Random Walk Distributed Dual Averaging Method For Decentralized Consensus Optimization

Cun Mu[‡][*] **Asim Kadav**[†], **Erik Kruus**[†], **Donald Goldfarb**[‡], **Martin Renqiang Min**[†]
Machine Learning Group, NEC Laboratories America[†]
Dept of Industrial Engineering and Operations Research, Columbia University[‡]

Abstract

In this paper, we address the problem of distributed learning over a large number of distributed sensors or geographically separated data centers, which suffer from sampling biases across nodes. We propose an algorithm called random walk distributed dual averaging (RW-DDA) method that only requires local updates and is fully distributed. Our RW-DDA method is robust to the change in network topology and amenable to asynchronous implementation. The theoretical analysis shows the algorithm has $O(1/\sqrt{t})$ convergence for non-smooth convex problems. Experimental results show that our algorithm outperforms competing methods in real-world scenarios, i.e. when trained over non-iid data and in the presence of communication link failures.

1 Introduction

With technological advancements in sensors, mobile devices, and data centers, machine learning algorithms are often applied to data distributed across these machines. However, in a real-world scenario, this data may not always be perfectly randomized. For example, physical sensors may collect non-iid data. Even data-centers often collect non-random data, biased towards the geography where they are located. It is often required to move the data to a central location to create perfectly random data. However, due to scale, or lack of a central coordinating resource, randomizing data may not always be possible and it is desirable to train over these nodes despite the presence of biased data at individual machines.

In this paper, we propose to solve this problem in the framework of *Decentralized Consensus Optimization (DCO)*, where all the nodes (agents), with their own utility functions, are connected through a network. The network system goal is to optimize the sum of all these utility functions, only through local computations and local information exchange with neighbors as specified by the communication graph of all nodes. Such framework, with applications ranging from large scale machine learning [11] to wireless sensor networks [6], tends to be scalable, simple to implement and robust to single points of failure. Consequently, many methods have been proposed for DCO recently [5]. Generally speaking, based on the style of the averaging/consensus step, these methods can be classified as model averaging methods [6, 7, 8, 14, 3, 4, 10], which average parameters, and dual averaging methods [2, 11, 12], which average subgradients. However, the successes of these methods heavily rely on the communication network to be static, which may not realistic due to node/edge failure.

The contribution of the paper is to propose an efficient algorithm, called *random walk distributed dual averaging (RW-DDA)* method, that is robust to the change in the network topology, and also suitable for stochastic subgradient and asynchronous implementation.

[*] Work done as a NEC Labs intern

2 Problem Statement

In mathematical terms, the optimization problem is defined on a connected undirected network and solved by n agents (computers) collectively,

$$\min_{\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d} \quad \bar{f}(\boldsymbol{x}) := \sum_{i=1}^{n} f_i(\boldsymbol{x}). \tag{2.1}$$

The feasible set $\mathcal{X}$ is a closed and convex set in $\mathbb{R}^d$ and is commonly known by all agents, whereas $f_i : \mathcal{X} \in \mathbb{R}$ is a convex function privately known by the agent i. Throughout the paper, we also assume that f_i is L-Lipschitz continuous over $\mathcal{X}$ with respect to the Euclidean norm $\|\cdot\|$. The network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, with the node set $\mathcal{N} = [n] := \{1, 2, \cdots, n\}$ and the edge set $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$, specifies the topological structure on how the information can be spread among agents through local agent interactions over time. In specific, each agent i can only send and retrieve information from its neighbors $\mathcal{N}(i) := \{j \mid (j, i) \in \mathcal{E}\}$ and himself.

3 RW-DDA method

Our random-walk distributed dual averaging (RW-DDA) method is shown step-by-step in Algorithm 1. Literally, in RW-DDA, each node i keeps a local estimate $\boldsymbol{x}_i$ and a dual variable $\boldsymbol{z}_i$ maintaining an accumulated subgradient. At iteration t, to update $\boldsymbol{z}_i$, each node needs to collect the $\boldsymbol{z}$-values of its neighbors, forms a convex combination with equal weight of the received information and adds its most recent local subgradient scaled by $|\mathcal{N}(i)| + 1$. After that, the dual variables $\boldsymbol{z}_i$ is projected to the primal space to obtain $\boldsymbol{x}_i$.

To implement RW-DDA, each node only needs to know its neighborhood information, which makes the algorithm robust to the change in network topology, frequently resulting from node failure or edge malfunction. As possibly inferred from the name, our RW-DDA method robustify the distributed dual averaging (DDA) method [2], based on the theory of random walk over undirected graph [9].

Algorithm 1 Random Walk Distributed Dual Averaging (RW-DDA) Method

```
Input: a predetermined nonnegative nonincreasing sequence $\{\alpha(t)\}$.
Initialization: $\boldsymbol{x}_i(0) = \boldsymbol{z}_i(0) = \boldsymbol{0}$, for all $i \in [n]$.
for $t = 0, 1, 2, \ldots,$ do
    1. Subgradient calculation:   $\boldsymbol{g}_i(t) \in \partial f_i(\boldsymbol{x}_i(t))$,   for each agent $i$.   (3.1)
    2. Dual updates:   $\boldsymbol{z}_i(t+1) = \frac{\sum_{j \in \mathcal{N}(i) \cup \{i\}} \boldsymbol{z}_j(t) + \boldsymbol{g}_i(t)}{|\mathcal{N}(i)| + 1}$,   for each agent $i$.   (3.2)
    3. Primal updates:
       $\boldsymbol{x}_i(t+1) = \mathcal{P}_{\mathcal{X}}[-\alpha(t)\boldsymbol{z}_i(t+1)] := \arg\min_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} + \alpha(t)\boldsymbol{z}_i(t+1)\|^2$,   for each agent $i$.   (3.3)
end for
```

4 Convergence Analysis

In this section, we will provide an $O(1/\sqrt{t})$-convergence result for Algorithm 1 when $\alpha(t)$ is properly chosen as $O(1/\sqrt{t})$. But prior to that, we will make an intuitive explanation to help understand the correctness of RW-DDA.

For notational convenience, we will define the matrix $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ with P_{ij} being $\frac{1}{|\mathcal{N}(i)+1|}$ for $j \in \mathcal{N}(i) \cup \{i\}$ and 0 otherwise. Clearly $\boldsymbol{P}$ is a row stochastic matrix, i.e. the sum of every row of $\boldsymbol{P}$ equals 1. We will also define the vector $\boldsymbol{\pi} \in \mathbb{R}^d$ with the i-th entry π_i being $\frac{|\mathcal{N}(i)|+1}{\beta}$, where $\beta := 2|\mathcal{V}| + |\mathcal{E}|$. It can easily verified that $\boldsymbol{\pi}$ is a probability vector, i.e. $\pi_i > 0$ and $\sum_{i \in [n]} \pi_i = 1$. With these notations, we are able to express (3.2) in a terser way. Imagine $\mathcal{X} \subseteq \mathbb{R}$, so $x_i(t)$, $z_i(t)$ and $g_i(t)$ are now all scalars. Then we can rewrite the update (3.2) as

$$\boldsymbol{z}(t+1) = \boldsymbol{P}\boldsymbol{z}(t) + \frac{1}{\beta}\operatorname{diag}(\boldsymbol{\pi})^{-1}\boldsymbol{g}(t) = \frac{1}{\beta}\sum_{s=0}^{t} \boldsymbol{P}^s \operatorname{diag}(\boldsymbol{\pi})^{-1}\boldsymbol{g}(t-s), \tag{4.1}$$

with $\boldsymbol{z}(t) = (z_1(t), z_2(t), \cdots, z_n(t))^\top$ and $\boldsymbol{g}(t) = (g_1(t), g_2(t), \cdots, g_n(t))^\top$. As we need each node to play the same role in the system, from (4.1), it is quite reasonable to require $\boldsymbol{P}^\infty \operatorname{diag}(\boldsymbol{\pi})^{-1} = \mathbb{1}_{n\times n}$, where $\boldsymbol{P}^\infty := \lim_{t\to\infty} \boldsymbol{P}^t$ and $\mathbb{1}_{n\times n}$ is the n by n matrix with all entries as one. Indeed, we can verify this requirement by the close connection between $\boldsymbol{P}$ and $\boldsymbol{\pi}$, as revealed in the following lemma, which can be regarded as a direct consequence of results for random walk under undirected graph [9]. This also justifies the appearance of random walk in the name of our algorithm.

Lemma 1. *$\boldsymbol{\pi}^\top \boldsymbol{P} = \boldsymbol{\pi}^\top$ and $\boldsymbol{P}^\infty := \lim_{t\to\infty} \boldsymbol{P}^t = \mathbf{1}\cdot\boldsymbol{\pi}^\top$.*

Proof. Consider a discrete-time Markov chain with state space as $\mathcal{V}$ and transition matrix specified by $\boldsymbol{P}$. It can be easily seen that this Markov chain is irreducible and aperiodic. Therefore, there exists a unique stationary distribution $\boldsymbol{d}$ satisfying $\boldsymbol{d} \geq 0$, $\mathbf{1}^\top \boldsymbol{d} = 1$, $\boldsymbol{d}^\top \boldsymbol{P} = \boldsymbol{d}^\top$ and $\boldsymbol{P}^\infty = \mathbf{1}\cdot\boldsymbol{d}^\top$. Since the probability vector $\boldsymbol{\pi}$ satisfies the so-called detailed balance equation, i.e. $\pi_i P_{ij} = \pi_j P_{ji}$, $\boldsymbol{\pi}$ is the stationary distribution, i.e. $\boldsymbol{d} = \boldsymbol{\pi}$. □

Next, we will state two lemmas that are similar in nature to Theorem 1 and Theorem 2 in [2]. The proofs can obtained by modifying arguments in [2], which we omit here.

Lemma 2. *Consider the sequences $\{\boldsymbol{x}_i(t)\}$ and $\{\boldsymbol{z}_i(t)\}$ generated by RW-DDA (Algorithm 1). Then for any $\boldsymbol{x}^\star \in \mathcal{X}$ and for each node $i \in [n]$, we have*

$$\begin{aligned}\bar{f}\left(\widehat{\boldsymbol{x}}_i\left(T\right)\right) - \bar{f}(\boldsymbol{x}^\star) \leq & \frac{L^2 n}{2\beta T}\sum_{t=1}^{T} \alpha(t-1) + \frac{\beta}{2nT\alpha(T)}\left\|x^\star\right\|^2 \\ & + \frac{L}{T}\sum_{t=1}^{T}\alpha(t)\left(\frac{2}{n}\sum_{j=1}^{n}\left\|\bar{\boldsymbol{z}}(t) - \boldsymbol{z}_j(t)\right\| + \left\|\bar{\boldsymbol{z}}(t) - \boldsymbol{z}_i(t)\right\|\right),\end{aligned} \tag{4.2}$$

where $\widehat{\boldsymbol{x}}_i(T) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_i(t)$ and $\bar{\boldsymbol{z}}(t) = \sum_{i=1}^{n}\pi_i \boldsymbol{z}_i(t)$.

Lemma 3. *Consider the sequences $\{\boldsymbol{x}_i(t)\}$ and $\{\boldsymbol{z}_i(t)\}$ generated by RW-DDA (Algorithm 1). Define $\bar{\boldsymbol{z}}(t) = \boldsymbol{\pi}^\top \boldsymbol{z}(t)$. Then we have,*

$$\left\|\bar{\boldsymbol{z}}(t) - \boldsymbol{z}_i(t)\right\| \leq \frac{L}{\beta \pi_{\min}}\sqrt{\frac{1-\pi_i}{\pi_i}}\frac{1}{1-\sigma_2(\boldsymbol{P})}, \tag{4.3}$$

where $\pi_{\min} = \min\{\pi_i\}$ and $\sigma_2(\cdot)$ denotes the second largest singular value.

Finally, we are ready to present the convergence theorem.

Theorem 1. *Consider the sequences $\{\boldsymbol{x}_i(t)\}$ and $\{\boldsymbol{z}_i(t)\}$ generated by RW-DDA (Algorithm 1). Define the running average at each node i as $\widehat{\boldsymbol{x}}_i(T) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_i(t)$. Then for any $\boldsymbol{x}^\star \in \mathcal{X}$ with $\|\boldsymbol{x}^\star\| \leq R$, and for each node $i \in [n]$, one has*

$$\bar{f}\left(\widehat{\boldsymbol{x}}_i\left(T\right)\right) - \bar{f}(\boldsymbol{x}^\star) \leq \frac{2LR}{\sqrt{nT(1-\sigma_2(\boldsymbol{P})\pi_{\min}^{3/4}}} \tag{4.4}$$

when the step size $\alpha(t)$ is chosen as $\frac{\beta(\pi_{\min})^{3/4}\sqrt{1-\sigma_2(\boldsymbol{P})}R}{4L\sqrt{n}}\cdot\frac{1}{\sqrt{t}}$.

Proof. Let us choose $\alpha(t)$ in the form of $c/\sqrt{t}$, where c is to be optimized later.

Plugging (4.3) into (4.2), we reach

$$\begin{aligned}\bar{f}\left(\widehat{\boldsymbol{x}}_i\left(T\right)\right) - \bar{f}(\boldsymbol{x}^\star) &\leq \frac{L^2 n}{\beta\sqrt{T}}\cdot c + \frac{\beta}{2n\sqrt{T}}R^2\cdot\frac{1}{c} + \frac{6L^2}{\sqrt{T}\beta\pi_{\min}^{3/2}(1-\sigma_2(\boldsymbol{P}))}\cdot c \\ &\leq \frac{7L^2}{\sqrt{T}\beta\pi_{\min}^{3/2}(1-\sigma_2(\boldsymbol{P}))}\cdot c + \frac{\beta}{2n\sqrt{T}}R^2\cdot\frac{1}{c},\end{aligned} \tag{4.5}$$

where we have used the fact that $\sum_{t=1}^{T} t^{-1/2} \leq \int_{t=0}^{T} t^{-1/2}dt = \sqrt{T}$.

The claimed result holds directly as we optimize the upper bound (4.5) with respect to the parameter c. □

5 Extension

Our RW-DDA method can be easily adapted to incorporate stochastic gradients to solve an optimization problem with a convex regularizer (e.g. ℓ_1, nuclear norm). In specific, Algorithm 2, a natural modification of RW-DDA (Algorithm 1), is capable of solving

$$\min_{\boldsymbol{x}} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}) + \phi(\boldsymbol{x}), \tag{5.1}$$

where $\phi(\boldsymbol{x})$ is a convex regularizer. Its convergence proof follows directly from combining our analysis above and arguments used in previous literature [13, 2], which we omit here.

Algorithm 2 Generalized Random Walk Distributed Dual Averaging (GRW-DDA) Method

```
**Input:** a predetermined nonnegative nonincreasing sequence $\{\alpha(t)\}$.
**Initialization:** $\boldsymbol{x}_i(0) = \boldsymbol{z}_i(0) = \mathbf{0}$, for all $i \in [n]$.
**for** $t = 0, 1, 2, \ldots,$ **do**
    1. Stochastic subgradient calculation:  $\mathbb{E}\left[\boldsymbol{g}_i(t)\right] \in \partial f_i(\boldsymbol{x}_i(t))$,  for each agent $i$.   (5.2)
    2. Dual updates:  $\boldsymbol{z}_i(t+1) = \frac{\sum_{j\in\mathcal{N}(i)\cup\{i\}} \boldsymbol{z}_j(t) + \boldsymbol{g}_i(t)}{|\mathcal{N}(i)|+1}$,  for each agent $i$.   (5.3)
    3. Primal updates:
        $\boldsymbol{x}_i(t+1) = \mathrm{Prox}_{t\alpha(t)\phi(\cdot)}[-\alpha(t)\boldsymbol{z}_i(t+1)]$
        $= \arg\min_{\boldsymbol{x}} \frac{1}{2}\left\|\boldsymbol{x} + \alpha(t)\boldsymbol{z}_i(t+1)\right\|^2 + t\alpha(t)\phi(\boldsymbol{x})$,  for each agent $i$.   (5.4)
**end for**
```

6 Experiments

We now evaluate the RW-DDA algorithm for the SVM application using the RCV1 dataset [1]. We compare RW-DDA with model averaging, both implemented over the MALT framework [4]. To improve performance, we perform the following three optimizations. First, instead of calculating the full gradient on every iteration, we only compute the sparse gradient and separately correct the regularizer [1]. Second, instead of sending z (or w for model averaging) after every example, nodes locally process examples (usually 500-5000), and then communicate z. We adjust the learning rate parameters to account for the batched communication. Finally, we maintain a running sum average over the dual, and only compute this sum only during reduce (incoming z parameters). We run all our experiments using 6 ranks. We compare the average training error over all ranks w.r.t. wall clock time for non-i.i.d. data and with failures as shown in Figure 1 (a) and (b) respectively. We find that RWDDA converges faster in time, and achieves a stable accuracy 2.6% better than model averaging after 2000 iterations.

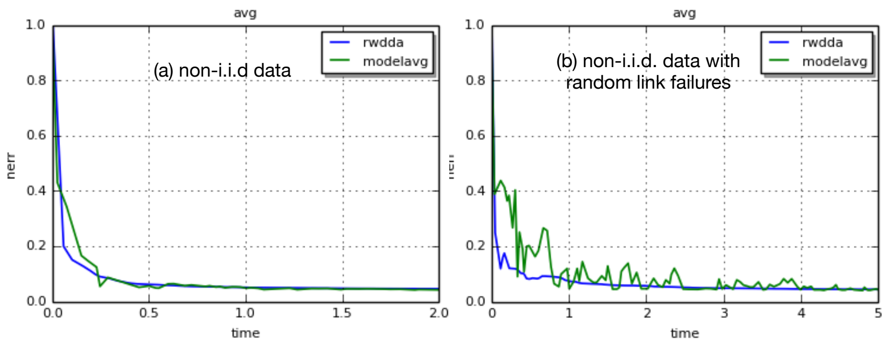

Figure 1: Figure (a) shows training error vs time for RW-DDA and model averaging for six ranks for non-iid dataset. Figure (b) shows the same experiment with intermittent link failures. Each machine communicates z after processing a local epoch (slightly over 3300 examples). We find RW-DDA method to be stable and faster to converge especially in the presence of failures.

References

[1] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[2] J. C. Duchi, A. Agarwal, and M. J. Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling. *Automatic control, IEEE Transactions on*, 57(3):592–606, 2012.

[3] D. Jakovetic, J. Xavier, and J. M. Moura. Fast distributed gradient methods. *Automatic Control, IEEE Transactions on*, 59(5):1131–1146, 2014.

[4] H. Li, A. Kadav, E. Kruus, and C. Ungureanu. Malt: distributed data-parallelism for existing ml applications. In *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015.

[5] A. Nedić. Distributed optimization. In *Encyclopedia of Systems and Control*, pages 1–12. Springer London, 2014.

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

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

[8] S. S. Ram, A. Nedić, and V. V. Venugopal. A new class of distributed optimization algorithms: Application to regression of distributed data. *Optimization Methods and Software*, 27(1):71–88, 2012.

[9] S. Ross. *Stochastic processes*, volume 2. John Wiley & Sons New York, 1996.

[10] W Shi, Q. Ling, G. Wu, and W. Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[11] K. Tsianos, S. Lawlor, and M. G. Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 1543–1550. IEEE, 2012.

[12] K. Tsianos and M. G. Rabbat. Distributed dual averaging for convex optimization under communication delays. In *American Control Conference (ACC), 2012*, pages 1067–1072. IEEE, 2012.

[13] L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems*, pages 2116–2124, 2009.

[14] K. Yuan, Q. Ling, and W. Yin. On the convergence of decentralized gradient descent. *arXiv preprint arXiv:1310.7063*, 2013.