Decentralized Gradient Descent: how network structure affects convergence

Jason Yang, Jun Wan, Hanshen Xiao

May 2021

Abstract

We investigate decentralized gradient descent among a network of nodes where an adversary has corrupted certain nodes. We focus on the case where the utility functions of all nodes are 1-dimensional quadratics, and where each corrupted node is connected to all honest nodes.

1 Introduction

Decentralized gradient descent (DGD) is a variant of gradient descent where multiple users have different cost functions and want to collectively minimize some combined version of all their cost functions. In DGD, the users never directly tell each other their own cost functions, but can tell less revealing information to each other. Furthermore, the communication network of the users is not necessarily the complete graph, so it may be impossible for certain pairs of users to directly communicate with each other without relaying information between intermediate users.

DGD has applications in machine learning problems that also require a level of confidentiality. For example, several hospitals could each hold some confidential medical data and want to collectively create a machine learning model that takes all of their data into account, without any of the hospitals directly sharing their data to each other or to a third party. If all hospitals agree to use the same kind of model for machine learning (e.g. if all hospitals use a neural network with the same number of input nodes, output nodes, number of hidden layers, and number of nodes in each hidden layer), then each hospital can use its data to create a cost function, and the objective is now for all hospitals to minimize the sum of all their cost functions.

Previous research has been done on DGD [1], including experimental results and asymptotic bounds on the error of the agents' final values from the optimum, i.e. the argument minimum of the objective function. We are most interested in seeing what happens when we add *corrupt* users to the graph, users who can send arbitrary information to try and throw off the DGD. We are also interested in how the density of the communication network affects DGD.

2 Model

For this paper, each agent's cost function $f_i(x)$ is of the form $(a_i x - v_i)^2$ for $x \in \mathbb{R}$ and uniformly random parameters $a_i \in [0, 1)$, $v_i \in [-100, 100)$. The function we want to minimize is denoted as $tf(x) = \frac{1}{N} \sum_i f_i(x)$. Note that all f_i and tf are upward-facing parabolic functions. Figure 1 shows what these functions $f_i(x)$ look like when graphed on the xy-plane.

The number of honest agents, N, is fixed at 50 throughout this paper. The agents are connected in a graph, where edge $(i \to j) \in E$ means that there exists a communication channel from agent i to agent j. Every honest agent also has a loop that goes to itself. Initially, all graphs we deal with will be generated as follows: a probability P is selected; for every unordered pair $\{i, j\}, i \neq j$, the edges $(i \to j)$ and $(j \to i)$ will be included in E with probability P; if the resulting graph is not connected, restart the process. Because all communication channels will be effectively two-way, we call these graphs "undirected". In Section 6, graphs will be directed, and in the generation procedure, the unordered pair is replaced with the ordered pair (i, j), the edges are replaced with the directed edge $(i \to j)$, and the condition is replaced with the condition 'strongly connected'.

At round 0, each agent i has a uniformly random initial value $x_i(0) \in [-200, 200)$. We take inspiration from [1] and use the following DGD procedure: at each round k, each agent updates their value as follows:

$$x_i(k) \leftarrow F(\{x_j(k-1),\ \forall j | (j \to i) \in E\}) - T f_i'(x_i(k-1)),$$

where F is some aggregation function that takes a set of real numbers and outputs a real number, and T is the step size. For this paper, we arbitrarily fix the total number of rounds for each decentralized gradient descent to 10 000.

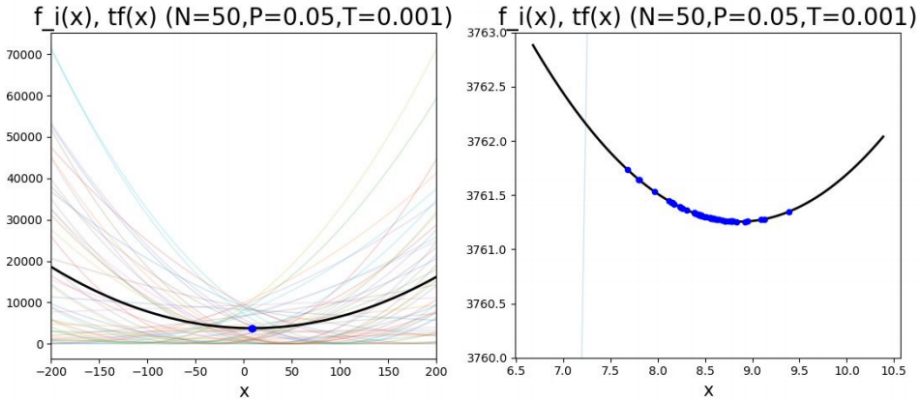

Figure 1: Graph plots of a sample randomly generated set of functions $f_i(x)$, along with final x_i values of all agents after 10 000 iterations of DGD, with the graph parameter $P = 0.05$.

To evaluate the DGD, we focus our attention on three quantities, which will usually (but not always) be displayed on logarithmic plots:

$$sd(k) := \frac{1}{N}\big(\sum_i x_i(k)\big) - \operatorname*{argmin}_x tf(x)$$

$$od(k) := \frac{1}{N}\big(\sum_i tf(x_i(k))\big) - \min_x tf(x)$$

$$s(k) := \texttt{stdev}_i(x_i(k))$$

Initially we will examine communication network consisting of only honest agents. Later we will add A corrupt nodes to the N honest nodes, each of whom has edges to all honest nodes (i.e. each corrupt agent can communicate with all honest agents), and each of whom can send arbitrary information to try and mess up the decentralized gradient descent. An example graph with $N = 6, A = 1$, with honest nodes in black and corrupt nodes in red, is shown in Figure 2.

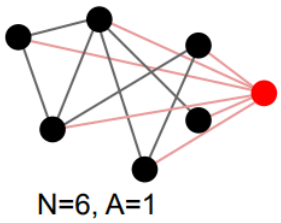

Figure 2: Example communication network with $N = 6$ honest nodes and $A = 1$ corrupt nodes.

3 No corrupt nodes

For the case $A = 0$, we set the function F to be the simple mean. We tested DGD on all values $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}$. For each P, we ran DGD for 10 000 iterations on each of 100 test sets, where each test set consists of a list of functions $f_i(x)$, a list of initial values $x_i(0)$, and a communication network, all randomly generated according to the rules described in the previous section; we then plotted the arithmetic means of the quantities $|sd(10000)|$ and $od(10000)$ obtained for all the test sets for that value of P. We repeated this procedure for each of 4 different step values $T \in \{0.01, 0.005, 0.002, 0.001\}$.

As shown in Figure 3, higher values of P tend to lead to lower mean $|sd(10000)|$ and $od(10000)$, which is to be expected, since higher P means higher connectivity between the honest nodes and thus better knowledge of the agents' $x_i(k)$. When $P = 1$, we observed an especially sharp decrease in mean $|sd(10000)|$ and $od(10000)$. The step size T did not seem to affect these two

quantities as much. However, lower T did lead to significantly lower $s(10000)$, meaning the agents' final values $x_i(10000)$ were much closer to each other for lower T.

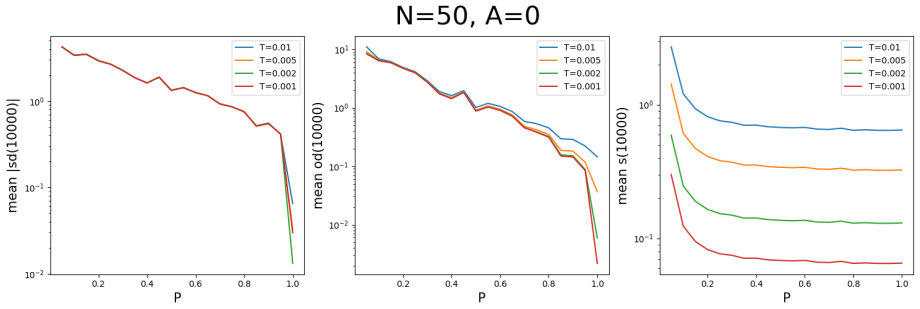

Figure 3: Graph of mean $|sd(10000)|$, $od(10000)$, and $s(10000)$ for parameters $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}, T \in \{0.01, 0.005, 0.002, 0.001\}$, and no corrupt nodes ($A = 0$). For each pair (P, T), DGD was tested on 100 randomly generated test sets.

4 One and two corrupt nodes

The first strategy we chose for the corrupt nodes was to send the value 1000 000 to every honest node during every round. This is because intuitively, the corrupt nodes want to maximally influence the values that the honest nodes calculate with the aggregation function F, preferably in a single direction, so we chose the large positive number direction. For comparison, all honest nodes' $x_i(k)$ values stay between -1000 and 1000.

In order for the DGD to become affected arbitrarily badly, we change the function F so that it is now a trimmed mean where the lowest A values and the highest A values are trimmed. We must trim on both sides, because otherwise, it would be easy for the corrupt nodes to counter our new DGD. If there is an honest node i that has $\leq 2A$ edges coming into it, then instead of using the formula $x_i(k) \leftarrow F(\{x_j(k-1),\ \forall j | (j \to i) \in E\}) - Tf_i'(x_i(k-1))$, we replace it with the standard single-user gradient descent formula: $x_i(k) \leftarrow x_i(k-1) - Tf_i'(x_i(k-1))$.

As shown in Figure 4, higher P also tends to lead to lower mean $|sd(10000)|$ and $od(10000)$, although this time there is a sharp decrease around low values of P instead of at around $P = 1$. Also, mean $|sd(10000)|$ and $od(10000)$ are both much higher in both the $A = 1$ and $A = 2$ cases than in the no corrupt nodes case. Except for $P = 0.05$, the step size T did not have a strong affect on mean $|sd(10000)|$ and $od(10000)$. However, lower T again correlated with lower $s(10000)$, although somehow this effect was rather diminished in the case of $A = 2,\ P < 0.2$. Additionally, the mean $s(10000)$ for $P \geq 0.2$ was similar across $A = 0, 1, 2$ and all values of T that we examined, meaning that the corrupt nodes

did not artificially spread out agents' final values $x_i(10000)$ for high enough P.

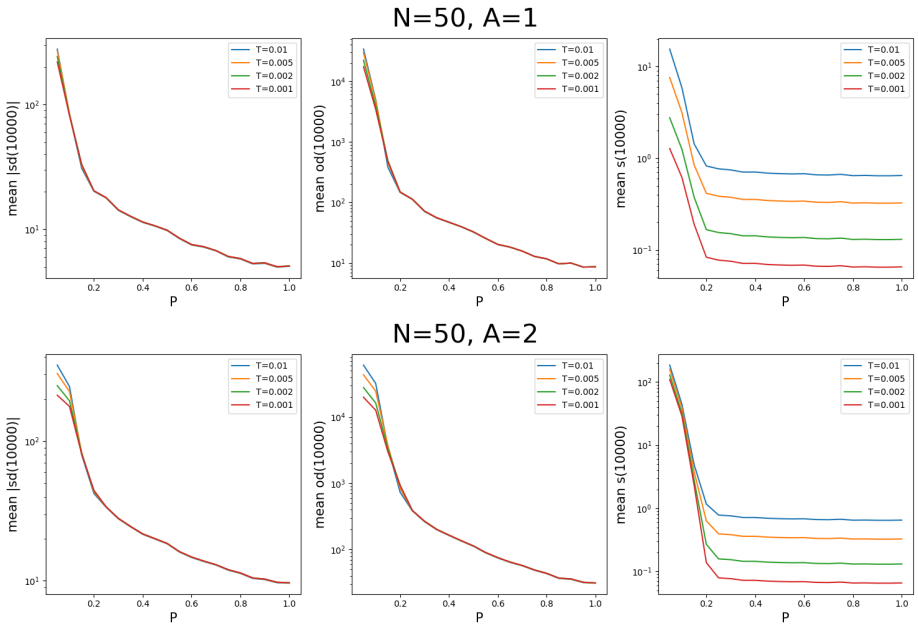

Figure 4: Graph of mean $|sd(10000)|$, $od(10000)$, and $s(10000)$ for parameters $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}, T \in \{0.01, 0.005, 0.002, 0.001\}$, and $A = 1, 2$ corrupt nodes. For each pair (P, T), DGD was tested on 100 randomly generated test sets.

To examine the effect of corrupt nodes on agents' final values $x_i(10000)$, we also plotted mean $sd(10000)$ (without the absolute value function) for $A = 0, 1, 2$ and the same sets of values for P, T. Figure 5 shows that for $A = 0$, $sd(10000)$ hovers around 0, whereas in $A = 1, 2$, $sd(10000)$ is a large positive number for low P and smaller positive number for higher P. This result indicates that the current strategy of corrupt nodes causes honest nodes' $x_i(k)$ values to be skewed higher than they should be.

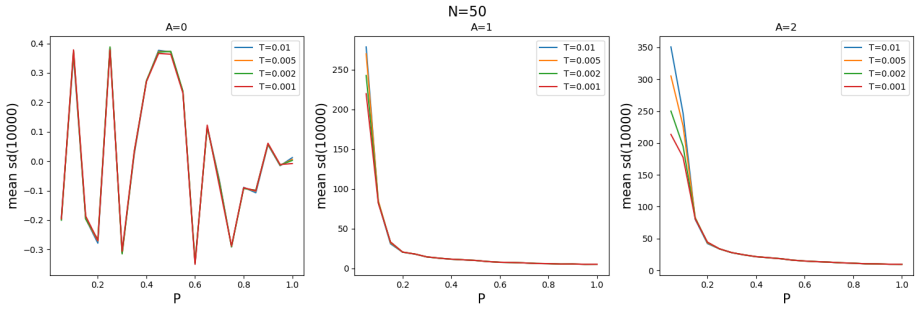

Figure 5: Graph of mean $sd(10000)$ for parameters $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}$, $T \in \{0.01, 0.005, 0.002, 0.001\}$, and $A = 0, 1, 2$ corrupt nodes. For each pair (P, T), DGD was tested on 100 randomly generated test sets.

5 Equivocation

For $A = 1$, another strategy for the corrupt node we tried was to send a very low value to half of the nodes and a very high value to the other half. In our implementation, the corrupt node sends the value 1000 000 to all honest nodes with index $i < \frac{N}{2}$, and the value -1000 000 to all other honest nodes. (Because all graphs were randomly generated with all vertices treated equally, this implementation is equivalent to having a randomly selected subset of $\frac{N}{2}$ honest nodes receive the very low value.) This means that for any single honest node, whether they receive a very low or a very high value from the corrupt node stays fixed throughout the entire DGD. Figure 6 shows that once again, higher P correlates to lower mean $|sd(10000)|$ and $od(10000)$.

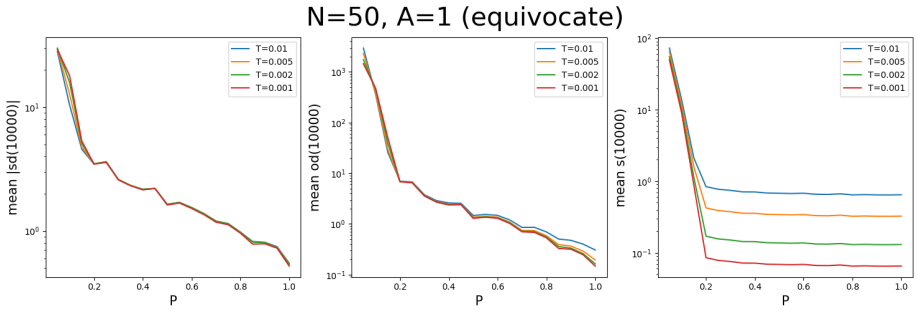

Figure 6: Graph of mean $|sd(10000)|$, $od(10000)$, and $s(10000)$ for parameters $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}, T \in \{0.01, 0.005, 0.002, 0.001\}$, and $A = 1$ equivocating corrupt node. For each pair (P, T), DGD was tested on 100 randomly generated test sets.

Although this strategy did not increase mean $|sd(10000)|$ and $od(10000)$ as much as the previous strategy for corrupt nodea did, the final $x_i(10000)$ values did end up becoming more spread out, as evidenced by mean $s(10000)$ being larger in $A = 1$+equivocate than in $A = 1$ for $P < 0.2$.

6 Directed Graphs

We repeated all the previous experiments on undirected graphs in this paper on directed graphs. The results are similar to those of undirected graphs, except for mean $sd(10000)$ (without the absolute value) being slightly higher for $A = 0, 1, 2$, and the plot of mean $sd(10000)$ for $A = 0$ looking different for directed graphs than for undirected graphs.

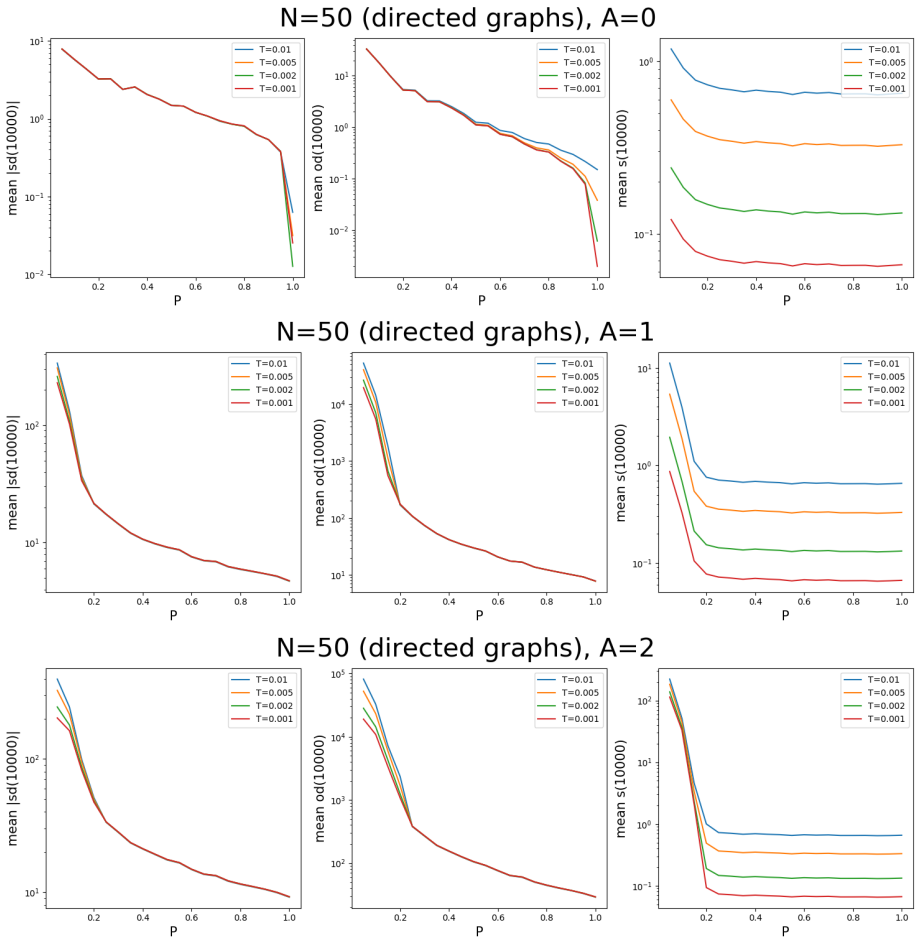

Figure 7: Graph of mean $|sd(10000)|$, $od(10000)$, and $s(10000)$ for parameters $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}, T \in \{0.01, 0.005, 0.002, 0.001\}$, $A = 0, 1, 2$ on directed graphs. For each pair (P, T), DGD was tested on 100 randomly generated test sets.

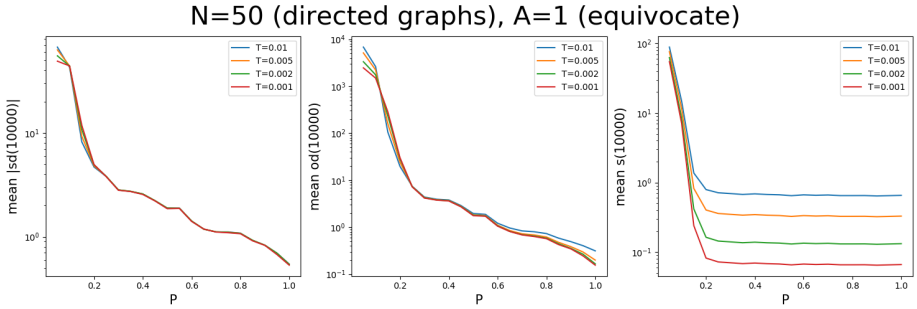

Figure 8: Graph of mean $|sd(10000)|$, $od(10000)$, and $s(10000)$ for parameters $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}, T \in \{0.01, 0.005, 0.002, 0.001\}$, $A =$ 1+equivocation on directed graphs. For each pair (P, T), DGD was tested on 100 randomly generated test sets.

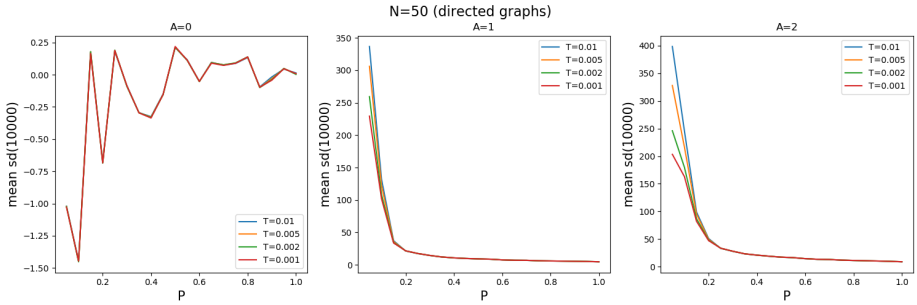

Figure 9: Graph of mean $sd(10000)$ for parameters $P \in \{0.05, 0.10, \ldots 0.95, 1.0\}$, $T \in \{0.01, 0.005, 0.002, 0.001\}$, and $A = 0, 1, 2$ corrupt nodes on directed graphs. For each pair (P, T), DGD was tested on 100 randomly generated test sets.

7 Conclusions and Future Directions

We have seen that the parameter P has a negative correlation with the error of decentralized gradient descent (DGD), measured with the functions $sd()$, $od()$, and $s()$. Adding a corrupt node that always sends a very high value to all honest nodes will significantly worsen the DGD by making all honest nodes' values skew higher than they should, and adding another corrupt node intensifies this effect. Having a single corrupt node that instead equivocates by sending a very low value to half of the honest agents to a very high value to the other half still disrupted the DGD, although not as much, but it also made the agents' final values more spread apart.

In future, we would like to perform the following:

- Expanding current experiments to d-dimensional functions $f_i(\overrightarrow{x})$ that are convex and L-Lipschitz continuous;
- Making more advanced strategies for DGD and corrupt nodes;
- Deriving asymptotic bounds on DGD error (whether using $sd()$, $od()$, $s()$, or some other metric) w.r.t. N, P, A, d;
- Investigating more pathological communication networks;

Our source code for this paper can be found at:
`https://github.com/jasonLLyang/Decentralized-Gradient-Descent`

8 Acknowledgments

We would like to thank Jun Wan and Hanshen Xiao for their mentorship and MIT PRIMES Computer Science for making this project possible.

References

[1] Kun Yuan, Qing Ling, Wotao Yin. On the Convergence of Decentralized Gradient Descent. arXiv:1310.7063, 2015.