Randomized Alternating Direction Methods for Efficient Distributed Optimization

Emmanuel Candès [1] **Qijia Jiang** [1] **Mert Pilanci** [1]

Abstract

We study large scale distributed optimization problems where the data are stored across different machines without shared memory. We propose a novel randomized variant of the Alternating Direction Method of Multipliers (ADMM) algorithm, where random projections are applied as a one-time pre-processing step for distributing the data, followed by iterative updates with little communication overhead. For distributed cone program solvers that crucially rely on solving a large number of linear systems as a subroutine, leveraging our algorithm we show that this step can be implemented efficiently – taking constant rounds of communication for convergence, independent of the condition number. We also extend our results to loss function of the form quadratic plus regularizer where the introduction of randomness allows us to obtain improved problem dependent convergence rates. Effectiveness of our proposed algorithm is demonstrated through empirical evaluations on real data.

1. Introduction

Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) is a popular and well-established method which is ideal for large scale optimization that has found applications in image processing, control, communication networking, among many others. ADMM is commonly employed for distributed optimization where each node solves a sub-problem using only local data and consensus is reached for global variable by passing local solutions around among the workers. Despite the powerful framework ADMM offers and its enormous use cases, the convergence rate of the algorithm crucially depends on problem dependent quantities such as conditioning of the local and global data matrices. Establishing convergence rates of ADMM is an important topic which gained recent interest (Shi et al., 2014; Hong & Luo, 2017) and lower bounds (Nishihara et al., 2015) confirm that the convergence rate can be slow under worst case data distribution and initialization.

In this paper, we propose a randomized version of ADMM where the data is distributed via random projections instead of naive sub-sampling. We give a powerful primitive for solving cone programs with distributed data storage, showing that for quadratic loss, the convergence rate can be made condition number free, without compromising the lightweight communication desideratum. We account for more flexible settings through incorporation of regularizers, and extend to high-dimensional regimes where the data matrices are split feature-wise instead of instance-wise, where in both cases, we obtain improved problem dependent convergence rates.

2. Related Work

Our work is close in spirit to sketching algorithms where the data is randomly projected to a lower dimensional space for computational efficiency (Vempala, 2004; Mahoney, 2011; Woodruff, 2014; Muthukrishnan, 2005). A common theme in these works is to build upon the geometric properties of random orthogonal matrices, and the concentration of measure phenomenon, to achieve compression while preserving essential information in the data. Sketching for least squares, constrained least squares and low rank approximation have previously been studied in the literature (Mahoney, 2011; Pilanci & Wainwright, 2014; Halko et al., 2011). More recent work considers iterative sketching methods, as well as statistical properties of sketching (Pilanci & Wainwright, 2016; Raskutti & Mahoney, 2015; Wang et al., 2017b; Sridhar et al., 2020). Among these, one line of work, which considers randomized pre-conditioning based on sketching (Rokhlin & Tygert, 2008; Avron et al., 2010), enjoys condition number independent linear convergence to an approximate solution of the *original* quadratic objective. A notable limitation of these methods, however, lies in the implementation of the preprocessing step where a centralized matrix decomposition is required, after which the $d \times d$ preconditioner needs to be communicated to all workers in order to obtain the claimed rate. Similar randomized preconditioner scheme is considered in (Wang et al., 2017a),

[1]Stanford University. Correspondence to: Qijia Jiang <qjiang2@stanford.edu>.

Preliminary work. Under review.

albeit working with quadratic loss only. At the other end sits (stochastic) first-order and quasi-second-order methods including accelerated gradient descent (Nesterov, 1983), randomized Kaczmarz (Strohmer & Vershynin), FISTA (Beck & Teboulle, 2009), L-BFGS (Nocedal, 1980), and approximate Newton type methods (Zhang & Xiao, 2015; Shamir et al., 2014), all of which are able to work with distributed data, at the cost of having the global iteration complexity that necessarily scales with the condition number in certain parameter regimes. This in turn entails considerable amount of synchronization throughout the execution, and makes matter even worse in presence of stragglers. Our proposed method, in contrast, takes fixed rounds of communication, inherently oblivious to the problem size and data on hand. For general conic programs, the options for distributed computing are much more scarce. State-of-the-art solvers (Brendan et al., 2016; Stellato et al., 2018) rely on operator splitting techniques, which reliably provide modest accuracy solution with computationally inexpensive operations, in the regime where interior-point methods run into scalability issues.

While it is known that various sketching techniques generally speed up computation in the centralized storage setting, investigating sketching in the context of distributed computing, *without* transmitting large matrices around during the iterations, while maintaining *condition-number free* rate, with guarantee for *high-accuracy* solution for any user-specified accuracy, hasn't been explored before to the best of our knowledge.

3. ADMM Lower Bound

The convergence of ADMM can be prohibitively slow depending on the conditioning of the local data matrices, assumed to be stored in two separate machines. Consider minimizing $f(x) = \frac{1}{2}\|Ax - b\|_2^2$, where

$$A := \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q^{1/2} \\ \sqrt{\delta} I_d \end{bmatrix} \quad \text{and} \quad b := \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0_d \\ 0_d \end{bmatrix}$$

with $m \cdot I_d \preceq Q \preceq L \cdot I_d$ a positive definite matrix. It is obvious that $x^* = 0_d$. With the data matrix and response vector split across 2 workers, ADMM will form the following augmented Lagrangian for the consensus constraint $x = y$ on the local variable:

$$\min_{x,y} \max_{\lambda} \frac{1}{2}\|A_1 x - b_1\|_2^2 + \frac{1}{2}\|A_2 y - b_2\|_2^2 + \lambda^\top (x - y) + \frac{\rho}{2}\|x - y\|_2^2 ,$$

and employ the update below for alternating minimization (combining linear and quadratic terms in augmented Lagrangian, and identifying u with λ/ρ)

$$\begin{aligned} x_{k+1} &= \arg\min_x \; x^\top Q x + \rho\|x - y_k + u_k\|_2^2 \\ &= \Big(\frac{1}{\rho} Q + I\Big)^{-1} (y_k - u_k) \\ y_{k+1} &= \arg\min_y \; \delta\|y\|_2^2 + \rho\|x_{k+1} - y + u_k\|_2^2 \\ &= \frac{\rho}{\rho + \delta}(x_{k+1} + u_k) \\ u_{k+1} &= u_k + x_{k+1} - y_{k+1} , \end{aligned}$$

where each machine keeps local variable x and y, and only length d vectors are communicated at each iteration. We can rewrite the update more concisely as

$$y_{k+1} := T y_k := \Big(\frac{\rho(\rho - \delta)}{\rho + \delta}(Q + \rho I)^{-1} + \frac{\delta}{\rho + \delta} I\Big) y_k,$$

where the eigenvalues of T are $1 - \frac{\rho(\lambda+\delta)}{(\rho+\delta)(\lambda+\rho)}$ and λ is an eigenvalue of Q (so $m \le \lambda \le L$). If we pick the step size $\rho = m^{1/2-\epsilon} L^{1/2+\epsilon}$, the convergence rate is lower bounded as follows with the argument in (Nishihara et al., 2015):

$$\|y_{k+1}\|_2 \ge \Big(1 - \frac{2}{1 + \kappa^{0.5+|\epsilon|}}\Big)^k \|y_1\|_2,$$

where $\kappa = L/m$ denotes the condition number of Q. Therefore we see the *unavoidable* dependence of the convergence rate of ADMM on the (potentially ill-behaved) worst local conditioning of the partitioned data.

4. RDMM For Quadratic Loss

In this section, we give the algorithm for solving large-scale quadratic problems in a distributed fashion with little communication overhead and condition-number free convergence rate. More general problems, such as logistic regression, which can be cast as solving a sequence of weighted least squares problems, can also benefit from the algorithm.

4.1. Randomized Preprocesing

Note that ADMM splits the data matrix $A \in \mathbb{R}^{n \times d}$ into $S_1 A$ and $S_2 A$ where $S_1 \in \mathbb{R}^{n_1 \times n}$ and $S_2 \in \mathbb{R}^{n_2 \times n}$ are sub-sampling matrices which satisfy $S_1^\top S_1 + S_2^\top S_2 = I_n$. Alternatively we can consider more general decompositions that satisfy the following property.

Definition 1. We say that $\{S_i\}$ provides a δ-*stable decomposition of identity* if it holds that $\sum_i S_i^\top S_i = I$ and $\forall i$,

$$\inf_{\|Az\|_2 = 1} \|S_i A z\|_2^2 \ge 1 - \delta, \qquad \sup_{\|Az\|_2 = 1} \|S_i A z\|_2^2 \le 1 + \delta$$

for some constant $0 < \delta < 1$.

This assumption says that S_i forms a subspace embedding for the column space of A, i.e., it preserves norms for vectors

lying in the column span of A up to an additive factor. One example of a stable decomposition is the pair of Subsampled Randomized Hadamard matrices (Tropp, 2011), i.e., $S_1 = P_1 HD$ and $S_2 = P_2 HD$ where P_1 and P_2 are two diagonal selection matrices denoting disjoint sets of indices, H is the Walsh-Hadamard matrix, and D is a random diagonal matrix with ± 1 entries. The following proposition establies that taking $n_i = \mathcal{O}(d \log d)$ gives $\delta = \mathcal{O}(1)$, but in general if $N \ll n/d$, the subspace embedding holds with $\delta = dN/n \ll 1$.

Proposition 1 (SRHT Preserves Geometry, generalized from (Tropp, 2011))**.** Fix a $n \times d$ matrix U with orthonormal columns, and draw a random $\ell \times n$ SRHT matrix S where the embedding dimension $l = n/N < n$ satisfies

$$\ell = \Omega\left(\frac{d\log(d) \vee \log(dn)\log(d)}{dN/n + (1 - dN/n)\log(1 - dN/n)}\right).$$

Then except with probability $\mathcal{O}(d^{-1})$,

$$\sqrt{1 - dN/n} \le \sigma_d(SU) \quad \text{and} \quad \sigma_1(SU) \le \sqrt{1 + dN/n}\,.$$

In fact replacing H with any unitary matrix with uniformly small entries that is equipped with fast matrix-vector multiply (i.e., order $\mathcal{O}(n \log n)$) will work as well. It is important to note that after the preprocessing step $S_i A$, which can be trivially parallelized across columns among workers, the local data $S_i A$ is well-conditioned with respect to the $\|\cdot\|_{A^\top A}$ norm, but not the $\|\cdot\|_2$ norm; therefore naive implementation of ADMM or first-order (accelerated) gradient descent on the randomized data won't exhibit comparable condition-number free performance.

4.2. Distributed and Parallel Computing

We proceed by performing updates with the preprocessed data matrix split among N workers across the rows, assuming that $\{S_i\}$ form δ-stable decomposition. This implies that each local data matrix $A^\top S_i^\top S_i A$ is invertible and is also equivalent to saying for U the left singular matrix of A,

$$(1-\delta)U^\top U \preceq U^\top S_i^\top S_i U \preceq (1+\delta)U^\top U\,. \tag{1}$$

We propose the following algorithm.

Algorithm 1 RDMM for LS (N Workers)

Input: Data batches $\{S_i A, S_i b\}_{i=1}^N$, stepsize μ
Initialize $\lambda_i^0 = 0_n\ \forall i \in [N]$.
for $k = 0$ **to** maxiter-1 **do**
$x_i^{k+1} = (A^\top S_i^\top S_i A)^{-1}(A^\top S_i^\top S_i b - \lambda_i'^k)\ \forall i \in [N]$
$\lambda_i'^{k+1} = \lambda_i'^k + \mu A^\top A(x_i^{k+1} - \bar{x}^{k+1})\ \forall i \in [N]$
end for
Output: Any one of x_i^{maxiter}.

This algorithm can be cast as a special instance of the dual decomposition method on the following consensus form for the least squares objective $f(x) = 1/2\|Ax - b\|_2^2$:

$$\begin{aligned} \underset{\{z_i\}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^N \|S_i A z_i - S_i b\|_2^2 \\ \text{subject to} \quad & Az_i = A\bar{z} \quad \forall i \in [N] \end{aligned} \tag{2}$$

for $\bar{z} = \frac{1}{N}\sum_{i=1}^N z_i$. Denoting $f_i(z_i) = \frac{1}{2}\|S_i A z_i - S_i b\|_2^2$, we have the following Langrangian,

$$\max_{\lambda_i}\ \min_{\{z_i\}}\quad \sum_{i=1}^N f_i(z_i) + \sum_{i=1}^N \lambda_i^\top\Big(Az_i - \frac{1}{N}\sum_{j=1}^N Az_j\Big)\,,$$

which letting $f_i^*(v) := \sup_u u^\top v - f_i(u)$ be the conjugate function, is the same as

$$\max_{\{\lambda_i\}}\quad \sum_{i=1}^N -f_i^*\Big(-A^\top\lambda_i + \frac{1}{N}A^\top\lambda_i + \frac{1}{N}\sum_{j\neq i}A^\top\lambda_j\Big)\,.$$

Rewriting the previous step, it reduces to solving the following convex problem at each iteration k, $\max_{\{\lambda_i^k\}}\ \sum_{i=1}^N -f_i^*(-A^\top\lambda_i^k + A^\top\bar{\lambda}^k)$, where we know the gradient of this function w.r.t λ_i is

$$\begin{aligned} &\Big(1 - \frac{1}{N}\Big)A\nabla f_i^*(-A^\top\lambda_i^k + A^\top\bar{\lambda}^k) \\ &\qquad - \frac{1}{N}A\sum_{j\neq i}\nabla f_j^*(-A^\top\lambda_j^k + A^\top\bar{\lambda}^k)\,. \end{aligned} \tag{3}$$

Note that if we initialize with $\frac{1}{N}\sum_i A^\top\lambda_i^0 = 0$, for $k = 0$, using Danskin's theorem, assuming $f_i(\cdot)$ is strongly convex:

$$\begin{aligned} \nabla f_i^*(-A^\top\lambda_i^k + A^\top\bar{\lambda}^k) &= \arg\min_{z_i}\ f_i(z_i) + \lambda_i^{k\top}Az_i \\ &= (A^\top S_i^\top S_i A)^{-1}(A^\top S_i^\top S_i b - A^\top\lambda_i^k) = x_i^{k+1}\,, \end{aligned}$$

which gives the desired dual ascent update on $\{\lambda_i\}$ when putting together with (3)

$$\lambda_i^{k+1} = \lambda_i^k + \mu\Big(Ax_i^{k+1} - \frac{1}{N}\sum_{j=1}^N Ax_j^{k+1}\Big) \tag{4}$$

with stepsize μ. Note the sum $\frac{1}{N}\sum_i A^\top\lambda_i^k = 0$ stays constant for all iterations k, which can be observed by summing up both sides of (4) for $i = 1\cdots, N$. Identifying $\lambda_i'^k := A^\top\lambda_i^k$ gives the algorithm. Therefore Algorithm 1 looks at problem (2) through the lens of the following unconstrained objective:

$$\begin{aligned} \min_{\{\lambda_i'\}}\ \sum_{i=1}^N &\frac{1}{2}(-\lambda_i' + \bar{\lambda}' + A^\top S_i^\top S_i b)^\top(A^\top S_i^\top S_i A)^{-1} \\ &(-\lambda_i' + \bar{\lambda}' + A^\top S_i^\top S_i b) - \frac{1}{2}b^\top S_i^\top S_i b\,. \end{aligned}$$

We note that the dependence on the scaling matrix $A^\top A$ in the dual update step in Algorithm 1, together with the assumption on S_iA, is crucial for our analysis for removing the *condition number* dependence in the convergence rate; this is also why instead of imposing the constraint $z_i = \bar{z}$ we incorporated A into the constraint, which effectively makes the algorithm affine-invariant. We state the convergence rate of the algorithm in the theorem below and defer the proof to Appendix A.2. The analysis hinges on showing a recursion of the form

$$\Delta\lambda_i^{k+1} = \Big[I - \mu Q_i\Big]\Delta\lambda_i^k + \frac{\mu}{N}\sum_{j\neq i}(Q_j - Q_i)\Delta\lambda_j^k$$

on the dual variable $\{\lambda_i\}_{i=1}^N$ for $Q_i = A(A^\top S_i^\top S_i A)^{-1}A^\top$, after which subspace embedding property (1) is invoked on Q_i to show $\kappa(Q_i) = \mathcal{O}(1)$ and the dual progress is translated back to the primal space for reaching the final claimed result.

Theorem 2 (RDMM for Quadratic Loss). *Let $\{S_i\}_{i\in[N]}$ form a δ-stable decomposition of I_n for $N = \mathcal{O}(\frac{n}{d\log(d)})$ and assume that the solution x^* to the overdetermined linear system is unique. Then Algorithm 1 using $\mu = (1+\delta)(1-\delta)(1-\frac{N-1}{N}\frac{2\delta}{1-\delta})^{-1}$ produces x_i^k such that for all $k \geq 1$,*

$$\frac{1}{N}\sum_{i=1}^N f(x_i^k) - f(x^*) \leq \frac{1}{(1-\delta)^2}\cdot\delta^{2(k-1)} f(x^*)\,.$$

A few remarks about implementation are in order. Note that QR factorization of the sketched submatrix $(A^\top S_i^\top S_i A)^{-1}$ can be cached locally on each machine so that each subsequent iteration only involves one matrix-vector multiply and one back-substitution for solving the linear system in step 1 of the iterative updates, i.e., complexity on the same order as computing gradient in first order methods. Alternatively, an inexact solver such as CG can be called upon to obtain an approximate local solution.

We find it helpful to elaborate on the implementation of step 2. At every iteration, each machine computes in parallel step 1, after which they communicate the length-d vector x_i^{k+1} back to the master node. The master node calculates $\bar{x}^{k+1}$, and send $\{x_i^{k+1}\}_{i=1}^N$ along with $\bar{x}^{k+1}$ back to the N worker nodes. Recall that since we have $A^\top A = A^\top(\sum_{i=1}^N S_i^\top S_i)A$, each worker node then proceeds by computing locally $A^\top S_i^\top S_i A(x_j^{k+1} - \bar{x}^{k+1})$ for all $j \in [N]$ and sends back the resulting N-by-d matrix to the master node for summation and performing the dual update. The updated $\lambda_i'^{k+1}$ are then distributed to each worker for the next iteration of the primal update. Throughout the process, at most N-by-d size matrices are transmitted across machines and no huge matrix movement is involved.

5. RDMM For Regularized Least Squares

In this section, we give the generalization of the algorithm for solving quadratic problem with regularizers.

5.1. High-dimensional ℓ_2 regression

In the case where one wants to fit regularized least squares on high-dimensional data with $d \gg n$, we can take the dual of $g(x) := 1/2\|Ax - b\|_2^2 + \eta/2\|x\|_2^2$ to get

$$\min_y\ f(y) := \min_y\ \frac{1}{2\eta}\|A^\top y\|^2 + \frac{1}{2}\|y\|^2 - y^\top b, \quad (5)$$

where the primal and dual solution are related by $x^* = \frac{1}{\eta}A^\top y^*$. If we perform row splitting on this objective, it will correspond to column splitting of the original data matrix A (which we assume to have $n \ll d$ and full row rank). We take $\eta = 1$ in what follows w.l.o.g.

Algorithm 2 RDMM for High-dimensional Ridge

Input: Data batches $\{S_iA^\top, b\}_{i=1}^N$, stepsize μ
Initialize $\lambda_i'^0 = 0_n\ \forall i$.
for $k = 0$ **to** maxiter-1 **do**
$y_i^{k+1} = (AS_i^\top S_i A^\top + \frac{1}{N}I)^{-1}(\frac{1}{N}b - \lambda_i'^k) \quad \forall i \in [N]$
$\lambda_i'^{k+1} = \lambda_i'^k + \mu(AA^\top + \frac{1}{N}I)(y_i^{k+1} - \bar{y}^{k+1}) \quad \forall i$
end for
Output: Any one of $\frac{1}{\eta}A^\top y_i^{\text{maxiter}}$.

For N workers, consider the following consensus form for solving (5): (where we denote $\hat{A}^\top = [A^\top; 1/\sqrt{N}\cdot I]$)

$$\begin{aligned}\underset{\{y_i\}}{\text{minimize}} \quad & \sum_{i=1}^N \frac{1}{2}\|S_iA^\top y_i\|_2^2 + \frac{1}{2N}\|y_i\|_2^2 - \frac{1}{N}y_i^\top b \\ \text{subject to} \quad & \hat{A}^\top\bar{y} = \hat{A}^\top y_i \quad \forall i \in [N].\end{aligned} \quad (6)$$

Denoting $f_i(y_i) = \frac{1}{2}\|S_iA^\top y_i\|_2^2 + \frac{1}{2N}\|y_i\|_2^2 - \frac{1}{N}y_i^\top b$, we have the following Langrangian,

$$\max_{\{\lambda_i\}}\ \min_{\{y_i\}}\quad \sum_{i=1}^N f_i(y_i) + \sum_{i=1}^N \lambda_i^\top\Big(\hat{A}^\top y_i - \frac{1}{N}\sum_{j=1}^N \hat{A}^\top y_j\Big)\,,$$

which letting $f_i^*(v) := \sup_u u^\top v - f_i(u)$ be the conjugate function, becomes the unconstrained objective

$$\max_{\{\lambda_i\}}\quad \sum_{i=1}^N -f_i^*\Big(-\hat{A}\lambda_i + \frac{1}{N}\hat{A}\lambda_i + \frac{1}{N}\sum_{j\neq i}\hat{A}\lambda_j\Big)\,. \quad (7)$$

Rewriting the previous step, (6) boils down to solving the following convex problem at each iteration k, $\max_{\{\lambda_i^k\}}\ \sum_{i=1}^N -f_i^*(-\hat{A}\lambda_i^k + \hat{A}\bar{\lambda}^k)$, where we know the

gradient of this function w.r.t λ_i is

$$\Big(1-\frac{1}{N}\Big)\hat{A}^\top\nabla f_i^*(-\hat{A}\lambda_i^k+\hat{A}\bar{\lambda}^k) - \frac{1}{N}\hat{A}^\top\sum_{j\neq i}\nabla f_j^*(-\hat{A}\lambda_j^k+\hat{A}\bar{\lambda}^k)\,.$$

Note that if we initialize with $\frac{1}{N}\sum_i \hat{A}\lambda_i^0=0$, we have for $k=0$,

$$\nabla f_i^*(-\hat{A}\lambda_i^k+\hat{A}\bar{\lambda}^k)=\arg\min_{z_i}\ f_i(z_i)+\lambda_i^{k\top}\hat{A}^\top z_i = \Big(AS_i^\top S_iA^\top+\frac{1}{N}I\Big)^{-1}\Big(\frac{1}{N}b-\hat{A}\lambda_i^k\Big)=y_i^{k+1}\,,$$

which gives the dual ascent update on $\{\lambda_i\}$ as

$$\lambda_i^{k+1}=\lambda_i^k+\mu\Big(\hat{A}^\top y_i^{k+1}-\frac{1}{N}\sum_{j=1}^N\hat{A}^\top y_j^{k+1}\Big)$$

with stepsize μ at each iteration. Now since the sum $\frac{1}{N}\sum_i\hat{A}\lambda_i^k=0$ stays constant for all iterations k, which we recognize by summing up both sides of the previous display for $i=1\cdots,N$, this is identical to Algorithm 2, where we denote $\lambda_i'^k:=\hat{A}\lambda_i^k$. Therefore to work out the convergence rate it suffices to look at iterate contraction of $\{\lambda_i\}$ for the objective (7). We give the result below and refer the reader to Appendix A.3 for the analysis. The argument begins by deriving a multiplicative bound on the progress made in dual variables at each iteration $\sum_i\|\Delta\lambda_i'^{k+1}\|_2^2\le\alpha^2\sum_i\|\Delta\lambda_i'^k\|_2^2$ for some $\alpha<1$, bearing resemblance to what is done in Theorem 2, after which it is translated to progress in function value $g(\cdot)$ through

$$\frac{1}{2}\|\Delta x_i^{k+1}\|_{A^\top A+\mathrm{I}}^2=\frac{1}{2}\|A^\top\Delta y_i^{k+1}\|_{A^\top A+\mathrm{I}}^2 = \frac{1}{2}\big\|A^\top(AS_i^\top S_iA^\top+N^{-1}I)^{-1}\Delta\lambda_i'^k\big\|_{A^\top A+\mathrm{I}}^2$$

to yield the conclusion below.

Theorem 3 (RDMM for High-dimensional Ridge Regression)**.** *Let $\{S_i\}_{i=1}^N$ form a δ-stable decomposition of I_d for $N=\mathcal{O}(\frac{n}{d\log(d)})$ and assume that the solution y^* to (5) is unique. Then Algorithm 2 with $\mu=(1+\delta)(1-\delta)(1-\frac{N-1}{N}\frac{2\delta}{1-\delta})^{-1}$ produces $x_i^k:=A^\top y_i^k$ such that*

$$\frac{1}{N}\sum_{i=1}^N g(x_i^{k+1})-g(x^*)\le\delta^{2k}\cdot C$$

for all $k\ge1$ and some problem dependent constant C that is a function of $\delta,\lambda(A),\|y^*\|_2$ and N.

The communication cost of performing the updates will be roughly the same as un-regularized case Algorithm 1 with all-broadcast of $\{y_i^{k+1}\}_{i=1}^N$ and each worker node sending back $(AS_i^\top S_iA^\top+\frac{1}{N^2}I)(y_j^{k+1}-\bar{y}^{k+1})$ for all $j\in[N]$. The master node collects to perform the summation and distributes the newly updated $\lambda_i'^{k+1}$ to each worker node to prepare for the next round.

5.2. Quadratic Loss with Smooth Penalty

One can extend the algorithm to incorporate smooth and convex penalty. We consider the case where we have a regularization term $g(x)$ assumed twice-differentiable, m-strongly-convex and L-smooth. We are interested in

$$\min_x\ f(x):=\min_x\ \frac{1}{2}\|Ax-b\|_2^2+g(x). \tag{8}$$

We propose the following Algorithm 3, which are natural updates when the splitting is performed as (for $\hat{A}:=[A\,;\sqrt{L}\cdot I_d]$)

$$\begin{aligned}\underset{x,y}{\text{minimize}}\quad & \frac{1}{2}\left(\|S_1Ax-S_1b\|_2^2+\|S_2Ay-S_2b\|_2^2\right)\\ & +\frac{1}{2}\left(g(x)+g(y)\right)\\ \text{subject to}\quad & \hat{A}x=\hat{A}y\,,\end{aligned} \tag{9}$$

where again we let $\lambda'=\hat{A}^\top\lambda$. It is not hard to see that the algorithm amounts to sequentially updating for the primal and dual variables after formulating the Langarian for the above consensus problem (9).

Algorithm 3 RDMM (Quadratic w/ Regularizer)

Input: Data batches $\{S_iA,S_ib\}_{i=1}^2$, stepsize μ, regularizer $g(\cdot)$
Initialize $\lambda_i'^0=0_d$.
for $k=0$ **to** maxiter-1 **do**
$x_{k+1}=\arg\min_x\ \frac{1}{2}\|S_1Ax-S_1b\|_2^2+\frac{1}{2}g(x)-\lambda_k'^\top x$
$y_{k+1}=\arg\min_y\ \frac{1}{2}\|S_2Ay-S_2b\|_2^2+\frac{1}{2}g(y)+\lambda_k'^\top y$
$\lambda_{k+1}'=\lambda_k'+\mu(A^\top A+L\cdot I_d)(y_{k+1}-x_{k+1})$
end for
Output: Either x_{maxiter} or y_{maxiter}.

Theorem 4 (RDMM for Regularized Least Squares)**.** *Suppose $\{S_i\}$ form a δ-stable decomposition of I_n and assume that the solution x^* to (8) is unique. Then Algorithm 3 with $\mu=\frac{1}{((1-\delta)^{-1}\vee L/m)+(1+\delta)^{-1}}$ produces x_k such that*

$$f(x_{k+1})-f(x^*)\le\left(\max\left\{\frac{L(1+\delta)-m}{L(1+\delta)+m}\,,\delta\right\}\right)^{2k}\cdot C$$

for all $k\ge1$, where we assume $m\cdot I_d\preceq\nabla^2g(x)\preceq L\cdot I_d\ \forall x\in dom(f)$ and C is a data-dependent constant that is a function of $\delta,L,m,\lambda(A)$ and $\|\nabla g(x^*)\|_2$.

Details for the convergence analysis and for handling inexact solves for the first two inner steps are elaborated in Appendix B, with extension to non-smooth penalty and additional use cases also discussed. We note that the convergence rate only depends on the strong convexity and smoothness parameter of the *regularizer* and not the condition number of the *data matrix*, in stark contrast to ADMM as illustrated in Section 3.

Here again, step 1 and 2 can be performed in parallel, and only length-d vectors x_{k+1}, y_{k+1} are communicated to compute $A^\top S_i^\top S_i A(y_{k+1} - x_{k+1})$ for carrying out step 3.

6. RDMM For Splitting Cone Solver

The distributed linear system solver introduced in Section 4 can be employed for implementing the subroutine required for general cone programs using the homogeneous self-dual embedding (Brendan et al., 2016). Standard form of cone program can be written as

$$\begin{aligned} \underset{x,s}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & s = b - Ax \in \mathcal{K} \end{aligned} \tag{10}$$

for $x \in \mathbb{R}^p$ and $s \in \mathbb{R}^q$ with $q \geq p$. Problem data here are $A \in \mathbb{R}^{q\times p}, b \in \mathbb{R}^q, c \in \mathbb{R}^p$ and convex cone $\mathcal{K}$. The embedding forms the KKT condition and reduces problem (10) to a feasibility problem as

$$\begin{aligned} \text{find} \quad & (u, v) \\ \text{such that} \quad & v = Qu, \quad (u, v) \in \mathcal{C} \times \mathcal{C}^* \end{aligned}$$

for

$$Q := \begin{bmatrix} 0 & A^\top & c \\ -A & 0 & b \\ -c^\top & -b^\top & 0 \end{bmatrix} \text{ and } \mathcal{C} := \mathbb{R}^p \times \mathcal{K}^* \times \mathbb{R}_+$$

where $\mathcal{K}^*$ denotes the dual cone. The splitting cone solver SCS (Brendan et al., 2016) then proceeds by transforming the two constraints to

$$\min \ \mathbb{1}_{\mathcal{C}\times\mathcal{C}^*}(u, v) + \mathbb{1}_{Qu=v}(\tilde{u}, \tilde{v}) \ \text{ s.t } \ (u, v) = (\tilde{u}, \tilde{v})$$

and performing ADMM on the problem gives the following algorithm at iteration k, after simplification:

$$\tilde{u}^{k+1} = (I + Q)^{-1}(u^k + v^k) \tag{11}$$
$$u_i^{k+1} = \Pi_{\mathbb{R}^p\times\mathcal{K}^*\times\mathbb{R}_+}(\tilde{u}^{k+1} - v_k) \tag{12}$$
$$v^{k+1} = v^k - \tilde{u}^{k+1} + u^{k+1}\,. \tag{13}$$

At a high level, the algorithm is performing alternating projection onto the two constraint sets (correspond to projection onto linear subspace (11) and convex cone projection (12), which typically admits closed-form solution); the last step (13) is a dual variable update for the consensus constraint. For large-scale setting, Conjugate Gradient is the default option in SCS for solving the first step and in this section, we consider replacing CG with RDMM, which works with distributed data and yields faster convergence, independent of the condition number.

6.1. Algorithm for SOCP using SCS

We specialize the application of SCS to Second-order Cone Program (SOCP) in our discussion, although the same methodology generalizes to other cone programs including LP and SDP. As done in Section 4.1 of (Brendan et al., 2016), the first step (11) amounts to solving for some w:

$$\begin{bmatrix} M & h \\ -h^\top & 1 \end{bmatrix} := \begin{bmatrix} I & A^\top & c \\ -A & I & b \\ -c^\top & -b^\top & 1 \end{bmatrix} \begin{bmatrix} \tilde{u}_x \\ \tilde{u}_y \\ \tilde{u}_\tau \end{bmatrix} = \begin{bmatrix} w_x \\ w_y \\ w_\tau \end{bmatrix}.$$

Block elimination followed by Sherman-Morrison gives

$$\tilde{u}_\tau = w_\tau + c^\top \tilde{u}_x + b^\top \tilde{u}_y\,, \tag{14}$$

$$\begin{bmatrix} \tilde{u}_x \\ \tilde{u}_y \end{bmatrix} = \left(M^{-1} - \frac{M^{-1}hh^\top M^{-1}}{1 + h^\top M^{-1} h}\right)\left(\begin{bmatrix} w_x \\ w_y \end{bmatrix} - w_\tau h\right).$$

Therefore we only need to focus on solving a linear system involving a reduced-size matrix, $M^{-1}(w_x, w_y)$ (the $M^{-1}h$ part can be solved once and cached). Rewriting, we have

$$\tilde{u}_x = (I + A^\top A)^{-1}(w_x - A^\top w_y) \tag{15}$$
$$\tilde{u}_y = w_y + A\tilde{u}_x\,, \tag{16}$$

which means that if A is full column-rank, $\tilde{u}_x$ is equivalently the solution to the following problem (where $\dagger$ denotes pseudo-inverse):

$$\arg\min_z \ \frac{1}{2}\|Az - ((A^\top)^\dagger w_x - w_y)\|_2^2 + \frac{1}{2}\|z\|_2^2\,,$$

and ignoring constant terms we can form the following consensus form (let $\hat{A} := [A; I]$), where again S_i are SRHT matrices and we perform row-splitting on $\hat{A}$:

$$\begin{aligned} \underset{\{z_i\}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^N \|S_i\hat{A}z_i\|_2^2 - \frac{1}{N} z_i^\top (w_x - A^\top w_y) \\ \text{subject to} \quad & \hat{A}z_i = \hat{A}\bar{z} \quad \forall i \in [N]\,. \end{aligned}$$

Leveraging the results from Section 4, RDMM update for $\tilde{u}_x$ can be shown as in Algorithm 4, where we denote $\lambda_i'^k = \hat{A}^\top \lambda_i^k$ and P_i is the row-subsampling operator. Note that since the LHS of the linear system doesn't change throughout the execution of the SCS algorithm, matrix inversion in step 1 can be carried out efficiently by factorizing once at the very beginning and caching the result.

Algorithm 4 RDMM for Solving (15) (N Workers)

Input: Data batches $\{S_i\hat{A}\}_{i=1}^N$, $\{A^\top P_i^\top P_i w_y\}_{i=1}^N$, vector w_x, stepsize μ
Initialize $\forall i$, $\lambda_i'^0 = (\sum_{i=1}^N A^\top P_i^\top P_i w_y - w_x)/N = (A^\top w_y - w_x)/N$.
for $k = 0$ **to** maxiter-1 **do**
  $z_i^{k+1} = -(\hat{A}^\top S_i^\top S_i \hat{A})^{-1}\lambda_i'^k \quad \forall i \in [N]$
  $\lambda_i'^{k+1} = \lambda_i'^k + \mu\hat{A}^\top\hat{A}(z_i^{k+1} - \bar{z}^{k+1}) \quad \forall i \in [N]$
end for
Output: Any one of z_i^{maxiter}.

We proceed by updating $\tilde{u}_y$ and $\tilde{u}_\tau$ as in (16) and (14), followed by projection (12) and vector addition (13) to get u^{k+1} and v^{k+1} for next iteration of the SCS update. Using that $\mathcal{K}_{\text{SOC}}$ is a self-dual cone, for any vector $e \in \mathbb{R}^n$, the projection onto second-order cone is in closed form as

$$\Pi_{\mathcal{K}_{\text{SOC}}}(e) = \begin{cases} e & \text{for } \|e_{1:n-1}\|_2 \le e_n \\ 0 & \text{for } \|e_{1:n-1}\|_2 \le -e_n \\ \frac{\|e_{1:n-1}\|_2 + e_n}{2}[\frac{e_{1:n-1}}{\|e_{1:n-1}\|_2}, 1]^\top & \text{otherwise}. \end{cases}$$

Below we formally state the guarantee for SCS embedding solved using RDMM, which is an almost immediate consequence of the result in Section 3.4 of (Brendan et al., 2016), before giving two concrete instantiations of the framework in the two sections that follow.

Corollary 1 (Aymptotic Convergence of SCS)**.** *With inner projection step (11) approximately solved to accuracy that satisfies at each iteration,*

$$\|\tilde{u}^{k+1} - (I+Q)^{-1}(u^k + v^k)\|_2 \le \zeta^k,$$

where $\zeta^k > 0$, $\sum_k \zeta^k \le \infty$, the algorithm returns (u^k, v^k) that eventually satisfies the optimality conditions for the homogeneous self-dual embedding to any desired accuracy.

6.2. Example I: Robust Least Squares

One application arises in robust least squares, which involves solving an SOCP. Consider the problem

$$\min_{x \in \mathbb{R}^d} \max_{a_i \in \mathcal{E}_i} \left(\sum_{i=1}^n (a_i^\top x - b_i)^2 \right)^{1/2}$$

where $\mathcal{E}_i = \{\bar{a}_i + P_i u \colon \|u\|_2 \le 1\}$ for some PSD matrix P_i. A short calculation reveals that it can be formulated as $\min_x \left(\sum_{i=1}^n (|\bar{a}_i^\top x - b_i| + \|P_i x\|)^2\right)^{1/2}$. Rewriting,

$$\begin{aligned} \underset{x,t,s}{\text{minimize}} \quad & s \\ \text{subject to} \quad & \|t\|_2 \le s \\ & |\bar{a}_i^\top x - b_i| + \|P_i x\|_2 \le t_i \quad \forall i \in [n]. \end{aligned}$$

Put this in the standard form (10), we can easily read off the problem parameter

$$\underset{x,t,s}{\text{minimize}} \quad \begin{bmatrix} 0_d & 0_n & 1 \end{bmatrix}^\top \begin{bmatrix} x & t & s \end{bmatrix}$$

subject to

$$s_1 = \begin{bmatrix} 0_n \\ 0 \end{bmatrix} - \begin{bmatrix} 0_{n\times d} & -I_n & 0_{n\times 1} \\ 0_d & 0_n & -1 \end{bmatrix} \begin{bmatrix} x \\ t \\ s \end{bmatrix} \in \mathcal{K}_{\text{SOC}} \subseteq \mathbb{R}^{n+1}$$

$$s_2 = \begin{bmatrix} 0_d \\ b_i \end{bmatrix} - \begin{bmatrix} -P_i & 0_{d\times n} & 0_{n\times 1} \\ \bar{a}_i & -e_i & 0 \end{bmatrix} \begin{bmatrix} x \\ t \\ s \end{bmatrix} \in \mathcal{K}_{\text{SOC}} \; \forall i \in [n]$$

$$s_3 = \begin{bmatrix} 0_d \\ -b_i \end{bmatrix} - \begin{bmatrix} -P_i & 0_{d\times n} & 0_{n\times 1} \\ -\bar{a}_i & -e_i & 0 \end{bmatrix} \begin{bmatrix} x \\ t \\ s \end{bmatrix} \in \mathcal{K}_{\text{SOC}} \; \forall i \in [n]$$

Altogether this gives a tall and skinny A matrix of size $n + 1 + 2(d+1)n$ by $d + n + 1$, assuming $n > d$.

6.3. Example II: Lasso with Sign Constraint

For data pairs (F, g), Lasso with sign constraint solves

$$\min_z \quad \frac{1}{2}\|Fz - g\|_2^2 + \mu\|z\|_1 \quad \text{s.t.} \quad z \ge 0,$$

which can be formulated as the following SOCP:

$$\begin{aligned} \underset{z,t,w}{\text{minimize}} \quad & \frac{1}{2}w + \mu 1^\top t \\ \text{subject to} \quad & -t \le z \le t,\ z \ge 0,\ \left\| \begin{matrix} 1-w \\ 2(Fz-g) \end{matrix} \right\|_2 \le 1 + w \end{aligned}$$

for $z, t \in \mathbb{R}^d$, $w \in \mathbb{R}$ and $F \in \mathbb{R}^{n\times d}, g \in \mathbb{R}^n$. It is easy to see that we can write the problem as

$$\underset{z,t,w}{\text{minimize}} \quad \begin{bmatrix} 1/2 & \mu 1_d & 0_d \end{bmatrix}^\top \begin{bmatrix} w & t & z \end{bmatrix}$$

subject to

$$s_1 = \begin{bmatrix} 1 \\ -2g \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0_d & 0_d \\ 0_n & 0_d & -2F \\ -1 & 0_d & 0_d \end{bmatrix} \begin{bmatrix} w \\ t \\ z \end{bmatrix} \in \mathcal{K}_{\text{SOC}} \subseteq \mathbb{R}^{n+2}$$

$$s_2 = \begin{bmatrix} 0_d \\ 0_d \\ 0_d \end{bmatrix} - \begin{bmatrix} 0_d & -I_d & I_d \\ 0_d & -I_d & -I_d \\ 0_d & 0_d & -I_d \end{bmatrix} \begin{bmatrix} w \\ t \\ z \end{bmatrix} \in \mathcal{K}_+ \subseteq \mathbb{R}^{3d}$$

Therefore $s \in \mathcal{K}_{\text{SOC}} \times \mathcal{K}_+$, where the non-negative cone is also self-dual with $\Pi_{\mathcal{K}_+}(e) = \max(0, e)$ for any vector e. This gives a matrix A of size $n + 3d + 2$ by $2d + 1$.

7. Numerical Experiments

In this section we present simulations which demonstrate the effectiveness of our algorithm. In all these examples, it can be seen that RDMM outperforms other alternative methods that work with distributed data. All experiments are done using `mpi4py` interface for message passing. Additional results are in Appendix C.

7.1. Quadratic Objective

We perform the experiment on 3 datasets[1] as shown in Table 1, 2, 5. The Discrete Cosine Transform (DCT) preprocessing step is done with 10 workers (in parallel across columns), after which the matrix is split across rows and each of the 5 workers access a subset of the (A,b) pairs. The same stopping criteria is used for the 3 methods where dual residual $\|s^k\| := \|\bar{x}^k - \bar{x}^{k-1}\| \le \sqrt{d}\times 10^{-4} + 10^{-4} \times \|\bar{x}^k\|$. In all cases, ridge is fitted with $\lambda = 10^{-3}$ to avoid numerical

[1]Link to source of data: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

	ADMM	RDMM	RT PCG
DCT	0s	3.15s	3.42s
cached QR	4.56s	5.1s	26.89s
iterative step	21.74s	20.29s	4.55s
# iter	264	55	6

Table 1. Epsilon Dataset (402,000-by-2,000)

	ADMM	RDMM	RT PCG
DCT	0s	664.57s	660.16s
cached QR	3119.62s	3268.89s	18806.36s
iterative step	6326.73s	5142.63s	326.69s
# iter	1069	623	13

Table 2. RCV1 Dataset (724,635-by-47,236)

issue. We also document the effect of stepsize on convergence for a gaussian i.i.d data matrix of size $2^{14} \times 2^9$ in Figure 1 below.

In the above, PCG (Rokhlin & Tygert, 2008) refers to the procedure where DCT sketching is performed on A (parallelized across 10 workers), followed by QR decomposition on the reduced size matrix at the central node to produce the perconditioner $R^{-1} \in \mathbb{R}^{d\times d}$. This $d \times d$ matrix is finally shared among the workers to perform iterative preconditioned CG updates using local data stored on 5 machines. Sketched dimension is picked to be $n/2$.

7.2. Second Order Cone Programming

We perform robust LS as outlined in Section 6.2 on 2 different datasets in Table 3, 4, comparing SCS embedding solved with CG versus RDMM. The DCT step for RDMM is done by 10 workers in parallel, after which the matrix is split across rows and each of the 5 workers gets a subset for solving the linear system in each iteration of the SCS update. We track the progress with (1) primal residual: $\|p^k\| = \|Ax^k + s^k - b\|$; (2) dual residual: $\|d^k\| = \|A^\top y^k + c\|$; (3) primal-dual gap: $|g^k| = |c^\top x^k + b^\top y^k|$. Stopping criteria in both cases is picked to be $\|p^k\|_2 \le \epsilon(1 + \|b\|_2)$ and

Phishing $(1,536,646 \times 11,124$ after embed)		
	CG	RDMM
DCT	0s	385.47s
iterative step	48993.09s	48541.72s

Table 3. Phishing Website Dataset (11,055-by-68)

German $(51,001 \times 1,025$ after embed)		
	CG	RDMM
DCT	0s	0.19s
iterative step	1724.68s	1674.83s

Table 4. German Statlog Dataset (1,000-by-24)

$\|d^k\|_2 \le \epsilon(1 + \|c\|_2)$ and $|g^k| \le \epsilon(1 + |c^\top x^k| + |b^\top y^k|)$ for $\epsilon = 10^{-3}$. We pick $P_i = 0.1 \cdot \mathrm{I}_d$ for the problem. For constrained Lasso as formulated in Section 6.3, we test on a synethetic data with 2 workers, where data matrix F with i.i.d gaussian entries of size 4096×256 are generated with response $g = Fz^* + \epsilon$ for z^* with 50% sparsity level and $\epsilon \sim \mathcal{N}(0, 0.1 \cdot \mathrm{I})$. We pick $\mu = 0.1$, and after a parallel DCT step with negligible time on the embedded matrix of size 5379×513, we observe the duality gap comparison as shown in Figure 2.

8. Discussion

In this paper, we present a randomized variant of ADMM and demonstrate its improvement over classical ADMM both theoretically and empirically. It is worth emphasizing that all our guarantees hold on the *original* objective (and not approximations thereof), for any user-specified accuracy. A one-time preprocessing, coupled with consensus form that takes in account the problem structure, leads to an algorithm with significant speed-ups for a large class of distributed optimization problems widely encountered in practice. We think our work makes an important step towards addressing scalability issues in this setting. More broadly, investigating theoretical properties of random projection based methods in the context of distributed consensus optimization is both a major open problem and promising direction.

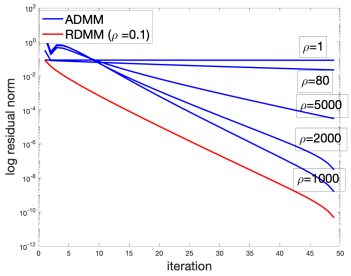

Figure 1. Effect of Stepsize (10 workers)

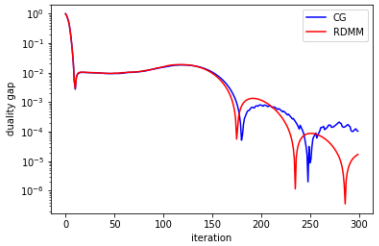

Figure 2. Duality Gap for Sign-Constrained Lasso

References

Avron, H., Maymounkov, P., and Toledo, S. Blendenpik: Supercharging lapack's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, March 2009. doi: 10.1137/080716542. URL https://doi.org/10.1137/080716542.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011. ISSN 1935-8237. doi: 10.1561/2200000016.

Brendan, O., Eric, C., Neal, P., and Stephen, B. Operator splitting for conic optimization via homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, Jun 2016. doi: 10.1007/s10957-016-0892-3.

Halko, N., Martinsson, P., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Hong, M. and Luo, Z.-Q. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1):165–199, Mar 2017. ISSN 1436-4646. doi: 10.1007/s10107-016-1034-2.

Mahoney, M. W. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning in Machine Learning*, 3(2), 2011.

Muthukrishnan, S. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005.

Nesterov, Y. A method for solving the convex programming problem with convergence rate o($1/k^2$). *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

Nishihara, R., Lessard, L., Recht, B., Packard, A., and Jordan, M. I. A general analysis of the convergence of admm. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 343–352. JMLR.org, 2015.

Nocedal, J. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980. ISSN 00255718, 10886842. URL http://www.jstor.org/stable/2006193.

Pilanci, M. and Wainwright, M. J. Randomized sketches of convex programs with sharp guarantees. Technical report, UC Berkeley, 2014. Full length version at arXiv:1404.7203; Presented in part at ISIT 2014.

Pilanci, M. and Wainwright, M. J. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

Raskutti, G. and Mahoney, M. Statistical and algorithmic perspectives on randomized sketching for ordinary least-squares. In *International Conference on Machine Learning*, pp. 617–625, 2015.

Rokhlin, V. and Tygert, M. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008. ISSN 0027-8424. doi: 10.1073/pnas.0804869105.

Shamir, O., Srebro, N., and Zhang, T. Communication-efficient distributed optimization using an approximate newton-type method. volume 32 of *Proceedings of Machine Learning Research*, pp. 1000–1008, Bejing, China, 22–24 Jun 2014. PMLR.

Shi, W., Ling, Q., Yuan, K., Wu, G., and Yin, W. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Trans. Signal Processing*, 62(7):1750–1761, 2014.

Sridhar, S., Pilanci, M., and Özgür, A. Lower bounds and a near-optimal shrinkage estimator for least squares using random projections. *arXiv preprint arXiv:2006.08160*, 2020.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. Osqp: An operator splitting solver for quadratic programs. In *2018 UKACC 12th International Conference on Control (CONTROL)*, pp. 339 – 339, Piscataway, NJ, 2018. IEEE. ISBN 978-1-5386-2864-5. doi: 10.1109/CONTROL.2018.8516834. 12th UKACC International Conference on Control (CONTROL 2018); Conference Location: Sheffield, UK; Conference Date: September 5-7, 2018.

Strohmer, T. and Vershynin, R. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, pp. 2009.

Tropp, J. A. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(1-2):115–126, 2011.

Vempala, S. *The Random Projection Method*. Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, Providence, RI, 2004.

Wang, J., Lee, J., Mahdavi, M., Kolar, M., and Srebro, N. Sketching Meets Random Projection in the Dual: A Provable Recovery Algorithm for Big and High-dimensional Data. In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1150–1158, Fort Lauderdale, FL, USA, 20–22 Apr 2017a. PMLR.

Wang, S., Gittens, A., and Mahoney, M. W. Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging. In *International Conference on Machine Learning*, pp. 3608–3616, 2017b.

Woodruff, D. P. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

Zhang, Y. and Xiao, L. Disco: Distributed optimization for self-concordant empirical loss. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 362–370. JMLR.org, 2015.

A. Missing Proofs

A.1. Proof of Proposition 1

Proof. From Lemma 3.4 in (Tropp, 2011), picking $\delta = \eta = k/l$ and for $N < n/k$,

$$\alpha = \frac{2}{k/l + (1-k/l)\log(1-k/l)} = \frac{2}{kN/n + (1-kN/n)\log(1-kN/n)}\,,$$

we have that for

$$l = \frac{n}{N} \geq \alpha M \log(k)\,,$$

the smallest and largest singular value are bounded as

$$\sqrt{1-k/l} \leq \sigma_{\min}(SU) \quad \text{and} \quad \sigma_{\max}(SU) \leq \sqrt{1+k/l}$$

with failure probability at most $2k^{-1}$. Now Lemma 3.3 in (Tropp, 2011) establishes that except with probability k^{-1},

$$M \leq [\sqrt{k} + \sqrt{8\log(kn)}]^2\,.$$

Altogether we have that for

$$l = \Omega\left(\frac{k\log(k) \vee \log(kn)\log(k)}{kN/n + (1-kN/n)\log(1-kN/n)}\right),$$

the sketching matrix satisfies the subspace embedding property (1) with parameter kN/n and probability at least $1 - 3k^{-1}$. □

A.2. Proof of Theorem 2

In this section, we present the proof of Theorem 2 for the convergence of RDMM on quadratic objective with N workers.

Proof. Let $\lambda_i'^* := A^\top \lambda_i^* := A^\top (S_i^\top S_i b^\perp)$, since $A^\top b^\perp = 0$, we have

$$A^\top \Big(\sum_{i=1}^N S_i^\top S_i\Big) b^\perp = A^\top b^\perp = 0 \quad \Rightarrow \quad \sum_{i=1}^N \lambda_i'^* = 0\,.$$

Introducing $\Delta x_i^k = x_i^k - x_i^*$ and $\Delta\lambda_i^k = \lambda_i^k - \lambda_i^*$, we can rewrite the update for $\{x_i\}$ as

$$\Delta x_i^{k+1} = (A^\top S_i^\top S_i A)^{-1}(A^\top S_i^\top S_i b^\perp - A^\top \lambda_i^k) = (A^\top S_i^\top S_i A)^{-1}(-A^\top \Delta\lambda_i^k)\,, \tag{17}$$

and the update for dual variables $\{\lambda_i\}$ becomes (let $Q_i := A(A^\top S_i^\top S_i A)^{-1}A^\top$)

$$\begin{aligned}
\Delta\lambda_i^{k+1} &= \Delta\lambda_i^k + \mu A\Big(\Delta x_i^{k+1} - \frac{1}{N}\sum_{j=1}^N \Delta x_j^{k+1}\Big) \\
&= \Delta\lambda_i^k + \mu A\Big[(A^\top S_i^\top S_i A)^{-1}(-A^\top \Delta\lambda_i^k) - \frac{1}{N}\sum_{j=1}^N (A^\top S_j^\top S_j A)^{-1}(-A^\top \Delta\lambda_j^k)\Big] \\
&= \Big[I - \mu A(A^\top S_i^\top S_i A)^{-1}A^\top\Big]\Delta\lambda_i^k + \frac{1}{N}\sum_{j=1}^N \mu A(A^\top S_j^\top S_j A)^{-1}(A^\top \Delta\lambda_j^k) \\
&= \Big[I - \mu Q_i\Big]\Delta\lambda_i^k + \frac{\mu}{N}\sum_{j\neq i}(Q_j - Q_i)\Delta\lambda_j^k
\end{aligned}$$

where in the last step we used $\sum_i \Delta\lambda_i^k = \sum_i \lambda_i^k - \sum_i \lambda_i^* = 0 - b^\perp = -b^\perp$ and $Q_j \sum_i \Delta\lambda_i^k = -Q_j b^\perp = 0$ for all j, which follows from (1) $\sum_i \lambda_i^0 = 0$; (2) $\sum_i \lambda_i^{k+1} = \sum_i \lambda_i^k$ for all iterations k. Taking norms on both sides, we end up with

$$\|\Delta\lambda_i^{k+1}\|_2 \leq \|I - \mu Q_i\|_{\text{op}}\|\Delta\lambda_i^k\|_2 + \frac{\mu}{N}\sum_{j\neq i}\|Q_j - Q_i\|_{\text{op}}\|\Delta\lambda_j^k\|_2\,.$$

Now by the assumption that

$$1-\delta \le \|S_i A z\|_2^2 \le 1+\delta \quad \text{for } \|Az\|_2 = 1\,,$$

we have for all $i \in [N]$ and $A = U\Sigma V^\top$ the singular value decomposition,

$$(1+\delta)^{-1} I \preceq Q_i = A(A^\top S_i^\top S_i A)^{-1} A^\top = U(U^\top S_i^\top S_i U)^{-1} U^\top \preceq (1-\delta)^{-1} I\,.$$

Consequently,

$$\begin{bmatrix} \|\Delta\lambda_1^{k+1}\| \\ \|\Delta\lambda_2^{k+1}\| \\ \vdots \\ \|\Delta\lambda_N^{k+1}\| \end{bmatrix} \le \begin{bmatrix} \|I-\mu Q_1\|_{\rm op} & \cdots & \frac{\mu}{N}\|Q_N - Q_1\|_{\rm op} \\ \vdots & \cdots & \vdots \\ \frac{\mu}{N}\|Q_1 - Q_N\|_{\rm op} & \cdots & \|I-\mu Q_N\|_{\rm op} \end{bmatrix} \begin{bmatrix} \|\Delta\lambda_1^{k}\| \\ \|\Delta\lambda_2^{k}\| \\ \vdots \\ \|\Delta\lambda_N^{k}\| \end{bmatrix}$$

which means from Gershgorin's circle theorem that all the eigenvalues of the matrix must be smaller than

$$1 - \frac{\mu}{1+\delta} + \frac{\mu(N-1)}{N}\Big(\frac{1}{1-\delta} - \frac{1}{1+\delta}\Big)\,.$$

Picking stepsize $\mu = \frac{(1+\delta)(1-\delta)}{1-\frac{N-1}{N}\frac{2\delta}{1-\delta}}$, under the assumption that $d/n < 1/(3N-2)$,

$$\sum_{i=1}^N \|\Delta\lambda_i^{k+1}\|_2^2 \le \delta^2 \sum_{i=1}^N \|\Delta\lambda_i^k\|_2^2\,.$$

This in turn gives geometric convergence of $\{x_i\}$ as well since from (17),

$$\begin{aligned} \|A\Delta x_i^{k+1}\|_2 &\le \|A(A^\top S_i^\top S_i A)^{-1} A^\top\|_{\rm op} \|\Delta\lambda_i^k\|_2 \\ &\le \frac{1}{1-\delta}\|\Delta\lambda_i^k\|_2\,, \end{aligned}$$

which implies since $\|\sum_{i=1}^N \lambda_i^*\|_2^2 = \|b^\perp\|_2^2 = 2f(x^*)$ for the quadratic objective,

$$\begin{aligned} \sum_{i=1}^N f(x_i^k) - f(x^*) &= \frac{1}{2}\sum_{i=1}^N \|\Delta x_i^k\|_{A^\top A}^2 \\ &\le \frac{1}{2}\left(\frac{1}{1-\delta}\right)^2 \sum_{i=1}^N \|\Delta\lambda_i^{k-1}\|_2^2 \\ &\le \frac{1}{2(1-\delta)^2}\cdot \delta^{2(k-1)} \sum_{i=1}^N \|\lambda_i^*\|_2^2 \\ &\le \frac{N}{(1-\delta)^2}\cdot \delta^{2(k-1)} f(x^*)\,, \end{aligned}$$

where we used (1) $\lambda_i^0 = 0_n\ \forall i$; (2)$\|\lambda_i^*\|_2 = \|S_i^\top S_i b^\perp\|_2 \le \|b^\perp\|_2$. This gives the claimed condition-number independent convergence rate. □

A.3. Proof of Theorem 3

In this section we include the proof for high-dimensional Ridge regression stated in Theorem 3.

Proof. By optimality condition we have $y^* = (AA^\top + I)^{-1} b$. Now the primal update gives

$$\Delta y_i^{k+1} = \left(AS_i^\top S_i A^\top + \frac{1}{N} I\right)^{-1}\left(\frac{1}{N} b - \lambda_i'^k\right) - (AA^\top + I)^{-1} b\,,$$

which implies for $\lambda_i'^* := \hat{A}\lambda_i^* := \frac{1}{N}b - (AS_i^\top S_i A^\top + \frac{1}{N}I)(AA^\top + I)^{-1}b$ (and therefore $\lambda_i^* = [\frac{1}{N}x^* - S_i^\top S_i x^*; 0_n]$), we have

$$\Delta y_i^{k+1} = -\left(AS_i^\top S_i A^\top + \frac{1}{N}I\right)^{-1}\Delta\lambda_i'^k\,.$$

The definition of $\lambda_i'^*$ yields

$$\begin{aligned}\sum_{i=1}^N \lambda_i'^* &= \sum_{i=1}^N \Big[\frac{1}{N}I - (AS_i^\top S_i A^\top + \frac{1}{N}I)(AA^\top + I)^{-1}\Big]b\\ &= b - (AA^\top + I)(AA^\top + I)^{-1}b = 0\,.\end{aligned}$$

The recursion in the dual variable gives (let $Q_i := (AA^\top + \frac{1}{N}I)(AS_i^\top S_i A^\top + \frac{1}{N}I)^{-1}$)

$$\begin{aligned}\Delta\lambda_i'^{k+1} &= \Delta\lambda_i'^k + \mu(AA^\top + \frac{1}{N}I)\Big(\Delta y_i^{k+1} - \frac{1}{N}\sum_{j=1}^N \Delta y_j^{k+1}\Big)\\ &= \Delta\lambda_i'^k + \mu(AA^\top + \frac{1}{N}I)\Big[(AS_i^\top S_i A^\top + \frac{1}{N}I)^{-1}(-A\Delta\lambda_i^k) - \frac{1}{N}\sum_{j=1}^N (AS_j^\top S_j A^\top + \frac{1}{N}I)^{-1}(-A\Delta\lambda_j^k)\Big]\\ &= \Big[I - \mu(AA^\top + \frac{1}{N}I)(AS_i^\top S_i A^\top + \frac{1}{N}I)^{-1}\Big]\Delta\lambda_i'^k + \frac{1}{N}\sum_{j=1}^N \mu(AA^\top + \frac{1}{N}I)(AS_j^\top S_j A^\top + \frac{1}{N}I)^{-1}\Delta\lambda_j'^k\\ &= \Big[I - \mu Q_i\Big]\Delta\lambda_i'^k + \frac{\mu}{N}\sum_{j\neq i}(Q_j - Q_i)\Delta\lambda_j'^k\,,\end{aligned}$$

where in the last step we used $\sum_i \Delta\lambda_i'^k = \sum_i \lambda_i'^k - \sum_i \lambda_i'^* = 0_n - 0_n$, which follows from (1) $\sum_i \lambda_i'^0 = 0_n$; (2) $\sum_i \lambda_i'^{k+1} = \sum_i \lambda_i'^k$ for all iterations k. Taking norms on both sides, we end up with

$$\|\Delta\lambda_i'^{k+1}\|_2 \le \|I - \mu Q_i\|_{\text{op}}\|\Delta\lambda_i'^k\|_2 + \frac{\mu}{N}\sum_{j\neq i}\|Q_j - Q_i\|_{\text{op}}\|\Delta\lambda_j'^k\|_2\,.$$

Now the assumption that

$$(1-\delta)\|A^\top z\|_2^2 \le \|S_i A^\top z\|_2^2 \le (1+\delta)\|A^\top z\|_2^2\,,$$

implies that for $A = U\Sigma V^\top$ the singular value decomposition, we have $\forall i \in [N]$,

$$(1-\delta)I_n \preceq V^\top S_i^\top S_i V \preceq (1+\delta)I_n\,,$$

therefore for

$$Q_i = \left(AA^\top + \frac{1}{N}I\right)\left(AS_i^\top S_i A^\top + \frac{1}{N}I\right)^{-1} = U\left(\Sigma^2 + \frac{1}{N}I\right)\left(\Sigma V^\top S_i^\top S_i V\Sigma + \frac{1}{N}I\right)^{-1}U^\top\,,$$

this gives

$$\min\left\{\frac{1}{1+\delta}, 1\right\}\cdot I \preceq Q_i \preceq \max\left\{\frac{1}{1-\delta}, 1\right\}\cdot I\,.$$

Consequently,

$$\begin{bmatrix}\|\Delta\lambda_1'^{k+1}\|\\ \|\Delta\lambda_2'^{k+1}\|\\ \vdots\\ \|\Delta\lambda_N'^{k+1}\|\end{bmatrix} \le \begin{bmatrix}\|I-\mu Q_1\|_{\text{op}} & \frac{\mu}{N}\|Q_2 - Q_1\|_{\text{op}} & \cdots & \frac{\mu}{N}\|Q_N - Q_1\|_{\text{op}}\\ \vdots & \cdots & \vdots & \\ \frac{\mu}{N}\|Q_1 - Q_N\|_{\text{op}} & \frac{\mu}{N}\|Q_2 - Q_N\|_{\text{op}} & \cdots & \|I-\mu Q_N\|_{\text{op}}\end{bmatrix}\begin{bmatrix}\|\Delta\lambda_1'^k\|\\ \|\Delta\lambda_2'^k\|\\ \vdots\\ \|\Delta\lambda_N'^k\|\end{bmatrix}$$

which means from Gershgorin's circle theorem that all the eigenvalues are smaller than

$$1 - \frac{\mu}{1+\delta} + \frac{\mu(N-1)}{N}\Big(\frac{1}{1-\delta} - \frac{1}{1+\delta}\Big)\,.$$

Picking stepsize $\mu = \frac{(1+\delta)(1-\delta)}{1-\frac{N-1}{N}\frac{2\delta}{1-\delta}}$, using the assumption that $d/n < 1/(3N-2)$, we have

$$\sum_{i=1}^{N} \|\Delta\lambda_i'^{k+1}\|_2^2 \leq \delta^2 \cdot \sum_{i=1}^{N} \|\Delta\lambda_i'^{k}\|_2^2 .$$

Convergence in $\{y_i\}$ follows trivially from convergence in $\{\lambda_i\}$ since (let $x_i^{k+1} = A^\top y_i^{k+1}$)

$$\begin{aligned}
\sum_{i=1}^{N} g(x_i^{k+1}) - g(x^*) &= \frac{1}{2}\|\sum_{i=1}^{N} \Delta x_i^{k+1}\|_{A^\top A+\mathrm{I}}^2 \\
&= \frac{1}{2}\|\sum_{i=1}^{N} A^\top \Delta y_i^{k+1}\|_{A^\top A+\mathrm{I}}^2 \\
&= \frac{1}{2}\left\|\sum_{i=1}^{N} A^\top (AS_i^\top S_i A^\top + \frac{1}{N}I)^{-1}\Delta\lambda_i'^{k}\right\|_{A^\top A+\mathrm{I}}^2 \\
&\leq \frac{1}{2}\max_j \left\|(AS_j^\top S_j A^\top + \frac{1}{N}I)^{-1}A(A^\top A + I)A^\top (AS_j^\top S_j A^\top + \frac{1}{N}I)^{-1}\right\|_{\mathrm{op}} \cdot \sum_{i=1}^{N} \|\Delta\lambda_i'^{k}\|_2^2 \\
&\leq \left(1 + \frac{1}{\lambda_{\min}(A)^2}\right)\frac{1}{2(1-\delta)^2} \times \delta^{2k} \cdot \sum_{i=1}^{N} \|\Delta\lambda_i'^{0}\|_2^2 .
\end{aligned}$$

Using $\lambda_i'^0 = 0_n \ \forall i$ and

$$\begin{aligned}
\|\lambda_i'^*\|_2^2 &= \left\|\frac{1}{N}b - (AS_i^\top S_i A^\top + \frac{1}{N}I)(AA^\top + I)^{-1}b\right\|_2^2 \\
&= \|\frac{1}{N}AA^\top y^* - AS_i^\top S_i A^\top y^*\|_2^2 \\
&\leq \left(1 + \delta - \frac{1}{N}\right)^2 \lambda_{\max}(A)^2 \|y^*\|_2^2
\end{aligned}$$

finish the proof. □

B. Regularized Least Squares

B.1. Convergence Analysis

We begin by presenting the proof for the convergence guarantee of Algorithm 3, as stated in Theorem 4.

Proof. Optimality conditions from step 1 and 2 imply that

$$A^\top S_1^\top S_1 A x_{k+1} = \lambda_k' + A^\top S_1^\top S_1 b - \frac{\nabla g(x_{k+1})}{2} ,$$

$$A^\top S_2^\top S_2 A y_{k+1} = -\lambda_k' + A^\top S_2^\top S_2 b - \frac{\nabla g(y_{k+1})}{2} .$$

Now since

$$A^\top (Ax^* - b) + \nabla g(x^*) = 0 \quad \text{and} \quad A^\top (Ay^* - b) + \nabla g(y^*) = 0 , \tag{18}$$

define $\Delta_{x_k} = x_k - x^*$ and $\Delta_{y_k} = y_k - y^*$, (note trivially $x^* = y^*$) we have

$$\begin{aligned}
&A^\top S_1^\top S_1 A \Delta_{x_{k+1}} \\
&= \lambda_k' + A^\top S_1^\top S_1 b - \frac{\nabla g(x_{k+1})}{2} - A^\top S_1^\top S_1 A x^*
\end{aligned}$$

$$
\begin{aligned}
&= \lambda'_k + A^\top S_1^\top S_1 b - \frac{\nabla g(x_{k+1})}{2} - A^\top S_1^\top S_1 A[(A^\top A)^{-1}(A^\top b - \nabla g(x^*))] \\
&= \lambda'_k + A^\top S_1^\top S_1 b^\perp - \frac{\nabla g(x_{k+1})}{2} + A^\top S_1^\top S_1 A(A^\top A)^{-1}\nabla g(x^*) \\
&= \lambda'_k + A^\top S_1^\top S_1 b^\perp + \nabla g(x^*) - \frac{1}{2}\nabla g(x_{k+1}) - (A^\top S_2^\top S_2 A)(A^\top A)^{-1}\nabla g(x^*) \\
&= \lambda'_k + A^\top S_1^\top S_1 b^\perp + \nabla g(x^*) - \frac{1}{2}\nabla g(x_{k+1}) + (A^\top S_2^\top S_2 A)(A^\top A)^{-1}A^\top(Ax^* - b)\,,
\end{aligned}
$$

where we let $b^\perp := b - (A^\top A)^{-1}A^\top b$ and used $A^\top S_1^\top S_1 A + A^\top S_2^\top S_2 A = I$. Similarly for y_k,

$$
A^\top S_2^\top S_2 A\Delta_{y_{k+1}} = -\lambda'_k + A^\top S_2^\top S_2 b^\perp + \nabla g(y^*) - \frac{1}{2}\nabla g(y_{k+1}) + A^\top S_1^\top S_1 A(A^\top A)^{-1}A^\top(Ay^* - b)\,.
$$

Define

$$
\begin{aligned}
\lambda'^* &= -A^\top S_1^\top S_1 b^\perp - \frac{\nabla g(x^*)}{2} - A^\top S_2^\top S_2 A(A^\top A)^{-1}A^\top(Ax^* - b) \\
&= A^\top S_2^\top S_2 b^\perp + \frac{\nabla g(y^*)}{2} + A^\top S_1^\top S_1 A(A^\top A)^{-1}A^\top(Ay^* - b)\,.
\end{aligned}
$$

The last equality is justified by observing that it implies

$$
-\frac{\nabla g(x^*)}{2} - \frac{\nabla g(y^*)}{2} = -\nabla g(x^*) = A^\top(Ax^* - b) + A^\top b^\perp = A^\top(Ax^* - b)\,,
$$

which holds in light of (18). The recursion then becomes for $\Delta_{\lambda'_k} = \lambda'_k - \lambda'^*$,

$$
\begin{aligned}
A^\top S_1^\top S_1 A\Delta_{x_{k+1}} &= \Delta_{\lambda'_k} + \frac{1}{2}(\nabla g(x^*) - \nabla g(x_{k+1})) \\
A^\top S_2^\top S_2 A\Delta_{y_{k+1}} &= -\Delta_{\lambda'_k} + \frac{1}{2}(\nabla g(y^*) - \nabla g(y_{k+1}))\,.
\end{aligned}
$$

Using the Mean Value Theorem we can expand $g(x)$ around x^* as

$$
\nabla g(x) = \nabla g(x^*) + \int_0^1 \nabla^2 g(x + t\Delta_x)\cdot\Delta x\, dt\,,
$$

and applying the above for $\Delta_{x_{k+1}}$ and $\Delta_{y_{k+1}}$ we obtain

$$
\begin{aligned}
\left(A^\top S_1^\top S_1 A + \frac{1}{2}\int_0^1 \nabla^2 g(x_{k+1} + t\Delta_{x_{k+1}})\, dt\right)\cdot\Delta_{x_{k+1}} &= \Delta_{\lambda'_k} \\
\left(A^\top S_2^\top S_2 A + \frac{1}{2}\int_0^1 \nabla^2 g(y_{k+1} + t\Delta_{y_{k+1}})\, dt\right)\cdot\Delta_{y_{k+1}} &= -\Delta_{\lambda'_k}\,.
\end{aligned}
$$

Defining the short hand H_x and H_y for the integrals on the LHS of the above display we obtain

$$
\begin{aligned}
\Delta_{\lambda'_{k+1}} &= \Delta_{\lambda'_k} + \mu(A^\top A + L\cdot I)(\Delta_{y_{k+1}} - \Delta_{x_{k+1}}) \\
&= \left[I_n - \mu(A^\top A + L\cdot I)\left((A^\top S_2^\top S_2 A + H_y)^{-1} + (A^\top S_1^\top S_1 A + H_x)^{-1}\right)\right]\Delta_{\lambda'_k} \\
&= \left[I_n - \mu V(\Sigma^2 + L\cdot I)\left((\Sigma U^\top S_2^\top S_2 U\Sigma + V^\top H_y V)^{-1} + (\Sigma U^\top S_1^\top S_1 U\Sigma + V^\top H_x V)^{-1}\right)V^\top\right]\Delta_{\lambda'_k}\,,
\end{aligned}
$$

where $A = U\Sigma V^\top$ is the singular value decomposition. Now using the assumption that

$$
(1-\delta)\cdot I_d \preceq U^\top S_i^\top S_i U \preceq (1+\delta)\cdot I_d \;\;\forall i \quad\text{and}\quad m\cdot I_d \preceq \nabla^2 g(x) \preceq L\cdot I_d\,,
$$

we have

$$
\min\left(\frac{1}{1+\delta}, 1\right)\cdot I \preceq (\Sigma^2 + L\cdot I_d)\cdot(\Sigma U^\top S_i^\top S_i U\Sigma + V^\top H_i V)^{-1} \preceq \max\left(\frac{1}{1-\delta}, \frac{L}{m}\right)\cdot I\,.
$$

Denoting $u := \max((1-\delta)^{-1}, L/m)$ and putting things together

$$\|\Delta_{\lambda'_{k+1}}\|_2 \leq \max\left\{\left|1-\left(\frac{2\mu}{1+\delta}\right)\right|, \left|2\mu \cdot u - 1\right|\right\} \|\Delta_{\lambda'_k}\|_2 \,.$$

It is easy to see that $\mu = \frac{1}{u+(1+\delta)^{-1}}$ minimizes the expression, yielding geometric convergence

$$\|\Delta_{\lambda'_{k+1}}\|_2 \leq \left(\frac{u(1+\delta)-1}{u(1+\delta)+1}\right) \|\Delta_{\lambda'_k}\|_2 \,,$$

which in turn gives

$$\|\Delta_{\lambda'_{k+1}}\|_2 \leq \max\left\{\frac{L(1+\delta)-m}{L(1+\delta)+m}, \delta\right\} \|\Delta_{\lambda'_k}\|_2 \,.$$

Convergence in primal variable follows immediately from that for dual variable since by smoothness of the regularizer we have

$$\begin{aligned} g(x_k) - g(x^*) &= (x_k - x^*)^\top \Big(\int_0^1 \int_0^p \nabla^2 g(x^* + z(x_k - x^*))\, dzdp \Big)(x_k - x^*) \\ &\leq \int_0^1 \int_0^p L \cdot \|x_k - x^*\|_2^2 \, dzdp = \frac{L}{2}\|x_k - x^*\|_2^2 \,. \end{aligned}$$

Therefore in terms of function value

$$\begin{aligned} f(x_{k+1}) - f(x^*) &\leq \frac{1}{2}\|\Delta x_{k+1}\|^2_{A^\top A + L\cdot \mathrm{I}} \\ &\leq \frac{1}{2}\|(A^\top S_1^\top S_1 A + H_x)^{-1}\Delta_{\lambda'_k}\|^2_{A^\top A + L\cdot \mathrm{I}} \\ &\leq \frac{1}{2}\|(A^\top S_1^\top S_1 A + H_x)^{-1}(A^\top A + L\cdot I)(A^\top S_1^\top S_1 A + H_x)^{-1}\|_{\mathrm{op}} \cdot \|\Delta\lambda'_k\|_2^2 \\ &\leq \frac{\max((1-\delta)^{-1}, L/m)}{2(\lambda_{\min}^2(1-\delta)+m)} \cdot \|\Delta\lambda'_k\|_2^2 \\ &\leq \frac{\max((1-\delta)^{-1}, L/m)}{2(\lambda_{\min}^2(1-\delta)+m)} \cdot \left(\max\left\{\frac{L(1+\delta)-m}{L(1+\delta)+m}, \delta\right\}\right)^{2k} \cdot \|\lambda'^*\|_2^2 \,, \end{aligned}$$

where in the above

$$\begin{aligned} \|\lambda'^*\|_2^2 &= \|A^\top S_2^\top S_2 b^\perp - \frac{1}{2}A^\top(Ax^* - b) + A^\top S_1^\top S_1 P_A(Ax^* - b)\|_2^2 \\ &= \left\|\frac{1}{2}A^\top S_1^\top S_1(Ax^* - b) - \frac{1}{2}A^\top S_2^\top S_2(Ax^* - b)\right\|_2^2 \\ &\leq \frac{1}{2}\|A^\top S_1^\top S_1(Ax^* - b)\|_2^2 + \frac{1}{2}\|A^\top S_2^\top S_2(Ax^* - b)\|_2^2 \\ &\leq \|A^\top(Ax^* - b)\|_2^2 = \|\nabla g(x^*)\|_2^2 \,, \end{aligned}$$

and we used partial isometry of S_i. Convergence in y_k follows analogously. □

B.2. Handling Non-smooth Penalty

For non-smooth penalty such as ℓ_1 norm, we can consider log-sum-exp function (with parameter μ) as smooth surrogate. Since $|x_i| = \max(x_i, -x_i, 0)$, we have

$$\|x\|_1 = \sum_{i=1}^d |x_i| \simeq \sum_{i=1}^d \mu \log(e^{x_i/\mu} + e^{-x_i/\mu} + 1) - \mu \log 3 \,.$$

Furthermore, since $f(x) - f^* \leq f_\mu(x) - f_\mu^* + \log(3)\mu d$, to achieve ϵ-accuracy, it suffices to pick μ such that $\log(3)\mu d \leq \frac{\epsilon}{2}$ and solve the smoothed objective to $\epsilon_\mu = \frac{\epsilon}{2}$ precision and the smoothness parameter of f_μ becomes $L_\mu = \frac{2d}{\mu}$. In particular, note that this function is both twice-differentiable and strictly convex, and μ can be decreased gradually during optimization. Similar smoothing techniques can be applied to $\|\cdot\|_2$ and $\|\cdot\|_{\mathrm{TV}}$, for example.

B.3. Inexact Inner Minimization

In practice, one might want to call iterative solver for solving step 1 and 2 in Algorithm 3 approximately, which will incur an additive term in the convergence rate bound, under appropriate conditions on the sub-optimality gap. Suppose at each iteration, the algorithm outputs an approximate minimizer $\tilde{x}_{k+1}$ such that

$$\|(A^\top A + L \cdot I) \cdot (\tilde{x}_{k+1} - x_{k+1})\|_2 \leq \epsilon \cdot \|(A^\top A + L \cdot I) \cdot \Delta_{x_{k+1}}\|_2$$

and similarly for y_{k+1}. Repeating the same argument in this case the error recursion for the dual parameter becomes

$$\begin{aligned}
\|\Delta_{\lambda'_{k+1}}\|_2 &= \|\Delta_{\lambda'_k} + \mu(A^\top A + L \cdot I)(\Delta_{\tilde{y}_{k+1}} - \Delta_{\tilde{x}_{k+1}})\|_2 \\
&\leq \left\| \left(I_n - \mu(A^\top A + L \cdot I)\left[(A^\top S_2^\top S_2 A + H_y)^{-1} + (A^\top S_1^\top S_1 A + H_x)^{-1}\right]\right) \Delta_{\lambda'_k} \right\|_2 \\
&\quad + \mu \left\| (A^\top A + L \cdot I)(\tilde{x}_{k+1} - x_{k+1}) \right\|_2 + \mu \left\| (A^\top A + L \cdot I)(\tilde{y}_{k+1} - y_{k+1}) \right\|_2 \\
&\leq \max\left\{ \left| 1 - \left(\frac{2\mu}{1+\delta}\right) \right|, \left| 2\mu \cdot u - 1 \right| \right\} \cdot \|\Delta_{\lambda'_k}\|_2 + \mu\epsilon \|(A^\top A + L \cdot I) \cdot \Delta_{x_{k+1}}\|_2 + \mu\epsilon \|(A^\top A + L \cdot I) \cdot \Delta_{y_{k+1}}\|_2 \\
&\leq \max\left\{ \left| 1 - \left(\frac{2\mu}{1+\delta}\right) \right|, \left| 2\mu \cdot u - 1 \right| \right\} \cdot \|\Delta_{\lambda'_k}\|_2 + \mu\epsilon \left\| (A^\top A + L \cdot I) \cdot (A^\top S_1^\top S_1 A + H_x)^{-1} \Delta_{\lambda'_k} \right\|_2 \\
&\quad + \mu\epsilon \left\| (A^\top A + L \cdot I) \cdot (A^\top S_2^\top S_2 A + H_y)^{-1} \Delta_{\lambda'_k} \right\|_2 \\
&\leq \left[\max\left\{ \left| 1 - \left(\frac{2\mu}{1+\delta}\right) \right|, \left| 2\mu \cdot u - 1 \right| \right\} + 2\mu\epsilon \cdot u \right] \cdot \|\Delta_{\lambda'_k}\|_2 .
\end{aligned}$$

This implies that (again picking $\mu = \frac{1}{u + (1+\delta)^{-1}}$),

$$\|\Delta_{\lambda'_{k+1}}\|_2 \leq \left(\frac{u(1+\delta) - 1}{u(1+\delta) + 1} + \frac{2\epsilon u (1+\delta)}{u(1+\delta) + 1} \right) \|\Delta_{\lambda'_k}\|_2 \leq \max\left\{ \frac{(2\epsilon + 1)L(1+\delta) - m}{L(1+\delta) + m}, \delta + \epsilon\delta + \epsilon \right\} \|\Delta_{\lambda'_k}\|_2 .$$

The relative error criterion implies that the minimization should be carried out more accurately as we approach the minimizer.

B.4. Additional Use Cases

In general we have the following primal-dual pairs, where $f^*(\cdot)$ denotes the Fenchel dual of the function $f(\cdot)$:

$$\begin{aligned}
\text{minimize}_x \quad & P(x) := f(x) + \delta_{\mathcal{C}}(Ax - b) \\
\text{maximize}_z \quad & D(z) := -b^\top z - \delta^*_{\mathcal{C}}(z) - f^*(-A^\top z)
\end{aligned}$$

Taking $f(\cdot) = \frac{1}{2}\|\cdot\|_2^2$, and $\mathcal{C}$ to be the unit $\|\cdot\|$-ball for example, we will have $\delta^*_{\mathcal{C}}(z) = \|z\|_*$ (where $\|\cdot\|_*$ denotes the dual norm), therefore we can equivalently write the dual problem as (assuming A is full row-rank with $d > n$)

$$\text{minimize}_z \quad \frac{1}{2}\|A^\dagger b - (-A^\top z)\|_2^2 + \|z\|_* ,$$

which is another quadratic problem with a different regularizer, therefore the above procedure can be brought to bear on a broader class of problems by looking at the dual.

Moreover, we have the primal-dual pairs

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + g(x) \Longleftrightarrow \max_y -\frac{1}{2}\|y\|^2 + y^\top b - g^*(-A^\top y) ,$$

where $g^*(\cdot)$ denotes the convex conjugate of $g(\cdot)$. Therefore the dual problem $y^* = \arg\min_y \frac{1}{2}\|y - b\|^2 + g^*(-A^\top y)$ is another regularized least squares problem, for which the above procedure can potentially benefit from.

C. Additional Experiments

We include additional plots and tables to support our finding in this section.

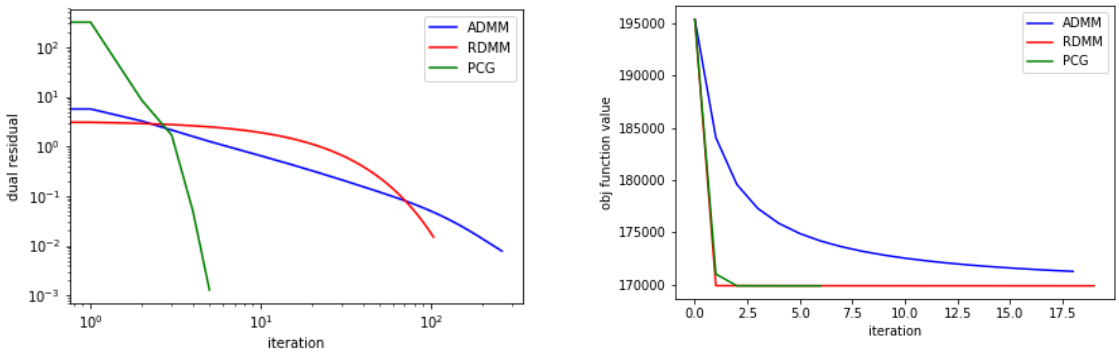

Figure 3. Comparison of the methods on Epsilon Dataset for Quadratic Objective

Criteo CTR (truncated to 850,000-by-50,000)			
	ADMM	RDMM	RT PCG
DCT	0s	194.6s	190.13s
cached QR	3929.57s	4037.58s	21307.11s
iterative step	8488.51s	4132.68s	228s
# iter	1389	712	20

Table 5. Quadratic objective for Criteo Click-through Rate Dataset