Non-linear Constrained Optimization Problem using Byzantine Distributed Optimization Algorithm

FT-report group 15

Ming-Yu Chung, Po-Yu Chen

December 2022

Outline

- **Introduction of Byzantine distributed optimization problem and our goal**
- **Linearization method for constrained optimization problem**
- **Proposed algorithm**

Introduction of Byzantine distributed optimization problem

Introduction of Byzantine distributed optimization problem

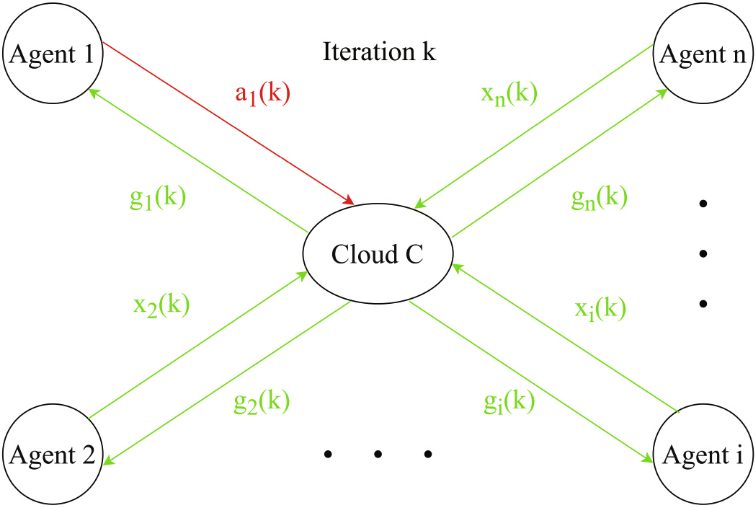
Byzantine distributed optimization problem can be discribed as following setting:

1. **Agent**: There are m agents. For i-th agent, he holds a cost function $C_i : \mathbb{R}^n \to \mathbb{R}$ and then send some information of $C_i(x)$ to the central server. We say the i-th agent is a Byzantine faulty agent if he send an incorrect information to central server. Otherwise, we call it is an honest agent.
2. **Central server:** Central server aims to utilize the information from each agent to solve following optimization problem:

$$\arg\min_{x \in B} \sum_{i \in \mathcal{H}} C_i(x), \tag{1}$$

where $B \subset \mathbb{R}^n$ is some compact set and $\mathcal{H}$ is the index set of honest agents.

Introduction of Byzantine distributed optimization problem



[XLH22]

Byzantine distributed optimization algorithm

[SV16, GV19a, GV19b, LGV21] have shown that the exact fault tolerance problem (1) cannot be solved without some specific condition ($2f$-redundancy). Hence, in this project, we always assume $2f$-redundancy is satisfied.

Byzantine distributed optimization algorithm (BDOA) is the algorithm to solve (1). For example:

- **Gradient-Filter-based Distributed Gradient Descent:** mitigates the detrimental impact of incorrect gradients

  E.g., comparative gradient elimination (CGE) [GLV20], coordinate-wise trimmed mean (CWTM) [SV16], geometric median-of-means (GMoM) [CSX17]

Our goal

In this project, we consider the following constrained optimization problem:

$$(P0) \begin{cases} \arg\min_{x \in B} & C_0(x) \\ \text{subject to} & C_i(x) \leq 0, \ \text{where } i \in [m]. \end{cases}$$

We want to utilize the technique of (BDOA) to enhance the performance of some existing constrained optimization algorithm on (P0). To be more specific, we want to improve linearization method [WP12].

Linearization method

At first, we approximate $(P0)$ as follows:

$$(P1)\begin{cases}\arg\min_p & C_0(x)+\langle C_0'(x),p\rangle+\|p\|^2\\ \text{subject to} & C_i(x)+\langle C_i'(x),p\rangle\le 0,\ \text{ where } i\in[m].\end{cases}$$

In [WP12], we can solve $(P1)$ iteratively to obtain the solution of $(P0)$. To more specific, we consider following algorithm:

1. Obtain direction p^k by solving $(P1)$ at x^k.
2. Approximate some suitable step size coefficient $\alpha^k > 0$.
3. Update $x^{k+1} \leftarrow x^k + \alpha^k \cdot p^k$.
4. $k \leftarrow k+1$ and back to step 1.

(Remark. If the linearization is terrible for some $C_i(x)$, the α^k will be very small and hence make the algorithm inefficient.)

Proposed Method

Proposed method

In this project, we aim develop an optimization algorithm to improve the performance of linearization method on (P0).

Here, we consider "linearization is terrible" as a fault (i.e. α is too small) and apply **(BDOA)** on (P1). We can obtain the optimal point of

$$(PB)\begin{cases} \arg\min_{x\in B} & C_0(x) \\ \text{subject to} & C_i(x) \leq 0, \ \text{where } i \in \mathcal{H}. \end{cases}$$

Proposed method

For the Byzantine faulty agents,

$$(PC)\begin{cases}\arg\min_{x\in B} & C_0(x)\\ \text{subject to} & C_i(x)\le 0,\ \text{ where } i\in\mathcal{B},\end{cases}$$

we utilize **central path method** and **proximal point method** [BV04].

Finally, in order to solve the original problem (P0). We utilize the **proximal gradient method** [BV04] to hybrid (PB) and (PC). In short, we consider following algorithm:

1. Update x^k with **(BDOA)** and identify the byzantine faulty agents on (P0).
2. Update x^{k+1} with **central path method** and **proximal point method** on (PC).
3. $k \leftarrow k+2$ and back to step 1.

(Some similar works : [LSH⁺10, XLH22])

Experiment

Experiment

The experiment has been conducted with three settings as Table 1 shown. The number of iterations is set as 20. We introduce a loss function to evaluate the optimization process:

$$\mathrm{Loss}(X) = f_0(X) + \sum_{i \in [m]} k_0 \cdot \mathrm{ReLU}(C_i(X))$$

where f_0 is the objective function, $[m]$ is the index set of constraints, and k_0 - the "penalty" coefficient - is a hyper-parameter greater than zero. Here we set k_0 as 10000. Then we plot the loss values during the optimization process, as shown in Fig. 1, Fig. 2 and Fig. 3. Note that the values are the results of taken the logarithm.

Table: Experiment settings

Setting	Objective Function	Constraints	Initial Point (X_0)
1	$f(x,y) = y^2 - 3$	$10^x - 10^{-1} \leqslant 0$ $10^y - 10^{-3} \leqslant 0$	$(3.0, -2.0)$ $(-4.0, 9.0)$
2	$f(x,y) = y^2 - 3$	$10^x - 10^{-1} \leqslant 0$ $10^y - 10^{-3} \leqslant 0$ $x^2 + y^2 - 20 \leqslant 0$	$(3.0, -2.0)$ $(3.0, -20.0)$
3	$f(x,y) = x^2 + y^2 + 10$	$x^6 + y^6 - 20 \leqslant 0$ $x^2 + y^2 - 10 \leqslant 0$ $y^8 - 20 \leqslant 0$ $x + y - 10 \leqslant 0$ $3x - 2y + 1 \leqslant 0$	$(30.0, -8.0)$ $(-100.0, 5.0)$

Experiment

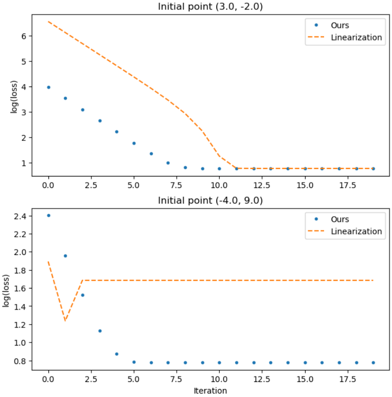


Figure: Experiment result of setting 1.

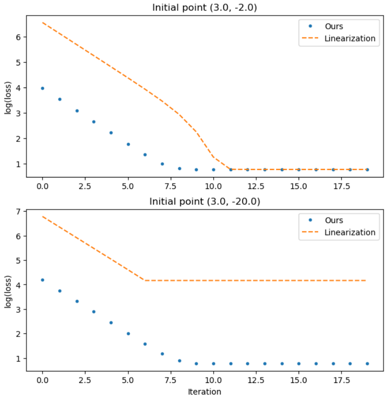


Figure: Experiment result of setting 2.

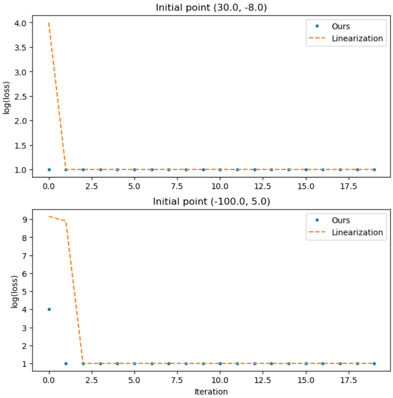


Figure: Experiment result of setting 3.

Conclusion

Conclusion

1. We proposed an algorithm based on BDOA and linearization method for non-linear constrained optimization problems.
2. The experimental results show that our algorithm is experimentally better than linearization method.

Future works

Future works

1. Provide theoretical proof to show our algorithm is better than linearization method, when the optimization problem which we consider is sufficiently unsuitable for linear approximation.
2. Provide an estimation of hyper-parameters.
3. Do more experiments to show the feasibility of our algorithm.

Thanks

Thanks for listening

References I

- Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.
- Yudong Chen, Lili Su, and Jiaming Xu, *Distributed statistical machine learning in adversarial settings: Byzantine gradient descent*, Proceedings of the ACM on Measurement and Analysis of Computing Systems **1** (2017), no. 2, 1–25.
- Nirupam Gupta, Shuo Liu, and Nitin H Vaidya, *Byzantine fault-tolerant distributed machine learning using stochastic gradient descent (sgd) and norm-based comparative gradient elimination (cge)*, arXiv preprint arXiv:2008.04699 (2020).
- Nirupam Gupta and Nitin H Vaidya, *Byzantine fault tolerant distributed linear regression*, arXiv preprint arXiv:1903.08752 (2019).

———, *Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression*, 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2019, pp. 415–420.

Shuo Liu, Nirupam Gupta, and Nitin H Vaidya, *Approximate byzantine fault-tolerance in distributed optimization*, Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing, 2021, pp. 379–389.

Wen Tao Li, Xiao Wei Shi, Yong Qiang Hei, Shu Fang Liu, and Jiang Zhu, *A hybrid optimization algorithm and its application for conformal array pattern synthesis*, IEEE Transactions on antennas and propagation **58** (2010), no. 10, 3401–3406.

Lili Su and Nitin H Vaidya, *Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms*, Proceedings of the 2016 ACM symposium on principles of distributed computing, 2016, pp. 425–434.

S.S. Wilson and B.N. Pshenichnyj, *The linearization method for constrained optimization*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2012.

Chentao Xu, Qingshan Liu, and Tingwen Huang, *Resilient penalty function method for distributed constrained optimization under byzantine attack*, Information Sciences **596** (2022), 362–379.