Manifold Identification for Ultimately Communication-Efficient Distributed Optimization

Yu-Sheng Li [*1] **Wei-Lin Chiang** [*1] **Ching-pei Lee** [2]

Abstract

This work proposes a progressive manifold identification approach for distributed optimization with sound theoretical justifications to greatly reduce both the rounds of communication and the bytes communicated per round for partly-smooth regularized problems such as the ℓ_1- and group-LASSO-regularized ones. Our two-stage method first uses an inexact proximal quasi-Newton method to iteratively identify a sequence of low-dimensional manifolds in which the final solution would lie, and restricts the model update within the current manifold to gradually lower the order of the per-round communication cost from the problem dimension to the dimension of the manifold that contains a solution and makes the problem within it smooth. After identifying this manifold, we take superlinear-convergent truncated semismooth Newton steps computed by preconditioned conjugate gradient to largely reduce the communication rounds by improving the convergence rate from the existing linear or sublinear ones to a superlinear rate. Experiments show that our method can be orders of magnitudes lower in the communication cost and an order of magnitude faster in the running time than the state of the art.

1. Introduction

Distributed computing that simultaneously utilizes multiple machines is essential for dealing with the huge volume of data faced nowadays. Therefore, distributed optimization has become an active research topic in machine learning in recent years. In distributed optimization, the additional computing power and storage from multiple machines can greatly reduce the time for computation, memory access, and disk I/O. On the other hand, expensive inter-machine communication is frequently needed to synchronize the current model and calculate the update steps, so the communication cost, which can be an order of magnitude more expensive than local computation, usually becomes the bottleneck.

It is clear that the overall communication cost is proportional to both (i) the number of communication rounds and (ii) the cost per round, so reducing either factor can lower the overall expense on communication. State-of-the-art communication-efficient distributed optimization methods such as (Zhang & Xiao, 2015; Zheng et al., 2017; Lee et al., 2017; Lee & Chang, 2019; Lee et al., 2019) mainly focused on improving the former, which is closely related to the iteration complexity or the convergence speed. On the other hand, the reduction of the latter is usually achieved by compression or coding techniques that communicate inexact information using fewer bytes (Aji & Heafield, 2017; Lin et al., 2017; Wen et al., 2017; Chen et al., 2018). These approaches can harm the convergence speed of the optimization process due to the introduced noise, so the overall communication cost might not be reduced because of the increased communication rounds, and relevant study is restricted to specific problem classes with limited theoretical support. This work aims at decreasing both factors simultaneously through a two-stage progressive manifold identification approach with thorough theoretical guarantees.

We consider a distributed setting of minimizing the following regularized problem using K machines.

$$\min_{\boldsymbol{w}\in\Re^d} \quad F(\boldsymbol{w}) \coloneqq \sum_{k=1}^{K} f_k(\boldsymbol{w}) + \Psi(\boldsymbol{w}), \tag{1}$$

where each f_k is differentiable and exclusively available on the kth machine, their sum is Lipschitz-continuously-differentiable (sometimes called smooth) but might not be convex, and the regularization term Ψ is convex, proper, and closed but possibly nonsmooth. We assume further that (1) has a solution set $\Omega \neq \emptyset$ and Ψ is partly smooth (Lewis, 2002) everywhere. Loosely speaking, a convex function

[*]Equal contribution [1]Department of Computer Science, National Taiwan University, Taipei, Taiwan [2]Department of Mathematics & Institute for Mathematical Sciences, National University of Singapore, Singapore. Correspondence to: Yu-Sheng Li <r07922087@csie.ntu.edu.tw>, Wei-Lin Chiang <r06922166@csie.ntu.edu.tw>, Ching-pei Lee <leechingpei@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Ψ is partly smooth at a point $\boldsymbol{w}$ relative to a $\mathcal{C}^2$ manifold $\mathcal{M}$ if $\Psi|_{\mathcal{M}}$ is $\mathcal{C}^2$ around $\boldsymbol{w}$ and for directions leaving $\mathcal{M}$, the change of Ψ is sharp. A more technical definition is in Definition 3.5. We focus on the case in which the manifolds are translates of subspaces of $\Re^d$ such that for some given matrices A_i, for each point $\boldsymbol{w}$ there is an index set $I_{\boldsymbol{w}}$ such that the manifold

$$\mathcal{M}_{\boldsymbol{w}} := \{\bar{\boldsymbol{w}} \mid A_i\bar{\boldsymbol{w}} = A_i\boldsymbol{w} \quad \forall i \in I_{\boldsymbol{w}}\} \tag{2}$$

contains $\boldsymbol{w}$ and makes $\Psi|_{\mathcal{M}_{\boldsymbol{w}}}$ $\mathcal{C}^2$ around $\boldsymbol{w}$. For example, when $\Psi(\cdot) = \|\cdot\|_1$, each A_i is the unit vector of the ith coordinate and $I_{\boldsymbol{w}} = \{i \mid \boldsymbol{w}_i = 0\}$.

An important class of partly smooth functions is the sparsity-inducing penalty functions such that Ψ is block-separable and for each block, the only point of nonsmoothness is the origin. Examples in machine learning include the ℓ_1 and the group-LASSO regularizations that promote sparsity (Tibshirani, 1996; Meier et al., 2008). Partly smoothness and a mild regularity condition (see (19)) ensures that an optimal solution lies in a low-dimensional smooth manifold, and identifying which helps us lower the communication cost.

We utilize partly smoothness to propose a two-stage progressive manifold identification approach for (1) that is ultimately communication-efficient and call it MADPQN (Manifold-Aware Distributed Proximal Quasi-Newton). The first stage is a trust-region-like distributed inexact proximal quasi-Newton method that predicts on the fly the manifold $\mathcal{M}^*$ in which a final solution would lie, and utilizes the current prediction to gradually reduce the communication cost from the $O(d)$ in existing work to $O(|\mathcal{M}^*|)$. Empirically, $|\mathcal{M}^*| \ll d$ is often observed. This stage conducts progressive manifold selection such that at each iteration, we identify a sub-manifold of the current one and confine the next iterate to it to keep the communication cost low. After the manifold becomes fixed, the second stage applies a superlinear-convergent truncated semismooth Newton (SSN) method with preconditioned conjugate gradient (PCG) on the problem restricted to the manifold. When the manifold contains an optimal solution, in contrast to the existing sublinear or linear rates, the superlinear convergence of the SSN steps greatly cuts the number of communication rounds while maintaining the $O(|\mathcal{M}^*|)$ per-round communication cost, making MADPQN even more communication-efficient, especially for obtaining high-precision solutions.

Contributions. This paper is the first to utilize manifold identification in distributed optimization for improving communication efficiency systematically. Unlike existing work that only switches to smooth optimization after the final manifold is identified but does nothing prior to that, our progressive strategy utilizes the intermediate manifolds identified in the process to reduce the costs of both computation and communication throughout. Our two-stage approach accelerates the optimization, whether the final manifold is of very low dimension or not. For the former case, the first stage makes the communication cost per round much smaller; for the latter, we reach the final manifold swiftly and the fast-convergent second stage then greatly cuts the number of communication rounds. In theory, unlike existing approaches that focus on only one factor, MADPQN has both lower per-round communication cost and fewer communication rounds such that the asymptotic communication complexity to achieve an ϵ-accurate solution for (1) can be as low as $O(|\mathcal{M}^*| \log\log(1/\epsilon))$, in contrast to the $O(d\log(1/\epsilon))$ or $O(d/\epsilon)$ cost of existing approaches. Experiments show that MADPQN can be more orders of magnitudes more communication-efficient than the state of the art and the running time can be an order of magnitude faster.

Organization. This paper is organized as follows. In Section 2, we give details of the proposed MADPQN algorithm. Section 3 then provides theoretical supports, including its convergence rate and the ability for manifold identification. Related research is reviewed in Section 4. Experiments in Section 5 showcase the empirical performance of MADPQN. Section 6 then concludes this work. Due to the space limit, proofs, implementation details, and additional experiments are all put in the appendices. Based on this study, we have released a package for use at `http://www.github.com/leepei/madpqn/`.

2. An Ultimately Communication-efficient Distributed Optimization Algorithm

MADPQN has three main components–(1) inexact proximal quasi-Newton, (2) progressive manifold selection, and (3) local acceleration through smooth optimization. Our two-stage method uses the first two components at the first stage, and at the second stage we switch to the third component. We will first introduce each component respectively, and then combine them into an integrated algorithm.

2.1. Where Do Communication Costs Occur?

Communication is the usual bottleneck in distributed optimization. We discuss where the major communication takes place. In our description, we assume an *all-reduce* model, meaning that each machine serves simultaneously as a master and a worker, such that each machine has its own local data, but all other information is global and all machines conduct the same operations.

We see that the summation in (1) requires distributing $\boldsymbol{w}$ to all machines, which takes a round of communication with cost up to $O(d)$. Summing the components up then takes an additional $O(1)$ communication cost.

In most efficient algorithms, we need to compute the gradient of the smooth part $f := \sum f_k$, which is of the form

$$\nabla f(\boldsymbol{w}) = \sum_{k=1}^{K} \nabla f_k(\boldsymbol{w}). \tag{3}$$

Since $\boldsymbol{w}$ has been made available on all machines when evaluating $F(\boldsymbol{w})$, computing (3) through all-reduce takes a round of $O(d)$ communication. Hence, the major communication at each iteration occurs when synchronizing $\boldsymbol{w}$ on all machines and computing the gradient, and each communicates a d-dimensional vector. A goal of our method design is reducing the size of the vectors to be communicated.

2.2. Distributed Inexact Proximal Quasi-Newton

We give an overview of the inexact distributed proximal quasi-Newton method (DPLBFGS) by Lee et al. (2019) and our modifications. This is the major building block of the first stage of our algorithm. We adopt DPLBFGS because of its fast convergence as a second-order method that can effectively reduce the communication rounds, and solving its subproblem requires little communication.

At the tth iteration, given the current iterate $\boldsymbol{w}^t$, DPLBFGS solves the following subproblem approximately to obtain a tentative update direction $\boldsymbol{p}$:

$$\min_{\boldsymbol{p}\in\mathbb{R}^d} Q_{H_t}(\boldsymbol{p};\boldsymbol{w}^t) := \nabla f(\boldsymbol{w}^t)^\top \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^\top H_t \boldsymbol{p} + \Psi(\boldsymbol{w}^t+\boldsymbol{p}) - \Psi(\boldsymbol{w}^t), \tag{4}$$

where H_t is the LBFGS approximation of the Hessian that we will give details below.

After $\boldsymbol{p}$ is obtained, we use the trust-region variant in (Lee et al., 2019) to ensure sufficient function decrease, as this approach can identify a manifold $\mathcal{M}^*$ containing a solution $\boldsymbol{w}^* \in \Omega$ to (1) within finite iterations (see Section 3). Given any symmetric and positive definite matrix H_t for defining the subproblem (4) and parameters $\sigma_1 \in (0,1]$ and $\theta > 1$, we accept the direction $\boldsymbol{p}$ as the update direction $\boldsymbol{p}^t$ if

$$F\left(\boldsymbol{w}^t+\boldsymbol{p}\right) \le F\left(\boldsymbol{w}^t\right) + \sigma_1 Q_{H_t}\left(\boldsymbol{p};\boldsymbol{w}^t\right); \tag{5}$$

otherwise we repeatedly update H_t by

$$H_t \leftarrow \theta H_t \tag{6}$$

and resolve (4) defined by the new H_t until (5) holds. The iterate is then updated by $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{p}^t$. We will denote the initial choice for H_t as $\tilde{H}_t$ and the final one as $\hat{H}_t$.

DPLBFGS uses the LBFGS approximation of the (generalized) Hessian (Liu & Nocedal, 1989) as $\tilde{H}_t$. Using the compact representation (Byrd et al., 1994), given a prespecified integer $m > 0$, let $m(t) := \min(m, t)$, and define

$$\boldsymbol{s}_i := \boldsymbol{w}^{i+1} - \boldsymbol{w}^i, \quad \boldsymbol{y}_i := \nabla f(\boldsymbol{w}^{i+1}) - \nabla f(\boldsymbol{w}^i), \quad \forall i,$$

Algorithm 1: MADPQN: An ultimately communication-efficient two-stage manifold identification method for (1)

Given $\theta > 1, \sigma_1 \in (0,1), \tilde{\epsilon}, \delta > 0$, integers $m, S, T > 0$, a sequence $\{\epsilon_j\} \ge \epsilon > 0$, an initial point $\boldsymbol{w}^{0,0}$

for $j = 0, 1, 2, \ldots$ **do**
  $U_{j,0} \leftarrow \emptyset, Z_{j,0} \leftarrow \emptyset, \tilde{\mathcal{M}}_{j,-1} = \tilde{\mathcal{M}}_{j,0} = \Re^d$
  for $t = 0, 1, 2, \ldots$ **do**
    Compute $P_{\tilde{\mathcal{M}}_{j,t-1}}(\nabla f(\boldsymbol{w}^{j,t}))$ by (13)
    if $t > 0$ **then**
      Compute $P_{\tilde{\mathcal{M}}_{j,t-1}}(\boldsymbol{y}_{t-1})$, and $\gamma_{j,t}$ by (15)
      if (14) *holds for* $i = t-1$ **then**
        Update $U_{j,t}$ by (8) and update $\tilde{Z}_{j,t}$ using $\tilde{Z}_{j,t-1}$ and $\boldsymbol{s}_{j,t-1}^\top U_t$
      end
      Select $\mathcal{M}_{j,t} \subset \mathcal{M}_{j,t-1}$ with $\boldsymbol{w}^{j,t} \in \mathcal{M}_{\boldsymbol{w}^{j,t}} \subseteq \mathcal{M}_{j,t}$ using $P_{\tilde{\mathcal{M}}_{j,t-1}}(\nabla f(\boldsymbol{w}^{j,t}))$ and decide a basis matrix $B_{j,t}$ of $\tilde{\mathcal{M}}_{j,t}$
    end
    if $\mathcal{M}_{j,t}$ *unchanged for* S *consecutive iterations (ignoring* $\mathcal{M}_{\tilde{j},0}$ *for all* $\tilde{j}$*), last update is not SSN, and* $\nabla^2 F|_{\mathcal{M}_{\boldsymbol{w}^{j,t}}}$ *is positive definite enough* **then**
      Compute a truncated SSN step $\boldsymbol{p}$ for $F|_{\mathcal{M}_{\boldsymbol{w}^{j,t}}}$ by PCG and line search
    end
    if $\mathcal{M}_{j,t}$ *unchanged for* S *consecutive iterations and last update is SSN* **then**
      $H_{j,t} \leftarrow \gamma_{j,t} I$
    end
    while *True* **do**
      if $H_{j,t}$ *is diagonal* **then**
        Solve (12) exactly to obtain $\boldsymbol{p} = B_{j,t}\boldsymbol{t}$ and the objective Q
      end
      Solve (12) approximately using SpaRSA to get $\boldsymbol{p} = B_{j,t}\boldsymbol{t}$ and the objective Q
      if $F(\boldsymbol{w}^{j,t}+\boldsymbol{p}) \le F(\boldsymbol{w}^{j,t}) + \sigma_1 Q$ **then**
        break
      end
      $H_{j,t} \leftarrow \theta H_{j,t}$
    end
    $\boldsymbol{w}^{j,t+1} \leftarrow \boldsymbol{w}^{j,t} + \boldsymbol{p}$
    if $|Q| \le \epsilon_j$ *and* $t \ge T$ **then**
      $\boldsymbol{w}^{j+1,0} \leftarrow \boldsymbol{w}^{j,t+1}$
      break
    end
  end
end

our $\tilde{H}_t$ is

$$\tilde{H}_t = \gamma_t I - U_t Z_t^{-1} U_t^\top, \tag{7}$$

where γ_t is bounded in a positive interval for all t, and

$$U_t \coloneqq [\gamma_t S_t, Y_t], \quad Z_t \coloneqq \begin{bmatrix} \gamma_t S_t^\top S_t, & L_t \\ L_t^\top & -D_t \end{bmatrix}, \tag{8}$$
$$S_t \coloneqq \left[\boldsymbol{s}_{t-m(t)}, \boldsymbol{s}_{t-m(t)+1}, \ldots, \boldsymbol{s}_{t-1}\right],$$
$$Y_t \coloneqq \left[\boldsymbol{y}_{t-m(t)}, \boldsymbol{y}_{t-m(t)+1}, \ldots, \boldsymbol{y}_{t-1}\right],$$
$$D_t \coloneqq \operatorname{diag}\left(S_t^\top Y_t\right), (L_t)_{i,j} \coloneqq \begin{cases} \left(S_t^\top Y_t\right)_{i,j} & \text{if } i > j, \\ 0 & \text{else.} \end{cases}$$

When $t = 0$, $\tilde{H}_0 = \gamma_0 I$ and we solve (4) exactly by simply applying one proximal operation.

To construct and utilize $\tilde{H}_t$ efficiently, especially when d is large, we pick a partition $\mathcal{J}_1, \ldots, \mathcal{J}_K$ of $\{1, \ldots, d\}$ and store $(U_t)_{\mathcal{J}_k,:}$ on the kth machine; since m is usually small, all machines keep a copy of the whole Z_t matrix.

If f is not strongly convex, it is possible that (7) is not positive definite, making (4) ill-conditioned. We thus follow Li & Fukushima (2001) to take a predefined $\delta > 0$ and discard all pairs of $(\boldsymbol{s}_i, \boldsymbol{y}_i)$ that do not satisfy $\boldsymbol{s}_i^\top \boldsymbol{y}_i \geq \delta \boldsymbol{s}_i^\top \boldsymbol{s}_i$.

For solving (4), as $\tilde{H}_t$ is generally not diagonal, there is no easy closed-form solution. Thus DPLBFGS uses an iterative solver SpaRSA (Wright et al., 2009), a proximal-gradient method with fast empirical convergence, to get an approximate solution to this subproblem. By treating (4) as another regularized problem with the smooth part being quadratic, at each round SpaRSA only requires computing the gradient of the smooth part and at most a couple of proximal operations and function value evaluations. These are all relatively cheap in comparison to calculating ∇f in general, as $\tilde{H}_t$ does not couple with the original function that might involve operating on a huge amount of data. The gradient of the smooth part at a subproblem iterate $\tilde{\boldsymbol{p}}$ is computed through

$$\nabla f(\boldsymbol{w}^t) + \tilde{H}_t \tilde{\boldsymbol{p}} = \nabla f(\boldsymbol{w}^t) + \gamma_t \tilde{\boldsymbol{p}} - U_t(Z_t^{-1}(U_t^\top \tilde{\boldsymbol{p}})). \tag{9}$$

The computation of $U_t^\top \tilde{\boldsymbol{p}}$ is distributed, through

$$U_t^\top \tilde{\boldsymbol{p}} = \sum_{k=1}^{K} (U_t)_{\mathcal{J}_k,:}^\top \tilde{\boldsymbol{p}}_{\mathcal{J}_k} \tag{10}$$

and a round of $O(m)$ communication. The $\mathcal{J}_k$ part of the gradient (9) is then computed locally on the kth machine as Z_t is maintained on all machines. Thus the iterate $\tilde{\boldsymbol{p}}$ is also computed and stored in a distributed manner according to the partition $\{\mathcal{J}_k\}$, consistent with the storage of U_t.

2.3. Progressive Manifold Selection

The usual bottleneck of DPLBFGS is the computation of ∇f, which involves going through the whole dataset and communicating a vector of length $O(d)$, and the latter can be much more costly in practice. However, as the trust-region DPLBFGS approach is able to identify a manifold $\mathcal{M}^*_{\boldsymbol{w}^*} \ni \boldsymbol{w}^*$ within finite iterations, it makes sense to confine all subsequent iterates to $\mathcal{M}^*_{\boldsymbol{w}^*}$, reducing the amount of communication per round from $O(d)$ bytes to $O(|\mathcal{M}^*_{\boldsymbol{w}^*}|)$.

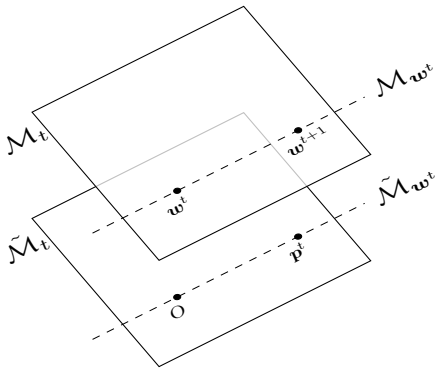


Figure 1. An example of updating within manifolds.

As the manifolds we consider in (2) are translates of subspaces of $\Re^d$, in the form

$$\mathcal{M}_{\boldsymbol{w}^t} = \boldsymbol{w}^t + \tilde{\mathcal{M}}_{\boldsymbol{w}^t} \tag{11}$$

for some subspace $\tilde{\mathcal{M}}_{\boldsymbol{w}^t}$, if we force $\boldsymbol{w}^i \in \mathcal{M}_{\boldsymbol{w}^t}$ for all $i > t$, we must have $\boldsymbol{p}^i \in \tilde{\mathcal{M}}_{\boldsymbol{w}^t}$ for all $i \geq t$. This is the central idea we use to reduce the communication and computation. At the tth iteration, with the currently manifold $\mathcal{M}_t$ selected by our algorithm that contains $\boldsymbol{w}^t$ and $\mathcal{M}_{\boldsymbol{w}^t}$ (the one making Ψ partly smooth around $\boldsymbol{w}^t$), we will select a sub-manifold $\mathcal{M}_{t+1} \subseteq \mathcal{M}_t$ and update $\boldsymbol{w}^{t+1}$ within $\mathcal{M}_{t+1}$. Equivalently, our algorithm selects a sequence of subspaces $\{\tilde{\mathcal{M}}_t\}$, and the update steps satisfy $\boldsymbol{p}^t \in \tilde{\mathcal{M}}_t$. Therefore, the subproblem (4) becomes $\min_{\boldsymbol{p} \in \tilde{\mathcal{M}}_t} Q_{H_t}(\boldsymbol{p}; \boldsymbol{w}^t)$. One thing to note is that $\mathcal{M}_t$ might not be the one that makes Ψ $\mathcal{C}^2$ at $\boldsymbol{w}^t$, which we denote by $\mathcal{M}_{\boldsymbol{w}^t}$, but we always have $\mathcal{M}_{\boldsymbol{w}^t} \subseteq \mathcal{M}_t$. We use a super-manifold of $\mathcal{M}_{\boldsymbol{w}^t}$ to allow flexibility because it is likely $\mathcal{M}_{\boldsymbol{w}^{t+1}} \nsubseteq \mathcal{M}_{\boldsymbol{w}^t}$, unless $\boldsymbol{w}^t$ is close to $\boldsymbol{w}^*$. An illustration is shown in Figure 1.

To really reduce the communication cost through the idea above, we need to conduct a change of basis first. As each $\tilde{\mathcal{M}}_t$ is a subspace of $\Re^d$, we can find an orthonormal basis $\{\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,N(t)}\}$ of it. We denote $B_t \coloneqq [\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,N(t)}]$ and approximately minimize the following equivalent problem

$$\begin{aligned} \min_{\boldsymbol{t} \in \Re^{N(t)}} \; & \nabla f(\boldsymbol{w}^t)^\top B_t \boldsymbol{t} + \frac{1}{2}(B_t \boldsymbol{t})^\top H_t B_t \boldsymbol{t} \\ & + \Psi(\boldsymbol{w}^t + B_t \boldsymbol{t}) - \Psi(\boldsymbol{w}^t), \end{aligned} \tag{12}$$

and only transmit the solution $\tilde{\boldsymbol{t}}$ to all machines for constructing $\boldsymbol{p} = B_t \tilde{\boldsymbol{t}}$. This reduces the communication cost for updating $\boldsymbol{w}^t$ from $O(d)$ to $O(N(t))$. When $\mathcal{M}_t = \mathcal{M}^*$,

we get $N(t) = O(|\mathcal{M}^*|)$ and the cost of this communication is greatly reduced. For the $\Psi(\boldsymbol{w}^t + B_t\boldsymbol{t})$ term in (12), the proximal operation seems harder to calculate, but since the basis is selected according to the structure of Ψ, this is not a problem. For the manifold selection, we use $\partial F|_{\mathcal{M}_t}(\boldsymbol{w}^{t+1})$ to decide how to shrink the current $\mathcal{M}_t$; see Appendix A.1 for more details.

Next, we reduce the communication cost for calculating the gradient. Let $P_{\tilde{\mathcal{M}}_t}$ be the Euclidean projection onto $\tilde{\mathcal{M}}_t$ and $\tilde{\mathcal{M}}_t^C$ be the complement space of $\tilde{\mathcal{M}}_t$. Because $\boldsymbol{v} = P_{\tilde{\mathcal{M}}_t}(\boldsymbol{v}) + P_{\tilde{\mathcal{M}}_t^C}(\boldsymbol{v})$ for any $\boldsymbol{v} \in \Re^d$, we always have

$$\nabla f(\boldsymbol{w}^t)^\top \boldsymbol{p} = P_{\tilde{\mathcal{M}}_t}(\nabla f(\boldsymbol{w}^t))^\top \boldsymbol{p}, \quad \forall \boldsymbol{p} \in \tilde{\mathcal{M}}_t.$$

Thus we only need $P_{\tilde{\mathcal{M}}_t}(\nabla f(\boldsymbol{w}^t))$ in (12) as $B_t\boldsymbol{t} \in \tilde{\mathcal{M}}_t$. On the other hand, we use $P_{\tilde{\mathcal{M}}_{t-1}}(\nabla f(\boldsymbol{w}^t))$ to select $\mathcal{M}_t$, and since $\tilde{\mathcal{M}}_t \subseteq \tilde{\mathcal{M}}_{t-1}$, we need to compute

$$P_{\tilde{\mathcal{M}}_{t-1}}(\nabla f(\boldsymbol{w}^t)) = B_{t-1}\sum_{i=1}^{K} B_{t-1}^\top \nabla f_k(\boldsymbol{w}^t). \tag{13}$$

The summation in (13) is where the communication occurs, and the bytes transmitted is now lowered to $O(N(t-1))$. Therefore, the communication cost per iteration has been reduced to $O(N(t-1) + N(t)) = O(N(t-1))$ as $N(t)$ is decreasing with respect to t. If B_j consists of columns of B_i for all $i \le j$, and the coordinates of the gradient are aligned with the columns in B_0 (as is the case when Ψ is a sparsity-inducing term like the ℓ_1 or $\ell_{2,1}$ norms), then $B_{t-1}^\top \nabla f_k(\boldsymbol{w}^t)$ is simply taking the coordinates of $\nabla f_k(\boldsymbol{w}^t)$ selected in B_{t-1}, so the computation is also cheaper than obtaining the whole $\nabla f(\boldsymbol{w}^t)$.

The remaining issue is to construct $\tilde{H}_t$ when we only have $P_{\tilde{\mathcal{M}}_{t-1}}(\nabla f(\boldsymbol{w}^t))$ and therefore $P_{\tilde{\mathcal{M}}_{t-1}}(\boldsymbol{y}_{t-1})$. Since $\tilde{H}_t$ appears in (12) in the form

$$B_t^\top \tilde{H}_t B_t = \gamma_t B_t^\top B_t - (B_t^\top U_t) Z_t^{-1} (B_t^\top U_t)^\top,$$

as long as we can construct Z_t correctly, $P_{\tilde{\mathcal{M}}_{t-1}}(\boldsymbol{y}_{t-1})$ is enough for obtaining $B_t^\top U_t$ and thus $B_t^\top \tilde{H}_t B_t$ as $\boldsymbol{s}_{t-1} = \boldsymbol{p}^{t-1} \in \tilde{\mathcal{M}}_{t-1}$. Note that because $\texttt{span}(B_{t+1}) = \tilde{\mathcal{M}}_{t+1} \subseteq \tilde{\mathcal{M}}_t = \texttt{span}(B_t)$, keeping $B_t^\top U_t$ alone is enough for updating $B_{t+1}^\top U_t$. To compute the correct Z_t with only $B_t^\top U_t$ available, the key is to reuse entries already appeared in Z_{t-1}. We thus only compute the newly appeared entries, which are $S_t^\top \boldsymbol{s}_{t-1}$ and $Y_t^\top \boldsymbol{s}_{t-1}$. As now we only compute $P_{\tilde{\mathcal{M}}_{t-1}}(\nabla f(\boldsymbol{w}^t))$, $P_{\tilde{\mathcal{M}}_{t-1}}(\boldsymbol{y}_{t-1})$ is used instead to update Z_t, and we call the matrix obtained this way $\tilde{Z}_t$. Section 3 will show that $\tilde{Z}_t = Z_t$, so our algorithm correctly maintains the LBFGS approximation and the corresponding fast convergence.

Our safeguard mechanism is also calculated using the components within the subspace only, as follows.

$$P_{\tilde{\mathcal{M}}_i}(\boldsymbol{s}_i)^\top P_{\tilde{\mathcal{M}}_i}(\boldsymbol{y}_i) \ge \delta P_{\tilde{\mathcal{M}}_i}(\boldsymbol{s}_i)^\top P_{\tilde{\mathcal{M}}_i}(\boldsymbol{s}_i). \tag{14}$$

For the choice of γ_t, we take a small $\tilde{\epsilon} > 0$ and set the following in our implementation.

$$\gamma_0 = \max\left\{\left|\frac{\nabla f(\boldsymbol{w}^0)^\top \nabla^2 f(\boldsymbol{w}^0) \nabla f(\boldsymbol{w}^0)}{\|\nabla f(\boldsymbol{w}^0)\|_2^2}\right|, \tilde{\epsilon}\right\},$$
$$\gamma_{t+1} = \begin{cases} \frac{P_{\tilde{\mathcal{M}}_t}(\boldsymbol{y}_t)^\top P_{\tilde{\mathcal{M}}_t}(\boldsymbol{y}_t)}{P_{\tilde{\mathcal{M}}_t}(\boldsymbol{y}_t)^\top P_{\tilde{\mathcal{M}}_t}(\boldsymbol{s}_t)} & \text{if (14)}, \\ \gamma_t & \text{otherwise}, \end{cases} \quad \forall t \ge 0. \tag{15}$$

2.4. Acceleration After Identifying the Active Set

The regularization Ψ we consider is partly smooth within $\mathcal{M}^*$ around a solution $\boldsymbol{w}^*$. Therefore, when $\mathcal{M}^*$ is correctly identified, the problem can be reduced to a smooth optimization problem by restricting the subsequent iterates within $\mathcal{M}^*$. We can therefore utilize smooth optimization algorithms that are both more efficient and faster in convergence. In particular, we use a truncated semismooth Newton (SSN) method that is superlinear-convergent when we are close to a solution (Qi & Sun, 1993), which is the case when $\mathcal{M}^*$ can be correctly identified. The idea of SSN is that since ∇f is Lipschitz continuous, f is twice-differentiable almost everywhere. Therefore, its generalized Hessian (denoted by $\nabla^2 f$) always exists (Hiriart-Urruty et al., 1984) and is hence used as the substitute of the real Hessian for computing the Newton step. When f is twice-differentiable, the algorithm reduces to the normal Newton method. We use preconditioned conjugate gradient (PCG) to approximately solve the SSN linear system and follow Hsia et al. (2018) to use the diagonal entries of $\nabla^2 f$ as the preconditioner. Backtracking line search is then applied to ensure sufficient objective decrease. To determine that we have found $\mathcal{M}^*$ to safely conduct SSN within the manifold, we assert $\mathcal{M}^* = \mathcal{M}_t$ when $\mathcal{M}_t$ remains unchanged for S consecutive iterations, for some predefined S.

To ensure convergence in case that $\nabla^2 f$ is not positive-definite, every time after an SSN step, we conduct a DPLBFGS step or a proximal gradient (PG) step (when we have entered the superlinear-convergent phase) on F restricted to $\mathcal{M}_t$. We also conduct manifold selection again after the DPLBFGS or PG step, and continue smooth optimization only when the manifold remains the same. More detailed description of our implementation is in Appendix A.2.

2.5. Safeguards

Our progressive manifold selection might sometimes be too aggressive and thus miss the correct manifold. To avoid this problem, we adopt a two-layer loop strategy. For any variable that we previously use t as its counter, now we use (j,t) to indicate that it is at the jth outer iteration and the tth inner iteration. At the beginning of the jth outer iteration, we discard $U_{j-1,t}$, $\tilde{Z}_{j-1,t}$ and the selected manifold to start afresh from the current iterate. The inner loop then conducts

the algorithm described above, until $|Q_{\hat{H}_{j,t}}(\boldsymbol{p}^{j,t};\boldsymbol{w}^{j,t})|$ is smaller than a predefined $\epsilon_j > 0$ and $t \geq T$ for some prespecified integer $T \geq 0$. We use an annealing strategy such that $\{\epsilon_j\}$ is a sequence decreasing to some $\epsilon > 0$. The overall algorithm is summarized in Algorithm 1.

3. Analysis

We first analyze the convergence of the proposed algorithm, and then show that $\mathcal{M}^*$ is identified in finite iterations. The latter justifies our approach of conducting progressive manifold identification on the fly and taking SSN steps after the manifold is fixed.

We remark that using the domain of $\Re^d$ in (1) is just for notational ease, and our algorithm and analysis extends directly to any Euclidean spaces.

3.1. Convergence

The following lemma shows that progressive manifold selection works correctly with the quasi-Newton approach.

Lemma 3.1. *For any $j = 0, 1, \ldots,$ under the progressive manifold identification scheme $\mathcal{M}_{j,t+1} \subseteq \mathcal{M}_{j,t}$ for all $t \geq 0$, $\tilde{Z}_{j,t} = Z_{j,t}$ always holds true. Moreover, $B_{j,t+1}^\top U_{j,t} = B_{j,t+1}^\top B_{j,t}(B_{j,t}^\top U_{j,t})$ for all i, t, so maintaining $B_{j,t}^\top U_{j,t}$ suffices for constructing (12).*

We now present the convergence rate of our algorithm, which provides an upper bound for the communication rounds. Parts of our analysis are extended from (Lee et al., 2019). We will use the following notation in this section.

$$Q_{H_{j,t}}^{j,t} := \min_{\boldsymbol{p}\in\tilde{\mathcal{M}}_{j,t}} \left(Q_{H_{j,t}}(\boldsymbol{p};\boldsymbol{w}^{j,t})\right).$$

Throughout the analysis, we assume that there is a fixed $\eta \in [0, 1)$ such that every time we solve (12) for $t > 0$, the solution $\boldsymbol{p}$ is η-approximate:

$$Q_{H_{j,t}}(\boldsymbol{p};\boldsymbol{w}^{j,t}) - Q_{H_{j,t}}^{j,t} \leq \eta(Q_{H_{j,t}}(\boldsymbol{0};\boldsymbol{w}^{j,t}) - Q_{H_{j,t}}^{j,t}), \tag{16}$$

and when $t = 0$, the solution is exact (as $H_{j,0} = \gamma_{j,0}I$). Condition (16) can easily be satisfied without the knowledge of $Q_{H_{j,t}}^{j,t}$, provided the eigenvalues of the quadratic term are bounded in a positive range and the subproblem solver is linear-convergent on strongly convex problems. See more discussions in (Lee & Wright, 2019; Lee et al., 2019). We thus assume (16) holds for all $t > 0$ with some fixed $\eta \in [0, 1)$, although its explicit value might be unknown. We first show that $\tilde{H}_{j,t}$ and $\hat{H}_{j,t}$ have bounded eigenvalues.

Lemma 3.2. *There are constants $\tilde{C}_1 \geq \tilde{C}_2 > 0$ such that $\tilde{C}_1 I \succeq \tilde{H}_{j,t} \succeq \tilde{C}_2 I$ for all j, t, and at every (j, t)th iteration, (6) is conducted at most a constant times before (5) is satisfied. Therefore, There are constants $C_1 \geq C_2 > 0$ such that $C_1 I \succeq \hat{H}_{j,t} \succeq \tilde{H}_{j,t} \succeq C_2 I$.*

We now show that the inner loops terminate finitely and give the convergence rate of the outer loop.

Lemma 3.3. *The jth inner loop ends in $o(1/\epsilon_j)$ steps. When f is convex and the quadratic growth condition*

$$g(\boldsymbol{w}) - \min_{\hat{\boldsymbol{w}}} g(\hat{\boldsymbol{w}}) \geq \min_{\boldsymbol{w}^*\in \mathit{argmin}\, g(\boldsymbol{w})} \mu\|\boldsymbol{w} - \boldsymbol{w}^*\|_2^2, \quad \forall \boldsymbol{w} \in \mathit{dom}(g), \tag{17}$$

for some fixed $\mu > 0$ is satisfied by $F|_{\mathcal{M}_{j,t}}$ for all j, t within the level set $\{\boldsymbol{w} \mid F|_{\mathcal{M}_{j,t}}(\boldsymbol{w}) \leq F|_{\mathcal{M}_{j,t}}(\boldsymbol{w}^{j,t})\}$, the inner loop ends in $O(\log(1/\epsilon_j))$ steps.

Theorem 3.4. *Consider (1), and fix $\epsilon > 0$. When f is convex, for reaching a point $\boldsymbol{w}$ with $F(\boldsymbol{w}) - F^* \leq \epsilon$, it takes $O(1/\epsilon)$ outer iterations if*

$$\max_{\boldsymbol{w}:F(\boldsymbol{w})\leq F(\boldsymbol{w}^{0,0})} \min_{\boldsymbol{w}^*\in\Omega} \|\boldsymbol{w} - \boldsymbol{w}^*\|_2 \tag{18}$$

is bounded, and $O(\log(1/\epsilon))$ if F satisfies (17). When f is nonconvex, it takes $o(1/\epsilon)$ outer iterations to have $|Q_{\hat{H}_{j,0}}(\boldsymbol{p}^{j,0};\boldsymbol{w}^{j,0})| \leq \epsilon$, and the first-order optimality $\boldsymbol{0} \in \partial F(\boldsymbol{w}^{j,0})$ holds if and only if $Q_{\hat{H}_{j,0}}(\boldsymbol{p}^{j,0};\boldsymbol{w}^{j,0}) = 0$.

3.2. Finite-time Manifold Identification

In what follows, we establish that the proposed algorithm can identify the correct manifold within finite time under the partly smoothness assumption (Lewis, 2002).

Definition 3.5 (Partly smooth functions)**.** *A convex function Ψ is partly smooth at $\boldsymbol{w}$ relative to a set $\mathcal{M}$ containing $\boldsymbol{w}$ if $\partial\Psi(\boldsymbol{w}) \neq \emptyset$ and:*

1. *Around $\boldsymbol{w}$, $\mathcal{M}$ is a $\mathcal{C}^2$-manifold and $\Psi|_{\mathcal{M}}$ is $\mathcal{C}^2$.*
2. *The affine span of $\partial\Psi(\boldsymbol{w})$ is a translate of the normal space to $\mathcal{M}$ at $\boldsymbol{w}$.*
3. *$\partial\Psi$ is continuous at $\boldsymbol{w}$ relative to $\mathcal{M}$.*

We first show that cluster points of the iterates are stationary.

Theorem 3.6. *All cluster points of $\{\boldsymbol{w}^{j,0}\}_{j=0}^\infty$ are critical. Moreover, if f is lower-bounded and Ψ is coercive, $\{\boldsymbol{w}^{j,0}\}_{j=0}^\infty$ has at least one cluster point.*

We now present the main result that $\mathcal{M}^*$ can be identified within finite iterations.

Theorem 3.7. *If a cluster point $\boldsymbol{w}^*$ of $\{\boldsymbol{w}^{j,0}\}$ satisfies*

$$-\nabla f(\boldsymbol{w}^*) \in \text{relint}\, \partial\Psi(\boldsymbol{w}^*), \tag{19}$$

where relint *denotes the relative interior, and Ψ is partly smooth relative to $\mathcal{M}^*$ around $\boldsymbol{w}^*$, then $\boldsymbol{w}^{j,1}$ identifies $\mathcal{M}^*$ in finite outer iterations. If moreover (12) is always solved exactly, then for j large enough and in the convergent subsequence, $\boldsymbol{w}^{j,t} \in \mathcal{M}^*$ for all $t > 0$.*

Remark 3.8. *Theorems 3.4 and 3.7 are applicable to the case in which F is nonconvex as long as Ψ is convex. Therefore, MADPQN is also applicable to nonconvex problems with a differentiable loss, including deep learning models.*

When the nondegenerate condition (19) fails, we are unable to identify $\mathcal{M}^*$. However, if Ψ is a mirror-stratifiable function (Fadili et al., 2018), we can still identify a superset manifold $\hat{\mathcal{M}}^*$ of $\mathcal{M}^*$, so MADPQN is still able to reduce the per-round communication cost to $O(|\hat{\mathcal{M}}^*|)$ at the first stage. However, it will not enter the second stage, so the overall communication cost is expected to be slightly higher, though still superior to existing approaches. See Appendix B.3 for the empirical effectiveness of the second stage.

3.3. Superlinear Convergence of the Second Stage

At the smooth optimization stage of MADPQN, eventually we alternate between a PG step and a truncated SSN step. If ∇f is semismooth (see Definition C.2), then we can obtain a two-step superlinear convergence of the iterates to the optimal solution when the truncated SSN steps are accurate enough. Due to the space limit, we leave the details of computing the truncated SSN steps and the superlinear convergence in Appendices A.2 and C.2.

4. Related Work

Manifold Identification. The major tool we use is manifold identification for partly smooth functions studied by Hare & Lewis (2004); Lewis (2002); Lewis & Zhang (2013). By applying their theory, many first-order algorithms are known to have the ability of manifold identification for partly smooth functions, including proximal-gradient-type methods (Nutini et al., 2017b; Liang et al., 2017a), the alternating direction method for multipliers (Liang et al., 2017b), regularized dual averaging (Lee & Wright, 2012), proximal coordinate descent (Wright, 2012; Nutini et al., 2017a), and variance reduction stochastic methods (Poon et al., 2018). However, these algorithms are usually not the choice for distributed optimization as non-stochastic first-order methods converge slowly in practice, and the frequent update of the iterate in stochastic methods incurs much more rounds of communication. Second-order algorithms tend to be superior for distributed optimization (see, *e.g.*, (Zhang & Xiao, 2015; Lee et al., 2019)), but current theory for their ability of manifold identification is restricted to the case in which (4) is solved exactly (Hare & Lewis, 2007; Hare, 2011; Sun et al., 2019). This work is the first to study and utilize such property in distributed optimization to improve communication efficiency. A key difference from all the works above is that instead of passively waiting until the final manifold is identified, our approach actively tries to find the manifold, and improves the computation and communication efficiency by utilizing from the beginning on the fact that the final solution lies in a lower-dimensional manifold.

Active Set Approaches. Another related idea is working-set or active-set methods for constrained problems in single-core optimization. These approaches force inequality constraints selected in the active-set to be at equality to save computation. Active-set methods can be extended to (1) when Ψ has an equivalent constrained form. For example, Yuan et al. (2012) used working set selection for ℓ_1-regularized logistic regression for proximal-Newton with coordinate descent as the subproblem solver, and Zhong et al. (2014) took a similar idea for ℓ_1-regularized M-estimators. These works showed that working-set heuristics can improve the empirical running time, but did not provide theoretical supports. Johnson & Guestrin (2015) considered a primal-dual approach for working set selection for ℓ_1-regularized problems with theoretical guarantees and nice practical performance, but it is hard to generalize the results to other regularizers. Our usage of progressive manifold identification is essentially a type of working set selection, but unlike existing works, MADPQN is fully supported by the theory of manifold identification and is applicable to all partly smooth regularizers, including but not limited to the ℓ_1 and group-LASSO norms. The stage of smooth optimization method is not present in the methods with working set heuristics either. Moreover, the purposes are totally different—these works use active sets to reduce their computation cost, while our progressive manifold identification is mainly for improving the communication efficiency.

Distributed Optimization for (1). For optimizing (1) in a distributed manner, although there are some heuristic approaches designed for specific regularizers that work well in practice, such as L-COMM (Chiang et al., 2018) and OWLQN (Andrew & Gao, 2007) for ℓ_1-regularized problems, the only method we are aware of that applies to general regularizers is the one proposed by Lee et al. (2019). They consider an inexact distributed proximal quasi-Newton method for (1) with theoretical convergence supports, and the first stage of MADPQN is extended from it. However, they did not utilize the partly smoothness of Ψ, so the communication cost per-round of their method is fixed to $O(d)$, while the communication cost per round of MADPQN is $O(N(t-1))$, which converges to $O(|\mathcal{M}^*|)$ rapidly. When $|\mathcal{M}^*| \ll d$, our method has a much lower communication cost per round (see Figure 3). Another difference between MADPQN and (Lee et al., 2019) is the truncated SSN steps, which greatly helps the convergence and reduces the number of communication rounds.

Distributed Optimization for Linear Models with Feature-wise Storage. For linear models such that $f_k(\boldsymbol{w}) = g_k(X_k^\top \boldsymbol{w})$ for some function g_k and some matrix X_k, instead of assuming each machine has exclusive access to an f_k, some works such as (Mahajan et al., 2017; Dünner et al., 2018) consider the scheme in which each node stores some rows of $X := [X_1, \ldots, X_K] \in \Re^{d\times n}$. These approaches communicate a vector of length $O(n)$ to synchronize $X^\top \boldsymbol{w}$ instead, so they might have a lower com-

munication cost when $n \ll d$. However, they do not work for nonlinear models. Even for linear models, $|\mathcal{M}^*| < n$ is often observed and these methods are unable to utilize the low-dimensional manifolds to reduce the communication cost. On the other hand, when $d \ll n$, their communication cost becomes prohibitively high, while MADPQN can handle both large d and large n.

5. Experiments

We conduct experiments to examine the empirical behavior of MADPQN on the Amazon cloud. For all experiments, we use 32 EC2 instances, each with one thread, connected through Open MPI. Convergence is compared through the relative difference to the optimal function value $(F(\boldsymbol{w}) - F^*)/F^*$, where F^* is the optimal objective value of (1), obtained approximately by solving the problem to a high precision.

We use the following fixed parameters for MADPQN throughout the experiments, and it works quite robustly, so there is no need to tune these parameters: $\theta = 2$, $m = 10$, $T = 2$, $S = 10$, $\sigma_1 = 10^{-4}$, $\tilde{\epsilon} = 10^{-10}$, $\delta = 10^{-10}$, $\epsilon = 10^{-14}$, $\epsilon_j = \max\{\epsilon, 10^{-4-3j}\}$.

Table 1. Data statistics for (20).

Data set	#instances (n)	#features (d)	$\lvert\mathcal{M}^*\rvert$
news20	19,996	1,355,191	506
webspam	350,000	16,609,143	793
avazu-site	25,832,830	999,962	11,867

Table 2. Data statistics for (21). Note that $d = \hat{d} \times c$.

Data set	n	$\hat{d}$	c	$\lvert\mathcal{M}^*\rvert$
sector.scale	6,412	55,197	105	94,920
rcv1-test.multiclass	518,571	47,236	53	386,529

5.1. Experiment on ℓ_1-regularized Problems

We first examine the performance of MADPQN on ℓ_1-regularized logistic regression, whose objective in (1) is

$$F(\boldsymbol{w}) = \|\boldsymbol{w}\|_1 + C\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i\right)\right), \quad (20)$$

where each $(\boldsymbol{x}_i, y_i) \in \Re^d \times \{-1, 1\}$ is an instance-label pair stored in one of the machines (so the sum over the data points on the kth machine forms f_k), and $C > 0$ is a given parameter to balance the two terms. For this problem, each A_i in (2) is the standard unit vector of the ith coordinate (so are the columns of $B_{j,t}$), $I_{\boldsymbol{w}} = \{i \mid \boldsymbol{w}_i = 0\}$, and $|\mathcal{M}^*|$ is the number of nonzero elements in the solution $\boldsymbol{w}^*$.

We consider public datasets listed in Table 1 (from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets) and set $C = 1$ in all experiments. For news20, we use 32 t2.micro instances, while for the remaining two larger datasets, we use 32 r5n.large instances. We first examine the effectiveness of our progressive manifold selection. Table 1 shows that $|\mathcal{M}^*|$ can be much smaller than d, proving the effectiveness of utilizing manifold identification to reduce the communication cost per round. We further compare the time spent on communication of our method with and without progressive manifold selection in Figure 3, and see that the progressively selected manifold quickly approaches $\mathcal{M}^*$ and effectively keeps the cumulative communication cost low as the manifold dimension drops.

We then compare MADPQN with the following state-of-the-art methods for (20) implemented in C/C++.

- DPLBFGS (Lee et al., 2019)
- OWLQN (Andrew & Gao, 2007)
- L-COMM (Chiang et al., 2018)

For all of them, we use information from the past 10 iterations and set the step size shrinking parameter to 0.5 to have a fair comparison with MADPQN (equivalent to $m = 10$ and $\theta = 2$). Note that distributed first-order methods are not included in our comparison because they have been shown by Lee et al. (2019) (for proximal-gradient-type methods) and Dünner et al. (2018) (for stochastic methods) to be inferior to the methods we compare with. Comparison with the feature-wise approach by Dünner et al. (2018) and other additional experiments are in Appendix B.

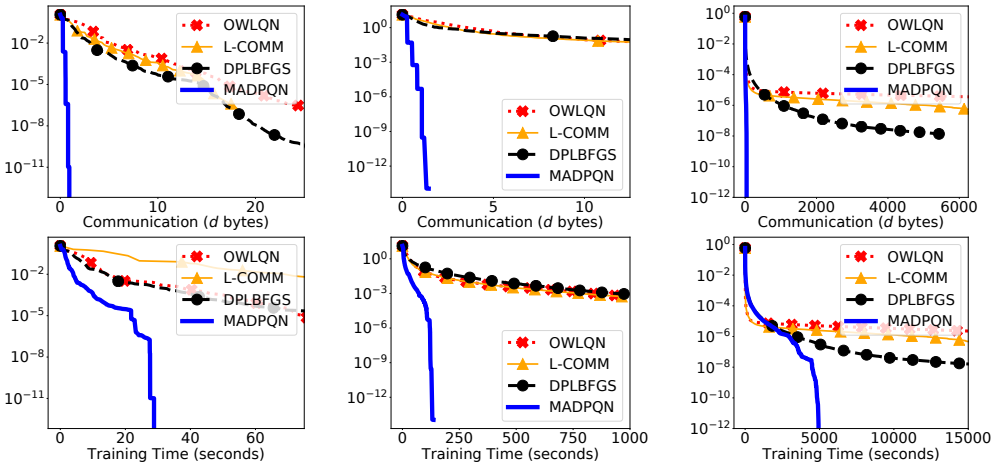
We compare the running time and the bytes communicated (divided by d) in Figure 2. We observe that the communication cost and the running time of MADPQN are both better than all other methods, although the difference in the running time is less significant. This is because that local computation still occupies a prominent portion of the total running time. Note that the manifold selection procedure in MADPQN takes less than 1% of the total running time so it is not the cause.

5.2. Experiment on Group-LASSO-regularized Problems

We next consider the group-LASSO-regularized multinomial logistic regression problem for multiclass classification:

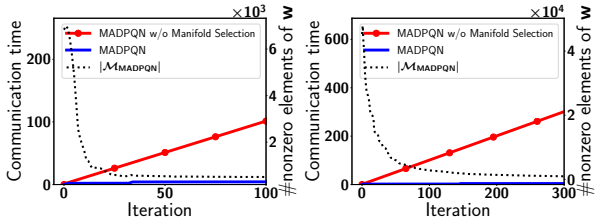
$$F(W) = \sum_{i=1}^{\hat{d}} \sqrt{\sum_{j=1}^{c} W_{i,j}^2} + C\sum_{i=1}^{n} l(W^\top \boldsymbol{x}_i; y_i), W \in \Re^{\hat{d} \times c}, \quad (21)$$

where $c > 0$ is the number of possible classes in the problem, each $(\boldsymbol{x}_i, y_i) \in \Re^{\hat{d}} \times \{1, \ldots, c\}$ is an instance-label



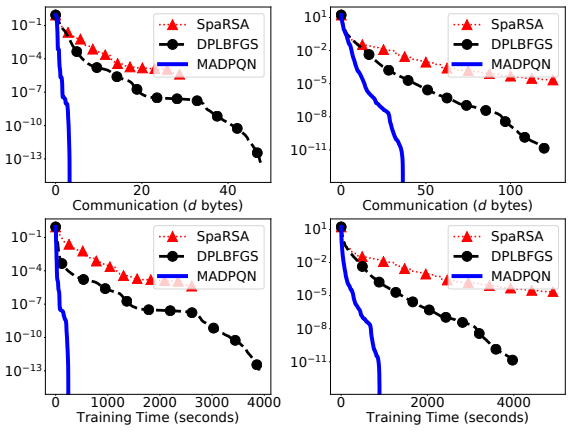
(a) news20 (b) webspam (c) avazu-site

Figure 2. Comparison of methods for (20). We present the number of bytes communicated divided by d (upper) and the training time (lower) vs. the relative difference to the optimal value.



(a) news20 (b) webspam

Figure 3. Iteration (x-axis) vs. total time spent on communication (solid lines, y-axis on the left) and the number of nonzero elements in the current $\boldsymbol{w}$ of MADPQN (dotted line, y-axis on the right).



(a) sector.scale (b) rcv1-test.multiclass

Figure 4. Comparison of methods for (21). We present the bytes communicated (upper) and the training time (lower) vs. the relative difference to the optimal value.

pair stored in one of the machines, and the loss term is defined as $l(\boldsymbol{z}; y) := -\log(\exp(z_y)/\sum_{i=1}^{c} \exp(z_i))$. For this problem, each A_i in (2) is the same as that for (20), $I_W = \{i \mid W_{i,j} = 0, \forall j\}$, and $|\mathcal{M}^*|$ is c times the number of rows in the solution W^* that contain at least one nonzero entry. Note that the problem dimension in this problem is $d = c \times \hat{d}$. For this problem, we compare MADPQN with DPLBFGS and SpaRSA (Wright et al., 2009) on the public datasets in Table 2, as OWLQN and L-COMM only apply to ℓ_1-regularized problems. The result of using 32 t2.micro instances is shown in Figure 4, and we see that MADPQN is significantly more efficient than existing approaches in both the communication cost and the running time.

6. Conclusions

In this work, we utilize manifold identification to greatly improve the communication efficiency and hence the overall running time of distributed optimization. Our algorithm cuts the bytes communicated per round through progressively selecting a sub-manifold where an optimal solution is likely to lie. After the correct manifold is identified, our algorithm further exploits partly smoothness of the problem to accelerate the convergence, and reduce the communication complexity by switching to a superlinear-convergent truncated semismooth Newton method. Experiments show that the overall communication cost and running time of our method are orders of magnitude lower than the state of the art. Future work includes the extension to nonlinear manifolds using Riemannian optimization, applying our algorithm to deep learning training with sparsity-inducing norms (Wen et al., 2016), and the analysis of manifold identification when the subproblems are solved inexactly.

Acknowledgement

This work was supported in part by the AWS Cloud Credits for Research program of Amazon Inc.

References

Aji, A. F. and Heafield, K. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 440–445, 2017.

Andrew, G. and Gao, J. Scalable training of L1-regularized log-linear models. In *Proceedings of the International Conference on Machine Learning*, 2007.

Byrd, R. H., Nocedal, J., and Schnabel, R. B. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3): 129–156, 1994.

Chen, T., Giannakis, G., Sun, T., and Yin, W. LAG: Lazily aggregated gradient for communication-efficient distributed learning. In *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2018.

Chiang, W.-L., Li, Y.-S., Lee, C.-p., and Lin, C.-J. Limited-memory common-directions method for distributed l1-regularized linear classification. In *Proceedings of SIAM International Conference on Data Mining*, 2018.

Dünner, C., Lucchi, A., Gargiani, M., Bian, A., Hofmann, T., and Jaggi, M. A distributed second-order algorithm you can trust. In *Proceedings of the International Conference on Machine Learning*, 2018.

Fadili, J., Malick, J., and Peyré, G. Sensitivity analysis for mirror-stratifiable convex functions. *SIAM Journal on Optimization*, 28(4):2975–3000, 2018.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008.

Hare, W. Identifying active manifolds in regularization problems. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 261–271. Springer, 2011.

Hare, W. L. and Lewis, A. S. Identifying active constraints via partial smoothness and prox-regularity. *Journal of Convex Analysis*, 11(2):251–266, 2004.

Hare, W. L. and Lewis, A. S. Identifying active manifolds. *Algorithmic Operations Research*, 2(2):75–75, 2007.

Hiriart-Urruty, J.-B., Strodiot, J.-J., and Nguyen, V. H. Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data. *Applied Mathematics & Optimization*, 11(1):43–56, 1984.

Hsia, C.-Y., Chiang, W.-L., and Lin, C.-J. Preconditioned conjugate gradient methods in truncated Newton frameworks for large-scale linear classification. In *Proceedings of the Asian Conference on Machine Learning*, 2018.

Johnson, T. and Guestrin, C. Blitz: A principled meta-algorithm for scaling sparse optimization. In *Proceedings of the International Conference on Machine Learning*, pp. 1171–1179, 2015.

Lee, C.-p. and Chang, K.-W. Distributed block-diagonal approximation methods for regularized empirical risk minimization. *Machine Learning*, 2019.

Lee, C.-p. and Wright, S. J. Inexact successive quadratic approximation for regularized optimization. *Computational Optimization and Applications*, 72:641–674, 2019.

Lee, C.-p., Wang, P.-W., Chen, W., and Lin, C.-J. Limited-memory common-directions method for distributed optimization and its application on empirical risk minimization. In *Proceedings of the SIAM International Conference on Data Mining*, 2017.

Lee, C.-p., Lim, C. H., and Wright, S. J. A distributed quasi-Newton algorithm for primal and dual regularized empirical risk minimization. Technical report, 2019. arXiv:1912.06508.

Lee, S. and Wright, S. J. Manifold identification in dual averaging for regularized stochastic online learning. *Journal of Machine Learning Research*, 13:1705–1744, 2012.

Lewis, A. S. Active sets, nonsmoothness, and sensitivity. *SIAM Journal on Optimization*, 13(3):702–725, 2002.

Lewis, A. S. and Zhang, S. Partial smoothness, tilt stability, and generalized hessians. *SIAM Journal on Optimization*, 23(1):74–94, 2013.

Li, D.-H. and Fukushima, M. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11 (4):1054–1064, 2001.

Liang, J., Fadili, J., and Peyré, G. Activity identification and local linear convergence of forward–backward-type methods. *SIAM Journal on Optimization*, 27(1):408–437, 2017a.

Liang, J., Fadili, J., and Peyré, G. Local convergence properties of douglas–rachford and alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 172(3):874–913, 2017b.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *NIPS Workshop on ML Systems*, 2017.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Mahajan, D., Keerthi, S. S., and Sundararajan, S. A distributed block coordinate descent method for training ℓ_1 regularized linear classifiers. *Journal of Machine Learning Research*, 18:1–35, 2017.

Meier, L., Van De Geer, S., and Bühlmann, P. The group LASSO for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70 (1):53–71, 2008.

Mifflin, R. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

Nocedal, J. and Wright, S. *Numerical Optimization*. Springer, second edition, 2006.

Nutini, J., Laradji, I., and Schmidt, M. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. Technical report, 2017a. arXiv:1712.08859.

Nutini, J., Schmidt, M., and Hare, W. "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? In *NIPS Workshop on Optimization for Machine Learning*, 2017b.

Poon, C., Liang, J., and Schönlieb, C.-B. Local convergence properties of SAGA/prox-SVRG and acceleration. In *Proceedings of the International Conference on Machine Learning*, 2018.

Qi, L. and Sun, J. A nonsmooth version of Newton's method. *Mathematical programming*, 58(1-3):353–367, 1993.

Rockafellar, R. T. and Wets, R. J.-B. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

Shi, W., Ling, Q., Wu, G., and Yin, W. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Sun, Y., Jeong, H., Nutini, J., and Schmidt, M. Are we there yet? manifold identification of gradient-related proximal methods. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1110–1119, 2019.

Tibshirani, R. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society Series B*, 58:267–288, 1996.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pp. 1509–1519, 2017.

Wright, S. J. Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization*, 22(1):159–186, 2012.

Wright, S. J., Nowak, R. D., and Figueiredo, M. A. T. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

Yuan, G.-X., Ho, C.-H., and Lin, C.-J. An improved GLMNET for L1-regularized logistic regression. *Journal of Machine Learning Research*, 13:1999–2030, 2012.

Zhang, Y. and Xiao, L. DiSCO: Distributed optimization for self-concordant empirical loss. In *International Conference on Machine Learning*, pp. 362–370, 2015.

Zheng, S., Wang, J., Xia, F., Xu, W., and Zhang, T. A general distributed dual coordinate optimization framework for regularized loss minimization. *Journal of Machine Learning Research*, 18:1–52, 2017.

Zhong, K., Yen, I. E.-H., Dhillon, I. S., and Ravikumar, P. K. Proximal quasi-newton for computationally intensive $l1$-regularized M-estimators. In *Advances in Neural Information Processing Systems*, 2014.