contains n data points, $\mathbf{x}_i$, $i = 1 \ldots n$. If each data point $\mathbf{x}_i$ has m_c categorical attributes, we write $\mathbf{x}_i = [x_{i1}, \ldots, x_{il}, \ldots, x_{im_c}]$, where x_{il} is the value of the l-th attribute of $\mathbf{x}_i$. Our anomaly score for each point makes use of the idea of an itemset (or set) from the frequent itemset mining literature (Agrawal and Srikant 1994). Let $\mathcal{I}$ be the set of all possible combinations of attributes and their values in the dataset $\mathcal{D}$. Then let S be the set of all sets d so that an attribute occurs only once in each set d:

$$S = \{d : d \in \text{PowerSet}(\mathcal{I}) \ \wedge \ \forall \ l, k \in d, \ l \neq k\}$$

where l, k represent attributes whose values appear in set d. We also define the length of set d, $|d|$, as the number of attribute values in d, and the *frequency* or *support* of set d, supp(d), as the number of data points $\mathbf{x}_i$ in dataset $\mathcal{D}$ which contain set d.

Following the reasoning stated earlier, a point is likely to be an outlier if it contains single values that are infrequent or sets of values that are infrequent. We say that a categorical value or a combination of values is infrequent if it appears less than σ times in our dataset, where σ is a user-defined threshold. Therefore, a good indicator to decide if point $\mathbf{x}_i$ is an outlier in the categorical attribute space is the score value, Score$_1$, defined below:

$$\text{Score}_1(\mathbf{x}_i) = \sum_{d \subseteq \mathbf{x}_i \ \wedge \ \text{supp}(d) < \sigma \ \wedge \ |d| \leq \text{MAXLEN}} \frac{1}{\text{supp}(d) \times |d|} \quad (1)$$

Essentially, we assign an anomaly score to each data point that depends on the infrequent subsets contained in this point. Otey et al. (2006) showed that we can obtain good detection accuracy by considering sets of length up to a user-entered MAXLEN. For example, let $\mathbf{x}_i = [a \ b \ c]$, and MAXLEN = 3, the possible subsets of are: a, b, c, ab, ac, bc, and abc. If subset d of $\mathbf{x}_i$ is infrequent, i.e. supp$(d) < \sigma$, we increase the score of $\mathbf{x}_i$ by the inverse of supp(d) times the length of d. In our example, if supp$(ab) = 3$ and $\sigma = 5$, ab is an infrequent subset of $\mathbf{x}_i$, and Score$_1(\mathbf{x}_i)$ will increase by the inverse of 3×2 or 0.167.

A higher Score$_1$ implies that it is more likely that the point is an outlier. If a point does not contain any infrequent subsets its score will be zero. The score in Eq. 1 is inversely proportional to the frequency, as well as to the length of each set d that occurs in point $\mathbf{x}_i$. Therefore, a point that has very infrequent single values will get a very high score; a point with moderate infrequent single values will get a moderately high score; and a point whose single values are all frequent and has a few infrequent subsets will get a moderately low score.

Based on Eq. 1, data points that have several infrequent categorical values are more likely to be outliers. For example, a point that contains two infrequent values is less likely to be an outlier than a point with ten infrequent values. As we add more and more values to set d, it is expected that the frequency of d will decrease. So, by taking the inverse of the length, we weigh single values more than their combinations.

We note that Score$_1$ is similar to the one in Otey et al. (2006); however, the latter score does not make any distinction between sets of different frequency. We use the frequency of the categorical value sets to further distinguish between data points that contain the same number of infrequent values. The benefit of our score becomes pronounced with larger datasets: for example, consider a dataset with a million data points and σ of 100 thousand or 10% of the data set. Also assume two categorical values, value a that appears only once in the dataset, and value b, that appears in the dataset slightly less than a hundred thousand times. Using our score, a data point containing value a (very infrequent) will have a much higher score than a point with value b. Using the score in Otey et al. (2006) the two values would contribute the same to the score. Therefore, our score better reflects the amount of irregularity of the categorical values in the dataset.

3.1.2 Modified categorical score

In the previous section, Eq. 1 implies that we have to increase the score for each and every infrequent subset of a data point starting with length 1 up to length MAXLEN. We can compute the score more efficiently by not considering all possible infrequent subsets of a data point $\mathbf{x}$.

First, note that if a set of categorical values, d, is infrequent, then all possible supersets of d are infrequent as well. This means that we can avoid calculations as follows: if we discover that set d belonging to point $\mathbf{x}$ is infrequent we do not consider any more combinations containing d. As our score in Eq. 1 is inversely proportional to the length of a set, and the frequency of a set is a good approximation for the frequency of its supersets, we are not missing valuable information.

Instead of gathering the frequency of all possible infrequent subsets of each point for the score in Eq. 1, we only gather the frequencies of certain categorical sets: the pruned candidates. This is also called negative border of frequent sets. Pruned candidates are those infrequent sets such that all their subsets are frequent. These are basically the sets that are pruned at each phase of a Frequent Itemset Mining algorithm such as Apriori (Agrawal and Srikant 1994). We show in the experiments section that this modification to the categorical score does not lead to reduction in the detection accuracy of our algorithm. In the "Appendix", we discuss a bound to the computational savings due to this modification. In the following example, we describe this process.

Example Assume we have two points, each with three categorical attributes: $\mathbf{x}_1 = [a\ b\ c]$ and $\mathbf{x}_2 = [a\ b\ d]$. If only single values a and c are infrequent with support equal to 5, the score is as follows:

$$\text{Score}_1(\mathbf{x}_1) = \frac{1}{\text{supp}(a)} + \frac{1}{\text{supp}(c)} = \frac{2}{5} = 0.4,\ \text{Score}_1(\mathbf{x}_2) = \frac{1}{\text{supp}(a)} = \frac{1}{5} = 0.2.$$

Since a and c are infrequent, we do not check any of their combinations with other values because they will also be infrequent. The sets we do not check are: ab, ac, ad, bc, cd. However, bd consists of frequent values, b and d, so we check its frequency. Assuming bd is infrequent, and $\text{supp}(bd) = 4$, we increase the score of $\mathbf{x}_2$:

$$\text{Score}_1(\mathbf{x}_2) = 0.2 + \frac{1}{\text{supp}(bd) \times |bd|} = 0.2 + \frac{1}{4 \times 2} = 0.325.$$

Note that at this point we stop increasing the score of both $\mathbf{x}_1$ and $\mathbf{x}_2$, because there are no more frequent sets. Therefore, in this scenario, we only need to check sets a, c, and bd, instead of all possible sets of length 1 to 3 contained in $\mathbf{x}_1$ and $\mathbf{x}_2$. □

3.1.3 Continuous score

Now that we have identified some outlier points based on their categorical characteristics, we consider the continuous attributes. Many existing methods detect outliers based on computing distances between each point and its nearest neighbors in the entire dataset. In addition to the fact that it is inefficient to consider the dataset in its entirety, especially in a distributed setting, it is very likely that, in doing so, we may miss outliers that are not globally obvious but easier to spot if we focus on a subset of our dataset.

Moreover, the notion of a nearest neighbor does not hold as well in high dimensional spaces as the distance between any two data points becomes almost the same (Beyer et al. 1999), which changes the notion of similarity between any two points. In our case of mixed attribute datasets, it is reasonable to believe that data points that share the same categorical value should also share similar continuous values. Therefore, we can restrict our search space by identifying and focusing on data points that share a categorical value, and then rank these points based on their similarity to each other.

One issue that arises is how to identify similarities between data points in high-dimensional datasets. The most prevalent similarity or distance metric is the Euclidean distance, or the L_2-norm. Even though the Euclidean distance is valuable in relatively small dimensionalities, its usefulness decreases as the dimensionality grows. Let us consider the four points below, taken from the KDDCup 1999 dataset (described in more detail in the Sect. 4.1.1): the first two points are normal and the second two points are outliers (we removed the columns that had identical values for all four points). Using Euclidean distance we find correctly that point 1 is closest to point 2 and vice versa. However, for both points 3 and 4 we find that each is closest in Euclidean distance to point 1, i.e. the two outlier points are more similar to a normal point than to each other.

1	0	0	.002	.002	0	0	0	0	1	0	0	.004	.004	1	0	1	0	0	0	0
2	8.2E−6	8.5E−6	.020	.02	0	0	0	0	.9	.2	.2	1	.878	.9	.01	.04	0	0	0	0
3	0	0	.002	.002	0	0	1	1	1	0	0	1	.004	0	.50	1	0	0	1	1
4	0	0	.002	.002	1	1	0	0	1	0	0	1	.004	0	.99	1	1	1	0	0

This is mainly because the Euclidean distance assigns equal importance to attributes with zero values as to attributes with non-zero values. In higher dimensionalities, "the presence of an attribute is typically a lot more important than the absence of an attribute" (Ertoz et al. 2003). The reason for this is that quite often the data points in high dimensionalities are sparse vectors, i.e. they only have a few non-zero attribute values (Ertoz et al. 2003). Cosine similarity is a commonly used similarity metric for clustering in very high-dimensional datasets, e.g. used for document clustering in Ertoz et al. (2003). The cosine similarity between two vectors is equal to the dot product of the two vectors divided by the individual vector norms. Assuming non-negative attribute values, the minimum cosine similarity is 0 (non-similar vectors) and the maximum is 1 (identical vectors). In our example with the four points above, the cosine function assigns highest similarity between points 1 and 2, and between points 3 and 4, so it correctly identifies similarity between normal points (points 1 and 2) and between outlier points (points 3 and 4).

In this paper, we used the cosine function to define similarities in the continuous space. Consider a data point $\mathbf{x}_i$ containing m_c categorical values and m_q continuous values. The categorical part of $\mathbf{x}_i$ is denoted by $\mathbf{x}_i^c$, and the continuous part of $\mathbf{x}_i$ by $\mathbf{x}_i^q$. Let a one of the categorical values of $\mathbf{x}_i^c$ that occurs with support $\text{supp}(a)$. We focus on a subset of the data, which contains the continuous vectors corresponding to the data points that share categorical value a: $\{\mathbf{x}_i^q : a \in \mathbf{x}_i^c,\ i = 1 \ldots n\}$, with a total of $\text{supp}(a)$ vectors. The mean vector of this set, μ_a, is below:

$$\mu_a = \frac{1}{\text{supp}(a)} \times \sum_{i=1 \,\wedge\, a \in \mathbf{x}_i^c}^{n} \mathbf{x}_i^q \tag{2}$$

Also, the cosine similarity between a point $\mathbf{x}_i^q$ and mean μ_a is given below:

$$\cos(\mathbf{x}_i^q, \mu_a) = \sum_{j=1}^{m_q} \left(\frac{x_{ij}^q}{\|\mathbf{x}_i^q\|} \times \frac{\mu_{aj}}{\|\mu_a\|} \right) \tag{3}$$

where $\|\mathbf{x}\|$ represents the L_2-norm of vector $\mathbf{x}$. Finally, we assign the following score to each point $\mathbf{x}_i$, for all categorical values a contained in $\mathbf{x}_i^c$:

$$\text{Score}_2(\mathbf{x}_i) = \frac{1}{|a \in \mathbf{x}_i^c|} \times \sum_{\forall a \in \mathbf{x}_i^c} \cos(\mathbf{x}_i^q, \mu_a) \tag{4}$$

which is the summation of all cosine similarities for all categorical values a divided by the total number of values in the categorical part of $\mathbf{x}_i$, $\mathbf{x}_i^c$. As minimum cosine similarity is 0 and maximum is 1, the data points with similarity close to 0 are more likely to be outliers.

Even though using the cosine similarity helps us better assess distances in a high-dimensional space, its use will not vastly improve our outlier detection accuracy in a large dataset with many different types of outliers. As we noted, at the beginning of this section, we focus on specific subsets of the continuous space so as to identify outliers in smaller settings. In the next section, we address the issue of having more than one outlier in a subset, and in Sect. 3.1.5 we outline which of the categorical values in $\mathbf{x}_i^c$ we use in order to compute Score_2 in Eq. 4.

3.1.4 Improving accuracy

In the previous sections, we outlined the basic concepts for detecting outliers in the categorical and the continuous space. In this section, we discuss ways to make further use of our knowledge from the categorical domain in order to improve our method of detecting outliers. Many methods such as Knorr et al. (2000), Angiulli and Pizzuti (2005) assume that outliers are the data points that are very irregular in comparison to the rest of the points, and that they can be globally detected. However, in many real datasets there are multiple outliers with different characteristics and their irregularity and detection depends on the rest of the outliers against which they are compared.

One problem that many outlier detection methods are prone to is the effect of *masking*. This is defined in an intuitive fashion in Acuna and Rodriguez (2004) as follows (for alternate definitions see Hawkins 1980, Barnett and Lewis 1978):

It is said that one outlier masks a second outlier, if the second outlier can be considered as an outlier only by itself, but not in the presence of the first outlier. Thus, after the deletion of the first outlier the second instance is emerged as an outlier.

In our case, the ideal scenario is to have only one type of outlier in every subset as discussed in 3.1.3. This way, we have more chances to detect outliers from each subset, since the outliers are different from the normal points in a particular subset. In reality however, we could also be confronted with the scenario in Fig. 1 where outlier point O_2 masks outlier point O_1. The solution that we propose is to further use the knowledge that we obtain from the categorical score to help alleviate the masking issue. Based on Eq. 1, data points with highly infrequent categorical values will have a very high Score_1. We can exclude points that have a high Score_1, from the calculation of our continuous score and from the computation of the mean used in our continuous score in Eqs. 2, 3, and 4.

Outlier points that already have a high categorical score (Score_1) can belong to one of the following categories in the continuous space: (a) highly irregular, (b) moderately irregular, or (c) normal. In Fig. 2, we illustrate three possible scenarios in the continuous space when trying to detect outliers with low categorical score in the presence of outliers with high categorical score. Figure 2 contains two hundred normal points (dots) and eighty one outliers (diamonds). Outlier points O_2 are assumed to score high in the categorical domain, thus we are not interested in these points in the continuous space. Instead, our goal is to detect outlier point O_1 in the presence of outlier points denoted by O_2.

Figure 2a illustrates case (a) where outliers O_2 score high in both categorical and continuous space; as a result they mask outlier O_1. This is the ideal case for our algorithm, because if we remove the O_2 points, outlier O_1 is easily detected due to its distance from the normal points. The mean of all data points in Fig. 2a before removing the O_2 points is roughly (0.4, 0.4), and after removal it is (0.2, 0.2).

Figure 2b has the case where outlier point O_1 is more extreme in the continuous space than already detected points O_2. In this case, removing the O_2 points will make the point we are trying to detect more pronounced and thus it also has a positive effect, though not as much as in case of Fig. 2a. The mean of all points in Fig. 2b is roughly (0.31, 0.35) before removing O_2, and after removal it is (0.2, 0.2).

Finally, in Fig. 2c, outliers O_2 are very close to normal points in the continuous space, and thus removing outliers O_2 will have little effect on the mean of the data points and thus the detection of outlier O_1. In conclusion, removing or excluding outliers with high categorical score from the mean of a subset in the continuous space will assist us in detecting masked outliers (as in the case of Fig. 2a) and will have little effect when outliers are not masked (as in the case of Fig. 2c).

The exclusion of outlier points can be done in the following manner: as we compute the frequencies and means for each categorical value in our dataset, we identify highly infrequent categorical values. Based on this information, we can update the means for

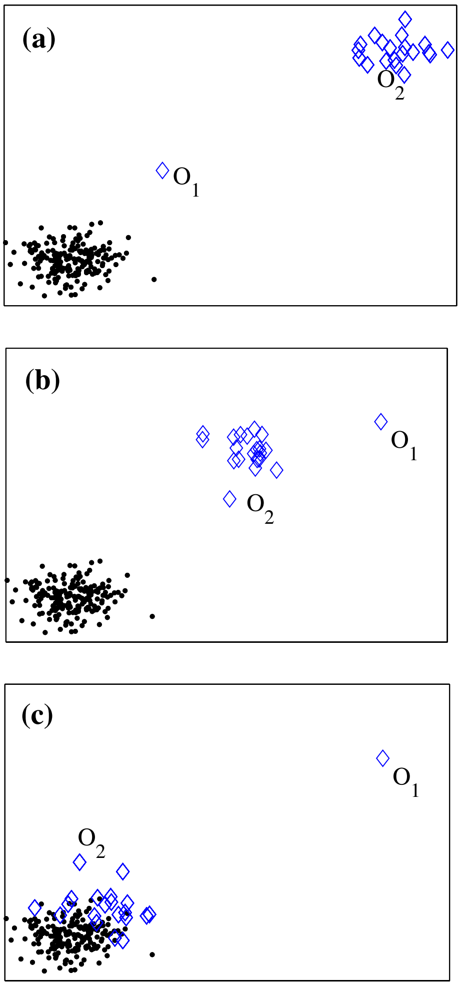

Fig. 2 Scenarios for outlier points and normal points in the continuous space (*black dots* denote normal points, *blue diamonds* denote outliers). Outlier points O_2 score high in the categorical domain. Outlier O_1 has low categorical score. In the continuous space, outliers O_2 are **a** highly irregular, **b** moderately irregular, or **c** normal, with respect to the rest of the points (normal points and outlier O_1)

the rest of the categorical values that co-occur with the highly infrequent values. The details on how we select the values to exclude from the continuous subsets are given in the following sections.

3.1.5 The centralized algorithm

In this section we provide the complete algorithm that uses the concepts and steps set forth in Sects. 3.1.1–3.1.4. The algorithm consists of two phases: the first phase constructs a model, while the second phase detects the outliers using the model. The model in the first phase includes the sets of categorical values, their frequencies, and the continuous mean vectors corresponding to categorical values. Figure 3 has the pseudocode for the first phase and Fig. 4 for the second phase. We describe the steps for our algorithm in more detail below:

```
Input  : Dataset $\mathcal{D}$ ($n$ points, $m_q$ and $m_c$ attributes),
         minimum support $\sigma$, maximum itemset length $MAXLEN$
Output: Model: $G$ - Pruned Candidates and their supports;
         $A$ - Categorical values, their means and frequencies

1 foreach point $\mathbf{x}_i$ $i = 1 \ldots n$ do
2     foreach categorical value $a \in \mathbf{x}_i^c$ do
3         $A \leftarrow$ Increase support $supp(a)$, Update mean $\mu_a$ with $\mathbf{x}_i^q$;
4     end
5     foreach $len = 2 \ldots MAXLEN$ do
6         $G \leftarrow$ Update pruned categorical sets and their supports;
7     end
8 end
```

Fig. 3 First phase of our outlier detection approach ODMAD

```
Input  : Dataset $\mathcal{D}$ ($n$ points, $m_q$ and $m_c$ attributes), $MAXLEN$,
         $\sigma$, $G$, $A$, $window$, $\Delta score_{c,q}$, $low\_sup$, $upper\_sup$
Output: Outliers detected

1  foreach point $\mathbf{x}_i, i = 1 \ldots n$ do
2      $Score_{1,2}(\mathbf{x}_i) = categorical\_values\_in\_range = 0$;
3      foreach categorical value $a \in \mathbf{x}_i^c$ do
4          if $supp(a) < \sigma$ then
5              $Score_1(\mathbf{x}_i) \mathrel{+}= \frac{1}{supp(a)}$;
6          end
7          if $low\_sup < supp(a) \leq upper\_sup$ then
8              $Score_2(\mathbf{x}_i) \mathrel{+}= \cos(\mathbf{x}_i^q, \mu_a)$;
9              $categorical\_values\_in\_range += 1$;
10
11         else if $supp(a) \leq low\_sup$ then
12             $Score_2(\mathbf{x}_i) = 1$;
13         end
14     end
15     $Score_2(\mathbf{x}_i) = Score_2(\mathbf{x}_i)/categorical\_values\_in\_range$;
16     foreach pruned set $d \in G \ \wedge \ d \in \mathbf{x}_i^c$ do
17         $Score_1(\mathbf{x}_i) \mathrel{+}= \frac{1}{supp(d) \times |d|}$;
18     end
19     if $Score_1(\mathbf{x}_i) > \Delta score_c \times$ average $Score_1$ in window or
20         $Score_2(\mathbf{x}_i) > 0 \wedge \Delta score_q \times Score_2(\mathbf{x}_i) <$ average $Score_2$ in window then
21         $\mathbf{x}_i \leftarrow outlier$;
22     else
23         $\mathbf{x}_i \leftarrow normal$; Add non-zero scores to window;
24     end
25 end
```

Fig. 4 Second phase of our outlier detection approach ODMAD

First phase: we scan the data in order to gather the single categorical values, sets of length 2 up to MAXLEN, and their respective frequencies (see Sect. 3.1.1). We first list the pruned candidates, i.e. all infrequent sets that contain only frequent subsets, in order to compute Score$_1$ (Sect. 3.1.2). Meanwhile, we compute the mean of each of the continuous subsets corresponding to categorical values (Sect. 3.1.3). As we identify categorical values and sets, we also update the corresponding mean vectors as discussed in Sect. 3.1.4.

We use a user-entered frequency threshold, called *low_sup*, to indicate what values we consider 'highly infrequent'; then categorical values with support ≤ *low_sup* are 'marked' as 'highly infrequent'. As we described in 3.1.4, we exclude the points that contain these 'highly infrequent' values from the mean in Eq. 2 of all other categorical values they co-occur with. For example: let point $\mathbf{x}$ with categorical value a, $\text{supp}(a) \leq low\_sup$, and value b, $\text{supp}(b) > low\_sup$. To exclude point $\mathbf{x}$, we subtract the continuous vector of $\mathbf{x}$ from the sum of all points with b, as follows:

$$\mu_b = \frac{1}{\text{supp}(b) - 1} \times \sum_{i=1 \wedge\, b \in \mathbf{x}_i^c}^{n} (\mathbf{x}_i^q - \mathbf{x}^q)$$

Second phase: we have all the information necessary to detect outliers based on their categorical values, i.e. Score_1 from Eq. 1, and based on their continuous values, i.e. Score_2 from Eq. 4. First, we increase Score_1 for each and every infrequent categorical value. Then, for each categorical value, we compute Score_2 for each point using the updated mean we computed in the first phase. The continuous vectors we use for the score are those that correspond to categorical values with support in $(low\_sup, upper\_sup]$. If a point has a value with support less than low_sup, its Score_2 will be equal to 1, as it contains a highly infrequent categorical value. If a point has no values with support in $(low\_sup, upper\_sup]$ it will have a Score_2 of 0. By applying a lower bound to the frequency range for Score_2 we are able to exclude values with very infrequent categorical values, and by applying an upper bound we can limit the amount of data points to which we assign a score in the continuous domain. In Sect. 4.4.1 we experiment with different values for this range $(low\_sup, upper\_sup]$ to explore how it affects our accuracy.

Finally, as we scan and score the data points, we maintain a window of categorical and continuous scores. We also employ a delta value for the detection of abnormal scores: $\Delta score_c$ for the categorical scores and $\Delta score_q$ for the continuous scores. As we go over the points in the second phase, if a point has a score larger (smaller in the case of Score_2) than the average score of the previous window of points by the corresponding Δ value, it is flagged as an outlier. Otherwise, the point is normal, and its non-zero scores are added to the window we maintain.

3.1.6 Complexity of the centralized algorithm

ODMAD computes the sets of categorical values of length 1 up to MAXLEN. It also needs to calculate the mean for the continuous subsets for each categorical value. Finally it needs to scan all points and calculate their score according to their subsets. Therefore if each of the m_c categorical attributes has an average of v distinct values, the complexity upper bound (worst case running time) is:

$$\begin{aligned} T &\leq n \times \left(v \times m_c \times m_q + \sum_{j=1}^{\text{MAXLEN}} v^j \times \binom{m_c}{j} \right) \\ &= O\left(n \times \left(v \times m_c \times m_q + (v \times m_c)^{m_c}\right)\right) \qquad (5) \end{aligned}$$

Therefore the execution time scales linearly with the number of data points, n, and with the number of continuous attributes, m_q, but scales exponentially with the number of categorical attributes, m_c. The memory requirements for ODMAD are the frequent itemset lattice, the pruned candidates (which is the negative border of the frequent sets on the lattice), and the sum vector of length m_q of the continuous vectors corresponding to each categorical value. Therefore, the memory load is $O\left(v \times m_c \times \left(m_q + 2^{m_c}\right)\right)$.

In comparison, the time complexity of the algorithm in Otey et al. (2006) is $O\left(n \times {m_q}^2 \times (v \times m_c)^{m_c}\right)$, while the memory requirements of their algorithm are $O\left(v \times m_c \times {m_q}^2 \times 2^{m_c}\right)$. As can be seen in Sect. 4 ODMAD is significantly faster than the approach by Otey et al. (2006), which we attribute to the fact that the latter maintains and checks a covariance matrix for each frequent itemset found in the dataset.

3.2 Distributed algorithm

As we discussed in Sect. 3.1.5, the algorithm consists of two phases: one that constructs the local model and one that detects the outliers using the model. In the distributed version of our algorithm, we assume that each node owns a unique subset of points (rows) of the entire dataset (horizontally distributed). The main concept is that each node computes its own local model using only its own dataset. It then exchanges its model with the other nodes and a global model is computed. Finally, this global model is broadcast to all nodes and used by each node to goes over its own dataset and detect outliers.

First, each node r enumerates all categorical sets of length 1 up to MAXLEN and their frequencies. Each node r also adds up the continuous vectors of all data points corresponding to all categorical sets. The quantities computed at local node r are as follows:

Support of set d: $\text{supp}^r(d)$

Sum of continuous vectors corresponding to set d: $\mathbf{s}_d^r = \sum_{d \in \mathbf{x}^{r,c}} \mathbf{x}^{r,q}$

where $\mathbf{x}^{r,q}$, $\mathbf{x}^{r,c}$ denote the continuous and categorical part of point $\mathbf{x}$ that resides in node r.

After all nodes have finished computing the above values, the sites engage in one round of communication to compute the global sets, frequencies, and means. Then we have the following global quantities denoted by G:

Global support of set d: $\text{supp}^G(d) = \sum_{r=1}^{R} \text{supp}^r(d)$

Global mean for set d: $\mu^G(d) = \frac{1}{\text{supp}^G(d)} \times \sum_{r=1}^{R} \mathbf{s}_d^r$

where R is the total number of nodes in our network.

From the global sets, each node obtains the pruned candidate sets (see 3.1.2) by deleting the sets that are frequent, and the sets that are infrequent and have infrequent subsets. After each node has computed and shared all necessary values, each node independently goes over its local dataset and selects the outliers as shown in Fig. 4 (second phase). This way, there is a need for only one round of communication between each node, in order to create the global model (sets, frequencies and means).

In the following, we discuss how we update the mean vectors to exclude highly infrequent categorical values (as in 3.1.4) in the distributed algorithm. As we calculate the categorical values and their frequencies (see Sect. 3.1.1), we also compute the frequency of each and every set of categorical values. We also compute the mean vector for each set of categorical values. Then, at the end of the first phase, we update the mean for each set of values where one of the values in the set has support less than the threshold low_sup.

In the following example, we show how we exclude the continuous information corresponding to a highly infrequent categorical value.

Example Assume two categorical values in our dataset, a and b. Let's assume that value a is highly infrequent: supp(a) $\leq low\_sup$, supp(b) $> low\_sup$, and a and b co-occur. As value a is highly infrequent, we wish to have the mean of all points containing value b and not a. The support of value b when value a is absent is denoted by supp($\overline{a}b$) (see Fig. 5a). This is equal to the support of b minus the support of the intersection of a and b:

$$\text{supp}(\overline{a}b) = \text{supp}(b) - \text{supp}(ab).$$

We store the summation of the continuous vectors corresponding to set ab. Then the updated mean for all values b will be the sum of the continuous vectors corresponding to b minus the sum of continuous vectors corresponding to ab, or:

$$\mu_{\overline{a}b} = \frac{1}{\text{supp}(\overline{a}b)} \times \left\{ \sum_{i=1 \wedge\, b \in \mathbf{x}_i}^{n} \mathbf{x}_i^q - \sum_{j=1 \wedge\, ab \in \mathbf{x}_j}^{n} \mathbf{x}_j^q \right\}$$

where: supp(a) $\leq low\_sup <$ supp(b) and supp($\overline{a}b$) $\neq 0$. Essentially, we subtract the continuous sum vector of the points containing ab from the sum vector of points that contain b. To achieve this, we store the support of b, supp(b), the support of ab, supp(ab), as well as the summation of their corresponding continuous subsets.

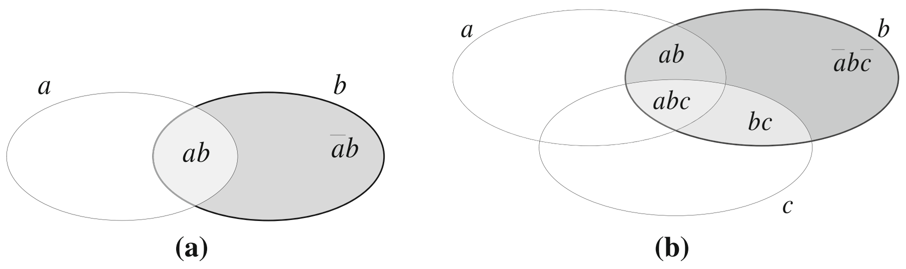

Fig. 5 Categorical value b co-occurs in our dataset with highly infrequent categorical **a** value a only; **b** values a and c

Figure 5b shows the case where categorical value b co-exists with two highly infrequent categorical values a and c, so we have the following:

$$\text{supp}(\overline{a}b\overline{c}) = \text{supp}(b) - \text{supp}(ab) - \text{supp}(bc) + \text{supp}(abc).$$

Similarly, when categorical value b co-exists with highly infrequent values a, c, d:

$$\begin{aligned}\text{supp}(\overline{a}b\overline{cd}) = {} & \text{supp}(b) - \text{supp}(ab) - \text{supp}(bc) - \text{supp}(bd) \\ & + \text{supp}(abc) + \text{supp}(abd) + \text{supp}(bcd) - \text{supp}(abcd).\end{aligned} \qquad \square$$

This concept generalizes in this way to the case where multiple categorical values are to be excluded. This is known as the *Inclusion/Exclusion Principle* (Knuth 1968). The update of the mean (Sect. 3.1.4) can be accomplished in a similar manner as in the example given above.

4 Experiments and results

4.1 Experimental setup

We implemented ODMAD and the approach in (Otey et al. 2006) using C++. We compare our method with the one by (Otey et al. 2006) as it is the only existing outlier detection approach for mixed attribute datasets that scales well with number of data points designed for distributed datasets. We ran our experiments on a workstation with a Pentium 4 1.99 GHz processor and 1 GB of RAM. For the distributed experiments, we implemented our approach using Boost.MPI[1] and Open MPI. We ran our experiments on a 16-node cluster, where each of the nodes had dual Opteron processors, 3GB of RAM, and 73GB disks. For all experiments we set MAXLEN to 4 and $window$ of 40 points, unless otherwise noted.

4.1.1 Datasets

KDDCup 1999 Dataset. We used the KDDCup 1999 intrusion detection dataset (Hettich and Bay 1999) which contains records that represent connections to a military computer network and multiple intrusions and attacks by unauthorized users. The raw binary TCP data were processed into features such as connection duration, protocol type, number of failed logins, etc. We obtained this dataset from the UCI KDD archive (Blake and Merz 1998). There are three available datasets: a training set, a test set, and a set with 10% of the training set. The KDD training set contains 4,898,430 data points and a dataset with 10% training data points. The testing data set is smaller and it contains several new intrusions not present in the training set. All the KDDCup 1999 sets contain 33 continuous attributes and 8 categorical attributes.

Due to the large number of attacks in these datasets, we preprocessed the datasets such that attack points are around 2% of the dataset. To construct our datasets we

[1] http://www.osl.iu.edu/~dgregor/boost.mpi.

picked a smaller number of outlier points entirely at random. Network traffic packets tend to occur in bursts for certain intrusions. While we preserved the proportions of the various attacks in the data, we selected our outlier points at random without necessarily preserving the length of the bursts. We followed the same concept as in Otey et al. (2006), and detected bursts of packets in the data set. Our processed dataset based on the entire training set contains 983,550 instances with 10,769 attack instances (22 distinct attack types); our 10% training dataset contains similar proportion of instances and attacks. The resulting testing set contains 61,924 instances and 1,331 attack instances (16 distinct attack types).

Artificially generated sets. Since there is no dataset generator publicly available to generate mixed attribute data, we created our own artificial datasets in order to experiment with various numbers of data points and dimensions. In these datasets we varied the number of points, n, the number of categorical attributes, m_c, and the number of continuous dimensions, m_q. Each dataset has a multimodal distribution where each mode corresponds to one set of categorical values of length m_c. Outliers contain a larger number of infrequent categorical values than normal points, and some of their values are more infrequent. Associated with each mode is a cluster of m_q dimensions. Specifically, we generate a cluster of random Gaussian data for the normal points and then create another smaller cluster for the outliers, while we apply random transformations to make the clusters different. Finally, we randomly shuffle the points to create the final dataset.

4.1.2 Evaluation

We evaluate both algorithms, ODMAD and Otey's, based on two measures:

- *Outlier Detection Accuracy rate*, which is the number of outliers correctly identified by each approach as outliers, and
- *False Positive rate*, reflecting the number of normal points erroneously identified as outliers.

We also compare the running time performance of the two algorithms using the same datasets.

4.2 Results

4.2.1 KDDCup 1999 dataset

Regarding the outlier detection accuracy or detection rate, in the KDDCup dataset, if we detect one point in a burst of packets as an outlier we mark all points in the burst as outliers and the burst as detected, similar to (Otey et al. 2006). In Table 2, we provide the detection rate achieved by ODMAD versus the approach in (Otey et al. 2006) using the KDDCup 1999 training and testing datasets (better rates are in bold). In Table 3 we show the execution time in seconds for the two approaches.

We experimented with several values for the parameters in Otey's approach, and in Table 2 we present best results (we used: $\delta = 35$; $\sigma = 50\%$ for the 10% set and testing

Table 2 Detection rate of ODMAD vs. Otey's on KDDCup 1999 (10%/Entire training and test sets; better rates are in bold)

Attack type	Detection rate (10% training)		Detection rate (entire training)	
	ODMAD	Otey's	ODMAD	Otey's
(a) *KDDCup 1999 10% training and entire training set*				
Back	**50**	**50**	75	**100**
Buffer overflow	**91**	36	**91**	**91**
FTP write	**75**	**75**	**100**	**100**
Guess password	**100**	**100**	**100**	**100**
Imap	**100**	**100**	**50**	**50**
IP sweep	**60**	30	**92**	76
Land	83	**100**	**100**	62
Load module	**100**	40	**100**	80
Multihop	**100**	75	**75**	**75**
Neptune	**100**	67	**100**	90
Nmap	**100**	**100**	**88**	63
Perl	**100**	67	**100**	67
Phf	0	**75**	**0**	**0**
Pod	**75**	50	**87**	53
Port Sweep	**100**	67	**100**	64
Root kit	**60**	40	**80**	40
Satan	**100**	67	**100**	50
Smurf	**57**	43	**88**	63
Spy	**100**	**100**	**100**	**100**
Teardrop	**100**	44	**100**	11
Warez client	**21**	4	9	**36**
Warez master	**100**	33	67	**100**

Attack type	ODMAD	Otey's approach
(b) *KDDCup 1999-test set*		
Apache 2	**100**	0
Buffer overflow	**63**	26
Guess password	**46**	7
IP sweep	**38**	33
Multihop	**86**	57
Named	**100**	58
Phf	**0**	**0**
Pod	**100**	42
Port sweep	**72**	38
Saint	**100**	42
Sendmail	**100**	56
Smurf	0	**11**
Snmpgetattack	1	**26**
Udpstorm	**100**	50
Xlock	**89**	67
Xsnoop	**100**	**100**

Table 3 Execution time (s) for ODMAD versus Otey's on the KDDCup 1999 datasets

Dataset	ODMAD	Otey's approach
10% Training set	3.7	617.4
Entire training set	38.0	6,014.9
Test set	2.1	331.2

set, and $\sigma = 10\%$ for the entire training set; $\Delta Score = 2$ for training and 1.5 for testing). For ODMAD we used: $upper\_sup = 10\%$ (40% test set); $low\_sup = 2\%$ (1% test set); $\Delta Score_c = 10$, $\Delta Score_q = 1.27$ (10% set); $\Delta Score_c = 7$, $\Delta Score_q = 1.18$ (entire training set); $\Delta Score_c = 5$, $\Delta Score_q = 1.2$ (test set).

From Table 2, ODMAD has equal or better detection rate than Otey's approach for all but two of the attacks on the 10% training set, and all but three of the attacks for the entire training set. Moreover, the detection rates in Table 2 for the 10% dataset were achieved with a false positive rate of 4.15% for ODMAD and 6.99% for Otey's, while the detection rates for the entire training set were achieved with a false positive rate of 7.09% for ODMAD, and 13.32% for Otey's.

For the test set we also perform better than Otey's approach for all but two attacks, while the false positive rate is 3.24% (ODMAD) versus 5.86% (Otey's). Overall, ODMAD achieved 81% average detection accuracy while Otey's had 62% (10% Training); 82 versus 67% (Entire Training); 68 versus 38% (Test set). Regarding the test set, it is noteworthy that it contains attacks not present in the training set. Our method performs very well for this set which was a challenge for KDDCup 1999 participants.

As can be seen in Table 2, ODMAD detects all attacks except one for the training sets (Phf), and two attacks for the testing set (Phf and Smurf). After inspecting the characteristics of these attacks, as well as Snmpgetattack (test set), we observed that these attacks have either high Score_2 (see Fig. 4, line 19), or their infrequent categorical values are below low_sup (see Fig. 4, lines 10–12), so they are not detected in the continuous space. On the other hand, their categorical score, Score_1, is not high enough for them to be detected as outliers, as there are other attacks with more irregular categorical values. For example, Snmpgetattack has no infrequent categorical values at $\sigma = 10\%$, and all of its points except one have very high Score_2.

In this dataset, the detection rate is affected by factors such as the type of attack, i.e. single connection attacks (e.g. Phf) versus attacks with multiple connections or bursts (see (Lazarevic et al. 2003) for an in-depth discussion on this issue). ODMAD does not use any special method, such as signature detection, to detect specific network intrusions, although these types of ideas such as in Lazarevic et al. (2003) could be used in addition to ODMAD to increase detection accuracy.

Another significant advantage of ODMAD versus Otey's approach with respect to running time is apparent in Table 3. For instance, ODMAD took 38 s to detect the outliers in the entire KDDCup 1999 training set, while Otey's approach needed 100 min to accomplish the same task. We attribute this to the fact that Otey's method needs to create and check a covariance matrix for each and every possible set of categorical attribute values, while our method looks at specific categorical values and the means of their continuous counterparts.

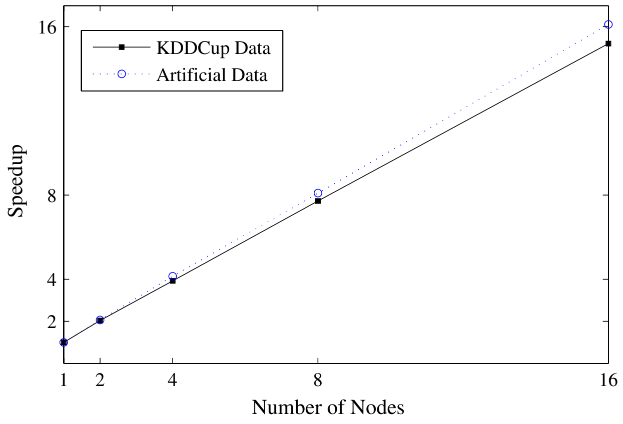

Fig. 6 Speedup of ODMAD as the number of nodes increases from 2 to 16 for the entire KDDCup 1999 dataset and artificial dataset (1 million normal points and 10 thousand outliers)

4.2.2 Artificial dataset

We ran experiments with an artificially generated dataset with 1 million normal points and 10,000 outliers, 9 categorical attributes and 20 continuous attributes. The parameters we used for ODMAD were: $low\_sup = 5\%$, $upper\_sup = 50\%$, $\Delta Score_c = 2.3$, $\Delta Score_q = 1.3$. For Otey's algorithm, we used $\sigma = 10\%$; $\delta = 35$; $window = 50$; $\Delta Score = 1$. ODMAD achieved a detection rate of 77.11% with a 1.69% false positive rate, while Otey's had a detection rate of 67.61% with 34.24% false positive rate. In closer inspection, normal points in this dataset also contain infrequent categorical values therefore Otey's score is high for these points as well as for outliers, while our score is able to better distinguish outliers due to the support of their values. Finally, ODMAD took 55 s to detect outliers in the artificial dataset, while Otey's took *81 min* for the same task.

4.3 Distributed experiments

In order to evaluate our distributed approach we performed experiments for the speedup of ODMAD as we increase the number of nodes in our distributed setting. For these experiments we used the entire KDD training set as well as an artificial dataset of 1 million normal data points and 10,000 outlier data points. Each dataset was evenly split amongst the available nodes. Figure 6 shows the speedup obtained when running our distributed approach on two, four, eight and sixteen nodes. The 'ideal' speedup is linear, i.e. when running an algorithm with linear speedup, doubling the number of nodes doubles the speed. As shown in Fig. 6, the speedup of ODMAD is close to linear. Since ODMAD relies on an exchange of sets of categorical values and the continuous vectors corresponding to these sets (sums), the communication overhead is minimal, and thus the speedup is very close to linear. As more nodes are added, the communication overhead increases, which is the reason the speedup is not exactly linear.

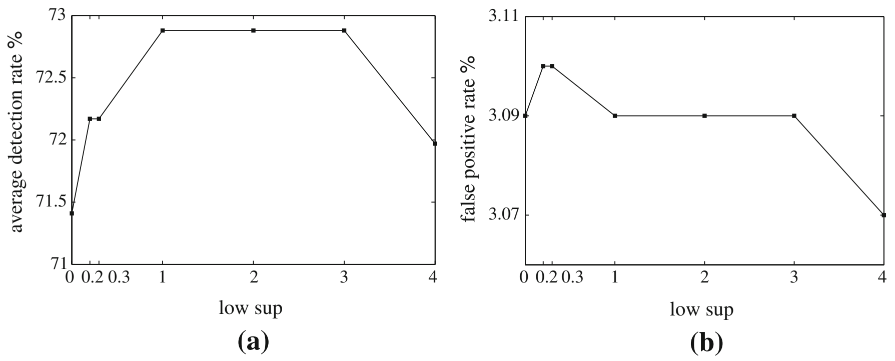

Fig. 7 Effect to **a** average detection accuracy and **b** false positive rate on the KDDCup 1999 10% dataset varying the lower threshold low_sup ($upper\_sup = \sigma = 10\%$; $\Delta score_c = 9$, $\Delta score_q = 1.6$)

4.4 Additional experiments

In this section, we present experimental results to explore how the performance of ODMAD varies with respect to the parameters of ODMAD. Specifically, we investigate how different values of the parameters low_sup and $upper\_sup$ affect ODMAD; how ODMAD responds to different number of attributes (categorical and continuous); and how calculating all subsets of a data point compares with counting only pruned candidates (see Sect. 3.1.2).

Regarding the $\Delta Score$ thresholds, we would like to note that the detection and the false positive rates decrease as $\Delta Score$ increases. In order for a point to be an outlier, its score must be larger (or smaller in the continuous case) than the average score in the previous window by this $\Delta Score$ value. Hence, $\Delta Score$ reflects the magnitude of difference between scores of different points. The larger $\Delta Score$ is, the higher the score difference needs to be for a point to be an outlier. The opposite happens as $\Delta Score$ decreases, accuracy and false positive rate increase, as more points can be outliers.

For the remaining ODMAD parameters, we note that experiments in (Otey et al. 2006) showed that a good value for MAXLEN regarding outlier detection accuracy is 3 or 4, which is why we used 4 for all our experiments. Furthermore, we used a σ of 10% and $window$ of 40 points for all experiments with ODMAD, as used by (Otey et al. 2006) in their experiments.

4.4.1 Effect of low_sup and upper_sup

We experimented with varying the two thresholds, low_sup and $upper\_sup$, in order to see the effect on the outlier detection accuracy and false positive rate of our technique. The results for the KDDCup 1999 10% set are shown in Figs. 7 (effect of low_sup) and 8 (effect of $upper\_sup$).

In Fig. 7, we kept $upper\_sup$ = 10% (equal to the minimum support σ) and varied low_sup from 0 to 4%. As Fig. 7a shows, the average detection accuracy is lower for 0% (i.e. no points are excluded from any subset), then peaks for low_sup = 1–3%.

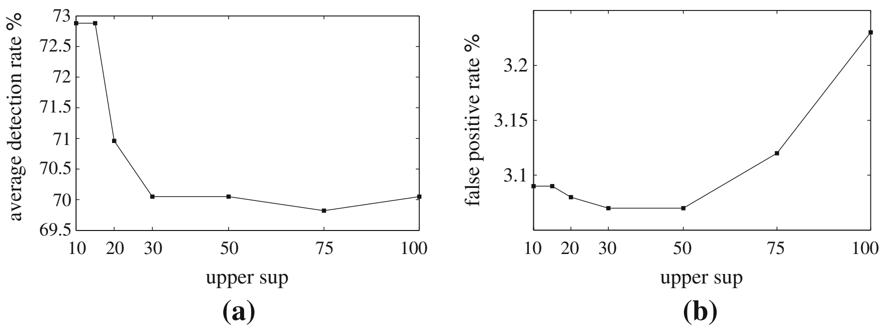

Fig. 8 Effect to **a** average detection accuracy and **b** false positive rate on the KDDCup 1999 10% dataset varying the upper threshold $upper\_sup$ ($low\_sup = 2\%$; $\sigma = 10\%$; $\Delta score_c = 9$, $\Delta score_q = 1.6$)

The average detection rate drops after 3% mainly because there are less categorical values with support between 4 and 10%, and thus fewer data points that contain these values. In Fig. 8, we kept low_sup at 2% and increased the $upper\_sup$ from 10 to 100%. As shown in Fig. 8a, the average detection accuracy drops as we include more categorical values and their subsets in the calculation of the continuous score.

The effect of varying low_sup and $upper\_sup$ on the false positive rate is shown in Figs. 7b and 8b, respectively. Increasing the low_sup threshold does not increase the false positive rate dramatically (Fig. 7b). Increasing $upper\_sup$ (Fig. 8b) results in a larger increase in the false positive rate. The overall results indicate that good values for $upper\_sup$ are close to the value for σ, and for low_sup close to 1–3%.

4.4.2 *Effect of number of attributes*

In Fig. 9 we evaluate how the execution time varies as the number of attributes, categorical and continuous, increases, using the KDDCup 1999 entire training set. For the first experiment in Fig. 9a we maintained the number of continuous attributes at 30, and varied the number of categorical attributes. For the second experiment in Fig. 9b, we varied the number of continuous attributes while the number of categorical attributes was 8. From these results, it can be seen that both times seem to have a polynomial dependence with the number of attributes. However, the execution time increases slower with respect to the increase in the number of continuous attributes than with respect to the increase in the number of categorical attributes.

4.4.3 *Using only pruned candidates for the categorical score*

Here we compare the categorical score we propose in Sect. 3.1.2 using only pruned candidates, i.e. infrequent sets that consist of frequent subsets, versus using all possible infrequent subsets of a data point. For this comparison, we used only the categorical score to detect outliers in the KDDCup 1999 10% dataset. In Table 4, we report the outlier detection accuracy as bursts of attacks detected for each attack type in this dataset. The accuracy reported in the second column of Table 4 is by assigning a score

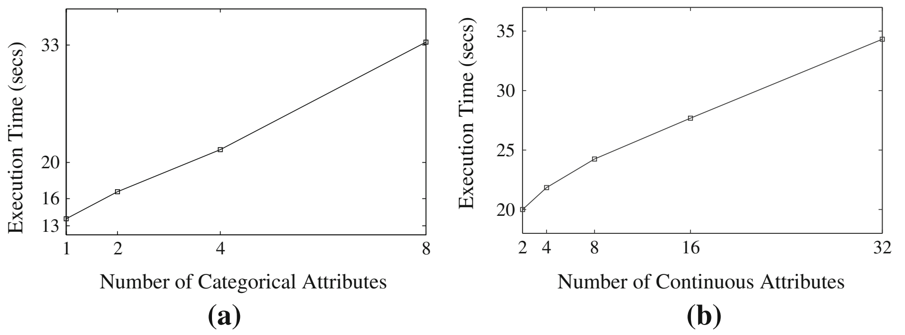

Fig. 9 Execution time in seconds as **a** the number of categorical attributes, and **b** the number of continuous attributes increases (Entire KDDCup 1999 set)

Table 4 Detection rate comparison using categorical score only: all infrequent subsets versus only pruned candidates using the KDDCup 1999 10% train dataset

Attack type	All infrequent subsets	Only pruned candidates
Back	50	50
Buffer overflow	82	82
FTP write	50	50
Guess password	100	100
Imap	100	100
IP sweep	55	65
Land	83	83
Load module	60	80
Multihop	75	75
Neptune	100	83
Nmap	83	83
Perl	100	100
Phf	0	0
Pod	75	75
Port sweep	100	89
Root kit	40	40
Satan	100	100
Smurf	43	71
Spy	100	100
Teardrop	0	0
Warez client	6	6
Warez master	67	67

to each data point based on all its infrequent subsets versus the third column which is using only the pruned candidates for Score$_1$ in Eq. 1. For this experiment, we used only the categorical attributes, and $\Delta Score_c = 8$.

As can be seen in Table 4, using only the pruned candidates for the computation of our categorical score in Eq. 1 has, in all but two cases, the same or higher detection rate compared to using all infrequent subsets of each data point. Furthermore, the false detection rate was 2.88% for the first case (all infrequent subsets) versus 2.99% for the second (only pruned candidates). We note that the accuracy rates are lower in Table 4 compared to Table 2 (e.g. Teardrop attack is not detected) because we are only using categorical attributes in this experiment.

Besides the comparison in Table 4, the "Appendix" contains a bound for the computational savings based on the second approach (only pruned candidates).

4.5 Discussion

From the previous paragraphs, one can see that ODMAD exhibits very good outlier detection rates and runtime performance. ODMAD explores the categorical space, and then the continuous space based on specific categorical values, in order to detect outliers quickly in large high-dimensional distributed data. Nevertheless, there are situations where the accuracy performance of ODMAD may start to degrade. In the case of data containing only continuous attributes, our method uses the cosine distance from a global mean, which might not work well for some data. We emphasize that our focus is to detect outliers in high-dimensional mixed-attribute data (e.g. network intrusion data), and not data with a single attribute type (e.g. astronomy data). Challenges with single-type attribute data faced by our method are also faced by the method in Otey et al. (2006). Extension of our ideas for data containing a single type of attribute, as well as data in other research areas, such as bioinformatics, is the topic of our future research.

5 Summary and conclusions

We have presented a fast distributed outlier detection algorithm for mixed attribute datasets that deals with sparse high-dimensional data. The algorithm called outlier detection for mixed attribute datasets (ODMAD) identifies outliers based on the categorical attributes first, and then focuses on subsets of data in the continuous space by utilizing information about these subsets from the categorical attribute space. We have experimented with the KDDCup1999 dataset, a benchmark outlier detection dataset, and artificially generated datasets in order to demonstrate the performance of ODMAD. We found that in most instances ODMAD exhibits higher outlier detection rates (accuracy) and lower false positive rates, compared to the existing state-of-the-art work in the literature (Otey et al. 2006). Furthermore, ODMAD relies on two dataset scans and its execution time is considerably faster than the competing work in Otey et al. (2006). Experiments showed that the distributed version of ODMAD exhibits close to linear speedup with the number of nodes. Future work includes extending ODMAD for vertically distributed datasets, as well as for other datasets and applications, such as sets with a single attribute type.

Acknowledgments This work was supported in part by NSF grants: 0341601, 0647018, 0717674, 0717680, 0647120, 0525429, 0806931, 0837332.

Appendix

In this section, we give a description of the computational savings in terms of number of sets we do not check for each point in the second phase of ODMAD (see Fig. 4) based on the reasoning in Sect. 3.1.2. Similar work was presented in Geerts et al. (2005), Calders et al. (2004) for a tight upper bound on number of candidates generated by frequent itemset mining methods and itemsets that can be derived based on smaller itemsets and their support.

In general, assume data point $\mathbf{x}_i$, where $i = 1 \ldots n$ (for the following analysis, we assume $\mathbf{x}_i$ to denote the categorical part of point $\mathbf{x}_i$). Point $\mathbf{x}_i$ has m_c attributes: $\mathbf{x}_i = [x_{i1}, \ldots, x_{il}, \ldots, x_{im_c}]$, where x_{il} denotes the l-th value of $\mathbf{x}_i$. As described in 3.1.2, if categorical value x_{il} is infrequent, all possible supersets of value x_{il} in point $\mathbf{x}_i$ are infrequent as well. The number of the supersets of value x_{il} contained in point $\mathbf{x}_i$ can easily be calculated since we know that $\mathbf{x}_i$ has m_c attributes (i.e. it is a vector with m_c values).

Let's assume that x_{il} is the only infrequent value in $\mathbf{x}_i$, and that all other values in $\mathbf{x}_i$ as well as their combinations are frequent. Therefore one value is fixed, i.e. x_{il}, and $m_c - 1$ values remain to form the supersets. Then, we have the following total number of supersets of x_{il} in $\mathbf{x}_i$ which are also infrequent:

$$\underbrace{\binom{m_c-1}{1}}_{\text{length } 1} + \underbrace{\binom{m_c-1}{2}}_{\text{length } 2} + \cdots \underbrace{\binom{m_c-1}{m_c-1}}_{\text{length } m_c} = m_c - 1 + \binom{m_c-1}{2} + \cdots + 1.$$

Now, if we assume 2 out of the m_c values of $\mathbf{x}_i$ are infrequent and the remaining $(m_c - 2)$ values are frequent, we have:

$$\underbrace{\left(2 \times \binom{m_c-2}{1} + \binom{2}{2}\right)}_{\text{length } 2} + \underbrace{\left(2 \times \binom{m_c-2}{2} + \binom{m_c-2}{1}\right)}_{\text{length } 2} + \cdots \underbrace{\binom{m_c-2}{m_c-2}}_{\text{length } m_c}$$

In general, for point $\mathbf{x}_i$ with m_c attributes, k infrequent categorical values, and any given length h, $1 < h \le m_c$, the total number of sets we do not check is equal to:

$$\sum_{j=1}^{h} \binom{m_c-k}{h-j} \times \binom{k}{j}$$

For example:

$$\text{Number of subsets of length } h = 2 : \binom{m_c-k}{1} \times k + \binom{k}{2},$$

$$\text{Number of subsets of length } h = 3 : \binom{m_c-k}{2} \times k + (m_c - k) \times \binom{k}{2} + \binom{k}{3}.$$

Therefore, if we maintain sets of length up to MAXLEN $\leq m_c$, the total number of sets we do not check, given k infrequent values in our data point and starting with length $h = 2$, is:

$$\sum_{h=2}^{\text{MAXLEN}} \sum_{j=1}^{h} \binom{m_c - k}{h - j} \times \binom{k}{j}$$

This means the following: Assume k infrequent values in our data and all of the subsequent supersets of the remaining $(m_c - k)$ values are frequent. The above is the lower bound on the infrequent sets we do not check for each point (based on 3.1.2). Below we include an example for this bound.

Example Assume point $\mathbf{x} = [a\ b\ c\ y\ p\ t\ w]$, with $m_c = 7$ categorical attributes. There are $k = 4$ infrequent values: a, b, c, and y. According to 3.1.2, we do not check any subsets of $\mathbf{x}$ that contain these k values as follows:

$$\text{Number of subsets length } 2 = \binom{m_c - k}{1} \times k + \binom{k}{2} = 4 \times (7 - 4) + \binom{4}{2} = 18.$$

Indeed, we get 6 sets by combining all k values with each other: ab, ac, ay, bc, by, cy, and 12 sets by combining $m_c - k = 3$ values with each of the k values: ap, at, aw, bp, bt, bw, cp, ct, cw, yp, yt, and yw.

Similarly, for length 3:

$$\binom{m_c - k}{2} \times k + (m_c - k) \times \binom{k}{2} + \binom{k}{3}$$
$$= 4 \times \binom{7 - 4}{2} + 3 \times \binom{4}{2} + \binom{4}{3} = 23.$$

Assuming MAXLEN is equal to 3, we do not check a total of $18 + 23 = 41$ sets for each point.

References

Acuna E, Rodriguez C (2004) A meta analysis study of outlier detection methods in classification. Technical paper, Department of Mathematics, University of Puerto Rico at Mayaguez. Available at http://academic.uprm.edu~eacuna/paperout.pdf

Aggarwal C, Yu P (2001) Outlier detection for high dimensional data. ACM SIGMOD Record 30(2):37–46

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the international conference on very large data bases, pp 487–499

Aha D, Bankert R (1994) Feature selection for case-based classification of cloud types: an empirical comparison. In: Proceedings of the 1994 AAAI workshop on case-based reasoning, pp 106–112

Angiulli F, Pizzuti C (2005) Outlier mining in large high-dimensional data sets. IEEE Transac Knowl Data Engin 17(2):203–215

Barnett V, Lewis T (1978) Outliers in statistical data. Wiley, NY

Bay S, Schwabacher M (2003) Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 29–38

Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is "nearest neighbor" meaningful? In: Proceedings of the 7th international conference on database theory, pp 217–235

Biba M, Esposito F, Ferilli S, Di Mauro N, Basile T (2007) Unsupervised discretization using kernel density estimation. In: Proceedings of the 20-th international conferece on artificial intelligence, pp 696–701

Blake C, Merz C (1998) UCI repository of machine learning databases. http://archive.ics.uci.edu. Accessed Sept 2008

Bolton R, Hand D (2002) Statistical fraud detection: a review. Stat Sci 17(3):235–255

Branch J, Szymanski B, Giannella C, Wolff R, Kargupta H (2006) In-network outlier detection in wireless sensor networks. In: Proceedings 26th international conference on distributed computing systems

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

Calders T, Rigotti C, Boulicaut J (2004) A survey on condensed representations for frequent sets. LNCS Constraint-Based Mining and Inductive Databases 3848:64–80

Catlett J (1991) Megainduction: machine learning on very large databases, PhD thesis, Basser Department of Computer Science, University of Sydney, Australia

Dean J, Ghemawat S (2004) MapReduce: simplified data processing on large clusters. In: USENIX symposium on operating systems design and implementation OSDI

Dokas P, Ertoz L, Kumar V, Lazarevic A, Srivastava J, Tan P (2002) Data mining for network intrusion detection. In: Proceedings NSF workshop on next generation data mining, pp 21–30

Ertoz L, Steinbach M, Kumar V (2003) Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: SIAM international conference on data mining, pp 47–58

Geerts F, Goethals B, Van den Bussche J (2005) Tight upper bounds on the number of candidate patterns. ACM Transac Database System (TODS) 30(2):333–363

Hawkins D (1980) Identification of outliers. Chapman and Hall, London

Hawkins S, He H, Williams G, Baxter R (2002) Outlier detection using replicator neural networks. In: Proceedings of the 4th international conference on data warehousing and knowledge discovery, pp 170–180

Hays C (2004) What Wal-Mart knows about customers habits. The New York Times, November 14

He Z, Xu X, Deng S, Calvanese D, De Giacomo G, Lenzerini M (2006) A fast greedy algorithm for outlier mining. In: Proceedings of 10th Pacific-Asia conference on knowledge and data discovery, pp 567–576

Hettich S, Bay S (1999) The UCI KDD archive. http://kdd.ics.uci.edu

Hodge V, Austin J (2004) A survey of outlier detection methodologies. Artif Intell Rev 22(2):85–126

Knorr E, Ng R (1998) Algorithms for mining distance-based outliers in large datasets. In: Proceedings of the 24th international conference on very large data bases, pp 392–403

Knorr E, Ng R, Tucakov V (2000) Distance-based outliers: algorithms and applications. Int J Very Large Data Bases VLDB 8(3):237–253

Knuth D (1968) The art of computer programming, vol 1. Addison-Wesley, Reading, MA

Koufakou A, Georgiopoulos M, Anagnostopoulos G (2008b) Detecting outliers in high-dimensional datasets with mixed attributes. In: International conference on data mining DMIN, pp 427–433

Koufakou A, Ortiz E, Georgiopoulos M, Anagnostopoulos G, Reynolds K (2007) A scalable and efficient outlier detection strategy for categorical data. In: IEEE international conference on tools with artificial intelligence ICTAI, pp 210–217

Koufakou A, Secretan J, Reeder J, Cardona K, Georgiopoulos M (2008a) Fast parallel outlier detection for categorical datasets using MapReduce. In: IEEE world congress on computational intelligence international joint conference on neural networks IJCNN, pp 3298–3304

Latecki L, Lazarevic A, Pokrajac D (2007) Outlier detection with kernel density functions. Lecture Notes in Computer Science 4571:61

Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J (2003) A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the 3rd SIAM international conference on data mining, p 25

Mehta S, Parthasarathy S, Yang H (2005) Toward unsupervised correlation preserving discretization. IEEE Transac Knowl Data Engin 17(9):1174–1185

Otey M, Ghoting A, Parthasarathy S (2006) Fast distributed outlier detection in mixed-attribute data sets. Data Mining Knowl Discov 12(2):203–228

Papadimitriou S, Kitagawa H, Gibbons P, Faloutsos C, (2003) LOCI: fast outlier detection using the local correlation integral. In: Proceedings 19th international conference on data engineering, pp 315–326

Penny K, Jolliffe I (2001) A comparison of multivariate outlier detection methods for clinical laboratory safety data. The Statistician 50(3):295–308

Preparata F, Shamos M (1985) Computational geometry: an introduction. Springer, Berlin

Roberts S, Tarassenko L (1994) A probabilistic resource allocating network for novelty detection. Neural Comput 6(2):270–284

Rousseeuw P (1985) Multivariate estimation with high breakdown point. Math Stat Appl 8:283–297

Rousseeuw P, Leroy A (1987) Robust regression and outlier detection. Wiley, NY

Tan P, Steinbach M, Kumar V (2005) Introduction to data mining. Pearson Addison Wesley, London

Tax D, Duin R (2004) Support vector data description. Mach Learn 54(1):45–66

Yu J, Qian W, Lu H, Zhou A (2006) Finding centric local outliers in categorical/numerical spaces. Knowl Inform Syst 9(3):309–338