Distributed Nesterov-like Gradient Algorithms

Dušan Jakovetić, José M. F. Moura, and João Xavier

***Abstract*— In classical, centralized optimization, the *Nesterov gradient* algorithm reduces the number of iterations to produce an ϵ-accurate solution (in terms of the cost function) with respect to ordinary gradient from $O(1/\epsilon)$ to $O(1/\sqrt{\epsilon})$. This improvement is achieved on a class of convex functions with Lipschitz continuous first derivative, and it comes at a very small additional computational cost per iteration. In this paper, we consider *distributed* optimization, where nodes in the network cooperatively minimize the sum of their private costs subject to a global constraint. To solve this problem, recent literature proposes distributed (sub)gradient algorithms, that are attractive due to computationally inexpensive iterations, but that converge slowly–the ϵ error is achieved in $O(1/\epsilon^2)$ iterations. Here, building from the Nesterov gradient algorithm, we present a distributed, constant step size, Nesterov-like gradient algorithm that converges much faster than existing distributed (sub)gradient methods, with zero additional communications and very small additional computations per iteration k. We show that our algorithm converges to a solution neighborhood, such that, for a convex compact constraint set and optimized stepsize, the convergence time is $O(1/\epsilon)$. We achieve this on a class of convex, coercive, continuously differentiable private costs with Lipschitz first derivative. We derive our algorithm through a useful penalty, network's Laplacian matrix-based reformulation of the original problem (referred to as the clone problem) – the proposed method is precisely the Nesterov-gradient applied on the clone problem. Finally, we illustrate the performance of our algorithm on distributed learning of a classifier via logistic loss.**

I. INTRODUCTION

The gradient algorithm proposed by Nesterov [1] significantly reduces the convergence time with respect to the ordinary gradient method from $O(1/\epsilon)$ to $O(1/\sqrt{\epsilon})$, on a class of convex functions with Lipschitz continuous first derivative. In this paper, building from the ideas of the (centralized) Nesterov gradient, we propose *distributed* Nesterov-like gradient algorithms for cooperative optimization in networks.

Specifically, we consider the problem where N nodes in the network cooperatively minimize the sum of their private convex costs $\sum_{i=1}^{N} f_i(y)$ subject to a globally known constraint $y \in \mathcal{Y}$, where $\mathcal{Y} \subset \mathbb{R}^d$ is a closed convex set. (Here by private cost $f_i(\cdot)$ we mean that the function $f_i(\cdot)$ is known only by node i.) Existing distributed (sub)gradient algorithms, e.g., the algorithm proposed in [2] and extended and analyzed in [3], [2], [4], [5], [6], [7], and the one proposed in [8] and extended and analyzed in [9], [10], converge slowly. For example, assuming possibly non-differentiable, convex f_i's, with bounded gradients over the constraint set, algorithm [2] with constant step size α, after k iterations, has the error in the cost $O(\alpha+1/(\alpha k))$, which, for the optimized α, gives $O(1/\epsilon^2)$ convergence time. Reference [8] shows (under the same class of the f_i's) that the algorithm therein achieves ϵ-accuracy in $O(1/\epsilon^2)$; reference [8] provides a tight estimate of the hidden constant as a number of nodes N and the network topology.

In this paper, we propose a distributed Nesterov-like gradient algorithm, that converges much faster than existing distributed (sub)gradient algorithms: with the step size α and after k iterations, the error in the cost function is $O(\alpha+1/(\alpha k^2))$, which gives $O(1/\epsilon)$ convergence time. The improvement comes with no additional communication cost per iteration k, and with a very small additional computational cost per k, when compared with existing methods [2], [8]. We achieve this on a class of convex, coercive private cost functions $f_i(\cdot)$ that are continuously differentiable, have Lipschitz continuous first derivative, and are Lipschitz on the constraint set $\mathcal{Y}$. (The Lipschitz condition holds, e.g., for any continuously differentiable $f_i(\cdot)$ when $\mathcal{Y}$ is compact.) A major contribution of this paper is to derive a useful penalty, graph Laplacian-matrix based reformulation of the original problem – referred to as the clone problem – and relate the clone and the original problems. The reformulation allows for a novel interpretation of the algorithm in [2] as being the (ordinary) gradient method on the clone problem. Then, our Nesterov-based method arises naturally as a way to speed up the (ordinary) gradient method on the clone problem. We demonstrate the effectiveness of our algorithm for distributed learning of the best linear classifier via logistic loss. The simulation example demonstrates that our algorithm converges significantly faster (in terms of k) than other distributed (sub)gradient algorithms.

We propose both diminishing and constant step size variants of the algorithm; the diminishing step size variant converges to the exact optimal value $f^\star$ at rate $O(\log k/k)$ (see [11]). In this paper, we focus on the constant step size variant.

We make further comments on the literature. For the problem of minimizing the sum of private costs $\sum_{i=1}^{N} f_i(y)$

The work of the first, and the third authors is partially supported by: the Carnegie Mellon—Portugal Program under a grant from the Fundação para a Ciência e Tecnologia (FCT) from Portugal; by FCT grants CMU-PT/SIA/0026/2009; and by ISR/IST plurianual funding (POSC program, FEDER). The work of J. M. F. Moura is partially supported by NSF under grants CCF-1011903 and CCF-1018509, and by AFOSR grant FA95501010291. D. Jakovetić holds a fellowship from FCT.

D. Jakovetić is with the Institute for Systems and Robotics (ISR), Instituto Superior Técnico (IST), Technical University of Lisbon, Lisbon, Portugal, and with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA djakovet@andrew.cmu.edu

J. M. F. Moura is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA moura@ece.cmu.edu

J. Xavier is with the Institute for Systems and Robotics (ISR), Instituto Superior Técnico (IST), Technical University of Lisbon, Lisbon, Portugal jxavier@isr.ist.utl.pt

subject to $y \in \mathcal{Y}$, existing work proposes: distributed (sub)gradient algorithms, e.g., [2], [8], that have simple, computationally cheap iterations k, and do not require sofisticated programming, but converge slowly; and augmented Lagrangian (or ordinary Lagrangian) dual methods, e.g., [12], [13], that have, in general, computationally expensive iterations k, as they require solving certain local optimization problems at each node i, at each k, but can reduce the number of iterations k when compared to current distributed (sub)gradient methods. This paper focuses on *distributed (sub)gradient algorithms* and proposes an algorithm that converges faster than [2], [8] while maintaining approximately the same iterations' computational cost. Finally, we note that generally, for distributed optimization, algorithms different than the (sub)gradients and augmented Lagrangians have been developed: reference [14] considers the network flow problem – a different optimization problem than the one we consider here – and proposes a family of distributed algorithms that use Newton approximate directions.

The paper is organized as follows. The next paragraph introduces notation. Section II introduces the optimization and network models that we assume and presents our distributed algorithm. Section III analyzes the convergence speed of our algorithms, and Section IV illustrates their performance on the classification problem with logistic loss. Finally, Section V concludes the paper.

Throughout, we denote by: $\mathbb{R}^d$ the d-dimensional real coordinate space, $d \geq 1$; A_{ij} the entry in the i-th row and j-th column of a matrix A; a_i the i-th entry of a vector a; $(\cdot)^\top$ the transpose; $\|\cdot\| = \|\cdot\|_2$ the Euclidean (respectively, spectral) norm of its vector (respectively, matrix) argument (We note that $\|\cdot\|$ also denotes the modulus of a scalar throughout); $\lambda_i(\cdot)$ the i-th smallest eigenvalue; $|\cdot|$ the cardinality of a set; $\nabla \mathcal{J}(y)$ the gradient evaluated at y of a function $\mathcal{J} : \mathbb{R}^d \to \mathbb{R}$, $d \geq 1$; $\mathcal{N}(\mu, \sigma^2)$ the normal distribution with mean μ and variance σ^2. Finally, the notation $r(k) = O(q(k))$ means existence of a $K > 0$ such that $r(k) \leq \mu\, q(k)$, for some $\mu > 0$, for all $k \geq K$.

II. Distributed Nesterov-like Gradient algorithm

A. Optimization and network models

Optimization model. We consider a distributed optimization problem where N nodes cooperatively solve:

$$\begin{array}{ll} \text{minimize} & f(y) := \sum_{i=1}^N f_i(y) \\ \text{subject to} & y \in \mathcal{Y}, \end{array} \tag{1}$$

where $\mathcal{Y} \subset \mathbb{R}^d$ is a global constraint set, known by all nodes. The function $f_i : \mathbb{R}^d \to \mathbb{R}$ is *private*, i.e., known only by node i. We impose the following structure on f_i's and $\mathcal{Y}$:

Assumption 1 (a) For all i, f_i is convex, coercive, and Lipschitz with respect to the Euclidean $\|\cdot\|$ norm on the set $\mathcal{Y}$, i.e., there exists $G' \in (0, \infty)$, such that:

$$\|f_i(y) - f_i(z)\| \leq G' \, \|y - z\| \,, \; \forall y, z \in \mathcal{Y}.$$

(b) f_i is continuously differentiable, with Lipschitz continuous first derivative of constant $L_{f_i} \in (0, +\infty)$, i.e.,

$$\|\nabla f_i(y) - \nabla f_i(z)\| \leq L_{f_i} \|y - z\|, \quad \forall y, z \in \mathbb{R}^d.$$

(c) The set $\mathcal{Y}$ is closed, convex, and non-empty.

We note that the condition 1 (a) on Lipschitz continuity of $f_i(\cdot)$ on $\mathcal{Y}$ holds for any function $f_i(\cdot)$ that satisfies the other Assumptions in 1 when $\mathcal{Y}$ is a compact set. By Assumption 1, problem (1) is solvable, the optimal value $f^\star > -\infty$, and the solution set is non-empty and compact, e.g., [15]. For simplicity of presentation, we let $d = 1$ and $y \in \mathbb{R}$ throughout the paper, but our results hold for generic d. (Throughout, we denote by $\|\cdot\|$ both the Euclidean l_2-norm of a vector and the modulus of a scalar.)

Communication model. We associate with problem (1) a network $\mathcal{V}$ of N nodes, described by the graph $\mathcal{G} = (\mathcal{V}, E)$, where $E \subset \mathcal{V} \times \mathcal{V}$ is the set of edges.

Assumption 2 The graph $\mathcal{G}$ is connected, undirected, simple (no self/multiple links,) and it does not change in time.

Denote by $\mathcal{A}$ the adjacency matrix, defined by $\mathcal{A}_{ij} = 1$, if $\{i, j\} \in E$ and zero otherwise ($\mathcal{A}_{ii} = 0$, for all i); $\mathcal{D}$ the diagonal matrix with $\ell_i := \mathcal{D}_{ii} = \sum_{j \neq i} \mathcal{A}_{ij}$ equal to the degree of node i; and the graph Laplacian matrix $\mathcal{L} = \mathcal{D} - \mathcal{A}$.[1] Because the graph is connected, we have that the algebraic connectivity $\lambda_2(\mathcal{L}) > 0$.

B. The algorithm

We propose the following *distributed* constant step size Nesterov-like gradient algorithm to solve (1). Each node i updates over iterations k: 1) its estimate $x_i(k)$ of a solution; and 2) an auxiliary variable $y_i(k)$. With the initialization $x_i(0) = y_i(0) \in \mathcal{Y}$, the update rule at node i is:

$$\begin{aligned} x_i(k) &= P_{\mathcal{Y}}\{(1 - \ell_i\, w_0)\, y_i(k-1) + w_0 \sum_{j \in O_i} y_j(k-1) \\ &\quad - \alpha \nabla f_i(y_i(k-1))\} \\ y_i(k) &= x_i(k) + \frac{k-1}{k+2} \left(x_i(k) - x_i(k-1)\right), \; k = 1, 2, ..., \end{aligned} \tag{2}$$

where $P_{\mathcal{Y}}$ is the Euclidean projection onto the set $\mathcal{Y}$:

$$P_{\mathcal{Y}}\{z_i\} = \arg\min_{y_i \in \mathcal{Y}} \|y_i - z_i\|^2.$$

In (2), $\alpha > 0$ is the step size, $w_0 = \alpha\, \rho > 0$ is the averaging weight, $\rho > 0$ is a parameter, and O_i is the neighborhood set of node i (excluding i), and $\ell_i = |O_i|$ is its cardinality. The algorithm operation is summarized as follows. Each node i, at each iteration k: 1) broadcasts its variable $y_i(k-1)$ to all its neighbors $j \in O_i$; 2) receives $y_j(k-1)$ from all its neighbors $j \in O_i$; 3) updates $x_i(k)$ and $y_i(k)$ via (2). When updating $x_i(k)$, node i makes a weighted combination of its own and the neighbors' variables $y_j(k-1)$, whereby the neighbors' variables are weighted by a positive weight w_0.

[1] We can allow for a weighted, symmetric, adjacency matrix, so that $\mathcal{A}_{ij}$ for $\{i, j\} \in E$ is a generic positive number, and the $\mathcal{A}_{ij}$'s can be mutually different over different edges.

(We detail the choice of $w_0 > 0$ and $\alpha > 0$ later.) Note that the variables $x_i(k)$ are feasible, $x_i(k) \in \mathcal{Y}$, for all k, while the variables $y_i(k)$ may not be feasible. When compared to the existing projected (sub)gradient methods in [5] and [8], algorithm (2) introduces very small additional computational cost, namely the cost of updating the variable $y_i(k)$. In terms of the communication cost per k, with all three algorithms, each node broadcasts a single variable to all its neighbors. (With all three algorithms, the transmitted variable is of the same size.)

We now write algorithm (2) in matrix form. Collect the averaging weights in the $N \times N$ matrix W and introduce the vector function $F(x)$ as:

$$W = I - w_0 \mathcal{L} \tag{3}$$
$$F(x) = F(x_1, x_2, ..., x_N) = (f_1(x_1), f_2(x_2), ..., f_N(x_N))^\top.$$

Then, with $x_i(0) = y_i(0) \in \mathcal{Y}$, our distributed Nesterov-like gradient algorithm (2) in matrix form is:

$$\begin{aligned} x(k) &= \mathcal{P}_{\mathcal{Y}} \left\{ W y(k-1) - \alpha \nabla F(y(k-1)) \right\} \\ y(k) &= x(k) + \frac{k-1}{k+2} \left(x(k) - x(k-1) \right), \ k = 1, 2, ..., \end{aligned} \tag{4}$$

where

$$\mathcal{P}_{\mathcal{Y}}(y) = (P_{\mathcal{Y}}(y_1), P_{\mathcal{Y}}(y_2), ..., P_{\mathcal{Y}}(y_N))^\top$$

is the projection of $y = (y_1, ..., y_N)^\top$ on $\mathcal{Y}^N \subset \mathbb{R}^N$ – the N-fold Cartesian product of $\mathcal{Y}$'s.

C. Derivation of the algorithm

We now explain how we derive the structure of our algorithm. Our derivation is based on the following clone optimization problem:

$$\begin{array}{ll} \text{minimize} & \Psi_\rho(x) := \sum_{i=1}^N f_i(x_i) + \frac{\rho}{2} x^\top \mathcal{L} x \\ \text{subject to} & x \in \mathcal{Y}^N, \end{array} . \tag{5}$$

where $\mathcal{Y}^N$ denotes the Cartesian product $\mathcal{Y}^N = \mathcal{Y} \times ... \times \mathcal{Y}$ ($\mathcal{Y}$ repeated N times.) By Assumption 1, problem (5) is solvable and has a compact solution set. Denote by $\Psi_\rho^\star$ the optimal value of (5). We have that $\Psi_\rho^\star \leq f^\star$. Namely, for $y^\star$, a solution to (1), construct a vector $x^\bullet = (y^\star, y^\star, ..., y^\star) \in \mathbb{R}^N$; $x^\bullet$ is feasible for (5), and $f^\star = \Psi_\rho(x^\bullet) \geq \Psi_\rho^\star$.

Problem (5) is related to the original problem (1), as we explain now. Problem (5) assigns a local copy x_i of the global variable y to each node; the quadratic term

$$\frac{\rho}{2} x^\top \mathcal{L} x = \frac{\rho}{2} \sum_{\{i,j\} \in E} \|x_i - x_j\|^2$$

enforces that, for a large ρ, nodes' local copies at a solution $x^\star(\rho)$ are close to each other, i.e., $x_i^\star(\rho) \approx x_j^\star(\rho)$. Importantly, at any node i, $x_i^\star(\rho)$ is an approximate solution to the original problem (1). Now, consider an algorithm that updates $x(k)$ as follows. At iteration k, perform the projected gradient step with respect to $\Psi_\rho(\cdot)$:

$$x(k) = \mathcal{P}_{\mathcal{Y}} \left\{ x(k-1) - \alpha \nabla \Psi_\rho(x(k-1)) \right\}. \tag{6}$$

Recall W in (3). By rearranging the terms, we can see that (6) is rewritten as:

$$x(k) = \mathcal{P}_{\mathcal{Y}} \left\{ W x(k-1) - \alpha \nabla F(x(k-1)) \right\}, \tag{7}$$

and hence is distributed: to update $x_i(k)$, node i needs only its own estimate $x_i(k-1)$, the estimates $x_j(k-1)$ from its neighbors, and its local gradient $\nabla f_i(x_i(k-1))$. To increase the convergence speed of (7), we apply the fast Nesterov gradient step to $\Psi_\rho(\cdot)$ instead of the ordinary gradient step:

$$\begin{aligned} x(k) &= \mathcal{P}_{\mathcal{Y}} \left\{ y(k-1) - \alpha \nabla \Psi_\rho(y(k-1)) \right\} \\ y(k) &= x(k) + \frac{k-1}{k+2} \left(x(k) - x(k-1) \right). \end{aligned} \tag{8}$$

Equation (8), after rearranging the terms, and recalling W in (3), is exactly (4). As desired, the resulting method is completely distributed.

From the above derivations, we can see that our algorithm (4) is exactly the Nesterov gradient applied to the clone problem (5), and hence $\Psi_\rho(x(k))$ converges to the optimal value $\Psi_\rho^\star$. Now, for a large ρ and for all nodes i, $f(x_i(k))$ converges to a neighborhood of $f^\star$, as desired, and as will be shown in the next section.

The intermediate algorithm in our derivation in (7) is the distributed (sub)gradient proposed in [2]. Hence, the algorithm in [2] can be interpreted as the ordinary gradient applied to the clone problem (5). It is natural to expect that our algorithm (4) converges faster than the one proposed in [2], because the Nesterov gradient converges faster than the ordinary gradient.

Algorithm (4) converges to a neighborhood of $f^\star$. We can force the algorithm to converge exactly to $f^\star$ by letting the function $\Psi_\rho(\cdot)$ and the step size α in (8) to be time varying. This is considered in [11].

We finish the Section by detailing the choice of parameters α and ρ. As required by the Nesterov gradient algorithm, the step size $\alpha \leq 1/L_\Psi$, where L_Ψ is a Lipschitz constant of the gradient of $\Psi_\rho(\cdot)$, which we can take as $L_\Psi = \max_{i=1,...,N} L_{f_i} + \rho \lambda_N(\mathcal{L})$. Thus:

$$\alpha \leq \frac{1}{\max_{i=1,...,N} L_{f_i} + \rho \lambda_N(\mathcal{L})}. \tag{9}$$

Note that (9) imposes a condition on the averaging weight $w_0 \leq \frac{\rho}{\max_{i=1,...,N} L_{f_i} + \rho \lambda_N(\mathcal{L})} < \frac{1}{\lambda_N(\mathcal{L})}$.

III. Convergence Analysis

We now study the convergence of the proposed distributed algorithm under constant step size. We have the following Theorem. (Note that we express the results below in terms of the *unnormailized* optimality gap $f(x_i) - f^\star$ like in, e.g., [2], while it is also possible (see, e.g., [8]) to expresses the results in terms of the normalized optimality gaps $\frac{1}{N}(f(x_i) - f^\star)$.)

Theorem 1 Consider algorithm (2) under Assumptions 1 and 2, with the step-size

$$\alpha = \frac{1}{\max_{i=1,...,N} L_{f_i} + \rho \lambda_N(\mathcal{L})}.$$

Then:

(a) for all $i = 1, ..., N$, for all $k = 1, 2, ...$:

$$f(x_i(k)) - f^\star \leq \frac{N^2(N-1)(G')^2}{2\,\rho} \tag{10}$$
$$+ \quad \frac{[2\ (\max_i L_{f_i} + \rho\lambda_N(\mathcal{L}))]\ \|x(0) - x^\star(\rho)\|^2}{k^2}.$$

(b) Let $\mathcal{Y}$ be a compact set with $\|y\| \leq B'$, for all $y \in \mathcal{Y}$, and fix the desired accuracy $\epsilon > 0$. Then, for:

$$\rho = \rho(\epsilon) = \frac{N^2(G')^2(N-1)}{\epsilon},$$

we have: $f(x_i(k)) - f^\star \leq \epsilon$, $\forall k \geq k_0(\epsilon), \forall i$, where:

$$k_0(\epsilon) = \frac{4N^2\sqrt{\lambda_N(\mathcal{L})}G'B'}{\epsilon} + \frac{4\sqrt{N}B'\sqrt{\max_i L_{f_i}}}{\sqrt{\epsilon}},$$

i.e., the ϵ-accuracy is achieved after at most $k_0(\epsilon)$ iterations.

Theorem 1 says that, with our proposed algorithm, for the compact set $\mathcal{Y}$ and appropriately set ρ, the convergence time for ϵ-accuracy in the cost function is $O(1/\epsilon)$. Hence, we reduce the convergence time with respect to [5], [8] by paying zero price in terms of the communication cost per k. Our simulation example shown here, as well as all other examples that we have tested, confirm that our algorithm significantly reduces convergence time. Of course, the improvement comes at the cost of restricting the admissible cost functions with respect to [5], [8]. Our admissible cost functions are continuously differentiable with Lipschitz continuous first derivative.

Proof: [Proof of Theorem 1] We first prove claim (a). The proof consists of two parts. First, we use the convergence results for the Nesterov method [16], [17] to estimate the error in terms of the clone function $\Psi_\rho(x(k)) - \Psi_\rho^\star$. Second, we relate the clone error $\Psi_\rho(x(k)) - \Psi_\rho^\star$ and the true error at any node j: $f(x_j(k)) - f^\star$.

Clone function error. By the convergence results for the Nesterov gradient method [16], and noting that the Lipschitz constant of Ψ_ρ equals $\max_{i=1,...,N} L_{f_i} + \rho\lambda_N(\mathcal{L})$, we have that, for all k:

$$\Psi_\rho(x(k)) - \Psi_\rho^\star \tag{11}$$
$$\leq \quad \frac{[2\ (\max_{i=1,...,N} L_{f_i} + \rho\lambda_N(\mathcal{L}))]\ \|x(0) - x^\star(\rho)\|^2}{k^2}$$
$$=: \quad \frac{C_\Psi}{k^2}.$$

Relating the clone and the true errors. We now fix a node j and start with the clone error:

$$\Psi_\rho(x(k)) - \Psi_\rho^\star \tag{12}$$
$$= \quad \sum_{i=1}^N f_i(x_i(k)) + \frac{1}{2}\,\rho\, x(k)^\top \mathcal{L}\, x(k) - \Psi_\rho^\star. \tag{13}$$

Consider equation (12), and fix a node j at which we want to estimate the true error. By Lipschitz continuity of $f_i(\cdot)$ on $\mathcal{Y}$ and by the fact that $x_i(k), x_j(k) \in \mathcal{Y}$, we have that:

$$\|f_i(x_i(k)) - f_i(x_j(k))\| \leq G'\|x_i(k) - x_j(k)\|. \tag{14}$$

Now, adding and subtracting $f^\star$ from (12) while using the fact that $\Psi_\rho^\star \leq f^\star$, and using (14) gives:

$$\Psi_\rho(x(k)) - \Psi_\rho^\star \geq \sum_{i=1}^N f_i(x_j(k)) - f^\star \tag{15}$$
$$- \quad \sum_{i=1}^N G'\|x_i(k) - x_j(k)\| + \frac{1}{2}\,\rho\, x(k)^\top \mathcal{L} x(k)$$
$$\geq \quad f(x_j(k)) - f^\star - N\,G'\left(\max_{i:\, i\neq j}\|x_i(k) - x_j(k)\|\right)$$
$$+ \quad \frac{1}{2}\,\rho\, x(k)^\top \mathcal{L} x(k).$$

We now lower bound the quadratic form

$$x^\top \mathcal{L} x = \sum_{\{i,j\}\in E} \|x_i - x_j\|^2,$$

for any $x \in \mathbb{R}^N$. Let $\max_{i:\, i\neq j}\|x_i - x_j\| =: \|x_s - x_j\|$. Because the graph is connected, there is a path of length D from node s to node j, say $(s = i_1) \to i_2 \to ... \to (i_{D+1} = j)$, where $1 \leq D \leq N-1$. Then:

$$x^\top \mathcal{L} x \geq \|x_s - x_{i_2}\|^2 + ... + \|x_{i_D} - x_j\|^2$$
$$= \quad D\left(\frac{1}{D}\|x_s - x_{i_2}\|^2 + ... + \frac{1}{D}\|x_{i_D} - x_j\|^2\right)$$
$$\geq \quad D\left\|\frac{1}{D}(x_s - x_{i_2}) + ... + \frac{1}{D}(x_{i_D} - x_j)\right\|^2 \tag{16}$$
$$= \quad \frac{1}{D}\|x_s - x_j\|^2 \geq \frac{1}{(N-1)}\|x_s - x_j\|^2,$$

where we use the fact that, for any path, $D \leq (N-1)$, and inequality (16) uses convexity of the quadratic function $z \mapsto \|z\|^2$. Using the latter bound for $x = x(k)$, we have:

$$\Psi_\rho(x(k)) - \Psi_\rho^\star \geq f(x_j(k)) - f^\star$$
$$- \quad NG'\left(\max_{i:i\neq j}\|x_i(k) - x_j(k)\|\right)$$
$$+ \quad \frac{1}{2}\frac{\rho}{(N-1)}\left(\max_{i:i\neq j}\|x_i(k) - x_j(k)\|\right)^2 \tag{17}$$
$$\geq \quad f(x_j(k)) - f^\star - \frac{N^2(N-1)(G')^2}{2\,\rho}, \tag{18}$$

where (18) follows by maximizing $NG'\delta - \frac{1}{2}\frac{\rho}{(N-1)}\delta^2$ over $\delta \in \mathbb{R}$. Equation (18) allows us to relate the clone and the true errors:

$$f(x_j(k)) - f^\star \leq \Psi_k(x(k)) - \Psi_\rho^\star + \frac{N^2(N-1)(G')^2}{2\,\rho}. \tag{19}$$

Equation (19), combined with (11), completes the proof of part (a).

We now prove part (b). Let the set $\mathcal{Y}$ be compact, such that $\|y\| \leq B'$, for all $y \in \mathcal{Y}$. Denote by $B := \sqrt{N}B'$. Then,

$$\|x(0) - x^\star(\rho)\| \leq \|x(0)\| + \|x^\star(\rho)\| \leq 2B,$$

which gives:

$$f(x_i(k)) - f^\star \leq \frac{N^2(N-1)(G')^2}{2\,\rho} \quad (20)$$

$$+ \quad \frac{[2\ (\max_i L_{f_i} + \rho\lambda_N(\mathcal{L}))]\ (2B)^2}{k^2}$$

$$= \quad \frac{C_1}{\rho} + \rho\frac{C_2}{k^2} + \frac{C_3}{k^2}, \quad (21)$$

with $C_1 = \frac{N^2(G')^2(N-1)}{2}$, $C_2 = 8B^2\lambda_N(\mathcal{L})$, and $C_3 = 8(\max_i L_{f_i})B^2$. Now, fix an $\epsilon > 0$, and consider the time $K(\rho)$–the smallest time k at which $f(x_i(k)) - f^\star \leq \epsilon$, for all i. Our goal is then to find $\rho > 0$ that minimizes $K(\rho)$ and to find the corresponding minimal value $K^\star(\epsilon)$. Instead of finding the actual minimum, it suffices for our purpose to find an upper bound on $K^\star(\epsilon)$, and a sub-optimal ρ, which we call $\rho(\epsilon)$. By (21), we have that

$$K^2(\rho) \leq \frac{\rho^2 C_2 + \rho\, C_3}{\epsilon\rho - C_1} =: \mathcal{M}(\epsilon, \rho),\ \rho > C_1/\epsilon. \quad (22)$$

Now, set $\rho(\epsilon) := \frac{2C_1}{\epsilon} = \frac{N^2(G')^2(N-1)}{\epsilon}$. This value, when plugged in the right hand side of (22), gives:

$$\mathcal{M}(\epsilon, \rho(\epsilon)) = \frac{4C_1C_2}{\epsilon^2} + \frac{2C_3}{\epsilon}.$$

From above, using inequality $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$, $x, y \geq 0$, we can conclude that:

$$K^\star(\epsilon) \leq \sqrt{\mathcal{M}(\epsilon, \rho(\epsilon))} \leq \frac{2\sqrt{C_1C_2}}{\epsilon} + \frac{\sqrt{2C_3}}{\sqrt{\epsilon}},$$

which, substituting the values of C_1, C_2, and C_3, yields the result (b). ∎

Remark. The bounds in Theorem 1 can be improved using the network's diameter Diam. Namely, replacing in (16) $\frac{\|x_s - x_j\|^2}{N-1}$ by $\frac{\|x_s - x_j\|^2}{\mathrm{Diam}}$, we obtain, for part (a) of Theorem 1 (equation (10)), that the term $\frac{N^2(N-1)(G')^2}{2\rho}$ is replaced by: $\frac{N^2\mathrm{Diam}(G')^2}{2\rho}$. Likewise, in part (b) of Theorem 1, setting $\rho'(\epsilon) = \frac{N^2\mathrm{Diam}(G')^2}{\epsilon}$, the convergence time is $k_0'(\epsilon) = \frac{4N^{3/2}\mathrm{Diam}^{1/2}\sqrt{\lambda_N(\mathcal{L})}G'B'}{\epsilon} + \frac{4\sqrt{N}B'\sqrt{\max_i L_{f_i}}}{\sqrt{\epsilon}}$.

IV. Simulation example

We demonstrate with an example of distributed learning of the best linear classifier via logistic loss that our algorithm converges much faster than existing distributed (sub)gradient methods [5], [8]. Training data is distributed across nodes in the network; each node has N_s data samples, $\{a_{ij}, b_{ij}\}_{j=1}^{N_s}$, where $a_{ij} \in \mathbb{R}^m$ is a feature vector (data vector,) and $b_{ij} \in \{-1, +1\}$ is the class label of the vector a_{ij}. Based on the data samples available at all nodes, we want to learn (in a distributed way) the linear classifier $a \mapsto \mathrm{sign}\left(a^\top x' + x''\right)$, i.e., to determine a sparse vector $x' \in \mathbb{R}^m$ and a scalar $x'' \in \mathbb{R}$, that minimizes the logistic loss:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N \sum_{j=1}^{N_s} \phi\left(-b_{ij}(a_{ij}^\top x' + x'')\right) \\ \text{subject to} & \|x'\| \leq c',\ x'' \in \mathbb{R}. \end{array} \quad (23)$$

Here $\phi(t) = \log(1 + e^{-t})$, and $\|x'\| \leq c' = 100$ is the regularization.

Setup. We consider a connected network with $N = 20$ nodes and 86 undirected links. The network is a geometric graph: nodes are uniformly randomly placed on a unit 2D square and the pairs of nodes whose distance is less than a radius are connected by an edge. We generate feature vectors a_{ij} independently over i and j, where we draw each entry of a_{ij} from $\mathcal{N}(0, 1)$. Each node has $N_s = 5$ data samples. We generate the "true" vector $x_0 = ((x')^\top, x'')^\top \in \mathbb{R}^4$ by drawing its entries independently from $\mathcal{N}(0, 1)$. The class labels are: $b_{ij} = \mathrm{sign}\left((x_0')^\top a_{ij} + x_0'' + \delta_{ij}\right)$, where δ_{ij}'s are drawn independently from $\mathcal{N}(0, 0.01)$. We numerically evaluate the optimal value $f^\star$ via the centralized projected gradient algorithm.

We compare our proposed distributed Nesterov-like gradient algorithm with the distributed (sub)gradient algorithm [2] and the distributed dual averaging algorithm [8]. As a metric, we use the relative error in the objective function averaged across nodes:

$$\mathrm{e}_f(k) = \frac{1}{N}\sum_{i=1}^N \frac{f(x_i(k)) - f^\star}{f^\star},\ f^\star \neq 0. \quad (24)$$

All three algorithms involve, as an intermediate step, a local averaging process. With all three algorithms, we use the Metropolis averaging weights: $W_{ij} = 1/(1 + \max(\ell_i, \ell_j))$, for $\{i, j\} \in E$, where we recall that ℓ_i is node i's degree; for $i \neq j$, $\{i, j\} \notin E$, $W_{ij} = 0$; and $W_{ii} = 1 - \sum_{j \neq i} W_{ij}$, for all i. Note that this corresponds to a straightforward generalization of our algorithm (2) to non-uniform weights w_0. With the algorithm in [8], we use a quadratic proximal function $z \mapsto \frac{1}{2}\|z\|^2$. With all algorithms, we use a constant step size $\alpha_k = \alpha$. In this situation, all the three algorithms converge to a solution neighborhood. For each algorithm, we adjust the step size so that our algorithm saturates at a lower level e_f than [5], [8]. Note that this is in favor of [5], [8], because, to achieve the same precision as our algorithm, we would have to further reduce the step size of [5], [8] and make these algorithms slower. We note that all three algorithms have similar computational cost per iteration k, and the same communication cost.

Results. Figure 1 plots $\mathrm{e}_f(k)$ versus the iteration number k (2), and the algorithms in [2] and [8]. We can see that our algorithm (2) performs much better than the other two: to achieve precision of $\mathrm{e}_f \approx 0.002$, (2) takes about 600 iterations, while the algorithms in [2] and [8] require more than 30000 iterations. Thus, (2) reduces the number of iterations more than 50 times, when compared with [2] and [8]. We also show the results for the diminishing step-sizes (Figure 2,) under which algorithms [2] and [8] converge to the exact value $f^\star$. Although we did not show here that our algorithm (2) converges exactly to $f^\star$ under the diminishing step size, simulations here suggest that the algorithm indeed converges to $f^\star$. With our algorithm, we set $\alpha_k = 1/k$. With the algorithms in [2] and in [8], we set $\alpha_k = 1/\sqrt{k}$. The best choice of α_k for [2], [8] under the function class

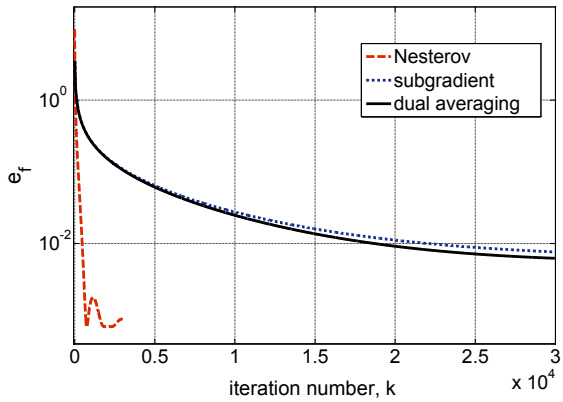

Fig. 1. Relative error in the objective function (24) versus iteration number k for the proposed algorithm in (2), dual averaging [8], and (sub)gradient [2], with constant step sizes.

that we study has not been proposed; [8] considers a wider class of functions than we do here (see [8] for details) and shows that the decay of order $1/\sqrt{k}$ with the algorithm therein is optimal; we thus set the decay order $1/\sqrt{k}$ for both [2] and [8]. Reference [2] does not recommend the decay order for the diminishing step size. We can see that our proposed algorithm again converges much faster than other algorithms. Interestingly, in the simulation shown in Figure 2, our algorithm converges linearly (geometrically) to the solution, at least in the iterations range $k = 5000$–$k = 15000$; we note that, in general, even when all the f_i's are strongly convex, the algorithm is not guaranteed to achieve the global linear convergence rate.

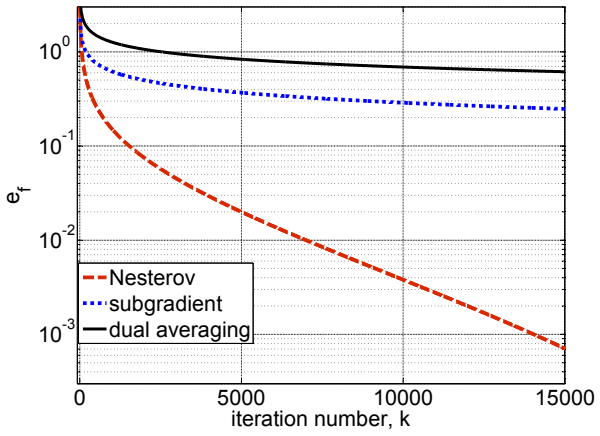

Fig. 2. Relative error in the objective function in (24) versus iteration number k for the proposed algorithm (2), dual averaging [8], and (sub)gradient [2], with diminishing step sizes.

V. Conclusion

We considered distributed optimization in networks, where N nodes cooperatively minimize the sum of their private cost functions subject to a a globally known constraint. We proposed a distributed Nesterov-like gradient algorithm that, when compared with existing (sub)gradient algorithms, significantly reduces the number of iterations for convergence without introducing additional communications per iteration k, and by maintaining computationally cheap and simple iterations, i.e., introducing very little additional computations per k. Our algorithm with step size α reduces the error in the cost function to $O(\alpha + 1/(\alpha k^2))$ after k iterations. This, for the optimized α, gives $O(1/\epsilon)$ convergence time for an ϵ–accurate solution, while existing distributed (sub)gradient algorithms require much longer time $O(1/\epsilon^2)$. We prove the result for convex, coercive, continuously differentiable private costs with Lipschitz continuous first derivative and a compact constraint set $\mathcal{Y}$. A simulation example for distributed learning of a classifier via logistic loss demonstrates the effectiveness of our distributed algorithm.

References

[1] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $o(1/k^2)$," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983, (in Russian).

[2] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, January 2009.

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

[4] I. Lobel and A. Ozdaglar, "Convergence analysis of distributed subgradient methods over random networks," in *46th Annual Allerton Conference onCommunication, Control, and Computing*, Monticello, Illinois, September 2008, pp. 353 – 360.

[5] A. Nedic, A. Ozdaglar, and A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, April 2010.

[6] S. S. Ram, A. Nedic, and V. Veeravalli, "Asynchronous gossip algorithms for stochastic optimization," in *CDC '09, 48th IEEE International Conference on Decision and Control*, Shanghai, China, December 2009, pp. 3581 – 3586.

[7] A. Nedic, "Asynchronous broadcast-based convex optimization over a network," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, June 2011.

[8] J. Duchi, A. Agarwal, and M. Wainwright, "Dual averaging for distributed optimization: Convergence and network scaling," *to appear in IEEE Transactions on Automatic Control*, January 2012.

[9] K. Tsianos and M. Rabbat, "Distributed dual averaging for convex optimization under communication delays," in *submitted to the American Control Conference*, 2012.

[10] ——, "Distributed consensus and optimization under communication delays," in *to apper in proc. 49th Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, Sept. 2011.

[11] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," November 2011, available at: http://arxiv.org/pdf/1112.2972.pdf.

[12] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, pp. 5262–5276, November 2010.

[13] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Cooperative convex optimization in networked systems: Augmented Lagrangian algorithms with directed gossip communication," *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3889–3902, August 2011.

[14] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network optimization," in *ACC '11, American Control Conference*, San Francisco, California, June 2011, pp. 2663–2668.

[15] A. Nedic, "Optimization I," August 2008, lecture notes. [Online]. Available: https://netfiles.uiuc.edu/angelia/ www/optimization_one.pdf

[16] L. Vandenberghe, "Optimization methods for large-scale systems," 2010, lecture notes, available at: http://www.ee.ucla.edu/ vandenbe/ ee236c.html.

[17] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.