Byzantine-tolerant federated Gaussian process regression for streaming data

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we consider Byzantine-tolerant federated learning for streaming data using Gaussian process regression (GPR). In particular, a cloud and a group of agents aim to collaboratively learn a latent function where some agents are subject to Byzantine attacks. We develop a Byzantine-tolerant federated GPR algorithm, which includes three modules: agent-based local GPR, cloud-based aggregated GPR and agent-based fused GPR. Specifically, the agent-based local GPR sends potentially compromised local predictions to the cloud, and the cloud-based aggregated GPR computes a global model by a Byzantine-tolerant Product-of-Experts aggregation rule. Then the cloud broadcasts the global model to all the agents. Agent-based fused GPR refines the predictions by fusing the model from the cloud-based GPR with that from the agent-based local GPR. We derive the upper bounds on prediction error between the mean from the cloud-based aggregated GPR and the target function provided that Byzantine agents are less than one quarter of all the agents. We also characterize the lower and upper bounds of the predictive variance. Experiments on a synthetic dataset and two real-world datasets are conducted to evaluate the proposed algorithm.

The Internet of Things (IoT) produces large volumes of spatially distributed data. A typical practice is to store data in a centralized entity, e.g., a cloud, and train machine learning models there. However, it poses challenges in communication overhead and computation efficiency. In addition, data storage on the centralized entity raises privacy concerns. Federated learning provides a promising paradigm to address the two challenges [1], [2], [3], [4]. First, only a finite number of model parameters are shared and the size of these parameters is much smaller than that of raw data; second, data samples are kept by agents and local models are trained without sharing raw data. However, these agents are vulnerable to Byzantine attacks, which force them to behave arbitrarily and send any message to the cloud. The cloud cannot access each agent's local private data and training process, and then cannot verify the correctness of each agent's updates. Byzantine attacks incur degradation of learning performance.

Related work. There have been many recent works to secure federated learning against Byzantine attacks [5], [6], [7], [8], [9]. Paper [5] proposes *Krum* as the aggregation rule where the cloud selects a gradient such that the overall distance between this gradient to a fixed number of its closest gradients is minimal. Paper [6] develops an algorithm which is based on integrated stochastic quantization, geometric median based outlier detection and secure model aggregation. Paper [7] analyzes two robust distributed gradient descent algorithms based on median and trimmed mean operations, respectively. Paper [8] proposes a high-dimensional robust mean estimation algorithm at the cloud to combat the adversary. In paper [9], the cloud partitions all the received local gradients into batches and computes the mean of each batch. After that, the cloud computes the geometric median of the batch means. The above set of papers can guarantee that the estimate of the parameter aggregated by the cloud will converge to a neighborhood of the minimizer of the loss function regardless of Byzantine attacks.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

However, the existing works only consider deep neural networks (DNN) as the learning model and are limited to static data and off-line learning.

On-line learning is demanded to process data that arrives sequentially across time [10], [11], [12]. Gaussian process regression (GPR) is a solution candidate to solve the problem [13], [14], [15], [16]. First of all, GPR is a non-parametric statistical learning model. With proper choice of prior covariance function and mild assumptions of the target function, GPR is able to approximate any continuous function [17]. Second, GPR can be implemented recursively, reducing computational complexity and memory [18]. Third, GPR is able to quantify learning uncertainties since it predicts the target function using a posterior probability distribution [19]. However, Byzantine-tolerant GPR has not been studied yet.

Contributions. This paper considers Byzantine-tolerant federated learning for streaming data using GPR where a cloud and a group of agents aim to collaboratively learn a latent function despite some agents are subject to Byzantine attacks. Major contributions are listed as follows:

1) We develop a Byzantine-tolerant federated GPR algorithm, which includes three modules: agent-based local GPR, cloud-based aggregated GPR and agent-based fused GPR. Specifically, the agent-based local GPR sends potentially compromised local predictions to the cloud, and the cloud-based aggregated GPR computes a global model by a Byzantine-tolerant PoE aggregation rule. Then the cloud broadcasts the current global model to all the agents. Agent-based fused GPR returns the predictions by fusing the models from the cloud-based GPR with those from the agent-based local GPR.
2) The predictive mean of the cloud-based aggregated GPR is theoretically guaranteed to be restricted within a neighborhood of the target function provided that Byzantine agents are less than one quarter of all the agents. The prediction error is positively proportional to the number of Byzantine agents and that of the agents with extreme values. Prediction uncertainty is quantified by lower and upper bounds, which are identical to that of the attack-free case.

Experiments on a synthetic dataset and two real-world datasets are conducted to demonstrate that the Byzantine-tolerant GPR algorithm is resilient to Byzantine attacks. To the best knowledge of authors, this is the first paper to study Byzantine-tolerant federated learning over streaming data using GPR.

Notations and notions. Throughout the paper, we use lower-case letters, e.g, a, to denote scalars, bold letters, e.g., $\boldsymbol{a}$, to denote vectors; we use upper-case letters, e.g., A, to denote matrices, calligraphic letters, e.g., $\mathcal{A}$, to denote sets, and bold calligraphic letters, e.g., $\boldsymbol{\mathcal{A}}$, to denote spaces. Denote $I_n \in \mathbb{R}^{n\times n}$ the n-by-n identity matrix. We denote by $\mathbb{R}$ the set of real numbers. The set of non-negative real numbers is denoted by $\mathbb{R}_+$, and the set of positive real numbers is denoted by $\mathbb{R}_{++}$. The cardinality of set $\mathcal{A}$ is denoted by $|\mathcal{A}|$. Define the distance metric $D(\boldsymbol{z},\boldsymbol{z}') \triangleq \|\boldsymbol{z}-\boldsymbol{z}'\|$, and the point to set distance as $D(\boldsymbol{z},\mathcal{Z}) \triangleq \inf_{\boldsymbol{z}'\in\mathcal{Z}} D(\boldsymbol{z},\boldsymbol{z}')$. Define $\text{proj}(\boldsymbol{z},\mathcal{Z}) \triangleq \{\boldsymbol{z}' \in \boldsymbol{\mathcal{Z}} | D(\boldsymbol{z},\boldsymbol{z}') = D(\boldsymbol{z},\mathcal{Z})\}$ the projection set of point $\boldsymbol{z}$ onto set $\mathcal{Z}$. Denote the supremum of a function $\eta : \boldsymbol{\mathcal{Z}} \to \mathbb{R}$ as $\|\eta\|_\infty \triangleq \sup_{\boldsymbol{z}\in\boldsymbol{\mathcal{Z}}} |\eta(\boldsymbol{z})|$. For a real number a, we denote the floor function by $\lfloor a \rfloor$. We denote by $\mathcal{O}(g(t))$ the limiting behavior of some function $f(t)$ if $\lim_{t\to\infty} \frac{f(t)}{g(t)} = a$ for some constant $a > 0$.

We give formal definitions of sub-Gaussian random variables and Gaussian process.

Definition 1 *[20] A random variable X with mean $\mu \triangleq \mathbb{E}[X]$ is sub-Gaussian (denoted by $X \sim subG(\sigma^2)$) if there exists a positive number σ such that*

$$\mathbb{E}\left[e^{\lambda(X-\mu)}\right] \le e^{\frac{\sigma^2\lambda^2}{2}}, \quad \forall\lambda \in \mathbb{R}. \tag{1}$$

Definition 2 *[13] A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

1 Preliminary

In this section, we provide basic knowledge of GPR. Let $\eta : \boldsymbol{\mathcal{Z}} \to \boldsymbol{\mathcal{Y}}$ be the target function, where $\boldsymbol{\mathcal{Z}} \subseteq \mathbb{R}^{n_z}$ and $\boldsymbol{\mathcal{Y}} \subseteq \mathbb{R}$. Given input $\boldsymbol{z}(t) \in \boldsymbol{\mathcal{Z}}$ at time t, the corresponding output is given by

$$y(t) = \eta(\boldsymbol{z}(t)) + e(t), \quad e(t) \sim \mathcal{N}(0, \sigma_e^2), \tag{2}$$

where $e(t)$ is the Gaussian measurement noise and the latent function η is sampled from a Gaussian process prior. Let training data be in the form $\mathcal{D} \triangleq (\mathcal{Z}, \boldsymbol{y})$, where $\mathcal{Z} \triangleq \{\boldsymbol{z}(1), \ldots, \boldsymbol{z}(n_s)\}$ is the set of input data and $\boldsymbol{y} \triangleq [y(1), \ldots, y(n_s)]^{\mathrm{T}}$ is the column vector aggregating the outputs. GPR aims to approximate the latent function η over a set of test data points $\mathcal{Z}_* \subset \boldsymbol{\mathcal{Z}}$ using $\mathcal{D}$.

Define kernel function $k : \mathbb{R}^{n_z} \times \mathbb{R}^{n_z} \to \mathbb{R}$ that is symmetric and positive semi-definite; i.e., $\int k(\boldsymbol{z}, \boldsymbol{z}') f(\boldsymbol{z}) f(\boldsymbol{z}') d\nu(\boldsymbol{z}) d\nu(\boldsymbol{z}') \geq 0$ for all $f \in \mathcal{L}_2(\boldsymbol{\mathcal{Z}}, \nu)$, where ν denotes a measure (see page 80, [13]). Let $\eta(\mathcal{Z})$ return a column vector such that the i-th entry of $\eta(\mathcal{Z})$ equals $\eta(\boldsymbol{z}(i))$. Assume that the latent function η is sampled from a GP prior specified by mean function μ and kernel function k. Then the training outputs $\boldsymbol{y}$ and the test outputs $\boldsymbol{\eta}(\mathcal{Z}_*)$ are jointly Gaussian distributed as:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\eta}(\mathcal{Z}_*) \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \boldsymbol{\mu}(\mathcal{Z}) \\ \boldsymbol{\mu}(\mathcal{Z}_*) \end{bmatrix}, \begin{bmatrix} k(\mathcal{Z}, \mathcal{Z}) + \sigma_e^2 I_{n_s} & k(\mathcal{Z}, \mathcal{Z}_*) \\ k(\mathcal{Z}_*, \mathcal{Z}) & k(\mathcal{Z}_*, \mathcal{Z}_*) \end{bmatrix}),$$

where $\boldsymbol{\mu}(\mathcal{Z})$ and $\boldsymbol{\mu}(\mathcal{Z}_*)$ return a vector consisting of $\mu(\boldsymbol{z}(i))$ and $\mu(\boldsymbol{z}_*(j))$; $k(\mathcal{Z}, \mathcal{Z}_*)$ returns a matrix such that the entry at the ith row and the jth column is $k(\boldsymbol{z}(i), \boldsymbol{z}_*(j))$, and $k(\mathcal{Z}, \mathcal{Z})$, $k(\mathcal{Z}_*, \mathcal{Z})$ and $k(\mathcal{Z}_*, \mathcal{Z}_*)$ are defined in an analogous way.

Utilizing identities of joint Gaussian distribution (see page 200 in [13]), GPR makes predictions of $\boldsymbol{\eta}$ on $\mathcal{Z}_*$ based on dataset $\mathcal{D}$ as $\boldsymbol{\eta}(\mathcal{Z}_*) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathcal{Z}_*|\mathcal{D}}, \Sigma_{\mathcal{Z}_*|\mathcal{D}})$, where

$$\begin{aligned} \boldsymbol{\mu}_{\mathcal{Z}_*|\mathcal{D}} &\triangleq \mu(\mathcal{Z}_*) + k(\mathcal{Z}_*, \mathcal{Z}) \mathring{k}(\mathcal{Z}, \mathcal{Z})^{-1} (\boldsymbol{y} - \mu(\mathcal{Z})), \\ \Sigma_{\mathcal{Z}_*|\mathcal{D}} &\triangleq k(\mathcal{Z}_*, \mathcal{Z}_*) - k(\mathcal{Z}_*, \mathcal{Z}) \mathring{k}(\mathcal{Z}, \mathcal{Z})^{-1} k(\mathcal{Z}, \mathcal{Z}_*), \end{aligned} \tag{3}$$

with $\mathring{k}(\mathcal{Z}, \mathcal{Z}) \triangleq k(\mathcal{Z}, \mathcal{Z}) + \sigma_e^2 I_{n_s}$. We refer (3) as full GPR. If y is multi-dimensional, GPR is performed for each element.

2 Problem formulation

2.1 Observation model

Consider a network consisting of n agents and a cloud. We denote the agent set by $\mathcal{V} \triangleq \{1, \ldots, n\}$. At each time instant t, the observation model of agent i is given as

$$y^{[i]}(t) = \eta(\boldsymbol{z}^{[i]}(t)) + e^{[i]}(t) \tag{4}$$

where $\boldsymbol{z}^{[i]}(t) \in \boldsymbol{\mathcal{Z}} \subseteq \mathbb{R}^{n_z}$ is the input of η, $y^{[i]}(t) \in \boldsymbol{\mathcal{Y}}$ is the observation of the agent, and $e^{[i]}(t)$ is independent zero-mean Gaussian noise with $e^{[i]}(t) \sim \mathcal{N}(0, (\sigma_e^{[i]})^2)$.

2.2 Attack model

All communications between the cloud and the agents are attack-free. Some agents are Byzantine agents, which can behave arbitrarily during learning process, and send arbitrary local predictions to the cloud. We assume that an α fraction of the n agents are Byzantine, and the remaining $1 - \alpha$ fraction are benign. We denote $\mathcal{B} \subset \mathcal{V}$ as the set of Byzantine agents, and thus $|\mathcal{B}| = \lfloor \alpha n \rfloor$. Both the cloud and the benign agents are unaware of the identities of Byzantine agents, but we assume the cloud knows that $\frac{1}{4}$ is an upper bound of α.

Assumption 1 *It holds that* $0 < \alpha < \frac{1}{4}$.

Most existing works on Byzantine-tolerant algorithms, e.g., [7], [5], [9], can tolerate Byzantine agents with $\alpha < \frac{1}{2}$. These algorithms only transmit one variable, e.g., the gradients of the loss function or the model parameter, between the cloud and the agents, whereas our federated GPR algorithm requires to exchange two variables, e.g., local mean and variance, to realize the aggregation of predictions.

2.3 Objective

The objective of this paper is to design an algorithm which can enable the agents and the cloud to correctly learn the function η without requiring the agents to share local streaming data $(\boldsymbol{z}^{[i]}(t), y^{[i]}(t))$ despite some agents are subject to Byzantine attacks.

3 Byzantine-tolerant federated GPR

In this section, we develop a Byzantine-tolerant federated GPR algorithm to solve the problem formulated in Section 2. This algorithm consists of three modules: agent-based local GPR, cloud-based aggregated GPR and agent-based fused GPR.

Algorithm 1 Byzantine-tolerant federated GPR

Input: test inputs $\mathcal{Z}_*$, prior mean function μ, kernel k and noise variance $(\sigma_e^{[i]})^2$ for $i \in \mathcal{V}$
Initialization: $\mathcal{D}(0) = \emptyset$.
for $t = 1, 2, \ldots$ **do**
  for $i \in \mathcal{V}$ **do** {Agent-based local GPR}
    $\mathcal{D}^{[i]}(t) = \mathcal{D}^{[i]}(t-1) \bigcup (\boldsymbol{z}^{[i]}(t), y^{[i]}(t))$
    $\check{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \check{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)} = \text{lGPR}(\mathcal{D}^{[i]}(t))$
  end for
  $\hat{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}(t)}, \hat{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}(t)} = \text{cGPR}(\check{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \check{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)})$ {Cloud-based aggregated GPR}
  for $i \in \mathcal{V}$ **do** {Agent-based fused GPR}
    $\tilde{\boldsymbol{\mu}}^{[i]}_{\mathcal{Z}_*|\mathcal{D}(t)}, (\tilde{\boldsymbol{\sigma}}^{[i]}_{\mathcal{Z}_*|\mathcal{D}(t)})^2 = \text{fGPR}(\check{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \check{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \hat{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}(t)}, \hat{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}(t)})$
  end for
end for

Specifically, first at each time instant t, each agent $i \in \mathcal{V}$ makes its own prediction of η through agent-based local GPR over a given set of points $\mathcal{Z}_* \subseteq \boldsymbol{\mathcal{Z}}$ using local streaming dataset $\mathcal{D}^{[i]}(t) \triangleq (\mathcal{Z}^{[i]}(t), \boldsymbol{y}^{[i]}(t))$ with local input data $\mathcal{Z}^{[i]}(t) \triangleq \{\boldsymbol{z}^{[i]}(1), \ldots, \boldsymbol{z}^{[i]}(t)\}$ and output $\boldsymbol{y}^{[i]}(t) \triangleq [y^{[i]}(1), \ldots, y^{[i]}(t)]^{\mathrm{T}}$. Then the benign agents send correct local predictions to the cloud and the Byzantine agents instead send arbitrary values. Second, the cloud collects all the local predictions and constructs two agent sets by trimming a fraction of outliers with respect to the predictive means and variances. Then the cloud uses a Byzantine-tolerant PoE aggregation rule to compute a global model, and broadcasts the global model to all the agents. Third, the agents refine the predictions on $\mathcal{Z}_*$ by fusing the model from the cloud-based GPR with that from the agent-based local GPR. Note that each agent only communicates local predictions with the cloud, and does not share its local streaming data with the cloud and other agents. The formal description of Byzantine-tolerant federated GPR is given in Algorithm 1.

Algorithm 2 Agent-based local GPR: $\text{lGPR}(\mathcal{D}^{[i]}(t))$

$\mathcal{Z}^{[i]}(t) = \{\boldsymbol{z}^{[i]}(1), \ldots, \boldsymbol{z}^{[i]}(t)\}$
$\boldsymbol{y}^{[i]}(t) = [y^{[i]}(1), \ldots, y^{[i]}(t)]^{\mathrm{T}}$
for $\boldsymbol{z}_* \in \mathcal{Z}_*$ **do**
  Choose $z_*^{[i]}(t) \in \text{proj}(\boldsymbol{z}_*, \mathcal{Z}^{[i]}(t))$
  Compute $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} \leftarrow \begin{cases} \check{\mu}_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} & \text{Benign agent,} \\ \star & \text{Byzantine agent} \end{cases}$
  Compute $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} \leftarrow \begin{cases} \check{\sigma}^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} & \text{Benign agent,} \\ \star & \text{Byzantine agent} \end{cases}$
  Send $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ and $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ to the cloud
end for
return $\check{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \check{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}$

3.1 Agent-based local GPR

To reduce computational complexity, we perform Nearest-neighbor (NN) GPR as the agent-based local GPR to predict for each test point $\boldsymbol{z}_* \in \mathcal{Z}_*$. Instead of feeding the whole training dataset to GPR, the agent-based local GPR only uses the nearest input denoted by $\boldsymbol{z}_*^{[i]}(t) \in \text{proj}(\boldsymbol{z}_*, \mathcal{Z}^{[i]}(t))$ and its corresponding output $y^{[i]}_{\boldsymbol{z}_*^{[i]}(t)}$ to compute the local predictions. Notice that, computation

complexity of the hyperparameter tuning for NN GPR is $\mathcal{O}(t^3)$. In fact, the computation complexity of the hyperparameter tuning can be reduced to $\mathcal{O}(1)$ by using recursive hyperparameter tuning [21]. NN GPR in prediction has computation complexity equivalent to nearest-neighbor search, whose worst case complexity is $\mathcal{O}(t)$, while full GPR has computation complexity $\mathcal{O}(t^3)$ [22]. Denote $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ and $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ the predictive mean and variance output from the agent-based local GPR. The predictive mean and variance of the benign agents are written in the following forms $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} = \check{\mu}_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$, $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} = \check{\sigma}^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ where

$$\begin{aligned}\check{\mu}_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} &\triangleq \mu(\boldsymbol{z}_*) + k(\boldsymbol{z}_*, \boldsymbol{z}_*^{[i]}(t))\mathring{k}(\boldsymbol{z}_*^{[i]}(t), \boldsymbol{z}_*^{[i]}(t))^{-1}(y^{[i]}_{\boldsymbol{z}_*^{[i]}(t)} - \mu(\boldsymbol{z}_*^{[i]}(t))),\\ \check{\sigma}^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} &\triangleq k(\boldsymbol{z}_*, \boldsymbol{z}_*) - k(\boldsymbol{z}_*, \boldsymbol{z}_*^{[i]}(t))\mathring{k}(\boldsymbol{z}_*^{[i]}(t), \boldsymbol{z}_*^{[i]}(t))^{-1}k(\boldsymbol{z}_*^{[i]}(t), \boldsymbol{z}_*)\end{aligned} \tag{5}$$

with $\mathring{k}(\boldsymbol{z}_*^{[i]}(t), \boldsymbol{z}_*^{[i]}(t)) \triangleq k(\boldsymbol{z}_*^{[i]}(t), \boldsymbol{z}_*^{[i]}(t)) + (\sigma_e^{[i]})^2 I_t$. When the agent is a Byzantine agent, its outputs can be arbitrary, and in this case we write $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} = \star$ and $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} = \star$. The implementation of the agent-based local GPR is provided in Algorithm 2.

3.2 Cloud-based aggregated GPR

In the sequel, we consider a Byzantine-tolerant PoE aggregation rule for the cloud. The algorithm is described as follows:

Step 1: For each $\boldsymbol{z}_* \in \mathcal{Z}_*$, the cloud collects potentially corrupted mean $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ and variance $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ from each agent i.

Step 2: The cloud constructs two agent sets to tolerate Byzantine attacks. Without loss of generality, for $i \in \mathcal{V}$, we sort $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ and $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ in ascending order and the corresponding agent index is kept in the sets $\mathcal{T}^{\mu}(t)$ and $\mathcal{T}^{\sigma}(t)$, such that for each $i \in \mathcal{T}^{\mu}(t)$, we have $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} \leq \check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i+1]}(t)}$ and for each $i \in \mathcal{T}^{\sigma}(t)$, we have $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} \leq \check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i+1]}(t)}$. We denote by $\mathcal{T}^{\mu}_{\max}(t) \triangleq \{n - \lfloor \beta n \rfloor + 1, \ldots, n\}$ a set of $\lfloor \beta n \rfloor$ agents with the largest local predictive means, and by $\mathcal{T}^{\mu}_{\min}(t) \triangleq \{1, \ldots, \lfloor \beta n \rfloor\}$ a set of $\lfloor \beta n \rfloor$ agents with the smallest local predictive means where $\beta \in [\alpha, \frac{1}{4})$. Similarly, we denote trimmed sets of variance by $\mathcal{T}^{\sigma}_{\max}(t)$ and $\mathcal{T}^{\sigma}_{\min}(t)$. Then, we define

$$\mathcal{M}(t) \triangleq \mathcal{V} \backslash (\mathcal{T}^{\mu}_{\max}(t) \bigcup \mathcal{T}^{\mu}_{\min}(t)), \quad \mathcal{C}(t) \triangleq \mathcal{V} \backslash (\mathcal{T}^{\sigma}_{\max}(t) \bigcup \mathcal{T}^{\sigma}_{\min}(t)). \tag{6}$$

Notice that $\mathcal{M}(t)$ and $\mathcal{C}(t)$ include $n - 2\lfloor \beta n \rfloor$ agents with extreme values removed. These are the agents whose predictions do not deviate much from the attack-free case.

Step 3: Define a global training dataset as $\mathcal{D}(t) \triangleq \bigcup_{i \in \mathcal{V}} \mathcal{D}^{[i]}(t)$, and a common set as $\mathcal{I}(t) \triangleq \mathcal{M}(t) \bigcap \mathcal{C}(t)$. The following lemma shows that $\mathcal{I}(t)$ is never empty.

Lemma 1 *Suppose that $\beta < \frac{1}{4}$, it holds that $(1 - 4\beta)n \leq |\mathcal{I}(t)| \leq (1 - 2\beta)n$ at each time instant t.*

The following lemma shows that $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ and $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$ for $i \in \mathcal{I}(t)$ are bounded by the values of the benign agents.

Algorithm 3 Cloud-based aggregated GPR: cGPR($\check{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \check{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}$)

```
for t = 1, 2 ... do
    D(t) = ⋃_{i∈V} D^[i](t)
    for z_* ∈ Z_* do
        Construct sets M(t) and C(t) according to (6)
        Compute I(t) = M(t) ⋂ C(t)
        Implement aggregation rule (7)
        Broadcast μ̂'_{Z_*|D(t)} and σ̂'^2_{Z_*|D(t)} to all the agents
    end for
end for
return μ̂'_{Z_*|D(t)}, σ̂'^2_{Z_*|D(t)}
```

Lemma 2 *Let $0 < \alpha \leq \beta < \frac{1}{4}$ hold. For $i \in \mathcal{I}(t)$, we have that $\min_{j\in\mathcal{V}\setminus\mathcal{B}} \left\{\check{\mu}_{\boldsymbol{z}_*|\mathcal{D}^{[j]}(t)}\right\} \leq \check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} \leq \max_{j\in\mathcal{V}\setminus\mathcal{B}} \left\{\check{\mu}_{\boldsymbol{z}_*|\mathcal{D}^{[j]}(t)}\right\}$, and $\min_{j\in\mathcal{V}\setminus\mathcal{B}} \left\{\check{\sigma}^2_{\boldsymbol{z}_*|\mathcal{D}^{[j]}(t)}\right\} \leq \check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} \leq \max_{j\in\mathcal{V}\setminus\mathcal{B}} \left\{\check{\sigma}^2_{\boldsymbol{z}_*|\mathcal{D}^{[j]}(t)}\right\}$.*

Given $\mathcal{I}(t)$, we propose a Byzantine-tolerant PoE aggregation rule below

$$\hat{\mu}'_{\boldsymbol{z}_*|\mathcal{D}(t)} \triangleq \frac{\hat{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}(t)}}{|\mathcal{I}(t)|} \sum_{i\in\mathcal{I}(t)} \check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} \check{\sigma}'^{-2}_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}, \quad \hat{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}(t)} \triangleq \frac{|\mathcal{I}(t)|}{\sum_{i\in\mathcal{I}(t)} \check{\sigma}'^{-2}_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}}. \tag{7}$$

Note that if all the agents are known to be benign, we can simply select $\beta = \alpha = 0$, and hence $\mathcal{M}(t) = \mathcal{C}(t) = \mathcal{V}$. In this case, (7) reduces to the standard PoE in [23]. Then the cloud-based aggregated GPR returns $\hat{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}(t)} \triangleq [\hat{\mu}'_{\boldsymbol{z}_*|\mathcal{D}(t)}]_{\boldsymbol{z}_*\in\mathcal{Z}_*}$ and $\hat{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}(t)} \triangleq [\hat{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}(t)}]_{\boldsymbol{z}_*\in\mathcal{Z}_*}$. The implementation of the cloud-based aggregated GPR is provided in Algorithm 3.

3.3 Agent-based fused GPR

Fused GPR aims to refine predictions of $\eta(\mathcal{Z}_*)$ by integrating the agent-based local GPR with the cloud-based aggregated GPR. When the cloud variance $\hat{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}(t)}$ is lower than the local predictive variances $\check{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)}$, the predictions from the cloud-based aggregated GPR are used. Otherwise, those from the agent-based local GPR are used. Our algorithm is one-round. The refined predictions in the agent-based local GPR will not be transmitted to the cloud, hence it cannot change the final aggregation. The implementation of the agent-based fused GPR is provided in Algorithm 4.

Algorithm 4 Agent-based fused GPR: fGPR$(\check{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \check{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}^{[i]}(t)}, \hat{\boldsymbol{\mu}}'_{\mathcal{Z}_*|\mathcal{D}(t)}, \hat{\boldsymbol{\sigma}}'^2_{\mathcal{Z}_*|\mathcal{D}(t)})$

```
for z_* ∈ Z_* do
    if σ̂'²_{z_*|D(t)} ≤ σ̌'²_{z_*|D^[i](t)} then
        μ̃^[i]_{z_*|D(t)} = μ̂'_{z_*|D(t)},   (σ̃^[i]_{z_*|D(t)})² = σ̂'²_{z_*|D(t)}
    else
        μ̃^[i]_{z_*|D(t)} = μ̌'_{z_*|D^[i](t)},   (σ̃^[i]_{z_*|D(t)})² = σ̌'²_{z_*|D^[i](t)}
    end if
end for
return μ̃^[i]_{Z_*|D(t)}, (σ̃^[i]_{Z_*|D(t)})²
```

3.4 Performance guarantee

This section discusses the robustness of the cloud-based aggregated GPR. We start with the assumptions of the kernel function k.

Assumption 2 *1) (Decomposition). The kernel function $k(\cdot,\cdot)$ can be decomposed in such a way that $k(\cdot,\cdot) = \kappa(D(\cdot,\cdot))$, where $\kappa : \mathbb{R}_+ \to \mathbb{R}_{++}$ is continuous.*

2) (Monotonicity). It holds that $\kappa(s)$ is monotonically decreasing in s and $\kappa(0) = \sigma_f^2$.

Many existing kernel functions can satisfy Assumption 2, e.g., squared exponential kernel [13]. As [13], any non-zero mean Gaussian process can be decomposed into a deterministic process and a zero-mean stochastic process. Without loss of generality and for notational simplicity, we assume η follows a zero-mean Gaussian process in the analysis.

Assumption 3 *It satisfies that $\eta \sim \mathcal{GP}(0, k)$ and η is Lipschitz continuous.*

The target function η is completely specified by Gaussian process with kernel function k. This assumption is common in the analysis of GPR [24]. Denote by $\ell_\eta > 0$ the Lipschitz constant of η, i.e., $\sup_{\boldsymbol{z},\boldsymbol{z}'} |\eta(\boldsymbol{z}) - \eta(\boldsymbol{z}')| \leq \ell_\eta D(\boldsymbol{z}, \boldsymbol{z}')$. For $i \in \mathcal{V}$, we define the dispersion of local data as $d^{[i]}(t) \triangleq \sup_{\boldsymbol{z}\in\boldsymbol{\mathcal{Z}}} D(\boldsymbol{z}, \mathcal{Z}^{[i]}(t))$, which is considered as a density measurement of sampled data distributed within a compact set. For notational simplicity, we denote the distance by $D^{\mathcal{Z}}_{\boldsymbol{z}} \triangleq D(\boldsymbol{z}, \mathcal{Z})$.

We denote $\sigma \triangleq \frac{\sigma_f^2 \sigma_e^{\max}}{\sigma_f^2 + (\sigma_e^{\min})^2}$ where $\sigma_e^{\max} \triangleq \max_{i \in \mathcal{V}}\{\sigma_e^{[i]}\}$ and $\sigma_e^{\min} \triangleq \min_{i \in \mathcal{V}}\{\sigma_e^{[i]}\}$. The following theorem shows that the prediction error from the cloud-based aggregated GPR is upper bounded.

Theorem 1 *(Byzantine-tolerant aggregated GPR: Mean) Let $0 < \alpha \leq \beta < \frac{1}{4}$ and Assumptions 2, 3 hold. For any $\boldsymbol{z}_* \in \boldsymbol{\mathcal{Z}}$ and $0 < \delta < 1$, with probability at least $1 - \delta$, $\left|\hat{\mu}'_{\boldsymbol{z}_*|\mathcal{D}(t)} - \eta(\boldsymbol{z}_*)\right| \leq (1 - \frac{\kappa(d^{\max}(t))}{\sigma_f^2 + (\sigma_e^{\max})^2})\|\eta\|_\infty + \ell_\eta d^{\max}(t) + \sqrt{\frac{(\sigma_e^{\min})^2}{\delta}} + \Delta$ where $\Delta \triangleq \frac{2\sqrt{2}\sigma\alpha\sqrt{(\ln(2n) - \ln\delta)}}{1-4\beta} \frac{\sigma_f^4 + \sigma_f^2(\sigma_e^{\max})^2 - \kappa(d^{\max}(t))^2}{\sigma_f^2(\sigma_e^{\min})^2}$.*

Corollary 1 *(Attack-free cloud-based GPR: Mean) Suppose Assumptions 2 and 3 hold. For any $\boldsymbol{z}_* \in \boldsymbol{\mathcal{Z}}$ and $0 < \delta < 1$, with probability at least $1 - \delta$, it holds that $\left|\hat{\mu}_{\boldsymbol{z}_*|\mathcal{D}(t)} - \eta(\boldsymbol{z}_*)\right| \leq (1 - \frac{\kappa(d^{\max}(t))}{\sigma_f^2 + (\sigma_e^{\max})^2})\|\eta\|_\infty + \ell_\eta d^{\max}(t) + \sqrt{\frac{(\sigma_e^{\min})^2}{\delta}}$.*

Combining Theorem 1 with Corollary 1, we can see that the performance of Byzantine-tolerant GPR can be decomposed into 2 terms: the performance of attack-free cloud-based GPR and the term Δ. Specifically, the term Δ characterizes the performance loss caused by Byzatine agents. Notice that Δ is positively proportional to α and goes to zero as α approaches zero. This implies that the prediction performance improves if the number of Byzantine agents is decreasing. When α is fixed, Δ is proportional to β. This implies that learning performance improves if the set $\mathcal{I}(t)$ expands.

The following theorem gives the lower bound and upper bound of the cloud variance.

Theorem 2 *(Byzantine-tolerant aggregated GPR: Variance) Let $0 < \alpha \leq \beta < \frac{1}{4}$ and Assumption 2 hold. For any $\boldsymbol{z}_* \in \boldsymbol{\mathcal{Z}}$, it holds that $\frac{\sigma_f^2(\sigma_e^{\min})^2}{\sigma_f^2 + (\sigma_e^{\max})^2} \leq \hat{\sigma}'^2_{\boldsymbol{z}_*|\mathcal{D}(t)} \leq \sigma_f^2 - \frac{\kappa(d^{\max}(t))^2}{\sigma_f^2 + (\sigma_e^{\max})^2}$.*

Corollary 2 *(Attack-free cloud-based GPR: Variance) Suppose Assumption 2 holds. For any $\boldsymbol{z}_* \in \boldsymbol{\mathcal{Z}}$, it holds that $\frac{\sigma_f^2(\sigma_e^{\min})^2}{\sigma_f^2 + (\sigma_e^{\max})^2} \leq \hat{\sigma}^2_{\boldsymbol{z}_*|\mathcal{D}(t)} \leq \sigma_f^2 - \frac{\kappa(d^{\max}(t))^2}{\sigma_f^2 + (\sigma_e^{\max})^2}$.*

From Theorem 2 and Corollary 2, we can see that the predictive variances of attack-free cloud-based GPR and Byzantine-tolerant GPR shares the same lower and upper bounds. This is due to the fact that the extreme values are removed by the Byzantine-tolerant PoE in the cloud-based GPR. The proofs of Lemmas 1, 2, Theorems 1, 2 and Corollaries 1, 2 are included in the supplementary document.

4 Numerical experiments

This section experimentally demonstrates the performance of the proposed Byzantine-tolerant federated GPR algorithm using one synthetic dataset and two real-world datasets.

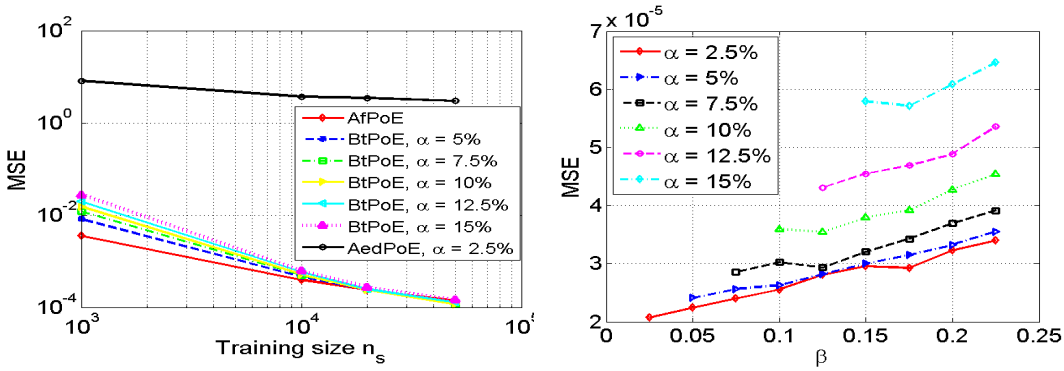
Synthetic dataset: For the observation model (2), we consider the target function introduced in [25]

$$\eta(\boldsymbol{z}) = 5\boldsymbol{z}^2 \sin(12\boldsymbol{z}) + (\boldsymbol{z}^3 - 0.5)\sin(3\boldsymbol{z} - 0.5) + 4\cos(2\boldsymbol{z}), \tag{8}$$

with $\boldsymbol{z} \in \mathbb{R}$ and $e \sim \mathcal{N}(0, 0.01)$. We generate $n_s = 10^3, 5 \times 10^3, 10^4, 5 \times 10^4$ training points in $[0, 1]$, respectively, and choose $n_t = 120$ test points randomly in $[0, 1]$. There are $n = 40$ agents in a network. We partition the training dataset into 40 disjoint groups, and each agent is assigned with n_s/n training data points. We use the following squared-exponential kernel $k(\boldsymbol{z}, \boldsymbol{z}_*) = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(\boldsymbol{z} - \boldsymbol{z}_*)^2)$, and use Mean Square Error (MSE) to evaluate the proposed algorithm in terms of consistency, the effect of α and the effect of β in the following experiments.

In the first experiment, we let $\alpha = 2.5\%, 5\%, 7.5\%, 10\%, 12.5\%, 15\%$. We randomly choose the agents in the network to be compromised by same-value attacks [26], and let $\beta = \alpha$. Specifically, for each test point $\boldsymbol{z}_*$, the Byzantine agents only change the local predictive means to 100, that is, $\check{\mu}'_{\boldsymbol{z}_*|\mathcal{D}^{[i]}(t)} = 100$ for all t, and send this incorrect prediction to the cloud. The local predictive variances for all the agents remain unchanged. Fig. 1(a) compares the performances of the proposed Byzantine-tolerant PoE (BtPoE) with the standard PoE. As a benchmark, the bottom curve depicts the prediction errors of the standard PoE (AfPoE) when there is no Byzantine attack in the network; it shows that the standard PoE is consistent in the attack-free scenario. However, when the aforementioned Byzantine attacks are launched, the top curve shows that the standard PoE can no longer

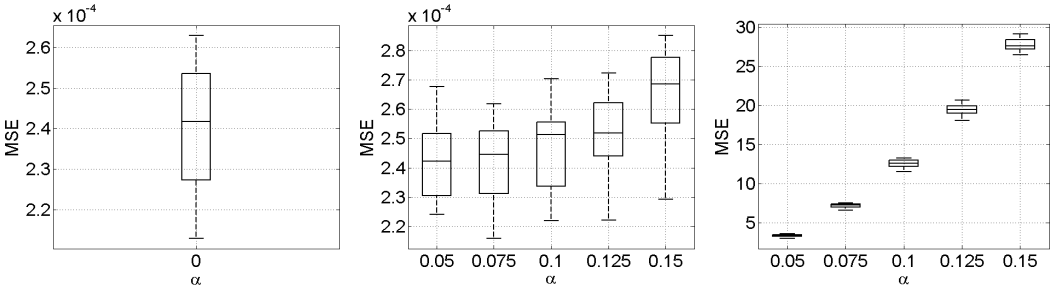
remain consistent. Note that there are 6 curves by attacked standard PoE (AedPoE), and only the closest one to the bottom curve is shown in Fig. 1(a)[1].



(a) Consistency evaluation (b) Prediction performance on different β

Figure 1: Prediction performance of the cloud-based aggregated GPR.

Notice that the performance of the Byzantine-tolerant PoE increases as α, the ratio of Byzantine agents, decreases. Fixing the training data size $n_s = 10^4$, we replicate the experiments for 50 times with different α and observation noise e. Figs. 2(a) and 2(b) show that the performances of the Byzantine-tolerant PoE are comparable to that of the attack-free standard PoE with consistency preserved. The prediction errors for both methods are in the order of 10^{-4}, while Fig. 2(c) shows that the attacked standard PoE has prediction errors ranged from 2.5 to 30. This demonstrates that the proposed PoE aggregation rule has strong ability to tolerate Byzantine attacks.



(a) Attack-free standard PoE (b) Byzantine-tolerant PoE (c) Attacked standard PoE

Figure 2: Comparison of the prediction performance on different α.

In the second experiment, we evaluate the prediction performance with regard to β. All associated parameters are the same as those in the above experiments. Notice that for each α, we initially let $\beta = \alpha$ and then increase β to 0.225 with increment 0.025, such that the assumption $\alpha \leq \beta < \frac{1}{4}$ is satisfied. The results are shown in Fig. 1(b). This shows that as the number of trimmed agents increases, the performance of the proposed algorithm grows worse, since more information is excluded in the aggregation. This result validates the conclusion in Theorem 1.

Table 1: Comparison of the prediction performance on different functions

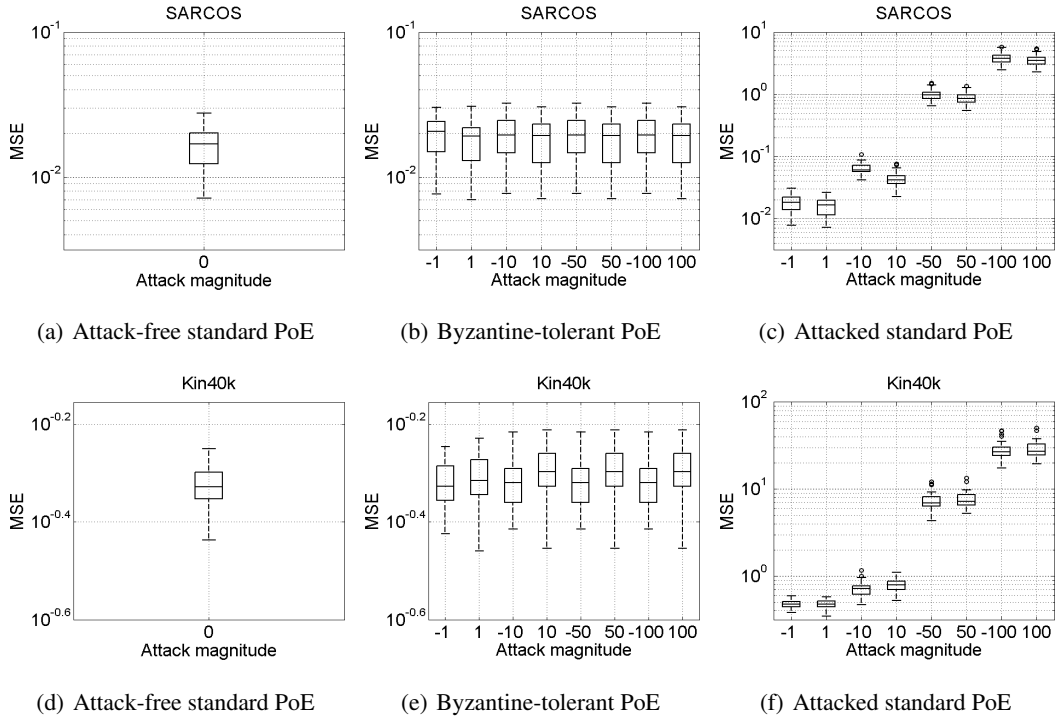
Algorithm	Attack-free standard PoE	Byzantine-tolerant PoE	Attacked standard PoE
MSE	$0.0049 \pm 0.007 \times 10^{-3}$	$0.0236 \pm 0.172 \times 10^{-3}$	$26.5339 \pm 0.019 \times 10^{-3}$

In the third experiment, we conduct Monte Carlo simulations to evaluate the performance of Byzantine-tolerant PoE on different functions. We repeat the experiment for 50 times by sampling different functions and observation points. Specifically, we replace $\sin(12\boldsymbol{z})$ in (8) with

[1]Due to space limitation, the prediction comparisons of these three PoEs are given in the supplementary document.

$\sin(12\boldsymbol{z}) + \epsilon'$ where $\epsilon' \sim \mathcal{N}(0, 0.01i)$, $i = 1, \ldots, 50$. We assign $n = 40$ agents in the network and sample $n_s = 10^4$ training points in $[0, 1]$, and then choose $n_t = 120$ test points randomly in $[0, 1]$. We also use the square-exponential kernel $k(\boldsymbol{z}, \boldsymbol{z}_*)$. There are six Byzantine agents, i.e., $\alpha = 15\%$, and we let $\beta = \alpha$. Experiment results are presented in Table 1, and it shows that the performance of the Byzantine-tolerant PoE is comparable to that of the attack-free standard PoE, noticing that the order of the prediction errors for the Byzantine-tolerant PoE are 10^{-2}. For comparison, the MSE by the attacked standard PoE is about 26.534. This demonstrates that the proposed Byzantine-tolerant aggregation rule is effective for learning different functions.

Real-world dataset: We conduct an experiment to evaluate the prediction performance with different attack magnitudes using two real-world datasets. The first dataset is collected from a seven degrees-of-freedom SARCOS anthropomorphic robot arm [13]. The goal is to learn a function η. The input of η is 21-dimensional including 7 joint positions, 7 joint velocities and 7 joint accelerations; whereas the output is 1-dimensional, representing the torque of a joint. We randomly sample 40,000 data points for training and 40 data points for testing. In the experiment, we partition $40,000$ training data into $n = 40$ disjoint groups, and assign each group to an agent. The second dataset *Kin40k* [27] is created using a robot arm simulator. We randomly choose $9,000$ training points and 40 test points. We also partition the training data evenly into 40 groups. The magnitudes of attack we consider are $\pm 0.1, \pm 1, \pm 10, \pm 50, \pm 100$. We randomly sample the training data points and repeat the experiments for 50 times. Fig. 3 shows the prediction performances. For both real-world datasets, when the attack magnitude is increasing, the prediction errors of the attacked standard PoE largely increases, whereas those of the attack-free standard PoE and the Byzantine-tolerant PoE are consistently maintained at a low level. It shows that the proposed Byzantine-tolerant PoE is resilient to the attack.



(a) Attack-free standard PoE (b) Byzantine-tolerant PoE (c) Attacked standard PoE

(d) Attack-free standard PoE (e) Byzantine-tolerant PoE (f) Attacked standard PoE

Figure 3: Prediction performance with different attack magnitudes on datasets SARCOS and *Kin40k*.

5 Conclusion

We propose a Byzantine-tolerant federated GPR algorithm, which is able to tolerate less than one quarter Byzantine agents in the network. We characterize the upper bounds on the prediction errors and the lower and upper bounds of the predictive variances. Experiments are conducted to demonstrate the robustness of Byzantine-tolerant GPR algorithm. Future work will focus on the comparison of the learning performance between the agent-based local GPR and the agent-based fused GPR, and the quantification of the benefits the cloud brings to the agents.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

[2] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proceedings of International Conference on Neural Information Processing Systems*, pp. 1–5, 2016.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[4] F. Yin, Z. Lin, Q. Kong, Y. Xu, D. Li, S. Theodoridis, and S. R. Cui, "Fedloc: Federated learning framework for data-driven cooperative localization and location data processing," *IEEE Open Journal of Signal Processing*, vol. 1, pp. 187 – 215, 2020.

[5] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of International Conference on Neural Information Processing Systems*, pp. 118–128, 2017.

[6] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[7] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the International Conference on Machine Learning*, pp. 5650–5659, 2018.

[8] D. Data and S. Diggavi, "Byzantine-resilient high-dimensional SGD with local iterations on heterogeneous data," in *Proceedings of the International Conference on Machine Learning*, pp. 2478–2488, 2021.

[9] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[10] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[11] L. Bottou, *Online algorithms and stochastic approximations*. Cambridge University Press, 1998.

[12] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[13] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[14] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gausian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[15] H. Xiao and C. Eckert, "Lazy Gaussian process committee for real-time online regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 969–976, 2013.

[16] K. Das and A. N. Srivastava, "Sparse inverse kernel Gaussian process regression," *Statistical Analysis and Data Mining*, vol. 6, no. 3, pp. 205–220, 2013.

[17] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels," *Journal of Machine Learning Research*, vol. 7, no. 95, pp. 2651–2667, 2006.

[18] Y. Kuang, T. Chen, F. Yin, and R. Zhong, "Recursive implementation of Gaussian process regression for spatial-temporal data modeling," in *International Conference on Wireless Communications and Signal Processing*, pp. 1–7, 2019.

[19] M. Liu, G. Chowdhary, B. C. Da Silva, S.-Y. Liu, and J. P. How, "Gaussian processes for learning and control: A tutorial with examples," *IEEE Control Systems Magazine*, vol. 38, no. 5, pp. 53–86, 2018.

[20] M. J. Wainwright, *High-dimensional statistics: A non-asymptotic viewpoint*. NY Cambridge University Press, 2019.

[21] M. F. Huber, "Recursive Gaussian process: On-line regression and learning," *Pattern Recognition Letters*, vol. 45, pp. 85 – 91, 2004.

[22] M. R. Abbasifard, B. Ghahremani, and H. Naderi, "A survey on nearest neighbor search methods," *International Journal of Computer Applications*, vol. 95, no. 25, 2014.

[23] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[24] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, 2012.

[25] H. Liu, J. Cai, Y. Wang, and Y. S. Ong, "Generalized robust Bayesian committee machine for large-scale Gaussian process regression," in *Proceedings of the International Conference on Machine Learning*, pp. 3131–3140, 2018.

[26] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1544–1551, 2019.

[27] M. W. Seeger, C. K. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *International Workshop on Artificial Intelligence and Statistics*, pp. 254–261, 2003.