Fair Detection of Poisoning Attacks in Federated Learning

Ashneet Khandpur Singh, Alberto Blanco-Justicia, Josep Domingo-Ferrer, David Sánchez, David Rebollo-Monedero
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, E-43007 Tarragona, Catalonia
{ashneet.singh, alberto.blanco, josep.domingo, david.sanchez, david.rebollo}@urv.cat

***Abstract*—Federated learning is a decentralized machine learning technique that aggregates partial models trained by a set of clients on their own private data to obtain a global model. This technique is vulnerable to security attacks, such as model poisoning, whereby malicious clients submit bad updates in order to prevent the model from converging or to introduce artificial bias in the classification. Applying anti-poisoning techniques might lead to the discrimination of minority groups whose data are significantly and legitimately different from those of the majority of clients. 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 a standard machine learning data set shows that employing our approach to tell legitimate from malicious updates produces models that are more accurate than those obtained with standard poisoning detection techniques.**

***Index Terms*—Federated learning, Security, Privacy, Fairness.**

I. Introduction

Technology advances allow exploiting ever larger volumes of data about individuals. However useful, this can also be worrisome from the privacy standpoint. Privacy concerns have led to the enactment of strict regulations, such as the European Union's General Data Protection Regulation (GDPR), which is binding for any organization operating in the EU.

To reduce centralized data collection for training machine learning models, a new machine learning technique called federated learning [11], [12] has been proposed. Federated learning works in a decentralized manner and allows learning a model based on data that are privately owned by a set of clients without the need to ever upload these data to a central server.

More specifically, a model manager first sends an initial machine learning model to the clients. Then, each client computes an update of the model based on her private data, and sends the update back to the manager. The manager aggregates these data and updates the global model, which is sent back again to the clients. The process is iterated until the model converges.

However, federated learning is vulnerable to security attacks whereby malicious clients sabotage the learning process by sending bad model updates. These attacks might result in the model failing to converge (Byzantine attacks) or to converge to biased models that do not truly represent the data (poisoning attacks). Poisoning attacks are described in [3], along with several solutions that thwart them. In [2], more effective attacks are investigated, that achieve 100% accuracy on the attacker's task within just a single learning round.

Techniques to prevent security attacks (discussed in Section II) 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 is effective to prevent attacks, systematically rejecting outlying updates might also lead to unfair trained models [15] if the outlying data correspond to a legitimately different minority.

Contribution and plan of this paper

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. We present two approaches to properly distinguish members of minority groups from potential model poisoners when carrying out robust aggregation of updates:

- The first approach is based on microaggregation [7]. 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, a federated learning 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 the sizes

of the clusters.
- In the second approach, we use Gaussian mixture models to characterize the distribution of the client-provided updates. Outliers are spotted in a more sophisticated way than just relying on the distance to an average client update. By signaling the presence of minority groups that differ from the majority group in some attributes, but that are homogeneous within themselves, this approach can be expected to identify honest individuals from minority groups as non-malicious.

The rest of this paper is organized as follows. Section II discusses related works on attack mitigation techniques in federated learning. Sections III and IV present our methods for fair detection of attacks based on microaggregation and Gaussian mixture models, respectively. Section V reports empirical results that illustrate the effectiveness of our methods. Finally, conclusions and future research lines are discussed in Section VI.

II. Related work

In federated learning the following attack detection methods have been proposed:

- Detection of malicious clients via model metrics. In [5], the authors propose a Byzantine gradient descent method based on the geometric median of means of the gradient. Any client updates that degrade the model performance are identified as malicious and discarded.
- 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 [18]. The model manager can compute how much do the distributions of distances in successive iterations change, for example using the Kullback-Leibler divergence metric.
- Krum aggregation. The authors of [4] propose an aggregation function that is resilient against f malicious clients. This function is called Krum. They show that averaging does not stand Byzantine attacks, while Krum does.The authors also evaluate a variant of Krum, Multi-Krum, which interpolates between Krum and averaging.
- Coordinate-wise median. In [18], 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 [18], 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 average/aggregation/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. The methods we introduce in the next sections aim at properly managing updates originated by clients in minority groups.

III. 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 [7]. It 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 all the records in the cluster, which is used as a replacement of the original records.

In our approach, we are interested only in the partition phase, where similar clients will be clustered 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 following solution based on collaborative microaggregation to distinguish malicious clients from genuine outliers/minority group members, which we will call in what follows *protected clients*:

1) Let each protected client publish, together with her pseudonym, demographic attribute values that characterize her as a member of a minority group, e.g. {Sex=female, Age=young, Ethnicity=black}.
2) The model manager and protected clients engage in decentralized microaggregation as specified in Protocol 1 to microaggregate protected clients according to their published attributes. Take microaggregation parameter k sufficiently large so that outliers can be distinguished within a group of k, and a collusion of k clients or of a significant proportion of k clients is unlikely.
3) For each cluster C of the microaggregation:
   a) Compute the centroid c_C of the updates sent by clients in C.
   b) Let λ_C be the set of clients $\mathcal{P}_i \in C$ such that the distance $dist_i$ from $\mathcal{P}_i$'s update to the centroid of the updates is not an outlier, more precisely

$$\lambda_C = \{\mathcal{P}_i \in C | Q1 - \tau IQR \leq dist_i \leq Q3 + \tau IQR\}, \quad (1)$$

   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.

c) If a client $\mathcal{P}_i \in \lambda_C$, then the client's update is considered a legitimate one because it is similar to most updates in $\mathcal{P}_i$'s cluster C. If a client $\mathcal{P}_i \in C \setminus \lambda_C$, then $\mathcal{P}_i$'s update is considered malicious because it is too outlying even for the minority group represented by C.

Protocol 1: Decentralized microaggregation

Input: Clients $\mathcal{P}_1, \ldots, \mathcal{P}_c$; microaggregation parameter k

1 Let qi_i be demographic attribute values of $\mathcal{P}_i$, for $i = 1, \ldots, c$;
2 Each $\mathcal{P}_i$ publishes qi_i;
3 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 $k \leq |C_j| \leq 2k - 1$;
4 The model manager publishes C_1, C_2, ..., C_p;
5 Each $\mathcal{P}_i$ can compute the above clusters and verify they are correct, and check that the cluster $C_{\mathcal{P}_i}$ where qi_i belongs contains k or more quasi-identifiers;

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, 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 1 we use the maximum distance to average vector (MDAV) algorithm, which is the most widely used microaggregation algorithm [7]. See Algorithm 1.

Algorithm 1: MDAV microaggregation algorithm

Input: Data set R, microaggregation parameter k

1 **while** $|R| \geq 3k$ **do**
2 Compute the average record $\bar{x}$ of all records in R;
3 Consider the most distant record x_r to $\bar{x}$;
4 Find the most distant record x_s from x_r considered in the previous step;
5 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;
6 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;
7 **end**
8 **if** *there are between* $3k-1$ *and* $2k$ *records in* R **then**
9 Compute the average record $\bar{x}$ of the remaining records in R;
10 Find the most distant record x_r from $\bar{x}$;
11 Form a cluster containing x_r and the $k-1$ records closest to x_r;
12 Form another cluster containing the rest of records;
13 **else**
14 Form a new cluster with the remaining records;
15 **end**

IV. Fair attack detection based on Gaussian mixtures and expectation-maximization

In this section, we propose a second approach to solve the problem of fair attack detection in federated learning based on Gaussian mixtures and the expectation-maximization algorithm.

Consider a data set $\{x_j\}_{j=1}^n$ of points in $\mathbb{R}^m$, representing client-provided model updates. Recall that the maximum-likelihood (ML) estimates of the mean μ and covariance Σ parameters of a multivariate Gaussian model

$$q_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^m \det \Sigma}} e^{-\frac{1}{2}\|x-\mu\|^2_{\Sigma^{-1}}}$$

match their respective sample moments. This means that we may view these moments as an expectation and covariance under a probability mass function p_X representing the relative frequencies of the observed samples, typically uniform and therefore equal to $1/n$ in the case of continuous alphabets, with samples that may now be construed as a random vector X distributed according to p_X. Precisely, the ML estimates are

$$\hat{\mu}_{\mathrm{ML}} = \mathrm{E}_{p_X} X = \frac{1}{n}\sum_{j=1}^n x_j \text{ and}$$
$$\hat{\Sigma}_{\mathrm{ML}} = \mathrm{Cov}_{p_X} X = \frac{1}{n}\sum_{j=1}^n (x_j - \mu)(x_j - \mu)^{\mathrm{T}}.$$

Under the simple assumption of uncorrelated, identically distributed dimensions and after convenient scaling, we may restrict the covariance parameter Σ to be the identity matrix, which finally leads us to conclude that the probability density $q_{\mu,I}(x)$ of any given point x is a decreasing function of the Euclidean norm $\|x - \mu\|$ with respect to the mean parameter μ.

The Gaussian mixture model (GMM) is a probabilistically weighted combination of Gaussian components each with its own mean and covariance. Representing the observed samples and the component index as random variables X and Y, respectively (one continuous, the other discrete), the GMM is a generative model with observation X and hidden label Y in which X given $Y = y$ is conditionally normally distributed with mean $\mu(y)$ and covariance $\Sigma(y)$. The weights of the components are the marginal probabilities $q_Y(y)$ of the component indices y. Therefore,

$$q_X(x) = \mathrm{E}_Y\, q_{X|Y}(x|Y) = \sum_y q_Y(y) q_{X|Y}(x|y)$$
$$= \sum_y q_Y(y) \frac{1}{\sqrt{(2\pi)^m \det \Sigma(y)}} e^{-\frac{1}{2}\|x-\mu(y)\|^2_{\Sigma(y)^{-1}}}.$$

This more expressive model comes with the challenge of selecting not only the means and covariances, but also the weights, jointly and in accordance with the ML criterion.

The joint estimation of the mean, covariance, and weight parameters for the GMM is typically accomplished via the expectation-maximization (EM) algorithm [17], guaranteed to monotonically converge to local optima under mild conditions [14]. Because of local convergence, the end result is ultimately driven by the initialization of the algorithm [13]. A simple, common choice for the mean parameters resorts to the k-means method, which itself requires initialization, for instance in the form of the k-means++ method [1].

The EM algorithm is an iterative method to find the maximum-likelihood estimates for model parameters in the presence of latent (hidden) variables.

To start, we need to pick the hyper parameter K that decides how many Gaussians we want in our model. The algorithm has the following two steps:

- E-step: For each point x_i, compute the probability that it belongs to cluster k. This can be written in terms of probability and using Bayes' theorem

$$r_{ik} = \frac{\mathcal{N}(x_i; \mu_k, \Sigma_k) \cdot \pi_k}{\sum_{j=1}^{k} \mathcal{N}(x_i; \mu_j, \Sigma_j) \cdot \pi_j}.$$

  This is just the normalized probability of each point belonging to one of the K Gaussians weighted by the mixture distribution (π_k). The value r_{ik} will be higher when the point is assigned to the right cluster and lower otherwise.
- M-step: After the E-step, we go back and update the π, μ and $\sum$ values. We want to maximize the likelihood function across all these values. These are updated in the following manner:
  1) The new density is defined by the ratio of the number of points in the cluster and the total number of points:

$$\pi_k = \frac{\text{number of points assigned to cluster}}{\text{total number of points}}$$

$$= \frac{1}{N} \sum_i r_{ik}.$$

     Next, we need to estimate the Gaussians.
  2) The mean and the covariance matrix are updated based on the values assigned to the distribution, in proportion to the probability values for the data point. Hence, a data point that has a higher probability of being part of that distribution will contribute a larger portion. This yields the following updates for the mean vector μ_k and the covariance matrix Σ_k:

$$\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}};$$

$$\Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r_{ik}}.$$

     Based on the updated values generated from this step, we calculate the new probabilities for each data point and update the values iteratively. This process is repeated in order to maximize the log-likelihood function.

To form clusters and find outliers, we use the model updates' weights and biases as data points.

During federated aggregation, we compute the per-sample average log likelihood of the data. The log-likelihood is a value that measures how good the fit is for any model. The higher the value, the better is the fit of the model; in contrast, an update yielding a low value may be a malicious outlier.

V. Empirical results

In this section we describe the experiments we conducted for the two proposed approaches and we discuss the obtained results.

As experimental data we used the Adult Income Dataset, which is a standard machine learning data set commonly used in the literature on data privacy. It was drawn from the 1994 United States Census Bureau data [8], and in it the learning task is to predict whether an individual will earn more or less than \$50,000 per year, according to a set of demographic attributes, and with two class values: $<$ \$50K and $\geq$ \$50K. The data set includes 14 demographic attributes that are a mix of categorical, ordinal, and numerical data types. There are a total of $48,842$ individual records, of which $3,620$ contain missing values, which we dropped, leaving $45,222$ records. We also dropped the $fnlwgt$ attribute, which is an artificial one, and recoded $Marital_status$ attribute into two values, $married$ and $unmarried$. We normalized numerical values and one-hot encoded categorical attributes.

Next, we preprocessed the data set in order to obtain three different sets of participants: a set of majority users, a set of users belonging to a minority and a set of attackers. We selected these sets taking into account the attributes that most contribute to the classification task; according to [9], these are the Sex and $Marital_status$ attributes. We obtained these sets according to the following rules:

1) In the entire data set consisting of $45,222$ records, we randomly chose half of the records with attribute $Sex = Female$ and flipped their assigned label to $>$ \$50K. We refer to these records as the *promoted women*.
2) Similarly, from the entire data set consisting of $45,222$ records, we randomly chose half of the records with attributes $Sex = Male$ and $Marital_status = unmarried$ and flipped their assigned label to $>$ \$50K. These are referred to as the *promoted unmarried men*.
3) By joining the two previous sets, we obtained the *promoted participants*. These are participants that differ from the majority users since, according to feature importance, women and unmarried men are less likely to earn more than \$50K.
4) Finally, to obtain the set of attackers, we created a set of records with randomly assigned attribute values.

We then defined 50 shards, following the approach of [12]. The step-by-step process was to sort the data, divide the data

into equally sized shards, and then assign one shard to each client. In the 50 shards, there were three types of clients:

- 30 shards contained unmodified records from the data set, and thus they corresponded to the majority clients. Each shard consisted of $1,500$ records, of which $1,394$ were reserved for training and 106 for testing.
- 15 shards contained the records of promoted participants. These shards corresponded to the minority clients. By doing the same as in the previous step, each shard consisted of $1,394$ records for training and 106 records for testing.
- The remaining 5 shards contained $1,500$ records each from the random data participants. These corresponded to the attacking clients.

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

1) *Baseline.* In the baseline experiment we trained a federated learning model using the 50 clients and detected outliers using the distance method in [6]. This represented the usual approach followed by attack detection mechanisms in the literature. 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 (1) with $\tau = 1.5$ were considered malicious and thus discarded.
2) *GMM.* In the second experiment, we collected all client-provided updates at the model manager, and fitted a GMM to the set of updates with parameters as defined in Scikit-learn [16], with $n_components = 3$ (3 mixture components) and $covariance_type = \text{“}full''$ (each component had its own general covariance matrix). Then, we obtained the log-likelihood (which lies in the range $(-\infty, 0]$) for each individual update belonging to the modeled distribution. Any update whose log-likelihood fell below $-20,000$ was considered a malicious outlier and was thus discarded.
3) Microaggregation. The last experiment modeled the microaggregation case. We split the 50 shards into two sets: the first set contained the 30 shards with unmodified records and 2.5 shards with random records; the second set contained the 15 shards with promoted participants' records and the remaining 2.5 shards with random records. Thus, the first set had 32.5 shards and the second one, 17.5 shards. Two models were independently trained using the two data sets. Malicious outliers were discarded using the distance criterion as in the baseline experiment. Finally, both models were merged by averaging them.

Performance measures and discussion

Having trained the baseline, the GMM and the microaggregation models, we evaluated them on the test data containing unmodified and promoted records. We recorded the following performance metrics:

- *Accuracy.* Number of correct predictions over all predictions made. We expected the accuracy to improve in the GMM and microaggregation models with respect to the baseline model, because legitimate client-provided updates were less likely to be discarded in the former two models.
- *Area under the ROC Curve (ROC AUC).* This metric measures discrimination between positive and negative classes. The receiver operating characteristic (ROC) curve visualizes the relationship between true positive and false positive rates over all possible acceptance thresholds. In [10], the authors show theoretically and empirically that AUC is a better measure than accuracy.
- *False Negative Rate (FNR).* We measured FNR on unmodified and promoted test sets independently. We expected the difference between FNR on unmodified and promoted test sets to decrease in the GMM and the microaggregation models with respect to the baseline model. A fair model should not only classify instances equally well regardless of the input data, but also fail at the same rate.

The results are summarized in Table I below.

TABLE I
PERFORMANCE MEASURES

	Baseline	***GMM***	***Microaggregation***
Accuracy	94.05%	93.45%	95.11%
ROC AUC	96.18%	96.89%	98.10%
FNR unmodified	4%	2.3%	1.7%
FNR promoted	0	0	0

Differences across the three approaches look minor at first sight, but we should consider that the baseline metrics were already very high (or very low) and, therefore, left little room for improvement. In this respect, the FNR for promoted clients was already perfect in the baseline.

Regarding our approaches, one the one hand we see that the GMM approach produces a model that is slightly better than the baseline regarding the most relevant metrics, ROC and FNR. On the other hand, the approach based on microaggregation improves on the baseline and the GMM approaches across all metrics. 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.

VI. 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 minority groups of clients, which would likely be classified as outlying/malicious by the standard attack detection mechanisms proposed in the literature. To make progress towards fair attack detection, we

propose two different methods, one based on microaggregation and another based on the Gaussian mixture model.

The empirical work we conducted on a standard machine learning data set shows 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.

As future work, we plan to further test our methods with other data sets and different configurations of attackers and minority groups. In particular, data sets with lower baseline accuracy will provide greater room to observe the differences across methods. We also plan to extend our work for non-i.i.d. data, which are especially challenging for statistical attack detection methods.

Acknowledgments and Disclaimer

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 grant 2017 SGR 705), and the Spanish Government (projects RTI2018-095094-B-C21 "Consent" and TIN2016-80250-R "Sec-MCloud"). 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

[1] D. Arthur and S. Vassilvitskii, "k-Means++: The advantages of careful seeding", in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SoDA), New Orleans, LA, Jan. 2007, pp. 10271035.

[2] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning", arXiv preprint arXiv:1807.00459, 2018.

[3] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens", arXiv preprint arXiv:1811.12470, 2018.

[4] P. Blanchard, E.-M. El-Mhambi, R. Guerraoui and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent", in Advances in Neural Information Processing Systems 30, 2017.

[5] Y. Chen, L. Su and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent", POMACS, vol. 25, pp. 1-44, 2017.

[6] J. Domingo-Ferrer, A. Blanco-Justicia, D. Sánchez and N. Jebreel, "Co-utile peer-to-peer decentralized computing", in 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing - CCGrid 2020, 2020, pp. 31-40.

[7] J. Domingo-Ferrer and V. Torra, "Ordinal, continuous and heterogeneous k-anonymity through microaggregation", Data Mining and Knowledge Discovery 11(2), 2005, pp. 195-212.

[8] D. Dua and C. Graff, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2019.

[9] R. Haffar, J. Domingo-Ferrer & D. Sánchez. "Explaining misclassification and attacks in deep learning via random forests", Modeling Decisions for Artificial Intelligence-MDAI 2020, pp. 273-285.

[10] J. Huang, and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms", IEEE Transactions on knowledge and Data Engineering, 17(3), 2005, pp. 299-310.

[11] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency", in in Advances in Neural Information Processing Systems, 2016.

[12] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data", arXiv preprint arXiv:1602.05629, 2016

[13] S. Michael and V. Melnykov, "An effective strategy for initializing the EM algorithm in finite mixture models", Advances in Data Analysis and Classification 10(4), 2016, pp. 563583.

[14] I. Naim and D. Gildea, "Convergence of the EM algorithm for Gaussian mixtures with unbalanced mixing coefficients", arXiv Preprint, no. 1206.6427, 2012.

[15] A. Narayanan. "Translation tutorial: 21 fairness definitions and their politics", In Proc. Conf. Fairness Accountability Transp., New York, USA, 2018.

[16] 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, 2011, pp. 2825-2830.

[17] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm", SIAM Review 26(2), 1984, pp. 195239.

[18] D. Yin, Y. Chen, K. Ramchandran and P. Bartlett, "Byzantine-robust distributed learning: towards optimal statistical rates", arXiv:1803.01498, 2018.