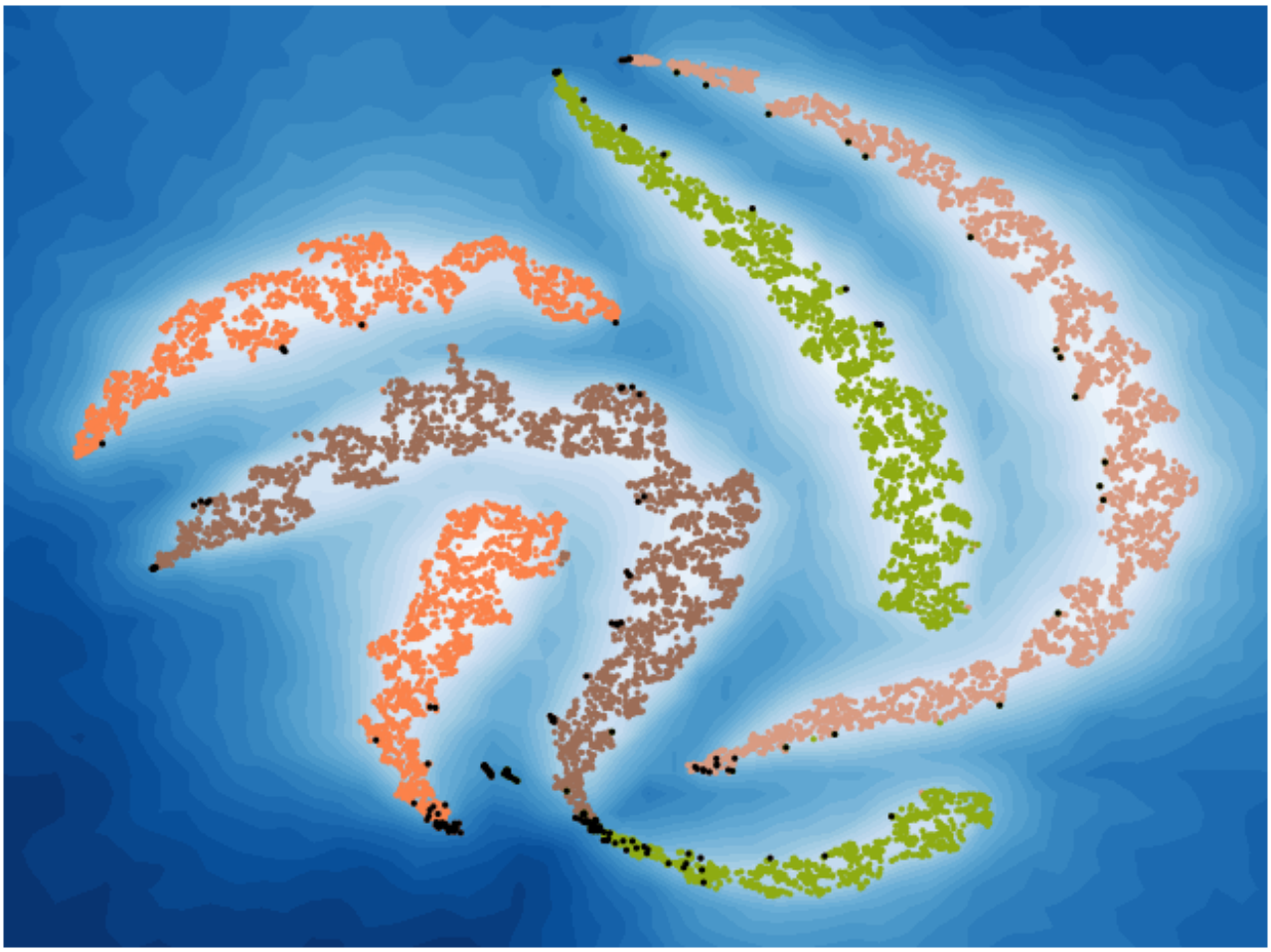
Multiclass Outlier Detection and Visualization Based on Isolation Forest



Master Thesis
16.11.2022

by Xianxiao Xu, 19-763-606

Supervisors:
Prof. Dr. Renato Pajarola
Haiyan Yang

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

University of Zurich[UZH]

VISUALIZATIONANDMULTIMEDIALAB

Abstract

Isolation forest is a popular anomaly detector due to its model-free structure and validity in detecting outliers across different types of datasets. This thesis presents solutions for two main issues in terms of isolation-based outlier detection algorithms. First, there is an adjustment to the isolation-based outlier detection model to improve the detection accuracy of Isolation Forest (iForest) and Extended Isolation Forest (EIF). The EIF is an extension of iForest, which addresses the block artifacts issue of iForest. Motivated by the failures of outlier detection on some real-world benchmark datasets by EIF, an adjusted EIF regarding the problem arising from the randomness of split hyperplane is presented. The outlier detection accuracy and precision of the adjusted EIF show that it is capable of enhancing the performance of both iForest and EIF. However, the drawback of the adjustment is that it is not time efficient. Second, this thesis proposes methods to generate a credible image presentation with outliers scattered in the relative areas based on isolation-based detection for multi-variate datasets. Inspired by the fact that neither iForest nor EIF can detect local outliers for each class in multi-class datasets, and few related works have been done in this direction, several class-wise detectors based on EIF and adjusted EIF are proposed in this thesis. By comparing the graphs, one of the methods achieves the best performance in providing insights for identifying potential outliers in clustering datasets.

Contents

1 Introduction

Outlier detection refers to finding patterns in data that do not match the expected behavior [CBK09]. With the digitization of society, outlier detection finds various applications in different domains, such as diagnosis, fraud detection, computer vision, sensor network, data cleaning, cybersecurity, etc. [WBH19]. Some researches focus on finding and removing the outliers from the datasets, such as when doing data preprocessing for machine learning, while others are interested in identifying the outliers and abstracting insights from them. Therefore, algorithms for outlier detection, also known as anomaly detection, have been widely studied. Basically, different outlier detection algorithms are designed to improve the accuracy and speed of identifying the outliers in different types of datasets. Among them, according to the documentation of PyOD [ZNL19], one of the most comprehensive Python libraries for outlier detection, Isolation Forest (iForest) [LTZ12], which is renowned for its model-free structure, is believed to be both effective and efficient on various types of data [HKB19].

1.1 Motivation

Drilled from the assumption that outliers are those rare and distinct data points in a dataset, iForest applies an *isolation* mechanism upon sub-sampled datasets to generate anomaly scores (possibility of being an anomaly) and find outliers based on the scores without calculating and building complex mathematical models. Explicitly speaking, first, the algorithm builds a binary search tree-like structure *iTree* by splitting a random sub-sample of the dataset based on a random split value on a randomly selected dimension each time iteratively till all the data points are isolated; second, assembles a certain number of iTrees with different sub-samples into a forest ;third, passes each test data point through the forest to generate the anomaly score by calculating the average depth from the root to the leaf node it reaches on each iTree in the forest. Unlike many other proposed algorithms, such as Proximity-Based and Neural Network-Based algorithms, iForest is computationally efficient with only linear time complexity but shows relatively stable effectiveness on different datasets. Therefore, iForest is suitable for large datasets.

However, iForest is subject to several limitations. First of all, because in iForest, the split is only on one dimension each time, it introduces block artifacts in the axis-parallel directions of the clusters. It assigns falsely low anomaly scores for them, which significantly reduces the detection accuracy. To address and remedy these problems, Extended Isolation Forest (EIF) [HKB19] was introduced. It shares a similar basic concept as the iForest, which is to split the data space until isolating each data point, but with a different implementation strategy to eliminate the mentioned issues. While each time iForest picks a random value in a random feature, which means doing an axis-parallel split, EIF rules the split hyperplane by randomly selecting a slope and a random intercept, so that the split will be non-axis-parallel. It has proved to be able to remove the "ghost" artifacts generated by iForest on 2-D synthetic data. In addition, another shortcoming of iForest is that it could only select limit features for splitting the high dimensional datasets, making it challenging to cover sufficient relevant dimensions to identify multifeatured anomalies [BTA+18]. For this problem, EIF allows the split hyperplane to intersect with different numbers of axes. Thus, each partition will correspondingly retain more dimensions.

Although EIF fixes the problems of artifacts and dimension loss, it is still unable to detect correct outliers in some real-world datasets, especially those high-dimensional datasets that do not have d-variate distribution (multi-variate normal distribution) properties. Due to the randomness of generating a split hyperplane, the EIF faces the problem that it might fail to split the data points in some iterations, and this risk is increased with the dimension of dataset growth [LBST21].

Furthermore, neither iForest nor EIF extends the method to identify and visualize outliers for multi-class data. Investigation shows that isolation-based algorithms are weak at detecting the local anomalies for each class, primarily those anomalies that lie in between the clusters [BTA+18]. Because it might vary if applying the

algorithms separately on each class, this thesis dedicates to generating a reasonable decision boundary of anomaly scores and intuitive visualization of the outliers for multi-variate datasets by applying isolation-based algorithms on each variate. Moreover, for high-dimensional data that do not present d-variate distribution, as EIF implements purely random split, it would be possible that the algorithm sometimes fails to split the dataset, making the anomaly scores biased.

1.2 Description of Work

Based on the aforementioned issues, this thesis proposes several solutions. Firstly, it adjusts the EIF on the generation of cutting hyperplane and the calculation of anomaly scores to improve detection accuracy on benchmark multidimensional datasets from Outlier Detection DataSets (ODDS) [Ray16]. The experiments show that the former of the adjustments significantly improves detection accuracy over some real-world high-dimensional datasets. Secondly, it tackles the detection of outliers and visualization of decision boundaries for multi-variate datasets. After calculating the anomaly scores for each variate, a score map depicts the distribution of aggregated anomaly scores for the multi-variate datasets, over which superimposed a scatter plot with local outliers indicated.

1.3 Thesis outlines

This thesis is outlined as follows. Chapter 1 introduces the background of the thesis and describes the contribution and work of the thesis. Chapter 2 summarizes the related work in the domain of anomaly detection, with a focus on isolation-based algorithms. In chapter 3, the two research problems of this thesis are stated and investigated in detail. Moreover, the next chapter poses solutions for each problem mentioned. Different solutions for each research goal are then implemented, tested, and compared in chapter 5 across both synthetic and real-world datasets. To the end, the last chapter briefly discusses the experiment results and makes a short conclusion.

2 Related Work

2.1 Anomaly detection algorithms

In the past decades, outlier detection algorithms that consider efficiency, accuracy, and the dimension of data based on different data characteristics such as *distance*, *density*, *cluster*,*statistical-related* have been widely researched [XPWW22]. As one of the most classical outlier detection techniques, Statistical approaches apply a statistical model. For instance, Iglewicz et al. utilized a robust measure of Median Absolute Deviation (MAD)[IH93] to identify the outliers instead of the standard deviation. Since the diversity and complexity of datasets increase explosively, the statistical methods are insufficient to detect anomalies due to limited assumptions on data distribution. Cluster-based methods, for example, Ordering Points To Identify the Clustering Structure (OPTICS) [ABKS99] relies on data clustering to exclude the outliers from clusters of normal points. However, most cluster-based methods require setting specific parameters learned from data distributions, and only provide a binary decision of whether a data point belongs to the cluster. For example, K-Means [Mac67] works as an anomaly detector only with a predefined number of clusters. Therefore, they can provide only a limited explanation for the outliers [BTA$^+$18]. Density-based and distance-based methods, such as Local Outlier Factor (LOF) [BKNS00], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [EKS$^+$96] and k-Nearest Neighbors (kNN) [AP02], assume that anomalies are points that lie in the area that is less dense or far from non-anomalies. These algorithms consider either the global or relative density of the data and compare the value with other data points, and overcome the limitation of setting data-related parameters manually. Nevertheless, the cost, especially the time complexity, is susceptible to the data size or number of neighbors, which means they are not scalable for large datasets. In recent years, deep learning-based outlier detection methods [RVG$^+$18, NM19, ZRF$^+$18] are also developed to better detect different types of anomalies on high-dimensional and complex sets. However, the effort for training can exponentially increase because of the neural network structures.

2.2 Isolation-based algorithms

On the other hand, the isolation-based approaches identify the anomalies by measuring the level of difficulty of isolating a point based on the assumption that outliers are few and distinct. iForest first iteratively partitions the subsets drawn from the dataset. Then a binary search tree can simulate the partition by sending samples to the left or right branches. The shorter path a point traverses on those trees, the more probable it is an anomaly. Though iForest outperforms both in speed and effectiveness for detecting outliers in some real-world benchmark datasets [ZNL19], it still has drawbacks. For instance, iForest has a bias on separating the space because of the axis-parallel partition [HKB19] strategy, and it lacks a theoretical explanation for the anomaly score assessment that uses the mean of all the path length [AFHR21, BHM20]. Therefore, several methods are proposed to remedy and improve the performance of iForest in different aspects.

The first aspect is to change how the split value and direction of the hyperplane are chosen when partitioning the data space. Rather than randomly select a cutting value in a random dimension, EIF [HKB19] shares a similar basic idea but defines the cutting hyperplane by random normal vector and intercepts. Generalized Isolation Forest (GIF) [LBST21] improves EIF by randomly selecting the split values on data projected on the randomly chosen slope to avoid an empty branch cut. PIDForest [GSW19] implements *PIDScore*, a new isolation-based method, for feature extraction and split points picking based on sparsity [XPWW22]. Probabilistic Generalization of Isolation Forest (PGIF) [TK22] introduces a new probability, enabling more effective partitions between clusters. These algorithms contribute to tackling the biased split issue introduced by iForest or EIF. However, most of them still suffer from algorithmic bias, and are unable to handle complicated datasets. For instance, PIDForest still has

the block artifacts issue similar to iForest though it generally achieves better performance on outlier detection. Hence, how to decide effective split hyperplanes to isolate data points is still a researchable problem.

The second aspect is to change the split criteria. For example, SCiF [LTZ10] proposes a new criterion for branch cut to construct a split hyperplane.

The third aspect is to change the definition of anomaly score. Instead of calculating the average paths that each point passes through the trees, Buschjäger et al. modify the anomaly scores of iForest and EIF by introducing an estimate of mixture coefficients of mixture distribution as the scoring method [BHM20, AFHR21].

The last aspect is to combine the distance-based algorithms with the isolation progress. For instance, Pang et al. instruct an ensemble method that uses the least similar nearest neighbors with linear time complexity [PTA15]. Since iForest is weak at detecting local outliers, which are close to normal points than the global outliers, Isolation-based anomaly detection using Nearest-Neighbor Ensembles (iNNE), however, is capable of isolating a point by building a hypersphere with a radius determined by nearest neighbors [BTA$^+$18].

2.3 isolation-based algorithm for multi-class datasets

Few studies focus on visualizing outliers for multi-variate datasets based on isolation-based algorithms. Fuzzy C-Mean-based Isolation Forest [KKPC21] enhances the iForest by applying *membership grades* of elements, indicating how possible a given instance belongs to a group of similar elements. This is a combination of the clustering algorithm and iForest. By weighting each anomaly score with membership grades, Fuzzy C-Mean-Based Isolation Forest is able to detect outliers more accurately than iForest. However, it does not examine and visualize the decision boundaries of outliers in multi-class clustering datasets, and there is no clue for how it performs on these data. Melquiades et al. propose a semi-supervised model based on Hybrid Isolation Forest to aggregate known anomalies in multiple classes [MdLN22]. The accuracy of detecting outliers improves in several multi-class datasets. However, it requires human effort to label and aggregate the data for training models as a semi-supervised technique. Therefore, further studies on unsupervised isolation-based methods to detect as well as visualize the decision boundary of anomalies in multi-class datasets are still needed.

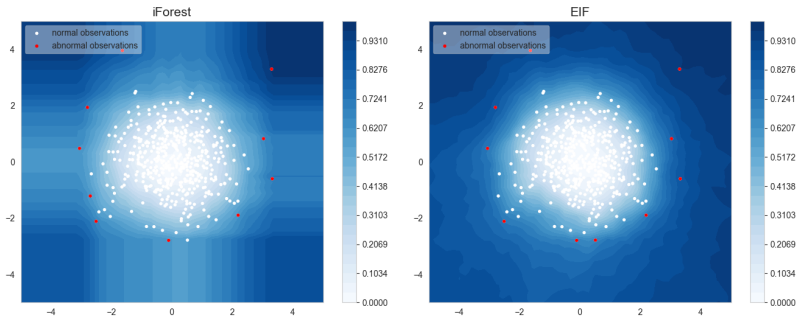
Inspired by these researches and the mentioned problems in iForest and EIF, this thesis attempts to adjust EIF on the following aspects. First, changing partitioning hyperplane decision rules by combining other algorithms. Second, altering anomaly scoring functions. Moreover, since the previous work of isolation-based algorithm and its extensions do not focus on the visualization of outliers on 2-D multi-variate datasets, this thesis also provides solutions to tackle it.

3 Problem Statement

This thesis is a combined work of improving the visualization of the isolation-based algorithm on multi-variate datasets and improving the performance of EIF on real-world datasets. This chapter explains why the improvements are needed explicitly in the following two sections.

3.1 Improvement of EIF

An isolation tree (iTree) is a binary tree structure used to isolate instances. In iForest and EIF, building an iTree represents splitting data instances iteratively so that each split can be regarded as one branch cut in the iTree. Because it is hard to show the process in a tree structure, a schematic diagram illustrating how to split samples on a 2-D plane is given in figure 3.2.



(a) iForest (b) EIF

Figure 3.1: Score map and data points visualization of iForest and EIF applying on a Gaussian-distributed blob

Figure 3.1 displays the score maps generated by iForest and EIF from a Gaussian-distributed blob. It demonstrates that EIF fixes the "ghost" region issue introduced by the coordinate-parallel split in iForest. However, because of the complete randomization of slope and intercept selection, EIF suffers from an empty branch cut problem. Figure 3.2 simulates two cuts in EIF. The black line represents the first cut, which splits the data points into two parts successfully. However, as the red line shows, the second cut fails since it actually does not split the data points due to the randomness of the cut hyperplane. At the same time, as the *height limit* of an iTree is restricted by the size of subsamples, the empty branch cut issue will make more data points stop at the same leaf nodes. Since the length that a point travels through iTrees decides its anomaly score, this problem makes more data points receive indistinguishable anomaly scores, which causes unwanted deviations in identifying outliers. Therefore, the accuracy of detection by EIF on various datasets might be affected to some extent. The solution is either to change the direction or the split value of the split hyperplane. And the effort to reduce the false positive rate on EIF has been made through this work.

3.2 Visualize outliers and plot score map on multi-class datasets

Although the global outliers, which are those scattered points that appear in a relatively low-density area, can be detected, neither iForest nor EIF has underperformance in identifying local anomalies, which are the points that show different attributes for each cluster. Figure 3.3 illustrates the visualization of both algorithms on a three-clusters dataset with equal numbers of data points in each class. The outliers are indicated with gray scatters

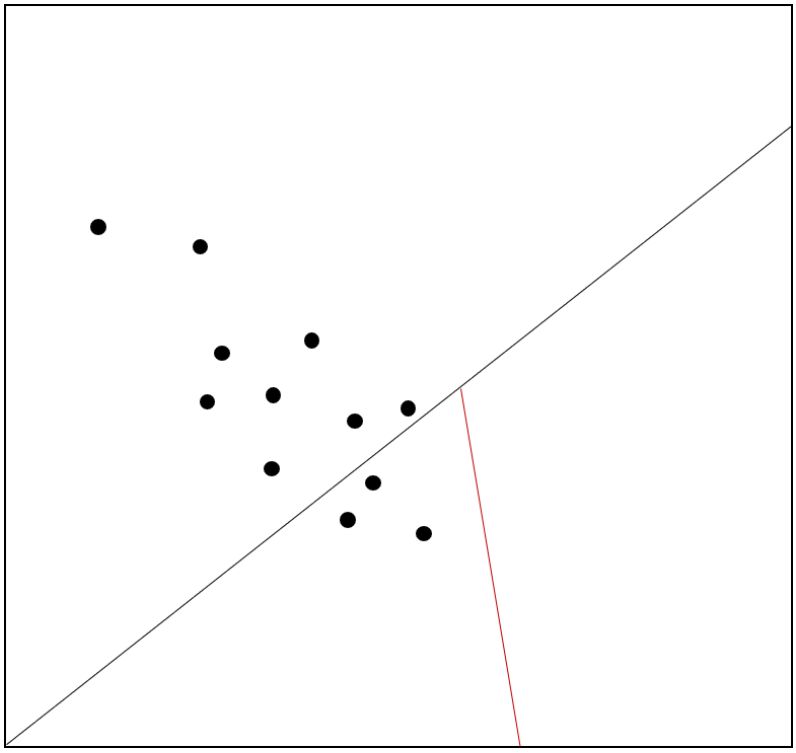
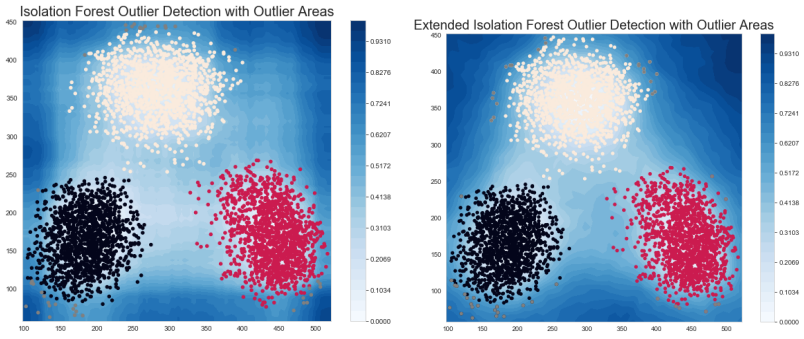


Figure 3.2: The simulation of a cut in EIF

in the figures, while others are normal points in each cluster. In observation, firstly, the scatter plot shows that both methods only classify the points around the data space rather than those far from each cluster as outliers. Secondly, the contour map indicating different degrees of anomaly scores is displayed. The darker the area, the higher the anomaly score is. It shows unreasonable results that if detecting outliers in each cluster, the detected outliers fall in the area with low anomaly scores. Therefore, how to combine the outlier visualization from cluster-wise detection with a reasonable score map showing the potential anomaly area is also an essential issue to be addressed.



(a) iForest (b) EIF

Figure 3.3: Score map and data points visualization of iForest and EIF applying on three-clusters dataset

4 Technical Solution

4.1 Overview of EIF

Generally, isolation-based algorithms [LTZ12, HKB19] have three phases to generate the final anomaly score for each data point:

Phase 1: Training an iTree

Phase 2: Ensemble iTrees into iForest

Phase 3: Evaluation and scoring data

To start, suppose there is a dataset X with M features and N instances. At phase 1, the algorithm first chooses a random subsample X^* with a size of n from the dataset X. Then a random slope for branching is selected, which is represented by constructing a M-dimension normal vector $\boldsymbol{n}$, where each coordinate of $\boldsymbol{n}$ is drawing from the standard normal distribution $\mathcal{N}(0, 1)$ [HKB19]. A parameter *extension level* l is set to randomly drop $M - l$ coordinates from the original normal vector here. By setting the extension level, the method can be more flexible in dealing with various-dimensional datasets. After that, it draws a random intercept $\boldsymbol{p}$, where each coordinate of $\boldsymbol{p}$ is drawn from a uniform distribution in the range of the subsample X^*. Once the slope and the intercept are decided, the method splits the sample X^* by a criterion as follows:

$$(\boldsymbol{x} - \boldsymbol{p}) \cdot \boldsymbol{n} \leq 0, \tag{4.1}$$

All the data points x_i that satisfy this inequality are sent to the left branch, and vise versa. This procedure splits the sample into two parts. EIF repeats the selection and branching process until one instance is *isolated* as a leaf node or the height limit $hlim$ is reached. Here, the limit of height in an iTree T_t is predefined by the depth of unsuccessful search in a Binary Search Tree (BST), which is $log_2 n$ [LTZ12]. After a number of iTrees are constructed, as in the above steps, the algorithm jumps to phase 2, which assembles the generated iTrees $T_t, t \in 1, 2, ..., K$ in an iForest $F, F = T_1, T_2, ..., T_K$. The training phase is complete till then.

In phase 3, each test data point x passes through each tree, whose direction is decided by the criteria as equation 4.1, and gains a path of length $h(x)$. By assuming the outlier is sparse and rare, the depth $h(x)$ of an outlier is expected to be significantly shorter than that of the normal data. An average path that the point x traversed down all iTrees in the iForest is then computed as $E(h(x))$. Based on this, an anomaly score for the point x is then calculated by the Equation:

$$s(x, n) = 2^{-E(h(x))/c(n)}, \tag{4.2}$$

where $c(n)$ is a normalized factor representing the mean value of depths of unsuccessful search in a size n BST, which is defined as:

$$c(n) = 2H(n - 1) - (2(n - 1)/n), \tag{4.3}$$

where n represents the size of subsample and $H(i)$ represents the harmonic number which can be estimated by Euler's constant $(ln(i) + 0.5772156649)$ [LTZ12]. Since the anomaly score $s(x, n)$ indicates the possibility of a point being an anomaly, the points with higher scores are believed to be outliers.

4.2 Isolation-based algorithm adjustment

In this thesis, several adjustments in phase 1 and phase 3 are posed, while the algorithm of phase 2 remains the same. As known, both EIF and iForest assume that outliers are more vulnerable to random *isolation* process

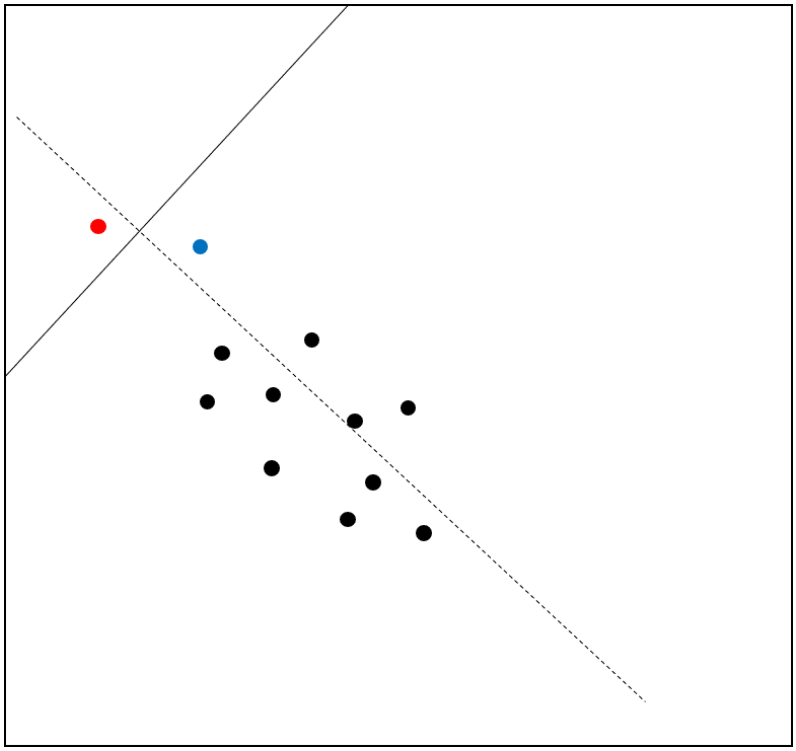


Figure 4.1: The simulation of a cut in adjusted EIF

regardless of the distribution of the datasets. However, although EIF eliminates the bias introduced by axes-parallel partition in the iForest, it sometimes fails to split the data points and has insufficient accuracy in detecting specific datasets.

4.2.1 Branching in building an iTree

Regarding the problem occurring because of randomness selection of slope and intercept, in phase 1, firstly, this work applies Principal Component Analysis (PCA) [SCSC03] to decide the normal vector of the split hyperplane. PCA is a statistical technique to extract main feature components through a linear transformation. Specifically, before each partition, selecting the first component of PCA as the normal vector and projection hyperplane, on which the data have the largest variance. This selection ensures that data points are away from each other as far as possible to be split. After that, projecting the samples on the first component and calculating the mean value of distance $d_i, i \in 1, ..., n$ between each point and their k nearest neighbors. Choose the intercept randomly in the range of the points with the highest values d_{i1} and second-highest values d_{i2}. Note that the k-nearest neighbors algorithm suffers from *curse of dimensionality* [KE11], thus, the projection process is executed before the distance calculation. This process balances the randomness and robustness to split the data points into two parts in an efficiency-performance trade-off and prioritizes a cut in a sparse region. Similar to EIF, this work continues to use extension levels to drop features randomly to keep flexibility on different datasets. Moreover, the branching criteria of EIF remain the same. That is, all the points fulfill $(\boldsymbol{x} - \boldsymbol{p}) \cdot \boldsymbol{n} \leq 0$ will be passed to the left branch, and the rest go to the other side.

Figure 4.1 visualizes the new branching process on the same simple 2-D data space as in figure 3.2. The dotted line represents the first component of PCA from the data. After projecting the data on this component, the method calculates the average distance between each point and three neighbors, finding the two with the highest distances, which are indicated by the red and blue dots. Then a random intercept $\boldsymbol{p}$ is drawn in the range of these two points, and the split is executed in the direction perpendicular to the component passing through the value.

4.2.2 Anomaly scoring

When evaluating the test points on the ensembled iForest, EIF records depths $h(x)$ that each data point passes through each iTree, and then calculates the average of the depths $E(h(x)$ for each piece of data. Different from EIF, a new normalized scoring is performed by applying the median instead of the mean to calculate the expected depth of the branches $E^{'}(h(x)$. This is inspired by the fact that the median is less affected by the extreme values, hence, catching a more representative depth for each instance.

Another tryout on the scoring is to retain the average depth of the branches $E(h(x)$, but switch the final scoring to the equation as follows:

$$s^{'}(x,n) = \frac{2}{1+e^{log_2 \frac{E(h(x))}{c(n)}}},$$

As a non-linear function, the scoring increases the anomaly score for the $E(h(x))$ that smaller than $c(n)$. Compared with the scoring in other iForest-based methods, the new one enables instances with lower depths than the depth of a tree to gain higher weight.

4.3 Visualization of cluster-wise detection

To address another issue mentioned in chapter 3, cluster-wise outlier detection and visualization for 2-D multivariate datasets, this work proposes a solution as follows. Assume there is a multi-class dataset X, for each class $c, c \in 1, ..., C$, there is $X_c \subset X$. The method first detects the outliers $O_c, O_c \subset X_c$ for each class c. Thereafter, for each pixel $p_(i,j)$ with coordinate (i,j) on the contour map, generating an anomaly score $s(p^c_{i,j})$ in each class c based on the adjusted EIF. Finally, the method provides several ways to calculate the final anomaly score for each pixel:

- Averaging C anomaly score, and normalized it by

$$s(p_{i,j}) = \frac{\sum_{c=1}^{C} s(p^c_{i,j})}{C}$$

,

- Computing the n-norm of C anomaly scores. It means computing the vector length formed by each class's scores in each pixel. Having $s(p_{i,j}, n)$ and normalizing it by

$$s(p_{i,j}) = \frac{s(p_{i,j}, n)}{C^{\frac{1}{n}}}$$

,

- Multiplying C anomaly scores, and normalized it by

$$s(p_{i,j}) = s(p_{i,j})^{\frac{1}{C}}$$

,

Finally, according to the anomaly score of each pixel, plot the contour map with data points (normal and outliers) overlapping on top of it.

5 Experiments and Results

This section presents the implementation of the algorithms proposed in chapter 4, and evaluates them on different 2-D uni- and multi-variate datasets and various high-dimensional benchmark datasets as required. Besides, measurements indicating the accuracy and precision of outlier detection in several widely-used unsupervised anomaly detection algorithms are computed to compare with the proposed method in section 5.1. In section 5.2, the presented EIF-based outlier detectors for multi-variate clustering datasets are compared with the original EIF through a score map and scatter of outliers. Further experiments concern the failure cases in section 5.2, and test two more methods. The first method combines the solutions of both two sections, and the other method avoids the deviation of anomaly scores from the previous methods. For all the situations, the experiments are conducted in Python 3.7 environment.

5.1 Isolation-based algorithm adjustment

5.1.1 Implementation

The implementation of the adjusted EIF described in subsection 4.2.1 is based on the python version of EIF [HKB19]. In the training phase, the work replaces the random selection of normal vector and intercept with the core process, as the pseudocode shows below. In detail, the PCA and Nearest Neighbor algorithms in this process utilize the decomposition and neighbors packages in Python Scikit-learn library [PVG+11].

```
# Applying the PCA on the subsample with d dimension
pca = PCA(n_components = dimension)
# Select the first component as the normal vector
normalVector = pca.component[0]
# Project the instances on the first component of PCA
projectedSample = pca.fit(Sample)
# Calculate the distances between each point and its two neighbors
distances = NearestNeighbors(n_neighbors = 3).fit(Sample)
# Select two samples with the largest mean distance
Sample[1], Sample[2] = argsort(mean(distances))[-1],argsort(mean(distances))[-2]
# Randomly draw a point in the range of the two samples
intercept = random.uniform(Sample[1], Sample[2])
```

In addition, another implementation of anomaly scoring modified as illustrated in section 4.2.2 performs in the evaluation phase. However, changing the anomaly scoring function from average to median or adapting the new normalization of anomaly score contributes little to improve the performance of EIF on either the real-world or synthetic datasets. Therefore, this chapter omits the experiments of this part and only discusses them in section 6, with results shown in Appendix. The following experiment is only based on the implementation of subsection 4.2.1.

5.1.2 real-world datasets

To evaluate the performance of adjusted EIF, in this subsection, ten real-world high-dimensional datasets from Outlier Detection DataSets (ODDS) [Ray16] are considered. Table 5.1 describes the properties, including name, sample size, number of dimensions, and the number as well as the proportion of anomalies of each real-world dataset. For comparison, as adopted partially in adjusted EIF, the results of KNN [AP02] and PCA [SCSC03]

Table 5.1: High-dimensional Benchmark Datasets

Name	Sample Size	Feature	Anomalies
Ionosphere	351	33	126 (35.8974%)
Optidigits	5216	64	150 (2.8758%)
Glass	214	9	3511 (4.2056%)
Vowels	1456	12	50 (3.4341%)
Musk	3062	166	97 (3.1679%)
Satimage-2	5803	36	71 (1.2235%)
Speech	3686	400	61 (1.6500%)
Satellite	6435	36	2036 (31.6395%)
Letter	1600	32	100 (6.2500%)
Cardio	1831	21	176 (9.6122%)

for anomaly detection are also included. Besides, another statistical-based method, HBOS [GD12], and a deep learning-based algorithm, DeepSVDD [RVG$^+$18], are considered. Therefore, the experiment can compare the performance of adjusted EIF with not only isolation-based algorithms but also with other classical density-, proximity- and neural network-based algorithms.

Table 5.2: Parameters setting of EIF

Parameters	Meaning	Value
sample_size n	The number of subsamples for each iTree	min(256, data size)
n_trees t	The number of iTree in the iForest	200
extension_level e_l	The number of coordinates kept for the normal vector	1
height_limit h_l	The maximum depth of an iTree	$log_2 n$

The area under the receiver operating characteristic curve (AUC-ROC) [Nar18] and the associated Precision/Recall (AUC-PRC) [SR15] are applied as indicators for performance evaluation. Specifically, AUC-ROC is a commonly used probability measurement to evaluate the prediction performance of machine learning binary-classifiers [SR15], where a higher AUC-ROC value usually represents a better prediction ability of an algorithm. And the AUC-PRC is an alternative indicator that evaluates the fraction of true positives among positive predictions [SR15], thus providing a more accurate measuring for imbalance binary classification missions, for exam-

Table 5.3: Comparison of AUC-ROC values for outlier detection algorithms

Name	adjusted EIF	iForest	EIF	KNN	PCA	HBOS	DeepSVDD
Ionosphere	**0.9095**	0.8569	0.8737	0.8623	0.8135	0.6674	0.5625
Optidigits	0.7483	0.6800	0.7062	0.3982	0.5166	**0.8528**	0.5786
Glass	**0.7888**	0.6982	0.7543	0.7408	0.6571	0.7718	0.5750
Vowels	0.8310	0.7528	0.7921	**0.9717**	0.5660	0.6461	0.5596
Musk	0.9995	0.9997	0.9993	0.7011	**1.000**	**1.000**	0.3500
Satimage-2	**0.9979**	0.9932	0.9955	0.9099	0.9862	0.9859	0.7141
Speech	0.4749	0.4768	0.4765	0.4732	0.4700	0.4764	**0.5207**
Satellite	0.7400	0.7117	0.7198	0.6461	0.6094	**0.7681**	0.5264
Letter	0.6606	0.5952	0.6312	**0.8671**	0.5270	0.5406	0.5416
Cardio	0.9276	0.9202	0.9240	0.7015	**0.9611**	0.8653	0.6728

Table 5.4: Comparison of AUC-PRC values for outlier detection algorithms

Name	adjusted EIF	iForest	EIF	KNN	PCA	HBOS	DeepSVDD
Ionosphere	**0.8803**	0.8108	0.8325	0.8207	0.7640	0.4395	0.4721
Optidigits	0.0553	0.0432	0.0489	0.0224	0.0275	**0.1339**	0.0316
Glass	0.0971	0.0776	0.0938	0.1097	0.1175	**0.1248**	0.0662
Vowels	0.1647	0.1253	0.1464	**0.7023**	0.1273	0.1950	0.0463
Musk	0.9915	0.9901	0.9802	0.1392	**1.0000**	**1.0000**	0.0703
Satimage-2	**0.9650**	0.9287	0.9485	0.2735	0.8438	0.7950	0.0348
Speech	0.0172	0.0171	0.0189	0.0173	0.0174	0.0163	**0.0354**
Satellite	**0.7054**	0.6650	0.6841	0.5017	0.6188	0.6984	0.3951
Letter	0.0940	0.0870	0.0867	**0.2731**	0.0879	0.0754	0.0912
Cardio	0.5338	0.5545	0.5717	0.3303	**0.6523**	0.5171	0.3095

ple, the outlier detection. Similarly, a higher AUC-PRC proves a higher capacity of a method to identify outliers. Therefore, this experiment compares both perform measurements across different algorithms.

Due to the randomness of the sampling and training process of isolation-based algorithms, the experiment iterates each algorithm ten times. It then calculates the average value for each measurement as the final result. The iForest algorithm is based on Scikit-learn library [PVG+11], and EIF is based on the source code published by Hariri and Kind (2018) [HKB19] with parameter setting as shown in the table 5.2. For those non-isolated-based detectors, the experiment is executed on the code of AD-Benchmark [HHH+22] with the best parameters tuned from 100 epochs of iterative training. Table 5.3 and table 5.4 summarize the results comparing the adjusted EIF with standard iForest, EIF, and other types of anomaly detection methods on the above real-world datasets.

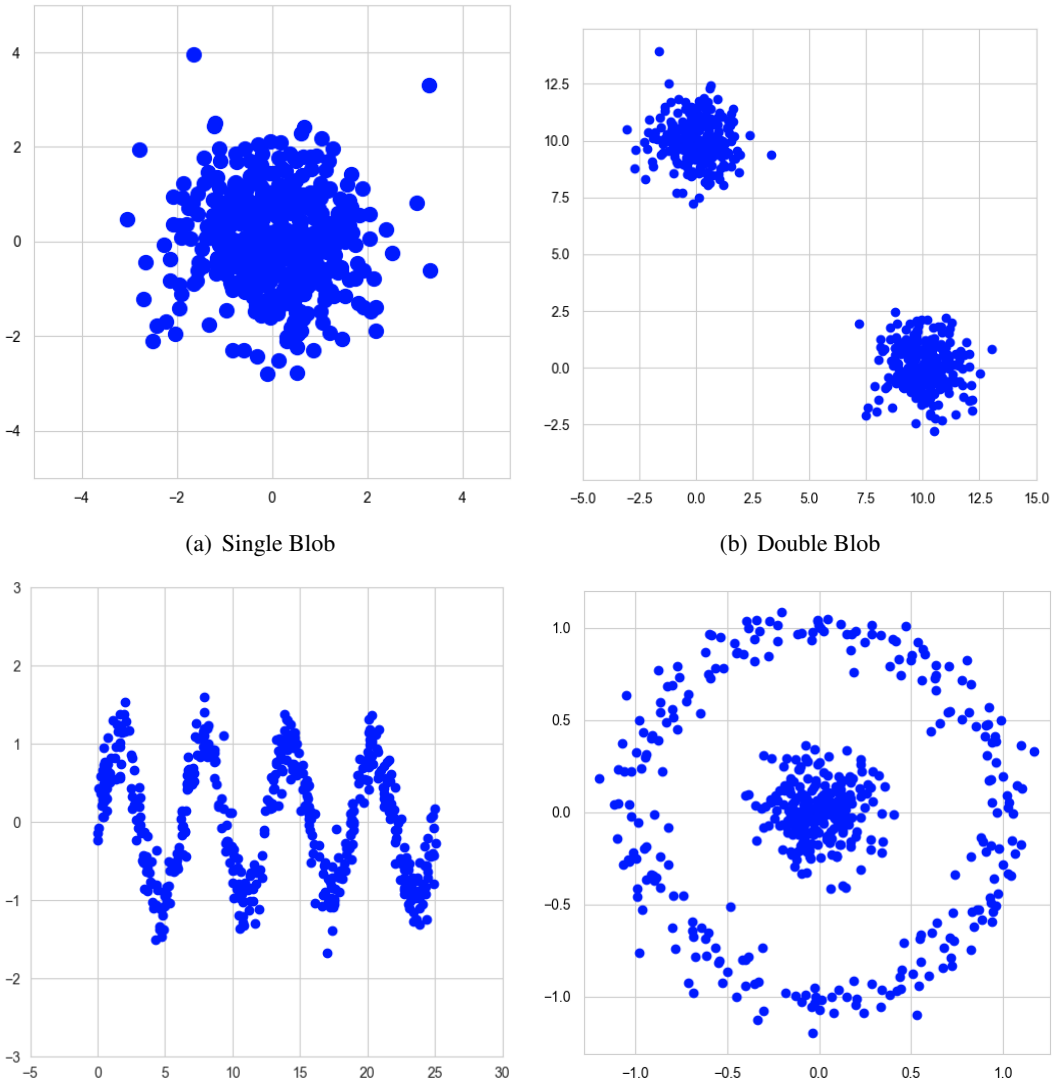
As shown in table 5.3 and table 5.4, the adjusted EIF moderately or considerably outperforms both iForest and EIF for most of the listed benchmark datasets on both AUC-ROC and AUC-PRC, except for datasets Musk, Speech, and Cardio. Adjusted EIF and EIF exhibit a slightly disadvantage on AUC-ROC than iForest on datasets Musk and Speech, while EIF outstands on the AUC-PRC value for the dataset Cardio. Compared to HBOS, isolation-based methods experience a hard time detecting some extremely high-dimensional datasets, such as Optidigits, Musk, and Speech. KNN, which uses the distance to the k-nearest neighbor as the outlier score [ZNL19], on the contrary, performs better on small datasets, for example, Glass, and Vowels. To conclude, as we can see, the adjusted EIF shows better accuracy in detecting high-dimensional real-world datasets than iForest and EIF in general. Furthermore, it is more advantageous to identify outliers in large datasets without too much dimensionality than other anomaly detection algorithms.

5.1.3 Synthetic datasets

In order to compare the difference between isolation-based algorithms intuitively, there are experiments on four 2-dimensional synthetic datasets. As depicted in figure 5.1, they are datasets with a single blob, double blobs, sinusoidal, and a circle around a blob. Each dataset is of 500 nodes following a deviation from the Gaussian distribution. The first three datasets are similar to that of the experiment conducted in [HKB19], while this experiment extends to have another circle-blob dataset to investigate the anomaly score between two nested clusters for isolation-based algorithms.

At first, the experiment detects the outliers of each dataset with each algorithm (iForest, EIF, and adjusted EIF) separately. Data points are shown on a scatter plot, where the outliers are highlighted in red. Second, for each pixel on a 50×50 mesh grid in the range of data space, it calculates the anomaly score and shows them by a contour map.

A comparison of the detection performance across iForest, EIF, and adjusted EIF for the first dataset is shown in figure 5.2. Intuitively, all methods are able to detect the outliers located around the dense cluster. However, in figure 5.2(a), a considerable cross artifact centered on the cluster is produced on the score map, with a relatively low anomaly score. It is counter-intuitive for a blob-shape dataset to have such a score distribution. As mentioned



(a) Single Blob

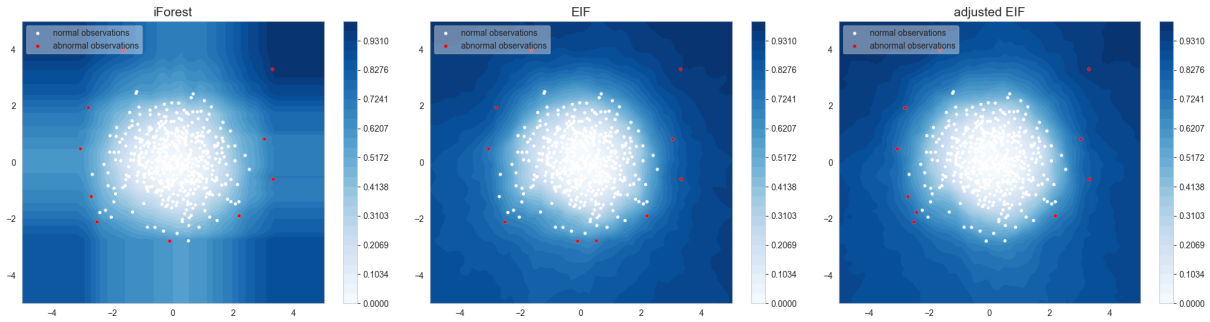
(b) Double Blob

(c) Sinusoidal

(d) Circle around Single Blob

Figure 5.1: Synthetic 2-D datasets

before, this happens since the axes-parallel split process in the training stage of iForest. On the other hand, the score maps of EIF and adjusted EIF show a nearly concentric circle shape, with higher scores on the outer part from the center of the cluster. The serrated shape for each contour on the score maps occurs due to the random split and the data distribution.

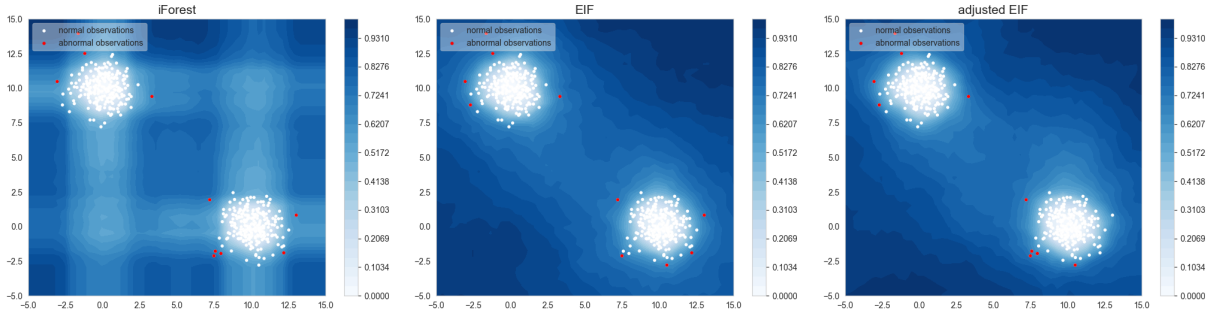


(a) iForest (b) EIF (c) adjusted EIF

Figure 5.2: Score map and scatter plot for iForest, EIF and adjusted EIF on the single blob dataset

In the second case, as figure 5.3 illustrates, both EIF and adjusted EIF overcomes the problem that iForest introduces axis-parallel rectangular areas around both blobs. In figure 5.3(a), it even produces another two "ghost" clusters at the overlaps of these rectangular, where low scores are assigned. As a lower anomaly score represents fewer probabilities of being an outlier, it will hardly be identified as an anomaly if there is a point located in the "ghost" region. Comparing the contour maps of the other two methods, basically, EIF and adjusted EIF share a similar score map. Nevertheless, the adjusted EIF has a slightly wider low score area assigned to the top of the right blob. It is probably because PCA is sensitive to the shape of the cluster. Since the outliers of the right blob locate far from the top, the projection on the direction with the highest data variance could lead to an irregular cutting.

Regarding the sinusoidal case, the same issue of rectangular artifacts is more severe, as demonstrated in figure 5.4(a). The contour of iForest is numerous crosses overlapping along the axes, forming a large band on the data space instead of a sinusoidal shape. Interestingly, unlike the other two methods, iForest regards points on the crest and trough as outliers, while EIF and adjusted EIF find most outliers on the far left and rightmost side with almost no difference. The explanation for this phenomenon remains the same as the previous illustration: iForest tends to ignore the points buried by the contour along the direction of axes. It can be proved that points far from the cluster but parallel to the main direction of data can be identified in EIF and adjusted EIF, where the contour of the score shows a precise sine shape.

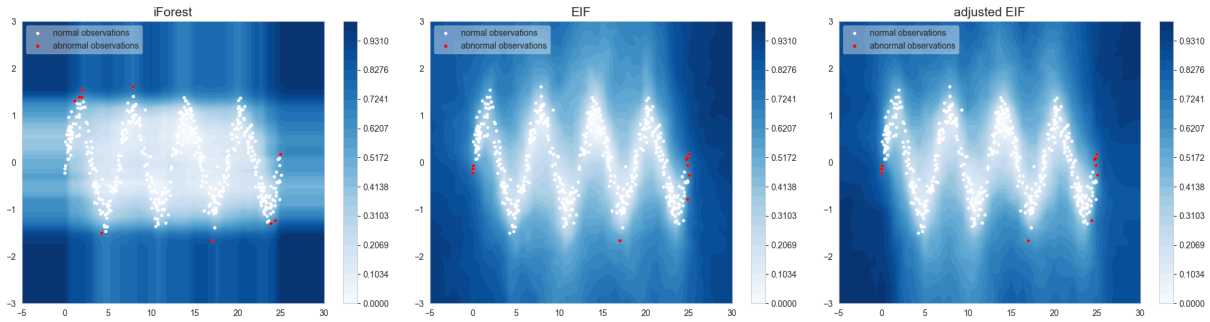


(a) iForest (b) EIF (c) adjusted EIF

Figure 5.3: Score map and scatter plot for iForest, EIF and adjusted EIF on the double-blobs dataset

The result of the last case is in figure 5.5. Similarly, iForest treated the composite graphics as a small cross upon a large cross. At the same time, other methods are able to separate the circle and the blob with a relatively

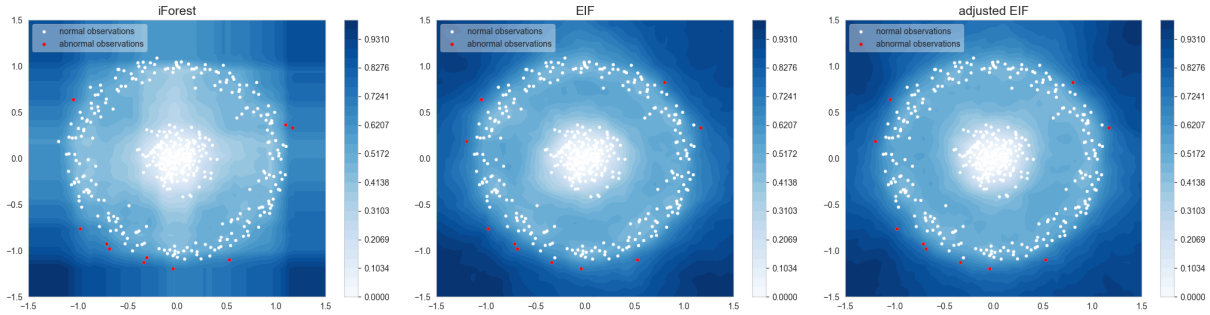
high score region between them. However, a new problem arises. None of the three mentioned algorithms can detect the outliers between the circle and the blob. From the visualization of detection, Only the points outside the circle are recognized as anomalies. This is also an issue stated on section3. Therefore, the following section introduces the implementation of solutions in subsection 4.3 for the multi-class detection issue.



(a) iForest (b) EIF (c) adjusted EIF

Figure 5.4: Score map and scatter plot for iForest, EIF and adjusted EIF on the sinusoidal dataset

In conclusion, this section implements the adjustment of EIF mentioned in section 4.2 and evaluates it on two types of datasets. Comparing the AUC-ROC and AUC-PRC with other anomaly detection algorithms on real-world high-dimensional datasets, the adjusted EIF is manageable to achieve relatively high accuracy in identifying outliers of most of the listed datasets. And for 2-D normally distributed datasets, basically, it has a similar performance as EIF, and thus can address the problem introduced by iForest.



(a) iForest (b) EIF (c) adjusted EIF

Figure 5.5: Score map and scatter plot for iForest, EIF and adjusted EIF on the circle around a blob dataset

5.2 Visualization of cluster-wise detection

This section contributes to solving another issue: isolation-based algorithms fail to detect the local outliers in each cluster for multi-class datasets as explained in chapter 3.2. In this section, experiments consider various datasets with various numbers and distributions of clusters representing different situations. As the EIF addresses the problem of unusual score maps for 2-D datasets in iForest, the following implementations and experiments are based on EIF.

5.2.1 Implementation

The work is mainly implemented based on NumPy library [HMdW+20] and EIF source code [HKB19] in Python 3.7 environment. Firstly, for the points in each class, identify the outliers by the decision value from EIF. Secondly, for each pixel in the 50×50 mesh grid in the range of minimum and maximum of each coordinate of data, it

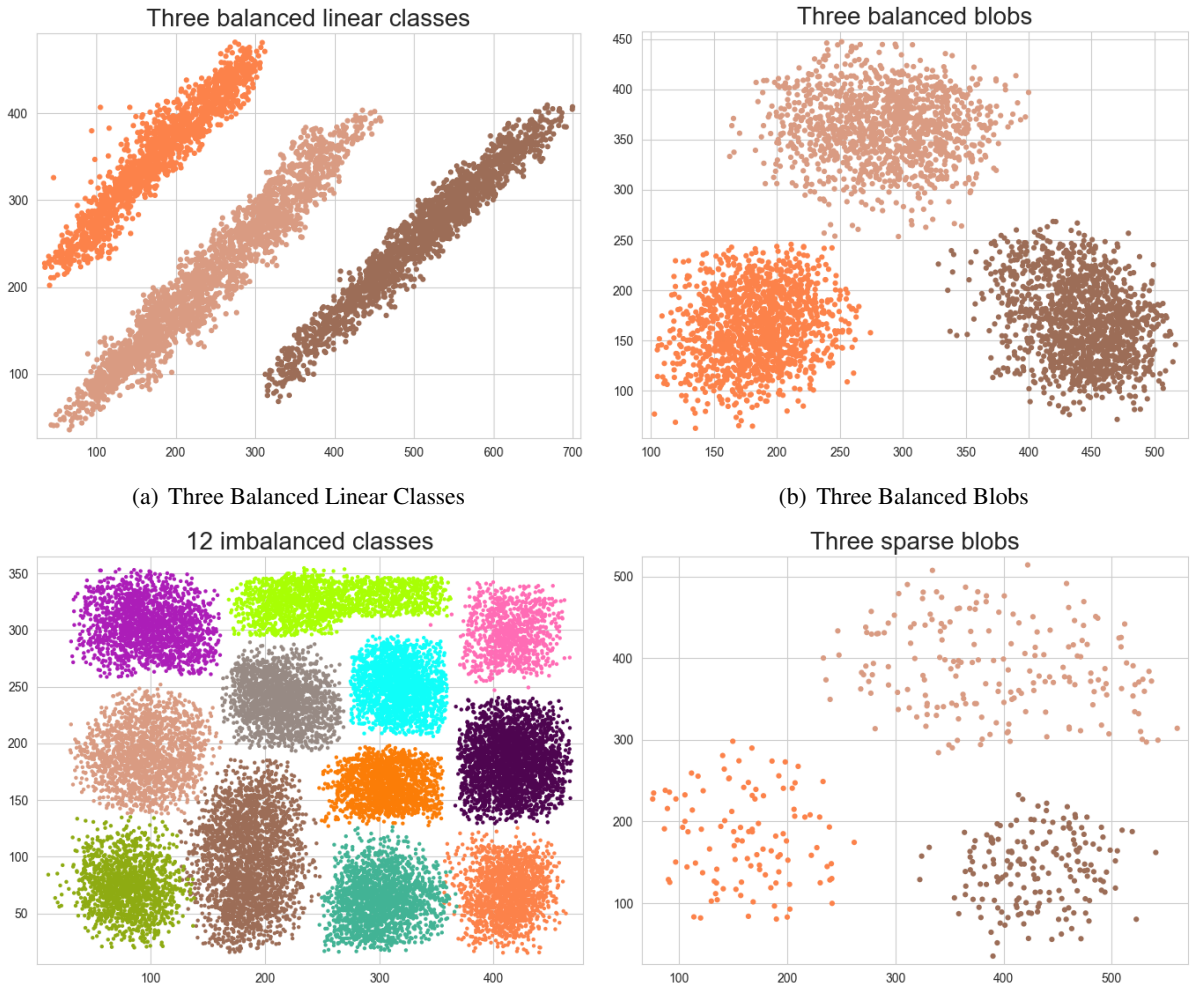
computes an anomaly score for each class. Thirdly, aggregate the anomaly scores in each pixel cluster-wisely and normalize the aggregation value to have a final score. There are three different tryouts to be tested here. They are computing the arithmetic mean, the n-norm (0.5-, 1-, 1.5-, and 2-norm), and the geometric mean of scores, respectively. The following pseudocode indicates the implementation of this method.

```
# Generate a 50*50 meshgrid in the range of dataset X
xx,yy = meshgrid(linspace(min(X[:,0]),max(X[:,0]),50), linspace(min(X[:,1]),max(X[:,1])
    ,50))
# For each class i in dataset X
for i in range(n_class):
# EIF implemented to detect outliers
    Begin
    # Train iTrees using nodes in class i
    F1 = iso.iForest(X[class_i])
    # Evaluate each node in class  i with an anomaly score
    S1 = F1.compute_paths(X_in = X[class_i])
    # Sort the anomaly score
    ss1 = argsort(S1)
    # Pick the 10 points with the highest anomaly score in that class as outliers
    X_outliers = X[cluster_i][ss1[-10:]]
# EIF implemented to generate score map
    # Evaluates each pixel on the meshgrid with an anomaly score
    S2_i = F1.compute_paths(X_in = (xx,yy))
    End
# Define an array that store the anomaly score of pixels in each class
S2 = [S2_1,...,S2_n_class]
# Compute the arithmetic mean and normalize
If (method = 'arithmetic mean'):
    Begin
    # For each pixel, average the sum of its score in each class
    S = sum(S2)/n_class
    # Normalize the results to [0,1]
    normalized_S = (S-min(S))/(max(S)-min(S))
    End
# Compute the norm of anomaly score vector
Elif (method = 'norm'):
    Begin
    # Calculate n-norm of the cluster-wise anomaly score for each pixel, and normalize
    S = norm([S2], norm = n)/(power(n_class,(1/n))
    # Normalize the results to [0,1]
    normalized_S = (S-min(S))/(max(S)-min(S))
    End
# Compute the geometric mean and normalize
Else:
    Begin
    # For each pixel, compute the n_class root of the product of scores in each pixel
    S = prod(S2)/power(Z,1/n_class)
    # Normalize the results to [0,1]
    normalized_S = (S-min(S))/(max(S)-min(S))
    End
```

The *normalized_S* in code represents the final anomaly score assigned to each pixel. A contour map of the grid data space and scatter plot with outliers highlighted are displayed based on the *normalized_S* eventually.

5.2.2 Synthetic datasets

Several synthetic datasets from [Yoo22] with different variate distributions are considered in this experiment. As figure 5.6 shows, three other datasets are applied except for the three-cluster dataset mentioned in section 3.



(a) Three Balanced Linear Classes

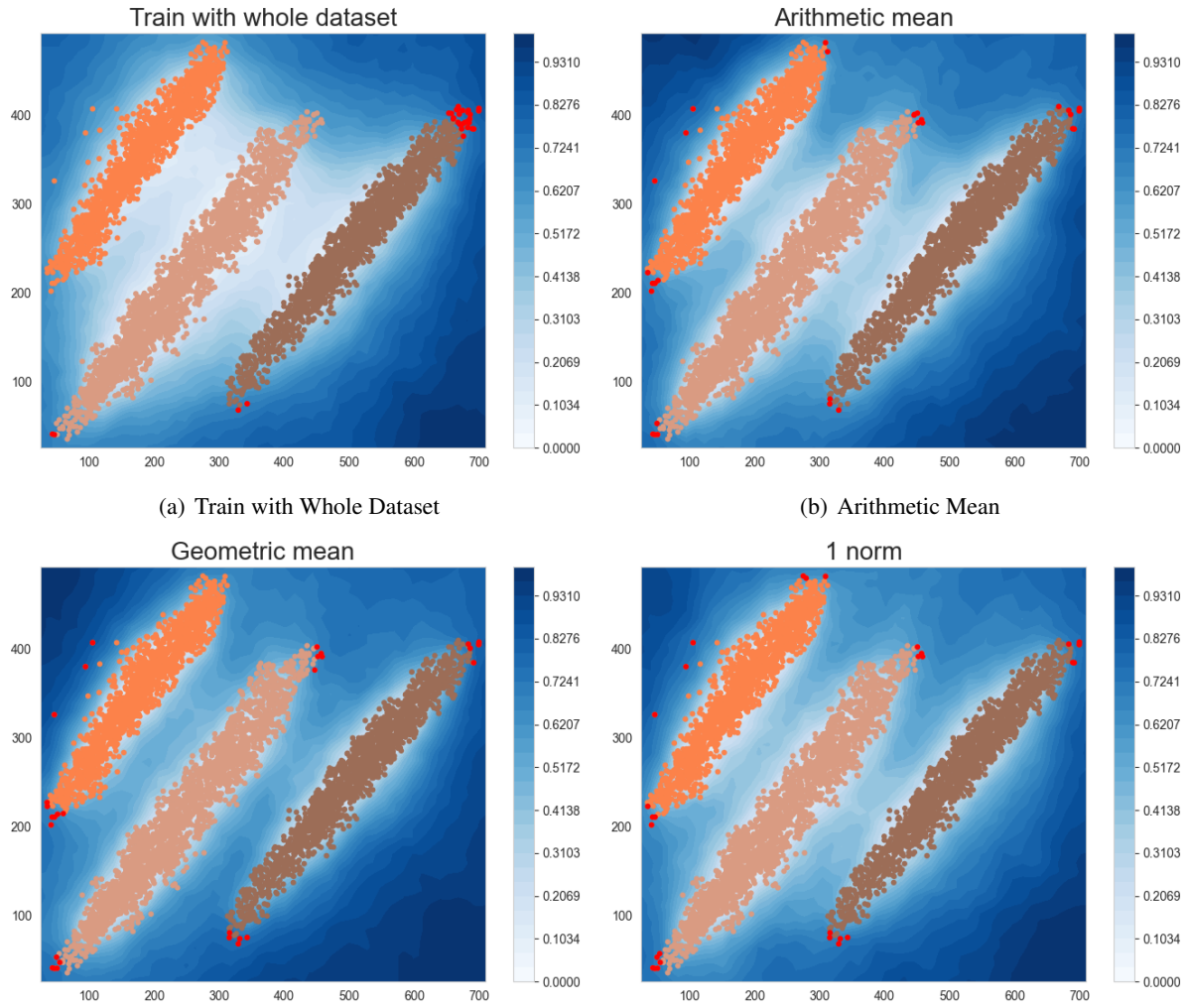
(b) Three Balanced Blobs

(c) Twelf Imbalanced Classes

(d) Three Sparse Blob

Figure 5.6: Synthetic 2-D datasets

Figure 5.6(a) illustrates a dataset with three linearly distributed classes that have similar numbers from left to right. The second dataset, visualized in figure 5.6(b), has three equal number blobs with several points scattered around each blob. The next dataset displayed in figure 5.6(c) contains 12 imbalanced clusters arranged in a square space. And the last one in figure 5.6(d) is formed by three sparse blob-shape classes, where several points are relatively far from each class. These datasets cover different situations for anomaly identification to ensure the proposed method can be utilized in general.



(a) Train with Whole Dataset

(b) Arithmetic Mean

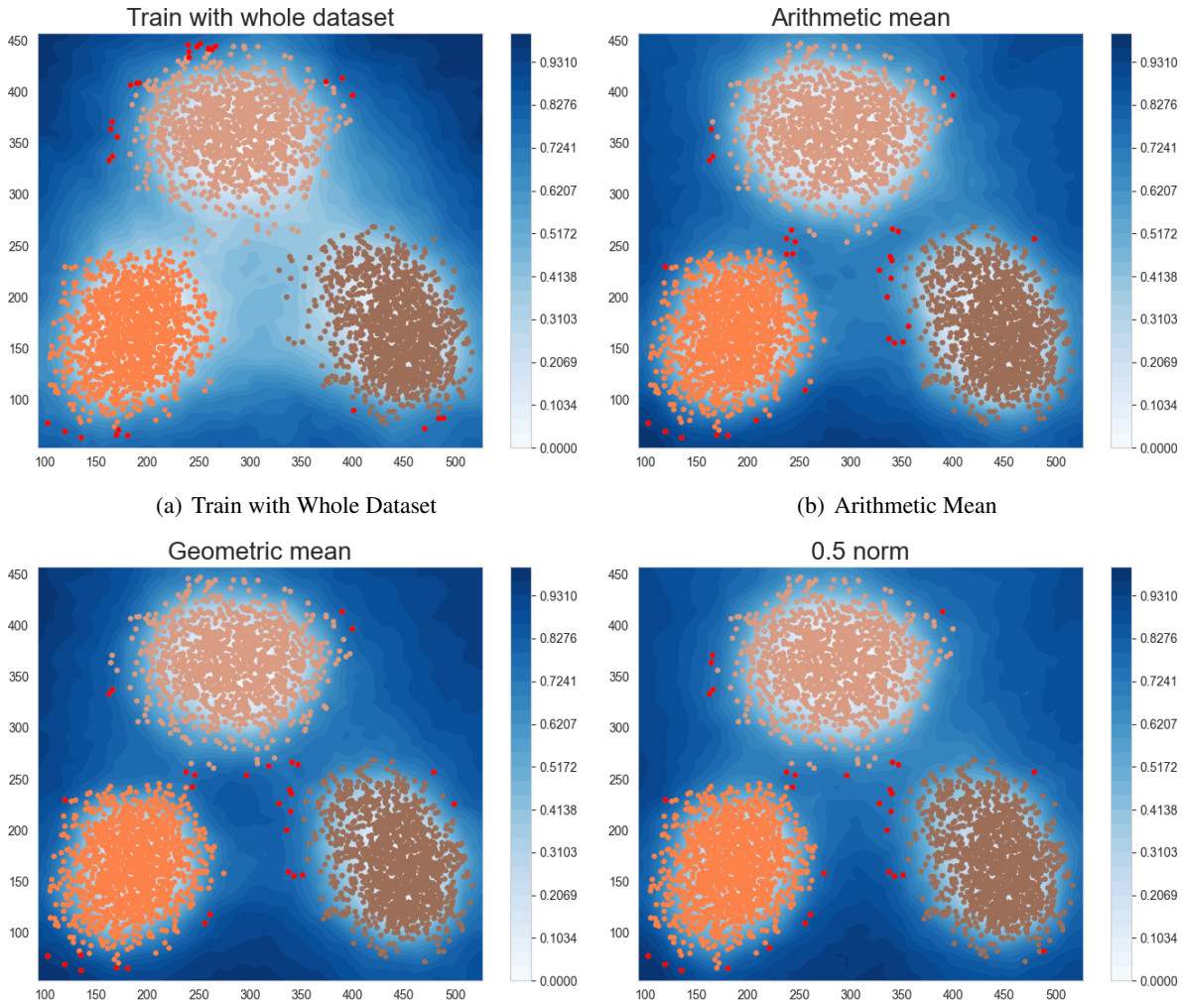
(c) Geometric Mean

(d) 1 Norm

Figure 5.7: Comparison of score map and scatter plot for treating the whole dataset as one with three proposed cluster-wise training methods on three balanced linear classes

Following the implementation steps mentioned in 5.2.1, different score maps joined with scatters are generated for each dataset. For a particular dataset, the scatter plot of each method shall remain similar because it is executed before the production of the score map. The comparison is made on the score map across the three aggregation approaches, as well as training the whole dataset directly with EIF. Precisely, for the n-norm method, as four kinds of norm (0.5, 1, 1.5, 2) are examined, this work only displays the one with the best performance and puts the rest of the results in the Appendix part. The outliers are defined as the ten points with the highest anomaly scores in each dataset class (the $numberofclass \times 10$ points with the highest scores for the case of full datasets training). The scatter colors each class distinctively, while the outliers are in red.

As illustrated in figure 5.7, there is no doubt that treating the whole dataset as one class is not a choice since it is not able to neither detect the obvious outliers on the left of the first class (the orange cluster) nor drawing a clear demarcation line to separate the classes, while the other proposed methods have a relatively better capacity



(a) Train with Whole Dataset

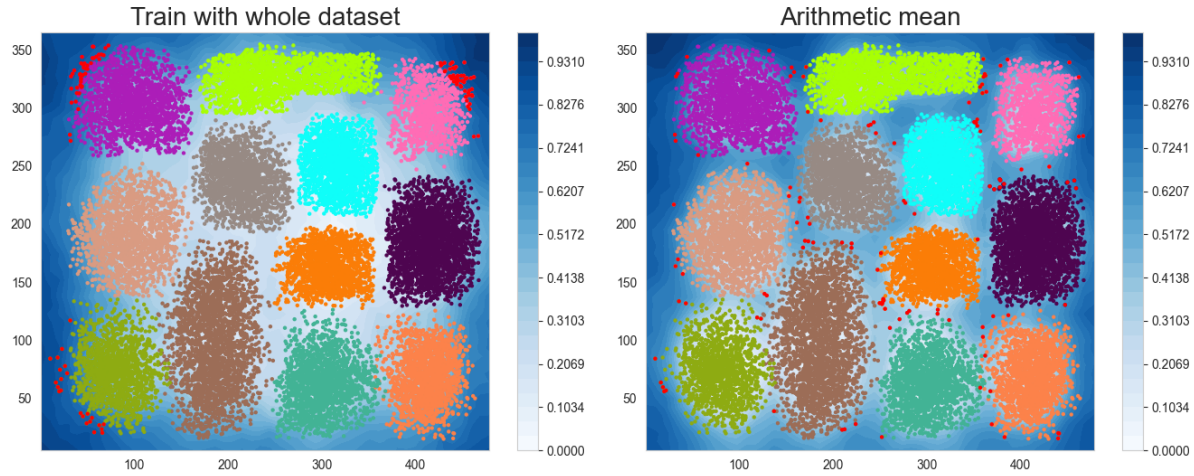
(b) Arithmetic Mean

(c) Geometric Mean

(d) 0.5 Norm

Figure 5.8: Comparison of score map and scatter plot for treating the whole dataset as one with three proposed cluster-wise training methods on the three balanced blobs dataset

of drawing decision boundary. As for the arithmetic mean method, it does not show an apparent contour between the middle (the flesh color cluster) and the right (the brown) classes. Although the 1-norm method can sketch a precise contour for each cluster, the geometric mean method handles separate classes smoothly, with higher anomaly scores assigned to the middle between classes.



(a) Train with Whole Dataset

(b) Arithmetic Mean

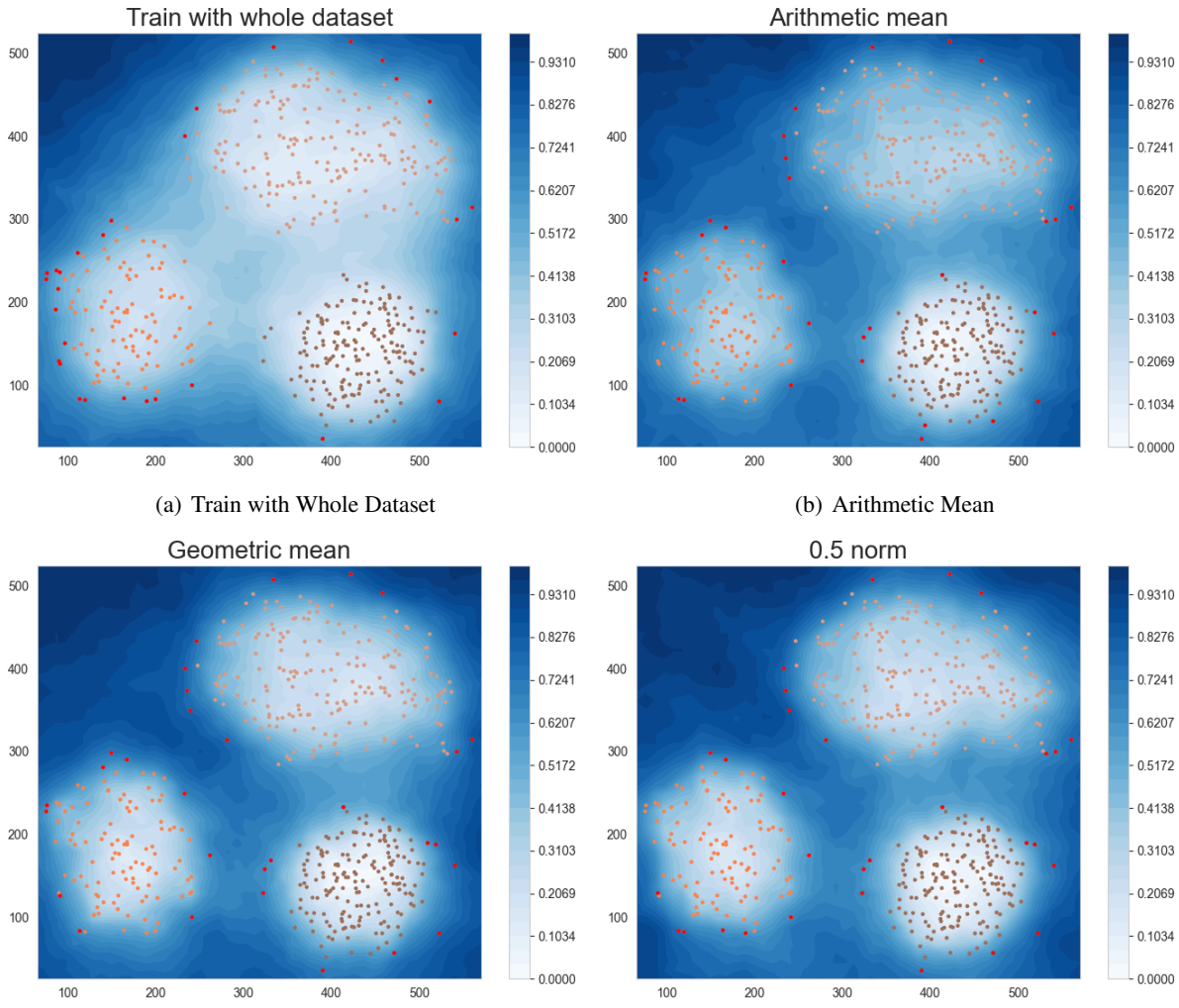
(c) Geometric Mean

(d) 0.5 Norm

Figure 5.9: Comparison of score map and scatter plot for treating the whole dataset as one with three proposed cluster-wise training methods on the 12 imbalanced classes dataset

In the second case, which is demonstrated in figure 5.8, the original EIF treats the whole dataset as a huge triangle-like shape and considers those points lying in the outer circle as anomalies. The arithmetic means method and 0.5-norm method share a similar distribution on the score map, while the geometric mean method gives higher anomalies in the surrounding area of each blob. In addition, the geometric mean method is the only one that designates the space center with high anomaly scores. It is an intuitively sensible result since the center is rather far away from the centroid of each blob.

The third case describes a rather complicated situation with 12 imbalanced classes. Figure 5.9 produces an interesting insight. The original EIF is even weaker at detecting local outliers in dense clusters since it only regards the corners of the rectangular space as outliers. And the arithmetic mean method, surprisingly, has a favorable performance in indicating the contour of each cluster, even slightly better than the 0.5-norm method case between some classes. And the geometric mean method, as previously, maps the outliers with higher anomaly scores and has higher scores for the in-between regions among several close classes.



(a) Train with Whole Dataset

(b) Arithmetic Mean

(c) Geometric Mean

(d) 0.5 Norm

Figure 5.10: Comparison of score map and scatter plot for treating the whole dataset as one with three proposed cluster-wise training methods on the three sparse blob dataset

The experiment result of the last case is portrayed in figure 5.10. For this unusually distributed sparse dataset, The EIF could generate a blur gap in the center of the graph, although the outliers of each class near the center can still be detected. The geometric mean method retains its strong cluster-wise detection ability. The band between the left and the top clusters is crystal clear, as they are physical with great distances. The 0.5-norm method allocates the upper left corner with a higher score, while the arithmetic mean method gives the center bottom a clear high score region.

To conclude, compared with the original EIF, which treats the whole dataset as one variate, all proposed approaches are able to indicate the area where the outliers and potential outliers sit with higher anomaly scores. Among them, the geometric mean method can better sketch the contour of each class on synthetic datasets, with the outliers always appearing on the high score space.

5.2.3 Real-world datasets

This subsection investigates eight real-world datasets involved in the work of [YXX+21] with more complicated distributed classes. However, to keep the work concise and representative, only the results of three datasets with the most representative situations are displayed and analyzed in this subsection, while the others are in the Appendix.

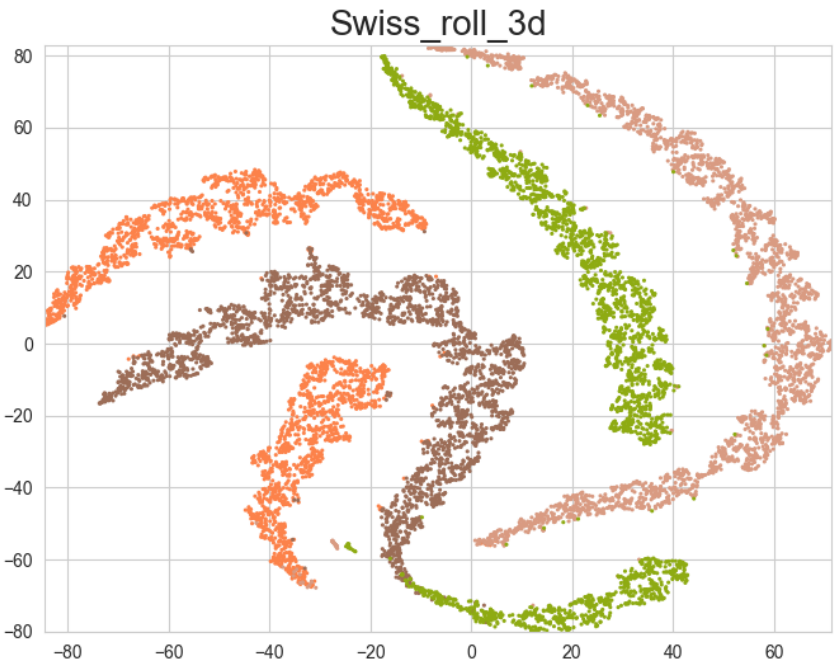
The detailed description of all eight datasets is presented in the table 5.5 as follows. The outliers are set as 2% of points in each dataset class. Most of the datasets in this subsection contain more classes and overlapping between classes. Therefore, these cases are more realistic, reflecting a more real-world situation than the ideal synthetic datasets. A similar experiment as in subsection 5.2.1 is executed across these datasets. It compares the score map of the original EIF that treats the whole dataset as one class with the three proposed cluster-wise detection methods. Similarly, here in the result figures, the work only presents the best performance $n - norm$ solution and puts other results from other norms in Appendix. Furthermore, the majority of the datasets have numerous cluttered data points on scatter plots, covering the whole score map and making it hard to recognize the detail. Therefore, in the following experiment, the results drop the scatter plot for the whole dataset and only retain the outliers. In addition, visualization of original data points is added to judge the quality of outlier detection.

Table 5.5: Real-world 2-D clustering datasets

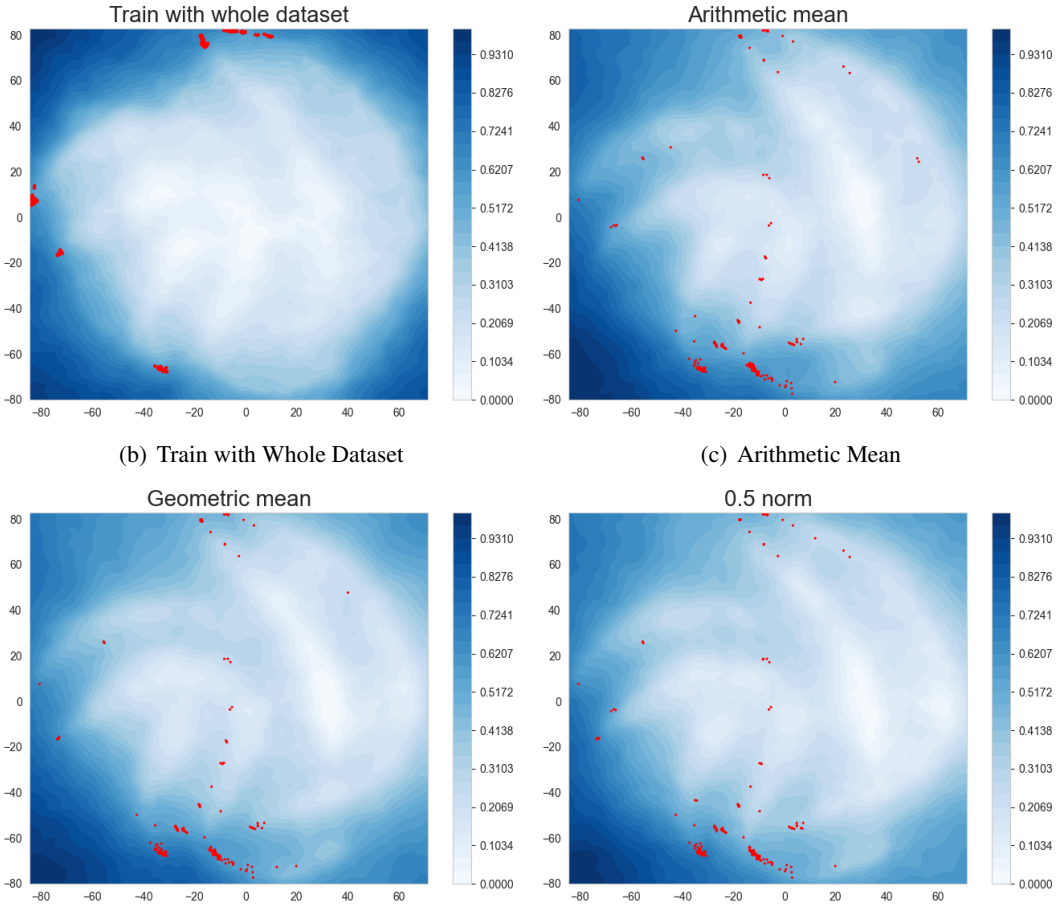
Name	Size	Class	Outlier
Abalone	4177	3	83
Crowdsourced_mapping	10845	6	216
Condition_based_maintenance	10000	9	200
Clothes	26569	10	531
Epileptic_seizure	11500	5	230
Mnist	70000	10	1400
Swiss_roll_2d	8000	4	160
Swiss_roll_3d	10000	4	200

The first case compares the graphs in figure 5.11(a). There are four curved classes indicated with orange, brown, green, and khaki, respectively. Note that some points are scattered in other classes and far away from their main clusters. All listed methods can identify those points, except the original EIF, which basically processes the whole dataset as a giant blob and considers the center as the dentist area. The other three methods are capable of sketching a blurred contour for each cluster and finding the points away from most of the points in one class. Generally, it is difficult to tell who performs best for this dataset. However, the arithmetic mean method places the outliers in relatively darker areas, especially for the outliers falling in the khaki class.

The following case investigates a rather sophisticated dataset, $Condition_based_maintenance$. Illustrated in figure 5.12(a), the dataset is a sparse nine-cluster set with the violet class intertwined with the dark purple class. Figure 5.12 demonstrates that the EIF is incapable of finding the outliers that fall in the "crack" between the



(a) Original Dataset



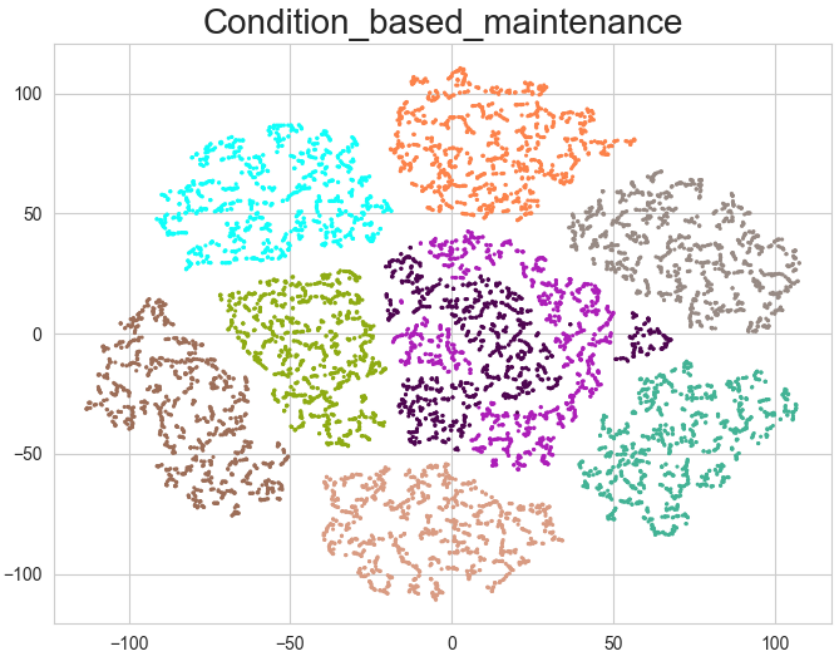
(b) Train with Whole Dataset

(c) Arithmetic Mean

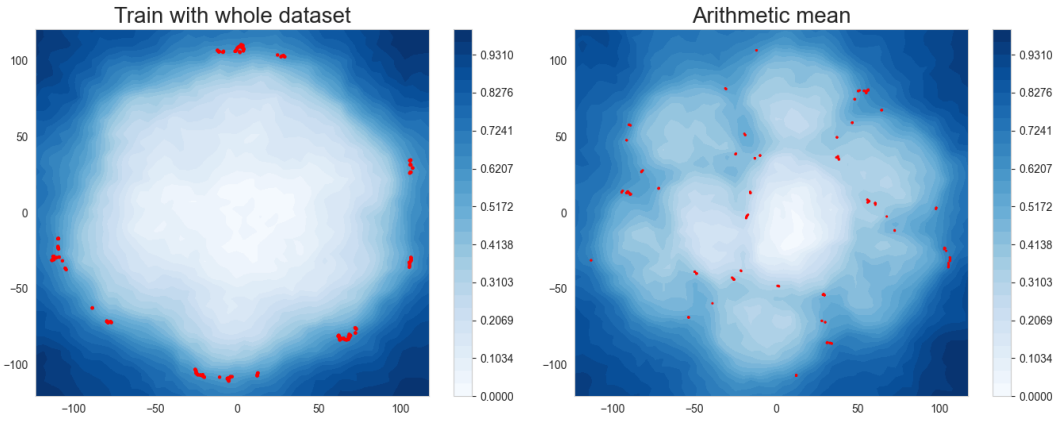
(d) Geometric Mean

(e) 0.5 Norm

Figure 5.11: Comparison of score map and outliers for treating the whole dataset as one with three proposed cluster-wise training methods on the *Swiss_roll_3d* dataset

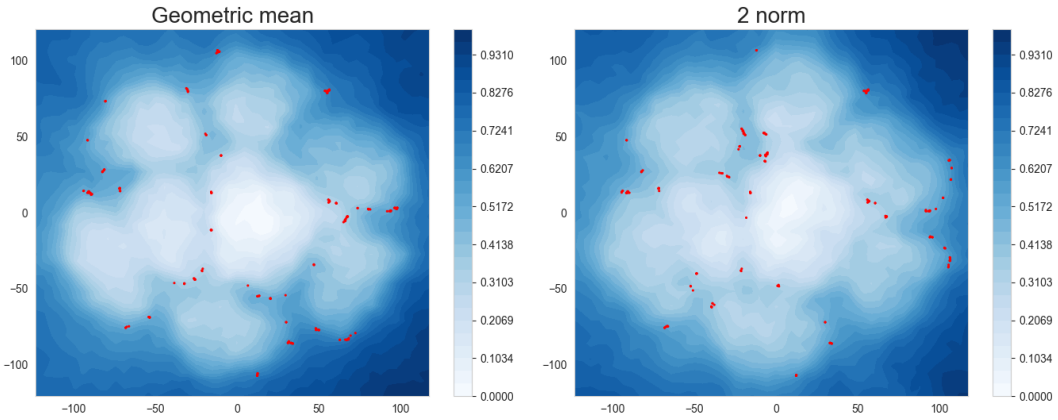


(a) Original Dataset



(b) Train with Whole Dataset

(c) Arithmetic Mean



(d) Geometric Mean

(e) 2 Norm

Figure 5.12: Comparison of score map and outliers for treating the whole dataset as one with three proposed cluster-wise training methods on the *Condition_based_maintenance* dataset

clusters. The proposed methods, however, merged the intersected classes and recognized this region as a place where outliers will have less possibility to appear. For the 2-norm method, the range of data points is assigned with a relatively evenly low score, although there are still obvious gaps around classes. The outliers in this method are scattered around each class as the arithmetic mean and the geometric mean methods do. Overall, the arithmetic mean method scores the four corners evenly, while the other two methods favor the right corners with higher scores.

A super complex dataset is displayed given the results in figure 5.13. In this biased dataset, the largest class has 7509 nodes, while the smallest has only 100. For the result figures, obviously, the EIF regards it as a sizeable sparse blob with outliers around the outer. Unexpectedly, other methods produce entirely distinct outcomes. The low score areas are shaped as a semi-oval, which is intuitively incompatible with the original dataset. This happens because for each pixel in the largest class (the brown class), though it assigned a low score in this class, the other five classes raise the scores as they are even weighted in the calculation. That means the lower score in one class is "diluted" by the other five higher scores in other classes. As scores of those points are pulled up, the region of the largest class is then invisible on the score map. In order to tackle this issue, a further experiment is conducted based on another method in the following subsection.

In conclusion, in the real-world 2-D dataset with no class overlapping on other classes, all the listed methods can detect the local outliers and generate clear contour lines by anomaly score to indicate the potential anomalies. Among them, in complex real-world cases, the arithmetic mean method is moderately better in viewing the score map with outliers. However, when the dataset is uneven and imbalanced, with points belonging to one class scattered in another class, the proposed methods have limitations in generating a reasonable score map that outliers can situate in appropriate locations. Based on this issue, further tryouts are examined. The first is to combine the adjusted EIF proposed in section 4.2 with the methods proposed in section 4.3. And the second is to devise a new visualizing method for each pixel, deciding the score by the minimum of scores in all clusters.

5.2.4 Further experiments

This subsection attempts to extend the experiment. Likewise, to keep it concise and interpretable, the two tryouts are examined on one synthetic dataset and two real-world datasets listed in this section. And the comparison is made between the two new methods and the geometric mean method since it generally has the most sensible results both in synthetic and real-world 2-D clustering datasets. At first, a brief description of the two methods are presented.

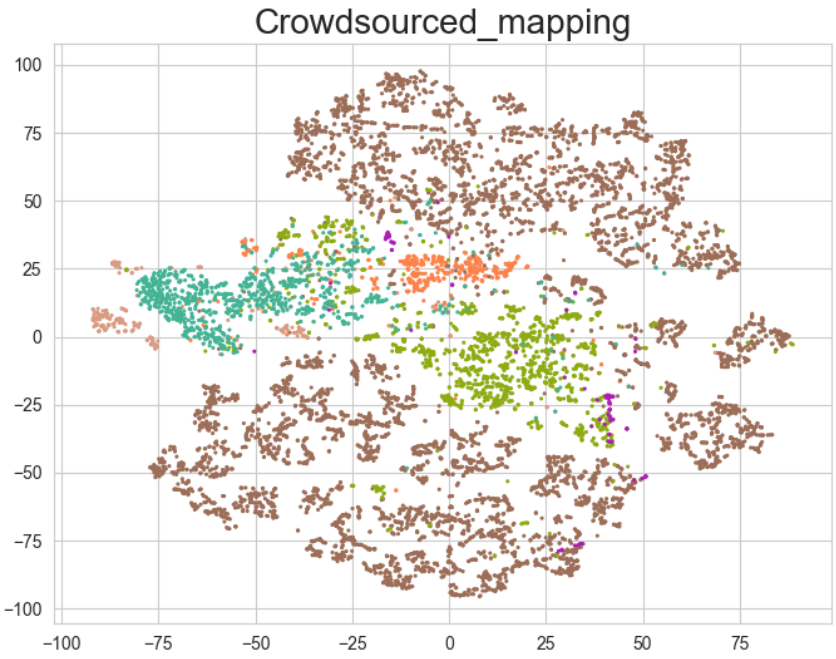
Combination of adjusted EIF with geometric mean method

Because the adjusted EIF handles improve the detection performance of EIF on real-world datasets, it is meaningful to evaluate it in the cluster-wise visualization mission. Furthermore, since the geometric mean method achieves a relatively better performance than the norm and arithmetic mean method in general, testing the combination of it and adjusted EIF is logically reasonable. The implementation of this idea is to replace the EIF with the adjusted EIF in each class detection, and the setting remains the same as the table 5.2.

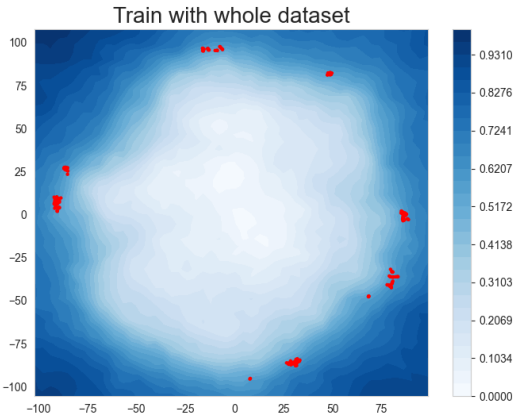
Minimum method

Due to the dilution that happens in imbalanced class cases, the minimum method is proposed. The method alleviates the effect of other classes by always choosing the minimum score across classes for each pixel. Moreover, the method still allocates a high score for a pixel away from data points since every class will still assign it with a high anomaly score. The implementation is similar to the pseudocode in subsection 5.2.1 by only switching the geometric mean to the minimum. Then the same linear normalization is conducted to scale the scores to $[0, 1]$ range.

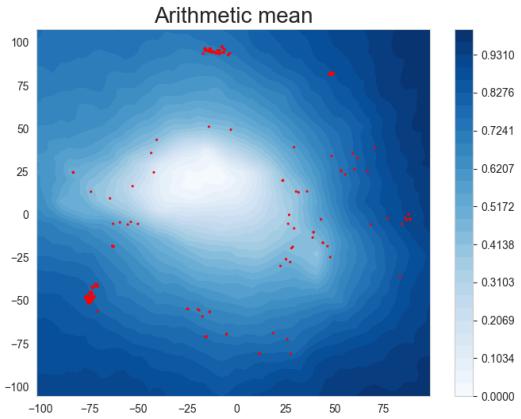
The experiments are based on the *three balanced linear classes*, *Condition_based maintenance*, and *Crowd-sourced_mapping* datasets, where the distributions have been instructed in the last subsections. Moreover, the visualization is similar to the corresponding results from the last subsections. Outliers are marked in red, and the



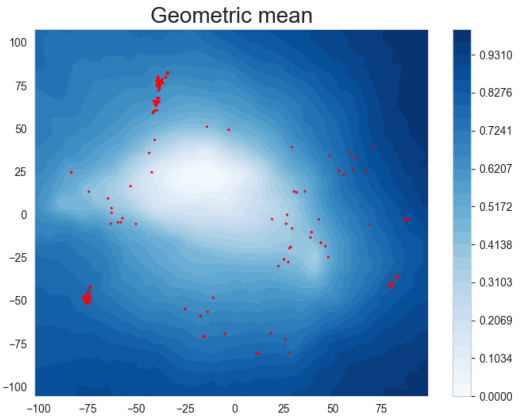
(a) Original Dataset



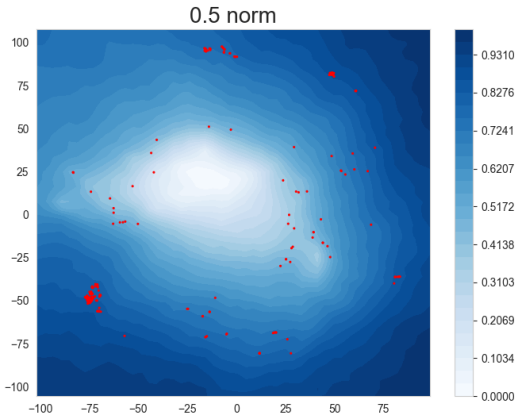
(b) Train with Whole Dataset



(c) Arithmetic Mean



(d) Geometric Mean

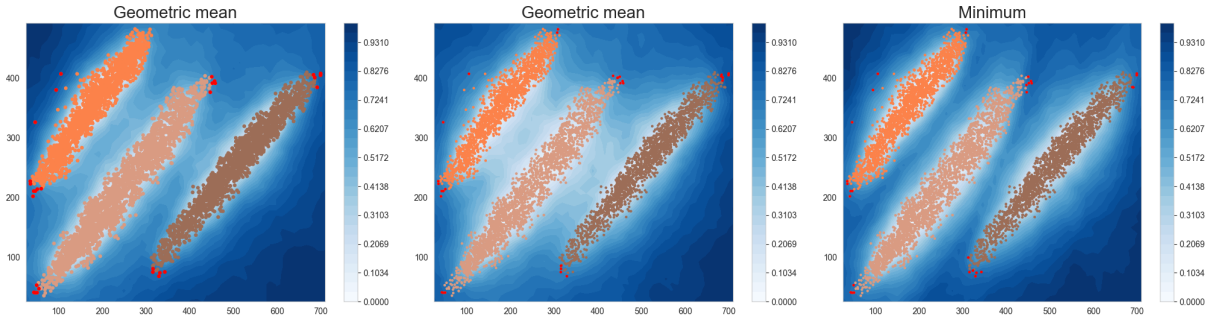


(e) 0.5 Norm

Figure 5.13: Comparison of score map and outliers for treating the whole dataset as one with three proposed cluster-wise training methods on the *Crowdsourced_mapping* dataset

score map from shallow to dark represents the smallest to largest scores. The comparison is made between two new methods with the geometric mean method upon EIF.

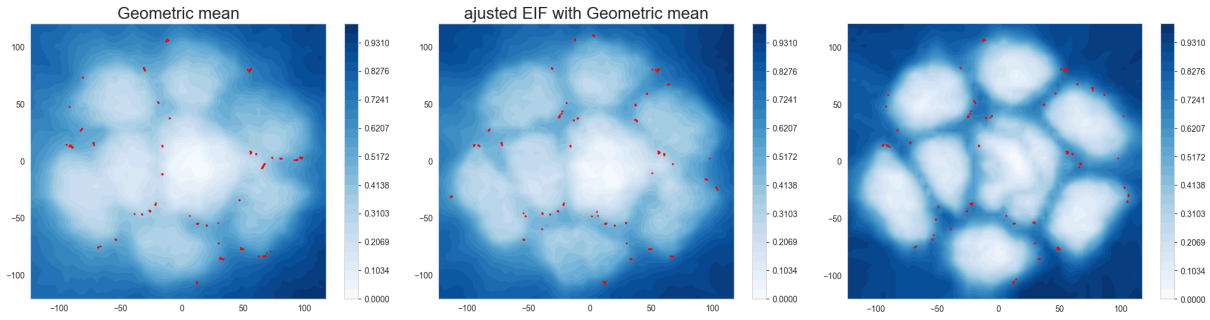
Figure 5.14 compares the first experiment's results. The adjusted EIF combined with the geometric mean generates uneven borders between clusters, with no outliers lying there. This phenomenon might occur due to the vertical cut in the direction of maximum variance from PCA. Since the algorithm always splits the data points in one class by that direction, the in-between areas always need more cuts than the end of clusters, which means they always gain lower scores, though still higher than the dense points area. However, the minimum method shows a powerful capability to delineate the clusters. Note that the whole areas around the clusters have very high scores, which frankly indicates the region where the potential outliers will sit.



(a) Geometric Mean (b) Adjusted EIF with Geometric Mean (c) Minimum

Figure 5.14: Comparison of score map and outliers for geometric mean method with two new proposed methods on the *three balanced linear classes*

The second results are delivered in figure 5.15. The first new method improves the class distinguishability of EIF in this case. The in-between "cracks" are more evident than the EIF to show where the outliers fall. However, the minimum method is somewhat more robust in that it produces perfectly noticeable gaps between each class, even the two interlocking classes. All the listed methods treated it as a whole blob in the two clusters, except the minimum method. It is reasonable that the points between the two classes should be outliers intuitively since they are far from each class.



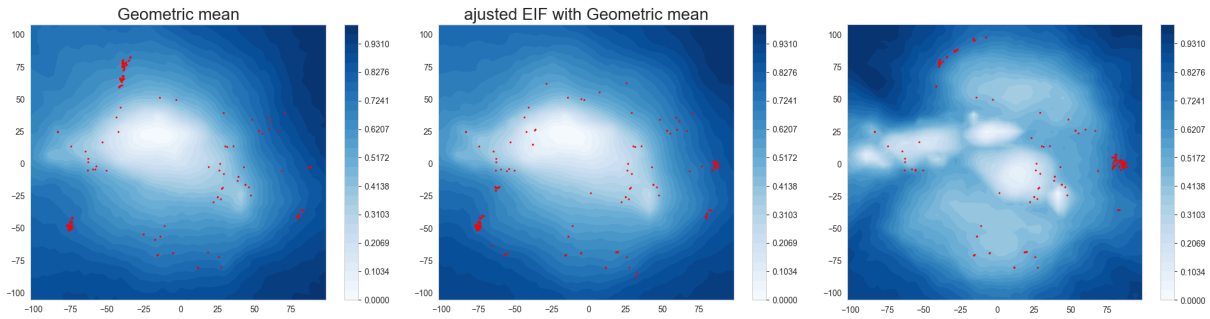
(a) Geometric Mean (b) Adjusted EIF with Geometric Mean (c) Minimum

Figure 5.15: Comparison of score map and outliers for geometric mean method with two new proposed methods on the *Condition_based maintenance* dataset

In the last experiment, as in figure 5.16, the first method basically shares a similar score map as the original geometric mean method. The only difference between them is the outliers detected. That for the first new method, the rightmost points are anomalies, while for the EIF-based geometric mean method, the small upper cluster is recognized as the outlier. Nevertheless, the advantage of the minimum method comes through in this highly complex dataset. In figure 5.16(c), each of the classes is colored in light, especially those dense small clusters, whereas in the other two methods, they only figure out a blur large semi-oval shallow area. The other insight that

can be abstracted from this graph is that the minimum method can indicate where the biggest class is, with a high score showing between it and other small clusters. The color for the outer class is rather shallow, since the data points there are relatively sparse.

In conclusion, the geometric mean method based on adjusted EIF achieves to handle a relatively complex dataset compared to the method based on EIF. However, when the class shows a distribution of linear, it may experience a hard time to split the clusters due to the limitation of PCA. On the other side, the minimum method achieves distinguishing each class and assigning the corresponding score in diverse clustering datasets, no matter how simple or complex the class distribution is. It can be explainable to identify potential outliers by viewing the score map.



(a) Geometric Mean (b) Adjusted EIF with Geometric Mean (c) Minimum

Figure 5.16: Comparison of score map and outliers for geometric mean method with two new proposed methods on the *Crowdsourced_mapping* dataset

To conclude this section, several experiments to compare different cluster-wise EIF/adjusted EIF-based methods are conducted. As the results show, the geometric mean method is able to detect local outliers and depict a sensible score map for simple synthetic datasets, while the n-norm and arithmetic mean methods are relatively weak at the contour map generation. The adjusted EIF has limitations to improving the EIF on specific datasets but has a similar to slightly better performance on other datasets. Among all the proposed methods, the minimum method is manageable for scoring the multi-variate classes with a plausible contour map, revealing the distribution of classes, and implying the potential outliers for the dataset.

5.3 Discussion

Note that this thesis dedicates to achieving two goals. The first is to improve the performance of isolation-based algorithms. And the second is to come up with a visualization solution for detecting outliers in multi-class datasets based on isolation-based algorithms.

5.3.1 Algorithm improvement

To address the problem of generating abnormal scores for axes-parallel areas in the score map, the EIF algorithm is presented [HKB19]. However, despite the fact that EIF solves the issue by creating the hyperplane determined by random normal vector and intercept, it still has shortcomings. For example, it fails to split the data points due to randomness. This thesis proposes adjusted EIF which modifies the random split hyperplane to a certain hyperplane decided by the first component of PCA and KNN. The results demonstrate that the adjustment makes significant progress in the detecting accuracy across several real-world datasets than classical anomaly detection algorithms, especially the iForst and EIF. The drawback of this adjustment is that as the dimension of the dataset rises, the training time will also increase exponentially. That is because the time complexity of PCA is $O(min(N^3, M^3))$ in an M-dimension dataset with N instances, as it needs to compute the covariance matrix and eigenvalues for the datasets. And the time complexity of KNN for the following value selection step is $O(N * M)$. Therefore, the PCA process on each partition limits the efficiency of the adjusted EIF. However,

thanks to the sampling and ensemble characteristics of isolation-based algorithms, the time of each cut is then reduced accordingly. In addition, since the distance calculation by KNN in the adjusted EIF is performed on the projection values of data on the 2-D plane, the complexity also substantially reduces. Given that the sample size and dimension are reduced in each split process, the adjusted EIF is still obviously faster than the PCA [SCSC03] and KNN [AP02] algorithms for anomaly detection. The generalized Isolation Forest (GIF) [LBST21], another isolation forest algorithm, is proposed by Lesouple et al. to deal with the empty branching issues in EIF. However, although it shortens the execution time for the training phase, it fails to improve the precision of outlier detection. Generally speaking, the proposed method enhances both EIF and iForest on the capacity of outlier detection in most high-dimensional real-world datasets.

Another tryout on modifying the scoring rules of EIF is proposed. Switching the scoring of anomaly from computing the mean of the depths a node travels in each tree to computing the median does not help strengthen the outlier identification. It might be because extreme values of depths for a point to pass all the iTrees are less likely to appear than normal values. Since the height of iTree limits the depth, the extreme value has little impact on the final score. That means, practically, the median might yield a roughly equivalent value to the mean. In addition, altering the normalization from exponential to sigmoid-like function contributes little to improve the accuracy, because revising the normalizing rules is simply scaling the final results. Thus, the rankings of anomaly scores for each point stay the same.

5.3.2 Visualization of outlier detection on multi-class dataset

This thesis also provides several solutions for the local outlier detection issue in isolation forest algorithms. That is, for each variate, detecting the corresponding outliers. Moreover, drawing a decision boundary by combining the collection of anomaly scores for each class is also researched. Specifically, this thesis proposes three methods (arithmetic mean, geometric mean, and n-norm of anomaly scores for each pixel in the data space) and compares them with the method treating the whole datasets as one variate based on EIF. The results on simple synthetic datasets demonstrate that all the proposed methods allow the identification of the local outliers for each class. Among them, the geometric mean method generates the most straightforward final decision boundaries to imply the potential outliers in simple, balanced datasets, while the arithmetic mean method provides a relatively more explainable score map for complex real-world datasets.

Inspired by the fact that all three listed methods are not well performed to draw an interpretable score map on datasets with intertwined classes, further experiments on the adjusted EIF-based geometric mean method and minimum method are conducted. One can see that the former has limited performance on particular datasets, for example, the three linear classes dataset. The reason is that for a class with linear distribution, each split of data points is more likely to happen at the end of the class rather than the middle because the PCA always selects the perpendicular direction of the main direction of the linear class. As this is the case, the pixels near the middle of the class would always need more cuts to be isolated and, thus, have lower anomaly scores. Surprisingly, the minimum method outperforms all three proposed methods in simple to interlaced and complex datasets. The reason might be, for in-class pixels, merely picking the minimum from the anomaly score sets of all classes always designates a low score for the in-class area. But for the regions out of classes, this method only picks the minimal score for each pixel, which is consistently higher than the in-class pixel. The final score map is then plausible by scaling the scores with a linear normalization.

6 Conclusion and Future work

In conclusion, this thesis studies two main problems. For the problem of performance improvement of isolation-based algorithms, an adjusted EIF approach is presented. The proposed method settles both the block artifacts problems introduced by iForest and the empty branching problems introduced by EIF. The limit of the proposed method is that the complexity of PCA limits the execution time. Generally, it has a similar score map as the EIF when visualizing the 2-D synthetic datasets. However, evaluating the method with an accuracy of detection, the results indicate that compared to that of both iForest and EIF, it manages to improve the detection accuracy as well as precision and reduce the false positive rate significantly. As for the multi-class outlier visualization problem, comparisons over several cluster-wise solutions are made. The minimum method is tested to be the most efficient approach in depicting the score map to indicate where the outliers and potential outliers are.

There are several directions for future work. The adjusted EIF somehow sacrifices the time efficiency to improve accuracy. The follow-up for the adjusted EIF algorithm is to shorten the execution time by modifying the PCA and KNN steps to make it more efficient while maintaining the accuracy in detecting outliers because the most distinct feature of isolation-based algorithms is their low time complexity. Meanwhile, it is worth investigating credible anomaly scoring functions on a scale of [0,1]. The scoring rules could vary based on different split rules. Furthermore, as in the experiments on real-world datasets, all studied isolation-based algorithms degrade the precision of outlier detection in some specific datasets, further research on why this happens can also provide insights for the improvement of isolation-based algorithms.

For the outliers and decision boundaries visualization of EIF on multi-class datasets, a simpler way to detect the local outliers that do not need to merge the anomaly scores is the direction of further work. Another finding from the experiments is that outliers that fall into other classes might be assigned low anomaly scores, reducing the convince of the indicators. Thus, methods such as weighting the anomaly scores from each class and combining isolation-based algorithms with other clustering algorithms deserve further study.

Acknowledgements

I would like to sincerely appreciate my supervisor, Ms. Haiyan Yang, and my responsible professor, Prof. Dr. Renato Pajarola, for providing me with enlightening topics and with the precious chance to finish my thesis in the VMML group. Without their inspiring ideas and patient instruction, it would have been impossible for me to complete this work. I would also like to thank Dimitris Effrosynidis for providing the case study and code of visualization of isolation-based algorithms, Minqi for the evaluation code of classical anomaly detection algorithms (ADBench), Shebuti Rayana for benchmark datasets (ODDS), Sahand Hariri and Matias Carrasco Kind for the source code of EIF and several synthetic datasets, and Joona Yoon for clustering datasets on Kaggle.

Bibliography

[ABKS99] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[AFHR21] Wahid Salman Al Farizi, Indriana Hidayah, and Muhammad Nur Rizal. Isolation forest based anomaly detection: A systematic literature review. In *2021 8th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE)*, pages 118–122, 2021.

[AP02] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *European conference on principles of data mining and knowledge discovery*, pages 15–27. Springer, 2002.

[BHM20] Sebastian Buschjäger, Philipp-Jan Honysz, and Katharina Morik. Randomized outlier detection with trees. *International Journal of Data Science and Analytics*, pages 1–14, 2020.

[BKNS00] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[BTA+18] Tharindu R Bandaragoda, Kai Ming Ting, David Albrecht, Fei Tony Liu, Ye Zhu, and Jonathan R Wells. Isolation-based anomaly detection using nearest-neighbor ensembles. *Computational Intelligence*, 34(4):968–998, 2018.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[EKS+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[GD12] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 9, 2012.

[GSW19] Parikshit Gopalan, Vatsal Sharan, and Udi Wieder. Pidforest: anomaly detection via partial identification. *Advances in Neural Information Processing Systems*, 32, 2019.

[HHH+22] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[HKB19] Sahand Hariri, Matias Carrasco Kind, and Robert J Brunner. Extended isolation forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–1489, 2019.

[HMdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J.van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[IH93] B. Iglewicz and D.C. Hoaglin. *How to Detect and Handle Outliers*. ASQC basic references in quality control. ASQC Quality Press, 1993.

[KE11] Nikolaos Kouiroukidis and Georgios Evangelidis. The effects of dimensionality curse in high dimensional knn search. In *2011 15th Panhellenic Conference on Informatics*, pages 41–45. IEEE, 2011.

[KKPC21] Paweł Karczmarek, Adam Kiersztyn, Witold Pedrycz, and Dariusz Czerwiński. Fuzzy c-means-based isolation forest. *Applied Soft Computing*, 106:107354, 2021.

[LBST21] Julien Lesouple, Cédric Baudoin, Marc Spigai, and Jean-Yves Tourneret. Generalized isolation forest for anomaly detection. *Pattern Recognition Letters*, 149:109–119, 2021.

[LTZ10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. On detecting clustered anomalies using sciforest. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–290. Springer, 2010.

[LTZ12] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012.

[Mac67] I MacQueen. Some methods for classifiction and analysis of multivariate observations. In *Proceedings 5th Berkeley Symposium on Mathematical Statistics Problems*, pages 281–297, 1967.

[MdLN22] Caio Melquiades and Fernando Buarque de Lima Neto. Isolation forest-based semi-supervised anomaly detection of multiple classes. In *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE, 2022.

[Nar18] Sarang Narkhede. Understanding auc-roc curve. *Towards Data Science*, 26(1):220–227, 2018.

[NM19] Trong Nguyen Nguyen and Jean Meunier. Hybrid deep network for anomaly detection. *arXiv preprint arXiv:1908.06347*, 2019.

[PTA15] Guansong Pang, Kai Ming Ting, and David Albrecht. Lesinn: Detecting anomalies by identifying least similar nearest neighbours. In *2015 IEEE international conference on data mining workshop (ICDMW)*, pages 623–630. IEEE, 2015.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Ray16] Shebuti Rayana. ODDS library, 2016.

[RVG+18] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[SCSC03] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, 2003.

[SR15] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[TK22] Mikhail Tokovarov and Paweł Karczmarek. A probabilistic generalization of isolation forest. *Information Sciences*, 584:433–449, 2022.

[WBH19] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. Progress in outlier detection techniques: A survey. *Ieee Access*, 7:107964–108000, 2019.

[XPWW22] Hongzuo Xu, Guansong Pang, Yijie Wang, and Yongjun Wang. Deep isolation forest for anomaly detection. *arXiv preprint arXiv:2206.06602*, 2022.

[Yoo22] Joonas Yoon. cluster-paint, 2022.

[YXX+21] J. Yuan, S. Xiang, J. Xia, L. Yu, and S. Liu. Evaluation of sampling methods for scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 27(1):1–1, 2021.

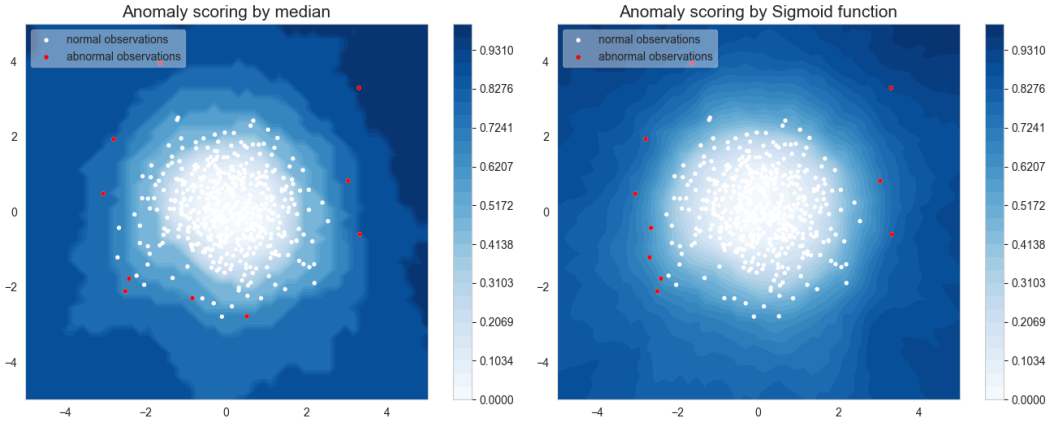
[ZNL19] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

[ZRF+18] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*, pages 727–736. IEEE, 2018.

Appendix

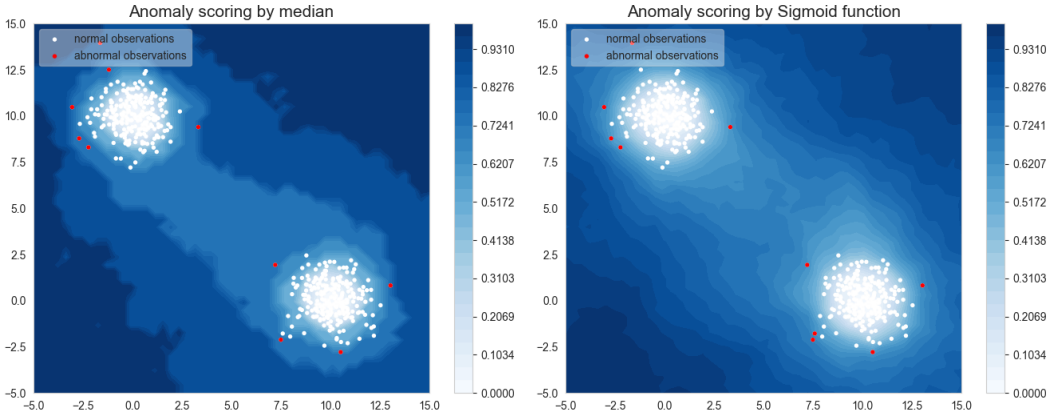
This appendix displays all the results of experiment that does not listed in the body oh thesis. First, for the adjustments of EIF, the results of AUC-ROC and AUC-PRC on real-world datasets, and also the score map of both methods on synthetic datasets are illustrated. Second, for the merging of cluster-wise score maps, figures of all the n-norm methods, the minimum method and the adjusted EIF based geometric mean method are shown in this chapter.

Results of adjusting anomaly scoring



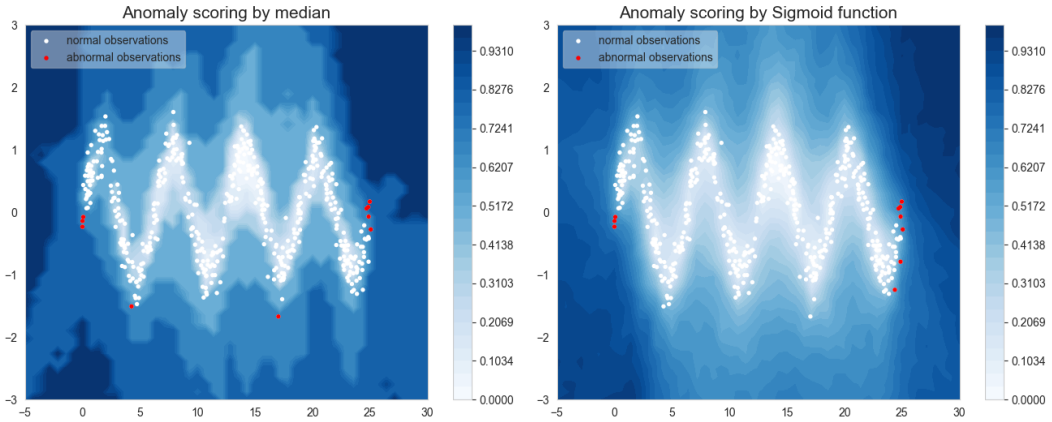
(a) Median (b) Sigmoid-based function

Figure 6.1: Comparison of score map and outliers of median scoring and sigmoid-based function scoring for single blob dataset



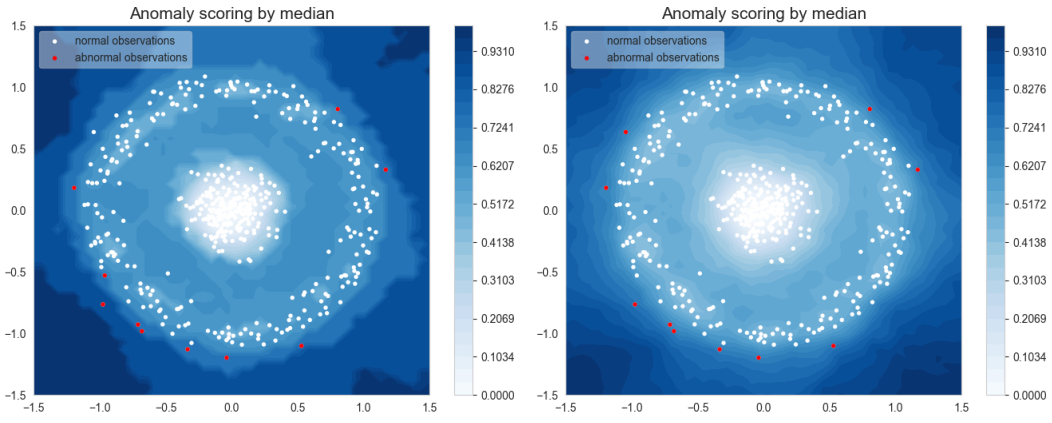
(a) Median (b) Sigmoid-based function

Figure 6.2: Comparison of score map and outliers of median scoring and sigmoid-based function scoring for double blobs dataset



(a) Median (b) Sigmoid-based function

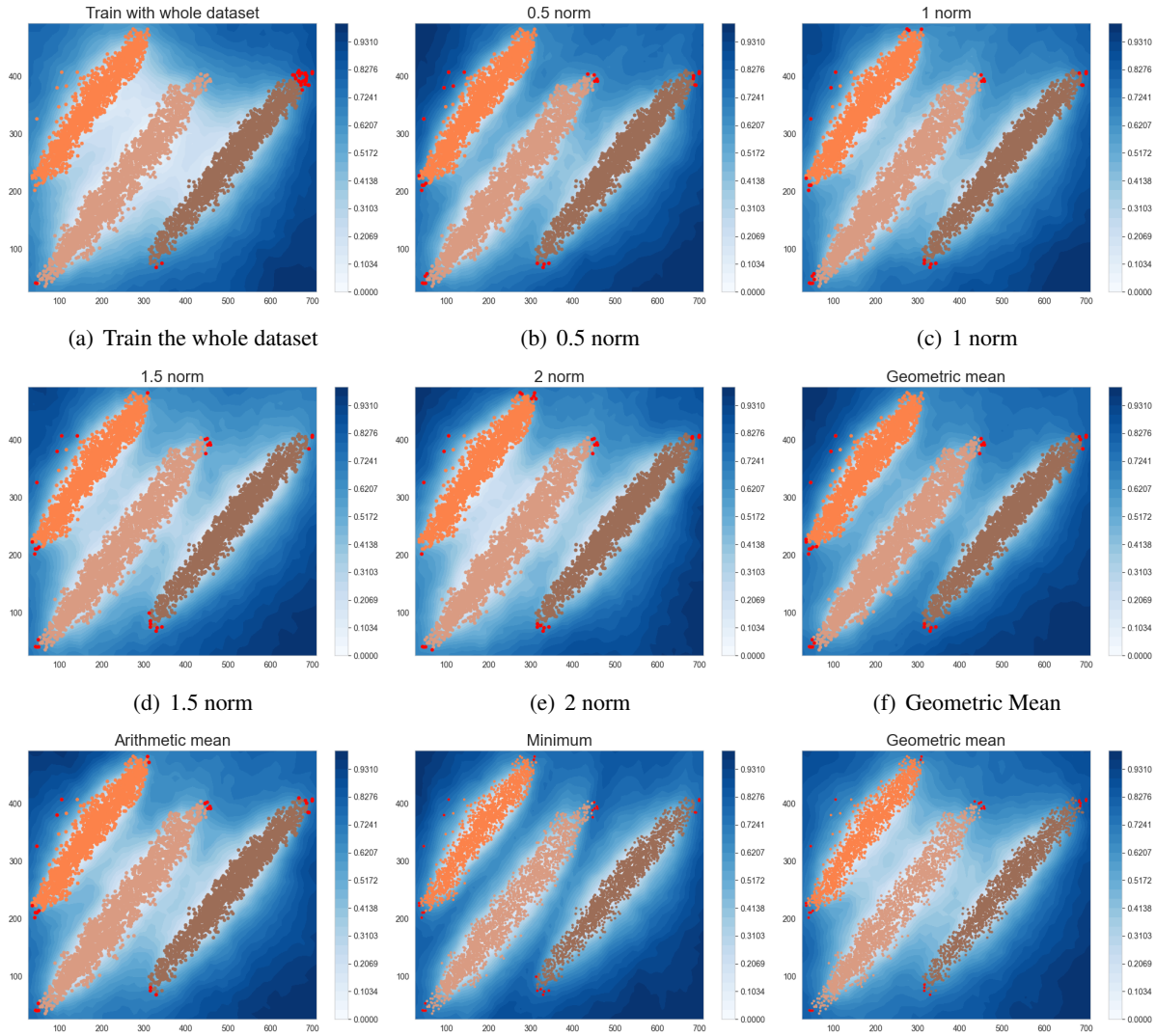
Figure 6.3: Comparison of score map and outliers of median scoring and sigmoid-based function scoring for sinusoidal dataset



(a) Median (b) Sigmoid-based function

Figure 6.4: Comparison of score map and outliers of median scoring and sigmoid-based function scoring for circle around single blob dataset

Results of proposed Outlier detectors for multivariate datasets

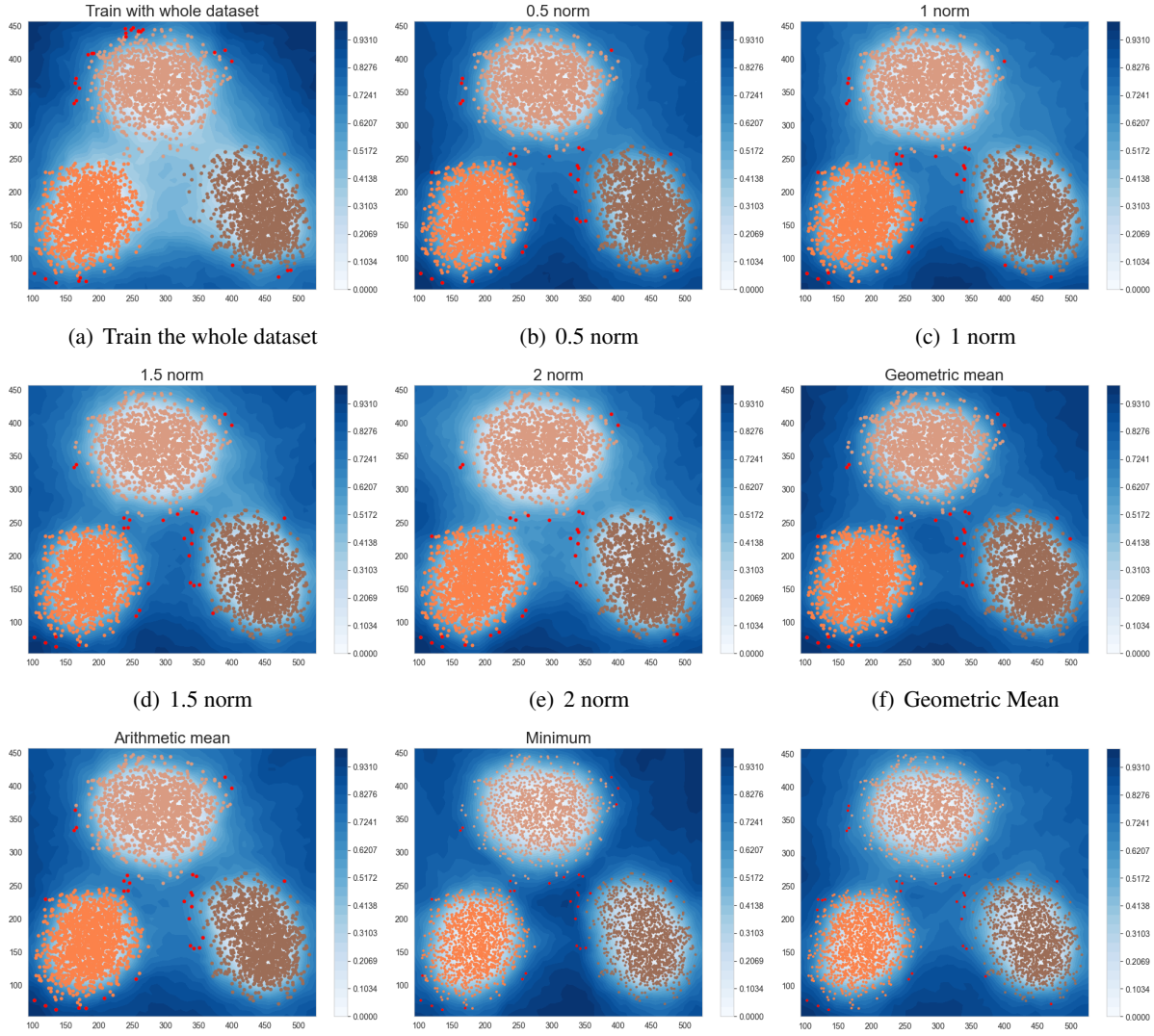


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.5: Comparison of score map and outliers of all proposed cluster-wise detection methods for three linear classes

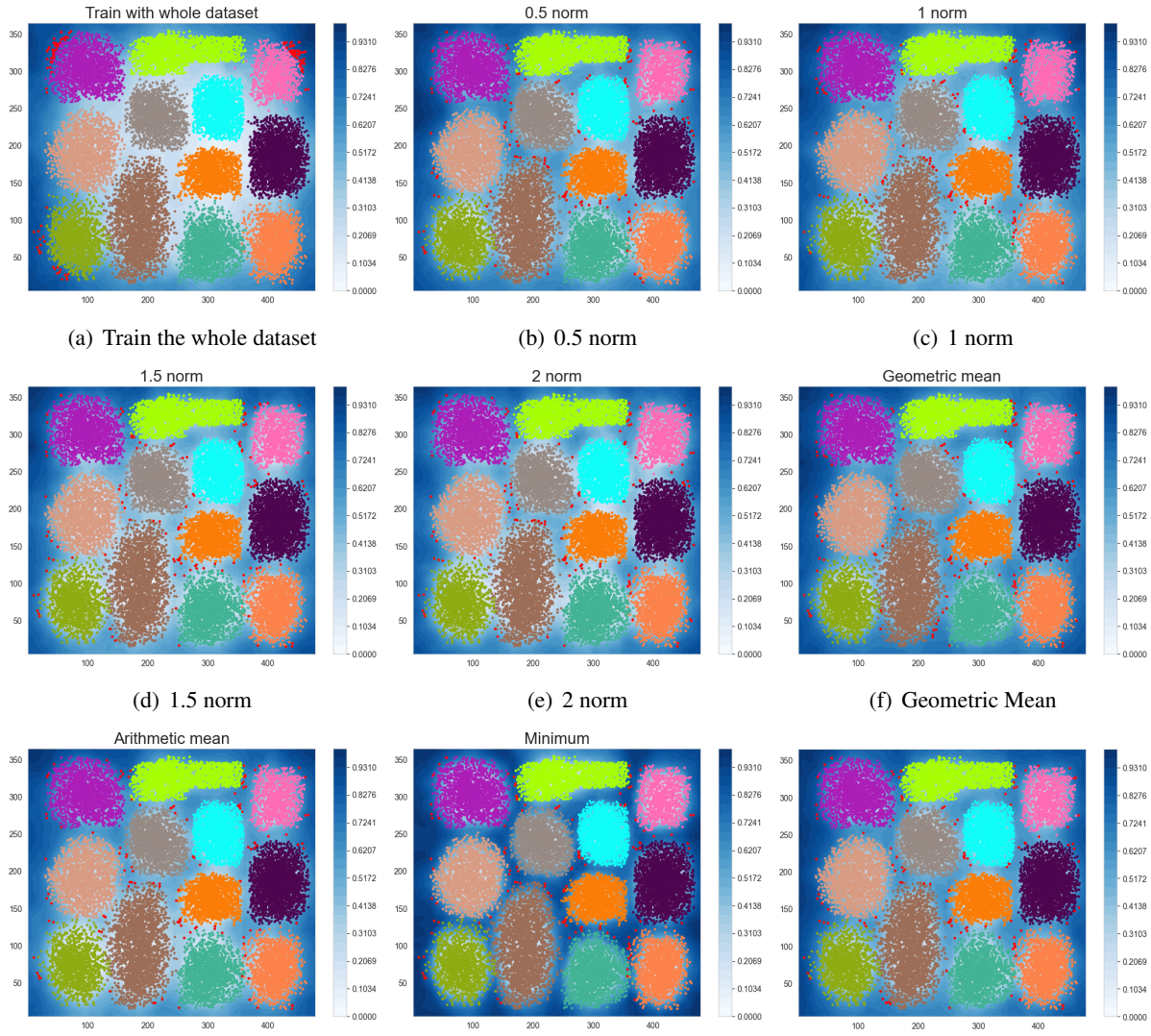


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.6: Comparison of score map and outliers of all proposed cluster-wise detection methods for the three balanced blobs dataset

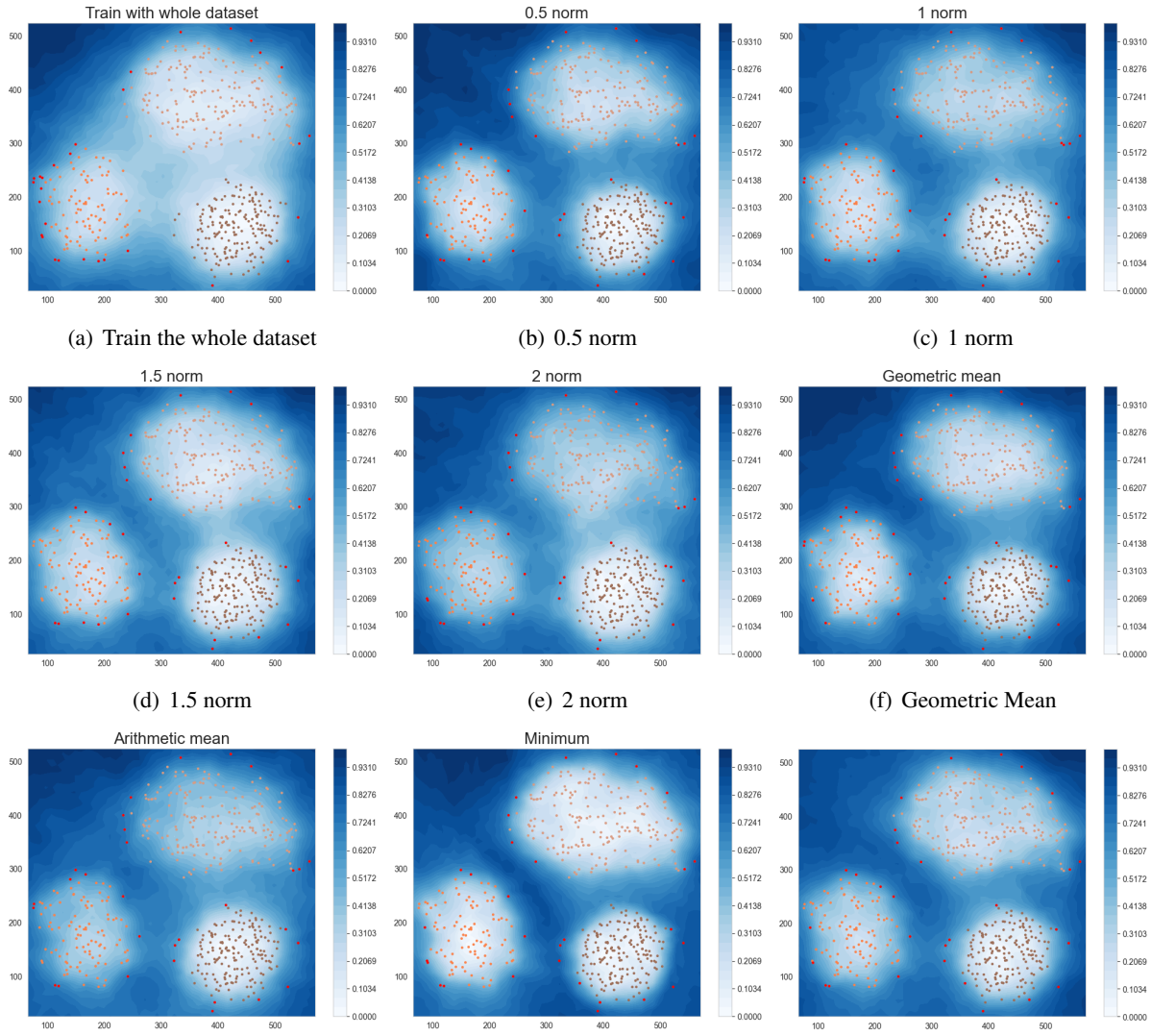


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.7: Comparison of score map and outliers of all proposed cluster-wise detection methods for the 12 imbalanced classes dataset

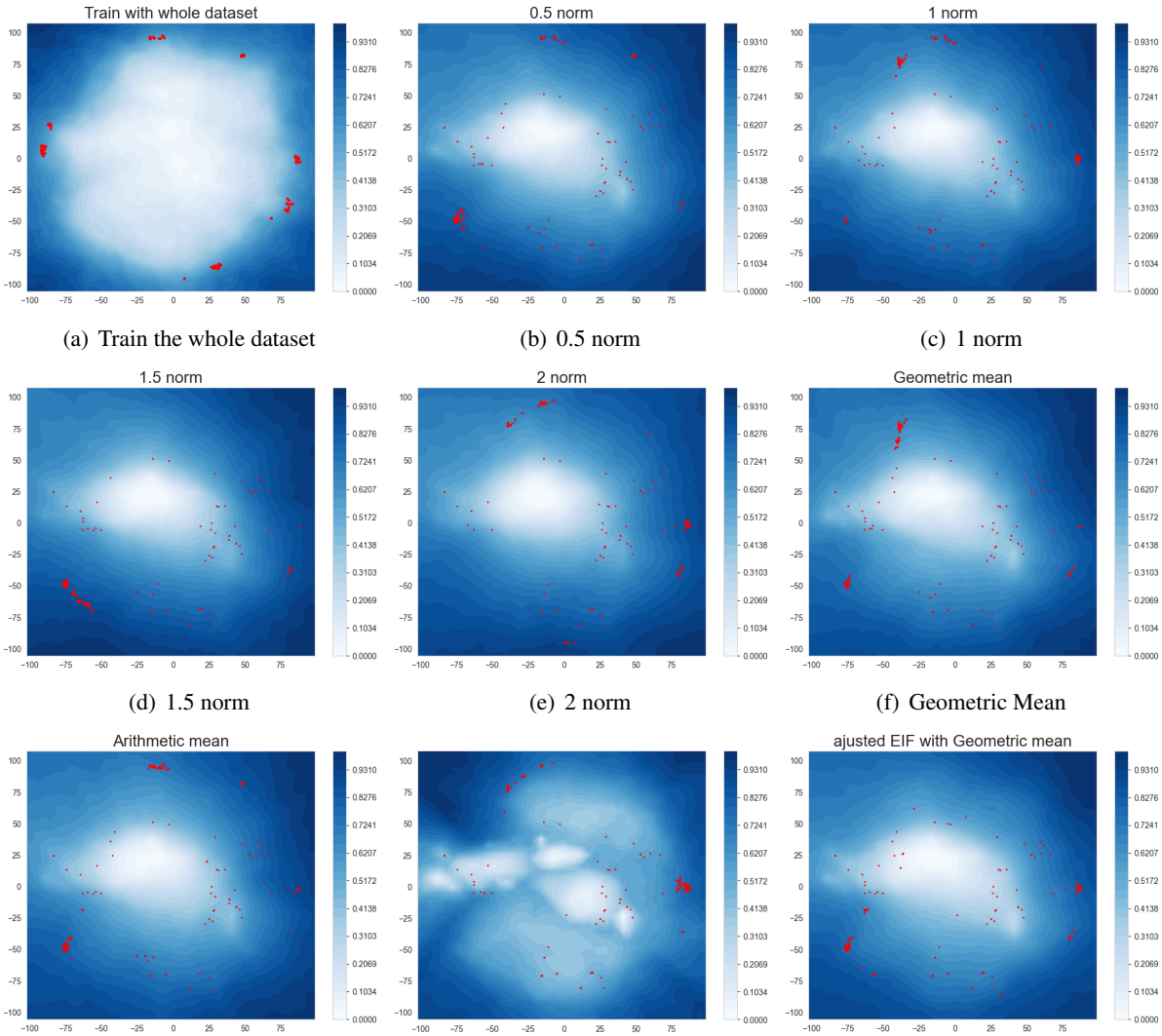


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.8: Comparison of score map and outliers of all proposed cluster-wise detection methods for the three sparse blobs dataset

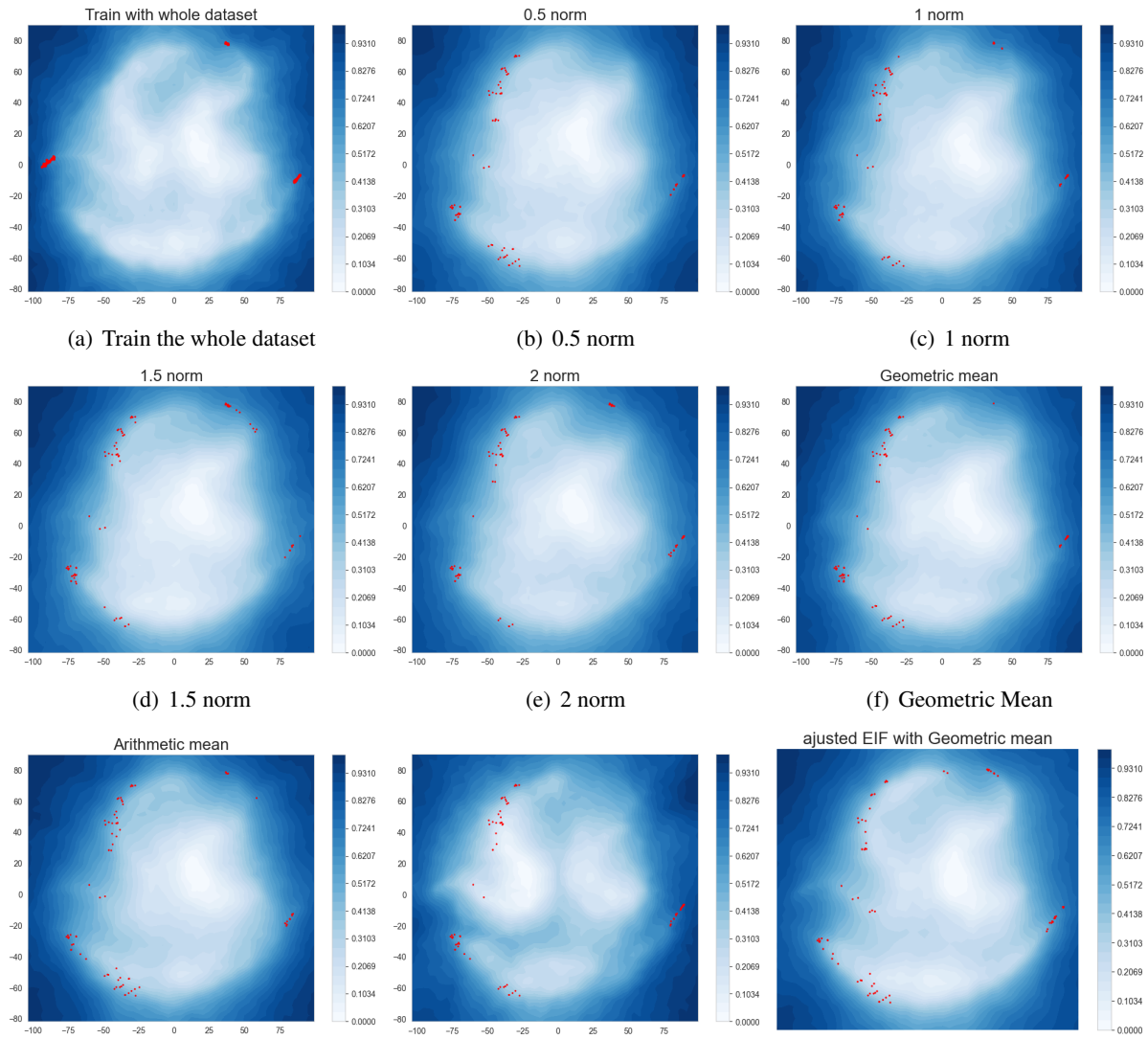


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.9: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Crowdsourced_mapping dataset

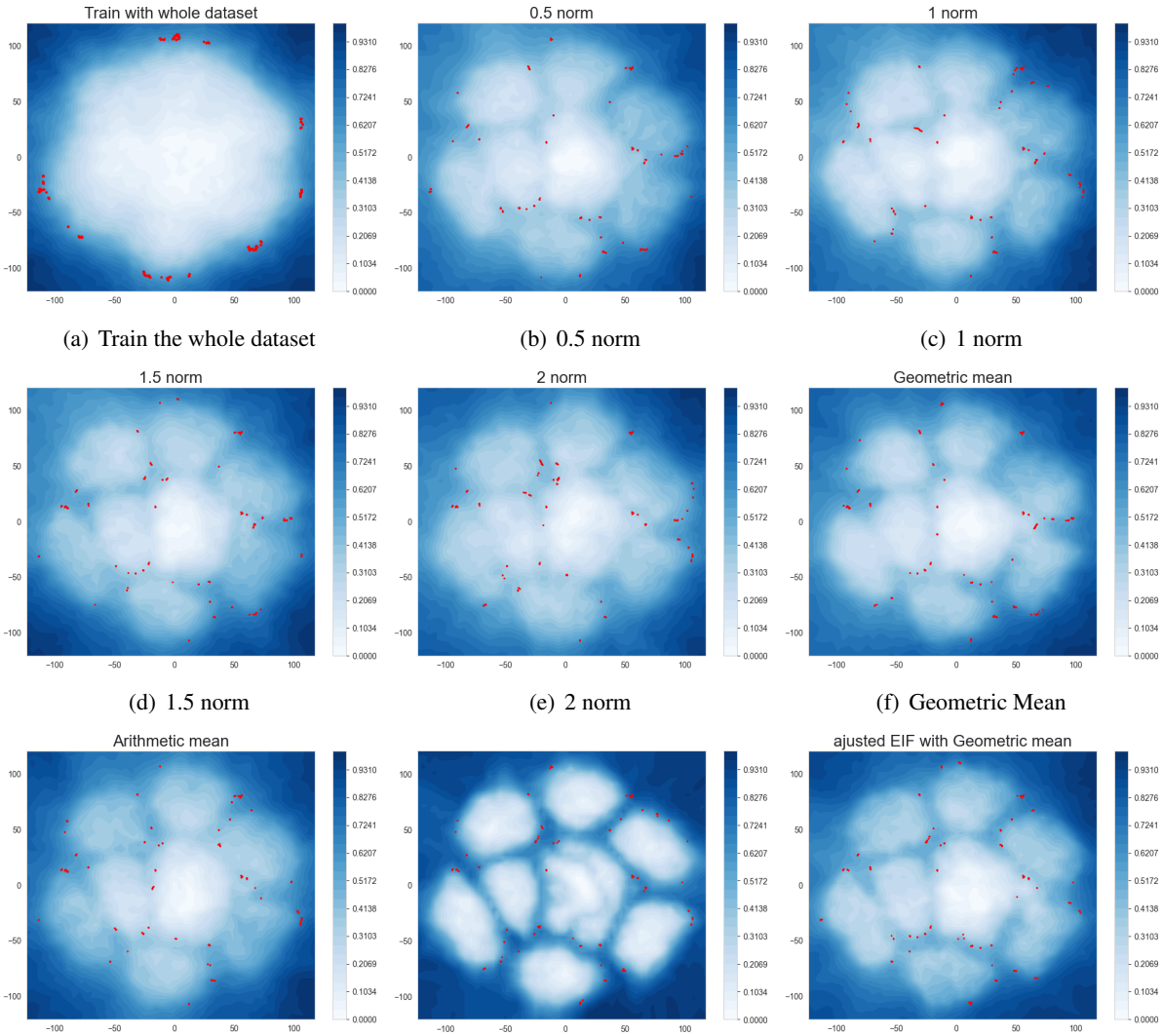


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.10: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Abalone dataset

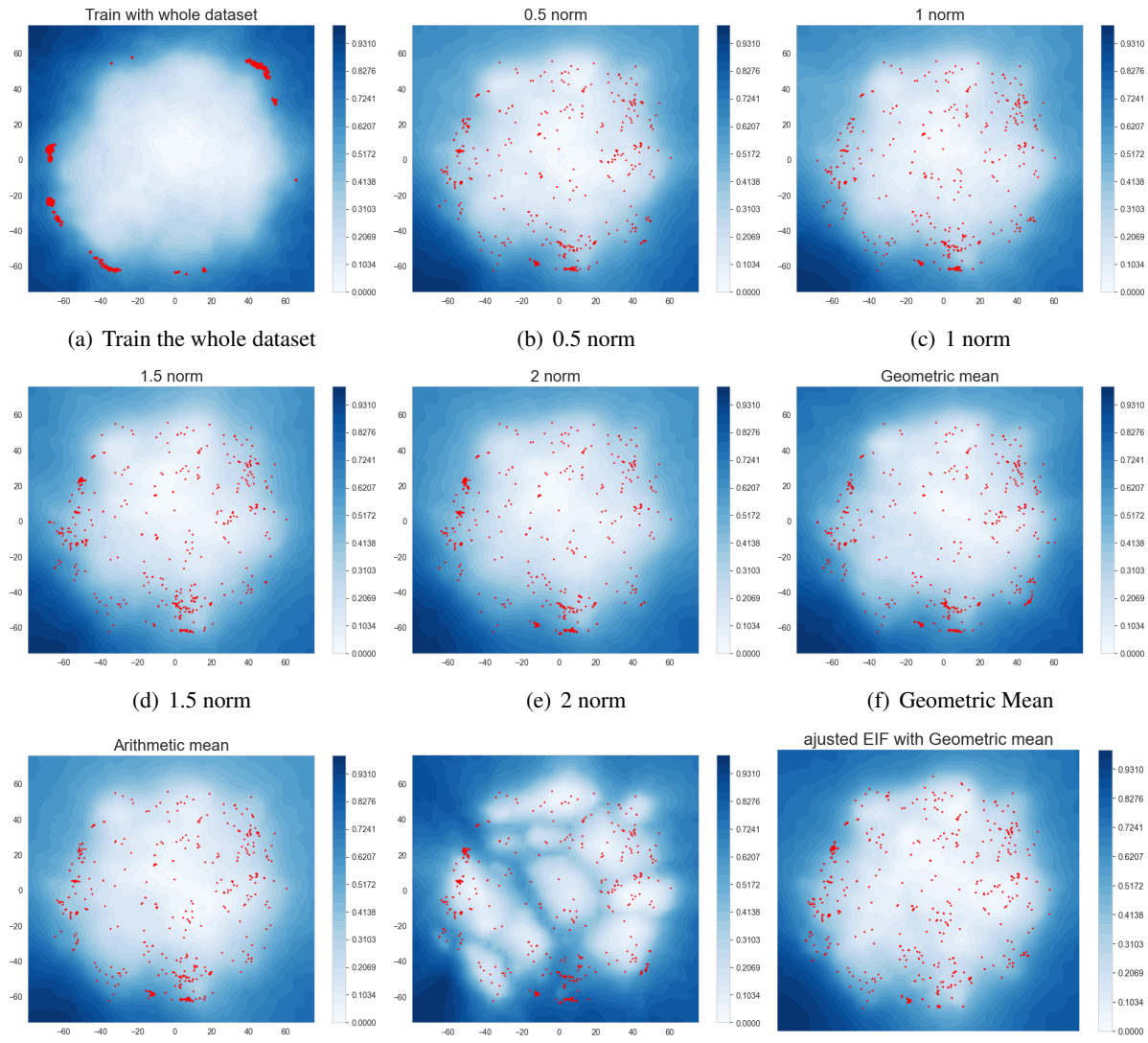


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.11: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Condition_based_maintenance dataset

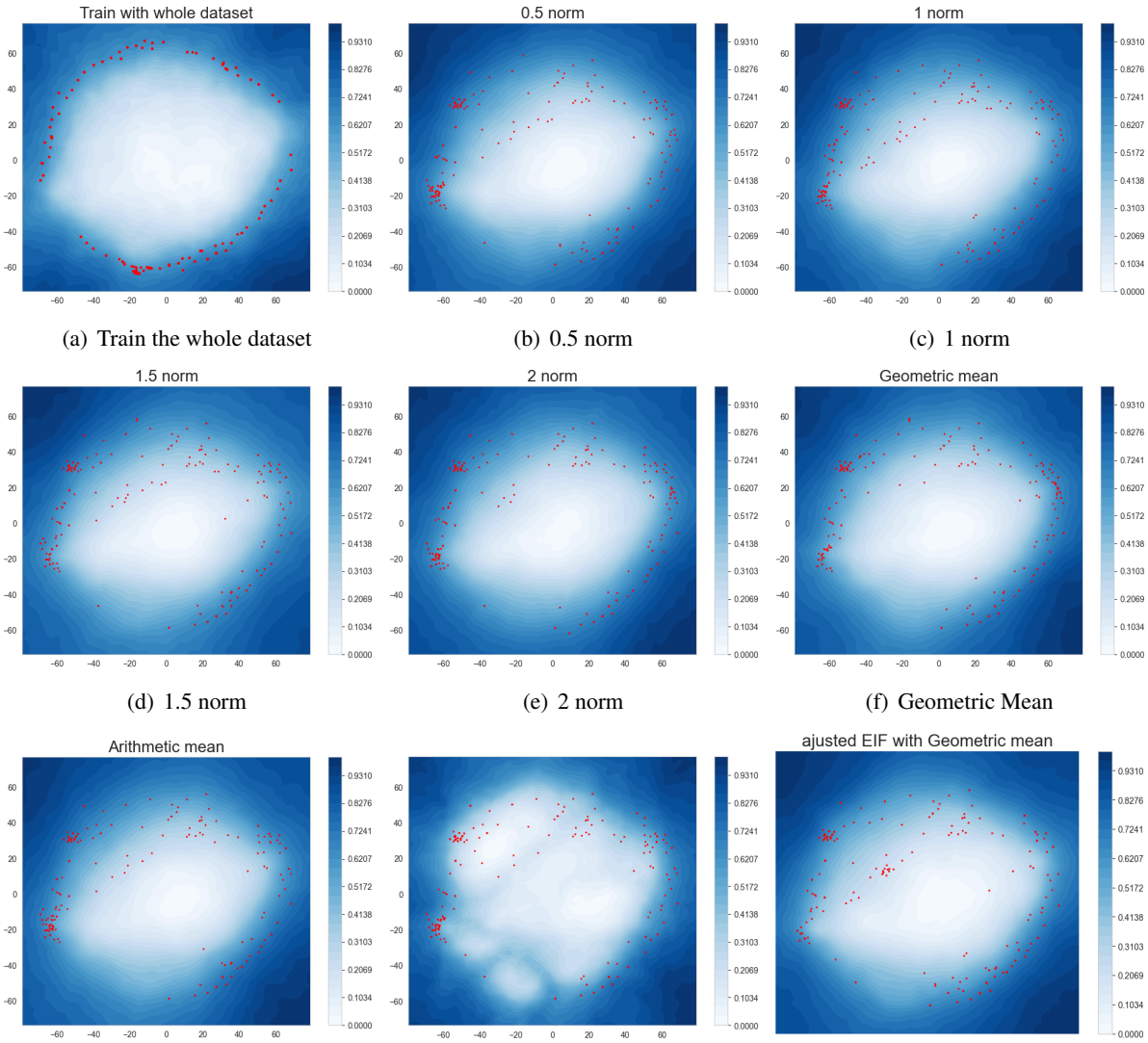


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.12: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Clothes dataset

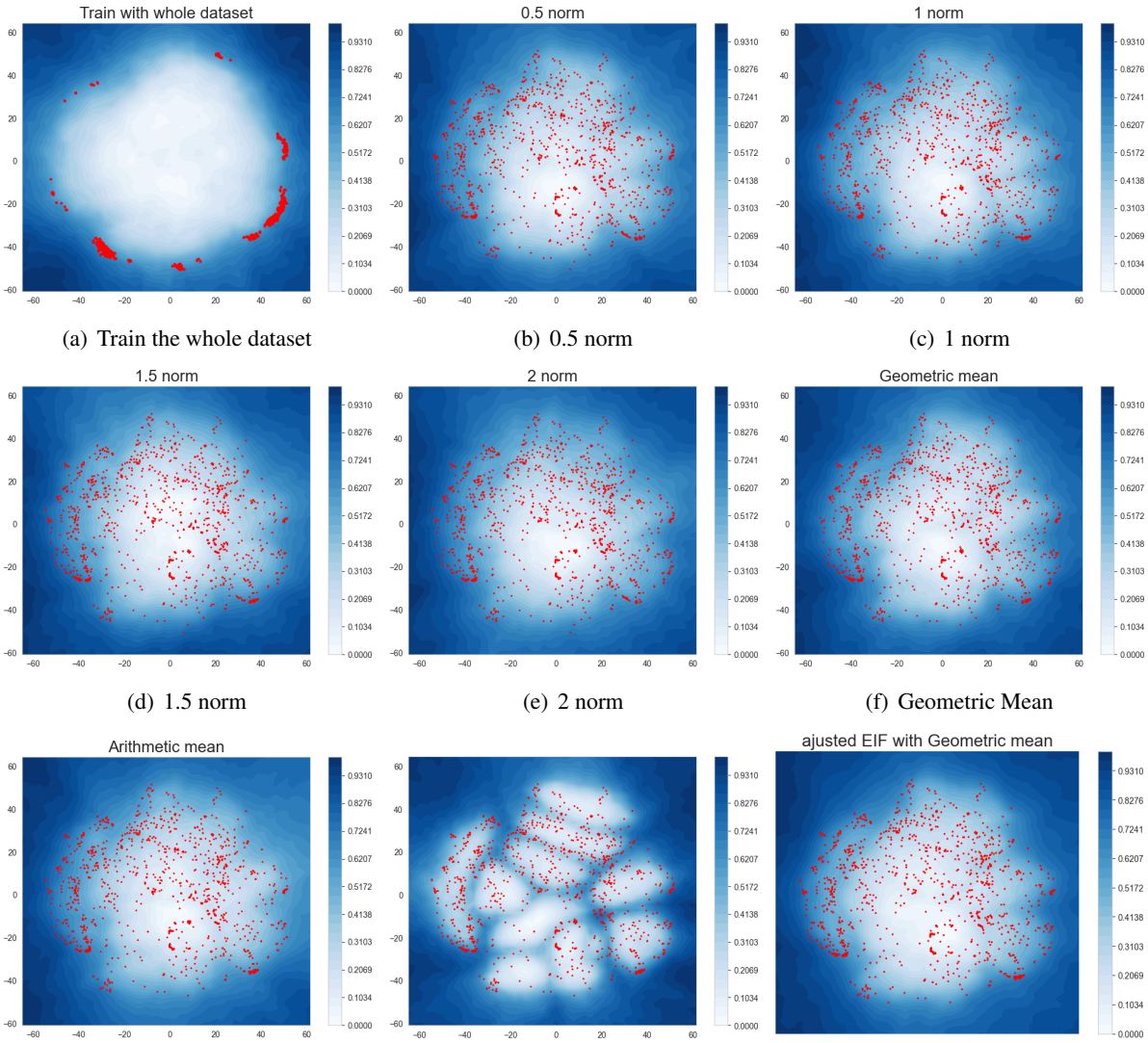


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.13: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Epilepic_seizure dataset

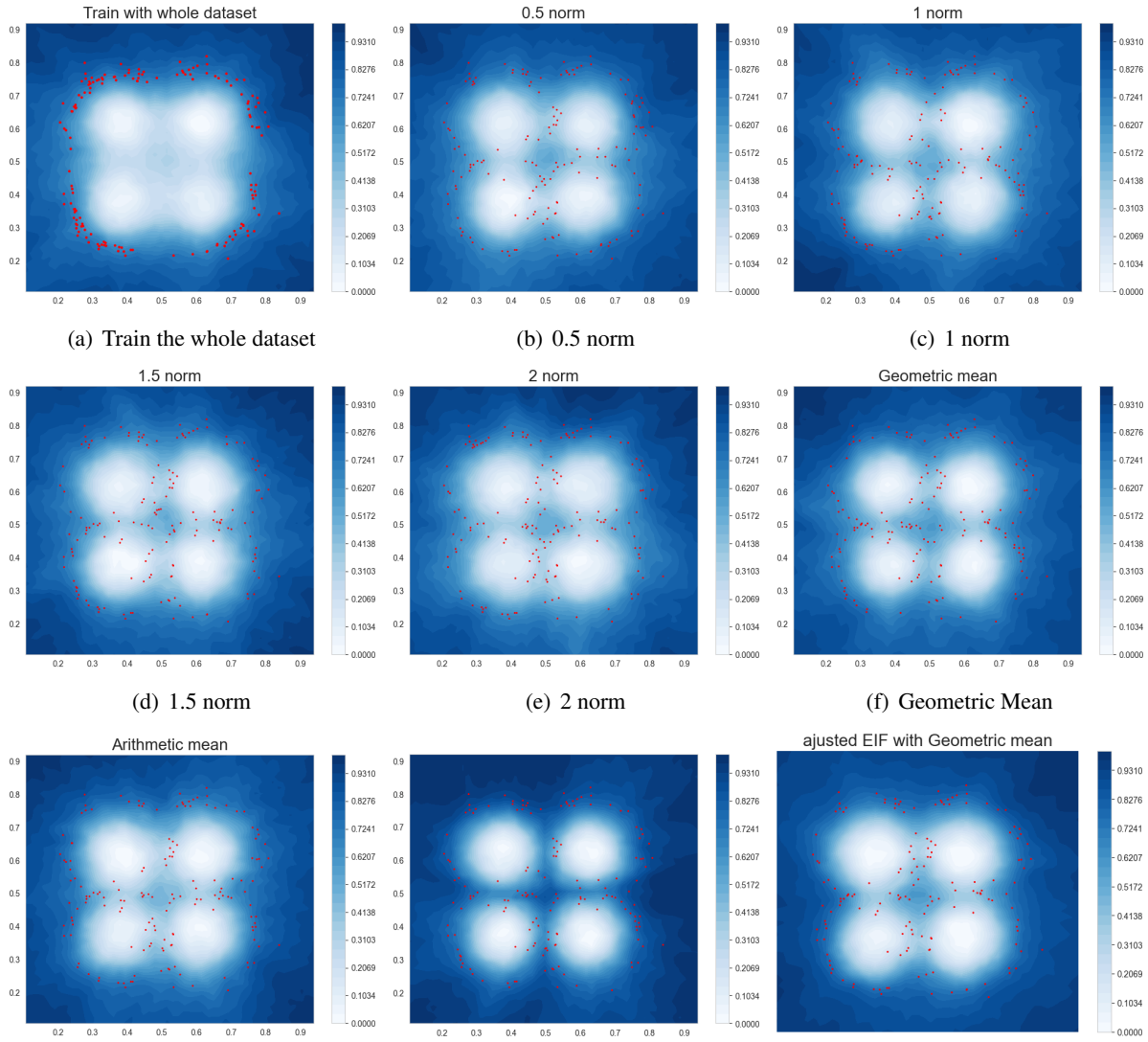


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.14: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Mnist dataset

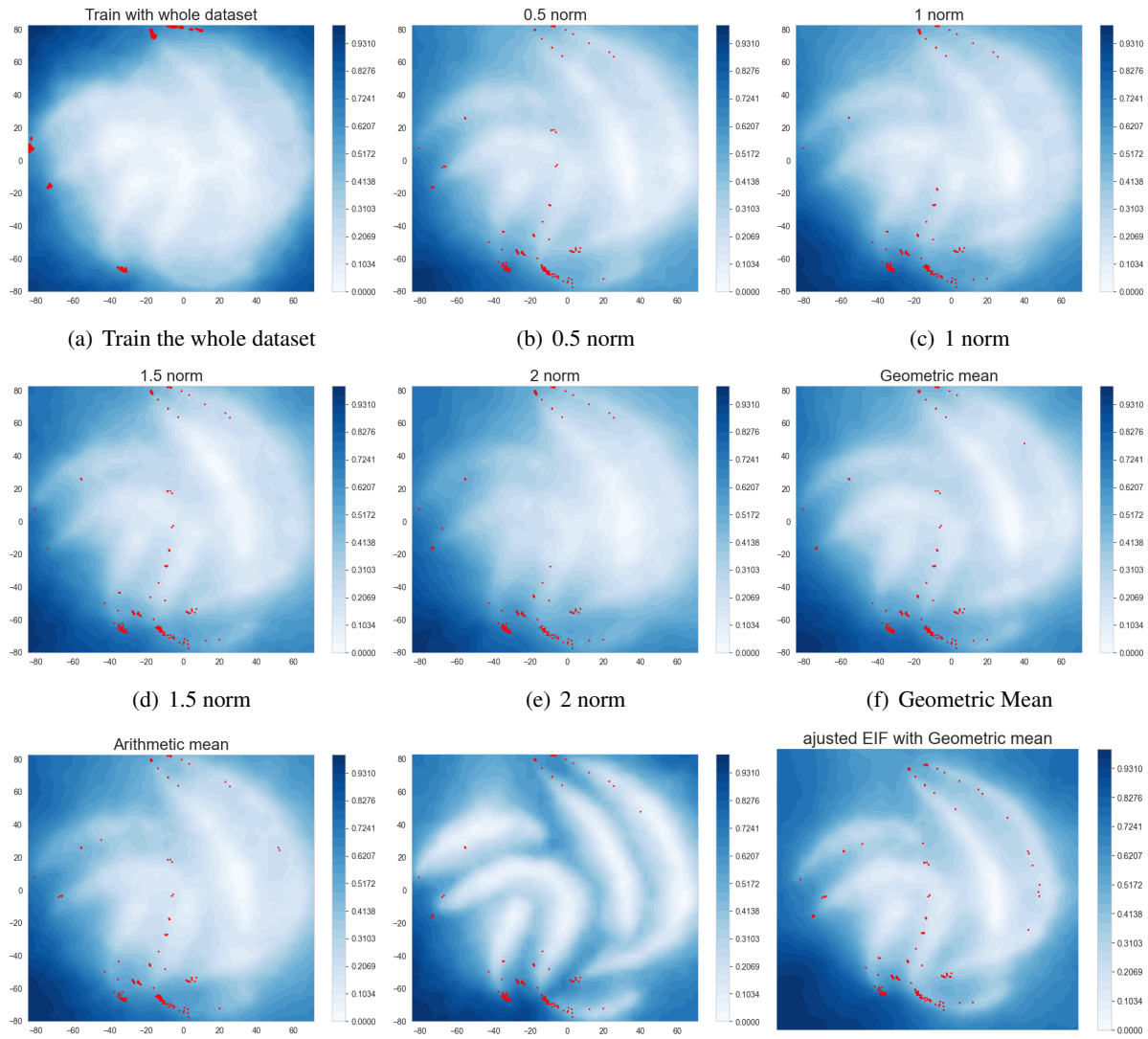


(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.15: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Swiss_roll_2d dataset



(a) Train the whole dataset (b) 0.5 norm (c) 1 norm

(d) 1.5 norm (e) 2 norm (f) Geometric Mean

(g) Arithmetic Mean (h) Minimum (i) adjusted EIF based Geometric Mean

Figure 6.16: Comparison of score map and outliers of all proposed cluster-wise detection methods for the Swiss_roll_3d dataset