International Journal of Engineering

Journal Homepage: www.ije.ir

A Game-theoretic Approach for Robust Federated Learning

E. Tahanian*, M. Amouei, H. Fateh, M. Rezvani

Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran

PAPER INFO

Paper history:
Received 20 November 2020
Received in revised form 28 December 2020
Accepted 14 January 2021

Keywords:
Federated Learning
Game Theory
Byzantine Model
Adaptive Averaging

ABSTRACT

Federated learning enables aggregating models trained over a large number of clients by sending the models to a central server, while data privacy is preserved since only the models are sent. Federated learning techniques are considerably vulnerable to poisoning attacks. In this paper, we explore the threat of poisoning attacks and introduce a game-based robust federated averaging algorithm to detect and discard bad updates provided by the clients. We model the aggregating process with a mixed-strategy game that is played between the server and each client. The valid actions of the clients are to send good or bad updates while the server can accept or ignore these updates as its valid actions. By employing the Nash Equilibrium property, the server determines the probability of providing good updates by each client. The experimental results show that our proposed game-based aggregation algorithm is significantly more robust to faulty and noisy clients in comparison with the most recently presented methods. According to these results, our algorithm converges after a maximum of 30 iterations and can detect 100% of the bad clients for all the investigated scenarios. In addition, the accuracy of the proposed algorithm is at least 15.8% and 2.3% better than the state of the art for flipping and noisy scenarios, respectively.

doi*: 10.5829/ije.2021.34.04a.09*

1. INTRODUCTION

As datasets grow, the optimization of learning model parameters needs distribution across multiple machines. The idea of federated learning has recently been proposed, in which a shared global model is trained with the cooperation of a central server and some participants named clients [1-7], as can be seen in Figure 1. In other words, the clients train the model using their own local datasets and send it back to the central server. The server aggregates the information sent by the clients to update the shared global model. Afterward, the server sends the updated global model to some of the clients and this process is repeated again. Since the clients send only the model and not the data to the server, the data privacy is preserved.

One of the most important concerns about federated learning is sending bad updates by faulty or malicious clients. The researchers showed that only one bad client can compromise the model as well as the result in a convergence problem [4]. Thus, the researchers have tried to mitigate this problem by proposing different robust federated learning approaches [1, 8-10].

However, some of these techniques impose computational cost in comparison with the conventional averaging such as Federated Averaging (FA) [1], especially for a large number of clients. In addition, most of these techniques do not consider the number of data points used by each client to train the local models.

In this paper, we propose a Game-based robust Federated Averaging algorithm (GFA) to detect and discard bad updates provided by the clients. The proposed method uses an iterative averaging algorithm to highlight the effect of the good updates sent by the majority of the clients. At the end of this iterative algorithm, a trustworthiness is assigned to each client that can be used to put the client in one of the good or bad sets. Finally, the server considers the probability of providing good updates to the model by each client. These probabilities can be computed by considering a mixed-strategy game between the central server and each client that exists in the good client set. The valid actions of the clients are to

*Corresponding Author Institutional Email: e.tahanian@shahroodut.ac.ir (E. Tahanian)

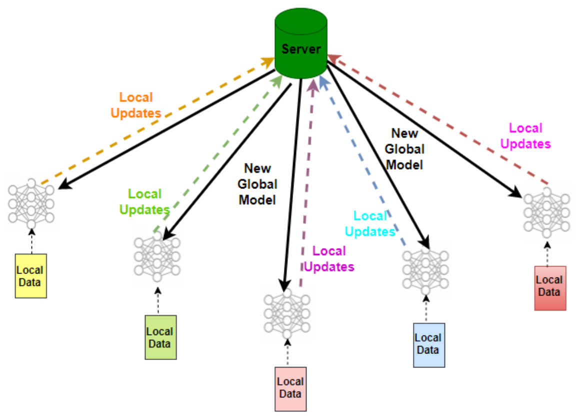


Figure 1. Architecture for a federated learning system with five benign clients that communicate with a central server periodically to learn a global model

send good or bad updates while the server can accept or ignore these updates. By employing the Nash Equilibrium property [11], the server determines the clients' probability to provide good updates to the model. We summarize our main contributions as follows:

- We propose an iterative averaging algorithm for the server to obtain the trustworthiness of each update and a robust estimate of the final model, simultaneously.
- We model the problem using a mixed-strategy game between the central server and each client.
- We apply the Nash Equilibrium property to compute the probability of providing good updates from each client.

We provide a thorough empirical evaluation of the effectiveness and efficiency of our proposed robust federated learning method. The results show that our method provides both higher accuracy and faster convergence than the existing methods. Specifically, our algorithm converges after a maximum of 30 iterations and can detect 100% of the bad clients for all the investigated scenarios. Furthermore, the accuracy of the proposed algorithm is at least 15.8% and 2.3% better than the state of the art for flipping and noisy scenarios, respectively.

The rest of this paper is organized as follows. The related work is discussed in Section 2. Section 3 describes the proposed federated learning as well as the aggregation algorithm. The experimental results are reported and discussed in Section 4, followed by the conclusion in Section 5.

2. RELATED WORK

Federated learning for the first time was implemented by Google to predict users' text input within a large number of mobile devices without sending private data [2, 12]. One of the main elements of Federated learning is the aggregation operator. Several federated aggregation operators have been presented in literature. FedAvg [13] updates the global model by averaging the parameters of the local models. This algorithm was used for recognizing out-of-vocabulary words [14] and improving the mobile keyboard prediction [15]. As a modification to FedAvg, Federated Stochastic Variance Reduced Gradient (FSVRG) [2] was presented to work with sparse data. In contrast to FedAvg and FSVRG, CO-OP [16] has been presented for asynchronous model updates. It merges any received client model with the global model. According to the difference in the age of the models, the local and global models merging is carried out using a weighting scheme, instead of directly averaging the models.

Due to the distributed scheme of federated learning, it is highly vulnerable to attacks against the learning models. As previously mentioned, sending bad updates by faulty or malicious clients is the most serious concern for federated learning. Consequently, the standard federated learning algorithms such as FedAvg [1, 13] are vulnerable to both model poisoning and data poisoning. To overcome this problem, researchers have proposed different robust averaging algorithms [8, 17, 18].

Some other researches have focused on vulnerability in federated learning known as a backdoor attack [3, 19]. In this kind of attack, the adversary tries to reduce the performance of the model on targeted tasks while maintaining good performance on the main task [19].

Authors in [8] proposed a byzantine-robust aggregation algorithm, referred to as KRUM, which is based on the similarity of the client updates. To solve the slow convergence problem of the KRUM, a faster algorithm known as MKRUM was introduced. Yin et al. [17] proposed a coordinate-wise median (COMED), a byzantine-robust statistical learning algorithm with a focus on statistical optimality.

Although the aforementioned researches take into account both model poisoning and data poisoning for a number of simple attack scenarios, the proposed methods can be computationally expensive when the number of clients is large. In contrast, the computational complexity of our method (GFA) can be shown to be considerably less. Moreover, the previous algorithms do not consider the number of data points used by each client to train the local models, while the GFA computes the averaging based on the dataset size of each client.

Moreover, our proposed method flexibly chooses the good clients based on the received information at each iteration. Therefore, it is more efficient in comparison with the works that use a pre-specified number of clients' information to update the global model, such as the work proposed by Xie et al. [18]. Especially, unlike the GFA, when all of the clients or the majority of them are good, the algorithm of Xie et al. [18] considers the pre-specified number of the clients as bad, which will affect the performance of the learning process.

Also, some works have focused on applying game theory to the federated learning system [20-22]. The authors in [20] proposed a contract theory-based incentive mechanism to motivate data owners that have high-quality local training data to join the learning processes. In [21], the authors presented the Stackelberg game model to analyze the transmission strategy and training data pricing strategy of the self-organized mobile device as well as the learning service subscription of the model owner in the cooperative federated learning system. Zou et al. [22] adopted an evolutionary game theory to model dynamic strategies of the mobile devices with bounded rationality in the federated learning system. Although all of these works used the game theory, none of them focused on the averaging algorithm in federated learning. In this paper, we use the game theory to propose a robust federated averaging algorithm to detect and discard bad updates provided by the clients.

3. FEDERATED LEARNING AND AGGREGATION ALGORITHM

In this section, we formulate the federated learning paradigm and propose our robust aggregation algorithm based on a game-theoretical approach.

3. 1. Federated Learning Model The main idea of federated learning is to perform the training of a deep neural network (DNN) using some clients by aggregating local models into a joint global model, as can be seen in Figure 1. Since the local training data never shared by the clients, the federated model can train on completely private data.

We suppose there are N clients where the ith client's dataset has n_i data points. At round t, the server randomly chooses a subset of clients (M_t) and sends them the latest global model (ω_t). Each client, for example, $i \in M_t$ that receives the model, updates it by training on its dataset, and derives a new local model (ω_{t+1}^i). Afterward, the chosen clients send back the new model to the central server. In this step, the server averages the received local models to achieve an updated global model according to Equation (1).

$$\omega_{t+1} = \sum_{i \subset M_t} \frac{n_i}{n} \omega_{t+1}^i, \qquad (1)$$

where n is obtained as follows:

$$n = \sum_{i \subset M_t} n_i \ . \qquad (2)$$

However, by using such an aggregation method, only a bad (malicious or faulty) client can lead to the wrong solution or prevent the DNN to be converged [4]. To solve this problem, in the next sections, we propose a novel aggregation algorithm based on the game theory in which the probability of providing good model updates by each client is considered. Table 1 contains a summary of the notations used in this paper.

3. 2. Adversary Model In this paper, we make the following assumptions regarding the adversary: (1) We assume that only less than half of the clients can be compromised; (2) the attacker controls the local training data of any compromised client; (3) it does not control the aggregation algorithm used by the server to average clients' updates and generate the new global model; (4) the attacker can not control the updates sent by the good clients and, (5) the data is distributed among the clients in an i.i.d fashion.

The adversary's goal in our work is to prevent the global model to converge. So, we propose a novel aggregation algorithm to overcome the convergence problem of the previous algorithms while the attacker follows the above-mentioned assumptions. In the rest of this paper, we use *bad* clients whenever we mean malicious or faulty clients.

3. 3. Aggregation Algorithm To implement robust federated learning, we should initially estimate the bad clients. To reach this goal, we propose a novel averaging algorithm as well as a game model in this Section.

3. 3. 1. Averaging Algorithm At each round, when the central server receives the local updates of the clients; it uses an adaptive averaging method. In this paper, we proposed to highlight the effect of the good updates sent by the majority of the clients [23, 24]. The proposed aggregation algorithm is detailed in Algorithm 1.

When the server receives the local updates of the clients, the server iteratively computes a weighed average as follows:

TABLE 1. The notation used in this paper.

Symbol	*Description*
N	The number of clients
n_i	The size of the ith client dataset
M_t	The number of clients that send updates at round t
m	The size of M_t
ω_{t+1}^i	The local model provided by the ith client at round t
ω_{t+1}	The global model sent to the clients
G_t	The set of clients considered good by the server
N_G	The size of G_t
AA_k	The weighed average of the received updates in kth iteration
y_i	The distance between ω_{t+1}^i and AA_{k-1}

$$AA_k = \frac{\sum_{i\in M_t} e^{-\alpha y_i} \omega_{t+1}^i}{\sum_{i\in M_t} e^{-\alpha y_i}}, \ \ k > 1 \tag{3}$$

where AA_k is the weighted average of the received updates in *k*th iteration and y_i is the distance between ω_{t+1}^i and the weighted average at the previous iteration (AA_{k-1}), obtained as follows:

$$y_i = |\boldsymbol{\omega_{t+1}^i} - \mathbf{AA_{k-1}}|$$

It is to be noted that we compute the distance of each client's model with the current estimate of the aggregation values to estimate the trustworthiness of the clients. In other words, there is an inverse relationship between the trustworthiness of a client and the distance of its local model with the aggregated model obtained at each iteration of our adaptive aggregation algorithm. We employed an exponential decaying function to model such an inverse relationship as it shows promising results in our experiments. One can choose any other decaying function to compute the trustworthiness from the distance value of the models. In Equation (3), α is a constant parameter that controls the amount of *trustworthiness* ($e^{-\alpha y_i}$) considered for each update.

It should be noted that the iterative procedure starts with giving equal credibility to all clients, i.e., with an initial value of 1. Consequently, the initial aggregated model at the first iteration of the algorithm is calculated using a simple averaging as follows:

$$AA_1 = \frac{\sum_{i\in M_t} \omega_{t+1}^i}{m} \tag{4}$$

Algorithm 1 Robust Aggregation Algorithm

```
Require: $M_t$, $n_i$, $\omega_{t+1}^i$, K, $\alpha$
$G_t \leftarrow \{i: i \in M_t\}$
$m$ : the size of $M_t$
for k = 1, 2, 3,..., K do
    if k = 1 then
        $\mathrm{AA_1} = \frac{\sum_{i\in M_t} \omega_{t+1}^i}{m}$
    else
        for $i \in M_t$ do
            $y_i = |\omega_{t+1}^i - \mathrm{AA_{k-1}}|$
        end for
        Compute $\mathrm{AA_k}$ according to Eq. (3)
    end if
end for
Apply k-means to set of $\{e^{-\alpha y_1}, \ldots, e^{-\alpha y_m}\}$ and form two sets, $G_t$ and $B_t$
Compute $p_t{}^i$ using Game model
$\omega_{t+1} \leftarrow \frac{\sum_{i\in G_t} p_t{}^i \, n_i \, \omega_{t+1}^i}{\sum_{i\in G_t} p_t{}^i \, n_i}$
return $\omega_{t+1}$ , $\mathrm{G_t}$
```

where m is the size of M_t. By considering a stopping criterion (K), according to the variation of the *trustworthiness* of the clients, the iterative algorithm will be stopped. The main idea for our proposed aggregation algorithm is inspired by the iterative filtering algorithm proposed by [in the literature [25]. In this reference, a class of voting systems based on iterative filtering has been presented. In other words, in the first round, the simple average of the votes is calculated. Then, proportional to the inverse of the distance from the calculated average, an averaging weight is considered for each vote to compute a next round average. This process continues until the majority votes are close enough to each other and therefore the minority votes are filtered. The proof of convergence has been provided in the literature for two different discriminant functions [25]. We believe that a similar method can be used to prove the convergence of the iterative process in Algorithm 1. We leave this proof as an interesting idea for our future work.

Now, we expect the *trustworthiness* of the good clients are similar enough and spaced far enough from the bad clients [23]. So, the server can apply a one-dimensional *k-means* algorithm to put the clients in two separate clusters, *bad* clients (B_t) and *good* clients (G_t) with more than half of the clients. Thus, the server can average the updates by good clients regarding to the fraction of training data points provided by each client (Equation (5)).

However, due to the similarity of the updates provided by the clients, some of them may be misdiagnosed. Therefore, the server should consider a probability ($p_t{}^i$) in the model aggregation algorithm for each client in G_t, according to Equation (5).

$$\omega_{t+1} = \frac{\sum_{i\in G_t} p_t{}^i \ n_i \ \omega_{t+1}^i}{\sum_{i\in G_t} p_t{}^i \ n_i} \tag{5}$$

The clients' probability to provide good updates to the model can be computed by considering a game between the central server and each client in G_t, as described in the next section.

3. 3. 2. Game Model

We suppose the server plays the game independently with each client. The valid actions of the clients are to send good or bad updates while the server can accept or ignore these updates as its valid actions. It should be noted that in our model the players (server and clients) follow a mixed-strategy in which the actions are randomly selected over the set of available actions according to some probability distribution. Afterward, the Nash Equilibrium property is applied to determine the probability of server and client actions. The Nash Equilibrium property is a popular Game Theory concept that describes strategies from which reasonable decision makers should not be deviated to maximize their utility.

The server uses the calculated probability of accepting the updates in our aggregation method as can be seen in Equation (5).

Figure 2 illustrates the normal form of the game including the valid actions and corresponding payoffs for the *i*th player.

$$A_i = \frac{n_i}{N_G} \left| \omega^i{}_{t+1} - \omega_t \right| \tag{6}$$

$$B_i = \frac{n_i}{N_G} \left| \omega^i{}_{t+1} \right| \tag{7}$$

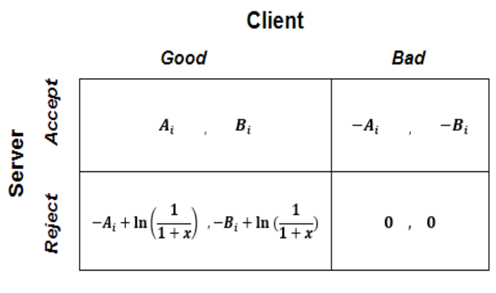
and,

$$x = \begin{cases} 0 & , if\ m^i{}_G \geq m^i{}_B \\ m^i{}_B - m^i{}_G & , if\ m^i{}_G < m^i{}_B \end{cases} \tag{8}$$

where $m^i{}_G$ and $m^i{}_B$ are the number of times that the ith client is in the good and bad set, respectively. Moreover, N_G is the total number of data points in G_t.

As one can see in Figure 2, each client can send good or bad updates while the server can accept or ignore these updates as valid actions. When the *i*th client sends good updates, if the server accepts these updates, it earns a payoff as large as A_i, which indicates the *i*th client contribution to the correction of the previous global model. On the other hand, if the server rejects these good updates, it losses this amount of payoff. Furthermore, we consider the client payoff equal to B_i, that is the client contribution in the global model at the next round, if the server accepts the good updates. It is worth noting that when a client is misdiagnosed several times, the server ignores it forever. Clearly, we should consider the effect of this wrong action of the server in the client payoff when it sends a good model. Therefore, in this situation, we add the term $ln \frac{1}{1+x}$ to the client payoff. In this term, x is related to the number of good model rejection by the server as explained in Equation (8).

Now, if the client is faulty or malicious and sends bad updates, the payoffs of the players can be determined as illustrated in Figure 2. When the server accepts the bad updates, both of them experience negative payoff. On the other hand, if the server rejects these updates, neither side will earn any payoffs.



Server \ Client	Good	Bad
Accept	A_i , B_i	$-A_i$, $-B_i$
Reject	$-A_i + \ln\left(\frac{1}{1+x}\right)$, $-B_i + \ln\left(\frac{1}{1+x}\right)$	0 , 0

Figure 2. The normal form of the game played between the server and the *i*th client

3. 3. 3. Nash Equilibrium Since in our model, the players follow a mixed-strategy, we can determine the probability of the server and clients actions by applying the Nash Equilibrium property. In other words, there is at least one Nash Equilibrium when we consider mixed-strategy [26]. A mixed strategy Nash Equilibrium involves at least one player playing a randomized strategy and no player is able to increase his or her expected payoff by playing an alternate strategy. At the mixed Nash Equilibrium, both players should be indifferent between their two strategies. Therefore, if the server is using a mixed strategy, it must be indifferent between accepting and rejecting the updates. So, we can write:

$$q_t{}^i A_i - A_i\left(1 - q_t{}^i\right) = q_t{}^i\left(-A_i + ln\left(\frac{1}{1+x}\right)\right) \tag{9}$$

and therefore we can calculate the probability of sending good updates by the *i*th client ($\boldsymbol{q_t{}^i}$), when it plays Nash Equilibrium, as follows:

$$\boldsymbol{q_t{}^i} = \frac{A_i}{3A_i - ln\frac{1}{1+x}} \tag{10}$$

On the other hand, when the server plays Nash Equilibrium, the *i*th client should be indifferent between its two actions. So, in a similar way, the probability of accepting the *i*th client updates ($\boldsymbol{p_t{}^i}$) by the server, when it plays Nash Equilibrium, can be derived as follows:

$$\boldsymbol{p_t{}^i} = \frac{B_i - ln\frac{1}{1+x}}{3B_i - ln\frac{1}{1+x}} \tag{11}$$

Finally, the server uses $\boldsymbol{p_t{}^i}$ in Equation (5) when it aggregates the received updates.

It is worth noting that this game has also two pure strategy Nash Equilibriums, i.e. (Good, Accept) and (Bad, Reject), which are quite obvious.

4. EXPERIMENTS

In this section, we report on a detailed experimental study that examines robustness and efficiency of our robust federated learning method. The objective of our experiments is to evaluate the robustness and efficiency of our approach for estimating the global model based on the model received from the clients in the presence of faults.

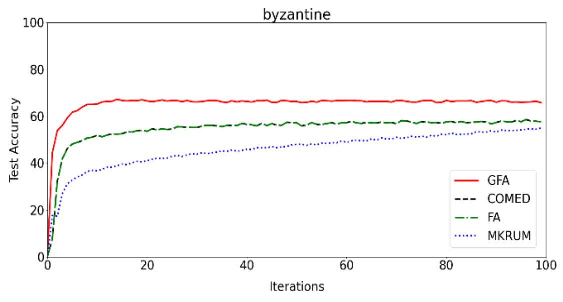
4. 1. Experimental Environment We conducted experiments on three datasets: CIFAR-10 [27], MNIST [28] and SPAMBASE. CIFAR-10 consists of 60000 32×32 color images in 10 classes while MNIST has 70000 28×28 handwritten digits in 10 classes. For

CIFAR we used VGG-11 which is a familiar convolutional neural network [29] and for MNIST we trained DNNs with $784 \times 512 \times 256 \times 10$ with learning rate 0.1 and Dropout probability 0.5. Also, the hidden layers and output layer activation functions are Leaky ReLU and Softmax, respectively. Furthermore, for SPAMBASE we trained DNNs with $54 \times 100 \times 50 \times 1$ with learning rate 0.05 and Dropout probability 0.5. Also, the hidden layers and output layer activation functions are Leaky ReLU and Sigmoid, respectively. We considered gradient descent as the optimization method where the batch size and epoch number are 200 and 10, respectively. For all the simulations, we set $\alpha = 5$ and we consider the number of clients 10 and 100. Moreover, we assume that all the clients are selected to send updates for the server, i. e. $M_t = N$.

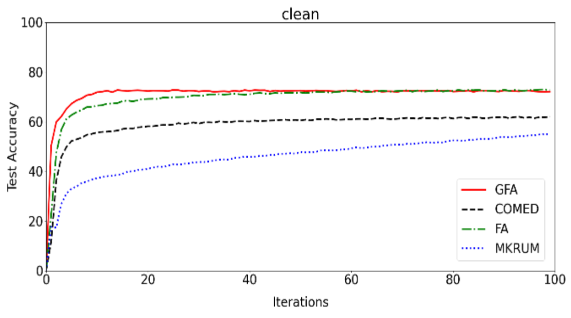
In this paper, we consider four different scenarios, namely, *clean*, *byzantine*, *flipping*, and *noisy* to evaluate our proposed method. For the clean scenario, all of the clients send good updates to the server. In the *byzantine* case, some of the clients are bad and send updates that are significantly different from the updates sent by the good clients. In this case, we consider a Gaussian distribution with mean zero and isotropic covariance matrix with standard deviation 20. For the third scenario that is *flipping*, we set all the labels of data points, used by the selected bad clients to train the model, to zero. Finally, in *noisy* case, we add uniform noise to all the pixels of the noisy clients.

4. 2. Evaluation Results In this section, we compare our proposed method (GFA) with the previous works, namely, Multi-KRUM (MKRUM) [8], Federated Averaging (FA) [1], and COMED [12]. Figures 3, 5, and 7 illustrate the test accuracy of all the algorithms for 10 clients as a function of the number of iterations on the CIFAR-10, MNIST, and SPAMBASE datasets, respectively. In addition, Figures 4, 6, and 8 illustrate the test accuracy of all the algorithms for 100 clients as a function of the number of iterations on the CIFAR-10, MNIST, and SPAMBASE datasets, respectively. According to these figures, we can analyze the convergence of these algorithms. As can be observed, the proposed algorithm converges for all the four scenarios over both datasets while other algorithms do not converge in at least one of the eight cases. For example, the FA and COMED algorithms do not converge for flipping scenarios on the CIFAR-10 while MKRUM and COMED have the same convergence problem on the MNIST. In addition, in the worst case, our algorithm converges after a maximum of 30 iterations.

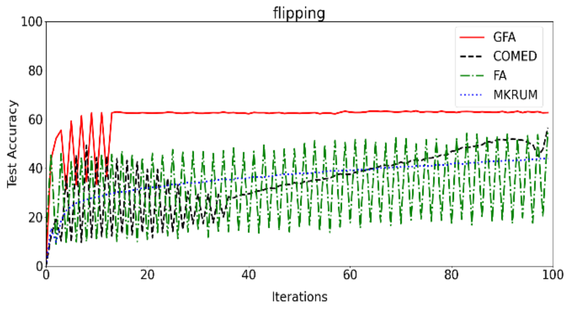
In addition, Figure 9 illustrates test accuracy as a function of the number of iterations for different values of α on two datasets, namely, CIFAR-10 and SPAMBASE. Accordingly, we set $\alpha = 5$ because it leads to the best accuracy.



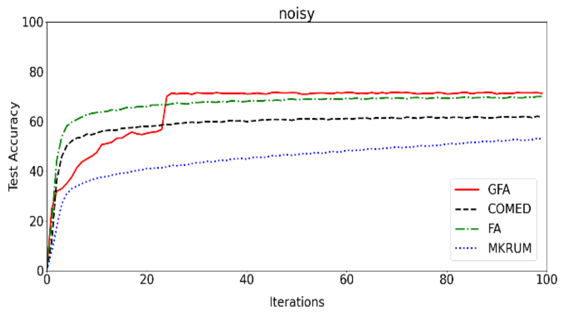
(a)



(b)



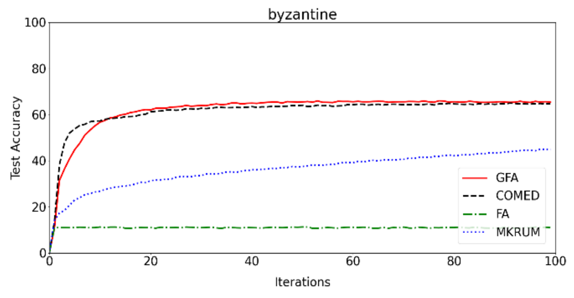
(c)



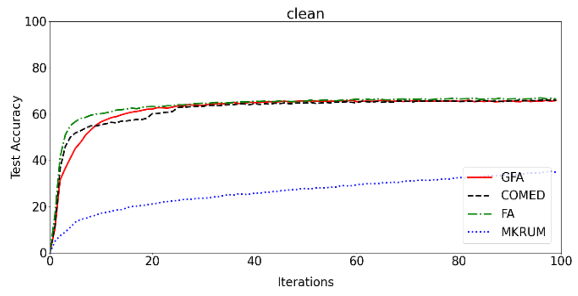
(d)

Figure 3. Test accuracy (%) as a function of the number of iterations for 10 clients and for different algorithms on CIFAR-10 for, a) clean (all benign clients), b) byzantine, c) flipping, and d) noisy clients

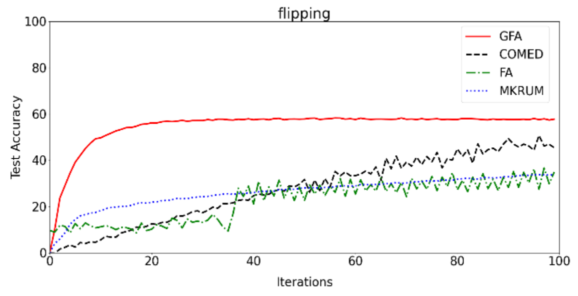
Even in the cases where the other methods converge, the proposed GFA algorithm is ultimately more accurate. Tables 2, 3, and 4 compare the ultimate test accuracy of the different algorithms over CIFAR-10, MNIST, and SPAMBASE, respectively. Accordingly, for CIFAR-10,



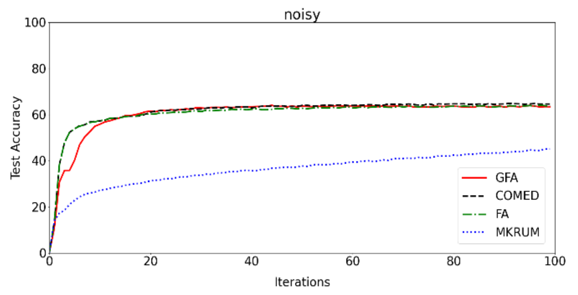
(a)



(b)



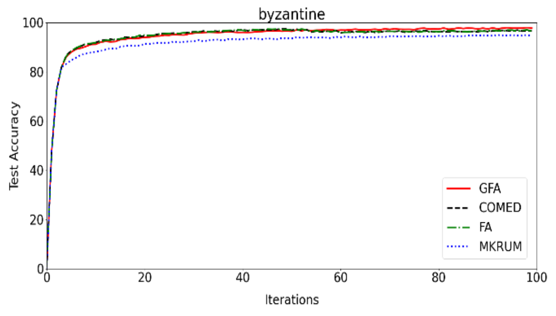
(c)



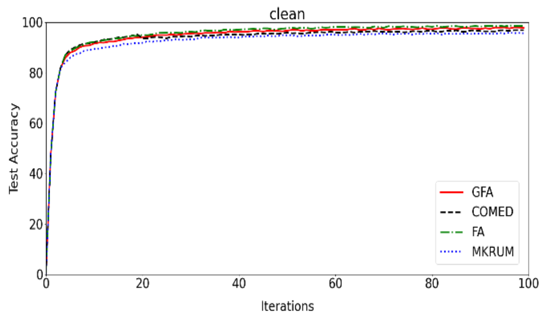
(d)

Figure 4. Test accuracy (%) as a function of the number of iterations for 100 clients and for different algorithms on CIFAR-10 for, a) clean (all benign clients), b) byzantine, c) flipping, and d) noisy clients

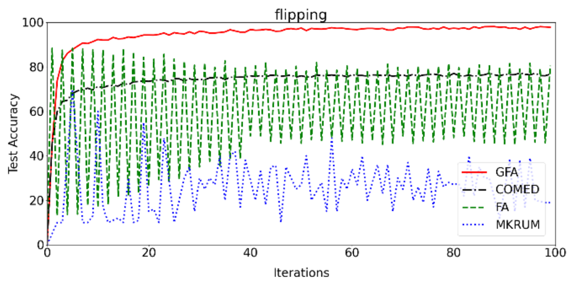
the test accuracy of the proposed algorithm is at least 14%, 15.8%, and 2.3% better than the others for byzantine, flipping, and noisy scenarios, respectively. For this dataset and in the case of clean scenario, the results show that the accuracy of the GFA is only 0.27% less than the standard FA algorithm. On the other hand, for MNIST, the simulations indicate a similar situation where the accuracy of our method is at least 0.4%, 27%, and 2.6% higher than the byzantine, flipping, and noisy scenarios, respectively. Again, the accuracy of the FA algorithm is a little (0.8%) better than the GFA algorithm in the case of clean.



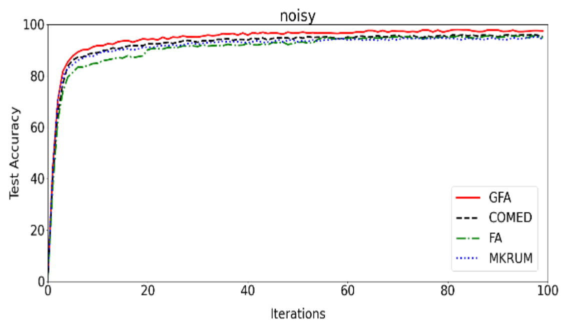
(a)



(b)

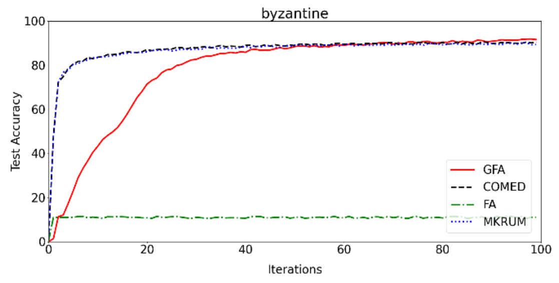


(c)

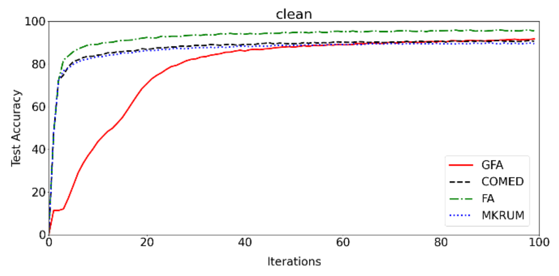


(d)

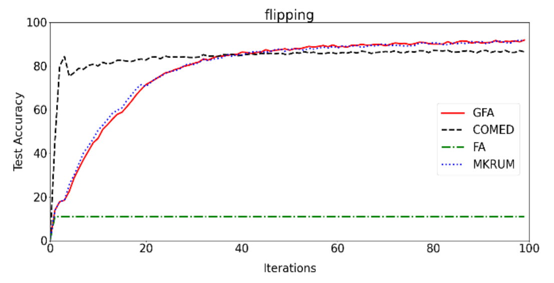
Figure 5. Test accuracy (%) as a function of the number of iterations for 10 clients and for different algorithms on MNIST for, a) clean (all benign clients), b) byzantine, c) flipping, and d) noisy clients



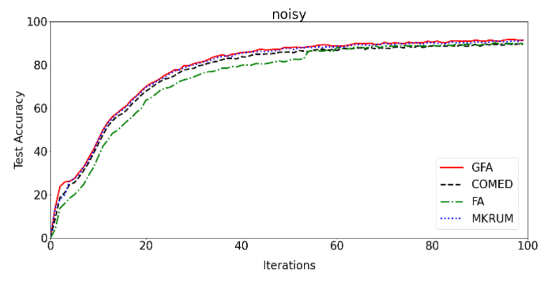
(a)



(b)

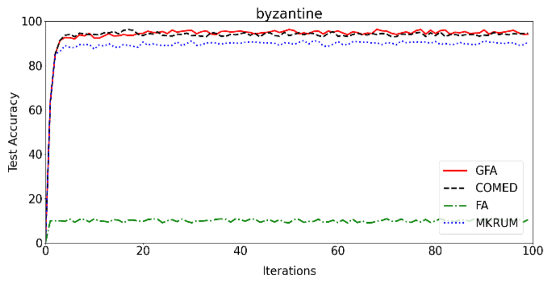


(c)

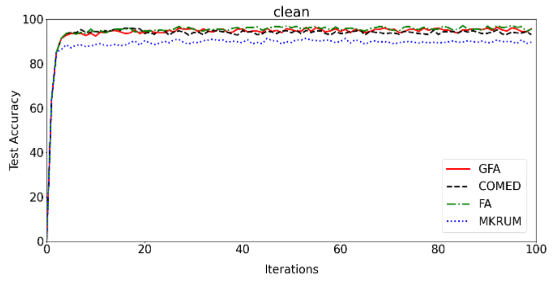


(d)

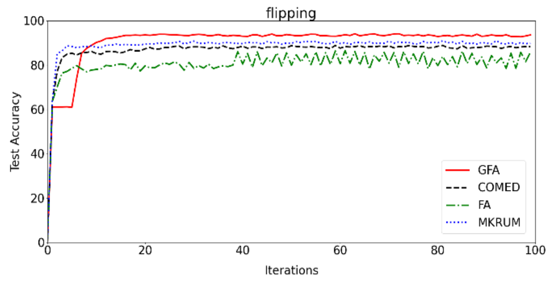
Figure 6. Test accuracy (%) as a function of the number of iterations for 100 clients and for different algorithms on MNIST for, a) clean (all benign clients), b) byzantine, c) flipping, and d) noisy clients



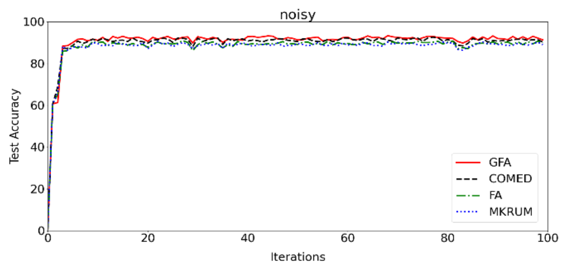
(a)



(b)

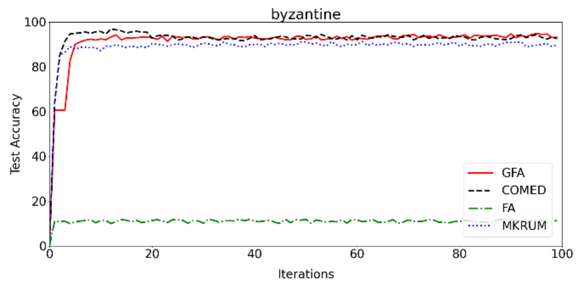


(c)

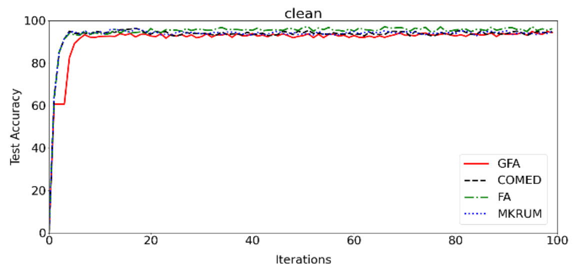


(d)

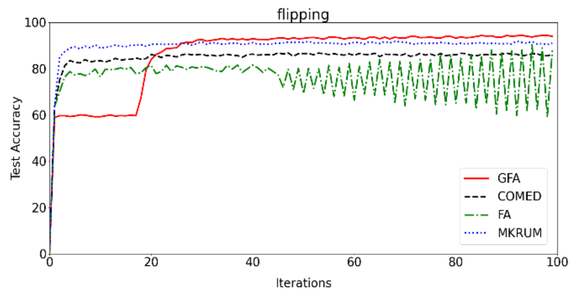
Figure 7. Test accuracy (%) as a function of the number of iterations for 10 clients and for different algorithms on SPAMBASE for, a) clean (all benign clients), b) byzantine, c) flipping, and d) noisy clients



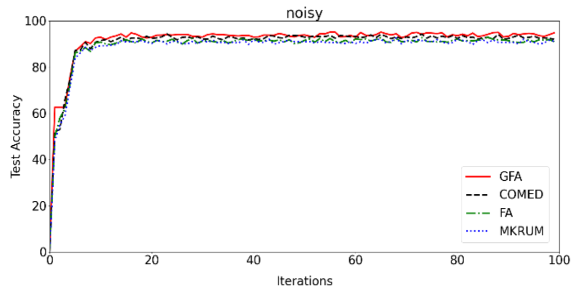
(a)



(b)

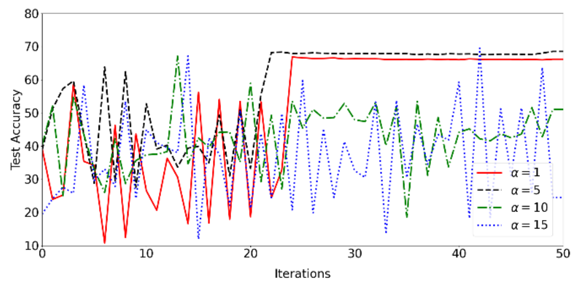


(c)

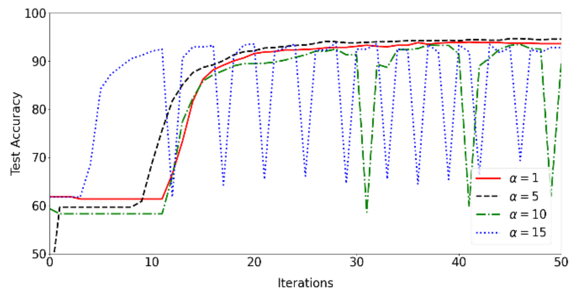


(d)

Figure 8. Test accuracy (%) as a function of the number of iterations for 100 clients and for different algorithms on SPAMBASE for, a) clean (all benign clients), b) byzantine, c) flipping, and d) noisy clients



(a)



(b)

Figure 9. Test accuracy (%) as a function of the number of iterations for different values of α on a) CIFAR-10 and b) SPAMBASE datasets

Furthermore, we investigated the detection rate for bad clients in Table 5. According to this table, for all of the scenarios containing malicious clients, the proposed algorithm in this paper can detect 100% of the bad clients for both datasets.

TABLE 2. The test accuracy of different algorithms for the CIFAR-10 dataset

Algorithm	Clean	Byzantine	Flipping	Noisy
GFA	72.83	67.06	63.48	71.69
FA	73.03	52.49	54.78	70.02
COMED	62.06	58.63	56.84	62.09
MKRUM	54.76	55.32	43.93	53.13

TABLE 3. The test accuracy of different algorithms for the MNIST dataset

Algorithm	Clean	Byzantine	Flipping	Noisy
GFA	98.09	98.01	98.23	98.09
FA	98.89	10.29	77.17	95.57
COMED	97.02	97.43	91.13	95.96
MKRUM	96.01	95.07	70.00	95.22

TABLE 4. The test accuracy of different algorithms for SPAMBASE dataset

Algorithm	Clean	Byzantine	Flipping	Noisy
GFA	96.42	96.65	94.18	93.50
FA	97.19	10.91	86.98	91.13
COMED	95.74	96.24	88.97	92.48
MKRUM	91.64	91.53	90.88	90.58

TABLE 5. The detection rate of GFA algorithm for bad clients on MNIST, CIFAR-10, and SPAMBASE datasets

Dataset	Byzantine	Flipping	Noisy
MNIST	100%	100%	100%
CIFAR10	100%	100%	100%
SPAMBASE	100%	100%	100%

5. CONCLUSION

In this paper, we introduced a game-based robust federated averaging algorithm to detect and discard bad updates provided by the clients. The proposed method uses an adaptive averaging method, in an iteration manner, to highlight the effect of the good updates sent by the majority of the clients. At the end of this iterative algorithm, a trustworthiness is assigned to each client that can be used to put the client in one of the good or bad sets. Finally, the server considers the probability of providing good updates by the clients to the model. These probabilities can be computed by considering a mixed-strategy game between the central server and each client that exists in the good client set. The valid actions of the clients are to send good or bad updates while the server can accept or ignore these updates. By employing the Nash Equilibrium property, the server determines the clients' probability to provide good updates to the model.

In experiments, we considered four scenarios, *clean*, *byzantine*, *flipping,* and *noisy* that were evaluated on MNIST, SIFAR-10, and SPAMBASE datasets for 10 and 100 clients. For all of the scenarios and both datasets, our algorithm converges after a maximum of 30 iterations. It should be noted that in all cases, 100% of the bad clients can be detected for both datasets. In addition, the test accuracy of the proposed algorithm is at least 15.8% and 2.3% better than the others for *flipping* and *noisy* scenarios, respectively.

In future work, we plan to use the Game Theory to detect the backdoor attack where a malicious client can use model replacement to introduce backdoor functionality into the global model.

6. REFERENCES

1. McMahan, B., Moore, E., Ramage, D., Hampson, S. and y Arcas, B.A., "Communication-efficient learning of deep networks from decentralized data", in Artificial Intelligence and Statistics, PMLR. 1273-1282.
2. Konečný, J., McMahan, H.B., Ramage, D. and Richtárik, P., "Federated optimization: Distributed machine learning for on-device intelligence", arXiv preprint arXiv:1610.02527, Vol., No., (2016).
3. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. and Shmatikov, V., "How to backdoor federated learning", in International Conference on Artificial Intelligence and Statistics, PMLR., 2938-2948.
4. Yang, Q., Liu, Y., Chen, T. and Tong, Y., "Federated machine learning: Concept and applications", ***ACM Transactions on Intelligent Systems and Technology (TIST)***, Vol. 10, No. 2, (2019), 1-19. Doi: 10.1145/3298981
5. Bhagoji, A.N., Chakraborty, S., Mittal, P. and Calo, S., "Analyzing federated learning through an adversarial lens", in International Conference on Machine Learning, PMLR. 634-643.
6. Li, T., Sahu, A.K., Talwalkar, A. and Smith, V., "Federated learning: Challenges, methods, and future directions", ***IEEE Signal Processing Magazine***, Vol. 37, No. 3, (2020), 50-60. Doi: 10.1109/msp.2020.2975749
7. Sattler, F., Wiedemann, S., Müller, K.-R. and Samek, W., "Robust and communication-efficient federated learning from non-iid data", ***IEEE Transactions on Neural Networks and Learning Systems***, Vol. 31, No. 9, (2019), 3400-3413. Doi: 10.1109/tnnls.2019.2944481
8. Blanchard, P., El Mhamdi, E.M., Guerraoui, R. and Stainer, J., "Machine learning with adversaries: Byzantine tolerant gradient descent", in Proceedings of the 31st International Conference on Neural Information Processing Systems. Vol., No., (Year), 118-128.
9. Damaskinos, G., El Mhamdi, E.M., Guerraoui, R., Guirguis, A.H.A. and Rouault, S.L.A., "Aggregathor: Byzantine machine learning via robust gradient aggregation", in The Conference on Systems and Machine Learning (SysML), 2019. Vol., No. CONF, (Year).
10. Mhamdi, E.M.E., Guerraoui, R. and Rouault, S., "The hidden vulnerability of distributed learning in byzantium", arXiv preprint arXiv:1802.07927, (2018).
11. Nash, J., "Non-cooperative games", Annals of Mathematics, (1951), 286-295.
12. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T. and Bacon, D., "Federated learning: Strategies for improving communication efficiency", arXiv preprint arXiv:1610.05492, (2016).
13. McMahan, H.B., Moore, E., Ramage, D. and y Arcas, B.A., "Federated learning of deep networks using model averaging", arXiv preprint arXiv:1602.05629, (2016).
14. Chen, M., Mathews, R., Ouyang, T. and Beaufays, F., "Federated learning of out-of-vocabulary words", arXiv preprint arXiv:1903.10635, (2019).
15. Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C. and Ramage, D., "Federated learning for mobile keyboard prediction", arXiv preprint arXiv:1811.03604, (2018).
16. Wang, Y., "Co-op: Cooperative machine learning from mobile devices", , (2017).
17. Yin, D., Chen, Y., Kannan, R. and Bartlett, P., "Byzantine-robust distributed learning: Towards optimal statistical rates", in International Conference on Machine Learning, PMLR., 5650-5659.
18. Xie, C., Koyejo, S. and Gupta, I., "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance", in International Conference on Machine Learning, PMLR., 6893-6901.
19. Sun, Z., Kairouz, P., Suresh, A.T. and McMahan, H.B., "Can you really backdoor federated learning?", arXiv preprint arXiv:1911.07963, (2019).
20. Kang, J., Xiong, Z., Niyato, D., Yu, H., Liang, Y.-C. and Kim, D.I., "Incentive design for efficient federated learning in mobile networks: A contract theory approach", in 2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS), IEEE., 1-5. Doi: 10.1109/vts-apwcs.2019.8851649
21. Feng, S., Niyato, D., Wang, P., Kim, D.I. and Liang, Y.-C., "Joint service pricing and cooperative relay communication for federated learning", in 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE., 815-820. Doi: 10.1109/ithings/greencom/cpscom/smartdata.2019.00148
22. Zou, Y., Feng, S., Niyato, D., Jiao, Y., Gong, S. and Cheng, W., "Mobile device training strategies in federated learning: An evolutionary game approach", in 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE., 874-879. Doi: 10.1109/ithings/greencom/cpscom/smartdata.2019.00157
23. Limam, N. and Boutaba, R., "Assessing software service quality and trustworthiness at selection time", ***IEEE Transactions on Software Engineering***, Vol. 36, No. 4, (2010), 559-574. Doi: 10.1109/tse.2010.2
24. Rehman, A.U., Jiang, A., Rehman, A. and Paul, A., "Weighted based trustworthiness ranking in social internet of things by using soft set theory", in 2019 IEEE 5th International Conference on Computer and Communications (ICCC), IEEE., 1644-1648. Doi: 10.1109/iccc47050.2019.9064242

25. De Kerchove, C. and Van Dooren, P., "Iterative filtering in reputation systems", ***SIAM Journal on Matrix Analysis and Applications***, Vol. 31, No. 4, (2010), 1812-1834. Doi: 10.1137/090748196

26. Myerson, R.B., "Game theory, Harvard university press, (2013). Doi: 10.1002/9781118547168

27. Krizhevsky, A. and Hinton, G., "Learning multiple layers of features from tiny images", Vol., No., (2009).

28. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., "Gradient-based learning applied to document recognition", ***Proceedings of the IEEE***, Vol. 86, No. 11, (1998), 2278-2324. Doi: 10.1109/5.726791

29. Liu, S. and Deng, W., "Very deep convolutional neural network based image classification using small training sample size", in 2015 3rd IAPR Asian conference on pattern recognition (ACPR), IEEE., 730-734. Doi: 10.1109/acpr.2015.7486599

Persian Abstract

چکیده

با استفاده ازیادگیری فدراسیونی قابلیت تجمیع مدلهای آموزش یافته بر روی تعداد زیادی از کلاینتها از طریق ارسال این مدلها به سرور مرکزی فراهم می‌گردد. این درحالیست که همچنان حریم خصوصی کلاینتها حفظ خواهد شد، زیرا تنها مدلها به سرور ارسال می‌شود. روشهای یادگیری فدراسیونی به شدت درمعرض حملات قرار دارند. در این مقاله ما یک الگوریتم میانگین‌گیری مقاوم براساس نظریه بازیها ارائه میکنیم. ما فرآیند میانگین‌گیری را با یک بازی با سناریوی میکس که در آن هر کلاینت و سرور به عنوان بازیکن می باشند، مدل میکنیم. اعمال مجاز کلاینتها در بازی شامل ارسال بروزرسانی‌های خوب و بد و نیز اعمال سرور شامل پذیرش یا رد این بروزرسانی‌ها می باشند. نتایج آزمایشها نشان میدهد به کار بردن روش میانگین گیری مبتنی بر نظریه بازیها بسیار مقاومتر از روشهای مشابه در مقابل کلاینتها مخرب می باشد. مطابق این نتایج، روش پیشنهادی حداکثر بعداز 30 تکرار همگرا میشود و قادر است 100 درصد از کلاینتهای مخرب را تشخیص دهد. همچنین روش ما از نظر دقت به ترتیب حداکثر 15.8 درصد و 2.3 درصد بهتر از روشهای پیشین برای دو سناریوی فلیپینگ و نویزی می باشد.