Thesis ideas

Thomas Werthenbach

January 27, 2023

1 FoolsGold in decentralized setting

Currently favorite idea. FoolsGold is a popular algorithm for mitigating Sybil attacks in federated learning environments. It detects Sybils by calculating the cosine similarity of the updates of a model, as all Sybils are likely to send similar models. However, it assumes that the aggregator has access to all models, which is the case for federated learning, but not for decentralized learning. In this proposal, we improve FoolsGold to work in a decentralized learning environment. We introduce a probabilistic gossiping mechanism, which sends a randomly selected model from its set of collected models to a peer. By doing so, peers build a database of models to perform cosine similarity-based aggregation with. We will undertake the following actions during the course of this research:

- We reproduce the results from FoolsGold using federated learning.
- We introduce *plain* FoolsGold to decentralized learning and compare the results.
- We describe a threat model as well as effective Sybil strategies against FoolsGold in a decentralized setting.
- We add our smart probabilistic gossiping mechanism and re-perform the experiments and find the results of our solution.

Note:

- We assume that nodes can not be fully connected, as this would result in the same result as federated learning.
- The Sybil attack will work best if there is a single attack edge to every node (assuming plain FoolsGold). By doing so, every node will be affected by the label-flipping or backdoor attack, but there is almost no cosine similarity between any update. Given the prior statement, a Sybil attack can be considered overkill, if the maximum degree of a node is unbounded, because it has the same effect as a single malicious node performing the label-flipping and backdoor attack. This is again similar to federated learning. Therefore, we assume that there exists a maximum degree on

each node, such that there exists and incentive for creating Sybils. This is similar to real-life situations in peer-to-peer networks, even though such limit does not exist in practice.

- If FoolsGold is applied directly, we already bound the maximum amount of *useful* attack edges to the same number as the amount of nodes in the network.
- Given that our novel gossip mechanism ensures that models travel at most s edges and the maximum degree of a node is d, then the amount of *useful* attack edges is equal to:

  $$u = \frac{n}{\frac{1-d^{(s+1)}}{1-d}}$$

  For example, if $s = 1$ and $d = 3$, then every attack edge has to be at least $2 * s + 1 = 3$ *hops* apart and every attack edge will spread the malicious model to $\frac{1-d^{(s+1)}}{1-d} = 4$ nodes.

  However, this comes with a trade-off, as the amount of models any given node will to calculate the cosine similarity between grows with $\frac{1-d^{(s+1)}}{1-d}$ as well.
- With Sybil attacks, adversaries typically aim to exploit a certain network property through creating multiple entity. We say that Sybil attacks are of no use when the maximum of theoretically *useful* attack edges is smaller than the degree, such that $u < d$. Note that this implies that nodes will need to keep track of $\frac{n}{d}$ nodes.
- Note: when the amount of attack edges exceeds u, there must be at least one node in the network detecting the Sybil attack, manifesting sub-optimal results.

However, regarding the analysis performed above, such strict bounds on a Sybil attack are in practice not required, as all honest nodes keep training the model after aggregation, having a fading effect on the label-flipping attack or the back-door attack as the distance increases from the sybil node.

2 Improve MeritRank

There are several possible points of improvement on MeritRank:

- It is currently centralized, does not work in decentralized settings. May require a smart way of distributing edge weights.
- We can use random floods rather than random walks for increased efficiency of transitivity decay
- For connectivity decay, we can use the netflow algorithm after performing the random flood on the accumulated weights of the edges (or should we do it on the default reputation weights?)

- possible new type of decay (?): the amount of paths between yourself and some other node, increases the trustworthiness of said node.

3 Framework for evaluating decentralized learning mechanisms and plug-in reputation mechanisms

We invent a novel methodology which can be used to structurally combine reputation mechanisms with decentralized learning approaches. We then use this methodology to study the performance change caused by the introduction of reputation by combining many reputation mechanisms with many decentralized learning approaches. Note that this will result in a large amount of experiments. An example of such methodology may be:

1. Upon receiving our neighbours' models, we calculate a score for every received model, e.g. the accuracy or a similarity function. This score will in turn affect the edge weight between the two nodes in the trust graph.
2. The selected reputation mechanism will use these scores to calculate the neighbours' reputations.
3. Given that all our aggregation methods work with some form of averaging, we linearly shift the received weights towards our own model depending on the reputation: $m'_j = (1 - r) \cdot m_i + r \cdot m_j$, where m_i is our own model, m_j is our neighbour's model and r is the reputation score provided by the selected reputation mechanism. By doing so, we simulate weights to the neighbours' models without modifying the internals of the existing decentralized learning method.
4. The existing decentralized learning methods now receive the linearly modified models and perform aggregation.

4 Continue with Bristle improvement

we continue with the current plan of integrating parts of Bristle and MeritRank and perform very thorough evaluation. In this case, we will need to clarify why we use transfer learning. The most obvious answers may be:

1. Decentralized learning may run on any device, including low-resource edge devices.
2. All peers aim to train a personalized model, given a pre-trained global model. However, with this reasoning one may wonder why we would perform the learning distributed at all.