Adversarial-resilient Machine Learning for the Internet-of-Things

Zhixiong Yang and Waheed U. Bajwa
Department of Electrical and Computer Engineering
Rutgers University–New Brunswick

http://www.inspirelab.us
Mathematically, machine learning is stochastic optimization:

\[ w^* = \arg \min_w \mathbb{E}_z[f(w, z)] \]

- SVM (supervised learning)
  
  \[ z = (x, y) \text{ and } f = \max(0, 1 - y(w^T x + b)) + \lambda \|w\|_2^2 \]

- K-means clustering (unsupervised learning)
  
  \[ z = x \text{ and } f = \sum_{i=1}^{K} \sum_{x \in S_i} d(x, \mu_i) \]

**Challenge:** Distribution of data ‘\( z \)’ is unknown

**Empirical Risk Minimization (ERM)**

- Use training data \( \mathcal{Z} = \{z_n\}_{n=1}^{N} \)
- Minimize the empirical risk:
  
  \[ \hat{w}_N = \arg \min_w \frac{1}{N} \sum_{n=1}^{N} f(w, z_n) \]

- Main result:
  
  \[ \mathbb{E}_z[f(\hat{w}_N, z)] \rightarrow \mathbb{E}_z[f(w^*, z)] \] (Vapnik’92)
What is decentralized machine learning?

Training data is geographically distributed across an interconnected set of devices, nodes, servers, data centers, etc.
But ... the world is a dangerous place for decentralized systems.

How to train decentralized machine learning models in the presence of malicious actors lurking within the network?
A node is said to have Byzantine failure if it *arbitrarily* deviates from its intended behavior within the network.

What can a Byzantine node do?
- Send out “bogus” data
- Collude with other Byzantine nodes
- Start acting normal under scrutiny
- and so much more …

But ... traditional decentralized optimization methods fail under Byzantine failures!
- An important (paraphrased) lesson from Su-Vaidya’15: Distributed empirical risk cannot exactly be minimized in the presence of even a single Byzantine node

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classifier</th>
<th>Optimization</th>
<th>Nodes</th>
<th>Byzantine nodes</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>SVM</td>
<td>DGD</td>
<td>100</td>
<td>1</td>
<td>9.8%</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>SVM</td>
<td>D-ADMM</td>
<td>100</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>
Byzantine-resilient decentralized machine learning

Network model
- A directed graph $G$ comprises $M$ nodes, out of which a maximum of $b$ can be Byzantine
- Each node has access to a local training set of cardinality $N$ (total # of samples = $NM$)

Basic setup
- Nodes cannot share raw data among themselves
- Node $j$ needs to learn a local model $w_j$
- Set of “good” nodes in the network is $J$
- Neighborhood of node $j$ in the network is $\mathcal{N}_j$
- $g_j(w, Z_j) = \frac{1}{N} \sum_{n=1}^{N} f(w, z_{jn})$

Goal: Develop a decentralized optimization method that ensures
- Closeness to $w^* = \arg \min_w \mathbb{E}_z[f(w, z)]$
- “Consensus” among nodes

$w_1 = w_2 = \cdots = w_M$
Algorithmic ingredient #1: Scalar-valued Byzantine-resilient decentralized optimization (Su-Vaidya’15)

Classic Distributed Gradient Descent (DGD) iteration (Nedic-Ozdaglar’09)

\[
w_{j}^{t+1} = \sum_{i \in \mathcal{N}_j \cup j} a_{ij} w_{i}^{t} - \rho^{t} \nabla_{w} g_{j}(w_{j}^{t}, \mathcal{Z}_{j})
\]

Su-Vaidya’15 robustifies DGD in the scalar case by using a “screening” idea similar to that of “trimmed mean” in robust statistics

Screening for Byzantine resilience

1. Sort the received (scalar) iterates at node \( j \)
2. Eliminate the top and the bottom \( b \) iterates
3. Take a mean of the rest of the iterates

\[
w_{j}^{t+1} = \frac{1}{|\mathcal{N}_j - 2b + 1|} \sum_{i \in \mathcal{N}_j^*} w_{i}^{t} - \rho^{t} \nabla_{w} g_{j}(w_{j}^{t}, \mathcal{Z}_{j})
\]

Main result:

\[
\forall j \in J', w_{j}^{t} \xrightarrow{t} \tilde{w}_{dis} = \arg \min_{\tilde{w}} \sum_{j \in J'} \alpha_{j} g_{j}(w, \mathcal{Z}_{j})
\]
Algorithmic ingredient #2: Coordinate descent

A $P$-dimensional optimization problem can be solved by solving $P$ scalar-valued subproblems, with convergence guarantees under various cases (Wright’15).

But ... we cannot solve the scalar-valued subproblems exactly in the presence of Byzantine nodes.
1. Start with the coordinate descent loop
2. In each iteration $r$ of CD, solve for the $k$-th subproblem using Byzantine-resilient scalar-valued DGD

\[ w^{t+1}_j(k) = \sum_{i \in \mathcal{N}_j^*} \frac{w^t_i(k)}{|\mathcal{N}_j^*|} - \rho^t \nabla w(k) g_j(w^t_j(k), \mathcal{Z}_j) \]

\[ \forall j \in J', w^t_j(k) \xrightarrow{t} \tilde{w}_{dis} = \arg \min_w \sum_{j \in J'} \alpha_j(r, k) g_j(w, \mathcal{Z}_j) \]

Does ByRDiE converge to something useful?
Convergence guarantee of ByRDiE

Reduced graph

- A subgraph of a graph is called a reduced graph if it is generated by:
  1. Removing all Byzantine nodes along with their incoming and outgoing edges
  2. Additionally, removing up to \( b \) incoming edges from each non-Byzantine node

Source component

- A source component of a graph is a collection of nodes in which each node in the collection has a directed path to every other node in the graph

Theorem (Convergence of ByRDiE) [Yang-Bajwa’18]

Suppose the candidate models \( w \) belong to a closed, compact set and the function \( f(w, z) \) is strictly convex and Lipschitz continuous. Then, as long as all reduced graphs generated from \( G' \) contain a source component of size at least \( (b + 1) \) and the training data are IID, ByRDiE guarantees with high probability that

\[
\forall j \in J', \mathbb{E}_z[f(w_j, z)] \xrightarrow{N, \bar{r}} \mathbb{E}_z[f(w^*, z)].
\]
Binary classification on MNIST dataset with linear classifier

- Strictly convex loss function, all assumptions fully satisfied
- DGD fails in the presence of Byzantine failures
- ByRDIE has better accuracy than training with only local data
- Trade-off between performance and robustness
Conclusion

- Technologies like IoT require decentralized machine learning
- Malicious actors cannot be ignored in decentralized machine learning

**Byzantine-resilient decentralized learning**
- Guarantees training of machine learning models from distributed data in the presence of Byzantine failures

**Open problems**
- Byzantine-resilient dual / second-order methods
- Non-smooth convex objective functions
- Nonconvex objective functions
- New screening methods

Preprints at [http://www.inspirelab.us](http://www.inspirelab.us)
Experimental setup

Network model
- Erdős–Rényi graph with 800 nodes and parameter $p = 0.5$, $b = 20$
- Each Byzantine node broadcasts a random scalar in each iteration

MNIST8M dataset
- Binary SVM on the most inseparable case of ‘5’ and ‘8’
- Training data: 250 images of each digit at each node
- Test data: 40,000 images

Performance metrics
- Learning: Average accuracy on the test set
- Consensus: 2-norm of pairwise differences

Methods
- Train an SVM at each node using ByRDiE / DGD
- Train an SVM at each node using only local data
- Centralized SVM on all 200,000 training samples (baseline)
Byzantine-resilient distributed gradient descent (BRIDGE)

Gradient descent

- Can be applied to nonconvex loss functions
- No need for dimension synchronization
- Cannot define “large” and “small” for vectors

BRIDGE

- Use dimension-wise trimmed mean as screening
- Update (simultaneously at dimension k):

\[
[w_{j}^{t+1}]_{k} = \frac{1}{|\mathcal{N}_{j} - 2b + 1|} \sum_{i \in \mathcal{N}_{j}^{*}(t, k)} [w_{i}^{t}]_{k} - \rho^{t}[\nabla_{w}g_{j}(w_{j}^{t}, \mathcal{Z}_{j})]_{k}
\]
Convergence guarantee of BRIDGE

Challenge in analysis

- The vectors “break” after screening
- Can no longer be expressed as a convex combination of neighbors

Theorem (Convergence of BRIDGE) [Yang-Bajwa’19]

Suppose the candidate models $w$ belong to a closed, compact set and the function $f(w, z)$ is strongly convex with Lipschitz gradient. Then, as long as all reduced graphs generated from $G$ contain a source component of size at least $(b + 1)$ and the training data are IID, ByRDiE guarantees with high probability that

$$\forall j \in J', w_j \xrightarrow{N,t} w^*.$$
Numerical results