Karlsruher Institut für Technologie

Karlsruhe Reports in Informatics 2017,1
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Hiding Outliers in HighDimensional Data Spaces

Georg Steinbuß, Klemens Böhm

2017

KIT – The Research University in the Helmholtz Association

Hiding Outliers in High-Dimensional Data Spaces

Georg Steinbuß
Karslruhe Institute of Technology
Germany
georg.steinbuss@kit.edu

Klemens Böhm
Karslruhe Institute of Technology
Germany
klemens.boehm@kit.edu

ABSTRACT

Detecting outliers in high-dimensional data is crucial in many domains. Due to the curse of dimensionality, one typically does not detect outliers in the full space, but in subspaces of it. More specifically, since the number of subspaces is huge, the detection takes place in only some subspaces. In consequence, one might miss *hidden outliers*, i.e., outliers only detectable in certain subspaces. In this paper, we take the opposite perspective, which is of practical relevance as well, and study how to hide outliers in high-dimensional data spaces. We formally prove characteristics of hidden outliers. We also propose an algorithm to place them in the data. It focuses on the regions close to existing data objects and is more efficient than an exhaustive approach. In experiments, we both evaluate our formal results and show the usefulness of our algorithm using different subspace selection schemes, outlier detection methods and data sets.

1. INTRODUCTION

Many applications in different domains, e.g., fraud detection, depend on the effective and efficient identification of outliers [3]. Due to the curse of dimensionality, outliers often occur in attribute subspaces. Such outliers are referred to as subspace outliers. In high-dimensional spaces, it is not feasible to inspect all subspaces for outliers, since their number grows exponentially with the dimensionality. Thus, most approaches only inspect a subset of the set of all subspaces. Depending on the subspaces inspected, the outlier detection method used and the distribution of the data, so-called *hidden outliers* may occur. A hidden outlier exhibits its outlier behaviour only in subspaces where no outlier detection takes place. Hence, the characteristic whether an outlier is hidden or not depends on the subspaces where one is looking for outliers. See Figure 1. The outlier in the figure is hidden when looking at each one-dimensional subspace in isolation. It can only be detected when looking at the two-dimensional subspace.

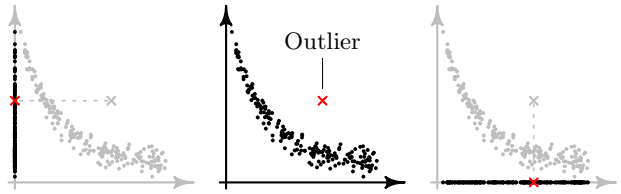


Figure 1: Example showing a hidden outlier

1.1 Motivation

In this article, we examine how to place hidden outliers in high-dimensional data spaces, and we quantify the risk of the data owner that such outliers can be placed in the data. We see three reasons why studying the issue is necessary, namely (1) increasing the reliability of critical infrastructures, (2) coping with attacks, and (3) systematic evaluation of outlier detection algorithms. We now elaborate on these points one by one.

1.1.1 Reliability of Critical Infrastructures

Think of data objects each representing a state of a critical infrastructure. Outliers are unusual system states which may represent any kind of fault or a state preceding a fault. Since faults of infrastructures that are critical may be catastrophic [13], any action preventing such faults pays off. However, data objects representing these states usually do not exist or are extremely rare. Hidden outliers represent combinations of values that remain undetected with existing models. If hidden outliers were detected, a domain expert could inspect them and assess how detrimental they are.

1.1.2 Attacking with hidden outliers

Research on classifier evasion [16] studies the behaviour of an adversary attempting to 'vanquish' a learner. Here, as Example 1 shows, the situation is analogous, including the motivation, i.e., studying the adverse behaviour in order to shield against it. While Example 1 is extreme for the sake of illustration, it is our running example due to its intuitiveness. The example also shows that hidden outliers pose a risk, and it is worthwhile to quantify this risk.

EXAMPLE 1. *Think of a criminal intending to commit credit-card fraud. He has inside information, i.e., he knows that the bank checks for fraud by means of an outlier-detection method on a high-dimensional representation of the credit-card transactions. More specifically, the bank uses*

a subspace-search method that is confined to subspaces consisting of few attributes only. The attacker then identifies regions of the data space where outliers are hidden from the detection method and designs fraudulent transactions whose representation falls into these regions.

1.1.3 Evaluation of subspace outlier detection

Being able to hide outliers is expected to help evaluate subspace outlier detection methods. Current evaluation schemes often either use an already existing minority class as outlying or downsample the data to one rare class [2]. However, these outliers are not necessarily subspace outliers, in contrast to hidden outliers generated with our approach. Placed hidden outliers are known to be outliers in certain subspaces, and one can quantify how well they are found. An evaluation scheme based on our approach would also allow differentiating between different kinds of hidden outliers and might be more systematic than one depending on the outliers which have been found so far. – The design and assessment of such evaluation schemes is a long-term research effort of us and is beyond the scope of this current article. Here, our concern is the effective and efficient placement of subspace outliers.

1.2 Challenges and Contribution

Up to our knowledge, the general question of how to place outliers in data sets has not been studied explicitly so far. Placing outliers such that they are hidden is even more unclear. Various challenges arise when designing such a placement, as follows.

Hidden outliers are inliers in certain subspaces. Hence, we cannot just use extreme values to obtain hidden outliers. Compared to Figure 1, the situation may also be more complex. For instance, one is not confined to just 'high dimensional versus low dimensional subspace'. Instead, a mix of the two is feasible as well. The variety of outlier definitions is another, orthogonal challenge. Some rely on statistical models, others on spatial proximity [12] or angles between objects [8]. We hypothesize that different outlier definitions lead to different regions where hidden outliers can be placed. When designing a general algorithm that hides outliers we cannot assume much on the outlier-detection method used. Another challenge relates to the relative size of the region where hidden outliers can be placed. In some scenarios, this region may be small, while it may be huge in others, e.g., close to the full data space. Placing many hidden outliers that are diverse requires methods that adapt to the size of this region. I.e., the placement should cover a broad range of positions if the region is huge. When the region is small in turn, the placement must be more fine-grained. A last challenge is that assessing the probability of success of attackers (the ones who hide the outliers) is not trivial. In contrast to Example 1, attackers may not have full access to the data. Hence, an attack is more likely to be successful if hiding is feasible without knowing much on the data. Any risk assessment should take into account the extent of knowledge which is necessary for the hiding.

In our work we start by deriving important characteristics of hidden outliers analytically, focusing on multivariate data following a normal distribution. A first result is that hidden outliers do exist in this setting. Second, correlation within subspaces can reduce the size of the region of hidden outliers. Third, having a higher number of dimensions has the same effect. Another contribution of ours is an algorithm that places hidden outliers. Its design is based on the hypothesis that hidden outliers tend to be close to real data objects. This is based on the property that hidden outliers must be inliers in some subspaces. Hence, our algorithm concentrates on placing hidden outliers in regions close to existing data objects, with adjustable tightness. This allows for a placement that concentrates on a small region, close to the data, or a rather large one. The algorithm does not rely on any assumption regarding the outlier detection method used, except for a non-restrictive one: Namely, the detection method must flag points as outliers or not. The output of any method we are aware of can be transformed without difficulty to have this characteristic (see e.g., [11]). Our algorithm also gives way to a rigid definition of the risk of an attacker being able to hide outliers. Finally, we have carried out various experiments. They confirm that some of our theoretical findings also hold in the absence of the underlying model assumptions, e.g., for other outlier detection methods and data sets. They also demonstrate that our algorithm is much better in hiding than a baseline. In particular, this holds for high-dimensional data sets. This paper is structured as follows:

1. Definition of Hidden Outlier. *Section 4.2*
2. Analytical derivations of characteristics of hidden outliers. *Section 4.3*
3. Algorithm to place hidden outliers. *Section 4.4*
4. Evaluation of concept and algorithm. *Section 5*

All our code and used data sets are publicly available.[1]

2. RELATED WORK

We are not aware of any comprehensive study of hidden outliers. [20] however describes the notion of *masked* outliers. 'masked' means that irrelevant attributes within a data set can hide the outlier behaviour to some extent. Our work is of course related to the various methods for outlier detection and subspace search. Some schemes exist solely for subspace search [14] [4], some with integrated outlier detection [10] [15] and numerous methods merely for outlier detection [1], [12], [8], [6]. All outlier detection methods compute whether existing data objects are outliers or not. This is different from our approach. We study how to place outliers in data sets.

To illustrate classifier evasion mentioned before explicitly, think of a spam filter. The idea now is that a spammer wants to send emails that are *as close as possible* to spam, but are classified as regular. However, existing approaches to find such positions [17], [19] rely on at least one instance of spam email, which we do not rely on in outlier detection. Secondly, we are not aware of any approach considering the effects of using subspaces.

Protecting privacy is another area in data analysis that is related. This is because some approaches for privacy protection add objects to the data. For example, [5] proposes an algorithm to add dummy objects to position data of individuals, in order to have better privacy. Clearly, the objective

[1]Our code and data: http://ipd.kit.edu/mitarbeiter/steinbuss/Experiments_HideOutlier.zip

is different: Privacy-protection approaches attempt to add data that behaves like the original data. Hence, the true data is hidden, while relevant information is still available. We in turn hide data objects which contradict the general structure of the data. Another difference is that such privacy approaches so far are global, i.e., not based upon subspaces.

Another related term is *robust statistics*. It deals with the fact that assumptions in statistics often are only approximations of reality. Violations of these assumptions are often interpreted as outliers. Hence, robust approaches take such possible violations into account to stabilize statistical models. [18] for instance proposes a modification of a subclass of Gauss-Markov models such that it is free from outlier hiding effects. Without these modifications, outliers might affect the model itself in a way that they are not detectable, i.e., are hidden. However, such approaches are not based upon subspaces, do not compute outlier regions or address the problem of hiding outliers in the data.

3. NOTATION

Let DB be a database containing n objects, each described by a d-dimensional real-valued data vector $\vec{y} = (y^{(1)}, \ldots, y^{(d)})^T$. The set $\mathcal{A} = \{1, \ldots, d\}$ denotes the full attribute space. W.l.o.g., we assume that each attribute lies within $[l, u]$ where $l,\ u \in \mathbb{R}$. An attribute subset $\mathcal{S} = \{a_1, \ldots, a_{\tilde{d}}\} \subseteq \mathcal{A}$ is called a $\tilde{d}$-dimensional subspace projection ($1 \leq \tilde{d} \leq d$). A set $Collection = \{\mathcal{S}_1, \ldots, \mathcal{S}_t\} \subseteq P(\mathcal{A})$ is a collection of t subspace projections ($1 \leq t \leq 2^d - 1$). The set $Full\mathcal{R} = \{\vec{y} \in [l, u]^d\}$ is the entire data space. When not stated different explicitly, for any region $\mathcal{R}$, it holds that $\mathcal{R} \subseteq Full\mathcal{R}$. Further, we assume that there exists a function $out^{\mathcal{S}}(\cdot)$ of the form:

$$out^{\mathcal{S}}(\vec{y}) := \begin{cases} 1 & \text{if } \vec{y} \text{ is outlier in } \mathcal{S}, \\ 0 & \text{if } \vec{y} \text{ is inlier in } \mathcal{S}. \end{cases} \quad (1)$$

The function $out^{\mathcal{S}}(\cdot)$ is a generic outlier definition. Different outlier detection methods which typically incorporate different definitions of this generic function are in use. Many such methods output a score instead of a binary value. However, we assume that these scores are transformed to a binary signal, e.g., by applying a threshold.

4. THE REGION OF HIDDEN OUTLIERS

In this section we formalize the notion of hidden outlier and derive important characteristics. Section 4.1 features some assumptions behind our formal results. In Section 4.2, we define hidden outliers and other relevant concepts. In Section 4.3, we derive our formal results. Section 4.4 features an algorithm to place hidden outliers. This algorithm also allows to define the *risk* of hidden outliers.

4.1 Assumptions

We assume that DB follows a multivariate normal (MVN) distribution with zero mean. Of course, Gaussian distributed data points have attribute limits $-\infty$ and $+\infty$. However, we assume that l and u are so large that even outliers will most likely be contained in the range spanned by l and u. With MVN data, the Mahalanobis distance [12] yields the likeliness of a data point. We assume data objects to be outliers if they are very unlikely according to that distance. We refer to the Mahalanobis distance of $\vec{y}$ in subspace $\mathcal{S}$ as $MDist^{\mathcal{S}}(\vec{y})$. $Quantile(\alpha,\ df)$ is the α quantile of a χ^2 distribution with df degrees of freedom. According to [12], we can instantiate our outlier definition as follows:

$$out^{\mathcal{S}}(\vec{y}) := \begin{cases} 1 & \text{if } \left[MDist^{\mathcal{S}}(\vec{y})\right]^2 > Quantile(0.975,\ |\mathcal{S}|), \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

4.2 Definition

We define the notion of hidden outlier as follows:

DEFINITION 1. *Let two disjunct sets of subspace projections $Collection_{outlier}$ and $Collection_{inlier}$ be given. $\vec{o} \in [l, u]^d$ is a* ***hidden outlier with respect to subspace collections $Collection_{inlier}$ and $Collection_{outlier}$*** *if $out^{\mathcal{S}}(\vec{o}) = 0\ \forall\ \mathcal{S} \in Collection_{inlier}$ and $\exists\ \mathcal{S} \in Collection_{outlier}$: $out^{\mathcal{S}}(\vec{o}) = 1$.*

The number of subspaces not in $Collection_{inlier}$ is usually rather high. Testing a subspace for outliers contained in it is expensive computationally. Thus, we focus on the case that the hidden outliers are outlier in at least one subspace of $Collection_{outlier}$ instead of any subspace not in $Collection_{inlier}$. $Collection_{inlier}$ and $Collection_{outlier}$ must always be disjunct. This is because there cannot be any point being an inlier and outlier in the same subspace. However, there can be overlapping attributes in subspaces of both sets. If there is no attribute within subspaces of both sets, the task of placing hidden outliers is rather simple. One creates an outlier for one of the subspace in $Collection_{outlier}$ and sets the values for the remaining attributes in $\mathcal{A}$ to the ones of any existing inlier object. Thus, in this article we focus on scenarios with such overlap. Based on this definition, we now formulate a hypothesis.

HYPOTHESIS 1. *Since hidden outliers are inliers for all subspaces in $Collection_{inlier}$, hidden outliers must be spatially close to the points in DB.*

We will return to this hypothesis when designing our algorithm (Section 4.4) and in the experiments (Section 5.4.5).

A core issue in this study is to identify the positions/region with the following characteristic: If we place a data point there, it is a hidden outlier. We now derive this region and present some characteristics of hidden outliers. To this end, we do not rely on any further assumption regarding $out^{\mathcal{S}}(\cdot)$.

DEFINITION 2. *The* ***region of inliers*** *is defined as*

$$In\mathcal{R}(Collection) := \{\vec{o} \in [l, u]^d \mid out^{\mathcal{S}}(\vec{o}) = 0\ \forall\ \mathcal{S} \in Collection\}$$

The ***region of outliers*** $Out\mathcal{R}(Collection)$ *is its complement.*

Definition 2 formalizes the notion of the region fulfilling one property of hidden outliers, i.e., regions with positions that are inlier or outlier for each subspace in *Collection*. This notion is a prerequisite before defining the region of hidden outliers. See Figures 2a and 2b for examples using the Mahalanobis distance. We discuss characteristics of it in Section 4.3.

LEMMA 1. *$In\mathcal{R}(Collection)$ is the intersection of $In\mathcal{R}(\{\mathcal{S}\})$ $\forall\ \mathcal{S} \in Collection$.*

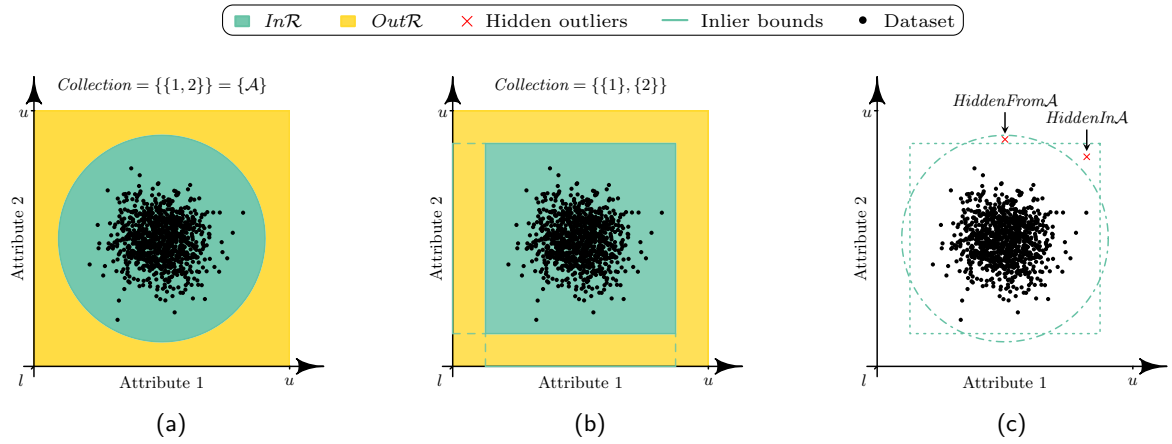


Figure 2: Example for *In$\mathcal{R}$(Collection)*, *Out$\mathcal{R}$(Collection)* and hidden outliers in *HiddenIn$\mathcal{A}$* and *HiddenFrom$\mathcal{A}$*.

Lemma 1 states that we can derive $In\mathcal{R}(Collection)$ using only intersections of $In\mathcal{R}(\{\mathcal{S}\})$, i.e., the inlier region in a single subspace. Detecting outliers in one subspace is well defined and has been explored intensively.

Definition 3. *Given two sets of subspace projections $Collection_{outlier}$ and $Collection_{inlier}$, the* ***region of hidden outliers*** *$Hidden(Collection_{outlier}, Collection_{inlier})$ is the intersection of $In\mathcal{R}(Collection_{inlier})$ and $Out\mathcal{R}(Collection_{outlier})$.*

Thus, every point in *Hidden* is a hidden outlier. We see that, up to intersections, unions and complements, *Hidden* solely depends upon outlier detection in a single subspace. However, $In\mathcal{R}(\{\mathcal{S}\})$ is of arbitrary shape – depending on $out^{\mathcal{S}}(\cdot)$. Hence, computing these intersections, unions and complements is arbitrarily complex.

The number of possible *Collection*s is huge: Having $|P(\mathcal{A})| = 2^d - 1$ subspaces yields $2^{|P(\mathcal{A})|} - 1$ possible *Collection*s. The number of possible combinations of two *Collection*s to obtain *Hidden* is even larger. Thus, in the first part of this work we focus on two cases, $Collection_1 = \{\mathcal{A}\}$ (the full space) and $Collection_2 = \{\{1\}, \ldots, \{d\}\}$ (each one-dimensional subspace).

Notation 1. ***HiddenIn$\mathcal{A}$*** *refers to*

$$Collection_{inlier} = \{\{1\}, \ldots, \{d\}\}, \quad Collection_{outlier} = \{\mathcal{A}\}$$

HiddenFrom$\mathcal{A}$ *to*

$$Collection_{inlier} = \{\mathcal{A}\}, \quad Collection_{outlier} = \{\{1\}, \ldots, \{d\}\}$$

HiddenIn$\mathcal{A}$ means that outliers are detectable in the full space, but not in any one-dimensional projection. *HiddenFrom$\mathcal{A}$* is the opposite: Outliers are not detectable in the full space, but in at least one of the one-dimensional projections.

Example 2. *In Figure 2c, the Mahalanobis distance is used to identify outliers. The red crosses are hidden outliers in the settings just proposed. The square represents the bound for inliers in both subspaces of Collection $= \{\{1\}, \{2\}\}$. The circle represents the inlier bound for points that are inliers in Collection $= \{\mathcal{A}\} = \{\{1, 2\}\}$. Hidden outliers in setting HiddenIn$\mathcal{A}$ are points inside the square but outside of the circle. Analogously, hidden outliers in setting HiddenFrom$\mathcal{A}$ are points outside of the square but inside the circle.*

In some cases we will refer to a more general form of *HiddenIn$\mathcal{A}$* and *HiddenFrom$\mathcal{A}$*, i.e., where one collection is an arbitrary partition of $\mathcal{A}$ into subspaces.

When analysing characteristics of *Hidden*, we will make use of the relative volume of a region. More explictly, we use it to bound the region of hidden outliers.

Definition 4. *Let a region $\mathcal{R} \in \mathbb{R}$ be given. The* ***relative volume of $\mathcal{R}$*** *is defined as $RelativeVolume(\mathcal{R}) := Volume(Full\mathcal{R} \cap \mathcal{R}) \div Volume(Full\mathcal{R})$.*

Lemma 2. *An upper bound on $RelativeVolume(Hidden)$ is the minimum of $RelativeVolume(In\mathcal{R}(Collection_{inlier}))$ and $RelativeVolume(Out\mathcal{R}(Collection_{outlier}))$.*

Thus, if the relative volume of $Out\mathcal{R}(Collection_{outlier})$ or $In\mathcal{R}(Collection_{inlier})$ is very small, e. g., zero, we know that the relative volume of *Hidden* cannot be larger.

In a next step, we investigate specific scenarios with an outlier-detection method using the Mahalanobis Distance. Having such a specific outlier notion allows to derive distinct characteristics of *Hidden*.

4.3 Formal Results

Motivation for Theorem 1: Figure 2c is a two-dimensional example illustrating hidden outliers. The lines are outlier boundaries using the Mahalanobis distance. In this two-dimensional case, hidden outliers for both settings *HiddenIn$\mathcal{A}$* and *HiddenFrom$\mathcal{A}$* can exist. We wonder whether this eventuality of having hidden outliers extends to higher dimensionalities and more general subspace selections. We answer this question by analysing a more general scenario. In our two sample settings, there are two kinds of subspace selections. We have $Collection_1 = \{\{1\}, \ldots, \{d\}\}$ and $Collection_2 = \{\mathcal{A}\}$. To generalize this, we replace $Collection_1$ with an arbitrary partition of the attribute space.

Theorem 1. *Let $\mathcal{A}$ be the full data space and Collection a non-trivial (i.e., $\neq \mathcal{A}$) partition of $\mathcal{A}$ into subspaces. Let the number of dimensions of $\mathcal{A}$ and of each subspace in*

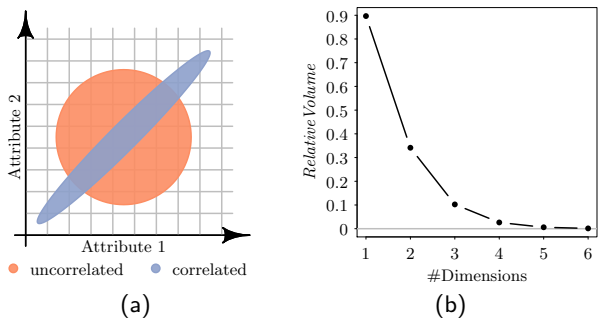


Figure 3: Motivation for Theorem 2 and 3

Collection converge to infinity. Let the data attributes be i.i.d. with $N(0,1)$. Then there exists a hidden outlier that is outlier in at least one subspace of Collection and inlier in $\mathcal{A}$. There also exists a hidden outlier that is outlier in $\mathcal{A}$ but inlier in each subspace of Collection.

All proofs are in the appendix (Section B). We have assumed that the dimensionality goes to infinity in order to approximate *Quantile* in the proof. However, Figure 2c shows that the theorem holds even in a two-dimensional case. Our experiments will show that it holds for other data sets as well.

Motivation for Theorem 2: Next, we consider the effect of correlation on the relative volume of $In\mathcal{R}(\{\mathcal{A}\})$. Figure 3a displays $In\mathcal{R}(\{\mathcal{A}\})$ in a two-dimensional example. The circle is for the case that Attributes 1 and 2 are uncorrelated. The ellipse stands for strong correlation. The volume of the ellipse is smaller than the one of the circle. Thus, a higher correlation seems to imply a smaller relative volume of $In\mathcal{R}(\{\mathcal{A}\})$ and a larger relative volume of $Out\mathcal{R}(\{\mathcal{A}\})$. Theorem 2 formalises this for data spaces of arbitrary dimensionality.

THEOREM 2. *Let subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$, both of dimensionality d and MVN distributed, be given, and let the attributes in $\mathcal{S}_1$ be i.i.d. with $N(0,\lambda)$. The covariance matrix in $\mathcal{S}_1$ is $\mathbf{\Sigma}_1$ and in $\mathcal{S}_2$ $\mathbf{\Sigma}_2$. It holds that $diag(\mathbf{\Sigma}_1) = diag(\mathbf{\Sigma}_2)$, and that $\mathbf{\Sigma}_2$ has off-diagonal elements (i.e., covariance). Then we have:*

$$RelativeVolume(In\mathcal{R}(\mathcal{S}_1)) \geq RelativeVolume(In\mathcal{R}(\mathcal{S}_2))$$
$$\Leftrightarrow RelativeVolume(Out\mathcal{R}(\mathcal{S}_1)) \leq RelativeVolume(Out\mathcal{R}(\mathcal{S}_2))$$

This theorem relies on one further technical assumption spelled out in the appendix which also contains the proof.

Motivation for Hypothesis 2: Theorem 2 reasons on the influence of correlation on inlier and outlier regions. In *HiddenIn*$\mathcal{A}$, the outlier region is $Out\mathcal{R}(\{\mathcal{A}\})$. In *HiddenFrom*$\mathcal{A}$, the inlier region is $In\mathcal{R}(\{\mathcal{A}\})$. In both settings, the respective other region depends on a *Collection* consisting of only one-dimensional subspaces. Correlation does not affect the distribution within these subspaces and hence does not affect the relative volume. Lemma 2 states that the minimum of the relative volumes of inlier and outlier region is an upper bound on the relative volume of *Hidden*. Thus, if one of them increases or decreases this bound might do so as well.

HYPOTHESIS 2. *Correlated data has a smaller relative volume of Hidden in setting HiddenFrom*$\mathcal{A}$ *than uncorrelated data. In HiddenIn*$\mathcal{A}$*, it is larger.*

If this hypothesis holds, it is more difficult to hide outliers in correlated subspaces in *HiddenFrom*$\mathcal{A}$ and less difficult in *HiddenIn*$\mathcal{A}$. We evaluate this assumption using various data sets and outlier-detection methods in Section 5.2.2.

Motivation for Theorem 3: Zimek et al. [20] have studied outlier detection in high-dimensional Euclidean spaces. They have shown that, if the radius is fixed, the relative volume of a hyper sphere becomes smaller for more dimensions. As we can see in Figure 3a, uncorrelated attributes give way to a spherical shape of $In\mathcal{R}(\{\mathcal{A}\})$. However, its radius depends on the $Quantile(0.975, |\mathcal{S}|)$ from Equation 2, and thus varies. In Figure 3b, we see that this varying radius does not seem to do away with that effect described in [20]. The relative volume of $In\mathcal{R}(\{\mathcal{A}\})$ tends to zero. Theorem 3 formalizes this characteristic for the inlier regions in both of our settings.

THEOREM 3. *Let a data space over $\mathcal{A}$ whose attributes are i.i.d. with $N(0,1)$ be given. Let $Collection = \{\{1\},\ldots,\{d\}\}$. Then:*

$$\lim_{d\to\infty} RelativeVolume(In\mathcal{R}(\{\mathcal{A}\})) = 0$$
$$\lim_{d\to\infty} RelativeVolume(In\mathcal{R}(Collection)) = 0$$

Lemma 2 tells us that the relative volume of *Hidden* is bounded by the relative volume of inlier and outlier regions. Theorem 3 states that in both our setting the relative volume of the inlier region tends to zero for arbitrarily large numbers of attributes. Thus, in both *HiddenIn*$\mathcal{A}$ and *HiddenFrom*$\mathcal{A}$ the relative volume of *Hidden* is close to 0 for many dimensions. Hence, it is difficult to hide outliers in both settings when the number of dimensions is high.

4.4 Algorithm for *Hidden*

So far, we have studied characteristics of *Hidden* analytically depending on certain assumptions. We now propose an algorithm which places hidden outliers efficiently, for any instantiation of $out^{\mathcal{S}}(\cdot)$, subspace selections and data set.

Our algorithm is randomized, i.e., checks for random points $\vec{x} \in [u,l]^d$ whether they are in *Hidden*. The baseline we propose samples these points according to a uniform distribution with domain $[u,l]^d$. However, inspecting such samples would not only be extremely expensive, it also would not take into account that *Hidden* can be a very small portion of $Full\mathcal{R}$. See Figure 2c. The red crosses indicate some areas of points in *HiddenIn*$\mathcal{A}$ and *HiddenFrom*$\mathcal{A}$. We have computed these areas by detecting outlier positions in each attribute in isolation as well as in the full space. While the Region *HiddenIn*$\mathcal{A}$ is rather large, *HiddenFrom*$\mathcal{A}$ is not. If *Hidden* is small, an algorithm which concentrates on this part of the data space is desirable. However, the algorithm should also inspect points exhaustively otherwise. A placement that is always exhaustive however would leave aside Hypothesis 1. It has stated that hidden outliers are close to *DB*. According to it, it is unlikely that an extreme position, a point far from *DB*, is a hidden outlier. To facilitate a placement that is adaptive in this spirit, we specify a parameter to model the probability of positions to be checked.

In particular, points next to existing data points $\vec{y} \in DB$ will have a higher likelihood. In the following enumerated list, we discuss the probability of positions being checked and how this check is performed.

4.4.1 Probability of a Position

A straightforward approach would be to only check points $\vec{x}$ closer than some threshold to existing data objects. However, we should also consider the distance of $\vec{x}$ to the attribute bounds. To this end we introduce the parameter $eps \in [0, 1]$. Figure 4 graphs the probability of a position being checked, for a single attribute and data point y. We use the log scale for better illustration. The area of both rectangles is 1. If $eps = 0$ we do not allow for any distance greater than 0 to y. Thus, the probability of checking y is 1 and of any other position 0. If $eps = 1$ we allow for any distance, as long as positions do not exceed the attribute bounds. We set the probability of checking x to be constant and thus to $\frac{1}{u-l}$. If $0 < eps < 1$, a point x between $y - eps \cdot (y-l)$ and $y + eps \cdot (u-y)$ is checked with a probability $\frac{1 \div eps}{u-l}$. Any point outside of these bounds is not checked.

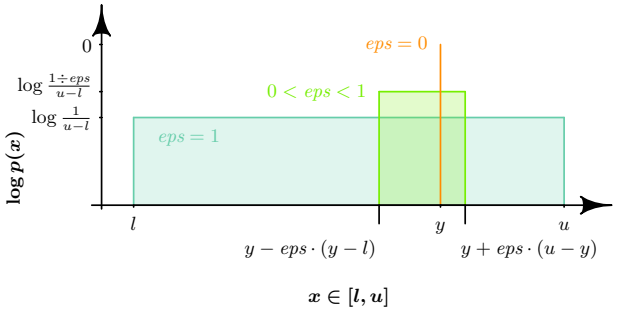


Figure 4: Exemplary probability density of checking random points regarding a single attribute and data object y.

Definition 5. *Let $\vec{y}_1, \ldots, \vec{y}_n \in [l, u]^d$ and $eps \in [0, 1]$ be given. The **surrounding region** of a single observation $\vec{y}_j$ is defined as:*

$$Surr\mathcal{R}^{eps}(\vec{y}_j) := \left\{ \vec{x} \in [l,u]^d \;\middle|\; \max_{i \in A} \left(\frac{y^{(i)} - x^{(i)}}{u - y^{(i)}}, \frac{x^{(i)} - y^{(i)}}{y^{(i)} - l} \right) \leq eps \right\}$$

*The **surrounding region** of several observations $\vec{y}_1, \ldots, \vec{y}_n$ is $Surr\mathcal{R}^{eps}(\vec{y}_1, \ldots, \vec{y}_n) := \bigcup_{j=1}^{n} Surr\mathcal{R}^{eps}(\vec{y}_j)$.*

Thus, $Surr\mathcal{R}^{eps}(\vec{y})$ consists of all $\vec{x}$ whose probability of being checked is greater than zero. Figure 5 displays the surrounding region for several points in an example where $|DB| = 4$.
The following two lemmas feature useful characteristics of the surrounding region.

Lemma 3. *If $eps = 0$ then $Surr\mathcal{R}^0(\vec{y}_j) = \{\vec{y}_j\}$ and $Surr\mathcal{R}^0(\vec{y}_1, \ldots, \vec{y}_n) = \{\vec{y}_1, \ldots, \vec{y}_n\}$.*

Lemma 4. *If $eps = 1$ then $Surr\mathcal{R}^1(\vec{y}_j) = Full\mathcal{R}$ and $Surr\mathcal{R}^1(\vec{y}_1, \ldots, \vec{y}_n) = Full\mathcal{R}$.*

Hence, the surrounding region can consist of solely the observations themselves, the entire data space or a middle ground

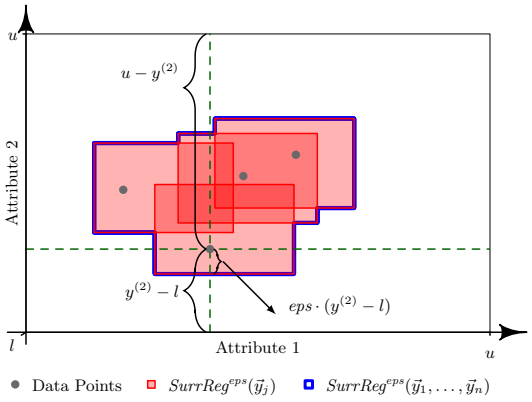


Figure 5: Example of range distance and surrounding region

between these extremes ($0 < eps < 1$). Placing random points only in $Surr\mathcal{R}^{eps}(DB)$ does away with the difficulties of an exhaustive placement. However, our approach so far features a parameter (eps), and it is unclear how to choose its value. We will discuss this in Section 4.4.4.

Regarding the selection of random points, a surrounding region gives way to a probability distribution from which to draw the samples, as follows

$$p(\vec{x}) \propto \frac{1}{n} \sum_{j=1}^{n} \mathbb{1}_{\vec{x} \in Surr\mathcal{R}^{eps}(\vec{y}_j)} \tag{3}$$

$p(\vec{x})$ is the multivariate generalisation of $p(x)$ in Figure 4. We now propose the algorithm in Figure 1 to sample points, given the density function. The first step to obtain $\#Samples$ random samples from $p(\vec{x})$ is to randomly draw $\#Samples$ data points from DB. For every attribute value of each such point, the algorithm calculates two new positions, one towards the upper attribute limit u and one towards the lower limit l. Both are scaled by eps. The algorithm then determines randomly whether the sample has the position next to u or l. This results in $\#Samples$ random samples from $p(\vec{x})$.

4.4.2 Checking Positions

The next step necessary to place hidden outliers is to check if a point is a hidden outlier or not. See Algorithm 2 for our algorithm. For a given point, it checks if it is an inlier in each subspace in $Collection_{inlier}$ and an outlier in at least one subspace of $Collection_{outlier}$. Thus we can filter sampled points for hidden outliers. Armed with these algorithms, it is now possible to hide outliers in $Surr\mathcal{R}^{eps}(DB)$, for given a data set DB, eps and $\#Samples$, as long as $RelativeVolume(Hidden) > 0$. In the following we will discuss an alternative interpretation of eps which allows to choose eps and define the risk of hidden outliers.

4.4.3 Interpreting eps

To motivate our interpretation we revisit Figure 5. The region $Surr\mathcal{R}^{eps}(\vec{y}_1, \ldots, \vec{y}_n)$ with the blue surrounding is a boundary for the observations. eps controls its tightness. Lemma 4 has stated that, if $eps = 0$, $Surr\mathcal{R}^1(DB)$ contains each point from DB. Thus $p(\vec{x})$ from Equation 3 is the empirical distribution function of DB. If $eps \approx 0$, $p(\vec{x})$ still

Algorithm 1 Sample points using the pdf from equation 3

Input: #Samples, *eps*, *DB*
Output: Samples $\vec{x}_1, \ldots, \vec{x}_{\#\text{Samples}}$

Sample $\vec{y}_1, \ldots, \vec{y}_{\#\text{Samples}}$ from *DB* (with replacement)
for $\vec{y}_j \in \vec{y}_1, \ldots, \vec{y}_{\#\text{Samples}}$ **do**
  for attribute i of $\vec{y}_j$ **do**
    lowerChange ← Sample from $Unif(0,\ y_j^{(i)} - l\ )$
    upperChange ← Sample from $Unif(0,\ u - y_j^{(i)}\ )$
    Select at random if
    $x_j^{(i)} \leftarrow y_j^{(i)} - eps \cdot$ lowerChange
    or
    $x_j^{(i)} \leftarrow y_j^{(i)} + eps \cdot$ upperChange
  end for
end for
return $\vec{x}_1, \ldots, \vec{x}_{\#\text{Samples}}$

is similar to the empirical distribution. However, the closer *eps* is to 1, the less similar the empirical data distribution is to $p(\vec{x})$. Their similarity quantifies how much knowledge $p(\vec{x})$ reveals on observations in *DB*. Thus, one can interpret *eps* as an indication of how much information on the data (*DB*) is used in the sampling procedure.

4.4.4 Choosing *eps*

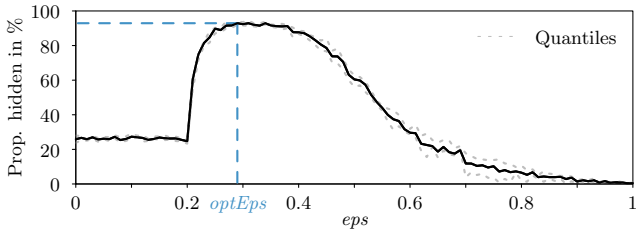


Figure 6: Demonstration of the connection of the proportion of hidden outliers in placed points and *eps*. Using *HiddenFrom*$\mathcal{A}$, Arrhythmia and DBOut (See Section 5.1). The risk is 0.44.

Up to here, *eps* is an exogenous parameter, without the flexibility envisioned. Thus we now add one step to the algorithm. Figure 6 graphs the proportion of sampled points that are hidden outliers in one example setting. The maximum is reached at $eps \approx 0.3$. Hence, in this example, 0.3 is the value that allows for the best placement of hidden outliers. We refer to the *eps* that maximises the proportion of hidden outliers as *optEps*. This is not just the optimum for *eps* but also allows for the computation of the risk of hidden outliers. Usually there is no knowledge on the dependency between *eps* and the proportion of hidden outliers. This is why we propose to use a genetic algorithm to find *optEps*.

4.4.5 The risk of hidden outliers

Definition 6. *Let DB, an outlier detection method* $out^{\mathcal{S}}(\cdot)$, $Collection_{inlier}$ *and* $Collection_{outlier}$ *be given. The* ***risk of attacker success*** *is the harmonic mean of optEps and the proportion of hidden outliers the attacker is able to hide in* $Surr\mathcal{R}^{optEps}(DB)$.

This risk has domain $[0, 1]$. If hiding outliers is difficult, the risk of the data owner is small. Recall our interpretation of *eps* as the amount of information known on the data.

Algorithm 2 Filter sample points for hidden outliers

Input: $\vec{x}_1, \ldots, \vec{x}_{\#\text{Samples}}$, *DB*, $Collection_{outlier}$, $Collection_{inlier}$, $out^{\mathcal{S}}(\cdot)$
Output: Set of hidden outliers *SamplesHidden*

for $\vec{x}_j \in \vec{x}_1, \ldots, \vec{x}_{\#\text{Samples}}$ **do**
  IsInlierInAll ← TRUE
  IsOutlier ← FALSE
  for type ∈ {outlier, inlier} **and for** $\mathcal{S} \in Collection_{\text{type}}$ **do**
    outRes ← $out^{\mathcal{S}}(\vec{x}_j)$
    if outRes ≠ type = inlier **then**
      IsInlierInAll ← FALSE
    end if
    if outRes = type = outlier **then**
      IsOutlier ← TRUE
    end if
  end for
  if IsInlierInAll & IsOutlier **then**
    Add $\vec{x}_j$ to *SamplesHidden*
  end if
end for
return *SamplesHidden*

optEps ≈ 1 means that an attacker does not need any information on *DB* to place hidden outliers. If both *optEps* and the maximal proportion of hidden outliers are high, it is easy to hide outliers, and the risk is high. If only one of them or both are low, the risk is also low. Hence, the risk is low if an attacker either needs much knowledge on the data and/or placed points rarely are hidden outliers.

4.4.6 Complexity

The algorithm we propose exclusively targets at high result quality. We deem absolute runtime less important, as long as it is not excessive, since a data owner will conduct the analysis proposed here offline. Having said this, we nevertheless discuss the worst case complexity of our solution. When placing hidden outliers for a given *eps*, the algorithms performs $\#Calc = \#Samples \cdot |Collection_{inlier}| \cdot |Collection_{outlier}|$ calculations. The maximal number of fitness-function evaluations by the genetic algorithm is *maxFitEval*.

Lemma 5. *The worst case complexity of our algorithm is* $O(\#Calc \cdot maxFitEval)$.

4.4.7 Summary of the Algorithm

The Algorithm needs four inputs: both subspace collections ($Collection_{inlier}$, $Collection_{outlier}$), an outlier detection method ($out^{\mathcal{S}}(\cdot)$) and the number of samples (#Samples). An additional optional input is *eps*; when not supplied, the algorithm itself determines *eps* (*optEps*). First, the algorithm obtains #Samples points from the probability distribution in Equation 3 (See Algorithm 1). Then these points are filtered. Only points that are inlier in each subspace of $Collection_{inlier}$ and outlier in at least one subspace of $Collection_{outlier}$ according to $out^{\mathcal{S}}(\cdot)$ remain. See Algorithm 2. They are hidden outliers. When *eps* is not given, the algorithm repeats this procedure for different values of *eps*. A heuristic generates values of *eps*. They target at maximising the share of hidden outliers in each sample.

5. EXPERIMENTS

In Section 4.3, we have derived characteristics of *Hidden* analytically, assuming a specific outlier definition and underlying data distribution. In our experiments we investigate its behaviour in terms of other outlier definitions and data sets using our algorithm. The experiments show the general ability of our algorithm to place hidden outliers and the vulnerability of different detection methods. We also study the role of *optEps*, e.g., whether there exists a unique one.

5.1 Experiment Setup

5.1.1 Outlier detection

Additionally to the Mahalanobis distance we investigate three other outlier definitions. One of them, follows the (k, $dmax$)-Outlier, short DBOut, proposed in [7]. A point is an outlier if at most k objects have a distance less than $dmax$. The distance used is the euclidean metric. Hence, solely the dimensionality of a subspace implies different magnitudes of distances [20]. That is why, instead of using a fixed $dmax$ for each subspace, we use an adaptive $dmax$. In particular, we set $dmax = \sqrt{|\mathcal{S}|}$. The last two methods we use are ABOD [8] and LoOP [9]. ABOD uses angles to obtain the outlierness of a point. These angles are much more stable in higher dimensions than $\mathcal{L}_p$-distances. [8] proposes three different implementations of ABOD which incorporate different tradeoffs between performance and result quality. We use the fastest implementation, FastABOD. LoOP is an adoption of the well known LOF [1] that incorporates a density based on the neighbourhood of data points. This allows to find density based outliers without assuming a specific distribution. In comparison to LOF, LoOP returns a score that lies in $[0, 1]$ and implies an outlier probability instead of a score in $[0, \infty]$. Except for FastABOD and LoOP all methods already output a binary signal if a data point is an outlier or not. FastABOD and LoOP output scores. Regarding LoOP a low score indicates usual observations, as for FastABOD a high score. In our experiments we need an automatic threshold that allows to transform that score to a binary signal. Regarding FastABOD we decided to use the empirical 2.5 % quantile of the resulting scores to this end. For LoOP we used a threshold of 0.5. Due to the high complexity of FastABOD, we only used datasets with at most 500 observations when using FastABOD. This applies to half of the datasets used. We set the neighbourhood size to $k = 5$.

5.1.2 Datasets

Two data sets we use are artificial and 14 are real-world benchmark data sets, including two high dimensional data sets from the UCI ML Repository [2], namely Madelon and Gisette (500 and 5,000 attributes). The remaining real world datasets are from [2]. We always use the normed data, and when the data has been down-sampled we use version one. The two artificial data sets are produced by sampling from a multivariate Gaussian distribution, each with 500 observations and 30 attributes. One data set is from a MVN distribution where each attribute is i.i.d N(0,1) , referred to as '*MVN* cor.'. In the second one '*MVN*', where attributes are N(0, 1) distributed as well, each pair of attributes is correlated with a covariance of 0.8. They are mostly used for the experiments studying Hypothesis 2. To obtain comparability across data sets, each data sets has been normalized (i. e., $l = 0$ and $u = 1$).

[2]UCI ML Repository: http://archive.ics.uci.edu/ml/

5.1.3 Subspace selection

To evaluate our theoretical findings, we have a deterministic procedure for subspace selection (*HiddenIn*$\mathcal{A}$ and *HiddenFrom*$\mathcal{A}$). Only for Theorem 1 we need to sample subspace partitions. However, when evaluating the general quality of our approach, instantiations of $Collection_{inlier}$ and $Collection_{outlier}$ are much less obvious. We need *realistic* and *diverse* instantiations. However, the number of possible combinations is daunting. On the other hand, the relationship of subspace size and number of possible subspaces, exemplary displayed in Figure 9a, implies the following: We assume that one normally checks low- and high-dimensional subspaces for outliers (green area). Hence, it is most likely that hidden outliers occur in the subspaces with a medium number of dimensions (red area). Thus, these outlier subspaces are a natural selection. However, even with these restrictions, the number of outlier and even inlier subspaces can still be infeasibly large. Thus, we sample them according to the procedure in Algorithm 3.

Algorithm 3 Sample inlier and outlier subspaces

Input: #Subspaces, $\mathcal{A} = \{1, \ldots, d\}$, $out^{\mathcal{S}}(\cdot)$
Output: Subspace collections for inlier and outlier

if $out^{\mathcal{S}}(\cdot)$ is FastABOD or LoOP **then**
  $\text{combs}_{\text{inlier}} \leftarrow \{\mathcal{S} \subseteq \mathcal{A} : |\mathcal{S}| \geq d - 2\}$
else
  $\text{combs}_{\text{inlier}} \leftarrow \{\mathcal{S} \subseteq \mathcal{A} : |\mathcal{S}| \leq 2\}$
end if
$Collection_{inlier} \leftarrow$ Sample #Subspaces from $\text{combs}_{\text{inlier}}$
$\tilde{\mathcal{A}} \leftarrow \{a \in \mathcal{S} : \exists\, \mathcal{S} \in \text{inlierCombs}\}$
$\text{combs}_{\text{outlier}} \leftarrow \{\mathcal{S} \subseteq \tilde{\mathcal{A}} : |\mathcal{S}| = \lfloor \frac{d}{2} \rfloor\}$
$Collection_{outlier} \leftarrow$ Sample #Subspaces from $\text{combs}_{\text{outlier}}$

return $Collection_{inlier}$, $Collection_{outlier}$

First we sample subspaces for $Collection_{inlier}$. For outlier detection methods for high-dimensional spaces (FastABOD and LoOP), we use the large subspaces as inlier subspaces (right green area). For the other outlier detection techniques, we use the smaller subspaces (left green area). Then we obtain the attributes that are contained in the sampled inlier subspaces. From those we sample the outlier subspaces. This guarantees that attributes from inlier and outlier subspaces overlap.

5.2 Evaluating Theoretical Findings

In the first experiments, we investigate the generalizability of our theoretical findings from Section 4.3. The experiments approximate the scenarios described in the theorems and hypotheses using various data sets and outlier detection methods, cf. Section 5.1.

5.2.1 Theorem 1

The theorem states that hidden outliers exist when either $Collection_{inlier}$ or $Collection_{outlier}$ is a partition of the full attribute space $\mathcal{A}$ into subspaces, and the other one is $\mathcal{A}$ itself. We investigate the generalizability of this statement by varying the data distribution and the outlier detection method.

For all data sets we create a number *Collection*s by randomly dividing $\mathcal{A}$ into partitions. For each outlier detection scheme and *Collection* we compute the maximal proportion of hidden outliers regarding two selections of $Collection_{inlier}$ and $Collection_{outlier}$. With the first one, $Collection_{inlier}$ equals *Collection* and $Collection_{outlier} = \mathcal{A}$. In the other case, $Collection_{outlier}$ equals *Collection* and $Collection_{inlier} = \mathcal{A}$. We record how often the proportion of hidden outliers is not 0, i.e., one can hide outliers. If Theorem 1 is generalizable, this should be possible. Table 5.2.1 lists the percentages of runs where we have been able to hide outliers.

	MDist	DBOut	LoOP	FastABOD
*	63.21	73.57	70.36	93.33
**	36.79	83.57	78.93	91.93

* $Collection_{outlier} = \mathcal{A}$, ** $Collection_{inlier} = \mathcal{A}$

Table 1: Percentage of runs with more than zero hidden outliers

In most cases our algorithm is able to place hidden outliers. Surprisingly, regarding *MDist* in particular, the success rate is rather low. This is in some contrast to our formal result that states there exist hidden outliers in these cases. The other detection methods show higher success rates. Thus, we conclude that Theorem 1 is generalizable to some extent.

5.2.2 Hypothesis 2

The hypothesis states that it is more difficult to place inliers in correlated subspaces in setting *HiddenFrom*$\mathcal{A}$ and less difficult in setting *HiddenIn*$\mathcal{A}$. To investigate this we try to hide outliers in both settings using different outlier detection schemes. In one data set, attributes are correlated (*MVN* corr.). In another one they are not (*MVN*). To focus on the effects of correlation, the data has only 10 attributes. If Hypothesis 2 holds we should see an increase in the proportion of hidden outliers from uncorrelated to correlated data in *HiddenIn*$\mathcal{A}$ and a decrease in *HiddenFrom*$\mathcal{A}$. Table 5.2.2 lists the results: the percentage obtained in each dataset, the raw difference between the two results and a relative difference. The last entry is obtained by dividing the raw difference by the maximal percentage the detection algorithm has obtained in any of the two datasets.

		MDist	DBOut	LoOP	FastABOD
*	*MVN*	68.58	1.04	13.30	20.68
	MVN cor.	71.84	1.32	63.06	57.18
	Difference	3.26	0.28	49.76	36.50
	Relative	4.54	21.21	78.91	63.83
**	*MVN*	0.62	21.60	31.28	0.68
	MVN cor.	0.36	19.90	20.76	0.40
	Difference	-0.26	-1.70	-10.52	-0.28
	Relative	-41.94	-7.87	-33.63	-41.18

* *HiddenIn*$\mathcal{A}$, ** *HiddenFrom*$\mathcal{A}$

Table 2: Difference in percentage of hidden outliers

All detection methods meet the expectation. We find it interesting that the magnitude of change of proportion is very different for *HiddenIn*$\mathcal{A}$ and *HiddenFrom*$\mathcal{A}$. However, the proportion of placed hidden outliers on each dataset also varies greatly. In summary, although the extents are different, the experiments confirm the hypothesis.

5.2.3 Generalising Theorem 3

The theorem states that hiding outliers in high dimensional data is likely to be difficult in both our settings. To test the theorems generalizability we try to hide outliers using several outlier detection schemes and data of different dimensionality. The data of different dimensionality should still be comparable. Thus, to obtain a data set of a specific dimensionality we sample the desired number of attributes from data sets with many attributes. To mitigate effects caused by peculiarities of individual attributes, we repeat the whole procedure and average the results. Figure 7 graphs the results for up to 100 dimensions.

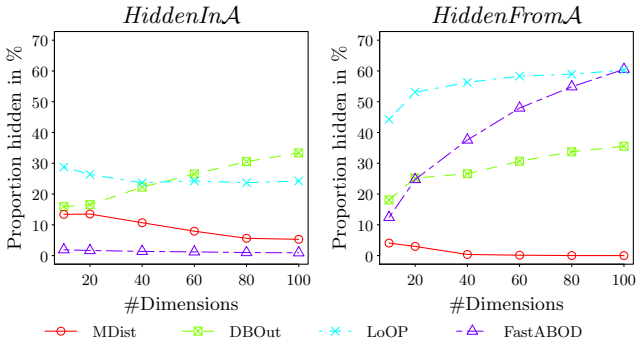


Figure 7: Proportion of hidden outliers versus dimensionality

The theorem does not seem to hold in a more general setting. Using DBOut and LoOP, that proportion tends to be low only in *HiddenIn*$\mathcal{A}$. With FastABOD the proportion does not tend to be low in any of the two settings.

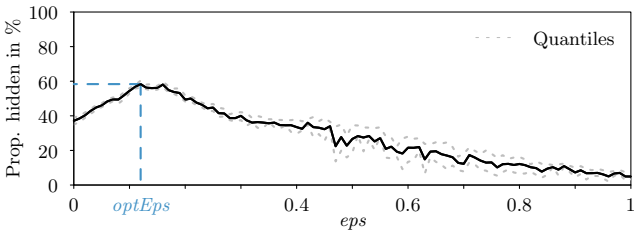
5.3 Investigating *optEps*

The next experiments target at a crucial parameter of our algorithm. The proposed algorithm is based on a sampling distribution parametrized by *eps*. The *eps* that maximizes the proportion of hidden outliers, *optEps*, is important: It allows to quantify the risk of data owners. We now investigate the dependency between *eps* and that proportion, to analyse if *optEps* usually exists, i.e., if there is a global maximum of the dependency.

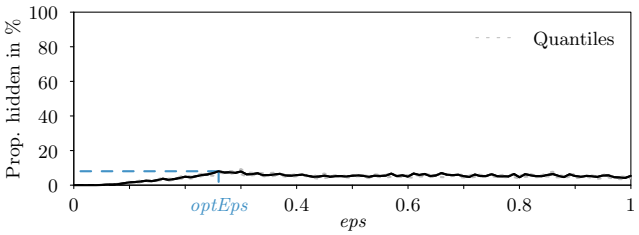
Figures 6 and 8 illustrate the dependency between the proportion of hidden outliers and *eps*. In Figures 6 and 8a we see a very distinct *optEps*. However, Figure 8b shows that this is not always the case. The risk is highest with 0.45 in Figure 6. *optEps* as well as the maximal proportion are relatively high. Figure 8b has the lowest risk with 0.12.

5.4 General Quality

In these final experiments, we study the general ability of our algorithm to place hidden outliers. We also compare our approach to a baseline. Since this article is first to study hidden outliers, there is no explicit competitor, and we choose uniform full space sampling as baseline, which is equivalent to fixing *eps* to 1. We will declare success if there is an increase in the quality of placing hidden outliers, and this increase is significant, e.g., a factor of at least two or three. Recall that our algorithm requires data, $out^{\mathcal{S}}(\cdot)$, $Collection_{inlier}$, $Collection_{outlier}$ as input. The subspace selection has been derived in Section 5.1.3. Regarding the data, we look at all datasets introduced in 5.1.2 and sample



(a) Using *HiddenFromA*, Parkinson and LoOP. Risk: 0.2



(b) Using *HiddenInA*, Lymphography and MDist. Risk: 0.12

Figure 8: Proportion of hidden outliers in placed points versus *eps*

different numbers of attributes and observations. However, we only downsample, upsampling would lead to data that is redundant. To vary $out^{\mathcal{S}}(\cdot)$, we use all outlier detection methods introduced in 5.1.1. Additionally, we obtain *eps* candidates by using a fixed sequence of values instead of a heuristic. This allows for futher analysis on the effect of *eps* and a straightforward comparison to the baseline. We summarize our results to highlight the effect of the number of attributes or observations, used dataset or detection method and *eps*.

5.4.1 Number of Attributes

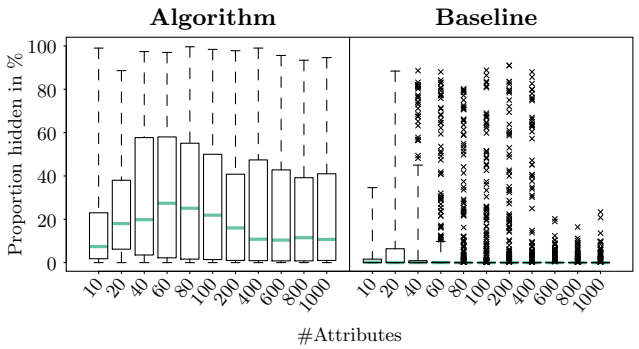


Figure 10: Effect of number of attributes

Figure 10 graphs the effect of the number of attributes. The y-axis displays the share of hidden outliers amongst sampled points, i.e., the success of the placement. For algorithm and baseline the figure shows boxplots of the share of hidden outliers. We observe a significant improvement of our algorithm over the baseline – for high dimensional data in particular. Second, for both alternatives it seems to be more difficult to place hidden outliers in high dimensional datasets.

5.4.2 Number of Observations

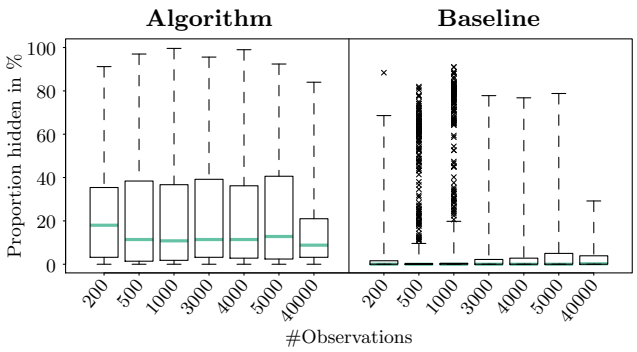


Figure 11: Effect of number of observations

Figure 11 plots the number of observations versus the share of hidden outliers. Again, our algorithm is better than the baseline, but with a bit less distinction. The algorithm improves the baseline by a factor of about 5-10.

5.4.3 Used Dataset

Figure 9(c) is a barplot showing the effect of the data set. We summarize the results over all experiments, i.e., with different samplings of observations and attributes. We see that there are drastic differences between data sets. While placing hidden outliers is successful when using Madelon, this is more difficult with, say, InternetAds. We speculate that the different data densities cause this effect. We have seen this effect in our experiments in Section 5.2.2 as well. As before, our algorithm outperforms the baseline significantly.

5.4.4 Outlier Detection Method Used

Values in %		*MDist*	DBOut	LoOP	FastABOD
*	Algorithm	94.70	80.86	100	100
	Baseline	8.01	61.44	35.62	100
**	Algorithm	6.58	30.14	29.75	25.97
	Baseline	0.03	10.32	2.62	23.67

* Percentage of runs with successfully placed hidden outliers
** Average proportion of hidden outliers

Table 3: Effect of used detection method

To determine effects of the method used, we aggregate the percentage of successful runs and the average share of hidden outliers of all experiments. We have done this for our algorithm as well as for the baseline. See Table 5.4.4. Some detection methods are very prone to hidden outliers while others are not. Next, the gain in success when using our algorithm varies among detection techniques. With *MDist* this gain is high, while it is negligible with FastABOD. However, it is possible to hide outliers with all five detection techniques.

5.4.5 Used eps

Figure 9(b) displays the proportion of hidden outliers versus *eps*. $eps = 1$ is our baseline, i.e., uniform sampling. The median has a peak when *eps* is 0.1. This value results in hidden outliers placed closely to other data points. This confirms Hypothesis 1, i.e., hidden outliers are spatially close to the points in *DB*. The figure also confirms the superiority of our algorithm over the baseline.

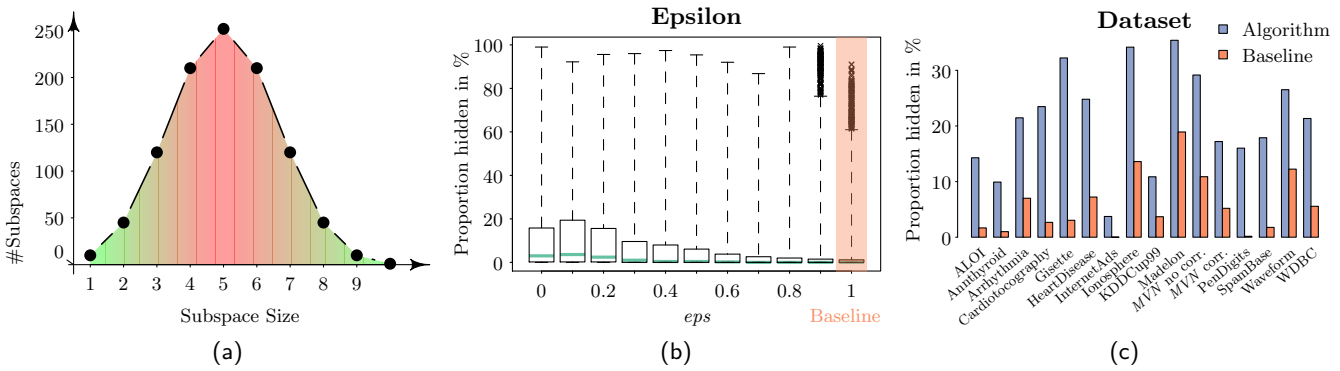


Figure 9: a) displays number of subspaces versus subspace size (using ten attributes) .b) - c) displays success in placement regarding number *eps* and dataset

Summary: The experiments have shown that in many scenarios our approach is able to place hidden outliers irrespective of the data set or the detection method used. Further, our approach is a significant improvement over the baseline. While not in all cases, our algorithm has improved the result by a factor of three or more in many settings.

6. CONCLUSIONS

In this work we have analysed characteristics of hidden outliers, i.e., outliers that are only detectable in certain attribute subspaces. This includes both formal results based on model assumptions and a proposal for an algorithm that places hidden outliers in data. Regarding the first kind of contribution, we prove the existence of hidden outliers in many scenarios and show that the extent of correlation or the number of attributes can have a significant effect on the ease of hiding outliers. The algorithm we have developed places hidden outliers in regions close to existing data objects. We evaluate the generalisability of our formal results experimentally with our algorithm. Some of these results do extend to scenarios not covered by the model assumptions. Further we have shown that our algorithm improves the results with a reference baseline significantly.

7. REFERENCES

[1] M. M. Breunig et al. LOF: Identifying Density-Based Local Outliers. *SIGMOD*, pages 93–104, 2000.

[2] G. O. Campos et al. On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study. *Data Mining and Knowledge Discovery*, pages 1–37, 2016.

[3] V. J. Hodge and J. Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, pages 85–126, 2004.

[4] F. Keller, E. Müller, and K. Böhm. HiCS: High Contrast Subspaces for Density-Based Outlier Ranking. *ICDE*, pages 1037–1048, 2012.

[5] H. Kido, Y. Yanagisawa, and T. Satoh. An Anonymous Communication Technique using Dummies for Location-based Services. *ICPS*, pages 88–97, 2005.

[6] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance Based Outliers in Large Datasets. *VLDB*, pages 392–403, 1998.

[7] G. Kollios et al. Efficient Biased Sampling for Approximate Clustering and Outlier Detection in Large Data Sets. *TKDE*, pages 1170–1187, 2003.

[8] H.-P. Kriegel et al. Angle-Based Outlier Detection in High-Dimensional Data. *SIGKDD*, pages 444–452, 2008.

[9] H.-P. Kriegel et al. LoOP: Local Outlier Probabilities. *CIKM*, pages 1649–1652, 2009.

[10] H.-P. Kriegel et al. Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data. *Advances in Knowledge Discovery and Data Mining*, pages 831–838, 2009.

[11] H.-P. Kriegel et al. Interpreting and Unifying Outlier Scores. *SDM*, pages 13–24, 2011.

[12] H.-P. Kriegel, P. Kröger, and A. Zimek. Outlier Detection Techniques. *Tutorial at the SIGKDD*, 2010.

[13] W. Kröger. Critical Infrastructures at Risk: A Need for a new Conceptual Approach and Extended Analytical Tools. *Reliability Engineering & System Safety*, pages 1781–1787, 2008.

[14] A. Lazarevic and V. Kumar. Feature Bagging for Outlier Detection. *SIGKDD*, pages 157–166, 2005.

[15] E. Müller, M. Schiffer, and T. Seidl. Statistical Selection of Relevant Subspace Projections for Outlier Ranking. *ICDE*, pages 434–445, 2011.

[16] B. Nelson et al. Classifier Evasion: Models and Open Problems. *Workshop on Privacy and Security Issues in Data Mining and Machine Learning*, pages 92–98, 2010.

[17] B. Nelson et al. Near-Optimal Evasion of Convex-Inducing Classifiers. *AISTATS*, pages 549–556, 2010.

[18] W. Prószyñski. On Outlier-Hiding Effects in Specific Gauss–Markov Models: Geodetic Examples. *Journal of Geodesy*, pages 581–589, 2000.

[19] W. Xu, Y. Qi, and D. Evans. Automatically evading classifiers. *Network and Distributed Systems Symposium*, 2016.

[20] A. Zimek, E. Schubert, and H.-P. Kriegel. A Survey on Unsupervised Outlier Detection in High-dimensional Numerical Data. *Statistical Analysis and Data Mining*, 5(5), 2012.

APPENDIX

A. PREREQUISITES FOR PROOFS

We will use the abbreviations $Collection =: \mathcal{SS}$, $In\mathcal{R}(\{\mathcal{S}\}) =: \mathcal{I}^{\mathcal{S}}$ and $u - l =: range$.

The Mahalanobis distance is defined as: $MDist^{\mathcal{A}}(\vec{y}) = \sqrt{(\vec{y}-\vec{\mu})^T \mathbf{\Sigma}^{-1}(\vec{y}-\vec{\mu})}$ where $\vec{\mu}$ is the mean vector and $\mathbf{\Sigma}$ the covariance matrix. W.l.o.g. we assume that $\vec{\mu} = \vec{0}$. As the data is MVN distributed, $MDist^2$ is χ^2_d distributed. The degrees of freedom are determined by the dimension of the data. We can rewrite:

$$\left[MDist^{\mathcal{A}}(\vec{y})\right]^2 = \vec{y}^T \mathbf{\Sigma}^{-1} \vec{y} = \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{A}} y^{(i)} \cdot \sigma^{(i,j)}_{-1} \cdot y^{(j)}$$

$\sigma^{(i,j)}_{-1}$ denotes the entry in the jth column and ith row of the inverse of the covariance matrix. If the attributes are i.i.d. N(0,1) distributed, this reduces to $\left[MDist^{\mathcal{A}}(\vec{y})\right]^2 = \sum_{i\in\mathcal{A}} \left[y^{(i)}\right]^2$. To test a data point for outlier or inlier, we used the outlier initialization in Equation 2. Although this initialization uses the 0.975 quantile, here we will use a general α quantile. The quantile function of a χ^2 distribution is not obtainable in closed form. Thus, we will make use of an approximation. Following the central limit theorem, for large degrees of freedom a χ^2_d distribution can be approximated by a $N(d, \sqrt{2d})$ distribution. Thus:

$$Quantile(\alpha,\ d) \approx d + \sqrt{2d}\, z_\alpha\ ; \quad \frac{\partial\ Quantile(\alpha,\ d)}{\partial\ d} \approx 1 + \frac{z_\alpha}{\sqrt{2d}} \tag{4}$$

where z_α is the α quantile of a standard normal distribution. We can derive that the approximation of the function $Quantile(\alpha,\ d)$ is strictly monotonic increasing. For a fixed distance, e.g., $\left[MDist^{\mathcal{A}}(\vec{y})\right]^2 = Quantile(\alpha,\ d)$, the Mahalanobis distance exhibits an ellipsoid form. I. e., having $\lambda_1, \ldots, \lambda_d$ eigenvalues and $\vec{v}_1, \ldots, \vec{v}_d$ eigenvectors of $\mathbf{\Sigma}$, the ellipsoid has centroid $\vec{\mu}$, and axes $\vec{v}_1, \ldots, \vec{v}_d$. Half the length of each axis is determined by $\sqrt{\lambda_i \cdot Quantile(\alpha,\ d)}$.

Introducing subspaces in this setting is quite trivial. We assume that the full data space is $MVN(\vec{0}, \mathbf{\Sigma})$ distributed. Hence, any subspace $\mathcal{S}$ is also Gaussian. To obtain its mean and covariance matrix we only need to drop the irrelevant variables from each parameter of the full space distribution.

B. PROOFS OF THEOREMS

B.1 Theorem 1

In this proof we will use the normal approximation given in Equation 4. From attributes i.i.d $N(0,1)$ and partitioning $\mathcal{SS}$ follows:

$$\left[MDist^{\mathcal{A}}(\vec{y})\right]^2 = \sum_{i\in\mathcal{A}} \left[y^{(i)}\right]^2 = \sum_{\mathcal{S}\in\mathcal{SS}} \sum_{i\in\mathcal{S}} \left[y^{(i)}\right]^2 = \sum_{\mathcal{S}\in\mathcal{SS}} \left[MDist^{\mathcal{S}}(\vec{y})\right]^2$$

We first prove that a data point $\vec{o}_1$ with

$$o_1^{(i)} = \begin{cases} \sqrt{\frac{Quantile(\alpha,\ |\mathcal{A}|)}{|\mathcal{S}|}} & \text{if } i \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

is an outlier for any $\mathcal{S} \in \mathcal{SS}$ but an inlier for $\mathcal{A}$. We know that $MDist^{\mathcal{A}}(\vec{o}_1) = Quantile(\alpha,\ |\mathcal{A}|)$ and $MDist^{\mathcal{S}}(\vec{o}_1) = Quantile(\alpha,\ |\mathcal{A}|) > Quantile(\alpha,\ |\mathcal{S}|)$. Due to the strictly monotonic increasing quantile function, $\vec{o}_1$ is an inlier regarding $\mathcal{A}$ but an outlier in $\mathcal{S}$. It is important to note that $\vec{o}_1$ is an outlier only regarding subspace $\mathcal{S}$ and not regarding any other subspace in $\mathcal{SS}$.

Moreover, a data point $\vec{o}_2$ defined by

$$o_2^{(i)} = \frac{\sqrt{Quantile(\alpha,\ |\mathcal{S}|)}}{|\mathcal{S}|} \qquad \text{for } \mathcal{S} \in \mathcal{SS}\ :\ i \in \mathcal{S}$$

is an outlier for $\mathcal{A}$ but an inlier for all $\mathcal{S} \in \mathcal{SS}$. It holds that: $MDist^{\mathcal{A}}(\vec{o}_2) = \sum_{\mathcal{S}\in\mathcal{SS}} Quantile(\alpha,\ |\mathcal{S}|)$ and $MDist^{\mathcal{S}}(\vec{o}_2) = Quantile(\alpha,\ |\mathcal{S}|)$. We know that $\vec{o}_2$ is an outlier if $MDist^{\mathcal{A}}(\vec{o}_2) > Quantile(\alpha,\ |\mathcal{A}|)$. We also know that $|\mathcal{A}| = \sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}|$. Hence, in order to show that $\vec{o}_2$ is an outlier in $\mathcal{A}$, we have to show:

$$\sum_{\mathcal{S}\in\mathcal{SS}} Quantile(\alpha,\ |\mathcal{S}|) \overset{!}{>} Quantile\left(\alpha,\ \sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}|\right)$$

$$\sum_{\mathcal{S}\in\mathcal{SS}} \left[|\mathcal{S}| + \sqrt{2|\mathcal{S}|}\ z_\alpha\right] > \sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}| + \sqrt{2 \sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}|}\ z_\alpha$$

$$\sum_{\mathcal{S}\in\mathcal{SS}} \sqrt{|\mathcal{S}|} > \sqrt{\sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}|}$$

$$\sum_{\mathcal{S}_1,\mathcal{S}_2\in\mathcal{SS}} \sqrt{|\mathcal{S}_1|}\sqrt{|\mathcal{S}_2|} > \sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}|$$

$$\sum_{\mathcal{S}_1\neq\mathcal{S}_2\in\mathcal{SS}} \sqrt{|\mathcal{S}_1|}\sqrt{|\mathcal{S}_2|} + \sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}| > \sum_{\mathcal{S}\in\mathcal{SS}} |\mathcal{S}|$$

As $\mathcal{SS}$ is a non-trivial partition, i.e., $\mathcal{SS} \neq \mathcal{A}$, the term $\sum_{\mathcal{S}_1\neq\mathcal{S}_2\in\mathcal{SS}} \sqrt{|\mathcal{S}_1|}\sqrt{|\mathcal{S}_2|}$ is greater than 0, and the inequality holds.

B.2 Theorem 2

This theorem relies on an assumption not explicitly listed in the body of the article. Let λ denote the eigenvalue of $\mathbf{\Sigma}_1$ (algebraic multiplicity of d). Further let $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_d$ denote the eigenvalues of $\mathbf{\Sigma}_2$. We introduce $\varepsilon_1, \ldots, \varepsilon_d$ which satisfy $\lambda + \varepsilon_i = \tilde{\lambda}_i$. Our assumption is that the ε_i's are symmetrical, i. e., for any $\varepsilon_j > 0$ there exist $\varepsilon_k = -\varepsilon_j$.

We know that, for both subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$, Volume($Full\mathcal{R}$) is equal. Using a constant Volume($Full\mathcal{R}$) we can write: $RelativeVolume\left(\mathcal{I}^{\mathcal{S}}\right) \propto \text{Volume}(\mathcal{I}^{\mathcal{S}})$. This volume is the one of a d-ellipse. Let $\lambda_1, \ldots, \lambda_d$ be the eigenvalues of the covariance matrix within a subspace $\mathcal{S}$, then:

$$RelativeVolume\left(\mathcal{I}^{\mathcal{S}}\right) \propto \frac{2\pi^{\frac{d}{2}}}{d\,\Gamma(\frac{d}{2})} \sqrt{Quantile(\alpha,\ |\mathcal{S}|) \prod_{i=1}^{d} \lambda_i} \tag{5}$$

We further know that $\sum_{i=1}^{d} \lambda_i$ is equal to the sum of the trace of the corresponding covariance matrix. The trace of $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ are the same. We have assumed that each attribute in $\mathcal{S}_1$ is i.i.d. $N(0, \lambda)$. Hence, $\mathbf{\Sigma}_1$ is a diagonal matrix with λ in each diagonal element. Thus, λ is the variance and each of the d eigenvalues of $\mathbf{\Sigma}_1$. $\mathbf{\Sigma}_2$ has off-diagonal elements. Hence, the eigenvalues can differ from the ones in $\mathbf{\Sigma}_1$. Using the equality of traces we infer that $\sum_{i=1}^{d} \varepsilon_i = 0$. In order to prove our theorem we need to show that:

$$\prod_{i=1}^{d} \lambda = \lambda^d \geq \prod_{i=1}^{d} \tilde{\lambda}_i = \prod_{i=1}^{d} (\lambda + \varepsilon_i) \tag{6}$$

We introduce $index_{>0} := \{i \in 1, \ldots, d \mid \varepsilon_i > 0\}$. Similarly, the index of $\varepsilon_i = 0$ as $index_{=0}$. Let further be $m = |index_{=0}|$. We can infer that $|index_{>0}| = \frac{d-m}{2}$. Using this, we can write:

$$\begin{aligned}
\prod_{i=1}^{d} (\lambda + \varepsilon_i) &= \left[\prod_{i \in index_{=0}} \lambda \right] \left[\prod_{j \in index_{>0}} (\lambda + \varepsilon_j)(\lambda - \varepsilon_j) \right] \\
&= \lambda^m \left[\prod_{j \in index_{>0}} (\lambda^2 - \varepsilon_j^2) \right] \\
&= \lambda^m \left(\lambda^2 \right)^{\frac{d-m}{2}} - \lambda^m \left[\prod_{j \in index_{>0}} \varepsilon_j^2 \right] \\
&= \lambda^d \underbrace{- \lambda^m \left[\prod_{j \in index_{>0}} \varepsilon_j^2 \right]}_{\leq 0}
\end{aligned}$$

Inserting this in Equation 6 directly proves the theorem. We can also infer that if there is an $\varepsilon_i > 0$, the statement of the theorem extends to $\mathit{RelativeVolume}(\mathcal{I}^{\mathcal{S}_1}) > \mathit{RelativeVolume}(\mathcal{I}^{\mathcal{S}_2})$

B.3 Theorem 3

We first prove the first statement. We know that having attributes that are i.i.d. $N(0,1)$ gives that $\prod_{i=1}^{d} \lambda_i = 1$. If d goes to infinity, the approximation of $\mathit{Quantile}(\alpha,\ |\mathcal{A}|)$ from Equation 4 is exact. This yields:

$$\begin{aligned}
\text{Volume}(\mathcal{I}^{\mathcal{A}}) &\approx \frac{2\pi^{\frac{d}{2}}}{d\,\Gamma(\frac{d}{2})} \sqrt{d + \sqrt{2d}\ z_\alpha} \\
\Rightarrow \quad \left[\lambda(\mathcal{I}^{\mathcal{A}})\right]^2 &\approx \frac{4\pi^d}{d^2\ \left[\Gamma(\frac{d}{2})\right]^2} \left(d + \sqrt{2d}\ z_\alpha\right) \\
&= \underbrace{\frac{4\pi^d}{d\ \left[\Gamma(\frac{d}{2})\right]^2}}_{=:p_1(d)} + \underbrace{\frac{2^{\frac{5}{2}} \pi^d z_\alpha}{\sqrt{d}^3\ \left[\Gamma(\frac{d}{2})\right]^2}}_{=:p_2(d)}
\end{aligned}$$

$p_1(d)$ and $p_2(d)$ are used to simplify the notation. We now derive a recursive formula for the volume of a full data space of dimensionality $d+2$, subsequently referred to as $\tilde{\mathcal{A}}$.

$$\begin{aligned}
\text{Volume}(\mathcal{I}^{\tilde{\mathcal{A}}}) &\approx \frac{4\pi^{(d+2)}}{(d+2)\ \left[\Gamma(\frac{d+2}{2})\right]^2} + \frac{2^{\frac{5}{2}} \pi^{(d+2)} z_\alpha}{\sqrt{d+2}^3\ \left[\Gamma(\frac{d+2}{2})\right]^2} \\
&= \frac{\pi^2 4\pi^d}{(d+2)\ \left[\frac{d}{2}\Gamma(\frac{d}{2})\right]^2} + \frac{2^{\frac{5}{2}} \pi^2 \pi^d z_\alpha}{\left(\sqrt{1+\frac{2}{d}}\sqrt{d}\right)^3\ \left[\frac{d}{2}\Gamma(\frac{d}{2})\right]^2} \\
&= \frac{\pi^2}{\left[\frac{d}{2}\right]^2} \left[\frac{d}{d+2}\, p_1(d) + \frac{1}{\sqrt{1+\frac{2}{d}}^3} p_2(d) \right] \\
&= \underbrace{\frac{\pi^2}{\left[\frac{d}{2}\right]^2}}_{\to 0} \left[\lambda(\mathcal{I}^{\mathcal{A}}) - \underbrace{\frac{2}{d+2}\, p_1(d)}_{\geq 0} - \underbrace{\frac{\sqrt{1+\frac{2}{d}}^3 - 1}{\sqrt{1+\frac{2}{d}}^3} p_2(d)}_{\geq 0} \right]
\end{aligned}$$

Thus, using this recursive formula, we can infer that the volume of the full space inlier region shrinks when increasing d. In Section 4.3 we had required l and u to be sufficiently large. Thus, we can assume that *range* is greater than 1, and that the volume of the full space region ($range^d$) increases with d. Putting all this together yields:

$$\lim_{d \to \infty} \mathit{RelativeVolume}\left(\mathcal{I}^{\mathcal{A}}\right) = \lim_{d \to \infty} \frac{\text{Volume}(\mathcal{I}^{\mathcal{A}})}{\text{Volume}(\mathit{Full}\mathcal{R})} = \frac{0}{\infty} = 0$$

We now turn to the second statement. Think of a point that is an inlier in an attribute. Its value in this attribute lies within $+\sqrt{\mathit{Quantile}(\alpha,\ 1)} := \varepsilon$ and $-\sqrt{\mathit{Quantile}(\alpha,\ 1)}$. We can infer that $\varepsilon < \frac{range}{2}$ as we have assumed *range* to be sufficiently large. Thus:

$$\lim_{d \to \infty} \mathit{RelativeVolume}(\mathit{In}\mathcal{R}(\mathcal{SS}_{inlier})) = \lim_{d \to \infty} \left(\frac{2\varepsilon}{range} \right)^d = 0$$