Calibration and Evaluation of Outlier Detection with Generated Data

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Georg Steinbuß

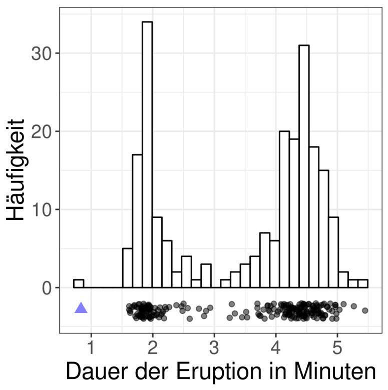
aus Köln

Tag der mündlichen Prüfung:	12.05.2020
Erster Gutachter:	Prof. Dr.-Ing. Klemens Böhm
Zweiter Gutachter:	Prof. Dr. Jörg Sander

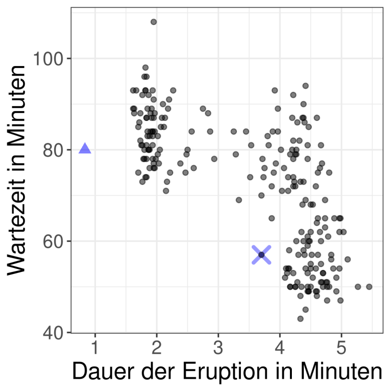
Deutsche Zusammenfassung

Ausreißer sind Beobachtungen, die von den meisten anderen Beobachtungen abweichen. Betrachten wir beispielsweise Daten zu den Eruptionen des Old Faithful Geysir im Yellowstone-Nationalpark (Abbildung 1a). Die Dauer der Eruptionen des Geysir sind in Abbildung 1b dargestellt[1]. Ein Ausreißer — eine extrem kurze Eruption — ist das blaue Dreieck links in Abbildung 1b.

(a) Der Old Faithful[2]



(b) Histogramm



(c) Streudiagramm

Abbildung 1: Eruptionen des Old Faithful Geysir.

Ausreißer können auf Defekte der Messinstrumente oder eines zugrundeliegenden Systems hinweisen. Beispielsweise ein verschlissenes Lager in einem Motor. Eine verlässliche Erkennung ist deshalb wichtig. Ausreißer sind allerdings selten, da sie von den meisten Beobachtungen abweichen. Ohne eine repräsentative Menge an Ausreißern ist es schwer, Methoden zu ihrer Erkennung zu entwickeln bzw. zu überprüfen, welche Typen von Ausreißern diese erkennen. Der Typ von Ausreißern definiert sich durch besondere Eigenschaften der betroffenen Beobachtung. Beispielsweise kann man Ausreißer als *global* oder *lokal* bezeichnen. Beispiel 1 illustriert Ausreißer verschiedenen Typs anhand der Daten des Old Faithful Geysir.

Beispiel 1 (Typen von Ausreißern) *Die Zeit ohne Vorkommnis, die einer Eruption vorausgeht, ist die Wartezeit. Abbildung 1c stellt diese Wartezeit zusammen mit der Dauer der Eruption dar. Auch in diesem Streudiagramm ist der Ausreißer aus Abbildung 1b gut zu erkennen. Es ist auch ein weiterer Ausreißer zu erkennen, dem eine sehr lange Wartezeit*

[1] Auszug aus https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/geyser.html

[2] Quelle: https://pixabay.com/photos/faithful-eruption-old-faithful-3939305/

vorausgeht (> 100 Minuten). Beide Ausreißer sind globale Ausreißer. *Sie fallen bei der Betrachtung aller Beobachtungen auf.* Lokale Ausreißer *hingehen fallen in ihrer lokalen Nachbarschaft auf. Die mit einem blauen Kreuz gekennzeichnete Eruption fällt beispielsweise auf, wenn man die Eruptionen rechts davon, mit einer leicht längeren Eruption aber ähnlicher Wartezeit, betrachtet.*

Mit Beispiel 1 lässt sich auch erkennen, dass sowohl die verschiedenen Typen von Ausreißern als auch die Eigenschaft "Ausreißer" an sich nicht sehr präzise definiert sind: Es ist unklar wie genau eine Beobachtung von anderen Beobachtungen abweichen muss, um definitiv als Ausreißer eines Typs zu gelten oder ob man wirklich jeden Typ als Ausreißer anerkennt. Beispielsweise ist die nötige Entfernung eines globalen oder lokalen Ausreißers zu anderen Beobachtungen subjektiv. Auch ob der lokale "Ausreißer" nicht doch eine ganz normale Beobachtung repräsentiert, ist zumindest fraglich.

Um die Güte von Methoden zur Erkennung von Ausreißern zu vergleichen, werden üblicherweise Daten herangezogen, die eine Kategorisierung in "Ausreißer" und "normale Beobachtungen" aufweisen. Solche Daten sind aber generell eher selten und die entsprechenden Ausreißer haben sehr unterschiedliche und teils widersprüchliche Eigenschaften. Eine klare Klassifizierung des Typs von Ausreißern (z. B. lokal oder global) existiert in solchen Daten ebenfalls nicht. Daher ist eine Evaluation der Qualität der Erkennung von Ausreißern schwierig, bzw. in Bezug auf verschiedenen Typen von Ausreißern gar nicht möglich.

Eine weitere Möglichkeit die Güte von Methoden zur Erkennung von Ausreißern zu vergleichen, ist künstliche Daten zur Evaluation von Methoden zur Erkennung von Ausreißern zu generieren. Dies bietet zwei nennenswerte Vorteile: Die Daten können in beliebigen Mengen generiert werden und die Eigenschaften bzw. Typen von Ausreißern können festgelegt werden.

Zentrale Forschungsfragen

Es gibt bereits einige Ansätze, Ausreißer künstlich zu generieren. Oft liegen den Ansätzen aber unterschiedliche Annahmen über die Eigenschaften von Ausreißern zugrunde, ohne dass die Auswirkung der Annahmen immer klar ist. Daraus leitet sich Forschungsfrage 1 dieser Arbeit ab, die wir in Kapitel 3 behandeln.

Forschungsfrage 1 *Welche Methoden zur Generierung von Ausreißern gibt es und wie werden sie genutzt?*

Um zur Erkennung von Ausreißern Methoden auszuwählen, ist es von großem Vorteil zu verstehen, welche Typen von Ausreißern von welcher Methode gut erkannt werden. Das erlaubt es, eine geeignete Vorauswahl an Methoden zu treffen, um möglichst verschiedene Typen erkennen zu können. Daraus leitet sich Forschungsfrage 2 ab, die wir in Kapitel 4 behandeln.

Forschungsfrage 2 *Lassen künstliche Daten eine differenziertere Evaluation in Bezug auf Ausreißer verschiedenen Typs zu?*

Die bisher erwähnten Typen von Ausreißern beziehen sich immer auf einen festen Datenraum. Bei sogenannten "versteckten" Ausreißern ist das nicht der Fall. Das sind Ausreißer, die als solche nur in bestimmten Teilräumen zu erkennen sind. Ein Teilraum ist jeweils eine Teilmenge der Datenattribute. Beispielsweise fällt der lokale Ausreißer in Beispiel 1 nur im gesamten Datenraum auf. In den beiden Datenattributen für sich betrachtet ist er nicht zu erkennen. Das heißt, in Bezug auf die einzelnen Datenattribute ist er versteckt. Für versteckte Ausreißer ist also nicht nur die Beziehung zu anderen Beobachtungen relevant, sondern auch die Teilräume, in denen die Ausreißer versteckt oder erkennbar sind.

Es existieren im Prinzip keine Daten in denen versteckte Ausreißer bekannt sind und es ist auch generell wenig über die Umstände die versteckte Ausreißer zulassen bekannt. Beides wäre aber wichtig, um Methoden die Ausreißer mithilfe von Teilräumen erkennen, zu evaluieren. Daraus leitet sich Forschungsfrage 3 ab, die wir in Kapitel 5 behandeln.

Forschungsfrage 3 *Lässt sich der Nutzen der Evaluation mit generierten Ausreißern auch auf Methoden übertragen, die Ausreißer mithilfe von Teilräumen erkennen?*

Herangehensweise und Ergebnisse

In diesem Abschnitt gehen wir kurz auf unsere Herangehensweise an die Forschungsfragen und die erreichten Ergebnisse ein.

Forschungsfrage 1

Fast alle der existierenden Ansätze zur Generierung von Ausreißern wurden mit dem Fokus entwickelt, die richtigen Parameter für Methoden zur Erkennung von Ausreißern zu finden. D. h., die Methoden zu kalibrieren. Die Parameter werden dabei so eingestellt, dass die generierten Ausreißer optimal erkannt werden. In Kapitel 3 beschreiben wir, wie diese Optimierung funktioniert und welche Ansätze zur Generierung künstlicher Ausreißer dazu in der Literatur existieren. Diese Ansätze haben wir dazu auf zwei voneinander unabhängige Wege gruppiert. Zum einen über die resultierenden Eigenschaften in Bezug auf die normalen Beobachtungen und zum andern über die Art wie die Ausreißer generiert werden. Beispielsweise haben wir Ansätze gruppiert, die Ausreißer nahe an normalen Beobachtungen generieren. Ein weiteres Beispiel sind Ansätze die Ausreißer generieren, indem von einer Verteilung eine Stichprobe gezogen wird. Beide Gruppierungen geben einen guten Überblick über die bereits existierenden Ansätze und zeigen das bereits großes Spektrum an Ansätzen zur Generierung von Ausreißern mit unterschiedlichen Eigenschaften auf.

In ausführlichen Experimenten haben wir die verschiedenen Ansätze in Bezug auf die Optimierung von Parametern mithilfe verschiedener Realweltdaten verglichen. Die Ergebnisse unterstreichen, dass es nicht den einen Ansatz gibt, der alle anderen Ansätze schlägt. D. h., um durch die Optimierung von Parametern mithilfe künstlicher Ausreißer eine gute Erkennung zu gewährleisten, sollte zumindest ungefähr bekannt sein wie sich die zu erkennenden echten Ausreißer verhalten.

Forschungsfrage 2

In Kapitel 4 formalisieren wir einen generischen Prozess, der es ermöglicht Daten zu generieren, die die Evaluation von Methoden zur Erkennung von Ausreißern ermöglichen. Der Prozess kann so initiiert werden, das die generierten Daten, Ausreißer eines bestimmten Typs enthalten. Vollständig künstliche Daten haben den Vorteil gegenüber Daten, in denen nur die Ausreißer generiert werden, dass sehr genaue Information über den Datenbestand verfügbar sind. Das ermöglicht eine aussagekräftige Interpretation der Ergebnisse, da man dadurch das Ergebnis einer perfekten Erkennung von Ausreißern bestimmen kann. Mit diesem kann man die Ergebnisse der anderen Methoden vergleichen. Dieser Vergleich kann zum Beispiel helfen, offene Potenziale der existierenden Methoden zu entdecken. Zudem begünstigt der formalisierte Prozess, dass die generierten Daten realistisch sind. Das ermöglicht es, die Ergebnisse einer Evaluation zum Teil auch auf Daten aus der realen Welt zu übertragen.

Um den Prozess zu demonstrieren, stellen wir in Kapitel 4 drei Instantiierungen vor. Beispielsweise stellen wir Instantiierungen vor, für lokale oder globale Ausreißer. In einer umfangreichen experimentellen Studie haben wir dann gängige Methoden zur Ausreißererkennung mit den vorgestellten Instantiierungen verglichen. Die Vergleiche zeigen deutliche Unterschiede bei der Genauigkeit der Erkennung von Ausreißern in Bezug auf unterschiedliche Typen. Zusätzlich haben wir Experimente durchgeführt, die aufzeigen, dass die von unseren Instantiierungen generierten künstlichen Daten tatsächlich realistisch sind.

Forschungsfrage 3

Um Forschungsfrage 3 anzugehen, beschäftigen wir uns in Kapitel 5 als Erstes mit der Frage, ob versteckten Ausreißer überhaupt existieren, bzw. unter welchen Umständen. Nur wenn versteckte Ausreißer überhaupt existieren, können sie für eine Evaluierung von Methoden, die Ausreißer mithilfe von Teilräumen erkennen, nützlich sein. Die Frage nach der Existenz von versteckten Ausreißern gehen wir zunächst theoretisch an. Die Aussagen, die wir dabei erhalten, hängen stark von den Teilräumen ab, die den versteckten Ausreißern zugrunde liegen. Um überhaupt klare Aussagen zu erhalten, betrachten wir deshalb unterschiedliche Extreme der Auswahl an verschiedenen Teilräumen. Selbst mit diesen spezifischeren Mengen an Teilräumen, benötigen wir aber sehr einschränkende Annahmen, um Umstände, die versteckte Ausreißer zulassen theoretisch zu analysieren. Eine dieser Annahmen ist beispielsweise, dass die Daten entsprechend einer multivariate Gaußverteilung verteilt sind. Mithilfe der einschränkenden Annahmen können wir aber zeigen, dass versteckte Ausreißer in verschiedenen Extremen von Teilräumen existieren und untersuchen welchen Effekt die Korrelation der Datenattribute auf die versteckten Ausreißer hat.

Als nächsten Schritt stellen wir ein Ansatz vor, um verstecke Ausreißer zu generieren. Die generierten Ausreißer nutzen wir, um unsere analytischen Ergebnisse mit weniger rigiden Annahmen zu bestätigen. Gleichzeitig stellen die generierten versteckten Ausreißern aber auch Schwachstellen von Methoden dar, die Ausreißer mithilfe von Teilräumen erkennen. Deshalb könnten die künstlichen versteckten Ausreißer auch sinnvoll für eine Evaluation entsprechender Methoden eingesetzt werden.

In Bezug auf unsere analytischen Ergebnisse zeigen Experimente mit verschiedenen Realweltdaten und Methoden zur Erkennung von Ausreißern, dass versteckte Ausreißer prinzipiell auch in Fällen existieren, wenn die Annahmen nicht gelten. Genauso verhält es sich mit dem Einfluss der Korrelation auf die versteckten Ausreißer.

Zusammenfassung der Ergebnisse und Ausblick

Mit meiner Dissertation beleuchte ich das Thema der Generierung von Ausreißern. In besonderem Fokus stehen dabei die Aspekte der Evaluierung und Kalibrierung von Methoden der Ausreißererkennung. Die einheitliche Beschreibung existierender Ansätze zur Generierung von Ausreißern hilft, die auftretenden Problem, als auch bereits existierende Lösungsansätze einfach nachzuvollziehen. Zusätzlich haben wir einen neuen Prozess vorgeschlagen, der auf Basis von generierten Daten, die Evaluation von Methoden zur Ausreißererkennung verbessert. Beziehungsweise, der Prozess ermöglicht die Evolution in Bezug auf manche Typen von Ausreißern erst. Die Untersuchungen zu versteckten Ausreißern führen zu interessanten Einsichten in Eigenschaften dieser sehr komplexen Art von Ausreißern, beispielsweise deren Existenz. Im Vergleich zu einer rein theoretischen Herangehensweise, konnten wir mit generierten versteckten Ausreißern die evaluierten Szenarien für versteckte Ausreißer wesentlich realistischer gestalten. Zusätzlich stellen die generierten künstlichen Ausreißer eine interessante Möglichkeit dar, speziell Methoden die Ausreißer mithilfe von Teilräumen erkennen, zu evaluieren.

Damit bietet die von mir geleistete Forschung auch einen guten Startpunkt für viele weitere interessante Forschungsvorhaben, beispielsweise in der Entwicklung neuer Ansätze zur Generierung von Ausreißern. Hier könnte insbesondere eine Verbindung mit Ansätzen die auf existierenden echten Ausreißern basieren große Erfolge versprechen. Der generelle Prozess, den wir entwickelt haben, kann um viele neue Instantiierungen erweitert werden. Diese Erweiterung ist beispielsweise nützlich, um andere Typen von Ausreißern abzudecken. Ein möglicher anderer Typ sind zum Beispiel Ausreißer, die nicht alleine "ausreißen", sondern als kleine Gruppe. Die künstliche Erstellung von versteckten Ausreißern, könnte zukünftig auch zu einer verbesserten Teilraumsuche führen. Ein iterativer Prozess, der abwechselnd versteckte Ausreißer generiert und mithilfe dieser, bessere Teilräume findet, wäre beispielsweise eine Möglichkeit diese Art der Evaluation direkt für eine optimale Methode zu nutzen.

In meiner Arbeit beschreibe ich systematisch den Nutzen und die Möglichkeiten der Generierung von künstlichen Ausreißern und leiste damit einen signifikanten Beitrag für die Erkennung echter Ausreißer. Beispielsweise ist es dank meiner Arbeit möglich, die Evaluation von Methoden zur Erkennung von Ausreißern, differenziert in Bezug auf Ausreißer mir bestimmten Typen durchzuführen. Mit der entwickelten Evaluationsmethodik können existierende Methoden zur Erkennung von Ausreißern entsprechend verglichen werden und mögliche Anhaltspunkte für Verbesserungspotential der Methoden identifiziert werden.

Abstract

Outlier detection is an essential part of data science — an area with increasing relevance in a plethora of domains. While there already exist numerous approaches for the detection of outliers, some significant challenges remain relevant. Two prominent such challenges are that outliers are rare and not precisely defined. They both have serious consequences, especially on the calibration and evaluation of detection methods. This thesis is concerned with a possible way of dealing with these challenges: the generation of outliers. It discusses existing techniques for generating outliers but specifically also their use in tackling the mentioned challenges. In the literature, the topic of outlier generation seems to have only little general structure so far — despite that many techniques were already proposed. Thus, the first contribution of this thesis is a unified and crisp description of the state-of-the-art in outlier generation and their usages. Given the variety of characteristics of the generated outliers and the variety of methods designed for the detection of real outliers, it becomes apparent that a comparison of detection performance should be more distinctive than state-of-the-art comparisons are. Such a distinctive comparison is tackled in the second central contribution of this thesis: a general process for the distinctive evaluation of outlier detection methods with generated data. The process developed in this thesis uses entirely artificial data in which the inliers are realistic representations of some real-world data and the outliers deviations from these inliers with specific characteristics. The realness of the inliers allows the generalization of performance evaluations to many other data domains. The carefully designed generation techniques for outliers allow insights on the effect of the characteristics of outliers. So-called hidden outliers represent a special type of outliers: they also depend on a set of selections of data attributes, i.e., a set of subspaces. Hidden outliers are only detectable in a particular set of subspaces. In the subspaces they are hidden from, they are not detectable. For outlier detection methods that make use of subspaces, hidden outliers are a blind-spot: if they hide from the subspaces, searched for outliers. Thus, hidden outliers are exciting to study, for the evaluation of detection methods that use subspaces in particular. The third central contribution of this thesis is a technique for the generation of hidden outliers. An analysis of the characteristics of such instances is featured as well. First, the concept of hidden outliers is broached theoretical for this analysis. Then the developed technique is also used to validate the theoretical findings in more realistic contexts. For example, to show that hidden outliers could appear in many real-world data sets. All in all, this dissertation gives the field of outlier generation needed structure and shows their usefulness in tackling prominent challenges of the outlier detection problem.

Acknowledgements

First, I want to thank my supervisor Prof. Dr.-Ing. Klemens Böhm. He did enable me to conduct my research in the first place, and the discussions with him have always helped me to improve the particular piece of work tremendously. This interaction has also improved my overall skill set in countless facets.

Next, I want to thank Prof. Dr. Jörg Sander: for being the second reviewer of my thesis and supporting my fantastic time at his research institute at the University of Alberta. In this regard, I want to thank the Karlsruhe House of Young Scientists (KHYS) as well: they have sponsored my research stay at the University of Alberta through their research travel grant.

I also want to thank every student I had the pleasure of supervising. Be it during their theses, their coursework, or as a student assistant. In particular, my thank goes to Lena Witterauf, Rudolf Biczok, and Leander Kurscheidt for helping me advance my research tremendously.

My next thank goes to all my colleagues from Prof. Dr.-Ing. Klemens Böhms chair. Discussions with you — be it for my own or your research — have always meant a great deal for me and my research. The time I spent with you has always been one of the highlights of my day. In particular, my thank goes to Fabian Laforet, Vadim Arzamasov, Micheal Vollmer, Jens Wilkomm, Holger Trittebach, Adrian Englhardt, Edouard Fouché, Jakob Bach, Natalia Arzamasova, and Elaheh Ordoni. I want to thank Barbara Breitenstein and Bettina Wagner from the chair for granting all of us with much support in the tangles of bureaucracy and enlightening our days.

My thank also goes to my friends and family for making me the person I am today and always supporting me. Especially my parents had encouraged me to pursue my scientific carrier from when I was young and always supported this career in every possible way.

Last but not least, I want to thank Magdalena, for just being there and sailing successfully with me trough sometimes stressful times.

Contents

Acronyms

Notation	Description
kNN	k-Nearest Neighbour Detection 83, 84
ABOD	Angle-Based Outlier Detection 63
ANOVA	Analysis of Variance 37–40, 42–44
AUC PR	Area Under the Precision Recall Curve 86, 87
BIC	Bayesian Information Criterion 81
C	Classify 79, 87
DBOut	DB(p, k)-Outlier 60, 62, 65, 66, 72
DF	Degrees of Freedom 38, 40, 55, 93, 94
FastABOD	Fast version of Angle-Based Outlier Detection 63–66, 71, 72
GAN	Generative Adversarial Network 15, 16, 21, 30, 37
KDE	Kernel Density Estimation 81, 83, 84, 88
LOF	Local Outlier Factor 8, 63, 78, 82–84
LoOP	Local Outlier Probabilities 63–67, 72
MCC	Matthews Correlation Coefficient 34, 37–46
MVN	Multivariate Normal Distribution 55, 56, 63, 65, 66, 69, 93, 94
NS	Negative Selection 28
OD	Overall Density 79, 80, 87, 88
PCA	Principal Component Analysis 69–71
PC	Principal Component 69
RD	Inlier Density 79, 87, 88
RNS	Real-Valued Negative Selection 28
ROC	Receiver Operating Characteristic 86
SVDD	Support Vector Data Description 10
SVM	Support Vector Machine 10, 24, 25, 35, 36, 46, 85
wkNN	Weighted k-Nearest Neighbour Detection 86

List of Symbols

Notation	Description
$\underline{N}$	Set of the nearest neighbors of an instance 24, 25, 29
$\underline{\Sigma}$	Covariance matrix of a multivariate random variable 22, 56, 81, 93–96
dens_{in}	Density of an instance with regard to the distribution of inliers 79, 80, 87
dens_{out}	Density of an instance with regard to the distribution of outliers 79, 80, 87
ε^*	Optimal parameter for generating hidden outliers 60–62, 66, 67
m_{in}	Generative model of inliers 80
m_{out}	Generative model of outliers 80
gen_{test}	Represents an outlier generation technique for testing 37, 39–42, 44
gen_{train}	Represents an outlier generation technique for training 37, 39–44, 46
d'	Number of data attributes in a subspace 51, 56, 64
$\mathbb{R}^d$	Instance space — a d-dimensional real-valued vector space 4, 14, 19, 54, 81
$\mathcal{R}_{in}$	Region only with inliers 52–54, 56, 96, 97
$\mathcal{R}_{hidden}$	Region of hidden outliers 54, 56, 57, 59, 60, 62, 67, 68, 72
$\mathcal{R}_{out}$	Region only with outliers 52–54, 56
$\mathcal{R}_{full}$	Region the whole data lies in (usually $[l, u]^d$) 4, 51–54, 56–59, 96
$\mathcal{SC}_{in}$	Set of subspaces a hidden outlier is inlier in 53, 54, 59, 61–67, 69, 71
$\mathcal{SC}_{out}$	Set of subspaces of that a hidden outlier is outlier in at least one 53, 54, 59, 61–66, 69
$\mathbb{E}$	Expectation of a random variable 21
$\mathbb{1}$	Indicator function 2, 21, 25, 59
μ	Expected value of a random variable xviii, 20–22, 38, 81, 93, 94

Notation	Description
n_{hidden}	Number of hidden outliers 61
n_{art}	Number of artificial instances 14–17, 19–24, 26, 27, 29, 31, 32, 36, 37, 59–62
n_{out}	Number of outliers 2
n_{genu}	Number of genuine instances 3, 18, 20, 22, 25, 36, 37, 39, 44–46, 58, 59, 68, 78, 80
σ	Standard deviation of a random variable 20, 21, 23, 56, 93, 96
$\mathbb{R}$	Set of real numbers xvii, 3, 4, 14, 19, 21, 51, 54, 78, 79, 81
ε	Parameter for generating hidden outliers xvii, 23, 57–62, 66, 67, 71, 72
$\vec{\mu}$	Expected value of a multivariate random variable 22, 93, 94
a	Value of one attribute of an artificial instance xix, 14, 16, 21, 22, 29–32, 56–61
c	Single classifer 31, 34, 37–41
$h_{1\|d}$	Setting in which outliers are hidden from the full space 52, 54–57, 60, 63, 65–67
$h_{\mathcal{F}}$	Setting in which outliers are hidden in the full space 52, 54–57, 63, 65–67
h	Value of one attribute of a hidden outlier xix, 61, 94, 95
l	Lower bound of an attribute xvii, 4, 19, 23, 51, 52, 55, 57–61, 63
m	Generative model xvii, 78–80
r	Random value xix, 22, 23, 27
t	Size of a subspace collection 51
u	Upper bound of an attribute xvii, 4, 19, 23, 51, 52, 55, 57–61, 63
x	Value of one attribute of a data instance xix, 3, 13, 14, 20–25, 27, 29, 31, 32, 51, 52, 55, 57–60, 81, 93, 94
y	Label of a data instance 13, 14
dect	Function that represents outlier detection 4, 32, 33, 52–56, 61, 62, 64, 66, 67, 78, 80
dens	Function that returns the probability density of instances xvii, 21, 58–60, 78–80, 87
diag	Function that returns diagonal of a matrix 56
fit	Function that fits a generative model to data 78, 80–82

Notation	**Description**
gauss	Function that transforms disagreement into a probability 31, 32
gen	Function that generates instances xvii, 21, 34, 35, 37, 39–44, 46, 78, 80
inter	Function that represents interaction between factors in an ANOVA 37, 40
margin	Function that computes the disagreement among classifiers 31, 32
match	Function that determines how well two detectors match 29
mdist	Function that computes Mahalanobis distance 55, 65–67, 71, 72, 93–95
modify	Function that modifies a generative model 78, 80–82
p	Function that is either probability mass or density function 13, 14
quant	Function that returns the quantile of a distribution 55, 60, 67, 93–96
surr	Function that returns surrounding region of an instance 58–61
vol	Function that returns volume of a region 54, 56, 59, 60, 67, 68, 72, 96, 97
$\vec{a}$	Artificial instance 14, 16, 21, 22, 29–32, 56–61
$\vec{h}$	Hidden outlier 61, 94, 95
$\vec{x}$	Data instance 3, 13, 14, 20, 22–25, 27, 29, 31, 32, 51, 52, 55, 57–60, 93, 94
$\vec{r}$	Vector of random values 22
$\vec{n}$	One of the nearest neighbors of an instance 24, 25, 29
$\vec{o}$	Outlier 22, 23, 25, 27, 53
$\mathcal{C}$	Set of classifiers 31, 32, 34
$\mathcal{F}$	Set of all data attributes — the fullspace xviii, 51–58, 63–67, 93–95
$\mathcal{G}$	Set of generation techniques 34
$\mathcal{M}$	Set of all generative models 78
$\mathcal{R}$	Region — a subset of the instance space xvii, 4, 51–54, 56–60, 62, 67, 68, 72, 96, 97
$\mathcal{SC}$	Set of subspaces xvii, 51–56, 59, 61–67, 69, 71, 94, 95
$\mathcal{S}$	Set of data attributes — a subspace 51–56, 61–64, 66, 67, 94–97

Notation	Description
d	Number of data attributes xvii, xviii, 3, 4, 14, 15, 19, 20, 23, 26, 27, 35, 36, 39, 44–46, 51, 52, 54–57, 60, 63–68, 78, 81, 95, 96
Var	Variance of a random variable 2
$\mathbf{I}$	Distribution of inliers 4, 17, 18, 26
$\mathbf{Y}$	Distribution of the label of instances 14
$\mathbf{O}$	Distribution of outliers 4, 16, 17, 19–21, 26
$\underline{O}_{art}$	Set of artificial outliers 22, 39, 42–44
$\underline{A}$	Set of artificial instances i, 4, 16, 17, 29, 31, 60, 61
$\underline{D}$	Set of instances that are an experimental design 14
$\underline{Z}$	Real-world data set augmented with artificial instances 14, 32, 36
$\underline{O}_{genu}$	Set of genuine outliers 4, 34, 39, 42, 44
$\underline{I}_{test}$	Set of inliers used for testing 34
$\underline{I}_{train}$	Set of inliers used for training 34
$\underline{I}$	Set of inliers i, 4, 34, 80
$\underline{A}_{inter}$	Set of interesting artificial instances 16, 17
$\underline{O}$	Set of outliers i, 4, 22, 34, 39, 42–44
$\underline{X}$	Real-world data set i, 3, 13, 14, 17–20, 23–27, 29, 31, 32, 34–37, 39, 40, 44, 45, 53, 55, 57, 59–61, 71, 80
$\underline{S}$	Synthetic data set 4, 80

1 Introduction

Data science is a rather novel paradigm with increasing relevance not only for researchers but also for industry. In a nutshell, the task of solving problems using data could describe data science (Carmichael and Marron, 2018). The broad range of problems that can or might be solved by the help of data science has undoubtedly spurred the vast development in this area of expertise. For example, the advances in artificial intelligence by the use of deep neural networks (Arel et al., 2010) — in essence, a data science method. Data science can be categorized by a few rather specific areas (Carmichael and Marron, 2018). Two very prominent ones for the topic of this dissertation are (1) data gathering, preparation, and exploration, and (2) data modeling. We will connect these two areas of data science to the scope of this dissertation in the following.

Overall this thesis is concerned with the detection of *outliers* (Aggarwal, 2017; Hawkins, 1980) often also called anomalies or novelties (Hodge and Austin, 2004). A common description of outliers was given by Hawkins (1980):

"An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

Figure 1.1 illustrates such data instances. It displays a histogram of the duration of eruptions from the Old Faithful geyser in the Yellowstone national park along with the actual data instances (objects below the histogram)[1]. The triangle represents a single eruption that had an extremely short and thus unusual duration (< 1 minute). Clearly, it qualifies as deviating *"so much [...] to arouse suspicions that it was generated by a different mechanism"* (Hawkins, 1980).

The Faithful geyser data is merely an illustration. However, almost all applications of data science are prone to outliers. They could result from a recording error or observations of a different process that resulted from a change in the monitored system. A recording error might occur due to misspellings when saving data to a database or invalid data that comes from a defect sensor. Such errors can almost always happen and affect the resulting analysis (see (Genschel, 2018), for example). In many cases, outliers can result from a change in the monitored system and are of great interest to detect (Singh and Upadhyaya, 2012): For example, in fraud detection, intrusion detection, or fault/damage detection. In each such application, instances from the class of interest (e.g., a faulty part) behave differently to most of the other data instances. In other words, a different mechanism generates instances from the class of interest. Having these two origins of outliers, the firm foundation of outliers in data science becomes apparent. Outliers that resulted from recording errors are an essential part of the data gathering, preparation, and exploration part. In the initial phase of any data analysis, such outliers should be

[1] Data available at https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/geyser.html

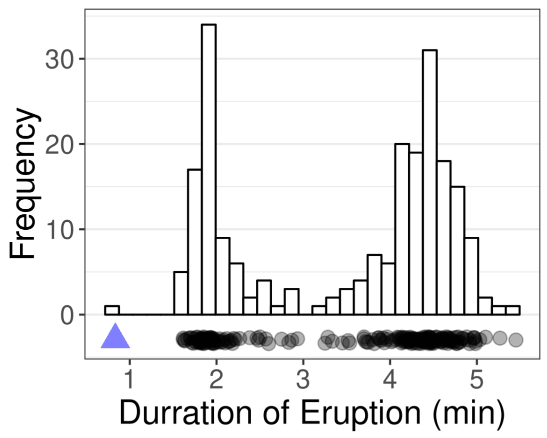


Figure 1.1: Outlier in the Faithful geyser data.

detected and possibly deleted. Outliers like the faulty parts — that are of great interest for the domain expert — should be part of the data model. For example, by the inclusion of an outlier detection method such that it detects if any part wears down. Note that there is no precise distinction between recording errors and unusual observations of interest. A defect sensor could be both, for example.

Of particular interest in this thesis are the calibration and evaluation of methods to detect outliers. To “calibrate” a detection method, means to tune the method for a specific setting by adjusting its parameters. The “evaluation” refers to a rating of the performance of detection methods. For example, this rating can base on a comparison of the detection performance to the performance of other detection methods or a particular ideal detection. For both calibration and evaluation, a performance measure in terms of outlier detection is crucial. In the literature, the primary usage of artificial outliers is the calibration of outlier detection methods. The evaluation of outlier detection methods through artificial outliers is something rather novel we propose in his thesis.

1.1 Central Challenges

The topic of outlier detection is far from solved, and many challenges remain. The following illustrates two central challenges in outlier detection. Both challenges are relevant, especially for the calibration and evaluation of methods to detect outliers since they affect any measure of detection performance.

1.1.1 There is Few Outliers

Outliers are rare (Emmott et al., 2015, 2013; Campos et al., 2016). If at all, there are only a few examples of the outliers in a specific data set. For the applications mentioned above, that is also the case: Parts of any system should rarely be faulty, or fraud should rarely happen. This scarcity of outliers poses great challenges for data science methods (Visa and Ralescu, 2005; Tax and Duin, 2001; Wang et al., 2018). Example 1 illustrates why the calibration and evaluation of outlier detection methods are affected by this scarcity in particular.

Example 1 *Let there be n_{out} outliers $\underline{O}$ and a detection method that successfully detects an outlier with some unknown probability p. An estimate for the performance of the detection method might be p itself which would be estimated by*[2]

$$\widehat{p} = \frac{1}{n_{out}} \sum_{\vec{o} \in \underline{O}} \mathbb{1}_{\{\vec{o} \text{ is detected as outlier}\}}. \tag{1.1}$$

The variance of $\widehat{p}$ then is

$$\mathrm{Var}(\widehat{p}) = \frac{p(1-p)}{n_{out}}. \tag{1.2}$$

Thus, the variance of $\widehat{p}$ increases with decreasing n_{out}.

The essential takeaway from Example 1 is that with fewer outliers (n_{out}), the variance of the estimate for the performance of the detection method is higher. Thus, the estimate that tells us if the detection method performs well is much less reliable with few outliers. In terms of evaluation and calibration, this poses a severe problem. A detection method that turned out to be the best using these few instances might not be a suitable detection method with other outliers. The same holds when calibrating detection methods using the few outliers.

1.1.2 Their Notion is Imprecise

Another issue with outliers is that they are not precisely defined (Zimek and Filzmoser, 2018). The description given by Hawkins (1980) already gives some leeway. For example, it does not specify how different instances must be to qualify as an outlier. Think again of the outlier in Figure 1.1: How much longer should the eruption last, such that the eruption is no longer outlying? Besides, there exist multiple *types of outliers* characterized by specific properties. For example, a common distinction is made between global and local outliers (Breunig et al., 2000; Schubert et al., 2014; Campos et al., 2016) illustrated in Example 2.

Example 2 *The time preceding an eruption is the waiting time of this eruption. Figure 1.2 displays the combination of waiting time and duration of an eruption for the Faithful geyser. The triangle from Figure 1.1 is clearly an outlier in this scatter plot as well. There is also another outlier that distinguishes itself with an incredibly long-lasting waiting time (> 100 minutes). Since they both stand out in comparison to all other data instances, they usually are characterized as* global outliers. Local outliers, *in contrast, stand out in their local neighborhood. The eruption marked by a blue cross, for instance, raises the suspicion to be different from the eruptions on its right with a slightly longer-lasting eruption but similar waiting time. However, from a global perspective, the eruption marked by the blue cross is rather close to the other instances.*

[2] For this we assume that the detection of one outlier is independent of the detection of other outliers.

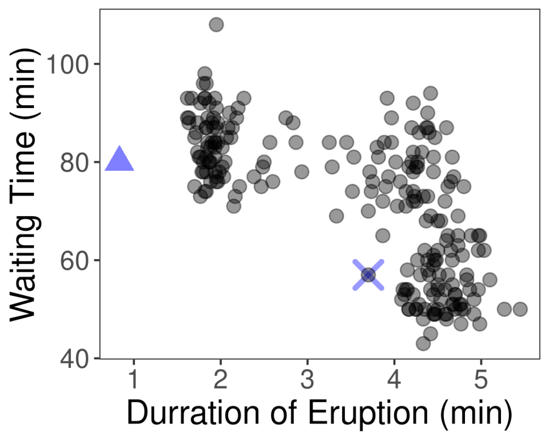


Figure 1.2: Scatter plot of the Faithful geyser data.

All this highlights the imprecise and somewhat subjective notion of what is and is not an outlier. This imprecise notion is also a very central issue, specifically with the estimation of detection performance. For many predictive data science methods, performance evaluation takes place on some annotated data (Hastie et al., 2009). In annotated data, the class membership (e.g., inlier or outlier) of instances is known. This holds in particular for outlier detection (Campos et al., 2016; Emmott et al., 2015, 2013; Goldstein and Uchida, 2016; Domingues et al., 2018). Clearly, for outliers of a different type, other methods for outlier detection perform well. However, a single annotation of instances can not cope with the imprecise notion of outliers and their types. We are also not aware of any data set with annotations concerning locality or any other specific type of outliers. We annotated the outliers mentioned in Example 2 with our subjective interpretation of outliers. Two geologists might come to different annotations — compared to ours but maybe also among themselves. A consequence of this is that the performance evaluation of outlier detection methods is difficult. In particular, in terms of outliers of different types.

Artificial outliers are generated such that they are outliers in terms of a set of inliers and represent one way to deal with the two mentioned challenges. Two properties of generated outliers are particularly beneficial in terms of the challenges described so far. They can usually be generated in any desired amount and thus are not rare. Since the generating procedure is known and can be adapted, generated outliers have quite distinct and controllable properties. Thus, generated outliers counteract the challenges described earlier.

1.2 Central Contributions

In the literature, there exists a vast amount of different techniques to generate outliers. However, it is not always clear what properties they share or in what aspects they might contradict each other. These properties might be algorithmic or refer to the characteristics of the generated outliers. For example, the technique proposed in (Tax and Duin, 2001) tries to generate outliers rather close to inliers. The technique proposed in (Pham et al., 2014) tries to generate outliers very far from inliers. Not always, there is a discussion of the relevance of such central characteristics of the generated outliers. This is the basis for Contribution 1 that is given mostly in Chapter 3[3].

Contribution 1 *This thesis gives a unified description of existing techniques to generate outliers and their usages.*

The central idea to most of the proposed techniques for generating outliers is to find proper parameters for the detection of genuine (i.e., not generated) outliers. This is, calibrating a detection method. Such a calibration should work reasonably well if the types of genuine outliers are at least roughly known. However, the focus of outlier detection in the data gathering, preparation, and exploration part of data science is to identify any instance that might be an outlier. Especially in this stage, it is usually not known what types of outliers might be present in the data. One way to approach this could be to apply a large selection of different state-of-the-art detection methods. This approach might enable the detection of any type of outlier. However, varying the parameters of a single detection method can already result in a massive set of different outlier detection results. Investigating each such result is time-consuming and difficult. A better approach would be to apply a rather small suite of detection methods that can detect outliers from a broad spectrum of different types well. For creating this suite of detection methods, it is essential to be able to investigate the performance of detection methods with diverse types of outliers. This results in Contribution 2 that is described in Chapter 4[4].

Contribution 2 *We propose a process that uses generated data to allow for the evaluation of outlier detection methods in terms of different outlier types.*

The outlier types mentioned so far, always related to a fixed set of data attributes. With so called "hidden outliers" (Steinbuss and Böhm, 2017) this is not the case. Hidden outliers are outliers only in specific subsets of the data attributes, commonly referred to as subspaces (Keller et al., 2012; Trittenbach and Böhm, 2019; Müller et al., 2011; Kriegel et al., 2009b). For example, the local outlier described in Example 2 does only deviate from other instances when observing both data attributes. When observing the duration of eruption or the waiting time in isolation, the outlier is close to other instances and not detectable as an outlier. Put differently, in terms of the one-dimensional views of the data the outlier is *hidden*. Hidden outliers represent a special type of outliers that is somewhat orthogonal to the types described so far: Hidden outliers can be of any such type in any of the subspaces. An essential property of hidden outliers is that they can be the blind spot of detection methods that make use of subspace — so-called Subspace Search Outlier Detection (SSOD). Hidden outliers are usefull to evaluate SSOD methods in particular. To illustrate, if a certain SSOD method does not allow for any hidden outliers, it probably has a high detection performance. However, there is not much knowledge of the circumstances that back the existence of hidden outliers. Thus, Contribution 3 that is given in Chapter 5[5] addresses hidden outliers specifically.

Contribution 3 *We analyze the concept of hidden outliers and develop an algorithm to generate them.*

[3] Published version: (Steinbuss and Böhm, 2020a)
[4] Published version: (Steinbuss and Böhm, 2020b)
[5] Published version: (Steinbuss and Böhm, 2017)

1.3 Fundamentals

This section reviews fundamental concepts for this thesis. The first part discusses the notion of objects (like outliers or inliers) we use throughout this thesis. Some additional notation is specific to a particular chapter and hence, introduced when needed (e.g., hidden outliers and subspaces). The second part of this section focuses on outlier detection itself.

1.3.1 Notion of Common Objects

Common types of objects in this dissertation are matrices, vectors, distributions, functions, and general sets. Within each type, the notation is similar. This will be explained and illustrated in the following.

A *matrix* is a collection of some real-valued numbers arranged ($\mathbb{R}$) in the form of a fixed number of rows and columns. The most common object of this type in this dissertation is a given real-world data set $\underline{X} \in \mathbb{R}^{n_{genu} \times d}$. Each row in $\underline{X}$ is an instance, and each column called an attribute. The number of instances in $\underline{X}$ is denoted n_{genu} and the number of attributes (the columns of $\underline{X}$) by d.

A *vector* is a collection of some real-valued numbers arranged in the form of a fixed number of attributes. For example, the instances in $\underline{X}$ denoted $\vec{x}$. The value of the ith attribute of an instance $\vec{x}$ is $\vec{x}^{(i)}$. i as a subscript refers to the ith instance from a set. For example, $\vec{x}_i$ is the ith instance from $\underline{X}$. This notation generalizes to other objects (like the distributions or general sets).

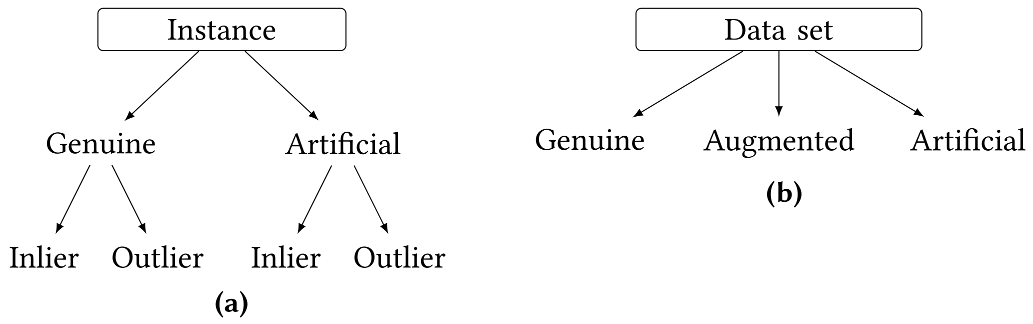


Figure 1.3: Terminology in respect to genuine or generated data.[6]

The most common objects of type vector are displayed in Figure 1.3a and of type matrix in Figure 1.3b. The real-world data set and the enclosed instances are genuine (i.e., not generated). Generated instances are referred to by "artificial". Any instance can either be an inlier or outlier. A corresponding set of instances (i.e., a matrix) is denoted $\underline{I}$ or $\underline{O}$. If the set refers explicitly to, say, genuine outliers, a corresponding subscript is added: $\underline{O}_{genu}$.

In terms of a data set (i.e., a matrix), there are three distinctions: genuine, augmented, and artificial. Genuine and artificial have the same meaning as with the instances. For example, the real world data set $\underline{X}$ is genuine. Augmented data is a mixture: genuine instances in combination with some artificial ones. For example, when adding outliers

[6] Adopted from (Steinbuss and Böhm, 2020a).

to real data that has only inliers. Note that in the literature, the synonym “synthetic” is rather frequent when referring to artificial data sets. However, for consistency, we usually use “artificial”.

A *distribution* describes the probability of values of a random variable. For example, the distribution of inliers **I** describes the probability of instances being an inlier or the one of outlier **O** of being an outlier.

There are many *functions* in this thesis. Usually, we denote these functions by plain lowercase letters. The most common function is an outlier detection function “dect”. It outputs either a binary signal stating if an instance is an outlier or not (Chapter 5), or a score that indicates how outlying an instance is (Chapter 4).

A *general set* is usually denoted by calligraphic letters. For example a region $\mathcal{R}$, which is a subset of the whole data region $\mathcal{R}_{full}$. The *whole data region* itself is the set of all feasible instances. In Chapter 3 $\mathcal{R}_{full} = \mathbb{R}^d$ or in Chapter 5 $\mathcal{R}_{full} = [l, u]^d$ for example.

1.3.2 Outlier Detection in This Thesis

This section reviews the central idea of outlier detection for this thesis. First, we describe the data that is the basis for outlier detection. Then we will shortly describe central concepts for outlier detection methods with such data.

The Data for Outlier Detection

A central identity in data — especially regarding performance — is the so-called “label of an instance”. The label represents the target of interest. To illustrate, classification methods partition a set of given instances into classes (Hastie et al., 2009). For example, classifying emails into “spam” and “no spam”. Then the label is a categorical variable y with $y \in \{\text{spam}, \text{no spam}\}$. With outlier detection, one usually has $y \in \{\text{inlier}, \text{outlier}\}$. The methods and techniques we design and discuss in this dissertation assume that there are no labeled examples of outliers. This is, they base on data that has either no labels $y \in \{\text{inlier}, \text{outlier}\}$ at all or that has no instance that has the label “outlier”. We deem these the most vital scenarios since outliers are rare.

Nonetheless, we will also make use of existing benchmark data for outlier detection that does come with labeled instances from both classes. Usually, this data is from the study by Campos et al. (2016). The reason for this is the validation or demonstration of the discussed methods and techniques. Note that the label in this data does not allow inference on the type of the outliers. This is, instances labeled “outlier” in the data from (Campos et al., 2016) could be of any type.

Taxonomy of Detection Methods

The usage of labels allows for the categorization of outlier detection into three areas (Hastie et al., 2009; Chapelle et al., 2010): supervised, unsupervised and semi-supervised. In terms of supervised outlier detection, we further differentiate between binary classification and One-Class Classification (OCC). This categorization is visualized in Figure 1.4.

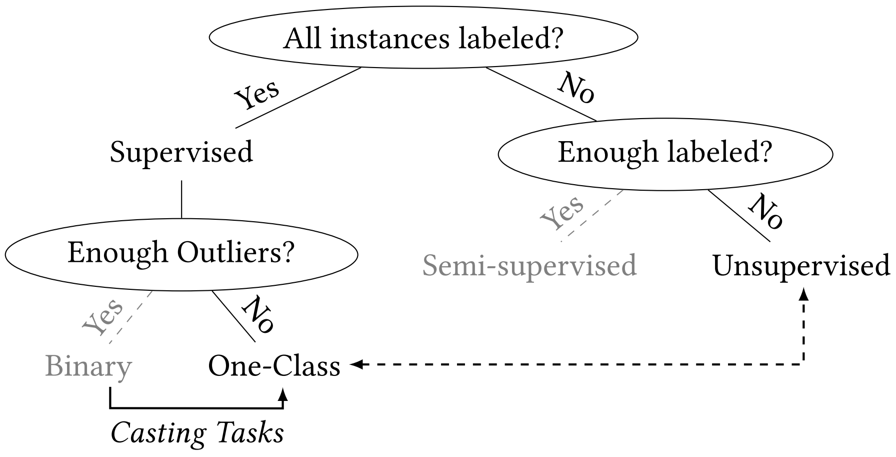


Figure 1.4: A taxonomy of outlier detection based on the availability of labels.

The methods that require no data with labeled outliers are either unsupervised or from OCC. Unsupervised methods assume that there are no labels for instances, while methods from OCC assume that the available instances are inliers. However, methods designed for OCC are usable also when there is no label available at all (i.e., the unsupervised case). See for example (Theiler and Michael Cai, 2003). The other way around, detection methods designed for the unsupervised case can also be used for OCC (Swersky et al., 2016).

Binary classification methods and methods for semi-supervised learning assume that there are some labeled outliers. Thus, such methods are not of direct interest in this thesis. However, binary classification remains important: In the literature the *casting-task* use case for artificial outliers (cf. Section 3.7) is common. This use case enables the usage of any binary classification method for OCC.

2 Related Work

The related work in this thesis has five parts. First, we review benchmarks for outlier detection — the usual way for evaluating outlier detection methods. We do not discuss related work for the calibration of outlier detection methods since we do not propose this methodology in this thesis but only summarize what others have contributed to this field. Thus, we leave a systematic review of alternative ways for calibrating outlier detection methods to the respective literature (see, e.g., (Wang et al., 2018)). The second part deals with data generation in general. The third part describes existing techniques to generate outliers without any genuine instances. In contrast, the fourth part discusses the generation of additional instances — not necessarily outliers — in terms of some genuine ones. Generating outliers in terms of a set of genuine instances is somewhat in-between parts three and four. Since the vast majority of techniques for generating outliers are of this kind, these techniques are the focus of Chapter 3 and not reviewed here. The fifth part of this section deals with related work specifically for hidden outliers and subspace search. In other words, that part is in particular relevant for our contribution in Chapter 5.[1]

2.1 Benchmarks for Outlier Detection

Benchmarks are large scale comparisons of the performance of different outlier detection techniques. They are essential for this thesis since they represent the state-of-the-art in evaluating outlier detection methods, and they summarize most of the available data sets with annotated outliers.

In terms of evaluation, the ideal outlier detection benchmark data set should fulfill the following requirements.

1. All instances in the data should have been assigned to the class of inliers or outliers.
2. The outliers should have characteristics that make them suite the meaning of outliers.
3. These characteristics should be known precisely.

The first requirement is vital for the performance measures themselves. For example, to identify the share of outliers that were detected by a method. The next two requirements

[1] The remainder of this chapter (except Sections 2.4 and 2.5) is an extraction and adaptation from (Steinbuss and Böhm, 2020b), which is submitted for review. Adjustments are to ensure consistency for this dissertation.

aim at the interpretation of the detection performance results, especially in terms of outliers of different types. In the following, we describe how well the current state-of-the-art in outlier detection benchmarks copes with these requirements.

Several benchmarks especially for unsupervised outlier detection exist (Domingues et al., 2018; Goldstein and Uchida, 2016; Campos et al., 2016; Emmott et al., 2013, 2015). They all use real-world data, usually from classification. One of the classes in the data set is selected and defined as the "outlier" class. Then this class is downsampled since outliers are rare. The class defined as the "outlier" class is usually selected based on its semantic meaning. For example, the patients that have a disease in data that records patients with and without the disease. While this addresses the first requirement for benchmark data given above, there remain at least two issues: (1) a semantic meaning does not necessarily translate to the given data representation, and (2) it does not specify at all of what type the outliers might be. Example 3[2] illustrates the mentioned issues.

Example 3 *Think of data in which one class represents patients that suffer from a brain tumor, and the other class represents patients without a tumor. Let us assume that the height of each patient is the only attribute given to us. Although the patients that suffer from a tumor are semantically meaningful as outliers, this does not translate to the height of the patients: regarding the height, there should be no difference. Even if we have attributes in which the two classes of patients are separable, it is still unclear how the two classes deviate from each other. Let us assume, we have a representation of images of the whole brain that — if applicable — includes the tumor of patients. In such data, the patients with tumors might be local outliers since, in large parts, the brains should be similar. If we have data about the overall well-being of the patients, on the other hand, the patients with tumors could be global outliers.*

Thus, with the described methodology, the second and third requirement is not directly addressed. Not addressing the requirements renders the interpretation of the results from a benchmark difficult or even impossible, concerning outliers of different types in particular.

In (Emmott et al., 2013, 2015), the issues just described are addressed to some extent through introducing four problem dimensions. These problem dimensions should result in a more systematic benchmark with real-world data. Sampling specific instances from the real-world data, allows varying the four dimensions within a data set. The more controlled versions of the data resulting from this are then useful to benchmark outlier detection methods. The four dimension are called "point difficulty", "relative frequency", "semantic variation" and "feature relevance". Relative frequency is the characteristic of outliers, that is addressed in other benchmarks as well: there should be only very few. Point difficulty measures how distant an outlier is to inliers. The rationale is that an outlier with greater distance is less difficult to detect. Semantic variation refers to the degree to which the outliers are spread across the instance space and not clustered. Feature importance addresses the issue of attributes in which outliers are not outlying the inliers. Clearly sampling instances according to the dimensions introduced in (Emmott et al.,

[2] A conversation with Prof. Dr. Jörg Sander inspired the example .

2013, 2015) does improve on the issues illustrated with Example 3. With carefully crafting appropriate problem dimensions, the methodology might even allow for an evaluation in terms of some different types of outliers. Thus, this methodology fulfills also the second requirement. However, the whole method crucially depends on the approaches to measure the dimensions in real-world data. These approaches are approximations themselves which can interfere with the interpretation of the results. For example, to measure point difficulty a binary classifier is proposed (Emmott et al., 2013, 2015). The rationale is that an outlier that has a high probability of being from the "outlier" class for this classifier is consequently easy to detect. The other way around, an outlier with a low probability is difficult to detect. However, the classifier gives only an approximation of this probability and thus might be wrong. Hence, this methodology does not fulfill the third requirement given above. In turn, the entirely artificial data we propose for the evaluation of outlier detection methods in Chapter 4, comes with precisely known characteristics of inliers and outliers. Thus, there is no estimation uncertainty in the computed ground truth.

There is also a benchmark for OCC (Swersky et al., 2016). The concept is similar to the benchmarks for unsupervised outlier detection, like (Campos et al., 2016). Two significant differences are that there is no downsampling of the class that represents outliers and that the assignment of classes in "outliers" and "inliers" is varied. This variation means that for the same data set a class might the "inlier" class in one experiment and the "outlier" class in another experiment. Thus, the issues illustrated earlier apply here as well.

2.2 Artificial Data Sets

Data generation, in general, is of high relevance for this thesis. There is much literature on generating a data set entirely, but here we focus on ones that are relevant for outlier detection. For broader reviews see (Zimmermann, 2019; McLachlan et al., 2019; Frasch et al., 2011; Steinley and Henson, 2005). First, we review domain-specific data generators. Then we review generators developed without a specific domain in mind. Lastly, we review generation approaches that use real data as a basis.

2.2.1 Domain Specific

Many artificial data sets are specific to a domain. This is, their generation process is designed with a certain application in mind, like chemical processes (Downs and Vogel, 1993), fraud detection (Barse et al., 2003) or application scoring (Kennedy, 2011). Such artificial data is also already used in benchmarks for unsupervised outlier detection. For example, in the benchmark (Campos et al., 2016) the data set Waveform[3] is artificial. Such data can be useful when comparing outlier detection approaches. However, domain-specific artificial data is usually not generated for outlier detection. For instance, the Waveform data is generated to evaluate methods for classification. Hence, if the data is useful for the benchmark of outlier detection methods is unclear. Even if like (Downs and Vogel, 1993; Barse et al., 2003; Kennedy, 2011), such data has a class that is semantically meaningful as outliers, it is costly to generate such data: a domain expert must be heavily

[3] http://archive.ics.uci.edu/ml/datasets/waveform+database+generator+(version+2)

involved in its design. Novel frameworks for crafting realistic data like the one introduced in (Mannino and Abouzied, 2019) might reduce the effort by utilizing smart visualizations and useful suggestions for attribute values, distributions, and dependencies among them. However, an expert must still be involved. For a broad benchmark of detection methods, it is necessary to generate data from many data domains. This generation is difficult with domain-specific generators. Besides, the data generators we are aware of usually feature very different generation approaches. Hence, it is difficult to compare data from the different generators in a benchmark beyond what is possible with real-world data already.

2.2.2 Domain Agnostic

There also exist many artificial data generators not situated in a certain domain, most of them for clustering (Iglesias et al., 2019; Sánchez-Monedero et al., 2013; Melnykov et al., 2012; Maitra and Melnykov, 2010; Qiu and Joe, 2006; Pei and Zaıane, 2006; Steinley and Henson, 2005; Waller et al., 1999; Milligan, 1985) or classification (Frasch et al., 2011; Rachkovskij and Kussul, 1998). A major difference between the generators for clustered data is the control of overlap (Sánchez-Monedero et al., 2013; Melnykov et al., 2012; Maitra and Melnykov, 2010; Qiu and Joe, 2006; Steinley and Henson, 2005; Milligan, 1985). In a nutshell, this overlap is a measure of how well separated the different clusters are from each other. In other words, it gives the degree to which instances from clusters can be confused with each other. How to parameterize the generators — for example, for a broad benchmark of outlier detection algorithms — remains an open question. Next, the probability densities that we use to obtain ideal outlier detection scores are not always available. Thus, there might be only little insights available from such data. However, in many cases, the techniques to generate domain agnostic data are the same as the ones used to generate data from real data. For example, Gaussian mixtures are common in the generation of domain agnostic data (Sánchez-Monedero et al., 2013; Melnykov et al., 2012; Frasch et al., 2011; Maitra and Melnykov, 2010; Milligan, 1985) but as we will see in the next section for generating data from real data as well. This usage of similar generation techniques might be one reason that the separation between domain agnostic generators and ones that generate data from real data is not precise.

2.2.3 From Real Data Sets

"Synthetic reconstruction" aims at generating data that matches the given real data. This term is known from survey data (Wan et al., 2019) but the principle is common in other disciplines as well (Mannino and Abouzied, 2019; Sun et al., 2018; Fazekas and Kiss, 2018; Albuquerque et al., 2011; Waller et al., 1999). Synthetic reconstruction is also the methodology we follow when we generate data in Chapter 4.

Often synthetic reconstruction is needed due to a limited amount of data or due to privacy issues with the original data (McLachlan et al., 2019). Some in principle domain agnostic data generators mentioned earlier also allow for some adaptation to real-world data. For example, in (Albuquerque et al., 2011) and (Iglesias et al., 2019), data generators are proposed that can cope with user defined distributions. Another example is the use-case example featured in (Waller et al., 1999). In their use-case example, Waller et al. (1999)

adopt the parameters of their generation technique to real-world data. Fitting statistical distributions to generate realistic artificial data is common as well (Sun et al., 2018; Fazekas and Kiss, 2018; Rogers et al., 2003). For example, Gaussian mixtures are useful to model protein spots in images of electrophoresis gel (Rogers et al., 2003). Artificial data is then generated from this model and used for evaluation purposes. Approaches like the Generative Adversarial Network (GAN) (Goodfellow et al., 2014a) recently received much attention for their astonishing capabilities in generating realistic artificial data. However, they do not provide much ground truth for the generated data.

In Chapter 4 we focus on approaches like (Sun et al., 2018; Fazekas and Kiss, 2018; Rogers et al., 2003) that fit statistical distributions to real-world data and draw samples from these models for generating data. The reasons for this focus are manifold, and we will cover them in detail in Section 4.4.2. In a nutshell, statistical distributions are simple and powerful, but give us at the same time the ability to compute useful ground truth annotations for the generated data.

2.3 Generating Outliers Without Real Data

The review for generating data in Section 2.2 focused on the techniques that exist to generate a whole data set, possibly with some classes or clusters. Here we want to focus specifically on the aspect of generating outliers that is also featured by some techniques and approaches already reviewed. However, we focus on techniques that do not use any real-world data to generate outliers. Techniques that do use real-world data for generating artificial outliers are detailed in Chapter 3.

Two of the benchmarks for unsupervised outlier detection reviewed in Section 2.1 feature artificial data with outliers (Domingues et al., 2018; Emmott et al., 2015). In (Emmott et al., 2015), data is generated in that inliers follow a simple multivariate Gaussian distribution and in (Domingues et al., 2018), inliers follow two separate Student's t-distributions. In both cases, outliers come from a uniform distribution surrounding the regular instances. As stated in (Emmott et al., 2015), such simple data does necessarily increase the possible takeaways from a benchmark in comparison to real-world data.

Generators for clustering introduced in Section 2.2.2 also feature the generation of outliers, usually from a uniform distribution within the instance space (Iglesias et al., 2019; Melnykov et al., 2012; Maitra and Melnykov, 2010; Qiu and Joe, 2006; Pei and Zaıane, 2006).

In (Iglesias et al., 2019), outliers are generated not by sampling from a uniform distribution, but by using the intersections of a grid. While we deem the resulting distribution close to uniformity, this is difficult to verify, given the purely algorithmic description. Interesting is that the proposed technique for generating outliers allows for the straightforward generation of outliers that are detectable in a specific subset of attributes. This generation is close to the hidden outliers detailed in Chapter 5. However, the technique is not useful for generating hidden outliers in real data like the technique we propose in Chapter 5.

In (Pei and Zaıane, 2006) and (Milligan, 1985), techniques for the non-uniform generation of outliers are proposed as well. In (Pei and Zaıane, 2006), outliers are generated such that they follow pre-specified patterns — lines, for example. In (Milligan, 1985), outliers are generated by sampling from a Gaussian with increased variance in comparison to the Gaussian that inliers are samples of. The technique is interesting since it is the only one to generate local outliers. Thus, one can generate outliers of a specific type, which is, as motivated earlier, one of the major concerns of this thesis. The possibility of generating outliers of a specific type is the reason we will use this generation technique for one instantiation of the process for benchmarking unsupervised outlier detection described in Chapter 4.

2.4 Generating Other Additional Instances

There also are a few approaches that generate instances but not specifically outliers. "Synthetic oversampling", for example, is one way for improving the classification of imbalanced data (Krawczyk, 2016). In imbalanced data, one class is underrepresented. The idea in synthetic oversampling is to increase the sample size of the minority class by generating instances similar to the other instance from this class. When adding these generated instances to the existing data, the class is no longer underrepresented. The most common approach for this is the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002). In essence, SMOTE interpolates between the different existing instances from the minority class. Within this interpolation, the generation of new artificial instances takes place. There also exist multiple adaptations to this idea (Krawczyk, 2016). However, the idea of SMOTE and imbalanced classification requires at least a few labeled examples from the minority class.

[4]"Classifier evasion" is another field in which instances are generated and added to data. To illustrate classifier evasion, 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 mail. However, existing approaches to find such instances (Nelson et al., 2010; Xu et al., 2016) rely on at least one instance of spam email.

"Adversarial examples" (Szegedy et al., 2013) that were recently introduced in the neural network community follow the concept of classifier evasion. An adversarial example is an instance that is classified wrongly due to some small changes to it. The crafting of such adversarial examples is formalizable as an optimization problem (Szegedy et al., 2013). The optimization is on the amount of modification on an instance such that it is classified differently. The smallest modification is optimal. Several approaches for crafting adversarial examples focus on the case that the classifier they invade is a neural network (Goodfellow et al., 2014b; Papernot et al., 2016). We will discuss this in greater length and also set it into perspective with artificial outliers in Section 3.8.3.

"Protecting privacy" is another area in data analysis that relates to the topic of generating instances. The reason is that serval approaches for privacy protection add generated instances to the data. For example, in (Kido et al., 2005), an algorithm is proposed to add dummy instances to geolocation data of individuals in order to increase privacy. However, the objective in protecting privacy is different from the one of generating outliers:

[4] The remainder of this section is an extraction of the related work section in (Steinbuss and Böhm, 2017), previously published in the International Journal of Data Science and Analytics.

Privacy-protection approaches attempt to add data that behaves like the original data. Hence, the real data is hidden, while relevant information is still available.

2.5 Hidden Outliers and Subspace Search

[5]We are not aware of any comprehensive study of hidden outliers. However, in (Zimek et al., 2012) the notion of "masked" outliers is described. Masked means that irrelevant attributes within a data set can hide the outlier behavior to some extent.

Hidden outliers are, of course, related to the various methods for SSOD. In this regard, there exist subspace search schemes that integrate subspaces in the outlier detection method (Kriegel et al., 2009b; Müller et al., 2011) but also schemes that first identify subspaces independent of a downstream outlier detection (Lazarevic and Kumar, 2005; Keller et al., 2012; Trittenbach and Böhm, 2019). One goal — specifically in the second category of schemes — is to find the combination of subspaces that allow for the best detection of outliers overall: First, such schemes find a set of subspaces that are promising for detecting various outliers. Then each subspace is searched for outliers, and the results from all subspaces are combined. For example, High Contrast Subspaces (HiCS) is a well-known method to find subspace independent of the outlier detection method. First, HiCS finds a set of subspaces that each has a high contrast. This contrast is essentially a multivariate dependency measure (Fouché and Böhm, 2019). Then in each subspace, an outlier detection method is applied. The last step is to combine the outlier detection results (a scoring that indicates how probable an instance is an outlier) from each subspace. The maximum score across each subspace or the average of all scores is useful for this combination (Keller et al., 2012). Especially for the category of schemes that find subspace independent of the outlier detection method, hidden outliers represent a new artifact for evaluation purposes. If there can be many outliers hidden from a specific selection of subspaces, this selection is likely to perform not good at detecting various outliers in the data.

[5] This section is extracted and adapted from the related work section in (Steinbuss and Böhm, 2017), previously published in the International Journal of Data Science and Analytics.

3 Generating Outliers from Genuine Instances

Section 2.3 has featured a discussion about the few techniques for generating outliers without real data. In this chapter, we review the vast majority of generation techniques: ones that do rely on real data (i.e., genuine instances).[1] There exists a large body of literature on such generation techniques (Steinbuss and Böhm, 2017; Wang et al., 2018; Curry and Heywood, 2009; Gonzalez et al., 2002; Shi and Horvath, 2006; Steinwart et al., 2005; Theiler and Michael Cai, 2003; Pham et al., 2014; Wang et al., 2009; Fan et al., 2004; Abe et al., 2006; Hempstalk et al., 2008; Neugebauer et al., 2016; Désir et al., 2013; Bánhalmi et al., 2007; Tax and Duin, 2001; Hastie et al., 2009). The idea is that one extends the given data set through accurate approximations of outliers and thus obtains augmented data. This augmented data is useful for finding the parameters of an outlier detection method (i.e., to calibrate it).

3.1 Purpose of Surveying Generation Techniques

There are two ways to calibrate a detection method with the help of augmented data: (1) casting an unsupervised learning task into a supervised one and (2) parameter tuning of one-class classifiers (see Section 3.7 for details). The common ground for these two different ways remains somewhat unclear, mainly due to a limited general perspective. That is, what artificial outliers are used for in general, and how existing techniques to generate them differ are currently not well formulated. The absence of a sophisticated general perspective makes it also difficult to connect the generation of artificial outliers to other research fields, such as generative modeling or adversarial learning. This integration, however, would be beneficial for both the generation of artificial outliers and related fields. One obstacle to such a general perspective, however, is that the terminology used in articles from different fields varies widely.

Possibly due to the missing general perspective, there is not much knowledge available on the performance of generating outliers or methods calibrated with them. For example, we are aware of only one comparison of the two ways to calibrate outlier detection methods by the usage of artificial outliers (Davenport et al., 2006). However, the comparison is only for a few somewhat similar generation techniques. Hence, we find it somewhat challenging to assess whether one of the two ways yields better outlier detection, irrespective of the generation technique used. A sizable study is also needed to investigate

[1] The remainder of this chapter is almost identical to (Steinbuss and Böhm, 2020a), which is submitted for publication. Adjustments are to ensure consistency for this dissertation.

the hypothesis that a high-quality result using a specific generation technique is not general. In other words, other generation techniques might be better on, say, other data sets.

3.2 Goals

In this chapter, we want to give the field of generating artificial outliers based on genuine instances a missing general perspective. This general perspective comprises clarifying the differences between the many diverse techniques to generate artificial outliers that already exist but also formulating and discussing a more general problem formulation.

Having some general perspective, we also aim at a sizable study that features systematic comparisons in terms of the ways for calibrating detection methods and generation techniques for artificial outliers. In particular, we want to compare (1) the performance of the different generation techniques, (2) the difference in outlier-detection performance of the two ways to calibrate outlier detection methods, and (3) analyze the characteristics of the data (e.g., the number of attributes) that influence the performance of techniques for generating artificial outliers. Another goal we have is to construct a concise set of advice that guide in the application of artificial outliers. These should simplify the usage of artificial outliers by much and thus might further increase their usage in the detection of real outliers.

3.3 Methods

We start this chapter by establishing a unified terminology around artificial outliers. We then describe the two different ways to calibrate outlier detection methods with artificial outliers. Following this, we highlight connections to other research fields and possible synergies. Given these connections, we produce a general problem formulation for the generation of artificial outliers and embed existing techniques into it. We describe each existing technique, using the unified terminology introduced. All this together results in the general perspective on the field of artificial outliers we aim at.

We then perform extensive experiments, comparing the two ways to calibrate detection methods with artificial outliers. These also allow us to analyze the performance of the different generation techniques with many benchmark-data sets on outlier detection. The effect of data characteristics, like the number of attributes, can be analyzed as well. Following the careful analysis of the results of our experiments we synthesize the findings obtained into a straightforward decision process that guides in the usage of artificial outliers.

3.4 Organization of This Chapter

The remainder of this chapter is structured as follows. First, we outline the scope of this chapter in Section 3.5. We describe the ambiguity of terminology from the literature in Section 3.6, and describe the usages of artificial outliers in Section 3.7. We then establish connections between the topic of generating artificial outliers with other research fields in

Section 3.8. In Section 3.9, we offer a general problem formulation. Section 3.10 describes the different generation techniques that presently exist. Section 3.11 outlines methods to filter artificial outliers for ones that give better results than the set of unfiltered ones. Section 3.12 contains the results of our extensive experimental study, and Section 3.13 presents our conclusions.

3.5 Scope of This Chapter

The study in this chapter focuses on the description, categorization, and comparison of the various existing generation techniques for artificial outliers based on some genuine instances. We do not propose any new generation technique but only compare the existing ones, mainly quantitatively. Beyond the detail that is necessary to this end, we do not carry out any further investigation of the behavior of the different techniques. We do also not actively question the value and purpose of artificial outliers in addition to what others have already observed. In this chapter, we do not use generation techniques to benchmark outlier-detection algorithms. Such a benchmark is the focus of Chapter 4. Nevertheless, the study in this chapter marks an excellent starting point for anyone interested in the topic of artificial outliers. We do show where the spectrum of the existing techniques is ranging, how well the techniques perform in specific settings, and what is currently achievable in terms of outlier-generation quality.

3.6 Ambiguous Terminology in the Literature

In the literature, many terminology differences make grasping an article difficult. In this section, we clarify notions encountered frequently in the literature and match this notion with ours. This matching greatly simplifies the reading of related articles.

Certain issues arise in the process of describing a data set. “Instances” are also referred to as “examples” (Abe et al., 2006; Bánhalmi et al., 2007; Curry and Heywood, 2009), “objects” (Tax and Duin, 2001; Theiler and Michael Cai, 2003; Wang et al., 2009), “observations” (Steinwart et al., 2005; Shi and Horvath, 2006), “vectors” (Gonzalez et al., 2002), “input/sample” (Lee et al., 2018), “data” (Wang et al., 2018), or “data points” (Dai et al., 2017). Throughout this thesis, we prefer the term “instances”. Another issue is the naming of the different characteristics of instances. Common terms are “attributes” (Theiler and Michael Cai, 2003; Bánhalmi et al., 2007; Hempstalk et al., 2008; Curry and Heywood, 2009; Wang et al., 2009; Désir et al., 2013; Steinbuss and Böhm, 2017), “features” (Gonzalez et al., 2002; Fan et al., 2004; Steinwart et al., 2005; Pham et al., 2014; Neugebauer et al., 2016) or “dimensions” (Tax and Duin, 2001; Wang et al., 2018). We use “attribute”. Another ambiguity is the term for the set of all possible instances. For example, when the data set consists of d real valued attributes, the set of all possible instances is some subset of $\mathbb{R}^d$.Possible terms are “feature space” (Tax and Duin, 2001; Wang et al., 2009; Neugebauer et al., 2016), “space” (Gonzalez et al., 2002), “domain” (Fan et al., 2004), “input space” (Dai et al., 2017) or “region” (Steinbuss and Böhm, 2017). We use “instance space”.

There also are ambiguities in the general field of outlier detection. Most central is the notion of outliers itself. Aside from “outlier” (Tax and Duin, 2001; Abe et al., 2006; Hempstalk et al., 2008; Curry and Heywood, 2009; Wang et al., 2009; Désir et al., 2013; Neugebauer et al., 2016; Steinbuss and Böhm, 2017; Wang et al., 2018), some authors use “anomaly” (Gonzalez et al., 2002; Theiler and Michael Cai, 2003; Fan et al., 2004; Steinwart et al., 2005), “out-of-distribution sample” (Lee et al., 2018), “negative example” (Gonzalez et al., 2002; Bánhalmi et al., 2007), “counter example” (Bánhalmi et al., 2007), “attack example” (Pham et al., 2014) or “infeasible example” (Neugebauer et al., 2016). We use “outlier”. The term for the counterpart of outliers is ambiguous as well. While they often are referred to as “normal” instances (Gonzalez et al., 2002; Theiler and Michael Cai, 2003; Fan et al., 2004; Steinwart et al., 2005; Abe et al., 2006; Hempstalk et al., 2008; Pham et al., 2014; Wang et al., 2018), other terms used include “inlier” (Steinbuss and Böhm, 2017), “positive” instance (Bánhalmi et al., 2007) or “feasible” instance (Neugebauer et al., 2016). We use “inlier”.

We see two notions with ambiguous terminology related to artificial outliers themselves. These outliers often are referred to as “artificial” outliers (Tax and Duin, 2001; Theiler and Michael Cai, 2003; Fan et al., 2004; Steinwart et al., 2005; Abe et al., 2006; Hempstalk et al., 2008; Curry and Heywood, 2009; Wang et al., 2009; Désir et al., 2013; Pham et al., 2014; Neugebauer et al., 2016), and the procedure that creates them is referred to as “generation” (Tax and Duin, 2001; Gonzalez et al., 2002; Fan et al., 2004; Steinwart et al., 2005; Abe et al., 2006; Shi and Horvath, 2006; Bánhalmi et al., 2007; Hempstalk et al., 2008; Curry and Heywood, 2009; Wang et al., 2009; Désir et al., 2013; Pham et al., 2014; Neugebauer et al., 2016; Wang et al., 2018). However, in (Wang et al., 2018) “pseudo” and in (Shi and Horvath, 2006) “synthetic” is used instead of “artificial”. Instead of “generated”, in (Steinbuss and Böhm, 2017) “placed”, and in (Lee et al., 2018; Dai et al., 2017) “sample” is used. We use “artificial” and “generated”.

3.7 Calibration with Artificial Outliers

In this section, we describe the two ways that one can calibrate an outlier detection method through artificial outliers, subsequently referred to as “use cases”. The joint description of these use cases is the first building block for our general perspective on artificial outliers.

We are aware of two use cases from the literature: (1) casting an unsupervised learning task into a supervised one (Gonzalez et al., 2002; Shi and Horvath, 2006; Steinwart et al., 2005; Theiler and Michael Cai, 2003; Pham et al., 2014; Fan et al., 2004; Abe et al., 2006; Hempstalk et al., 2008; Neugebauer et al., 2016; Désir et al., 2013; Bánhalmi et al., 2007; Hastie et al., 2009; El-Yaniv and Nisenson, 2007), subsequently referred to as “casting task”; and (2) hyperparameter tuning of one-class classifiers (Wang et al., 2018, 2009; Tax and Duin, 2001; Dai et al., 2017), referred to as “one-class tuning”.

3.7.1 Casting Task

The basis for this use case is the observation that the augmented data consists of two fully labeled classes: genuine and artificial instances. The genuine instances result from observations in the real world, while artificial instances result from generation. Thus, one can apply any classifier to set the two apart. Genuine instances mostly are inliers, and the artificial outliers have been generated so that they are outliers. The classifier thus learns to distinguish between inliers and outliers. Next, the number of artificial outliers is controllable. Thus, unlike "classical" supervised outlier detection, this classification does not even have to be unbalanced.

One reason this approach is common might be that it has a theoretical basis (Hastie et al., 2009; Hempstalk et al., 2008; Abe et al., 2006; Theiler and Michael Cai, 2003; Steinwart et al., 2005; El-Yaniv and Nisenson, 2007). Given some data with unknown distribution, one can use a classifier that distinguishes genuine from artificial instances, to obtain a density estimation of the genuine instances. This density estimation allows for identifying unlikely instances. Section 14.2.4 ("Unsupervised as Supervised Learning") in (Hastie et al., 2009) and (Steinwart et al., 2005) show this for different types of classifiers.

3.7.2 One-Class Tuning

Another use case is hyperparameter tuning for one-class classifiers (Wang et al., 2018, 2009; Tax and Duin, 2001). The training of a one-class classifier uses only instances from one class to learn to separate new instances belonging to this class from those that do not (Hempstalk et al., 2008). Instances not belonging to the class are deemed outliers. A common one-class classifier belongs to the category of Support Vector Machine (SVM): the Support Vector Data Description (SVDD) introduced in Tax and Duin (1999). It has hyperparameters s and ν (Tax and Duin, 2001) where s is the kernel width, and ν is an upper bound on the fraction of genuine instances classified as outlying. To choose values for both parameters, one must optimize the error rate of the resulting one-class classifier (Tax and Duin, 2001). However, since one-class classification is applied when there is either no outliers or not a sufficient number of outliers, estimating this error is difficult. Multiple techniques for the generation of artificial outliers have been developed to estimate the error (Wang et al., 2018, 2009; Tax and Duin, 2001).

While the two use cases described are different, their outcome is the same: a calibrated classifier for outlier detection. In both use cases, the artificial outliers help train the classifier. A useful generation technique yields a high detection rate on outliers, be they genuine or artificial. To investigate the quality differences in terms of outlier detection between the two cases, we have performed experiments, see Section 3.12.7. We have found that there are some differences, but none of the two use cases is always preferable in terms of detection quality.

3.8 Connection to Other Fields

The specifics of the use cases for artificial outliers given in Section 3.7 allow us now to connect the generation of artificial outliers to other research fields. This connection is one of the building blocks for our general perspective on artificial outliers.

We see at least three broad research fields closely connected to generating artificial outliers based on genuine instances: generative models, design of experiments, and adversarial machine learning. The first two are fields from statistics, while the last one is a relatively new paradigm, mostly from computer science. In the following, we will discuss the general ideas of each of these research fields and their connection to the generation of artificial outliers.

3.8.1 Generative Models

Our source for the following discussion is mostly (Bernardo et al., 2007) that discusses the connection between discriminative and generative models.

In machine learning, one often tries to predict a label y_i that belongs to an instance $\vec{x}_i$. In the remainder of this section, y_i identifies $\vec{x}_i$ as "inlier" or "outlier" (classification). The goal then is to determine the conditional probability $\mathrm{p}(y \mid \vec{x})$ from a given data set $\underline{X}$ (i.e., the distribution of y given an instance $\vec{x}_i$). Two frequent approaches to do so are discriminative or generative, respectively. Discriminative models directly approximate $\mathrm{p}(y \mid \vec{x})$, while generative ones first try to find the joint distribution $\mathrm{p}(y,\ \vec{x})$. By sampling from this joint distribution, it is possible to generate instances. Hence, these models are called "generative". Specifying the joint distribution $\mathrm{p}(y,\ \vec{x})$ is usually done by defining a distribution for the classes $\mathrm{p}(y)$ and a class-conditional distribution for the instances $\mathrm{p}(\vec{x} \mid y)$, along with finding the best fit to the instances in $\underline{X}$. This specification gives the joint distribution by

$$\mathrm{p}(y,\ \vec{x}) = \mathrm{p}(\vec{x} \mid y) \cdot \mathrm{p}(y)\,. \tag{3.1}$$

We have omitted the distribution parameters that are fitted using $\underline{X}$ for the sake of clarity.

Since y can only take two distinct values, the generative model is fully specified if $\mathrm{p}(\vec{x} \mid y = \textit{inlier})$, $\mathrm{p}(y = \textit{inlier}) =: p_{\textit{in}}$, $\mathrm{p}(\vec{x} \mid y = \textit{outlier})$ and $\mathrm{p}(y = \textit{outlier}) =: p_{\textit{out}}$ are specified. Artificial outliers are essentially samples from $\mathrm{p}(\vec{x} \mid y = \textit{outlier})$ or at least approximations of these samples. To generate the artificial outliers, one explicitly or implicitly defines $\mathrm{p}(\vec{x} \mid y = \textit{outlier})$. With the number of samples generated, $p_{\textit{in}}$ and $p_{\textit{out}}$ are defined as well. Thus, when generating artificial outliers, most parts of the generative model are also defined. The only missing part is the distribution of inliers $\mathrm{p}(\vec{x} \mid y = \textit{inlier})$. Hence, if we explicitly define $\mathrm{p}(\vec{x} \mid y = \textit{outlier})$ and estimate $\mathrm{p}(\vec{x} \mid y = \textit{inlier})$ from the data, we end up with a generative model for outlier detection. However, this is not the only connection between artificial outliers and generative models. A generative model can also be used to classify instances as outlier or inlier. This classification is also what artificial outliers facilitate in the use cases *casting task* and *one-class tuning*. Interestingly, outliers do not need to be generated for the generative model since their distribution only needs to be defined. The issue with such an approach, however, is that estimating $\mathrm{p}(\vec{x} \mid y = \textit{inlier})$ is not simple. The generation of outliers is often more straightforward than this estimation. Thus, using some artificial outliers to train or tune a classifier is simpler or sometimes merely more effective than is specifying the generative model. The connection between artificial outliers and generative models is goes deep. If it is simple to, for instance, estimate $\mathrm{p}(\vec{x} \mid y = \mathit{inlier})$ in some setting, one might prefer the generative model over artificial outliers.

Another insight in this context comes from p_{out}. We find it surprising that many inventors of generation techniques do not discuss its importance. Since p_{out} is part of the generative model, it does affect the decision of whether an instance is an outlier or not. Recall that in the case of artificial outliers, p_{out} is essentially given by n_{art}. Hence, n_{art} also determines whether an instance is an outlier or not.

3.8.2 Design of Experiments

Our basis for the following description is (Lovric, 2011).

The "Design of Experiments" deals with modeling the dependence of a random variable **Y** on some deterministic factors $x^{(1)}, \dots, x^{(d)}$ (i.e., attribute values). A combination of the d deterministic factors yields an artificial instance $\vec{a}$. As in the previous section, y (a realization of **Y**) identifies $\vec{a}$ as *inlier* or *outlier*. The topic "Design of Experiments" aims to find a set of such factor combinations $\underline{D} = \{\vec{a}_1, \dots, \vec{a}_{n_{art}}\}$ that give optimal results regarding **Y**. To illustrate, "optimal" can mean that our classification with regard to **Y** yields a perfect accuracy. One does not need to estimate this classification from $\underline{D}$ alone. It is also reasonable to consider that it is learned from $\underline{Z} = \underline{D} \cup \underline{X}$ (i.e., augmented data). Hence, seeing the generation of artificial outliers as a subfield of the design of experiments is conceivable. Although it is difficult to make the definition of "optimal" more concrete, we approach this in Section 3.9. The design of experiments encompasses extensive theoretical work. We believe that establishing a rigid connection of artificial outliers to this broad field may facilitate a rather formal derivation of relevant concepts and approaches.

o our knowledge, there exists no previous work regarding artificial outliers in the field of design of experiments. However, some rather general techniques to a good $\underline{D}$ seem to be applicable. One such technique is already common when generating artificial outliers (UNIFBOX, see Section 3.10.1) (Lovric, 2011). It relies entirely on random sampling. This reliance makes it difficult to ensure that the whole instance space (e.g., $\mathbb{R}^d$) is evenly covered. However, such behavior is often a desirable property, since it is usually not known *a priori* which regions of the instance space have to be covered. The Latin hypercube design ensures that the instances spread evenly in the instance space (Santner et al., 2013). See Definition 1.

Definition 1 (LHS) *LHS is an technique to generate artificial instances using the so-called* Latin hypercube design, *as follows: To generate n_{art} instances, partition the value range of each attribute into n_{art} equally sized intervals. This yields a grid with $(n_{art})^d$ cells. Assign the integers $1, \dots, n_{art}$ to cells so that each integer appears only once in any dimension of the grid. Now randomly select an integer $i \in \{1, \dots, n_{art}\}$. Finally, generate n_{art} instances by sampling uniformly within the n_{art} cells which integer i has been assigned to.*

The only generation parameter of LHS is n_{art}. Figure 3.1 features an illustration of the LHS technique.

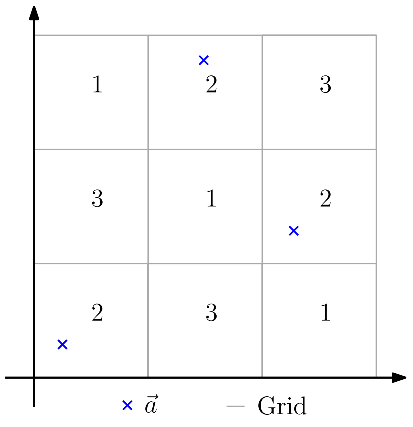


Figure 3.1: Illustration of the generation technique LHS. $n_{art} = 3$ and $i = 2$.

In our experiments, we let the artificial instances generated with LHS compete against the output of techniques specifically designed for the generation of outliers. In the two use cases *casting task* and *one-class tuning*, we find that the instances generated with LHS yield comparable outlier-detection quality.

3.8.3 Adversarial Machine Learning

The recent development of the GAN (Goodfellow et al., 2014a) has given adversarial machine learning much attention. In general, the field is concerned with the robustness of machine learning in terms of adversarial input and countermeasures (Biggio and Roli, 2017). Such adversarial input is artificial data deemed either evasive or poisonous (Kumar et al., 2017). Evasive instances fool a trained classifier, yielding the wrong classification (e.g., spam email that is not classified as such). Poisonous instances, on the other hand, prevent a classifier from being trained correctly.

In our view, the generation of adversarial input is similar to that of artificial outliers. In essence, an outlier that is wrongly classified as an inlier can be a very useful artificial outlier, as illustrated later in Section 3.9. The fact that there is an outlier-generation technique using GANs (Lee et al., 2018; Dai et al., 2017) (see Section 3.10.1 for details) further emphasizes the strong connection between artificial outliers and adversarial machine learning. The idea of GANs (Goodfellow et al., 2014a) is to have two models, a generative and a discriminative one, that compete against each other. The generating model tries to generate instances that the discriminator model cannot tell apart from genuine ones. The generative model is thus encouraged to generate instances as close as possible to genuine ones. This idea is similar to a generation technique proposed in (Hempstalk et al., 2008) (Definition 7). However, techniques to generate adversarial inputs tend to be very specific to a classifier or task they are supposed to attack (Brendel et al., 2017). Thus, one cannot always use them for the generation of artificial outliers.

3.9 Problem Definition

One of the central building blocks for a general perspective on artificial outliers is a unified problem formulation. Such a formulation — that takes into account the different use cases for artificial outliers described in Section 3.7 — now follows. The integration of artificial outliers within other research fields given in Section 3.8 is essential here as well since it frames the distinctive ideas from this field.

Artificial outliers are expected to approximate instances from **O**. When we know **O**, obtaining artificial outliers becomes trivial: We sample from the distribution that matches our knowledge. An exemplary scenario is when we want to detect faults in a system, and the maintainer knows the distribution of these faults. However, one usually does not have any or has only minimal knowledge of **O**. To illustrate, think again about the system maintainer. It is highly unlikely in the exemplary scenario just sketched that the system maintainer knows the distribution of faults without having multiple faulty instances. Thus, we have to rely on assumptions on outliers that allow the generation of instances approximating ones from **O**. To reflect our limited or missing knowledge on **O**, we make assumptions so that outliers generated are as *uninformative* as possible (Theiler and Michael Cai, 2003). That is, they should disclose only very few characteristics of outliers and hence result in the detection of many possible types. However, at the same time, we want to make the generated artificial outliers as *interesting* as possible. Definition 2 formalizes the concept of interesting artificial outliers, and Example 4 illustrates it.

Definition 2 ($\underline{A}_{inter}$) *A set of* interesting artificial outliers $\underline{A}_{inter} = \{\vec{a}_1, \ldots, \vec{a}_{n_{art}}\}$ *is a set of instances that solve a use case for artificial outliers well.*

Recall the use cases introduced earlier, *casting task* or *one-class tuning*, and consider the following example.

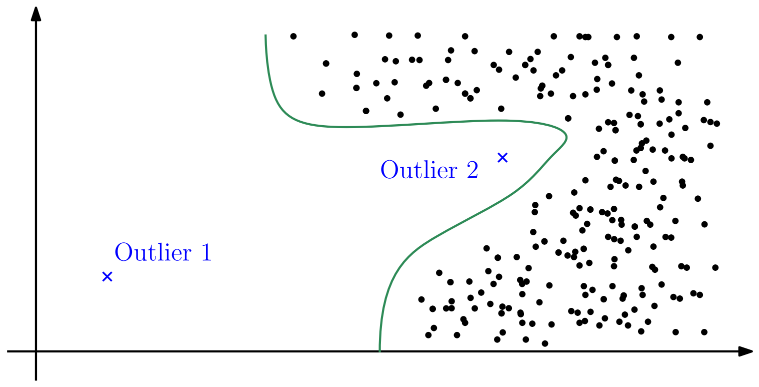


Figure 3.2: Illustration of interesting artificial outliers.

Example 4 *The use case in this example is* casting task *(i.e., training a classifier for outlier detection with training data that contains only inliers). The use case is solved well if the outlier-detection accuracy is later high. To train the classifier, we use artificial outliers. See Figure 3.2. The green line is the best decision boundary between inliers and outliers. Outlier*

1 is far from any inlier. Such an outlier is not very useful when training the classifier. It is rather trivial to classify it as outlying, and it might even pull the decision boundary of the classifier away from inliers. Outlier 2, by contrast, is helpful when learning the correct decision boundary and is thus rather interesting.

The interestingness of artificial outliers depends heavily on the specific application — e.g., if we aim at density estimation uniformity of the outliers is desirable (Steinwart et al., 2005; Hastie et al., 2009). If we are interested in exploring the instance space instead of just calibrating detection methods, artificial outliers far away from inliers might be interesting as well. For example, when analyzing properties of hidden outliers using generated ones, an exhaustive generation can be useful (cf. Chapter 5). This relationship makes a precise and general definition of interesting artificial outliers difficult. Another issue is that the interestingness of artificial outliers also depends on the other generated instances. It might well be that Outliers 1 and 2, in combination, lead to a better decision boundary. Definition 2 has reflected this possibility.

However, the situation is even more complicated since the number of generated outliers is of importance as well. Any additional artificial instance increases the computational effort. Thus, we want to generate as few artificial outliers as possible. This aim leads to the following definition.

Definition 3 (Minimal $\underline{A}_{inter}$) *A* minimal *set $\underline{A}_{inter}$ of artificial outliers is a set of interesting ones that has a minimal number of elements n_{art} and is still interesting.*

When generating artificial outliers for a use case, one would like to have a minimal set $\underline{A}_{inter}$. However, there is a trade-off. Interesting outliers are often counter to uninformative outliers. Consider Example 4 where we suppose that outliers occur close to genuine instances and not everywhere. With such additional assumptions, one loses some generality. One could argue that having some uninteresting artificial outliers is better than losing this generality. However, in high-dimensional spaces in particular, including uninteresting artificial outliers can soon become very expensive computationally (Tax and Duin, 2001; Hempstalk et al., 2008; Steinbuss and Böhm, 2017; Davenport et al., 2006). Hence, existing techniques make different assumptions about outliers in order to obtain a minimal set $\underline{A}_{inter}$. These assumptions will become apparent in Section 3.10 when we describe the techniques. However, having a specific use case in mind, one must be careful that the assumptions fit the use case. For instance, as mentioned before, if one wants to perform an explorative analysis, generating instances only very close to the boundary of inliers tends not to be good.

If artificial outliers are used, usually not only **O** is missing, but also **I**. Otherwise, a generative model might be preferable, see Section 3.8.1. Hence, the generation is based only on samples from **I** possibly mixed with some from **O** (i.e., on $\underline{X}$). Of course, it is possible that the instances from $\underline{X}$ are not sufficient to represent **I**. Think of the case that there is no instance from $\underline{X}$ in parts of the instance space that inliers can lie in. An artificial outlier in this part might then be an outlier regarding $\underline{X}$ but not regarding **I**. However, an essential assumption behind all generation techniques is that there is a sufficient number of genuine instances (n_{genu}) available.

Next to the actual generation of artificial outliers, techniques also exist to filter existing artificial outliers for interesting ones. That is, instead of generating instances at a particular location (e.g., very close to the boundary), one generates many artificial outliers with some simple technique and tests which ones are interesting. Some of these filtering techniques have been proposed together with a specific technique to generate the outliers. However, they might also work well when the generating technique is a different one. Thus, in the following two sections, we first describe the different techniques relevant to generate artificial outliers and then filtering techniques.

3.10 Generating Techniques

In this section, we review the various generating techniques with context to the problem formulation from Section 3.9. This review is another building block of our general perspective. We start by classifying the generation techniques in terms of how they relate to the characteristics of $\underline{X}$. We then describe each technique. Techniques with a similar generating procedure are described together in order to reduce redundancy and improve comprehension. This description results in two somewhat orthogonal classifications of generation techniques: one based on the characteristic of $\underline{X}$ and one in terms of similar generation procedures.

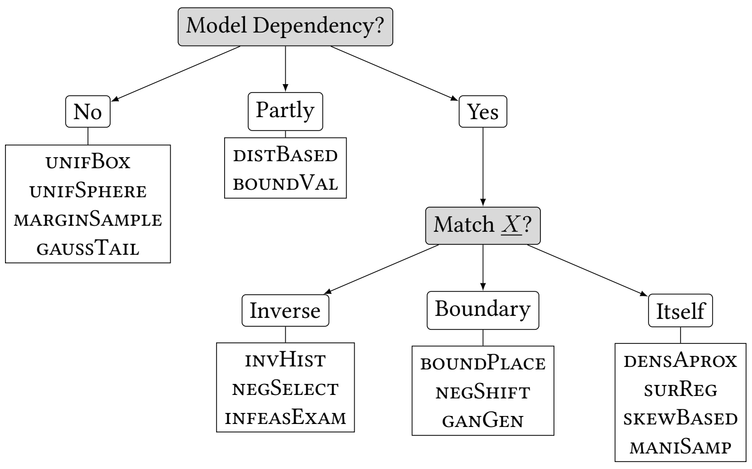


Figure 3.3: Classification of generating techniques.

In terms of retaining characteristics from $\underline{X}$, we group the techniques in six groups, see Figure 3.3. They differ in the extent of modeling the dependency on $\underline{X}$, and how well they match with instances from $\underline{X}$. In terms of the dependency, they either do not model it, do so only partly, or model all of it. Regarding the match with instances from $\underline{X}$, the artificial outliers can be somewhat inversely distributed, close to their boundary, or entirely similar. In the following, we describe the existing techniques, grouped by generation paradigms.

3.10.1 Sampling from a Distribution

Sampling from a distribution is a common way to generate data. There also exist techniques to generate outliers with such sampling. The difference among these techniques is the distribution **O** from which they sample.

Uniform within a Hyper-Rectangle

Definition 4 features the most frequently used distribution (Steinbuss and Böhm, 2017; Hastie et al., 2009; Hempstalk et al., 2008; Shi and Horvath, 2006; Abe et al., 2006; Steinwart et al., 2005; Fan et al., 2004; Theiler and Michael Cai, 2003; Fan et al., 2001; Tax and Duin, 2001; El-Yaniv and Nisenson, 2007; Davenport et al., 2006).

Definition 4 (unifBox) $\mathbf{O}_{unifBox}$ *is a uniform distribution within a hyper-rectangle encapsulating all genuine instances. The parameters are* n_{art} *and the bounds* $l, u \in \mathbb{R}^d$ *for the hyper-rectangle.*

Instances from $\underline{X}$ usually determine the bounds $l, u \in \mathbb{R}^d$. For this reason, this technique needs them as input. In (Tax and Duin, 2001) and (Fan et al., 2004), these bounds are chosen so that the hyper-rectangle encapsulates all genuine instances. In (Steinbuss and Böhm, 2017) the minimum and maximum for each attribute obtained from $\underline{X}$ are used. In (Theiler and Michael Cai, 2003), it is mentioned that the boundary does not need to be far beyond these boundaries. In (Abe et al., 2006), the rule that the boundary should expand the minimum and maximum by 10% is proposed, while in (Désir et al., 2013) to expand the boundary by 20%. In Section 3.12.4, we describe the boundaries used in our experiments.

Uniform within a Hyper-Sphere

In (Tax and Duin, 2001), a straightforward adaptation of the distribution from the unifBox technique is proposed. The technique emphasizes generating outliers close to genuine instances.

Definition 5 (unifSphere) $\mathbf{O}_{unifSphere}$ *is a uniform distribution in the minimal bounding sphere encapsulating all genuine instances. The only generation parameter is* n_{art}.

There are various techniques to obtain or approximate the minimal bounding sphere (e.g., see (Larsson, 2008)). In (Tax and Duin, 2001), it is proposed to use the optimization approach also used when fitting a SVDD. Sampling uniformly from a hyper-sphere is not simple. In (Tax and Duin, 2001), a method using transformed samples from a multivariate Gaussian distribution is proposed to this end.

Manifold Sampling

The generation technique maniSamp (Davenport et al., 2006) also uses hyper-spheres. Similar to unifSphere, the aim is to generate instances close to genuine ones. More specifically, they want to generate instances within the manifold in which the genuine instances lie. To model this manifold, they use multiple hyper-spheres. The sampling distribution **O** is formalized in Definition 6.

Definition 6 (MANISAMP) *For each $\vec{x}_i \in \underline{X}$, let $\overline{d}_i^k$ be the average distance to its k nearest neighbors. Then the sampling distribution $\mathbf{O}_{\text{MANISAMP}}$ is the union of the hyper-spheres with center $\vec{x}_i$ and radius $\overline{d}_i^k$ for $i \in \{1, \ldots, n_{genu}\}$. The parameters are n_{art} and k.*

Using Density Estimation

In (Hempstalk et al., 2008) a reformulation of the *casting task* use case is approached so that the set of artificial outliers is close to minimal. For this objective, they find that the ideal distribution of artificial outliers should be the distribution of inliers. However, since the distribution of inliers usually is not known, they propose to estimate it with any density-estimation technique (see Definition 7).

Definition 7 (DENSAPROX) $\mathbf{O}_{\text{DENSAPROX}}$ *is the result of density estimation on genuine instances. The parameters are n_{art} and the density-estimation technique.*

In (Hempstalk et al., 2008), it is stated that any density-estimation technique can be used in principle, as long as it is possible to draw samples from the density estimate. In the experiments of (Hempstalk et al., 2008), two variants of density estimation are used. In both cases, they assume a specific distribution and estimate its parameters from $\underline{X}$. These distributions are as follows:

- A multivariate Gaussian distribution having a covariance matrix with only diagonal elements. That is, attributes are independent.[2]
- A product of d Gaussian mixtures, one for each attribute.

Outside of a Confidence Interval

In (Pham et al., 2014), it is found that outlier instances should be very different from inliers. Thus, in (Pham et al., 2014), it is proposed to generate outliers far from most genuine instances by using the distribution from Definition 8.

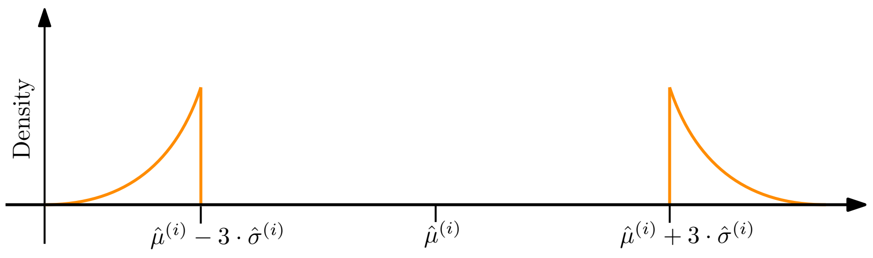


Figure 3.4: Illustration of $\mathbf{D}^{(i)}$ from GAUSSTAIL technique.

[2] This actually results in the same generation process Abe et al. (2006) proposes for the MARGINSAMPLE technique.

Definition 8 (gaussTail) *Let $\hat{\mu}^{(i)}$ and $\hat{\sigma}^{(i)}$ be the mean and standard deviation estimated from attribute i in $\underline{X}$. The distribution $\mathbf{D}^{(i)}$ has density zero for any value*

$$x^{(i)} \in \underbrace{\left[\hat{\mu}^{(i)} - 3 \cdot \hat{\sigma}^{(i)},\ \hat{\mu}^{(i)} + 3 \cdot \hat{\sigma}^{(i)}\right]}_{=: \mathcal{R}_{zero}}. \tag{3.2}$$

Then $\mathbf{O}_{\text{gaussTail}}$ is the product of $\mathbf{D}^{(i)}$ for all attributes. The only parameter is n_{art}.

In (Pham et al., 2014), the form of the density outside the interval $\mathcal{R}_{zero}$ is not discussed. In our experiments, we assume $\mathbf{D}^{(i)}$ to be a Gaussian with the density for $x^{(i)} \in \mathcal{R}_{zero}$ set to zero. See Figure 3.4.

Inverse Histogram

In (Désir et al., 2013), it is proposed that the distribution of outliers should be exactly complementary to the distribution of instances. In other words, they propose to use the distribution from Definition 9.

Definition 9 (invHist) *Let H_{in} be the normalized histogram of inliers. Then $\mathbf{O}_{\text{invHist}}$ has probability density function $1 - H_{in}$. The parameters are n_{art} and the histogram-estimation technique.*

In (Désir et al., 2013), details on how to compute the normalized histogram are not discussed. They say that the instance-space boundary (i.e., minimum and maximum of each attribute) should be increased by 20%.

Generative Adversarial Networks

In (Dai et al., 2017) and (Lee et al., 2018) it is proposed to use a GAN (Goodfellow et al., 2014a) to generate artificial outliers. The generator from a trained GAN architecture is an implicit generative model (Dai et al., 2017). Hence, it can generate instances that are similar to the instances it was trained with (the genuine ones) but does not provide a closed-form of their density. The generator aims at maximizing the similarity of generated and genuine instances. Hence, using the generator to generate outliers is not straightforward. To achieve the generation of outliers, in (Dai et al., 2017) and (Lee et al., 2018), the same strategy is proposed. A penalty term is added to the objective function of the generator. This penalty encourages a generation of instances further away from genuine ones. Since both formulations are based on the same idea (Lee et al., 2018), Definition 10 features the formulation from Dai et al. (2017). We deem it more illustrative for our purpose.

Definition 10 (ganGen) *Let $\mathbf{Z}$ be the distribution of the prior input noise for the generator function* $\text{gen}\colon \text{supp}(\mathbf{Z}) \to \mathbb{R}$. *Let* $\text{disc}\colon \mathbb{R} \to [0,1]$ *be the discriminator function outputting the probability that an instance is not generated and* $\text{dens}(\cdot)$ *an estimate of the density function of genuine instances. $\mathbf{O}_{\text{ganGen}}$ is then the distribution of instances sampled from* $\text{gen}(\mathbf{Z})$ *optimized according to*

$$\min_{\text{gen}} \ \mathbb{E}_{\vec{a}\sim\text{gen}(\mathbf{Z})}\left[\log(\text{dens}(\vec{a}))\ \mathbb{1}_{\{\text{dens}(\vec{a})>\varepsilon\}}\right] + \underbrace{\mathbb{E}_{\vec{a}\sim\text{gen}(\mathbf{Z})}\left[\log(1-\text{disc}(\vec{a}))\right]}_{\textit{Original}\ \text{GAN}\ \textit{(Goodfellow et al., 2014a)}}. \tag{3.3}$$

The parameters are n_{art}, ε, the network structure for the GAN *architecture, and the density estimation technique from which* dens($\cdot$) *resulted.*

The first term in Equation (3.3) intuitively punishes the generator for generating instances that have a very high density ($> \varepsilon$) according to dens($\cdot$). Hence, the generation of instances takes place in regions of the instance space with rather low density.

3.10.2 Shifting Genuine Instances

Techniques in this category modify attribute values of genuine instances to generate outliers. The techniques INFEASEXAM, SKEWBASED, and SURREG add random noise to genuine instances. Both BOUNDPLACE and NEGSHIFT shift genuine instances so that they move away from other genuine ones.

Plain Gaussian Noise

In (Neugebauer et al., 2016), it is proposed to alter instances with Gaussian noise and then filter the resulting instances for those far from inliers (i.e., having a certain distance to them). Only in the first iteration are inliers altered; then, only the resulting artificial outliers are. See INFEASEXAM in Algorithm 1. The technique requires not just genuine instances but genuine inliers.

Algorithm 1 INFEASEXAM.

```
Input: $n_{art}$, $\vec{\mu}$, $\underline{\Sigma}$, $\alpha$, $\epsilon$
 1: for $i \in \{1, \dots, n_{genu}\}$ do
 2:     $\vec{r}$ = Sample from Gaussian with $\vec{\mu}$ and $\underline{\Sigma}$
 3:     $\vec{a}_i = \vec{x}_i + \vec{r} \cdot \alpha$
 4:     $d_i$ = distance of $\vec{a}_i$ to closest inlier
 5:     if $d_i \geq \epsilon$ then
 6:         Add $\vec{a}_i$ to artificial outliers $\underline{O}_{art}$
 7:     end if
 8: end for
 9: repeat
10:     Randomly choose $\vec{a}_i$ from $\underline{O}_{art}$
11:     $\vec{r}$ = Sample from Gaussian with $\vec{\mu}$ and $\underline{\Sigma}$
12:     $\vec{o}_i = \vec{a} + \vec{r} \cdot \alpha$
13:     $d_i$ = distance of $\vec{o}_i$ to closest inlier
14:     if $d_i \geq \epsilon$ then
15:         Add $\vec{o}_i$ to artificial outliers $\underline{O}_{art}$
16:     end if
17: until $|\underline{O}_{art}| = n_{art}$
```

Scaled Gaussian Noise

In (Deng and Xu, 2007), a so-called skewness-based generation technique for artificial outliers is proposed. The technique uses noise added to genuine instances. Similar to the case in Algorithm 1, this noise is Gaussian; a parameter α scales it. However, the technique in (Deng and Xu, 2007) does not make use of any filtering or several iterations. See Definition 11.

Definition 11 (skewBased) *Let $\hat{\sigma}^{(i)}$ be the standard deviation estimated for Attribute $i \in \{1, \ldots, d\}$, and let each $r^{(i)}$ be a random value drawn from a standard normal distribution. Further, let*

$$v^{(i)} := \frac{\hat{\sigma}^{(i)}}{\sum_{j=1}^{d} \hat{\sigma}^{(j)}}, \quad \zeta^{(i)} := \frac{r^{(i)}}{\sum_{j=1}^{d} r^{(j)}}. \tag{3.4}$$

An outlier $\vec{o}_{\text{SKEWBASED}}$ is then generated by

$$\vec{o}_{\text{SKEWBASED}} = \vec{x} + \alpha \cdot (v^{(1)} \cdot \zeta^{(1)}, \ldots, v^{(d)} \cdot \zeta^{(d)}), \tag{3.5}$$

where $\vec{x}$ is a randomly drawn genuine instance. The parameters are n_{art} and α.

Uniform Noise

In (Steinbuss and Böhm, 2017) we propose another technique to generate outliers.[3] The rationale is to adjust the tightness of artificial outliers around inliers. The technique adds uniform noise parameterized with $\varepsilon \in [0, 1]$ to genuine instances. If $\varepsilon = 1$, the generation will result in samples from a uniform distribution. If $\varepsilon = 0$, there will be samples of genuine instances. See Definition 12.

Definition 12 (surReg) *Let $\vec{x}$ be a randomly drawn instance from $\underline{X}$. W.l.o.g., $\vec{x} \in [l, u]^d$. Further, let $r^{(i)}$ $i \in \{1, \ldots, d\}$ be random values drawn uniformly from the range $\varepsilon \cdot (x^{(i)} - l)$ to $\varepsilon \cdot (u - x^{(i)})$. Then an outlier $\vec{o}_{\text{SURREG}}$ is generated by*

$$\vec{o}_{\text{SURREG}} = \vec{x} + (r^{(i)}, \ \ldots, \ r^{(d)}). \tag{3.6}$$

This procedure is repeated until n_{art} outliers are generated. The parameters are n_{art} and ε.

Our experiments in Section 5.6 indicate that 0.1 can be a good value for ε in particular if there are many attributes. Note that for the surReg technique, the data set must have the bounds $[0, 1]$ for every attribute.

Using Boundary Instances

In (Bánhalmi et al., 2007) and (Wang et al., 2018), a similar idea to generate artificial outliers very tightly around the boundary of genuine instances is presented. The idea is to have a two-stage process. In the first stage, one finds boundary instances (i.e., instances that *surround* all other genuine instances). They are then used in the second stage to shift

[3] The technique is detailed more closely in Chapter 5 but outlined here for completeness as well.

genuine instances away from others. See Figure 3.5 for an illustration of the technique proposed in (Bánhalmi et al., 2007). The techniques proposed in (Bánhalmi et al., 2007) and (Wang et al., 2018) differ in the following respects:

1. How boundary instances are found.
2. Which instances are shifted.
3. The magnitude and direction of the shift.

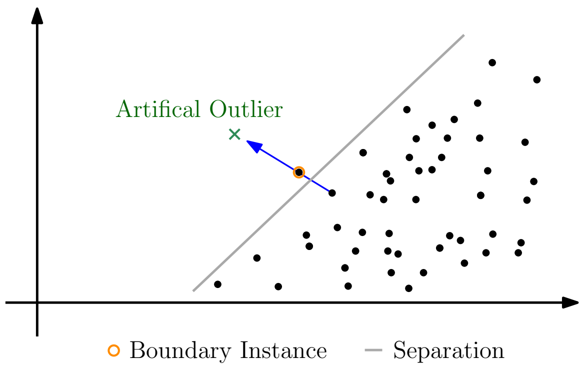


Figure 3.5: Illustration of BOUNDPLACE technique (Bánhalmi et al., 2007).

In (Bánhalmi et al., 2007), it is proposed to determine boundary instances with Algorithm 2. The idea is that an instance is on the boundary if it is linearly separable from its k-nearest neighbors. A hard margin SVM is thus fitted to separate the instance under consideration from its k-nearest neighbors. If it finds such a separation, the instance is deemed on the boundary.

Algorithm 2 Boundary detection (Bánhalmi et al., 2007).

```
Input: $k$
1: for $\vec{x} \in \underline{X}$ do
2:     $\underline{N}$ = $k$-nearest neighbors of $\vec{x}$ in $\underline{X}$
3:     $e_i = \frac{\vec{n}_i - \vec{x}}{\|\vec{n}_i - \vec{x}\|} \quad \forall\ i \in \{1, \dots, k\},\ \vec{n}_i \in \underline{N}$
4:     Separate $e_i$ from origin with hard margin SVM
5:     if Separation succeeds then
6:         $\vec{x}$ is boundary instances
7:         Save $v_{\vec{x}} = \sum_{i=1}^{k} \alpha_i \cdot e_i$
8:     end if
9: end for
```

The vector $v_{\vec{x}}$ in Algorithm 2 is used to compute the shift direction of genuine instances, see Definition 13. The α_is result from fitting the SVM. They weight the contribution of e_i to the final separation.[4]

[4] See (Bánhalmi et al., 2007) for two refinements of this technique that increase the maximally possible n_{art}.

Definition 13 (boundPlace) *Let $\underline{B}$ be the set of boundary instances found with Algorithm 2, with $\mathcal{V}$ as the set of the vectors $v_{\vec{x}}$ saved for each boundary instance found in Algorithm 2. For an instance $\vec{x} \in \underline{X} \setminus \underline{B}$, let $\vec{b} \in \underline{B}$ be the closest boundary instance to $\vec{x}$ with $v_{\vec{x}} \in \mathcal{V}$. Let $\Delta := \vec{b} - \vec{x}$. Further,*

$$\omega = \frac{v_{\vec{x}}^T \cdot (-\Delta)}{\|v_{\vec{x}}\| \cdot \|\Delta\|} \quad \textit{and} \quad \kappa = \frac{\theta}{\theta \cdot \gamma + \omega}, \tag{3.7}$$

where θ and γ are parameters. The instance $\vec{x}$ is then shifted by

$$\vec{o}_{\text{BOUNDPLACE}} = \vec{x} + \Delta \cdot \left(1 + \frac{\kappa}{\|\Delta\|}\right). \tag{3.8}$$

Then $\vec{o}_{\text{BOUNDPLACE}}$ is an artificial outlier generated with the boundPlace technique, if it is deemed a boundary instance. This procedure is repeated for every $\vec{x} \in \underline{X} \setminus \underline{B}$. The parameters are k, θ and γ.

In (Wang et al., 2018) different instantiations for Items 1 to 3 are proposed. To detect boundary instances, the technique from (Wang et al., 2018) relies on Algorithm 3, the Border-Edge Pattern Selection (BEPS) algorithm (Li and Maguire, 2011). Like Algorithm 2, BEPS also relies on the k-nearest neighbors to decide whether an instance $\vec{x}$ is on the boundary or not. However, instead of checking for linear separability using a hard margin SVM, it uses a technical condition on the vectors from a neighbor to the instance $\vec{x}$ (v_i). See (Li and Maguire, 2011; Wang et al., 2018) for details.

Algorithm 3 BEPS algorithm as given in (Wang et al., 2018).

```
Input: None
1:  $k = \lceil 5 \log_{10}(n_{genu}) \rceil, \ \tau = 0.1$
2:  for $\vec{x} \in \underline{X}$ do
3:      $\underline{N} = k$-nearest neighbors of $\vec{x}$
4:      $v_i = \frac{\vec{x} - \vec{n}_i}{\|\vec{x} - \vec{n}_i\|} \quad \forall \ i \in \{1, \ldots, k\}, \ \vec{n}_i \in \underline{N}$
5:      Calculate $\varphi = \sum_{i=1}^{k} v_i$
6:      $\theta_i = v_i^T \cdot \varphi \quad \forall \ i \in \{1, \ldots, k\}$
7:      $l = \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}_{\{\theta_i \geq 0\}}$
8:      if $l \geq 1 - \tau$ then
9:          $\vec{x}$ is boundary instances
10:         Save $\varphi$
11:     end if
12: end for
```

Definition 14 (negShift) *Let $\underline{B}$ be the boundary instances found with Algorithm 3. Further,*

$$\kappa = \frac{1}{|\underline{B}| \cdot k} \sum_{\vec{b} \in \underline{B}} \sum_{i=1}^{k} |\vec{b} - \vec{n}_i|, \tag{3.9}$$

where $\vec{n}_i$ is the i-th neighbor of an instance $\vec{b} \in$ Bounds. $\vec{b}$ is then shifted by

$$\vec{o}_{\textit{NEGSHIFT}} = \vec{b} + \frac{\varphi}{|\varphi|} \cdot \kappa, \tag{3.10}$$

where φ comes from Algorithm 3. This procedure is repeated for every $\vec{b} \in \underline{B}$. There is no parameter.

The κ value determines how far a boundary instance should be shifted. This shift uses the distance of each boundary instance to its k-nearest neighbors. Shifting a boundary instance in the direction of φ then generates an artificial outlier.

3.10.3 Sampling Instance Values

Techniques from this category generate outliers similar to the ones in Section 3.10.2. They do so by directly using genuine instances. However, instead of creating new attribute values, the current techniques recombine existing values of genuine instances to form artificial outliers.

From the Marginals

Several articles propose to use marginal sampling to generate outliers (Shi and Horvath, 2006; Theiler and Michael Cai, 2003; Abe et al., 2006; Hastie et al., 2009).

Definition 15 (marginSample) *Let $\mathbf{I}^{(i)}$ be the distribution of attribute i. Then*

$$\mathbf{O}_{\textit{MARGINSAMPLE}} = \mathbf{I}^{(1)} \cdot \ \cdots \ \cdot \mathbf{I}^{(d)}. \tag{3.11}$$

That is, one can generate outliers from $\mathbf{O}_{\textit{MARGINSAMPLE}}$ by sampling a value from each attribute independently. The only parameter is n_{art}.

From Definition 15, it follows that $\mathbf{O}_{\text{MARGINSAMPLE}}$ and $\mathbf{I}$ have the same marginal distributions. However, in $\mathbf{O}_{\text{MARGINSAMPLE}}$, the attributes are mutually independent, while in $\mathbf{I}$, they are often not. Definition 15 gives the distribution outliers are generated with explicitly. Thus, marginSample is closely related to the techniques sampling from a distribution (cf. Section 3.10.1).

In Sparse Regions

In (Fan et al., 2001, 2004) the distribution-based generation technique is introduced. Our foundation for the following summary is our understanding of the respective publications which do not come with an open implementation. The idea is to generate outliers close to genuine instances while generating more in sparse regions of the instance space. In (Fan et al., 2004), it is speculated that “sparse regions are characterized by infrequent values of individual features”. Based on this speculation, in (Fan et al., 2001, 2004) Algorithm 4 is proposed for the generation of outliers.

Algorithm 4 distBased.

```
Input: Number of runs
 1: for $i \in \{1, \dots, d\}$ do
 2:     $\Phi$ = unique values for attribute $i$
 3:     $\phi*$ = most frequent value of attribute $i$
 4:     $n_{\phi*}$ = number of instances with $\phi$
 5:     for $\phi \in \Phi$ do
 6:         $n_{\phi}$ = number of instances with $\phi$
 7:         for $j \in \{n_{\phi}, \dots, n_{\phi*}\}$ do
 8:             Choose $\vec{x} \in \underline{X}$ randomly
 9:             Choose $r \in \Phi \setminus (\phi \cup x^{(i)})$ randomly
10:             $\vec{o} = \vec{x}$ with value $r$ for attribute $i$
11:         end for
12:     end for
13: end for
```

In Algorithm 4, random instances are drawn from $\underline{X}$ for each possible value of each attribute. The number of instances sampled is anti-proportional to the frequency of that value. Finally, the value of the sampled instances for an attribute is replaced with some random value from that attribute. Algorithm 4 can be run several times to generate more outliers.

Minimal and Maximal Value

Algorithm 5 boundVal.

```
Input: $n_{art}$
 1: for $i \in \{1, \dots, n_{art}\}$ do
 2:     Choose attribute $j$ and $l$ randomly
 3:     $\phi_j^+, \phi_l^+$ maximal value of attribute $j$ or $l$
 4:     $\phi_j^-, \phi_l^-$ minimal value of attribute $j$ or $l$
 5:     $\phi_j^*$ = randomly choose $\phi_j^+$ or $\phi_j^-$
 6:     $\phi_l^*$ = randomly choose $\phi_l^+$ or $\phi_l^-$
 7:     Choose $\vec{x} \in \underline{X}$ randomly
 8:     $\vec{o}_i = \vec{x}$ with $\phi_j^*$ and $\phi_l^*$ for attribute $j$ and $ml$
 9: end for
```

In (Wang et al., 2009), the boundary value technique is proposed. The idea is to generate artificial outliers so that they surround the genuine instances in each attribute (see Algorithm 5). For each artificial outlier, the values of two[5] randomly chosen attributes of a randomly chosen genuine instance are replaced with the minimum or maximum of the corresponding attribute. Whether a value is replaced by the minimum or maximum of the attribute is also decided by chance.

[5] If the data set has only two attributes, our implementation replaces the values of only one attribute.

3.10.4 Real-Valued Negative Selection

The technique described next does not fit any of the previous categories. The generation is an adaption of the Negative Selection (NS) algorithm (Forrest et al., 1994) from the field of artificial immune systems. The idea of NS is inspired by T cells from the human immune system. They distinguish cells that belong to the human body (*self*) from ones that do not (*other*). In NS, one implements a set of detectors (resembling the T cells) that are then used to distinguish inlier (*self*) from outlier (*other*) instances. However, NS is usable only when the data set is representable in binary form, which is to say that each attribute takes the value either 0 or 1. Thus, in (Gonzalez et al., 2002) the Real-Valued Negative Selection (RNS) algorithm is introduced (see also González and Dasgupta (2003)). The algorithm tries to find a set of detectors that cover the real-valued instance space not occupied by inliers. Each such detector is a hyper-sphere. Figure 3.6 serves as an illustration. The green line is the boundary between inliers and outliers. The gray circles are the detectors, with the black crosses as their centers.

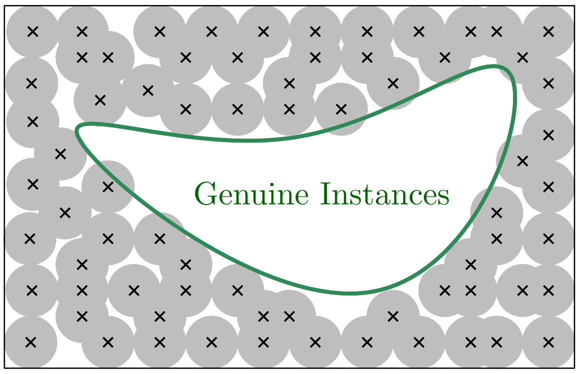


Figure 3.6: Illustration of real-valued negative selection.

In negative selection, the detectors themselves detect the outlier instances (e.g., by checking whether an instance falls into their vicinity). However, in (Gonzalez et al., 2002), the usage of the centers of the detectors as artificial outliers is proposed (see Algorithm 6). An initial set of randomly chosen detectors[6] is iteratively optimized. In each iteration, the detectors are moved away from genuine instances (*medDists* $< r$) or separated from other detectors (*medDists* $\geq r$) (see Definition 16).

Definition 16 (negSelect) *The* negSelect *technique is used to generate artificial outliers, outlined in Algorithm 6. Here, function* match$(\cdot,\cdot)$ *is given by*

$$\text{match}(\vec{a}_1, \vec{a}_2) = e^{-\frac{\|\vec{a}_1 - \vec{a}_2\|^2}{2r^2}}, \tag{3.12}$$

$\varphi_{\vec{x}}$ *by*

$$\varphi_{\vec{x}} = \frac{\sum_{\vec{n} \in \underline{N}} \vec{a} - \vec{n}}{|\underline{N}|} \tag{3.13}$$

[6] In (Gonzalez et al., 2002), it is not discussed how these are obtained. We use the unifBox technique to this end.

Algorithm 6 negSelect.

```
Input: $n_{art}$, $r$, $\eta_0$, , $\tau$, $t$, $k$, $i^+$, match$(\cdot,\cdot)$
1: $\underline{A} = n_{art}$ random detectors with age 0
2: for $i \in \{1, \dots, i^+\}$ do
3:     $\eta_i = \eta_0^{-\frac{i}{\tau}}$
4:     for $\vec{a} \in \underline{A}$ do
5:         $\underline{N} = k$-nearest neighbors of $\vec{a}$ in $\underline{X}$
6:         $\mathcal{D} =$ distances of $\vec{a}$ to $\underline{N}$
7:         $d_{0.5} =$ median of $\mathcal{D}$
8:         if $d_{0.5} < r$ then
9:             if age of $\vec{a} > t$ then
10:                Replace $\vec{a}$ by new random detector
11:            else
12:                Increase age of $\vec{a}$ by one
13:                $\vec{a} = \vec{a} + \eta_i \cdot \varphi_{\vec{x}}$
14:            end if
15:        else
16:            Set age of $\vec{a} = 0$
17:            $\vec{a} = \vec{a} + \eta_i \cdot \varphi_{\vec{a}}$
18:        end if
19:    end for
20: end for
```

and $\varphi_{\vec{a}}$ by

$$\varphi_{\vec{a}} = \frac{\sum_{\vec{a}' \in \underline{A}} \text{match}\big(\vec{a}, \vec{a}'\big)\,(\vec{a} - \vec{a}')}{\sum_{\vec{a}' \in \underline{A}} \text{match}\big(\vec{a}, \vec{a}'\big)}. \tag{3.14}$$

The parameters are $\underline{A}$, r, η_0, , τ, t, k, and i^+.

The function match$(\cdot,\cdot)$ in Definition 16 determines how well two detectors match (i.e., cover the same instance space), while $\varphi_{\vec{x}}$ is the direction in which a detector is shifted to move it away from genuine instances. The direction $\varphi_{\vec{a}}$ is used to move a detector away from other detectors.

3.10.5 Discussion

We conclude this section with a summary and a general comparison of the generation techniques presented. We have classified the techniques by their connection to the genuine instances (cf. Figure 3.3) and by the type of procedure used to generate the outliers (Sections 3.10.1 to 3.10.4). The results of our experimental study suggest that for the calibration of detection methods, artificial outliers similar to genuine instances (e.g., densAprox or skewBased) seem to be interesting (cf. Definition 2). Hence, Figure 3.3 offers a useful resource to guide the selection of a suitable generation technique.

A comparison of techniques within a specific category like *sampling from a distribution* is difficult, since generation techniques tend to differ significantly also within a category. For example, within the category just mentioned, the technique based on a simple uniform distribution (unifBox) requires only the attribute bounds to be estimated. The technique utilizing GANs (ganGen), in turn, requires a trained deep neural network. Differences like this one arise throughout the surveyed techniques and make finding general benefits or drawbacks difficult. Regardless of these difficulties, we give some general results from comparisons in the following. The category described in Section 3.10.1 comprises the highest number of techniques. Their joint idea is to fit a distribution to the data first and then generate artificial outliers by sampling from this distribution. The fitting of the distribution can be quite resource-intensive, for instance, for the ganGen technique, but it is easy to generate any amount of artificial outliers with sampling. As mentioned, not all techniques require many resources to fit the distribution, however. Techniques that generate outliers by shifting genuine instances (Section 3.10.2) or sampling instance values (Section 3.10.3) usually require less computational effort in advance of the generation of outliers. Additionally, the direct use of genuine instances tends to yield artificial outliers close to these genuine instances (cf. Figure 3.3). Sampling instance values (Section 3.10.3), has similar drawbacks and benefits but is simple to perform. The negSelect technique is the only one described in Section 3.10.4. It features a generation paradigm that differs substantially from the procedure of other techniques. The iterative optimization of the initial set of artificial outliers is resource-intensive but does not allow for the straightforward generation of more artificial outliers *a posteriori*, unlike techniques that sample from a distribution.

In summary, we find it difficult to say which procedure or connection to genuine instances is preferable. All techniques presented incorporate ideas that can be useful in certain aspects.

3.11 Filtering Techniques

Having described the existing generation techniques, we now turn to the techniques that filter generated instances for interesting ones. This description is the last building block for our general perspective on artificial outliers.

We group the filter techniques in two groups: those that use a classifier and those that compute and use some statistics. "Artificial instances" refer to instances generated, and "artificial outliers" to those resulting from filtering the artificial instances. Therefore, the artificial outliers should be interesting and close to minimal (see Definitions 2 and 3).

3.11.1 Using a Classifier

In (Fan et al., 2004), an iterative technique to filter artificial instances is proposed so that they are further from genuine ones (see Algorithm 7). In each iteration, a classifier is trained to distinguish between the genuine instances from $\underline{X}$ and the generated instances. Then, the artificial instances that are classified as genuine are replaced with newly generated instances. This process is repeated until only very few artificial instances are removed in an iteration.

Algorithm 7 Filter using a classifier (Fan et al., 2004).

```
Input: $\underline{X}$, Generation Technique, Classifier Model, $\gamma*$
 1: $\underline{A} = n_{art}$ artificial instances
 2: repeat
 3:     Train classifier with $\underline{A} \cup \underline{X}$
 4:     $\gamma = 0$
 5:     for $\vec{a} \in \underline{A}$ do
 6:         Predict class of $\vec{a}$ with classifier
 7:         if Predicted class is genuine then
 8:             Replace $\vec{a}$ with new generated instance
 9:             $\gamma = \gamma + 1$
10:         end if
11:     end for
12: until $\gamma \leq \gamma*$
```

In (Abe et al., 2006), a filter called ensemble-based minimum margin active learning is applied. It combines the ideas of query by committee and of ensembles. Several classifiers are trained one after another, each one on a sample of the genuine and the artificial instances. In the end, all classifiers are combined in an ensemble, yielding the final classifier. The filter is the sampling procedure that selects the instances used to train a new ensemble member.

Definition 17 (Filter with Query by Committee) *Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ be a set of m classifiers that have been trained one after another. Let $c^{out}(\vec{x})$ be the probability that Classifier $c \in \mathcal{C}$ classifies $\vec{x}$ as an outlier. Analogously, $c^{in}(\vec{x})$ is the probability of c classifying $\vec{x}$ as inlier. Let*

$$\text{margin}(\mathcal{C}, \vec{x}) = \sum_{c \in \mathcal{C}} c^{out}(\vec{x}) - c^{norm}(\vec{x}) \tag{3.15}$$

and

$$\text{gauss}(\mu, \sigma, \xi) = \int_{\xi}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx, \tag{3.16}$$

where Equation (3.16) *is used only to simplify Equation* (3.17). *Then, the* filter with query by committee *is as follows: An instance $\vec{x} \in \underline{Z}$ is kept with probability*

$$\text{gauss}\left(\mu = \frac{m}{2}, \sigma = \frac{\sqrt{m}}{2}, \xi = \frac{m + \text{margin}(\mathcal{C}, \vec{x})}{2}\right). \tag{3.17}$$

The function margin$(\cdot, \cdot)$ computes the disagreement among the classifiers on the class of $\vec{x}$. The function gauss$(\cdot, \cdot, \cdot)$ transforms this disagreement into a probability and is similar to the CDF of a Gaussian distribution. Thus, artificial instances for which there is much disagreement among the classifiers are kept with a higher likelihood. Note that the filtering from Definition 17 is probabilistic. That is, rerunning the filter can lead to other artificial outliers. A very similar idea is proposed in (Curry and Heywood, 2009). However, instead of the filtering from Definition 17, the so-called balanced block algorithm (Curry et al., 2007) is used.

3.11.2 Using a Statistic

Instead of using a classifier to filter artificial instances, several filtering techniques make use of a statistic computed based on the artificial and genuine instances. Definition 13 has featured the filter introduced in (Bánhalmi et al., 2007). The statistic computed comes from Algorithm 2, which checks whether an instance is a boundary instance. Any generated instance that is not a boundary instance, according to Algorithm 2, is filtered out. The filter proposed in (Neugebauer et al., 2016) has also been described already, in Algorithm 1. The statistic computed is the distance from an artificial instance to its nearest genuine neighbor. Only if this distance is greater than a certain threshold is the artificial instance deemed an outlier.

In (Davenport et al., 2006), thinning is proposed. The idea is to filter artificial instances so that the remaining ones are well spread across the whole instance space and have a distance to each other that is as large as possible. This proposal resembles the idea behind the LHS technique from Definition 1 (see Definition 18).

Definition 18 (Filter through Thinning) *Let $d_{i,j}$ be the Euclidean distance between two artificial instances $\vec{a}_i$ and $\vec{a}_j$. Then the* filter through thinning *is as follows:*

1. *Find $i \neq j$ $(i, j \in \{1, \dots, n_{art}\})$ for which $d_{i,j}$ is the smallest.*
2. *Remove the instance from $\{\vec{a}_i, \vec{a}_j\}$ which has a lower distance to its nearest neighbor.*

The filter from Definition 18 must be applied several times to ensure evenly spread artificial outliers.

Definition 19 (Filter with Unsupervised Detection) [7] *Let* $\mathrm{dect}(\cdot)$ *be an unsupervised outlier-detection technique; that is, given a set of instances,* $\mathrm{dect}(\cdot)$ *determines which ones are inlier or outlier. The* filter with unsupervised detection *works as follows: An artificial instance $\vec{a}$ is kept if either* $\mathrm{dect}(\vec{a}) =$ *outlier or* $\mathrm{dect}(\vec{a}) =$ *inlier holds. Whether "inlier" or "outlier" has to hold is a parameter of this filter.*

Note that in Chapter 5, depending on the attribute subset, $\mathrm{dect}(\cdot)$ sometimes filters artificial instances deemed an outlier and sometimes ones deemed an inlier. To obtain artificial outliers that are rather far away from genuine instances, similarly to Algorithm 7, one only needs to filter for artificial instances $\mathrm{dect}(\cdot)$ deems outlying.

[7] This filter is introduced in Chapter 5 of this thesis but described here as well for completeness.

3.11.3 Discussion

The number of techniques to filter generated instances is much smaller than the number of techniques to generate them. We categorize the known techniques in two groups: techniques from one group, use a classifier to filter generated instances, and techniques from the other group, specific statistics. Filter techniques that use a classifier are usually more time-consuming since the training of the classifiers has to happen multiple times. The filter techniques that utilize statistics are usually much faster. However, the filter using unsupervised outlier detection methods can be computationally heavy as well, depending on the detection method used.

Whether it makes sense to use a filter technique ultimately depends on the use case and the technique used to generate artificial outliers. The thinning filter from Definition 18, for instance, can be quite useful in the *casting task* use case. It renders the artificial outliers more uniformly distributed within the instance space, which can be advantageous for that use case (Steinwart et al., 2005).

3.12 Experiments

So far, we have presented our general perspective on artificial outliers, including the various techniques to generate artificial outliers in Section 3.10 in particular. We now experiment with them for insights that also extend to a practical level. We list three aims behind such experiments.

Aim 1: To our knowledge, most presented generation techniques have never been compared to each other systematically. We aim at precisely this comparison.

Aim 2: Section 3.7.2 has explained that the two use cases *casting task* and *one-class tuning* are similar. Both result in outlier detection based on classification. Thus, another aim of our experiments is to study the quality difference in the resulting detection. Since both use cases have a foundation in classification, we refer to them by the respective classifiers.

Aim 3: Some characteristics of certain types of artificial outliers in terms of the underlying data set are known. For example, consider that the outlier-detection quality with some generation techniques decreases with an increasing number of attributes (Tax and Duin, 2001; Hempstalk et al., 2008; Steinbuss and Böhm, 2017; Davenport et al., 2006). In our experiments, we also analyze these characteristics more closely; for example, how prominent such effects are.

In the remainder of this section, we first describe the workflow of our experiments. We then describe the data sets and classifiers used and discuss the parametrization of the generation techniques. Next, we describe the statistical tools we use to analyze our experimental results. We then describe general outcomes from the experiments. Finally, we analyze the results in terms of the different classifiers, the generating techniques used, and the data-set characteristics.

3.12.1 Workflow

Algorithm 8 is the workflow for our experiments. For each data set and each classifier, we use the generating techniques presented in this survey for training. We then test each classifier on all types of outliers. With *types of outlier* we refer to outliers generated with some technique as well as the genuine outliers. For instance, one type of outliers is "genuine outliers", while another is "artificial outliers generated with unifBox". The label for training or testing the classifier is whether an instance is a genuine inlier or an outlier. To evaluate a detection method, we use the Matthews Correlation Coefficient (MCC), which is particularly suited if the classes can be imbalanced (Boughorbel et al., 2017). This MCC is essentially the correlation between predicted and ground-truth instance labels. We repeat each such experiment 20 times. The code for our experiments from this chapter are publicly available.[8]

Algorithm 8 Experiment workflow.

```
Input: A set of data sets, a set of classifiers $\mathcal{C}$ and a set of generation techniques[9] $\mathcal{G}$.
1: for each $\underline{X}$ in the set of data sets do
2:     for each $c \in \mathcal{C}$ do
3:         $\underline{I}$ = inliers from $\underline{X}$
4:         $\underline{I}_{train}$ = random sample of $\underline{I}$ with 70% of $\underline{I}$s size
5:         $\underline{I}_{test} = \underline{I} \setminus \underline{I}_{train}$
6:         $\underline{O}_{genu}$ = genuine outliers from $\underline{X}$
7:         for each $\text{gen}(\cdot) \in \mathcal{G}$ do
8:             Train $c$ with $\text{gen}(\underline{I}_{train}) \cup \underline{I}_{train}$
9:             for each $\text{gen}(\cdot) \in \mathcal{G}$ do
10:                Predict class of $\text{gen}(\underline{I}_{train}) \cup \underline{I}_{test}$ with $c$
11:                Save Matthews correlation coefficient (MCC)
12:            end for
13:            Predict class of $\underline{O}_{genu} \cup \underline{I}_{test}$ with $c$
14:            Save Matthews correlation coefficient (MCC)
15:        end for
16:    end for
17: end for
```

3.12.2 Data Sets Used

The data sets we use are a common outlier-detection-benchmark data sets. We use most data sets proposed in (Campos et al., 2016). These are mostly classification data sets in which one class is deemed outlying. We exclude the data sets Arrythmia and InternetAds due to their very high number of attributes (259 and 1555). These data sets would immensely increase the runtime of our experiments. We add, however, the musk2 data sets

[8] Available at ipd.kit.edu/mitarbeiter/steinbussg/exp-artificial-outliers-FINAL-V3.zip.

[9] We represent a generation technique in this algorithm by a function gen$(\cdot)$, which has only a data set as input. Some techniques require additional inputs. See Table 3.1 for their values.

from (Dheeru and Karra Taniskidou, 2017) with a reasonable number of attributes. This addition leaves us with the data sets displayed in Table 3.6. As proposed in (Campos et al., 2016), each data set is scaled so that $\underline{X} \in [0,1]^d$, and duplicate instances are removed. To reduce the run time of our experiments, we downsample data sets with more than 1000 instances to 1000 instances. We downsampled the outlier and inlier classes such that their initial ratio remains.

3.12.3 Classifiers Used

The difference in the use cases *casting task* and *one-class tuning* from Section 3.7 is that *casting task* uses a binary classifier and *one-class tuning* a one-class classifier. There is one family of classifiers that exists in the binary case as well as in the one-class case, namely the SVM. For this reason, we use SVMs in our experiments. While the one-class SVM only needs a single class of instances (e.g., inliers) for training, the binary SVM needs two. Both SVMs use a simple separation (e.g., linear for the binary case) to perform their classification. Projecting the data into a kernel space generalizes the simple separation to take any complex form (see (Hastie et al., 2009)). The binary SVM tries to find the best separation of the two classes. One version of the one-class SVM, in turn, tries to separate all available instances from the origin of the transformed space. This trick allows the one-class SVM to train with instances from only a single class (see (Schölkopf et al., 2001)).

The binary, as well as the one-class SVM, have two different formulations. The binary SVM can be formulated as C-SVM or ν-SVM (Chang and Lin, 2001). The main difference is that they feature different parameters. While the C-SVM features a parameter $C \in (0,\infty)$, the ν-SVM features $\nu \in (0,1]$. In our experiments, we use both, as described later. The one-class SVM is formulated as described above in (Schölkopf et al., 2001) and is called ν-support classifier. In (Tax and Duin, 2001) another version, the SVDD, is formulated. The two types, however, give identical decision functions when using the Gaussian kernel (Lampert, 2009). We use the formulation from (Schölkopf et al., 2001) in our experiments with this kernel.

Algorithm 9 Hyperparameter tuning.

```
Input: $\underline{Z}$, $\nu_{range}$, $s_{range}$
 1: Set $Err_{best} = \inf$
 2: for each hyperparameter combination $(\nu, s)$ do
 3:     Train SVM with $(\nu, s)$
 4:     $Err_{Art}$ = error on artificial outliers from $\underline{Z}$.
 5:     $Err_{Genu}$ = error on genuine instances from $\underline{Z}$.
 6:     $Err = 0.5 * Err_{Art} + 0.5 * Err_{Genu}$
 7:     if $Err_{best} > Err$ then
 8:         $Err_{Best} = Err$
 9:         $(\nu*, s*) = (\nu, s)$
10:     end if
11: end for
```

The one-class tuning use case aims at finding optimal hyperparameters. For this search we use the technique introduced in (Wang et al., 2018) that is displayed in Algorithm 9. It is a grid search over hyperparameters ν and s. The values with the lowest error are chosen as the final model. As in (Wang et al., 2018) we use $\nu_{range} = \{0.001, 0.05, 0.1\}$ and $s_{range} = \{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$. This approach is referred to as *one-class*. For the binary SVM in the *casting task* use case, we have implemented two techniques. One approach is to just use a C-SVM with the default values from the respective implementation $C = 1$ and $s = \frac{1}{d}$, subsequently referred to as *binary*. The second approach is to optimize ν and s of a ν-SVM using Algorithm 9. We refer to this as *binaryGrid*. In summary, we use three types of classifiers: *binary* and *binaryGrid* for the *casting task* use case and *one-class* for the *one-class tuning* use case.

3.12.4 Generation Techniques

Table 3.1: Overview of used parameters and techniques. If applicable n_{art} is set to n_{genu}.

Technique	Suggested Parameter Value(s)	Used Parameter Value
UNIFBOX	Increase of bounds: 0%, 10%, 20%	10%
LHS	—	—
UNIFSPHERE	—	—
MANISAMP	Number of nearest neighbors: 10	10
MARGINSAMPLE	—	—
BOUNDVAL	—	—
DENSAPROX	Single Gaussian or mixture	Single Gaussian
SURREG	$\varepsilon = 0.1$	0.1
SKEWBASED	$\alpha = 2$	2
NEGSHIFT	—	—
GAUSSTAIL	—	—

We do not vary the parameters of the generation techniques but use their default values if applicable. They are listed in Table 3.1. The parameter n_{art} is always set to n_{genu} if applicable in our experiments. That is, the number of genuine and artificial instances is equal if the technique has this parameter. The bounds for the UNIFBOX technique are extended by 10%, as proposed in (Abe et al., 2006). We find this a good compromise between no extension and the 20% increase proposed in (Désir et al., 2013). We perform density estimation in the DENSAPROX technique with a multivariate Gaussian having a covariance matrix with only diagonal elements. For the SURREG technique, we choose $\varepsilon = 0.1$, as suggested by our other experiments in Section 5.6. We exclude the INVHIST and INFEASEXAM techniques from our experiments since they have been explicitly proposed for data sets with very few attributes. We have excluded BOUNDPLACE, DISTBASED, and negSelect, because of the enormous runtimes of our respective implementations, which an experienced programmer from our institution has put together. A single execution of distBased — the fastest technique of these three excluded ones — takes more than 50 seconds on a data set with 30 attributes and 650 genuine instances. These numbers roughly represent the average size of the data sets in our experiments. We have to execute each technique for 20 iterations, 16 data sets, and for training as well as for testing three classifiers in combination with 11 other generation techniques (cf. Algorithm 8). With the distBased technique, the runtime is very high because our data sets are not categorical. Thus, counting the number of occurrences of the values of all attributes in addition to the procedure to look up a new random value becomes very expensive. The requirement "low runtime" also is the reason that we have not implemented nor included ganGen. Training a GAN architecture is extremely resource-intensive.

3.12.5 Statistical Tools

A direct comparison of the MCC scores is not very useful due to the many factors influencing the scores. Hence, we want to analyze our results statistically by performing an Analysis of Variance (ANOVA) (Hartung et al., 2012). This analysis allows us to check whether and how strongly the experimental parameters affect the MCC scores. We then analyze these effects in more detail with a *post hoc* analysis (McHugh, 2011). Finally, we use Kendall's Tau coefficient (Hartung et al., 2012) to determine the effect of specific data characteristics.

Analysis of Variance

From Algorithm 8, we see that there are four parameters for a specific experiment. The classifier (c), the underlying data set ($\underline{X}$), the generation technique the classifier is trained with (gen$_{train}$), and the type of outliers the classifier is tested on (gen$_{test}$). We refer to these four as main factors. The ANOVA partitions the variation of a dependent variable, here the MCC score, according to so-called sources. There are three types of sources: the main factors just mentioned, their interactions, and the residuals. The interactions between main factors, denoted by inter$(\cdot)$, are used to account for the joint effect of several main factors, for example, if the choice of the classifier is not independent of the underlying data set (inter$(1,4)$ in Table 3.2). To explain residuals, observe that the basis of the ANOVA is regression. The residual source is the variation that cannot be accounted for using this regression. Each source except for the residual one has a specific number of levels. A level of a source is a particular value that it takes. For the main factor $c(\cdot)$ for example, the levels are *binary*, *binaryGrid* and *one-class*. To perform the ANOVA, one obtains the sum of squares attributed to the different sources from the regression model. These and their Degrees of Freedom (DF) are used to compute the F-Value. The number of DF of a source is given by the number of levels of the source minus one. For example, the classifier has three levels. Hence, the number of DF for this source is 2. With the F-Value, one can perform a statistical test, given in Hypothesis Test 1. The p-value of this test is computed using the F distribution parameterized by the number of DF — the distribution of the F-Value under the null hypothesis.

Hypothesis Test 1 (ANOVA F-test) *Let μ_i be the mean of the dependent variable for level i from a source with L levels. Then the null and alternative hypothesis of the* ANOVA *F-test are*

$$H_0 : \forall\, i, j \in \{1, \dots, L\} : \mu_i = \mu_j \tag{3.18}$$

and

$$H_A : \exists\, i, j \in \{1, \dots, L\} : \mu_i \neq \mu_j. \tag{3.19}$$

The ANOVA F-test thus checks whether the mean of at least one level is different from the mean of the other levels. In addition to this test, one can compute the partial omega squared (ω_P^2) values (Olejnik and Algina, 2003) using the ANOVA. The value for ω_P^2 gives the importance of the respective source in explaining the variation in the MCC score. A high ω_P^2 means that this source accounts for a rather large part of the variation in the MCC score.

Post Hoc Analysis

Following an ANOVA, one usually performs a *post hoc* analysis (McHugh, 2011). Any source for which the Hypothesis Test 1 is significant is analyzed in more detail. With the ANOVA F-test, one can conclude only that the mean of at least one level differs significantly from the mean of at least one other level. However, it usually is interesting for which levels this is the case. Hence, for each pair of levels in a significant source, one computes whether there is a difference or not. This procedure is the *post hoc* analysis. The pairwise tests in a *post hoc* analysis are a case of multiple testing (McHugh, 2011). Hence, p-values need to be adjusted accordingly. We use the Holm–Bonferroni method (Holm, 1979) to this end. Usually, a simple Student's t-test is used to compare the means of two levels (McHugh, 2011). However, since there are many significant interactions in our ANOVA result, the assumptions behind the Student's t-test are usually violated. Hence, we use a non-parametric alternative: the Mann–Whitney U test (Mann and Whitney, 1947).

Hypothesis Test 2 (Mann-Whitney U Test) *Let L_i and L_j be the distribution function of the dependent variable within Levels i and j of a source. Then the null and alternative hypothesis of the Mann-Whitney U test is*

$$H_0 : L_i(x) = L_j(x)\ \forall\, x \in [0, 1] \tag{3.20}$$

and

$$H_A : L_i(x) > L_j(x) \text{ or } L_i(x) < L_j(x)\ \forall\, x \in [0, 1]. \tag{3.21}$$

That is, one variable is stochastically larger or smaller than the other one.

A level that is stochastically greater than another indicates a preference. To illustrate, if the one-class classifier is stochastically greater than the binary one, it usually yields higher MCC scores. Along with the pairwise test from Hypothesis Test 2, we provide level-wise means, medians, and density plots of the MCC score when applicable. These measures and plots indicate the direction of the stochastic order. For greater clarity, the result of pairwise tests can be presented in the form of letters (Piepho, 2004). We use the letters “a” to “z”. Every level is assigned a combination of letters, often only a single one. If two levels share a letter, the respective test is not significant; that is, H_A from Hypothesis Test 2 cannot be accepted.

Kendall’s Tau Coefficient

When analyzing the results of our experiments in terms of the different $\underline{X}$, we are interested in the effects of the number of instances n_{genu} and the number of attributes d. Hence, we are interested in the dependency of the MCC score on n_{genu} or d. A common estimate for a monotonic relationship between two random variables is Kendall’s Tau (τ). The situation of $\tau = 0$ indicates that there is no dependency, $\tau > 0$ stands for a joint increase, and $\tau < 0$ indicates that an increase in one variable leads to a decrease in the other. We make use of the Tau test formalized in Hypothesis Test 3 to test whether the estimated τ is significant.

Hypothesis Test 3 (Tau Test) *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be two random variables with* $\tau = \tau_0$. *The null and alternative hypothesis of the tau test are*

$$H_0 : \tau_0 = 0 \quad \textit{and} \quad H_A : \tau_0 \neq 0. \tag{3.22}$$

Similarly to the previous test, we have to account for multiple testing. We again apply the Holm–Bonferroni method (Holm, 1979).

3.12.6 Performing the Analysis of Variance

The factors in our experiments have many levels: $c(\cdot)$ has 3; $\underline{X}$, 16; gen$_{train}$, 11; and gen$_{test}$, 12. We think that this large number of levels and factors makes a full ANOVA with all possible interactions difficult to interpret. To reduce the number of levels, we use an aggregated type of gen$_{test}$: the genuine outliers from $\underline{X}$ ($\underline{O}_{genu}$) and the median of all generating techniques ($\underline{O}_{art}$). We aggregate the result on all generation techniques since we are not very interested in the detection quality of a single type of artificial outliers. A low detection quality could, for example, mean simply that these outliers are easy to detect and not offer any insights into the ability of the specific classifier to detect various types of outliers. The results of the ANOVA are displayed in Table 3.2.

The F-Tests for each source yield a highly significant result (p-values $< 6 \cdot e^{-10}$). Thus, at least one mean value within the different levels of each source significantly differs from the other levels. We conclude that each source listed in Table 3.2 determines to some extent whether the MCC score of an experiment is high or low on average. The significance of all possible interactions means that the main factors influence each other. For example, the choice of a classifier type has an impact on the generation technique for outliers that results in a high MCC score on average. This finding coincides with what is hypothesized in (Hastie et al., 2009) and (Steinwart et al., 2005). Note that the necessary assumption for ANOVA of standard normal residuals with equal variance is not fully met in our case. Thus, the p-values of the F-Test and the ω_P^2 values might not be exact. Our subsequent *post hoc* analysis does, however, confirm the tests regarding the main factors.

Table 3.2: Four-Way ANOVA from our experiments.

Source	Sum of Squares	DF	F-Value	ω_P^2
$^1\underline{X}$	366.82	15	2319.90	0.62
$^2\text{gen}_{train}$	265.90	10	2522.51	0.54
inter$(1,3)$	191.31	15	1209.93	0.46
inter$(1,2)$	182.84	150	115.63	0.45
inter$(1,2,3)$	166.33	150	105.19	0.43
$^3\text{gen}_{test}$	148.82	1	14117.71	0.40
inter$(1,2,4)$	124.99	300	39.52	0.35
inter$(2,4)$	106.72	20	506.20	0.32
inter$(2,3)$	87.96	10	834.43	0.28
inter$(1,2,4,3)$	61.52	300	19.45	0.21
inter$(1,4)$	44.32	30	140.13	0.17
inter$(1,4,3)$	18.25	30	57.71	0.07
inter$(2,4,3)$	17.54	20	83.22	0.07
$^4c(\cdot)$	1.26	2	59.59	0.01
inter$(4,3)$	0.45	2	21.34	0.00
Residuals	211.50	20064		

3.12.7 Classifier Comparison

From the ω_P^2 values in Table 3.2, we see that the type of classifier ($c(\cdot)$) itself has a rather small impact on the variation of the resulting MCC scores. As such, it is not vital for explaining a high or low MCC score. Interestingly, some interactions involving the classifier are ranked much higher, the one of classifier and gen$_{train}$, for example. Hence, it is more important that the classifier and the generation technique for outliers fit. However, we are interested in the exact differences between the types of classifiers. Table 3.3 features the results of the respective pairwise Mann-Whitney U test.

Table 3.3: Comparison of different types of classifier.

$c(\cdot)$	Mean	Median	Test Result
binaryGrid	0.31	0.24	b
one-class	0.30	0.24	b
binary	0.29	0.23	a

We see that the one-class and binaryGrid classifiers have the same group letter. Hence, there seem to be few reasons to prefer one over the other. The binary classifier is in a group alone, indicating a significant difference from the former two. The mean and median MCC score are lowest. However, the difference is tiny. This result differs somewhat from the results of the small study performed in (Davenport et al., 2006). The results in (Davenport et al., 2006) suggest that the *casting tasks* use case (binary/binaryGrid classifier) is preferable. To further elaborate on the difference in distributions of the classifier types, we visualize the estimated probability density of the MCC score for the different classifier types in Figure 3.7. The figure visually supports that the differences are not substantial, but the binaryGrid or one-class classifier is more likely to result in high MCC scores.

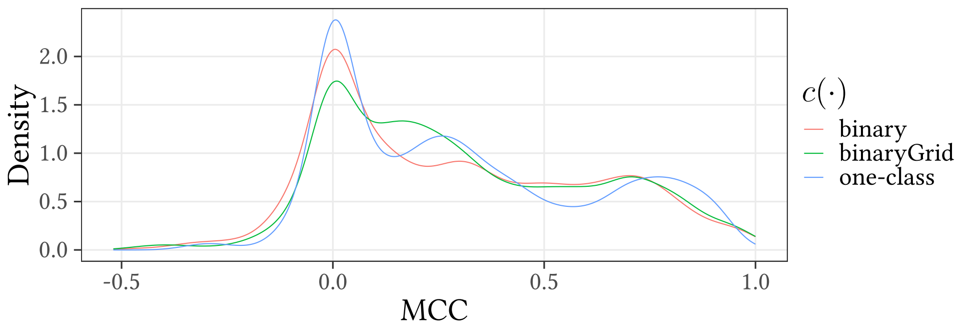


Figure 3.7: Density of MCC score with different classifier types.

3.12.8 Comparison of Generation Techniques

The ω_P^2 values in Table 3.2 for the two factors that relate to generation techniques, gen_{train} and gen_{test}, are quite high — for gen_{train} in particular. Thus, they account for a large part of the variation in the MCC score. The interpretation of the generation techniques for gen_{train} and gen_{test} differ greatly. We start with the results in terms of gen_{train} and then analyze the results regarding gen_{test}. Figure 3.8 displays the MCC score probability density regarding the levels of gen_{train} and gen_{test}.

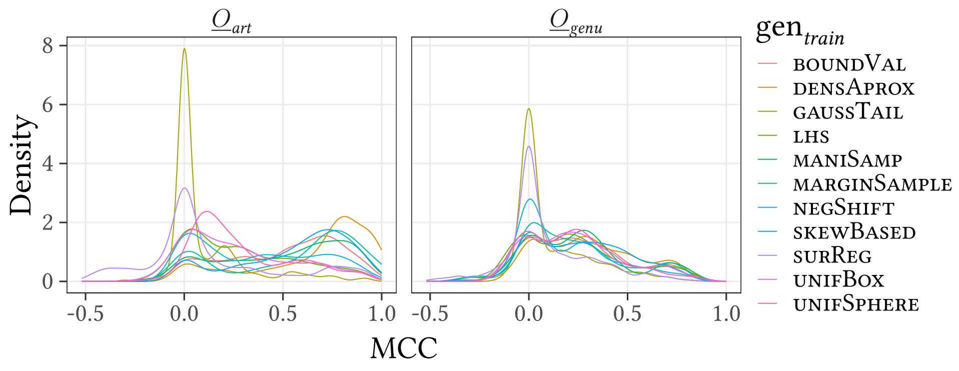


Figure 3.8: Density of MCC score with different generation techniques.

Artificial Outliers for Training

The ω_P^2 of gen$_{train}$ is the second highest in Table 3.2. Thus, certain significant differences in detection quality appear when using different generation techniques to train the classifiers. Table 3.4 lists the results of the pairwise Mann-Whitney U tests. The techniques boundVal and maniSamp form a group, and unifBox, lhs, unifSphere and negShift form one as well. The elements of the second group, in particular, are conceptually quite similar. For example, three of the four techniques spread outliers uniformly. The techniques with the highest mean and median are densAprox and skewBased. Both try to generate outliers similar to genuine instances. The densAprox technique does this quite literally. Hence, we conclude that this is generally a useful technique for generating outliers to train classifiers. Somewhat contradictory to this conclusion, however, is that the surReg technique, which also generates outliers similar to genuine instances, shares the lowest mean and median with gaussTail. From Figure 3.8 we see that both techniques often result in an MCC score close to zero. Hence, training the classifier using artificial outliers generated by surReg or gaussTail seems to result in a rather low detection quality. For the gaussTail technique, we think that this is because the generated outliers are too far from genuine instances to be interesting (cf. Example 4). The attribute bounds heavily influence the distribution of outliers generated by the surReg technique. This influence might lead to uneven coverage of the instance space around genuine instances. It might also be that instances generated with surReg are too close to genuine ones to be interesting, which would explain the low MCC score when outliers generated with surReg are used to test a classifier (cf. Table 3.5).

Table 3.4: Comparison of artificial outliers for training.

gen$_{train}$	Mean	Median	Test Result
densAprox	0.47	0.49	b
skewBased	0.41	0.42	e
marginSample	0.39	0.34	a e
boundVal	0.37	0.35	a
maniSamp	0.37	0.33	a
negShift	0.28	0.22	d
lhs	0.26	0.21	d
unifBox	0.26	0.21	d
unifSphere	0.25	0.21	d
gaussTail	0.11	0.00	c
surReg	0.11	0.00	f

Artificial Outliers for Testing

Since we aggregate all artificial outlier generation-techniques for the ANOVA, gen_{test} has only two values, $\underline{O}_{art}$ and $\underline{O}_{genu}$. Hence, we can conclude immediately that there is a significant difference in the mean MCC score of the two (cf. Table 3.5). This difference implies that there is quite a gap between the quality we assign to a detection method when we evaluate it with the artificial outlier types presented or the labeled ground truth outliers from the benchmark data sets. This is also clearly visible in Figure 3.8. We hypothesize that this is mainly because most artificial outliers are much simpler to identify as such. For example, artificial outliers generated with the unifBox technique tend to be quite far from genuine instances and are hence trivial to classify as outlying. Thus, when using artificial outliers to assess the quality of an outlier-detection method, one should consider how difficult the generated outliers generally are to detect. We also think that using several types of outliers (i.e., generated with different techniques) offers much useful insight into the performance of outlier-detection methods.

Table 3.5: Comparison of artificial outliers for testing.

gen_{test}	Mean	Median	Test Result
gaussTail	0.68	0.88	c
unifBox	0.65	0.84	i
lhs	0.64	0.82	d
unifSphere	0.62	0.81	d
densAprox	0.29	0.21	a
boundVal	0.29	0.28	ab
maniSamp	0.29	0.19	b
skewBased	0.28	0.09	e g
$\underline{O}_{genu}$	0.21	0.18	g
negShift	0.21	0.06	f
marginSample	0.20	0.12	e
surReg	0.05	0.00	h
$\underline{O}_{art}$	0.38	0.35	

Although we have aggregated the generation techniques used for testing the classifiers when performing the ANOVA, we are nevertheless interested in the differences of the generation techniques presented. Note that a high MCC value here means that the outliers generated are generally easy to detect. Table 3.5 lists the results of corresponding pairwise Mann-Whitney U tests. We listed the mean and median of the aggregated version $\underline{O}_{art}$ as references. Techniques used to test the classifiers do not group much, only lhs and unifSphere are in one group. We also observe that the ranking of generation techniques in Table 3.5 is to some extent inverse to the one in Table 3.4. For example, gaussTail is listed first in Table 3.5 but second-to-last in Table 3.4. We think that this listing further supports our previous hypothesis on the successful generation techniques to train a classifier. Training a classifier with outliers that are somewhat similar to the genuine instances, and hence more difficult to detect, results in a better detection method.

3.12.9 Data Characteristics

From the ANOVA results in Table 3.2, we see that the underlying data set $\underline{X}$ is quite important to determine if the MCC score of an experiment is rather high or low on average. A more detailed analysis regarding the effect of $\underline{X}$ can be found in Table 3.6 with the pairwise Mann-Whitney U test. Some data sets share a letter and hence do not allow for the acceptance of the alternative hypothesis that one is stochastically larger than the other one, but most are in different groups.

Table 3.6: Comparison of different data sets.

$\underline{X}$	n_{genu}	d	Mean	Median	Test Result
PageBlocks	1000	10	0.61	0.67	j
Ionosphere	351	32	0.48	0.62	g
Stamps	340	9	0.42	0.43	e
Glass	214	7	0.42	0.39	e
KDDCup99	1000	40	0.38	0.31	h
Wilt	1000	5	0.33	0.21	bc l
Cardiotocography	1000	21	0.32	0.28	d
Pima	768	8	0.29	0.21	cd
Annthyroid	1000	21	0.28	0.25	c
SpamBase	1000	57	0.23	0.18	kl
ALOI	1000	27	0.22	0.06	ab
Parkinson	195	22	0.22	0.22	k
WPBC	198	33	0.16	0.08	f i
musk2	1000	166	0.16	0.00	i
HeartDisease	270	13	0.15	0.13	a
Hepatitis	80	19	0.13	0.09	f

We hypothesize that most of the difference in detection quality is due to the distribution of the different data sets. However, the numbers of genuine instances or attributes also have an effect. We also think that the generation technique used to train the classifier has a strong influence on this effect. Thus, for each level of gen$_{train}$, we estimate τ between the MCC and the number of attributes d as well as the number of genuine instances n_{genu}. For each τ we also perform a tau test. The results are displayed in Table 3.7, and those with a significant τ (p-value < 0.05) are in bold. The p-value is abbreviated as p_{val}.

Table 3.7: Correlation with d and n_{genu}.

gen_{train}	$\hat{\tau}\ d$	$p_{val}\ d$	$\hat{\tau}\ n_{genu}$	$p_{val}\ n_{genu}$
boundVal	**-0.13**	0.00	0.03	0.21
densAprox	**0.05**	0.00	**0.21**	0.00
gaussTail	**-0.07**	0.00	**0.05**	0.03
lhs	**-0.24**	0.00	-0.04	0.08
maniSamp	-0.01	1.00	**0.17**	0.00
marginSample	0.01	1.00	**0.10**	0.00
negShift	**-0.23**	0.00	0.01	0.49
skewBased	-0.03	0.16	**0.17**	0.00
surReg	**-0.06**	0.00	**0.28**	0.00
unifBox	**-0.28**	0.00	-0.04	0.07
unifSphere	**-0.26**	0.00	-0.03	0.11

The τ regarding d tend to be negative, indicating a decreasing effect on the MCC score for an increasing number of attributes. This relation seems to be particularly strong with the unifBox technique. Most of the τ regarding n_{genu} are significant and positive. Hence, if a higher number of instances affects the resulting MCC score at all, the effect is usually a positive one.

3.12.10 Summary of Experiments

A core insight from our study is that there are considerable differences in the outlier-detection quality for different generation techniques and data sets. When used to train a classifier, the overall best performing technique has been densAprox, with a median MCC of 0.49. The worst ones have been gaussTail and surReg, both with a median MCC of 0. This result is comparable to those obtained by random guesses. The data sets form only a few groups with no significant difference in terms of the overall MCC to other data sets. However, there are significant differences between the groups. For example, with the Ionosphere data set from the group with letter “g”, the median MCC is 0.62, while it is only 0.08 with the WPBC from the group with letters “fi”. All interactions between the main factors are significant. Thus, the choice of a generation technique in a specific scenario is not reducible to, for example, “densAprox performs best”. Depending on the classifier a scenario requires or on the data set given by the scenario, different techniques might be suitable. This realization has motivated us to propose a three-step process, displayed in Figure 3.9, in order to choose a generation technique in a specific scenario. The steps are based on the results of our experiments. They help to make the necessary decisions when the goal is to detect outliers with the help of artificial outliers.

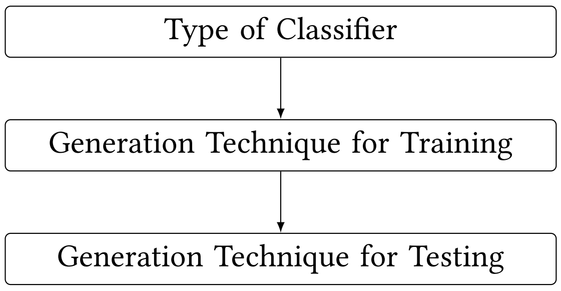

Type of Classifier
Generation Technique for Training
Generation Technique for Testing

Figure 3.9: Process to choose outlier-generation technique.

Step 1: Type of Classifier

When the ultimate goal is a method to detect outliers, any of the use cases *casting tasks* and *one-class tuning* is applicable. In our experiments, we have found that the *one-class* or *binaryGrid* classifier yield similar MCC scores. However, in a specific scenario, the huge massive of binary classifiers available for the *casting task* can be advantageous. One example is when the outlier-detection result should be easily interpretable. A decision tree might then be a better fit than a one-class SVM.

Step 2: Generation Technique for Training

One can now check for outlier generating techniques that are more suitable for the classifier chosen in Step 1. The experimental results displayed in this survey may be beneficial for this but are not necessarily sufficient to this end. Observe, however, that this article does not explicitly feature the result in every useful representation, to ensure that this survey still has a reasonable length. The full results are available, however, in combination with our code. Others can also use the code to test further combinations; this may be particularly useful when new types of classifiers become available. In addition to the type of classifier used, the data set of the scenario is of importance. One can, for example, check whether this data set is similar to one of the data sets from our experiments, or if the number of attributes and genuine instances is high or low. Depending on these two factors (classifier and data set), one can then choose the best-suited outlier-generation technique to train the classifier.

Step 3: Generation Technique for Testing

We have seen in Section 3.12.8 that one needs to be careful when assessing the quality of outlier detection using artificial outliers. We think that this assessment, nevertheless, offers useful insights into the outlier-detection quality. In a real-world scenario, there might be some knowledge of potential genuine outliers available. Consider a system administrator who has a rough idea of the distribution of possible outliers. Suppose further that this distribution is somewhat similar to the distribution of artificial outliers generated by the NEGSHIFT technique. Detection quality in terms of artificial outliers generated with NEGSHIFT is then clearly a reliable estimate for the detection quality of genuine instances. However, if there is no such knowledge, which might be the much more likely case, we conclude that one of the quite general and uninformative techniques to artificial outliers, like LHS, could be well-suited. To gain a better feel for which types of outliers are and are not well detected, a quality assessment using a variety of the generation techniques described might also be suitable. However, we leave a systematic study of this idea to Chapter 4 because this is not straightforward at all, and it goes well beyond the scope of this chapter.

3.13 Chapter Summary

By definition, outliers are instances that are rarely observed in reality, so it is difficult to learn anything with them. Various techniques to generate artificial outliers have been proposed to compensate for this shortage of data. This chapter is a survey of such techniques. As a first step, we have connected the field of artificial outliers to other research fields. This step allows us to narrow down the field of artificial outliers somewhat. The generation techniques described next represent different ways to generate artificial outliers. They form separate groups, depending on the similarity of the generated instances to genuine ones or on the general generation concept. All this fulfills our aim of having a general perspective on artificial outliers.

Our experiments confirm the hypothesis of some authors that, for the *one-class tuning* or *casting task* use case, artificial outliers similar to genuine instances seem to be performing well. That is, the outlier detection performance is high. The experiments also confirm that this performance heavily depends on the setting (e.g., the data set used). In terms of the use cases themselves, our experiments suggest that there are no distinctive differences in outlier detection performance. Analyzing the effect of some data set characteristics with different generation techniques confirms that these can heavily influence outlier-detection performance.

To this end, we have also developed a decision process, building on the results of our experiments that guide the choice of a proper generation technique. In other words, the process targets at finding a generation technique that yields high outlier-detection quality.

The study in this chapter has featured a great variety of techniques for the generation of artificial outliers based on genuine instances. The experimental study we have conducted is a basis for the decision-making process towards a well-performing outlier-generation technique. As such, this study is likely to support individuals from diverse fields when developing advanced techniques for the generation of artificial outliers.

4 Benchmarking Unsupervised Detection

So far, we have given a structured overview of existing techniques to generate outliers. We categorized them into a few techniques that do so without explicitly using genuine instances (Section 2.3) and the majority of techniques that do use genuine instances (Chapter 3). From the literature, the primary use for outliers generated with techniques from the second category is the calibration of detection methods. The evaluation of detection methods (i.e., comparing their performance to each other or some ideal performance) has been done rarely with generated outliers. Nonetheless, we think generated outliers can be extremely useful in this sense, which is the reason we address such an evaluation in this chapter specifically. We focus on unsupervised outlier-detection methods since we deem a comparison in terms of what types of outliers can be detected by which detection method of upmost importance in this scenario. For example, to get a suite of a few detection methods that can detect various types of outliers well.[1]

Outliers tend to be rare and are not precisely defined (cf. Section 1.1). These two properties render the evaluation of detection performance, and hence comparisons of outlier-detection methods difficult. Researchers tend to use benchmark data sets with labeled instances to this end (Domingues et al., 2018; Goldstein and Uchida, 2016; Campos et al., 2016; Emmott et al., 2013, 2015). Since the data sets often are from some real-world classification problems, identifying the characteristics of outliers and seeing how these characteristics compare to outliers from another data set can be very difficult. Often one can only approximate the process generating outliers or inliers. See Example 5 and Section 2.1.

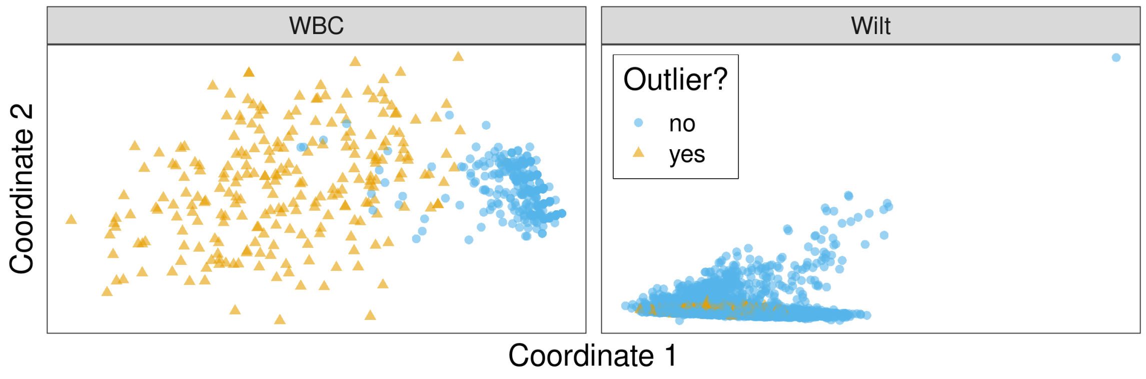


Figure 4.1: Result from multidimensional scaling.

[1] The remainder of this chapter is almost identical to (Steinbuss and Böhm, 2020a), which is submitted for review. Adjustments are to ensure consistency for this dissertation.

Example 5 *The* WBC *(Campos et al., 2016) data set comprises instances on benign and malignant (=: outliers) cancer. The* Wilt *(Campos et al., 2016) data set holds satellite images of diseased trees (=: outliers) and other land cover. Figure 4.1 displays two-dimensional representatives of both data sets obtained with multidimensional scaling (Hastie et al., 2009). This scaling aims at keeping the pairwise distances from the original data for the lower-dimensional representatives. The figure illustrates that the outliers from both data sets feature very different characteristics in terms of their pairwise distance (i.e., they are of a very different type). The outliers from WBC are somewhat distant from each other and the inliers in particular. In the Wilt data set, outliers are very close to each other and the inliers.*

Clearly, an improvement to the situation described in Example 5 is achievable by using the techniques for generating outliers described in Chapter 3. Since the generating procedure for outliers is known and can be adopted, artificial outliers have similar characteristics. However, using augmented data (i.e., inliers are genuine and outliers artificial) for benchmarks does not allow for precise information on the characteristics of outliers. For example, think of the case when generating outliers from a uniform distribution. While most of the outliers will be very different from any inlier, some outliers might be remarkably similar to inliers. Without precise knowledge on the inliers, any measure for the similarity of generated outliers to inliers remains an estimation. For this reason, we focus in this chapter on the generation of entirely artificial data. Thus, we have to generate outliers and inliers. To this end, we develop and perform a benchmark with the explicit objective of the artificial data being realistic and featuring outliers with insightful characteristics. These properties then give way to interpretations beyond pure detection accuracy. An example is how well detection methods handle local outliers. We refer to artificial outliers of a specific type (e.g., local ones) as "characterizable".

4.1 Challenges for Artificial Benchmarks

A major issue with existing outlier detection benchmarks is the absence of precise ground truth (cf. Section 2.1). This ground truth includes information on whether an instance is an outlier or not. However, it should also give information on the characteristics of the outliers. For example, information on the different types of outliers that are within the data. Due to the known generation technique for artificial data, it is feasible to obtain a ground truth that gives exact information on the types of outliers. However, not with all generation techniques, precise ground truth is automatically available. Think of generating outliers with one of the rather complex generation techniques described in Chapter 3 (e.g., ganGen or negShift). The precise characteristics of such outliers can be challenging to determine — just like with genuine outliers.

Another challenge is coverage. Similar to existing benchmarks, we are interested in comparisons of detection performance not just for a specific data domain, like the cancer types from WBC in Example 5, but comparisons across many diverse domains. Existing benchmarks (Domingues et al., 2018; Goldstein and Uchida, 2016; Campos et al., 2016; Emmott et al., 2013, 2015) approach this by using many data sets from different domains. With artificial data, coverage of several domains is challenging. Data generators for a specific domain are usually handcrafted and hence difficult to compare to each other. To illustrate, in (Barse et al., 2003), data from the fraud detection domain is generated by simulating users, while in (Downs and Vogel, 1993) data from a plant is generated by modeling chemical processes. Artificial data generators that are domain agnostic, like MDCGen (Iglesias et al., 2019), need to be parameterized. For example, the number of clusters or their form needs to be specified. To our knowledge, there currently is no widely accepted way to choose these parameters so that the resulting data sets cover different domains sufficiently.

4.2 Contributions Towards Artificial Benchmarks

The idea central to this chapter is to use a real-world data set as a basis for generating an artificial one. We propose to generate inliers and outliers, but differently: Artificial inliers follow a model of the existing real inliers. Outliers are in line with a characterizable deviation from this model. This fundamental design decision has various ramifications: The inliers are realistic, and repeating the process for different real-world data sets also gives way to good coverage (i.e., consideration of different domains). On the other side, generating outliers similar to the ones labeled as such in real-world data would not improve the insights obtainable from a respective benchmark by much. Recall that instances labeled as outliers in the real-world data may be of any type (cf. Example 5), which synthetic reconstructions would exhibit as well. Thus, we propose to generate outliers as a characterizable deviation from the model of inliers. In our evaluation, we use the real-world data sets from a recent benchmark on unsupervised outlier detection (Campos et al., 2016) to this end. They have already been used for a broad benchmark and cover different domains. Additionally, the labels existing in such data are handy for our study: First, we can compare the results from our artificial benchmark with the ones on real-world data. Second, we can assess the realness of our artificial data.

To have a sophisticated ground truth, we demand that for any generated instance — be it outlying, be it inlying — we have access to its probability density. This access allows for the introduction of several ideal outlier scorings. For example, one is similar to the Bayes error rate (Tumer and Ghosh, 1996) from supervised classification tasks. This rate gives insight into the prediction error that is unavoidable. In our case, this error comes from generated outliers indistinguishable from inliers.

Our first contribution is formalizing the generic process sketched so far. As a second contribution, we perform an extensive benchmark featuring different characterizable deviations from inliers. To this end, we present and use three such instantiations of our generic process: one for local outliers, one for outliers in the dependency structure of attributes, and one for global outliers. Our third contribution is to validate the realness of our artificial data through experiments. They confirm that the instantiations just mentioned do result in artificial data close to the corresponding real-world data set.

4.3 Organization of This Chapter

The remainder of this chapter is structured as follows. In Section 4.4, we propose a general process to generate realistic artificial data. In Section 4.5, we introduce three instantiations of this process for our benchmark. In Section 4.6, we describe the design of our experiments. In Section 4.7 and Section 4.8, we discuss our results and Section 4.9 concludes this chapter.

4.4 The Generic Process

In this section, we formalize the generic process we propose for realistic artificial benchmarks for unsupervised outlier detection. First, we introduce some notions, then our requirements regarding the instantiations of our generative process. We then introduce our ideal scorings and provide an overview of the process.

4.4.1 Special Notion

A generative model $m \in \mathcal{M}$ is a description of a set of instances that allows for the generation of artificial instances. $\mathcal{M}$ is the set of all possible models, used here for purely notational purposes. A generative model in our context is associated with the four functions

$$\text{fit}\colon \mathbb{R}^{n_{genu} \times d} \to \mathcal{M}, \tag{4.1}$$

$$\text{gen}\colon \mathcal{M} \to \mathbb{R}^{n_{genu} \times d}, \tag{4.2}$$

$$\text{dens}\colon \mathbb{R}^{n_{genu} \times d} \times \mathcal{M} \to \mathbb{R}^{n_{genu}} \quad \text{and} \tag{4.3}$$

$$\text{modify}\colon \mathcal{M} \to \mathcal{M}. \tag{4.4}$$

The function fit$(\cdot)$ creates the generative model from a set of instances and gen$(\cdot)$ generates instances from the generative model. Based on a generative model, dens$(\cdot,\ \cdot)$ returns the density of a set of instances. The function modify$(\cdot)$ modifies a generative model so that the resulting model can generate outliers that are characterizable.

In this chapter the function

$$\text{dect}\colon \mathbb{R}^{n_{genu} \times d} \to \mathbb{R}^{n_{genu}} \tag{4.5}$$

denotes an unsupervised outlier detection method. It outputs a score, given a set of instances. The scores must have a meaningful ordering in terms of outliers. For example, with the Local Outlier Factor (LOF) (Breunig et al., 2000) a high score indicates outliers.

4.4.2 Requirements on Generation

In Section 4.2, we have argued that fitting the labeled inliers *and* the outliers from real-world data is not very insightful. Hence, in this benchmark, we reconstruct only the inliers by fitting a generative model to them. Outliers are then generated based on a *characterizable* deviation from this model.

Section 2.2.3, described already existing techniques for the artificial reconstruction of a data set (basically $\text{fit}(\cdot)$ and $\text{gen}(\cdot)$). To decide on a suitable one for our process, we gather requirements that it must fulfill.

R1: Be applicable to many kinds of real-world data.

R2: Generate realistic artificial data.

R3: Feature a generation that is comprehensible.

R4: Give access to the density of generated instances.

R5: Allow for generating characterizable outliers.

R1 and R2 are essential to achieve good coverage of different data domains and R3 to R5 for a proper interpretation of the results from a benchmark. Except for R4, the requirements do not allow for a rigid definition. Terms like "many kinds of real-world data", "realistic artificial data" or "comprehensibility" are subjective. To simplify the search for a suitable technique in terms of R2, Section 4.7 will showcase a possible approach to assess the realness of synthetic reconstructions.

We have found that statistical distributions like Gaussian mixtures usually fulfill the requirements well — all our instantiations we describe in Section 4.5 are of this kind. Naturally they allow for access to the density of instances (R4) and are applicable to any real-world numeric data (R1). In Section 4.5, we will show that the ones we use also allow for characterizable outliers (R5) and in Section 4.7 that they are reasonably realistic (R2). Finally, models (m) that describe statistical distributions are well understood (R3).

4.4.3 Ideal Scores

Definition 20 introduces our three ideal scorings Classify (C), Inlier Density (RD), and Overall Density (OD).

Definition 20 (Ideal Scorings) *Let* $\text{dens}_{in} \in \mathbb{R}$ *be the density of an instance with regard to the distribution of inliers and* $\text{dens}_{out} \in \mathbb{R}$ *the one with regard to the distribution of outliers. Then the ideal scores for this instance are*

$$\text{RD} = \text{dens}_{in}, \tag{4.6}$$

$$\text{OD} = \xi \cdot \text{dens}_{out} + (1-\xi) \cdot \text{dens}_{in} \quad \textit{and} \tag{4.7}$$

$$\text{C} = \frac{\xi \cdot \text{dens}_{out}}{(1-\xi) \cdot \text{dens}_{in}}, \tag{4.8}$$

where ξ *is the frequency of outliers.* RD *and* OD *give outliers lower scores, and* C *assigns them higher scores.*

Each scoring gives insights regarding the characteristics of the generated data. C represents an ideal classifier that knows both probability distributions — similarly to the idea of the Bayes error rate (Tumer and Ghosh, 1996). The nominator is proportional to the probability of an instance being an outlier and the denominator to the one being inlier. Thus, a score higher than 1 is a prediction for outliers and a lower score for inliers. The score gives insight regarding the possibility of distinguishing between outliers and inliers. This is similar to the *difficulty* of an outlier detection problem introduced in (Emmott et al., 2015, 2013). Since C emulates a supervised scenario (where outliers and inliers are available), detection performance achievable in an unsupervised setting might be much lower.

RD and OD are both densities. The idea is that unsupervised outlier detection is very close to density estimation. The existence of many density-based methods for outlier detection and also the firm connection of distance-based ones to density (Zimek and Filzmoser, 2018) support this. RD is the density regarding the distribution of inliers. Hence, any instance that is different from inliers will have a low score. OD is the overall density of the data (i.e., regarding the outliers and inliers). The difference between RD and OD is that highly clustered outliers will have a low score in RD but not in OD. However, OD is much closer to the unsupervised setting in which it is tough to characterize inliers on their own. Hence, in particular, OD can give insights into an ideal unsupervised scoring.

4.4.4 Overview of the Process

Algorithm 10 gives an overview of the benchmark process with a specific instantiation of the generation process for artificial data (i.e., $\text{fit}(\cdot)$, $\text{gen}(\cdot)$, $\text{dect}(\cdot)$ and $\text{modify}(\cdot)$). The input of the algorithm is a set of real-world data sets, multiple unsupervised detection methods, and the frequency of outliers (ξ). For each data set, the process fits a generative model to the inliers (line 3). In line 4, the process modifies this model to obtain a model for generating outliers. In lines 5 – 7, both models are used to create the artificial data set. Then the densities from both models are used to compute the ideal scorings in lines 8 – 10. Following this, the process applies each detection method to the artificial data (line 12) and computes their detection performance (line 13). In line 14, the process determines the relationship to the ideal scorings (e.g., by computing the correlation of the scorings).

4.5 Our Instantiations

Here we introduce our instantiations to the generic process, first for local outliers, then for outliers in the dependency structure and lastly for global outliers.

4.5.1 Local Outliers

Local outliers are outlying “relative to their local neighborhoods [...]” (Breunig et al., 2000). Here we define a local neighborhood to be a cluster. More specifically, we follow the technique from (Milligan, 1985) to generate outliers in artificial data from Gaussian Mixtures. The $\text{fit}(\cdot)$ function returns the parameters of a Gaussian Mixture: the number of components $G \in \{1, 2, 3, \dots\}$, the mixing proportion $\pi \in [0, 1]^G$, the mean vector of each component $\mu_i \in \mathbb{R}^d$ and the covariance matrix of each component $\Sigma_i \in \mathbb{R}^{d \times d}$. The model for generating outliers has the same parameters, but the covariance matrix is scaled with

Algorithm 10 Overview of our benchmark process.

```
Input: $\underline{X}$, set of dect$(\cdot)$ and $\xi \in [0,1]$
 1: for every $\underline{X}$ do
 2:     $\underline{I}$ = Inliers from $\underline{X}$
 3:     $m_{in} = \text{fit}(\underline{I})$
 4:     $m_{out} = \text{modify}(m_{in})$
 5:     $\underline{S}_{in} = (1-\xi) \cdot n_{genu}$ instances with gen$(m_{in})$
 6:     $\underline{S}_{out} = \xi \cdot n_{genu}$ instances with gen$(m_{out})$
 7:     $\underline{S} = \underline{S}_{in} \cup \underline{S}_{out}$
 8:     $\text{dens}_{in} = \text{dens}(\underline{S},\ m_{in})$
 9:     $\text{dens}_{out} = \text{dens}(\underline{S},\ m_{out})$
10:     Compute ideal scorings
11:     for every dect$(\cdot)$ do
12:         $\vec{s}_{\text{dens}} = \text{dect}(\underline{S})$
13:         Compute detection performance
14:         Determine relation of $\vec{s}_{\text{dens}}$ and ideals
15:     end for
16: end for
```

$\alpha > 1$ to generate the local outliers. This is, $\widetilde{\underline{\Sigma}}_i = \alpha \underline{\Sigma}_i$ where $i \in \{1, \ldots, G\}$. Instances generated with the increased covariance matrix are still close to inliers generated from the cluster with the same π_i. However, they will often be outlying the inliers in this cluster. So they are local outliers.

Although in (Milligan, 1985) $\alpha = 9$ is proposed, we use $\alpha = 5$. This α yields outlier that are detectable (i.e., sufficiently far away from inliers), but still close to the corresponding cluster of inliers. Both properties are essential since the outliers should be local. The value of G is chosen from the range of $1, \ldots, 9$ using the Bayesian Information Criterion (BIC) (Hastie et al., 2009) and the covariance parameters are restricted[2] to speed up computations.

4.5.2 Dependency Outliers

Let $F_i(\cdot)$ be the cumulative distribution and $f_i(\cdot)$ the probability density function of data attribute i. If the distribution of each attribute is absolutely continuous, the full multivariate probability density function $f(\cdot)$ can be written (Aas et al., 2009) as

$$f(x_1, \ldots, x_d) = f(x_1) \cdot \ldots \cdot f(x_d) \ \cdot\ c(F_1^{-1}(x_1), \ldots, F_d^{-1}(x_d)). \tag{4.9}$$

The copula $c(\cdot)$ provides "a way of isolating the [...] dependency structure." (Aas et al., 2009). We use it to generate outliers that do not follow the dependency structure, which inliers follow. Outliers in the dependency have recently attracted some attention (see for instance (Ren et al., 2017)), and there also exist proposals for generating outliers in this spirit (see marginSample from Definition 15). A compelling variant of modeling the

[2] With the VEI model in the mclust R package.

copula is the vine (Aas et al., 2009). In principle, the vine decomposes the multivariate distribution into multiple bi-variate building blocks. Each bi-variate distribution can be rather simple, but their combination allows modeling complex dependency patterns.

Here, fit$(\cdot)$ returns distribution estimates for each attribute and a vine copula. modify$(\cdot)$ returns the distribution estimates for the attributes without any modification of the estimates. However, we set the vine copula to complete independence. Hence, generated outliers will not follow any dependency. To estimate the distributions of each attribute we use Kernel Density Estimation (KDE) (Hastie et al., 2009). To select the distribution family[3] for each bi-variate copula in the vine we make again use of the BIC (Hastie et al., 2009).

4.5.3 Global Outliers

Generating uniform outliers has been described many times (Iglesias et al., 2019; Melnykov et al., 2012; Maitra and Melnykov, 2010; Qiu and Joe, 2006; Pei and Zaıane, 2006). Since they scatter across the whole instance space, we dub them global outliers.

The fit$(\cdot)$ function here has three possible forms. In one form it returns a uniform distribution with its bounds being the minimum and maximum of each attribute. This form is a crude and somewhat unrealistic fit to the given genuine inliers. Hence, we also allow fit$(\cdot)$ to be the corresponding function from our previous two instantiations. Thus, the other two forms return a Gaussian Mixture or a vine copula fitted to the data. With any form of fit$(\cdot)$, modify$(\cdot)$ returns a uniform distribution. Its bounds also use the maximum and minimum of each attribute, but the values are increased by 10%. Hence, even when artificial inliers are from a uniform distribution, there are outliers generated.

4.6 Workflow of Experiments

In this section, we describe our experiments by first giving a broad overview. We then introduce the real-world data sets used.

4.6.1 Overview

Our experiments were coded in the R language[4] using the batchtools (Lang et al., 2017) package for organization and parallelism[5]. Thus, to illustrate the workflow of our experiments, we use two entities from batchtools: *problems* and *algorithms*.

A problem describes the data we use with an algorithm. Any problem is based on a real-world data set and might replace inliers, outliers or both with artificial instances. Thus, for each real-world data set described in Section 4.6.2, there are four problem types.

- Real: The real-world data itself

[3] We choose between all families provided by the rvinecopulib R package.
[4] https://www.r-project.org/
[5] Our code is available at http://ipd.kit.edu/mitarbeiter/steinbussg/synth-benchmark-code-V2.zip.

- ArtInliers: we synthetically reconstruct inliers, but outliers remain the ones from the real data (i.e., genuine).
- ArtOutliers: Outliers are artificial (not necessarily reconstructing the genuine ones), but inliers are genuine.
- Art: Outliers are artificial, and we synthetically reconstruct inliers.

For the benchmark we make use of Real and Art. We use the other problems to assess the realness of our data.

An algorithm is a classifier (Classify) to assess the realness of our data or an outlier detection method like LOF (Detect) for the benchmark. We now describe what we use the two algorithms for and how.

Algorithm for Realness

The idea behind the Classify algorithm was proposed in (Sun et al., 2018), to show how well the introduced model fits real-world data. The idea is to test how much worse the classifier performs when trained on artificial instead of genuine data. The workflow of the Classify algorithm requires a problem and a real-world data set and is as follows.

S1: Split the real-world data into 70% training and 30% test with unchanged class frequencies.

S2: Apply the problem on the training data. For example, with ArtInliers replace the inliers with a synthetic reconstruction. The Real problem leaves the training data unchanged.

S3: Train a classifier to distinguish outliers and inliers in the possibly artificial training data.

S4: Report the performance of the classifier evaluated on the test data.

Since we test the classifier on unseen and non-artificial data, the drop in classification performance using artificial data can serve as an indication of the realness of the artificial data. If the artificial data is close to the genuine data, the drop should be small.

For the classifier we used a random forest implemented by the ranger R package and optimized its parameter using ten repetitions of 10-fold cross-validation from the caret R package[6]. Since the two classes in our data are usually imbalanced, we measure classification performance with Cohen's Kappa (Porwik et al., 2016). It lies in [-1,1] where 1 indicates perfect agreement of prediction and ground truth, 0 stands for random guessing, and anything below 0 indicates a prediction worse than this.

Algorithms for the Benchmark

The Detect algorithm performs our benchmark for unsupervised outlier detection methods. Detect used with the Real problem is just a conventional benchmark. With Art, it

[6] For the grid of possible parameter values of the random forest we used the defaults from the caret R package.

follows our general process given in Algorithm 10. In both cases, outliers are up to 5% of the entire data. With Real, this percentage is achieved by sampling from the given outliers, like in (Campos et al., 2016).

In our benchmark we compare the outlier detection methods that are competitive according to existing benchmarks on real-world data (Domingues et al., 2018; Goldstein and Uchida, 2016; Campos et al., 2016; Emmott et al., 2013, 2015). While (Domingues et al., 2018; Emmott et al., 2013, 2015) recommend the isolation Forrest (Liu et al., 2008), the benchmarks (Goldstein and Uchida, 2016; Campos et al., 2016) result in recommendations for k-Nearest Neighbour Detection (kNN) (Ramaswamy et al., 2000) and the LOF (Breunig et al., 2000). In addition to these methods we included KDE (Hastie et al., 2009) since our ideal scores are strongly related to density.

We tried using some meaningful default parameters and not vary them extensively, to keep our experiments comprehensible and to reduce their run time. For the isolation Forrest, we used the parameter values proposed in its original publication (Liu et al., 2008), i.e., $\psi = 256$ and $t = 100$. For the paremters minPts from LOF and k from kNN, we used the values 5 and 100. While 5 is a very local choice, 100 is a rather global one. Instead of the plain kNNwe used the Weighted k-Nearest Neighbour Detection (wkNN) (Angiulli and Pizzuti, 2002), which yields better density estimates (Biau et al., 2011). For the bandwidth parameter γ of KDE, we used the default rule of thumb from the np R package.

4.6.2 Data Sets

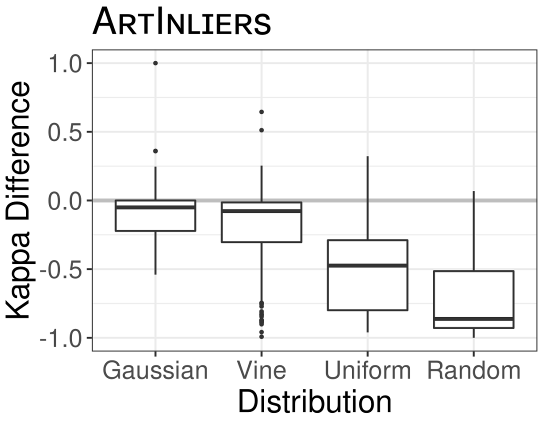
We used most of the data sets proposed in (Campos et al., 2016). Thus we have a total of 19 different data sets. We excluded *Lymphography* and *InternetAds* due to the categorical nature of their attributes: None of their attributes had more than ten distinct values. The data sets *Arrhythmia* with 166 attributes and *SpamBase* with 53 attributes were excluded to keep the run time of our experiments at a reasonable level. We also kept domain-specific artificial data sets, as described in Section 2.2.1. *Waveform* is an example. Such data is engineered for a specific real-world use case, and we deem it realistic.

As described in (Campos et al., 2016), we only kept numeric attributes in each data set, removed duplicate values, and normalized each attribute to $[0, 1]$. We also excluded any attribute with less than ten distinct values. In such cases, the attribute is most likely categorical (e.g., gender gender represented by $\{0, 1\}$). Fitting a continuous distribution to such attributes yields unrealistic artificial data. When we fit statistical distributions to the data, we add some noise to the attribute values according to the procedure described in (Nagler, 2017). This noise should improve the fitting process.

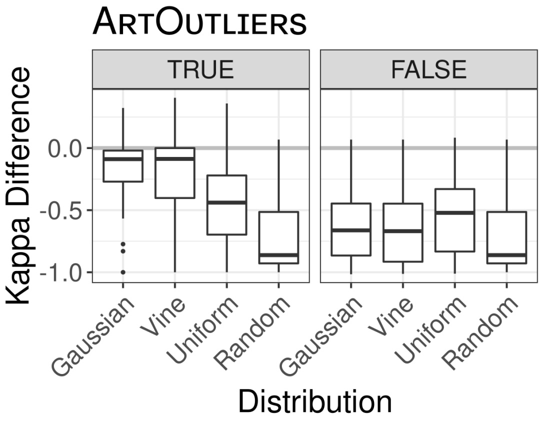
4.7 Results for Realness

First we discuss our results regarding the realness of our data. To do so, we use the Classify algorithm and the problems Real, ArtInlier and ArtOutlier. For realness, we will not use the Art problem (i.e., entirely artificial data). The reason is that we are interested in the realistic reconstruction of inliers. Whether these remain realistic in combination with the different types of generated outliers is of less importance. However, how the artificial outliers impact classification performance on their own, is investigated with ArtOutlier.

4.7.1 Generating Inliers



(a) Using artificial inliers.



(b) Using artificial outliers.

Figure 4.2: Reduction in Kappa.

Figure 4.2a graphs the realness of our synthetically reconstructed inliers. The x-axis lists the distributions that we fitted to inliers. *Random* is not connected to some distribution but shows the drop in classification performance when randomly guessing the class of instances. We include it as a reference point. The y-axis shows the drop in Kappa when we train the classifier with artificial inliers instead of genuine ones.

From Figure 4.2a, we can see that artificial inliers from every distribution yield a better classifier than random guessing — even from the uniform distribution. Next, the Gaussian mixtures give the best reconstruction overall, but the vine is not far from it. Since the median is close to zero, we conclude that our reconstructed inliers are realistic.

4.7.2 Generating Outliers

Outliers should be credible and insightful deviations from the inliers. It is only for completeness that we also investigate the drop in classification performance when training with artificial outliers. See Figure 4.2b. The left plot contains the results when we derive the distribution of the artificial outliers from the genuine outliers and not a characterizable deviation from the model of inliers. Results on the right are from artificial outliers that we generate from our deviations introduced in Section 4.5. The left plot is very similar to Figure 4.2a: Gaussian Mixtures and Vine result in only a small drop in Kappa overall. However, compared to inliers, the Vine seems to perform slightly better for outliers. We find this particularly interesting in two regards. (1) It seems to indicate that the outliers in the benchmark data are not extremely complex. Standard statistical distributions fit them reasonably well. (2) The uniform distribution is not a very good fit for the genuine outliers in the benchmark data. We think that this questions the realness of artificial data with only uniform outliers. The distributions shown in the right plot are entirely independent of the genuine outliers, which is why the drop in classification performance is rather significant. Interestingly the uniform distribution results in the smallest drop in Kappa. We hypothesize that this is because of the one-class nature of the respective resulting classifier. Steinwart et al. (2005) discuss this for an SVM classifier. With outliers generated from a uniform distribution, the classifier might become a well-calibrated one-class classifier. This connection would explain its good detection performance regarding genuine outliers.

4.8 Outlier Detection Benchmark

In the benchmark, we assess detection performance with the Area Under the Precision Recall Curve (AUC PR). We adjust it so that random guessing has value 0. We do not use the Receiver Operating Characteristic (ROC) curve since the data is imbalanced (only 5% are outliers by design). With imbalance, precision-recall curves are preferable (Saito and Rehmsmeier, 2015).

Next, we compare results from our artificial data to ones from real-world data. Then we analyze the correlation of the ideal scorings and the scores from the different outlier detection methods.

4.8.1 Comparison to Genuine Data

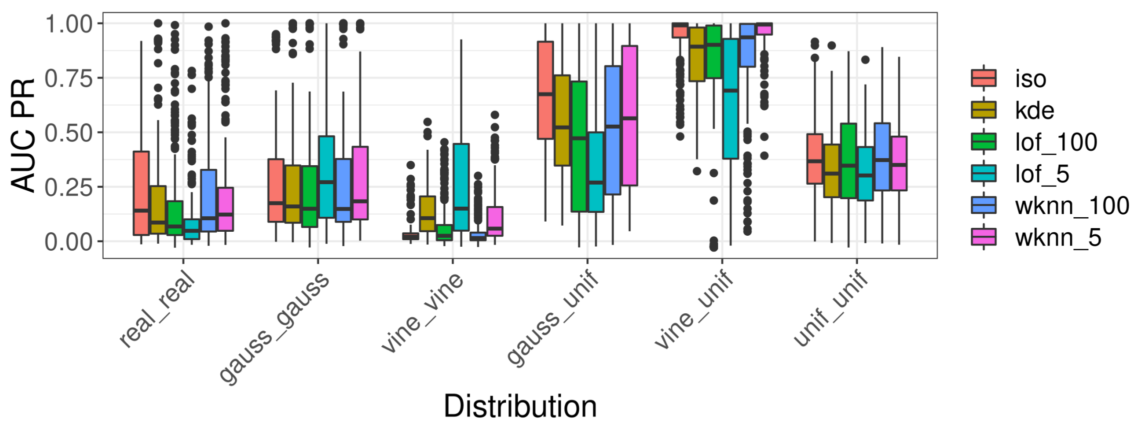


Figure 4.3: Performance of the different detection methods on synthetic and genuine data.

Figure 4.3 contains results regarding our benchmark for different instantiations and genuine data. The y-axis gives the respective AUC PR, and the x-axis gives the distribution inliers (term before "_") or outliers (term after "_") are samples from. The results with entirely genuine data suggest that the Isolation Forrest or wkNN method performs well overall — just as observed with previous benchmarks. Detection methods rank similar when using artificial data that uses the uniform distribution to generate outliers (gauss_unif, vine_unif, and unif_unif). The local method “lof_5” performs worst overall for all three types of data with global outliers. This is expected.

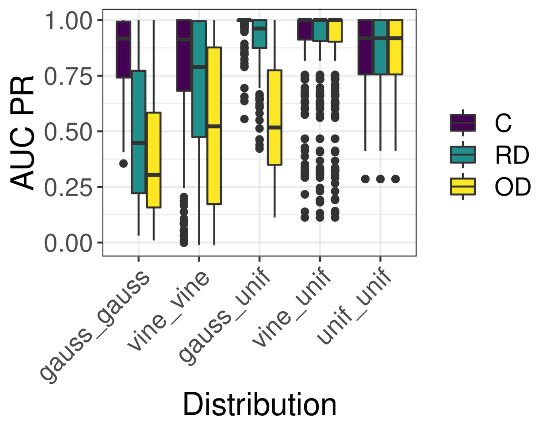
The two types of artificial data that do not use the uniform distribution give a different ranking. Both suggest the local method “lof_5” to be best overall. For the artificial data with outliers from a Gaussian mixture, this is what we would expect. We designed the outliers to be local ones. Nevertheless, this nicely confirms that this type of artificial data is well suited to evaluate methods for local outlier detection. With outliers in the dependency structure of the data (vine_vine) the isolation Forest, “lof_100” and “wknn_100” do not perform well overall. Their inliers AUC PR is close to 0. Hence, local methods seem to detect such outliers much better.

The overall AUC PR, is the highest for artificial data with uniform outliers. Every detection method has a higher average AUC PR than with genuine data. This high score is close to what is found in (Emmott et al., 2015) with simple artificial data. With the other two forms of artificial data, this is not the case — regarding the vine in particular. However, we think, the bare value of the inliers AUC PR is not very meaningful for artificial data. For example, with our instantiation based on Gaussian Mixtures, it depends directly on α, which can be adjusted accordingly. Instead, the ranking in terms of different characteristics is more conclusive.

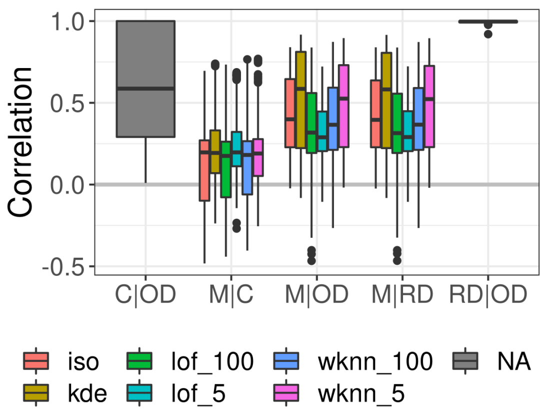
4.8.2 Analyzing Ideal Scores

Section 4.4.3 has introduced three ideal scorings. Here we analyze our benchmark results in terms of them. First, we investigate the detection performance (AUC PR) of our ideal scorings. Then we study the relationship of the ranking of instances from the detection methods to the ones from the ideal scorings.

Detection Performance



(a) Detection quality.



(b) Correlation.

Figure 4.4: Ideal scores.

Figure 4.4a displays the detection quality of each of the ideal scorings. The ranking reflects our expectations: C has the highest AUC PR, followed by RD and then OD. Thus, the fully supervised case (C) yields better results than the case in which the distribution of outliers is not modeled (RD), or both classes are modeled together (OD). When inliers and outliers are from a uniform distribution, all ideal scores are the same. The reason is that in this case the densities dens_{out} and dens_{in} only differ by a constant. Since we compute AUC PR by varying the threshold on the scorings, the specific value for the threshold does not affect it. So all ideal scores give the same AUC PR. When outliers come from a uniform distribution, the AUC PR is highest overall. This high AUC PR confirms their rather global nature.

Relationship to Detection Methods

To analyze how well the scorings from the detection methods coincide with our ideal ones, we use Kendall's Tau coefficient (Hartung et al., 2012) It quantifies the similarity of the rankings with the two scorings. See Figure 4.4b. We abbreviate the scores of the outlier detection methods with "M".

The two ideal scorings based on density (RD and OD) have a firm positive correlation. Hence, using the overall density as the density for inliers has little impact on the overall ranking of instances. We find this very interesting: In the unsupervised setting, computing the overall density is simple — no labels are needed. Getting an estimate of the density of inliers is much more challenging. However, it seems that the small difference in the overall ranking does affect outlier detection performance. This performance varies substantially for the different rankings (cf. Figure 4.4a). The classification scoring does not correlate much with the low-density scorings. We expected this: The classify scoring uses the actual distributions of outliers and inliers. Thus, it can also distinguish instances that might not be meaningful outliers. An example is artificial outliers that are incredibly close to inliers. In general, the methods correlate much more with the density scoring than with the classify scoring. In particular, KDE has a strong correlation with density — which we expected. However, since it is not the best outlier detector (cf. Figure 4.3), there seems to be more to outlier detection than just density estimation. While "lof_5" does not correlate with the density as well as the other methods, it does correlate with the classify scoring more than some other methods. We hypothesize that this comes from its detection capabilities with the Gaussian Mixture but also vine outliers.

4.9 Chapter Summary

Benchmarking unsupervised outlier detection methods continues to be difficult — especially in terms of outliers of different types. We have shown that the problem can be dealt with using entirely artificial data, where outliers are characterizable deviations from inliers. In this chapter, we have proposed a process that yields such artificial data. Using existing real-world data sets as the basis for the generation renders the generated data realistic. The combination of realistic data and outliers as characterizable deviations allows for high interpretability. We also propose concrete techniques for three types of outliers, each one exhibiting different characteristics. An extensive benchmark with state-of-the-art unsupervised outlier detection methods confirms the usefulness of our proposed process. Our results suggest that the relative performance of the detection methods does indeed differ with the various types of outliers. Put differently: no method is optimal for all types. The artificial nature of our data also allowed for the computation of some ideal scorings. The gap between the performance of the detection methods and this ideal scoring might be a useful indicator for improved future detection methods.

All in all, our proposed process allows for a realistic comparison of detection methods and meaningful interpretations of results.

5 Characteristics of Hidden Outliers

In the previous chapters, we have given an overview of existing techniques for generating artificial outliers (Section 2.3 and Chapter 3) and described a process for the evaluation of unsupervised outlier detection methods which uses generated data (Chapter 4). In this chapter, we want to analyze a special type of outlier that adds a layer of complexity to the types investigated so far: hidden outliers.[1]

A hidden outlier exhibits its outlier behavior only in subspaces (i.e., subsets of data attributes) where no outlier detection takes place. Hence, whether an outlier is hidden or not depends on the subspaces where one is looking for outliers. Figure 5.1 displays a low-dimensional illustrative example. 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.

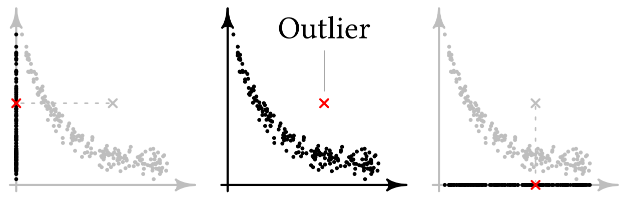


Figure 5.1: Example illustrating a hidden outlier.

In Section 2.1 and Chapter 4, we have discussed that the current approach to the evaluation of outlier detection methods is not entirely satisfactory. Throughout this thesis, it has also become apparent that generated outliers can improve on this. This improvement applies to hidden outliers as well: Generated hidden outliers are known to be outliers in certain subspaces, and one can quantify how well they are found. An evaluation scheme based on generated hidden outliers would also allow differentiating between different types of hidden outliers (in the spirit of the process in Chapter 4). Thus, such an evaluation scheme might be more systematic than one with the usual outlier detection benchmark data. In terms of such an evaluation scheme, we ultimately also come up with a risk measure that quantifies how easy it is to generate hidden outliers for an adversarial attacker. Such an attacker is someone who wants to circumvent detection by carefully manipulating the values of the corresponding instance (cf. Section 3.8.3).

However, the complete design and assessment of an evaluation scheme for SSOD with artificial outliers is beyond the scope of this dissertation. Mainly this is due to some aspects in a full evaluation scheme that needs to be determined by the SSOD community.

[1] The remainder of this chapter is almost identical to (Steinbuss and Böhm, 2017), previously published in the International Journal of Data Science and Analytics. Adjustments are to ensure consistency for this dissertation.

For example, the community needs to find a guideline on the selection of subspaces that should be evaluated. This is, in which subspaces the hidden outliers are detectable and in which they are not. In this thesis, our concern is the effective and comprehensive generation of hidden outliers and their characteristics.

5.1 Challenges for Generating Hidden Outliers

Hidden outliers are inliers in certain subspaces. Thus, we cannot just use extreme values to obtain hidden outliers. Compared to Figure 5.1, the situation may also be more complicated. For instance, when it is not just high dimensional versus low dimensional subspace. Instead, a mix of the two is feasible as well.

The imprecise notion of outliers is another, orthogonal challenge. This imprecision allows for diverse types of outliers. We hypothesize that different outlier types lead to different regions where the generation of hidden outliers is possible. When designing a general algorithm that generates hidden outliers, we cannot assume much about the possible types of hidden outliers.

Another challenge relates to the relative size of the region where hidden outliers can exist. In some scenarios, this region may be small, while it may be huge in others (e.g., close to the full instance space). Generating hidden outliers exhaustively in the feasible region requires techniques that adapt to the size of this region. The generation should cover a broad range of positions if the region is enormous. When the region is small, in turn, the generation must be more fine-grained.

5.2 Contributions Towards Hidden Outliers

In this chapter, we start by deriving important characteristics of hidden outliers analytically, focusing on multivariate data following a normal distribution. One result is that hidden outliers do exist in this setting. Another one is that correlation within subspaces can reduce the size of the region of hidden outliers.

Another contribution of ours is a generation algorithm that generates hidden outliers. The introduced algorithm is essentially a generation technique in combination with a filter (cf. Chapter 3). The algorithm relies on only one mild assumption: a provided outlier detection method defines the types of outliers. The outliers found by different methods are likely of different types — like it is the case in our benchmark in Section 4.8. Hence, the algorithm does not require the assumptions behind our theoretical analyses (e.g., the normal distribution assumption). The algorithm design bases on the hypothesis that the distance of generated hidden outliers and genuine inliers should be adaptive. Thus, our generation technique features a parameter that allows adopting the closeness of generated outliers to genuine instances. This adaptiveness allows for a generation that can concentrate on a small region, close to the genuine data, or a rather large region. The algorithm we propose 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 is transformable in this spirit (see (Kriegel et al., 2011)). 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). The experiments also demonstrate that our adaptive generation technique is much better in generating hidden outliers than a baseline (a technique that is not adaptive). In particular, this holds for high-dimensional data sets.

5.3 Organization of This Chapter

The remainder of this chapter is structured as follows. In Section 5.5.1, we formalize our definition of hidden outliers. Section 5.5.3 features our analytical derivations of characteristics of hidden outliers. In Section 5.5.4, we develop our algorithm to generate hidden outliers. In Section 5.6, we validate our analytical derivations and our developed algorithm.

5.4 Special Notation

The set $\mathcal{F} = \{1, \dots, d\}$ denotes the full attribute space. W.l.o.g., we assume here that each attribute lies within $[l, u]$ where $l,\ u \in \mathbb{R}$. An attribute subset $\mathcal{S} \subseteq \mathcal{F}$ is called a d'-dimensional subspace projection ($1 \leq d' \leq d$). A set $\mathcal{SC} = \{\mathcal{S}_1, \dots, \mathcal{S}_t\} \subseteq \mathcal{P}(\mathcal{F})$[2] is a collection of t subspace projections ($1 \leq t \leq 2^d - 1$). The set $\mathcal{R}_{full} = \{\vec{x} \in [l, u]^d\}$ is the entire data space. When not stated different explicitly, for any region $\mathcal{R}$, it holds that $\mathcal{R} \subseteq \mathcal{R}_{full}$. Further, we assume that there exists a function $\text{dect}^{\mathcal{S}}(\cdot)$ of the form

$$\text{dect}^{\mathcal{S}}(\vec{x}) := \begin{cases} 1 & \text{if } \vec{x} \text{ is outlier in } \mathcal{S}, \\ 0 & \text{if } \vec{x} \text{ is inlier in } \mathcal{S}. \end{cases} \tag{5.1}$$

The function $\text{dect}^{\mathcal{S}}(\cdot)$ here is a generic unsupervised outlier definition with binary output that takes subspace into account. Different outlier detection methods, which typically result in the detection of different types of outliers, are in use. In this chapter, we rely on the detection method to define a specific type of outliers. For this reason, we sometimes refer to the detection methods as "outlier definitions". Many such methods output a score instead of a binary value. However, we assume that these scores are transformable to a binary signal (e.g., by applying a threshold).

5.5 The Region of Hidden Outliers

In this section, we formalize the notion of hidden outliers and derive important characteristics. Section 5.5.2 features some assumptions behind our formal results. In Section 5.5.1,

[2] $\mathcal{P}(\mathcal{F})$ is the power set of $\mathcal{F}$

we define hidden outliers and other relevant concepts. In Section 5.5.3, we derive our formal results. Section 5.5.4 features an algorithm to generate hidden outliers. This algorithm also allows defining the "risk" of hidden outliers, which quantifies assessing the probability of success of an adversarial attacker (someone that wants to circumvent detection).

5.5.1 Definition

We briefly review the well known multiple view property (Müller et al., 2012) of outliers before presenting our underlying definitions. It is crucial for the notion of hidden outliers. The property refers to the case that an instance can be an outlier in some subspaces while being an inlier in others. Hence, there are several subspaces, each containing outliers that could have different meanings. For instance, think of a data set from a bank. Further, let there be a subspace related to the family of customers and another one regarding their employment. The outliers in the family characteristics subspace likely have a different meaning than those in the employment subspace. Subspace outlier detection methods find promising subspaces in a first step; in a second step, these methods apply a conventional outlier detection method to each of these subspaces.

Bearing the multiple view property in mind, we define the notion of a hidden outlier as follows:

Definition 21 (Hidden Outlier) *Let two disjunct sets of subspace projections $\mathcal{SC}_{out}$ and $\mathcal{SC}_{in}$ be given. Then $\vec{o} \in \mathcal{R}_{full}$ is a* hidden outlier with respect to subspace collections $\mathcal{SC}_{out}$ and $\mathcal{SC}_{in}$ *if*

$$\text{dect}^{\mathcal{S}}(\vec{o}) = 0 \quad \forall\, \mathcal{S} \in \mathcal{SC}_{in} \tag{5.2}$$

$$\text{and} \quad \exists\, \mathcal{S} \in \mathcal{SC}_{out} \colon \text{dect}^{\mathcal{S}}(\vec{o}) = 1. \tag{5.3}$$

The number of subspaces not in $\mathcal{SC}_{in}$ 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 $\mathcal{SC}_{out}$ instead of any subspace not in $\mathcal{SC}_{in}$. $\mathcal{SC}_{in}$ and $\mathcal{SC}_{out}$ must always be disjunct. They are disjunct because there cannot be any instance, 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 generating hidden outliers is rather simple. One creates an outlier for one of the subspaces in $\mathcal{SC}_{out}$ and sets the values for the remaining attributes in $\mathcal{F}$ to the ones of any genuine inlier. Thus, we focus here on scenarios with such overlap. Based on Definition 21, we now formulate a hypothesis.

Hypothesis 1 *Since hidden outliers are inliers for all subspaces in $\mathcal{SC}_{in}$, hidden outliers must be spatially close to the instances in $\underline{X}$.*

We will return to Hypothesis 1 when designing our algorithm (Section 5.5.4) and in the experiments (Section 5.6.4).

A core issue in this study is to identify the positions/region with the following characteristic: If we generate an instance there, it is a hidden outlier. We now derive this region

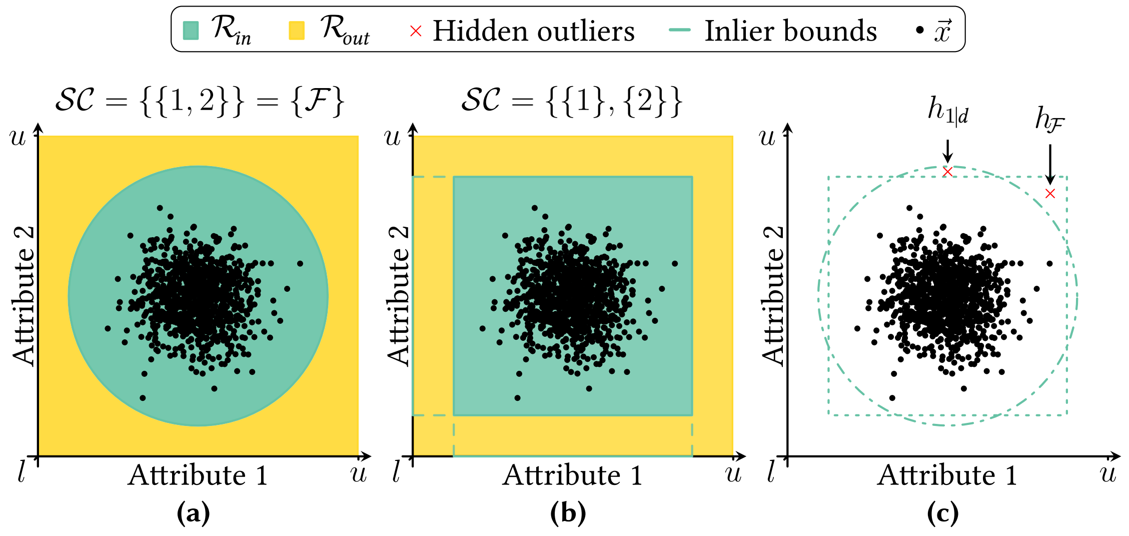


Figure 5.2: Example for $\mathcal{R}_{in}^{\mathcal{SC}}$, $\mathcal{R}_{out}^{\mathcal{SC}}$ and hidden outliers in $h_{\mathcal{F}}$ and $h_{1|d}$.

and present some characteristics of hidden outliers. To this end, we do not rely on any further assumption regarding $\text{dect}^{\mathcal{S}}(\cdot)$.

Definition 22 ($\mathcal{R}_{in}$ and $\mathcal{R}_{out}$) *The region of inliers $\mathcal{R}_{in}^{\mathcal{SC}}$ is defined as*

$$\{\vec{o} \in \mathcal{R}_{full} \,|\, \text{dect}^{\mathcal{S}}(\vec{o}) = 0 \quad \forall\, \mathcal{S} \in \mathcal{SC}\}. \tag{5.4}$$

The region of outliers $\mathcal{R}_{out}^{\mathcal{SC}}$ is its complement.

Definition 22 formalizes the notion of the region fulfilling one property of hidden outliers: regions with positions that are inlier or outlier for each subspace in $\mathcal{SC}$. This notion is a prerequisite before defining the region of hidden outliers. See Figures 5.2a and 5.2b for examples using the Mahalanobis distance (Kriegel et al., 2010) which is detailed in Section 5.5.2.

Lemma 1 *The region $\mathcal{R}_{in}^{\mathcal{SC}}$ is the intersection of the regions $\mathcal{R}_{in}^{\mathcal{S}}$ $\forall\, \mathcal{S} \in \mathcal{SC}$.*

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

Definition 23 (Region of Hidden Outliers) *Let two sets of subspace projections $\mathcal{SC}_{out}$ and $\mathcal{SC}_{in}$ be given. The region of hidden outliers $\mathcal{R}_{hidden}$ is the intersection of the region $\mathcal{R}_{in}^{\mathcal{SC}_{in}}$ and the region $\mathcal{R}_{out}^{\mathcal{SC}_{out}}$.*

Every instance in $\mathcal{R}_{hidden}$ is a hidden outlier. We see that, up to intersections, unions and complements, $\mathcal{R}_{hidden}$ solely depends upon outlier detection in a single subspace. However, $\mathcal{R}_{in}^{\mathcal{S}}$ is of arbitrary shape – depending on $\text{dect}^{\mathcal{S}}(\cdot)$. Hence, computing these intersections, unions, and complements is arbitrarily complex.

The number of possible $\mathcal{SC}$s is huge: Having $|\mathcal{P}(\mathcal{F})|$ $(= 2^d - 1)$ subspaces yields $2^{|\mathcal{P}(\mathcal{F})|} - 1$ possible $\mathcal{SC}$s. The number of possible combinations of two $\mathcal{SC}$s to obtain $\mathcal{R}_{hidden}$ is even larger. Thus, in the first part of this chapter we focus on two cases, $\mathcal{SC}_1 = \{\mathcal{F}\}$ (the full space) and $\mathcal{SC}_2 = \{\{1\}, \dots, \{d\}\}$ (each one-dimensional subspace).

Notation 1 ($h_{\mathcal{F}}$ and $h_{1|d}$) *$h_{\mathcal{F}}$ refers to the setting when $\mathcal{SC}_{in} = \{\{1\}, \dots, \{d\}\}$ and $\mathcal{SC}_{out} = \{\mathcal{F}\}$. $h_{1|d}$ to the setting when vice versa $\mathcal{SC}_{in} = \{\mathcal{F}\}$ and $\mathcal{SC}_{out} = \{\{1\}, \dots, \{d\}\}$.*

In setting $h_{\mathcal{F}}$, outliers are detectable in the full space, but not in any one-dimensional projection. Setting $h_{1|d}$ is the opposite: Outliers are not detectable in the full space but at least in one of the one-dimensional projections.

Example 6 *In Figure 5.2c, 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 instances that are inliers in both subspaces of $\mathcal{SC} = \{\{1\}, \{2\}\}$. The circle represents the inlier bound for points that are inliers in $\mathcal{SC} = \{\mathcal{F}\} = \{\{1, 2\}\}$. Hidden outliers in setting $h_{\mathcal{F}}$ are instances inside the square but outside the circle. Analogously, hidden outliers in setting $h_{1|d}$ are instances outside the square but inside the circle.*

In some cases we will refer to a more general form of the settings $h_{\mathcal{F}}$ and $h_{1|d}$: where one collection is an arbitrary partition of $\mathcal{F}$ into subspaces.

When analyzing the characteristics of $\mathcal{R}_{hidden}$, we will make use of the relative volume of a region. More explicitly, we use it to bound the region of hidden outliers.

Definition 24 ($\mathrm{vol}_{rel}(\cdot)$) *Let a region $\mathcal{R} \subseteq \mathbb{R}^d$ be given. The* relative volume *of $\mathcal{R}$ is defined as*

$$\mathrm{vol}_{rel}(\mathcal{R}) := \frac{\mathrm{vol}(\mathcal{R}_{full} \cap \mathcal{R})}{\mathrm{vol}(\mathcal{R}_{full})}. \tag{5.5}$$

Lemma 2 *An upper bound on $\mathrm{vol}_{rel}(\mathcal{R}_{hidden})$ is*

$$\min \left[\mathrm{vol}_{rel}(\mathcal{R}_{in}^{\mathcal{SC}_{in}}) \, , \, \mathrm{vol}_{rel}(\mathcal{R}_{out}^{\mathcal{SC}_{out}})\right] . \tag{5.6}$$

Thus, if the relative volume of $\mathcal{R}_{out}^{\mathcal{SC}_{out}}$ or $\mathcal{R}_{in}^{\mathcal{SC}_{in}}$ is very small, e.g., zero, we know that the relative volume of $\mathcal{R}_{hidden}$ cannot be larger.

In the next step, we investigate specific scenarios with an outlier-detection method using the Mahalanobis Distance; having such a specific outlier notion allows deriving distinct characteristics of $\mathcal{R}_{hidden}$.

5.5.2 Assumptions for Formal Results

We assume that $\underline{X}$ follows a Multivariate Normal Distribution (MVN) with zero mean. Of course, Gaussian distributed instances 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 yields the likeliness of instance. We assume instances to be outliers if they are unlikely according to that distance. We refer to the Mahalanobis distance of $\vec{x}$ in subspace $\mathcal{S}$ as $\mathrm{mdist}^{\mathcal{S}}(\vec{x})$.

quant(α, DF) gives the α-quantile of a χ^2 distribution with DF Degrees of Freedom. We can instantiate our outlier definition as follows (Kriegel et al., 2010):

$$\text{dect}^{\mathcal{S}}(\vec{x}) := \begin{cases} 1 & \text{if } \left[\text{mdist}^{\mathcal{S}}(\vec{x})\right]^2 > \text{quant}(0.975, |\mathcal{S}|)\,, \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

5.5.3 Formal Results

Motivation for Theorem 1: Figure 5.2c is a two-dimensional example illustrating hidden outliers. The lines are outlier boundaries using the Mahalanobis distance. In this two-dimensional case, hidden outliers can exist for both settings $h_{\mathcal{F}}$ and $h_{1|d}$. We wonder whether this existence of hidden outliers extends to higher dimensionalities and more general subspace selections. We answer this question by analyzing a more general scenario. In our two exemplary settings, there are two kinds of subspace selections. We have $\mathcal{SC}_1 = \{\{1\}, \ldots, \{d\}\}$ and $\mathcal{SC}_2 = \{\mathcal{F}\}$. To generalize this, we replace $\mathcal{SC}_1$ with an arbitrary partition of the attribute space.

Theorem 1 *Let $\mathcal{SC}$ be a non-trivial (i.e., $\neq \mathcal{F}$) partition of $\mathcal{F}$ into subspaces. Let the number of dimensions of $\mathcal{F}$ and of each subspace in $\mathcal{SC}$ converge to infinity. Let each data attribute be i.i.d. according to a standard normal distribution. Then there exists a hidden outlier that is an outlier in at least one subspace of $\mathcal{SC}$ and inlier in $\mathcal{F}$. There also exists a hidden outlier that is outlier in $\mathcal{F}$ but inlier in each subspace of $\mathcal{SC}$.*

All proofs are in the appendix. We have assumed that the dimensionality goes to infinity in order to approximate quant($\cdot$, $\cdot$) in the proof. However, Figure 5.2c shows that the theorem holds even in a two-dimensional case. Our experiments will show that it is likely to hold for other data sets and detection methods as well.

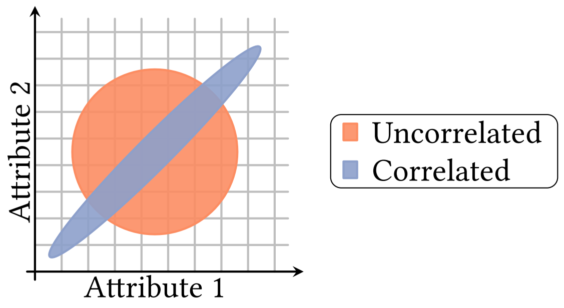


Figure 5.3: Motivation for Theorem 2 and Hypothesis 2.

Motivation for Theorem 2: Next, we consider the effect of correlation on the relative volume of $\mathcal{R}_{in}^{\mathcal{F}}$. Figure 5.3 displays $\mathcal{R}_{in}^{\mathcal{F}}$ in a two-dimensional example. The circle is for the case that Attributes 1 and 2 are uncorrelated, and the ellipse represents a 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 $\mathcal{R}_{in}^{\mathcal{F}}$ and a larger relative volume of $\mathcal{R}_{out}^{\mathcal{F}}$. Theorem 2 formalizes 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. according to a normal distribution with zero mean and variance σ. The covariance matrix in $\mathcal{S}_1$ is $\underline{\Sigma}_1$ and in $\mathcal{S}_2$ $\underline{\Sigma}_2$. For these matrices, it holds that* $\mathrm{diag}(\underline{\Sigma}_1) = \mathrm{diag}(\underline{\Sigma}_2)$, *and that $\underline{\Sigma}_2$ has off-diagonal elements (i.e., there is correlation). Then we have:*

$$\mathrm{vol}_{rel}\big(\mathcal{R}_{in}^{\mathcal{S}_1}\big) \geq \mathrm{vol}_{rel}\big(\mathcal{R}_{in}^{\mathcal{S}_2}\big) . \tag{5.8}$$

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

Lemma 3 *From $\mathcal{R}_{out}^{\mathcal{S}} = \overline{\mathcal{R}_{in}^{\mathcal{S}}}$ it follows that*

$$\mathrm{vol}_{rel}\big(\mathcal{R}_{out}^{\mathcal{S}_1}\big) \leq \mathrm{vol}_{rel}\big(\mathcal{R}_{out}^{\mathcal{S}_2}\big) . \tag{5.9}$$

Motivation for Hypothesis 2: Theorem 2 reasons on the influence of correlation on inlier and outlier regions. In $h_{\mathcal{F}}$, the outlier region is $\mathcal{R}_{out}^{\mathcal{F}}$ (i.e., involves the full space). In $h_{1|d}$, the inlier region $\mathcal{R}_{in}^{\mathcal{F}}$ involves the full space. In both settings, the respective other region depends on a $\mathcal{SC}$ 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 regions is an upper bound on the relative volume of $\mathcal{R}_{hidden}$. Thus, if one of the relative volumes of inlier and outlier increases or decreases, this bound might do so as well.

Hypothesis 2 *Correlated data leads to a smaller relative volume of $\mathcal{R}_{hidden}$ in setting $h_{1|d}$ than uncorrelated data. In $h_{\mathcal{F}}$, it is larger.*

If this hypothesis holds, it is more difficult to hide outliers in correlated subspaces in $h_{1|d}$ and less difficult in $h_{\mathcal{F}}$. We evaluate this assumption using various data sets and outlier-detection methods in Section 5.6.2.

5.5.4 Algorithm for Generating Hidden Outliers

So far, we have studied the characteristics of $\mathcal{R}_{hidden}$ analytically depending on certain assumptions. We now propose an algorithm that generates hidden outliers for any instantiation of $\mathrm{dect}^{\mathcal{S}}(\cdot)$, subspace selections, and data set.

Our proposed algorithm is randomized. This is, a set of instances $\vec{a} \in \mathcal{R}_{full}$ is randomly generated and then filtered for hidden ones. The baseline we propose generates the initial instances according to a uniform distribution with domain $\mathcal{R}_{full}$ (the unifBox technique from Definition 4). However, inspecting such instances would not only be extremely expensive, but it also would not take into account that $\mathcal{R}_{hidden}$ can be a tiny portion of $\mathcal{R}_{full}$. See Figure 5.2c. The red crosses indicate some instances in $h_{\mathcal{F}}$ and $h_{1|d}$. We computed these areas by detecting outliers in each attribute in isolation as well as in the full space. While the Region $h_{\mathcal{F}}$ is rather large, $h_{1|d}$ is not. If $\mathcal{R}_{hidden}$ is small, an algorithm that concentrates on this part of the data space is desirable. However, the algorithm should also inspect feasible positions exhaustively otherwise. A generation that is always exhaustive, however, would leave aside Hypothesis 1. It has stated that hidden outliers are close to $\underline{X}$.

According to it, it is unlikely that an extreme position, an instance far from $\underline{X}$, is a hidden outlier. To facilitate a generation that is adaptive in this spirit, we specify a parameter to model the probability of generating at certain positions. In particular, instances next to genuine ones $\vec{x} \in \underline{X}$ will have a higher likelihood.

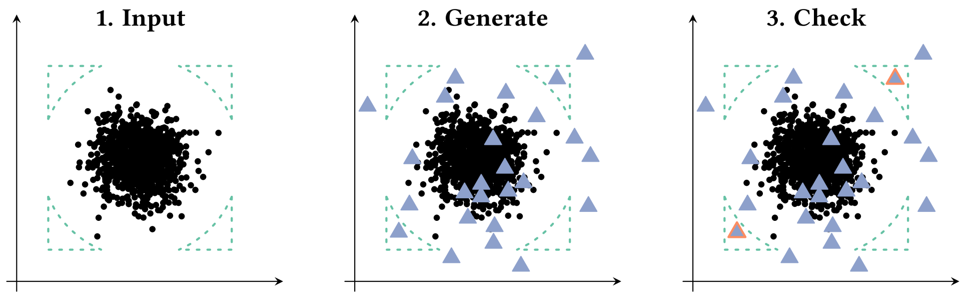


Figure 5.4: Exemplary flow of our generation algorithm.

Our algorithm consists of two steps. First, the algorithm generates random instances within $\mathcal{R}_{full}$. Second, the algorithm filters the generated instances for those that are hidden outliers. Figure 5.4 illustrates this flow. The example resembles the scenario from Figure 5.2. The method detecting outliers uses the Mahalanobis distance, and our data has two attributes. In Figure 5.4, we generate hidden outliers in the $h_{\mathcal{F}}$ setting. Thus, the hidden outliers are outliers in the full space but must not be detectable in any one-dimensional projection. The leftmost plot shows the data set that is part of the input of the algorithm. The area framed by the dotted lines are positions where generated instances will be hidden outliers. This area depends on other inputs of the algorithm (e.g., the detection method used in a subspace). The other two plots display the actual generation. The second plot visualizes random instances generated in the instance space ($\mathcal{R}_{full}$). For the generation, we used the baseline generation technique UNIFBOX. In the third plot, a filtering of the generated instances followed, keeping only the desired hidden outliers (triangles framed in orange).

In the following, we discuss the technique we propose for generating the initial instances. For this, we discuss the probability distribution of instances generated with this technique. Then we discuss the filter for hidden outliers.

Probability of Generating a Certain Instance

A straightforward technique for generating the artificial instances $\vec{a}$ is to generate them only in some close surroundings of genuine instances, like many techniques described in Chapter 3. However, we also want to consider the distance of $\vec{a}$ to the attribute bounds. We deem this important to adapt the generation in-between an exhaustive generation (UNIFBOX) and a generation extremely close to genuine instances. To this end we introduce the parameter $\varepsilon \in [0, 1]$. Figure 5.5 graphs the probability of an instance being generated at a certain position, for a single attribute and genuine instance x. We use the log scale for better illustration. The area of both rectangles is 1. If $\varepsilon = 0$ we do not allow for any distance greater than 0 to x. Thus, the probability of generating an instance with value x is 1 and with any other value 0. If $\varepsilon = 1$, we allow for any distance as long as the instances do not exceed the attribute bounds. We set the probability of generating a to be constant and thus to $\frac{1}{u-l}$. If $0 < \varepsilon < 1$, an artificial instance a between $x - \varepsilon \cdot (x - l)$ and $x + \varepsilon \cdot (u - x)$ is generated with a probability of $\frac{1 \div \varepsilon}{u-l}$. Outside of these bounds, there is no generation.

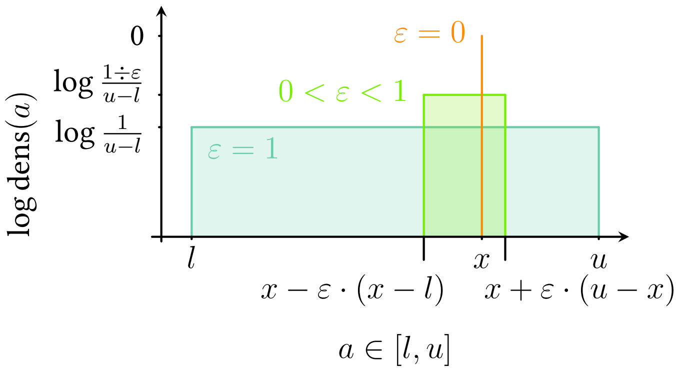


Figure 5.5: Exemplary probability density of generating instances regarding a single attribute and genuine instance x.

Definition 25 (surr) *Let $\vec{x}_1, \ldots, \vec{x}_{n_{genu}} \in \mathcal{R}_{full}$ and $\varepsilon \in [0,1]$ be given. The* surrounding region *of a single instance $\vec{x}_j$ is defined as:*

$$\mathrm{surr}^{\varepsilon}(\vec{x}_j) := \left\{ \vec{a} \in \mathcal{R}_{full} \mid \xi^{\vec{x}_j}(\vec{a}) \leq \varepsilon \right\}, \tag{5.10}$$

where

$$\xi^{\vec{x}}(\vec{a}) := \max_{i \in \mathcal{F}} \left(\frac{x^{(i)} - a^{(i)}}{u - x^{(i)}}, \frac{a^{(i)} - x^{(i)}}{x^{(i)} - l} \right). \tag{5.11}$$

The surrounding region *of several instances $\vec{x}_1, \ldots, \vec{x}_n$ is defined:*

$$\mathrm{surr}^{\varepsilon}(\vec{x}_1, \ldots, \vec{x}_{n_{genu}}) := \bigcup_{j=1}^{n_{genu}} \mathrm{surr}^{\varepsilon}(\vec{x}_j). \tag{5.12}$$

The region $\mathrm{surr}^{\varepsilon}(\vec{x})$ consists of all $\vec{a}$ whose probability of being generated is greater than zero. Figure 5.6 illustrates the surrounding region in an example with four genuine instances.

The following lemma features useful characteristics of the surrounding region.

Lemma 4 *If $\varepsilon = 0$ then*

$$\mathrm{surr}^{0}(\vec{x}_j) = \{\vec{x}_j\} \quad \forall\, j \in 1, \ldots, n_{genu} \quad \textit{and} \tag{5.13}$$

$$\mathrm{surr}^{0}(\vec{x}_1, \ldots, \vec{x}_{n_{genu}}) = \{\vec{x}_1, \ldots, \vec{x}_{n_{genu}}\}. \tag{5.14}$$

and if $\varepsilon = 1$ then

$$\mathrm{surr}^{1}(\vec{x}_j) = \mathrm{surr}^{1}(\vec{x}_1, \ldots, \vec{x}_{n_{genu}}) = \mathcal{R}_{full} \quad \forall\, j \in 1, \ldots, n_{genu}. \tag{5.15}$$

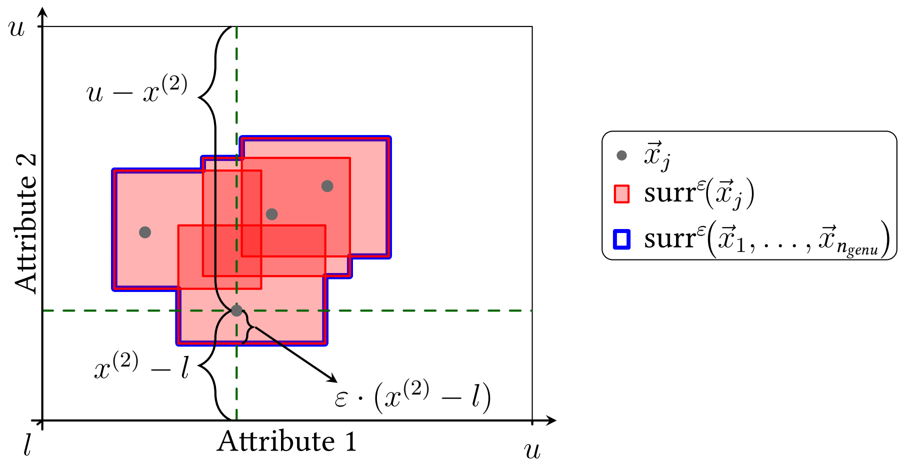

u
$u - x^{(2)}$
Attribute 2
$x^{(2)} - l$
$\varepsilon \cdot (x^{(2)} - l)$
l
Attribute 1
u
● $\vec{x}_j$
🟥 $\mathrm{surr}^{\varepsilon}(\vec{x}_j)$
🟦 $\mathrm{surr}^{\varepsilon}(\vec{x}_1, \ldots, \vec{x}_{n_{genu}})$

Figure 5.6: Example of range distance and surrounding region.

The surrounding region can consist of solely the genuine instances themselves, the entire instance space, or a middle ground between these extremes ($0 < \varepsilon < 1$). Generating instances only in $\mathrm{surr}^{\varepsilon}(\underline{X})$ does away with the difficulties of a purely exhaustive generation. However, our technique so far features a parameter (ε), and it is unclear how to choose its value. We will discuss this in Section 5.5.5.

Regarding the generation of instances, a surrounding region gives way to a probability distribution from which to draw the samples, as follows:

$$\mathrm{dens}(\vec{a}) \propto \frac{1}{n_{genu}} \sum_{j=1}^{n_{genu}} \mathbb{1}_{\{\vec{a} \,\in\, \mathrm{surr}^{\varepsilon}(\vec{x}_j)\}}. \tag{5.16}$$

The function $\mathrm{dens}(\cdot)$ in Equation (5.16) is the multivariate generalization of $\mathrm{dens}(\cdot)$ from Figure 5.5. Algorithm 11 describes our technique to generate instances, given $\mathrm{dens}(\cdot)$. The first step to generate n_{art} instances with $\mathrm{dens}(\cdot)$ is to randomly draw n_{art} genuine instances from $\underline{X}$. For every attribute value of each such instance, the algorithm calculates two new values, one towards the upper attribute limit u and one towards the lower limit l. Both are scaled by ε. The algorithm then determines randomly whether the sample takes the value next to u or l. This results in n_{art} random samples from $\mathrm{dens}(\cdot)$.

Filtering the Generated Instances

The next step necessary to generate hidden outliers is to check if an artificial instance is a hidden outlier or not. See Algorithm 12 for our approach to this. For a given instance, the algorithm checks if it is an inlier in each subspace in $\mathcal{SC}_{in}$ and an outlier in at least one subspace of $\mathcal{SC}_{out}$. Thus, we can filter generated instances for hidden outliers. Armed with this algorithm and our generation technique, it is now possible to generate hidden outliers in $\mathrm{surr}^{\varepsilon}(\underline{X})$, for given a data set $\underline{X}$, ε and n_{art}, as long as $\mathrm{vol}_{rel}(\mathcal{R}_{hidden}) > 0$.

Algorithm 11 Generate instances using the density from Equation (5.16).

```
Input: $n_{art}$, $\varepsilon$, $\underline{X}$
Output: Artificial instances $\vec{a}_1, \ldots, \vec{a}_{n_{art}} = \underline{A}$
 1: Sample $\vec{x}_1, \ldots, \vec{x}_{n_{art}}$ from $\underline{X}$ with replacement
 2: for $\vec{x}_j \in \{\vec{x}_1, \ldots, \vec{x}_{n_{art}}\}$ do
 3:     for each attribute $i$ of $\vec{x}_j$ do
 4:         $\Delta l :=$ Sample a value uniformly from $[0, x_j^{(i)} - l]$
 5:         $\Delta u :=$ Sample a value uniformly from $[0, u - x_j^{(i)}]$
 6:         Select at random if
 7:         $a_j^{(i)} := x_j^{(i)} - \varepsilon \cdot \Delta l$
 8:         or
 9:         $a_j^{(i)} := x_j^{(i)} + \varepsilon \cdot \Delta u$
10:     end for
11: end for
12: return $\underline{A}$
```

5.5.5 The Parameter of Our Technique

In the following, we will discuss more on the meaning and properties of the parameter ε featured by our generation technique. First, we will offer an alternative interpretation of this parameter, followed by a method to choose its value. Then we will detail the definition of the risk of hidden outliers and give the algorithmic complexity of the proposed algorithm. The section concludes with a summary of the whole algorithm.

Interpreting the Parameter

To motivate our interpretation we revisit Figure 5.6. The region $\text{surr}^{\varepsilon}(\underline{X})$ with the blue surrounding is a boundary for the genuine instances. ε controls its tightness. Lemma 4 has stated that, if $\varepsilon = 0$, $\text{surr}^{0}(\underline{X})$ consists of the instance from $\underline{X}$. Thus, $\text{dens}(\cdot)$ from Equation (5.16) will reveal exact information (on attribute values and frequency) about the genuine instances in $\underline{X}$. If $\varepsilon \approx 0$, the information encoded in $\text{dens}(\cdot)$ will not be exact but still give a good approximation. However, the closer ε is to 1, the less information on genuine instances is conveyed in $\text{dens}(\cdot)$. Thus, one can interpret ε as an indication of how much information the generation technique has on the data ($\underline{X}$).

Choosing the Parameter

Up to here, ε is an exogenous parameter, without the flexibility envisioned. Thus, we now add one step to the algorithm. Figure 5.7 graphs the proportion of generated instances that are hidden outliers in one example setting. The maximum is reached at $\varepsilon \approx 0.3$. Hence, in this example, 0.3 is the value that allows for the best generation of hidden outliers. We refer to the ε that maximizes the proportion of hidden outliers as ε^*. The ε^* value is not just the optimum for ε but also allows for the computation of the risk of hidden outliers. Usually, there is no knowledge of the dependency between ε and the proportion of hidden

Algorithm 12 Filter instances for hidden outliers.

```
Input: $\mathcal{SC}_{out}$, $\mathcal{SC}_{in}$, $\underline{A}$, $\underline{X}$, $\text{dect}^{\mathcal{S}}(\cdot)$
Output: Set of hidden outliers $\{\vec{h}, \dots, \vec{h}_{n_{hidden}}\}$ where $n_{hidden} \leq n_{art}$
 1: for $\vec{a}_j \in \underline{A}$ do
 2:     $b_{in}$ = TRUE
 3:     $b_{out}$ = FALSE
 4:     for $t \in \{out, in\}$ do
 5:         for $\mathcal{S} \in \mathcal{SC}_t$ do
 6:             $r := \text{dect}^{\mathcal{S}}(\vec{a}_j)$
 7:             if $t = in$ and $r \neq 0$ then
 8:                 $b_{in}$ = FALSE
 9:             end if
10:             if $t = out$ and $r = 1$ then
11:                 $b_{out}$ = TRUE
12:             end if
13:         end for
14:     end for
15:     if $b_{in} = b_{out}$ = TRUE then
16:         Add $\vec{a}_j$ to set of hidden outliers
17:     end if
18: end for
19: return $\{\vec{h}, \dots, \vec{h}_{n_{hidden}}\}$
```

outliers. Because of this missing knowledge, we propose to use a genetic algorithm to find ε^*.

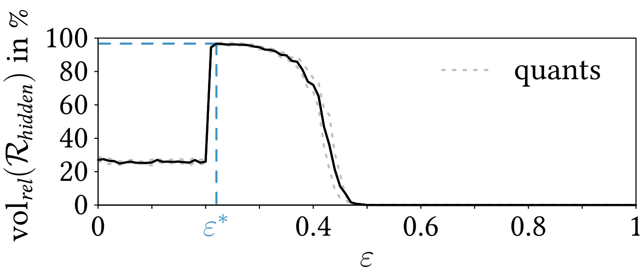


Figure 5.7: Demonstration of the connection of the proportion of hidden outliers in generated instances and ε. Using $h_{1|d}$, Arrhythmia and DBOut (see Section 5.6.1). The risk is 0.44.

The Risk of Hidden Outliers

Definition 26 (Risk of Hidden Outliers) *Let $\underline{X}$, $\text{dect}^{\mathcal{S}}(\cdot)$, $\mathcal{SC}_{in}$ and $\mathcal{SC}_{out}$ be given. The* risk of attacker success *is the harmonic mean of ε^* and the proportion of hidden outliers the attacker is able to hide in the surrounding region* $\text{surr}^{\varepsilon^*}(\underline{X})$.

This risk has domain $[0,1]$. If hiding outliers is difficult, the risk of the data owner is small. Recall our interpretation of ε as the amount of information known on the data. $\varepsilon^* \approx 1$ means that an attacker does not need any information on $\underline{X}$ to generate hidden outliers (except l and u). If both ε^* and the maximal proportion of hidden outliers are high, it is easy to generate hidden 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 or generated instances rarely are hidden outliers.

Complexity

The algorithm we propose targets at high result quality. We deem absolute runtime less critical, 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 generating hidden outliers for a given ε, the algorithms performs

$$\#C = n_{art} \cdot |\mathcal{SC}_{in}| \cdot |\mathcal{SC}_{out}| \tag{5.17}$$

calculations. Let $\#G$ be the maximal number of fitness-function evaluations by the genetic algorithm.

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

Summary of the Algorithm

The algorithm needs four inputs: the two subspace collections ($\mathcal{SC}_{in}$, $\mathcal{SC}_{out}$, an outlier detection method ($\text{dect}^{\mathcal{S}}(\cdot)$) and the number of instances initially generated (n_{art}). An additional optional input is ε; when not supplied, the algorithm itself determines ε by the means of a genetic algorithm (ε^*). First, the algorithm generates n_{art} instances from the probability distribution in Equation (5.16) (see Algorithm 11). Then these artificial instances are filtered. Only instances that are inlier in each subspace of $\mathcal{SC}_{in}$ and outlier in at least one subspace of $\mathcal{SC}_{out}$ according to $\text{dect}^{\mathcal{S}}(\cdot)$ remain. See Algorithm 12. The result is a (possibly empty) set of hidden outliers. When not provided withε, the algorithm repeats this procedure for different values of ε. A heuristic generates values of ε, that targets at maximizing the share of hidden outliers in each execution.

5.6 Experiments

In Section 5.5.3, we have derived characteristics of $\mathcal{R}_{hidden}$ analytically, assuming a specific outlier detection method and underlying data distribution. In our experiments, we investigate its behavior in terms of other outlier detection methods and data sets using our algorithm. The experiments show the general ability of our algorithm to generate hidden outliers and the vulnerability of different detection methods. We also study the role of ε^* (e.g., whether there exists a unique one).

5.6.1 Experiment Setup

We first describe the outlier detection methods and data sets used in the experiments. In essence, these methods define the types of outliers in each subspace, which is the reason we refer to them here as outlier definitions. Then we describe the subspace selections we will use. All our code and used data sets from this chapter are publicly available.[3]

Outlier Detection

Additionally, to the Mahalanobis distance (further denoted mdist), we investigate three other outlier definitions. One of them, follows the DB(p, k)-Outlier (DBOut), proposed in (Kollios et al., 2003). An instance is an outlier if at most p instances have a distance less than k. The distance used is the euclidean metric. Hence, solely the dimensionality of a subspace implies different magnitudes of distances (Zimek et al., 2012). These different magnitudes are the reason we use an adaptive k for each subspace instead of a fixed one. In particular, we set

$$k = 0.2 \cdot \sqrt{\mid \mathcal{S} \mid}. \tag{5.18}$$

The factor $\sqrt{\mid \mathcal{S} \mid}$ is used to scale the distances to the size of the subspace. If the subspace is of size 1, the maximal Euclidean distance is 1 = $\sqrt{1}$ (the attributes are normalized such that $l = 0$ and $u = 1$). In general, the maximal Euclidean distance between two instances is $\sqrt{\mid \mathcal{S} \mid}$. Hence, the distance of an inlier to its nearest neighbor can be at most 20% of the maximal distance.

The last two methods we use are Angle-Based Outlier Detection (ABOD) (Kriegel et al., 2008) and Local Outlier Probabilities (LoOP) (Kriegel et al., 2009a). ABOD uses angles to determine the outlierness of an instance. These angles are deemed more reliable in higher dimensions than the typical L^p-distance (Kriegel et al., 2008). In (Kriegel et al., 2008) three different implementations of ABOD are proposed, which incorporate different trade-offs between performance and result quality. We use the fastest implementation, FastABOD. LoOP is an adoption of the well known LOF (Breunig et al., 2000). 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 an instance is an outlier or not. FastABOD and LoOP output scores. Regarding LoOP a low score indicates inliers, as for FastABOD a high score. In our experiments, we need an automatic threshold that allows transforming 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. We set the neighborhood size to 5.

Data sets

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 (Dheeru and Karra Taniskidou, 2017), namely Madelon and Gisette (500 and 5,000 attributes). The remaining real-world data sets are from (Campos et al., 2016). We always use the normed data, and with

[3] http://ipd.kit.edu/mitarbeiter/steinbussg/Experiments_HideOutlier.zip

downsampled data, we use version one. The two artificial data sets are generated by sampling from a MVN, each with 500 instances and 30 attributes. In one data set, each attribute is i.i.d. according to a standard normal. We refer to this data set with “MVN”. In the second one “MVN cor.”, attributes follow a standard normal distribution as well, but each pair of attributes has a covariance of 0.8. Thus, with “MVN cor.” attributes are correlated. Mostly we use the artificial data sets for the experiments studying Hypothesis 2. To obtain comparability across data sets, each data sets has been normalized (i.e., $l = 0$ and $u = 1$).

Subspace Selection

To evaluate our theoretical findings, we have a deterministic procedure for subspace selection ($h_{\mathcal{F}}$ and $h_{1|d}$). Only for Theorem 1 we need to sample subspace partitions. However, when evaluating the general quality of our approach, instantiations of $\mathcal{SC}_{in}$ and $\mathcal{SC}_{out}$ 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 the number of possible subspaces, exemplary displayed in Figure 5.8, implies the following: We assume that one checks typically 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 attributes (red area). Thus, these outlier subspaces are a natural selection. However, even with these restrictions, the number of the outlier and even inlier subspaces can still be infeasibly large. Thus, we sample them according to the procedure in Algorithm 13.

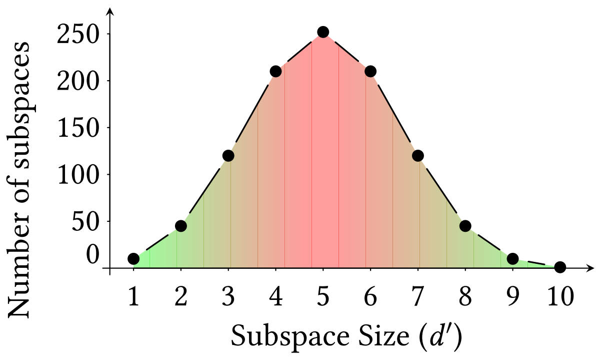


Figure 5.8: Number of subspaces versus subspace size ($d = 10$).

First candidate subspaces for $\mathcal{SC}_{in}$ are sampled in Algorithm 13. With outlier detection methods for high-dimensional spaces (FastABOD or LoOP), we use the large subspaces as inlier subspaces (right green area). For the other outlier detection methods, 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 sampling guarantees that attributes from inlier and outlier subspaces overlap.

Algorithm 13 Sample inlier and outlier subspaces.

```
Input: Number of subspace $|\mathcal{SC}|$, $\mathcal{F}$, $\text{dect}^{\mathcal{S}}(\cdot)$
Output: Set of inlier and outlier subspaces
 1: if $\text{dect}^{\mathcal{S}}(\cdot)$ is FastABOD or LoOP then
 2:     $\mathcal{SC}_{cand} = \{\mathcal{S} \subseteq \mathcal{F} \colon |\mathcal{S}| \geq d - 2\}$
 3: else
 4:     $\mathcal{SC}_{cand} = \{\mathcal{S} \subseteq \mathcal{F} \colon |\mathcal{S}| \leq 2\}$
 5: end if
 6: $\mathcal{SC}_{in}$ = Sample $|\mathcal{SC}|$ subspaces from $\mathcal{SC}_{cand}$
 7: $\widetilde{\mathcal{F}}$ = Set of all attributes that are in at least one subspace of $\mathcal{SC}_{in}$
 8: $\mathcal{SC}_{cand} = \{\mathcal{S} \subseteq \widetilde{\mathcal{F}} \colon |\mathcal{S}| = \lfloor \frac{d}{2} \rfloor\}$
 9: $\mathcal{SC}_{out}$ = Sample $|\mathcal{SC}|$ from combs $\mathcal{SC}_{cand}$
10: return $\mathcal{SC}_{in}$, $\mathcal{SC}_{out}$
```

5.6.2 Evaluating Our Theoretical Findings

In the first experiments, we investigate the generalizability of our theoretical findings from Section 5.5.3. The experiments approximate the scenarios described in the theorems and hypotheses using various data sets and outlier detection methods, cf. Section 5.6.1.

Theorem 1

Theorem 1 states that hidden outliers exist when either $\mathcal{SC}_{in}$ or $\mathcal{SC}_{out}$ is a partition of the full attribute space $\mathcal{F}$ into subspaces, and the other one is $\mathcal{F}$ 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 of $\mathcal{SC}$s by randomly dividing $\mathcal{F}$ into partitions. For each outlier detection scheme and $\mathcal{SC}$ we compute the maximal proportion of hidden outliers regarding two selections of $\mathcal{SC}_{in}$ and $\mathcal{SC}_{out}$. With the first one, $\mathcal{SC}_{in}$ equals $\mathcal{SC}$ and $\mathcal{SC}_{out} = \mathcal{F}$. Vice versa in the other case, $\mathcal{SC}_{out}$ equals $\mathcal{SC}$ and $\mathcal{SC}_{in} = \mathcal{F}$. 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 5.1 lists the percentages of runs where we have been able to hide outliers.

Table 5.1: Percentage of runs with more than zero hidden outliers.

(Values in %)	mdist	DBOut	LoOP	FastABOD
$\mathcal{SC}_{out} = \mathcal{F}$	64.29	77.86	87.14	95.36
$\mathcal{SC}_{in} = \mathcal{F}$	37.86	87.50	96.43	95.71

In most cases, our algorithm can generate hidden outliers. Surprisingly, regarding mdist in particular, the success rate is rather low. This low success 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.

Hypothesis 2

Hypothesis 2 states that it is more difficult to generate inliers in correlated subspaces in setting $h_{1|d}$ and less difficult in setting $h_{\mathcal{F}}$. To investigate this hypothesis, we try to hide outliers in both settings using different outlier detection schemes. In one data set, we have correlated attributes (MVN cor.). In another one, they are not (MVN). If Hypothesis 2 holds we should see an increase in the proportion of hidden outliers from uncorrelated to correlated data in $h_{\mathcal{F}}$ and a decrease in $h_{1|d}$. Table 5.2 lists the results: the percentage obtained in each data set, the raw difference between the two results and a relative difference. We obtain the last entry by dividing the raw difference by the maximal percentage the detection algorithm has obtained in any of the two data sets.

Table 5.2: Difference in percentage of hidden outliers.

(Values in %)	mdist	DBOut	LoOP	FastABOD
MVN	28.14	56.54	12.96	1.12
MVN cor.	64.78	69.84	69.20	1.26
Difference	36.64	13.30	56.24	0.14
Relative	56.56	19.04	81.27	11.11

(a) $h_{\mathcal{F}}$

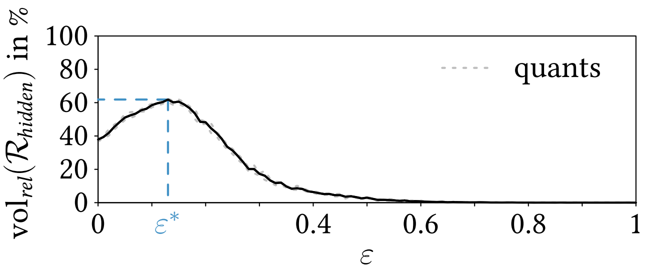
(Values in %)	mdist	DBOut	LoOP	FastABOD
MVN	0.84	0.62	29.08	21.50
MVN cor.	0.36	0.30	19.68	22.00
Difference	-0.48	-0.32	-9.40	-0.50
Relative	-57.14	-51.61	-32.32	2.27

(b) $h_{1|d}$

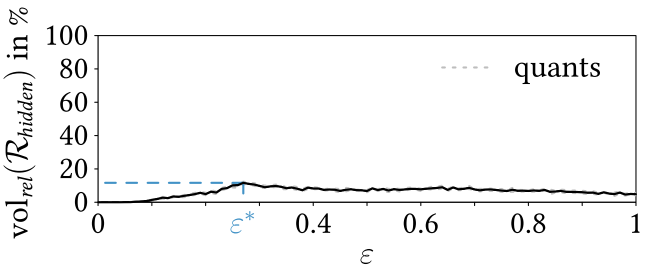
All detection methods except for FastABOD meet the expectation. We find it interesting that the magnitude of change of proportion is very different for $h_{\mathcal{F}}$ and $h_{1|d}$. However, the proportion of generated hidden outliers on each data set also varies greatly. In summary, although the extents are different, the experiments confirm the hypothesis to some extent.

5.6.3 Investigating the Optimal Parameter

The next experiments target at a crucial parameter of our generation technique. The proposed technique uses a sampling distribution parametrized by ε. The ε that maximizes the proportion of hidden outliers, defined as ε^*, is important: It allows quantifying the risk of data owners. We now investigate the dependency between ε and the proportion of generated hidden outliers, to analyze if ε^* usually exists (i.e., if there is a global maximum of the dependency).



(a) $h_{1|d}$, Parkinson and LoOP. Risk: 0.21.



(b) $h_{\mathcal{F}}$, Lymphography and mdist. Risk: 0.16.

Figure 5.9: Proportion of hidden outliers in generated instances versus ε.

Figures 5.7 and 5.9 illustrate the dependency between the proportion of hidden outliers and ε. In both figures, there is a very distinct ε^*. However, Figure 5.9b shows that this is not always the case. The risk is highest with 0.45 in Figure 5.7. The ε^* as well as the maximal proportion are relatively high. Figure 5.9b has the lowest risk with 0.12.

5.6.4 General Quality

In these final experiments, we study the general ability of our algorithm to generate hidden outliers. We also compare our technique to a baseline that does not feature the parameter ε. We choose uniform full space sampling as this baseline, which is equivalent to fixing ε to 1 (i.e., a purely exhaustive strategy). We will declare success if there is an increase in the quality of generating hidden outliers, and this increase is significant (e.g., a factor of at least two or three). Recall that our algorithm requires data, $\text{dect}^{\mathcal{S}}(\cdot)$, $\mathcal{SC}_{in}$ and $\mathcal{SC}_{out}$ as input. The subspace selection has been derived in Section 5.6.1. Regarding the data, we look at all data sets introduced in Section 5.6.1 and sample different numbers of attributes and instances. However, we only downsample, upsampling would lead to redundant data. To vary $\text{dect}^{\mathcal{S}}(\cdot)$, we use all outlier detection methods introduced in Section 5.6.1. Additionally, we obtain candidates for ε by using a fixed sequence of values instead of a heuristic. This way of obtaining candidates allows for further analysis of the effect of ε and a straightforward comparison to the baseline. We summarize our results to highlight the effect of the number of attributes or instances, used data set or detection method, and ε.

Number of Attributes

Figure 5.10 graphs the effect of the number of attributes. The y-axis displays the share of hidden outliers among the artificial instances (i.e., the success of the generation). For our technique and the baseline, the figure shows box plots of the share of hidden outliers. We observe a significant improvement of our technique over the baseline — for high dimensional data in particular. Second, for both alternatives, it seems to be more challenging to generate hidden outliers in high dimensional data sets. Two effects cause the shape of the left plot. One is that it is challenging to generate hidden outliers with few attributes. Any subspace-based outlier detection scheme will most likely detect them. If we increase the number of attributes, this effect decreases. However, the other influence is that increasing the number of attributes also increases the number of subspaces searched for outliers. So it is tough to find instances that are inliers in each subspace of $\mathcal{SC}_{in}$. In consequence, it also is difficult to generate hidden outliers.

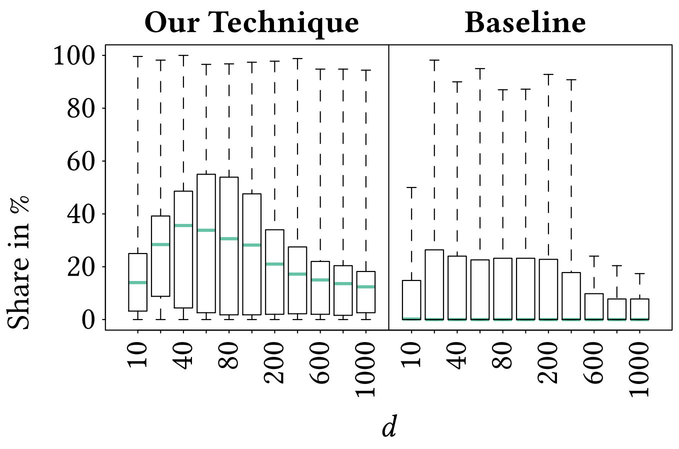


Figure 5.10: Effect of the number of attributes.

Number of Instances

Figure 5.11 plots the number of instances 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. In both cases, the effect of the number of instances is not very significant. This insignificance is because the number of instances does not change the data distribution much: dense areas remain dense and sparse areas sparse.

Data Sets Used

Table 5.3 displays the effect of the data set. We summarize the results across all experiments (i.e., with different samplings of instances and attributes). As before, our algorithm

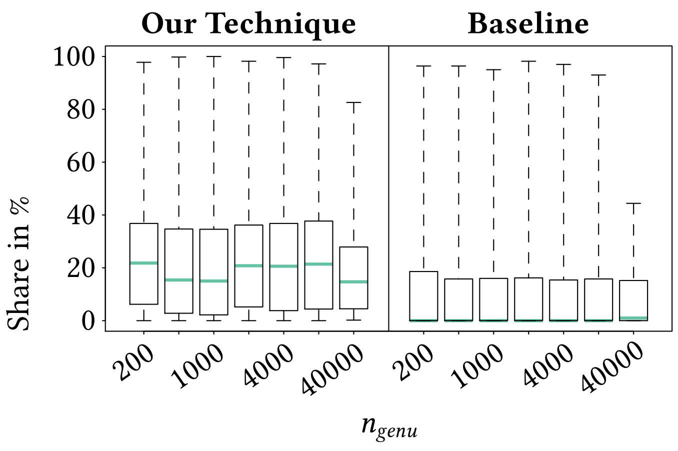


Figure 5.11: Effect of the number of instances.

outperforms the baseline significantly. However, we see that there are drastic differences between data sets. While generating hidden outliers is successful when using Madelon, this is more difficult with, say, InternetAds. We speculate that different data densities cause this effect. We have seen this effect in our experiments in Section 5.6.2 as well.

Measuring Data Set Characteristics

We now want to investigate further the difference in the proportion of successfully generated hidden outliers regarding the data sets used. Thus, we have conducted an intensive and systematic study, as follows: We have developed some measures that we then have correlated with the characteristics in Table 5.3. Our measures are divide into three categories that address the subspaces selected, the number of irrelevant attributes, and the quality of outlier detection. The first category of measure addresses the correlation in subspaces. As part of our formal results, we have already derived that this correlation should affect the region of hidden outliers. In particular, we take the average of the Spearman and Pearson correlation in both subspace collections ($\mathcal{SC}_{in}$ and $\mathcal{SC}_{out}$). When a subspace has more than two attributes, the correlation measure is the average of each pair of attributes. Regarding the second category, one measure uses a Principal Component Analysis (PCA) computed on each data set. Each Principal Component (PC) has a score dedicated to its importance in explaining the data instances. If there are only a few PCs with a very high score and many with low scores, the data set has a rather low intrinsic dimensionality. A low intrinsic dimensionality means that only a few transformed attributes are necessary to account for most variation within the data set. With few noisy (i.e. highly variable) attributes there might also be few irrelevant attributes, which might affect our generation: The number of irrelevant attributes could reduce the share of hidden outliers by making it difficult to generate them. However, it might also be that it increases their proportion, by blurring low dimensional outliers so that they are inliers in higher dimensions. This blurring is why we have quantified the skewness of the PCA scores. If this skewness is high, there are likely only a few PCs with a high score. All our data sets are benchmark data sets for outlier detection. Hence, they include labels for outliers and inliers so that one can use them to evaluate outlier detection methods (cf. Section 2.1). For the remain-

Table 5.3: Proportion of hidden outliers regarding data set used.

Data set	Our Technique (in %)	Baseline (in %)
ALOI	18.12	7.08
Annthyroid	13.07	7.13
Arrhythmia	23.64	8.23
Cardiotocography	24.57	7.29
Gisette	33.10	7.72
HeartDisease	29.53	6.85
InternetAds	6.29	3.54
Ionosphere	40.65	15.13
KDDCup99	16.32	10.73
Madelon	33.84	20.03
MVN	30.08	11.17
MVN cor.	18.14	5.49
PenDigits	21.72	5.34
SpamBase	21.27	7.06
Waveform	27.08	13.28
WDBC	22.26	5.93

ing measures, we have fitted a random forest to each data set that distinguishes between the labeled inliers and outliers. We can derive measures from this random forest fit for two of our categories: the importance of attributes that belongs to the second category and the quality of the inlier/outlier-classification, which is of the third category. In line with the usual definition, the importance of an attribute is the Gini index decrease with this attribute. If it is high, it is crucial for the classification. From the attribute importance values, we compute the mean value, variance, and skewness. Thus we measure, how vital the attributes in general, how much this varies, and whether there only are a few vital attributes. From the classification, we obtain accuracy, sensitivity (how well are outliers detected), and specificity (how well are inliers detected). We then have computed the correlation between all these measures and the percentage of hidden outliers generated by our algorithm. The results are listed in Table 5.4. We have separated them by the type of inlier subspaces: low- (up to 2 attributes) or high-dimensional (at least d-2 attributes).

We see that many measures influence the percentage of hidden outliers generated. As expected, the correlation affects this percentage. However, the effect in the low-dimensional setting is not very prominent. This small effect may be a result of the diverse subspaces we average the scores from, in the low dimensional setting in particular. The feature "importance" has an overall negative effect, although the mean is not significant in the high dimensional setting. This negative effect makes sense: If there are only a few

Table 5.4: Correlation of data measurements with percentage of hidden outliers.

Measure	Low	High
Average Pearson (Inlier)	0.32	0.30
Average Spearman (Inlier)	0.04	0.14
Average Pearson (Outlier)	0.21	0.45
Average Spearman (Outlier)	-0.06	0.20
Importance Mean	-0.24	0.06
Importance Variance	-0.29	-0.07
Importance Skewness	-0.20	-0.39
PCA Skewness	0.00	0.16
Accuracy	-0.35	-0.45
Specificity	-0.16	-0.42
Sensitivity	-0.41	0.24

vital attributes that are well suited for outlier detection, it is more difficult to hide from these. To illustrate, think of a data set with four attributes. The first two contain outliers that are easily detectable (see Figure 5.1, for instance). In the other two attributes, the outlier and inlier class are scattered randomly and are indifferent. Since outliers and inliers are indifferent, the subspaces likely follow a distribution that makes it difficult for an instance to be an outlier. For example, this is the case with an independent uniform distribution. However, hidden outliers must be outlying in certain subspaces. If this subspace consists of irrelevant attributes (here, the third and the fourth attribute), generating outliers might be difficult. In the high-dimensional setting, the PCA skewness seems to have an effect. This effect might be because the attributes that are more than the intrinsic dimensionality allow for a generation of many hidden outliers by blurring the detection result. Remember that in the high-dimensional setting, the inlier subspaces ($\mathcal{SC}_{in}$) have more attributes. The accuracy has a negative effect. Clearly, the better the random forest is in detecting outliers, the more difficult it is to hide such. Regarding specificity, the effect is the same. The better the random forest can detect inliers, the less effective is our generation. Regarding the sensitivity, we do not see an obvious interpretation of the results. The effect is opposite for the high- and the low-dimensional settings.

To conclude, we have experimented with various measures to quantify effects that go along with different shares of successfully generated hidden outliers. However, the main takeaway is that there is not one single measure or a few measures in combination correlated with a successful generation.

Outlier Detection Method Used

To determine the 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 technique as well as for the baseline. See Table 5.5. 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 methods.

Table 5.5: Effect of the used detection method on the success in hiding outliers and their average proportion.

(Values in %)	mdist	DBOut	LoOP	FastABOD
Our Technique	99.32	81.19	97.70	96.06
Baseline	4.46	36.49	21.81	95.78

(a) Percentage of runs with generated hidden outliers.

(Values in %)	mdist	DBOut	LoOP	FastABOD
Our Technique	7.67	30.28	30.13	26.41
Baseline	0.03	8.61	3.31	22.26

(b) Average share of hidden outliers.

Parameter Used

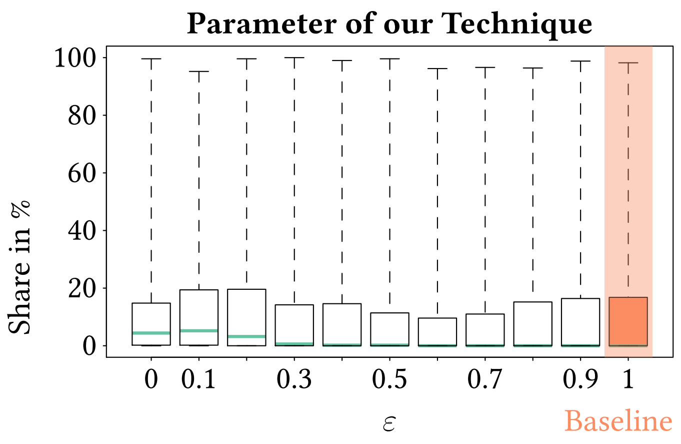


Figure 5.12: Effect of ε.

Figure 5.12 displays the proportion of hidden outliers versus ε. The value $\varepsilon = 1$ indicates our baseline (i.e., uniform sampling). The median has a peak when ε is about 0.1. This value results in hidden outliers generated such that they closely surround other instances, which confirms Hypothesis 1. Thus, hidden outliers are spatially close to the instances from $\underline{X}$. The figure also confirms the superiority of our algorithm over the baseline.

Summary of Experiments

The experiments have shown that in many scenarios, our algorithm can generate hidden outliers irrespective of the data set or the detection method used. Further, our generation technique is a significant improvement over the baseline: our technique has improved the result by a factor of three or more in many settings.

5.7 Chapter Summary

In this chapter, we have analyzed the characteristics of hidden outliers. These are outliers that are only detectable in certain attribute subspaces. Our analysis includes both formal results based on model assumptions and a proposal for an algorithm that generates 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 can have a significant effect on the ease of hiding outliers. We evaluate the generalizability 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 the generation technique we propose improves the results of our algorithm with a reference baseline significantly. We deem this algorithm a central aspect in a proof-of-concept for the evaluation of SSOD methods through generated hidden outliers.

6 Conclusions and Outlook

The reliable detection of outliers is essential, be it to clean a data set from any corruption, understand the data better, or predict and possibly prevent faults. This thesis has analyzed the concept of generated outliers in great depth: From the techniques to generate them to different ways of making use of them. Unlike genuine outliers, generated outliers are not rare and feature rather precise characteristics. Both properties help to face central challenges in the outlier detection problem, especially in the calibration and evaluation of detection methods.

One contribution of this thesis is a structured overview of the state-of-the-art in outlier generation. Of specific relevance in this regard are outliers generated in terms of available genuine instances (i.e., real ones). Ultimately the artificial outliers are used in combination with the genuine instances to calibrate outlier detection methods. In principle, this means finding suitable parameters. With the overview developed in this thesis, a comparison of the workflow, goals, and characteristics of the generated outliers from existing techniques is simple. A developed description of common problems faced when generating outliers increases the comprehension of the topic by much. For example, the discussion of the possible effects and uses for different distances of generated outliers to genuine instances. In addition to this, we perform experiments that compare the different generation techniques but also two different ways of calibrating detection methods through artificial outliers. A key takeaway from these experiments is that no generation technique always performs better than all others. Thus, we developed a simple decision process that guides practitioners in the usage of artificial outliers for calibrating detection methods.

From the literature on outlier detection methods but also from our overview of generation techniques, we observe that there exist many types of outliers. Different types of outliers feature different characteristics. Conventional approaches for evaluating and comparing outlier detection methods can cope with different types of outliers to some extent only. Thus, we strived for a more controlled and distinctive evaluation scheme by using artificial data. In artificial data, outliers and inliers come from some generative model and are thus not genuine. It is possible to generate outliers in such data with precise and controllable characteristics (i.e., types). To this end, our second contribution is a general process for the extensive evaluation of outlier detection methods using artificial data. In a nutshell, the process describes the generation of artificial data with realistic inliers and outliers of a specific type. We also present three instantiations of this process for some common types of outliers. The process and its instantiations are useful for finding a suite of well-performing outlier detection methods in terms of different types. This suite is then useful in detecting outliers without much prior knowledge on the types of outliers that might be present in some data set.

There exist outlier detection methods that detect outliers, not in the full data space, but a set of subspaces. Each subspace is a subset of the data attributes. Hidden outliers are, in principle, a blind spot for methods that detect outliers by the use of subspaces. Such outliers can be of any type mentioned before but additionally depend on the selected subspaces. In some subspaces, hidden outliers are detectable; in others, they are not. From the subspaces they are not detectable in, they are hidden. Our third contribution is to extend the usage of generated outliers to the concept of hidden outliers. Of particular interest for this are the circumstances under which hidden outliers exist at all. First, we analyze these circumstances theoretical. This analysis requires restrictive assumptions, which is the reason we extend the theoretical study through generated hidden outliers. Developing and using a tailored generation technique also allows us to analyze more general circumstances that allow for hidden outliers and characteristics of therein. For example, the usage of generated hidden outliers confirms the hypothesis that they tend to be spatial close to genuine inliers. Hence, our analysis of hidden outliers does not just reveal that they exist but also highlights other factors that affect their occurrence. As mentioned, the generated hidden outliers are blind spots in terms of a specific selection of subspaces. Thus, the algorithm we designed to generate hidden outliers might ultimately be useful in systematic evaluations for outlier detection methods based on subspaces.

The two central challenges for outlier detection faced in this thesis are that outliers are rare, and their notion is not precise. Our first contribution tackles, especially the challenge that outliers are rare. The overview of existing techniques and experiments comparing them illustrates many ways in which the calibration of outlier detection methods is possible without any genuine outliers. In tackling the imprecise notion of outliers, the overview is helpful as well: That there can be different outliers with characteristics defined by generation processes becomes apparent with it. This observation is the basis for the process from our second contribution. In its distinctive and controllable way of evaluating and comparing detection methods, the process reduces the negative effect of the overall imprecise notion of outliers. Our third contribution tackles both challenges as well. By generating hidden outliers, we do not just overcome the absence of labeled examples in real data but also identify key characteristics that confine the notion of hidden outliers.

The research carried out within this dissertation does give various starting points for future research. For instance, the overview of the state-of-the-art in generating outliers is one starting point. From the overview and the overall issues encountered by generation techniques, numerous challenges that require attention arise. For example, the precise distance of generated outliers and genuine instances remains unclear. Our experiments show that outliers generated close to genuine instances usually perform well in calibrating outlier detection methods. In the limit, when generating outliers precisely like the genuine instances, the outliers might not be useful anymore. Thus, it seems interesting to investigate the limits of the closeness of generated outliers and genuine instances. However, this does depend on many other factors, like the types of outliers generated. For this reason, we deem answering this question with some generality an exciting challenge for future research. A similar question arises from the effect of the number of artificial outliers generated. This number can affect the calibration of detection methods. However, we are not aware of any sophisticated general guidelines for choosing it. Another direction of future research we deem highly profitable is the possible synergy of techniques presented in this thesis and techniques that generate outliers using some genuine outliers. Outliers generated with the techniques presented in this thesis could extend the variation of the few genuine outliers. Thus, it might be that calibrating detection methods using artificial outliers generated with and without genuine examples achieves a better generalization.

The types of outliers we propose with the three instantiations of our general process in our second contribution are common. However, there are many more types of outliers possible. Thus, we deem it a vital task of future research to develop more instantiations of our process. For example, there is yet no instantiation for outliers that are outliers as a small group and not just on their own. The development of further instantiations of our process might lead to a broad but somewhat complete categorization of types of outliers and thus resolve the imprecise notion of outliers ultimately. For this goal, a categorization of available genuine outliers in terms of different types might be of great help as well.

Our evaluation of the concept of hidden outliers can also serve as the basis for further research. For example, the factors that influence the occurrence of hidden outliers we identified might lead to an improved technique for generating them. The effect of correlation among data attributes might be useful in this spirit. Such an improved technique, but also our current technique, could be adopted to improve schemes for subspace search outlier detection themselves. In essence, hidden outliers represent blind spots of outlier detection methods that use subspaces. This observation could be the basis for an iterative scheme to find a set of subspace that is not prone to any hidden outliers: In one iteration, outliers are generated such that they are hidden from an initial set of subspaces. In the following step, the selection of subspaces is modified such that the hidden outliers generated in the previous step can be detected. Then the first step is repeated with the new selection of subspaces. Repeating this process might result in a subspace selection that is least prone to hidden outliers. However, the details of an iterative process in this spirit are somewhat unclear — for example, how to modify the selection of subspaces. This is the reason we leave this idea for future work.

All in all, this dissertation yields a detailed and structured overview of the topic of generated outliers and shows why such outliers are useful for the calibration and evaluation of outlier detection methods.

A Appendix

The appendix is almost identical to (Steinbuss and Böhm, 2017), previously published in the International Journal of Data Science and Analytics. Adjustments are to ensure consistency for this dissertation.

A.1 Prerequisites for Proofs

The Mahalanobis distance is defined as

$$\text{mdist}^{\mathcal{F}}(\vec{x}) = \sqrt{(\vec{x}-\vec{\mu})^T \underline{\Sigma}^{-1} (\vec{x}-\vec{\mu})}, \tag{A.1}$$

where $\vec{\mu}$ is the mean vector and $\underline{\Sigma}$ the covariance matrix. W.l.o.g. we assume that $\vec{\mu} = \vec{0}$. Since the data is MVN distributed, mdist^2 is χ^2_{DF} distributed. The degrees of freedom (DF) are determined by the dimension of the data. We can write

$$\begin{aligned}\left[\text{mdist}^{\mathcal{F}}(\vec{x})\right]^2 &= \vec{x}^T \underline{\Sigma}^{-1} \vec{x} & \text{(A.2)}\\ &= \sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} x^{(i)} \cdot \sigma_{-1}^{(i,j)} \cdot x^{(j)}, & \text{(A.3)}\end{aligned}$$

where $\sigma_{-1}^{(i,j)}$ denotes the entry in the jth column and ith row of the inverse of the covariance matrix. If the attributes are i.i.d. according to a standard normal, this reduces to

$$\left[\text{mdist}^{\mathcal{F}}(\vec{x})\right]^2 = \sum_{i\in\mathcal{F}} \left[x^{(i)}\right]^2. \tag{A.4}$$

To test an instance for outlier or inlier, we used the outlier initialization formalized in Section 5.5.2. 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_{DF} distribution can be approximated by a normal distribution with mean DF and variance $\sqrt{2 \cdot \text{DF}}$). Thus,

$$\begin{aligned}\text{quant}(\alpha,\ \text{DF}) &\approx \text{DF} + \sqrt{2\cdot\text{DF}}\, z_\alpha \quad \text{and} & \text{(A.5)}\\ \frac{\partial\, \text{quant}(\alpha,\ \text{DF})}{\partial\, \text{DF}} &\approx 1 + \frac{z_\alpha}{\sqrt{2\cdot\text{DF}}}, & \text{(A.6)}\end{aligned}$$

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

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

A.2 Proofs of Theorems

A.2.1 Theorem 1

In this proof we will use the normal approximation given in Equation (A.5). From attributes being i.i.d. according to a standard normal distribution and partitioning $\mathcal{SC}$ of $\mathcal{F}$ into subspaces follows

$$\left[\mathrm{mdist}^{\mathcal{F}}(\vec{x})\right]^2 = \sum_{i\in\mathcal{F}} \left[x^{(i)}\right]^2 = \sum_{\mathcal{S}\in\mathcal{SC}} \sum_{i\in\mathcal{S}} \left[x^{(i)}\right]^2 \tag{A.7}$$

$$= \sum_{\mathcal{S}\in\mathcal{SC}} \left[\mathrm{mdist}^{\mathcal{S}}(\vec{x})\right]^2 . \tag{A.8}$$

We first prove that an instance $\vec{h}_1$ with

$$h_1^{(i)} = \begin{cases} \sqrt{\frac{\mathrm{quant}(\alpha,\ |\mathcal{F}|)}{|\mathcal{S}|}} & \text{if } i \in \mathcal{S}, \\ 0 & \text{otherwise,} \end{cases} \tag{A.9}$$

is an outlier for a $\mathcal{S} \in \mathcal{SC}$ but an inlier for $\mathcal{F}$. We know that

$$\mathrm{mdist}^{\mathcal{F}}\left(\vec{h}_1\right) = \mathrm{quant}(\alpha,\ |\mathcal{F}|) = \mathrm{mdist}^{\mathcal{S}}\left(\vec{h}_1\right) . \tag{A.10}$$

Since the quantile function is strictly monotonic increasing (cf. Equation (A.5)),

$$\mathrm{quant}(\alpha,\ |\mathcal{F}|) > \mathrm{quant}(\alpha,\ |\mathcal{S}|) . \tag{A.11}$$

Thus, $\vec{h}_1$ is an inlier regarding $\mathcal{F}$ but an outlier in $\mathcal{S}$. It is important to note that $\vec{h}_1$ is an outlier only regarding subspace $\mathcal{S}$ and not regarding any other subspace in $\mathcal{SC}$.

Moreover, an instance $\vec{h}_2$ defined by

$$h_2^{(i)} = \frac{\sqrt{\mathrm{quant}(\alpha,\ |\mathcal{S}|)}}{|\mathcal{S}|} \qquad i \in \mathcal{S},\ \forall\, \mathcal{S} \in \mathcal{SC} \tag{A.12}$$

is an outlier for $\mathcal{F}$ but an inlier for all $\mathcal{S} \in \mathcal{SC}$. It holds that

$$\operatorname{mdist}^{\mathcal{F}}\left(\vec{h}_2\right) = \sum_{\mathcal{S} \in \mathcal{SC}} \operatorname{quant}(\alpha,\ |\mathcal{S}|)\,, \tag{A.13}$$

$$\operatorname{mdist}^{\mathcal{S}}\left(\vec{h}_2\right) = \operatorname{quant}(\alpha,\ |\mathcal{S}|)\,. \tag{A.14}$$

Further, we know that $\vec{h}_2$ is an outlier in the full space if

$$\operatorname{mdist}^{\mathcal{F}}\left(\vec{h}_2\right) > \operatorname{quant}(\alpha,\ |\mathcal{F}|)\,. \tag{A.15}$$

We also know that $|\mathcal{F}| = \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}|$. Hence, in order to show that $\vec{h}_2$ is a hidden outlier as specified, we have to show

$$\sum_{\mathcal{S} \in \mathcal{SC}} \operatorname{quant}(\alpha,\ |\mathcal{S}|) \overset{!}{>} \operatorname{quant}\left(\alpha,\ \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}|\right) \tag{A.16}$$

$$\sum_{\mathcal{S} \in \mathcal{SC}} \left[|\mathcal{S}| + \sqrt{2|\mathcal{S}|}\ z_\alpha\right] > \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}| + \sqrt{2 \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}|}\ z_\alpha \tag{A.17}$$

$$\sum_{\mathcal{S} \in \mathcal{SC}} \sqrt{2|\mathcal{S}|} > \sqrt{2 \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}|} \tag{A.18}$$

$$\sum_{\mathcal{S} \in \mathcal{SC}} \sqrt{|\mathcal{S}|} > \sqrt{\sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}|} \tag{A.19}$$

$$\left(\sum_{\mathcal{S} \in \mathcal{SC}} \sqrt{|\mathcal{S}|}\right)^2 > \sum_{\mathcal{S} \in \mathcal{SS}} |\mathcal{S}| \tag{A.20}$$

$$\sum_{\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{SC}} \sqrt{|\mathcal{S}_1|}\sqrt{|\mathcal{S}_2|} > \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}| \tag{A.21}$$

$$\sum_{\mathcal{S}_1 \neq \mathcal{S}_2 \in \mathcal{SC}} \sqrt{|\mathcal{S}_1|}\sqrt{|\mathcal{S}_2|} + \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}| > \sum_{\mathcal{S} \in \mathcal{SC}} |\mathcal{S}|. \tag{A.22}$$

Since $\mathcal{SC}$ is a non-trivial partition, i.e., $\mathcal{SC} \neq \{\mathcal{F}\}$, the term

$$\sum_{\mathcal{S}_1 \neq \mathcal{S}_2 \in \mathcal{SC}} \sqrt{|\mathcal{S}_1|}\sqrt{|\mathcal{S}_2|} \tag{A.23}$$

is greater than 0, and the inequality holds.

A.2.2 Theorem 2

This theorem relies on an assumption not explicitly listed in the body of the article. Let λ denote the eigenvalue of $\underline{\Sigma}_1$ (algebraic multiplicity of *d*). Further, let $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_d$ denote the eigenvalues of $\underline{\Sigma}_2$. We introduce $\zeta_1, \ldots, \zeta_d$ which satisfy $\lambda + \zeta_i = \tilde{\lambda}_i$. Our assumption is that the ζ_i's are symmetrical, i. e., for any $\zeta_j > 0$ there exist $\zeta_k = -\zeta_j$.

We know that, for both subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$, the volume of the full instance space $\text{vol}\big(\mathcal{R}_{full}\big)$ is equal. Using this we can write

$$\text{vol}_{rel}\big(\mathcal{R}^{\mathcal{S}}_{in}\big) \propto \text{vol}\big(\mathcal{R}^{\mathcal{S}}_{in}\big)\,. \tag{A.24}$$

$\text{vol}\big(\mathcal{R}^{\mathcal{S}}_{in}\big)$ is the volume of a d-ellipse. Let $\zeta_1, \ldots, \zeta_d$ be the eigenvalues of the covariance matrix within a subspace $\mathcal{S}$. Then

$$\text{vol}_{rel}\big(\mathcal{R}^{\mathcal{S}}_{in}\big) \propto \frac{2\pi^{\frac{d}{2}}}{d \cdot \Gamma(\frac{d}{2})} \sqrt{\text{quant}(\alpha,\ |\mathcal{S}|) \prod_{i=1}^{d} \zeta_i}. \tag{A.25}$$

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 $\underline{\Sigma}_1$ and $\underline{\Sigma}_2$ are the same. We have assumed that each attribute in $\mathcal{S}_1$ is i.i.d. according to a univariate Gaussian distribution with variance σ. Hence, $\underline{\Sigma}_1$ is a diagonal matrix with σ in each diagonal element. Thus, $\lambda = \sigma$ is the variance and each of the d eigenvalues of $\underline{\Sigma}_1$. $\underline{\Sigma}_2$ has off-diagonal elements. Hence, the eigenvalues can differ from the ones in $\underline{\Sigma}_1$. Using the equality of traces we infer that $\sum_{i=1}^{d} \zeta_i = 0$. In order to prove our theorem we need to show that

$$\prod_{i=1}^{d} \lambda = \lambda^d \geq \prod_{i=1}^{d} \tilde{\lambda}_i = \prod_{i=1}^{d} (\lambda + \zeta_i). \tag{A.26}$$

We introduce

$$\mathcal{I}_{>0} := \{i \in \{1, \ldots, d\} \mid \zeta_i > 0\}, \tag{A.27}$$
$$\mathcal{I}_{=0} := \{i \in \{1, \ldots, d\} \mid \zeta_i = 0\}. \tag{A.28}$$

Let further $m = |\mathcal{I}_{=0}|$. We can infer that $|\mathcal{I}_{>0}| = \frac{d-m}{2}$. Using this, we can write

$$\prod_{i=1}^{d} (\lambda + \zeta_i) = \left[\prod_{i \in \mathcal{I}_{=0}} \lambda\right] \left[\prod_{j \in \mathcal{I}_{>0}} (\lambda + \varepsilon_j)(\lambda - \zeta_j)\right] \tag{A.29}$$
$$= \lambda^m \left[\prod_{j \in \mathcal{I}_{>0}} (\lambda^2 - \zeta_j^2)\right] \tag{A.30}$$
$$= \lambda^m \left(\lambda^2\right)^{\frac{d-m}{2}} - \lambda^m \left[\prod_{j \in \mathcal{I}_{>0}} \zeta_j^2\right] \tag{A.31}$$
$$= \lambda^d \underbrace{-\lambda^m \left[\prod_{j \in \mathcal{I}_{>0}} \zeta_j^2\right]}_{\leq 0}. \tag{A.32}$$

Inserting this in Equation (A.26) directly proves the theorem. We can also infer that if there is a $\zeta_i > 0$, the statement of the theorem extends to

$$\mathrm{vol}_{rel}\left(\mathcal{R}_{in}^{\mathcal{S}_1}\right) > \mathrm{vol}_{rel}\left(\mathcal{R}_{in}^{\mathcal{S}_2}\right) . \tag{A.33}$$

Bibliography

K. Aas, C. Czado, A. Frigessi, and H. Bakken. Pair-Copula Constructions of Multiple Dependence. *Insurance: Mathematics and Economics*, 2009.

N. Abe, B. Zadrozny, and J. Langford. Outlier Detection by Active Learning. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 504–509, 2006.

C. C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2 edition, 2017. doi: 10.1007/978-3-319-47578-3.

G. Albuquerque, T. Lowe, and M. Magnor. Synthetic Generation of High-Dimensional Datasets. *IEEE Transactions on Visualization and Computer Graphics*, Dec. 2011.

F. Angiulli and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, 2002.

I. Arel, D. C. Rose, and T. P. Karnowski. Deep Machine Learning - A New Frontier in Artificial Intelligence Research. *IEEE Computational Intelligence Magazine*, 5(4):13–18, Nov. 2010. doi: 10.1109/MCI.2010.938364.

A. Bánhalmi, A. Kocsor, and R. Busa-Fekete. Counter-Example Generation-Based One-Class Classification. In *Machine Learning*, pages 543–550, 2007.

E. L. Barse, H. Kvarnstrom, and E. Jonsson. Synthesizing Test Data for Fraud Detection Systems. In *Computer Security Applications Conference*, Dec. 2003.

J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. Smith, and M. West. Generative or Discriminative? Getting the Best of Both Worlds. *Bayesian Statistics*, pages 3–24, 2007.

G. Biau, F. Chazal, D. Cohen-Steiner, L. Devroye, and C. Rodríguez. A Weighted k-nearest Neighbor Density Estimate for Geometric Inference. *Electronic Journal of Statistics*, 2011.

B. Biggio and F. Roli. Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning. *arXiv preprint arXiv:1712.03141*, 2017.

S. Boughorbel, F. Jarray, and M. El-Anbari. Optimal Classifier for Imbalanced Data Using Matthews Correlation Coefficient Metric. *PloS one*, page e0177678, 2017.

W. Brendel, J. Rauber, and M. Bethge. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *arXiv preprint arXiv:1712.04248*, 2017.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. *ACM Sigmod Record*, pages 93–104, 2000.

G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study. *Data Mining and Knowledge Discovery*, pages 1–37, 2016.

I. Carmichael and J. S. Marron. Data Science vs. Statistics: Two Cultures? *Japanese Journal of Statistics and Data Science*, 1(1):117–138, June 2018. doi: 10.1007/s42081-018-0009-3.

C. C. Chang and C. J. Lin. Training Nu-Support Vector Classifiers: Theory and Algorithms. *Neural Computation*, pages 2119–2147, 2001.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. Adaptive Computation and Machine Learning Series. MIT Press, 2010. ISBN 978-0-262-03358-9 978-0-262-51412-5.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002. doi: 10.1613/jair.953.

R. Curry and M. I. Heywood. One-Class Genetic Programming. In *Genetic Programming*, pages 1–12, 2009.

R. Curry, P. Lichodzijewski, and M. I. Heywood. Scaling Genetic Programming to Large Datasets Using Hierarchical Dynamic Subset Selection. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics.*, pages 1065–1073, 2007.

Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov. Good Semi-Supervised Learning That Requires a Bad GAN. In *Advances in Neural Information Processing Systems 30*, pages 6510–6520, 2017.

M. A. Davenport, R. G. Baraniuk, and C. D. Scott. Learning Minimum Volume Sets with Support Vector Machines. In *16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pages 301–306, 2006.

H. Deng and R. Xu. Model Selection for Anomaly Detection in Wireless Ad Hoc Networks. In *IEEE Symposium on Computational Intelligence and Data Mining*, pages 540–546, 2007.

C. Désir, S. Bernard, C. Petitjean, and L. Heutte. One Class Random Forests. *Pattern Recognition*, pages 3490–3506, 2013.

D. Dheeru and E. Karra Taniskidou. UCI Machine Learning Repository, 2017. URL http://archive.ics.uci.edu/ml.

R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. A Comparative Evaluation of Outlier Detection Algorithms: Experiments and Analyses. *Pattern Recognition*, Feb. 2018.

J. J. Downs and E. F. Vogel. A Plant-Wide Industrial Process Control Problem. *Computers & Chemical Engineering*, Mar. 1993.

R. El-Yaniv and M. Nisenson. Optimal Single-Class Classification Strategies. In *Advances in Neural Information Processing Systems*, pages 377–384, 2007.

A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. A Meta-Analysis of the Anomaly Detection Problem. *arXiv:1503.01158 [cs, stat]*, Mar. 2015.

A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. Systematic Construction of Anomaly Detection Benchmarks from Real Data. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, 2013.

W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan. Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 123–130, 2001.

W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan. Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. *Knowledge and Information Systems*, pages 507–527, 2004.

B. Fazekas and A. Kiss. Statistical Data Generation Using Sample Data. In *New Trends in Databases and Information Systems*, 2018.

S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 202–212, 1994.

E. Fouché and K. Böhm. Monte Carlo Dependency Estimation. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management*, pages 13–24. Association for Computing Machinery, 2019. doi: 10.1145/3335783.3335795.

J. V. Frasch, A. Lodwich, F. Shafait, and T. M. Breuel. A Bayes-True Data Generator for Evaluation of Supervised and Unsupervised Learning Methods. *Pattern Recognition Letters*, Aug. 2011.

U. Genschel. The Effect of Data Contamination in Sliced Inverse Regression and Finite Sample Breakdown Point. *Sankhya A*, 80(1):28–58, Feb. 2018. doi: 10.1007/s13171-017-0102-x.

M. Goldstein and S. Uchida. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE*, Apr. 2016.

F. Gonzalez, D. Dasgupta, and R. Kozma. Combining Negative Selection and Classification Techniques for Anomaly Detection. In *Proceedings of the 2002 Congress on Evolutionary Computation.*, pages 705–710, 2002.

F. A. González and D. Dasgupta. Anomaly Detection Using Real-Valued Negative Selection. *Genetic Programming and Evolvable Machines*, pages 383–403, 2003.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014a.

J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014b.

J. Hartung, B. Elpelt, and K.-H. Klösener. *Statistik: Lehr-und Handbuch der angewandten Statistik*. Walter de Gruyter, 2012.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*, volume 1 of *Springer Series in Statistics*. Springer, New York, second edition, 2009.

D. Hawkins. *Identification of Outliers*. Monographs on Statistics and Applied Probability. Springer Netherlands, 1980. doi: 10.1007/978-94-015-3994-4.

K. Hempstalk, E. Frank, and I. H. Witten. One-Class Classification by Combining Density and Class Probability Estimation. In *Machine Learning and Knowledge Discovery in Databases*, pages 505–519, 2008.

V. Hodge and J. Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126, Oct. 2004. doi: 10.1023/B:AIRE.0000045502.10941.a9.

S. Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics, Theory and Applications*, pages 65–70, 1979.

F. Iglesias, T. Zseby, D. Ferreira, and A. Zimek. MDCGen: Multidimensional Dataset Generator for Clustering. *Journal of Classification*, Apr. 2019.

F. Keller, E. Müller, and K. Böhm. HiCS: High Contrast Subspaces for Density-Based Outlier Ranking. In *International Conference on Data Engineering*, pages 1037–1048, 2012.

K. Kennedy. A Framework for Generating Data to Simulate Application Scoring. In *Credit Scoring and Credit Control XII*, 2011.

H. Kido, Y. Yanagisawa, and T. Satoh. An Anonymous Communication Technique using Dummies for Location-Based Services. In *Proceedings of the International Conference on Pervasive Services*, pages 88–97, 2005.

G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. Efficient Biased Sampling for Approximate Clustering and Outlier Detection in Large Data Sets. *Transactions on Knowledge and Data Engineering*, pages 1170–1187, 2003.

B. Krawczyk. Learning from Imbalanced Data: Open Challenges and Future Directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016. doi: 10.1007/s13748-016-0094-0.

H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-Based Outlier Detection in High-Dimensional Data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 444–452, 2008.

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. LoOP: Local Outlier Probabilities. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 1649–1652, 2009a.

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 831–838, 2009b.

H.-P. Kriegel, P. Kröger, and A. Zimek. Outlier Detection Techniques. *Tutorial at Knowledge Discovery and Data Mining*, 2010.

H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Interpreting and Unifying Outlier Scores. In *Proceedings of the SIAM International Conference on Data Mining*, pages 13–24, 2011.

A. Kumar, S. Mehta, and D. Vijaykeerthy. An Introduction to Adversarial Machine Learning. In *Big Data Analytics*, pages 293–299, 2017.

C. H. Lampert. Kernel Methods in Computer Vision. *Foundations and Trends®in Computer Graphics and Vision*, pages 193–285, 2009.

M. Lang, B. Bischl, and D. Surmann. batchtools: Tools for R to Work on Batch Systems. *The Journal of Open Source Software*, Feb. 2017.

T. Larsson. Fast and Tight Fitting Bounding Spheres. In *Proceedings of the Annual SIGRAD Conference*, pages 27–30, 2008.

A. Lazarevic and V. Kumar. Feature Bagging for Outlier Detection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 157–166, 2005.

K. Lee, H. Lee, K. Lee, and J. Shin. Training Confidence-Calibrated Classifiers for Detecting Out-Of-Distribution Samples. *arXiv preprint arXiv:1711.09325*, 2018.

Y. Li and L. Maguire. Selecting Critical Patterns Based on Local Geometrical and Statistical Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1189–1201, 2011.

F. T. Liu, K. M. Ting, and Z. Zhou. Isolation Forest. In *IEEE International Conference on Data Mining*, Dec. 2008.

M. Lovric. *International Encyclopedia of Statistical Science*. Springer, Berlin, Heidelberg, 2011.

R. Maitra and V. Melnykov. Simulating Data to Study Performance of Finite Mixture Modeling and Clustering Algorithms. *Journal of Computational and Graphical Statistics*, Jan. 2010.

H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, pages 50–60, 1947.

M. Mannino and A. Abouzied. Is this Real?: Generating Synthetic Data that Looks Real. In *ACM Symposium on User Interface Software and Technology*, Oct. 2019.

M. L. McHugh. Multiple Comparison Analysis Testing in ANOVA. *Biochemia Medica*, pages 203–209, 2011.

S. McLachlan, K. Dube, T. Gallagher, J. A. Simmonds, and N. Fenton. Realistic Synthetic Data Generation: The ATEN Framework. In *Biomedical Engineering Systems and Technologies*, 2019.

V. Melnykov, W.-C. Chen, and R. Maitra. MixSim: An R Package for Simulating Data to Study Performance of Clustering Algorithms. *Journal of Statistical Software*, 2012.

G. W. Milligan. An Algorithm for Generating Artificial Test Clusters. *Psychometrika*, Mar. 1985.

E. Müller, M. Schiffer, and T. Seidl. Statistical Selection of Relevant Subspace Projections for Outlier Ranking. In *International Conference on Data Engineering*, pages 434–445, 2011.

E. Müller, I. Assent, P. Iglesias, Y. Mülle, and K. Böhm. Outlier Ranking via Subspace Analysis in Multiple Views of the Data. In *12th International Conference on Data Mining*, pages 529–538, 2012.

T. Nagler. A Generic Approach to Nonparametric Function Estimation with Mixed Data. *arXiv:1704.07457 [stat]*, Apr. 2017.

B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S.-h. Lau, S. J. Lee, S. Rao, A. Tran, and J. D. Tygar. Near-Optimal Evasion of Convex-Inducing Classifiers. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 549–556, 2010.

J. Neugebauer, O. Kramer, and M. Sonnenschein. Instance Selection and Outlier Generation to Improve the Cascade Classifier Precision. In *Agents and Artificial Intelligence*, pages 151–170, 2016.

S. Olejnik and J. Algina. Generalized Eta and Omega Squared Statistics: Measures of Effect Size for Some Common Research Designs. *Psychological Methods*, pages 434–447, 2003.

N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy*, pages 372–387, 2016.

Y. Pei and O. Zaıane. A Synthetic Data Generator for Clustering and Outlier Analysis. Technical Report, University of Alberta, 2006.

T. S. Pham, Q. U. Nguyen, and X. H. Nguyen. Generating Artificial Attack Data for Intrusion Detection Using Machine Learning. In *Proceedings of the Fifth Symposium on Information and Communication Technology*, pages 286–291, 2014.

H.-P. Piepho. An Algorithm for a Letter-Based Representation of All-Pairwise Comparisons. *Journal of Computational and Graphical Statistics.*, pages 456–466, 2004.

P. Porwik, T. Orczyk, M. Lewandowski, and M. Cholewa. Feature Projection k-NN Classifier Model for Imbalanced and Incomplete Medical Data. *Biocybernetics and Biomedical Engineering*, 2016.

W. Qiu and H. Joe. Generation of Random Clusters with Specified Degree of Separation. *Journal of Classification*, Sept. 2006.

D. A. Rachkovskij and E. M. Kussul. DataGen: a Generator of Datasets for Evaluation of Classification Algorithms. *Pattern Recognition Letters*, May 1998.

S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. In *ACM Sigmod Record*, 2000.

X. Ren, K. Zhu, T. Cai, and S. Li. Fault Detection and Diagnosis for Nonlinear and Non-Gaussian Processes Based on Copula Subspace Division. *Industrial & Engineering Chemistry Research*, Oct. 2017.

M. Rogers, J. Graham, and R. P. Tonge. Using Statistical Image Models for Objective Evaluation of Spot Detection in Two-Dimensional Gels. *Proteomics*, June 2003.

T. Saito and M. Rehmsmeier. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PloS one*, Mar. 2015.

T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments.* Springer, New York, Mar. 2013. ISBN 9781475737998.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural computation*, pages 1443–1471, 2001.

E. Schubert, A. Zimek, and H.-P. Kriegel. Local Outlier Detection Reconsidered: A Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection. *Data Mining and Knowledge Discovery*, 28(1):190–237, Jan. 2014. doi: 10.1007/s10618-012-0300-z.

T. Shi and S. Horvath. Unsupervised Learning With Random Forest Predictors. *Journal of Computational and Graphical Statistics.*, pages 118–138, 2006.

K. Singh and D. S. Upadhyaya. Outlier Detection: Applications And Techniques. *International Journal of Computer Science*, 9(1):307–323, 2012.

G. Steinbuss and K. Böhm. Generating Artificial Outliers in the Absence of Genuine Ones — a Survey. *arXiv preprint arXiv:2006.03646*, 2020a.

G. Steinbuss and K. Böhm. Benchmarking Unsupervised Outlier Detection with Realistic Synthetic Data. *arXiv preprint arXiv:2004.06947*, 2020b.

G. Steinbuss and K. Böhm. Hiding Outliers in High-Dimensional Data Spaces. *International Journal of Data Science and Analytics*, pages 173–189, 2017.

D. Steinley and R. Henson. OCLUS: An Analytic Method for Generating Clusters with Known Overlap. *Journal of Classification*, Sept. 2005.

I. Steinwart, D. Hush, and C. Scovel. A Classification Framework for Anomaly Detection. *Journal of Machine Learning Research.*, pages 211–232, 2005.

Y. Sun, A. Cuesta-Infante, and K. Veeramachaneni. Learning Vine Copula Models For Synthetic Data Generation. *arXiv:1812.01226 [cs, stat]*, Dec. 2018.

L. Swersky, H. O. Marques, J. Sander, R. J. G. B. Campello, and A. Zimek. On the Evaluation of Outlier Detection and One-Class Classification Methods. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2016.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. *arXiv preprint arXiv:1312.6199*, 2013.

J. Sánchez-Monedero, P. A. Gutiérrez, M. Pérez-Ortiz, and C. Hervás-Martínez. An n-Spheres Based Synthetic Data Generator for Supervised Classification. In *Advances in Computational Intelligence*, 2013.

D. M. J. Tax and R. P. W. Duin. Support Vector Domain Description. *Pattern Recognition Letters*, pages 1191–1199, 1999.

D. M. J. Tax and R. P. W. Duin. Uniform Object Generation for Optimizing One-class Classifiers. *Journal of Machine Learning Research*, pages 155–173, 2001.

J. P. Theiler and D. Michael Cai. Resampling Approach for Anomaly Detection in Multispectral Images. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery IX*, pages 230–240, 2003.

H. Trittenbach and K. Böhm. Dimension-Based Subspace Search for Outlier Detection. *International Journal of Data Science and Analytics*, 7(2):87–101, Mar. 2019. doi: 10.1007/s41060-018-0137-7.

K. Tumer and J. Ghosh. Estimating the Bayes Error Rate Through Classifier Combining. In *International Conference on Pattern Recognition*, Aug. 1996.

S. Visa and A. Ralescu. Issues in Mining Imbalanced Data Sets - A Review Paper. In *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*, pages 67–73, 2005.

N. G. Waller, J. M. Underhill, and H. A. Kaiser. A Method for Generating Simulated Plasmodes and Artificial Test Clusters with User-Defined Shape, Size, and Orientation. *Multivariate Behavioral Research*, Apr. 1999.

C. Wan, Z. Li, and Y. Zhao. SynC: A Unified Framework for Generating Synthetic Population with Gaussian Copula. *arXiv:1904.07998 [cs, stat]*, Apr. 2019.

C.-K. Wang, Y. Ting, Y.-H. Liu, and G. Hariyanto. A Novel Approach to Generate Artificial Outliers for Support Vector Data Description. In *International Symposium on Industrial Electronics*, pages 2202–2207, 2009.

S. Wang, Q. Liu, E. Zhu, F. Porikli, and J. Yin. Hyperparameter Selection of One-Class Support Vector Machine by Self-Adaptive Data Shifting. *Pattern Recognition*, pages 198–211, 2018.

W. Xu, Y. Qi, and D. Evans. Automatically Evading Classifiers. In *Proceedings of the Network and Distributed Systems Symposium*, pages 21–24, 2016.

A. Zimek and P. Filzmoser. There and Back Again: Outlier Detection Between Statistical Reasoning and Data Mining Algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2018.

A. Zimek, E. Schubert, and H.-P. Kriegel. A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data. *Statistical Analysis and Data Mining*, pages 363–387, 2012.

A. Zimmermann. Method Evaluation, Parameterization, and Result Validation in Unsupervised Data Mining: A Critical Survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, July 2019.