Improved First-Order Methods for Certain Classes of Convex Optimization

Trevor Squires

Problem Setting

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$ using first-order, deterministic algorithms.

Here,

- f is a real-valued, convex function
- $\mathbb{R}^n$ is a high dimensional space
- $X \subseteq \mathbb{R}^n$ closed and convex

Specifically for smooth convex optimization,

- ∇f exists and is Lipschitz continuous with Lipschitz constant L, i.e. f is L-smooth
- the projection onto X is computationally feasible

Solving (CO)

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$ using first-order, deterministic algorithms.

We desire algorithms with the following properties

- computationally efficient
  - > low computational time
  - > avoids potentially difficult tasks
- generic
  - > does not make assumptions on f or X
  - > covers a wide variety of applications
- parameter free
  - > does not require knowledge of problem dependent information
  - > potentially adapts to problem setting

Upper Complexity Bound

We'll use oracle complexity theory to analyze computational efficiency

→ Identify an oracle $\mathcal{O}_{f,X} : \mathbb{R}^n \to S$ that encompasses the costly operations of an algorithm

→ Count the number of oracle calls an algorithm requires to compute a solution

Example: Gradient Descent (GD)

$$\begin{aligned} x_k &= \operatorname*{argmin}_{u \in X} \left|\left| u - \left(x_{k-1} - \frac{1}{\eta_k} \nabla f(x_{k-1}) \right) \right|\right|^2 \\ &= \operatorname*{argmin}_{u \in X} \langle \nabla f(x_{k-1}), u \rangle + \frac{\eta_k}{2} ||u - x_{k-1}||^2 \end{aligned}$$

- for properly chosen η_k, GD achieves an ε-solution in $\mathcal{O}(1/\varepsilon)$ iterations when applied to sufficiently smooth problem instances of (CO)
- if $O(x) = (\nabla f(x))$, then since GD makes 1 oracle call per iteration, it requires $\mathcal{O}(1/\varepsilon)$ first-order oracle calls to compute an ε-solution

As a result, GD provides an upper complexity bound for problems of the form (CO). This provides a limit on how "hard" a problem instance can be.

Lower Complexity Bound

We can similarly discuss limitations on how "good" algorithms can be through the idea of a lower complexity bound.

→ Specify algorithm class and problem setting

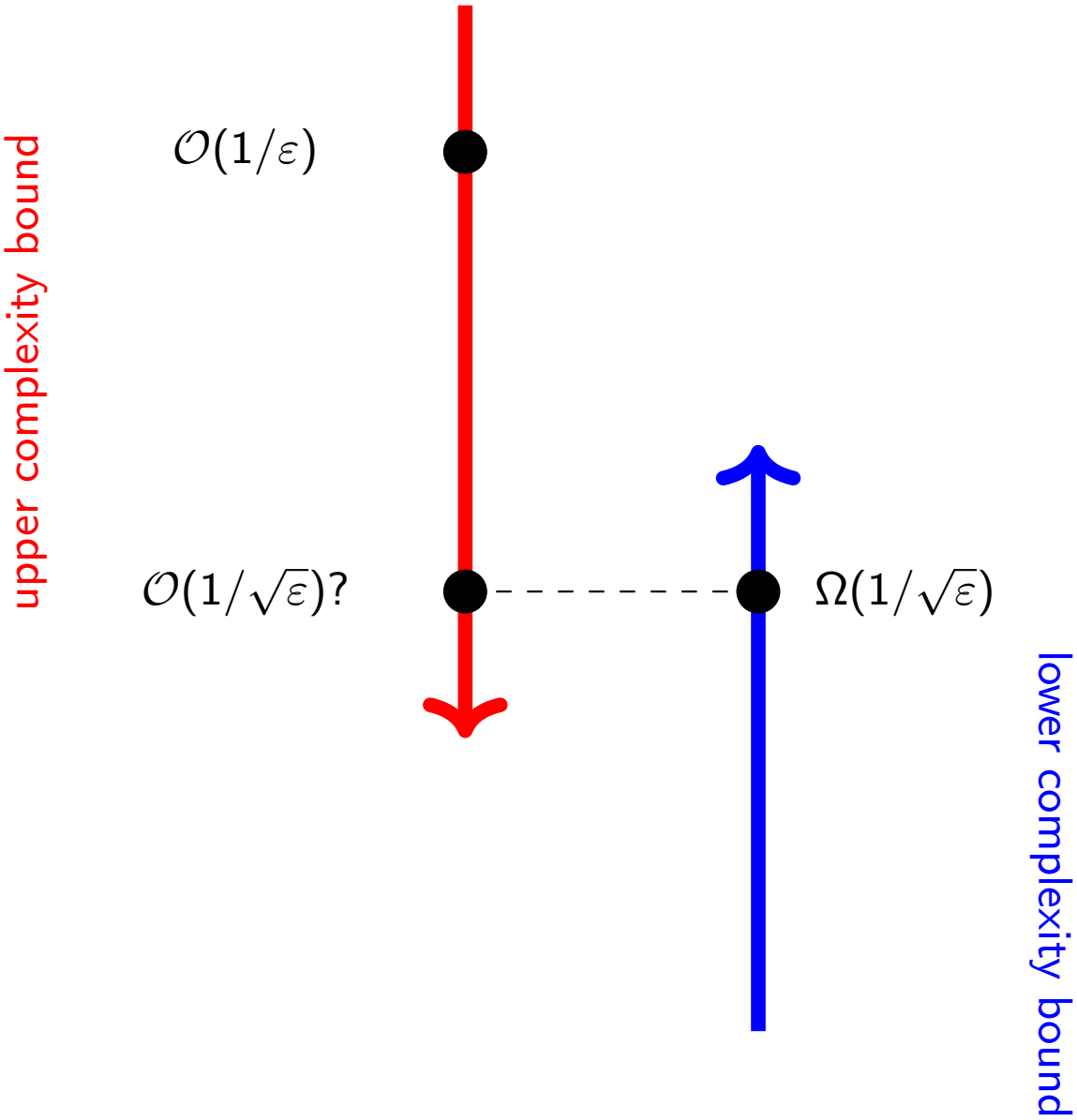
→ Given any algorithm, search for some "difficult" problem instance such that said algorithm struggles to solve with respect to certain oracle

Example: Smooth Convex Optimization (Nemirovski, 1983)

- for any method $\mathcal{M}$, there exists a quadratic function g such that if $\mathcal{M}$ is applied to minimize g at least $\Omega(1/\sqrt{\varepsilon})$ first-order oracle calls must be made
- since quadratics are a subset of smooth convex optimization, there cannot exist a method which solves all smooth convex problems in less than $\mathcal{O}(1/\sqrt{\varepsilon})$ gradient evaluations

The worst case problem instance above provides a lower complexity bound for smooth convex optimization with respect to the first-order oracle.

Complexity Bounds



Algorithm Nesterov's accelerated gradient descent (NAGD)

Start: Choose $x_0 \in X$. Set $y_0 := x_0$
for $k = 1, \ldots, N$ **do**

$$
\begin{aligned}
z_k &= (1-\gamma_k) y_{k-1} + \gamma_k x_{k-1}, \\
x_k &= \operatorname*{argmin}_{u \in X} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2, \\
y_k &= (1-\gamma_k) y_{k-1} + \gamma_k x_k.
\end{aligned}
$$

end for
Output y_N.

- minimizes linear approximation proximal problem
- reduces to gradient descent when $\gamma_k \equiv 1$
- computes ε-solution in only $\mathcal{O}(1/\sqrt{\varepsilon})$ iterations
- computes ε-solution in only $\mathcal{O}(1/\sqrt{\varepsilon})$ oracle calls under the first-order oracle
- optimal algorithm under this setting due to Nemirovski
- requires knowledge of L to set η_k appropriately

Research Directions in First-Order Optimization

We'll now take a further look at the following two settings:

1. Hölder smooth extensions of conditional gradient sliding methods
   - Literature Review
   - The UCGS algorithm
   - Numerical Experiments

2. Gradient sliding techniques applied to ADMM type algorithms
   - Literature Review
   - Gradient Sliding ADMM
   - Regularization techniques
   - Achieving the lower complexity bound
   - Numerical Experiments

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$ using first-order, deterministic algorithms.

NAGD is optimal in terms of gradient evaluations, but we can continue to make improvements by relaxing assumptions. Consider our projection assumption.

- certain sets can be as difficult to project to as the underlying problem is to solve
  - Convex Hull: $X = \text{conv}(v_1, \ldots, v_p)$
  - Standard Spectrahedron: $X = \{Y \in \mathbb{R}^{n \times n} : \text{tr}(Y) = 1, Y \succeq 0\}$
- NAGD is of no use when projection is more difficult than (CO)
- want to design algorithms that do not require difficult optimizations over X, i.e. projection free methods

Algorithm Conditional Gradient (CG) method

Start: Select parameters $\gamma_k \in (0, 1], \eta_k > 0$. Choose $y_0 \in X$.
for $k = 1, \ldots, N$ **do**

$$x_k = \operatorname*{argmin}_{x \in X} \langle \nabla f(y_{k-1}), x \rangle$$

$$y_k = (1 - \alpha_k) y_{k-1} + \alpha_k x_k$$

end for
Output y_N.

- run NAGD with a linear optimization rather than a projection by removing proximal term
  - when X is a convex hull, the linear optimization is a linear program
  - when X is the standard spectrahedron, the linear optimization is a smallest eigenvalue problem
- requires $\mathcal{O}(1/\varepsilon)$ number of iterations to obtain ε-solution
- more gradient evaluations and linear optimizations, but no projections at all
- multiple objective considerations
- optimal number of linear optimizations (Jaggi, 2013), but clearly suboptimal gradient evaluation complexity

We would like to get back the optimal gradient evaluation complexity.

Algorithm Conditional Gradient Sliding (CGS)

Start: Choose $x_0 \in X$. Set $y_0 := x_0$
for $k = 1, \ldots, N$ **do**

$$\begin{aligned} z_k &= (1-\gamma_k) y_{k-1} + \gamma_k x_{k-1}, \\ x_k &= CG(\nabla f(z_k), x_{k-1}, \eta_k, \varepsilon_k) \\ y_k &= (1-\gamma_k) y_{k-1} + \gamma_k x_k. \end{aligned}$$

end for
Output y_N.

- uses NAGD as a framework for achieving good gradient complexity
- solves x_k subproblem approximately with linear optimizations only
- fewer CG iterations implies fewer linear optimizations, but more gradient evaluations
- if parameters are chosen properly, CGS computes ε-solution in $\mathcal{O}(1/\sqrt{\varepsilon})$ gradient evaluations and $\mathcal{O}(1/\varepsilon)$ linear optimizations (Lan, 2016)
- optimal with respect to both oracles, but still requires L

We are now ready to relax our final assumption, the Lipschitz smoothness of f. We propose replacing it with the following assumption

Relaxed Assumption - Hölder Smooth

Assume that there exists a Hölder exponent $\nu \in [0, 1]$ and constant $M_\nu > 0$ such that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M_\nu}{1+\nu} ||x - y||^{1+\nu}, \ \forall x, y \in X.$$

This is a generalization of Lipschitz continuous gradient. In particular,

- any convex smooth f with Lipschitz continuous gradient M_1 is Hölder smooth with $\nu = 1$
- any convex nonsmooth Lipschitz continuous f with is Hölder smooth with $\nu = 0$
- any convex smooth f satisfying

$$||\nabla f(y) - \nabla f(x)|| \leq M_\nu ||y - x||^\nu, \ \forall x, y \in X$$

is Hölder smooth with $\nu \in (0, 1)$

The UCGS Algorithm

How can we address the loss of the Lipschitz smoothness? It appeared in two different aspects of algorithm analysis

- it was used in the analysis to bound $f(y_k) - f(z_k)$
- it was used explicitly in the parameter setting

To address these issues as simply as possible, we can search for a substitute constant L_k that satisfies an approximate Lipschitz condition locally

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} ||y_k - z_k||^2 + e_k.$$

through a backtracking linesearch process. We can then proceed with the rest of the CGS algorithm.

Algorithm Universal Conditional Gradient Sliding (UCGS)

Start: Choose $x_0 \in X$ and $\varepsilon > 0$. Set $y_0 = x_0$

for $k = 1, \ldots, N$ **do**

Decide $L_k > 0$ satisfying

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} ||y_k - z_k||^2 + \frac{\varepsilon}{2}\gamma_k$$

where

$$\begin{aligned} z_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1}, \\ x_k &= ACG(\nabla f(z_k), x_{k-1}, \eta_k, \varepsilon_k, \delta_k) \\ y_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_k. \end{aligned}$$

Terminate if

$$\max_{x \in X} f(y_k) - \ell_k(x) \leq \varepsilon$$

where

$$\ell_k(x) = \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} \left(f(z_i) + \langle \nabla f(z_i, x - z_i \rangle \right)$$

end for

Theorem

Suppose that we apply UCGS with parameters

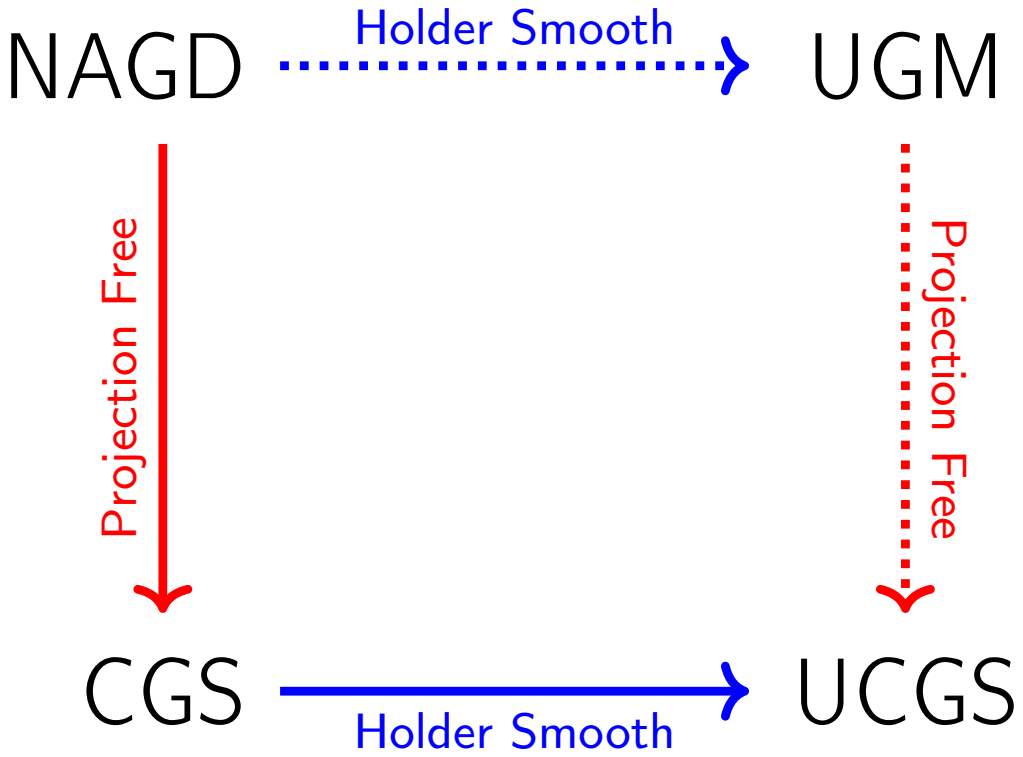
$$\beta_k = L_k\gamma_k,\ \eta_k = \frac{L_k\gamma_k D_X^2}{k}, \Gamma_k = \frac{L\gamma_k^2}{k},\ \text{and}\ \varepsilon_k = \frac{\sigma L_k\gamma_k^2 D_X^2}{2},$$

with γ_k satisfying $\Gamma_k = (1-\gamma_k)\Gamma_{k-1}$, and $\delta_t = \sigma\beta_k D_X^2/t$ in the ACG procedure, where $\sigma \geq 0$ is a parameter related to the accuracy of approximately solving linear objective optimization subproblems.Then UCGS terminates with an ε-solution after at most N_{iter} outer iterations where

$$N_{\text{iter}} := \left\lceil 16\left(\frac{(3+\sigma)^{\frac{1+\nu}{2}} M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{2}{1+3\nu}}\right\rceil$$

Consequently, the total number of gradient evaluations and linear objective optimizations performed by UCGS to find an ε-solution can be bounded by $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ and $\mathcal{O}((1/\varepsilon)^{\frac{4}{1+3\nu}})$ respectively.

The UCGS Algorithm



The UCGS Algorithm

Properties of UCGS

- Theoretical
  - obtains the optimal $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ gradient evaluations for an ε-solution (Nemirovski, 1983)
  - improves the best known number of linear optimizations required by the Universal CG variant (Nesterov, 2018) from $\mathcal{O}((1/\varepsilon)^{\frac{1}{\nu}})$ to $\mathcal{O}((1/\varepsilon)^{\frac{4}{1+3\nu}})$
  - achievable by novel choice of γ_k
- Practical
  - contains a stopping condition
  - allows linear optimization problems to be solved approximately by analyzing with ACG method
  - does not require knowledge of (ν, M_ν) for setting of parameters
  - adaptive to local geometry

Numerical Results

We consider the problem

$$\min_{x \in \text{conv}(V)} f(x) := ||Ax - b||_2$$

with $V = \{v_1, \ldots, v_p\} \subseteq \mathbb{R}^n$, $\text{conv}(V) := \{x \in \mathbb{R}^n : \exists \lambda \in \Delta_p \text{ s.t. } x = \sum_{j=1}^p \lambda_i v_i\}$, and $\Delta_p := \{\lambda \in \mathbb{R}^p : \sum_{i=1}^p \lambda_i = 1, \lambda_i \geq 0\}$ is the standard simplex.

		UCGS				Universal CG		
n	d	**GE**	**LO**	**Time**	**Error**	**Iter**	**Time**	**Error**
2500	0.2	66	2690	6.71	$9.945e-4$	572	13.42	$9.7086e1$
2500	0.4	60	3679	9.08	$9.976e-4$	524	18.17	$1.404e2$
2500	0.6	62	245	2.64	$9.678e-4$	146	5.29	$5.598e2$
2500	0.8	57	3176	8.45	$9.768e-4$	399	16.93	$2.400e2$
5000	0.2	71	286	7.13	$9.882e-4$	178	14.32	$6.037e2$
5000	0.4	42	52	4.89	$9.585e-4$	84	9.81	$1.689e3$
5000	0.6	68	4564	36.14	$9.727e-4$	483	72.40	$3.527e2$
5000	0.8	67	419	12.91	$9.815e-4$	161	25.94	$1.165e3$
10000	0.2	85	12269	150.51	$9.96e-4$	915	301.21	$2.449e2$
10000	0.4	69	12614	157.39	$9.916e-4$	636	315.27	$4.734e2$
10000	0.6	70	16063	205.87	$9.821e-4$	653	412.14	$5.423e2$
10000	0.8	69	12707	180.65	$9.862e-4$	473	361.73	$8.162e2$

Research Directions in First-Order Optimization

We'll be looking at the following two settings:

1. Hölder smooth extensions of conditional gradient sliding methods
   - Literature Review
   - The UCGS algorithm
   - Numerical Experiments

2. Gradient sliding techniques applied to ADMM type algorithms
   - Literature Review
   - Gradient Sliding ADMM
   - Regularization techniques
   - Achieving the lower complexity bound
   - Numerical Experiments

Affinely Constrained Optimization

Consider a new class of smooth convex optimization problems of the form

$$F^* := \min_{x \in X} f(x) + h(Kx) = \min_{x \in X, z \in Z} f(x) + h(z) \text{ s.t. } Kx = z. \qquad \text{(ACO)}$$

where

- f is L-smooth and X is easy to project to
- h is Lipschitz continuous but possibly nonsmooth
- the proximal mapping problem involving $h(\cdot)$ is easy, i.e.

$$\min_{w \in \mathbb{R}^m} h(w) + \frac{\rho}{2} \|w - z\|^2$$

  can be solved quickly

Such problems are common in machine learning and imaging science where h is some regularization/loss term, e.g. $h = ||\cdot||_1$.

Algorithm Alternating direction method of multipliers (ADMM)

Start: Start: Choose $x_0 \in X$, $y_0 \in \mathbb{R}^m$, and $z_0 := Kx_0$.
for $k = 1, \ldots, N$ **do**

$$\begin{aligned} x_k &= \operatorname*{argmin}_{u \in X} \mathcal{L}_\rho(u, y_{k-1}, z_{k-1}) \\ z_k &= \operatorname*{argmin}_{w \in \mathbb{R}^m} \mathcal{L}_\rho(x_k, y_{k-1}, w) \\ y_k &= y_{k-1} + \rho(Kx_k - z_k) \end{aligned}$$

end for
Output x_N.

- minimizes Augmented Lagrangian instead

$$\mathcal{L}_\rho(u, v, w) = f(u) + h(w) + \langle v, Ku - w \rangle + \frac{\rho}{2} \|Ku - w\|^2$$

- alternates updating primal and dual variables
- only requires $\mathcal{O}(||K|| / \varepsilon)$ iterations, but potential problematic x_k subproblem

Algorithm Linearized Alternating Direction Method of Multipliers (L-ADMM)

Start: Choose $x_0 \in X$. Set $y_0 := 0$ and $z_0 := Kx_0$.
for $k = 1, \ldots, N$ **do**

$$x_k = \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(x_k), u \rangle + K^\top (y_{k-1} + \theta_k (Kx_{k-1} - z_{k-1})), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2$$

$$z_k = \underset{w \in \mathbb{R}^m}{\operatorname{argmin}} - \langle y_{k-1}, w \rangle + h(w) + \frac{\tau_k}{2} \|Kx_k - w\|^2$$

$$y_k = y_{k-1} + \rho_k (Kx_k - z_k)$$

end for
Output x_N.

- linearizes the problematic subproblem and reduces it to a projection
- computes ε-solution in $\mathcal{O}((L + ||K||)/\varepsilon)$ oracle calls of $O(x, y) = (\nabla f(x), Kx, K^T y)$

Algorithm Accelerated Alternating Direction Method of Multipliers (A-ADMM)

Start: Choose $x_0 \in X$. Set $\overline{x}_0 := x_0$, $y_0 := 0$, and $z_0 := Kx_0$.
for $k = 1, \ldots, N$ **do**

$$
\begin{aligned}
\underline{x}_k &= (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1}, \\
x_k &= \operatorname*{argmin}_{u \in X} \langle \nabla f(\underline{x}_k), u \rangle + K^\top (y_{k-1} + \theta_k (Kx_{k-1} - z_{k-1})), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2 \\
z_k &= \operatorname*{argmin}_{w \in \mathbb{R}^m} - \langle y_{k-1}, w \rangle + h(w) + \frac{\tau_k}{2} \|Kx_k - w\|^2, \\
y_k &= y_{k-1} + \rho_k (Kx_k - z_k). \\
\overline{x}_k &= (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_k.
\end{aligned}
$$

end for
Output $\overline{x}_N$.

- acceleration motivated by NAGD
- was shown in that it computes ε-solution in $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| / \varepsilon)$ oracle calls of $O(x, y) = (\nabla f(x), Kx, K^T y)$

A few remarks

- it was shown that $\Omega(\sqrt{L/\varepsilon} + ||K|| /\varepsilon)$ is the lower complexity bound for problems of the form (ACO) using oracle $O(x, y) = (\nabla f(x), Kx, K^T y)$ (Ouyang and Xu, 2019)
- using oracle $O(x, y) = (\nabla f(x), Kx, K^T y)$ for analysis assumes that any algorithm makes 1 gradient evaluation for every operator evaluation
- although it makes sense to use $O(x, y)$ for analyzing A-ADMM, we may be able to achieve better bounds by separating the oracles into one for gradients and one for operators
- by splitting the oracles, we can focus on minimizing the calls to one or the other rather than both simultaneously

In particular, we may be able to achieve better results by specifically minimizing the calls to a single oracle. Similarly to CGS, perhaps there is an algorithm that minimizes the gradient evaluations while keeping the operator evaluations low.

Gradient Sliding ADMM

Algorithm Gradient sliding alternating direction method of multipliers (GS-ADMM)

Start: Choose $x_0 \in X$ and set $\overline{x}_0 := x_0$
for $k = 1, \ldots, N$ **do**

$$\begin{aligned} \underline{x}_k &= (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1} \\ (\tilde{x}_k, x_k, y_k, z_k) &= \text{ApproxGS}(\nabla f(\underline{x}_k), x_{k-1}, y_{k-1}, z_{k-1}) \\ \overline{x}_k &= (1-\gamma_k)\overline{x}_{k-1} + \gamma_k \tilde{x}_k \end{aligned}$$

end for
Output $\overline{x}_N$.

- uses NAGD as a framework for achieving good gradient complexity
- must deal with subproblem in a sophisticated way
- subproblem is approximately solved using ApproxGS

Gradient Sliding ADMM

The NAGD subproblem when applied to (ACO) takes the form

$$\min \phi(u, w) := \langle g, u \rangle + h(w) + \frac{\beta}{2}\|u - x\|^2 \text{ s.t. } Ku = w$$

for which we can use L-ADMM to solve.

Algorithm Approximate GS-ADMM subproblem (ApproxGS)

Start: Choose $x_0 \in X$. Set $y_0 := 0$ and $z_0 := Kx_0$.
for $t = 1, \ldots, T$ **do**

$$x_t = \underset{u \in X}{\operatorname{argmin}} \langle g, u \rangle + \frac{\beta}{2}||u - x||^2 + \langle K^T(y_{t-1} + \theta_t(Kx_{t-1} - z_{t-1})), u \rangle + \frac{\eta_t}{2}||u - x_{t-1}||^2$$

$$z_t = \underset{w \in \mathbb{R}^m}{\operatorname{argmin}} - \langle y_{t-1}, w \rangle + h(w) + \frac{\tau_t}{2}\|Kx_t - w\|^2$$

$$y_t = y_{t-1} + \rho_t(Kx_t - z_t)$$

end for

Gradient Sliding ADMM

Theorem

Suppose that the parameters in GS-ADMM are set to

$$T_k = \left\lceil \frac{N\,||K||\,D_Y}{LD_X} \right\rceil, \gamma_k = \frac{2}{k+1}, \beta_k = \frac{2L}{k}$$

and the parameters in the k-th call to the ApproxGs procedure are set to

$$\eta_t = \frac{\sigma N}{k}\,||K||^2 + \beta_k(t-1), \tau_t = \theta_t \equiv \frac{\sigma N}{k}, \rho_t \equiv \frac{\sigma k}{N}$$

where $\sigma = D_Y/||K||\,D_X$. Then the number of gradient evaluations of ∇f operator evaluations are bounded by

$$N_{\nabla f} := \sqrt{\frac{5LD_X^2}{\varepsilon}} \text{ and } N_K := \frac{5\|K\|D_X D_Y}{\varepsilon} + \sqrt{\frac{5LD_X^2}{\varepsilon}}$$

respectively.

- under appropriate parameter settings, GS-ADMM computes ε-solution in only $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluations and $\mathcal{O}(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$ operator evaluations
- clear improvement from A-ADMM

Regularization Techniques

Unfortunately, in order to complete the convergence analysis for GS-ADMM, we are required to set

$$\tau_{t,k} = \frac{\sigma N}{k}, \theta_{t,k} = \frac{\sigma N}{k}, \rho_{t,k} = \frac{\sigma k}{N}$$

and other parameters similarly where N is the total number of iterations to be computed. Such reliance on N is not preferred. Lan proposed a technique called dual regularization to alleviate such issues (Lan, 2020). We can adapt this idea to our sliding algorithms.

Key Idea: Add additional terms in the optimization subproblems that do not change the computational efficiency of solving them. Then, leverage these extra terms in the analysis to provide flexibility in the parameter selection process.

To remove N, we propose replacing ApproxGS in GS-ADMM with the following algorithm.

Algorithm Approximate regularized GS-ADMM subproblem (ApproxGSR)

Start: Choose $x_0 \in X$. Set $y_0 := 0$ and $z_0 := Kx_0$.
for $t = 1, \ldots, T$ **do**

$$x_t = \underset{u \in X}{\operatorname{argmin}} \langle g, u \rangle + \tfrac{\beta}{2}\|u - x\|^2 + \langle K^\top(y_{t-1} + \theta_t(Kx_{t-1} - z_{t-1})) - \alpha_t(Kx_{t-1} - Kx), u \rangle + \tfrac{\eta_t}{2}\|u - x_{t-1}\|^2 + \tfrac{\xi_t}{2}\|u - x\|^2$$

$$z_t = \underset{w \in Z}{\operatorname{argmin}}\, h(w) - \langle y_{t-1}, w \rangle + \tfrac{\tau_t}{2}\|w - Kx_t\|^2 + \tfrac{\zeta_t}{2}\|w - z_0\|^2$$

$$y_t = \tfrac{1}{1+\rho_t\delta_t} y_{t-1} + \tfrac{\rho_t\delta_t}{1+\rho_t\delta_t} y_0 + \tfrac{\rho_t}{1+\rho_t\delta_t}(Kx_t - z_t)$$

end for
Output x_N.

- adds regularization terms
- does not change the difficulty of solving each subproblem

Theorem

Suppose that the parameters in Regularized GS-ADMM are set to

$$T_k = \lceil \mu k \rceil, \beta_k = \frac{2L}{k}, \gamma_k = \frac{2}{k+1}$$

and the parameters in the k-th call to the ApproxGSR procedure are set to

$$\tau_t = \theta_t \equiv \sigma, \rho_t \equiv \rho, \zeta_t = \alpha_t = \frac{\sigma}{t}, \xi_t = \frac{\sigma}{t}\|K\|^2, \eta_t = \beta_k\left(\frac{t-1}{2}\right) + \sigma\|K\|^2, \text{ and } \delta_t = \frac{1}{\rho t}$$

where

$$\sigma = \chi\frac{\lceil \mu k \rceil}{\mu k}, \rho = \chi\frac{\mu k}{\lceil \mu k \rceil}, \chi = \frac{D_Y}{\|K\| D_X}, \text{ and } \mu = \frac{\|K\| D_Y}{L D_X}.$$

Then the number of gradient evaluations of ∇f operator evaluations are bounded by

$$N_{\nabla f} := \sqrt{\frac{7LD_X^2}{\varepsilon}} \text{ and } N_K := \frac{7\|K\| D_X D_Y}{\varepsilon} + \sqrt{\frac{7LD_X^2}{\varepsilon}}$$

respectively.

- corresponding (R-)GS-ADMM algorithm can achieve the same complexities
- new parameter setting does not require N

Achieving the Lower Complexity Bound

The idea of sliding has been very crucial thus far. We can summarize this fundamental idea for problems with multiple oracles:

1. Choose outer iteration algorithm that computes an ε-solution with optimal (outer) oracle calls
2. Choose inner iteration algorithm that resolves potential subproblems in the outer iterations without additional outer oracle calls and with good complexity of inner oracle calls
3. Adjust the analysis/parameter setting such that both objectives are achieved simultaneously

For the above GS-ADMM method, NAGD and L-ADMM satisfied the conditions in 1-2 and were chosen as a result. Step 3 required some modifications and novel analysis techniques. What if we wanted to reduce the calls to the operator oracle?

Achieving the Lower Complexity Bound

Algorithm Operator sliding alternating direction method of multipliers (OS-ADMM)

Start: Choose $u_0 \in X$. Set $\overline{u}_0 := u_0$, $\tilde{u}_0 := u_0$, $v_0 := 0$ and $w_0 := Ku_0$.
for $t = 1, \ldots, T$ **do**

$$
\begin{aligned}
l_t &= K^\top (v_{t-1} + \theta_k (K\tilde{u}_{t-1} - w_{t-1})) \\
(\tilde{u}_t, u_t) &= \text{ApproxOS}(l_t, u_{t-1}, \overline{u}_{t-1}, \eta_t, N_t) \\
\overline{u}_t &= (1-\lambda_t)\overline{u}_{t-1} + \lambda \tilde{u}_t \\
w_t &= \operatorname*{argmin}_{w \in \mathbb{R}^m} -\langle v_{k-1}, w\rangle + h(w) + \tfrac{\tau_k}{2}\|K\tilde{u}_t - w\|^2 \\
v_t &= v_{t-1} + \rho_k (K\tilde{u}_t - w_t)
\end{aligned}
$$

end for
Output $\overline{x}_N$.

- uses ADMM as a framework for achieving good operator complexity
- must deal with subproblem in a sophisticated way
- subproblem is approximately solved using a variant of NAGD called ApproxOS

Achieving the Lower Complexity Bound

The ADMM subproblem when applied to (ACO) takes the form

$$\min_{u \in X} \psi(u) := f(u) + \langle l, u \rangle + \tfrac{\eta}{2}\|u - x\|^2.$$

for which we can use a modified NAGD method to solve.

Algorithm Approximate OS-ADMM subproblem (ApproxOS)

Start: Choose $x_0 \in X$. Set $\tilde{x}_0 := x_0$

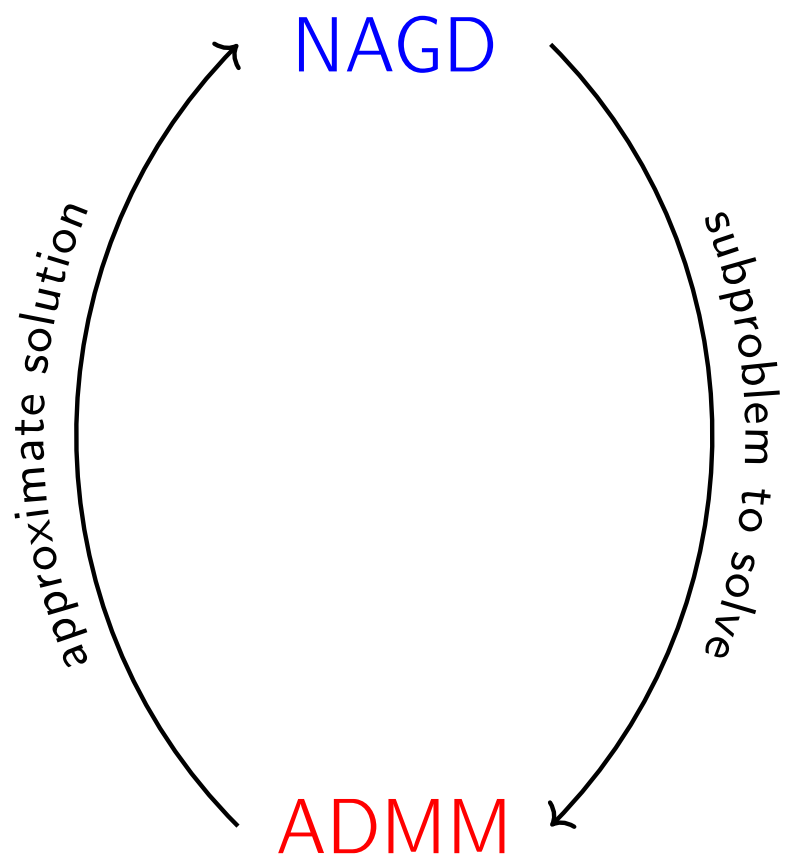
for $k = 1, \ldots, N$ **do**

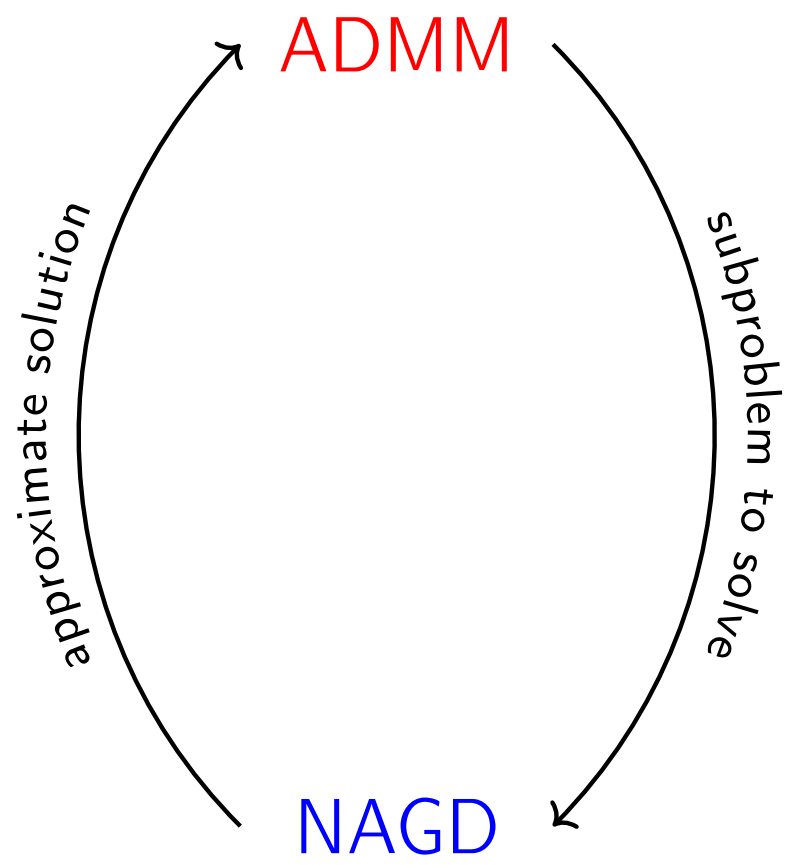
Compute

$$\begin{aligned}
\underline{x}_k &= (1-\lambda)\tilde{x}_0 + \lambda(1-\gamma_k)\tilde{x}_{k-1} + \lambda\gamma_k x_{k-1}, \\
x_k &= \operatorname*{argmin}_{u \in X} \langle \nabla f(\underline{x}_k) + l, u \rangle + \tfrac{\eta}{2}\|u - x\|^2 + \tfrac{\beta_k}{2}\|u - x_{k-1}\|^2, \\
\tilde{x}_k &= (1-\gamma_k)\tilde{x}_{k-1} + \gamma_k x_k.
\end{aligned}$$

end for

Achieving the Lower Complexity Bound





Achieving the Lower Complexity Bound

Theorem

Suppose that the parameters in OS-ADMM are set to

$$N_t \equiv \left\lceil \sqrt{\frac{LD_X}{T\|K\|D_Y}} \right\rceil, \lambda_t = \frac{2}{t+1}, \ \eta_t = \frac{\sigma T}{t}\|K\|^2, \ \tau_t = \theta_t = \frac{\sigma T}{t}, \ \rho_t = \frac{\sigma t}{T},$$

and that the parameters in the k-th call to the ApproxOS procedure are set to

$$\gamma_k = \frac{2}{k+1}, \ \beta_k = \frac{2L}{kt}$$

where $\sigma = \frac{D_Y}{\|K\|D_X}$. Then the number of gradient evaluations and operator evaluations are bounded by

$$T_K := \frac{9\|K\|D_X D_Y}{\varepsilon} \text{ and } T_{\nabla f} := \sqrt{\frac{9LD_X^2}{\varepsilon}} + \frac{9\|K\|D_X D_Y}{\varepsilon}$$

respectively.

- computes ε-solution in only $\mathcal{O}(||K||/\varepsilon)$ operator evaluations and $\mathcal{O}(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$ gradient evaluations
- clear improvement from A-ADMM

Achieving the Lower Complexity Bound

There is an apparent tradeoff in choosing between GS-ADMM and OS-ADMM. Consider the quantity $\pi := \frac{||K||}{\sqrt{\varepsilon L}}$.
If $\pi \geq 1$,

$$N_K = \sqrt{\frac{5L}{\varepsilon}} + \frac{5\,||K||}{\varepsilon} \leq \frac{||K||}{\sqrt{\varepsilon L}} \cdot \sqrt{\frac{5L}{\varepsilon}} + \frac{5\,||K||}{\varepsilon} = \frac{(5+\sqrt{5})\,||K||}{\varepsilon}.$$

One the other hand, if $\pi \leq 1$

$$T_{\nabla f} = \sqrt{\frac{9L}{\varepsilon}} + \frac{9\,||K||}{\varepsilon} \leq \sqrt{\frac{9L}{\varepsilon}} + \frac{\sqrt{\varepsilon L}}{||K||} \cdot \frac{9\,||K||}{\varepsilon} = 12\sqrt{\frac{L}{\varepsilon}}.$$

Thus, either GS-ADMM or OS-ADMM will have both optimal gradient and operator oracle complexities.

Achieving the Lower Complexity Bound

Algorithm Sliding alternating direction method of multipliers (S-ADMM)

Compute $\pi := ||K|| / \sqrt{\varepsilon L}$
if $\pi \geq 1$ **then**
    Apply GS-ADMM.
else
    Apply OS-ADMM.
end if

- chooses to slide on best oracle
- computes ε-solution in only $\mathcal{O}(||K|| / \varepsilon)$ operator evaluations and $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluations
- improves both oracle complexities

Summary of Contributions

GS-ADMM

- reduces gradient evaluations required as compared to A-ADMM
- crucial step was the modification of ADMM to facilitate analysis

(R-)GS-ADMM

- demonstrates that regularization modifications can also be applied to sliding ADMM methods by removing the need for N in the parameter settings of GS-ADMM
- crucial step was the adaptation of the previous ApproxGS analysis
- we believe similar modifications can be made to OS-ADMM using the same adaptations

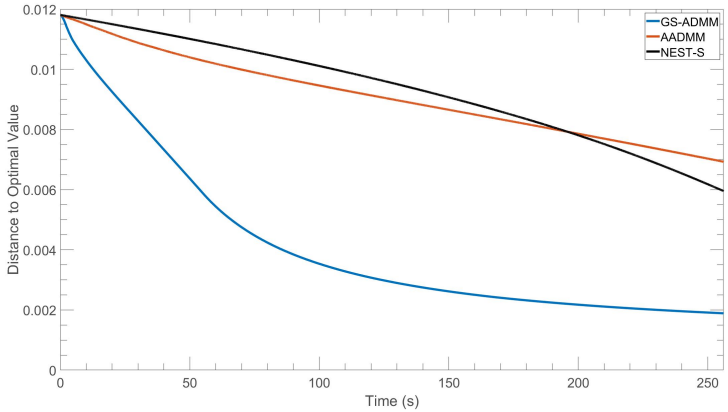
OS-ADMM

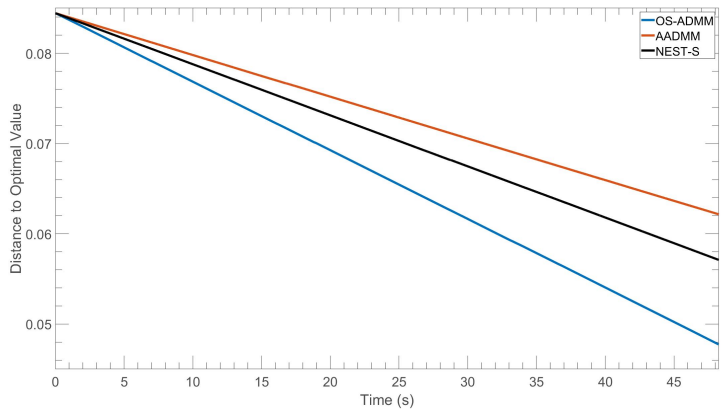
- reduces operator evaluations required as compared to A-ADMM
- completes the lower complexity bound picture when combined with GS-ADMM to obtain S-ADMM
- crucial step was the modification of NAGD to facilitate analysis

Numerical Experiments

Problem setting:

- motivated from worst-case instance in Ouyang and Xu, 2019.
- $f(x) = \frac{1}{2}x^T Qx - q^T x, h(x) = ||Kx - b||_2$
- one sparse, one dense matrix for each experiment





- GS-ADMM better when gradient evaluations are expensive and A-ADMM preforms as worse as possible
- OS-ADMM better when operator evaluations are expensive and A-ADMM preforms as worse as possible
- very contrived examples
- better to do more realistic experiments

Numerical Experiments

Problem setting:

- popular image reconstruction problem in imaging science
- $f(x) = \frac{1}{2}\,||Ax - f||^2$ $h(x) = \lambda\,||Dx||_{2,1}$
- we let S-ADMM determine sliding

		S-ADMM				A-ADMM		
$\sqrt{n}$	$\log\lambda$	**GE**	**OE**	**Obj Val**	**Rel Err**	**Iter**	**Obj Val**	**Rel Err**
100	0	500	125250	$1.00e+05$	11.51	18742	$1.03e+05$	11.47
150	0	500	125250	$1.86e+05$	9.52	8073	$2.17e+05$	9.23
200	0	500	125250	$2.86e+05$	8.19	4663	$3.76e+05$	7.28
100	-1	500	125250	$1.33e+04$	7.69	15135	$1.36e+04$	7.68
150	-1	500	125250	$2.47e+04$	6.03	3976	$2.88e+04$	6.96
200	-1	500	125250	$3.72e+04$	4.95	2608	$4.66e+04$	6.35
100	-2	500	125250	$1.65e+03$	9.52	14613	$1.42e+03$	7.48
150	-2	500	125250	$3.18e+03$	8.08	4228	$3.14e+03$	7.61
200	-2	500	125250	$4.97e+03$	7.10	2558	$7.07e+03$	12.53
100	-3	500	125250	$5.78e+02$	49.17	14351	$1.44e+02$	7.32
150	-3	500	125250	$1.30e+03$	49.43	4900	$4.36e+02$	13.16
200	-3	500	125250	$2.31e+03$	49.54	2772	$1.82e+03$	39.62

- S-ADMM preferable as n gets large and for common problems in imaging science

Thanks for your attention. The discussed works and their publication information can be found at http://tsquire.people.clemson.edu/ or by email tsquire[at]clemson[dot]edu