Asynchronous decentralized convex optimization through short-term gradient averaging

Jerome Fellus[1], David Picard[1] and Philippe-Henri Gosselin[1] *

1- ETIS - UMR CNRS 8051 - ENSEA - Universite de Cergy-Pontoise

Abstract. This paper considers decentralized convex optimization over a network in large scale contexts, where *large* simultaneously applies to number of training examples, dimensionality and number of networking nodes. We first propose a centralized optimization scheme that generalizes successful existing methods based on gradient averaging, improving their flexibility by making the number of averaged gradients an explicit parameter of the method. We then propose an asynchronous distributed algorithm that implements this original scheme for large decentralized computing networks.

1 Introduction

Recent successes of stochastic optimization methods based on averaged gradients have renewed the interest for gradient-based schemes to solve large convex problems [7, 8]. However, very large problems such as training a SVM [2] with Terabytes of input data are still computationally challenging. Meanwhile, the burgeoning of data-collecting devices (*e.g.*, mobile phones, sensor networks, *etc*) leads to a logical spread of potential input data into many remote collections that can rarely be gathered into a single site. These computational and massive data spread challenges raise the need for distributed optimization algorithms. In spite of at least three decades of intensive research in this field [13], massively distributed optimization algorithms have proved inescapable in applicative contexts only a few years ago.

Stochastic Gradient Descent schemes (SGD) [3], where only one training example is randomly drawn at each optimization step, have unlocked the problematic complexity of the original full-gradient descent method. Unfortunately, computing gradients on single examples entails a high instability that must be compensated by a decreasing step size to ensure convergence. Nesterov [7] noticed that SGD integrated new gradients with smaller weights than old ones, thus explaining its poor convergence rate in $\mathcal{O}(\frac{\ln(t)}{\sqrt{t}})$. To circumvent this drawback, he proposed the Dual Averaging method (DA) where *all* past gradients are averaged and taken into account with the same weight when performing the update step. Although this leads to an improved convergence rate in $\mathcal{O}(\frac{1}{\sqrt{t}})$, this scheme still exhibits sub-linear convergence. In 2012, Le Roux et al [8] introduced the Stochastic Averaged Gradient method (SAG), achieving a nice exponential convergence rate in $\mathcal{O}(\rho^t), \rho < 1$ (a rate comparable to the deterministic full gradient scheme) for strongly-convex objectives with a constant step size. They built on the intuition that, in case of finite datasets, only the *most recent gradient* with respect to each training vector should be taken into account in the averaged gradient (instead

*This work is funded by the Culture 3D Cloud project.

of all past ones in DA). Exponential convergence is then neither achieved when averaging *one* nor *all* previous gradients, but rather when taking into account some recent gradients and discarding older ones. Unfortunately, SAG's convergence proof [9] does not easily extends to the case where the number of averaged gradients is different from the number of training vectors. Whether exponential or quasi-exponential convergence still holds when averaging an arbitrary number of gradients remains an interesting open question. In this paper, we tackle it from an empirical perspective.

The main contribution of this paper is two-fold : We first introduce a Short-Term Averaged Gradient scheme to optimize a convex objective function, where the number of averaged gradients explicitly appears as a parameter. Secondly, we extend this scheme to distributed environments, and in particular decentralized networking contexts, using an asynchronous gossip averaging protocol. After formally stating our problem, we introduce our Short-Term Average gradient scheme and Gossip averaging. We then present our proposed Asynchronous Gossip Short-Term Averaged Gradient algorithm and report experimental results in centralized and distributed setups, before concluding.

2 Problem statement

Given a strongly-connected network $\mathcal{G}$ of N nodes, each node i holding a sample $\mathbf{X}_i = [\mathbf{x}_1^{(i)} \dots \mathbf{x}_{n_i}^{(i)}]$, $\mathbf{x}_j^{(i)} \in \mathbb{R}^D$, our goal is to compute the unique minimizer $\mathbf{y}^\star \in \mathbb{R}^Q$ of the following strongly-convex objective function f:

$$\forall \mathbf{y} \in \mathbb{R}^Q, \qquad f(\mathbf{y}) = \sum_{i=1}^{N} \sum_{j=1}^{n_i} f_j^{(i)}(\mathbf{y}), \tag{1}$$

where each function $f_j^{(i)}$ is a convex function associated with $\mathbf{x}_j^{(i)}$. For this, each node i starts with a random local estimate $\mathbf{y}^{(i)}$ and updates it using node-local computations on its own sample $\mathbf{X}^{(i)}$ and communication with neighboring nodes. We aim at making all $\mathbf{y}^{(i)}$ converge to $\mathbf{y}^\star$. In addition, three constraints must be respected:

C1. No sample exchange - Only estimates and/or gradients can be exchanged.

C2. Asynchrony - Nodes must never wait for each other.

C3. Decentralization - All nodes and links must play the same role (*i.e.*, all nodes run the same algorithm and links are randomly selected).

3 Short-Term Stochastic Averaged Gradient

First, consider the centralized case ($N = 1$, thus (1) is a traditional convex optimization problem). This problem can be solved using a Stochastic Averaged Gradient Descent scheme, that is, with the following general update rule:

$$\forall i, \quad \mathbf{g}(t) = \nabla f_j(\mathbf{y}(t)), \quad j \text{ randomly drawn in } \{1 \dots n\} \tag{2}$$

$$\mathbf{y}(t+1) = \mathbf{y}(t) - \eta_t \sum_{\tau=1}^{t} \alpha_{t-\tau+1} \mathbf{g}(\tau), \tag{3}$$

where (η_t) is a non-increasing sequence of step sizes and $(\alpha_{t-\tau+1})_{\tau=1}^{t}$ a sequence of t weights associated with the t gradients seen so far (the most recent gradient takes weight α_1 and the oldest one takes α_t). (2-3) encompasses several existing algorithms:

Clearly, SGD [2] corresponds to setting $\alpha_1 = 1$ and $\forall k > 1, \alpha_k = 0$ since only the most recent gradient is kept. Variants integrating a momentum term [11] with fixed weight β also fit to (2-3), with $\alpha_k = \beta^k$.

In its initial statement, SAG [8] does not fit to (2-3), as it keeps the most recent gradient *for each training example*. If training vectors are uniformly drawn with replacement, these are not necessarily the same as the last n gradients. However, as highlighted in [10], the most efficient strategy to draw training examples is generally to select them *without replacement*. This can be implemented by performing deterministic passes over the full data and shuffling the dataset at the begin of each epoch. With this selection strategy, SAG is equivalent to (2-3) with $\alpha_k = \frac{1}{n}$ if $k \leq n$ and 0 otherwise.

Dual Averaging [7] is a proximal method, *i.e.*, $\mathbf{y}(t+1)$ is not computed from $\mathbf{y}(t)$ but rather by directly projecting the averaged gradient using a proximal mapping. However, for simple (*e.g.*, quadratic) proximal functions, one can derive from it an averaged gradient descent (2-3) where all gradients are given an equal weight $\alpha_k = 1, \forall k \in \{1 \dots t\}$.

SAG enjoys the fastest convergence rate of the above-cited algorithms. On the other hand, SAG memorizes as many gradients as training examples, resulting in a storage cost in $\mathcal{O}(nD)$, which can hardly be afforded in large scale setups. A natural idea is then to design an intermediary scheme, where only the M most recent gradients are kept in memory and averaged, where M is a user-defined parameter. We call this scheme *Short-Term Averaged Gradient* (STAG). Intuitively, for M close to 1, we get a behavior similar to SGD, and for M close to n, we get a SAG-like convergence.

4 Asynchronous gossip averaging

Now, consider a distributed case, that is, $N > 1$. To extend STAG to this setup, we would ideally provide all nodes with the average of the M most recent gradients seen over the network for each iteration. Obviously, **C2** and **C3** prevent us from computing this average by gathering local estimates in a master node. Instead, we rely on an asynchronous gossip averaging protocol, which is a special kind of Consensus protocols [1]. Consensus protocols solve distributed averaging over a network by iteratively averaging estimates between neighboring nodes. Assuming each node i holds an arbitrary initial estimate $s_i(0)$ and a weight $w_i(0)$, and setting $\mathbf{s}(t) = (s_1(t), \dots, s_N(t))$ and $\mathbf{w}(t) = (w_1(t), \dots, w_N(t))$, consensus protocols compute the network average $\mu = \frac{\sum_i^N s_i(0)}{\sum_i^N w_i(0)}$ using the following update:

$$\mathbf{s}(t+1)^\top = \mathbf{s}(t)^\top \mathbf{K}(t+1), \quad \mathbf{w}(t+1)^\top = \mathbf{w}(t)^\top \mathbf{K}(t+1), \tag{4}$$

where $\mathbf{K}(t)$ is a constant primitive doubly stochastic matrix respecting $\mathcal{G}$ (*i.e.*, $(\mathbf{K}(t))_{ij} \neq 0$ only if i and j are connected in $\mathcal{G}$). For constant $\mathbf{K}(t) = \mathbf{K}$, we have $\mathbf{s}(t)^\top = \mathbf{s}(0)^T \mathbf{K}^t$ and $\mathbf{w}(t)^\top = \mathbf{w}(0)^T \mathbf{K}^t$. By Perron-Frobenius theorem and stochasticity of $\mathbf{K}$ (*i.e.*, $\mathbf{K1} = \mathbf{1}$), the rows of $\mathbf{K}^t$ tend to equality at rate $\mathcal{O}(\lambda_2^t)$ where λ_2 is

the sub-dominant eigenvalue of $\mathbf{K}$. Consequently,

$$\lim_{t\to\infty}\frac{s_i(t)}{w_i(t)}=\lim_{t\to\infty}\frac{\sum_j^N s_j(0)(\mathbf{K}^t)_{ji}}{\sum_j^N w_j(0)(\mathbf{K}^t)_{ji}}=\lim_{t\to\infty}\frac{(\mathbf{K}^t)_{1,i}\sum_j^N s_j(0)}{(\mathbf{K}^t)_{1,i}\sum_j^N w_j(0)}=\mu \qquad (5)$$

Consensus protocols have been successfully used in the distributed extension of Dual Averaging (DDA [4]). However, constant and doubly-stochastic consensus matrix $\mathbf{K}$ are not compatible with **C2** and **C3**. Hopefully, as shown in [1], both assumptions can be relaxed : any random row-stochastic $\mathbf{K}(t)$ still satisfy (5) as long as $\prod_t \mathbf{K}(t)$ is primitive. Such randomized protocols, called asynchronous Gossip protocols, were recently applied to distributed SGD (for SVM training [5]) and Dual Averaging (PS-DDA [12]).

We define our asynchronous Gossip protocol by $\mathbf{K}(t)=\frac{1}{2}\mathbf{e}_i(\mathbf{e}_i^\top-\mathbf{e}_j^\top)$ where (i,j) is uniformly drawn among the edge set of $\mathcal{G}$. This update rule appears very simple: at any time t, a sender node i is uniformly drawn, and sends half of its estimates to a uniformly chosen neighbor j which adds it to is own estimate:

$$\left\{\begin{array}{l} s_i(t+1)=\frac{1}{2}s_i(t)\\ w_i(t+1)=\frac{1}{2}w_i(t)\end{array}\right. , \quad \left\{\begin{array}{l} s_j(t+1)=s_j(t)+\frac{1}{2}s_i(t)\\ w_j(t+1)=w_j(t)+\frac{1}{2}w_i(t)\end{array}\right. \qquad (6)$$

Remark that, contrarily to (4) that can require simultaneous messages between multiple nodes, update (6) only involves a single random link at a time, thus allowing a fully asynchronous (**C2**) and decentralized (**C3**) functioning.

5 The AGSTAG algorithm

In this section, we describe our proposed distributed optimization algorithm called AGSTAG (Asynchronous Gossip Short-Term Averaged Gradient). It is built on the STAG scheme described in section 3, associated with the asynchronous Gossip averaging protocol defined by (6). In AGSTAG, every nodes run the same local procedure (Algorithm 1). Each node i repeatedly perform the following steps without any synchronization with other nodes. First, it selects a training example (without replacement) to evaluate a new gradient. This gradient is added to a node-local gradients buffer in replacement of the oldest stored one, following a FIFO rule. This buffer thus always stores the M most recently evaluated local gradients. The difference between the oldest gradient and the newest one, respectively leaving and entering the buffer, is added to a local estimate $\mathbf{z}$, while w counts the number of gradients currently stored in the buffer. $(\mathbf{z},w)$ is then iteratively exchanged with random neighboring nodes according to (6).

Remark that this algorithm is asynchronous because there is no message back from the receiver to the sender (senders emit independently of received messages). Notice that the updates of $\mathbf{z}(t)$ imply that the global sum of the dual variables $\sum_i^N \mathbf{z}^{(i)}(t)$ is always equal to the global sum of the gradients stored in all queues. By (5), during the course of AGSTAG, the local dual variables $\mathbf{z}^{(i)}(t)$ behave as estimators of the network-wide average of all currently stored gradients, thus mimicking a centralized STAG, with the difference that there are now NM gradients globally averaged in the network instead of M. Consequently, M can be set N-times smaller to get the same behavior as the centralized scheme, resulting in a much lower memory cost.

Algorithm 1 AGSTAG (independently run at each networking node i)

Parameters : (η_t) : sequence of step sizes. M : capacity of the gradients buffer.
S : number of messages sent at each iteration
Init : $\mathbf{y} \in \mathbb{R}^Q$ at random ; $\mathbf{z} \in \mathbb{R}^Q \leftarrow \mathbf{0}$; $w \leftarrow 0$; FIFO $\leftarrow$ empty with M slots
loop
  $\mathbf{g} \leftarrow \nabla f_j^{(i)}(\mathbf{y})$, where j uniformly drawn without replacement in $\{1 \dots n_i\}$
  If FIFO.size = M **then** $\mathbf{h} \leftarrow$ FIFO.pull() **else** $\mathbf{h} \leftarrow \mathbf{0}$ **EndIf**
  FIFO.push($\mathbf{g}$)
  $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{g} - \mathbf{h}$
  If the FIFO is not full **then** $w \leftarrow w + 1$ **EndIf**
  loop S **times**
    For each received $(\mathbf{z}', w')$ **do** $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{z}'$; $w \leftarrow w + w'$ **EndFor**
    $\mathbf{z} \leftarrow \frac{1}{2}\mathbf{z}$; $w \leftarrow \frac{1}{2}w$
    send $(\mathbf{z}, w)$ to a randomly drawn neighboring node
  end loop
  $\mathbf{y} \leftarrow \mathbf{y} - \frac{\eta_t}{w}\mathbf{z}$
end loop

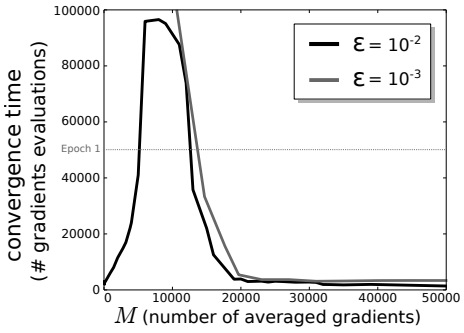

Fig. 1: Convergence time of STAG versus gradient buffer size M

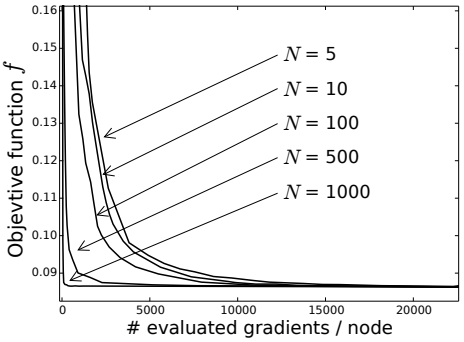

Fig. 2: Convergence of AGSTAG for various network sizes N. Here, $M = n_i/10$.

6 Experiments

We now present experimental results for both the STAG scheme and its distributed AGSTAG extension on the MNIST dataset [6] ($D = 784, n = 60000$). Both algorithms are evaluated in a binary L_2 linear SVM training task (similar setup as [3] with $\lambda = 0.1$). We only report results for the best-performing constant step-size and a complete network connectivity. The number of messages per iteration S was fixed to 10.

Fig. 1 displays the convergence time of STAG in terms of the number of gradient evaluations needed to reach an ε-accurate minimizer to the objective f defined in (1). Observe that STAG still enjoys a low convergence time when M (the number of averaged gradient) is only half the size of the dataset. It also behaves well for small values of M and $\varepsilon = 10^{-2}$ because instability is contained within this threshold, but to obtain convergence under $\varepsilon = 10^{-3}$ we must use very low step sizes, implying much higher

convergence times. This is not the case for large values of M, where STAG fastly converges up to any arbitrarily-low ε.

Fig. 2 shows the convergence slope of AGSTAG for various network sizes N, where training examples are uniformly spread over the nodes. At each node, we chose $M = n_i/10$, that is, local gradients buffers are 90% smaller than local datasets. We can see that using a large network can drastically reduce the number of gradient evaluations per node. Besides, we observed that larger networks allowed greater constant step-sizes.

7 Conclusion

We introduced an asynchronous algorithm for distributed convex optimization called AGSTAG, that is made of two original parts: a Short-Term averaged gradient (STAG) optimization scheme and an asynchronous and decentralized Gossip averaging protocol. Unlike other methods, STAG turns the number of averaged gradients into a user-defined parameter, improving flexibility. The proposed asynchronous Gossip averaging protocol offers a simple and efficient communication mechanism to extend STAG to large scale distributed environments, as shown in our experimental evaluation.

References

[1] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli. Weighted gossip: Distributed averaging using non-doubly stochastic matrices.

[2] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, 2010.

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

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

[5] C. Hensel and H. Dutta. Gadget svm: a gossip-based sub-gradient svm solver. In *International Conference on Machine Learning (ICML), Numerical Mathematics in Machine Learning Workshop*, 2009.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[7] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[8] N. L. Roux, M. Schmidt, and F. R. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. *NIPS*, pages 2663–2671, 2012.

[9] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[10] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *J. Mach. Learn. Res.*, 14(1):567–599, Feb. 2013.

[11] P. Tseng. An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[12] K. I. Tsianos, S. Lawlor, and M. G. Rabbat. Push-sum distributed dual averaging for convex optimization. In *CDC*, pages 5453–5458, 2012.

[13] J. N. Tsitsiklis. *Problems in decentralized decision making and computation*. PhD thesis, Massachusetts Institute of Technology, 1984.