Fast and Scalable Outlier Detection with Metric Access Methods

Altamir Gomes Bispo Junior

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Altamir Gomes Bispo Junior

Fast and Scalable Outlier Detection with Metric Access Methods

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Robson Leonardo Ferreira Cordeiro

USP – São Carlos
September 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

G633f Gomes Bispo Junior, Altamir
Fast and Scalable Outlier Detection with Metric
Access Methods / Altamir Gomes Bispo Junior;
orientador Robson Leonardo Ferreira Cordeiro. --
São Carlos, 2019.
88 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2019.

1. Mineração de Dados. 2. Detecção de Anomalias.
3. Métodos de Acesso Métrico. I. Leonardo Ferreira
Cordeiro, Robson, orient. II. Título.

Altamir Gomes Bispo Junior

Detecção Rápida e Escalável de Casos de Exceção com Métodos de Acesso Métrico

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Robson Leonardo Ferreira Cordeiro

USP – São Carlos
Setembro de 2019

I dedicate this work to God in all His infinite, enduring and everlasting Love. To my mother Darci, who took me here with her passionate support, motivating words and valuable knowledge. To my supportive father Altamir, brother Alexandre, and uncles Benedito and Fátima. To my Master's Dissertation Advisor, Professor Robson, to my Undergraduate Project Advisor, Professor Marcela, and to Professors Caetano and Agma – more than lending their valuable technical input, they have vested me with a complete way of life into the realm of scientific research. To my colleagues at the Data Bases and Images Group. To my comrades of our – now extinct – study group on human sciences, philosophy and theology, with all the valuable instruction therein, that showed to me sincere efforts for acquiring new knowledge always pay off. And to my brothers at the Ministério Universidades Renovadas (MUR).

ACKNOWLEDGEMENTS

This work was partially supported by the São Paulo Research Foundation (FAPESP) – Grant N° 2018/05714-5, by the Coordination for Improvement of Higher Education Personnel (CAPES) – Finance Code 001, and by the National Council for Scientific and Technological Development (CNPq).

*“Truth is not determined by a majority vote.”
(Pope Benedict XVI)*

RESUMO

GOMES, A. G. B. **Detecção Rápida e Escalável de Casos de Exceção com Métodos de Acesso Métrico**. 2019. 88 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

É conhecido e notável que os modelos teóricos existentes empregados na detecção de outliers realizam suposições que podem não refletir a verdadeira natureza dos outliers em cada aplicação. Esta dissertação descreve um estudo empírico sobre *detecção de outliers não-supervisionada* usando 8 algoritmos do estado-da-arte e 8 conjuntos de dados que foram extraídos de uma variedade de tarefas do mundo real de relevância prática, tais como a detecção de ataques cibernéticos, patologias clínicas e anormalidades naturais. Apresentam-se considerações sobre os resultados obtidos, apontando os pontos positivos e negativos de cada técnica do *ponto de vista do especialista da aplicação*, o que representa uma mudança do embasamento rotineiro no ponto de vista do desenvolvedor da técnica. A maioria das técnicas estudadas apresentou requerimentos de tempo impraticáveis ou falhou em encontrar o que os especialistas consideraram como outliers nos conjuntos de dados confeccionados por eles próprios. Para lidar-se com esta questão, foi desenvolvido o método *MetricABOD*: um novo algoritmo baseado no ABOD que torna a análise *milhares de vezes mais veloz*, sendo ainda em média *26% mais acurada* do que o trabalho relacionado mais acurado. Esta melhoria equivale a tornar a busca por outliers uma tarefa factível em muitas aplicações do mundo real para as quais os métodos existentes apresentam resultados instáveis ou requerimentos de tempo impassíveis de realização. Finalmente, foram também estudadas duas coleções de dados adimensionais para mostrar que o novo *MetricABOD* funciona também para dados puramente métricos.

Palavras-chave: Ciência Computacional Aplicada, Dados Complexos, Mineração de Dados, Detecção de Outliers Não-supervisionada, Métodos de Acesso Métrico.

ABSTRACT

GOMES, A. G. B. **Fast and Scalable Outlier Detection with Metric Access Methods**. 2019. 88 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

It is well-known that the existing theoretical models for outlier detection make assumptions that may not reflect the true nature of outliers in every real application. This dissertation describes an empirical study performed on *unsupervised outlier detection* using 8 algorithms from the state-of-the-art and 8 datasets that refer to a variety of real-world tasks of practical relevance, such as spotting cyberattacks, clinical pathologies and abnormalities occurring in nature. We present our lowdown on the results obtained, pointing out to the strengths and weaknesses of each technique from the *application specialist's point of view*, which is a shift from the designer-based point of view that is commonly adopted. Many of the techniques had unfeasibly high runtime requirements or failed to spot what the specialists consider as outliers in their own data. To tackle this issue, we propose *MetricABOD*: a novel ABOD-based algorithm that makes the analysis up to *thousands of times faster*, still being in average *26% more accurate* than the most accurate related work. This improvement is tantamount to practical outlier detection in many real-world applications for which the existing methods present unstable accuracy or unfeasible runtime requirements. Finally, we studied two collections of text data to show that our *MetricABOD* works also for adimensional, purely metric data.

Keywords: Applied Computational Sciences, Complex Data, Data Mining, Unsupervised Outlier Detection, Metric Access Methods.

LIST OF FIGURES

Figure 1 – A simple example of clustering from the classical Iris dataset (Fischer, 1936).	36
Figure 2 – ABOD’s angle-based analysis for an inlier (left) and an outlier (right).	38
Figure 3 – Angle values and their standard deviations (square roots of variances) as outlieriness scores for data in Figure 2.	38
Figure 4 – Basic clustering example where the points inside the cluster are inliers and the points outside the cluster are outliers.	40
Figure 5 – A convex hull made of outliers. They are highlighted here by a red contour.	40
Figure 6 – Dissimilarity matrix for two toy datasets with 10 random objects each. a) 5 attributes; b) 100 attributes.	41
Figure 7 – 2-dimensional objects at: a) level h of a tree-based MAM structure; b) level $h - 1$ of the same structure.	43
Figure 8 – An example of reachability distance as employed by LOF.	46
Figure 9 – Four counting neighborhoods and one sampling neighborhood.	50
Figure 10 – Intuition behind the ABOD algorithm. It is expected that an object inside a cluster will present the highest variance of angles. These angles are generated by the object itself and an arbitrary pair of other objects.	52
Figure 11 – Graph showing the spectra of angles for different types of points. Extremely high values of this spectra are assigned to inner objects and the converse is true for the outliers.	53
Figure 12 – Our proposed angle-based analysis for an object P . Circles in the 2-dimensional space illustrate the leaf nodes of any tree-based MAM. The new <i>MetricABOD</i> algorithm uses only the representatives A , B , C and D to compute the outlieriness score of P .	60
Figure 13 – Accuracy for the BreastW dataset as the number of retrieved objects increases.	67
Figure 14 – Accuracy for the Cardio dataset as the number of retrieved objects increases.	67
Figure 15 – Accuracy for the Satimage-2 dataset as the number of retrieved objects increases.	67
Figure 16 – Accuracy for the Shuttle dataset as the number of retrieved objects increases.	67
Figure 17 – Accuracy for the Annnthyroid dataset as the number of retrieved objects increases.	67
Figure 18 – Accuracy for the Mammography dataset as the number of retrieved objects increases.	67
Figure 19 – Accuracy for the Http dataset as the number of retrieved objects increases.	68

Figure 20 – Accuracy for the Musk dataset as the number of retrieved objects increases.	68
Figure 21 – Rank of Brazilian names as per their average L-Edit distances to all others. Orange lines are the top-20 outliers; lines with text are the top-5 ones. . . .	68
Figure 22 – Rank of NSF abstracts as per their average Jaccard distances to all others. Orange lines are the top-20 outliers; lines with text are the top-5 ones. . . .	68
Figure 23 – Histogram of position distances between same object for HiCS vs. aLOCI rankings – Cardio dataset.	80
Figure 24 – Histogram of position distances between same object for HiCS vs. KNN- Outlier rankings – BreastW dataset.	80
Figure 25 – Histogram of position distances between same object for HiCS vs. LOF rankings – BreastW dataset.	80
Figure 26 – Histogram of position distances between same object for aLOCI vs. KNN- Outlier rankings – BreastW dataset.	80
Figure 27 – Histogram of position distances between same object for aLOCI vs. LOF rankings – BreastW dataset.	80
Figure 28 – Histogram of position distances between same object for KNN-Outlier vs. LOF rankings – BreastW dataset.	80
Figure 29 – Histogram of position distances between same object for ABOD vs. FastA- BOD rankings – BreastW dataset.	81
Figure 30 – Histogram of position distances between same object for ABOD vs. LB- ABOD rankings – BreastW dataset.	81
Figure 31 – Histogram of position distances between same object for ABOD vs. FastVOA rankings – BreastW dataset.	81
Figure 32 – Histogram of position distances between same object for ABOD vs. <i>MetricA- BOD</i> rankings – BreastW dataset.	81
Figure 33 – Histogram of position distances between same object for ABOD vs. L1- Depth rankings – BreastW dataset.	81
Figure 34 – Histogram of position distances between same object for ABOD vs. Fast L1-Depth rankings – BreastW dataset.	81
Figure 35 – Histogram of position distances between same object for FastABOD vs. LB-ABOD rankings – BreastW dataset.	82
Figure 36 – Histogram of position distances between same object for FastABOD vs. FastVOA rankings – BreastW dataset.	82
Figure 37 – Histogram of position distances between same object for FastABOD vs. <i>MetricABOD</i> rankings – BreastW dataset.	82
Figure 38 – Histogram of position distances between same object for FastABOD vs. L1D rankings – BreastW dataset.	82
Figure 39 – Histogram of position distances between same object for FastABOD vs. FL1D rankings – BreastW dataset.	82

Figure 40 – Histogram of position distances between same object for LB-ABOD vs. FastVOA rankings – BreastW dataset.	82
Figure 41 – Histogram of position distances between same object for LB-ABOD vs. <i>MetricABOD</i> rankings – BreastW dataset.	83
Figure 42 – Histogram of position distances between same object for LB-ABOD vs. L1-Depth rankings – BreastW dataset.	83
Figure 43 – Histogram of position distances between same object for LB-ABOD vs. Fast L1-Depth rankings – BreastW dataset.	83
Figure 44 – Histogram of position distances between same object for FastVOA vs. <i>MetricABOD</i> rankings – BreastW dataset.	83
Figure 45 – Histogram of position distances between same object for FastVOA vs. L1-Depth rankings – BreastW dataset.	83
Figure 46 – Histogram of position distances between same object for FastVOA vs. Fast L1-Depth rankings – BreastW dataset.	83
Figure 47 – Histogram of position distances between same object for <i>MetricABOD</i> vs. L1-Depth rankings – BreastW dataset.	84
Figure 48 – Histogram of position distances between same object for HiCS vs. Fast L1-Depth rankings – Cardio dataset.	84
Figure 49 – Histogram of position distances between same object for L1-Depth vs. Fast L1-Depth rankings – BreastW dataset.	84
Figure 50 – Histogram of position distances between same object for HiCS vs. aLOCI rankings – Cardio dataset.	84
Figure 51 – Histogram of position distances between same object for HiCS vs. KNN-Outlier rankings – Cardio dataset.	84
Figure 52 – Histogram of position distances between same object for HiCS vs. LOF rankings – Cardio dataset.	84
Figure 53 – Histogram of position distances between same object for aLOCI vs. KNN-Outlier rankings – Cardio dataset.	85
Figure 54 – Histogram of position distances between same object for aLOCI vs. LOF rankings – Cardio dataset.	85
Figure 55 – Histogram of position distances between same object for KNN-Outlier vs. LOF rankings – Cardio dataset.	85
Figure 56 – Histogram of position distances between same object for ABOD vs. FastABOD rankings – Cardio dataset.	85
Figure 57 – Histogram of position distances between same object for ABOD vs. LB-ABOD rankings – Cardio dataset.	85
Figure 58 – Histogram of position distances between same object for ABOD vs. FastVOA rankings – Cardio dataset.	85

Figure 59 – Histogram of position distances between same object for ABOD vs. <i>MetricABOD</i> rankings – Cardio dataset.	86
Figure 60 – Histogram of position distances between same object for ABOD vs. L1-Depth rankings – Cardio dataset.	86
Figure 61 – Histogram of position distances between same object for ABOD vs. Fast L1-Depth rankings – Cardio dataset.	86
Figure 62 – Histogram of position distances between same object for FastABOD vs. LB-ABOD rankings – Cardio dataset.	86
Figure 63 – Histogram of position distances between same object for FastABOD vs. FastVOA rankings – Cardio dataset.	86
Figure 64 – Histogram of position distances between same object for FastABOD vs. <i>MetricABOD</i> rankings – Cardio dataset.	86
Figure 65 – Histogram of position distances between same object for FastABOD vs. L1-Depth rankings – Cardio dataset.	87
Figure 66 – Histogram of position distances between same object for FastABOD vs. Fast L1-Depth rankings – Cardio dataset.	87
Figure 67 – Histogram of position distances between same object for LB-ABOD vs. FastVOA rankings – Cardio dataset.	87
Figure 68 – Histogram of position distances between same object for LB-ABOD vs. <i>MetricABOD</i> rankings – Cardio dataset.	87
Figure 69 – Histogram of position distances between same object for LB-ABOD vs. L1-Depth rankings – Cardio dataset.	87
Figure 70 – Histogram of position distances between same object for LB-ABOD vs. Fast L1-Depth rankings – Cardio dataset.	87
Figure 71 – Histogram of position distances between same object for FastVOA vs. <i>MetricABOD</i> rankings – Cardio dataset.	88
Figure 72 – Histogram of position distances between same object for FastVOA vs. L1-Depth rankings – Cardio dataset.	88
Figure 73 – Histogram of position distances between same object for FastVOA vs. Fast L1-Depth rankings – Cardio dataset.	88
Figure 74 – Histogram of position distances between same object for <i>MetricABOD</i> vs. L1-Depth rankings – Cardio dataset.	88
Figure 75 – Histogram of position distances between same object for <i>MetricABOD</i> vs. Fast L1-Depth rankings – Cardio dataset.	88
Figure 76 – Histogram of position distances between same object for L1-Depth vs. Fast L1-Depth rankings – Cardio dataset.	88

LIST OF ALGORITHMS

Algorithm 1 – Orca outlier detection algorithm.	46
Algorithm 2 – RBRP outlier detection algorithm. Phase 1.	49
Algorithm 3 – RBRP outlier detection algorithm. Phase 2.	49
Algorithm 4 – LOCI outlier detection algorithm.	51
Algorithm 5 – Approximate aLOCI outlier detection algorithm.	52
Algorithm 6 – OUTRES outlier detection algorithm.	53
Algorithm 7 – The <i>MetricABOD</i> algorithm	59

LIST OF TABLES

Table 1 – Outlier detection methods compared, as for their suitability to high dimensional data.	54
Table 2 – Summary of datasets.	65
Table 3 – Accuracy results.	66
Table 4 – Runtime results in minutes.	66
Table 5 – Accuracy results for ABOD-based methods and L_1 -Depth-based ones.	78
Table 6 – Kendall rank correlation coefficients for the BreastW dataset.	78
Table 7 – Kendall rank correlation coefficients for the Cardio dataset.	79

LIST OF ABBREVIATIONS AND ACRONYMS

ABOD	Angle-Based Outlier Detection
aLOCI	Approximate Local Correlation Integral
DB-Outlier	Distance-Based Outlier
DBSCAN	Density Based Spatial Clustering of Application with Noise
FastABOD	Fast Angle-Based Outlier Detection
FastVOA	Fast Variance of Angles
HiCS	High-Contrast Subspaces
KNN-Outlier	K^{th} Nearest Neighbor Outlier
LB-ABOD	Lower Bound Angle-Based Outlier Detection
LOCI	Local Correlation Integral
LOF	Local Outlier Factor
OPTICS	Ordering Points to Identify Cluster Structure
OUTRANK	Outlier Ranking
OUTRES	Adaptive Outlierness for Subspace Outlier Ranking
RBRP	Recursive Binning and Re-Projection
VOA	Variance of Angles

LIST OF SYMBOLS

μ — expected value of a given variable

σ — standard deviation of a given variable

$d(a,b)$ — Distance from object a to the object b

$\check{d}(a,b)$ — Lower bound of the distance from object a to the object b

CONTENTS

1	INTRODUCTION	29
1.1	Motivation	29
1.2	Problem definition, objectives and hypothesis	30
1.3	Main contributions of this MSc work	31
1.4	Conclusions	32
2	BACKGROUND CONCEPTS	33
2.1	Data distributions	33
2.2	Extreme values in the data distribution	33
2.3	Outlier detection in Data Mining	34
2.3.1	<i>Summary of related concepts and initial presentation on state-of-art outlier detection methods</i>	35
2.4	Types of outlier detection methods	38
2.4.1	<i>Proximity-based outlier detection</i>	39
2.4.1.1	<i>Distance-based outlier detection</i>	39
2.4.1.2	<i>Density-based outlier detection</i>	39
2.4.1.3	<i>Clustering-based outlier detection</i>	39
2.4.2	<i>Depth based outlier detection</i>	40
2.5	Curse of high dimensionality	41
2.6	Metric Access Methods	41
2.7	Semantic knowledge and semantic awareness	43
2.8	Conclusions	44
3	OUTLIER DETECTION METHODS	45
3.1	State-of-the-art methods for outlier detection	45
3.1.1	<i>Orca</i>	45
3.1.2	<i>LOF</i>	45
3.1.3	<i>HiCS</i>	47
3.1.4	<i>OUTRANK</i>	47
3.1.5	<i>RBRP</i>	48
3.1.6	<i>LOCI</i>	50
3.1.7	<i>aLOCI</i>	51
3.1.8	<i>ABOD</i>	51

3.1.9	<i>OUTRES</i>	51
3.2	Summary of the existing approaches	53
3.3	Evaluation of outlier detection methods	54
3.4	Conclusions	55
4	METRIC ABOD	57
4.1	Conclusions	60
5	OUTLIER DETECTION AT WORK	61
5.1	Setup	61
5.2	Datasets	63
5.3	Results	64
5.4	Results: spotting outliers in adimensional data	67
5.5	Conclusions	69
6	CONCLUSION	71
BIBLIOGRAPHY		73
ANNEX A	ANNEX	77
A.1	L_1 -Depth	77
A.2	Correlation between outlier detection methods	77

INTRODUCTION

1.1 Motivation

An important task with high-impact real-world consequences is the forecasting and detection of extreme events and exception cases, like frauds in the financial sector, cyberattacks, clinical pathologies and abnormalities in nature. Such phenomena are known as outliers. The volume of data being collected by enterprises from a variety of scientific areas is increasing exponentially over time, and a neat aspect of this thing are the innumerable and potentially interesting patterns are hitherto hidden, thus calling into action the task of automatic outlier detection (FAN; LI, 2006; JOHNSTONE; TITTERINGTON, 2009).

After a thorough revision process, we came to the conclusion that the literature on the subject lacks ample and thoughtful comparison between the many existing outlier detection methods. One recent work (CAMPOS *et al.*, 2016) succeeded into reducing this gap, although it is limited to a “meta-analysis”. In other words, their work targets the clarification of ill-understood concepts discussed either thoroughly or just *en passant* by the literature, being thus as generalist as possible and, as a natural consequence of this fact, that work does not focus on any application nor domain and it does not discuss the accuracy aspect from the point of view of the specialist users, i.e., “what the application specialists expect from the data”. Their work has merit in that it presents useful and very needed pointers to guide further investigation on the subject. They also leverage important questions, such as the need for well-established benchmarks and the need to draw proper scopes as for where the validity of conclusions for any proposed method remains significant. Nevertheless, the ability of most of the existing methods to spot what the specialists consider as outliers in datasets of distinct natures, especially those collected in the many real-world applications that can benefit from the detection of outliers, is still unclear.

It is well-known in the literature that the existing theoretical models for unsupervised outlier detection make certain assumptions that may not reflect the true nature of specific outliers

in every single application (AGGARWAL, 2013; CAMPOS *et al.*, 2016). With that in mind, we performed an empirical study to evaluate 8 state-of-the-art outlier detection algorithms using 8 datasets from a variety of real-world applications with ground truth data manually created by specialist users; we were focused on verifying whether or not the algorithms are able to spot in an automatic and timely manner what the specialists selected as outliers. Interestingly, many of the algorithms had unfeasibly high runtime requirements or failed to return what the specialists expect from the data that was feed into them.

1.2 Problem definition, objectives and hypothesis

Outlier detection performed using computational methods is, in reality, a very specific form of classification. The general classification task does not usually deal with very imbalanced classes and, furthermore, to loose the constraints against imbalanced classes hardens the task considerably. For classes with a very small number of objects, every false positive and false negative with relation to such kind of class have magnified weight. Also, a class with a small number of objects may engulf within a class with very large number of objects altogether, according to the sensitivity of the classification method. Removal of outliers is also desirable for classification as outliers are unwanted, since they steer the classification method into deviating from the expected result.

Taking argument to reason, imbalanced classes and outlying objects really do not make sense in much of the context of machine learning. For a wide variety of scenarios, the diverse classes tend to have approximately the same number of objects. Most classification methods take advantage of this constraint and simplify the problem. In turn, they end up not tuned to deal with such very specific scenarios. Outlier detection is a classification task that deals precisely with these difficult issues. Although, in the context of machine learning, outlier detection is not as mainstream as clustering and classification, when detection of these objects is called into action, it is usually of utmost importance. Under the context of a factory, outliers mean severe failures in a production facility; under the context of neighboring natural events, outliers might be dangerous tornados or giant sea waves; and under the context of a financial transaction, outliers mean costly frauds and abuse of the system; and so on.

Translated to practice, the high sensitivity to the biases that are collateral to outlier detection methods inhibits generalization of the notion of outlierness. Hence, methods that perform well under certain groups of tasks, tend to perform poorly in other tasks, leading thus to the increasing interest in outlier ensembles (AGGARWAL, 2013), as they are more adaptable to the diverse kinds of tasks. Since there is a small room to simplify the assumptions of any given outlier detection method that is geared to be reasonably accurate, these methods cannot speed up so easily as the bulk of the classification methods. There is an additional caveat, that is the manner in which the literature approaches the experimental facet of scientific validation.

Experiments are performed under very controlled environments and synthetic datasets, without too much regard to common and important real-world tasks. The outlier detection field of study lacks standardized benchmarks, a fact that hardens comparisons between methods (CAMPOS *et al.*, 2016).

In this work, we restrict our focus to real-world datasets only, and application domains of utmost relevance. We make comparisons between diverse baseline methods. Some methods stand out, theoretically, per their low computational complexity, other methods for their resilience against deteriorating effects of high dimensionality, other methods for their high average accuracy, etc. We have, as a first objective, to evaluate how they perform with real-world datasets, both in terms of accuracy of results and runtime requirements. Naturally, experimental observations lend to conclusions. During the course of the experiments, we concluded that one method, ABOD, demonstrated high average accuracy; nevertheless, it was one of the slowest approaches studied. Recurring to literature, there are already existing approximation methods based on ABOD, with improved computational complexity, but we witnessed their collateral deterioration on the accuracy side. After careful evaluation, we noted that these approximation methods rely on data sampling, but the sampling process itself is performed in a very straightforward manner. Such a fact leads to the main hypothesis of this MSc work:

Hypothesis: the use of an appropriate data sampling strategy allows one to improve ABOD's theoretical computational complexity and its practical runtimes without losing accuracy.

1.3 Main contributions of this MSc work

This work's main contributions are:

1. *Empirical evaluation*: we evaluated 8 state-of-the-art outlier detection methods from the *application specialist's point of view*, and report results focused on quantifying how useful they can be for a variety of real world tasks of high impact, such as spotting breast cancer, heart and thyroid anomalies, detecting cyberattacks, and even identifying abnormalities in nature, like extreme natural events that might compromise preservation areas;
2. *MetricABOD*: we carefully designed a new ABOD-based algorithm that makes the analysis up to *thousands of times faster*, still being in average 26% *more accurate* than the most accurate related work. The main innovation is the usage of tree-based data structures known as Metric Access Methods (MAM) (TRAINA JR. *et al.*, 2002) to perform data sampling. Such structures were originally designed to index complex data, like images, audio, large graphs and fingerprints. This improvement was essential to enable outlier detection in many of the datasets that we studied, for which the existing methods lead to

unexpected results or unfeasible runtime requirements. Finally, we studied real collections of text data to show that our *MetricABOD* works also for adimensional, purely metric data.

1.4 Conclusions

The rest of this work is organized as follows: Related Work and Concepts, Outlier Detection Methods, Metric ABOD, Outlier Detection at Work and conclusions.

Content under this Master's dissertation is closely related to what was presented and discussed in our research paper ([JUNIOR; CORDEIRO, 2019](#)). In fact, our research paper has the same objectives and its structure closely resembles this Master's dissertation, although in a summarized and compact form. We did not resource to merely print out our paper in this document, but we took notes and wording straight from it, with slight modifications and added extra connecting sections wherever required.

BACKGROUND CONCEPTS

2.1 Data distributions

Objects that are stored in datasets or databases can be aggregated by a certain data distribution and summarized by the parameters of this distribution. For simplicity, we may assume that these objects are of numerical type, but it does not have to be the case. Distance or similarity measures between objects can also be summarized by a data distribution. Furthermore, if we have a distance function df or a similarity function sf and we are able to output as one number how distant or similar objects are one another, then we can treat each one of these numbers as objects, and these objects will obey some data distribution.

If the objects are themselves multidimensional, i.e., have several dimensions or attributes, the data distribution is also multidimensional. In the field of statistics, this is known as a multivariate data distribution. A distribution is described by its moments. These moments help describe the process that generated the data, and thus help labeling objects correlated to the distribution as normal or abnormal. For example, an object's (ab)normality can be gauged according to its variance with regard to the moment related to the central measure of tendency, i.e., first moment of the distribution. The variance itself is a specific moment of the distribution, i.e., second moment of the distribution.

2.2 Extreme values in the data distribution

Objects close to the central measure of the data distribution are considered normal and similar between themselves. Likewise, objects sufficiently away from the central measure or, in other words, away from the dominant pattern in the dataset, are considered abnormal or anomalous. By intuition, anomalous objects are rare. The three-sigma rule, when applied to objects inside of a dataset that obeys the normal distribution, says that objects whose actual value is more than three standard deviations distant from their expected value are unusual and, in

general, unexpected.

A more general mathematical formalism that helps demonstrate the intrinsic rarity of anomalous objects is the Chebyshev's inequality. The formula for the Chebyshev's inequality is $Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$, where $Pr(x)$ is the probability function for x , X is a random variable, μ is the expected value of X and σ^2 is the variance of X .

If we apply Chebyshev's inequality to the sample space spanned by a dataset with n unique objects whose data distribution behaves just like the normal distribution, the validity of assumptions from the inequality together with the three-sigma rule, says that in the vast majority of situations, no more than $\frac{1}{3^2}n = \frac{1}{9}n$ objects of a given dataset are unusual. Or, in other words, anomalous or outlying objects.

Drawing a parallel with the clustering task in data mining, clustering puts objects that are similar between themselves inside the same cluster. Clustering does not have to forcibly make every object pertain to any of the discovered clusters. Some objects are allowed to lie inside some cluster and the remaining objects may lie outside any clusters. Objects outside any clusters are clearly instances of extreme values, due to the fact that they are dissimilar to any object that has a significant degree of similarity to a significant number of other objects. Furthermore, objects located inside very specific and unusual examples of clusters might be also instances of extreme values, if the number of objects inside a such cluster is proportionally small, with relation to the cardinality of the majority of other clusters. We call these clusters with small number of objects "microclusters". As such, objects inside microclusters are not representatives of the dominant pattern in the dataset. Border objects with relation to some cluster, that is, objects that lie close to the borders of clusters, may also be cases of extreme values.

Let us now consider objects spread throughout space and some density measure like those employed in methods such as DBSCAN and OPTICS (ESTER *et al.*, 1996; ANKERST *et al.*, 1999). Sparsely populated regions in the space have smaller density, as objects in these regions are, in average, far apart their nearest neighbors. As such, sparsely populated regions in the space have a strong correspondence with extreme values in the data distribution.

2.3 Outlier detection in Data Mining

Knowledge Discovery in Databases, or KDD for short, has as its main goal to extract valuable knowledge from raw data. Fayyad, Piatetsky-shapiro and Smyth (1996) catalogue the KDD process in five steps: data selection, which refers to filtering the data that is most pertinent for the analysis; preprocessing, which deals with the removal of redundancy, noise, and the perks of missing data; transformation, which converts data to a presumably more "data mining friendly" format; data mining; and interpretation and evaluation of results. The data mining component is the most relevant one; it leverages machine learning, pattern recognition and statistics to find new patterns that are hidden into the data.

Outlier detection is one possible application of the data mining process. More specifically, outlier detection is a form of the classification task in machine learning and statistics, but to develop methods for the former is significantly more difficult than for the latter, thus posing hard challenges to researchers (AGGARWAL, 2013). Tracing comparisons to clustering, which is one of the most famous tasks in data mining, outlier detection has nearly the inverse objective: while clustering searches for groups of objects that are most similar between themselves inside a group – see Figure 1 for an example, outlier detection searches for the objects that are most dissimilar to the remaining objects. There are supervised outlier detection methods and unsupervised ones. Let us highlight specifics of both supervised and unsupervised techniques. Supervised outlier detection techniques incorporate domain-specific data and use machine learning to improve results in terms of the accuracy and relevance of the detected outlier objects. Because the availability of such data is often limited due to the very nature of outliers, which are rare, the development of supervised outlier detection methods might be unfeasible in a number of cases. Worse, work on such data is prone to further perks due to data contamination. One relevant example of data contamination is the wrong labeling of classes for specific data instances/objects. Combined to class imbalance, where the “normal” class is generally over-represented in comparison to the “abnormal” class, wrong labeling of classes makes the outlier detection task significantly more difficult and error-prone. On the other hand, unsupervised outlier detection techniques employ methods that discover the distribution of the data, or assume some distribution for the data, and try to locate objects that deviate markedly from this distribution. Such approach has a number of weaknesses. For example, it may return uninteresting noise instead of actual outlier objects. However it is, in a number of cases, significantly more practical and feasible than supervised outlier detection (AGGARWAL, 2013). In this MSc work, we focus on unsupervised outlier detection methods only. Therefore, we refer to it as simply “outlier detection” in the remaining parts of this dissertation.

2.3.1 Summary of related concepts and initial presentation on state-of-art outlier detection methods

There are many definitions for outliers in the literature. The distance-based one (KNORR; NG, 1998) is one of the most famous conceptual foundations that is commonly employed as a basis for time-efficient outlier detection algorithms: “An object P in a dataset T is a $DB(f, \xi)$ -outlier if at least one fraction f of the objects in T have a distance to P that is greater than ξ .” Here, term $DB(f, \xi)$ -outlier is a shorthand notation for a Distance-Based outlier with supporting parameters f and ξ . This definition is significantly independent of the data distribution at hand and it is also intuitive and simple, because it does not introduce sophisticated concepts beyond neighborhoods and some distance measure. Furthermore, it has been long proven that this definition is also practical, as it serves as foundation to distance-based outlier detection algorithms (BREUNIG *et al.*, 2000; BAY; SCHWABACHER, 2003).

Figure 1 – A simple example of clustering from the classical Iris dataset (Fischer, 1936).

Source: elaborated by the author

There is not a single, universally accepted definition for what an outlier is, neither an unanimous way to compute outlieriness scores. The issue of defining what an outlier is and how to find it matters to statisticians since long ago. It is common practice in the domain of Statistics to assume a parametric distribution for the data, following a fixed set of parameters. But, when taking into consideration most real-world scenarios, it is not safe to assume, a priori, any parametric distribution underneath a dataset. And there is more to it, because even if it is assumed as non-parametric, the existing unsupervised outlier detection algorithms end up also assuming a specific definition of outliers underneath the distribution to create a model, and then use the model to search for data objects that fit this definition. Obviously, different algorithms employ distinct models, thus they may obtain different results. Since there is no consensus between the different models, it is frequent that two given algorithms' results disagree markedly (MARQUES *et al.*, 2015).

Let us highlight that the application specialist user expects results of his/her true interest, and he/her does not bother excessively with the model and algorithm at hand. Model and algorithm are used as a mere medium to the desired goals. They are not the goal itself. At this point, we make a shift from the designer-based point of view that is routinely seen in the literature, to the user-based one. Even elaborate and theoretically sound techniques may not translate to acceptable results for the end user. Also, many real-world datasets from which spotting outliers is desirable have millions of objects and hundreds of attributes. Outlier detection in these data is a demanding task (FAN; LI, 2006). In fact, it is truly an unfeasible task in most cases. The main challenges are: (a) huge runtime requirements, even for approximate methods; (b) low availability of ground truth to assess a method's accuracy; (c) data of high dimensionality and its unwanted effects, and; (d) huge sensitivity to input parameter values.

To evaluate the distances from objects to their k -th nearest neighbors is one of simplest approaches to spot outliers. The *KNN-Outlier* method ranks the objects' respective distances to their k -th nearest neighbors (KNN distances), from longest to shortest, and gives higher outlier-ness scores to objects with higher KNN distances. KNN-Outlier is the basis for many other works, such as *Orca* (BAY; SCHWABACHER, 2003) and *RBRP* (GHOTING; PARTHASARATHY; OTEY, 2008).

LOF (BREUNIG *et al.*, 2000) is another well-known outlier detection method. LOF introduced in outlier detection the concepts of core distance and reachability distance, inspiring many posterior works, such as *LOCI* (PAPADIMITRIOU *et al.*, 2003), *aLOCI* (PAPADIMITRIOU *et al.*, 2003) and *ABOD* (KRIEGEL; SCHUBERT; ZIMEK, 2008). According to *LOF*, a *core object* is an object that has at least a certain number of neighbours in its ξ -neighborhood. An object x is *density-reachable* from object y if there is a chain of objects $p_1, p_2, \dots, p_q$, where $p_1 = y$ and $p_q = x$, such that p_{i+1} is in the ξ -neighborhood of p_i for $i \in \{1, 2, \dots, q-1\}$. The *core-distance* of an object x is the smallest distance ξ that makes it a core object. The *reachability-distance* of x is the smallest distance ξ that puts it in the ξ -neighbourhood of a core object y . LOF employs these concepts to calculate outlierness scores for all data objects according to their ξ -neighbourhoods' density of objects in the feature space.

From here on, we briefly describe methods that were developed having in mind the problems inherent to high dimensional data. Two basic approaches have been explored in the literature so far: subspace analysis and angle-based analysis. A subspace is a representation of the original space that excludes some of the dimensions/attributes present in the latter. Since a proper subspace has lower dimensionality than that of the original space, the data projection can alleviate the dilution and noise effects present in higher dimensional spaces and may also improve the contrast for outlier detection methods that use distances as a basis. Unfortunately, the number of distinct subspaces to analyze grows exponentially with regard to the number of attributes of a dataset. Therefore, exhaustive enumeration of subspace for search is not feasible.

Aggarwal and Yu (2001) faced this issue using an evolutionary algorithm. Their method works by iteratively improving candidate sets of subspaces until a final set of feasible subspace candidates is generated. OUTRES (MULLER; SEIDL, 2011) works by performing statistical tests upon subspace projections of dataset objects, starting from 1-dimensional projections and adding new dimensions one-by-one. Only subspace projections that pass their statistical test are considered. OUTRANK (MULLER *et al.*, 2008) enumerates as outliers the objects with low overlap, considering their presence inside clusters found in distinct subspaces. HiCS (KELLER; MULLER; BOHM, 2012) selects feasible subspaces by performing a statistical test upon each candidate subspace. HiCS itself does not measure the outlierness of objects; it merely retrieves and aggregates the results obtained by a coupled outlier detection method, such as LOF. Thus, HiCS is described by its authors as a "meta" outlier detection method. Finally, note that all of the aforementioned methods allow to set thresholds in an attempt to prune uninteresting subspace

projections.

ABOD (KRIEGEL; SCHUBERT; ZIMEK, 2008) uses a different approach. It was inspired by previous research works in the domain of text comparison that had verified that angles are more resilient than distances to the “curse of high dimensionality”. The authors of ABOD noted that such resilience also remains true when detecting outliers from high-dimensional data in a general context. For each object P , ABOD calculates the angles between every possible pair of objects (x, y) using P as the pivot, such that $x \neq y \neq P$. The variance of angles is then attributed to P as its outlieriness score. Small variances of angles indicate outliers; larger variances refer to inliers. The intuition here is that outliers tend to be in the borders of the feature space, so they have neighbors concentrated in one specific direction, while the inliers’ neighbours are spread all over the space. Figure 2 illustrates this idea considering an inlier (Figure 2a) and an outlier (Figure 2b). In both cases, colored semi-circles represent some of the angles associated to pairs of objects when taking the object of interest P as the pivot. Figure 3 complements this example by plotting the corresponding angle values and variances.

Figure 2 – ABOD’s angle-based analysis for an inlier (left) and an outlier (right).

Figure 3 – Angle values and their standard deviations (square roots of variances) as outlieriness scores for data in Figure 2.

ABOD’s performance may not be limited in practice by the data dimensionality, except that it does not address the issue of irrelevant attributes in high dimensional data (KRIEGEL *et al.*, 2009). However, due to scalability issues, its use is clearly impractical for data of high cardinality. It computes angles for every possible triple of objects, so ABOD is $O(n^3)$ where n is the number of objects. LB-ABOD (KRIEGEL; SCHUBERT; ZIMEK, 2008), FastABOD (KRIEGEL; SCHUBERT; ZIMEK, 2008) and FastVOA (PHAM; PAGH, 2012) improve upon ABOD, being focused on tackling its scalability issues. Unfortunately, they lead to either a minor speedup or to a considerable accuracy degradation.

2.4 Types of outlier detection methods

We now refer to the taxonomy of outlier detection methods as described by (HAN; KAMBER; PEI, 2011) and (AGGARWAL, 2013).

2.4.1 Proximity-based outlier detection

Proximity-based outlier detection methods rely on some notion of proximity between objects in the feature space. A such notion is to assume that the proximity of a given outlier object to its nearest neighbors significantly deviates from the proximity of the object to most of other objects in the dataset (HAN; KAMBER; PEI, 2011). Proximity-based outlier detection methods can be further divided in distance-based methods, density-based methods and clustering-based methods.

2.4.1.1 Distance-based outlier detection

One of the simplest ways to understand the distance-based outlier detection approach is through envisage of the distance calculation of each point to its respective k^{th} nearest neighbor. Top outliers, then, are points that hold the highest values recorded. In a more general sense, distance-based outlier detection methods quantify the dissimilarities between objects and their neighborhoods, and they estimate outlierness for a given object by weighting these dissimilarities in a manner that is suited to the data distribution beneath the dataset objects. One example is *DB-Outlier* (KNORR; NG, 1998), the seminal distance-based outlier detection method.

2.4.1.2 Density-based outlier detection

The main intuition behind density-based outlier detection is that the density around an outlier object is significantly lower than the average density around its neighbors. Thus, the relative density of an object of interest, with relation to its neighbors, can be used as an indicator of the outlierness degree for the object of interest. It is interesting to note that density measures and distance measures are converses of each other. One can part from calculations derived from a distance function and arrive to a density function and vice-versa.

2.4.1.3 Clustering-based outlier detection

In the clustering-based approach, considering the full feature space, outliers belong to small or sparse clusters, or do not belong to any cluster. The scientific literature enumerates many clustering-based outlier detection methods (CHRISTY; GANDHI; VAITHYASUBRAMANIAN, 2015; QAZI; KADRI, 2018), and there are also many methods that employ clustering as a pass of the complete method (MULLER *et al.*, 2008; DUAN *et al.*, 2009). This phenomenon happens as a byproduct of the multitude of clustering methods. Let us illustrate the basic idea in Figure 4. A single cluster is represented by a ellipse, so that inliers are objects inside the cluster. Points outside the ellipse are the outliers.

Figure 4 – Basic clustering example where the points inside the cluster are inliers and the points outside the cluster are outliers.

Source: elaborated by the author

2.4.2 Depth based outlier detection

This approach works as follows: given multidimensional data and a statistical depth function, a depth-based method gets a center-outward ordering from the deepest object. It detects outliers as the observations that appear extreme relative to the rest. The resulting n -dimensional convex hull forms the set of the top outliers for the dataset. An illustrative example is presented in Figure 5. This concept can be extended further, for calculating degrees of outlierness. The intuition here is that with a depth function, the actual statistical moments, like mean, variance and standard deviation need not be known. The center-outward ordering given in convex layers of the dataset objects is enough to attribute degrees of outlierness for the objects. For example, the central objects in the convex hull are attributed a depth of 0, being the least outlying objects. And the outermost objects that lie on the convex hull can be given a depth of k .

Figure 5 – A convex hull made of outliers. They are highlighted here by a red contour.

Source: elaborated by the author

The next sections discuss the main undesirable effects of dealing with data of high dimensionality; we also present the basics of Metric Access Methods.

2.5 Curse of high dimensionality

As the number of dimensions of a dataset increases, the variance of distances between any two data objects tends to zero (AGGARWAL, 2013). Figure 6 illustrates this fact by presenting dissimilarity matrices for two toy datasets with 10 objects each. The objects were randomly picked from a multivariate normal distribution. Figures 6a and 6b respectively refer to data of low and high dimensionality, i.e., 5 and 100 dimensions. Darker colors represent smaller distances; lighter colors refer to larger distances. As it can be seen, it is much harder to discriminate the distinct dissimilarities in the high-dimensional scenario than it is in the low-dimensional one. This simple example illustrates the evanescence of the variance of pairwise dissimilarities between objects, as the dimensionality increases.

In this fashion, according to the distance-based outlier definition, and also to most of the other existing definitions, all objects in a dataset of high dimensionality are potential outliers. To tackle this problem with subspace selection or dimensionality reduction is not obvious. According to (AGGARWAL, 2013), the main difficulties are: (a) circular references between neighborhoods and subspaces; (b) noise with its masking and dilution effects, and; (c) rarity-based subspace outliers. State-of-the-art solutions to the problem perform angle-based analysis, rather than the traditional distance-based one, as we discuss later in Chapter 3 of this monograph.

Figure 6 – Dissimilarity matrix for two toy datasets with 10 random objects each. a) 5 attributes; b) 100 attributes.

2.6 Metric Access Methods

Metric Access Methods (MAM) (TRAINA JR. *et al.*, 2002) allow efficient queries upon datasets embedded in a *metric space*. A metric space is formally defined by a data domain X and

a distance function $d : X \times X \rightarrow \mathbb{R}$, such that the rules as follows apply:

- i) Positivity: for all $x, y \in X$, $0 \leq d(x, y) < \infty$
- ii) Non-degenerated: for all $x, y \in X$, $d(x, y) = 0 \leftrightarrow x = y$
- iii) Symmetry: for all $x, y \in X$, $d(x, y) = d(y, x)$
- iv) Triangle inequality: for all $x, y, z \in X$, $d(x, y) \leq d(x, z) + d(z, y)$

A metric space may not allow sorting objects. Thus, MAM structures must use distances only; nothing else.

The distance function of a metric space is called a metric. Subspaces of metric spaces will also be metric, so queries of a data type upon which some metric is defined will form a finite metric space, subject to the same constraints. Metric spaces keep properties that allow us to derive upper and lower bounds without knowing the exact form of the distance in question. One interesting property in this context is triangularity. Triangularity may be interpreted as the concept of overlap for naturally defined regions such as every object within a given distance of some center object. If we loosen the triangularity restriction and violate it, two seemingly non-overlapping balls could still share objects. If we assume that our dissimilarity space is, in fact, a metric space, the essential next step is to construct bounds (HETLAND, 2009).

Sample objects, also known as pivots or centers, are selected as a basis for the indexing of datasets in metric spaces, and their associated distance bounds. Let we select some pivot object p from a given dataset, with the distance $d(p, o)$ from the pivot to every other indexed object already computed. This collection of precomputed distances then becomes an index. When performing a query, the distance from query to pivot, $d(q, p)$, is also computed. The distance $d(q, o)$ has as its lower bound:

$$\check{d}p(q, o) = |d(q, p) - d(p, o)|$$

An high-performance search may then be performed by filtering out objects within this lower bound. This approach to search potentially reduces the total number of distance computations (HETLAND, 2009).

State-of-the-art MAMs are tree-based. Every tree node stores a subset of objects. One of them is selected to be the node's representative, which is also known as the pivot. The representative is usually the object with the shortest average distance to all others. The node's region of representation is a hyper-sphere centered at the representative object with a radius that is greater than or equal to the distance from it to any other object in the same node.

Figure 7 illustrates a tree-based MAM upon objects in a two-dimensional space. We assume that each node stores up to 4 objects. In Figure 7a, four nodes with representatives A , B , C and D are defined out of all objects at the lowest level h . At level $h - 1$, only these representatives are considered; as it can be seen in Figure 7b, object D is selected to represent the previous level's representatives into a single root node. When using a MAM, the triangle inequality property and

Figure 7 – 2-dimensional objects at: a) level h of a tree-based MAM structure; b) level $h-1$ of the same structure.

the representatives serve as a basis to prune unnecessary distance calculations, thus leading to efficient data manipulation.

2.7 Semantic knowledge and semantic awareness

Semantic awareness infused to the outlier search method is a path to improve detection accuracy using heuristics, presumably without affecting detection speed dramatically. If the outlier search method does not return what the user is searching for, we might say there is a “semantic gap” between what the search method proposes and what the user wants.

“The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.” (SMEULDERS; WORRING; SANTINI, 2000)

Experiments reported in (CAMPOS *et al.*, 2016) suggest that any method for outlier detection tends to be most suited and useful for specific application domains. This observation can be extended on and developed further into the realm of outlier detection ensembles, which light up interest for their usefulness (AGGARWAL; SATHE, 2017). Ensembles are often computationally very demanding, adding to that the outlier detection task is itself already costly. With the asset of semantic knowledge about the application domain, one it then is able to build slimmer ensembles, that is, ensembles that are lighter in terms of computational demand.

2.8 Conclusions

We presented in this chapter the concept of data distributions, the concept of outliers, and a couple of other related concepts. We presented also a taxonomy for the diverse types of outlier detection methods, presented the curse of high dimensionality, Metric Access Methods and semantic aspects of data. These bases help lay ground to the development of efficient and accurate outlier detection techniques, which we discuss in greater detail further in this document.

OUTLIER DETECTION METHODS

3.1 State-of-the-art methods for outlier detection

This section describes state-of-the-art methods for outlier detection that form a sensible list of representatives of the main ideas and techniques behind high accuracy of results and significant runtime improvements over naive methods. It is important to note that all of these methods are designed to work on numeric data only.

3.1.1 *Orca*

Orca (BAY; SCHWABACHER, 2003) attempts to find the top outliers within a multidimensional data set. It employs a nearest-neighbor method. The intuition behind Orca is that if there are sufficiently many examples that are close to a candidate outlier in the feature space, then it is best to discard the candidate. Otherwise, the candidate is not discarded. Probable outliers lie in regions of the feature space in which the nearest neighbor density estimate is small.

Orca's pseudo-code is given in Algorithm 1. It receives as input: (i) k , the number of nearest neighbors; (ii) n , the number of outliers to return, and; (iii) D , a set of data objects in random order. It outputs a set of outliers O . In the code, $\text{maxdist}(x, Y)$ refers to the maximum distance between a given object x and any objects of a dataset Y ; similarly, $\text{Closest}(x, Y, k)$ returns the k closest objects of Y regarding x .

3.1.2 *LOF*

LOF (Local Outlier Factor) (BREUNIG *et al.*, 2000) introduces the concept of reachability. Reachability is defined as the maximum of two values for the distances: between a pair of objects A and B and B 's k -th nearest neighbor distance from B itself. This basilar concept helps build the concept of local density, that indicates the degree of outlierness of a given object. Low densities indicate objects with high degrees of outlierness.

Algorithm 1 – Orca outlier detection algorithm.

```

1: procedure ORCA( $k, n, D$ )
2:    $c \leftarrow 0$ 
3:    $O \leftarrow \emptyset$ 
4:   while  $B \leftarrow \text{get-next-block}(D)$  do
5:      $\text{Neighbors}(b) \leftarrow \emptyset$  for all  $b$  in  $B$ 
6:     for each  $d$  in  $D$  do
7:       for each  $b$  in  $B, b \neq d$  do
8:         if  $|\text{Neighbors}(b)| < k$  or  $\text{distance}(b, d) < \text{maxdist}(b, \text{Neighbors}(b))$  then
9:            $\text{Neighbors}(b) \leftarrow \text{Closest}(b, \text{Neighbors}(b) \cup d, k)$ 
10:          if  $\text{score}(b, \text{Neighbors}(b)) < c$  then  $\triangleright$   $\text{score}$  is an “outlierness” estimate
      for object  $b$ 
11:        remove  $b$  from  $B$ 
12:      end if
13:    end if
14:  end for
15: end for
16:    $O \leftarrow \text{Top}(B \cup O, n)$ , the top  $n$  outliers  $\triangleright$   $\text{Top}$  returns the  $n$  objects with the highest
    “outlierness” estimates
17:    $c \leftarrow \min(\text{score}(o))$  for all  $o$  in  $O$ 
18: end while

```

Figure 8 – An example of reachability distance as employed by LOF.

Source: (BREUNIG *et al.*, 2000)

Let $k\text{-distance}(A)$ be the distance of an object A to its k -th nearest neighbor. The set of the k nearest neighbors includes all objects at this distance, which in the case of a “tie” at the borderline are more than k objects. $N_k(A)$ is the set of k nearest neighbors. Then the reachability distance is:

$$\text{reachability-distance}_k(A, B) = \max\{k\text{-distance}(B), d(A, B)\}$$

Reachability distance is in fact a distance function of an object A from B , bounded at the minimum as the k -distance of B . Objects that belong to the k nearest neighbors of B are considered to be equidistant, leading to more stable results. Figure 8 illustrates the concept of

reachability distance as employed by LOF with a parameter $k=3$ and its choice of the maximal distance between objects. Since o 's distance to its 3^{rd} nearest neighbor is higher than its distance to p_1 , $reach-dist(p_1)$ assumes the value of o 's distance to its 3^{rd} nearest neighbor. $reach-dist(p_2)$, however, assumes the value of the Euclidean distance between o and p_2 .

The local reachability density of an object A is defined by

$$\mathbf{lrd}(A) := 1 / \left(\frac{\sum_{B \in N_k(A)} \mathbf{reachability-distance}_k(A, B)}{|N_k(A)|} \right)$$

as the inverse of the average reachability distance of the object A from its neighbors.

The local reachability densities are then stacked up against those of the neighbors using

$$\mathbf{LOF}_k(A) := \frac{\sum_{B \in N_k(A)} \frac{\mathbf{lrd}(B)}{\mathbf{lrd}(A)}}{|N_k(A)|} = \frac{\sum_{B \in N_k(A)} \mathbf{lrd}(B)}{|N_k(A)|} / \mathbf{lrd}(A)$$

as the average local reachability density of the neighbors divided by the object's own local reachability density. A value of *approximately 1* indicates that the object is not an outlier. A value *below 1* indicates a denser region, a potential inlier thus. Values *significantly larger than 1*, being "significantly larger" an object of scrutiny, indicate outliers.

3.1.3 HiCS

HiCS (High Contrast Between Subspaces) (KELLER; MULLER; BOHM, 2012) performs subspace selection and outlier detection in separate. It is based on the fact that subspaces that present high contrast have markedly strong correlation among their superspace's individual attributes, showing thus a peculiar probability density function that deviates from the expected function if such attributes were, in fact, independent.

Let $X_1, \dots, X_K$ be K continuous random variables. Then, for each $i = 1, \dots, K$, the probability density function of the random variable X_i , denoted by $f_{X_i}(x)$, is called a marginal probability density function. Suppose X and Y are continuous random variables with joint probability density function $f(x, y)$ and marginal probability density functions $f_X(x)$ and $f_Y(y)$, respectively. Then, the conditional probability density function of Y given $X = x$ is defined as: $h(y|x) = f(x, y) / f_X(x)$, provided that $f_X(x) > 0$.

HiCS makes comparisons of the marginal probability density function and the conditional probability density function for each candidate subspace. If the difference between these functions is "large enough", then the subspace is sent to a plug-in outlier detection method for inspection. By default, LOF is used as the plugged-in outlier detection method. It is used to compute density scores over all subspaces that present high contrast.

3.1.4 OUTRANK

OUTRANK (MULLER *et al.*, 2008) uses clustering in subspaces and employs an unified scoring system that assesses the outlierness degree of objects in distinct subspaces. The main

idea behind this method is that clusters are more stable than outliers to identify in the diverse subspaces. Since OUTRANK tests a diverse number of subspaces, from low-dimensional to high-dimensional, clusters may overlap.

At first, OUTRANK verifies how any specific object behaves in distinct subspaces. If such object lies mainly in clusters found in few or low-dimensional subspaces, it receives a high outlierness score. OUTRANK also considers density measurements for the object, in order to assign outlierness scores. It then standardizes outlierness scores for the data instances between distinct subspaces, weighted by the dimensionality of the subspace and the cluster size. Depending on the level of discrimination for the subspace clusters, this method may return a large set of potential outliers to be verified.

3.1.5 RBRP

RBRP (Recursive Binning and Re-Projection) (GHOTING; PARTHASARATHY; OTEY, 2008) splits the space recursively in bins or containers using divisive hierarchical clustering. Then, restricted at the level of the containers, it searches for the k nearest neighbors of the objects that are inside the containers to identify possible outliers, under the definition that outliers are the n objects whose distances to their respective k -nearest neighbors are the largest. Therefore, RBRP can be seen as an approximate version of Orca. Unfortunately, RBRP uses distance measures that, in theory, deteriorate rapidly as a result of increased dimensionality and, furthermore, it is not able to search for subspaces, which would make the effects of high dimensionality more amenable.

RBRP is divided in two phases; Algorithms 2 and 3 are the corresponding pseudo-codes. At phase 1, the method starts with k random centers and assigns each point to its closest center, creating k partitions. Then, it iteratively finds k centers for these k partitions. Once these iterations are over, the process continues for each of these partitions, recursively, if the size of the partition is greater than a user-defined threshold. This strategy forces points that are close to each other to be allocated in the same bin.

At phase 2, the method sequentially scans through each bin to find the approximate nearest neighbors of each data point. To speed up convergence to the approximate nearest neighbors, the method reorganizes the data points in each bin as per their order in the projection along the principal component – i.e., the axis in which variance is maximized – of the points inside the bin. The expected speed-up is because the approximate nearest neighbors of a data point are found during a sequential scan, again, when data points are ordered as per their projection along the principal component.

Algorithm 2 – RBRP outlier detection algorithm. Phase 1.

```

1: procedure RBRP-BIN( $n$ : number of top outliers to compute,  $k$ : number of partitions,  $D$ :
   the set of objects)
2:    $c \leftarrow \{c_1, c_2, \dots, c_k\}$  ▷ the set of  $k$  random centers
3:    $p \leftarrow \{p_1, p_2, \dots, p_k\}$  ▷ the set of  $k$  partitions of  $D$ 
4:   for an arbitrary number of iterations do
5:     Empty all  $k$  partitions in  $p$ 
6:     for each  $d$  in  $D$  do
7:        $j \leftarrow \text{Closest}(c, d)$ 
8:       Insert( $d, j$ )
9:     end for
10:     $c \leftarrow \{\}$ 
11:    RecomputeCenters( $c, p$ )
12:  end for
13:  for each  $p_i$  in  $p$  do
14:    if size of  $p_i > \text{Binsize}$  then
15:      RBRP-Bin( $\text{Binsize}, k, \text{it}, p_i$ )
16:    else
17:      Reorganize data points in  $p_i$ , ordered as per their projection along the principal
      component of  $p_i$ 
18:      Add  $p_i$  to  $B$ 
19:    end if
20:  end for

```

Algorithm 3 – RBRP outlier detection algorithm. Phase 2.

```

1: procedure RBRP-FIND-OUTLIERS( $n$ : number of top outliers to compute,  $k$ : number of
   partitions,  $B$ : the set of partitions)
2:    $c \leftarrow 0$  ▷  $c$  is the cutoff threshold
3:    $O \leftarrow \{\}$ 
4:   for each bin  $b$  in  $B$  do
5:     for each  $d$  in  $b$  do
6:       Neighbors( $d$ )  $\leftarrow \{\}$ 
7:       for each  $t$  in  $B$ , ordered by increasing distance to  $b$  do
8:         for each  $p$  in  $t$  such that  $p \neq d$  do
9:           if  $|\text{Neighbors}(d)| < k$  or  $\text{Distance}(d, p) < \text{Maxdist}(d, \text{Neighbors}(d)) \leftarrow$ 
           Closest( $d, \text{Neighbors}(d) \cup p, k$ ) then
10:            end if
11:            if  $|\text{Neighbors}(d)| \geq k$  and  $c > \text{Distance}(p, d)$  then
12:              break
13:            end if
14:          end for
15:        end for
16:      end for
17:       $O \leftarrow \text{TopOutliers}(O \cup b, n)$ 
18:       $c \leftarrow \text{MaxThreshold}(O)$ 
19:    end for

```

Figure 9 – Four counting neighborhoods and one sampling neighborhood.

Source: adapted from (QAZI; KADRI, 2018)

3.1.6 LOCI

LOCI (Local Correlation Integral) (PAPADIMITRIOU *et al.*, 2003) employs the multi-granularity deviation factor (MDEF). MDEF works with local density variations in the feature space, given a concept of local density, and is able to detect outlying clusters, that is, neighborhoods with outlier objects only.

The authors introduce two concepts. One is the counting neighborhood for some object, given a radius with a certain value. The other concept is the sampling neighborhood for the same object, whose radius is always larger than that of the counting neighborhood. Figure 9 shows four counting neighborhoods and a sampling neighborhood. LOCI calculates how much the densities of the counting neighborhoods deviate from the density of the sampling neighborhood. If the deviation is "large enough", the point lying at the center of the sampling neighborhood is set as an outlier. Multiple values are used for the radii of the sampling neighborhoods.

As it can be seen in Algorithm 4, LOCI marks a data point as an outlier if its counting neighborhoods have a density that is more than three standard deviations over the expected density for every of the considered sampling neighborhoods. This procedure naturally leads to a scoring system for the degree of outlierness. Because multiple values for the radius of sampling neighborhoods are employed, it is possible to generate multi-granularity plots of the outlier detection results, leading to further insight for the data analysis, as compared to the other methods. α represents the factor < 1 applied to the radius of the counting neighborhoods.

Algorithm 4 – LOCI outlier detection algorithm.

```

1: procedure LOCI( $\alpha$ ,  $P$ : the set of objects) ▷ Pre-processing
2:   for each  $p_i \in P$  do
3:     Perform a range-search
4:     for  $N_i \leftarrow \{p \in P | d(p_i, p) \leq r_{max}\}$  do
5:       end for
6:       From  $N_i$ , construct a sorted list  $D_i$  of the critical and  $\alpha$  – critical distances of  $p_i$ 
7:     end for ▷ Post-processing
8:     for each radii  $r \in D_i$  (ascending) do
9:       Update  $n(p_i, \alpha r)$  and  $\hat{n}(p_i, r, \alpha)$ 
10:      From  $n$  and  $\hat{n}$ , compute
11:       $MDEF(p_i, r, \alpha)$  and  $\sigma MDEF(p_i, r, \alpha)$ 
12:      if  $MDEF(p_i, r, \alpha) > 3\sigma MDEF(p_i, r, \alpha)$  then
13:        flag  $p_i$ 
14:      end if
15:    end for

```

3.1.7 aLOCI

aLOCI (Approximate Local Correlation Integral) (PAPADIMITRIOU *et al.*, 2003) improves upon the concepts of LOCI and employs counting boxes instead of dealing with the objects directly as to perform approximate distance calculations, achieving in this way linear time complexity with relation to the number of objects, while searching for outliers. An undesired side-effect of using counting boxes – instead of dealing with the objects directly – is the introduction of counting errors into the calculations. Exactly as with LOCI, α represents the factor < 1 applied to the radius of the counting neighborhoods. Algorithm 5 provides the full pseudo-code.

3.1.8 ABOD

ABOD (KRIEGEL; HUBERT; ZIMEK, 2008) attempts to detect outliers making use of variance of angles calculation for every object of interest. Such angles are constructed by the difference vector of the vectors $\overrightarrow{OA}$ and $\overrightarrow{OB}$, where O is the object of interest itself, and A and B are distinct neighbors in the d -dimensional data, since FastVOA does not weight the distances from the object of interest to the wings as ABOD does.

3.1.9 OUTRES

OUTRES (MULLER; SCHIFFER; SEIDL, 2010) attempts to tackle the issues of the high-dimensionality by employing heuristics that select only relevant subspaces out of all possible subspaces. The subspaces are selected via statistical tests. The selection itself is performed in an approximate manner, thus avoiding the high computational complexity of processing every possible subspace. As it can be seen in Algorithm 6, OUTRES begins its search with lower dimensional subspaces to find relevant projections where the data objects are not uniformly

Algorithm 5 – Approximate aLOCI outlier detection algorithm.

```

1: procedure ALOCI( $\alpha$ ,  $g$ : the number of grids,  $P$ : the set of objects)           ▷ Initialization
2:   Select set of shifts  $S = \{s_0, s_1, \dots, s_g\}$ , where  $s_0 = 0$            ▷ The shifts are applied to the
   objects, possibly “wrapping” some or all of them around the counting boxes in the grid.
3:    $l_\alpha \leftarrow -lg(\alpha)$ 
4:   for each  $s_i \in S$  do
5:     Initialize quadtree  $Q(s_i)$                                            ▷ Pre-processing stage
6:     for each  $p_i \in P$  do
7:       for each  $s_i \in S$  do
8:         Insert  $p_i$  in  $Q(s_i)$ 
9:       end for
10:    end for                                                                ▷ Post-processing stage
11:    for each  $p_i \in P$  do
12:      for each level  $l$  do
13:        Select cell  $C_i$  in  $Q(s_a)$  with side  $d_i = R_P/2^l$  and center closest to  $p_i$ 
14:        Select cell  $C_i$  in  $Q(s_b)$  with side  $d_i = R_P/2^{l=l_\alpha}$  and center closest to center of
         $C_i$ 
15:        Estimate  $MDEF(p_i, \frac{d_i}{2}, \alpha)$  and  $\sigma MDEF(p_i, \frac{d_i}{2}, \alpha)$ 
16:        if  $MDEF(p_i, \frac{d_i}{2}, \alpha) > 3\sigma MDEF(p_i, \frac{d_i}{2}, \alpha)$  then
17:          flag  $p_i$ 
18:        end if
19:      end for
20:    end for

```

Figure 10 – Intuition behind the ABOD algorithm. It is expected that an object inside a cluster will present the highest variance of angles. These angles are generated by the object itself and an arbitrary pair of other objects.

Source: (KRIEGEL; SCHUBERT; ZIMEK, 2008)

distributed. After this phase, the method pinpoints higher-dimensional subspaces and repeats the process for each newly discovered subspace.

Additionally, the authors introduce a concept of adaptive neighborhood. The extent of

Figure 11 – Graph showing the spectra of angles for different types of points. Extremely high values of this spectra are assigned to inner objects and the converse is true for the outliers.

Source: (KRIEGEL; SCHUBERT; ZIMEK, 2008)

the neighborhood for a given outlier that the algorithm takes into account is defined by an epsilon parameter. Information about the neighborhood is used to assess how much a given data instance is an outlier, in conjunction with the calculation of local object deviation for each object, with relation to its neighborhood. These calculations lead to the objects' "outlierness" scores. OUTRES is able to standardize "outlierness" scores between subspaces of different dimensionalities, being regarded as a robust outlier detection method (ZIMEK; SCHUBERT; KRIEGEL, 2012). Unfortunately, it can be very slow in practice. For example, in preliminary experiments performed in this MSc work, OUTRES took entire days to process datasets that other algorithms analyze in minutes.

Algorithm 6 – OUTRES outlier detection algorithm.

```

1: procedure OUTRES(Data Point:  $\bar{X}$ , Subspace:  $S$ )
2:   for each attribute  $i$  not in  $S$  do
3:     if  $S_i \leftarrow S \cup \{i\}$  passes non-uniformity test then
4:       Compute  $OS(S_i, \bar{X})$ 
5:        $O(X) \leftarrow OS(S_i, \bar{X}) \cdot O(\bar{X})$ 
6:        $OUTRES(\bar{X}, S_i)$ 
7:     end if
8:   end for

```

3.2 Summary of the existing approaches

Table 1 presents a succinct comparison of the methods described in the previous sections with regard to their ability to harvest subspaces after outliers and also to their robustness when dealing with data of high dimensionality. It is expected that every method that detects

outliers in specific subspaces of the full-dimensional space can also withstand the curse of high dimensionality. It is worth mentioning that, in practice, methods that theoretically should not perform well with high-dimensionality data, such as LOF and Orca, can still present state-of-the-art results in some specific cases (CAMPOS *et al.*, 2016).

Table 1 – Outlier detection methods compared, as for their suitability to high dimensional data.

Method	Can detect outliers in subspaces	Can withstand the curse of high dimensionality
Orca	N	N
LOF	N	N
HiCS	Y	Y
OUTRANK	Y	Y
RBRP	N	N
LOCI	N	N
aLOCI	N	N
ABOD	N	Y
OUTRES	Y	Y

3.3 Evaluation of outlier detection methods

Marques *et al.* (2015) argue that evaluation and validation “...problems have been investigated for decades in the area of unsupervised data clustering, but are rarely mentioned and are virtually untouched in the area of outlier detection.” Our literature review directs us into accepting the validness of such affirmation. The very characteristic nature of outliers challenge the evaluation of outlier detection methods, e.g., outliers do not appear to follow any general pattern, a fact that hardens comparisons between these methods.

Campos *et al.* (2016) also comment on the inherent challenges of the evaluation of unsupervised outlier detection methods, pointing out that its main goal should be restricted upon analysis of the strengths and weaknesses that each method presents, with all methods being tested within a wide spectrum of datasets. They also write on the easiness of favoring any of the available methods over the others, by picking appropriate parameter values. This issue on favoring methods should be countered by testing all methods over multiple parameters that cover most or all of the possible usage scenarios. For a more practical point of view on this issue, it is said in Aggarwal (2013) that “At the end of the day, comparing two algorithms is somewhat of an art-form in the unsupervised setting, and one must rely at least a little on the experience and good judgement of the analyst in making proper experimental design choices.” We interpret this sentence as an advice for that we should, during research, pursue an ever improved knowledge of the inner workings of outlier detection, and update our evaluation methods accordingly. We must also be aware of and be sensible to time and resource constraints.

With that being said, we succinctly explain in the following two types of evaluation methods: internal evaluation methods and external evaluation methods (ZAKI; JR, 2014).

Internal evaluation methods: in order to perform evaluation, *mathematical models* that capture characteristics inherent to the task of detecting outliers are used. This capture includes specific characteristics of the dataset itself at hand. One of such evaluation methods is IREOS, by [Marques *et al.* \(2015\)](#).

External evaluation methods: those methods rely on *previous knowledge* about the data objects and attempt to take advantage of such availability for reporting deviations of the results obtained with relation to external immutable data, as it happens when ground truth data is available. Ground truth is basically a key part of a dataset in which the expected outcome for the reported results for many or all of the data objects is already known in advance.

3.4 Conclusions

This chapter discussed state-of-the-art outlier detection methods and their peculiarities. We summarized the methods with two key features as parameters: the ability to detect outliers in subspaces and the ability to withstand the curse of high dimensionality. We also presented ideas behind a robust evaluation of outlier detection methods.

METRIC ABOD

ABOD is one of the outlier detection methods able to cope with the effects of high dimensionality. Unfortunately, ABOD's cubic time complexity on the number of objects hampers its usefulness in scenarios where the cardinality of data is higher than a few hundred elements. Data dimensionality up to a few dozen attributes is not the main factor that impacts ABOD's runtime, and even less with the asset of precomputation of inner products to speed up the algorithm. A natural option that arises is to reduce the number of calculations *per object* that ABOD performs, without degrading accuracy. It may be a viable alternative that leads potentially to better scalability and also to lower constant terms in the complexity equation. In this chapter, we present a train of thought that begins with simple and well-known strategies and it evolves as we progress into the proposal of a novel method.

As it was discussed before, ABOD uses every data object P as the pivot for angle calculations on the order of $O(n^2)$, where n is the total number of objects. Specifically, the outlierness score of an object P is the variance of angles between every possible pair of objects (x, y) with P as the pivot. Obviously, x , y and P must be distinct objects. (x, y) is, for the sake of simplicity, the same pair as (y, x) , since they lead to exactly the same term in the variance of angles formulation. Figures 10 and 11 from the previous Chapter 3 illustrate the process.

To calculate angles only for pairs (x, y) taken from a *sample dataset* is one clever way to achieve better scalability, still obtaining a robust outlierness score for *every* data object. The initial $O(n^3)$ complexity is lowered into $O(n \cdot m^2)$ with $n \gg m$, where n and m are the full dataset cardinality and the sample size respectively. Let us keep in mind that the use of samples leads to approximate outlierness scores that may negatively impact the accuracy of results. So, the main question here is:

- How to select appropriate objects to be in the sample?

Random sampling is a computationally efficient and unburdensome way to answer this

question. In fact, it has already been considered in one of the existing ABOD variations, the FastABOD algorithm. However, in high-dimensional spaces, random sampling might increase the sampling error of its associated multivariate data distribution, thus leading to skewed variances of angles. Due to this fact, FastABOD's authors suggest the use of one distinct sample for each data object: its own k nearest neighbours. The rationale behind this approach is that the nearest neighbors have more weight in the calculation of variances of angles. FastABOD computes approximate outlierness scores for each object P by using it as the pivot for angle calculations among its own k -nearest neighbours. Unfortunately, k -nn sampling considerably degrades accuracy, as it could be observed in the results obtained during this MSc work; see the upcoming Chapter 5. It is also not computationally as efficient as random sampling.

This chapter presents a novel sampling strategy to improve ABOD by taking advantage of a specific class of Metric Access Methods (MAMs): hierarchical structures based on hyper-spheres. As it was described in the previous Chapter 2, tree-based MAMs have nodes that follow an hierarchical organization. Each node stores a set of objects; one of them is selected to be the node's representative, which is usually the object with the shortest average distance to the others. The leaf nodes store all dataset objects; their representatives are stored redundantly in nodes of the immediate higher tree level, from which representatives are also selected. This organization continues recursively up to the root of the tree; see Figure 7 from the previous Chapter 2 for an illustrative example.

The nodes of a tree-based MAM are usually built in a bottom-up fashion, like in a B-tree, so to minimize the number of levels, the number of nodes per level and the overlap among the nodes' hyper-spheres of representation in the feature space. This fact is *essential* to our proposal: it means that the representatives of nodes in any tree level preserve key characteristics of the full dataset, with more or less details according to the level. The highest degree of detail is in the leaf nodes' representatives. Since the representative is usually the object with the shortest average distance to the others, it is akin to the first moment or center of mass of the node's hyper-sphere. So, it is similar to the median for the node's objects. In fact, as it happens with random sampling, there is a strong correspondence between the representatives' statistical moments and those of the full dataset. Note, however, that low density regions in the feature space end up underrepresented with random sampling, i.e. collected samples do not give statistical hints as for the presence of such regions, while the choice of MAM representatives by our method is able to improve representativeness of every object located within the volume of the hyper-spheres, independently of the objects' local densities. The representatives also compare favorably with FastABOD's k -nn sampling. Although the k nearest neighbours of an object account for most of its variance of angles, to use them as references does not necessarily discriminate outliers correctly. For example, FastABOD tends to suffer from accuracy degradation wherever happens that an outlier object whose k nearest neighbours are spread in many directions, while the rest of the dataset, i.e., the vast majority of it, concentrates in a single direction. Note that this case would not be a problem with MAM representatives; most of them would also concentrate in one direction, just

like in the full data. A similar behavior is expected with more intricate data distributions.

With that in mind, we *propose* to use the leaf nodes' representatives from hierarchical hyper-sphere-based MAMs as samples to speed-up ABOD. Algorithm 7 gives the full pseudo-code; let us call it the *MetricABOD* algorithm. We argue that the leaf node's representatives lead to a reasonable estimate for the exact variances of angles, and experimental results that support our claim are reported in this work. This estimate for the variance of angles performs appropriately, as using the representatives even improved accuracy for the datasets that we studied. Figure 12 illustrates how our *MetricABOD* works. To easy comprehension, the same toy data of our running example from Figure 7 is reused here. There are $n = 15$ objects with two attributes each. At first, a tree-based MAM is created. Four circles in the 2-dimensional space illustrate its leaf nodes. The representative objects are A, B, C and D . In this setting, the approximate outlierness score for an object of interest P is computed by using it as the pivot for angle calculations among pairs of objects (x, y) , such that $x, y \in \{A, B, C, D\}$ and $x \neq y \neq P$. Colored semi-circles illustrate some of the angles to be computed. Specifically, only $C_2^4 = \frac{4!}{2!(4-2)!} = 6$ angle calculations are performed to process P . Note that the original ABOD algorithm would require $C_2^{n-1} = \frac{14!}{2!(14-2)!} = 91$ calculations in the same scenario.

Algorithm 7 – The *MetricABOD* algorithm

```

1: procedure MABOD(Data – input dataset)
2:   Build a tree-based MAM for dataset Data. Let it be Tree;
3:   Sample =  $\emptyset$ ;
4:   for each leaf node n of Tree do
5:     Let r be the representative object of n;
6:     Sample = Sample  $\cup$   $\{\mathbf{r}\}$ ;
7:   end for
8:   Result =  $\emptyset$ ;
9:   for each object P in set Data do
10:    Angles =  $\emptyset$ ;
11:    for each object x  $\neq$  P in set Sample do
12:      for each object y  $\neq$  x  $\neq$  P in set Sample do
13:        Angle(x,y) = the angle  $\theta_{\vec{P_x P_y}}$  between the difference vectors  $\vec{P_x}$  and  $\vec{P_y}$ ;
14:        Angles = Angles  $\cup$   $\{\text{Angle}_{(x,y)}\}$ ;
15:      end for
16:    end for
17:    Score = the variance of angles in Angles;
18:    Result = Result  $\cup$   $\{(\mathbf{P}, \text{Score})\}$ ;
19:  end for
20:  Sort Result in ascending order with regard to the Score values.
21: end procedure

```

In a general scenario, our *MetricABOD* avoids computing $C_2^{n-1} - C_2^{m-1}$ angles *per object*, with $n \gg m$, where n and m are respectively the full dataset cardinality and the number of leaf nodes, i.e., the sample cardinality. So, it turns ABOD's $O(n^3)$ overall complexity into

$O(n.m^2)$. Note that the time complexities to build and traverse a tree-based MAM are respectively $O(n \log n)$ and $O(n)$ for the most typical scenarios, so the additional tree-related cost does not modify the aforementioned overall complexity of *MetricABOD*. It is even possible to load balance between MAM construction and actual outlier detection, as a tree with a very small number of representatives, something desired for ABOD, takes more time to finish building when its height is also small and its node capacity is close to maximum. It is worth noting that the sample size m is linearly correlated with the degree of the tree, i.e., the maximum capacity of a node, so m can be easily tuned for any case of use.

Figure 12 – Our proposed angle-based analysis for an object P . Circles in the 2-dimensional space illustrate the leaf nodes of any tree-based MAM. The new *MetricABOD* algorithm uses only the representatives A, B, C and D to compute the outlierness score of P .

4.1 Conclusions

This chapter presented *MetricABOD*, a new ABOD-based algorithm for unsupervised outlier detection. Its main innovation is the usage of tree-based data structures known as Metric Access Methods to perform data sampling. In the next chapter, we show that this improvement is desirable to enable outlier detection in many real-world datasets, for which the existing methods lead to unexpected results or unfeasible runtime requirements. Finally, we also show that our *MetricABOD* allows spotting outliers in adimensional, purely metric data.

OUTLIER DETECTION AT WORK

After carefully revising the literature during this MSc work, we noted that many of the state-of-the-art methods in outlier detection validate their results only on synthetic data of low cardinality. Some methods are also tested for one or two real applications using data with up to a few thousand objects, but not for a broad range of applications in which diversity takes place, nor using datasets that are as large as the ones required for commercial use. Therefore, it is still unclear how efficient and effective these methods are to be used in the real world.

In this chapter, we shrink this limitation considering a variety of real-world tasks of high impact, like spotting cyberattacks, clinical pathologies and abnormalities in nature. We studied the behaviour of 8 state-of-the-art algorithms face to 8 datasets of distinct natures, that is, data from diverse real applications with varying cardinality and dimensionality. To make it possible, we assume that the ground truth created by specialist users is correct, and verify whether or not the algorithms are able to obtain similar results in an automatic and timely manner. The main motivation here is that outlier detection must be useful in practice, not only in academia. Specifically, we focus on answering the questions as follows:

- Q1** How effective and efficient are the state-of-the-art algorithms to be used in diverse real world applications?
- Q2** What are the best applications for each algorithm?
- Q3** Is there any algorithm with remarkably high accuracy in a general context?

5.1 Setup

The algorithms studied are: MetricABOD, KNN-Outlier, LOF, ABOD, FastABOD, LB-ABOD, FastVOA, aLOCI and HiCS. We used the Elki (<https://elki-project.github.io>) implementation for all algorithms, except for MetricABOD and FastVOA that were evaluated

with implementations of our own. For KNN-Outlier, LOF, aLOCI and HiCS, we set distance calculations to the L^2 norm, also known as the Euclidean metric. For MetricABOD, ABOD, FastABOD, LB-ABOD and FastVOA, we set angle calculations on a polynomial kernel feature space, with polynomials of degree 2. The Slim-tree (TRAINA JR. *et al.*, 2002) MAM was used as the supporting tree structure for MetricABOD. The entire Slim-tree structure was allocated on working memory. The algorithms were tested in a Xeon E5-2640v3 machine with 16 GB of RAM, running Debian 9 64-bit. The number of samples k was set to 100 in both FastABOD and LB-ABOD. For a fair comparison with the related works, our MetricABOD was evaluated with a tree degree tuned to generate nearly $m = 100$ leaf nodes' representatives in every experiment; every leaf node was used for each one of the experiments; note that it is the same value used for FastABOD's and LB-ABOD's similar parameter k . For LB-ABOD, we also set parameter l as two times the number of outliers known to exist in the ground truth. The same selection was used for parameter k in KNN-Outlier, LOF and HiCS. Finally, FastVOA was tuned with $t=100$, $s1=1,600$ and $s2=10$, following its authors recommendation.

A commonly used metric in the evaluation of results in outlier detection is the Area Under the Curve (AUC), which is derived from the curve of true positive rate versus false positive rate – the Receiver Operating Characteristic. We observe that such metric does not account for the number of objects retrieved per method, which can be arbitrarily large and impractical. In real-world practice, a human specialist would clamp the number of top results per outlier detection method to a value proportional to the expected number of outliers for each dataset. Therefore, both to simplify the evaluation process and to best represent what real users would find, our evaluation was performed initially by asking each algorithm for the o objects with the highest outlierness scores, where o is the number of outliers present in the ground truth. Further accuracy evaluation was conducted by using the methods to return a slightly larger number of objects, up to $2o$. We envisage scenarios where the application specialist allows for a few more objects, even if it is guaranteed that the number of false positives will be high. One scenario is risk evaluation, where the cost of not detecting an outlier is normally higher than the cost of analyzing a slightly higher number of objects.

The evaluation was performed by asking each algorithm for the o objects with the highest outlierness scores, where o is the number of outliers present in the ground truth. Considering correctly detected outliers as true positives (tp), the accuracy of each algorithm was calculated as $\frac{tp}{o}$. We flag each outlier in the ground truth as correctly detected if it appears at least one time between the o objects with top outlierness scores. This procedure was done with each of the rankings retrieved per method. Except when we explicitly state otherwise, every experiment was ran 5 times and we report average accuracy, the corresponding standard deviation and average runtime.

We also observe that the Java runtime optimizer aggressively optimizes for the ABOD implementation in Elki. Thus, we use the following flags for the g++ compiler: `-O3 -march=native`

-mtune=native -ffast-math. As result, implementations in both languages benefit from a high level of optimization when turned into machine code.

5.2 Datasets

Table 2 summarizes the datasets studied. They are available at the ODDS repository (<http://odds.cs.stonybrook.edu/>). All datasets come from real-world applications and include ground truth created by specialist users. For brevity, they are described in the following with focus on the practical benefits of spotting outliers in each case. Detailed descriptions are found in the data source website.

- **BreastW and Mammography:** two datasets with features from medical exams of breast cancer suspicious cases. They were collected in separate from two distinct locations. Spotting outliers here means detecting severe health problems, that is, distinguishing between the benign and the malignant cases.
- **Cardio:** consists of measurements of Fetal Heart Rate (FHR) and Uterine Contraction (UC) features on cardiotocograms classified by expert obstetricians. This dataset has 1831 instances and has dimensionality 21. It is an adaptation of the "Cardiotocography dataset" from UCI repository. The classes are normal, suspect, and pathologic. The pathologic class is downsampled to 176 instances and is considered as outliers. The suspect class is discarded. To detect outliers in this scenario is important to prevent and treat heart malfunction.
- **Annthyroid:** consists of thyroid exams. It is an adaptation of the "Thyroid Disease Data Set" classification dataset from UCI machine learning repository. The original dataset has 3772 training instances and 3428 testing instances and has 15 categorical and 6 real attributes. The problem is to determine whether a patient referred to the clinic is hypothyroid. Therefore three classes are built: normal (not hypothyroid), hyperfunction and subnormal functioning. For outlier detection, both training and testing instances are used, totalizing 7200 instances, but with only 6 real attributes. Only the hyperfunction and subnormal classes are treated as one outlier class with 534 instances. Here, to detect outliers refers to the identification of the hypothyroid disease in human patients.
- **Satimage-2:** this dataset is an adaptation of the "Statlog (Landsat Satellite) Data Set". The original dataset consists of multi-spectral values of pixels in 3x3 neighborhoods in a satellite image, with the classification associated with the central pixel in each neighborhood. The training and test instances are combined. Class 2 is downsampled to 71 instances which are considered as outliers, while all the other classes are combined to form an inlier class. The modified dataset is referred to as Satimage-2. Satimage-2 has

5803 instances and dimensionality 36. The utility of finding outliers here relates to the study of abnormal and unexpected patterns of topography, land use etc.

- Shuttle: consists of an adaptation of the "Statlog (Shuttle) Data Set". The original Statlog (Shuttle) dataset from UCI machine learning repository is a multi-class classification dataset. It combines the training and test instances. The smallest five classes are combined to form the outliers class, while the dominating class forms the inlier class. Data for class 4 is discarded. Shuttle has 49097 instances and dimensionality 9. Finding outliers in this case is equivalent to spotting potentially dangerous abnormalities during a space shuttle flight.
- Http (KDDCUP99): consists of an adaptation of the "KDD Cup 1999 dataset" from UCI machine learning repository. The original dataset contains 41 attributes which reduced to only 4 attributes (service, duration, src_bytes, dst_bytes), where only "service" is categorical. The categories of the "service" attribute are http, smtp, ftp, ftp_data, others. Only "http" service data is used. The original data set has 3925651 attacks (80.1%) out of 4898431 records. A smaller set is forged by having only 3377 attacks (0.35%) of 976157 records. From this forged dataset, 567479 "http" service data instances are used to construct the http (KDDCUP99) dataset. To spot outliers here means distinguishing between cyber attacks and regular transactions.
- Musk: the original "Musk (Version 2) Data Set" dataset from UCI machine learning repository contains several musk and non-musk classes. The non-musk classes j146, j147, and 252 are combined to form the inliers, while the musk classes 213 and 211 are added as outliers without downsampling. Other classes are discarded. As such, it totals 3062 instances with dimensionality 166. It is a medium-to-high-dimensionality dataset that stresses the ability of the outlier detection method to cope with the noise and dillution effects in higher-dimensional spaces. This dataset has features extracted from molecules classified as musk or non-musk by human experts in chemistry. Spotting outliers in such kind of data helps discovering new material that may be valuable to pharmaceutical corporations.

5.3 Results

Tables 3 and 4 respectively report accuracy and runtime results for the algorithms and datasets that we studied. As it was expected, some of the algorithms had unfeasibly high runtime requirements for our largest datasets Shuttle and Http (KDDCUP99), with $\sim 50k$ and $\sim 500k$ objects respectively. Therefore, we represent with "N/A" the cases that exceeded either a timeout limit of 24 *hours* or a memory limit of 16 *GiB*. With regard to accuracy, note that many of the state-of-the-art techniques failed to return what the application specialists expect from their own

Table 2 – Summary of datasets.

Dataset	Number of Axes	Number of Objects	Number of Outliers
BreastW	9	683	239
Cardio	21	1831	176
Satimage-2	36	5803	71
Shuttle	9	49097	3511
Anthyroid	6	7200	534
Mammography	6	11183	260
Http (KDDCUP99)	3	567479	2211
Smtip (KDDCUP99)	3	95156	30
ForestCover	10	286048	2747
Musk	166	3062	97

data. LOF, HiCS, FastABOD and FastVOA had the lowest average accuracies, overall. On the other hand, *MetricABOD*, aLOCI, ABOD, LB-ABOD and KNN-Outlier performed considerably better, being the most accurate methods in average. Let us highlight that the original ABOD method obtained the third best accuracy, but its use is unfortunately prohibitive due to scalability issues. In fact, ABOD is the worst methods in terms of runtime. The variation LB-ABOD is slightly faster without losing much in accuracy, although its runtime requirements are still far from being acceptable for most real-world uses. Finally, FastABOD is tens of times faster than ABOD and LB-ABOD in average, but its accuracy degrades considerably, that is, it spots correct outliers in only 34% of the times.

The last columns of Tables 3 and 4 report the results obtained with our *MetricABOD*. Note that the tree-related costs are *included* in the runtime results. As it was expected, *MetricABOD* coped with low-to-high dimensional data being considerably faster than the original ABOD method; surprisingly, it even improved ABOD’s accuracy for the datasets that we studied. In fact, our proposed algorithm made the analysis up to *thousands of times faster* when compared with the 8 related works evaluated, still being in average 26% *more accurate* than the most accurate related work, i.e., KNN-outlier. These results indicate that our *MetricABOD* is the best option, among those, which were considered, for use with a broad range of applications in which diversity takes place, especially when considering datasets that are as large as those required for commercial use.

To better understand why *MetricABOD* outperformed ABOD in accuracy, we investigated the case with the highest discrepancy; that is, the analysis of the *Musk* dataset. Ground truth shows that there are 97 outliers in this data. ABOD and *MetricABOD* respectively identified 7% and 77% of them. There are 29 common objects among the 97 most outlying objects indicated by each technique, and only 7 of them are true outliers. Interestingly, the common objects are among ABOD’s most outlying objects, while they figure among the least outlying ones for *MetricABOD*,

thus further corroborating the superiority of our proposal’s results for this dataset. Let us highlight that *Musk* has the largest dimensionality among all datasets studied, with 166 attributes. Due to this fact, we conjecture that the dimensionality was high enough to affect ABOD’s results, even with its angle-based analysis that is less susceptible than distances to the curse of the high dimensionality. On the other hand, there are evidences in the literature (ZIMEK; SCHUBERT; KRIEGEL, 2012) indicating that to spot appropriate representatives for a dataset is somehow similar to reducing its dimensionality. Thus, we believe that the tree nodes’ representatives obtained from *Musk* reduced the effects of the high dimensionality, thus enabling the angle-based analysis to obtain better results.

Table 3 – Accuracy results.

Dataset	ABOD	LABOD	FABOD	LOF	KNN	aLOCI	HiCS	FVOA	MetricABOD
BreastW	0.95	0.95	0.94	0.32	0.94	0.62	0.22	0.47	0.95
Cardio	0.52	0.52	0.36	0.23	0.51	0.04	0.32	0.19	0.61
Sat-2	0.88	0.88	0.29	0.07	0.91	0.00	0.11	0.11	0.90
Shuttle	<i>N/A</i>	<i>N/A</i>	0.31	0.19	0.34	0.88	<i>N/A</i>	0.44	0.58
Annth	0.25	0.24	0.26	0.32	0.24	0.10	0.15	0.15	0.24
Mammo	0.28	0.11	0.26	0.23	0.27	0.08	0.21	0.17	0.26
Http	<i>N/A</i>	<i>N/A</i>	<i>N/A</i>	0.04	0.03	0.98	0.05	0.00	0.98
Musk	0.07	0.07	0.02	0.00	1.00	0.98	0.95	0.05	0.77
Average	0.49	0.46	0.34	0.17	0.53	0.46	0.28	0.19	0.66
Stddev	0.32	0.34	0.26	0.11	0.34	0.42	0.08	0.16	0.27

Table 4 – Runtime results in minutes.

Dataset	ABOD	LABOD	FABOD	LOF	KNN	aLOCI	HiCS	FVOA	MetricABOD
BreastW	0.04	0.04	0.00	0.00	0.00	0.00	1.16	0.16	0.00
Cardio	1.23	1.25	0.01	0.00	0.00	0.00	10.0	0.46	0.00
Sat-2	58.9	61.9	0.06	0.02	0.02	0.00	36.5	1.59	0.01
Shuttle	<i>N/A</i>	<i>N/A</i>	9.24*	0.70	0.68	9.28	<i>N/A</i>	20.8	0.09
Annth	123	81.8	0.09	0.02	0.01	0.00	28.7	2.06	0.01
Mammo	253*	223*	0.28	0.02	0.04	0.03	60.9	3.42	0.03
Http	<i>N/A</i>	<i>N/A</i>	<i>N/A</i>	24.0	58.7	0.24	194	272*	2.98
Musk	7.50	7.33	0.03	0.03	0.03	0.00	8.89	0.80	0.01
Average	73.9	62.5	1.38	3.09	7.43	1.19	48.5	37.6	0.39

* symbol means result from a single-run obtained on one AMD EPYC 7571 machine with 128GB RAM

Finally, we show in Figures 13- 20 accuracy plots for retrieval of objects whose number of objects retrieved is up to two times the cardinality of the ground truth dataset for each of the complete datasets. Note that *MetricABOD*’s accuracy increases steadily relative to other methods in several of the studied datasets, namely the Cardio, Shuttle, Mammography and Musk datasets.

aLOCI manifests initially very low accuracy on the Annthyroid dataset, on the order of 0.1, but at 20 it surpasses all other methods. HiCS also shows marked increase on the BreastW dataset, with accuracy values ranging from its initial 0.22 to 0.9.

Figure 13 – Accuracy for the BreastW dataset as the number of retrieved objects increases.

Figure 14 – Accuracy for the Cardio dataset as the number of retrieved objects increases.

Figure 15 – Accuracy for the Satimage-2 dataset as the number of retrieved objects increases.

Figure 16 – Accuracy for the Shuttle dataset as the number of retrieved objects increases.

Figure 17 – Accuracy for the Annthyroid dataset as the number of retrieved objects increases.

Figure 18 – Accuracy for the Mammography dataset as the number of retrieved objects increases.

5.4 Results: spotting outliers in adimensional data

This section describes additional experiments performed with two real collections of text data. We aim at showing that the new *MetricABOD* algorithm works also for adimensional, purely metric data. As we discussed before, *MetricABOD* must compute angles among pairs of objects (x,y) using other object P as the pivot. But, how to compute angles in an adimensional data

Figure 19 – Accuracy for the Http dataset as the number of retrieved objects increases.

Figure 20 – Accuracy for the Musk dataset as the number of retrieved objects increases.

space? To tackle the problem, we used a simple angle estimation strategy based on distances only: given x , y and P , the corresponding angle was computed from a triangle in the 2-dimensional space, whose side sizes are distances $d(x, y)$, $d(x, P)$ and $d(y, P)$.

Figure 21 – Rank of Brazilian names as per their average L-Edit distances to all others. Orange lines are the top-20 outliers; lines with text are the top-5 ones.

Figure 22 – Rank of NSF abstracts as per their average Jaccard distances to all others. Orange lines are the top-20 outliers; lines with text are the top-5 ones.

The datasets studied are:

- NSF abstracts: the NSF Research Award Abstracts 1990-2003 from the UCI repository (archive.ics.uci.edu/ml/datasets/). It has 129,000 abstracts describing awards granted by the National Science Foundation for basic research from 1990 to 2003. The abstracts were represented as sets of steamed words, and compared using the Jaccard distance function;
- Brazilian names: a set of 2,755 first names used in Brazil. It is available at: <https://gabrielrb.net/2011/10/18/dados-prontos-em-formato-sql-e-csv/>. The L-Edit distance function was applied to compare the names.

We ran *MetricABOD* in both datasets, thus obtaining the corresponding outlierness scores. Since there is no ground truth for these data, accuracy was empirically evaluated by verifying whether or not the highest scores are in objects far away from the others. Figure 21 ranks the Brazilian names according to their average distances, on metric space, to all other

names. Figure 22 ranks the NSF abstracts in a similar way. Calculation of average distances for each object of interest, as an outlier detection method on its own regard, was able to spot outliers in the 8 multidimensional datasets with an average accuracy of 51%. Since adimensional data are restricted to distance measures, we naturally extend the calculation of average distances to our own reference method for the experiments.

In each illustration, orange vertical lines indicate the top-20 outliers as per MetricABOD, and in arbitrary order, and blue dots indicate objects sorted as per their average distances to all other objects, i.e. as per x -axis order. Five of these lines report the actual names or abstracts' NSF identifiers; they are the top-5 outliers. Let us emphasize that in most cases the outliers are far away from any other object; note the log scale in the horizontal axes. Finally, as Brazilian citizens, we also argue that the most outlying names are indeed very distinct from names of people that are regularly used in Brazil.

5.5 Conclusions

This chapter presented an empirical evaluation performed with state-of-the-art outlier detection methods from the application specialist's point of view, and report results focused on quantifying how useful they can be for a variety of real world tasks of high impact, such as spotting breast cancer, heart and thyroid anomalies, detecting cyberattacks, and even identifying abnormalities in nature, like extreme natural events that might compromise preservation areas. We also report that our proposed *MetricABOD* algorithm surpassed the existing works both in runtime and accuracy of results, besides working with adimensional, purely metric data. The next chapter presents the conclusions of this MSc work.

CONCLUSION

This MSc work described an empirical study performed on unsupervised outlier detection, taking as a basis 8 datasets with ground truth that refer to a variety of real-world tasks of high relevance and impact, and 8 algorithms from the state-of-the-art. We provided feedback, as to the algorithms' performance, upon the summary of results we obtained. We pointed out to the strengths and weaknesses of each technique from the *application specialist's point of view*, which is commonly underrated in the outlier detection literature. The viewpoint of and input from the designer of outlier detection methods and methods related to machine learning in general, is undoubtedly of high importance. However, to cloister research within the designer viewpoint may have come to a stage of diminishing returns. We suggest, as a future direction for research in the area of outlier detection, to capture more feedback from the community outside academia.

Interestingly, many of the techniques that we studied had unfeasibly high runtime requirements or didn't succeed to spot what the specialists consider as outliers. With that in mind, we carefully designed *MetricABOD*: a novel ABOD-based algorithm that made the analysis up to *thousands of times faster*, still being in average 26% *more accurate* than the most accurate related work. The main innovation presented in our work is a new usage for tree-based Metric Access Methods that were originally designed to index complex data, but also have a variety of other uses, specially in data mining tasks, e.g., clustering (APPEL *et al.*, 2007; BARIONI *et al.*, 2008). Improvements such as those achieved by our work are essential to enable outlier detection in many real-world applications for which the existing methods lead to unexpected results or unfeasible runtime requirements. Additionally, we studied two real collections of text data to show that our *MetricABOD* works also for adimensional, purely metric data.

We must highlight that all theoretical models existing for unsupervised outlier detection draw assumptions that may not reflect the true nature of outliers in every single application and thus are not guaranteed to be effective in practice even if they are structurally robust and are built upon a sound theoretical basis (CAMPOS *et al.*, 2016; AGGARWAL, 2013). The results

reported in our work naturally corroborate this fact, since conclusions upon these results are glaring to notice. They demonstrate experimentally how a given method can be very accurate in specific cases and still fail in others, in the sense of returning or not what the application specialist expects from his or her data. Here, one must remember that specific data mining tasks all have biases that are not well-understood; such biases are well able to hinder a number of techniques that we studied. As an alternative to unsupervised outlier detection, we observe that supervised outlier detection can be useful in tasks where unsupervised methods do not perform acceptably. Further discussion can also be conducted as to the validity of the ground truth used and their representativeness as outliers. Nevertheless, a *few* exceptions apart, it is always desirable that outlier detection techniques return what the application specialists understand as outliers in their own data. The status of any technique should not be given as per its degree of sophistication, but as per its ability to respond well to practical demands.

As a suggestion for future improvements, we observe that for *top-q rankings*, or the ranking of the q most outlying objects in the dataset, as in *top-n LOF* (YAN; CAO; RUNDEN-STEINER, 2017), only the set of doable candidates, that is, the set of objects that are not clearly inliers, has to be considered. Irrelevant objects are pruned before the later stages of computation, which lowers the runtime. An interesting observation is that the dataset of representatives or dataset of samples retrieved by *MetricABOD* is an approximation to the original dataset of objects, with a data distribution that serves as an estimate to that of the original set. Therefore, any object in the dataset of sample objects that lies within a region with high density estimates probably has a high density estimate also in the original space. Their representatives, by extension, are most likely in high-density regions of the space. These objects could be discarded previously just by analyzing the set of sample objects and would not have to be checked. A top- q ranking can be computed using *MetricABOD* via such a strategy.

Finally, let us highlight that this MSc work generated the following publications:

1. Altamir Gomes Bispo Junior, Robson Leonardo Ferreira Cordeiro: Fast and Scalable Outlier Detection with Metric Access Methods. In: 19th International Conference on Computational Science – ICCS, 2019, Faro, Portugal. p. 189-203. (International Conference **Qualis A1**);
2. Lucas Felipe Kunze, Thábata Amaral, Leonardo Mauro Pereira Moraes, Jadson José Monteiro Oliveira, Altamir Gomes Bispo Junior, Elaine Parros Machado de Sousa, Robson Leonardo Ferreira Cordeiro: Classification Analysis of NDVI Time Series in Metric Spaces for Sugarcane Identification. In: 20th International Conference on Enterprise Information Systems – ICEIS, 2018, Funchal, Portugal. p. 162-169. (International Conference **Qualis B2**);

BIBLIOGRAPHY

AGGARWAL, C. C. **Outlier Analysis**. [S.l.]: Springer, 2013. Citations on pages [30](#), [35](#), [38](#), [41](#), [54](#), and [71](#).

AGGARWAL, C. C.; SATHE, S. **Outlier Ensembles - An Introduction**. [S.l.]: Springer, 2017. Citation on page [43](#).

AGGARWAL, C. C.; YU, P. S. Outlier detection for high dimensional data. **SIGMOD Rec.**, ACM, v. 30, n. 2, 2001. Citation on page [37](#).

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. Optics: Ordering points to identify the clustering structure. In: **Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data**. [S.l.]: ACM, 1999. Citation on page [34](#).

APPEL, A. P.; PATERLINI, A. A.; SOUSA, E. P. M. de; TRAINA, A. J. M.; TRAINA JR., C. Biased box sampling - a density-biased sampling for clustering. In: **Proceedings of the 2007 ACM Symposium on Applied Computing**. [S.l.]: ACM, 2007. Citation on page [71](#).

BARIONI, M. C. N.; RAZENTE, H. L.; TRAINA, A. J. M.; TRAINA JR., C. Accelerating k-medoid-based algorithms through metric access methods. **J. Syst. Softw.**, Elsevier Science Inc., v. 81, n. 3, 2008. Citation on page [71](#).

BAY, S. D.; SCHWABACHER, M. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: **ACM SIGKDD**. [S.l.: s.n.], 2003. Citations on pages [35](#), [37](#), and [45](#).

BREUNIG, M. M.; KRIEGEL, H.-P.; NG, R. T.; SANDER, J. LOF: Identifying density-based local outliers. **SIGMOD Rec.**, ACM, v. 29, n. 2, 2000. Citations on pages [35](#), [37](#), [45](#), and [46](#).

CAMPOS, G. O.; ZIMEK, A.; SANDER, J.; CAMPELLO, R. J. G. B.; MICENKOVÁ, B.; SCHUBERT, E.; ASSENT, I.; HOULE, M. E. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. **DMKD**, v. 30, n. 4, 2016. Citations on pages [29](#), [30](#), [31](#), [43](#), [54](#), and [71](#).

CHRISTY, A.; GANDHI, G. M.; VAITHYASUBRAMANIAN, S. Cluster based outlier detection algorithm for healthcare data. **Procedia Computer Science**, v. 50, 2015. Citation on page [39](#).

DUAN, L.; XU, L.; LIU, Y.; LEE, J. Cluster-based outlier detection. **Annals of Operations Research**, v. 168, n. 1, 2009. Citation on page [39](#).

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining**. [S.l.]: AAAI Press, 1996. Citation on page [34](#).

FAN, J.; LI, R. Statistical challenges with high dimensionality: feature selection in knowledge discovery. In: **Congress of Mathematicians**. [S.l.: s.n.], 2006. Citations on pages [29](#) and [36](#).

FAYYAD, U.; PIATETSKY-SHAPIO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, v. 17, 1996. Citation on page 34.

GHOTING, A.; PARTHASARATHY, S.; OTEY, M. E. Fast mining of distance-based outliers in high-dimensional datasets. **Data Mining and Knowledge Discovery**, v. 16, n. 3, 2008. Citations on pages 37 and 48.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. [S.l.]: Morgan Kaufmann Publishers Inc., 2011. Citations on pages 38 and 39.

HETLAND, M. L. The basic principles of metric indexing. In: _____. **Swarm Intelligence for Multi-objective Problems in Data Mining**. [S.l.]: Springer Berlin Heidelberg, 2009. Citation on page 42.

JOHNSTONE, I. M.; TITTERINGTON, D. M. Statistical challenges of high-dimensional data. **Philosophical Transactions A**, v. 367, n. 1906, 2009. Citation on page 29.

JUNIOR, A. G. B.; CORDEIRO, R. L. F. Fast and scalable outlier detection with metric access methods. In: **Computational Science – ICCS 2019**. [S.l.]: Springer International Publishing, 2019. Citation on page 32.

KELLER, F.; MULLER, E.; BOHM, K. Hics: High contrast subspaces for density-based outlier ranking. In: **2012 IEEE 28th International Conference on Data Engineering**. [S.l.: s.n.], 2012. Citations on pages 37 and 47.

KNORR, E. M.; NG, R. T. Algorithms for mining distance-based outliers in large datasets. In: **Proceedings of the 24rd International Conference on Very Large Data Bases**. [S.l.]: Morgan Kaufmann Publishers Inc., 1998. Citations on pages 35 and 39.

KRIEGEL, H.-P.; HUBERT, M. S.; ZIMEK, A. Angle-based outlier detection in high-dimensional data. In: **Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.]: ACM, 2008. Citation on page 51.

KRIEGEL, H.-P.; KRÖGER, P.; SCHUBERT, E.; ZIMEK, A. Outlier detection in axis-parallel subspaces of high dimensional data. In: **Advances in Knowledge Discovery and Data Mining**. [S.l.]: Springer Berlin Heidelberg, 2009. Citation on page 38.

KRIEGEL, H.-P.; SCHUBERT, M.; ZIMEK, A. Angle-based outlier detection in high-dimensional data. In: **ACM SIGKDD**. [S.l.: s.n.], 2008. Citations on pages 37, 38, 52, and 53.

MARQUES, H. O.; CAMPELLO, R. J. G. B.; ZIMEK, A.; SANDER, J. On the internal evaluation of unsupervised outlier detection. In: **SSDBM**. [S.l.: s.n.], 2015. Citations on pages 36, 54, and 55.

MULLER, E.; ASSENT, I.; STEINHAUSEN, U.; SEIDL, T. Outrank: ranking outliers in high dimensional data. In: **2008 IEEE 24th International Conference on Data Engineering Workshop**. [S.l.: s.n.], 2008. Citations on pages 37, 39, and 47.

MULLER, E.; SCHIFFER, M.; SEIDL, T. Adaptive outlierness for subspace outlier ranking. In: **Proceedings of the 19th ACM International Conference on Information and Knowledge Management**. [S.l.]: ACM, 2010. Citation on page 51.

MULLER, M. S. E.; SEIDL, T. Statistical selection of relevant subspace projections for outlier ranking. In: **IEEE ICDE**. [S.l.: s.n.], 2011. Citation on page 37.

PAPADIMITRIOU, S.; KITAGAWA, H.; GIBBONS, P. B.; FALOUTSOS, C. In: **ICDE**. [S.l.]: IEEE Computer Society, 2003. Citations on pages [37](#), [50](#), and [51](#).

PHAM, N. L1-depth revisited: A robust angle-based outlier factor in high-dimensional space. In: **Machine Learning and Knowledge Discovery in Databases**. [S.l.]: Springer International Publishing, 2019. Citation on page [77](#).

PHAM, N.; PAGH, R. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In: **ACM SIGKDD**. [S.l.: s.n.], 2012. Citations on pages [38](#) and [51](#).

QAZI, S.; KADRI, M. Revisiting constraint based geolocation: Improving accuracy through removal of outliers. v. 15, 2018. Citations on pages [39](#) and [50](#).

SMEULDERS, A. W. M.; WORRING, M.; SANTINI, S. Content-based image retrieval at the end of the early years. In: . [S.l.: s.n.], 2000. Citation on page [43](#).

TRAINA JR., C.; TRAINA, A.; FALOUTSOS, C.; SEEGER, B. Fast indexing and visualization of metric data sets using slim-trees. **IEEE TKDE**, v. 14, n. 2, 2002. Citations on pages [31](#), [41](#), and [62](#).

VARDI, Y.; ZHANG, C.-H. The multivariate l1-median and associated data depth. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 97, n. 4, 2000. Citation on page [77](#).

YAN, Y.; CAO, L.; RUNDENSTEINER, E. A. Scalable top-n local outlier detection. In: **Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.]: ACM, 2017. Citation on page [72](#).

ZAKI, M. J.; JR, W. M. **Data Mining and Analysis: Fundamental Concepts and Algorithms**. [S.l.]: Cambridge University Press, 2014. Citation on page [54](#).

ZIMEK, A.; SCHUBERT, E.; KRIEGEL, H.-P. A survey on unsupervised outlier detection in high-dimensional numerical data. **ASA Data Science**, v. 5, n. 5, 2012. Citations on pages [53](#) and [66](#).

ANNEX

A.1 L_1 -Depth

The L_1 -Depth method was developed by statisticians (VARDI; ZHANG, 2000) and is well-known. It is a depth-based outlier detection method whose conceptual relation to angle-based outlier detection methods has been explained in the literature (PHAM, 2019). L_1 -Depth is also computationally more efficient than ABOD, since its time complexity is linear per object. ABOD iterates over all distinct combinations of objects that form triplets whereas L_1 -Depth iterates over all distinct combinations of objects that form doubles.

MetricABOD takes advantage of Metric Access Methods to reduce the number of triplets that the method tests. However, in the case of *MetricABOD*, calculations on the order of quadratic complexity in the number of samples are still performed, for each object of interest. An alternative that reduces runtime even further is to substitute the angle-based algorithm at the outlier detection pass, after the selection of representatives, with L_1 -Depth. Table 5 presents accuracy results for the multidimensional datasets. *LI* is the base L_1 -Depth algorithm and *FLI* is a version of L_1 -Depth that benefits from the same improvements as our *MetricABOD* implementation for multidimensional data.

We highlight that L_1 -Depth was not intended in any way for adimensional, general metric data. Its accuracy guarantees hold for multidimensional data in $\mathbb{R}^d$. Nevertheless, since we showed promising results after we apply the concepts within ABOD on general metric spaces, there is room for exploration.

A.2 Correlation between outlier detection methods

We proceed to build a correlation matrix for the BreastW and Cardio datasets with Kendall rank correlation coefficients to verify by how much the rankings that are output per the

Table 7 – Kendall rank correlation coefficients for the Cardio dataset.

	ABOD	FABOD	LABOD	FVOA	aLOCI	KNN	LOF	<i>MetricABOD</i>	L1D	FL1D
Hi	-.030/.050	-.013/.381	.006/.686	.003/.825	-.048/.001	.025/.095	.003/.842	.003/.802	.022/.152	.028/.063
AB	*	-.017/.261	.131/.000	.018/.241	.024/.119	.015/.310	.007/.612	.044/.004	.037/.016	.016/.290
FA	*	*	.003/.822	-.005/.724	.044/.004	.033/.031	.005/.717	.023/.127	-.000/.973	.028/.065
LA	*	*	*	.024/.111	.027/.073	.011/.462	.019/.201	.038/.014	.017/.255	.044/.004
FV	*	*	*	*	-.010/.511	.000/.955	.019/.222	.038/.014	.015/.306	.037/.015
aL	*	*	*	*	*	-.021/.175	.008/.573	-.017/.261	-.024/.119	-.018/.229
KN	*	*	*	*	*	*	-.006/.667	.039/.011	.041/.007	.016/.292
LO	*	*	*	*	*	*	*	-.003/.811	.026/.090	.004/.775
MA	*	*	*	*	*	*	*	*	.084/.000	.054/.000
L1	*	*	*	*	*	*	*	*	*	.007/.640

It is seen that the rankings for the sibling methods ABOD and LB-ABOD are highly correlated, whereas the rankings for exact versus approximate methods, e.g. ABOD and FastABOD, LOF and aLOCI, are not as correlated. Due to a reasonable sample size of 680+ instances, there is high confidence that the statistics presented on Tables 6 and 7 are robust and significant. Kendall rank correlation, although powerful enough to find correlations between rankings, does not tell the entire story when similar rankings with high number of discordant pairs, albeit discordant due to small position “shifts”, are taken into consideration. Objects at discordant pairs may still exhibit some correlation in the form of closeness between the two rankings.

Figures 23 to 76 exhibit histograms of position distances between identical objects in the BreastW and Cardio datasets. Position distance is defined by the absolute value of the difference between the position of an object in the ranking *A* and the position of the first occurrence of the same object in ranking *B*. The values over the bar on vertical axis over each of the boxes discriminate the number of pairs of identical objects that lie within each range of position distances that are determined by the bar on the horizontal axis. One way to interpret correlation is to set a threshold in terms of position distance and calculate the ratio of the number of pairs of objects concentrated to the left up until the threshold to the total number of pairs. We split the methods that output the rankings into two classes: distance-based methods (HiCS, aLOCI, KNN-Outlier, LOF) and angle-based methods (ABOD, FastABOD, LB-ABOD, FastVOA, *MetricABOD*, L1-Depth and Fast L1-Depth).

Figure 23 – Histogram of position distances between same object for HiCS vs. aLOCI rankings – Cardio dataset.

Figure 24 – Histogram of position distances between same object for HiCS vs. KNN-Outlier rankings – BreastW dataset.

Figure 25 – Histogram of position distances between same object for HiCS vs. LOF rankings – BreastW dataset.

Figure 26 – Histogram of position distances between same object for aLOCI vs. KNN-Outlier rankings – BreastW dataset.

Figure 27 – Histogram of position distances between same object for aLOCI vs. LOF rankings – BreastW dataset.

Figure 28 – Histogram of position distances between same object for KNN-Outlier vs. LOF rankings – BreastW dataset.

Figure 29 – Histogram of position distances between same object for ABOD vs. FastABOD rankings – BreastW dataset.

Figure 30 – Histogram of position distances between same object for ABOD vs. LB-ABOD rankings – BreastW dataset.

Figure 31 – Histogram of position distances between same object for ABOD vs. FastVOA rankings – BreastW dataset.

Figure 32 – Histogram of position distances between same object for ABOD vs. *MetricABOD* rankings – BreastW dataset.

Figure 33 – Histogram of position distances between same object for ABOD vs. L1-Depth rankings – BreastW dataset.

Figure 34 – Histogram of position distances between same object for ABOD vs. Fast L1-Depth rankings – BreastW dataset.

Figure 35 – Histogram of position distances between same object for FastABOD vs. LB-ABOD rankings – BreastW dataset.

Figure 36 – Histogram of position distances between same object for FastABOD vs. FastVOA rankings – BreastW dataset.

Figure 37 – Histogram of position distances between same object for FastABOD vs. *MetricABOD* rankings – BreastW dataset.

Figure 38 – Histogram of position distances between same object for FastABOD vs. L1D rankings – BreastW dataset.

Figure 39 – Histogram of position distances between same object for FastABOD vs. FL1D rankings – BreastW dataset.

Figure 40 – Histogram of position distances between same object for LB-ABOD vs. FastVOA rankings – BreastW dataset.

Figure 41 – Histogram of position distances between same object for LB-ABOD vs. *MetricABOD* rankings – BreastW dataset.

Figure 42 – Histogram of position distances between same object for LB-ABOD vs. L1-Depth rankings – BreastW dataset.

Figure 43 – Histogram of position distances between same object for LB-ABOD vs. Fast L1-Depth rankings – BreastW dataset.

Figure 44 – Histogram of position distances between same object for FastVOA vs. *MetricABOD* rankings – BreastW dataset.

Figure 45 – Histogram of position distances between same object for FastVOA vs. L1-Depth rankings – BreastW dataset.

Figure 46 – Histogram of position distances between same object for FastVOA vs. Fast L1-Depth rankings – BreastW dataset.

Figure 47 – Histogram of position distances between same object for *Metri-cABOD* vs. L1-Depth rankings – BreastW dataset.

Figure 48 – Histogram of position distances between same object for HiCS vs. Fast L1-Depth rankings – Cardio dataset.

Figure 49 – Histogram of position distances between same object for L1-Depth vs. Fast L1-Depth rankings – BreastW dataset.

Figure 50 – Histogram of position distances between same object for HiCS vs. aLOCI rankings – Cardio dataset.

Figure 51 – Histogram of position distances between same object for HiCS vs. KNN-Outlier rankings – Cardio dataset.

Figure 52 – Histogram of position distances between same object for HiCS vs. LOF rankings – Cardio dataset.

Figure 53 – Histogram of position distances between same object for aLOCI vs. KNN-Outlier rankings – Cardio dataset.

Figure 54 – Histogram of position distances between same object for aLOCI vs. LOF rankings – Cardio dataset.

Figure 55 – Histogram of position distances between same object for KNN-Outlier vs. LOF rankings – Cardio dataset.

Figure 56 – Histogram of position distances between same object for ABOD vs. FastABOD rankings – Cardio dataset.

Figure 57 – Histogram of position distances between same object for ABOD vs. LB-ABOD rankings – Cardio dataset.

Figure 58 – Histogram of position distances between same object for ABOD vs. FastVOA rankings – Cardio dataset.

Figure 59 – Histogram of position distances between same object for ABOD vs. *MetricABOD* rankings – Cardio dataset.

Figure 60 – Histogram of position distances between same object for ABOD vs. L1-Depth rankings – Cardio dataset.

Figure 61 – Histogram of position distances between same object for ABOD vs. Fast L1-Depth rankings – Cardio dataset.

Figure 62 – Histogram of position distances between same object for FastABOD vs. LB-ABOD rankings – Cardio dataset.

Figure 63 – Histogram of position distances between same object for FastABOD vs. FastVOA rankings – Cardio dataset.

Figure 64 – Histogram of position distances between same object for FastABOD vs. *MetricABOD* rankings – Cardio dataset.

Figure 65 – Histogram of position distances between same object for FastABOD vs. L1-Depth rankings – Cardio dataset.

Figure 66 – Histogram of position distances between same object for FastABOD vs. Fast L1-Depth rankings – Cardio dataset.

Figure 67 – Histogram of position distances between same object for LB-ABOD vs. FastVOA rankings – Cardio dataset.

Figure 68 – Histogram of position distances between same object for LB-ABOD vs. MetricABOD rankings – Cardio dataset.

Figure 69 – Histogram of position distances between same object for LB-ABOD vs. L1-Depth rankings – Cardio dataset.

Figure 70 – Histogram of position distances between same object for LB-ABOD vs. Fast L1-Depth rankings – Cardio dataset.

Figure 71 – Histogram of position distances between same object for FastVOA vs. *MetricABOD* rankings – Cardio dataset.

Figure 72 – Histogram of position distances between same object for FastVOA vs. L1-Depth rankings – Cardio dataset.

Figure 73 – Histogram of position distances between same object for FastVOA vs. Fast L1-Depth rankings – Cardio dataset.

Figure 74 – Histogram of position distances between same object for *MetricABOD* vs. L1-Depth rankings – Cardio dataset.

Figure 75 – Histogram of position distances between same object for *MetricABOD* vs. Fast L1-Depth rankings – Cardio dataset.

Figure 76 – Histogram of position distances between same object for L1-Depth vs. Fast L1-Depth rankings – Cardio dataset.