Networked Optimization with Adaptive Communication

Konstantinos I. Tsianos, Sean F. Lawlor, Jun Ye Yu, and Michael G. Rabbat
Department of Electrical and Computer Engineering
McGill University
Montréal, Québec, Canada
Email: {konstantinos.tsianos, sean.lawlor, jun.y.yu}@mail.mcgill.ca, michael.rabbat@mcgill.ca

***Abstract*—Methods for distributed optimization are necessary to solve large-scale problems such as those becoming more common in machine learning. The communication cost associated with transmitting large messages can become a serious performance bottleneck. We propose a consensus-based distributed algorithm to minimize a convex separable objective. Each node holds one component of the objective function, and the nodes alternate between a computation phase, where local gradient steps are performed based on the local objective, and a communication phase, where consensus steps are performed to bring the local states into agreement. The nodes use local decision rules to adaptively determine when communication is not necessary. This results in significantly lower communication costs and allows a user to tradeoff the amount of communication with the accuracy of the final output. Experiments on a cluster using simulated and real datasets illustrate the tradeoff.**

I. Introduction

In this paper we present a consensus-based distributed optimization algorithm aimed at adaptively eliminating unnecessary or redundant communications. Our algorithm interleaves communication and computation phases in order to bring a network of processors into agreement on the minimum of a convex objective. The appeal of our algorithm is the ability to find the solution by dynamically adjusting the amount of communication during the communication phase locally at each node. This can lead to significant reductions in the amount of information communicated, and communication costs can be critical in large scale problems where transmitting data over a network could be very slow. The adaptivity of each node's communication behavior is controlled by a tunable parameter which also controls the desired final accuracy of the solution. As a byproduct, our distributed algorithm endows the nodes with the ability to adaptively determine when to stop computing and transmitting information without using a centralized coordinator. Simulation results confirm that indeed we can solve machine learning tasks in a distributed manner while dramatically reducing the number of messages exchanged.

The need for distributed optimization algorithms has become prominent in the area of machine learning since the size of available datasets has been consistently pushing computers to their limits [1]. The recent work of Agarwal et al. [2], in which a classifier is trained using 1,000 machines on a human genome dataset with 50,000,000 training instances (roughly 3 Terabytes in size) is instructive of this trend. Out of the numerous distributed approaches that have been proposed in the literature, in this work we focus on the class of consensus-based distributed optimization algorithms [3]–[8]. At a high level all these algorithms employ a local gradient-based optimization scheme combined with a communication phase where the nodes exchange information with their neighbors in order to drive their local states towards a consensus on the global optimum. These algorithms are particularly suitable for optimizing separable objectives of the form

$$\underset{x\in\mathcal{X}}{\text{minimize}}\ F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{1}$$

where each component $f_i(x)$ in (1) is assigned to a different node in the 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 [9]). In addition, they combine simplicity of implementation with robustness to node failures and are resilient to communication delays [10].

The main drawback of consensus-based algorithms comes from the potentially high communication cost associated with distributed consensus. This problem can be quite significant in machine learning applications where the data being exchanged over the network could be several MB in size. To decrease the communication overhead, we describe an algorithm where each node uses its own local rule to determine whether or not to continue communicating with its neighbours. To that end, we employ the idea of local silencing rules proposed in [11], in combination with Distributed Dual Averaging (DDA) [3]. Using such a strategy at each node can result in a significant reduction in communication which, in practice, could translate into tremendous speedups (see, e.g., experiments in [2]). An appealing aspect of our algorithm is the local nature of the proposed approach; nodes make independent decisions based on their current estimate of the progress towards the global solution. To illustrate the ideas and benefits of the proposed algorithm, we include an experimental evaluation using both real and synthetic data.

The rest of the paper is organized as follows. Section II reviews relevant background. Section III describes the proposed algorithm in detail, and Section IV presents the experimental evaluation and findings. The paper concludes with a summary and possible future extensions in Section V.

II. Background

This section provides necessary background on the two building blocks of the proposed algorithm: local silencing rules for distributed averaging, and distributed dual averaging. For the rest of the paper we assume that we are given a network of n nodes organized as a graph $G = (V, E)$ with $|V| = n$. The nodes are only allowed to exchange information over the edges in the set E. For this paper, we restrict to the case of pairwise interactions meaning that at any time only two nodes communicate over an edge and exchange information.

A. Average Consensus with Local Stopping Rules

Assume that each of the n nodes holds an initial value $x_i(0)$. The question of average consensus is how to use communication over the edges of G to coordinate the nodes so that they reach consensus on the average of the initial values. If $x_i(t)$ is node i's value at time t, we thus require that $x_i(t) \to \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$ as $t \to \infty$. This very simple problem can appear as a building block in a large variety of signal processing tasks [12] including distributed optimization.

One simple and elegant distributed algorithm to solve the average consensus problem is Randomized Gossip, first proposed by Boyd et al. [13]. It is an asynchronous and fully distributed algorithm which does not require routing as nodes only communicate with their neighbours on the graph. The communication overhead per node at each iteration is minimal as each communication is between a pair of adjacent nodes. Specifically, at each time instant, a node wakes up and selects a neighbour at random. The two nodes update their values by averaging. Over time, if the network remains connected, all nodes' values converge to the average of their initial values. It is worth mentioning that the drawback of distributed averaging algorithms is that their diffusive nature can lead to slow convergence on some topologies such as grids and random geometric graphs [13].

Given that gossip algorithms are fully distributed, a key challenge is how to determine when the algorithm has converged and the nodes can stop transmitting data. This is a non-trivial task in the absence of any centralized coordination mechanism. In the majority of the existing literature, if the network size is known one can predefine a maximum number of iterations based on worst-case upper bounds for the number of transmissions required to reach a certain level of accuracy. However, such bounds can be very loose, resulting in an unavoidable waste of communication bandwidth. We employ in this paper the results of [11] where a simple silencing rule for randomized gossip is proposed.

Under the silencing rule, each node maintains an individual counter. Whenever a node updates its value, it increases its counter by one if its value does not change by more than a threshold, $\tau > 0$, and it resets the counter to 0 otherwise. When the counter reaches an upper limit, C, the node still responds to a gossip round initiated by a neighbouring node, but it stops initiating gossip rounds itself. In [11], it is shown that all nodes eventually stop transmitting and the final accuracy of the results depends on the two parameters, τ and C.

More specifically, let n be the number of nodes in the network, $|E|$ the number of edges, d_{max}, the maximum degree of any node, and $\delta > 0$ the desired accuracy. Let A denote the adjacency matrix of G, and let $D = \mathrm{diag}(\mathbf{1}^T A)$ be a diagonal matrix formed by the node degrees. The Laplacian matrix is $L = D - A$. Let λ_2 denote the second smallest eigenvalue of the graph Laplacian.

Theorem 1 (Daher et al. [11]): By setting

$$C = d_{\max}(\log(d_{\max}) + 2\log(n)) \tag{2}$$

$$\tau = \sqrt{\frac{\lambda_2 \delta^2}{4|E|(C-1)^2}} \tag{3}$$

all nodes eventually stop transmitting almost surely, and with probability at least $1 - 1/n$, the error is bounded as

$$||x(K) - \bar{x}|| \leq \delta \tag{4}$$

where $x(t)$ is the vector of node values at iteration t, and K is the first iteration when all nodes stop transmitting.

B. Distributed Dual Averaging (DDA) [3]

DDA solves the problem (1). Each node i knows a function $f_i(x) : \mathbb{R}^d \to \mathbb{R}$. Nodes communicate with one another using the available communication channels indicated by the edge set E. Each f_i is assumed convex and Lipschitz continuous with respect to the same norm $\|\cdot\|$; i.e., $|f_i(x) - f_i(y)| \leq L\|x - y\|, \forall x, y \in \mathcal{X}$. As a consequence, for any $x \in \mathcal{X}$ and any subgradient $g_i \in \partial f_i(x)$ we have $\|g_i\|_* \leq L$ where $\|v\|_* = \sup_{\|u\|=1}\langle u, v\rangle$ is the dual norm.

Let us select a 1-strongly convex function $\psi : \mathbb{R}^d \to \mathbb{R}$ such that $\psi(x) \geq 0$ and $\psi(0) = 0$. For example, take $\psi(x) = \|x\|_2^2$. Also select a decreasing sequence of positive step sizes $a(t) = O(\frac{1}{\sqrt{T}})$. The adaptivity of the step size sequence is standard in subgradient optimization methods to ensure convergence.[1] We also select a doubly stochastic matrix $P = [p_{ij}]$ that respects the structure of G in the sense that $p_{ij} > 0$ only if $i = j$ or $(i, j) \in E$. DDA proceeds as follows. Each node maintains a primal variable $x_i(t)$ and a dual variable $z_i(t)$. At iteration t, node i goes through a phase of communication followed by a phase of computation described next:

1. Communicate: Send $z_i(t)$ to and receive $z_j(t)$ from neighbors.

2. Compute: Update the primal and dual variables by setting

$$z_i(t+1) = \sum_{j=1}^{n} p_{ij} z_j(t) - g_i(t) \tag{5}$$

$$x_i(t+1) = \Pi_{\mathcal{X}}^{\psi}\left(z_i(t+1), a(t)\right) \tag{6}$$

[1] Standard conditions are that the step size sequence is square summable but not summable.

where $g_i(t) \in \partial f_i(x_i(t))$ is a subgradient of f_i at $x_i(t)$, and the projection operator $\Pi_{\mathcal{X}}^{\psi}(\cdot,\cdot)$ is defined as

$$\Pi_{\mathcal{X}}^{\psi}(z,a) = \operatorname*{argmin}_{x\in\mathcal{X}} \left\{ \langle z, x\rangle + \frac{1}{a}\psi(x) \right\}. \tag{7}$$

In [3] it is shown that, for the updates above, the local running average $\hat{x}_i(T) = \frac{1}{T}\sum_{t=1}^{T} x_i(t)$ at every node i converges to the optimum at a rate $F(\hat{x}_i(T)) - F(x^*) = O(\frac{\log(\sqrt{n}T)}{\sqrt{T}})$. This version of DDA describes the algorithm with synchronous updates where every node sends out its current dual variable to all of its neighbours. This is exactly where the communication overhead can become excessive. It is proven however, that DDA has the same convergence rate if a consensus protocol such as randomized gossip is used [3, Theorem 3], and we capitalize on this property in the next section where we describe the proposed algorithm.

III. Proposed Algorithm

Our proposed algorithm is given in pseudo-code in Alg. 1. Nodes initialize $z_i(0) = 0$, $\tilde{z}_i = 0$, and $count_i = 0$. Each node i first updates its estimate $x_i(t)$ and local running average $\hat{x}_i(t)$ and then takes a gradient step. If this gradient step was significant enough (as indicated by the selected threshold τ) relative to the last time it was updated due to incoming information from a neighbour (recorded by the intermediate variable $\tilde{z}_i$), then the node resets its local counter to zero, meaning that this node should communicate (i.e., "something important to say"). In the opposite case, the node increases its counter taking a step closer to silence. If the node's counter is low enough, then the node remains active and communicates its information to a random neighbour. As a result, both nodes update their dual variables and set them to the average of the two. After that, both nodes must check if the new information was significant enough or if they need to take another step towards silence.

The adaptivity of the nodes to the current state of the underlying optimization procedure is what leads to increased sparsity of communication and thus significant savings in execution time when there is limited bandwidth and/or large amounts of information to be transmitted over the network. The more the nodes enter their silence mode, the more time they spend in local gradient updates and the less time they spend communicating. As a pleasant side effect, the algorithm benefits from the local stoping rules theory and all nodes eventually terminate automatically without centralized coordination.

There are a number of subtleties in employing the stopping rules in a distributed optimization algorithm. First of all, notice that for optimization we need to keep an update of the estimates $x_i(t)$ and $\hat{x}_i(t)$ at each node but the consensus phase involves updates in the dual variables. Since the local gradient step moves the estimates and the dual variables, this is a version of a dynamic consensus problem where the consensus value moves. Most importantly, the dual variables are vectors which continuously grow in magnitude. Thus, one needs to be careful when applying the silencing rule since the threshold τ cannot be applied as is in this dynamic situation. The solution is to normalize the norm of the difference in dual variables and thus make the notion of innovation and "enough change" independent of the magnitude of the dual variables. Finally, notice that, as also explained in [13], the type of pairwise averaging just described does yield (time-varying) doubly stochastic matrices as required by the DDA convergence theory.

Algorithm 1 DDA with Adaptive Communications

$x_i(t) = \Pi_{\mathcal{X}}^{\psi}(z_i(t), a(t))$
$\hat{x}_i(t) = \frac{1}{t}\sum_{t=1}^{t} x_i(t)$
$z_i(t+1) = z_i(t) + g_i(t)$
if $\frac{\|\tilde{z}_i - z_i(t+1)\|}{\|\tilde{z}_i\|} \leq \tau$ **then**
  $count_i = count_i + 1$
else
  $count_i = 0$
end if
if $count_i < C$ **then**
  select random neighbour j
  /* Repeat the following steps for i and j */
  $z_i(t+1) = z_j(t+1) \leftarrow \frac{z_i(t+1)+z_j(t+1)}{2}$
  if $\frac{\|\tilde{z}_i - z_i(t+1)\|}{\|z_i(t+1)\|} \leq \tau$ **then**
    $count_i = count_i + 1$
  else
    $count_i = 0$
  end if
end if
if $count_i \leq C$ **then**
  $\tilde{z}_i = z_i(t+1)$
end if

IV. Simulations

To validate that the proposed algorithm works as described in the previous section, we present two simulations: one with synthetic data and one using the well know MNIST digits that are widely used in classification tasks. For both scenarios, we consider a network of $n = 10$ nodes organized as a realization of a random geometric graph. We repeat each run 10 times and average the results.

A. Synthetic Data

For the first set of experiments, we feed each processor with a stream of i.i.d. data to emulate the solution of a least squares problem online. For node i, the instantaneous cost function is defined as

$$f_i(x) = (x - c_i)^T(x - c_i), \quad x \in \mathbb{R}^{20} \tag{8}$$

where $c_i = i\mathbf{1} + \epsilon$. In other words each node receives data that define a quadratic cost, and each data point c_i is corrupted by some uniform noise $\epsilon \in [0, 1]$. For this problem we know the optimal solution, and thus we can track the mean square error (MSE). Notice that the centers of the quadratics are different, so without communication the nodes cannot converge to the

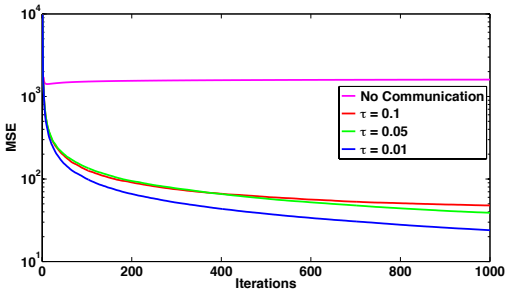

Fig. 1. Distributed optimization with adaptive communications on synthetic data. Network of 10 nodes.

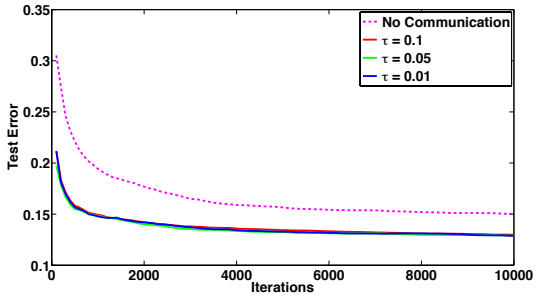

Fig. 2. Distributed optimization to train an SVM classifier for the MNIST digit data with Noodle. Network of 10 nodes.

	$\tau = 0.1$	$\tau = 0.05$	$\tau = 0.01$
Synthetic	131934	179636	200000
MNIST	84974	161056	200000

TABLE I
TOTAL NUMBER OF TRANSMISSIONS ON A 10 NODE RANDOM NETWORKS.

same globally optimal solution. This is clearly seen in Figure 1. In addition, the figure shows that as we increase the value for τ the discrepancies we allow between the nodes are larger. These results should be examined in combination with Table I where we count the total number of transmissions cumulatively among all nodes after 1,000 iterations. As we see with $\tau = 0.01$ the nodes end up communicating at every iteration while increasing this threshold can significantly reduce that communication overhead.

B. MNIST Digits Data

As a proof of concept for a real machine learning problem, we used the MNIST digit dataset to train an SVM classifier. The data consist of 60,000 handwritten 28×28 pixel images of the ten digits 0 through 9. Each image is represented as a vector $w_j \in \mathbb{R}^{784}$. The goal is to train a classifier that can correctly predict the label of an unseen digit image. To simplify the exposition, we design a two-class problem where any digit w_j below 5 is labeled negative ($y_j = -1$) and every digit above 5 is labeled positive ($y_j = +1$). The SVM objective is the hinge loss, given by

$$f_i(x_i) = \sum_{j=1}^{m} \max\{0, 1 - y_{j|i} \cdot w_{j|i}^T x_i\},$$

where $(w_{j|i}, y_{j|i})$ is the jth data point at node i. We separate 50,000 data points for training and kept the rest as an independent test set. Thus, each node in our network has access only to 5,000 data points that are processed in an online/incremental fashion. Figure 2 shows that convergence is very slow without communication. However, when the nodes are allowed to exchange information, convergence is much faster. In addition, the nodes all converge to the same solution so the adaptivity of the threshold and the induced sparsity in communication does not affect the final solution. On a real distributed hardware, saving more than 50% of the transmissions (see Table I) could lead to a significantly faster runtime.

V. CONCLUSIONS AND FUTURE WORK

We present a consensus-based distributed optimization algorithm that uses local rules at the nodes to reduce the amount of unnecessary communication. This could be very beneficial in terms of runtime and bandwidth utilization when applied to large scale distributed problems. We show, in simulations using both synthetic and real data, that the proposed approach reduces the amount of communication without significantly increasing the error. In the future we would like to conduct a formal convergence analysis of the algorithm.

REFERENCES

[1] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up Machine Learning, Parallel and Distributed Approaches*. Cambridge University Press, 2011.
[2] A. Agarwal, O. Chapelle, M. Dudik, and J. Langford, "A reliable effective terascale linear learning system," *arXiv:1110.4198v2*, 2012.
[3] 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.
[4] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, January 2009.
[5] 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.
[6] 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.
[7] J. Chen and A. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, August 2012.
[8] D. Jakovetic, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *arXiv:1112.2972v1*, 2011.
[9] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Neural Information Processing Systems*, 2011.
[10] K. I. Tsianos and M. G. Rabbat, "Distributed dual averaging for convex optimization under communication delays," in *American Control Conference (ACC)*, 2012.
[11] A. Daher, M. Rabbat, and V. Lau, "Local silencing rules for randomized gossip," in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, june 2011, pp. 1 –8.
[12] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847 – 1864, November 2010.
[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, pp. 2508–2530, 2006.