Noname manuscript No.
(will be inserted by the editor)

Fair Detection of Poisoning Attacks in Federated Learning on Non-i.i.d. data

Ashneet Khandpur Singh, Alberto Blanco-Justicia, Josep Domingo-Ferrer

Received: date / Accepted: date

Abstract Reconciling machine learning with individual privacy is one of the main motivations behind federated learning (FL), a decentralized machine learning technique that aggregates partial models trained by clients on their own private data to obtain a global deep learning model. Even if FL provides stronger privacy guarantees to the participating clients than centralized learning collecting the clients' data in a central server, FL is vulnerable to some attacks whereby malicious clients submit bad updates in order to prevent the model from converging or, more subtly, to introduce artificial bias in the classification (poisoning). Poisoning detection techniques compute statistics on the updates to identify malicious clients. A downside of anti-poisoning techniques is that they might lead to discriminate minority groups whose data are significantly and legitimately different from those of the majority of clients. This would not only be unfair, but would yield poorer models that would fail to capture the knowledge in the training data, especially when data are not independent and identically distributed (non-i.i.d.). In this work, we strive to strike a balance between fighting poisoning and accommodating diversity to help learning fairer and less discriminatory federated learning models. In this way, we forestall the exclusion of diverse clients while still ensuring detection of poisoning attacks. Empirical work on three data sets shows that employing our approach to tell legitimate from malicious updates produces models that are more accurate than those obtained with state-of-the-art poisoning detection techniques. Additionally, we explore the impact of our proposal on the performance of models on non-i.i.d local training data.

Keywords Federated learning · Security · Privacy · Fairness · Minorities.

1 Introduction

In this digital age, data are key assets. Sources of data often include edge devices, such as smartphones, IoT sensors attached to industrial equipment, or activities conducted at organizations or other entities, such as hospitals. However, collecting, sharing, or releasing these data can lead to many privacy concerns. As companies and institutions collect growing amounts of data on their clients, they need to ensure that the privacy of clients is not violated and that data protection regulations are enforced. The data collected from everyday objects like cell phones, smartwatches, or fitness trackers almost invariably end up in centralized servers where they are aggregated, packaged and then, more often than not, shared with or sold to third parties. This may create privacy issues, since these data sets can include a person's confidential data, such as her browsing history, sexuality, political affiliation, and even medical conditions. These issues have led to the enactment of strict data protection laws, such as the European Union's General Data Protection Regulation (GDPR), which is binding for any organization operating in the EU.

Universitat Rovira i Virgili,
Department of Computer Engineering and Mathematics,
CYBERCAT-Center for Cybersecurity Research of Catalonia,
UNESCO Chair in Data Privacy,
Av. Països Catalans 26, 43007 Tarragona, Catalonia E-mail: {ashneet.singh, alberto.blanco, josep.domingo}@urv.cat

Privacy concerns have become more prominent during the Covid-19 pandemic because, on the one hand, life has become more digital than before and, on the other hand, data collection aimed at controlling the spread of the virus might be perceived as a double-edged sword. While contact and mobility tracing are powerful instruments to preserve public health, their potential for misuse is high. More generally, the privacy expectations of individuals are confronted with the data-hungry artificial intelligence (AI) methods increasingly adopted by organizations. Specifically, for deep learning to be effective vast amounts of data are required to train the models. Service providers collect data at massive scales for such training purposes. Traditionally, these large amounts of data have been stored in centralized databases and processed in central servers owned or hired by the service providers. Such central facilities need tight protection to prevent data leaks. Even if no leaks arise, central data collection and processing generate an asymmetry between the service provider and the customer, because the former accumulates a wealth of personal data on the latter.

Federated learning (FL) (Konečnỳ et al., 2016; McMahan et al., 2017) attempts to solve these problems. FL is a machine learning technique that operates in a decentralized manner and allows learning models with the help of a set of clients, each of whom privately owns a local data set. In FL clients receive an initial global model from a service provider, often called model manager. Then each client updates the received model based on her private local data, and then uploads the model update to the model manager. The model manager aggregates the client updates to produce a new version of the global model. In this way, the global model can be iteratively improved and shared without the model manager ever accessing the private data of clients. The process iterates until the model converges.

The most usual situation in FL is that there is a crisp divide between the model manager, who orchestrates the different steps of the process, and the clients, who update the global model based on their private data. Yet, it is also conceivable to use federated learning in a peer-to-peer scenario, where each peer may be both a model manager (of her own model) and a client (who updates the models of other peers). In any case, clients transmit to the model manager the bare minimum data to improve the model. This is inherently more privacy-preserving than centralized approaches in which client data are collected by a central server to build a machine learning model. Another advantage of FL is that the learning effort is distributed among the clients, instead of being centralized in a single entity.

For all its many advantages, FL is not free of issues. In particular, it is vulnerable to security attacks whereby malicious clients sabotage the learning process by sending bad model updates. These attacks may seek to prevent convergence to a model (Byzantine attacks) or to cause convergence to a flawed model whose output is determined by the attacker at least for designated inputs (poisoning attacks). Poisoning attacks are described in Bhagoji et al. (2019), along with several solutions that thwart them. A well-known poisoning attack is label flipping, where the attacker is assumed to be able to flip the labels of a fraction of training points. In this work, we restrict to the detection of label flipping attacks and leave for future work the prevention of other types of poisoning attacks (see Section 5 in Kairouz et al. (2019) for an overview of different poisoning and backdoor attacks).

Techniques to prevent security attacks (discussed in Section 3) compute statistics on the client updates to detect outlying values. Since abnormal and malicious behaviors are usually associated with outlying updates, these are filtered out as an attack prevention strategy when updating the global model. Even though this type of approaches are effective to prevent attacks, systematically rejecting outlying updates might also lead to unfair global models (Narayanan, 2018) if the outlying data correspond to a legitimately different minority. Apart from facing the unfairness issue, attack prevention countermeasures for FL often struggle to correctly treat non-i.i.d. (non-independently and identically distributed) data. Most research proposals assume that the clients' private data are i.i.d.

In this work, we explore the tensions between data privacy, partially achieved by the use of federated learning, model robustness against label flipping attacks, and fairness in classification tasks. As outlined above, federated learning is vulnerable to poisoning attacks, and in particular to label flipping attacks. Mechanisms to protect against these attacks are based on filtering outlying model updates. However, it is not known *ex ante* whether these outliers come from attackers or from benign clients whose data are genuinely different from those of the majority of clients, either because the data are non-i.i.d. or because those benign clients belong to a minority group. Thus,

attack protection mechanisms in the literature provide model robustness at the cost of classification fairness. We propose mechanisms to better model the updates provided by the clients, by finding similarities among outliers that can indicate the existence of minority groups and by only discarding those updates which are completely isolated. Other tensions among desirable properties of ML models are explored in Wang et al. (2021).

1.1 Contribution and plan of this paper

No honest client ought to be discriminated in FL due to the genuine attribute values of the persons/records represented by the client when interacting with other clients or the model manager. In other words, all honest clients should be able to contribute to the training process, because this is the best way to obtain not only fair but also good-quality decision models. Note that ignoring minority groups in the training process decreases the quality of the learned global model.

However, as introduced above, being inclusive with respect to minorities often clashes with the ability to detect attacks against FL models. A common detection approach is for the model manager to compute the Euclidean distance between each of the client-provided model updates and the average of such updates, and then discard as potentially malicious any update too far from the average, according to some threshold or rule. In the presence of non-i.i.d. data, or when some of the clients represent individuals from minority groups, this approach might lead to treating genuinely different individuals as potential attackers. This would not only be unfair to minorities, but would result in a biased model.

Our aim is to strike a balance between anti-poisoning and diversity accommodation. By including diverse clients, we aim at making it possible to learn less discriminatory machine learning models. In Khandpur Singh et al. (2020), a preliminary conference version of this work, we presented two approaches to properly distinguish members of minority groups from potential model poisoners when carrying out robust aggregation of updates. In addition, this article presents a third approach and also studies the differences between the cases of i.i.d. and non-i.i.d. data. Thus, the contributions in this paper are:

- A first method to distinguish minority members from attackers based on microaggregation (Domingo-Ferrer and Mateo-Sanz, 2002). Clients who identify themselves as belonging to a minority group announce some relevant attributes to their peers, such as their *gender*, their *sexual orientation*, or their *ethnicity*. From these attributes, the peers carry out a clustering process via collaborative microaggregation. In this way, the majority group and the minority groups are clustered separately. After that, an FL model is trained for each cluster. Since peers have been already clustered according to some of their attributes, outliers within clusters are likely to be attackers because their updates are unusual even for a minority group. Finally, a weighted aggregation of the different cluster-level models is computed, where the weights are proportional to the sizes of the clusters.
- A second method where we use Gaussian mixture models to characterize the distribution of the client-provided updates and classify outliers in a more sophisticated way than just relying on the distance to an average client update. In the presence of minority groups that differ from the majority group in some attributes, but that are homogeneous within themselves, we expect this approach to label honest individuals from minority groups as non-malicious.
- A third method predicated on density-based clustering. Specifically, we use the DBSCAN algorithm to identify clusters of any shape among the client updates. In the FL setting the assumption is that the objectives for all clients approximate the global objectives. However, this is not the case with non-i.i.d. data. DBSCAN can help correctly characterize the distribution of updates from clients with non-i.i.d. data.

The rest of the paper is organized as follows. Section 2 gathers background concepts. Section 3 discusses related work on attack mitigation techniques in FL. Section 4 presents our three methods for fair detection of attacks based on microaggregation, Gaussian mixture models and DBSCAN, respectively. Section 5 reports empirical results that illustrate the effectiveness of our approaches for both i.i.d. and non-i.i.d. private local data. Finally, conclusions and future research lines are discussed in Section 6.

2 Background

In this section, we first present the general form of federated learning. Then, we introduce the notions of fairness used in this article and how non-i.i.d-ness works in FL.

2.1 Federated learning

In an FL scenario, a model manager initializes a learning model, such as a neural network, with weights θ^0, loss function L and learning rate ρ. Other hyper-parameters may apply, such as dropout rate, decay or momentum, but we restrict to a general model using stochastic gradient descent (SGD). The model manager may or may not pre-train the model with available public or private data already in her possession. Each of the m clients, whose devices are called client or edge devices, has access to a data set $D_u = \{x_i^u, y_i^u\}_{i=1}^{n_u}$ of size n_u. The total size of the available data is $n = \sum_{u=1}^{m} n_u$. At epoch t —where epoch means learning iteration—, the model manager sends the current global model θ^{t-1} to all clients; these use their devices to train local models from the global model using their respective private data sets D_u; then, clients send their respective updates δ_u^t to the model manager, who updates the global model θ^{t-1} into θ^t by averaging the updates, possibly subject to a parameter η which regulates the model substitution rate. Additionally, a vector $\vec{\alpha}$ can be used to adjust the weight of each client's contribution in the federated aggregate. The intuition of FL is depicted in Figure 1.

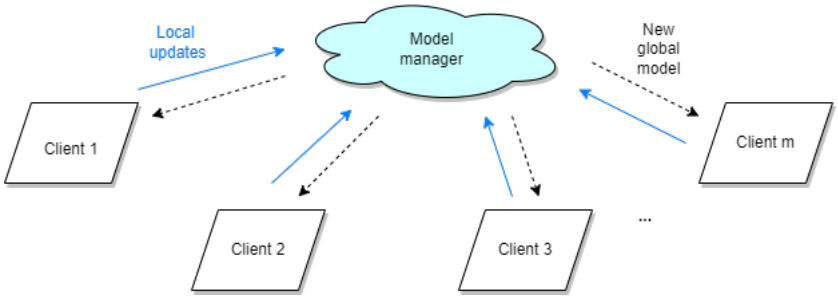


Fig. 1: Overview of federated learning

A possible choice is for all components of $\vec{\alpha}$ to be $1/m$, in which case the clients have the same influence. If the client data sets are of very different sizes, an alternative choice giving weight $\alpha_u = n_u/n$ to the u-th client might make sense. Also, in case a client is found malicious, her α_u value can be set to 0 to exclude her contributions from the aggregate. This approach to aggregating updates is the most usual one and is known as federaged averaging (FedAvg, (McMahan et al., 2017)). See its pseudocode in Protocol 1.

Protocol 1: FedAvg

Initialize model parameters θ^0 and distribute them among clients;
while *termination condition not met* **do**
  foreach *client* $u \in S$ *;* `// S: set of m clients`
  do
    $\delta_u^t \leftarrow \frac{\rho}{n_u} \sum_i \nabla L(\{x_i^u, y_i^u\}, \theta^{t-1})$;
    Send δ_u^t to the model manager;
  end
  $\vec{\alpha} \leftarrow AttackerDetection(\theta^{t-1}, \{\delta_u^t\})$;
  $\theta^t \leftarrow \theta^{t-1} + \eta \sum_u \alpha_u \delta_u^t$;
  Distribute θ^t among clients;
end

2.2 Notions of fairness

To ensure high-quality learning, the FL model manager should refrain from making decisions that unfairly (dis)favor any particular group of clients. On the one hand, unfair treatment can discourage clients from joining the FL training. On the other hand, blindly treating all clients equally without regard to their potentially diverse contributions can yield FL models that do not generalize well. Hence, ensuring fairness in FL is essential, as it is the key to sustainable healthy collaboration in such an ecosystem. In this work, we use the following notions of fairness from Verma and Rubin (2018). In the definitions below, $A \in \{0, 1\}$ is the protected attribute (that distinguishes the minority/protected group from the non-minority/unprotected group), $Y \in \{0, 1\}$ is the target decision variable, and $\hat{Y} \in \{0, 1\}$ is a binary predictor.

Definition 1 (**False positive error rate balance** (also called **predictive equality** (PE))). A prediction algorithm satisfies this definition if the subjects in the protected and unprotected groups have equal FPR (false positive rate). That is, the probability of a subject in the minority group to have a wrongly predicted positive outcome is the same as for a subject in the majority group:

$$P(\hat{Y} = 1 \mid Y = 0, A = 0) = P(\hat{Y} = 1 \mid Y = 0, A = 1).$$

Definition 2 (**False negative error rate balance** (also called **equal opportunity**(EO))). A prediction algorithm satisfies this definition if the subjects in the protected and unprotected groups have equal FNR (false negative rate). That is, the probability of a subject in the majority group to have a wrongly predicted negative value is the same as for a subject in the minority group:

$$P(\hat{Y} = 0 \mid Y = 1, A = 0) = P(\hat{Y} = 0 \mid Y = 1, A = 1).$$

In our context, a positive prediction for a client means that her update is accepted by the manager, whereas a negative prediction means that it is discarded (as being potentially malicious).

2.3 Non-i.i.d. data in FL

The fact that training data are often non-i.i.d. among clients is a challenge faced by FL that also has fairness ramifications. In this setting, the distribution of the local data at each client is not representative of the distribution of the global data (those that would be obtained if all the clients' local data were pooled). Non-i.i.d. data make it difficult for FL to learn models that are as good as those obtained with centralized learning.

Non-i.i.d.-ness can be measured by the differences between the gradients obtained by the clients on their respective local data and the gradients of the global model. For a non-negative real value δ, Zhang et al. (2020) characterize δ-non-i.i.d. data in FL with the following condition

$$||\nabla f_u(\theta) - \nabla f(\theta)|| \leq \delta, \forall u, \tag{1}$$

where θ are the model parameters, $\nabla f_u(\theta)$ are the gradients obtained by client u after a local training phase, and $\nabla f(\theta)$ are the global model gradients. Expression (1) limits to δ the difference in the distributions of the gradients of individual users and those of the global model.

3 Related work

Several solutions have been proposed in the literature to detect attacks or abnormal behaviors in machine learning (Gander et al., 2012; George and Vidyapeetham, 2012). In the specific context of FL, where the model manager has access to the individual updates from the clients, the following classes of attack detection methods have been proposed:

– *Detection of malicious clients via model metrics.* The model manager can reconstruct the individual updated models for every client u and compare the model performance metrics, such as accuracy or loss, against a validation data set with respect to the model obtained by

aggregating all updates except that of client u. The model manager can mark as anomalous and possibly discard any client updates that degrade the model performance according to some rule or threshold. Note that the model manager needs a suitable validation data set, which may not always be available in the FL scenario. Moreover, re-evaluating the model accuracy after each update is extremely costly, and introduces an unacceptable overhead in the FL process.

- *Detection of malicious clients via update statistics.* A very common and natural approach for the model manager is to observe the statistics of the magnitudes of the updates (Yin et al., 2018). The model manager can compute how much do the distributions of distances in successive iterations change, for example using the Kullback-Leibler divergence metric. In a scenario with colluding malicious clients, these might have enough influence on the computed centroid to render the previous countermeasures ineffective. To gain additional protection, the model manager can compute the centroid as a median rather than as an average. The median is more robust in front of outlying updates submitted by malicious clients. More costly alternatives are presented in Li et al. (2019a), where anomalous clients are detected by generating low-dimensional surrogates of model weight vectors, and in Li et al. (2020), in which a spectral anomaly detection is performed by the model manager.
  A decentralized approach based on update statistics is presented in Domingo-Ferrer et al. (2020). A client's model update is considered legitimate if its distance to the centroid of all client updates is roughly between the first and the third quartiles of the set of distances between all client updates and the centroid.
- *Krum aggregation.* The authors of Blanchard et al. (2017) propose an aggregation function that is resilient against f malicious clients. This function is called Krum. The authors show that averaging does not stand Byzantine attacks, while Krum does. An important advantage of Krum is its (local) time complexity $\mathcal{O}(m^2 \cdot d)$, which is linear in the dimension of the updates. The authors also evaluate a variant of Krum, Multi-Krum, which interpolates between Krum and averaging.
- *Coordinate-wise median.* In Yin et al. (2018), a median-based distributed algorithm is proposed that selects the coordinate-wise median instead of the coordinate-wise average. Since the median is a more robust statistic than the mean (i.e. it is less influenced by outliers), the obtained global model is less influenced by potential malicious peers.
- *Coordinate-wise trimmed mean.* Also in Yin et al. (2018), a second distributed algorithm is proposed, called coordinate-wise trimmed mean, that can achieve order-optimal error rate under weaker assumptions than the coordinate-wise median algorithm.

In the approaches above, updates that are statistical outliers departing from a global aggregate model are considered malicious. However, it may also be the case that honest clients have genuinely outlying local data and therefore generate genuinely outlying updates. This may be a consequence of the clients belonging to a minority group.

There is a growing interest in the development of fair models for machine learning. In federated settings, Li et al. (2021) propose DITTO, a multi-task learning framework, to address the competing constraints of accuracy, fairness and robustness in FL. The authors of this work define fairness as each client achieving equal test performance on the federated model. In Lyu et al. (2020), a collaborative fair federated learning framework (CFFL) is proposed. In this work, fairness is achieved by adjusting the performance of the models allocated to each participant based on their contributions. Also, Du et al. (2021) aim at achieving group fairness in FL. The authors mimic the centralized fair learning setting by very frequently exchanging information for each local update, rather than for each round of local training.

Several works have dealt with non-i.i.d. data in a federated learning setting. As prior studies show, decentralized learning algorithms lose significant model accuracy in the non-i.i.d. setting. In Zhao et al. (2018), the authors propose a strategy to improve training on non-i.i.d. data by creating a small subset of data which are globally shared among all the edge devices. However, this relies on a substantial amount of public data being available for a given task. In Jeong et al. (2018), the authors propose federated augmentation (FAug), where clients collectively train a generative model, and thereby augment their local data towards yielding an i.i.d. data set. The authors of Li et al. (2019b) analyze the convergence of federated averaging on non-i.i.d. data and establish a convergence rate of $\mathcal{O}(\frac{1}{T})$ for strongly convex and smooth problems, where T is the number of rounds of local SGD updates. The commonly used FedAvg (McMahan et al., 2017) makes no

special adjustments when encountering non-i.i.d data and therefore suffers from a deterioration in the accuracy of FL (Hsieh et al., 2020). This performance degradation can chiefly be attributed to weight divergence of the local models resulting from non-i.i.d data.

A systematic study on local model poisoning attacks to FL is offered in Fang et al. (2020), including the attacks mentioned above. The authors simulate FL with different non-i.i.d. training data distributions. They generalize two defenses against data poisoning attacks, which are effective in some cases but not in others; this highlights the need for new defenses against local model poisoning. For further background on attacks and defenses in FL, see the surveys Kairouz et al. (2019) and Blanco-Justicia et al. (2020). The methods we introduce in the next sections depart from the state of the art in that they aim at properly managing updates originated by clients with local data on minorities.

4 Fair attack detection methods

To evaluate the performance of the trained model, the fairness notions of Section 2.2 can be readily applied to centralized model training. However, with non-i.i.d data in FL, low levels of fairness are likely. To address this problem, one must pay attention to the distribution of outlying updates. If these are concentrated, then this could signal a minority, rather than attackers. Fairness comes from differentiating attackers from minorities, so that the latter can avoid rejection of their updates.

4.1 Fair attack detection based on microaggregation

In this section, we introduce our microaggregation-based approach for fair detection of attacks in federated learning.

Microaggregation is a perturbative method for statistical disclosure control of quantitative microdata. The method was introduced in Domingo-Ferrer and Mateo-Sanz (2002) for numerical data, and Domingo-Ferrer and Torra (2005); Torra (2004) extended it for categorical data. Microaggregation is based on two steps:

1. *Partition.* The records in the original data set are partitioned into a number of clusters, each of them containing at least k records (the minimum cluster size) and no more than $2k-1$ records. To minimize information loss in the following step, records in each cluster should be as close to one another as possible.
2. *Aggregation.* An aggregation operator is used to compute the centroid of the records in each cluster. Then the records in the cluster are replaced by their centroid.

In our approach, we are interested only in the partition step, whereby similar clients will be clustered together based on their demographic attributes. The superiority of microaggregation over standard clustering for our purposes lies in that the former ensures that clusters will have at least size k. In this way, we avoid training models for clusters that are too small. Note that it is impossible to detect any outliers if too small clusters are allowed.

We propose the solution in Protocol 2 based on collaborative microaggregation to distinguish malicious clients from clients with outlying updates computed on genuine minority data, which we will call in what follows *protected clients*.

In Line 1 of Protocol 2, the demographic attributes that characterize a minority client might for example be {Sex=female, Age=young, Ethnicity=black}; we implicitly assume that clients holding local minority data have themselves minority demographic attributes. In the microaggregation called in Line 2, parameter k must be taken large enough so that outliers can be distinguished within a group of k, and a collusion of k clients or of a significant fraction of k clients is unlikely. In Line 7 assigning a nonzero weight α_u to P_u's update means accepting the update as legitimate (because it is similar to most updates in $\mathcal{P}_u$'s cluster C). In contrast, in Line 9 assigning $\alpha_u = 0$ means discarding the update (because it is too outlying even for the minority group represented by C).

Note that microaggregation attempts to create clusters such that the published attributes of protected clients in each cluster are maximally similar. Therefore, if clients within a cluster are similar, it is natural to expect that the updates they send are also similar. As a consequence,

Protocol 2: AttackerDetectionMicro

Each protected client publishes, together with her pseudonym, demographic attribute values that characterize her as a minority group client;
The model manager and protected clients engage in decentralized microaggregation as per Protocol 3 to microaggregate protected clients according to their published attributes;
for *each cluster C of the microaggregation* **do**
  Compute the centroid c_C of the updates sent by clients in C;
  Let λ_C be the set of clients $\mathcal{P}_u \in C$ such that the distance $dist_u$ from $\mathcal{P}_u$'s update δ_u^t to the centroid of the updates is not an outlier, more precisely

$$\lambda_C = \{\mathcal{P}_u \in C | Q1 - \tau \times IQR \leq dist_u \leq Q3 + \tau \times IQR\}, \quad (2)$$

  where $Q1$ and $Q3$ are, respectively, the first and the third quartile of the set of distances, $IQR = Q3 - Q1$ is the interquartile range and τ is a tolerance parameter;
  if *a client $\mathcal{P}_u \in \lambda_C$* **then**
    $\alpha_u = 1/m$ (or n_u/n);
  else
    $\alpha_u = 0$;
  end
end

Protocol 3: Decentralized microaggregation

Input: Clients $\mathcal{P}_1, \ldots, \mathcal{P}_m$; microaggregation parameter k;
Output: A partition of clients $\{C_i\}_{i=1}^p$;
Let qi_u be demographic attribute values of $\mathcal{P}_u$, for $u = 1, \ldots, m$;
Each $\mathcal{P}_u$ publishes qi_u;
The model manager uses a microaggregation algorithm based on the quasi-identifiers $qi_1, \ldots, qi_c$ to obtain clusters C_1, C_2, ..., C_p, such that $k \leq |C_j| \leq 2k - 1$;
The model manager publishes C_1, C_2, ..., C_p;
Each $\mathcal{P}_u$ can compute the above clusters, verify they are correct, and check that the cluster $C_{\mathcal{P}_u}$ where qi_u belongs contains k or more quasi-identifiers;

if an update differs very much from the others, it is not unreasonable to treat the client having contributed it as malicious.

To create homogeneous clusters in an efficient way, in Protocol 3 we use the maximum distance to average vector (MDAV) algorithm, detailed in Algorithm 1, which is the most widely used microaggregation algorithm (Domingo-Ferrer and Torra, 2005).

Algorithm 1: MDAV microaggregation algorithm

Input: Data set R, microaggregation parameter k;
Output: A partition of R, with sets of minimum size k;
while $|R| \geq 3k$ **do**
  Compute the average record $\bar{x}$ of all records in R;
  Consider the record x_r most distant from $\bar{x}$;
  Find the record x_s most distant from x_r considered in the previous step;
  Form two clusters around x_r and x_s, respectively, one containing x_r and the $k-1$ records closest to x_r and the other cluster containing x_s and the $k-1$ records closest to x_s;
  Take as a new data set R the previous data set R minus the clusters formed around x_r and x_s in the last instance of Step 5;
end
if *there are between $3k-1$ and $2k$ records in R* **then**
  Compute the average record $\bar{x}$ of the remaining records in R;
  Find the record x_r most distant from $\bar{x}$;
  Form a cluster containing x_r and the $k-1$ records closest to x_r;
  Form another cluster containing the rest of records;
else
  Form a new cluster with the remaining records;
end

MDAV is a heuristic algorithm that clusters records in a data set so that each cluster is guaranteed to contain at least k records. At each iteration, two records are selected: the record x_r farthest

from the average record $\bar{x}$ of the data set and the record x_s farthest from x_r. Then, a cluster is formed with x_r and its closest $k-1$ records, and another cluster with x_s and its closest $k-1$ records. The records in both clusters are removed from the data set in the next iteration.

4.2 **Fair attack detection based on Gaussian mixtures and expectation-maximization**

In this section, we propose a second approach to tackle the problem of fair attack detection in federated learning. It is based on Gaussian mixture models (GMM) and the expectation-maximization (EM) algorithm.

Gaussian mixture models are probabilistically weighted combinations of Gaussian components, each with its own mean and covariance. Mixture models, in general, are better suited than single distributions at modeling populations where differences between sub-populations exist. We leverage this property of Gaussian mixture models to capture the differences among different sub-populations (*e.g.*, minorities) while still being able to determine whether some data points are too far from the distribution modeling the population.

The expectation-maximization algorithm is an iterative method to find the maximum-likelihood estimates for GMM parameters in the presence of latent (hidden) variables. The expectation-maximization algorithm takes the number of Gaussians to model the data, K [1], and iteratively finds for each Gaussian $k \in \{1, \ldots, K\}$ its weight π_k, its mean μ, and its covariance matrix Σ_k. Given these parameters, we are able to compute how likely each point is to belong to the mixture of Gaussians.

Algorithm 2 shows how we use the expectation-maximization algorithm to detect potential malicious updates in federated learning aggregation. This algorithm is used at each global learning step, that is, at the time of aggregating local updates. Once the model manager receives all updates from clients, it fits a GMM to the received updates using the expectation-maximization algorithm. Then, each individual update is evaluated according to the log-likelihood that it follows the derived distribution. Those updates with a log-likelihood below a parameter τ (*i.e.*, those updates that are significantly different from the rest) are flagged as potentially malicious and discarded.

Algorithm 2: AttackerDetectionEM

Input: Client updates $\{\delta_u^t\}_{u=1}^m$
$\pi, \mu, \Sigma \leftarrow EM(\{\delta_u^t\}_{u=1}^m)$;
for u *in* S **do**
  $\ell \leftarrow \log(\sum_k \mathcal{N}(\delta_u^t; \mu_k, \Sigma_k) \cdot \pi_k)$;
  if $\ell < \tau$ **then**
    $\alpha_u = 0$;
  else
    $\alpha_u = 1/m$ (or n_u/n);
  end
end

4.3 **Fair attack detection based on DBSCAN**

In this section we propose a third approach to tackle the problem of fair attack detection in federated learning. It is based on a commonly used data clustering algorithm, *i.e.* density-based spatial clustering of applications with noise (DBSCAN). In Ester et al. (1996), a new notion called density-based clustering was introduced, whereby clusters of any shape can be identified in data sets containing noise and outliers. The goal of DBSCAN is to identify dense regions, which can be measured by the number of objects close to a given point.

DBSCAN requires two parameters:

- Epsilon (*Eps*): maximum radius of the neighborhood around a point.

[1] Choosing K is not trivial since it is not known how many clusters there are. One can vary K until obtaining useful results.

– Minimum points ($MinPts$): minimum number of points in the Eps-neighborhood of a point. This Eps-neighborhood of a point p can be defined as $N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\}$, where D is the total set of points.

Any point p in the data set with a neighbor count at least $MinPts$ is marked as a *core point*. A point p is a *border point* if the number of its neighbors is less than $MinPts$, but it belongs to the Eps-neighborhood of some core point. If a point is neither a core point or a border point, then it is called a *noise point* or an outlier.

Those points that do not belong to any cluster are treated as outliers or noise. One limitation of DBSCAN is that it is sensitive to the choice of parameters, especially if clusters have different densities. If Eps is too small, a cluster whose point-to-point distances are greater than Eps will be taken as noise. In contrast, if Eps is too large, clusters whose inter-cluster distance is less than Eps may be merged together.

In Algorithm 3 we show how DBSCAN can be used for attacker detection in federated learning. The model manager fits the model to the updates and goes through each individual update to check if it is a noise point. If the latter happens, then the model manager flags the update as malicious.

Algorithm 3: AttackerDetectionDBSCAN

Input: Set of clients S, client updates $\{\delta_u^t\}_{u=1}^m, Eps, MinPts$

$Core, Border, Noise \leftarrow DBSCAN(\{\delta_u^t\}_{u=1}^m, Eps, MinPts)$;
for u *in* S **do**
  if $\delta_u^t \in Noise$ **then**
    $\alpha_u = 0$;
  else
    $\alpha_u = 1/m$ (or n_u/n);
  end
end

5 Experimental results

We conducted experiments to examine the effectiveness of our attack detection mechanisms in FL with minority groups and non-i.i.d. data. To that end, we chose three publicly available data sets, namely (i) the Adult Income data set (Dua and Graff, 2017), (ii) the Athletes data set (Griffin, 2018), and (iii) the Bank Marketing data set (S. Moro and Rita, 2014).

In the next sections we describe these data sets and the preprocessing we conducted on them to emulate both clients with local minority data and clients bases with non-i.i.d. data.

5.1 Data sets, preprocessing, and baseline scenarios

Here, we present a summary table (Table 1) of the data sets we use in our experiments, along with how we compute the initial baseline metrics.

Table 1: Characteristics of data sets

	Adult	***Athletes***	***Bank Marketing***
# Records	45,222	206,165	45,211
Sensitive attribute	Race	Height and Country	Marital status
Balanced data set	No	Yes	No
Attributes	14	17	17
Class label	Income	Height	Term deposit

For the three data sets, we first cleaned missing values and identified (sensitive) attributes that split the population of individuals in each data set into majority and minority groups.

To measure potential biases in the data sets, we trained a federated learning baseline model for each of them and we computed performance metrics. In all three cases, the baseline models were built using Keras and consisted of a multilayer perceptron (MLP) with two hidden layers using the ReLU activation function. Since we were training binary classifiers, the output layer used a sigmoid activation function. We used binary cross-entropy as the loss function and the Adam optimizer with a learning rate $3 \cdot 10^{-4}$.

After this evaluation, we prepared the data sets for further experiments in a federated scenario with fair malicious client detection. We considered three different kinds of clients: majority clients, minority clients, and malicious clients. We followed the approach of McMahan et al. (2017). The step-by-step process was to first sort the data, then divide the data into equally sized shards, and finally assign each of the shards to a different client.

The next subsections provide details on these procedures for each of the data sets.

5.1.1 Adult Income data set

The Adult data set has been typically used to train classifiers which predict whether an individual earns more or less than $\$50,000$ per year, according to a set of demographic attributes, and with two class values: $\leq$ \$50K (class 0) and $>$ \$50K (class 1). The two classes are distributed as follows: 34,014 individuals (75.21%) earn up to $\$50K$ per year and 11,208 individuals (24.78%) earn more. The race attribute has 5 distinct values: 1. White: 38,903 (86.02%); 2. Black: 4,228 (9.34%); 3. Asian-Pac-Islander: 1,303 (2.88%); 4. Amer-Indian-Eskimo: 435 (0.96%); 5. Other: 353 (0.78%).

We observe that White and Asian-Pacific-Islander are more likely to be in the $>$ $\$50K$ class than the rest, with over 25% of the observations of these two in that class. In comparison, only around 12% of black individuals belong to this class. Therefore, we found that this data set had a bias towards White and Asian-Pacific-Islander individuals, so we first wanted to establish whether a machine learning model trained with these data also showed this bias.

To that end, we first set to finding which proportion of black individuals were misclassified as earning less than $\$50K$, in comparison to individuals from other ethnicities. The whole data set is clearly imbalanced towards the low-earning class, and so we expected a certain FNR (taking the class $<$ $\$50K$ as class 0) across all ethnicities. We wanted to know if this FNR was balanced across all ethnicities. First, we recoded the 'Black' individuals in the variable 'race' as 0 and the rest as 1, who were the 'nonblack' individuals. This resulted in 4,228 black individuals and 40,994 nonblack individuals. Then, we trained an MLP as described above in a centralized manner by using a random 75-25% train-test split: the training set consisted of 33,916 samples and the test set had 11,306 samples.

Table 2 shows the centralized baseline scenario results for Adult.

Table 2: Performance measures for the Adult centralized baseline scenario

	Black	***Nonblack***
Accuracy	87.31%	90.18%
FNR	0.1203	0.0881
FPR	0.0797	0.0702
ROC AUC	0.80	0.82

From Table 2, we can confirm that the model is biased to favor the 'nonblack' individuals since their FNR is smaller (8.81%) than for the 'Black' individuals (12.03%), *i.e.*, black individuals are incorrectly assigned to be in the low income category in a bigger proportion than nonblack individuals.

Next, we prepared the Adult data set for a federated learning scenario with malicious clients. Adult was split into 50 client shards, as follows:

- 30 shards contained records that only included nonblack individuals, *i.e.*, $race = 1$. These corresponded to the majority clients. Each shard consisted of 1,500 records, of which 1,400 were reserved for training and 100 for testing.

- 19 shards contained records of black individuals, *i.e.*, $race = 0$ across the two income classes. These shards corresponded to the minority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- The remaining shard contained 55% of rich black individuals and 45% of low-income black individuals whose labels had been flipped to high-income. This shard represented a malicious client, trying to make the model misclassify low-income black individuals as high-income.

Our objective was to detect and remove those updates computed on poisoned records without contributing to the bias against the minority clients. That is, without punishing the genuine high-income black individuals.

5.1.2 Athletes data set

The second data set contains information on all the athletes that have competed in any of the Olympic games from Athens 1896 to Rio 2016. We replaced all missing values in the 'Medal' attribute with 'No Medal', and, after dropping the rest of missing values, we were left with $206,165$ records.

For our study, we used 'Country' and 'Height' as sensitive attributes. We made a new column 'Country_height', which has the values 'SouthAsian', labeled as 0 and 'NSA', labeled as 1. The first class included countries from South Asia and South East Asia present in the data set (Indonesia, Vietnam, Philippines, Malaysia, Sri Lanka, Thailand, Singapore, India, Pakistan, Maldives, Afghanistan, Bangladesh, Bhutan, Nepal, Brunei, Cambodia, Laos, Myanmar, Japan) for which the data shows people are more likely to have a height attribute below the data set median height (175.0 cm). The second value was the non South Asian countries, which included the rest of the countries.

We considered 'tall' (class 1) those athletes with height 175.0 (the data set median) or above ($110,618$ athletes) and 'short' (class 0) those athletes with height below 175.0 ($95,547$). This was the classification task. We were interested in studying the biases in models trained on these data, by focusing on male South Asian athletes as the protected minority. First, we created a new data set with only male athletes, where $7,851$ were from South Asian countries and $131,603$ were from non South Asian countries. Then, we trained an MLP as described above in a centralized manner, in which the training set consisted of $104,590$ samples and the test set of $34,864$ samples. This was the baseline scenario, whose metrics we show in Figure 3.

Table 3: Performance measures for the Athletes centralized baseline scenario

	SouthAsian	***NonSouthAsian***
Accuracy	89.42%	92.81%
FNR	0.0794	0.0701
FPR	0.0566	0.0513
ROC AUC	0.82	0.83

Then, we prepared the data set to be evaluated in a federated learning scenario with fair malicious client detection as described above. In this case, we split the data into 90 shards, again, with three different kinds of clients:

- 60 shards contained records of non South Asian athletes, *i.e.*, 'Country' = 1. These corresponded to the majority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- 29 shards contained records of South Asian athletes, *i.e.* 'Country' = 0. These shards corresponded to the minority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- The last shard contained both tall South Asian athletes (25%) and short South Asian athlethes (75%) whose labels had been flipped to tall. This shard is assigned to the attacker.

Again, our purpose was to detect the attacker and discard its updates without affecting the performance of the model or causing South Asian athletes to be misclassified in a larger proportion than in the baseline scenario.

5.1.3 Bank Marketing data set

The last data set we used is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. Its classification goal is to predict whether the individual will subscribe to a term deposit ('yes' = 0, 'no' = 1). It is an imbalanced data set with the class 1 ('no', that is, not subscribed) being the majority class (88.7%).

In this case, we used the $Marital-status$ attribute as the sensitive attribute. The majority of the individuals are married (60.19%) and these are the most likely to subscribe to a term deposit. The rest are singles (28.29%) or divorced (11.52%). To find out whether this imbalance caused bias in classification, we first labeled 'Divorced' individuals in the variable 'Marital-status' as 0 and the rest as 1, which are the 'non-divorced'. $5,207$ records belonged to divorced individuals and $40,004$ belonged to individuals who are not. Then, we trained an MLP as described above in a centralized manner, with a training data set of size $33,908$ and a testing data set of size $11,303$. We show the metrics for this baseline scenario in Table 4.

Table 4: Performance measures for the Marketing centralized baseline scenario

	Divorced	***NonDivorced***
Accuracy	85.74%	88%
FNR	0.1102	0.1063
FPR	0.0787	0.0709
ROC AUC	0.79	0.80

To prepare the data set for experiments with malicious clients, we split it in 50 shards as follows:

- 30 shards contained records of non-divorced clients, i.e. 'Marital-status' = 1. These corresponded to the majority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- 19 shards contained records of divorced individuals. These shards corresponded to the minority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- 1 shard of 1,500 records included 622 divorced individuals who had not subscribed to a term deposit. The label for 90% of those 622 records was flipped from 'no' to 'yes'. This shard represented a malicious client trying to poison the model into misclassifying non-suscribers to term deposits as subscribers.

5.2 Detection of malicious updates

We compared four approaches to detect malicious updates on the generated data sets:

1. *FL baseline.* In the FL baseline experiment we trained a federated learning model using a distance-based method to detect outliers as in Domingo-Ferrer et al. (2020). In summary, an average update was computed from all client-provided updates. Then, the Euclidean distance between every individual update and the average update was computed. All updates whose distance fell outside the bounds in Expression (2) with $\tau = 1.5$ were considered malicious and thus discarded. *This FL baseline must not be confused with the centralized baseline of Tables 2, 3 and 4.*
2. *Microaggregation.* The second experiment modeled FL with microaggregation-based attack detection. We used different parameters for the experiments to show how varying them affects the metrics. The local learning steps varied among 1, 2 and 5, and for parameter k we chose 1, 3 and 5. Note that *microaggregation with $k = 1$ (no clusters) yields the FL baseline described in the previous paragraph.*

3. *GMM.* This experiment tested the GMM-based approach. We used the GMM implementation in Scikit-learn (Pedregosa et al., 2011), with parameters $K = n_components = 3$ (3 mixture components) and $covariance_type =$ "$full$" (each component had its own general covariance matrix). Any update whose log-likelihood fell below $\tau = -20,000$ was considered malicious and was discarded. We used different parameters to see how this affected the metrics. The local learning steps were the same as in the previous experiment, the mean μ took values 0 and 2, and the covariance σ^2 took values 1 and 4. To get the optimal number of clusters, we used the Bayesian Information Criterion (BIC) function. The optimal value is the one that minimizes BIC.
4. *DBSCAN.* The last experiment was similar to the previous one, but using Algorithm 3 to detect malicious updates. The algorithm used the DBSCAN implementation provided in Scikit-learn with parameters $Eps = 0.5, 3, 5$, and $minPts$ depending on the data set as follows:
   - Adult Income data set: $minPts = 5, 15, 28$.
   - Athletes data and Bank Marketing data sets: $minPts = 5, 18, 34$.

   Values for Eps other than 0.5 and values for $MinPts$ other than 5 were selected following the procedure in Section 4.2 of Sander et al. (1998).

5.3 Performance measures and discussion

In Table 5 we count the number of good updates sent by genuine minority clients but flagged as malicious, for each of the four approaches and for each of the three data sets. The numbers are cumulative over all epochs (100 epochs were used for all approaches). Clearly, the three methods we propose misclassified genuine minority updates as malicious in a smaller proportion than the FL baseline for the three studied data sets. In particular, the method based on microaggregation offered the best results among our three proposals.

Table 5: Number of genuine minority updates misclassified as malicious

	FL baseline	***Microaggregation***	***GMM***	***DBSCAN***
Adult data set	183	102	131	126
Athletes data set	385	213	227	234
Bank Marketing data set	271	158	172	164

Further, we used Accuracy and ROC AUC as performance metrics for majority, minority, and attacker clients. The basic objective of our mechanisms was to increase these performance metrics for both majority and minority clients while ensuring attackers achieved worse results. Additionally, we used the above mentioned Predictive Equality (PE) and Equal Opportunity (EO) metrics to detect the presence of unfairness towards the majority or the minority. According to these two metrics, the closer PE and EO to 0, the fairer is a method[2].

Results are summarized in the following tables. Tables 6, 7 and 8 report the above metrics for the microaggregation experiment with different parameter k and learning steps LS. Tables 9, 10 and 11 report the above metrics for the GMM experiment with different mean, covariance and learning steps. Finally, Tables 12, 13 and 14 report the above metrics for the DBSCAN experiment with different Eps, $minPts$, and learning steps.

The results show that all methods perform comparably or slightly better than the FL baseline scenario (microaggregation with $k = 1$) in terms of accuracy and ROC AUC. These results indicate that our methods allow the models to better capture the differences present between the majority and minority groups, while still being able to discard malicious updates. In all cases, attacker clients achieve worse accuracy than legitimate majority and minority clients. Additionally, all methods

[2] Note that fairness metrics computed in what follows refer to *clients*, more precisely to the decision *made by the model manager* to accept or reject a client's update. This is different from fairness referred to *subjects*, when the decision is *made by the classifier* to classify a subject's record into the positive or negative category (*e.g.* > \$50K, resp. ≤ \$50K in the case of Adult). To avoid confusion between both types of fairness, we did not compute PE or EO in the centralized baseline tables (Tables 2, 3, and 4).

Table 6: Performance measures for Adult Income data set with different microaggregation parameters. 'B' stands for black (minority clients), 'NB' for non-black (majority clients), 'Att' for attacker clients.

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	B	NB	Att	B	NB	Att	—	—
k=1, LS=1	0.9003	0.9307	0.7712	0.82	0.83	0.68	0.0323	0.238
k=1, LS=2	0.9125	0.9362	0.8298	0.82	0.84	0.72	0.0271	0.0205
k=1, LS=5	0.9072	0.9251	0.7809	0.83	0.83	0.71	0.0308	0.0251
k=3, LS=1	0.9395	0.9591	0.7421	0.84	0.85	0.72	0.0097	0.0061
k=3, LS=2	0.9594	0.9873	0.8406	0.86	0.87	0.73	0.0022	0.031
k=3, LS=5	0.9775	0.9849	0.7421	0.89	0.90	0.78	0.0035	0.0029
k=5, LS=1	0.92	0.9401	0.8103	0.84	0.83	0.73	0.0106	0.0095
k=5, LS=2	0.9261	0.9387	0.8164	0.83	0.84	0.71	0.0182	0.0206
k=5, LS=5	0.9122	0.9394	0.8027	0.82	0.83	0.72	0.0213	0.0285

Table 7: Performance measures for Athletes data set with different microaggregation parameters. 'SA' stands for South Asian (minority clients), 'NSA' for non-South-Asian (majority clients), 'Att' for attacker clients.

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	SA	NSA	Att	SA	NSA	Att	—	—
k=1, LS=1	0.9227	0.9546	0.8115	0.83	0.85	0.71	0.0118	0.0241
k=1, LS=2	0.9395	0.9591	0.802	0.82	0.85	0.72	0.0172	0.0283
k=1, LS=5	0.9282	0.9487	0.8146	0.82	0.84	0.71	0.0105	0.0199
k=3, LS=1	0.9582	0.9721	0.7903	0.87	0.89	0.69	0.0081	0.0065
k=3, LS=2	0.9689	0.9984	0.8049	0.88	0.9	0.72	0.0037	0.0103
k=3, LS=5	0.9508	0.9714	0.8021	0.87	0.89	0.71	0.0073	0.0091
k=5, LS=1	0.9372	0.9522	0.8241	0.85	0.86	0.73	0.011	0.0183
k=5, LS=2	0.9388	0.9564	0.8173	0.84	0.86	0.72	0.0256	0.0174
k=5, LS=5	0.9261	0.947	0.8153	0.84	0.85	0.72	0.0234	0.0201

Table 8: Performance measures for Bank Marketing data set with different microaggregation parameters. 'D' stands for divorced (minority clients), 'ND' stands for non-divorced (majority clients), 'Att' for attacker clients.

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	D	ND	Att	D	ND	Att	—	—
k=1, LS=1	0.9106	0.9212	0.7572	0.81	0.83	0.68	0.0147	0.0132
k=1, LS=2	0.9163	0.9387	0.7681	0.83	0.84	0.72	0.0212	0.0263
k=1, LS=5	0.9201	0.93	0.7657	0.83	0.83	0.72	0.0178	0.0224
k=3, LS=1	0.9695	0.9691	0.7521	0.86	0.87	0.74	0.0098	0.0113
k=3, LS=2	0.9738	0.9888	0.7806	0.87	0.89	0.75	0.0096	0.0077
k=3, LS=5	0.9572	0.9614	0.7521	0.87	0.88	0.74	0.0103	0.0094
k=5, LS=1	0.9362	0.9584	0.7662	0.86	0.86	0.73	0.0152	0.188
k=5, LS=2	0.9344	0.9482	0.78	0.85	0.86	0.73	0.0183	0.0205
k=5, LS=5	0.9272	0.9439	0.7638	0.85	0.85	0.72	0.0175	0.0226

reduce in most cases the differences between majority and minority groups with respect to FL baseline; this can be observed with the EO and PE metrics.

Regarding the different parameters, we can see in the results for microaggregation that the accuracy with $k = 3$ is better than with $k = 1$. However, further increasing k to 5 decreases accuracy. A plausible explanation is that larger values of k yield larger clusters, which entails some information loss and thus a performance degradation. Thus, $k = 3$ seems best for accuracy, and it also yields the best values (closer to 0) for fairness metrics PE and EO between majority and minority groups.

Table 9: Performance measures for Adult Income data set with different GMM parameters

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	B	NB	Att	B	NB	Att	—	—
$\mu = 0, \sigma^2 = 1$, **LS=1**	0.9117	0.9398	0.7251	0.83	0.83	0.72	0.042	0.0158
$\mu = 0, \sigma^2 = 1$, **LS=2**	0.9295	0.9401	0.7288	0.83	0.84	0.71	0.0301	0.0242
$\mu = 0, \sigma^2 = 1$, **LS=5**	0.9272	0.9464	0.73	0.83	0.84	0.71	0.0323	0.0168
$\mu = 0, \sigma^2 = 4$, **LS=1**	0.939	0.9591	0.7321	0.84	0.84	0.70	0.0285	0.0315
$\mu = 0, \sigma^2 = 4$, **LS=2**	0.9208	0.9446	0.74	0.83	0.83	0.71	0.0273	0.0205
$\mu = 0, \sigma^2 = 4$, **LS=5**	0.9384	0.9516	0.7461	0.84	0.85	0.72	0.0207	0.0186
$\mu = 2, \sigma^2 = 4$, **LS=1**	0.9473	0.9628	0.7582	0.87	0.89	0.73	0.0096	0.105
$\mu = 2, \sigma^2 = 4$, **LS=2**	0.9578	0.9763	0.7534	0.88	0.91	0.74	0.0062	0.0091
$\mu = 2, \sigma^2 = 4$, **LS=5**	0.9455	0.9682	0.7495	0.87	0.89	0.73	0.0071	0.0112

Table 10: Performance measures for Athletes data set with different GMM parameters

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	SA	NSA	Att	SA	NSA	Att	—	—
$\mu = 0, \sigma^2 = 1$, **LS=1**	0.9109	0.9423	0.7762	0.83	0.84	0.69	0.0289	0.0213
$\mu = 0, \sigma^2 = 1$, **LS=2**	0.9154	0.9365	0.7864	0.84	0.84	0.7	0.0162	0.0197
$\mu = 0, \sigma^2 = 1$, **LS=5**	0.9277	0.9488	0.7742	0.83	0.83	0.7	0.0256	0.0301
$\mu = 0, \sigma^2 = 4$, **LS=1**	0.9293	0.9491	0.7821	0.84	0.84	0.71	0.0143	0.0106
$\mu = 0, \sigma^2 = 4$, **LS=2**	0.9207	0.9456	0.7895	0.83	0.85	0.71	0.0187	0.0259
$\mu = 0, \sigma^2 = 4$, **LS=5**	0.9156	0.9327	0.7759	0.83	0.83	0.7	0.0175	0.0201
$\mu = 2, \sigma^2 = 4$, **LS=1**	0.9476	0.9702	0.8143	0.86	0.89	0.73	0.0084	0.0107
$\mu = 2, \sigma^2 = 4$, **LS=2**	0.9587	0.9765	0.8278	0.87	0.89	0.73	0.0051	0.0038
$\mu = 2, \sigma^2 = 4$, **LS=5**	0.9443	0.9631	0.8109	0.87	0.88	0.72	0.0096	0.012

Table 11: Performance measures for Bank Marketing data set with different GMM parameters

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	D	ND	Att	D	ND	Att	—	—
$\mu = 0, \sigma^2 = 1$, **LS=1**	0.8906	0.9273	0.8064	0.82	0.84	0.69	0.0359	0.0401
$\mu = 0, \sigma^2 = 1$, **LS=2**	0.9159	0.9365	0.7929	0.83	0.84	0.7	0.0225	0.0362
$\mu = 0, \sigma^2 = 1$, **LS=5**	0.9066	0.9284	0.7942	0.84	0.83	0.7	0.0274	0.0207
$\mu = 0, \sigma^2 = 4$, **LS=1**	0.9224	0.9391	0.7831	0.84	0.84	0.71	0.0148	0.0262
$\mu = 0, \sigma^2 = 4$, **LS=2**	0.9282	0.9256	0.7995	0.83	0.85	0.71	0.0197	0.0211
$\mu = 0, \sigma^2 = 4$, **LS=5**	0.9144	0.9397	0.8059	0.83	0.84	0.7	0.025	0.0193
$\mu = 2, \sigma^2 = 4$, **LS=1**	0.9376	0.9551	0.8142	0.86	0.89	0.73	0.0123	0.0085
$\mu = 2, \sigma^2 = 4$, **LS=2**	0.9554	0.9672	0.8178	0.87	0.89	0.73	0.011	0.0152
$\mu = 2, \sigma^2 = 4$, **LS=5**	0.9437	0.9614	0.8109	0.87	0.88	0.72	0.0097	0.0104

In the case of GMM, the results improve as we increase the mean. When the means are too low, then the maximum-likelihood of the model fits Gaussians that may encompass legitimate users not distinguishable from malicious ones.

For the last method, with low $minPts$, the outliers are more clear. This is because with a higher number of $minPts$, smaller clusters will be incorporated into the larger ones, making it difficult to differentiate between majority and minority groups. Moreover, the accuracy and ROC AUC are better when minPts are lower.

Also, for the three methods, taking $LS = 2$ local learning steps appears as a better choice than $LS = 1, 5$.

Again, this allows us to conclude that our proposed outlier detection mechanisms are capable of distinguishing between genuine minority groups and attackers. In particular, the microaggregation-based method achieves the best performance in most cases. This was to be expected because microaggregation implements a finer-grained assessment of inter-client likeness not only to a prototypical majority, but to prototypical minorities. In this way, minority groups are properly (and fairly) considered and only true outliers within these minority groups are discarded.

Table 12: Performance measures for the Adult Income data set with different DBSCAN parameters

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	B	NB	Att	B	NB	Att	—	—
$Eps = 0.5$, **minPts=5, LS=1**	0.9632	0.9861	0.7994	0.89	0.91	0.72	0.0062	0.0031
$Eps = 0.5$, **minPts=5, LS=2**	0.9777	0.9636	0.7885	0.9	0.9	0.71	0.0045	0.0081
$Eps = 0.5$, **minPts=5, LS=5**	0.9592	0.9622	0.7839	0.89	0.9	0.71	0.0093	0.0106
$Eps = 3$, **minPts=15, LS=1**	0.94	0.9546	0.8047	0.85	0.85	0.7	0.0126	0.0174
$Eps = 3$, **minPts=15, LS=2**	0.9584	0.9603	0.8021	0.85	0.86	0.69	0.0117	0.0109
$Eps = 3$, **minPts=15, LS=5**	0.9401	0.9514	0.7962	0.84	0.85	0.7	0.0183	0.0196
$Eps = 5$, **minPts=28, LS=1**	0.9225	0.9347	0.8139	0.83	0.84	0.7	0.0241	0.0273
$Eps = 5$, **minPts=28, LS=2**	0.9332	0.9289	0.8016	0.84	0.84	0.71	0.0191	0.0262
$Eps = 5$, **minPts=28, LS=5**	0.9187	0.9311	0.7812	0.83	0.84	0.72	0.0285	0.0306

Table 13: Performance measures for the Athletes data set with different DBSCAN parameters

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	SA	NSA	Att	SA	NSA	Att	—	—
$Eps = 0.5$, **minPts=5, LS=1**	0.9505	0.9682	0.8082	0.88	0.89	0.76	0.0105	0.0138
$Eps = 0.5$, **minPts=5, LS=2**	0.9602	0.9688	0.7943	0.89	0.9	0.75	0.0078	0.0063
$Eps = 0.5$, **minPts=5, LS=5**	0.9558	0.9592	0.8031	0.88	0.89	0.76	0.0097	0.0115
$Eps = 3$, **minPts=18, LS=1**	0.9372	0.9546	0.8147	0.85	0.85	0.71	0.0167	0.0199
$Eps = 3$, **minPts=18, LS=2**	0.9434	0.9501	0.8104	0.85	0.86	0.7	0.0202	0.0186
$Eps = 3$, **minPts=18, LS=5**	0.9386	0.9427	0.8175	0.84	0.83	0.7	0.0231	0.0295
$Eps = 5$, **minPts=34, LS=1**	0.9269	0.9335	0.8268	0.83	0.84	0.72	0.0282	0.0334
$Eps = 5$, **minPts=34, LS=2**	0.9037	0.9284	0.82	0.82	0.83	0.7	0.0227	0.0285
$Eps = 5$, **minPts=34, LS=5**	0.8948	0.9005	0.8293	0.82	0.82	0.73	0.0312	0.0294

Table 14: Performance measures for the Bank Marketing data set with different DBSCAN parameters

	Accuracy			**ROC AUC**			**PE**	**EO**
Parameters	D	ND	Att	D	ND	Att	—	—
$Eps = 0.5$, **minPts=5, LS=1**	0.9522	0.9568	0.8121	0.88	0.89	0.72	0.0061	0.0074
$Eps = 0.5$, **minPts=5, LS=2**	0.9641	0.9759	0.8195	0.89	0.88	0.71	0.0094	0.0103
$Eps = 0.5$, **minPts=5, LS=5**	0.9548	0.9607	0.805	0.88	0.88	0.71	0.0114	0.0101
$Eps = 3$, **minPts=18, LS=1**	0.9476	0.9532	0.8048	0.85	0.86	0.7	0.0182	0.0192
$Eps = 3$, **minPts=18, LS=2**	0.94	0.9503	0.8021	0.85	0.86	0.7	0.0174	0.0152
$Eps = 3$, **minPts=18, LS=5**	0.9366	0.9407	0.7932	0.84	0.85	0.7	0.0252	0.0138
$Eps = 5$, **minPts=34, LS=1**	0.913	0.9328	0.8039	0.82	0.84	0.7	0.0322	0.0285
$Eps = 5$, **minPts=34, LS=2**	0.9174	0.9349	0.7963	0.83	0.84	0.71	0.0387	0.0393
$Eps = 5$, **minPts=34, LS=5**	0.9021	0.9136	0.7812	0.82	0.82	0.71	0.0372	0.0414

Finally, from the related work approaches mentioned in Section 3 for non-i.i.d. data in FL, we took Zhao et al. (2018) and its implementation[3]. We implemented our three methods on top of their approach and measured how much performance improvement they brought on the non-i.i.d. shards described above for the Adult data set. Table 15 shows the results, where FL_Zhao, Microaggregation_Zhago, GMM_Zhao, and DBSCAN_Zhao are, respectively, the method in Zhao et al. (2018), and our microaggregation, GMM, and DBSCAN-based methods on top of Zhao et al. (2018). The reported accuracy is for the minority clients, that is, those with shards corresponding to black individuals. See the table caption about the parameters used in the methods. We observe that using any of our methods improves on the plain method of Zhao et al. (2018). In particular, we see that the best results are obtained with our microaggregation method.

[3] https://github.com/yjlee22/FedShare

Table 15: Performance measures for Adult Income data set with Zhao et al. (2018) approach. The reported accuracy is for the minority clients. Microaggregation parameter: $k = 3$. GMM parameters: $\mu = 0$, $\sigma^2 = 1$. DBSCAN parameters: $Eps = 0.5$, $minPts = 5$. Local learning steps: $LS = 1$ in all methods.

	FL_Zhao	***Microaggregation_Zhao***	***GMM_Zhao***	***DBSCAN_Zhao***
Accuracy	0.8108	0.9075	0.9023	0.864
PE	0.0195	0.009	0.0103	0.0110
EO	0.0351	0.0192	0.0201	0.0273

6 Conclusions and future research

In this work, we have dealt with the problem of distinguishing abnormal/malicious behaviors from legitimate ones in federated learning. We focus on scenarios with clients having legitimate minority data, whose updates are likely to be classified as outlying/malicious by the standard attack detection mechanisms proposed in the literature. To make progress towards fair attack detection, we propose three different methods, one based on microaggregation, another based on the Gaussian mixture model and the third one based on DBSCAN.

To evaluate and compare the performance of these methods, we computed standard evaluation metrics, namely accuracy, ROC AUC, PE and EO. Our results indicate that the microaggregation method is especially effective at differentiating malicious model updates from normal (even minority) model updates. This results in improvements in all observed evaluation metrics. From a more qualitative perspective, our approach avoids discriminating minority groups.

Beyond fairness being an ethical value to be satisfied, it brings rewards in terms of model quality: taking into account the updates from minority groups enriches the resulting model.

As future work, we plan to apply similar approaches to those proposed in this paper to protect against other kinds of poisoning attacks, such as collusion attacks, while taking the fairness of the classification tasks into account.

Acknowledgements We are indebted to the late David Rebollo-Monedero for his helpful comments and contributions on an earlier version of this paper. The following funding sources are gratefully acknowledged: European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab"), the Government of Catalonia (ICREA Acadèmia Prize to J. Domingo-Ferrer), and the Spanish MCIN/AEI /10.13039/501100011033 /FEDER, UE under project PID2021-123637NB-I00 "CURLING". The authors are with the UNESCO Chair in Data Privacy, but the views in this paper are their own and are not necessarily shared by UNESCO.

References

Bhagoji AN, Chakraborty S, Mittal P, Calo S (2019) Analyzing federated learning through an adversarial lens. In: International Conference on Machine Learning, PMLR, pp 634–643

Blanchard P, Guerraoui R, Stainer J, et al. (2017) Machine learning with adversaries: Byzantine tolerant gradient descent. In: Advances in Neural Information Processing Systems, pp 119–129

Blanco-Justicia A, Domingo-Ferrer J, Martínez S, Sánchez D, Flanagan A, Tan KE (2020) Achieving security and privacy in federated learning systems: survey, research challenges and future directions. arXiv preprint arXiv:201206810

Domingo-Ferrer J, Mateo-Sanz JM (2002) Practical data-oriented microaggregation for statistical disclosure control. IEEE Transactions on Knowledge and data Engineering 14(1):189–201

Domingo-Ferrer J, Torra V (2005) Ordinal, continuous and heterogeneous k-anonymity through microaggregation. Data Mining and Knowledge Discovery 11(2):195–212

Domingo-Ferrer J, Blanco-Justicia A, Sánchez D, Jebreel N (2020) Co-utile peer-to-peer decentralized computing. In: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), IEEE, pp 31–40

Du W, Xu D, Wu X, Tong H (2021) Fairness-aware agnostic federated learning. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), SIAM, pp 181–189

Dua D, Graff C (2017) Uci machine learning repository. `http://archive.ics.uci.edu/ml`

Ester M, Kriegel HP, Sander J, Xu X (1996) Density-based spatial clustering of applications with noise. In: Int. Conf. Knowledge Discovery and Data Mining, vol 240, p 6

Fang M, Cao X, Jia J, Gong N (2020) Local model poisoning attacks to byzantine-robust federated learning. In: 29th USENIX Security Symposium (USENIX Security 20), pp 1605–1622

Gander M, Felderer M, Katt B, Tolbaru A, Breu R, Moschitti A (2012) Anomaly detection in the cloud: Detecting security incidents via machine learning. In: International Workshop on Eternal Systems, Springer, pp 103–116

George A, Vidyapeetham A (2012) Anomaly detection based on machine learning: dimensionality reduction using pca and classification using svm. International Journal of Computer Applications 47(21):5–8

Griffin RH (2018) 120 years of Olympic history: Athletes and results. `https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results`

Hsieh K, Phanishayee A, Mutlu O, Gibbons P (2020) The non-iid data quagmire of decentralized machine learning. In: International Conference on Machine Learning, PMLR, pp 4387–4398

Jeong E, Oh S, Kim H, Park J, Bennis M, Kim SL (2018) Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. arXiv preprint arXiv:181111479

Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, et al. (2019) Advances and open problems in federated learning. arXiv preprint arXiv:191204977

Khandpur Singh A, Blanco-Justicia A, Domingo-Ferrer J, Sánchez D, Rebollo Monedero D (2020) Fair detection of poisoning attacks in federated learning. In: 32th IEEE International Conference on Tools with Artificial Intelligence – ICTAI 2020, IEEE, pp 224–229

Konečnỳ J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D (2016) Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:161005492

Li S, Cheng Y, Liu Y, Wang W, Chen T (2019a) Abnormal client behavior detection in federated learning. arXiv preprint arXiv:191009933

Li S, Cheng Y, Wang W, Liu Y, Chen T (2020) Learning to detect malicious clients for robust federated learning. arXiv preprint arXiv:200200211

Li T, Hu S, Beirami A, Smith V (2021) Ditto: Fair and robust federated learning through personalization. In: International Conference on Machine Learning, PMLR, pp 6357–6368

Li X, Huang K, Yang W, Wang S, Zhang Z (2019b) On the convergence of fedavg on non-iid data. arXiv preprint arXiv:190702189

Lyu L, Xu X, Wang Q, Yu H (2020) Collaborative fairness in federated learning. In: Federated Learning, Springer, pp 189–204

McMahan B, Moore E, Ramage D, Hampson S, Aguera-y Arcas B (2017) Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, PMLR, pp 1273–1282

Narayanan A (2018) Translation tutorial: 21 fairness definitions and their politics. In: Proc. Conf. Fairness Accountability Transp., New York, USA, vol 1170

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. (2011) Scikit-learn: Machine learning in python. the Journal of machine Learning research 12:2825–2830

S Moro PC, Rita P (2014) Bank marketing dataset. `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing`

Sander J, Ester M, Kriegel HP, Xu X (1998) Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data mining and knowledge discovery 2(2):169–194

Torra V (2004) Microaggregation for categorical variables: a median based approach. In: International Workshop on Privacy in Statistical Databases, Springer, pp 162–174

Verma S, Rubin J (2018) Fairness definitions explained. In: 2018 ieee/acm international workshop on software fairness (fairware), IEEE, pp 1–7

Wang J, Charles Z, Xu Z, Joshi G, McMahan HB, Aguera-y Arcas B, Al-Shedivat M, Andrew G, Avestimehr S, Daly K, Data D, Diggavi S, Eichner H, Gadhikar A, Garrett Z, Girgis AM, Hanzely F, Hard A, He C, Horvath S, Huo Z, Ingerman A, Jaggi M, Javidi T, Kairouz P, Kale S, Karimireddy SP, Konecny J, Koyejo S, Li T, Liu L, Mohri M, Qi H, Reddi SJ, Richtarik P, Singhal K, Smith V, Soltanolkotabi M, Song W, Suresh AT, Stich SU, Talwakar A, Wang H,

Woodworth B, Wu S, Yu FX, Yuan H, Zaheer M, Zhang M, Zhang T, Zheng C, Zhu C, Zhu W (2021) A field guide to federated optimization. arXiv preprint arXiv:210706917v1

Yin D, Chen Y, Ramchandran K, Bartlett P (2018) Byzantine-robust distributed learning: Towards optimal statistical rates. arXiv preprint arXiv:180301498

Zhang X, Hong M, Dhople S, Yin W, Liu Y (2020) Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. arXiv preprint arXiv:200511418

Zhao Y, Li M, Lai L, Suda N, Civin D, Chandra V (2018) Federated learning with non-iid data. arXiv preprint arXiv:180600582