Journal of Machine Learning Research 23 (2022) 1-31 Submitted 9/20; Revised 7/21; Published 1/22

Asymptotic Network Independence and Step-Size for a Distributed Subgradient Method

Alex Olshevsky ALEXOLS@BU.EDU
Department of Electrical and Computer Engineering
Division of Systems Engineering
Boston University Boston, MA, 02134

Editor: Suvrit Sra

Abstract

We consider whether distributed subgradient methods can achieve a linear speedup over a centralized subgradient method. While it might be hoped that distributed network of n nodes that can compute n times more subgradients in parallel compared to a single node might, as a result, be n times faster, existing bounds for distributed optimization methods are often consistent with a slowdown rather than speedup compared to a single node.

We show that a distributed subgradient method has this "linear speedup" property when using a class of square-summable-but-not-summable step-sizes which include $1/t^\beta$ when $\beta \in (1/2, 1)$; for such step-sizes, we show that after a transient period whose size depends on the spectral gap of the network, the method achieves a performance guarantee that does not depend on the network or the number of nodes. We also show that the same method can fail to have this "asymptotic network independence" property under the optimally decaying step-size $1/\sqrt{t}$ and, as a consequence, can fail to provide a linear speedup compared to a single node with $1/\sqrt{t}$ step-size.

Keywords: Distributed Systems, Convex Optimization, Subgradient Method

1. Introduction

We consider the standard setting of distributed convex optimization: $f_1(x), \ldots, f_n(x)$ are convex functions from $\mathbb{R}^d$ to $\mathbb{R}$, with node i of the network being the only node which can compute subgradients of the function $f_i(x)$. The goal is to compute a minimizer

$$x^* \in \arg\min_{x \in \Omega} F(x), \tag{1}$$

where

$$F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

and Ω is a closed compact convex set. The underlying method must be decentralized, relying only on local subgradient computations and peer-to-peer message exchanges in a certain graph G. In particular, we will consider the "standard model" of distributed optimization where at each step, node i computes a subgradient of its local function, possibly performs a projection step onto the set Ω, and broadcasts a message to its neighbors.

This problem setup is a natural model for machine learning over a network of processors. Minimizing the function $F(x)$ typically comes from empirical loss minimization. The

function $F(x)$ typicaly measures how well a model parametrized by the vector x can fit a collection of data points; distributing the data points among n processors will result in the problem formulation of Eq. (1). While it is possible to make additional assumptions, such as the functions $f_i(x)$ being identical or nearly identical (Haddadpour et al. (2019); Stich (2018)), or having gradients or Hessians that are not too far apart (Karimireddy et al. (2020); Khaled et al. (2020)), in this work we will assume that the functions $f_i(x)$ can be arbitrarily different.

A variation on this setup considers the situation when the underlying graph G is taken to be the star graph, sometimes called "local gradient descent" (see Stich (2019)). The advantage of using the star graph is that one can design simple protocols involving rounds of interaction between the center and the leaf nodes which are not available in the setting where G is an arbitrary graph. However, a disadvantage of using the star graph is that, as the number of nodes gets large, the number of bits that need to be transmitted to the center increases as well. One way to avoid this problem is to consider optimization over arbitrary graphs G instead, as we do in this paper. Changing topology to an arbitrary graph or even to a time-varying sequence of graphs G has been empirically shown to speed up distributed training (Assran et al. (2019)).

This problem formulation is now classical; it first analyzed in Nedic and Ozdaglar (2009), where a distributed subgradient method was proposed for the unconstrained case when $\Omega = \mathbb{R}^n$. The case with the constraint Ω was first analyzed in Nedic et al. (2010). Both papers proposed methods inspired by the "average consensus" literature, where nodes mix subgradient steps on their local functions with linear combinations of their neighbors iterates.

Distributed optimization methods have attracted considerable attention since the publication of Nedic and Ozdaglar (2009) for several reasons. First, it is hoped that distributed empirical loss minimization in machine learning could result in faster training. Second, many problems in control and signal processing among network of nodes involve nodes acting to maximize a global objective from local information, and Eq. (1) is thought to be among the simplest problems of this type. Over the past decade, thousands of papers have been written on different variations of this problem, and it would be impossible to survey all this related work; instead, we refer the reader to the recent survey Nedić et al. (2018).

We next launch into a discussion of the main motivating concern of this paper, namely how the performance of distributed optimization methods compares to their centralized counterparts. We begin by discussing the available guarantees for the (centralized) subgradient method, so that we can can contrast those guarantees to the available distributed bounds in our survey of previous work, which will follow.

1.1 The subgradient method

The projected subgradient method run on the function $F(x)$ takes the form

$$y(t+1) = P_\Omega \left[y(t) - \alpha(t) g_F(t) \right],$$

where $g_F(t)$ is a subgradient of the function $F(\cdot)$ at $y(t)$, and P_Ω is the projection onto Ω.

The standard reference for an analysis of this method is the set of lecture notes Boyd et al. (2003). It is usually assumed that $||g_F(t)||_2 \leq L$ for all t, i.e., all subgradients are

bounded; and Ω is assumed to have diameter at most D. The function $F(x)$ may have more than one minimizer over Ω; we select one minimizer arbitrarily and call it x^*.

The step-size $\alpha(t)$ needs to be properly chosen. There are two choices that are typically analyzed in this setting. One is to set $\alpha(t) = 1/\sqrt{t}$, which turns out to be the optimal decay rate. The other is to choose $\alpha(t)$ to be "square summable but not summable" as in the following assumption.

Assumption 1 *The sequence $\alpha(t)$ satisfies*

$$\begin{aligned} \sum_{t=1}^{+\infty} \alpha^2(t) &< \infty \\ \sum_{t=1}^{+\infty} \alpha(t) &= +\infty \end{aligned}$$

We now briefly summarize the standard analysis of the method from Boyd et al. (2003), which the reader can consult for details. The analysis is based on the following recurrence relation, to the effect that, up to second order terms, the method gets closer to the set of minimizers at every step:

$$||y(k+1) - x^*||_2^2 \leq ||y(k) - x^*||_2^2 - 2\alpha(k)(F(y(k)) - F^*) + L^2\alpha^2(k),$$

It is standard to re-arrange this into a telescoping sum as

$$2\alpha(k)(F(y(k)) - F^*) \leq ||y(k) - x^*||_2^2 - ||y(k+1) - x^*||_2^2 + L^2\alpha^2(k), \tag{2}$$

and then sum it up over $k = 1, \ldots, t$. Indeed, defining

$$y_\alpha(t) := \frac{\sum_{k=1}^t \alpha(k) y(k)}{\sum_{k=1}^t \alpha(k)},$$

and summing up Eq. (2) and then appealing to the convexity of $F(x)$ we can obtain that

$$F(y_\alpha(t)) - F^* \leq \frac{D^2 + L^2 \sum_{k=1}^t \alpha^2(k)}{2 \sum_{k=1}^t \alpha(k)}, \tag{3}$$

where $||y(0) - x^*||_2^2 \leq D^2$ as Ω was assumed to have diameter D. Finally, by Assumption 1, the right-hand side goes to zero, and so we obtain that the subgradient method works.

A variation on this argument can get rid of the dependence on L in Eq. (3). This requires the following assumption.

Assumption 2 *There is a constant C_α such that for all positive integers t,*

$$\sum_{k=1}^{t} \alpha(k) \leq C_\alpha \sum_{k=\lceil t/2 \rceil}^{t} \alpha(k).$$

This assumption can be motivated by observing that it is satisfied by step-sizes that decay polynomially as $\alpha(t) = 1/t^\beta$ when $\beta > 0$.

With this assumption in place, one can set $t' = \lceil t/2 \rceil$ and instead sum Eq. (2) from t' to t. Defining the running average from time t' to t as

$$y'_\alpha(t) := \frac{\sum_{k=t'}^t \alpha(k) y(k)}{\sum_{k=t'}^t \alpha(k)}$$

this immediately yields the following proposition.

Proposition 1 *Suppose Assumptions 1 and 2 on the step-size are satisfied, $F(x)$ is a convex functions whose subgradients are upper bounded by L in the Euclidean norm, and t is large enough so that we have the upper bound*

$$\sum_{k=\lfloor t/2 \rfloor}^{+\infty} \alpha^2(k) \leq \frac{D^2}{L^2}. \tag{4}$$

Then

$$F(y'_\alpha(t)) - F^* \leq \frac{D^2 C_\alpha}{\sum_{k=1}^t \alpha(k)}.$$

This result has no dependence on L, but at the expense of multiplying the dependence on D by the constant C_α. For example, if $\alpha(t) = 1/t^{3/4}$, it is an exercise to verify that one can take $C_\alpha = 6$. We note that since the step-size $\alpha^2(t)$ is square summable, Eq. (4) is guaranteed to hold for large enough t.

The bound of this proposition suggests to take $\alpha(t)$ decaying as slowly as possible (so that $\sum_{k=1}^t \alpha(k)$ grows as fast as possible) while still keeping $\alpha(t)$ square summable but not summable. There is no optimal choice, but in general one wants to pick $\alpha(t) = 1/t^\beta$ where β is close to $1/2$, but not $1/2$ since $\alpha(t) = 1/\sqrt{t}$ is not square summable. The result will be a decay rate of $F(y'_\alpha(t)) - F^* = O(1/t^{1-\beta})$.

One can redo the above argument with the rate of decay of $\alpha(t) = 1/\sqrt{t}$ to obtain an optimal rate of decay. In that case, because this is not a square summable step-size, the dependence on L cannot be avoided. However, since $\sum_{k=t'}^t 1/t = O(1)$, we can simply repeat all the steps above to give the bound

$$F(y_\alpha(t)) - F^* \leq O\left(\frac{D^2 + L^2}{\sqrt{t}}\right), \tag{5}$$

An advantage to optimizing over a closed convex set Ω is that in the case where it has finite diameter, the stndard analysis sketched here allows us to avoid extraneous logarithmic factors without fixing the number of steps ahead of time. One can also choose $\alpha(t)$ depending on the constants D and L to obtain better scaling with respect to those constants; however, in this paper, for simplicity we restrict our attention to unoptimized step-sizes of the form $\alpha(t) = 1/t^\beta$.

We next compare these results for the centralized subgradient to available convergence times in the distributed case.

1.2 Convergence times of distributed subgradient methods

A number of distributed subgradient methods have been proposed in the literature, with the simplest being

$$x(t+1) = Wx(t) - \alpha(t)g(t), \tag{6}$$

which was analyzed in Nedic and Ozdaglar (2009). Here $x(t)$ is an $n \times d$ matrix, with the i'th row of $x(t)$ being controlled by agent i; we will use $x_i(t)$ to denote the same i'th row. The matrix $g(t)$ is also $n \times d$ and it's i'th row, which we will denote by $g_i(t)$, is a subgradient of the function $f_i(x)$ at $x = x_i(t)$. The matrix W is doubly stochastic and needs to satisfy some connectivity and non-aperiodicity conditions; it suffices to assume that W has positive diagonal and that the directed graph corresponding to the positive entries of W is strongly connected. The underlying assumption in Eq. (6) is that the time it takes to compute a subgradient and the time it takes to broadcast $x(t)$ to neighbors are comparable, as both are being done between iteration t and iteration $t+1$.

It was shown in Nedic and Ozdaglar (2009) that, for small enough constant stepsize $\alpha(t) = \alpha$, this method results in a final error that scales linearly in α. The projected version

$$x(t+1) = P_\Omega \left[Wx(t) - \alpha(t)g'(t) \right], \tag{7}$$

was studied in Nedic et al. (2010); here the projection operator P_Ω acts on each row of the matrix and $g'(t)$ is composed of subgradients evaluated at $Wx(t)$. It was shown that, under an appropriately decaying step-size, this scheme results convergence to an optimal solution.

Our interest is in the convergence rate of these methods; in particular, we want to see if the parallelization inherent in having n nodes query subgradients at the same time helps convergence. A useful benchmark is to consider a single node, which knows all the functions $f_i(x), i = 1, \ldots, n$, and can compute the gradient of one of these functions at every time step. We will call the rate obtained in this setup by performing full-batch subgradient descent (i.e., by computing the gradient of $F(x)$ by querying the subgradients of $f_1(x), \ldots, f_n(x)$ in n steps) the *single-node rate*. The single node rate consists in multiplying all the rates obtained in the previous section by n, consistent with n steps to compute a single subgradient of $F(x)$. For example, the bound of Proposition 1 becomes

$$F(y'_\alpha(t)) - F^* \leq \frac{nD^2C_\alpha}{\sum_{k=1}^t \alpha(k)}.$$

Ideally, one hopes for a factor n speedup over the single note rate, since the n-node network can compute n subgradients in parallel at every step. This corresponds to a convergence time that removes the factor of n from the last equation[1].

Most of the existing convergence analyses do not attempt to write out all the scalings for the convergence times of distributed optimization methods; many papers write out the scaling with t but do not focus on scaling with the number of nodes. Unfortunately, once those scalings are traced out within the course of the proof, they tend to scale with $(1-\sigma)^{-1}$, where σ is the second-largest singular value associated with the matrix W. The quantity

1. Centralized full-batch subgradient descent is a natural point of comparison for the distributed subgradient method since both methods work under similar assumptions (e.g., no randomization) and the existing bounds for it are not worse than any alternatives (such as incremental subgradient).

$(1-\sigma)^{-1}$ can scale as much as $O(n^2)$ in the worst-case over all graphs (see Nedić et al. (2018)), so the underlying scaling could actually be worse than the single-node rate.

A concrete example of this comes from the survey paper Nedić et al. (2018), where a worse case rate is explicitly written out. The unconstrained case is studied, with step-size $\alpha = 1/\sqrt{T}$ and the algorithm is run for T steps. It is shown in Nedić et al. (2018) that

$$F(y_\alpha(t)) - F^* \leq O\left(\frac{\left|\left|\frac{1}{n}\sum_{i=1}^n x_i(t) - x^*\right|\right|_2^2 + L^2(1-\sigma)^{-1}}{\sqrt{T}}\right) \tag{8}$$

Comparing this with Eq. (5), we see that, in the worst case when $(1-\sigma)^{-1} \approx \Theta(n^2)$, this is a factor of n slower than the single node rate – in spite of the fact that the network can compute n gradients in parallel. Similar issues affect all the upper bounds in this setting that have been derived in the previous literature, in particular the bounds derived in Duchi et al. (2011) for dual subgradient, in Scaman et al. (2018) (for the standard setting of distributed optimization where a single message exchange in neighbors is possible per step; Scaman et al. (2018) also explored methods where potentially $(1-\sigma)^{-1/2}$ steps of gossip are possible per step, in which case the dependence on $1-\sigma$ for the number of subgradients computed can be removed), and those implicit in Nedic et al. (2010) for square-summable-but-not-summable step-sizes.

In the paper Olshevsky (2017), it was shown how, for a particular way to choose the matrix W in a distributed way, it is possible to replace the $(1-\sigma)^{-1}$ with an $O(n)$ factor, matching the single-node rate. The idea was to use Nesterov acceleration, which allows us replace $(1-\sigma)^{-1}$ with $(1-\sigma)^{-1/2}$, and argue that for a certain particularly chosen set of weights the latter quantity is $O(n)$. However, this required slightly stronger assumptions (namely, knowing either the total number of nodes or a reasonably accurate upper bound on it). A similar idea was explored in Scaman et al. (2018). While this does not offer a speedup over the single-node rate, at least it matches it.

To attain a linear speedup over the single node rate, we would need to remove $(1-\sigma)^{-1}$ from the numerator in the scaling above. In this paper, we explore when this can be done and when it cannot.

2. Related work

Among methods that converge to the optimal solution, the first examples of distributed algorithms that obtained a linear speedup were Lian et al. (2017) and Morral et al. (2017). In Lian et al. (2017), the case of stochastic non-convex gradient descent was considered, and it was shown that when the number of iterations is large enough, the distributed method achieves a linear speedup over the centralized. A similar result was shown in Morral et al. (2017) for the case of strongly convex distributed stochastic gradient descent; specifically, using tools from stochastic approximation, Morral et al. (2017) derived an expression for the limiting variance of the decentralized method which matches the performance of the centralized method. These papers have spawned a number of follow-up works (e.g., Spiridonoff et al. (2020); Koloskova et al. (2019); Assran et al. (2019); Wang et al. (2019); Lian et al. (2018); Tang et al. (2018); Jiang and Agrawal (2018); Nazari et al. (2019), among others), making a number of further refinements in terms of the communication topology,

communication requirements, transient times, and testing on real world benchmarks. In the earlier paper Towfic et al. (2016), a linear speedup was shown under the assumptions that all the functions $f_i(\cdot)$ have the same minimum. The corresponding result in the constant step-size case, where the system converges to a neighborhood of the optimal solution, was shown even earlier in Chen and Sayed (2015b,a), where it was proved that the limiting mean-squared error corresponding to any graph is the same as that of the complete graph. More recently, it was shown in Richards et al. (2020) that distributed SGD can attain optimal statistical rates for generalization if all functions $f_i(\cdot)$ are sums of the quadratic sampled from the same distribution and the number of total samples per agent is sufficiently large. In Hendrikx et al. (2019) the case when every node knows a finite sum of functions from which it can sample was considered, and a linear speedup was shown in the case when the network was not too large.

In contrast to our work, all of these papers studied the case of stochastic gradients, and also made stronger assumptions such as Lipschitz continuity of the gradient or strong convexity. By contrast, we study the usual (i.e., non-stochastic) subgradient method under only the assumption that the function is convex.

This setup has been analyzed in a number of settings in the control literature, as we already discussed, and without showing any linear speedup. Additional works to be mentioned are a primal-dual approach was explored in Zhu and Martínez (2011), using the dual subgradient method instead of the ordinary subgradient method was studied in Duchi et al. (2011). An analysis that elaborates on scaling with the network was given in Ram et al. (2010). Finally, we mention that our paper is close in spirit to the recent works Neglia et al. (2019, 2020), which are also concerned with similar questions. A number of expressions for scaling with topology are derived in Neglia et al. (2019) which also considers variability in the time it takes to compute subgradients. The paper Neglia et al. (2020) gives a convergence rate that does not depend on the underlying network when the variability of the gradients is zero or close to zero.

2.1 Our contributions

We will analyze a minor variation of Eq. (6) and Eq. (7):

$$x(t+1) = WP_{\Omega}\left[x(t) - \alpha(t)g(t)\right], \tag{9}$$

This is slightly more natural than Eq. (7), since $g(t)$ here is the subgradient evaluated at $x(t)$, and not at $Wx(t)$ as in Eq. (7). This makes analysis somewhat neater.

Our main result will be to show that a linear speedup is achieved by this iteration on a class of square-summable-but-not-summable stepsizes which include $\alpha(t) = 1/t^{\beta}$ with $\beta \in (1/2, 1)$. This is done by showing that, provided t is large enough, we can give a performance bound that does not depend on $(1-\sigma)^{-1}$, i.e., is network independent. We will also show that the same assertions fail for the optimally decaying step-size $\alpha(t) = 1/\sqrt{t}$.

We next give a formal statement of our main results. First, let us state our assumptions formally as follows.

Assumption 3 *Each function $f_i(x) : \mathbb{R}^d \to R$ is a convex with all of its subgradients bounded by L in the Euclidean norm. Moreover, the set Ω is a closed convex set. Each node begins with an identical initial condition $x_i(0) \in \Omega$.*

Assumption 4 *The matrix W is nonnegative, doubly stochastic, and with positive diagonal. The directed graph corresponding to the positive entries of W is strongly connected.*

Secondly, we will be making an additional assumption on step-size.

Assumption 5 *The sequence $\alpha(t)$ is nonincreasing and there is a constant $C_{\alpha'}$ such that for all t,*

$$\alpha(\lfloor t/2 \rfloor) \leq C'_{\alpha}\alpha(t).$$

This assumption essentially bounds how fast $\alpha(t)$ can decrease over the period of $t/2, \ldots, t$. It is motivated by observing that it holds for step-sizes of the form $\alpha(t) = 1/t^{\beta}$.

Finally, let us introduce the notation

$$\overline{x}(t) = \frac{1}{n}\sum_{i=1}^{n} x_i(t),$$

for the average of the iterates at time t. We adopt a general convention that, for a vector or a matrix, putting an overline will mean referring to the average of the rows.

Similarly to what was done in the previous subsection, we define

$$x'_{\alpha}(t) = \frac{\sum_{k=t'}^{t} \alpha(k)\overline{x}(k)}{\sum_{k=t'}^{t} \alpha(k)}$$

Our first main result is the following theorem.

Theorem 2 (Asymptotic Network Independence with Square Summable Step-Sizes) *Suppose Assumptions 1, 2, and 5 on the step-size, Assumption 3 on the functions, and Assumption 4 on the mixing matrix W all hold.*

Then if t is large enough so that the tail sum satisfies the upper bound

$$\sum_{k=\lfloor t/2 \rfloor}^{+\infty} \alpha^2(t) \leq \frac{D^2(1-\sigma)}{10C'_{\alpha}L^2} \tag{10}$$

and also

$$t \geq \Omega\left(\frac{1}{1-\sigma}\log\left[(1-\sigma)t\alpha_{\max}/(C'_{\alpha}\alpha(t))\right]\right) \tag{11}$$

then we have the network-independent bound

$$F(x'_{\alpha}(t)) - F^* \leq \frac{D^2 C_{\alpha}}{\sum_{k=0}^{t}\alpha(k)} \tag{12}$$

In particular, if $\alpha(t) = 1/t^{\beta}$ where β lies in the range $(1/2, 1)$, then when t additionally satisfies

$$t^{2\beta-1} \geq \Omega_{\beta}\left(\frac{L^2}{D^2(1-\sigma)}\right) \tag{13}$$

we have the network-independent bound

$$F(x'_\alpha(t)) - F^* \leq O_\beta\left(\frac{D^2}{t^{1-\beta}}\right), \tag{14}$$

where the subscript of β denotes that the constants in the $O(\cdot)$ and $\Omega(\cdot)$ notation depend on β.

At the risk of being repetitive, we note that the performance guaranteed by this theorem is asymptotically network independent, as the only dependence on the spectral gap $1-\sigma$ is in the transient. The point of this theorem is to contrast Eq. (12) with Proposition (1). The two guarantees are identical, which implies that the distributed optimization method analyzed in Theorem 2 gives us a linear time speedup over the single-node rate using the same-step size. Likewise, Eq. (14) gives a network-independent bound (though, again, the size of the transient until it holds depends on the network), and can be thought of as a linear-time speedup over the corresponding single-node rate.

Such linear speedups are significant in that they provide a strong motivation for distributed optimization: one can claim that, over a network with n nodes, the distributed optimization is n times faster than a centralized one, at least provided t is large enough. Without such speed-ups, it is much more challenging to motivate the study of distributed optimization in the first place.

Note that the lower bound of Eq. (11) on the transient is not fully explicit, as t actually appears on both sides. However, for large enough t the inequality always holds. Informally, this is because t on the right-hand side is inside the logarithm. Slightly more formally, it is immediate that, if the bound of Eq. (11) fails to hold, then $\alpha(t) \leq C_1 t e^{-C_2 t}$ for all t where C_1, C_2 depend on $1-\sigma$,$\alpha_{\max}$, and C_α; and this would contradict the non-summability of $\alpha(t)$ in Assumption 1.

Finally, note that this theorem contains bounds on the performance achieved by $x'_\alpha(t)$, which a particular kind of running average across the iterates of all the nodes in he network. This can be tracked in a distributed way by the nodes if the number of iterations T to be performed is known ahead of time: each node can take a running average of its own iterates and, after T iterations have been performed, the nodes do a run of average consensus to compute the average of these running averages. The algorithm from Olshevsky (2017) takes $O(n\log(1/\epsilon))$ iterations to come ϵ-close to the average on any undirected n-node graph, so this incurs an extra cost which does not depend on T (moreover, that consensus method along with the method from Rebeschini and Tatikonda (2017) takes $O(D\log\frac{1}{\epsilon})$ steps on many common graphs of diameter D, e.g., grids). It is also possible to modify this idea to handle the case when the number of iterations to be done is not known ahead of time (e.g., each node can restart computation of the running average when the number of iterations is a power of 2) at the cost of losing of a constant multiplicative factor in the error $F(\cdot) - F^*$.

Finally, we remark that linear speedup theorems such as this one validate the idea of distributed optimization over an arbitrary graph G. One might also consider a hierarchical structure where nodes forward information along a spanning tree to a central coordinator which forwards back an average of the gradients. However, no variation on such a scheme is currently known to achieve network independence in this setting.

n	Graph	Lower bound on iterations to network independence with $t^{-3/4}$ step-size
10	Line	940
50	Line	577,842
100	Line	9,240,904
16	2DGrid	58
49	2Dgrid	558
100	2Dgrid	2373
16	MGG Expander	8
49	MGG Expander	16
100	MGG Expander	18
8	3Dgrid	4
64	3Dgrid	103
125	3Dgrid	259
8	Binomial Tree	343
64	Binomial Tree	50,540
128	Binomial Tree	224,529

Table 1: This table shows what the lower bound of Eq. (13) looks like for a variety of scenarios. MGG expander refers to the Margulis-Gabber-Galil expander, and all other graph names are standard. All graphs were generated using the NetworkX package: `http://networkx.org`. We used $L = 1, D = 1$ for simplicity. In each case, we used the so-called Metropolis matrix which puts $\max\{\deg(i) + 1, \deg(j) + 1\}^{-1}$ as W_{ij}, and sets the diagonal entries of W to make W stochastic. The results show a very strong dependence on the graph with quite reasonable scaling for the 3D grid. Unlike expanders which will have "long range" connections, the 3Dgrid can be implemented directly with only neighbor-to-neighbor communications in a cluster where the processors are stacked on top of each other. The code which produced the bounds in this table may be found on GitHub: `https://github.com/alexolshevsky/NetworkIndependenceSubgradient/blob/main/Network_Independence_Bounds.ipynb`

Unfortunately, the previous theorem does not apply to $\beta = 1/2$ which, as discussed earlier, is the best rate of decay for the subgradient method. In fact, our next theorem will show something quite different occurs when $\beta = 1/2$: we will construct a counterexample where the distributed method has network-dependent performance regardless of how large t is.

We next give a formal statement of this result. Our first step is to describe how we will choose the matrix W depending on the underlying graph. Let us adopt the convention that, given an undirected graph $G = (V, E)$ without self-loops, we will define the symmetric stochastic matrix $W_{G,\epsilon}$ as

$$[W_{G,\epsilon}]_{ij} = \begin{cases} \epsilon & (i,j) \in E \\ 0 & \text{else} \end{cases},$$

and we set diagonal entries $[W_G]_{ii}$ to whatever values result in a stochastic matrix. Clearly, ϵ should be strictly smaller than the largest degree in G.

We next define G'_n to be a graph on $2n$ nodes obtained as follows: two complete graphs on nodes $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ are joined by connecting each u_i to v_i (i.e., u_1 is connected to v_1, u_2 to v_2, and so forth). We note that because the largest degree in this graph is n, any ϵ used to construct a stochastic $W_{G,\epsilon}$ should be upper bounded by $1/n$.

Our second main result shows that when we run Eq. (9) on this graph G'_n, then with an appropriate choice of functions we will never obtain a performance independent ϵ^{-1}; and since, as we just, ϵ^{-1} grows as $\Omega(n)$, the performance will always scale with n no matter how long we wait.

Theorem 3 (Lack of Asymptotic Network Independence with $1/\sqrt{t}$ Step-Size) *Consider the distributed optimization method of Eq. (9) with*

- *The functions*

  $$f_i(x) = \gamma|x|,$$

  when $i \in \{u_1, \ldots, u_n\}$ *and*

  $$f_i(x) = \frac{1}{2}|x - 1|,$$

  for $i \in \{v_1, \ldots, v_n\}$
- *Step-size* $\alpha(t) = 1/\sqrt{t}$.
- *Constraint set* $\Omega = [-a, a]$.
- *Initial conditions* $x_i(0) = 0$.

Then, there exists a choice of the constants $\gamma > 1$ *and* a *(appearing in the above definitions), both independent of* n *or* ϵ*, such that, on the graph* G'_n *with* $n \geq 4$ *and any choice of* $\epsilon \leq 1/n$*, there exists an infinite sequence* $g_i(t)$ *such that the quantity* $x(t)$ *defined through Eq. (9) satisfies:*

- *For* $i \in \{v_1, \ldots, v_n\}$, $x_i(t) - x^*$ *is a nonnegative sequence that does not depend on* i *and satisfies*

  $$x_i - x^* = \Omega\left(\frac{\epsilon^{-1}}{\sqrt{t}}\right), \text{ for all } i \in \{v_1, \ldots, v_n\}, \tag{15}$$

  for all large enough t.

- $x_i(t) = x^*$ *for all* $i \in \{u_1, \ldots, u_n\}$ *and all* t.

To parse the statement of this theorem, note that there is some freedom to choose the subgradient of functions like $|x|$ at $|x-1|$ at zero and one, respectively. Consequently, the theorem has to assert that there is a choice of subgradients such that the solution of Eq. (9) has the desired behavior.

To put the guarantees of this theorem into context, observe that, as a consequence of Eq. (15), not only does the average $(1/n)\sum_{i=1}^n x_i(t) - x^*$ scale as $O(\epsilon^{-1}/\sqrt{t})$ but so does any convex combination of this quantity over the various t's (in particular, the quantities $x_\alpha(t)$ or $x'_\alpha(t)$ discussed earlier). It immediately follows that $F(\cdot) - F^*$ for all of these quantities also scales linearly with ϵ^{-1}, i.e.,

$$F(x_\alpha(t)) - F^* \geq \Omega\left(\frac{\epsilon^{-1}}{\sqrt{t}}\right) \tag{16}$$

$$F(x'_\alpha(t)) - F^* \geq \Omega\left(\frac{\epsilon^{-1}}{\sqrt{t}}\right) \tag{17}$$

Moreover, since $\epsilon \leq 1/n$, the performance of Eq. (9) with $1/\sqrt{t}$ step-size does not attain a speedup over the corresponding single-node rate. This is to be contrasted with Eq. (12) and Eq. (14) which, provided one waits long enough, attain a linear speedup over the single-node rate.

In fact, it is easy to see that, for the matrix $W_{G'_n}$, the inverse spectral gap will grow with ϵ^{-1}, so that the presence of ϵ^{-1} in Eq. (16) and Eq. (17) is equivalent to scaling with $(1-\sigma)^{-1}$. For completeness, we give a proof of this assertion later in this paper as Proposition 8.

The main result of this paper is the contrast between Theorems 2 and Theorem 3. In the former case, we have asymptotic network independence and a linear speedup whenever the step-size is $1/t^\beta$ when $\beta > 1/2$. Unfortunately, the latter theorem shows that setting $\beta = 1/2$ can ruin this.

We remark that the last theorem has some similarities with Eq. (75) of Neglia et al. (2020). That work considers a distributed optimization problem where the functions are linear, so the optimal solution is either plus or minus infinity. It then considers choosing these linear functions so that the gradient is exactly an eigenvector of the matrix W, and observes that the speed at which the resulting method will move towards plus or minus infinity will then depend on the spectrum of W. By contrast, this theorem consider a case when x^* exists and the distributed method converges to it.

Finally, we remark that a visual illustration of this is given in Section 5, and the interested reader can skip ahead to see what the difference in performance looks like in a simulation of this example between $1/\sqrt{t}$ step-size and $1/t^\beta$ step-size where $\beta \in (1/2, 1)$.

2.2 Organization of this paper

We give a proof of Theorem 2 in Section 3 and a proof of Theorem 3 in Section 4. A simulation of our counterexample to network independence with $1/\sqrt{t}$ step-size is given in Section 5, which shows what the contrast between the presence of absence of network independence looks like numerically.

Our results raise the possibility that while a more slowly decaying step-size might be better for the centralized subgradient method, the opposite might be true in the distributed case (note that the transient bound of Theorem 2 improves with higher β). We show that this kind of "step size inversion" does indeed occur on a class of randomly generated problems in Section 6. Finally, our concluding Section 7 mentions several open problems and future directions.

3. Proof of Theorem 2

In this section, we provide a proof of Theorem 2. Our first step is to rewrite Eq. (9) in a way that will be easier to analyze. We set $s(t)$ to be the "gradient mapping" defined as

$$s(t) := \frac{x(t) - P_\Omega\left[x(t) - \alpha(t)g(t)\right]}{\alpha(t)}$$

so that Eq. (9) can be written as

$$x(t+1) = W\left[x(t) - \alpha(t)s(t)\right] \tag{18}$$

Consistent with our previous notation, we will use $s_i(t)$ to denote the i'th row of the matix $s(t)$.

In this formulation, we no longer have to explicitly deal with the projection, which is incorporated into the definition of $s(t)$. It is well known that the quantity $s(t)$, which is typically known as the "gradient mapping" in the case where the functions are smooth, has some properties similar to the properties of the a subgradient. We have been unable to find a reference for the facts below; to our knowledge, in the current literature, various properties of $s(t)$ are only listed in the case where the functions $f_i(\cdot)$ are smooth, which is assumption we do not make here. The next lemma is our first statement to this effect, showing that $s_i(t)$ inherits any upper bound on $g_i(t)$; it is an immediate consequence of the non-expansiveness of the projection operator (Lemma 3.2.1 in Bertsekas (2015)), as the proof makes clear.

Lemma 4 *If* $||g_i(t)||_2 \leq L$ *then* $||s_i(t)||_2 \leq L$.

Proof We first observe that for all i, t we have that $x_i(t) \in \Omega$. Indeed, $x_i(t)$ is obtained as a convex combination of vectors projected onto Ω and so itself belongs to Ω by convexity. We then use this, along with the fact that projection onto convex sets is nonexansive, to argue that

$$\begin{aligned}
||s_i(t)||_2 &= \frac{||x_i(t) - P_\Omega\left[x_i(t) - \alpha(t)g_i(t)\right]||_2}{|\alpha(t)|} \\
&= \frac{||P_\Omega\left[x_i(t)\right] - P_\Omega\left[x_i(t) - \alpha(t)g_i(t)\right]||_2}{|\alpha(t)|} \\
&\leq \frac{||\alpha(t)g_i(t)||_2}{|\alpha(t)|} \\
&\leq ||g_i(t)||_2 \\
&\leq L.
\end{aligned}$$

■

Next, we note that it is standard that the subgradient $g_i(t)$ of the convex function $f_i(x)$ at $x_i(t)$ satisfies the relation

$$g_i(t)(x_i(t) - x^*)^T \geq f(x_i(t)) - f_i(x^*). \tag{19}$$

Our next lemma shows that $s_i(t)$ satisfies a similar inequality up to a "higher order" term. It is similar to Proposition 3.2.2 in Bertsekas (2015).

Lemma 5 *Under Assumption 3, we have that for all $i = 1, \ldots, n$,*

$$\alpha(t)s_i(t)(x_i(t) - x^*)^T \geq \alpha(t)(f(x_i(t)) - f_i(x^*)) - \frac{\alpha^2(t)}{2}L^2.$$

Note that whereas Eq. (19) does not contain the step-size $\alpha(t)$, Lemma 5 does. This is because the definition of the quantity $s_i(t)$ contained the step-size $\alpha(t)$ (unlike the subgradient $g_i(t)$ which is obviously defined independently of step-size).

Proof We start from the relation

$$x_i(t) - \alpha(t)s_i(t) = P_\Omega\left[x_i(t) - \alpha(t)g_i(t)\right],$$

which is just a rearrangement of the definition of $s(t)$. Our next step is to subtract x^* and take the squared Euclidean norm of both sides. On the left-hand side, we have

$$||x_i(t) - x^*||^2 - 2\alpha(t)s_i(t)(x_i(t) - x^*)^T + \alpha^2(t)||s_i(t)||_2^2.$$

On the right-hand side, we use the fact that projecting onto Ω cannot increase Euclidean distance from x^* to obtain an upper bound of

$$||x_i(t) - x^*||_2^2 - 2\alpha(t)g_i(t)(x_i(t) - x^*)^T + \alpha^2(t)||g_i(t)||_2^2$$

Putting these two facts together, we obtain the inequality

$$-2\alpha(t)s_i(t)(x_i(t) - x^*)^T \leq -2\alpha(t)g_i(t)(x_i(t) - x^*)^T + \alpha^2(t)||g_i(t)||_2^2$$

or

$$2\alpha(t)s_i(t)(x_i(t) - x^*)^T \geq 2\alpha(t)g_i(t)(x_i(t) - x^*)^T - \alpha^2(t)||g_i(t)||_2^2$$

Now using Assumption 3 and Eq. (19), we obtain

$$2\alpha(t)s_i(t)^T(x_i(t) - x^*) \geq 2\alpha(t)(f_i(x_i(t)) - f_i(x^*)) - \alpha^2(t)L^2,$$

which proves the lemma. ■

The final lemma we need bounds the distance between each $x_i(t)$ and the network-wide average $\overline{x}(t)$ as $O(\alpha(t))$ (where the constant inside this $O(\cdot)$-notation will depend on the matrix W). Such bounds are standard in the distributed optimization literature.

We introduce some new notation which we will find convenient to use. We will use $\mathbf{1}$ to denote the all-ones vector in $\mathbb{R}^n$, so that $\mathbf{1}\overline{x}(t)^T$ has the same dimensions as $x(t)$. We adopt the notation σ to denote the second-largest singular value of the matrix W; under Assumption 4, we have that $\sigma < 1$ while the largest singular value is 1 corresponding to the all-ones vector[2]. In the sequel, for a vector y we will use the inequality

$$||W(y - \overline{y})||_2^2 \leq \sigma^2 ||y - \overline{y}||_2^2 \leq \sigma^2 ||y||_2^2. \tag{20}$$

With these preliminaries in place, we have the following lemma, which uses a standard analysis distributed optimization updates (for a similar estimate see Theorem 8 of Nedić et al. (2018)).

Lemma 6 *Suppose Assumptions 1, 2, and 5 on the step-size, Assumption 3 on the functions, and Assumption 4 on the mixing matrix W all hold. When*

$$t \geq \Omega\left(\frac{1}{1-\sigma}\log\frac{(1-\sigma)t\alpha_{\max}}{C'_\alpha \alpha(t)}\right)$$

we have that

$$||x(t) - \mathbf{1}\overline{x}(t)^T||_F \leq \frac{2C'_\alpha \alpha(t) L\sqrt{n}}{1-\sigma}.$$

Proof Recall that, by assumption, the initial conditions are identical; let us denote them all by x_1. Thus starting from Eq. (18) we have that

$$x(t) = W^{t-1}x_1 - \alpha(1)W^{t-1}s(1) - \cdots - \alpha(t-1)Ws(t-1)$$

we can use the fact that multiplication by a doubly stochastic matrix doesn't affect the mean of a vector to obtain that

$$\overline{x}(t) = x_1 - \alpha(1)\overline{s}(1) - \cdots - \alpha(t-1)\overline{s}(t-1).$$

2. Formally, this is implied from Lemma 4 of Nedic et al. (2009). That lemma implies that $||Wx||_2 \leq ||x||_2$ and, under Assumption 4, equality holds if and only if x is a multiple of $\mathbf{1}$.

Next, using the "MATLAB notation" $[A]_{:,i}$ for the i'th column of a matrix A, we can apply Eq. (20) in the following sequence of equations:

$$\begin{aligned}
||x(t) - \mathbf{1}\overline{x}(t)^T||_F &\leq \sum_{k=1}^{t-1} \alpha(k)||W^{t-k}(s(k) - \mathbf{1}\overline{s}(k))^T||_F \\
&= \sum_{k=1}^{t-1} \alpha(k)\sqrt{\sum_{i=1}^{n} ||W^{t-k}[s(k) - \mathbf{1}\overline{s}(k)^T]_{:,i}||_2^2} \\
&\leq \sum_{k=1}^{t-1} \alpha(k)\sqrt{\sum_{i=1}^{n} \sigma^{2(t-k)}||[s(k)]_{:,i}||_2^2} \\
&\leq \sum_{k=1}^{t-1} \alpha(k)\sigma^{t-k}||s(k)||_F.
\end{aligned}$$

Let us break the last sum at $t' = t - \lceil t/2 \rceil$ and bound each of the two pieces separately. The first piece, over the range $t = 1, \ldots, t'$ is bounded simply using the fact that all subgradients are upper bounded by L in the Euclidean norm (and consequently $||s(t-k)||_F \leq L\sqrt{n}$); whereas the second piece, over the last $t/2$ steps, is upper bounded as a geometric sum. The result is

$$||x(t) - \mathbf{1}\overline{x}(t)^T||_F \leq \frac{t}{2}\alpha_{\max} L\sqrt{n}\sigma^{\lceil t/2 \rceil} + \frac{L\sqrt{n}}{1-\sigma}\alpha(\lfloor t/2 \rfloor).$$

We next use that $x^{1/(1-x)} \leq e^{-1}$ when $x \in [0,1]$ as well as Assumption 5 to obtain that

$$||x(t) - \mathbf{1}\overline{x}(t)^T||_F \leq t\alpha_{\max} L\sqrt{n}e^{-t(1-\sigma)/2} + \frac{L\sqrt{n}C'_\alpha \alpha(t)}{1-\sigma}.$$

When $t \geq \Omega((1-\sigma)^{-1}\log[(1-\sigma)(t\alpha_{\max}/(C'_\alpha \alpha(t))]$, the first term is upper bounded by the second and the lemma is proved.

■

With all these lemmas in place, we are now ready to give a proof of our first main result.

Proof [Proof of Theorem 2] Starting from Eq. (18) we obtain

$$\overline{x}(t+1) - x^* = \overline{x}(t) - \alpha(t)\overline{s}(t) - x^*$$

so that

$$\begin{aligned}
||\overline{x}(t+1)-x^*||_2^2 =&||\overline{x}(t)-x^*||_2^2+\alpha^2(t)||\overline{s}(t)||^2-2\alpha(t)\overline{s}(t)(\overline{x}(t)-x^*)^T\\
=&||\overline{x}(t)-x^*||_2^2+\alpha^2(t)||\overline{s}(t)||_2^2-2\alpha(t)\left(\frac{1}{n}\sum_{i=1}^n s_i(t)(\overline{x}(t)-x^*)^T\right)\\
=&||\overline{x}(t)-x^*||_2^2+\alpha^2(t)||\overline{s}(t)||_2^2-2\alpha(t)\left(\frac{1}{n}\sum_{i=1}^n s_i(t)(x_i(t)-x^*)^T\right)\\
&+2\alpha(t)\left(\frac{1}{n}\sum_{i=1}^n s_i(t)(x_i(t)-\overline{x}(t))^T\right)\\
\leq&||\overline{x}(t)-x^*||_2^2+\alpha^2(t)L^2-2\alpha(t)\frac{1}{n}\sum_{i=1}^n f_i(x_i(t))-f_i(x^*)\\
&+L^2\alpha^2(t)+2\alpha(t)\frac{1}{n}\sum_{i=1}^n L||x_i(t)-\overline{x}(t)||_2,
\end{aligned}$$

where, in the above sequence of inequalities, we used Lemma 4 to bound the norm of $||\overline{s}(t)||_2^2$, Lemma 5 to bound $s_i(t)^T(x_i(t)-x^*)$, and Cauchy-Schwarz in the very last step. Now using the fact that each $f_i(\cdot)$ is L-Lipschitz, which follows from Assumption 3, we have

$$\begin{aligned}
||\overline{x}(t+1)-x^*||_2^2 \leq ||\overline{x}(t)-x^*||_2^2+2\alpha^2(t)L^2-2\alpha(t)\frac{1}{n}\sum_{i=1}^n f_i(\overline{x}(t))-f_i(x^*)\\
+4\alpha(t)L\frac{1}{n}\sum_{i=1}^n||x_i(t)-\overline{x}(t)||_2. \qquad (21)
\end{aligned}$$

We next bound the very last term in the sequence of inequalities above. Our starting point is the observation that

$$\sum_{i=1}^n||x_i(t)-\overline{x}(t)||_2\leq\sqrt{n}||x(t)-\overline{x}(t)||_F,$$

which follows by an application of Cauchy-Schwarz. We then use Lemma 6 to bound the right-hand side. This yields that, for t large enough to satisfy the assumptions of that lemma,

$$||\overline{x}(t+1)-x^*||_2^2\leq||\overline{x}(t)-x^*||_2^2+2\alpha^2(t)L^2-2\alpha(t)\frac{1}{n}\sum_{i=1}^n f_i(\overline{x}(t))-f_i(x^*)+4\alpha(t)L\frac{2C'_\alpha\alpha(t)L}{1-\sigma},$$

implying that

$$2\alpha(t)\left[F(\overline{x}(t))-F^*\right]\leq||\overline{x}(t)-x^*||_2^2-||\overline{x}(t+1)-x^*||_2^2+2\alpha^2(t)L^2+8\alpha^2(t)\frac{C'_\alpha L^2}{1-\sigma},$$

As before, let $t'=\lfloor t/2\rfloor$. We sum the last inequality up frome time t' to time t to obtain

$$2\sum_{k=t'}^t\alpha(k)\left[F(\overline{x}(k))-F^*\right]\leq||\overline{x}(t')-x^*||_2^2+\frac{10C'_\alpha L^2}{1-\sigma}\sum_{k=t'}^t\alpha^2(k),$$

where we used that $C'_\alpha \geq 1$ (because $\alpha(t)$ is nonincreasing) and that $\sigma < 1$ to combine the terms involving $\alpha^2(t)$.

Dividing both sides by $2\sum_{k=t'}^{t}\alpha(t)$ and using convexity of $F(x)$, we obtain

$$F(\overline{x}_\alpha(t)) - F^* \leq \frac{||\overline{x}(t') - x^*||_2^2}{2\sum_{k=t'}^{t}\alpha(k)} + \frac{10C'_\alpha L^2}{1-\sigma}\frac{\sum_{k=t'}^{t}\alpha^2(k)}{2\sum_{k=t'}^{t}\alpha(k)} \tag{22}$$

Now because $\alpha(t)$ is square summable, we have that

$$\lim_{t\to\infty}\sum_{k=t'}^{t}\alpha^2(k) \leq \lim_{t\to\infty}\sum_{k=\lfloor t/2\rfloor}^{+\infty}\alpha^2(k) = 0.$$

In particular, Eq. (10) will eventually be satisfied, and when that happens, the second term of Eq. (22) will be upper bounded by the first. We will then have

$$F(\overline{x}_\alpha(t)) - F^* \leq \frac{D^2}{\sum_{k=t'}^{t}\alpha(k)}$$

The first part of the theorem, namely Eq. (12), now follows immediately.

Next, we suppose that $\alpha(k) = 1/k^\beta$ where $\beta \in (1/2, 1)$. This step-size satisfies all the assumptions we have made; however, starting from Eq. (22), we can write down some more effective bounds. Indeed, by the usual method of upper/bounding sums by the corresponding integrals, we have that

$$\begin{aligned}\sum_{k=t'}^{t}\frac{1}{k^\beta} \geq& \Omega\left(\left(1-\frac{1}{2^{-\beta+1}}\right)\frac{t^{-\beta+1}}{-\beta+1}\right) = \Omega_\beta\left(t^{-\beta+1}\right)\\ \sum_{k=t'}^{t}\left(\frac{1}{k^\beta}\right)^2 \leq& O\left(\left(1-\frac{1}{2^{-2\beta+1}}\right)\frac{t^{-2\beta+1}}{-2\beta+1}\right) = O_\beta\left(t^{-2\beta+1}\right),\end{aligned}$$

where the subscript of β denotes that the constant could depend on β. Plugging this into Eq. (22), we have

$$\begin{aligned}F(\overline{x}_\alpha(t)) - F^* &\leq O_\beta\left(\frac{D^2}{t^{1-\beta}} + \frac{L^2 t^{-2\beta+1}}{(1-\sigma)t^{-\beta+1}}\right)\\ &\leq O_\beta\left(\frac{D^2}{t^{1-\beta}} + \frac{L^2}{(1-\sigma)t^\beta}\right)\end{aligned} \tag{23}$$

Therefore when

$$t^{2\beta-1} \geq \Omega_\beta\left(\frac{L^2}{D^2(1-\sigma)}\right)$$

we have that the first term of Eq. (23) dominates and

$$F(\overline{x}_\alpha(t)) - F^* \leq O_\beta\left(\frac{D^2}{t^{1-\beta}}\right)$$

This proves Eq. (14) and the proof of the theorem is now complete. ■

4. Proof of Theorem 3

The proof below will analyze an explicit example of a graph where the dependence on spectral gap never disappears, no matter how large t is. The graph is G'_n, which is a graph on $2n$ vertices $\{u_1, \ldots, u_n\} \cup \{v_1, \ldots, v_n\}$ defined shortly before the statement of Theorem 3, and the underlying matrix W is $W = W_{G'_n,\epsilon}$, defined in the same place.

We begin with a brief description of the intuition underlying the counter-example. The high-level idea is that the performance of distributed subgradient descent can be thought of in terms of a recursion that moves towards the optimal solution, perturbed by some error due to network disagreement. Indeed, this is the form taken by the proof of Theorem 2: we wrote down a recursion satisfied by $F(\overline{x}(t)) - F^*$, and the updates of that recursion featured terms that depended on the network-wide disagreement $\sum_{i=1}^n ||x_i(t) - \mathbf{1}\overline{x}(t)||$ (see Eq. (21) above).

In the absence of any disagreement – i.e., if we could magically set $\sum_{i=1}^n ||x_i(t)-\mathbf{1}\overline{x}(t)|| = 0$ at every step – the recursion of Eq. (21) will converge to the optimal solution at a rate $F(\cdot) - F^* = O(1/\sqrt{t})$. Now if the disagreement decays faster than $O(1/\sqrt{t})$, it is intuitive that it has no effect on the ultimate convergence rate. On the other hand, if the disagreement decays as $\sim 1/\sqrt{t}$, then it may well dominate. Furthermore, it should also be intuitive that the term $\sum_{i=1}^n ||x_i(t) - \mathbf{1}\overline{x}(t)||$ can only be bounded in terms of the spectral gap (since, in effect, this term measures how successful repeated multiplication by the matrix W is in driving all the nodes closer together), and this way the spectral gap will appear in the final performance of the method.

In short, we need to argue that the recursion of Eq. (21) is, in some sense, "tight" when the step-size is $1/\sqrt{t}$. Unfortunately, we know of no pleasant way to make this argument. The only way we are able to do this is to come up with an example where we can write down an explicit formula for the solution of Eq. (6) at any time t, and this is in effect what we do below on the graph G'_n. The graph G'_n is particularly well-suited for the purpose because it is symmetric: we can write down a solution where all $x_{u_i}(t)$ have the same value, and all $x_{v_i}(t)$ have the same value. Unfortunately, even in this simple case, the argument turns out to be somewhat involved.

We next turn to the proof itself. The key ingredient will be the following technical lemma.

Lemma 7 *Consider the update rule determined by $y(1) = 0$ and*

$$y(t+1) = (1-\epsilon)\, y(t) - \frac{(1/2)(1-\epsilon)\text{sign}(y(t)-1) + \epsilon\Delta(t)}{\sqrt{t}}, \tag{24}$$

where

$$\Delta(t) = \frac{\epsilon\sqrt{t}y(t) - (\epsilon/2)\text{sign}(y(t)-1)}{1-\epsilon}.$$

Here $\text{sign}(x)$ *is the function which returns* 1 *when* $x \geq 0$ *and* -1 *otherwise.*

Then when $\epsilon \in (0, 1/4]$, we have that $y(t) \in [0, 2]$ for all t; and furthermore,

$$y(t) \leq \frac{2\epsilon^{-1}}{\sqrt{t}}, \tag{25}$$

for all t. Finally, for all t larger than some t_1, we also have

$$y(t) \geq \frac{\epsilon^{-1}}{16\sqrt{t}}. \tag{26}$$

We postpone the proof of this lemma for now, as we think the reader will be more interested in it once it is seen how the recursion of Eq. (24) naturally appears in the analysis of distributed subgradient descent on the graph G'_n. Thus, we will first give a proof of Theorem 3 which relies on this lemma, and then we will go back and supply a proof of the lemma.

Proof [Proof of Theorem 3] We argue that

$$\begin{cases} x_i(t) = 0 & i \in \{u_1, \ldots, u_n\} \\ x_i(t) = y(t) & i \in \{v_1, \ldots, v_n\} \end{cases} \tag{27}$$

is a valid trajectory of Eq. (6) with $W = W_{G'_n,\epsilon}$ where, recall, $W_{G'_n,\epsilon}$ is defined shortly before the statement of Theorem 3. Here $y(t)$ is defined in the statement of Lemma 7 and by "valid trajectory" we mean that there exists a sequence of vectors $g_i(t)$, with $g_i(t)$ being a valid subgradient of $f_i(x)$ at $x_i(t)$, resulting in our main update of Eq. (6) taking the values specified in Eq. (27) for all t.

Our proof of this, given next, will depend on a particular choice of the constants $\gamma > 1$ and a from the statement of Theorem 3; these constants will be defined in the course of the proof. Once the validity of Eq. (27) is proved, Theorem 3 is immediate, conditional on Lemma 7. Indeed, it is easy to see that because $\gamma > 1 > 1/2$, we have that $x^* = 0$; and Eq. (26) then provides the lower bound claimed in statement of Theorem 3.

Our proof of the validity of Eq. (27) is by induction. At time $t = 1$, we just have $x_i(t) = 0$, so there is nothing to prove. Suppose Eq. (27) is a valid trajectory over times $1, \ldots, t$, and let us consider time $t + 1$. Our first step is to argue that, for an appropriately large choice of the constant a, we can simply omit the projection step in Eq. (6).

Indeed, observe that by definition of the functions $f_i(\cdot)$ (see statement of Theorem 3), we have that the subgradients are in $[-\gamma, \gamma]$ for $i \in \{u_1, \ldots, u_n\}$, and in $[-1/2, 1/2]$ for $i \in \{v_1, \ldots, v_n\}$; and recall that later we will specify $\gamma > 1$. Using Lemma 7, which asserts that $y(t) \in [0, 2]$ for all t, we have that $x_i(t) \in [0, 2]$ for all i, t; and what follows from all this that $x_i(t) - \alpha(t)g_i(t) \in [-\gamma, 2 + \gamma]$ for all i, t.

To be able to omit the projection step from Eq. (9), it suffices to have all $x_i(t) - \alpha(t)g_i(t)$ be in the interior of $[-a, a]$. But since we have just argued that $x_i(t) - \alpha(t)g_i(t) \in [-\gamma, 2+\gamma]$ for all i, t, we see that it suffices to choose e.g., $a = 3 + \gamma$.

The update of Eq. (6) then becomes

$$x(t + 1) = W_{G'_n,\epsilon} \left[x(t) - \alpha(t)g(t) \right].$$

Our next step is to work out what this gives for nodes $i \in \{u_1, \ldots, u_n\}$ given the particular form of $W_{G'_n,\epsilon}$. Since all $g_{u_i}(t), i = 1, \ldots, n$ are subgradients of the same function evaluated

at zero, we will be considering the case when they are the same. In that case, we have that

$$\begin{aligned} x_{u_i}(t+1) &= (1-n\epsilon)x_{u_i}(t) + (n-1)\epsilon u_i(t) + \epsilon x_{v_i}(t) - \alpha(t)[W_{G'_n,\epsilon}g(t)]_{u_i} \\ &= x_{u_i}(t) + \epsilon\left(x_{v_i}(t) - x_{u_i}(t)\right) - \frac{(1-\epsilon)g_{u_i}(t) + \epsilon g_{v_i}(t)}{\sqrt{t}}. \end{aligned}$$

Multiplying both sides by $\sqrt{t}$ and using that $x_{v_i}(t) = y(t), x_{u_i}(t) = 0$ by the inductive hypothesis, we obtain

$$\sqrt{t}x_{u_i}(t+1) = \epsilon\sqrt{t}y(t) - (1-\epsilon)g_{u_i}(t) - \epsilon g_{v_i}(t).$$

In order to show that Eq. (27) holds at time $t+1$ for the node u_i, we need to have $x_{u_i}(t+1) = 0$. The last equation allows us to see what is needed for this to be the case; setting the left-hand side to zero, we obtain

$$g_{u_i}(t) = (1-\epsilon)^{-1}\left(\epsilon\sqrt{t}y(t) - \epsilon g_{v_i}(t)\right). \tag{28}$$

Our argument shows that as long we "select" this number as the local subgradient chosen by all the nodes u_i at time t, then Eq. (27) holds at time $t+1$ for the nodes u_i. But is this number a valid choice of subgradient for all the functions $f_i(\cdot)$ at the point 0?

Observe that, by Lemma 7, we have that $y(t) \leq 2\epsilon^{-1}/\sqrt{t}$ for all t. Consequently, $\epsilon\sqrt{t}y(t) \in [0,2]$. Since $g_{v_i}(t) \in [-1/2, 1/2]$ and $\epsilon \leq 1/4$ (because we assumed $n \geq 4$ and $\epsilon \leq 1/n$), we have that the expression in parenthesis on right-hand side of Eq. (28) lies between $-1/2$ and $17/8$. Using $\epsilon \leq 1/4$ again, we see that the right-hand side of Eq. (28) is at most $17/6$ in absolute value. Thus, as long as we choose γ sufficiently big, e.g., $\gamma = 3$, the choice of $g_{u_i}(t)$ in Eq. (28) is a valid subgradient.

To recap, we have just shown that Eq. (27) holds at time $t+1$ for at least for the nodes u_i by specifying what the local subgradients at nodes u_i should be at time t. For this choice of subgradient to be valid, we had to choose $\gamma = 3$ and also $a = 3 + \gamma = 6$. We next argue that, with these choices of γ, a and $g_{u_i}(t)$, we can also choose a number to return as the local subgradient of all the nodes v_i such that Eq. (27) holds at time $t+1$ for the nodes v_i as well.

Indeed, by induction we have that for $i \in \{v_1, \ldots, v_n\}$,

$$x_{v_i}(t+1) = x_{v_i}(t) + \epsilon(0 - x_{v_i}(t)) - \frac{(1-\epsilon)g_{v_i}(t) + \epsilon g_{u_i}(t)}{\sqrt{t}}, \tag{29}$$

where we used that $x_{u_i}(t) = 0$ by the inductive hypothesis, and $x_{v_i}(t) = x_{v_j}(t)$ for all i, j, also by the inductive hypothesis. We want to show that there is a choice of $g_{v_i}(t)$ that turns the left-hand side into $y(t+1)$. But the choice $g_{v_i}(t) = (1/2)\text{sign}(y(t) - 1)$ is valid and turns the pair of Eq. (29 and Eq. (28) into exactly Eq. (24), so it certainly results in $x_{v_i}(t+1) = y(t+1)$.

To summarize, we have shown how to choose valid subgradients at each step so that Eq. (9) turns into the recursion relation satisfied by Eq. (27). The proof is now complete. ■

It remains to prove Lemma 7.

Proof [Proof of Lemma 7] We first argue that $y(t) \in [0,2]$ for all t. This will follow from the following three assertions.

First: that $y(t)$ decreases whenever it is above one. Indeed, from Eq. (24), we have that whenever $y(t) \geq 1$,

$$y(t+1) \leq (1-\epsilon)y(t) + \frac{\epsilon^2/2}{1-\epsilon} = y(t) - \epsilon\left(y(t) - \frac{\epsilon/2}{1-\epsilon}\right).$$

Since we have assumed $\epsilon \in (0, 1/4]$ and $y(t) \geq 1$, the expression in parenthesis is positive and $y(t+1) < y(t)$.

Second: that $y(t)$ cannot decrease below zero. Indeed, if $y(t) \in [0,1)$, then from Eq. (24),

$$\begin{aligned} y(t+1) &\geq (1-\epsilon)y(t) + \frac{(1-\epsilon)/2}{\sqrt{t}} - \frac{\epsilon^2\sqrt{t}y(t)}{\sqrt{t}(1-\epsilon)} - \frac{\epsilon^2/2}{(1-\epsilon)\sqrt{t}} \\ &= \left(1 - \epsilon - \frac{\epsilon^2}{1-\epsilon}\right)y(t) + \frac{(1-\epsilon)/2}{\sqrt{t}} - \frac{\epsilon^2/2}{(1-\epsilon)\sqrt{t}} \\ &\geq 0, \end{aligned}$$

where the last step follows because $\epsilon \leq 1/4$ implies that

$$1 - \epsilon - \frac{\epsilon^2}{1-\epsilon} \geq 0$$

and

$$\frac{1-\epsilon}{2} \geq \frac{\epsilon^2/2}{1-\epsilon}.$$

On the other hand, if $y(t) \geq 1$, then

$$\begin{aligned} y(t+1) &\geq (1-\epsilon)y(t) - \frac{(1-\epsilon)/2}{\sqrt{t}} - \frac{\epsilon^2\sqrt{t}y(t)}{\sqrt{t}(1-\epsilon)} \\ &= \left(1 - \epsilon - \frac{\epsilon^2}{1-\epsilon}\right)y(t) - \frac{(1-\epsilon)/2}{\sqrt{t}} \\ &\geq \frac{1}{2}y(t) - \frac{1/2}{\sqrt{t}} \\ &\geq \frac{1}{2} - \frac{1}{2} = 0, \end{aligned}$$

where we used that $\epsilon \leq 1/4$ again. Thus, regardless of whether $y(t) \in [0,1)$ or $y(t) \geq 1$, we have that $y(t+1) \geq 0$.

Third: that if $y(t) \in [0,1)$, then the increase to the next step is at most $1/(2\sqrt{t})$. Indeed, for $y(t) \in [0,1)$ we have from Eq. (24),

$$\begin{aligned} y(t+1) &\leq \left(1 - \epsilon - \frac{\epsilon^2}{1-\epsilon}\right)y(t) + \frac{(1-\epsilon)/2}{\sqrt{t}} \\ &\leq y(t) + \frac{1}{2\sqrt{t}} \end{aligned}$$

Putting the three assertions above together, we obtain that $y(t) \in [0, 2]$ for all t.

We next prove that $y(t)$ decays as $O(\epsilon^{-1}/\sqrt{t})$. Indeed, since $|\text{sign}(y(t) - 1)| \leq 1$, we have that,

$$y(t+1) \leq \left(1 - \epsilon - \frac{\epsilon^2}{1-\epsilon}\right) y(t) + \frac{1}{2\sqrt{t}}.$$

Defining $z(t)$ via

$$z(t+1) = (1-\epsilon)z(t) + \frac{1}{2\sqrt{t}}, \tag{30}$$

with $z(1) = 0$, we have that $y(t) \leq z(t)$. To prove an upper bound on $y(t)$, we simply need to establish the same upper bound for $z(t)$.

First, we argue that

$$z(t) \leq \sum_{k=1}^{t-1} \frac{1}{2\sqrt{k}} = \frac{1}{2} + \sum_{k=2}^{t-1} \frac{1}{2\sqrt{k}} \leq \frac{1}{2} + \int_1^{t-1} \frac{1}{2\sqrt{u}}\, du \leq \sqrt{t} \tag{31}$$

Next, we multiply both sides of Eq. (30) by $\sqrt{t+1}$ to obtain

$$\sqrt{t+1}z(t+1) = (1-\epsilon)\sqrt{t+1}z(t) + \frac{1}{2}\sqrt{\frac{t+1}{t}},$$

and now using concavity of square root we obtain

$$\sqrt{t+1}z(t+1) \leq (1-\epsilon)\left(\sqrt{t} + \frac{1}{2\sqrt{t}}\right) z(t) + \frac{1}{2}\sqrt{\frac{t+1}{t}}.$$

Using Eq. (31), this implies that

$$\sqrt{t+1}z(t+1) \leq (1-\epsilon)\sqrt{t}z(t) + 2,$$

which gives that $\sqrt{t}z(t) \leq 2/\epsilon$. This proves that $y(t) \leq 2\epsilon^{-1}/\sqrt{t}$ and concludes the proof of Eq. (25).

It only remains to prove Eq. (26). Since, from Eq. (25) which we have just established, we know that $y(t) \to 0$, it follows that, for all t larger than some t_0, $\text{sign}(y(t) - 1) = -1$; and therefore, for such t,

$$y(t+1) = \left(1 - \epsilon - \frac{\epsilon^2}{1-\epsilon}\right) y(t) + \frac{(1-\epsilon) - \epsilon^2/(1-\epsilon)}{2\sqrt{t}},$$

which, because $\epsilon \in (0, 1/4]$, implies that

$$y(t+1) \geq (1-2\epsilon)y(t) + \frac{1}{4\sqrt{t}}.$$

This means that for all $t \geq t_0 + 1$

$$y(t) \geq \sum_{k=t_0}^{t-1} \frac{1}{4\sqrt{k}}(1-2\epsilon)^{t-(k+1)} \geq \frac{1}{4\sqrt{t}} \sum_{k=t_0}^{t-1} (1-2\epsilon)^{t-(k+1)}.$$

As $t \to \infty$, the geometric sum in the final term above approaches $1/(2\epsilon)$. It follows that, when t is bigger than some t_1, it is larger than half of that, so that

$$y(t) \geq \frac{\epsilon^{-1}}{16\sqrt{t}}.$$

This completes the proof. ■

Finally, we put Theorem 3 into context by arguing that the inverse spectral gap of the matrix $W_{G'_n,\epsilon}$ is $\Theta\left(\epsilon^{-1}\right)$ when ϵ is small enough. This explains that scaling with ϵ^{-1} is the same scaling with the inverse spectral gap of $W_{G'_n,\epsilon}$.

Proposition 8 *Let $1 = \lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of the matrix $W_{G'_n,\epsilon}$ arranged in descending order (because $W_{G'_n,\epsilon}$ is symmetric, its eigenvalues are real). Then*

$$\lambda_2(G_n) = 1 - 2\epsilon.$$

Proof The main idea is that it is quite easy to diagonalize $W_{G'_n,\epsilon}$ explicitly. Indeed, let L_{K_n} be the Laplacian of the complete graph on n vertices and let us adopt the notation I_n for the $n \times n$ identity matrix. Then it is immediate that

$$W_{G_n,\epsilon} = I_{2n} - \epsilon \begin{pmatrix} L_{K_n} & 0 \\ 0 & L_{K_n} \end{pmatrix} - \epsilon \begin{pmatrix} I_n & -I_n \\ -I_n & I_n \end{pmatrix}$$

Now it is well known how to diagonalize the Laplacian of the complete graph: $L_{K_n}\mathbf{1} = 0, L_{K_n}x = nx$ for any vector x orthogonal to $\mathbf{1}$. It follows that the eigenvectors of $W_{G'_n,\epsilon}$ can be written out as follows. Pick $n-1$ linearly independent vectors $x_1, \ldots, x_{n-1}$ whose entries sum to zero, and observe that the collection of vectors $\{[x_i, x_i]^T, [x_i, -x_i]^T \mid i = 1, \ldots, n-1\} \cup \{[\mathbf{1}, \mathbf{1}]^T, [\mathbf{1}, -\mathbf{1}]^T\}$ is an orthogonal basis for $\mathbb{R}^{2n}$.

We next verify that all of these vectors are in fact eigenvectors of $W_{G_n,\epsilon}$. Indeed:

$$W_{G'_n,\epsilon} \begin{pmatrix} \mathbf{1} \\ \mathbf{1} \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ \mathbf{1} \end{pmatrix},$$

$$W_{G'_n,\epsilon} \begin{pmatrix} \mathbf{1} \\ -\mathbf{1} \end{pmatrix} = (1 - 2\epsilon) \begin{pmatrix} \mathbf{1} \\ -\mathbf{1} \end{pmatrix},$$

and for any x orthogonal to the all-ones vector,

$$W_{G'_n,\epsilon} \begin{pmatrix} x \\ x \end{pmatrix} = (1 - n\epsilon) \begin{pmatrix} x \\ x \end{pmatrix},$$

while

$$W_{G'_n,\epsilon} \begin{pmatrix} x \\ -x \end{pmatrix} = (1 - n\epsilon - 2\epsilon) \begin{pmatrix} x \\ -x \end{pmatrix}.$$

Thus the set of eigenvalues of $W_{G'_n,\epsilon}$ is $\{1, 1-2\epsilon, 1-n\epsilon, 1-(n+2)\epsilon\}$. This completes the proof. ■

5. What does network independence look like? Simulating the counterexample.

We next give a numerical illustration of what network independence looks like. Specifically, we simulate the example constructed in the proof of Theorem 3. Our main purpose in doing so is to show how network independence, and its lack, are very stark phenomena that can be instantly "read off" the simulation results.

First, we found that our choice of γ and a in the course of the proof were somewhat conservative; numerically, we find that we can choose the slightly smaller values $\gamma = 2$ and $a = 5$. We simulated the step-size of $1/t^\beta$ for two choices of β. Specifically, the step-size choice of $\beta = 1/2$ is shown in Figure 1a while the choice of $\beta = 3/4$ is shown in Figure 1b. Each simulation shows three different values of n.

It is crucial to note that *the y-axis of both figures shows* $t^{1-\beta}(F(\overline{x}(t) - F^*)$. Intuitively, choosing a step-size of $\alpha(t) = 1/t^\beta$ will result in error $F(x(t)) - F^*$ that decays like $1/t^{1-\beta}$. What we are interested is whether the constant in front of this depends on the network, so we rescale the error to make this clear.

Comparing Figure 1a and Figure 1b illustrates our main result. In Figure 1a, we have argued in Theorem 3 that network independence does not occur. In particular, the underlying network has spectral gap that grows linearly with n (see Proposition 8) and, as a result, the performance of the method ultimately behaves like $\sim n/\sqrt{t}$. Thus the effect of n is never forgotten.

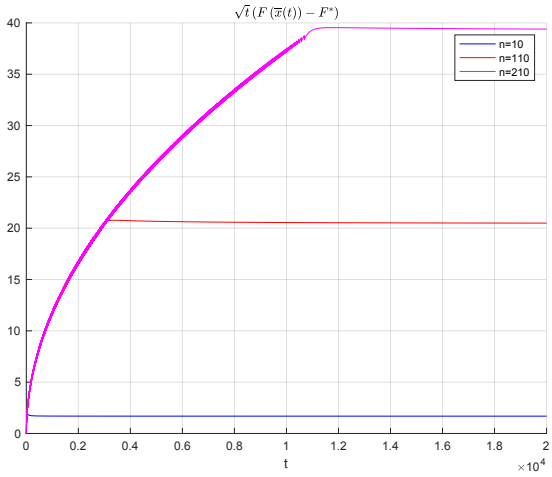
On the other hand, in Figure 1b, we see a peak whose height/length may depend on the spectral gap, and on n, in some fashion; but eventually, every curve drops below 1. In other words, we have that, eventually, $t^{1-\beta}(F(\overline{x}(t) - F^*) \leq 1$. We see that, after a transient period, the performance satisfies an $O(t^{-(1-\beta)})$ decay bound that does not depend on n or the spectral gap. Of course, here one must include the usual caveat that the size of the transient until this happens will depend on the spectral gap, and thus on n in graph families where the spectral gap depends on n.

6. Step-size inversion

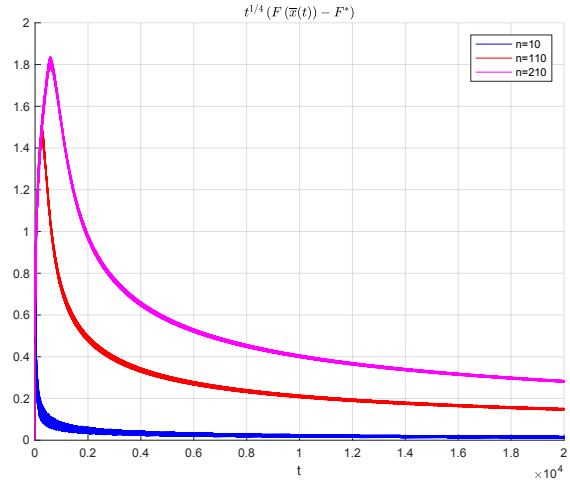
The contrast between Theorems 2 and 3 raises an interesting question: could it help in practice to choose a more slowly decaying step-size in distributed optimization?

Asymptotically, the answer is clearly no. Choosing a step-size that decays like $1/t^\beta, \beta \in [1/2, 1)$ results in an error $F(\cdot) - F^*$ that decays like $1/t^{1-\beta}$, so a lower β is better if we wait long enough. But this is an asymptotic statement, and it says nothing about what happens in practice when we count iterations until we are close to the optimal solution.

Informally, we may think of the distributed subgradient method as having two sources of error: the first kind, which is error that comes from the asymptotic convergence rate; and the second kind, which is the error that comes purely from the network disagreement effects. Theorems 2 and Theorem 3 suggest the second source of error decays faster for larger β (this is why the transient size given in Theorem 2 decreses with larger β). Even though asymptotically, the first source of error dominates, if we only count iterations until we get to a certain neighborhood of the optimal solution, it might be that the second kind of error dominates in such a "non-asymptotic" experiment.



(a) Step-size $\alpha(t) = 1/\sqrt{t}$.



(b) Step-size $\alpha(t) = 1/t^{3/4}$

Figure 1: Simulation of the counter-example constructed in the proof of Theorem 3 for two different choices of step-size. The figure on the right is consistent with asymptotic network independence, while the figure on the left is consistent with its absence.

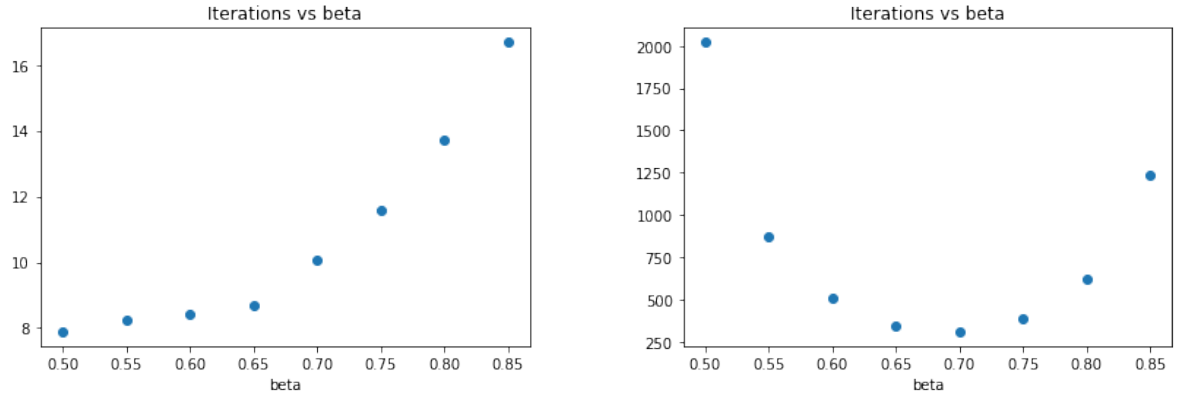
We next give evidence that this is sometimes the case. Specifically, we describe a class of problems with random data, and simulate the centralized subgradient method; as expected, performance (measured as number of iterations until the gradient mapping $s(t)$ is small) gets worse as we increase β in step-size $1/t^\beta$. We then simulate the distributed subgradient method and observe a more complicated relationship between step-size and performance; in particular, performance could improve as we increase β over some range of values. We call this phenomenon step-size inversion (since the effect of increasing β could be opposite for centralized and distributed methods).

Our starting point is that we want to consider a sufficiently simple class of problems to which it makes sense to apply the subgradient method. Thus, the problem should be non-smooth. On the other hand, it should not be something so simple as a quadratic with an $||\cdot||_1$ regularizer, as that can be solved via proximal methods. A natural choice is to consider problems with an "elastic net" regularization Zou and Hastie (2005), which a sum $||\cdot||_1$ and $||\cdot||_2$ regularizers. For our objective, we consider a regression-like problem where squares are replaces by fourth-powers, so that the ultimate objective is

$$F(\theta) = \frac{1}{K}\sum_{i=1}^{K} \lambda_0 ||a_i^T\theta - b_i||_2^4 + \lambda_1||\theta||_2^2 + \lambda_2||\theta||_1.$$

The logic of using fourth powers, or any exponent higher than 2, is that the resulting regression problem is very sensitive to large deviations – at the cost of being less sensitive to small errors. It also makes the problem more difficult for the subgradient method (and if the exponents were quadratic, there would actually be no good reason to solve this problem with a subgradient method in the first place).

We generate the matrix A which stacks up the vectors a_i to be Gaussian with unit variance; each a_i belongs to $\mathbb{R}^2$. We set $b_i = a_i^T\mathbf{1} + w_i$, where w_i is a Gaussian random



(a) Centralized subgradient method

(b) Distributed subgradient method

Figure 2: The relationship between the step-size $1/t^\beta$ and the number of iterations needed until the gradient mapping $s(t)$ is small. Note that, for the distributed method, the number of iterations we are plotting the first t in Eq. (6) until the termination condition $||s(t)||_1 < 0.03$ is met. For the centralized method, each iteration consists of computing the gradient of the full function $F(t)$. In other words, a single "iteration" of the centralized method has the same cost in terms of the number of gradient computations as a single iteration of the distributed "method" (both involve computing all n subgradients of the functions $f_1(\cdot), \dots f_n(\cdot)$). The code which produced this figure can be found on GitHub: `https://github.com/alexolshevsky/NetworkIndependenceSubgradient/blob/main/Step_size_inversion.ipynb`

variable of variance 4. We chose $\lambda_0 = 1/40, \lambda_1 = 1, \lambda_2 = 1/20$. We perform the optimization over the unit square $\Omega = [0, 1]^2$.

The results are shown in Figure 2. We use $n = 9$ agents on the 2D grid graph. The two figures plot β vs the number of iterations. Each data point is an average over 1,000 runs; we terminate when[3] $||s(t)||_1 < 0.03$. A slight difficulty comes from the fact that we have considerable freedom to choose the subgradient of $||\cdot||_1$ at the origin and, somewhat problematically, round-off error will preclude any component of $s(t)$ from being zero exactly. Moreover, choosing a subgradient has to be done carefully: it is in principle possible that, when the algorithm is at the optimal point, it chooses a nonzero subgradient and moves away from it. Since our termination condition is achieving a sufficiently small subgradient, a natural approach is to choose $s(t)$ so that $||s(t)||_1$ to be the smallest possible among all possible choices of subgradient. Furthermore, to account for round-off error, we treat the entries of $s(t)$ in a small enough interval around zero as indistinguishable from zero.

Note further that a small subgradient 1-norm means we are close to the optimal solution due to the compactness of the set Ω. Indeed, suppose we terminate at some point θ_t with $||\tilde{\nabla} F(\theta_t)||_1 \leq \Delta$. We then have by convexity for any feasible $\theta \in \Omega$,

$$F(\theta) \geq F(\theta_t) + \tilde{\nabla} F(\theta_t)^T (\theta - \theta_t),$$

so that

$$F(\theta) \geq F(\theta_t) - ||\tilde{\nabla} F(\theta_t)||_1 ||\theta - \theta_t||_\infty \geq F(\theta_t) - \Delta,$$

where we used that Ω is the unit square. Thus $F(\theta_t)$ is indeed close to $\min_{\theta \in \Omega} F(\theta)$.

Of course, for some functions it may be the case that a small optimality gap is achieved without a small subgradient. e.g., $f(x) = |x|$ when x is close to zero but nonzero. Nevertheless, since a small subgradient always implies a small optimality gap over a compact set, it is a natural choice for a stopping criterion.

Finally, note that the minimum norm subgradient here is used only as a stopping criterion: since we are solving an optimization problem whose solution we do not know ahead of time, to compare the convergence time of two methods (cenralized vs distributed) we compare the time until a minimum norm gradient at a running average is small. The algorithm update itself does not attempt to minimize the norm of the subgradient.

Glancing at Figure 2a, we see that the performance of the centralized subgradient method is as expected: higher β leads to slower performance. On the other hand, the behavior of the distributed method in Figure 2b is more complicated. We do see that it may be possible for one β to be better than another in the centralized case, only to have this flip in the distributed case, which is the "step-size" inversion from the title of this section.

We end this discussion with two caveats. First, this behavior is not true for all practical examples; our point is that step-size inversion *can* happen in measures of average-case performance, which is something that is hinted by our results but has not been observed before to our knowledge. In our simulations, step-size inversion seems to be tied to the oscillations of the subgradient when the optimal solution occurs at a point of non-smoothness (i.e., when some component of θ^* equals zero in the above example). This is how the parameter values described above were chosen. We wanted to choose values that resulted in occasional solutions with components close to zero, but we didn't want this to happen

3. This threshold was chosen to make the simulation converge in a reasonable amount of time.

too often (as then the simulation would take too long, as these are exactly the cases when it takes a very long time converge), which is why we chose $\beta = 0.1$. Similarly, the choice of 0.03 for the 1-norm of $s(t)$ in the termination condition was picked to achieve a reasonable running time.

7. Conclusion

Our goal was to understand when one can obtain network independence and a linear speedup in the distributed subgradient method when compared to its centralized counterpart. We showed that this is possible when the step-size decays like $1/t^\beta$ when $\beta > 1/2$, but not when $\beta = 1/2$. The bounds we derived on the transient time until the linear speedup kicks in increased as β decreased, suggesting that it might be the case that faster-decaying step-sizes are better in the distributed case, even when the opposite is true in the centralized case. We simulated one class of problems with random data where this was indeed the case for a range of step-sizes.

Our results point to a number of open questions. First, we have no theory to explain the performance we see in Figure 2b. Second, it would be interesting to conduct a large scale computational study across many problems of interest to see if the step-size inversion phenomenon holds more broadly than was demonstrated here. Since this is a primarily theoretical paper, it is out of scope for the present work.

Most importantly, it would be interesting to examine whether a linear speedup can be obtained for the optimal step-size decay $\beta = 1/2$ using a different algorithm (in the standard model of distributed optimization where a single message exchange and a single subgradient computation are possible at each node at each step). We are not aware of any lower bounds ruling this out, nor any algorithms known to achieve this.

References

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353, 2019.

Dimitri Bertsekas. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *Lecture Notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.

Jianshu Chen and Ali H Sayed. On the learning behavior of adaptive networks—part ii: Performance analysis. *IEEE Transactions on Information Theory*, 61(6):3518–3548, 2015a.

Jianshu Chen and Ali H Sayed. On the learning behavior of adaptive networks—part i: Transient analysis. *IEEE Transactions on Information Theory*, 61(6):3487–3517, 2015b.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2011.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. *Advances in Neural Information Processing Systems*, 32:11082–11094, 2019.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. An accelerated decentralized stochastic proximal algorithm for finite sums. In *Advances in Neural Information Processing Systems*, pages 954–964, 2019.

Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, pages 2525–2536, 2018.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pages 3478–3487, 2019.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pages 3043–3052, 2018.

Gemma Morral, Pascal Bianchi, and Gersende Fort. Success and failure of adaptation-diffusion algorithms with decaying step size in multiagent networks. *IEEE Transactions on Signal Processing*, 65(11):2798–2813, 2017.

Parvin Nazari, Davoud Ataee Tarzanagh, and George Michailidis. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Angelia Nedic, Alex Olshevsky, Asuman Ozdaglar, and John N Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

Angelia Nedic, Asuman Ozdaglar, and Pablo A Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4): 922–938, 2010.

Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

Giovanni Neglia, Gianmarco Calbi, Don Towsley, and Gayane Vardoyan. The role of network topology for distributed machine learning. In *INFOCOM- the IEEE Conference on Computer Communications*, pages 2350–2358. IEEE, 2019.

Giovanni Neglia, Chuan Xu, Don Towsley, and Gianmarco Calbi. Decentralized gradient methods: does topology matter? In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

Alex Olshevsky. Linear time average consensus and distributed optimization on fixed graphs. *SIAM Journal on Control and Optimization*, 55(6):3990–4014, 2017.

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.

Patrick Rebeschini and Sekhar C Tatikonda. Accelerated consensus via min-sum splitting. In *Advances in Neural Information Processing Systems*, pages 1374–1384, 2017.

Dominic Richards, Patrick Rebeschini, and Lorenzo Rosasco. Decentralised learning with random features and distributed gradient descent. In *International Conference on Machine Learning*, 2020.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2740–2749, 2018.

Artin Spiridonoff, Alex Olshevsky, and Ioannis Ch Paschalidis. Robust asynchronous stochastic gradient-push: Asymptotically optimal and network-independent performance for strongly convex functions. *Journal of Machine Learning Research*, 21(58):1–47, 2020.

Sebastian Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2019.

Sebastian U Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2018.

Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pages 4848–4856, 2018.

Zaid J Towfic, Jianshu Chen, and Ali H Sayed. Excess-risk of distributed stochastic learners. *IEEE Transactions on Information Theory*, 62(10):5753–5785, 2016.

Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. SlowMo: Improving communication-efficient distributed sgd with slow momentum. In *International Conference on Learning Representations*, 2019.

Minghui Zhu and Sonia Martínez. On distributed convex optimization under inequality and equality constraints. *IEEE Transactions on Automatic Control*, 57(1):151–164, 2011.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320, 2005.