UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

MASTER THESIS IN COMPUTER ENGINEERING

Comparison of Isolation Forest and Clustering Methods for Outlier Detection

CANDIDATE
Xhacu Bejaj
Student ID 2055172

SUPERVISOR
Prof. Andrea Alberto Pietracaprina
University of Padova

ACADEMIC YEAR 2024/2025
July 7th, 2025

Abstract

In this thesis, we study the relationship between the notion of outlier employed by Isolation Forest and the 3-approximation algorithm for solving the k-center with z outliers problem. Both algorithms' strategy is influenced by the concept of density, which motivates our comparison. We also design a new method employing Isolation Forest as a preprocessing step for efficiently solving the k-center with z outliers problem. Through our experimental analysis we find that, depending on outlier type, these methods do not always return similar sets of outliers but nonetheless the returned outliers are of comparable outlying degree. Furthermore, the proposed method shows substantial efficiency gains, with a nearly linear complexity as opposed to the more than quadratic complexity of the classical 3-approximation algorithm.

Sommario

In questa tesi, studiamo la relazione tra la nozione di outlier impiegata da Isolation Forest e l'algoritmo di 3-approssimazione per la risoluzione del problema k-center con z outlier. La strategia di entrambi gli algoritmi è influenzata dal concetto di densità, il chè motiva il nostro confronto. Proponiamo inoltre un nuovo metodo che impiega Isolation Forest come fase di preprocessing per risolvere efficientemente il problema k-center con z outlier. Attraverso la nostra analisi sperimentale scopriamo che, a seconda del tipo di outlier, questi metodi non restituiscono sempre insiemi simili di outlier, tuttavia gli outlier restituiti presentano un comparabile grado di anomalia. Inoltre, il metodo proposto mostra guadagni sostanziali in termini di efficienza, con una complessità quasi lineare, rispetto alla complessità più che quadratica dell'algoritmo classico di 3-approssimazione.

Contents

Chapter 1

Introduction

In an increasingly data-driven world, the ability to identify unusual patterns in data has become crucial across numerous domains, such as healthcare, cybersecurity and fault detection in industrial settings. The task of outlier detection directly addresses this issue, yet poses several significant difficulties. One of these challenges is the problem of definition itself. Hawkins provided what has become a seminal definition: "*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism*"[1].

The vagueness in this definition is noteworthy, as data instances can deviate from normal behavior in many different ways. Consequently, the field has evolved such that different outlier detection methods operate with their own specific definitions of outlier, closely tied to their methodological approach. Any definition of outlier inherently makes assumptions about what constitutes normal behavior, with outliers being those data instances that do not fit the method-specific normal model.

Another significant challenge in outlier detection is the inherent scarcity of labeled data, leading most detection methods to adopt unsupervised approaches. This preference is further justified by the fact that when the goal is to discover anomalous patterns that were never encountered before, labels could even be undesirable [2].

The terms outlier and anomaly are often used interchangeably, but there is a subtle distinction. The difference is primarily semantic and depends on what the analyst considers to be an interesting deviation from normal behavior [3]. Anomalies represent meaningful deviations that are of interest to the analyst, while outliers could be either true anomalies (strong outliers) or merely noise (weak outliers). Given this application-specific and subjective nature, most detection methods focus on identifying any unusual patterns.

Of particular relevance to this thesis is Isolation Forest [4], a widely adopted outlier detection algorithm due to its computational efficiency and effectivness in numerous contexts.

Outlier detection is closely tied with the clustering problem, which asks to partition data into clusters such that similar instances are placed in the same cluster and dissimilar instances into different clusters. In this setting outliers are instances that do not belong to any cluster.

A popular formulation of the clustering problem called k-center clustering requires that, given an input set of data instances, k instances referred to as *centers* are determined, which minimize the maximum distance of any data instance to its closest center. While NP-Hard, an approximate solution can be efficiently found by the algorithm proposed in [5], which also provides theoretical guarantees on its solution's quality.

Notably k-center clustering does not account for the presence of outliers in the data, but, nonetheless, outliers can, in principle, affect the quality of the solutions significantly. To address this limitation, [6] introduced a variant called k-center with z outliers. The problem asks to find k centers that minimize the maximum distance of any instance to its closest center, excluding the z farthest instances, which are considered anomalous, hence not belonging to any cluster.

In [6] the authors also propose a 3-approximation algorithm for the k-center variant with z outliers, which is referred to as KCenterOUT. The main idea behind the algorithm is that centers should be found in the densest regions of the feature space. Its high computational complexity, however, makes it unsuitable in the context of big data.

These issues inspired us to devise a new methodology for efficiently solving the problem which combines known outlier detection and k-center clustering algorithms.

1.1 Objectives and Contributions of the Thesis

The objectives of this thesis are to study the relationship between the concept of outlier employed by *Isolation Forest* [4] and *KCenterOUT* [6], and understand whether Isolation Forest can be used as a preprocessing step to efficiently solve the k-center with z outliers problem. Both algorithms leverage density-based principles: Isolation Forest for outlier detection, and KCenterOUT for producing outlier-robust clustering.

To determine whether the density-based strategies that both algorithms employ translate to similar notions of outlier, we first define an outlier score based on the solution (i.e., the set of k centers) returned by KCenterOUT and then compare the outlier sets identified by each algorithm as well as the rankings of all instances provided by the respective outlier scores.

For the second objective, we devise a new approach using Isolation Forest for outlier detection and *Farthest First Traversal* [5] to compute a solution for the k-center problem on the cleaned dataset where z outliers identified by Isolation Forest have been removed, and compare it with the solution obtained by applying KCenterOUT to the full dataset. More specifically, we compare the values of the objective function for the k-center with z outliers problem computed using the respective sets of centers returned by each method.

To understand how the type of outlier influences the experimental analysis, we created artificial datasets of different dimensionality for each type of outlier. To validate the practical applicability of the proposed method, experiments were also conducted on real-world datasets.

We found that the solution returned by the proposed method is only slightly worse than the one returned by KCenterOUT, yielding values of objective function which are higher by a factor at most two in most cases. Interestingly, the degree of this performance gap depended on the proximity relationship between outliers and normal instances, and was independent of outlier's distributional properties such as clusteredness/scatteredness. On the other hand, the efficiency advantage is substantial and increases with dataset size. The proposed method was at least one order of magnitude faster, and up to three orders of magnitude faster for larger datasets. The empirical evidence also showed that even when Isolation Forest and KCenterOUT return considerably different sets of outliers, the respective solutions yield similar values of objective function, suggesting that the outliers returned by the two methods are of comparable outlying degree.

1.2 Thesis Structure

The rest of this thesis is organized as follows. In Chapter 2 we provide a commonly used taxonomy to categorize outlier types and review the most prominent outlier detection approaches. In Chapter 3 we provide a formal definition of the k-center clustering problem and its variant with z outliers, followed by a detailed review of the three algorithms involved in our experimental analysis. Chapter 4 describes the experiments we designed and the results obtained, the procedures implemented to create artificial datasets, and the rationale behind our choice of real-world datasets. In conclusion, in Chapter 5 we briefly summarize the results obtained and provide suggestions for future research.

Chapter 2

Outlier Detection

The design of outlier detection algorithms requires a precise definition of outlier. However, what constitutes an anomalous pattern is inherently context-dependent [7], making a universal definition unachievable. Nonetheless, outliers can be categorized based on fundamental characteristics such as their distributional properties and proximity relationship with normal instances [8].

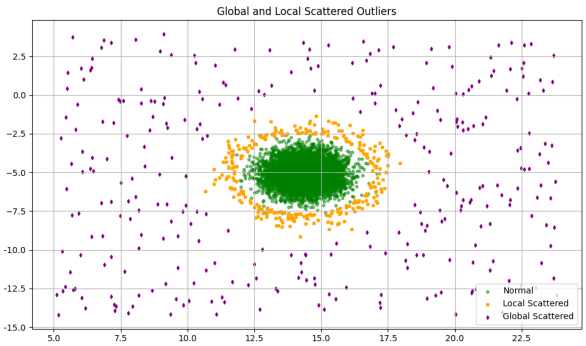
The following sections present not only a commonly used taxonomy [9] to categorize outlier types, based on the aforementioned characteristics, but also a classification of outlier detection methods, illustrating the most prominent works for each class of algorithms.

2.1 Types of Outliers

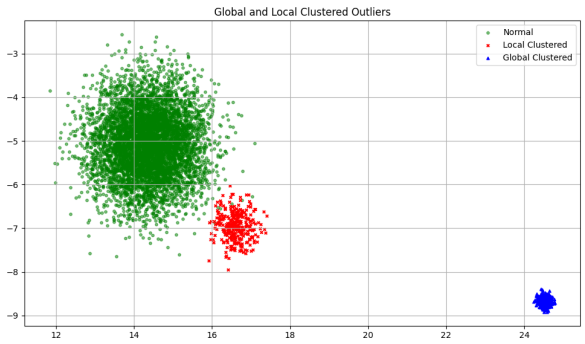
Based on outliers' data distribution, they can be divided into *scattered* or *clustered*. Scattered outliers involve single instances that deviate from normal patterns, while clustered outliers consist of a collection of related data instances, forming an anomalous pattern.

Furthermore, considering their proximity relationship with normal instances, they can be categorized as *local* or *global* outliers. Local outliers are anomalous only within their neighborhood but may appear normal when considered against the entire dataset. Global outliers instead, deviate significantly from the overall data distribution.

These categories are not mutually exclusive, for example, outliers can be global and clustered, meaning an anomalous cluster that significantly deviates from the overall data distribution.



(a) Scattered Outliers



(b) Clustered Outliers

Figure 2.1: Examples of clustered and scattered outliers

2.2 Overview of Outlier Detection Approaches

Outlier detection algorithms are commonly classified based on the adopted detection strategy, which in turn, is informed by the notion of outlier used by algorithm designers. The main classes of outlier detection algorithms include proximity-based and projection-based approaches [8]. Proximity-based approaches define outliers based on the relationship with their neighborhoods, while projection-based methods project instances into a new space with lower complexity, where outliers are more easily distinguished from normal instances.

2.2.1 Proximity-Based Approaches

Proximity-based approaches can be further divided, depending on the notion of proximity, into nearest-neighbor methods where outliers are identified based on instances' local density patterns, and clustering-based methods where outlying behavior is defined by the relationship between instances and clusters. The fundamental assumption shared by all proximity-based algorithms, is that normal instances are located in high density regions, while outliers lie in sparse areas of the feature space.

A nearest-neighbor algorithm, such as [10] defines outliers as data instances whose neighborhood, defined by a radius r, contains fewer than p instances. This definition assumes that normal instances must obey the same minimun density requirements, since parameters r and p, are the same for all instances of the dataset. Figure 2.2 illustrates a dataset where this assumption does not hold. The dataset is composed of two normal clusters, with one cluster being much sparser than the other, and a single local outlier located near the dense normal cluster. In this case there are no values of r and p that can detect the local outlier near the dense cluster, without mistakenly labeling some instances of the sparse cluster as outliers. For this reason, this method is effective when global outliers are present in the dataset.

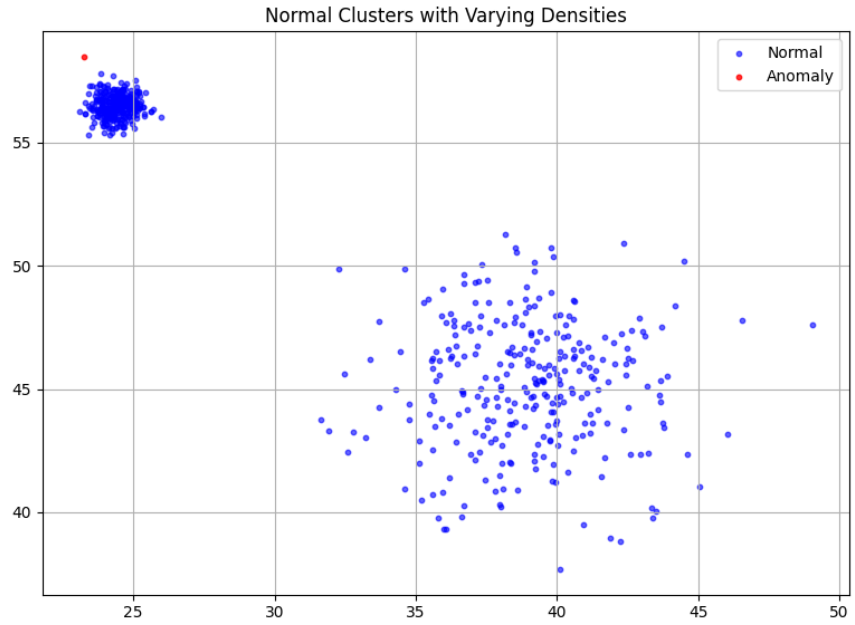


Figure 2.2: In blue normal clusters of varying density. In red a single local outlier, close to the dense cluster

Other nearest-neighbor based methods such as Local Outlier Factor (LOF)[11], are concerned with detecting local outliers. Differently from [10], which returns binary labels(outlier or normal), LOF computes an outlier score based on the ratio between the density of an instance and the average density of its neighbors. The assumption here, is that the density of local outliers is lower than the density of its neighbors. An outlier score defined in this manner is relative to each instance's neighborhood, therefore it accounts for normal clusters with different densities.

The main drawback of nearest-neighbor approaches is their reliance on pairwise distance computations to compute each instance's density, resulting in quadratic complexity. On the other hand, these pairwise distance computations provide a more detailed analysis, than for instance clustering-based algorithms, making nearest-neighbors methods capable of distinguishing outliers from possibly uninteresting deviations, such as randomly distributed noise [3].

Clustering-based methods take a different approach by first partitioning the dataset into clusters, then computing outlier scores based on instances' relationship with these clusters. The assumption is that outliers are located in sparse and small clusters. Cluster-Based Local Outlier Factor (CBLOF) [12] uses a heuristic rule to identify *large* clusters, which are assumed to represent normal patterns. This procedure makes CBLOF more robust to clustered outliers than nearest-neighbor algorithms. The method then computes an outlier score based on an instance's distance from its closest large cluster.

A fundamental issue concerning this class of algorithms is that their effectiveness depends on the quality of the clustering, and, in the particular case of CBLOF, also on the parameters used to define large (and hence normal) clusters.

2.2.2 Projection-Based Approaches

Differently from proximity-based approaches which work in the original feature space, projection-based ones project the data into a new space, where outliers are more easily distinguishable, based on properties of the projection space[8]. For this reason, the assumptions these algorithms make, vary depending on the specific projection, and are not shared as was the case for proximity-based approaches.

The most promising subclass projection-based outlier detection approaches are the tree-based, whose main representative is Isolation Forest [4], [13]. In contrast with the algorithms described above, Isolation Forest does not make strong assumptions about anomalous patterns. It assumes outliers to be rare data instances with attribute values that are significantly different from those of normal instances. These properties make outliers easier to isolate with a tree-based structure.

The algorithm consists of a training phase, where t small subsamples are drawn from the dataset, then each subsample is recursively partitioned by randomly selecting a split attribute, and a corresponding split value. The partitioning ends when all instances in the subsample are isolated, or when a specific tree depth is reached, creating a structure akin to binary search trees, called *iTree*. Subsequently, in the evaluation phase, each data instance traverses the t iTrees that make up the isolation forest, and an outlier score is computed based on the average path length of the traversal. The idea behind this procedure is that since outliers are easier to isolate, their path length is shorter than that of normal instances.

As it will be discussed in depth in the next chapter, Isolation Forest is highly efficient and effective against various types of outliers, with the exception of local outliers.

Several variants have been developed that tackle its main limitations. One such limitation lies in the partitioning strategy: axis-parallel partitions introduce artifacts that give instances of similar anomalous degree a different outlier score, when ideally the score should depend only on how anomalous an instance is, based on the algorithm's assumptions about what constitutes outlying behavior.

To address this issue, Extended Isolation Forest [14], selects multiple attributes for each split, resulting in partitioning hyperplanes with random slopes, which remove the inherent bias of axis-parallel splits.

Chapter 3

Isolation Forest and *K*-Center with Z Outliers

This chapter provides a description of the algorithms involved in the experimental evaluation. The proposed methodology consist of solving the k-center with z outliers problem by first employing Isolation Forest to detect and remove outliers from the dataset, then applying *FFT*(*Farthest First Traversal*) [5] to solve the k-center clustering problem, on the cleaned dataset. Furthermore, we describe *KcenterOUT* [6] a 3-approximation algorithm specifically designed for the k-center with z outliers variant.

3.1 Isolation Forest

Isolation Forest is the first algorithm to employ *isolation* to detect outliers, where isolation refers to any process that separates data instances. The method assumes that outliers comprise few instances exibiting attribute values significantly different from those of normal instances. These assumed characteristics make outliers more easily separable, which justifies the use of an isolation mechanism for detection.

To realize the idea of isolation, [4], [13] propose a binary search tree structure, named *isolation tree* (iTree), which naturally separates instances. Outliers, for the properties stated above, are likely to be isolated closer to the root, therefore the path length of an instance traversing an iTree, is used as a measure of outlying degree.

The method is composed of a training and an evaluation stage. During the training stage, an ensemble of isolation trees is constructed. In the evaluation stage, the outlying degree of an instance x, is estimated by averaging the path length, $h(x)$, from root node, to a terminal node, across all trees of the ensemble. After the evaluation stage has terminated, an outlier score is computed for each instance.

Algorithm 1, in Figure 3.1, builds the isolation forest. It extracts a subsample of size ψ from dataset X, sets the height limit based on ψ and calls Algorithm 2 to construct t isolation trees.

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - subsampling size

Output: a set of t *iTrees*

Initialize $Forest$
set height limit $l = ceiling(\log_2 \psi)$
for $i = 1$ to t **do**
    $X' \leftarrow sample(X, \psi)$
    $Forest \leftarrow Forest \cup iTree(X', 0, l)$
end for
return $Forest$

Figure 3.1: Algorithm to build an isolation forest (source [13])

Algorithm 2, in Figure 3.2, is used to build an isolation tree. The algorithm receives in input a subsample of the dataset, *X'*, and a height limit *l*. It then partitions X' into subsets X_l, X_r by randomly choosing a split attribute q, and drawing a split value p, uniformly at random from the selected attribute's range. Subsets X_l, X_r are represented by the respective internal nodes *Left, Right*. Each internal node stores the split attribute q and split value p, as they are necessary information for tree traversal, performed in the evaluation stage.

The process continues recursively for each internal node until either all instances have been isolated, or the height limit is reached. Upon termination, the size of the remaining subset is stored in the corresponding external (leaf) node, marking the conclusion of the partitioning process for that branch.

Algorithm 2 : $iTree(X, e, l)$

Inputs: X - input data, e - current tree height, l - height limit

Output: an iTree

if $e \geq l$ or $|X| \leq 1$ **then**
    return *exNode*$\{Size \leftarrow |X|\}$
else
    let Q be a list of attributes in X
    randomly select an attribute $q \in Q$
    randomly select a split point p from max and min values of attribute q in X
    $X_l \leftarrow filter(X, q < p)$
    $X_r \leftarrow filter(X, q \geq p)$
    return *inNode*$\{Left \leftarrow iTree(X_l, e+1, l)$,
                $Right \leftarrow iTree(X_r, e+1, l)$,
                $SplitAtt \leftarrow q$,
                $SplitValue \leftarrow p\}$
end if

Figure 3.2: Algorithm to build an iTree (source [13])

In the worst case, to build t iTrees, all of the ψ instances of the subsample would reach the height limit $\log(\psi)$ in each tree, resulting in $O(t\psi \log(\psi))$ time complexity and

$O(t\psi)$ space complexity. Since both t and ψ are constants independent of dataset size, the training stage has constant complexity.

Algorithm 3 : $PathLength(x, T, e)$

Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of x

1: **if** T is an external node **then**
2: return $e + c(T.size)$
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

Figure 3.3: Path Length estimation (source [13])

In the evaluation stage, Algorithm 3, in Figure 3.3, estimates the path length of an instance x, as it traverses an iTree T. The algorithm assumes that tree traversal of outlier instances terminates before reaching the height limit, as they are expected to be easily isolatable. Normal instances instead, are assumed to reach an external node T at height limit depth, hence there could be additional partitioning steps that would have occurred in a complete tree. To account for the expected contribution of this unbuilt subtree represented by the external node, an adjustment $c(T.size)$ is added to the path length, where $T.size$ represents the number of instances remaining in the external node [15]. The adjustment is defined as

$$c(T.size) = \begin{cases} 2H(T.size - 1) - 2(T.size - 1)/T.size & \text{for } T.size > 2, \\ 1 & \text{for } T.size = 2, \\ 0 & \text{otherwise,} \end{cases}$$

where $H(T.size)$ is the harmonic number and it can be estimated as $ln(T.size) + 0.5772156649$ (Euler's constant) [4].

The average path length of instance x across all isolation trees in the ensemble is then used to compute the outlier score. Given that $c(\psi)$ represents the expected path length of an instance x, traversing a tree of size ψ, the outlier score of x given ψ is computed as:

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(\psi)}}$$

Evaluating all n instances of the dataset results in a time complexity of $O(nt\log(\psi))$, since, in the worst case, $\log(\psi)$ splits could be required to isolate each instance across each tree.

The score compares the reported average path length $E(h(x))$, with $c(\psi)$, which acts as a baseline for normality. Since each tree is built by randomly sampling: ψ instances from the dataset, an attribute q and split value p, each path length $h_i(x)$ for $i = 1, \ldots, t$ is a random variable with expectation $\mu(x, \psi, l)$, hence $E(h(x)) = \frac{1}{t}\sum_{i=1}^{t} h_i(x)$ is an approximation of $\mu(x, \psi, l)$.

Figure 3.4: Distribution of values for an attribute q. Green dots represent values of normal instances, while red dots represent values of outliers. In blue, the range of split values that can separate them

This characterization of the outlier score reveals some flaws of Isolation Forest. A hidden assumption of this method is that choosing an attribute q and a split value p, uniformly at random, is a good strategy for isolating outliers.

Figure 3.4 shows the distribution of values of an attribute q, where the assumption holds. Anomalous attribute values are very different from normal values, therefore there is a high probability that Isolation Forest selects a split value p that separates outliers, from normal instances. In such cases, $\mu(x, \psi, l)$ reflects an instance's outlying degree.

The assumption does not hold in two cases, described below.

When dealing with local outliers whose values are similar to those of neighboring normal instances, the range of effective split values narrows significantly, reducing the probability of successful isolation through random splitting, increasing path length of the instance being evaluated.

When a subset of attributes of the anomalous instance exibith normal values, which we refer to as *irrelevant attributes*, there are no split values that can separate the outlier from normal instances, hence irrelevant attributes increase path length, distorting the outlier score.

3.2 Farthest First Traversal

Given a dataset P of size N, and an integer $k < N$, the *k-center clustering problem* requires finding a set $S \subset P$, of k centers that minimize the maximum distance of any instance to its closest center.

The k-center clustering problem is NP-Hard, therefore algorithms for solving it only provide approximate solutions. One such algorithm is *FFT* (*Farthest First Traversal*) [5]. The idea behind FFT is that centers should be spread out in the feature space so that no instance is too far from its closest cluster center.

The algorithm selects the first center arbitrarily from dataset P. The subsequent centers are chosen, in a greedy fashion, as the farthest instances from the current set of centers S. The process continues until k centers are found, hence solution S is reported in output.

Algorithm 4 : *Farthest First Traversal(P,k)*

Inputs: P - Set of N instances, k - number of centers
Output: A set S of k centers

1: $S \leftarrow \{c_1\}$ ▷ $c_1 \in P$ arbitrary point
2: **for** $i = 2$ **to** k **do**
3: Find the point $c_i \in P - S$ that maximizes $d(c, S)$
4: $S \leftarrow S \cup \{c_i\}$
5: **end for**
6: **return** S

To find the required number of centers, k distance computations are needed for each instance, yielding a computational complexity of $O(Nk)$. Furthermore, *FFT* is a 2-approximation algorithm for the k-center clustering problem, which is the best possible approximation factor [5].

3.3 *K*CenterOUT

Given a dataset P of size N, and integers k, z, with $k + z <$ N, the *k-center with z outliers clustering problem* requires finding a set $S \subset P$, of k centers that minimize the maximum distance of an instance to its closest center, excluding the z farthest instances.

To solve this problem, [6] propose KCenterOUT. The unweighted variant of this algorithm is described below.

Algorithm 5 : *KCenterOUT(P,k,z,α)*

Inputs: P - Set of N instances, k - number of centers, z - number of outliers in P, α - scaling factor for radius r

Output: A set S of k centers

$r \leftarrow$ (min distance between first $k + z + 1$ instances)/2
while true do
    $Z \leftarrow P$; $S \leftarrow \emptyset$; $W_Z = |P|$
    while $(|S| < k)$ **AND** $(W_Z > 0)$ **do**
        max $\leftarrow 0$
        for $x \in P$ **do**
            ball-weight $\leftarrow |B_Z(x, (1 + 2\alpha)r)|$
            if ball-weight $>$ max **then**
                max $\leftarrow$ ball-weight
                newcenter $\leftarrow x$
            end if
        end for
        $S \leftarrow S \cup \{\text{newcenter}\}$
        for $y \in B_Z(\text{newcenter}, (3 + 4\alpha)r)$ **do**
            remove y from Z
            $W_Z \leftarrow W_Z - 1$
        end for
    end while
    if $W_Z \leq z$ **then**
        return S
    else
        $r \leftarrow 2r$
    end if
end while

This is a clustering algorithm, therefore its main concern is to find a suitable set of centers. Outliers are found as a byproduct of the clustering process.

The core assumption of KCenterOUT is that cluster centers should be located in the densest regions of the feature space. Furthermore, to provide a good coverage of the dataset, centers should not be close to each other.

To realize this idea, the algorithm defines an initial radius guess r and attempts to find k centers that cover, within a certain radius, all but z instances. If, once k centers are found, more than z instances remain uncovered, then the radius guess is doubled and the process is repeated, until the coverage constraint is satisfied.

For each radius guess, the algorithm counts the number of uncovered instances inside the neighborhood of each instance x, defined as the ball of radius $(1 + 2\alpha)r$ where r is the current radius guess and α a scaling parameter.

The instance with the densest neighborhood is selected as the new center. Once a center is selected, instances within its extended neighborhood defined as the ball of radius $(3 + 4\alpha)$r, are considered as *covered*. The covered instances, can be selected as centers in the next iterations, but they do not contribute to the density calculation. This ensures that the next centers avoid regions already covered by previously selected centers.

The center selection strategy requires, in the worst case, $O(n^2)$ distance computations. Since there are k centers to select, the algorithm has a computational complexity of $O(kn^2)$ per radius guess.

The advantage of this algorithm is in its tight approximation guarantees. The unweighted version of KCenterOUT with $\alpha = 0$, returns a 3-approximate solution to the k-center clustering problem with z outliers [6].

3.3.1 Outlier Scoring Method for KCenterOUT

Once the algorithm has returned the set of centers S, the outlier score of each instance is computed as the distance from its closest center. Then the z instances farthest from their closest cluster center are reported as outliers.

Chapter 4

Experimental Analysis

The objectives of this thesis are to understand the relationship between the concept of outlier of Isolation Forest and KCenterOUT, and evaluate whether Isolation Forest can be used as a preprocessing step to efficiently solve the k-center with z outliers problem. For these purposes, we designed two sets of experiments.

In the first set of experiments we configure Isolation Forest's and KCenterOUT's parameters to ground truth values and compare the sets of outliers returned, as well as the outlier rankings of all instances of the dataset.

In the second set of experiments we compare the objective functions, for the k-center with z outliers problem, of the proposed methodology and KCenterOUT, varying the algorithms' parameters such as the number of outliers to report, z, and the number k of centers to be returned.

Experiments have been conducted on both real-world and artificial datasets. Each artificial dataset contains a specific type of outlier, consistently with the categorization described in Section 2.1. The simple data distribution and known cluster structure of artificial datasets allow for more interpretable results, while the complex distributions that characterize real-world datasets provide more meaningful insights.

All experiments were conducted on a machine running Windows 11 on a AMD Ryzen 5 5600 6-Core 3.50 GHz processor with 16 GB RAM. Algorithms were implemented in Java 21.

The following sections first briefly define the metrics used in this Chapter, then provide a description of the datasets used and of the experiments carried out, concluding with a discussion of the results.

4.1 Technical Notions

In this section we provide a brief definition of the various metrics used in the next sections.

Mahalanobis distance: Let $\Sigma \in \mathbb{R}^{d \times d}$ be the covariance matrix of a dataset. Then, the *Mahalanobis distance* between two d-dimensional data points x, y is defined as:

$$D_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top \Sigma^{-1} (\mathbf{x} - \mathbf{y})}$$

True Positive (TP): Outliers correctly identified as outliers.
False Positive (FP): Normal instances incorrectly identified as outliers.
True Negative (TN): Normal instances correctly identified as normal.
False Negative (FN): Outliers incorrectly identified as normal.

AUC ROC: The ROC curve plots True Positive Rate ($TPR = TP/(TP+FN)$) against False Positive Rate ($FPR = FP/(FP+TN)$) at various thresholds. AUC ROC measures the area under this curve, ranging from 0 to 1, where 0.5 indicates random performance and 1.0 indicates perfect classification. In the context of this thesis, the AUC ROC score indicates the probability that when picking one outlier and one normal instance uniformly at random, the outlier has a higher outlier score than the normal instance.

AUC PR: The Precision-Recall curve plots Precision ($TP/(TP+FP)$) against Recall ($TP/(TP+FN)$) at various thresholds. AUC PR measures the area under this curve, ranging from 0 to 1, where higher values indicate better performance. AUC PR is more suitable for imbalanced datasets like outlier detection problems.

Silhouette Score: For each instance, the *silhouette score* is calculated as

$$\frac{b - a}{\max(a, b)}$$

where a is the mean distance to other instances in the same cluster and b is the mean distance to instances in the nearest neighboring cluster. The overall silhouette score is the average across all instances, ranging from -1 to 1, where higher values indicate better clustering quality with well-separated and cohesive clusters.

Davies-Bouldin Index: Measures clustering quality by evaluating the average similarity between each cluster and its most similar cluster. For each cluster i, it computes

$$\max_{j \neq i} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

where σ_i is the average distance from all points in cluster i to the cluster centroid c_i, and $d(ci, cj)$ is the distance between centroids of clusters i and j. The Davies-Bouldin Index is the average of these values across all clusters, ranging from 0 to infinity, where lower values indicate better clustering with more compact and well-separated clusters.

Jaccard Similarity: Measures the similarity between two sets by calculating the ratio of their intersection to their union:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where A and B are two sets. The Jaccard similarity ranges from 0 to 1, where 0 indicates no overlap between sets and 1 indicates identical sets.

Spearman's Rank Correlation: Measures the monotonic relationship between two ranked variables by calculating the Pearson correlation coefficient of the rank values:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where d_i is the difference between the ranks of corresponding values and n is the number of observations. Spearman's correlation ranges from -1 to 1, where -1 indicates perfect negative correlation, 0 indicates no correlation, and 1 indicates perfect positive correlation.

4.2 Dataset Description

This section illustrates the reasoning behind our choice of real-world datasets, as well as a description of the procedures used to create the artificial datasets.

4.2.1 Real-World Datasets

Real-world datasets were selected from the ADBench project [16] which includes many commonly used datasets for outlier detection tasks. Selection was guided by constraints imposed by algorithms involved and the available computational resources.

Farthest First Traversal and KCenterOUT rely on Euclidean distance computations, hence to mitigate the effects of the curse of dimensionality, selection was limited to datasets with low number of dimensions. Moreover, KCenterOUT has more than quadratic complexity with respect to dataset size, further constraining the choice to relatively small datasets.

For meaningful experimentation, also algorithms' performance was taken into account. KCenterOUT and Farthest First Traversal are guaranteed to provide good solutions for their respective problems, regardless of dataset used, therefore to establish comparable evaluation conditions to all three algorithms, selection focused on datasets where Isolation Forest exhibits high detection accuracy when compared to other outlier detection methods [16].

Only three datasets satisfy these constraints: *Annthyroid*, *Thyroid*, *Shuttle*. Table 4.1 summarizes the available information about these datasets.

Table 4.1: Real-world datasets and Isolation Forest performance

Dataset	**Size**	**# Dimensions**	**# Outliers**	**% Outliers**	**AUC ROC**	**AUC PR**
Annthyroid	7200	6	534	7.42	0.82	0.35
Thyroid	3772	6	93	2.47	0.98	0.63
Shuttle	49097	9	3511	7.15	0.98	0.97

4.2.2 Artificial Datasets

To ensure diversity, we created datasets from a suitable mixture of distributions. Two-dimensional datasets were particularly useful to visualize algorithms' behavior such as selected centers and detected outliers, while 4-dimensional and 10-dimensional datasets provided more realistic experimental conditions. We created three instances of each dataset to minimize dataset-specific bias on our experiments, hence all results described in Section 4.3 report averages over the three versions.

The fundamental purpose of the artificial datasets is to investigate how outlier types influence results. For this reason, we created a dataset for each outlier type.

The dataset creation procedure consists of first creating normal instances and subsequently generate a specific type of outlier to inject. Decoupling the generation of normal and outlier instances, allows us to isolate the influence of outlier types, by having datasets share the normal data instances, which we refer to as the *base distribution*.

The base distribution is composed of k randomly positioned Gaussian clusters, of similar size. Each cluster C_i is generated from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, where $\mu_i \in \mathbb{R}^d$, $\Sigma_i \in \mathbb{R}^{d \times d}$ and d is the number of attributes. Each element of the mean vector μ_i is sampled uniformly at random from a user-specified interval. The same procedure is applied to generate each element of the covariance matrix Σ_i. To ensure cluster separation we computed Silhouette score [17] and Davies-Bouldin index [18]. All created datasets achieve a Silhouette score greater than 0.88 and a Davies-Bouldin index of 0.2 or smaller.

Outliers are then generated following a two-step process. First a set of candidate outliers is generated, then, a filtering process, based on Mahalanobis distance, selects the candidates that satisfy the criteria imposed by the specific type of outlier being generated. Candidate outlier generation and filtering procedures vary depending on the target outlier category.

Global scattered outliers are instances that do not form any inherent grouping and are far from all normal instances. To generate this type of outlier, we extract the value range of each attribute of the base distribution, enlarge it, and sample attribute values from this enlarged range, following a uniform distribution to avoid cluster creation. Then to ensure globality, candidate outliers are filtered by computing the Mahalanobis distance between a candidate and all k clusters of the base distribution. We found sufficient a threshold of 7 for the Mahalanobis distance.

The process generating the other three types of outliers is based on modifying parameters that describe the base distribution such as mean or covariance matrix, and sampling candidates from the modified distribution.

Global clustered outliers are generated by creating new Gaussian shaped clusters positioned away from the base distribution. First, we define the mean of each outlier cluster by scaling the mean matrix of the base distribution by a parameter α, effectively placing the outlier clusters to regions distant from base clusters. The covariance matrix values for each outlier cluster are randomly sampled from a range ensuring tighter outlier clusters than base clusters, hence increasing detection difficulty. Candidates are then sampled from these outlier distributions and filtered using the same Mahalanobis distance criterion applied to global scattered outliers.

To create local clustered outliers we follow a similar procedure but with a locality constraint. Instead of placing outlier clusters far from all base clusters, the filtering process ensures they are positioned near a single target base cluster, and further from any other cluster.

The distribution generating local scattered outliers is obtained scaling the covariance

matrix ($n\_attributes, n\_attributes$) of a target base cluster by a user-specified parameter β. The effect of this operation is a widening of the Gaussian distributions that produced the clusters of the base distribution. Candidate outliers are then sampled from these wider distributions. However, since the outlier distributions overlap with the base clusters, a two-step filtering process is necessary to remove candidates inside a base cluster.

First, candidates that fall too close to their target base cluster, below a minimum Mahalanobis distance threshold, are removed. Then, to ensure locality, we keep only candidates closer to their target cluster than to any other base cluster.

Parameters such as dataset size and number of outliers were selected to be comparable with those of real-world datasets. Table 4.2 summarizes the parameters describing each artificial dataset, while Table 4.3 illustrates Isolation Forest's performance.

Table 4.2: Artificial Datasets

Dataset	**Size**	**#Base Clusters**	**#Outlier Clusters**	**#Outliers**	**%Outliers**
Global_Scattered2D	21000	3	0	1000	4.76
Global_Clustered2D	21000	3	2	1000	4.76
Local_Scattered2D	21000	3	0	1000	4.76
Local_Clustered2D	21000	3	2	1000	4.76
Global_Scattered4D	21000	3	0	1000	4.76
Global_Clustered4D	21000	3	2	1000	4.76
Local_Scattered4D	21000	3	0	1000	4.76
Local_Clustered4D	21000	3	2	1000	4.76
Global_Scattered10D	31500	10	0	1500	4.76
Global_Clustered10D	31500	10	10	1500	4.76
Local_Scattered10D	31500	10	0	1500	4.76
Local_Clustered10D	31500	10	10	1500	4.76

Table 4.3: Isolation Forest Performance on Artificial Datasets

Dataset	**AUC ROC**	**AUC PR**
Global_Scattered2D	0.99	0.99
Global_Clustered2D	0.99	0.99
Local_Scattered2D	0.95	0.75
Local_Clustered2D	0.94	0.60
Global_Scattered4D	0.99	0.99
Global_Clustered4D	0.99	0.99
Local_Scattered4D	0.95	0.66
Local_Clustered4D	0.95	0.58
Global_Scattered10D	0.99	0.99
Global_Clustered10D	0.99	0.99
Local_Scattered10D	0.92	0.48
Local_Clustered10D	0.93	0.36

4.3 Experiment Description and Objectives

The experimental analysis is composed of four tests.

In *Test 1* we compute the Jaccard similarity between the sets of outliers detected by Isolation Forest and KCenterOUT to quantify how much the two algorithms agree on which instances constitute anomalous behavior. Parameters k and z of KCenterOUT are set to ground truth values for artificial datasets.

For real-world datasets the number of centers is empirically determined by running KCenterOUT with $k \in [2, \ldots, 10]$, plotting the objective function and selecting the elbow point where further increases in k yield negligible improvements. Parameter z is known [16].

Isolation Forest parameters are set to default values as specified in [4]. To ensure comparable results, we extract the top z instances ranked by outlier score, matching the number of outliers identified by KCenterOUT.

Test 2 determines the similarity between the outlier rankings produced by Isolation Forest and KCenterOUT across the entire dataset, by computing the Spearman's rank correlation coefficient. Algorithm parameters are configured as in *Test 1*.

The purpose of this first set of experiments is to understand the relationship between the concept of outlier used by Isolation Forest and KCenterOUT, by comparing the rankings and outlier sets of the two algorithms.

The second set of experiments has the additional purpose of determining whether the proposed methodology can be used to efficiently solve the k-center with z outliers problem.

The proposed method consists of using Isolation Forest to detect and subsequently remove the z instances with the highest outlier score. Then Farthest First Traversal is applied to the cleaned dataset, returning a set S of k centers.

The procedure to compute the values of objective function of the proposed method and KCenterOUT, respectively $Objective_{PM}$ and $Objective_{KCenterOUT}$ is the following. Given a set S of k centers, for each instance of the full dataset we compute the distance to its closest center. Then we sort each distance in ascending order so that: $d_1 \leq d_2 \leq \ldots \leq d_n$, where n is the dataset size. Finally we report as value of objective function the $(z+1)-th$ largest distance, effectively ignoring the z largest distances are required by the k center with z outliers problem.

Test 3 compares the objective functions of the proposed methodology and KCenterOUT, by computing the ratio: $\frac{Objective_{PM}}{Objective_{KCenterOUT}}$, as we vary the number of outliers to report z, while parameter k is constant and set to the true value. Values of z range from 15% to 1% of dataset size.

In *Test 4*, we again compare the objective functions of the two methods, but vary the number of centers k, while z is set to the ground truth value. For the 10-dimensional artificial datasets we used $k \in [5, \ldots, 15]$, while for all other datasets $k \in [2, \ldots, 10]$. The use of different intervals accounts for the different clustering structure of the datasets.

The idea behind these tests is that if the proposed method can be used for solving the k-center with z outliers problem, then objective functions should be similar even when k or z deviate from the ground truth values.

The basis for our method is on the assumption that Isolation Forest and KCenterOUT use a similar notion of outlier, given that the strategy they employ to solve their respective problems is influenced by the concept of density.

If results conferm our assumption then the proposed method could be used to solve the k-center with z outlier problem much more efficiently than KCenterOUT. Furthermore, since z is not a parameter of the proposed method, once Isolation Forest returns all outlier scores, solutions for the k-center with z outliers problem can be found for any value of z by giving in input to Farthest First Traversal the dataset cleaned of the z most outlying instances. For KCenterOUT instead, the outlier scoring process must be executed for each value of z.

4.4 Analysis of the Results

The following comprises a discussion of the results obtained in each test.

4.4.1 Test 1

Below, Table 4.4 illustrates the results of *Test 1*. Artificial datasets show a clear pattern. For global outliers Isolation Forest and KCenterOUT mostly agree on which instances constitute anomalous behavior, while for local outliers, we observe low similarity of the returned outlier sets. Interestingly, as dataset dimensionality increases, this pattern becomes more pronounced.

A possible explanation of the low agreement on local outliers can be found by observing that Isolation Forest performs poorly on these types of outliers [4], and also recalling that normal clusters of the artificial datasets follow a Gaussian distribution, which favors KCenterOUT. Gaussian clusters have their highest density at the center and decrease radially outward. KCenterOUT selects as centers the instances with the densest neighborhood, where the neighborhood is defined as a ball of a certain radius. Hence the particular geometry of Gaussian clusters aligns with the center selection strategy of the algorithm. Enabling KCenterOUT to correctly identify the true local outliers that are positioned at the cluster boundaries, while Isolation Forest fails to detect them effectively.

For what concerns real-world datasets, we observe similar results as those for artificial datasets with local outliers. Of course this doesn't mean that outliers of real-world datasets are mostly local, in fact Isolation Forest's detection accuracy for the *Shuttle* dataset is almost optimal (Table 4.1) and yet it returns a fairly different set of outliers than KCenterOUT.

Table 4.4: Jaccard Similarity and Spearman's Rank Correlation

Dataset	Jaccard Similarity	Rank Correlation
Annthyroid	0.51	0.50
Thyroid	0.26	0.54
Shuttle	0.26	0.49
Global_Scattered2D	0.93	0.72
Global_Clustered2D	0.76	0.85
Local_Scattered2D	0.49	0.89
Local_Clustered2D	0.49	0.92
Global_Scattered4D	0.99	0.76
Global_Clustered4D	0.99	0.86
Local_Scattered4D	0.37	0.85
Local_Clustered4D	0.41	0.78
Global_Scattered10D	1.0	0.31
Global_Clustered10D	0.97	0.45
Local_Scattered10D	0.21	0.48
Local_Clustered10D	0.18	0.43

4.4.2 Test 2

Results are shown in Table 4.4. We observe high rank correlation when also Jaccard similarity is high, suggesting that when the two algorithms agree on which instances to report as outliers, they also rank similarly all other instances of the dataset.

For real-world and part of the 10-dimensional artificial datasets, rank correlation is at best moderate. In this case, Isolation Forest and KCenterOUT rank similarly the most normal instances while ranking dissimilarly all other instances.

Interestingly for two-dimensional and 4-dimensional artificial datasets with local outliers we observe high rank correlation. This can be explained by the fact that outliers constitute only 4.76% of the dataset. The algorithms agree on approximately 40% to 50% of the outliers and rank the remaining outliers differently, hence the high rank correlation is primarily driven by their similar ranking of most normal instances, which comprise the vast majority of the data.

4.4.3 Test 3

For artificial datasets we observed the same patterns regardless of dataset dimensionality. As illustrated in Figure 4.1(a), when global outliers are present in the dataset, the ratio between the objective functions of the proposed methodology and KCenterOUT remains approximately constant at a value of 1.6, as long as $z > z^*$, where z is value used, while z^* is the true number of outliers present in the dataset.

When $z < z^*$ instead, the two objective functions start to converge and the ratio between the two is very close to 1. These results can be explained by examining the centers selected by the two methods.

When we overestimate the number of outliers present in the dataset ($z > z^*$), no outlier remains undetected by Isolation Forest and Farthest First Traversal cannot select an outlier center. Under these conditions also KCenterOUT does not select an anomalous center. Farthest First Traversal selects the first center by sampling an instance uniformly

at random, and the next centers as the most distant instances from the previously selected ones. Ideally this strategy would tend to select as the first center an instance close to the dense centers of the normal clusters, however this is not guaranteed, leading to a slightly worse solution than KCenterOUT whose strategy also favored by the structure of normal clusters, is more sofisticated and completely deterministic.

When we underestimate the number of outliers instead ($z < z^*$), an increasing number of outliers is not detected by Isolation Forest and Farthest First Traversal can and does select an outlier center, but so does KCenterOUT. Outlier centers worsen both objective functions, but affect KCenterOUT more severely, causing the two objective functions to converge.

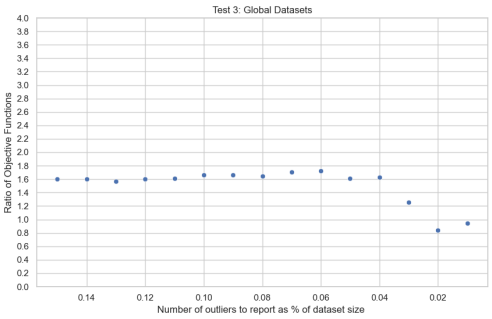
For artificial datasets with local outliers, shown in Figure 4.1(b), we observe a similar pattern as long as $z > z^*$. For $z < z^*$ instead, we observe a different pattern. The values of the two objective functions do not converge, they remain constant at a ratio of roughly 1.5 , regardless of z used.

In examining the nature of the centers selected by the two methods, we observe that KCenterOUT does not select any outlier centers, regardless of z used. In the case of the proposed method instead, while Isolation Forest misses many outliers (for any value of z) Farthest First Traversal selects an outlier center only for $z << z^*$, but the objective function does not increase dramatically. This can be explained by the fact that local outliers are close to their target normal cluster, hence even if Farthest First Traversal chooses an outlier center, the objective function shows marginal increase.

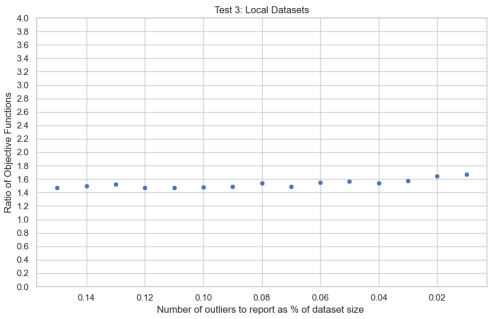
For *Thyroid*, we have a roughly constant ratio of 1.2. By examining the selected centers, we find that regardless of z, the two methods select the same number of outlier centers and the two objective functions are always similar.

For *Annthyroid*, with $z > z^*$ the ratio starts at 1.6 and decreases to 1.2 as z approaches z^*. When $z < z^*$ instead, the ratio jumps at 1.6 again and also decreases to 1.2 as we decrease z.

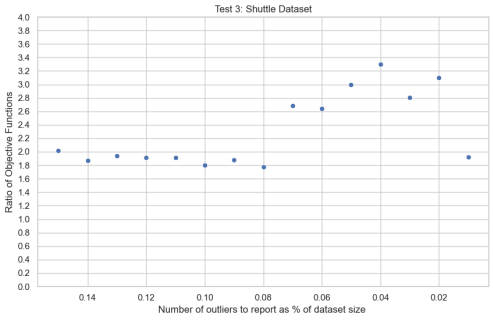
The *Shuttle* dataset, shown in Figure 4.1(c), exibiths the most interesting pattern. Surprisingly, for $z < z^*$, even if some outliers are not detected by Isolation Forest, Farthest First Traversal does not select any outlier center, while KCenter out selects two outlier centers out of 5 available. For $z > z^*$ outliers with an increasing outlier score are not detected by Isolation Forest leading Farthest First Travesal to select an outlier center, this dramatically increases the objective function and the ratio increases accordingly. KCenterOUT always selects two outlier centers, except for the last value of z where it selects only one, increasing its objective function and bringing it closer to that of the proposed method.



(a) Global datasets



(b) Local datasets



(c) Shuttle dataset

Figure 4.1: Results of Test 3. The x-axis represents parameter z, while the y-axis shows the ratio between the objective functions of the two methods

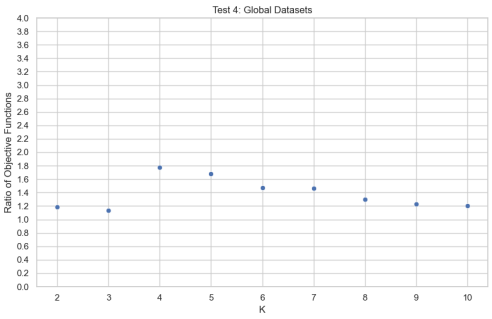
4.4.4 Test 4

In *Test 4*, we observed three distinct patterns, illustrated in Figure 4.2.

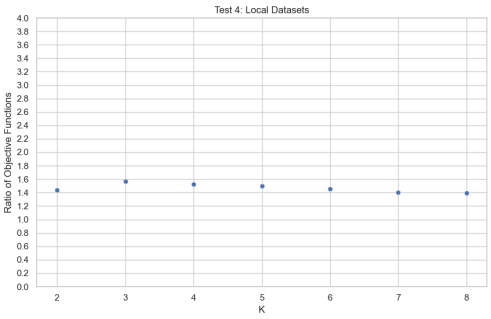
For artificial datasets with global outliers, in Figure 4.2(a), regardless of dimensionality, the ratio between objective functions is close to 1 until $k < k^*$, then, with $k = k^*$, the ratio jumps to values close to 2, mainly because the objective of KCenterOUT decreases sharply. As we increase k the ratio slowly descreases until it reaches values close to the case $k < k^*$. In this case, it's the objective of the proposed methodology that decreases faster than that of KCenterOUT.

For datasets with local outliers, shown in Figure 4.2(b), the ratio remains almost constant regardless of the value of k. *Thyroid* and *Annthyroid* exibith the same pattern. By analyzing the centers selected by the two algorithms we find that, for these datasets, whenever Farthest First Traversal chooses an outlier center, so does KCenterOUT and the objective functions change by similar amounts, keeping the ratio approximately constant throughout the test.

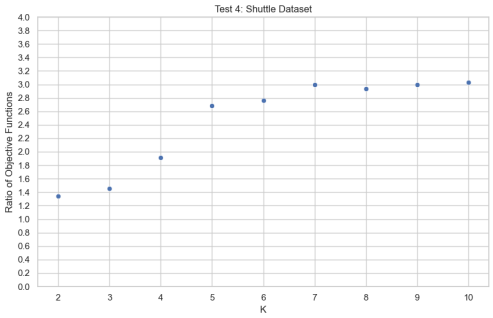
For the *Shuttle* dataset, in Figure 4.2(c), the ratio exhibits a dramatic increase when $k < k^*$, whereas for $k > k^*$, the ratio remains approximately constant. Examining the centers selected by the respective algorithms we find that Isolation Forests fails to identify approximately 150 out of 3511 outliers and Farthest First Traversal consistently selects one outlier as a center regardless of value of k. KCenterOUT demonstrates similar



(a) Global datasets



(b) Local datasets



(c) Shuttle dataset

Figure 4.2: Results of Test 4. The x-axis represents parameter k, while the y-axis shows the ratio between the objective functions of the two methods

behavior, selecting at least one outlier center, and for $k > 3$, it selects two outlier centers. When $k = k^*$, the objective function of KCenterOUT exhibits a sharp decline, followed by a gradual decrease as k increases. Conversely, the objective function of the proposed method is relatively stable, showing minimal variation regardless of the value of k.

Table 4.5: Time Comparison between Proposed Method and KCenterOUT

Dataset	**Time Proposed Method(ms)**	**Time KCenterOUT(ms)**
Annthyroid	48	1924
Thyroid	30	365
Shuttle	306	204516
Global_Scattered2D	129	20207
Global_Clustered2D	173	24210
Local_Scattered2D	187	23064
Local_Clustered2D	160	21205
Global_Scattered4D	114	13895
Global_Clustered4D	173	20800
Local_Scattered4D	188	14994
Local_Clustered4D	189	18201
Global_Scattered10D	181	43784
Global_Clustered10D	263	116436
Local_Scattered10D	263	77810
Local_Clustered10D	261	106245

The results obtained from *Test 3* and *Test 4* indicate that KCenterOUT consistently outperforms the proposed method in terms of solution quality, irrespective of the parameter settings. However, the proposed methodology exhibits substantial computational efficiency advantages, in some cases it is three orders of magnitude faster than KCenterOUT. A performance comparison is provided in Table 4.5, above.

Chapter 5

Conclusions

In this thesis, we studied the similarity in terms of notion of outlier of Isolation Forest and KCenterOUT. The latter is a state-of-the-art k-center with z outliers clustering algorithm which searches for centers in the densest regions of the feature space, thus assuming that outliers lie in sparse regions. Similarly, Isolation Forest is an outlier detection algorithm that assumes outliers to be easily isolatable by partitioning the feature space with axis-parallel cuts, hence inherently assuming they are located in low-density areas.

The density-informed strategies both algorithms employ inspired our interest in their comparison, while the superior computational efficiency of Isolation Forest compared to the more than quadratic complexity of KCenterOUT, led us to devise a method using this outlier detection algorithm as a preprocessing step to efficiently solve the k-center with z outliers problem.

The proposed method consists of a first stage where Isolation Forest is used to detect outliers followed Farthest First Traversal to solve the k-center problem on the cleaned dataset. The nearly linear computational complexity of both algorithms contributes to the overall efficiency of the approach.

The empirical evidence suggests that, when global outliers are present in the data, Isolation Forest and KCenterOUT rank all instances similarly. In contrast, they rank local outliers significantly differently, which can be explained by the poor detection accuracy of Isolation Forest for this type of outlier.

Our analysis revealed that, even when the proposed method and KCenterOUT return relatively different sets of outliers, the values of objective function for the k-center with z outliers problem, computed using the solutions returned by the two methods, do not differ substantially. This suggests that even when the specific outlier instances returned differ, they are of similar outlying degree.

Furthermore we found that the difference in value of objective function was influenced by the proximity relationship between outliers and normal instances and was independent of the distributional characteristics of outliers, such as whether they form clusters or are scattered in the feature space.

Althought KCenterOUT consistently returnes a better solution, the substancial effi-

ciency gains of the proposed method justify its adoption in favor of KCenterOUT. The nearly linear computational complexity of the proposed method makes it at least one order of magnitude faster, with efficiency gains increasing with dataset size.

The proposed methodology has the further advantage that once the outlier scores of the data instances have been determined with Isolation Forest, solutions of k-center with z outliers can be found for several values of z with a computational complexity independent of z. This is a very important feature considering that, in many practical settings, the actual number of outliers is unknown and probing different values of z might be useful.

5.1 Future Work

In this thesis, we employed the original Isolation Forest algorithm developed by [4]. The success of this algorithm has led research to design new algorithms that improve on the flaws of the original algorithm, such as its low detection accuracy for local outliers.

One promising direction would be to revise our method by substituting Isolation Forest with one of the algorithms it heavily inspired, such as [14].

Another insteresting direction involves investigating whether it is possible to find theoretical bounds on the approximation factor of the proposed method for the k-center with z outliers problem.

Furthermore our findings could potentially be extended to distributed settings given that Isolation Forest is easily parallelizable and that extensive research has been conducted on the k-center with z outliers problem in this setting.

In particular, it would be interesting to compare our proposed method with the distributed coreset-based strategy proposed in [19] which is more efficient yet less accurate than KCenterOUT.

Bibliography

[1] D. M. Hawkins, *Identification of Outliers*. Monographs on Statistics and Applied Probability, Dordrecht: Springer, 1980.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, pp. 1–58, July 2009.

[3] C. C. Aggarwal, *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd ed., 2016.

[4] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, Mar. 2012.

[5] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[6] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan, "Algorithms for facility location problems with outliers," in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, (USA), p. 642–651, Society for Industrial and Applied Mathematics, 2001.

[7] Z. Li, Y. Zhu, and M. Van Leeuwen, "A survey on explainable anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 18, Sept. 2023.

[8] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier detection: Methods, models, and classification," *ACM Comput. Surv.*, vol. 53, June 2020.

[9] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "On detecting clustered anomalies using sciforest," in *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2010* (J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, eds.), vol. 6322 of *Lecture Notes in Computer Science*, (Berlin, Heidelberg), pp. 274–289, Springer, 2010.

[10] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *The VLDB Journal*, vol. 8, pp. 237–253, feb 2000.

[11] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, (New York, NY, USA), p. 93–104, Association for Computing Machinery, 2000.

[12] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recogn. Lett.*, vol. 24, p. 1641–1650, June 2003.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[14] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, p. 1479–1489, Apr. 2021.

[15] D. E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley Professional, 2nd ed., Apr. 1998.

[16] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," 2022.

[17] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[18] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[19] M. Ceccarello, A. Pietracaprina, and G. Pucci, "Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially," *Proc. VLDB Endow.*, vol. 12, p. 766–778, Mar. 2019.