Investigating the robustness of peer-to-peer machine learning

Richard Tomsett
IBM Research, UK

Kevin Chan
ARL, USA

Supriyo Chakraborty
IBM Research, USA

***Abstract*—Future military coalition operations will increasingly rely on machine learning (ML) methods to improve situational understanding. The coalition context presents unique challenges for ML: the tactical environment creates significant computing and communications limitations while also having to deal with an adversarial presence. Further, coalition operations must operate in a distributed manner, while coping with the constraints posed by the operational environment. Envisioned ML deployments in military assets must be resilient to these challenges. Here, we focus on the susceptibility of distributed ML models to be poisoned during training. We present results from investigations into model poisoning attacks on distributed learning systems without a central parameter aggregation node (peer-to-peer learning). *This paper is a summary of research originally published at SPIE, 2019*.**

I. Introduction

We describe experiments into targeted model poisoning attacks [1] in a peer-to-peer machine learning setting. Peer-to-peer (P2P) learning is a promising method for distributing model learning over a network of devices, without requiring coordination from a centralized location. It is therefore well-suited to the coalition setting. However, adversaries will seek to disrupt model learning to impede coalition operations. One method of doing this is by inserting a malicious node into the network of P2P learners, that transmits poisoned updates to the other peers. This attack method has yet to be studied in P2P learning. We present some results illustrating that this kind of attach is both possible, and improves in efficiency when peers share parameters more frequently during learning.

II. Methods

We extend prior work on poisoning distributed learning to the P2P learning setting. The learning dynamics of P2P training methods are different from those that rely on parameter aggregation servers such as federated learning, and so we expect the effects of any particular model poisoning attack to be different as well. In this preliminary study, we chose to focus on targeted model poisoning when learning using gossiping stochastic gradient descent [2], [3].

In our learning setup, each peer has access to an IID subset of size S/K of the full data set, where S is the total number of training data points and K is the number of peers. Every peer starts with a copy of a neural network model with the same architecture and initial parameters, and attempts to minimize its loss on an unseen test data set by training using its local data and sharing parameters with other peers.

Peers are fully connected. We use *random gossip* to share model parameters between peers, specifically the GoSGD algorithm [2], [3]. This algorithm consists of two steps: the gradient update, where each peer k updates its model parameters (weights and biases) w_k using SGD on one randomly chosen mini-batch from its local data; and the mixing step, where each k transmits its parameters to at most one other peer, with probability p_k. If k transmits its parameters, it picks a random peer with uniform probability (i.e. $1/K$) to send them to. It also sends its mixing factor α_k, which is used to adjust the weight given to its parameters when combining them with other peers' parameters. Prior to transmitting, the mixing factor is updated by setting $\alpha_k \leftarrow \alpha_k/2$. Thus α_k shrinks every time k shares its parameters.

Prior to each gradient update, each peer k processes any messages from other peers — if it has received > 1 message, it processes them in the order they arrived. Processing a message from peer j means merging the received parameters w_j from j with k's own parameters w_k according to the mixing factors α_j, α_k using the rule $w_k \leftarrow \alpha_k w_k/(\alpha_j + \alpha_k) + \alpha_j w_j/(\alpha_j + \alpha_k)$. The mixing factor is also updated on processing a message, setting $\alpha_k \leftarrow \alpha_k + \alpha_j$. Thus α_k increases every time k receives parameters. All mixing factors are initialized as $\alpha_k \leftarrow 1/K$.

We adapt the targeted model poisoning attack described by Bhagoji et al. [1] to this peer-to-peer learning setup. In this attack, the single malicious peer m attempts to trick the benign peers to learn to classify a set of data points $\{\mathbf{x_i}\}_{i=1}^{r}$ with specific, incorrect labels $\{\tau_i\}_{i=1}^{r}$ instead of their true labels $\{y_i\}_{i=1}^{r}$. This set is available to the malicious peer, but not the benign peers, during training. As in Bhagoji et al. [1], we choose to learn to mis-classify a single example, i.e. $r = 1$.

We use Bhagoji et al.'s alternating minimization attack strategy [1]: the malicious peer m updates its own model weights by first minimizing an adversarial objective (minimizing the loss on $\{\mathbf{x_i}\}_{i=1}^{r}$) and updating its parameters, then minimizing the global loss using a mini-batch from its allocated training data, and updating its parameters.

All our experiments were run with 9 benign peers and 1 malicious peer (10 peers in total). We used the Fashion-MNIST dataset [4], split into a training set of 60,000 and a test set of 10,000 images, and categorized into 10 classes. With our 10 peers, this means that each peer has access to a random subset of 6,000 images from the training set, and is tested at each step on the full 10,000–image test-set. For our