Trust-Adaptive Krum Algorithm for Federated Machine Learning

Spring 2024 CS 525 Survey Report (G15)

Lilesh Kurella
lileshk2@illinois.edu

Vineet Chinthakindi
vineetc2@illinois.edu

Shruthik Musukula
srm14@illinois.edu

Abstract—Federated Learning (FL) has emerged as a pivotal paradigm for distributed processing and model training. This paper addresses the limitation of FL systems and Krum's algorithm [1], which lacks a mechanism of trustworthiness. We propose a novel trust metric that complements Krum's scoring by incorporating a historical rolling average of node contributions and taking into account the quality of contribution. We detail our methodology for optimizing the trust metric through a grid search of coefficient values conducted under simulated conditions with variable proportions of adversarial nodes. This framework offers a novel approach to trust-based evaluation that promises to mitigate the impact of adversarial behavior and enhance model integrity, thereby setting a new standard for robust distributed ML practices.

1 INTRODUCTION

Machine Learning (ML) has become the cornerstone of technological advancement. The growth in ML use cases has led to an influx in volume, velocity, and variety of data that traditional data processing software cannot handle[14]. Federated Learning (FL), is a strategy where a distributed network of nodes each process their own subset of data and then combine their updates to form a collective model [10]. Distributed ML and FL introduce new challenges, particularly in terms of system security and integrity [13].

Krum's algorithm[1], is a solution that employs stochastic gradient descent (SGD) to score gradients from client nodes in an FL system. This strategy helps impose robustness and tolerate adversarial attacks from client nodes against the global model, characterizing it as a byzantine-tolerant system. While this method bolsters system resilience, it does not actively deter nodes from repetitively compromising the global model. Krum's algorithm does not incorporate a mechanism for evaluating the trustworthiness of individual nodes.

1.1 Proposed Solution: Establishing Trust

We introduce a scalable trust metric designed to enrich Krum's algorithm by integrating a trust-based scoring mechanism for client gradients. The trust metric draws upon the historical contributions' quality from each client node, by employing a rolling average calculated over the last hundred iterations for each client node. This average encapsulates the deviation between a node's Krum score and that of the node whose gradient was ultimately chosen for the model update. Through this approach, we aim to provide a more nuanced assessment of each node's reliability and contribution quality, ensuring a more robust and trustworthy FL environment.

$$S_{\text{final}} = (1 - \beta) \times S_{\text{Krum}} + \beta \times S_{\text{Trust}} \tag{1}$$

The final score, S_{final}, is derived by merging the Krum score, S_{Krum}, which identifies and mitigates outlier contributions, with the Trust score, S_{Trust}, which assesses a client's historical reliability. The weighting factor, β ranging between [0,1), modulates the influence of each component in the final score, as shown in Equation 1, balancing immediate data quality against the consistency of past contributions.

1.2 Planned Approach

We plan to create a FL model utilizing the MNIST dataset [2] to address computer vision classification. The model will be evaluated based on accuracy, precision, recall and F1 score. In order to optimize the performance of our FL system in the presence of malicious nodes, we plan to conduct a comprehensive grid search to identify the most effective coefficients for our trust metric.

The grid search strategy will be executed under different simulated conditions, each with a specific percentage of nodes designated as malicious. This will allow us to observe the robustness of the system against varying levels of adversarial behavior. The resulting data will be analyzed to determine the most effective strategy for model accuracy.

1.2.1 Exploratory Alternatives.

(1) **Threshold Metric** We plan to enhance the trust scoring system by including a threshold metric that measures a client node's consistency, based on how often its gradient ranks in the top 50 percent of all evaluated gradients. This aims to gauge both

the node's performance deviation and its comparative ranking.

(2) **Automated Trust-Based Client Fine tuning** Another approach worth considering is manipulating the learning rate for client nodes based on their trust scores. This approach should serve as a deterrent against consistently divergent contributions to our central learning model.

2 RELATED WORK

One of the earliest papers that looked into creating a reputation management stream for peer to peer networks introduced the *EigenTrust* algorithm [9]. The algorithm treats trust within a network as a quantifiable measure assigning each node in a network a trust value based on behavior and feedback from other nodes. This concept can be borrowed for use in FL since information must be shared securely and accurately while malicious nodes are present in the system. Trust can be used used a metric to help maintain a robust global model that can be distributed to the clients in the network.

2.1 Distributed SGD Network Topologies

When dealing with distributed SGD architectures, there exist e two major communication methods: synchronous and asynchronous [3]. The synchronous architecture consists of the traditional server and multi-client model, where each worker node calculates model gradients in parallel before a parameter server collects and produces an updated model for downstream workers. The asynchronous architecture consists of two approaches: one with multiple servers and clients and one where each node takes on the role of server and client. The first approach involves multiple servers communicating with multiple clients in a decentralized setting. The latter topology is similar to a peer-to-peer network. Workers communicate with their neighbors to share calculated model parameters and aggregate data locally. We focus on the centralized, synchronous distributed SGD architecture when applying our approach discussed in the previous section.

2.2 Reputation as an Incentive Mechanism

Current approaches to FL lack a reliable trustworthy incentive mechanism [15]. A proposed secure and trustworthy blockchain framework (SRB-FL) is designed to address this major challenge [12]. SRB-FL focuses on improving the reliability of data provided by users in a system designed for FL workloads through an incentive mechanism known as subjective multi-weight logic. This technique encourages other FL devices to provide reliable model updates to maintain their reputation. While SRB-FL is designed to work in a blockchain ecosystem, it provides major takeaways to enhance the security and reliability of FL devices which we aim to explore in the context of a centralized FL system.

2.3 Byzantine-Tolerant FL

In Google's first FL system implementation [11], a server-side aggregation algorithm was implemented to determine global model updates from client calculated gradients. By training on local clients using SGD to minimize loss, the corresponding gradients would be returned by the clients and averaged in the central server before weight changes were passed to clients for them to update their local model. However, this algorithm does not guarantee robustness against malicious byzantine behavior.

In order to ensure Byzantine tolerance a novel new aggregation algorithm called *Krum* was created to better handle arbitrary byzantine failures in a system [1]. The authors demonstrate that no general gradient descent update that uses a linear combination of vectors can truly tolerate a byzantine failure. The *Krum* algorithm is designed for robust aggregation of gradients by selecting one or more vectors from a set proposed by different workers in a distributed system.

Given a set of vectors, *Krum* computes pairwise distances between all vectors. For each vector v_i, it calculates the sum of distances to its k nearest neighbors, where $k = n - f - 2$, with n being the total number of vectors and f the maximum number of tolerable Byzantine workers. The vector with the smallest sum of distances is chosen for the model update.

In the *m-Krum* extension, the algorithm selects m vectors by iteratively applying *Krum*. After each selection, the chosen vector is removed, and *Krum* is reapplied, under the condition $n - m > 2f + 2$. This approach utilizes more information from the workers while maintaining robustness. Other systems that build on the proposed *Krum* algorithm include *Median-Krum* [6], a variation on top of the regular *Krum* algorithm, where the aggregation calculation uses the median of the immediate neighbor gradient vectors rather than the mean.

However, malicious clients can produce improper local models and corrupt the global model. An existing approach exists to combat this issue by incorporating the idea of bootstrapping trust [5]. In this approach, the service provider is trained on its local model (the root dataset) and maintains a model (the server model) that produces new gradients as well. All subsequent local model updates from client nodes are normalized to reduce the effects of malicious client data.

Each client receives a trust score based on whether or not its model update is in the direction of the server model update. Finally, the server computes the average of all provided model weights scaled by the assigned trust score.

The main goal of most FL approaches is to protect the privacy of client data while accurately training a global model. However, without directly viewing and evaluating client data, such systems are left vulnerable to poisoning methods like backdoor and label-flipping attacks. *FedBayes* [16] aims to mitigate the effects of such attacks by calculating a normalized distribution for the global model based on the mean and standard deviations of model weights. Using a cumulative distribution function, the probability of each weight provided by client models can then be calculated against the expected model weights in the global model. The lower the probability of a weight, the less it should be trusted when updating the global model. We aim to incorporate additional data such as the history of a client's contributions to evaluate trust at a given iteration.

A novel aggregation rule named *Zeno* is introduced for FL [4], which effectively handles an unlimited number of Byzantine (malicious or faulty) clients while ensuring strong convergence properties. Zeno operates by assigning a reliability score to each participating node. This score, derived using a first-order stochastic oracle, is a combination the predicted reduction in the loss function and the estimated size of the update from each node. At the central server, the average gradient is calculated from the updates of nodes with the highest scores. The server then compares this average gradient with the actual gradient values to determine the likelihood of an update being malicious.

2.4 Identifying Malicious Clients

Another system known as *PeerReview* solves the issue of identifying malicious nodes and defending benign data by ensuring that Byzantine faults are eventually detected and linked to a malicious client. Additionally, it ensures that a benign client cannot be incorrectly identified as a malicious node. *PeerReview* [8] maintains a secure log of all client messages. The use of logs and authenticators builds a safe, append-only record of system activity. By scanning collected logs, it ensures that malicious clients are identified by benign clients. This solution has been proven to work in the context of a network filesystem, a peer-to-peer system, and an overlay multicast system, but not a traditional server and multi-client FL environment.

Another approach for specific identification of malicious attacks in FL was presented by Fung et al. [7] using a *FoolsGold* algorithm. *FoolsGold* is designed to mitigate the effects of Sybil attacks in FL environments. In a Sybil attack, a malicious entity creates multiple fake identities (Sybils) to influence the training process disproportionately. *FoolsGold* analyzes the similarity of updates contributed by different clients since Sybils are assumed to provide similar updates. The algorithm weights updates accordingly in the aggregation process based on this similarity analysis; thus, similar updates are weighted lightly. This enhances the robustness of FL systems against coordinated attacks without relying on any central trust authority. *FoolsGold* operates under the assumption that honest participants' data are diverse enough such that their updates are sufficiently distinct from each other.

3 TIMELINE/EXPECTED MILESTONES

- **Survey Report** - [Feb 25]
- **System Setup** - [Feb 26 - Mar 11]
  1. Dataset Acummulation (MNIST)
  2. VM initialization
  3. Client Node setup, Malicious Node partitioning
  4. Central Node Pipeline Setup, Krum implementation, Trust Scoring calculations
- **Initial Experimentation** - [Mar 12 - Mar 30]
  1. Gather Baseline Krum Statistics with Model Evaluation
  2. Assess Krum + Trust Statistics with Model Evaluation
- **Midterm Report** - [Mar 31]
- **Robust Experimentation** [Apr 1 - May 4]
  1. Run Analysis with Different Coeffecients
  2. Experiment over varying percentage of malicious nodes
- **Final Report** - [May 5]

REFERENCES

[1] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Byzantine-tolerant machine learning. *arXiv preprint arXiv:1703.02757*.

[2] L. Bottou et al. 1994. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*. Vol. 2, 77–82 vol.2. doi: 10.1109/ICPR.1994.576879.

[3] Djamila Bouhata, Hamouma Moumen, Jocelyn Ahmed Mazari, and Ahcène Bounceur. 2022. Byzantine fault tolerance in distributed machine learning : a survey. (2022). arXiv: 2205.02572 [cs.DC].

[4] Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. Zeno: an interactive framework for behavioral evaluation of machine learning. doi: 10.1145/3544548.3581268.

[5] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2020. Fltrust: byzantine-robust federated learning via trust bootstrapping. *CoRR*, abs/2012.13995. https://arxiv.org/abs/2012.13995 arXiv: 2012.13995.

[6] Francesco Colosimo and Floriano De Rango. 2023. Median-krum: a joint distance-statistical based byzantine-robust algorithm in federated learning. isbn: 9798400703676. doi: 10.1145/3616390.3618283.

[7] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2020. Mitigating sybils in federated learning poisoning. arXiv: 1808.04866 [cs.LG].

[8] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. 2007. Peerreview: practical accountability for distributed systems. *SIGOPS Oper. Syst. Rev.*, 41, 6, (Oct. 2007), 175–188. doi: 10.1145/1323293.1294279.

[9] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. 2003. The eigentrust algorithm for reputation management in p2p networks. *Proceedings of the 12th international conference on World Wide Web*.

[10] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[11] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2023. Communication-efficient learning of deep networks from decentralized data. arXiv: 1602.05629 [cs.LG].

[12] Hajar Moudoud, Soumaya Cherkaoui, and Lyes Khoukhi. 2022. Towards a secure and reliable federated learning using blockchain. *CoRR*, abs/2201.11311. https://arxiv.org/abs/2201.11311 arXiv: 2201.11311.

[13] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. 2020. Federated learning for wireless communications: motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58, 6, 46–51.

[14] Oracle. 2024. What is big data? https://www.oracle.com/big-data/what-is-big-data/. Accessed: 2024-02-25. (2024).

[15] Asadullah Tariq, Mohamed Adel Serhani, Farag Sallabi, Tariq Qayyum, Ezedin S. Barka, and Khaled A. Shuaib. 2023. Trustworthy federated learning: a survey. (2023). arXiv: 2305.11537 [cs.AI].

[16] Marc Vucovich, Devin Quinn, Kevin Choi, Christopher Redino, Abdul Rahman, and Edward Bowen. 2023. Fedbayes: a zero-trust federated learning aggregation to defend against adversarial attacks. (2023). arXiv: 2312.04587 [cs.CR].