Detect insider attacks using CNN in Decentralized Optimization

45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2020)

Gangqiang Li

Shenzhen University

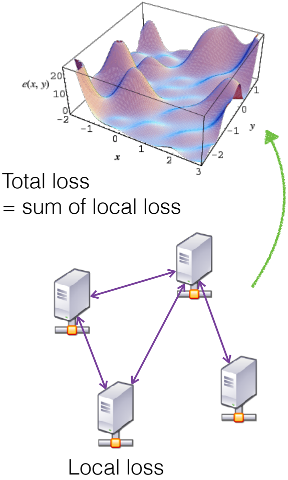
Joint work with:
Xiaoxiao Wu (SZU), Shengli Zhang (SZU), Qiang Li (UESTC)

Big Data Network Challenges

Basic idea: large-scale, massive number of users, computing everywhere...

Context of the Talk: Gossip-based Algorithms

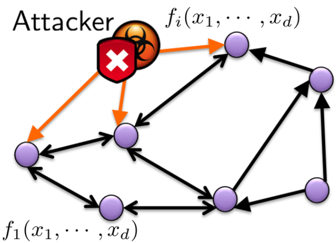
- Data are often measured distributively over large networks
- Gossip-based algorithms: solve multi-agent coordination and optimization problems in a decentralized manner
  - synchronous
  - asynchronous
- Key features:
  - ✓ built-in fault tolerance to intermittent computation/communication.
  - ✓ self reorganization to automatic failure correction.
- Vast literature (see e.g. [NO09, BPC+11, PC06, MBG10, BGPS06, DKM+10, RN04, JXM14])



Problem Statement

The general distributed multi-agent optimization problem has the following syntax:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{x} \in X \ . \tag{1}$$

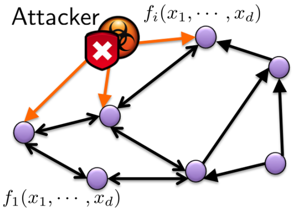


- $f_i : \mathbb{R}^d \to \mathbb{R}$ is a differentiable function over X.
- Each agent i can compute a local estimate $x_i(t) = [\boldsymbol{x}(t)]_i$ of the optimal solution to the problem (at time t), which we refer to as the agent *state*.
- The constraints can be handled either through the use of the Lagrangian or the use of a projection on the constraint set.

The Graph Model

Consider a network of agents described by a connected possibly undirected graph:

- At time t, the network is described by $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E})$. Where $\mathcal{V} = [n] = \{1, ..., n\}$, and $\mathcal{E}(t) \subseteq [n] \times [n]$ is the edge set.
- The weighted adjacency matrix $\boldsymbol{A}(t) \in \mathbb{R}^{n\times n}$ where $[\boldsymbol{A}(t)]_{ij} := A_{ij}(t) = 0$ if $(j, i) \notin \mathcal{E}(t)$. The time varying graph is defined as $\mathcal{G}(t) := (\mathcal{V}, \mathcal{E}(t))$ with $\mathcal{E} := \cup_{t=1}^{\infty} \mathcal{E}(t)$. We have the following assumption:



Assumption 1

There exists a scalar $\eta \in (0, 1)$ such that for all $t \geq 1$ and $i = 1, \cdots, n$:

- $A_{ij}(t) \geq \eta$ if $(i, j) \in \mathcal{E}(t)$, $\boldsymbol{A}(t)\mathbf{1} = \mathbf{1}$, $\boldsymbol{A}^\top(t)\mathbf{1} = \mathbf{1}$;
- The graph $(\mathcal{V}, \cup_{\ell=1}^{B_0} \mathcal{E}(t + \ell))$ is connected for $B_0 < \infty$.

The Protocol for Trustworthy Agents

- At the t^{th} recursion, trustworthy agents follow a typical gossip-based distributed projected gradient (DPG) algorithm:

$$\begin{aligned} \boldsymbol{x}_i(t+1) &= P_X\left(\bar{\boldsymbol{x}}_i(t) - \gamma(t)\nabla f_i\left(\bar{\boldsymbol{x}}_i(t)\right)\right) . \\ \bar{\boldsymbol{x}}_i(t) &= \textstyle\sum_{j=1}^{n} A_{ij}(t)\boldsymbol{x}_j(t) . \end{aligned} \tag{2}$$

  for $t \geq 1$, where $\gamma(t) > 0$ is a diminishing step size. P_X denotes the Euclidean projection onto the set X and $\nabla f_i\left(\bar{\boldsymbol{x}}_i(t)\right)$ is a gradient of the agent i private function $f_i(\boldsymbol{x})$ at $\bar{\boldsymbol{x}}_i(t)$. For convex problems, it was shown in [WWS$^+$18, RNV10] that the DPG method converges to an optimal solution of (1):

Fact 1

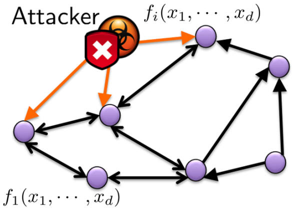
Under Assumption 1. If $\|\nabla f_i(\boldsymbol{x})\| \leq C_1$ for some C_1 and for all $\boldsymbol{x} \in X$, and the step size satisfies $\sum_{t=1}^{\infty} \gamma(t) = \infty$, $\sum_{t=1}^{\infty} \gamma^2(t) < \infty$, then for all $i, j \in \mathcal{V}$ we have

$$\lim_{t\to\infty} f(\boldsymbol{x}_i(t)) = f^\star \quad \text{and} \quad \lim_{t\to\infty} \|\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)\| = 0 . \tag{3}$$

Without attacker, each node will converge and consensus to the global optimum.

Data Injection Attack from Insiders

- Let $\mathcal{V} = \mathcal{V}_t \cup \mathcal{V}_m, \; n = |\mathcal{V}_t| + |\mathcal{V}_m|$.
- **Coordinated Attack**:
  Disguised consensus injection:
  $$\boldsymbol{x}_j(t) = \underbrace{\boldsymbol{\alpha}}_{\text{attack target value}} + \underbrace{\boldsymbol{r}_j(t)}_{\text{noise to disguise}} \qquad \forall\, j \in \mathcal{V}_m,$$
  where $\lim_{t\to\infty} \|\boldsymbol{z}_j(t)\| \to 0$.



- (Assumption 2). There exists $B_1, B_2 < \infty$ such that for all $t \geq 1$, (a) the composite sub-graph $(\mathcal{V}_t, \cup_{\ell=t+1}^{t+B_1} \mathcal{E}(\mathcal{V}_t; \ell))$ is connected;(b) there exists a pair $i \in \mathcal{V}_t$, $j \in \mathcal{V}_m$ with $(i,j) \in \mathcal{E}(t) \cup \ldots \cup \mathcal{E}(t + B_2 - 1)$.

Fact 2

Under Assumptions 1 and 2. If $\|\nabla f_i(\boldsymbol{x})\| \leq C_2$ for some C_2 and for all $\boldsymbol{x} \in X$, and $\gamma(t) \to 0$, we have:
$$\lim_{t\to\infty} \max_{i\in\mathcal{V}_t} \|\boldsymbol{x}_i(t) - \boldsymbol{\alpha}\| = 0 \,. \tag{4}$$

Attacks succeed: agents will converge and consensus to attackers' desirable value.

Attacker Detection and Localization

We define $\mathcal{H}_0 : \mathcal{V}_m = \emptyset$ and $\mathcal{H}_1 : \mathcal{V}_m \neq \emptyset$ as two scenarios in the network, namely 'no-attacker' and 'attacker is present'. As for the DPG algorithm, we define the following events to execute these two neighborhood tasks:

$$\mathcal{H}_0^i : \mathcal{N}_i \cap \mathcal{V}_m = \emptyset; \quad \mathcal{H}_1^i : \mathcal{N}_i \cap \mathcal{V}_m \neq \emptyset. \tag{5}$$

$$\mathcal{H}_0^{ij} : j \notin \mathcal{V}_m; \quad \mathcal{H}_1^{ij} : j \in \mathcal{V}_m. \tag{6}$$

$\mathcal{H}_0^i$ and $\mathcal{H}_1^i$ as two events of the trustworthy agent i for the neighborhood detection task, i.e., events $\mathcal{H}_0^{ij}$and $\mathcal{H}_1^{ij}$ for neighborhood localization task.

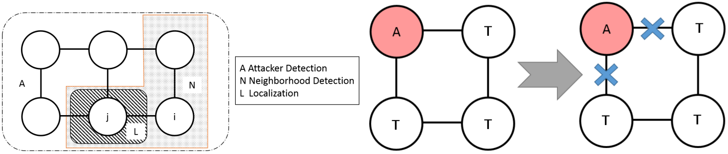


Figure: (Left) Different tasks involved in the attack detection scheme. (Right) Each "trustworthy" agent T performs detection and localization independently, therefore isolating an attacker A from the network.

Neighborhood detection with spatial data

- $\mathcal{H}_0^i$ –there is no attacker in $\mathcal{N}_i$, i.e., $\mathcal{V}_m \cap \mathcal{N}_i = \emptyset$;
- $\mathcal{H}_1^i$ –there exists an attacker in $\mathcal{N}_i$, i.e., $\mathcal{V}_m \cap \mathcal{N}_i \neq \emptyset$.

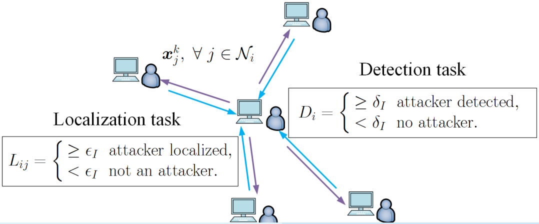
After running the DPG for K different instances, the detection task corresponds to —

Neighborhood Detection Task:

$$\varphi_{ij}^k := \sum_{t=0}^{T} \left(\boldsymbol{x}_j^k(t) - \overline{\boldsymbol{x}}_i^k(t) \right), \tag{7}$$

$$D_i := \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \left(\frac{1}{K} \sum_{k=1}^{K} \frac{\mathbf{1}^\top \varphi_{ij}^k}{d} \right)^2 \underset{\mathcal{H}_1^i}{\overset{\mathcal{H}_0^i}{\lessgtr}} \delta_I. \tag{8}$$

where $\overline{\boldsymbol{x}}_i^k(t) = (1/|\mathcal{N}_i|) \sum_{j \in \mathcal{N}_i} \boldsymbol{x}_j^k(t)$. δ_I is a pre-designed threshold, and d is the state dimension of agents.



Neighborhood localization with spatial data

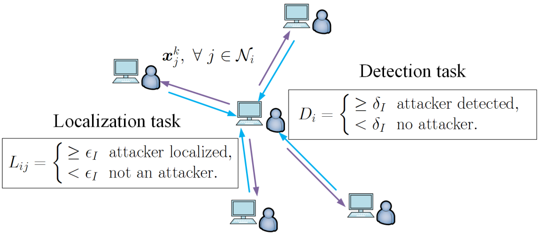
- $\mathcal{H}_0^{ij}$ –agent j is not an attacker, i.e., $j \notin \mathcal{V}_m$;
- $\mathcal{H}_1^{ij}$ –agent j is an attacker, i.e., $j \in \mathcal{V}_m$.

For the localization task, we compare the state of agent j and agent i to check if the neighbor agent is an attacker. We propose checking the metric for localization:

Neighborhood Localization Task:

$$\tilde{\boldsymbol{\varphi}}_{ij}^k := \sum_{t=0}^{T} \left(\boldsymbol{x}_j^k(t) - \boldsymbol{x}_i^k(t) \right) - \boldsymbol{\varphi}_{ii}^k, \tag{9}$$

$$L_{ij} := \left(\frac{1}{K} \sum_{k=1}^{K} \frac{\mathbf{1}^\top \tilde{\boldsymbol{\varphi}}_{ij}^k}{d} \right)^2 \underset{\mathcal{H}_1^{ij}}{\overset{\mathcal{H}_0^{ij}}{\lessgtr}} \epsilon_I. \tag{10}$$



Tackle this Problem using Convolutional Neural Networks

- D_i and L_{ij} are roughly linear functions which fuse the state vector obtained by node i into a scalar score for classification.
- We propose to apply a CNN system to fuse $\{\boldsymbol{x}_i\}_{j \in \mathcal{N}_i}$ for the detection and localization task.
- We consider the detection and localization process as a classification problem.
- The CNN can be trained in an offline manner by using data collected from the neighbors of a trustworthy agent i. We train the CNN in an offline manner and the same CNN can be deployed on each trustworthy agent.

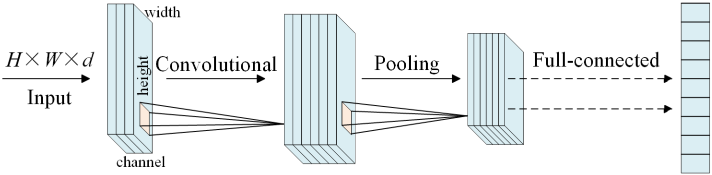


Figure: The structure of CNN.

Detection and localization via CNN

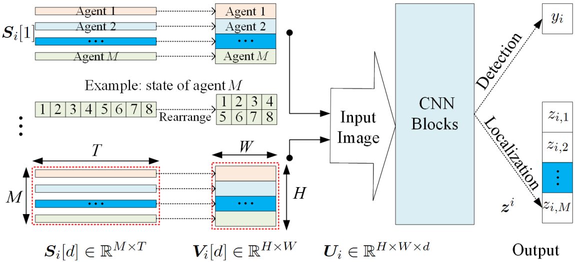
For ease of exposition, we assume $K = 1$ and all the agents have M neighbors. Let T be the number of iterations that the DPG algorithm runs before convergence.

- $\widetilde{x}_i^\ell(t) \in \mathbb{R}$ for $t = 1, \dots, T$ and $\ell = 1, \dots, d$ be the ℓth dimension state vector of agent i at the tth iteration.
- By putting $\{\widetilde{x}_i^\ell(t)\}_{t,\ell}$ together, we get a state matrix $\widetilde{\boldsymbol{X}}_i \in \mathbb{R}^{T\times d}$ of the ith agent,
$$\widetilde{\boldsymbol{X}}_i = \begin{bmatrix} \widetilde{x}_i^1(1), & \dots, & \widetilde{x}_i^d(1) \\ \vdots & \vdots & \vdots \\ \widetilde{x}_i^1(T), & \dots, & \widetilde{x}_i^d(T) \end{bmatrix} \in \mathbb{R}^{T\times d}.$$
Denote $\widetilde{\boldsymbol{x}}_i[\ell] \in \mathbb{R}^{T\times 1}, \ell = 1, \dots, d$ as the ℓth column vector of $\widetilde{\boldsymbol{X}}_i$.
- We then construct the neighborhood state matrix associated with the ℓth dimension of the ith agent,
$$\boldsymbol{S}_i[\ell] = \left[\widetilde{\boldsymbol{x}}_{i(1)}[\ell], \dots, \widetilde{\boldsymbol{x}}_{i(M)}[\ell]\right]^T \in \mathbb{R}^{M\times T}$$
- Next we reshape $\boldsymbol{S}_i[\ell]$ into an image $\boldsymbol{V}_i[\ell] \in \mathbb{R}^{H\times W}$ with height H and width W. We virtually obtain an image with d channels, i.e.,
$$\boldsymbol{U}_i = \{\boldsymbol{V}_i[1], \dots, \boldsymbol{V}_i[d]\} \in \mathbb{R}^{H\times W\times d}, \ i = 1, \dots, n.$$

Detection and localization via CNN

We employ CNNs to detect and localize the malicious agents, as illustrated in the following figure.

- The detection is made according to the following rule in (CNND), wherein $y_i \in \mathbb{R}$.
- The localization is done by using the following rule in (CNNL). We define $\boldsymbol{z}_i = [z_{i,1}, \cdots, z_{i,M}] \in \mathbb{R}^M$ as the output of CNN for localization.
- $\delta_{II} \in [0, 1]$ and $\epsilon_{II} \in [0, 1]$ are some prescribed threshold.



$$\text{(CNND)} \quad y_i \underset{\mathcal{H}_0^i}{\overset{\mathcal{H}_1^i}{\gtrless}} \delta_{II}; \qquad \text{(CNNL)} \quad z_{i,j} \underset{\mathcal{H}_0^{ij}}{\overset{\mathcal{H}_1^{ij}}{\gtrless}} \epsilon_{II}$$

Simulation Settings

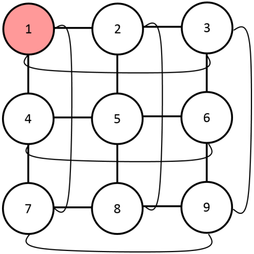


Figure: The Manhattan network topology is considered. We select only node 1 as an attacker, while all other nodes are trustworthy agents.

We take an example of the least square problem; i.e.,

$$f^k(\boldsymbol{x}) = \sum_{i=1}^n f_i^k(\boldsymbol{x}) = \sum_{i=1}^n \left|(\boldsymbol{a}_i^k)^\top \boldsymbol{x}^k - b_i^k\right|^2, k = 1, ..., K.$$

Herein, f_i^k can be seen as a utility function for instance k, $b_i^k = (\boldsymbol{a}_i^k)^T (\boldsymbol{x}^\star)^k$.

Simulation Settings

- We set $\mathbb{E}\left[\boldsymbol{A}(t)\right] = \boldsymbol{I} - \frac{1}{2n}\boldsymbol{\Sigma} + \frac{\boldsymbol{P}+\boldsymbol{P}^\top}{2n}$ with $[\boldsymbol{\Sigma}]_{ii} = \boldsymbol{\Sigma}_{j=1}^{n}(P_{ij} + P_{ji})$, where $\boldsymbol{\Sigma}$ is a diagonal matrix, $P_{ij} = \frac{1}{|\mathcal{N}_i|}$ is the probability between agents i and j at time t.
- A trustworthy agent $i \in \mathcal{V}_t$ is initial by $\boldsymbol{x}_i^k$, $\boldsymbol{x}^k(0) \sim \mathcal{U}[0,1]^d$.
- An attacker agent $j \in \mathcal{V}_m$ follow a update rule (4). We set $\boldsymbol{\alpha}^k \sim \mathcal{U}[-0.5, 0.5]^d$ and $\boldsymbol{r}_j^k(t) \sim \mathcal{U}[-\hat{\lambda}^t, \hat{\lambda}^t]$, where λ is the second largest eigenvalue of $\mathbb{E}[\boldsymbol{A}(t)]$.
- $b_i^k = (\boldsymbol{a}_i^k)^T(\boldsymbol{x}^\star)^k$, where $\boldsymbol{a}_i^k \sim \mathcal{U}[0.5, 2.5]^d$, $(\boldsymbol{x}^\star)^k \sim \mathcal{U}[0,1]^d$.
- $k \in [K] = \{1, 2, \cdots, K\}$, $dim \in [d] = \{1, 2, \cdots, d\}$.
  Among them, K is the numbers of instances and we take $d = 3$; *dim* means that among total d dimensions, how many of which is observed to calculate the metric.

ROC Curves for SD and CNN

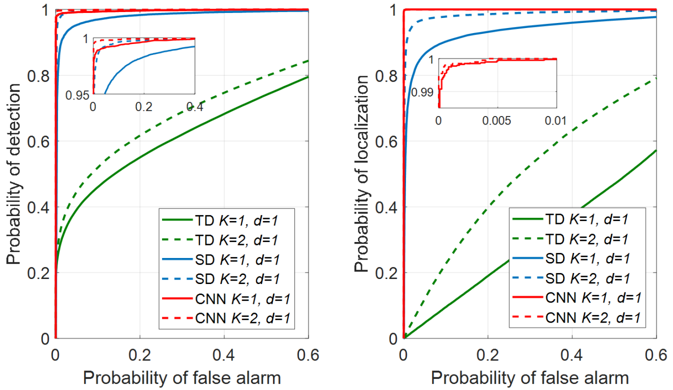


Figure: ROCs performance of TD, SD and CNN methods with $d = 1$.Left: Detection performance at the neighboring nodes of the attacker; Right: Localization performance at the neighboring nodes of the attacker.

Note: This result implies that transient states do provide us more information to identify the malicious agent.

ROC Curves for SD and CNN

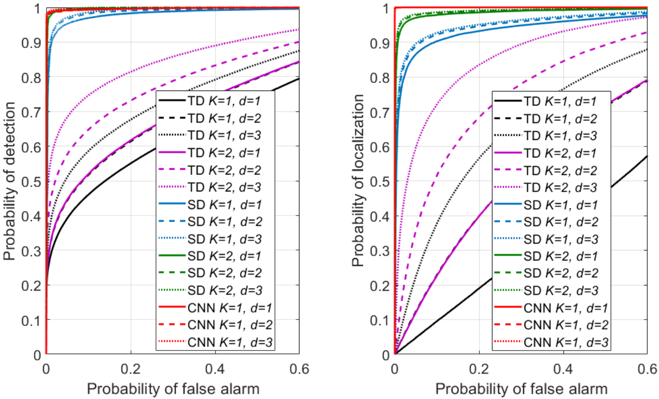


Figure: ROCs of CNN and SD in different dimensions: where d is the state dimension of neighbor agents. Left: Detection performance at the neighboring nodes of the attacker; Right: Localization performance at the neighboring nodes of the attacker.

Note: Increasing K and *dim* can improve the performance.

Summary

- In this work, we proposed two defense strategies for the gossip-based DPG optimization algorithm.
- The first one is a score-based method employing the transient state information from the agents. It can outperform our previous score-based method which only considers initial state and steady state information.
- We further adopt the CNN to secure the DPG algorithm. CNN can automatically learn effective features from original state information without complex calculations.
- We numerically verify the efficiency of the detector for the optimization algorithm based on the least square functions.

References

[BGPS06] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and D. Shah.
Randomized gossip algorithms.
IEEE Trans. on Information Theory, 52(6):2508–2530, June 2006.

[BPC+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein.
Distributed optimization and statistical learning via the alternating direction method of multipliers.
Found. Trends Mach. Learn., 3(1):1–122, January 2011.

[DKM+10] A.G. Dimakis, S. Kar, J.M.F. Moura, M.G. Rabbat, and A. Scaglione.
Gossip algorithms for distributed signal processing.
Proceedings of the IEEE, 98(11):1847–1864, Nov 2010.

[JXM14] Dusan Jakovetic, Joao Xavier, and Jose M. F. Moura.
Fast distributed gradient methods.
IEEE Trans. Autom. Control, 59(5):1131–1146, May 2014.

[MBG10] G. Mateos, J. A. Bazerque, and G. B. Giannakis.
Distributed sparse linear regression.
IEEE Transactions on Signal Processing, 58(10):5262–5276, Oct 2010.

[NO09] A. Nedić and A. Ozdaglar.
Distributed subgradient methods for multi-agent optimization.
IEEE Transactions on Automatic Control, 54(1):48–61, 2009.

[PC06] D. P. Palomar and Mung Chiang.
A tutorial on decomposition methods for network utility maximization.
IEEE Journal on Selected Areas in Communications, 24(8):1439–1451, Aug 2006.

[RN04] Michael Rabbat and Robert Nowak.
Distributed optimization in sensor networks.
In Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, IPSN '04, pages 20–27, New York, NY, USA, 2004. ACM.

[RNV10] S Sundhar Ram, Angelia Nedić, and Venugopal V Veeravalli.
Distributed stochastic subgradient projection algorithms for convex optimization.
Journal of optimization theory and applications, 147(3):516–545, 2010.

[WWS+18] Sissi Xiaoxiao Wu, Hoi-To Wai, Anna Scaglione, Angelia Nedić, and Amir Leshem.
Data injection attack on decentralized optimization.
In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3644–3648. IEEE, 2018.

Thank You

&&

Question Welcomed!