LEGATO: A LayerwisE Gradient AggregaTiOn Algorithm for Mitigating Byzantine Attacks in Federated Learning

Kamala Varma, Yi Zhou, Nathalie Baracaldo, and Ali Anwar, *IBM Research, San Jose, CA, USA*

INTRODUCTION

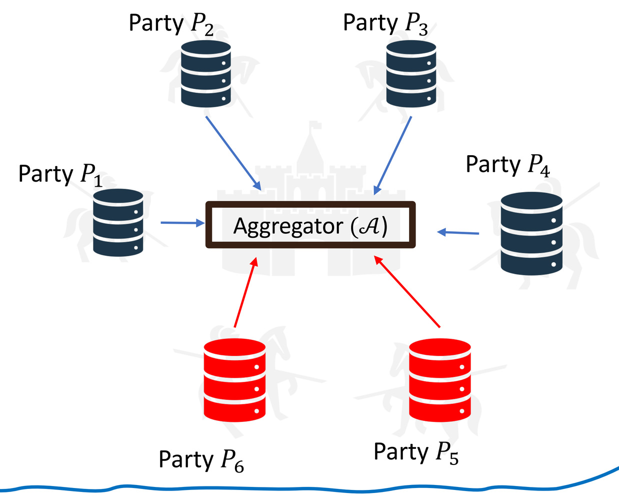
How do we train a model without violating privacy or regulatory constraints?

Federated Learning:

- A group of parties collaboratively trains a machine learning model without sharing/revealing training data
- Only model updates, such as model weights or gradients are shared
- More data better models (Google coined the term in 2016)

Existing Challenges:

- Data heterogeneity: each party may have a dissimilar data distribution (NON-IID)
- Byzantine threats:
  - Byzantine failures: parties crash/stall when sending updates, computation errors
  - Byzantine attacks: the existence of malicious parties



How to deal with Byzantine threats in federated learning without compromising model performance?

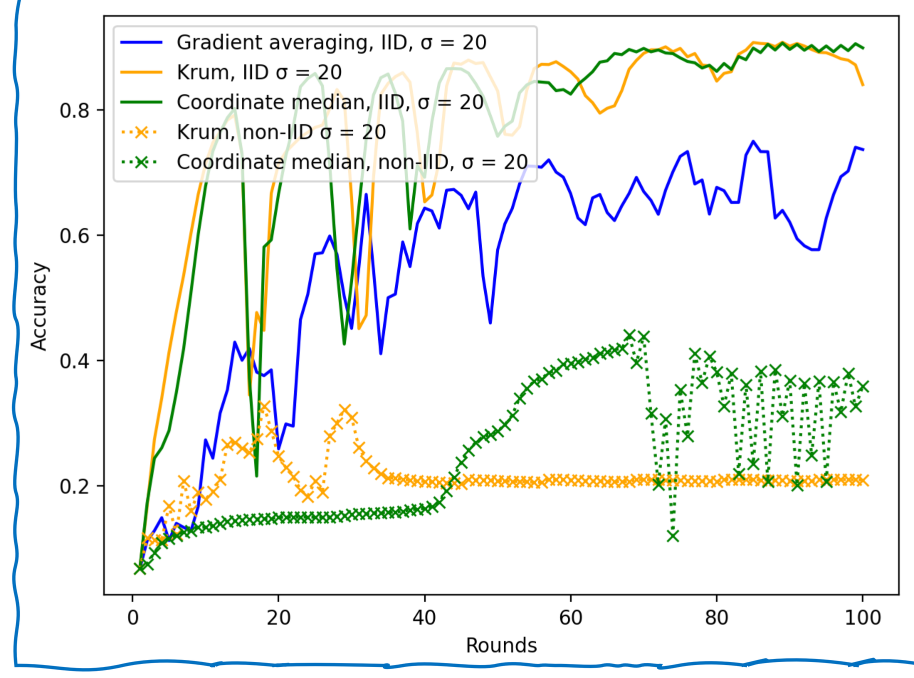
RELATED WORK

- Robust statistics:
  - Coordinate median [1] - use coordinate-wise geometric median as the aggregated gradient
  - Geometric median of means, trimmed mean, repeated median
- Pruning updates from malicious parties:
  - Krum [2] - chooses one party's gradients having the smallest ℓ_2 norms with all other parties' gradients, computation complexity $\mathcal{O}(n^2)$
  - Based on Krum: Multi-Krum and Bulyan
- Others:
  - Distributed momentum [3] – uses momentum at the party side to strengthen existing robust aggregation algorithms (i.e. Krum, median)
  - Residual-based reweighting [4] – reweights updates by party based on gradient residuals from a repeated median regression line
  - FoolsGold [5] – adaptive learning rate for each party based on contribution similarity

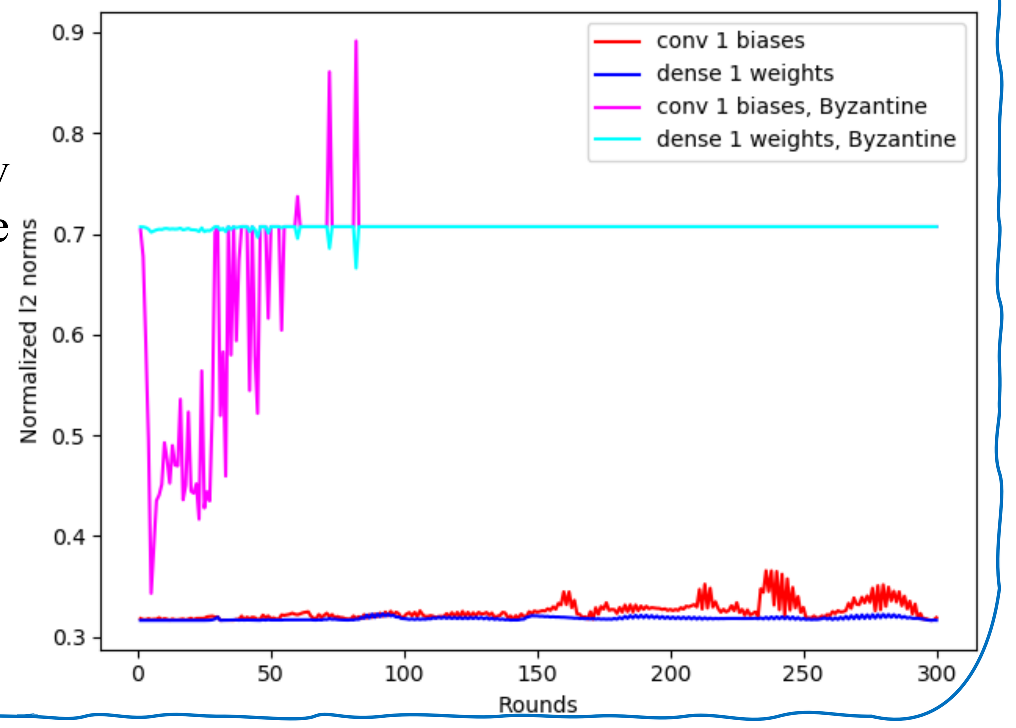
Drawbacks: Rely on assumptions of bounded honest gradients, which DOES NOT hold in NON-IID case.

PILOT STUDY

Existing robust aggregation methods **failed** in NON-IID case



Certain layers (conv 1 in right figure) are **more vulnerable** than other layers (dense) to Byzantine attacks



Exp. Setup: each parties with 1K data points from MNIST. Byzantine parties executing a Gaussian attack with μ= 0 and σ= xx

OUR SOLUTION - LEGATO

Algorithm 4: FEDERATED LEARNING WITH LEGATO.

Aggregator *Maximum global round K, and a learning rate policy $\{\eta_k\}$.*
  Initialize w_0;
  for *round* $k = 1, \ldots, K$ **do**
    $\mathcal{G}^k \leftarrow$ new list;
    Query party $p \in \mathcal{P}$ with the current global model weights w_{k-1} for its current gradient G_p^k and add it to $\mathcal{G}^k$;
    `// Aggregates gradients`
    $\mathcal{G}_{agg}^k$ = LEGATO($\mathcal{G}^k$) `// (Algorithm 5)`
    $\boxed{w_k = w_{k-1} - \eta_k \mathcal{G}_{agg}^k;}$
  return w_K
Party *Each party $p \in \mathcal{P}$ owns its local dataset, $\mathcal{D}_p$, training batch-size B and the current model weights w_k*
  Initialize the local model with $w_0 = w_k$;
  $\boxed{g = \nabla \ell(w_0; B)}$; `// ℓ denotes the loss function.`
  return g

Algorithm 5: LEGATO. An aggregation algorithm to aggregate gradients at round k.

Aggregator *A list of current parties' gradients $\mathcal{G}^k$, a log of recent past party's gradients with maximum size m $GLog := [\mathcal{G}^{k-m}, \mathcal{G}^{k-m+1}, \ldots, \mathcal{G}^{k-1}]$ where* $\mathcal{G}^k = [G_1^k, G_2^k, \ldots, G_n^k]$
  if $k = 1$ **then**
    Initialize an empty log $GLog$;
  else
    UpdateGradientLog($GLog$, $\mathcal{G}^t$);
  for $\mathcal{G}^k$ *in* $GLog$ **do**
    for *Each party p in $\mathcal{P}$* **do**
      $x_p = \| G_p^k \|_2$;
    for *each layer l* **do**
      $P_l^k \leftarrow \frac{\|[g_{1,l}^k, g_{2,l}^k, \ldots, g_{n,l}^k]\|_2}{\|[x_0, x_1, \ldots, x_n]\|_1}$;
for *each layer l* **do**
  $w_l \leftarrow \text{Normalize}\left(\frac{1}{\sqrt{\text{Var}(P_l^1, \ldots, P_l^k)}}\right)$;

Evaluating robustness factor of each layer

for *p in $|\mathcal{P}|$ and each layer l* **do**
  $G_{p,l}^* \leftarrow w_l G_{p,l}^k + \frac{1-w_l}{m-1} \sum_{j=1}^{m-1} G_{p,l}^{k-j}$;

Layerwise gradient reweighting

return $\mathcal{G}_{agg}^k \leftarrow \sum_{p \in \mathcal{P}} G_{p,l}^*$;

UpdateGradientLog($GLog$, $\mathcal{G}^k$):
  $GLog \leftarrow GLog + \mathcal{G}^k$;
  if $len(GLog) > m$ **then**
    $GLog \leftarrow GLog[1:]$

Maintenance of the log of historic gradients from parties

Threat Model:

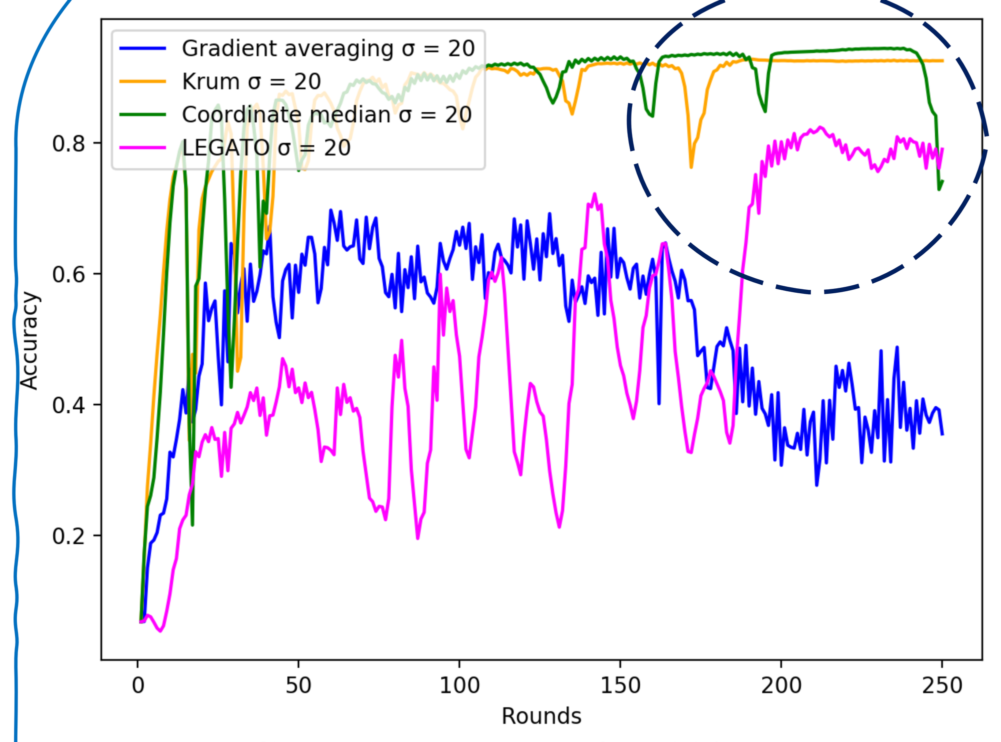
- The aggregator is honest and wants to detect malicious or erroneous gradients received during the training process.
- Parties may be dishonest and may collude with each other to evade detection.

Proposition 1. *LEGATO has time complexity $\mathcal{O}(dn + d)$.*
Proposition 2. *LEGATO has space complexity $\mathcal{O}(dnm)$.*

n: number of parties; d: layer dimension; m: size of the gradient log.

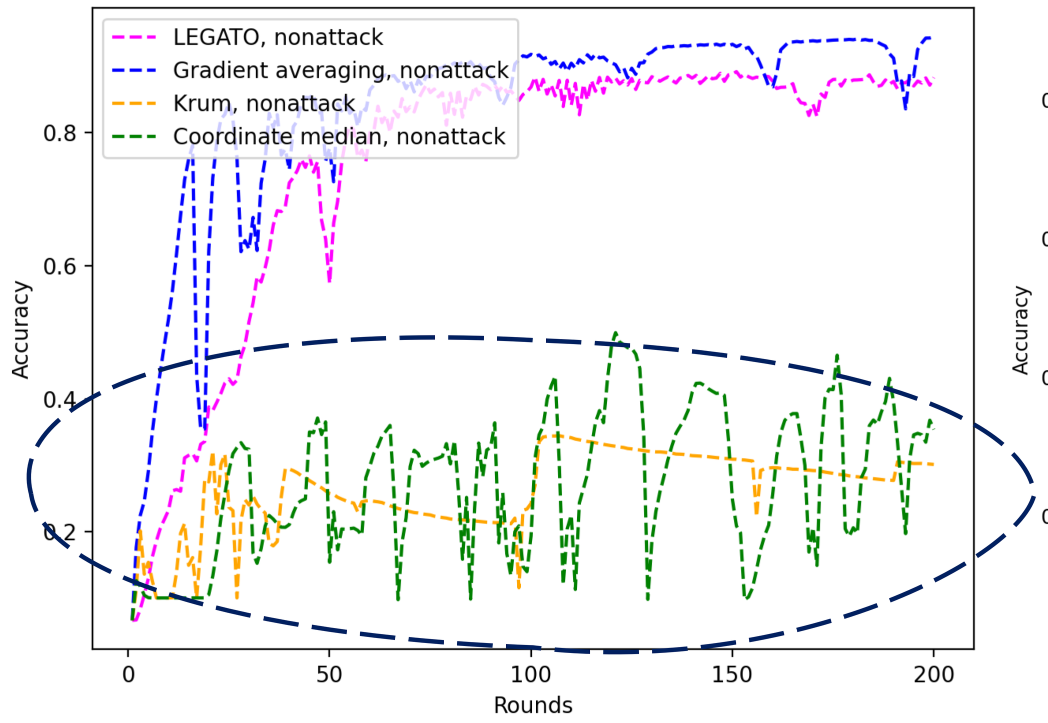
PRELIMINARY EXPERIMENTS

Gaussian attack, IID, 4 Byzantine parties



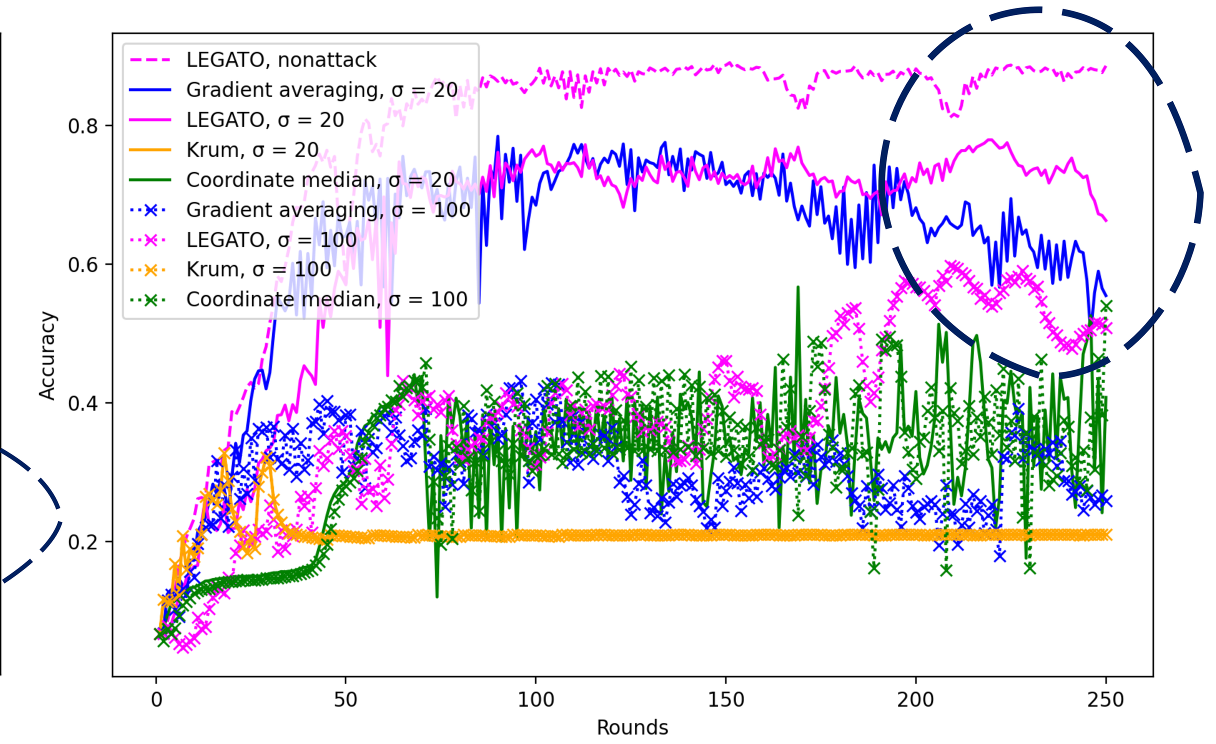
LEGATO catches up after 200 rounds

Non-attack, NON-IID data, WITHOUT Byzantine parties



Robust algorithms fail to produce good models for NON-IID case!

Gaussian attack, NON-IID data, 4 Byzantine parties



LEGATO consistently performs better than baselines for NON-IID case!

Experimental setup: MNIST, 25 total parties, learning rate= .03, batch size = 50, 1K points per party, log size 10. For NON-IID case, each party only has data from one class.

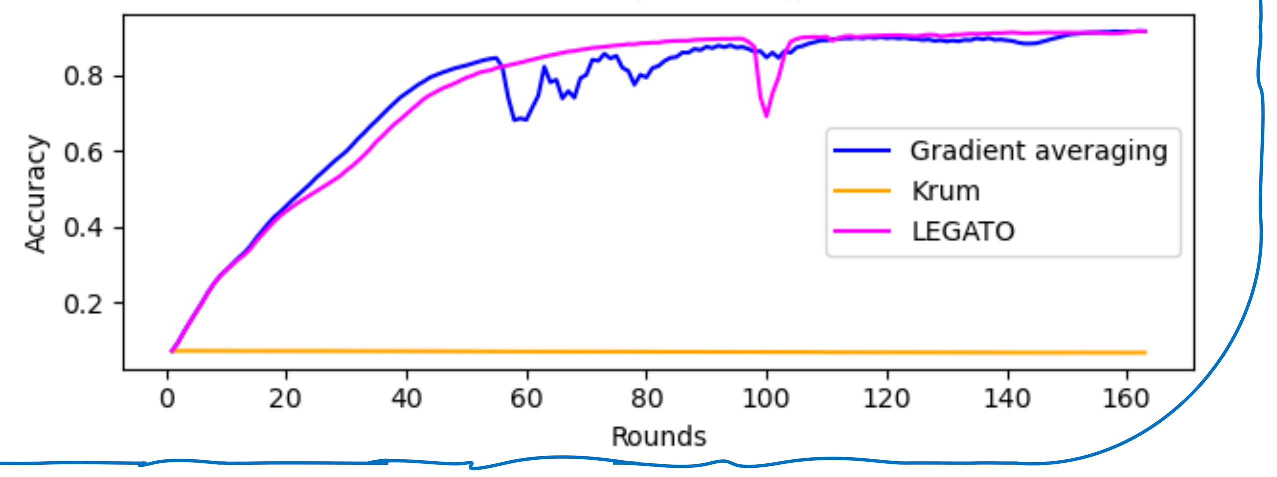
Attacks:

- Gaussian attack [6] - effective, common, replies randomly drawn from $N(0, \sigma I)$
- Fall of Empires (FoE) [7] – designed to break Krum, a group of parties to craft the attack with $u_1 = u_2 = \cdots = u_n = -\epsilon/m \sum_{i=1}^{m} v_i$

$U = \{u_1, \ldots, u_n\}$ *byzantine gradients* $V = \{v_1, \ldots, v_m\}$ *honest gradients*

Robust algorithms fail to produce good models, LEGATO is not affected by this attack!

FoE attack, IID data, 11 Byzantine parties, $\epsilon = 0.001$



[1] Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.
[2] Blanchard, P., Guerraoui, R., Stainer, J., et al. Machine learning with adversaries: Byzantine tolerant gradient descent. NIPS, pp. 118–128, 2017.
[3] El-Mhamdi, E., Guerraoui R., and Rouault, S., Distributed Momentum for Byzantine-resilient Learning. *arXiv preprint arXiv:2003.00010*, 2020.
[4] Fu, S., Xie, C., Li, B., and Chen, Q., "Attack-resistant federated learning with residual-based reweighting," *arXiv preprint arXiv:1912.11464*, 2019.
[5] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.
[6] Xie, Cong, Oluwasanmi Koyejo, and Indranil Gupta. "Generalized byzantine-tolerant sgd." *arXiv preprint arXiv:1802.10116* (2018).
[7] Xie, Cong, Oluwasanmi Koyejo, and Indranil Gupta. "Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation." *Uncertainty in Artificial Intelligence*. PMLR, 2020.