Communication/Computation Tradeoffs in Consensus-Based Distributed Optimization

Konstantinos I. Tsianos, Sean Lawlor, and Michael G. Rabbat
Department of Electrical and Computer Engineering
McGill University, Montréal, Canada
{konstantinos.tsianos, sean.lawlor}@mail.mcgill.ca
michael.rabbat@mcgill.ca

Abstract

We study the scalability of consensus-based distributed optimization algorithms by considering two questions: How many processors should we use for a given problem, and how often should they communicate when communication is not free? Central to our analysis is a problem-specific value r which quantifies the communication/computation tradeoff. We show that organizing the communication among nodes as a k-regular expander graph [1] yields speedups, while when all pairs of nodes communicate (as in a complete graph), there is an optimal number of processors that depends on r. Surprisingly, a speedup can be obtained, in terms of the time to reach a fixed level of accuracy, by communicating less and less frequently as the computation progresses. Experiments on a real cluster solving metric learning and non-smooth convex minimization tasks demonstrate strong agreement between theory and practice.

1 Introduction

How many processors should we use and how often should they communicate for large-scale distributed optimization? We address these questions by studying the performance and limitations of a class of distributed algorithms that solve the general optimization problem

$$\underset{x\in\mathcal{X}}{\text{minimize}}\ F(x) = \frac{1}{m}\sum_{j=1}^{m} l_j(x) \tag{1}$$

where each function $l_j(x)$ is convex over a convex set $\mathcal{X} \subseteq \mathbb{R}^d$. This formulation applies widely in machine learning scenarios, where $l_j(x)$ measures the loss of model x with respect to data point j, and $F(x)$ is the cumulative loss over all m data points.

Although efficient serial algorithms exist [2], the increasing size of available data and problem dimensionality are pushing computers to their limits and the need for parallelization arises [3]. Among many proposed distributed approaches for solving (1), we focus on consensus-based distributed optimization [4, 5, 6, 7] where each component function in (1) is assigned to a different node in a network (i.e., the data is partitioned among the nodes), and the nodes interleave local gradient-based optimization updates with communication using a consensus protocol to collectively converge to a minimizer of $F(x)$.

Consensus-based algorithms are attractive because they make distributed optimization possible without requiring centralized coordination or significant network infrastructure (as opposed to, e.g., hierarchical schemes [8]). In addition, they combine simplicity of implementation with robustness to node failures and are resilient to communication delays [9]. These qualities are important in clusters, which are typically shared among many users, and algorithms need to be immune to slow nodes that

use part of their computation and communication resources for unrelated tasks. The main drawback of consensus-based optimization algorithms comes from the potentially high communication cost associated with distributed consensus. At the same time, existing convergence bounds in terms of iterations (e.g., (7) below) suggest that increasing the number of processors slows down convergence, which contradicts the intuition that more computing resources are better.

This paper focuses on understanding the limitations and potential for scalability of consensus-based optimization. We build on the distributed dual averaging framework [4]. The key to our analysis is to attach to each iteration a cost that involves two competing terms: a computation cost per iteration which decreases as we add more processors, and a communication cost which depends on the network. Our cost expression quantifies the communication/computation tradeoff by a parameter r that is easy to estimate for a given problem and platform. The role of r is essential; for example, when nodes communicate at every iteration, we show that in complete graph topologies, there exists an optimal number of processors $n_{opt} = \frac{1}{\sqrt{r}}$, while for k-regular expander graphs [1], increasing the network size yields a diminishing speedup. Similar results are obtained when nodes communicate every $h > 1$ iterations and even when h increases with time. We validate our analysis with experiments on a cluster. Our results show a remarkable agreement between theory and practice.

In Section 2 we formalize the distributed optimization problem and summarize the distributed dual averaging algorithm. Section 3 introduces the communication/computation tradeoff and contains the basic analysis where nodes communicate at every iteration. The general case of sparsifying communication is treated in Section 4. Section 5 tests our theorical results on a real cluster implementation and Section 6 discusses some future extensions.

2 Distributed Convex Optimization

Assume we have at our disposal a cluster with n processors to solve (1), and suppose without loss of generality that m is divisible by n. In the absence of any other information, we partition the data evenly among the processors and our objective becomes to solve the optimization problem,

$$\underset{x\in\mathcal{X}}{\text{minimize}}\ F(x) = \frac{1}{m}\sum_{j=1}^{m} l_j(x) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{n}{m}\sum_{j=1}^{\frac{m}{n}} l_{j|i}(x)\right) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{2}$$

where we use the notation $l_{j|i}$ to denote loss associated with the jth local data point at processor i (i.e., $j|i = (i-1)\frac{m}{n} + j$). The local objective functions $f_i(x)$ at each node are assumed to be L-Lipschitz and convex. The recent distributed optimization literature contains multiple consensus-based algorithms with similar rates of convergence for solving this type of problem. We adopt the *distributed dual averaging* (DDA) framework [4] because its analysis admits a clear separation between the standard (centralized) optimization error and the error due to distributing computation over a network, facilitating our investigation of the communication/computation tradeoff.

2.1 Distributed Dual Averaging (DDA)

In DDA, nodes iteratively communicate and update optimization variables to solve (2). Nodes only communicate if they are neighbors in a communication graph $G = (V, E)$, with the $|V| = n$ vertices being the processors. The communication graph is user-defined (application layer) and does not necessarily correspond to the physical interconnections between processors. DDA requires three additional quantities: a 1-strongly convex proximal function $\psi : \mathbb{R}^d \to \mathbb{R}$ satisfying $\psi(x) \geq 0$ and $\psi(0) = 0$ (e.g., $\psi(x) = \frac{1}{2}x^T x$); a positive step size sequence $a(t) = O(\frac{1}{\sqrt{t}})$; and a $n \times n$ doubly stochastic consensus matrix P with entries $p_{ij} > 0$ only if either $i = j$ or $(j, i) \in E$ and $p_{ij} = 0$ otherwise. The algorithm repeats for each node i in discrete steps t, the following updates:

$$z_i(t) = \sum_{j=1}^{n} p_{ij} z_j(t-1) + g_i(t-1) \tag{3}$$

$$x_i(t) = \underset{x\in\mathcal{X}}{\text{argmin}}\left\{\langle z_i(t), x\rangle + \frac{1}{a(t)}\psi(x)\right\} \tag{4}$$

$$\hat{x}_i(t) = \frac{1}{t}\big((t-1)\cdot\hat{x}_i(t-1) + x_i(t)\big) \tag{5}$$

where $g_i(t-1) \in \partial f_i(x_i(t-1))$ is a subgradient of $f_i(x)$ evaluated at $x_i(t-1)$. In (3), the variable $z_i(t) \in \mathbb{R}^d$ maintains an accumulated subgradient up to time t and represents node i's belief of the direction of the optimum. To update $z_i(t)$ in (3), each node must communicate to exchange the variables $z_j(t)$ with its neighbors in G. If $\psi(x^*) \leq R^2$, for the local running averages $\hat{x}_i(t)$ defined in (5), the error from a minimizer x^* of $F(x)$ after T iterations is bounded by (Theorem 1, [4])

$$\begin{aligned}\mathrm{Err}_i(T) = F(\hat{x}_i(T)) - F(x^*) \leq & \frac{R^2}{Ta(T)} + \frac{L^2}{2T}\sum_{t=1}^{T} a(t-1) \\ & + \frac{L}{T}\sum_{t=1}^{T} a(t)\left(\frac{2}{n}\sum_{j=1}^{n}\|\bar{z}(t) - z_j(t)\|_* + \|\bar{z}(t) - z_i(t)\|_*\right) \quad (6)\end{aligned}$$

where L is the Lipschitz constant, $\|\cdot\|_*$ indicates the dual norm, $\bar{z}(t) = \frac{1}{n}\sum_{i=1}^{n} z_i(t)$, and $\|\bar{z}(t) - z_i(t)\|_*$ quantifies the network error as a disagreement between the direction to the optimum at node i and the consensus direction $\bar{z}(t)$ at time t. Furthermore, from Theorem 2 in [4], with $a(t) = \frac{A}{\sqrt{t}}$, after optimizing for A we have a bound on the error,

$$\mathrm{Err}_i(T) \leq C_1 \frac{\log(T\sqrt{n})}{\sqrt{T}}, \qquad C_1 = 2LR\sqrt{19 + \frac{12}{1-\sqrt{\lambda_2}}}, \quad (7)$$

where λ_2 is the second largest eigenvalue of P. The dependence on the communication topology is reflected through λ_2, since the sparsity structure of P is determined by G. According to (7), increasing n slows down the rate of convergence even if λ_2 does not depend on n.

3 Communication/Computation Tradeoff

In consensus-based distributed optimization algorithms such as DDA, the communication graph G and the cost of transmitting a message have an important influence on convergence speed, especially when communicating one message requires a non-trivial amount of time (e.g., if the dimension d of the problem is very high).

We are interested in the shortest time to obtain an ϵ-accurate solution (i.e., $\mathrm{Err}_i(T) \leq \epsilon$). From (7), convergence is faster for topologies with good expansion properties; i.e., when the spectral gap $1 - \sqrt{\lambda_2}$ does not shrink too quickly as n grows. In addition, it is preferable to have a balanced network, where each node has the same number of neighbors so that all nodes spend roughly the same amount of time communicating per iteration. Below we focus on two particular cases and take G to be either a complete graph (i.e., all pairs of nodes communicate) or a k-regular expander [1].

By using more processors, the total amount of communication inevitably increases. At the same time, more data can be processed in parallel in the same amount of time. We focus on the scenario where the size m of the dataset is fixed but possibly very large. To understand whether there is room for speedup, we move away from measuring iterations and employ a time model that explicitly accounts for communication cost. This will allow us to study the communication/computation tradeoff and draw conclusions based on the total amount of time to reach an ϵ accuracy solution.

3.1 Time model

At each iteration, in step (3), processor i computes a local subgradient on its subset of the data:

$$g_i(x) = \frac{\partial f_i(x)}{\partial x} = \frac{n}{m}\sum_{j=1}^{\frac{m}{n}} \frac{\partial l_{j|i}(x)}{\partial x}. \quad (8)$$

The cost of this computation increases linearly with the subset size. Let us normalize time so that one processor compute a subgradient on the full dataset of size m in 1 time unit. Then, using n cpus, each local gradient will take $\frac{1}{n}$ time units to compute. We ignore the time required to compute the projection in step (4); often this can be done very efficiently and requires negligible time when m is large compared to n and d.

We account for the cost of communication as follows. In the consensus update (3), each pair of neighbors in G transmits and receives one variable $z_j(t-1)$. Since the message size depends only on the problem dimension d and does not change with m or n, we denote by r the time required to transmit and receive one message, relative to the 1 time unit required to compute the full gradient on all the data. If every node has k neighbors, the cost of one iteration in a network of n nodes is

$$\frac{1}{n} + kr \text{ time units / iteration.} \tag{9}$$

Using this time model, we study the convergence rate bound (7) after attaching an appropriate time unit cost per iteration. To obtain a speedup by increasing the number of processors n for a given problem, we must ensure that ϵ-accuracy is achieved in fewer time units.

3.2 Simple Case: Communicate at every Iteration

In the original DDA description (3)-(5), nodes communicate at every iteration. According to our time model, T iterations will cost $\tau = T(\frac{1}{n} + kr)$ time units. From (7), the time $\tau(\epsilon)$ to reach error ϵ is found by substituting for T and solving for $\tau(\epsilon)$. Ignoring the log factor in (7), we get

$$C_1 \frac{1}{\sqrt{\frac{\tau(\epsilon)}{\frac{1}{n}+kr}}} = \epsilon \quad \Longrightarrow \quad \tau(\epsilon) = \frac{C_1^2}{\epsilon^2}\Big(\frac{1}{n} + kr\Big) \text{ time units.} \tag{10}$$

This simple manipulation reveals some important facts. If communication is free, then $r = 0$. If in addition the network G is a k-regular expander, then λ_2 is fixed [10], C_1 is independent of n and $\tau(\epsilon) = C_1^2/(\epsilon^2 n)$. Thus, in the ideal situation, we obtain a linear speedup by increasing the number of processors, as one would expect. In reality, of course, communication is not free.

Complete graph. Suppose that G is the complete graph, where $k = n - 1$ and $\lambda_2 = 0$. In this scenario we cannot keep increasing the network size without eventually harming performance due to the excessive communication cost. For a problem with a communication/computation tradeoff r, the optimal number of processors is calculated by minimizing $\tau(\epsilon)$ for n:

$$\frac{\partial \tau(\epsilon)}{\partial n} = 0 \quad \Longrightarrow \quad n_{opt} = \frac{1}{\sqrt{r}}. \tag{11}$$

Again, in accordance with intuition, if the communication cost is too high (i.e., $r \geq 1$) and it takes more time to transmit and receive a gradient than it takes to compute it, using a complete graph cannot speedup the optimization. We reiterate that r is a quantity that can be easily measured for a given hardware and a given optimization problem. As we report in Section 5, the optimal value predicted by our theory agrees very well with experimental performance on a real cluster.

Expander. For the case where G is a k-regular expander, the communication cost per node remains constant as n increases. From (10) and the expression for C_1 in (7), we see that n can be increased without losing performance, although the benefit diminishes (relative to kr) as n grows.

4 General Case: Sparse Communication

The previous section analyzes the case where processors communicate at every iteration. Next we investigate the more general situation where we adjust the frequency of communication.

4.1 Bounded Intercommunication Intervals

Suppose that a consensus step takes place once every $h+1$ iterations. That is, the algorithm repeats $h \geq 1$ cheap iterations (no communication) of cost $\frac{1}{n}$ time units followed by an expensive iteration (with communication) with cost $\frac{1}{n} + kr$. This strategy clearly reduces the overall average cost per iteration. The caveat is that the network error $\|\bar{z}(t) - z_i(t)\|_*$ is higher because of having executed fewer consensus steps.

In a cheap iteration we replace the update (3) by $z_i(t) = z_i(t-1) + g_i(t-1)$. After some straightforward algebra we can show that [for (12), (16) please consult the supplementary material]:

$$z_i(t) = \sum_{w=0}^{H_t-1} \sum_{k=0}^{h-1} \sum_{j=1}^{n} \left[P^{H_t-w}\right]_{ij} g_j(wh+k) + \sum_{k=0}^{Q_t-1} g_i(t-Q_t+k). \tag{12}$$

where $H_t = \lfloor \frac{t-1}{h} \rfloor$ counts the number of communication steps in t iterations, and $Q_t = \text{mod}(t,h)$ if $\text{mod}(t,h) > 0$ and $Q_t = h$ otherwise. Using the fact that $P\mathbf{1} = \mathbf{1}$, we obtain

$$\bar{z}(t) - z_i(t) = \frac{1}{n}\sum_{s=1}^{n} z_s(t) - z_i(t) = \sum_{w=0}^{H_t-1}\sum_{j=1}^{n}\left(\frac{1}{n} - \left[P^{H_t-w}\right]_{ij}\right)\sum_{k=0}^{h-1} g_j(wh+k) \tag{13}$$

$$+ \frac{1}{n}\sum_{s=1}^{n}\sum_{k=0}^{Q_t-1}\big(g_s(t-Q_t+k) - g_i(t-Q_t+k)\big). \tag{14}$$

Taking norms, recalling that the f_i are convex and Lipschitz, and since $Q_t \leq h$, we arrive at

$$\|\bar{z}(t) - z_i(t)\|_* \leq \sum_{w=0}^{H_t-1}\left\|\frac{1}{n}\mathbf{1}^T - \left[P^{H_t-w}\right]_{i,:}\right\|_1 hL + 2hL \tag{15}$$

Using a technique similar to that in [4] to bound the ℓ_1 distance of row i of P^{H_t-w} to its stationary distribution as t grows, we can show that

$$\|\bar{z}(t) - z_i(t)\|_* \leq 2hL\frac{\log(T\sqrt{n})}{1-\sqrt{\lambda_2}} + 3hL \tag{16}$$

for all $t \leq T$. Comparing (16) to equation (29) in [4], the network error within t iterations is no more than h times larger when a consensus step is only performed once every $h+1$ iterations. Finally, we substitute the network error in (6). For $a(t) = \frac{A}{\sqrt{t}}$, we have $\sum_{t=1}^{T} a(t) \leq 2A\sqrt{T}$, and

$$\text{Err}_i(T) \leq \left(\frac{R^2}{A} + AL^2\left(1 + \frac{12h}{1-\sqrt{\lambda_2}} + 18h\right)\right)\frac{\log\left(T\sqrt{n}\right)}{\sqrt{T}} = C_h\frac{\log\left(T\sqrt{n}\right)}{\sqrt{T}}. \tag{17}$$

We minimize the leading term C_h over A to obtain

$$A = \frac{R}{L}\left(\sqrt{1 + 18h + \frac{12h}{1-\sqrt{\lambda_2}}}\right)^{-1} \text{ and } \quad C_h = 2RL\sqrt{1 + 18h + \frac{12h}{1-\sqrt{\lambda_2}}}. \tag{18}$$

Of the T iterations, only $H_T = \lfloor \frac{T-1}{h} \rfloor$ involve communication. So, T iterations will take

$$\tau = (T - H_T)\frac{1}{n} + H_T\left(\frac{1}{n} + kr\right) = \frac{T}{n} + H_T kr \text{ time units.} \tag{19}$$

To achieve ϵ-accuracy, ignoring again the logarithmic factor, we need $T = \frac{C_h^2}{\epsilon^2}$ iterations, or

$$\tau(\epsilon) = \left(\frac{T}{n} + \left\lfloor\frac{T-1}{h}\right\rfloor kr\right) \leq \frac{C_h^2}{\epsilon^2}\left(\frac{1}{n} + \frac{kr}{h}\right) \text{ time units.} \tag{20}$$

From the last expression, for a fixed number of processors n, there exists an optimal value for h that depends on the network size and communication graph G:

$$h_{opt} = \sqrt{\frac{nkr}{18 + \frac{12}{1-\sqrt{\lambda_2}}}}. \tag{21}$$

If the network is a complete graph, using h_{opt} yields $\tau(\epsilon) = O(n)$; i.e., using more processors hurts performance when not communicating every iteration. On the other hand, if the network is a k-regular expander then $\tau(\epsilon) = \frac{c_1}{\sqrt{n}} + c_2$ for constants c_1, c_2, and we obtain a diminishing speedup.

4.2 Increasingly Sparse Communication

Next, we consider progressively increasing the intercommunication intervals. This captures the intuition that as the optimization moves closer to the solution, progress slows down and a processor should have "something significantly new to say" before it communicates. Let $h_j - 1$ denote the number of cheap iterations performed between the $(j-1)$st and jth expensive iteration; i.e., the first communication is at iteration h_1, the second at iteration $h_1 + h_2$, and so on. We consider schemes

where $h_j = j^p$ for $p \geq 0$. The number of iterations that nodes communicate out of the first T total iterations is given by $H_T = \max\{H\colon \sum_{j=1}^{H} h_j \leq T\}$. We have

$$\int_{y=1}^{H_T} y^p dy \leq \sum_{j=1}^{H_T} j^p \leq 1 + \int_{y=1}^{H_T} y^p dy \quad \Longrightarrow \quad \frac{H_T^{p+1} - 1}{p+1} \leq T \leq \frac{H_T^{p+1} + p}{p+1}, \tag{22}$$

which means that $H_T = \Theta(T^{\frac{1}{p+1}})$ as $T \to \infty$. Similar to (15), the network error is bounded as

$$\|\bar{z}(t) - z_i(t)\|_* \leq \sum_{w=0}^{H_t-1} \left\| \frac{1}{n}\mathbf{1}^T - \left[P^{H_t-w}\right]_{i,:} \right\|_1 \sum_{k=0}^{h_w-1} L + 2h_t L = L \sum_{w=0}^{H_t-1} \|\cdot\|_1 h_w + 2h_t L. \tag{23}$$

We split the sum into two terms based on whether or not the powers of P have converged. Using the split point $\hat{t} = \frac{\log(T\sqrt{n})}{1-\sqrt{\lambda_2}}$, the ℓ_1 term is bounded by 2 when w is large and by $\frac{1}{T}$ when w is small:

$$\begin{aligned}
\|\bar{z}(t) - z_i(t)\|_* \leq & L \sum_{w=0}^{H_t-1-\hat{t}} \|\cdot\|_1 h_w + L \sum_{w=H_t-\hat{t}}^{H_t-1} \|\cdot\|_1 h_w + 2h_t L && (24)\\
\leq & \frac{L}{T} \sum_{w=0}^{H_t-1-\hat{t}} w^p + 2L \sum_{w=H_t-\hat{t}}^{H_t-1} w^p + 2t^p L && (25)\\
\leq & \frac{L}{T} \frac{(H_t - \hat{t} - 1)^{\frac{1}{p+1}} + p}{p+1} + 2L\hat{t}(H_t-1)^p + 2t^p L && (26)\\
\leq & \frac{L}{p+1} + \frac{Lp}{T(p+1)} + 2L\hat{t}H_t^p + 2t^p L && (27)
\end{aligned}$$

since $T > H_t - \hat{t} - 1$. Substituting this bound into (6) and taking the step size sequence to be $a(t) = \frac{A}{t^q}$ with A and q to be determined, we get

$$\begin{aligned}
\text{Err}_i(T) \leq & \frac{R^2}{AT^{1-q}} + \frac{L^2 A}{2(1-q)T^q} + \frac{3L^2 A}{(p+1)(1-q)T^q} + \frac{3L^2 pA}{(p+1)(1-q)T^{1+q}} \\
& + \frac{6L^2\hat{t}A}{T} \sum_{t=1}^{T} \frac{H_t^p}{t^q} + \frac{6L^2 A}{T} \sum_{t=1}^{T} t^{p-q}.
\end{aligned} \tag{28}$$

The first four summands converge to zero when $0 < q < 1$. Since $H_t = \Theta(t^{\frac{1}{p+1}})$,

$$\frac{1}{T}\sum_{t=1}^{T} \frac{H_t^p}{t^q} \leq \frac{1}{T}\sum_{t=1}^{T} \frac{O(t^{\frac{1}{p+1}})^p}{t^q} \leq O\left(\frac{T^{\frac{p}{p+1}-q+1}}{T}\right) = O\left(T^{\frac{p}{p+1}-q}\right) \tag{29}$$

which converges to zero if $\frac{p}{p+1} < q$. To bound the last term, note that $\frac{1}{T}\sum_{t=1}^{T} t^{p-q} \leq \frac{T^{p-q}}{p-q+1}$, so the term goes to zero as $T \to \infty$ if $p < q$. In conclusion, $\text{Err}_i(T)$ converges no slower than $O(\frac{\log(T\sqrt{n})}{T^{q-p}})$ since $\frac{1}{T^{q-\frac{p}{p+1}}} < \frac{1}{T^{q-p}}$. If we choose $q = \frac{1}{2}$ to balance the first three summands, for small $p > 0$, the rate of convergence is arbitrarily close to $O(\frac{\log(T\sqrt{n})}{\sqrt{T}})$, while nodes communicate increasingly infrequently as $T \to \infty$.

Out of T total iterations, DDA executes $H_T = \Theta(T^{\frac{p}{p+1}})$ expensive iterations involving communication and $T - H_T$ cheap iterations without communication, so

$$\tau(\epsilon) = O\left(\frac{T}{n} + T^{\frac{p}{p+1}} kr\right) = O\left(T\left(\frac{1}{n} + \frac{kr}{T^{\frac{1}{p+1}}}\right)\right). \tag{30}$$

In this case, the communication cost kr becomes a less and less significant proportion of $\tau(\epsilon)$ as T increases. So for any $0 < p < \frac{1}{2}$, if k is fixed, we approach a linear speedup behaviour $\Theta(\frac{T}{n})$. To get $\text{Err}_i(T) \leq \epsilon$, ignoring the logarithmic factor, we need

$$T = \left(\frac{C_p}{\epsilon}\right)^{\frac{2}{1-2p}} \text{ iterations, with } C_p = 2LR\sqrt{7 + \frac{12p+12}{(3p+1)(1-\sqrt{\lambda_2})} + \frac{12}{2p+1}}. \tag{31}$$

From this last equation we see that for $0 < p < \frac{1}{2}$ we have $C_p < C_1$, so using increasingly sparse communication should, in fact, be faster than communicating at every iteration.

5 Experimental Evaluation

To verify our theoretical findings, we implement DDA on a cluster of 14 nodes with 3.2 GHz Pentium 4HT processors and 1 GB of memory each, connected via ethernet that allows for roughly 11 MB/sec throughput per node. Our implementation is in C++ using the send and receive functions of OpenMPI v1.4.4 for communication. The Armadillo v2.3.91 library, linked to LAPACK and BLAS, is used for efficient numerical computations.

5.1 Application to Metric Learning

Metric learning [11, 12, 13] is a computationally intensive problem where the goal is to find a distance metric $D(u, v)$ such that points that are related have a very small distance under D while for unrelated points D is large. Following the formulation in [14], we have a data set $\{u_j, v_j, s_j\}_{j=1}^m$ with $u_j, v_j \in \mathbb{R}^d$ and $s_j = \{-1, 1\}$ signifying whether or not u_j is similar to v_j (e.g., similar if they are from the same class). Our goal is to find a symmetric positive semi-definite matrix $A \succeq 0$ to define a pseudo-metric of the form $D_A(u, v) = \sqrt{(u-v)^T A(u-v)}$. To that end, we use a hinge-type loss function $l_j(A, b) = \max\{0, s_j\left(D_A(u_j, v_j)^2 - b\right) + 1\}$ where $b \geq 1$ is a threshold that determines whether two points are dissimilar according to $D_A(\cdot, \cdot)$. In the batch setting, we formulate the convex optimization problem

$$\underset{A,b}{\text{minimize}} \quad F(A, b) = \sum_{j=1}^{m} l_j(A, b) \quad \text{subject to} \quad A \succeq 0, b \geq 1. \tag{32}$$

The subgradient of l_j at (A, b) is zero if $s_j(D_A(u_j, v_j)^2 - b) \leq -1$. Otherwise

$$\frac{\partial l_j(A, b)}{\partial A} = s_j(u_j - v_j)^T(u_j - v_j), \quad \text{and} \quad \frac{\partial l_j(A, b)}{\partial b} = -s_j. \tag{33}$$

Since DDA uses vectors $x_i(t)$ and $z_i(t)$, we represent each pair $(A_i(t), b_i(t))$ as a d^2+1 dimensional vector. The communication cost is thus quadratic in the dimension. In step (3) of DDA, we use the proximal function $\psi(x) = \frac{1}{2}x^T x$, in which case (4) simplifies to taking $x_i(t) = -a(t-1)z_i(t)$, followed by projecting $x_i(t)$ to the constraint set by setting $b_i(t) \leftarrow \max\{1, b_i(t)\}$ and projecting $A_i(t)$ to the set of positive semi-definite matrices by first taking its eigenvalue decomposition and reconstructing $A_i(t)$ after forcing any negative eigenvalues to zero.

We use the MNIST digits dataset which consists of 28×28 pixel images of handwritten digits 0 through 9. Representing images as vectors, we have $d = 28^2 = 784$ and a problem with $d^2 + 1 = 614657$ dimensions trying to learn a 784×784 matrix A. With double precision arithmetic, each DDA message has a size approximately 4.7 MB. We construct a dataset by randomly selecting 5000 pairs from the full MNIST data. One node needs 29 seconds to compute a gradient on this dataset, and sending and receiving 4.7 MB takes 0.85 seconds. The communication/computation tradeoff value is estimated as $r = \frac{0.85}{29} \approx 0.0293$. According to (11), when G is a complete graph, we expect to have optimal performance when using $n_{opt} = \frac{1}{\sqrt{r}} = 5.8$ nodes. Figure 1(left) shows the evolution of the average function value $\bar{F}(t) = \frac{1}{n}\sum_i F(\hat{x}_i(t))$ for 1 to 14 processors connected as a complete graph, where $\hat{x}_i(t)$ is as defined in (5). There is a very good match between theory and practice since the fastest convergence is achieved with $n = 6$ nodes.

In the second experiment, to make r closer to 0, we apply PCA to the original data and keep the top 87 principal components, containing 90% of the energy. The dimension of the problem is reduced dramatically to $87 \cdot 87 + 1 = 7570$ and the message size to 59 KB. Using 60000 random pairs of MNIST data, the time to compute one gradient on the entire dataset with one node is 2.1 seconds, while the time to transmit and receive 59 KB is only 0.0104 seconds. Again, for a complete graph, Figure 1(right) illustrates the evolution of $\bar{F}(t)$ for 1 to 14 nodes. As we see, increasing n speeds up the computation. The speedup we get is close to linear at first, but diminishes since communication is not entirely free. In this case $r = \frac{0.0104}{2.1} = 0.005$ and $n_{opt} = 14.15$.

5.2 Nonsmooth Convex Minimization

Next we create an artificial problem where the minima of the components $f_i(x)$ at each node are very different, so that communication is essential in order to obtain an accurate optimizer of $F(x)$.

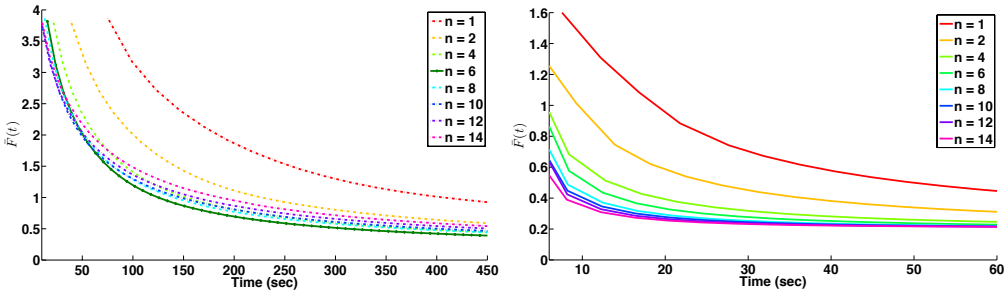

Figure 1: (Left) In a subset of the Full MNIST data for our specific hardware, $n_{opt} = \frac{1}{\sqrt{r}} = 5.8$. The fastest convergence is achieved on a complete graph of 6 nodes. (Right) In the reduced MNIST data using PCA, the communication cost drops and a speedup is achieved by scaling up to 14 processors.

We define $f_i(x)$ as a sum of high dimensional quadratics,

$$f_i(x) = \sum_{j=1}^{M} \max\left(l^1_{j|i}(x), l^2_{j|i}(x)\right), \quad l^\xi_{j|i}(x) = (x - c^\xi_{j|i})^T(x - c^\xi_{j|i}), \quad \xi \in \{1, 2\}, \tag{34}$$

where $x \in \mathbb{R}^{10,000}$, $M = 15,000$ and $c^1_{j|i}, c^2_{j|i}$ are the centers of the quadratics. Figure 2 illustrates again the average function value $\bar{F}(t)$ for 10 nodes in a complete graph topology. The baseline performance is when nodes communicate at every iteration ($h = 1$). For this problem $r = 0.00089$ and, from (21), $h_{opt} = 1$. Naturally communicating every 2 iterations ($h = 2$) slows down convergence. Over the duration of the experiment, with $h = 2$, each node communicates with its peers 55 times. We selected $p = 0.3$ for increasingly sparse communication, and got $H_T = 53$ communications per node. As we see, even though nodes communicate as much as the $h = 2$ case, convergence is even faster than communicating at every iteration. This verifies our intuition that communication is more important in the beginning. Finally, the case where $p = 1$ is shown. This value is out of the permissible range, and as expected DDA does not converge to the right solution.

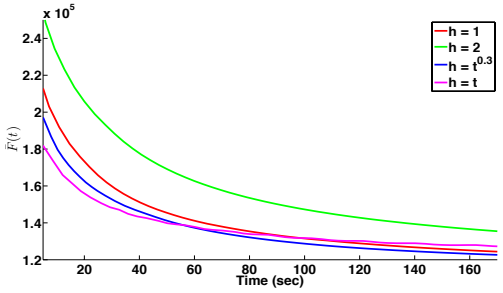

Figure 2: Sparsifying communication to minimize (34) with 10 nodes in a complete graph topology. When waiting $t^{0.3}$ iterations between consensus steps, convergence is faster than communicating at every iteration ($h = 1$), even though the total number of consensus steps performed over the duration of the experiment is equal to communicating every 2 iterations ($h = 2$). When waiting a linear number of iterations between consensus steps ($h = t$) DDA does not converge to the right solution. Note: all methods are initialized from the same value; the x-axis starts at 5 sec.

6 Conclusions and Future Work

The analysis and experimental evaluation in this paper focus on distributed dual averaging and reveal the capability of distributed dual averaging to scale with the network size. We expect that similar results hold for other consensus-based algorithms such as [5] as well as various distributed averaging-type algorithms (e.g., [15, 16, 17]). In the future we will extend the analysis to the case of stochastic optimization, where $h_t = t^p$ could correspond to using increasingly larger mini-batches.

References

[1] O. Reingold, S. Vadhan, and A. Wigderson, "Entropy waves, the zig-zag graph product, and new constant-degree expanders," *Annals of Mathematics*, vol. 155, no. 2, pp. 157–187, 2002.

[2] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical Programming Series B*, vol. 120, pp. 221–259, 2009.

[3] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up Machine Learning, Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[4] J. Duchi, A. Agarwal, and M. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2011.

[5] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, January 2009.

[6] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Control and Optimization*, vol. 20, no. 3, 2009.

[7] S. S. Ram, A. Nedic, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optimization Theory and Applications*, vol. 147, no. 3, pp. 516–545, 2011.

[8] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Neural Information Processing Systems*, 2011.

[9] K. I. Tsianos and M. G. Rabbat, "Distributed dual averaging for convex optimization under communication delays," in *American Control Conference (ACC)*, 2012.

[10] F. Chung, *Spectral Graph Theory*. AMS, 1998.

[11] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Neural Information Processing Systems*, 2003.

[12] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Optimization Theory and Applications*, vol. 10, pp. 207–244, 2009.

[13] K. Q. Weinberger, F. Sha, and L. K. Saul, "Convex optimizations for distance metric learning and pattern classification," *IEEE Signal Processing Magazine*, 2010.

[14] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng, "Online and batch learning of pseudo-metrics," in *ICML*, 2004, pp. 743–750.

[15] M. A. Zinkevich, M. Weimer, A. Smola, and L. Li, "Parallelized stochastic gradient descent," in *Neural Information Processing Systems*, 2010.

[16] R. McDonald, K. Hall, and G. Mann, "Distributed training strategies for the structured perceptron," in *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2012, pp. 456–464.

[17] G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. D. Walker, "Efficient large-scale distributed training of conditional maximum entropy models," in *Neural Information Processing Systems*, 2009, pp. 1231–1239.