FEDERATED LEARNING SVDD FOR OUTLIER DETECTION

Massimo Frasson
Università degli Studi di Milano
Milano
massimo.frasson@studenti.unimi.it

ABSTRACT

As the number of sensors in our society continues to grow, monitoring the functioning of various systems, from industrial equipment to human well-being has become increasingly common. An important aspect of this monitoring is the ability to detect anomalies or novel information, which can provide valuable insights or serve as early warning signs in the event of an incident.

Support Vector Machines (SVMs) are a well-established technique for the task being resistant to the class imbalance, common in anomaly detection. Nevertheless, like other traditional anomaly detection algorithms it requires centralizing data for training, which poses privacy concerns. Federated Learning techniques, on the other hand, enable the training of models where the data is generated, thereby reducing privacy risks.

This work presents a novel approach, called Federated Learning SVDD (FLSVDD), which applies the principles of Federated Learning (FL) to address the Outlier Detection problem using SVMs. By leveraging the benefits of FL, FLSVDD offers a privacy-preserving solution while achieving high accuracy in detecting anomalies. Through experimental evaluation, we demonstrate the effectiveness of FLSVDD in detecting anomalies in a federated setting, highlighting its potential for real-world applications where data privacy is a critical concern.

***K*eywords** Federated Learning · Support Vectors · Outlier Detection

1 Introduction

In the earliest days, detecting outliers was motivated by data cleansing: removing outliers from the dataset so that parametric statistical models could fit the training data more smoothly. Soon more attention has been turned toward outliers themselves as outliers often represent interesting and critical information, e.g., cyber-attacks in a network, mechanical faults caused by defective industrial equipment, and so on. [1]

According to [2], outliers have the following characteristics:

- Outliers are different from the norm with respect to their features and
- They are rare in a dataset compared to normal instances.

Since the data about anomalies is scarce and often unrepresentative of a system's possible anomalies, most of the research focus has been on unsupervised machine learning techniques. Many algorithms and approaches have been proposed, as explained in [1] and [2]. In particular, the fundamental SVMs approaches are the ones presented in [3] and [4]. The core idea of the SVM approach is to create a boundary that surrounds the normal data, the former contribution uses a hyperplane while the latter uses a hyper-sphere. One key advantage of SVMs is that they require less parameter tuning than neighbor-based approaches and are less sensitive to training set data density.

Moreover, thanks to the adoption of various kernels via Mercer's theorem [5], SVM-based techniques can create boundaries of any shape, not limited to ellipsoids or other standard geometrical shapes. In this work, [3] will serve as one of the baselines, while [4] will form the basis for the new federated approach. However, the parameter nomenclature for SVDD [4] will be derived from [6], which offers a clear introduction to the SVDD math.

However, the aforementioned SVMs techniques require centralizing the data for algorithm training, raising serious privacy concerns. Federated Learning offers a solution to this issue, enabling training to occur on edge devices such as smartphones and IoT devices.[7] The model update is the only information sent to the server, providing clear privacy benefits, and distributed computation is a byproduct. The primary focus of this work is to introduce and assess Federated Learning primitives for SVDD. To achieve this, a custom implementation of SVDD was developed. The evaluation of the custom implementation of SVDD, as well as its Federated Learning adaptation and OneClassSVM by Sklearn [3], were performed using the same methodology as presented in [2] to ensure that the results are comparable.

2 Related Works

The field of anomaly detection has gathered significant attention from the computer science community over the past few decades. In this paper, we provide an overview of the notable algorithms proposed in Section 2.1, shedding light on their key contributions. Additionally, in Section 2.2, we introduce the concept of Federated Learning, exploring the foundational research in this area and examining the recent endeavors made to address anomaly detection within this framework.

2.1 Outlier Detection

Outlier detection, also known as anomaly detection, is a well-studied problem in machine learning and data mining. The goal of outlier detection is to identify data points that are significantly different from the majority of the data, which may indicate errors, fraud, or novel phenomena. In recent years, outlier detection has gained importance in many fields such as finance, security, healthcare, and social media analysis.

One comprehensive survey of outlier detection methodologies is presented in the work by [2]. It provides an overview of various outlier detection techniques, including statistical methods, proximity-based methods, and clustering-based methods. It also discusses the challenges and evaluation criteria for outlier detection, such as scalability, interpretability, and robustness.

One of the fundamental algorithms, evaluated in [2], is Local Outlier Factor (LOF). LOF, proposed by [8], calculates the ratio of the average density of a point's k nearest neighbors to the point's own density. Points with a significantly lower LOF score than their neighbors are considered local outliers. Their work provides a thorough evaluation of the LOF technique on various synthetic and real-world datasets, demonstrating its effectiveness in identifying local outliers. However, the algorithm is very sensitive to parameter selection and can be too computationally expensive on large datasets [2].

Another fundamental algorithm is CBLOF, which was introduced to outperform LOF [9]. CBLOF identifies outliers based on their deviation from the local cluster density. It first partitions the dataset into clusters and then calculates the outlier score for each point. The score is based on the point's distance from the centroid of the cluster it belongs to and the average distance of the points within the same cluster. However, like LOF, the algorithm is sensitive to parameter selection and may be computationally expensive for large datasets [2].

A statistical approach is HBOS by [10]. The HBOS algorithm uses a histogram-based approach to model the distribution of each feature in the dataset and combines the scores of individual features to obtain the final outlier score for each data point. The authors provide a thorough evaluation of the HBOS on various synthetic and real-world datasets, demonstrating its effectiveness in identifying global outliers. The technique is shown to outperform existing outlier detection methods such as LOF and k-nearest neighbor (k-NN) in terms of both speed and accuracy. However, the algorithm assumes that the features are independent and identically distributed, which may not hold true for all datasets [10].

Support Vector Machines (SVM) based techniques have indeed been proposed to address the problem of Outlier Detection. Two of the main techniques in this area are ocSVM, introduced by [3], and SVDD (Support Vector Data Description), proposed in [4]

ocSVM focuses on constructing a hyperplane with the maximum distance from the normal data and the outliers. The idea is to find a decision boundary that separates the normal data from the outliers with the largest margin possible. By doing so, it aims to identify the region where outliers lie.

On the other hand, SVDD aims to build the minimum volume hypersphere that encloses the normal data points. The goal is to find a hypersphere that tightly fits the normal data instances while excluding the outliers. The size of the hypersphere is minimized, making it robust against outliers.

SVDD has demonstrated generally comparable results to Nearest Neighbor (NN) techniques. However, the authors have shown an advantage over NN in scenarios involving sparse or complex datasets.

The work on Federated Learning SVDD (FLSVDD) builds upon the foundation established by [4]. However, the notation used in FLSVDD is taken from [6], which explains the concepts and techniques related to SVDD in a clear manner.

Despite the potential of the above techniques for outlier detection, they may face practical challenges in distributed and privacy-sensitive settings. In many real-world scenarios, the data is scattered across multiple sources or entities, and it may not be feasible or desirable to centralize the training data. The main contribution of this work is to make the SVDD algorithm privacy-sensitive and distributed, thus eliminating the need for centralizing the training data and reducing the risk of privacy breaches.

2.2 Federated Learning

The aim of this work is to conduct outlier detection within a federated learning framework, prioritizing both data privacy and distributed computation. Federated Learning, which was first introduced in [7], is a method for training deep neural networks using data stored on various devices without centralization. The method includes iterative training of local models on each device, followed by aggregation of model updates on a central server. The method's comparable accuracy to centralized training methods has significant implications in situations where data privacy or computational constraints make centralizing data inconvenient.

Recent works on federated outlier detection, including [11] and [12], focus solely on neural network models and exclude other techniques. In contrast, this work incorporates SVDD (Support Vector Data Description) into federated learning, providing an alternative tool for outlier detection.

In summary, our work contributes to the field of federated learning by presenting a new approach for outlier detection that complements current methods based on neural networks. This approach could be highly beneficial in situations where neural networks are not optimal or when interpretability and computational efficiency are crucial factors.

3 Model

From [7] we can infer a meta-algorithm common to all the Federated Learning approaches.

Algorithm 1 Federated Learning training meta-algorithm

$g \leftarrow$ Init()
for each round $t \leftarrow 1$ to L **do**
  $m \leftarrow max(C * K, 1)$
  $S_t \leftarrow$ (m random clients)
  $u \leftarrow \emptyset$
  for each client $k \in S_t$ **in parallel do**
    $u_k \leftarrow$ ClientUpdate(k, g)
  end for
  $g \leftarrow$ GlobalCombine(u, g)
end for

The Federated Learning process involves a central server that coordinates the algorithm, and K clients. At each round t, a fraction C of random clients S_t is selected from the total set of K clients.

For each selected client $k \in S_t$, the central server sends the previous round's global model g to update the client's local model. The client then computes its update by training on its own data, starting from g. Once the client finishes training, its model update is sent to the central server, which combines the updates from all participating clients with its global model to generate a new global model.

This process continues until the total number of rounds L is reached.

3.1 Global Model Initialization

The SVDD algorithm, explained in [6], is used to compute the minimum volume hypersphere enclosing all the normal data, i.e., not outliers. The distance from a new data point to the center of the hypersphere is used to determine whether it is an outlier or not. The hypersphere is built using the slackness coefficient C, the Gaussian kernel size q, and a set

of normal data points X. The chosen support vectors $xs = x_0, x_1, ...x_i \in X$ and their Lagrange multipliers β are obtained after training, and with these values, we can describe the hyper sphere and compute the distance of any point from its center.

To adapt SVDD for federated learning, we propose an initialization primitive that starts without any data, support vectors, or Lagrange multipliers. The slackness parameter C is set to its maximum value of 1 to create a stricter representation of the data initially, and it is loosened only after the model's performance benefits from it. The kernel width parameter is set by the user as it strongly depends on the data being treated. Strategies to initialize and improve q at each iteration, as explained in [6], are to be explored in future work.

From now on the hyper-sphere will be referred as model $g = (q, C, xs, \beta)$.

3.2 Client Update

For each iteration t, each client $k \in S_t$ simultaneously chooses some of their data points as support vectors for the global model. Given the global model g, each client trains an SVDD model with the same q and C as in g. The training data given to SVDD is a concatenation between the support vectors xs coming from g and a batch B of the client's own data.

At the end of the procedure, all the elected support vectors and bounded support vectors are sent to the central server to contribute to the improvement of the global model.

The clients' batch size is a hyper-parameter of FLSVDD. The clients never use the same data twice, so an higher batch size means fewer rounds in which they can participate.

3.3 Global Combine

The global combine is in charge of updating the global model g at each iteration t. The first step is concatenating all the candidate support vectors coming from the clients' updates together with the xs in g. Then, they are marked as a normal class if their Lagrange multiplier is less or equal to C_g. Once built this new dataset X_t, the procedure trains an SVDD model over it. The goal is to build a new global model g, keeping just enough support vectors among all the candidates coming from the previous g and the updates. However, during the training, the procedure tries to improve the hyper-parameters q and C by randomly testing other values. At the end of the procedure, g is updated with the new C, the new q, and the elected support vectors and multipliers.

4 Experiments

The experiments conducted in this work follow the methodology proposed in [2] to ensure comparability with their results.

The evaluation of the models is done in an unsupervised manner, with no separation between training and testing data. Each record in the dataset is assigned a score by the models, with higher scores indicating a greater likelihood of the point being an outlier. The final classification is determined by a threshold, with points above the threshold being classified as outliers.

4.1 Dataset

The "Pen-Based Recognition of Handwritten Text (global)" UCI dataset[13] contains the hand-written digits 0–9 from 45 different writers. The authors only keep the digit 8 as the normal class and sample the 10 digits from the remaining classes as anomalies. It results in one big normal cluster and global anomalies sparsely distributed. The resulting pen-global dataset has 16 dimensions and 809 instances including a large number of anomalies (11.1%). [2]

As in [2], the dataset features are normalized through min-max, where every feature is scaled to the interval [0, 1].

4.2 Evaluation metrics

The models generate a score representing the degree of "outlierness" for each data point, and the classification of the data point is based on a threshold. However, selecting an appropriate threshold without prior knowledge is challenging. To address this issue and determine the best threshold, the Receiver Operating Characteristic (ROC) is used, which shows the relationship between True Positive Rate (TPR) and False Positive Rate (FPR) at various thresholds. The Area

Under the ROC Curve (AUC) is used as a metric to facilitate comparisons between models, as it is a standard measure for anomaly detection, as explained in [2].

However, in federated learning, we cannot guarantee that two classes will be available at each stage, making it impossible to calculate ROC-AUC, which requires two classes. Because SVDD allows for one-class training, adding the constraint of two classes would be counterproductive. Thus, in the experiments, both ROC-AUC and Average Precision (AP) are computed for the baseline models, while only AP is computed for the federated learning model.

AP is a suitable alternative to ROC-AUC since it summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight, and is related to AUC [14].

By computing average precision for the baselines, this work allows comparing the Federated Learning SVDD results against the oc-SVM ones presented in [2].

4.3 Baselines

To evaluate the performance of FLSVDD, it is compared with two baseline methods: oc-SVM and SVDD.

The oc-SVM algorithm, evaluated in [2], is a machine learning algorithm based on support vector machines for outlier detection. The authors report a standard deviation for their results, since they tested and aggregated multiple experiments with different hyperparameters. In particular, they sampled η in the interval [0.2, 0.8], the slackness parameter, and adopted an automatic gamma tuning technique to choose the Gaussian kernel width parameter γ. We use the implementation of oc-SVM available in [15] (OneClassSVM) to replicate their results. We sample η in the same interval, and γ in the arbitrary interval [0, 3], since we do not have access to their automatic gamma tuning technique. We collect 100 samples to evaluate OneClassSVM.

Another baseline method we use is SVDD, based on the paper [6]. Like OneClassSVM, SVDD also relies on Support Vector Machines and the Gaussian Kernel to separate normal data from outliers. We implement SVDD ourselves and use the same hyperparameter ranges and nomenclature as in [6]. Specifically, we sample the slackness parameter C in the interval [0.2, 0.8], and the Gaussian kernel width parameter q in the interval [0, 3]. We also collect 100 samples to evaluate SVDD.

To validate the correctness of our SVDD implementation, we compare its performance with OneClassSVM. We expect them to perform similarly since they are based on the same principles and use similar hyperparameters.

4.4 Federated Learning SVDD: IID vs Non-IID

Typically, machine learning algorithms are evaluated on Independent and Identically Distributed (IID) data, which simplifies the modeling process and allows for a wide range of statistical and machine learning algorithms to be utilized. However, in real-world scenarios, data is often not IID, which is especially true for Federated Learning. In Federated Learning, each edge device uses its own collected data to contribute to the learning process. This data can be collected at different times, in different locations, and in different amounts, among other factors.

To evaluate the performance of our Federated Learning SVDD approach in real-world scenarios, we conducted an experiment comparing IID and non-IID datasets. In the IID experiment, we shuffled the dataset and divided it into K equal partitions, with each partition assigned to a client. In the non-IID experiment, we clustered the shuffled dataset using the KMeans [16] algorithm with K clusters. Then we distributed the data among the K clients based on their KMeans cluster assignment. In the non-IID scenario, some clients may have fewer dataset points available than the others, resulting in them potentially stopping their contribution to the learning process earlier.

Federated Learning SVDD has three hyperparameters: the fraction of clients randomly extracted for training at each round C, the number of rounds of training L, and the batch size of the client B. Since each client data point is used only once during training, the number of rounds, the batch size, and the times the client is selected for training must be chosen carefully taking into account the dataset size. If we set those parameters too high, we will soon run out of points during training, preventing the exhausted clients from contributing to the training.

To understand the hyperparameters' impact on performances, we conducted a grid search over the following ranges: number of rounds [1, 3, 5], client fraction [.1, .5, 1], and batch size [1, 5, 10].

	AUC	Average Precision
oc-SVM	0.9721 ± 0.0102	-
OneClassSVM	0.9901 ± 0.0088	0.9886 ± 0.0109
SVDD	0.9975 ± 0.0074	0.9978 ± 0.0072
FL-SVDD IID	-	0.9905 ± 0.0080
FL-SVDD Non-IID	-	0.9662 ± 0.0316

Table 1: The table shows the results of the experiments using various outlier detection algorithms. The oc-SVM results are the ones reported in [2], with AP not available. OneClassSVM and SVDD serve as baselines for comparison, with both AUC and AP metrics reported. For Federated Learning SVDD, experiments are conducted under IID and non-IID data settings, with only AP metric reported as ROC-AUC is not applicable with one class in training.

5 Results

The objective of this study is to extend support vector machine (SVM) techniques for outlier detection to Federated Settings without compromising performance.

Previous research on unsupervised outlier detection techniques [2], presents oc-SVM as one of the suitable SVM techniques for the task. The results achieved in the survey are reported in Table 1's first row. This study attempts to replicate the survey results using OneClassSVM, an implementation of the oc-SVM algorithm provided by [15]. The obtained results are quite similar, although a difference in the selection of the γ parameter may account for the difference. In the survey, the authors used "automatic gamma tuning" to determine its value, while this study arbitrarily sampled its value from the $[0, 3]$ interval.

As discussed in section 4.2, replicating the survey results enables the introduction of a new metric for all experiments: Average Precision.

To extend SVMs to perform outlier detection in a Federated Learning (FL) setting, this study relied on the SVDD algorithm presented in [4]. Although SVDD and OneClassSVM by [3] have different starting points, their algorithms are very similar. The results presented in Table 1 for the SVDD implementation confirm the algorithms' similarity and demonstrate that the results of this study are comparable to those in the survey paper.

FLSVDD-IID replicated the SVDD experiment using the FL primitives described in section 3. The results presented in Table 1 are almost identical, indicating that SVMs can be extended to work in an FL setting without loss of performance, allowing an SVM to be trained without centralizing data or computation.

The previous results were achieved in a controlled setting where data was IID, but in real-life scenarios, data is non-IID, and each client has unique characteristics reflected in its data. The FLSVDD-nonIID experiments stressed the algorithm with a real-life scenario, still achieving excellent performance. The results in Table 1 show that the performance loss is significant, but this is expected given the worsening of the setting.

Although hyperparameter selection did not impact the previous experiments significantly, it plays a critical role in the final experiment, as seen in Figure 2. The amount of data points seen during training depends on the choice of hyperparameters. Table 5 shows that selecting low values for client fraction C or max training rounds L has the worst impact on algorithm performance.

The presented results are encouraging, and their consistency with increasingly challenging conditions is promising. However, it's worth noting that the dataset properties may favor this algorithm since the chosen dataset consists mostly of global outliers, as explained in section 4.1. Future research can explore the performance of different datasets with novel properties.

6 Conclusion & Future Work

In conclusion, this study successfully demonstrates the extension of Support Vector Domain Description in a Federated Learning setting without loss of performance. As shown in section 5, when dealing with IID data, similar performances to the standard SVDD are achieved. Although in the non-IID scenario, a slight decrease in performance is observed as expected.

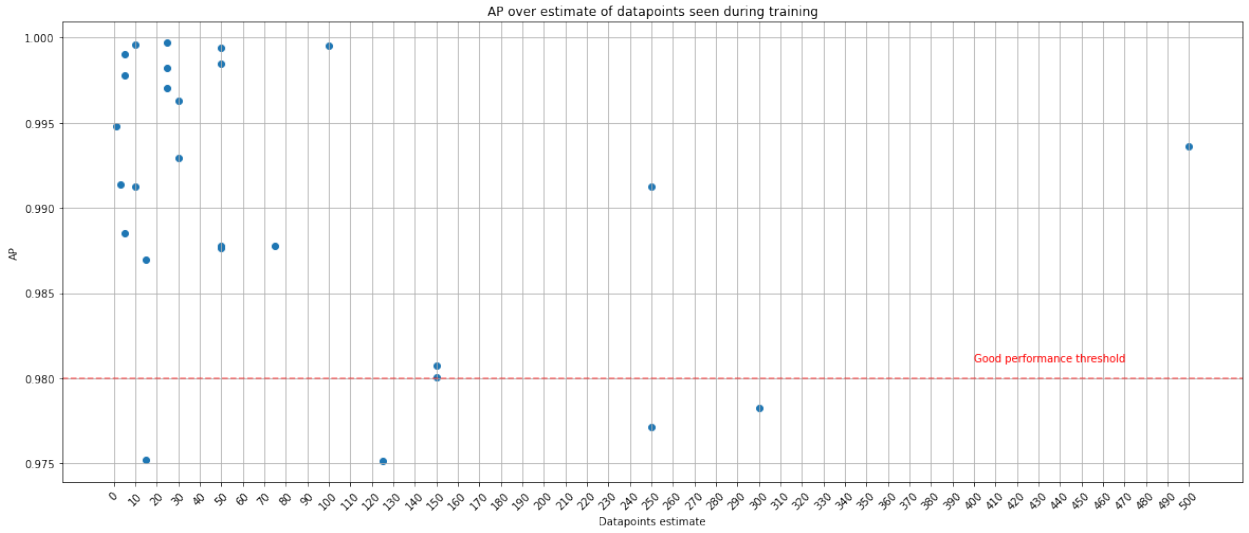

Figure 1: The figure illustrates the Federated Learning SVDD performance in terms of Average Precision (AP) over the estimated number of datapoints seen during training under IID data setting. The estimated number of datapoints seen is calculated as the product of maximum rounds, batch size, client count, and client fraction. A good performance threshold is set arbitrarily. There is no discernible correlation between the number of datapoints seen and AP.

		AVG AP	**STD AP**
C	.1	0.9472	0.0327
	.5	0.9717	0.0332
	1	0.9796	0.0235
L	1	0.9373	0.0341
	3	0.9798	0.0167
	5	0.9813	0.0227
B	1	0.9617	0.0406
	5	0.9638	0.0319
	10	0.9729	0.024

Table 2: The table displays the results of the hyperparameter search for FLSVDD-nonIID. The hyperparameters C, L, and B denote the fraction of clients randomly selected in each round, the maximum number of rounds, and the batch size of each client update, respectively. For each row in the table, one of the hyperparameters is fixed while the others are varied through a grid search, and the results are presented in terms of the mean and standard deviation. The table highlights that extremely low values for L and C lead to worse performance.

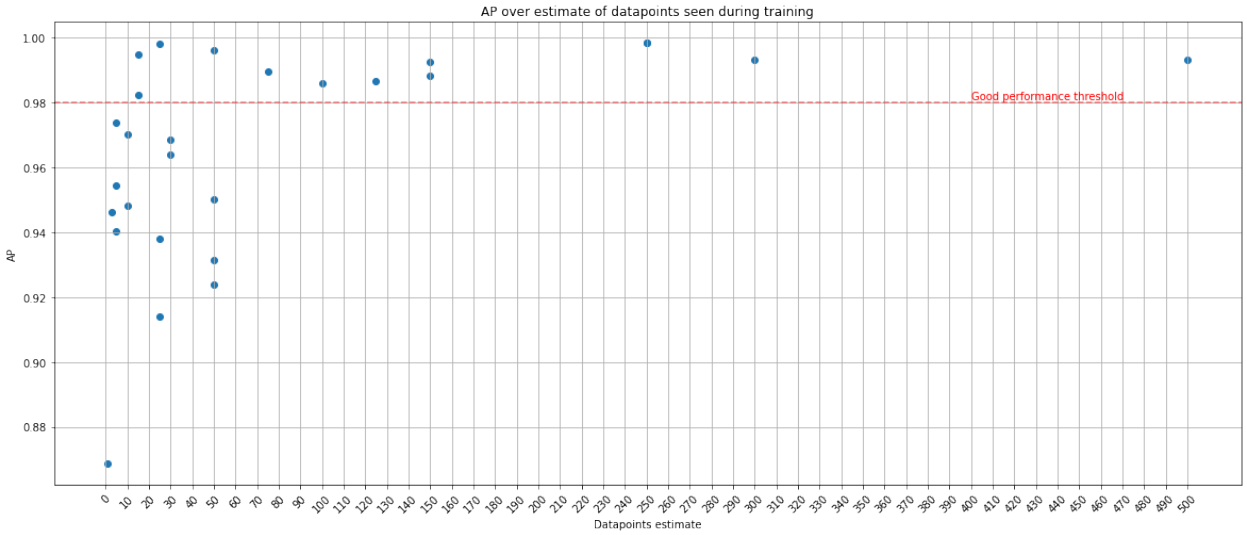

Figure 2: The figure displays the Average Precision of FLSVDD-non-IID as a function of an estimate of the number of data points seen in training. The estimate is computed as the product of max rounds, batch size, client count, and client fraction. A threshold for good performance is set arbitrarily. The plot shows that in a non-IID setting, lower numbers of data points seen during training result in worse performance, which is expected since drawing from a non-IID distribution makes it more challenging to find suitable points for generalization.

The presented results make this research path worthwhile being pursued. However, one of the main challenges to tackle in future works is the poor privacy of the proposed primitives. Sending a set of support vector candidates from each client update compromises the privacy of the client's data, even though the amount of client data sent to the server is reduced. Adding differential privacy techniques could be a potential solution to this issue.

Another important future work initiative could be testing the primitives against more datasets from [2]. The dataset used in this study mostly contained global outliers, making our results less generalizable. Thus, testing the primitives on different types of datasets would improve the validity of the findings.

Improving the hyperparameter initialization could be another area of future work. For example, parameter q could be initialized using the maximum distance between two data points, as shown in [6]. In general, involving clients in the initialization process by allowing them to propose initial hyperparameter values that are then merged server-side could be beneficial.

In this study, we showed data in batches at each client update, and they were never repeated during training. Investigating the impact of showing the data multiple times would be valuable for future studies.

Disclaimer

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university, and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

References

[1] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. Outlier detection: Methods, models, and classification. *ACM Comput. Surv.*, 53(3), jun 2020.

[2] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS ONE*, 11(4):1–31, 04 2016.

[3] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[4] David M.J Tax and Robert P.W Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11):1191–1199, 1999.

[5] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the Royal Society of London. Series A, Containing papers of a mathematical or physical character*, 209:415–446, 1909.

[6] Asa Ben-Hur, David Horn, Hava T Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001.

[7] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. 2016.

[8] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, may 2000.

[9] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.

[10] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. 09 2012.

[11] Raed Abdel Sater and A. Ben Hamza. A federated learning approach to anomaly detection in smart buildings, 2021.

[12] Mirko Nardi, Lorenzo Valerio, and Andrea Passarella. Anomaly detection through unsupervised federated learning, 2022.

[13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[14] Wanhua Su, Yan Yuan, and Mu Zhu. A relationship between the average precision and the area under the roc curve. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, page 349–352, New York, NY, USA, 2015. Association for Computing Machinery.

[15] 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.

[16] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.