Everything about ADMM

Hailiang ZHAO @ ZJU-CS
http://hliangzhao.me

November 8, 2022

Outline

Outline

μ-Strongly Convex Function

A function $f : C \to \mathbb{R}$ is μ-strongly convex if $C \subseteq \mathbb{R}^n$ is a convex set and there exists a constant $\mu > 0$ such that

$$\begin{aligned} f\Big(\alpha x + (1-\alpha)y\Big) \le {} & \alpha f(x) + (1-\alpha)f(y) \\ & - \frac{\mu\alpha(1-\alpha)}{2}\|y - x\|^2, \forall x, y \in C. \end{aligned} \tag{1}$$

Compared with convex, the inequality holds even if a norm term is subtracted. Thus we can understand how strong the convexity is.

Note that every norm appears in this slide is Euclidean norm. Besides, please carefully distinguish the math symbol used is a scalar or a vector according to context.

L-Smooth Function

A function $f : C \to \mathbb{R}$ is L-smooth (a.k.a has Lipschitz continuous gradients) if there exists $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|, \forall x, y \in C. \tag{2}$$

Informally, smooth is an alias of *continuously differentiable.* L-smooth function's gradient is controlled by a constant L from the upper bound.

An L-smooth (continuously differentiable) function is not necessarily to be convex, and vice versa.

L-Smooth Function (Cont'd)

For L-smooth function f we have

$$|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \leq \frac{L}{2}\|y - x\|^2, \forall x, y \in C. \quad (3)$$

The result can be obtained with Taylor expansion. It indicates that the gap between smooth function f and its linear approximation is *upper bounded* by the change of x. The result indicates that **smooth functions are good to be linearized**. Their linearization is easy to compute, and the deviation is not too large.

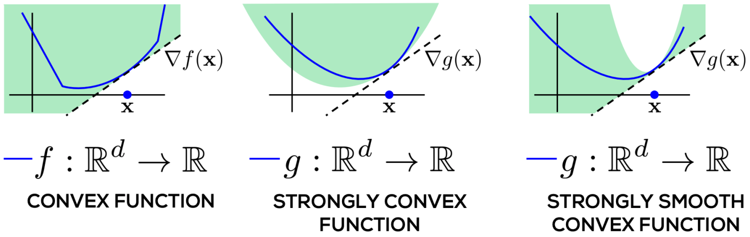
If f is both L-smooth and convex, we have: $\forall x, y \in C$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2. \quad (4)$$

It indicates that, when f is differentiable and convex, we can use the change of the gradient to bound the linearization gap *from the lower side*.

Convex, Strongly Convex, and Smooth

A convex function is lower bounded by its own tangent at all points. Strongly convex and smooth functions are, respectively, lower and upper bounded in the rate at which they may grow, by quadratic functions and cannot, again respectively, grow too slowly or too fast.



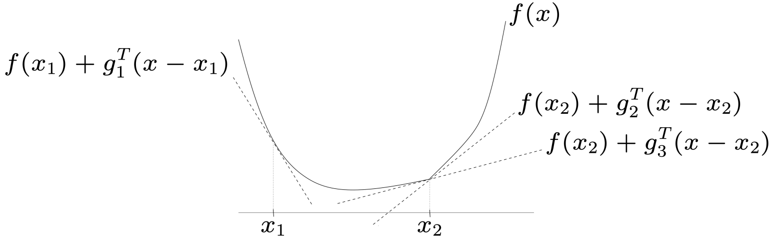
In each figure, the shaded area describes regions the function curve is permitted to pass through. We can find that L-smooth functions should not be too smooth because its change of gradient is controlled from the upper bound :-).

Subgradient of Convex Functions

A vector g is a subgradient of a convex function f at $x \in C$ if

$$f(y) \geq f(x) + \langle g, y - x \rangle, \forall y \in C. \tag{5}$$

$\partial f(x)$ is the collection of all the g's at x. We can find that, the linearization of f at x with the subgradient g is always smaller than f (the straight line is below the curve f).


$f(x)$
$f(x_1) + g_1^T(x - x_1)$
$f(x_2) + g_2^T(x - x_2)$
$f(x_2) + g_3^T(x - x_2)$
x_1
x_2

In this figure, g_1 is the subgradient at x_1. g_2 and g_3 are the subgradients at x_2.

Subgradient of Convex Functions (Cont'd)

For convex f, its subgradient exists at every interior point of its domain.

We only care about the subgradient of *convex functions*. It's useful especially f is continuous but not differentiable. If convex f is differentiable, then $\partial f(x)$ is a singleton and the only element in it is equal to $\nabla f(x)$. (For example, in the figure above, $g_1 = \nabla f(x_1)$.)

Actually, subgradient is an extension of gradient to non-differentiable functions.

Subgradient of μ-Strongly Convex Functions

For μ-strongly convex f we have

$$f(y) \geq f(x) + \langle g, y - x \rangle + \frac{\mu}{2}\|y - x\|^2, \forall g \in \partial f(x). \quad (6)$$

It indicates that, the gap between f and its linearization with g is *at least* the norm term $\frac{\mu}{2}\|y - x\|^2$. We can imagine how strong the convexity is. We further have

$$f(x) - f(x^*) \geq \frac{\mu}{2}\|x - x^*\|^2, \forall x \in C. \quad (7)$$

The result is obvious because $0 \in \partial f(x^*)$ (the critical point).

It's interesting to compare (6) with (4). (4) holds when f is smooth and convex while (6) holds when f is strongly convex. The minimum gap between f and its linearization is thus controlled by different terms.

Gradient of μ-Strongly Convex Functions

If the μ-strongly convex f is differentiable, we have

$$f(x) - f(x^*) \leq \frac{1}{2\mu} \|\nabla f(x)\|^2, \forall x \in C. \tag{8}$$

It indicates that the gap between any value of f and the minimum is *upper bounded* by the gradient of f at that point.

We also have

$$\frac{\langle \nabla f(x) - \nabla f(y), x - y \rangle}{\|x - y\|^2} \geq \mu, \forall x, y \in C. \tag{9}$$

It indicates that, to be μ-strongly convex, at any point, the change of gradient is *at least* μ.

Monotonicity of Subgradient

If a single-valued mapping f satisfies

$$\langle x-y, f(x)-f(y)\rangle \geq 0, \forall x, y \in \mathbf{dom} f,$$

then we say f is a monotone function. The subgradient of a convex function f is monotone, i.e.,

$$\langle x-y, g_x-g_y\rangle \geq 0, \forall x, y \in C, \tag{10}$$

where $g_x \in \partial f(x), g_y \in \partial f(y)$. This inequality will be frequently used when analyzing the convergence of ADMMs. It will be useful when building the structure

$$\|obj(x^{k+1}) - obj(x^*)\| \leq \alpha\|obj(x^k) - obj(x^*)\|.$$

When f is μ-strongly convex, we have

$$\langle x-y, g_x-g_y\rangle \geq \mu\|x-y\|^2, \forall x, y \in C, \tag{11}$$

where $g_x \in \partial f(x), g_y \in \partial f(y)$. This result is similar to (9) for gradient of μ-strongly convex functions.

Bregman Distance of Convex Functions

If a convex function ϕ is differentiable, then the associated Bregman distance is

$$D_\phi(y, x) := \phi(y) - \phi(x) - \langle \nabla\phi(x), y - x\rangle. \tag{12}$$

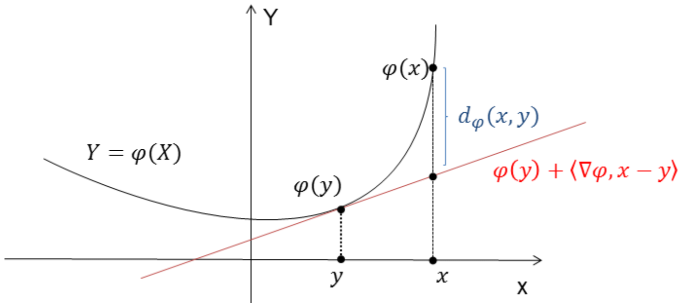
If a convex function ϕ is not differentiable, then

$$D_\phi^v(y, x) := \phi(y) - \phi(x) - \langle v, y - x\rangle, \tag{13}$$

where $v \in \partial f(x)$.

Bregman Distance of Convex Functions (Cont'd)

The Bregman distance is actually the gap between a convex function f and its linearization:



Thus, when ϕ is μ-strongly convex, similar to (6), we have

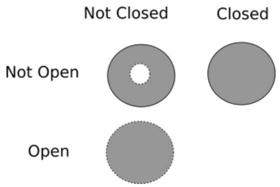
$$D_\phi(y,x) \geq \frac{\mu}{2}\|y-x\|^2. \tag{14}$$

Closed and Bounded

f is a closed function if its epigraph

$$\textbf{epi}f := \{(x, t) \mid t \geq f(x)\} \tag{15}$$

is a closed set. Note that a set $\mathcal{S}$ is called closed if $\forall x \in \mathcal{S}$, its limit $\lim_{y \to x} y$ is also in $\mathcal{S}$. We care about closed functions because the limit operator is well defined on them.



A set $\mathcal{S}$ is called *compact* if it is both *bounded* and *closed.*

Conjugate Function and its Properties

The conjugate function of f is

$$f^*(u) := \sup_{z \in \mathbf{dom} f} \big(\langle z, u\rangle - f(z)\big). \tag{16}$$

f^*'s domain is $\{u \mid f^*(u) < \infty\}$.

In the following, we show briefly that f^* is always a convex function whatever f is or not. *Note that*

$$\forall z \in \boldsymbol{dom} f, h(u, z) := \langle z, u\rangle - f(z)$$

is a convex function of u, thus $\boldsymbol{epi} h(\cdot, z)$ *is a convex set. Besides,* $f^*(u) := \sup_{z \in \boldsymbol{dom} f} h(u, z)$ *satisfies that*

$$\boldsymbol{epi} f^* = \cap_{z \in \boldsymbol{dom} f} \boldsymbol{epi} h(\cdot, z), \tag{17}$$

which is also a convex set. Therefore, $f^*(u)$ *is convex.*

Conjugate Function and its Properties (Cont'd)

By definition we also have

$$f(x) + f^*(y) \geq \langle x, y \rangle, \forall x, y. \tag{18}$$

It is called the *Fenchel-Young Inequality*. Thus we have the relation between the biconjugate and the original function ——

$$\begin{aligned} f^{**}(x) &:= \sup_{y \in \mathbf{dom} f^*} \big(\langle x, y \rangle - f^*(y) \big) \qquad (19) \\ &\leq \sup_{y \in \mathbf{dom} f^*} \big(\langle x, y \rangle + f(x) - \langle x, y \rangle \big) = f(x). \qquad (20) \end{aligned}$$

When f is closed and convex, we have $f^{**}(x) = f(x)$. The proof can be found at here.

More properties:

- If f is closed and convex, then $y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y)$.
- If f is L-smooth on $\mathbf{dom} f$, then f^* is L^{-1}-strongly convex on $\mathbf{dom} f^*$. Vice Versa.

Understand Conjugate Physically

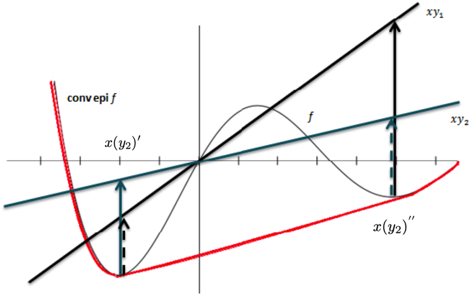
From the definition of conjugate function, we want to know, at any legal point x, the maximum of $y^T x - f(x)$. Suppose $x(y)$ is the extremum w.r.t. y, then we must have

$$\nabla_x y^T x = \nabla_x f(x),$$

i.e., $y = f(x(y))$. It means the slope of the line $y^T x$ is equal to the slope of the curve f at point $x(y)$. Thus, we can move the crossing-zero straight line $y^T x$ up and down until we find several points at which the straight line tangents to f.

Understand Conjugate Physically (Cont'd)

From the figure below, we can find that, when $y = y_2$, for the two extremes $x(y_2)'$ and $x(y_2)''$, $x(y_2)'$ is what we want because the gap between $x(y_2)'^T y$ and $f(x(y_2)')$ is larger.



At any y, we can understand the gap in this way. Then we can find that, the conjugate function encodes the convex hull of **epi**f in terms of its supporting hyperplanes. Actually, this also explains the weak duality (the red curve is always under f), which will be introduced later.

Lagrangian Function

The vanilla ADMM is suitable for the problems with a linear equality constraint. They can be written in the form of:

$$\mathcal{P}_1 : \min_{x \in \mathbb{R}^n} f(x), \quad s.t. \quad Ax = b, \quad g(x) \leq 0, \tag{21}$$

where $A \in \mathbb{R}^{m \times n}$ and $g(x) := [g_1(x), ..., g_p(x)]^T$. We divide $g(x) \leq 0$ into $g_{i_1 \in \mathcal{I}_1}(x) \leq 0$ and $g_{i_2 \in \mathcal{I}_2}(x) \leq 0$, where $\mathcal{I}_1 \subseteq [p]$ and $\mathcal{I}_2 \subseteq [p]$ are the set of indices of linear and nonlinear inequality constraints. The feasible set is denoted by C.

The Lagrangian function is

$$L(x, u, v) = f(x) + \langle u, Ax - b \rangle + \langle v, g(x) \rangle, \tag{22}$$

where $v \in \mathbb{R}^p \geq 0$ and u are called the Lagrangian multipliers.

The Dual Function and The Dual Problem

Note f is a function of x. Correspondingly, the dual function

$$d(u, v) := \min_{x \in C} L(x, u, v) \tag{23}$$

is a function of the Lagrangian multipliers (u, v), where $\mathbf{dom} d := \{(u, v) \mid v \geq 0, d(u, v) > -\infty\}$. Obviously we have

$$(Ax^\star - b, g(x^\star)) \in \partial d(u, v), \tag{24}$$

where $x^\star \in \operatorname{argmin}_{x \in C} L(x, u, v)$.

The dual function d is always concave whatever f is convex or not. The reason is that, $L(x, u, v)$ is affine in u and v, and by definition $d(u, v)$ is the pointwise infimum of a set of affine functions.

The Dual Problem

We define the dual problem as:

$$\max_{(u,v)\in\mathbf{dom}\, d} d(u, v). \tag{25}$$

The primal problem and its dual problem are nicely connected by the Lagrangian function. Specifically, we can find that the primal and the dual optimal values, f^* and d^*, always satisfy

$$f^* \geq d^*. \tag{26}$$

This is called the weak duality.

Proof Sketch of the Weak Duality

In the following, we give a simple proof of the weak duality.

We consider a problem with inequality constraints:

$$\begin{aligned} \min_x \quad & f(x) \\ s.t. \quad & g_i(x) \leq 0, i \in [m]. \end{aligned}$$

Our target is to find the optimal (maximal) lower bound of f. Firstly, for any $v \in \mathbb{R}$, how to make it be a lower bound of f? Actually, if the following equation system on x has no solution, then we can say v is a lower bound of f:

$$\begin{cases} f(x) < v \\ g_i(x) \leq 0, i \in [m] \end{cases} \tag{27}$$

Proof Sketch of the Weak Duality (Cont'd)

If (27) *has a solution, then, for any* $\lambda \geq 0$, *the following equation of* x

$$f(x) + \sum_{i \in [m]} \lambda_i g_i(x) < v \tag{28}$$

has a solution. According to the equivalence of contrapositives, we have: For any $\lambda \geq 0$, *if* (28) *has no solution, then* (27) *has no solution. On the other hand,* (28) *has no solution for any given* $\lambda \geq 0$ *iff the following inequality holds for any given* $\lambda \geq 0$*:*

$$\min_x f(x) + \sum_{i \in [m]} \lambda_i g_i(x) \geq v. \tag{29}$$

Proof Sketch of the Weak Duality (Cont'd)

Combing the above results, we have: If (29) *holds for any given* $\lambda \geq 0$*, then* v *is a lower bound of* f*. Note that we want to find the maximal lower bound of* f*, i.e.*

$$v^* = \max_{\lambda \geq 0} \underbrace{\left(\min_x \left[\overbrace{f(x) + \sum_{i \in [m]} \lambda_i g_i(x)}^{L(x,\lambda)} \right] \right)}_{d(\lambda) := \min_x L(x,\lambda)}. \tag{30}$$

As a infimum of f*, we have* $v^* = \min_{x^*} f(x^*)$*. Therefore, we have:*

$$\min_{x^*} f(x^*) \geq \max_{\lambda^*} d(\lambda^*). \tag{31}$$

■

Strong Duality

Strong duality holds if and only if the duality gap is equal to 0, i.e.,

$$f^* = d^*. \tag{32}$$

For the primal $\mathcal{P}_1$, if there exists an $x_0 \in C$ such that

$$Ax_0 = b, \quad \{g_{i_1}(x_0) \leq 0\}_{i_1 \in \mathcal{I}_1}, \quad \{g_{i_2}(x_0) < 0\}_{i_2 \in \mathcal{I}_2}, \tag{33}$$

then we say the Slater's condition holds.

When the Slater's condition holds, the strong duality holds.

KKT Point and KKT Condition

(x, u, v) is called a Karush-Kuhn-Tucker (KKT) point of $\mathcal{P}_1$ if

1. *Stationary*: $0 \in \partial f(x) + A^T u + \sum_{i=1}^{p} v_i \partial g_i(x)$.
2. *Primal feasibility*: $Ax = b, g_i(x) \leq 0, \forall i \in [p]$.
3. *Complementary slackness*: $v_i g_i(x) = 0, \forall i \in [p]$.
4. *Dual feasibility*: $v_i \geq 0, \forall i \in [p]$.

The above conditions are called the KKT condition of $\mathcal{P}_1$.

They are the optimality condition of $\mathcal{P}_1$ when

1. $f(x)$ and $g_i(x), i \in [p]$ are convex, and
2. $\mathcal{P}_1$ satisfies the Slater's condition.

KKT Point and KKT Condition (Cont'd)

When $f(x)$ and $g_i(x), i \in [p]$ in $\mathcal{P}_1$ are all convex, then

1. every KKT point (x^*, u^*, v^*) is a saddle point of the Lagrangian function, i.e.,

$$L(x^*, u, v) \leq L(x^*, u^*, v^*) \leq L(x, u^*, v^*), \quad \forall x \in C, (u, v) \in \mathbf{dom}\, d, \tag{34}$$

   and

2. (x^*, u^*, v^*) is a pair of the primal and the dual solutions with zero dual gap *iff* it satisfies the KKT condition.

Danskin's Theorem

Danskin's Theorem

Let $\mathcal{Z}$ be a compact subset of $\mathbb{R}^m$, and let $\phi : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ be continuous and such that $\phi(\cdot, z) : \mathbb{R}^n \to \mathbb{R}$ is convex for each $z \in \mathcal{Z}$. Define $f : \mathbb{R}^n \to \mathbb{R}$ by $f(x) = \max_{z\in\mathcal{Z}} \phi(x, z)$ and

$$\mathcal{Z}(x) = \left\{ \bar{z} \mid \phi(x, \bar{z}) = \max_{z\in\mathcal{Z}} \phi(x, z) \right\}. \tag{35}$$

$f(x)$ is differentiable at x if $\mathcal{Z}(x)$ is a singleton. If $\phi(\cdot, z)$ is differentiable for all $z \in \mathcal{Z}$ and $\nabla_x \phi(x, \cdot)$ is continuous on $\mathcal{Z}$ for each x, then $\forall x \in \mathbb{R}^n$,

$$\partial f(x) = conv\left\{ \nabla_x \phi(x, z) \mid z \in \mathcal{Z}(x) \right\}. \tag{36}$$

Danskin's Theorem will be used when we calculate the (sub)gradient of the dual problem —— $d(\lambda) = \min_x L(\lambda, x)$.

Hoffman's Bound

Consider the non-empty polyhedron

$$\mathcal{X} = \{x \mid Ax = a, Bx \leq b\}. \tag{37}$$

Then there exists a constant θ, depending only on $[A^T, B^T]^T$, such that for any x we have

$$\mathrm{dist}(x, \mathcal{X})^2 \leq \theta^2 \left(\|Ax - a\|^2 + \left\| [Bx - b]_+ \right\|^2 \right)^2, \tag{38}$$

where $[\cdot]_+$ means the projection to the non-negative orthant, i.e., $[\cdot]_+ = \max\{\cdot, 0\}$.

Functions Used in Nonconvex Analysis

Proper Function

A function $g : \mathbb{R}^n \to (-\infty, +\infty]$ *is said to be proper if dom* $g \neq \varnothing$, *where dom* $g = \{x \in \mathbb{R}^n \mid g(x) < +\infty\}$.

We only consider proper functions.

Coercive Function

f *is called coercive if* $\lim_{\|x\| \to \infty} f(x) \to \infty$.

A coercive function is a function that “grows rapidly” at the extremes of the space on which it is defined.

Lower Semicontinuous Function

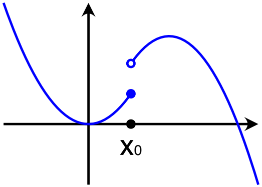
A function $g : \mathbb{R}^n \to (-\infty, +\infty]$ *is said to be lower semicontinuous at point* x_0 *if* $\liminf_{x \to x_0} g(x) \geq g(x_0)$.

A function is called lower semicontinuous if the function is lower semicontinuous at every point of its domain.

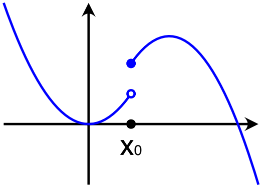
More about Semi-Continuity

Semi-continuity is a property of extended real-valued functions that is weaker than continuity.

An extended real-valued function f is upper (respectively, lower) semicontinuous at a point x_0 if, roughly speaking, the function values for arguments near x_0 are not much higher (respectively, lower) than $f(x_0)$.



(a)



(b)

In the figures above, the curve in (a) is lower semicontinuous while the curve in (b) is upper semicontinuous. Lower semicontinuous will be required for nonconvex analysis.

Subdifferential and its Properties

Subdifferential

Let f be a proper and lower semicontinuous function.

1. *For a given $x \in \mathrm{dom}\, f$, the Fréchet subdifferential of f at x, written as $\hat{\partial} f(x)$, is the set of all vectors $u \in \mathbb{R}^n$, which satisfies*

$$\liminf_{y \neq x, y \to x} \frac{f(y) - f(x) - \langle u, y - x \rangle}{\|y - x\|} \geq 0. \tag{39}$$

2. *The limiting subdifferential, or simply the subdifferential, of f at $x \in \mathbb{R}^n$, written as $\partial f(x)$, is defined through the following closure process:*

$$\begin{aligned} \partial f(x) = \big\{ u \in \mathbb{R}^n \,|\, \exists x_k \to x, f(x_k) \to f(x), \\ u_k \in \hat{\partial} f(x_k) \to u, k \to \infty \big\}. \end{aligned} \tag{40}$$

Subdifferential and its Properties (Cont'd)

Some properties of subdifferential:

1. In the nonconvex context, Fermat's rule remains unchanged: If $x \in \mathbb{R}^n$ is a local minimizer of g, then $0 \in \partial g(x)$.
2. Let (x_k, u_k) be a sequence such that $x_k \to x$, $u_k \to u$, $g(x_k) \to g(x)$, and $u_k \in \partial g(x_k)$, then $u \in \partial g(x)$.
3. If f is a continuously differentiable function, then

$$\partial(f + g)(x) = \nabla f(x) + \partial g(x). \tag{41}$$

Measurement of Convergence Rate

For a sequence $\{x^k\}_k$ that converges to x^*,

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} = C \tag{42}$$

must satisfy $C \leq 1$. The reason is intuitive. Since $x^k \to x^*$, there must exist infinitely many indices such that $\|x^{k+1} - x^*\|/\|x^k - x^*\| \leq 1$.

Obviously, the smaller C is, the faster the convergence speed. Specifically,

1. If $C = 1$, we say $\{x^k\}$ is *sublinear* convergent (e.g., $\{\frac{1}{k}\}$ and $\{\frac{1}{k^2}\}$);
2. If $C \in (0, 1)$, we say $\{x^k\}$ is *linear* convergent (e.g., $\{\frac{1}{2^k}\}$);
3. If $C = 0$, we say $\{x^k\}$ is *superlinear* convergent (e.g., $\{\frac{1}{2^{2^k}}\}$).

Measurement of Convergence Rate

For a superlinear convergent sequence $\{x^k\}$, if $\exists p > 1$ such that

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^p} = C', \tag{43}$$

and C' is a constant, then we say $\{x^k\}$ *converges with order p*.

Actually, linear convergence is fast enough, and we can find that the sequence's curve decreases exponentially. Thus we also call linear convergence *exponential convergence*. Correspondingly, the curve of sublinearly-convergent sequences decreases polynomially, thus we also call sublinear convergence *polynomial convergence*.

Measurement of Convergence Rate

Many optimization algorithms cannot achieve linear convergence. Thus, we need a more precise way to measure sublinear convergence. For example, if some algorithm $\mathcal{A}$ satisfies

$$\|f(x^k) - f^*\| \leq \frac{\|x^0 - x^*\|^2}{2tk}, \tag{44}$$

then we say: With $\mathcal{A}$, $\{f(x^k)\}_k$ converges to f^* with the speed $\mathcal{O}(\frac{1}{k})$, where $\frac{1}{k}$ is the gap between f^k and the optimum. If k is doubled, i.e., the iteration number is doubled, then the gap shrinks to half.

Measurement of Convergence Rate

Another way to represent the convergence speed is $\mathcal{O}(\frac{1}{\epsilon})$. If we want to limit the gap to ϵ, i.e., $\|f(x^k) - f^*\| \leq \frac{\|x^0 - x^*\|^2}{2tk} \leq \epsilon$, then $k \geq \frac{\|x^0 - x^*\|^2}{2t\epsilon} = \mathcal{O}(1/\epsilon)$. In other words, if we want to reduce the gap in half, we need to double the iteration number.

With these representations, $\mathcal{O}(1/k^2)$, i.e., $\mathcal{O}(1/\sqrt{\epsilon})$ gap, is much faster than $\mathcal{O}(1/k)$, although both of them are sublinear.

Similarly, we have

- For linear convergence: $\mathcal{O}(\gamma^{-k})$, or $\mathcal{O}(\log \frac{1}{\epsilon})$;
- For quadratic convergence ($p = 2$): $\mathcal{O}(\gamma^{-2^k})$, or $\mathcal{O}(\log \log \frac{1}{\epsilon})$.

Laplacian Matrix

Denote a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the node and the edge sets, respectively. $e_{ij} = (i, j) \in \mathcal{E}$ indicates that nodes i and j are connected. Define $\mathcal{V}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$ to be the neighborhood of node i, i.e., the index set of the nodes that are connected to node i. *The Laplacian matrix L of the graph* is defined as

$$Lij = \begin{cases} |\mathcal{V}_i| & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } (i, j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

L has the following properties:

1. $L \succeq 0$;
2. $\operatorname{rank}(L) = n - c$, where c is the number of connected components in the graph, and the eigenvector associated to 0 is 1_n.

Outline

Linearly Constrained Convex Problem

We firstly consider the following linearly constrained convex problem:

$$\mathcal{P}_2 : \min_x f(x), \quad s.t. \quad Ax = b, \tag{46}$$

where f is (i) proper, (ii) closed and (iii) convex. We only care about proper and closed function. Thus the two adjectives will be omitted for simplification.

With the preliminaries mentioned in the previous section, for $\mathcal{P}_2$ we have the following conclusions:

- Strong duality holds.
- Obtained by Danskin's Theorem —— $d(\lambda) := -f^*(-A^T\lambda) - \langle \lambda, b\rangle$ is differentiable when f is strictly convex and in this case $\forall \lambda, \nabla d(\lambda) = A\bar{x} - b$, where $\bar{x}$ is the minimizer of $L(x, \lambda)$.

Dual Descent

Based on the above analysis, we can have the following iterations to solve $\mathcal{P}_2$:

$$x^{k+1} = \operatorname*{argmin}_x L(x, \lambda^k) \tag{47}$$

$$\lambda^{k+1} = \lambda^k + \alpha_k(Ax^{k+1} - b), \tag{48}$$

where (47) is to solve a subproblem on x with the newest λ; (48) is a (sub)gradient ascent to update λ. α_k is the stepsize at iteration k.

This algorithm is called *Dual Descent*.

Augmented Lagrangian Method

Note that (48) is a gradient ascent, rather than subgradient ascent, only when f is strictly convex. However, this greatly limits the application of Dual Descent because subgradient ascent is much slower.

Nevertheless, we can use the augmented Lagrangian function to remedy this:

$$L_\beta(x, \lambda) := f(x) + \langle \lambda, Ax - b \rangle + \frac{\beta}{2}\|Ax - b\|^2. \tag{49}$$

The corresponding dual function is $d_\beta(\lambda) = \min_x L_\beta(x, \lambda)$.

Augmented Lagrangian Method

Because the optimal solution of x^* satisfies that $Ax^* = b$, thus we have $d_\beta(\lambda^*) \leq f(x^*)$. Moreover, for any λ we have $d(\lambda) \leq d_\beta(\lambda)$. Since $d(\lambda^*) = f(x^*)$, we can conclude that

$$d(\lambda^*) = d_\beta(\lambda^*) = f(x^*). \tag{50}$$

In other words, the augmented term does not change the solution. However, using the augmented Lagrangian function brings great benefits: *For $d_\beta(\lambda)$ to be differentiable we only require f to be convex, not strictly convex.*

The augmented Lagrangian Method (a.k.a. *Method of Multipliers*) has the following iterations:

$$x^{k+1} = \operatorname*{argmin}_x L_\beta(x, \lambda^k) \tag{51}$$

$$\lambda^{k+1} = \lambda^k + \beta(Ax^{k+1} - b), \tag{52}$$

where the stepsize in (52) is fixed as β.

Augmented Lagrangian Method

The augmented Lagrangian method can also be derived from the dual problem. Recall that the dual problem is $\min_\lambda f^*(-A^T\lambda) + \langle \lambda, b \rangle$. We use the Proximal Point Method to solve it:

$$\lambda^{k+1} = \operatorname*{argmin}_{\lambda} \Big(f^*(-A^T\lambda) + \langle \lambda, b \rangle + \frac{1}{2\beta} \| \lambda - \lambda^k \|^2 \Big). \tag{53}$$

The optimality condition is

$$0 \in -A\partial f^*\big(-A^T\lambda^{k+1}\big) + b + \frac{1}{\beta}\big(\lambda^{k+1} - \lambda^k\big). \tag{54}$$

(54) means that there exists

$$x^{k+1} \in \partial f^*\big(-A^T\lambda^{k+1}\big) \tag{55}$$

such that $0 = -Ax^{k+1} + b + \frac{1}{\beta}\big(\lambda^{k+1} - \lambda^k\big)$, which leads to (52).

Augmented Lagrangian Method

On the other hand, with the properties of the conjugate of a convex function and (55), we have

$$-A^T\lambda^{k+1} \in \partial f(x^{k+1}), \tag{56}$$

which means

$$\begin{aligned} 0 &\in \partial f(x^{k+1}) + A^T\lambda^{k+1} \\ &= \partial f(x^{k+1}) + A^T\big(\lambda^k + \beta(Ax^{k+1} - b)\big). \end{aligned} \tag{57}$$

(57) leads to (51).

Note that we have used two pages to derive the iterations of the augmented Lagrangian method based on *the Proximal Point method* on the dual problem.

Alternating Direction Method of Multipliers

Consider a speicial case of $\mathcal{P}_2$, named as $\mathcal{P}_2'$, which has a *separable* structure:

$$\mathcal{P}_2' : \min_{x,y} f(x) + g(y), \quad s.t. \quad Ax + By = b. \tag{58}$$

The augmented Lagrangian function is

$$\begin{aligned} L_\beta(x, y, \lambda) = f(x) + g(y) + \langle Ax + by - b, \lambda \rangle \\ + \frac{\beta}{2} \|Ax + By - b\|^2. \end{aligned} \tag{59}$$

With the augmented Lagrangian method, we need to solve one subproblem as follows.

$$(x^{k+1}, y^{y+1}) = \operatorname*{argmin}_{x,y} L_\beta(x, y, \lambda). \tag{60}$$

Alternating Direction Method of Multipliers

Sometimes, it is much simpler when we *solve $\mathcal{P}_2'$ for x and y separately*, which motivates the ADMM [appeared in 1970s]. Different from the augmented Lagrangian method, ADMM updates x and y in an *alternating* (or called sequential) fashion:

$$x^{k+1} = \operatorname*{argmin}_x L_\beta(x, y^k, \lambda^k) \tag{61}$$

$$y^{k+1} = \operatorname*{argmin}_y L_\beta(x^{k+1}, y, \lambda^k) \tag{62}$$

$$\lambda^{k+1} = \lambda^k + \beta\big(Ax^{k+1} + By^{k+1} - b\big). \tag{63}$$

We call it *ADMM*. ADMM is superior to the augmented Lagrangian method when the x and y subproblems can be more efficiently solved.

$$\text{Dual Descent} \xrightarrow[\textit{add aug. term}]{} \text{Method of Multipliers}$$

$$\text{Method of Multipliers} \xrightarrow[\textit{separate primal vars.}]{} \text{ADMM}$$

Convergence of ADMM

ADMM is guaranteed to converge when the objective is convex.

Theorem II-1 (Existence of Convergence)

Suppose that *$f(x)$ and $g(y)$ are convex*. When using ADMM to solve $\mathcal{P}_2'$, we have

$$f(x^{k+1}) - f(x^*) + g(y^{k+1}) - g(y^*) \to 0, \tag{64}$$

$$Ax^{k+1} + by^{k+1} - b \to 0, \tag{65}$$

when $k \to 0$.

Sublinear Convergence Rate of ADMM

The following result was firstly proved in 2015.

Theorem II-2 (Sublinear Non-Ergodic Convergence Rate)

Suppose that *$f(x)$ and $g(y)$ are convex*. When using ADMM to solve $\mathcal{P}_2'$, we have

$$
\begin{aligned}
-\|\lambda^*\|\sqrt{\frac{C}{\beta(K+1)}} &\leq f(x^{k+1}) - f(x^*) + g(y^{k+1}) - g(y^*) \\
&\leq \frac{C}{K+1} + \frac{2C}{\sqrt{K+1}} + \|\lambda^*\|\sqrt{\frac{C}{\beta(K+1)}},
\end{aligned}
\tag{66}
$$

where $C = \frac{1}{\beta}\|\lambda^0 - \lambda^*\|^2 + \beta\|By^0 - By^*\|^2$.

Sublinear Convergence Rate of ADMM

The following result was firstly proved in 2012.

Theorem II-3 (Sublinear Ergodic Convergence Rate)

Suppose that *$f(x)$ and $g(y)$ are convex*. When using ADMM to solve $\mathcal{P}_2'$, we have

$$\begin{aligned} &|f(\hat{x}^{K+1}) - f(x^*) + g(\hat{y}^{K+1}) - g(y^*)| \\ &\qquad\qquad \le \frac{C}{2(K+1)} + \frac{2\sqrt{C}\|\lambda^*\|}{\sqrt{\beta}(K+1)}, \end{aligned} \tag{67}$$

$$\|A\hat{x}^{K+1} + B\hat{y}^{K+1} - b\| \le \frac{2\sqrt{C}}{\sqrt{\beta}(K+1)}. \tag{68}$$

where

$$\hat{x}^{K+1} = \frac{1}{K+1}\sum_{k=1}^{K+1} x^k, \quad \hat{y}^{K+1} = \frac{1}{K+1}\sum_{k=1}^{K+1} y^k. \tag{69}$$

Linear Convergence Rate of ADMM

The following result was firstly proved in 2016. To achieve linear convergence rate, more requirements need to be satisfied.

Theorem II-4 (Linear Convergence Rate with Assumption #1)

Suppose that *$f(x)$ is convex and $g(y)$ is μ-strongly convex and L-smooth*. Assume that *$\forall \lambda, \|B^T \lambda\| \geq \sigma \|\lambda\|$, where $\sigma > 0$. This means B should be fully rank*. Let $\beta = \frac{\sqrt{\mu L}}{\sigma \|B\|}$. When using ADMM to solve $\mathcal{P}_2'$, we have

$$\begin{aligned}
&\frac{1}{2\beta}\|\lambda^{k+1} - \lambda^*\|^2 + \frac{\beta}{2}\|By^{k+1} - By^*\|^2 \\
\leq &\left(1 + \frac{1}{2}\sqrt{\frac{\mu}{L}}\frac{\sigma}{\|B\|}\right)^{-1}\left(\frac{1}{2\beta}\|\lambda^k - \lambda^*\|^2 + \frac{\beta}{2}\|By^k - By^*\|^2\right).
\end{aligned} \tag{70}$$

Linear Convergence Rate of ADMM

If the error bound condition is satisfied, even though g is not strongly convex and smooth, linear convergence rate can still be achieved. We firstly introduce what is error bound condition. For $\mathcal{P}_2'$, we define

$$\phi(x, y, \lambda) := \begin{pmatrix} \partial f(x) + A^T\lambda \\ \partial \mathrm{g}(y) + B^T\lambda \\ Ax + By - b \end{pmatrix}. \tag{71}$$

Correspondingly,

$$\phi^{-1}(s) := \{(x, y, \lambda) \mid s \in \phi(x, y, \lambda)\} \tag{72}$$

Obviously, (x^*, y^*, λ^*) is a KKT point *iff* $0 \in \phi(x^*, y^*, \lambda^*)$.

Linear Convergence Rate of ADMM

Recall that the distance between a vector and a set/space is defined as

$$\operatorname{dist}(x, \mathcal{S}) := \min_{y \in \mathcal{S}} \|x - y\|. \tag{73}$$

The distance relative to a matrix is defined as

$$\begin{aligned} \operatorname{dist}_H(x, \mathcal{S}) &:= \min_{y \in \mathcal{S}} \|x - y\|_H \\ &= \min_{y \in \mathcal{S}} \sqrt{(x - y)^T H (x - y)}. \end{aligned} \tag{74}$$

Linear Convergence Rate of ADMM

Definition II-1 (Error Bound Condition)

The set-value mapping $\phi(w)$ satisfies the (global) error bound condition, if there exists a constant $\kappa > 0$ such that, $\forall w$,

$$dist_H(w, \phi^{-1}(0)) \leq \kappa dist(0, \phi(w)), \tag{75}$$

where

$$H := \begin{pmatrix} 0 & 0 & 0 \\ 0 & \beta B^T B & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix}. \tag{76}$$

The LHS of (75) is the matrix-relevant distance between any solution $w = (x, y, \lambda)$ and the KKT point set. The RHS is the distance between optimal (sub)gradient (i.e., 0) at the critical point and the (sub)gradients at w.

Linear Convergence Rate of ADMM

The following result was firstly proved in 2018.

Theorem II-5 (Linear Convergence Rate with Assumption #2)

Suppose that *$f(x)$ and $g(y)$ are convex and $\phi(w)$ satisfies the error bound condition* (75). When using ADMM to solve $\mathcal{P}_2'$, we have

$$\begin{aligned}&\operatorname{dist}_H^2((x^{k+1}, y^{k+1}, \lambda^{k+1}), \phi^{-1}(0)) \\ \leq& \left[1 + \frac{1}{\kappa^2(\beta\|A\|^2 + \frac{1}{\beta})}\right]^{-1} \operatorname{dist}_H^2((x^k, y^k, \lambda^k), \phi^{-1}(0)).\end{aligned} \quad (77)$$

Bregman ADMM

Note that in ADMM, (61) and (62) are time-consuming. We can use linearization technique to make the subproblems compuatationally efficient:

$$x^{k+1} = \underset{x}{\operatorname{argmin}} \left(L_\beta(x, y^k, \lambda^k) + D_\phi(x, x^k) \right) \tag{78}$$

$$y^{k+1} = \underset{y}{\operatorname{argmin}} \left(L_\beta(x^{k+1}, y, \lambda^k) + D_\Psi(y, y^k) \right) \tag{79}$$

$$\lambda^{k+1} = \lambda^k + \beta\left(Ax^{k+1} + By^{k+1} - b\right). \tag{80}$$

The above algorithm is called *Bregman ADMM*. Wherein, D_ϕ and D_Ψ are Bregman distance w.r.t. ϕ and Ψ, respectively.

Bregman ADMM appeared at 2002, and it was in-depth studied in 2011 $\sim$ 2015.

Sublinear Convergence Rate of Bregman ADMM

Bregman ADMM is easily computable with linearization, but with the cost of more computation times / iterations. Even though, it can still achieve sublinear convergence rate without any additional requirements.

Theorem II-6 (Sublinear Convergence Rate)

Suppose that $f(x)$ and $g(y)$ are convex. When using Bregman ADMM to solve $\mathcal{P}_2'$, we have

$$
\begin{aligned}
&|f(\hat{x}^{K+1}) + g(\hat{y}^{K+1}) - f(x^*) - g(y^*)| \\
&\qquad\qquad \leq \frac{D}{2(K+1)} + \frac{2\sqrt{D}\|\lambda^*\|}{\sqrt{\beta}(K+1)}, \qquad (81)
\end{aligned}
$$

$$
\|A\hat{x}^{K+1} + B\hat{y}^{K+1} - b\| \leq \frac{2\sqrt{D}}{\sqrt{\beta}(K+1)}. \qquad (82)
$$

$\hat{x}^{K+1}$ and $\hat{y}^{K+1}$ are defined in (69), *and D is a constant relative to the element-wise gap between (x^0, y^0, λ^0) and (x^*, y^*, λ^*).*

LADMM-1

When we set different ϕ and Ψ, different ADMMs can be obtained. If we set

$$\phi(x) = \frac{\beta\|A\|^2}{2}\|x - u_1\|^2 - \frac{\beta}{2}\|Ax - u_2\|^2 \tag{83}$$

$$\Psi(y) = \frac{\beta\|B\|^2}{2}\|y - v_1\|^2 - \frac{\beta}{2}\|By - v_2\|^2. \tag{84}$$

u_1, u_2, v_1, and v_2 are constants. Then (78) reduces to

$$\begin{aligned} x^{k+1} = \operatorname*{argmin}_x \Big(& f(x) + g(y^k) + \langle \lambda^k, Ax + By^k - b \rangle \\ & + \mathcal{A}\Big(\frac{\beta}{2}\|Ax + By^k - b\|^2\Big) + \frac{\beta\|A\|^2}{2}\|x - x^k\|^2 \Big), \end{aligned} \tag{85}$$

LADMM-1

Then (79) reduces to (cont'd)

$$\begin{aligned} y^{k+1} = \operatorname*{argmin}_y \Big(& f(x^{k+1}) + g(y) + \langle \lambda^k, Ax^{k+1} + By - b \rangle \\ & + \mathcal{A}\Big(\frac{\beta}{2}\|Ax^{k+1} + By - b\|^2\Big) + \frac{\beta\|B\|^2}{2}\|y - y^k\|^2 \Big), \end{aligned} \quad (86)$$

while (80) keeps unchanged.

LADMM-1

In (85), $\mathcal{A}\left(\frac{\beta}{2}\|Ax + By^k - b\|^2\right)$ is the linear approximation of $\frac{\beta}{2}\|Ax + By^k - b\|^2$ at x^k. It has the form:

$$\underbrace{\frac{\beta}{2}\|Ax^k + By^k - b\|^2}_{h(x^k)} + \underbrace{\beta\langle A^T(Ax^k + By^k - b), x - x^k\rangle}_{\langle \nabla h(x^k), x - x^k\rangle}. \quad (87)$$

In (86), $\mathcal{A}\left(\frac{\beta}{2}\|Ax^{k+1} + By - b\|^2\right)$ is the linear approximation of $\frac{\beta}{2}\|Ax^{k+1} + By - b\|^2$ at y^k. It has the form:

$$\frac{\beta}{2}\|Ax^{k+1} + By^k - b\|^2 + \beta\langle B^T(Ax^{k+1} + By^k - b), y - y^k\rangle. \quad (88)$$

This variant of Bregman ADMM, i.e., (85), (86), and (80), is called *LADMM-1*.

LADMM-2

Obviously, LADMM-1 is good to use when the proximal mappings of f and g, i.e., $\operatorname{argmin}_x \left(f(x) + \frac{\beta\|A\|^2}{2}\|x - x^k\|^2\right)$ and $\operatorname{argmin}_y \left(g(y) + \frac{\beta\|B\|^2}{2}\|y - y^k\|^2\right)$ are easily to compute.

When the proximal terms are not easily computable (note that we still require them to be convex), but f and g are L_f-smooth and L_g-smooth, respectively, we may choose

$$\phi(x) = \frac{L_f + \beta\|A\|^2}{2}\|x - u_1\|^2 - \frac{\beta}{2}\|Ax - u_2\|^2 - f(x) \tag{89}$$

$$\Psi(y) = \frac{L_g + \beta\|B\|^2}{2}\|y - v_1\|^2 - \frac{\beta}{2}\|By - v_2\|^2 - g(y). \tag{90}$$

Note that f and g are subtracted. It means they are also approximated.

LADMM-2

Specifically, with (89) and (90), we have

$$\begin{aligned} x^{k+1} = \operatorname*{argmin}_{x} \Big(& g(y^k) + \langle \lambda^k, Ax + By^k - b \rangle \\ & + \mathcal{A}\Big(f(x) + \frac{\beta}{2}\|Ax + By^k - b\|^2\Big) \\ & + \frac{L_f + \beta\|A\|^2}{2}\|x - x^k\|^2\Big), \end{aligned} \tag{91}$$

where $\mathcal{A}(f(x) + \frac{\beta}{2}\|Ax + By^k - b\|^2)$ is the linear approximation of $f(x) + \frac{\beta}{2}\|Ax + By^k - b\|^2$. It has the form:

$$\begin{aligned} & f(x^k) + \langle \nabla f(x^k), x - x^k \rangle \\ + & \frac{\beta}{2}\|Ax^k + By^k - b\|^2 + \beta\langle A^T(Ax^k + By^k - b, x - x^k)\rangle. \end{aligned}$$

LADMM-2

With (89) and (90), we further have (cont'd)

$$\begin{aligned} y^{k+1} = \operatorname*{argmin}_y \Big(& f(x^{k+1}) + \langle \lambda^k, Ax^{k+1} + By - b \rangle \\ & + \mathcal{A}\Big(g(y) + \frac{\beta}{2}\|Ax^{k+1} + By - b\|^2 \Big) \\ & + \frac{L_g + \beta\|B\|^2}{2}\|y - y^k\|^2 \Big), \end{aligned} \tag{92}$$

where $\mathcal{A}(g(y) + \frac{\beta}{2}\|Ax^{k+1} + By - b\|^2)$ is the linear approximation of $g(y) + \frac{\beta}{2}\|Ax^{k+1} + By - b\|^2$. It has the form:

$$\begin{aligned} & g(y^k) + \langle \nabla g(y^k), y - y^k \rangle \\ + \frac{\beta}{2}\|Ax^{k+1} + By^k - b\|^2 & + \beta\langle B^T(Ax^{k+1} + By^k - b), y - y^k \rangle. \end{aligned}$$

This variant of Bregman ADMM, i.e., (91), (92), and (80), is called *LADMM-2*.

Linear Convergence Rate of Bregman ADMM

If we add more requirements on convexity, linear convergence rate of Bregman ADMM can be achieved. Specifically,

1. If f **is generally convex, but** g **is** μ_g**-strongly convex and** L_g**-smooth**, then we linearize the second subproblem and the augmented term, i.e., $\phi \equiv 0$.
2. If f **is** μ_f**-strongly convex, and** g **is** μ_g**-strongly convex and** L_g**-smooth**, then we linearize both subproblems and the augmented term (as what LADMM-2 does).

For the above two scenarios, linear convergence rate can be achieved by Bregman ADMM.

Comparison Between ADMM and LADMMs

Linear complexity comparisons between ADMM and two variants of linearized ADMM, for solving $\mathcal{P}_2'$:

METHOD	RATE	LINEARIZATION
ADMM	$\mathcal{O}(\sqrt{\frac{L_g}{\mu_g}}\frac{\lVert B\rVert_2}{\sigma}\log\frac{1}{\epsilon})$	None
LADMM-1	$\mathcal{O}((\sqrt{\frac{L_g}{\mu_g}}\frac{\lVert B\rVert_2}{\sigma}+\frac{\lVert B\rVert_2^2}{\sigma^2})\log\frac{1}{\epsilon})$	On aug.
LADMM-2	$\mathcal{O}((\frac{\lVert B\rVert_2^2}{\sigma^2}+\frac{L_g}{\mu_g})\log\frac{1}{\epsilon})$	On f, g and aug.

Note that all the three complexities in this table are linear (with smooth and strongly convex assumptions). However, LADMMs is slower than ADMM. We have explained this —— Linearized ADMM is easily computable with linearization, but with the cost of more computation times / iterations.

Nesterov's Acceleration Techniques

Still for $\mathcal{P}_2'$, if f or g is smooth, or they have a smooth part, then we can use Nesterov's acceleration techniques to accelerate the LADMMs [proposed in 2015, SIAM].

We firstly consider the following assumption.

Assumption II-1

In $\mathcal{P}_2'$, both f and g are convex and g is L_g-smooth.

In this case, we can linearize g at the auxiliary variable v^k in the y update step.

Acc-LADMM-1

Specifically we have:

$$x^{k+1} = \operatorname*{argmin}_x L_\beta(x, y^k, \lambda^k), \quad \text{(same as (61))} \tag{93}$$

$$v^k = \theta_k y^k + (1-\theta_k)\tilde{y}^k, \tag{94}$$

$$\begin{aligned} y^{k+1} = \operatorname*{argmin}_y \Big(& f(x^{k+1}) + \langle \lambda^k, Ax^{k+1} + By - b\rangle \\ & + \mathcal{A}_1\Big(g(y)\Big) + \mathcal{A}_2\left(\frac{\beta}{2}\|Ax^{k+1} + By - b\|^2\right) \\ & + \frac{L_g\theta_k + \beta\|B\|^2}{2}\|y - y^k\|^2\Big), \end{aligned} \tag{95}$$

where $\mathcal{A}_1(g(y)) := g(v^k) + \langle \nabla g(v^k), y - v^k\rangle$ is the linearization approximation at v^k; $\mathcal{A}_2(\frac{\beta}{2}\|Ax^{k+1} + By - b\|^2)$ is the linear approximation at y^k, which is the same as (88).

Acc-LADMM-1

Specifically we have (cont'd):

$$\tilde{x}^{k+1} = \theta_k x^{k+1} + (1-\theta_k)\tilde{x}^k, \tag{96}$$

$$\tilde{y}^{k+1} = \theta_k y^{k+1} + (1-\theta_k)\tilde{y}^k, \tag{97}$$

$$\lambda^{k+1} = \lambda^k + \beta\left(Ax^{k+1} + By^{k+1} - b\right). \quad \text{(unchanged)} \tag{98}$$

(93) $\sim$ (98) are called *Acc-LADMM-1. It linearizes* g *at the auxiliary variable* v^k *in the* y *update step while keeping the calculate of* x *unchanged.*

Compare Acc-LADMM-1 with LADMM-2

It's interesting to compare (95) with (92). The main difference is that, in Acc-LADMM-1, the linearization of g is taken at the auxiliary variable v^k, rather than y^k, which is introduced by the Nesterov's acceleration techniques.

Although not easy, but we can prove that Acc-LADMM-1 is still sublinear convergent: $\mathcal{O}(\frac{1}{K} + \frac{L_g}{K^2})$. Acc-LADMM-1 is faster than LADMM-2, and it does not require f to be smooth.

Acc-LADMM-2

If both f and g are smooth, or both of them have a smooth part, we can linearize both of them [proposed in 2019]. Firstly, we formalize our assumption:

Assumption II-2

In $\mathcal{P}_2'$, both f and g are convex and they have composite structures:

$$f(x) = f_1(x) + f_2(x), \quad g(y) = g_1(y) + g_2(y), \tag{99}$$

where both f_2 and g_2 are L-smooth.

We give the following new iterations:

$$u^k = x^k + \frac{\theta_k(1-\theta_{k-1})}{\theta_{k-1}}(x^k - x^{k-1}), \tag{100}$$

$$v^k = y^k + \frac{\theta_k(1-\theta_{k-1})}{\theta_{k-1}}(y^k - y^{k-1}), \tag{101}$$

Acc-LADMM-2

We give the following new iterations (cont'd):

$$x^{k+1} = \operatorname*{argmin}_x \Big(g(v^k) + \langle \lambda^k, Ax + Bv^k - b \rangle + f_1(x) + \mathcal{A}\Big(f_2(x) + \frac{\beta}{2\theta_k} \|Ax + Bv^k - b\|^2 \Big) + \frac{L + \frac{\beta}{\theta_k}\|A\|^2}{2} \|x - u^k\|^2 \Big), \tag{102}$$

where

$$\mathcal{A}\Big(f_2(x) + \frac{\beta}{2\theta_k} \|Ax + Bv^k - b\|^2 \Big) = f_2(u^k) + \langle \nabla f_2(u^k), x - u^k \rangle + \frac{\beta}{2\theta_k} \|Au^k + Bv^k - b\|^2 + \frac{\beta}{\theta_k} \langle A^T(Au^k + Bv^k - b), x - u^k \rangle. \tag{103}$$

Acc-LADMM-2

We give the following new iterations (cont'd):

$$
\begin{aligned}
y^{k+1} = \operatorname*{argmin}_{y} \bigg(& f(u^{k+1}) + \langle \lambda^k, Au^{k+1} + By - b \rangle \\
& + g_1(y) + \mathcal{A}\left(g_2(y) + \frac{\beta}{2\theta_k} \|Au^{k+1} + By - b\|^2 \right) \\
& + \frac{L + \frac{\beta}{\theta_k}\|B\|^2}{2} \|y - v^k\|^2 \bigg),
\end{aligned} \tag{104}
$$

where

$$
\begin{aligned}
\mathcal{A}\left(g_2(y) + \frac{\beta}{2\theta_k} \|Au^{k+1} + By - b\|^2 \right) = {} & g_2(v^k) \\
& + \langle \nabla g_2(v^k), y - v^k \rangle + \frac{\beta}{2\theta_k} \|Au^{k+1} + Bv^k - b\|^2 \\
& + \frac{\beta}{\theta_k} \langle B^T(Au^{k+1} + Bv^k - b), y - v^k \rangle.
\end{aligned} \tag{105}
$$

Acc-LADMM-2

We give the following new iterations (cont'd):

$$\lambda^{k+1} = \lambda^k + \beta\tau\left(Ax^{k+1} + By^{k+1} - b\right), \tag{106}$$

where $\beta\tau$ is the stepsize. We call (100), (101), (102), (104), and (106) *Acc-LADMM-2*. Obviously, when $\forall k, \theta_k \equiv 1$ and $\tau = 1$, Acc-LADMM-2 reduces to non-accelerated LADMMs.

With the smooth requirements defined in Assumption II-2, Acc-LADMM-2 converges faster than Acc-LADMM-1. But its convergence rate is still sublinear.

Acc-LADMM-3

Recall that Acc-LADMM-1 works under Assumption II-1, which requires f and g to be convex and g to be L_g-smooth. Wherein, (95) is the acceleration of (92). If we further assume that g is μ_g-strongly convex, then (95) can be further accelerated.

Specifically, we have

$$v^{k+1} = \theta y^k + (1-\theta)\tilde{y}^k, \tag{107}$$

$$x^{k+1} = \operatorname*{argmin}_x \Big(f(x) + \langle \lambda^k, Ax + By^k - b \rangle + \frac{\beta\theta}{2}\|Ax + By^k - b\|^2 \Big). \tag{108}$$

Acc-LADMM-3

We further have (cont'd):

$$y^{k+1} = \operatorname*{argmin}_y \Big(\langle \nabla g(v^k), y \rangle + \langle \lambda^k, By \rangle + \beta\theta \langle B^T(Ax^{k+1} + By^k - b), y \rangle + \frac{1}{2}\left(\frac{\theta}{\alpha} + \mu_g\right) \left\| y - \frac{1}{\frac{\theta}{\alpha} + \mu_g}\left(\frac{\theta}{\alpha} y^k + \mu_g v^k\right) \right\|^2 \Big). \quad (109)$$

From (109) we can find that, $g(y)$ is linearly approximated at v^k while $\frac{\beta\theta}{2}\|Ax + By^k - b\|^2$ is approximated at y^k.

For the proximal term, we wants y to approach the weighted average of y^k and v^k, where their weights are $\frac{\theta}{\alpha}$ and μ_g, respectively.

Acc-LADMM-3

We further have (cont'd):

$$\tilde{x}^{k+1} = \theta x^{k+1} + (1-\theta)\tilde{x}^k, \tag{110}$$

$$\tilde{y}^{k+1} = \theta y^{k+1} + (1-\theta)\tilde{y}^k, \tag{111}$$

$$\lambda^{k+1} = \lambda^k + \beta\theta\big(Ax^{k+1} + By^{k+1} - b\big). \tag{112}$$

This method, (107) $\sim$ (112), is called *Acc-LADMM-3*, which was proposed in 2020.

Remember that Acc-LADMM-3 works fine under the assumption that, f is convex, and g is L_g-smooth and μ_g-strongly convex.

Comparison of Complexities

For solving $\mathcal{P}_2'$, LADMM-2 is slower than the vanilla ADMM, but, as we have analyzed, it is more easily computable. Acc-LADMM-3 is both easily computable and fast, although is has strong convexity requirement on g.

Method	Rate
ADMM	$\mathcal{O}(\sqrt{\frac{L_g}{\mu_g}}\frac{\|B\|_2}{\sigma}\log\frac{1}{\epsilon})$
LADMM-2	$\mathcal{O}((\frac{\|B\|_2^2}{\sigma^2}+\frac{L_g}{\mu_g})\log\frac{1}{\epsilon})$
Acc-LADMM-3	$\mathcal{O}(\sqrt{\frac{L_g}{\mu_g}}\frac{\|B\|_2}{\sigma}\log\frac{1}{\epsilon})$

Note that all the three complexities in this table are linear. However, Acc-LADMM-3 is able to overcome the slow convergence of LADMM-2, which is the same fast as ADMM.

Conclusion

Time to make a conclusion.

In the pages till now, we try to solve a linearly constrained problem $\mathcal{P}_2'$, located at (58).

- We firstly proposed a *Dual Descent method*, for $\mathcal{P}_2$, which updates the primal and dual variables alternatively. However, it works fine only when f is strictly convex. To remove this restrict we propose *the augmented Lagrangian method*, which adds an augmented term to the Lagrangian function to make its curve upward swinging.
- Further, based on the augmented Lagrangian method, if the objective of $\mathcal{P}_2$ is separatable, i.e., in the shape of $\mathcal{P}_2'$, then we can update the primal variables x and y alternatively. This is ADMM.

Conclusion (Cont'd)

- For ADMM, to achieve a sublinear convergence rate, f and g are required to be convex. This is the basic requirement for ADMM to convergent. Nevertheless, if more requirements can be met, for example,
    1. g is further μ-strongly convex and L-smooth, or
    2. the erorr bound condition (75) is satisfied,

  then linear convergence rate can be achieved.
- ADMM is criticized for its time-consuming computation on subproblems (61) and (62). To speed up the computation, linearization is introduced. This is Bregman ADMM, which is easily computable because part of the augmented Lagrangian function is linearized. However, its speed up is at the cost of more computation iterations. The reason behind is intuitive: *Minimizing the linearization of the objective cannot completely replace the objective after all.*

Conclusion (Cont'd)

- If the objective is smooth, Bregman ADMM works more efficiently. This is intuitive, because smooth functions can be better approximated by their linearizations. We introduce two different variant of Bregman ADMM, LADMM-1 and LADMM-2. They linearize different parts of the augmented Lagrangian function.
- For Bregman ADMM, with general assumptions, such as the objective is convex and smooth, sublinear convergence rate can be achieved. If strong convexity satisfies, linear convergence can be achieved.

Conclusion (Cont'd)

- Another goodness of smooth is that, the update of primal variables can be accelerated with Nesterov's acceleration techniques. Combing this technique with linearization, we have Acc-LADMM-1, Acc-LADMM-2, and Acc-LADMM-3.
- Acc-LADMM-1 accelerates the calculation of y^k by introducing the auxiliary variable v^k. Acc-LADMM-2 accelerates both the calculation of x^k and y^k. Thus Acc-LADMM-2 is faster than Acc-LADMM-1.
- If g is further strong convex, the calculation of y^k can be further accelerated. This is want Acc-LADMM-3 does. Correspondingly, Acc-ALDMM-3 is linear convergent.

Multi-Block Problems

$\mathcal{P}_2'$ is a two-block problem, where the objective can be separated into two disjoint functions, $f(x)$ and $g(y)$. Now we consider the multi-block version, $\mathcal{P}_3$.

$$\mathcal{P}_3 : \min_{\{x_i\}_{i\in[m]}} f(x) \equiv \sum_{i=1}^{m} f_i(x_i), \quad s.t. \quad \sum_{i=1}^{m} A_i x_i = b. \tag{113}$$

Wherein $\forall i \in [m]$, $x_i \in \mathbb{R}^{d_i}$, $A_i \in \mathbb{R}^{n\times d_i}$, and $b \in \mathbb{R}^n$. We define $x = [x_1^T, ..., x_m^T]^T \in \mathbb{R}^{\sum_{i=1}^m d_i}$ and $A = [A_1, ..., A_m] \in \mathbb{R}^{n\times \sum_{i=1}^m d_i}$.

Correspondingly, $\mathcal{P}_3$ can be represented as

$$\mathcal{P}_3 : \min_x f(x), \quad s.t. \quad Ax = b. \tag{114}$$

ADMM for Multi-Block Problems

The vanilla ADMM can solve $\mathcal{P}_3$. We name it *MB-ADMM*:

$$\begin{aligned} &\forall i \in [m] \textit{ in turn}: \\ &\quad \tilde{x}_i^{k+1} = \operatorname*{argmin}_{x_i} L_\beta(\tilde{x}_1^{k+1}, \dots, \tilde{x}_{i-1}^{k+1}, x_i, x_{i+1}^k, \dots, x_m^k, \lambda^k), \qquad (115) \\ &\lambda^{k+1} = \lambda^k + \beta(A\tilde{x}^{k+1} - b), \qquad (116) \\ &x^{k+1} = \tilde{x}^{k+1}, \forall i \in [m]. \qquad (117) \end{aligned}$$

where

$$L_\beta(x, \lambda) = f(x) + \langle \lambda, Ax - b \rangle + \frac{\beta}{2}\|Ax - b\|^2. \qquad (118)$$

$\tilde{x}^k$ is introduced for the following algorithms. In MB-ADMM, $\forall k, \tilde{x}^k \equiv x^k$. Unfortunately, it is proved that ADMM-MB might not converge. Thus, we need to devise new technique that at least guarantee the convergence.

ADMM-GBS

Although not easy, but based on the optimality condition of primal update, i.e.,

$$0 \in \partial_{x_i} L_\beta\Big(\{\tilde{x}_{i_1}^{k+1}\}_{i_1=1,\dots,i-1}, x_i, \{x_{i_2}^k\}_{i_2=i+1,\dots,m}, \lambda^k\Big), \tag{119}$$

and the monotone of subgradient, i.e., (10), we have

$$\begin{aligned} & f(\tilde{x}^{k+1}) - f(x^*) + \langle \lambda^*, A\tilde{x}^{k+1} - b\rangle \\ & \quad \le \frac{1}{2\beta}\Big(\|\lambda^k - \lambda^*\|^2 - \|\lambda^{k+1} - \lambda^*\|^2\Big) - \frac{\beta}{2}\|\tilde{x}^{k+1} - x^k\|_H^2 \\ & \quad - \beta(x^k - x^*)^T M(\tilde{x}^{k+1} - x^*). \end{aligned} \tag{120}$$

Actually, this intermediate result is frequently used in ADMMs' convergence analysis.

ADMM-GBS

In (120),

$$M = \begin{pmatrix} A_1^T A_1 & 0 & \cdots & 0 \\ A_2^T A_1 & A_2^T A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A_m^T A_1 & A_m^T A_2 & \cdots & A_m^T A_m \end{pmatrix} \tag{121}$$

is a lower triangular matrix, and

$$H == \begin{pmatrix} A_1^T A_1 & 0 & \cdots & 0 \\ 0 & A_2^T A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & A_m^T A_m \end{pmatrix} \tag{122}$$

is a diagonal matrix.

ADMM-GBS

To build a convergence analysis, we hope that the RHS of (120) has the shape

$$\begin{aligned}\frac{1}{2\beta}&\left(\|\lambda^k - \lambda^*\|^2 - \|\lambda^{k+1} - \lambda^*\|^2\right)\\ &+ \frac{\beta}{2}\left(\|x^k - x^*\|_G^2 - \|x^{k+1} - x^*\|_G^2\right),\end{aligned} \tag{123}$$

If we let

$$x^{k+1} = x^k + D(\tilde{x}^{k+1} - x^k), \tag{124}$$

which works as **correcting** x^{k+1} **from the predicted** $\tilde{x}^{k+1}$, and bring it into (123), we have

$$D^T GD = H, \quad GD = H. \tag{125}$$

ADMM-GBS

Assume that all A_i's are full column rank, then M and H are invertible. Combing (120), (124), and (125), we have

$$x^{k+1} = x^k + M^{-T} H(\tilde{x}^{k+1} - x^k). \tag{126}$$

Note that the matrix-vector multiplication in (126) can be calculated efficiently by Gaussian back substitution.

In this case, the iterations (115), (116), and (126), which are designed to solve $\mathcal{P}_3$, are called *ADMM-GBS* [2012, SIAM]. ADMM-GBS is able to achieve $\mathcal{O}(1/K)$ convergence rate.

ADMM-PC

Is ADMM-GBS simple enough? Yes, but it is computation-intensive. The reason is that in every iteration, by (126), we have to solve m linear systems. Pay attention to (115), actually, to solve it, we only need $A_i x_i$, rather than x_i. In other words, we don't need to compute x_i explicitly.

Denote

$$L = \begin{pmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ I & I & \cdots & I \end{pmatrix}, \quad P = \begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & A_m \end{pmatrix}. \tag{127}$$

Then

$$M = P^T L P, \quad H = P^T P. \tag{128}$$

ADMM-PC

Similarly, we hope that the RHS of (120) has the shape

$$\frac{1}{2\beta}\left(\|\lambda^k-\lambda^*\|^2-\|\lambda^{k+1}-\lambda^*\|^2\right) \\ +\frac{\beta}{2}\left(\|Px^k-Px^*\|_{G'}^2-\|Px^{k+1}-Px^*\|_{G'}^2\right), \tag{129}$$

If we let

$$Px^{k+1}=Px^k+D'(P\tilde{x}^{k+1}-Px^k), \tag{130}$$

then we have

$$D'=L^{-T}, \quad G'=LL^T. \tag{131}$$

ADMM-PC

In this case, (130) is equal to

$$\begin{pmatrix} A_1 x_1^{k+1} \\ A_2 x_2^{k+1} \\ \vdots \\ A_{m-1} x_{m-1}^{k+1} A_m x_m^{k+1} \end{pmatrix} = \begin{pmatrix} A_1 \tilde{x}_1^{k+1} + A_2 x_2^k - A_2 \tilde{x}_2^{k+1} \\ A_2 \tilde{x}_2^{k+1} + A_3 x_3^k - A_3 \tilde{x}_3^{k+1} \\ \vdots \\ A_{m-1} \tilde{x}_{m-1}^{k+1} + A_m x_m^k - A_m \tilde{x}_m^{k+1} \\ A_m \tilde{x}_m^{k+1} \end{pmatrix}. \tag{132}$$

In other words, we can obtain $A_i x_i^{k+1}$ directly without solving linear systems.

This looks good. However, we have to mention that, $A_i x_i$ exists is not equal to x_i exists. Thus, this method may not lead to a feasible solution.

ADMM-PC

Based on this, we present ADMM-PC [proposed in 2021]. In the following formulas, ξ_i is used to replace $A_i x_i$.

$$\forall i \in [m] \textit{ in turn}:$$
$$\tilde{x}_i^{k+1} = \underset{x_i}{\operatorname{argmin}} \tilde{L}_i(\tilde{x}_1^{k+1}, \ldots, \tilde{x}_{i-1}^{k+1}, x_i, \xi_{i+1}^k, \ldots, \xi_m^k, \lambda^k), \tag{133}$$
$$\lambda^{k+1} = \lambda^k + \beta(A\tilde{x}^{k+1} - b), \tag{134}$$
$$\xi^{k+1} = \xi^k + L^{-T}(P\tilde{x}^{k+1} - \xi^k), \forall i \in [m], \tag{135}$$

where $\tilde{L}_i(x_1, \ldots, x_i, \xi_{i+1}, \ldots, \xi_m, \lambda)$ is defined as

$$\sum_{j=1}^{i} f_i(x_j) + \langle \lambda, \sum_{j=1}^{i} A_j x_j + \sum_{j=i+1}^{m} \xi_j - b \rangle$$
$$+\frac{\beta}{2} \| \sum_{j=1}^{i} A_j x_j + \sum_{j=i+1}^{m} \xi_j - b \|^2, \tag{136}$$

and $\xi = (\xi_1^T, \ldots, \xi_m^T)^T$.

Drawbacks of Prediction-Corrections

ADMM-PC also has a $\mathcal{O}(1/K)$ convergence rate but compared with ADMM-GBS, the coefficient is smaller.

ADMM-GBS and ADMM-PC have the following drawbacks:

1. They need to maintain two groups of variables, the predicted and the corrected, thus increasing the memory cost (at least doubled)
2. Neither x_i^k (ADMM-GBS) nor ξ^k (ADMM-PC) could be sparse or low-rank, even in sparse or low-rank problems. Thus the memory consumption can be more

LADMM-PS

If we could remove the memory-intensive prediction-correction procedures and update all x_i's **parallelly**, then the convergence can be speed up. To do this, let's put our focus back to the linearization techniques.

Based on the Bregman ADMM (78) or (79), for $\mathcal{P}_3$, when we set

$$\phi(x) := \frac{\beta}{2}\|x\|_L^2 - \frac{\beta}{2}\|Ax\|^2, \tag{137}$$

The update of primal variables are:

$$x^{k+1} = \operatorname*{argmin}_x \Big(f(x) + \langle \lambda^k, Ax - b\rangle + \beta\langle A^T(Ax-b), x - x^k\rangle + \frac{\beta}{2}\|x - x^k\|_L^2 \Big), \tag{138}$$

which coincides with (85) of LADMM-1.

LADMM-PS

Note that in (138), $L = \text{diag}(L_1 I_{d_1}, ..., L_m I_{d_m})$, and the Bregman distance is meaningful only when ϕ is convex. Because

$$\begin{aligned}
\phi(x) &= \frac{\beta}{2} \sum_{i=1}^{m} x_i^T (L_i I_{d_i}) x_i - \frac{\beta}{2} \| \sum_{i=1}^{m} A_i x_i \|^2 \\
&\geq \sum_{i=1}^{m} x_i^T (L_i I_{d_i}) x_i - \frac{\beta}{2} m \sum_{i=1}^{m} \| A_i x_i \|^2 \\
&\geq \sum_{i=1}^{m} x_i^T (L_i I_{d_i}) x_i - \frac{\beta}{2} m \sum_{i=1}^{m} \| A_i \|^2 \| x_i \|^2 \\
&= \frac{\beta}{2} \sum_{i=1}^{m} x_i^T (L_i - m\|A_i\|^2) I_{d_i} x_i, \qquad (139)
\end{aligned}$$

Thus, to guarantee the convexity of ϕ, we should let

$$L_i \geq m\|A_i\|^2, \forall i \in [m]. \qquad (140)$$

LADMM-PS

Thus, with L satisfying (140), (138) is separable for each x_i:

$$x_i^{k+1} = \operatorname*{argmin}_{x_i} \Big(f_i(x_i) + \langle \lambda^k, A_i x_i \rangle + \beta \langle A_i^T (Ax^k - b), x_i \rangle + \frac{\beta L_i}{2} \|x_i - x_i^k\|^2 \Big), \tag{141}$$

which can be solved in parallel for each $i \in [m]$. The update of the dual variables is not changed:

$$\lambda^{k+1} = \lambda^k + \beta(Ax^{k+1} - b). \tag{142}$$

We call (141) and (142) *LADMM-PS* (Linearized ADMM with Parallel Splitting) [2013, 2015]. The same as LADMM-1, LADMM-PS can achieve sublinear and linear convergence rate under certain conditions.

LADMM-SPU

LADMM-PS to $\mathcal{P}_3$ is similar to Method of Multipliers to $\mathcal{P}_2'$. However, as we have analyzed, ADMM is faster than Method of Multipliers, i.e., the alternating update of primal variables is faster than update the primal variables in parallel. This inspires us to combine the alternating update and LADMM-PS for multi-block problems.

Specifically, we can divide the m blocks in $\mathcal{P}_3$ into two partitions:

$$1, \ldots, m' \text{ and } m' + 1, \ldots, m,$$

and then update the two partitions in serial, while updating the blocks in the same partition in parallel. This method is called *LADMM-SPU* (Linearized ADMM with the Serial and the Parallel Update Orders) [TPAMI, 2018].

LADMM-SPU

LADMM-SPU has the following iterations:

$$\begin{aligned} x_i^{k+1} &= \operatorname*{argmin}_{x_i} \Big(f_i(x_i) + \langle \lambda^k, A_i x_i \rangle + \beta \langle A_i^T (Ax^k - b), x_i \rangle \\ &\quad + \frac{m' \beta \|A_i\|^2}{2} \|x_i - x_i^k\|^2 \Big), \forall i \in [1, m'] \text{ in parallel}, \\ x_j^{k+1} &= \operatorname*{argmin}_{x_j} \Big(f_j(x_j) + \langle \lambda^k, A_j x_j \rangle \\ &\quad + \beta \langle A_j^T \Big(\sum_{t=1}^{m'} A_t x_t^{k+1} + \sum_{t=m'+1}^{m} A_t x_t^k - b \Big), x_j \rangle \\ &\quad + \frac{(m - m')\beta \|A_j\|^2}{2} \|x_j - x_j^k\|^2 \Big), \forall j \in [m'+1, m] \text{ in parallel}. \end{aligned}$$

In above formulas, L_i is set as $m' \|A_i\|^2$ while L_j is set as $(m - m')\|A_i\|^2$. LADMM-SPU is also a variant of Bregman ADMM. But it runs faster than LADMM-PS.

Conclusion

Time to make a conclusion.

So far, we have discussed several algorithms for solving the multi-block problem $\mathcal{P}_3$, defined in (113).

- The vanilla ADMM, i.e., MB-ADMM, is not guaranteed to converge. To make the ADMM-like procedure converges, we try to find suitable D and G such that

$$\begin{aligned}
&f(\tilde{x}^{k+1}) - f(x^*) + \langle \lambda^*, A\tilde{x}^{k+1} - b \rangle \\
&\quad \le \frac{1}{2\beta}\left(\|\lambda^k - \lambda^*\|^2 - \|\lambda^{k+1} - \lambda^*\|^2 \right) \\
&\quad + \frac{\beta}{2}\left(\|Px^k - Px^*\|_{G'}^2 - \|Px^{k+1} - Px^*\|_{G'}^2 \right).
\end{aligned}$$

  Then we get ADMM-GBS.

Conclusion (Cont'd)

- ADMM-GBS is criticized for its intensive computation —— It needs to solve m linear systems in every single iteration. When m is large, it could be very slow. Fortunately, we can replace $A_i x_i$ by ξ_i and remove the matrix-vector multiplication. Then the computation can be greatly reduced. This is ADMM-PC.
- ADMM-PC is still not good enough. Actually, ADMM-GBS and ADMM-PC have the following drawbacks:
  1. They need to maintain two groups of variables, the predicted and the corrected, thus increasing the memory cost (at least doubled)
  2. Neither x_i^k (ADMM-GBS) nor ξ^k (ADMM-PC) could be sparse or low-rank, even in sparse or low-rank problems. Thus the memory consumption can be more

Conclusion (Cont'd)

- Thus, we have LADMM-PS, which removes the memory-intensive prediction-correction procedures and update all x_i's in parallel.
- LADMM-PS to $\mathcal{P}_3$ is similar to Method of Multipliers to $\mathcal{P}_2'$. However, as we have analyzed, ADMM is faster than Method of Multipliers, i.e., the alternating update of primal variables is faster than update the primal variables in parallel. This inspires us to combine the alternating update and LADMM-PS in multi-block problems.
- Then we get LADMM-SPU. It devides the m x_i's into two partitions. Each partition is updated in serial while the variables in the same partition are updated in parallel.

Conclusion (Cont'd)

So far, we have discussed several ADMMs for determinisitc linear convex problems. The problem must

1. has a (several) linear constraint (constraints), and
2. does not allowed to have nonlinear constraints (inequality constraints).
3. Further, the objective must be convex.

These conditions must be met for the convergence of ADMMs. If strongly convex or smooth can be met, linear convergence could be achieved.

It's interesting to mention that, all the ADMMs we introduced so far can be derived from the perspective of Variational Inequality. More information can be found from Sec. 3.6 of the ADMM book.

Outline

Nonlinear Convex Problems

Now we consider the following problem $\mathcal{P}_4$, which has two separable blocks and nonlinear constraints.

$$\begin{aligned} \mathcal{P}_4 : \quad & \min_{x,y} f(x) + g(y) \\ s.t. \quad & h_0(x) \leq 0 \\ & p_0(y) \leq 0 \\ & Ax + By = b. \end{aligned}$$

$h_0(x)$ and $p_0(y)$ are scalars, i.e., $h_0(x) \in \mathbb{R}$, $p_0(y) \in \mathbb{R}$. In $\mathcal{P}_4$, we require f, g, h_0, and p_0 to be convex. By introducing $h(x) = \max\{0, h_0(x)\}$ and $p(y) = \max\{0, p_0(y)\}$, we can have the following problem instead:

$$\begin{aligned} \mathcal{P}_4' : \quad & \min_{x,y} f(x) + g(y) \\ s.t. \quad & h(x) = 0, \quad p(y) = 0, \quad Ax + By = b. \end{aligned}$$

ADMM-NC

Obviously, $\mathcal{P}_4'$ is solvable by ADMMs. The augmented Lagrangian function is

$$\begin{aligned} & L_{\rho_1,\rho_2,\beta}(x, y, \gamma, \tau, \lambda) \\ = {} & f(x) + g(y) + \gamma h(x) + \frac{\rho_1}{2} h^2(x) + \tau g(y) + \frac{\rho_2}{2} p^2(y) \\ & + \langle \lambda, Ax + By - b \rangle + \frac{\beta}{2} \|Ax + By - b\|^2. \end{aligned} \tag{143}$$

ADMM-NC

Correspondingly, we have the following iterations:

$$x^{k+1} = \underset{x}{\operatorname{argmin}} L_{\rho_1,\rho_2,\beta}(x, y^k, \gamma^k, \tau^k, \lambda^k), \tag{144}$$

$$y^{k+1} = \underset{y}{\operatorname{argmin}} L_{\rho_1,\rho_2,\beta}(x^{k+1}, y, \gamma^k, \tau^k, \lambda^k), \tag{145}$$

$$\gamma^{k+1} = \gamma^k + \rho_1 h(x^{k+1}), \tag{146}$$

$$\tau^{k+1} = \tau^k + \rho_2 p(y^{k+1}), \tag{147}$$

$$\lambda^{k+1} = \lambda^k + \beta(Ax^{k+1} + By^{k+1} - b). \tag{148}$$

We call (144) $\sim$ (148) *ADMM-NC* (ADMM for Nonlinear Convex problems). Obviously, every convergence result for ADMM is suitable for ADMM-NC.

Any other ADMMs proposed in Sec. II can solve $\mathcal{P}_4$ ($\mathcal{P}_4'$). We will not list them here.

General Nonliner Convex Problems

We extend $\mathcal{P}_4$ to the general nonlinear case:

$$\mathcal{P}_5: \quad \min_x f(x) \equiv \sum_{i=1}^{m} f_i(x) \tag{149}$$

$$s.t. \quad h_i(x) \leq 0, \forall i \in [m], \tag{150}$$

$$Ax = b. \tag{151}$$

$\forall i \in [m], H_i(x)$ are scalars. Note that $\mathcal{P}_5$ is different from $\mathcal{P}_3$, we have only one primal variable x, other than $\{x_i\}_{i\in[m]}$. But we can transform it to the shape of $\mathcal{P}_3$.

In this case, we introduce the auxiliary variables z and $\{x_i\}_{i\in[m]}$, and they satisfy

$$z \equiv x_i, \forall i \in [m]. \tag{152}$$

ADMM-GNC

Then $\mathcal{P}_5$ can be transformed into $\mathcal{P}_5'$:

$$\begin{aligned} \mathcal{P}_5' : \quad & \min_{\{x_i\}_{i\in[m]}, z} \sum_{i=1}^{m} f_i(x_i) && (153) \\ & s.t. \quad h_i(x_i) \leq 0, \forall i \in [m], && (154) \\ & A'z - I'x' = b', && (155) \end{aligned}$$

where (155) is to guarantee that (152) holds and it has the shape:

$$\begin{pmatrix} I \\ I \\ \vdots \\ I \\ A \end{pmatrix} z - \begin{pmatrix} I & 0 & \cdots & 0 \\ 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I \\ 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b \end{pmatrix}. \qquad (156)$$

By solving $\mathcal{P}_5'$, we solve $\mathcal{P}_5$ with $z \equiv x_i \equiv x$.

ADMM-GNC

We can further introduce $H_i(x_i) = \max\{h_i(x), 0\}, \forall i \in [m]$. Then we get $\mathcal{P}_5^{''}$:

$$\mathcal{P}_5^{''}: \quad \min_{\{x_i\}_{i\in[m]}, z} \sum_{i=1}^{m} f_i(x_i) \tag{157}$$

$$s.t. \quad H_i(x_i) = 0, \forall i \in [m], \tag{158}$$

$$A'z - I'x' = b'. \tag{159}$$

For $\mathcal{P}_5^{''}$, the augmented Lagrangian function is:

$$L_{\{\rho_i\}_i,\beta}(x', z, \gamma, \lambda) = \sum_{i=1}^{m} f_i(x_i) + \sum_{i=1}^{m} \gamma_i H_i(x_i) + \sum_{i=1}^{m} \frac{\rho_i}{2} H_i^2(x_i) + \langle \lambda, A'z - I'x' - b \rangle + \frac{\beta}{2} \|A'z - I'x' - b'\|^2,$$

where $\gamma = \{\gamma_i\}_{i\in[m]}$.

ADMM-GNC

We then have the following iterations:

$$(x')^{k+1} = \underset{x'}{\operatorname{argmin}} \, L_{\{\rho_i\}_i,\beta}(x', z^k, \gamma^k, \lambda^k), \tag{160}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} \, L_{\{\rho_i\}_i,\beta}((x')^{k+1}, z, \gamma^k, \lambda^k), \tag{161}$$

$$\gamma_i^{k+1} = \gamma_i^k + \rho_i H_i(x_i^{k+1}), \forall i \in [m], \tag{162}$$

$$\lambda^{k+1} = \lambda^k + \beta(A' z^{k+1} - I'(x')^{k+1} - b'). \tag{163}$$

Wherein, (160) can be solved by solving each x_i in parallel. We name this method *ADMM-GNC* (ADMM for General Nonlinear Convex problems). This method is actually an extension of ADMM-SPU.

Conclusion

- The nonlinear convex optimization problems are transformed to the linear ones when solving them. The technique is introducing auxiliary variables or functions.
- For a speicial case of nonlinear convex problems, i.e., $\mathcal{P}_4$, which has two nonlinear constraints $h_0(x) \leq 0$ and $p_0(y) \leq 0$, ADMM-NC solves it by introducing $h(x) = \max\{0, h_0(x)\}$ and $p(y) = \max\{0, p_0(y)\}$.
- For the general nonconvex problems $\mathcal{P}_5$, we use two transformations, which introduces both slack variables and auxiliary functions. Then, ADMM-GNC is proposed, which is an extension of ADMM-SPU.

Outline

Linear Nonconvex Problems

In the following, we firstly introduce the ADMMs for linear nonconvex problems, since nonlinear problems can be easily transformed to linear ones.

We consider the following multi-block problem $\mathcal{P}_6$:

$$\mathcal{P}_6 : \quad \min_{\{x_i\}_{i\in[m]},y} \sum_{i=1}^{m} f_i(x_i) + g(y) \tag{164}$$

$$s.t. \quad \sum_{i=1}^{m} A_i x_i + By = b, \tag{165}$$

where $x_i \in \mathbb{R}^{d_i}$, $y \in \mathbb{R}^{d_y}$, $A_i \in \mathbb{R}^{n\times d_i}$, $B \in \mathbb{R}^{n\times d_y}$, and $b \in \mathbb{R}^n$. Similarly, we define $x = (x_1^T, ..., x_m^T)^T \in \mathbb{R}^{\sum_i d_i}$, $A = [A_1, ..., A_m] \in \mathbb{R}^{n\times \sum_i d_i}$.

Multi-Block Bregman ADMM

$\mathcal{P}_6$ satisfies the following constraint:

Assumption IV-1

$\forall i \in [m]$, f_i are lower semicontinuous and g is L-smooth.

Note that all f_i's and g could be nonconvex. Since f_i's amd g could be nonconvex, the vanilla ADMM is not suitable for $\mathcal{P}_6$ because the subproblems of x_i of y are difficult to solve. Let's try Bregman ADMM, specifically, LADMM-2, for solving it.

The multi-block Bregman ADMM has the following iterations:

$$\begin{aligned} x_i^{k+1} = \operatorname*{argmin}_{x_i} \Bigg(& f_i(x_i) + \langle \lambda^k, A_i x_i \rangle \\ & + \frac{\beta}{2} \| \sum_{j<i} A_j x_j^{k+1} + A_i x_i + \sum_{j>i} A_j x_j^k + B y^k - b \|^2 \\ & + D_{\phi_i}(x_i, x_i^k) \Bigg). \qquad \textit{(for each } i \in [m] \textit{ in turn)} \end{aligned} \tag{166}$$

LADMM-MB

The multi-block Bregman ADMM has the following iterations (cont'd):

$$y_i^{k+1} = \operatorname*{argmin}_y \Big(g(y) + \langle \lambda^k, By \rangle + \frac{\beta}{2} \| \sum_{i=1}^m A_i x_i^{k+1} + By - b \|^2 + D_{\phi_0}(y, y^k) \Big), \tag{167}$$

$$\lambda^{k+1} = \lambda^k + \beta (\sum_{i=1}^m A_i x_i^{k+1} + By^{k+1} - b). \tag{168}$$

In (166) and (167), $\{D_{\phi_i}\}_{i=0,1,\ldots,m}$ are Bregman distances. We call (166), (167), (168) *LADMM-MB* (Linearized ADMM for Multi-Block problems) [2018-2020].

LADMM-MB

Similar to (90), we consider the following ϕ_0:

$$\phi_0(y) := \frac{L+\beta\|B\|^2}{2}\|y\|^2 - \frac{\beta}{2}\|By\|^2 - g(y), \tag{169}$$

then the y-update reduces to

$$\begin{aligned} y_i^{k+1} &= \operatorname*{argmin}_y \Big(\langle \lambda^k, By\rangle + \langle \nabla g(y^k), y\rangle \\ &\quad + \beta\langle B^T(Ax^{k+1} + By^k - b), y\rangle \\ &\quad + \frac{L+\beta\|B\|^2}{2}\|y-y^k\|^2 \Big) \\ &= y^k - \frac{1}{L+\beta\|B\|^2}\nabla_y L_\beta(x^{k+1}, y, \lambda^k)|_{y=y^k}, \end{aligned} \tag{170}$$

where L_β is the augmented Lagrangian function of $\mathcal{P}_6$.

LADMM-MB

Now let's assume all $\{\phi_i\}_{i=0,1,\dots,m}$ are ρ-strongly convex and L_i-smooth with $\rho > \frac{12(L^2+2L_0^2)}{\sigma^2\beta}$.

From (166), we have: $\forall i \in [m]$,

$$L_\beta(x_{j\le i}^{k+1}, x_{j>i}^k, y^k, \lambda^k) + D_{\phi_i}(x_i^{k+1}, x_i^k) \le L_\beta(x_{j<i}^{k+1}, x_{j\ge i}^k, y^k, \lambda^k). \tag{171}$$

With (14), we then have

$$\frac{\rho}{2}\|x_i^{k+1} - x_i^k\|^2 \le L_\beta(x_{j<i}^{k+1}, x_{j\ge i}^k, y^k, \lambda^k) - L_\beta(x_{j\le i}^{k+1}, x_{j>i}^k, y^k, \lambda^k). \tag{172}$$

Summing over $i = 1, \dots, m$, we have

$$\frac{\rho}{2}\|x^{k+1} - x^k\|^2 \le L_\beta(x^k, y^k, \lambda^k) - L_\beta(x^{k+1}, y^k, \lambda^k). \tag{173}$$

LADMM-MB

Similarly, from (167), we have

$$\frac{\rho}{2}\|y^{k+1}-y^k\|^2 \leq L_\beta(x^{k+1},y^k,\lambda^k)-L_\beta(x^{k+1},y^{k+1},\lambda^k). \quad (174)$$

From (168), we have

$$-\frac{1}{\beta}\|\lambda^{k+1}-\lambda^k\|^2 = L_\beta(x^{k+1},y^{k+1},\lambda^k)-L_\beta(x^{k+1},y^{k+1},\lambda^{k+1}). \quad (175)$$

Summing up the three results, we have

$$\begin{aligned}&\frac{\rho}{2}\|x^{k+1}-x^k\|^2+\frac{\rho}{2}\|y^{k+1}-y^k\|^2-\frac{1}{\beta}\|\lambda^{k+1}-\lambda^k\|^2\\ &\qquad \leq L_\beta(x^k,y^k,\lambda^k)-L_\beta(x^{k+1},y^{k+1},\lambda^{k+1}). \quad (176)\end{aligned}$$

LADMM-MB

On the other hand, from the optimality condition of the y update, we have

$$\begin{aligned} 0 &= \nabla g(y^{k+1}) + \beta B^T\lambda^k + \beta B^T(Ax^{k+1} + By^{k+1} - b) \\ &\qquad + \nabla\phi_0(y^{k+1}) - \nabla\phi_0(y^k) \\ &= \nabla g(y^{k+1}) + B^T\lambda^{k+1} + \nabla\phi_0(y^{k+1}) - \nabla\phi_0(y^k). \end{aligned} \tag{177}$$

Now, we further assume that the surjectiveness of B holds, i.e., there exists $\sigma > 0$ such that $\|B^T\lambda\| \geq \sigma\|\lambda\|$ for all λ. It means that B needs to be fully rank. Then we have

$$\begin{aligned} \sigma^2\|\lambda^{k+1} - \lambda^k\|^2 &\leq \|B^T(\lambda^{k+1} - \lambda^k)\|^2 \\ &= \|\nabla g(y^{k+1}) - \nabla g(y^k) + \nabla\phi_0(y^{k+1}) - \nabla\phi_0(y^k) \\ &\quad - (\nabla\phi_0(y^k) - \nabla\phi_0(y^{k-1}))\|^2. \end{aligned} \tag{178}$$

LADMM-MB

The RHS of (178) further satisfies

$$\begin{aligned} RHS &\le 3\|\nabla g(y^{k+1}) - \nabla g(y^k)\|^2 + 3\|\nabla \phi_0(y^{k+1}) - \nabla \phi_0(y^k)\| \\ &\quad + 3\|\nabla \phi_0(y^k) - \nabla \phi_0(y^{k-1})\|^2 \\ &\le 3(L^2 + L_0^2)\|y^{k+1} - y^k\|^2 + 3L_0^2\|y^k - y^{k-1}\|^2, \end{aligned} \tag{179}$$

where we use the definition of smooth functions. Combing (176) and (179), we have

$$\begin{aligned} &\frac{\rho}{2}\|x^{k+1} - x^k\|^2 + \frac{\rho}{2}\|y^{k+1} - y^k\|^2 + \frac{1}{\beta}\|\lambda^{k+1} - \lambda^k\|^2 \\ &\qquad \le L_\beta(x^k, y^k, \lambda^k) - L_\beta(x^{k+1}, y^{k+1}, \lambda^{k+1}). \\ &\quad + \frac{6(L^2 + L_0^2)}{\sigma^2\beta}\|y^{k+1} - y^k\|^2 + \frac{6L_0^2}{\sigma^2\beta}\|y^k - y^{k-1}\|^2. \end{aligned} \tag{180}$$

LADMM-MB

Defining

$$\Phi^k := L_\beta(x^k, y^k, \lambda^k) + \frac{6L_0^2}{\sigma^2\beta}\|y^k - y^{k-1}\|^2, \tag{181}$$

we have

$$\begin{aligned}\frac{\rho}{2}\|x^{k+1} - x^k\|^2 + \left(\frac{\rho}{2} - \frac{6L^2 + 12L_0^2}{\sigma^2\beta}\right)\|y^{k+1} - y^k\|^2 \\ + \frac{1}{\beta}\|\lambda^{k+1} - \lambda^k\|^2 \le \Phi^k - \Phi^{k+1}.\end{aligned} \tag{182}$$

If we further assume that $\{x^k, y^k, \lambda^k\}_k$ are bounded and $\sum_i f_i(x_i) + g(y)$ is bounded below, we know that Φ^k is lower bounded by some Φ^*.

LADMM-MB

Summing the above inequality over $k = 0, \ldots, K$, we have

$$\min_{k\leq K}\left\{\frac{\rho}{2}\|x^{k+1}-x^k\|^2+\left(\frac{\rho}{2}-\frac{6L^2+12L_0^2}{\sigma^2\beta}\right)\|y^{k+1}-y^k\|^2 + \frac{1}{\beta}\|\lambda^{k+1}-\lambda^k\|^2\right\} \leq \frac{\Phi^0-\Phi^*}{K+1}. \tag{183}$$

The result means that LADMM-MB needs $\mathcal{O}(1/\epsilon^2)$ iterations to find $(x^{k+1}, y^{k+1}, \lambda^{k+1})$ such that

$$\|x^{k+1}-x^k\| \leq \epsilon, \quad \|y^{k+1}-y^k\| \leq \epsilon, \quad \|\lambda^{k+1}-\lambda^k\| \leq \epsilon. \tag{184}$$

LADMM-MB

From (178) we have

$$\begin{aligned}\|\nabla g(y^{k+1}) + B^T\lambda^{k+1}\| &= \|\nabla\phi_0(y^{k+1}) - \nabla\phi_0(y^k)\| \\ &\leq L_0\|y^{k+1} - y^k\| \leq \mathcal{O}(\epsilon). \qquad (185)\end{aligned}$$

Further, from the update of λ we have

$$\|Ax^{k+1} + By^{k+1} - b\| = \frac{1}{\beta}\|\lambda^{k+1} - \lambda^k\| \leq \mathcal{O}(\epsilon). \qquad (186)$$

Further, from the optimality of x_i, we have that there exists $\tilde{\nabla} f_i(x_i^{k+1}) \in \partial f_i(x_i^{k+1})$ such that

$$\begin{aligned}\tilde{\nabla} f_i(x_i^{k+1}) + A_i^T\Big[\lambda^k + \beta\Big(\sum_{j\leq i} A_j x_j^{k+1} + \sum_{j>i} A_j x_j^k + By - b\Big)\Big] \\ + \nabla\phi_i(x_i^{k+1}) - \nabla\phi_i(x_i^k) = 0. \qquad (187)\end{aligned}$$

LADMM-MB

Therefore we have

$$\begin{aligned}&\|\tilde{\nabla} f_i(x_i^{k+1}) + A_i^T\lambda^{k+1}\| \leq \|\nabla\phi_i(x_i^{k+1}) - \nabla\phi_i(x_i^k)\|\\ &+\beta\left\|A_i^T\left(\sum_{j>i} A_j(x_j^{k+1} - x_j^k) + B(y^{k+1} - y^k)\right)\right\| \leq \mathcal{O}(\epsilon). \quad (188)\end{aligned}$$

The above analysis shows that, under some conditions, LADMM-MB needs $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations to find an ϵ-approximate KKT point. The result is formally presented in Theorem IV-1.

LADMM-MB

The following proof was refined in 2020.

Theorem IV-1 (Convergence Rate of LADMM-MB)

Assume that Assumption IV-1 and the surjectiveness of B holds (there exists $\sigma > 0$ such that $\|B^T\lambda\| \geq \sigma\|\lambda\|$ for all λ, it means that B needs to be fully rank), and ϕ_i is ρ-strongly convex and L_i-smooth with

$$\rho > \frac{12(L^2 + 2L_0^2)}{\sigma^2\beta}, i = 0, ..., m. \tag{189}$$

Suppose that the sequence $\{(x^k, y^k, \lambda^k)\}_k$ is bounded and $\sum_{i=1}^m f_i(x_i) + g(y)$ is bounded below with bounded (x, y). Then LADMM-MB needs $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations to find an ϵ-approximate KKT point $(x^{k+1}, y^{k+1}, \lambda^{k+1})$.

LADMM-MB

Theorem IV-1 (cont'd) (Convergence Rate of LADMM-MB)

Namely,

$$\Big\| \sum_{i=1}^{m} A_i x_i^{k+1} + By^{k+1} - b \Big\| \leq \mathcal{O}(\epsilon), \tag{190}$$

$$\|\nabla g(y^{k+1}) + B^T \lambda^{k+1}\| \leq \mathcal{O}(\epsilon), \tag{191}$$

$$dist\big(-A_i^T \lambda^{k+1}, \partial f_i(x_i^{k+1})\big) \leq \mathcal{O}(\epsilon), \forall i \in [m]. \tag{192}$$

In this theorem, we assume that the objectives are proper and lower semicontinuous such that the *subdifferential* is well defined (Otherwise (171) may not hold). We assume that the objective is *coercive* over the entire space.

LADMM-MB

In the proof of Theorem IV-1, a crucial step is to *bound the dual variables by the primal ones*, i.e., (178) and (179):

$$\sigma^2\|\lambda^{k+1}-\lambda^k\|^2 \le 3(L^2+L_0^2)\|y^{k+1}-y^k\|^2 + 3L_0^2\|y^k-y^{k-1}\|^2,$$

which is established via the surjectiveness assumption:

$$\|B^T\lambda\| \ge \sigma\|\lambda\|, \forall \lambda. \tag{193}$$

Nevertheless, we can replace it by a weaker assumption:

$$\mathrm{Im}(A_i) \subseteq \mathrm{Im}(B), \forall i \in [m]. \tag{194}$$

In this case, we have

$$\lambda^{k+1}-\lambda^k = \beta(Ax^{k+1}+By-b) \in \mathrm{Im}(B). \tag{195}$$

LADMM-MB

Suppose that the SVD of B is $B = U\Sigma V^T$. Then we may write $\lambda^{k+1} - \lambda^k = U\alpha$ for some α. Further we have

$$\begin{aligned} \|B^T(\lambda^{k+1} - \lambda^k)\|^2 &= \|V\Sigma U^T U\alpha\|^2 = \|\Sigma\alpha\|^2 \\ &\geq \lambda_+(BB^T)\|\alpha\|^2 = \lambda_+(BB^T)\|\lambda^{k+1} - \lambda^k\|^2, \end{aligned} \tag{196}$$

where $\lambda_+(BB^T)$ is the smallest strictly positive eigenvalue of BB^T. Then we get (193). In other words, *with a weaker assumption $Im(A_i) \subseteq Im(B)$, we can still get the intermediate results* (178) *and* (179). In this case, we do not require B to be fully rank any more.

Besides, it's easy to check that when the problem $\mathcal{P}_6$ has only one block, the nwe assumption can be removed because (194) always holds.

The Vanilla ADMM for $\mathcal{P}_6$

LADMM-MB is actually an application of Bregman ADMM. What if we remove the Bregman distance from the subproblems (166) and (167)? In this case LADMM-MB reduces to the vanilla ADMM. If we want the vanilla ADMM works fine for solving $\mathcal{P}_6$, we need to further assume that

1. $\forall i \in [m]$, the smallest eigenvalue of $A_i^T A_i$ is positive, and
2. β should be chosen large enough

such that $L_\beta(x, y, \lambda)$ is μ-strongly convex w.r.t. x_i (assume that f_i is smooth if necessary). Then for the first step, we have

$$L_\beta(x_{j<i}^{k+1}, x_{j\geq i}^k, y^k, \lambda^k) - L_\beta(x_{j\leq i}^{k+1}, x_{j>i}^k, y^k, \lambda^k) \geq \frac{\mu}{2}\|x_i^{k+1} - x_i^k\|^2,$$

and the proof can continue.

LADMM-MB

Recall that LADMM-MB is $\mathcal{O}(1/\epsilon^2)$ when solving $\mathcal{P}_6$ under the condition that

1. Assumption IV-1 holds, and
2. the surjectiveness of B holds (or $\mathrm{Im}(A_i) \subseteq \mathrm{Im}(B), \forall i \in [m]$).

If we let LADMM-MB works under the following new assumption:

Assumption IV-2

All f_i's and g are L-smooth.

In other words, we have more assumptions on the functions f_i and g, other than A_i and B. Then the $\mathcal{O}(1/\epsilon^2)$-complexity can still be achieved. The proof details is omitted.

PADMM-EA

In the y-update of LADMM-MB (formula (170)), the Bregman distance essentially results in adding a proximal term, i.e., $\frac{\beta'}{2}\|y - y^k\|^2$, to the augmented Lagrangian function. What if we use another proximal term?

In the following, we introduce *PADMM-EA* (Proximal ADMM with Exponential Averaging) [SIAM, 2020], which uses $\frac{\beta'}{2}\|y - z^k\|^2$ instead, in which z^k is an exponential averaging of

$$y^0, y^1, \ldots, y^k.$$

PADMM-EA

Now we consider a more general form of $\mathcal{P}_6$, which is named as $\mathcal{P}_7$:

$$\mathcal{P}_7: \quad \min_{\{x_i\}_{i\in[m]}} f(x_1, \dots, x_m) \tag{197}$$

$$s.t. \quad \sum_{i=1}^{m} A_i x_i = b, \tag{198}$$

with a **non-separable** objective. Equally, denote

$$x = (x_1^T, \dots, x_m^T)^T, \quad A = [A_1, \dots, A_m].$$

Define the following proximal augmented Lagrangian function

$$P(x, y, \lambda) = f(x) + \langle \lambda, Ax - b\rangle + \frac{\beta}{2}\|Ax - b\|^2 + \frac{\rho}{2}\|x - z\|^2. \tag{199}$$

PADMM-EA

PADMM-EA consists of the following iterations:

for $j \in [m]$ in turn:

$$x_j^{k+1} = x_j^k - \alpha_1 \nabla_{x_j} P\Big(\{x_i^{k+1}\}_{1 \le i \le j-1}, x_j, \{x_i^k\}_{j+1 \le i \le m}, z^k, \lambda^k\Big), \tag{200}$$

$$\lambda^{k+1} = \lambda^k + \alpha_2(Ax^{k+1} - b), \tag{201}$$

$$z^{k+1} = z^k + \alpha_3(x^{k+1} - z^k). \tag{202}$$

Fomr (202) we have

$$z^{k+1} = \sum_{t=0}^{k} \alpha_3(1-\alpha_3)^{k-t} x^{t+1} + (1-\alpha_3)^{k+1} z^0. \tag{203}$$

In every iteration k, we do not want x^{k+1} to deviate too much from z^k.

PADMM-EA

When f is L-smooth w.r.t. x, if we choose α_1, α_2, and α_3 appropriately and letting $\rho > L$, then PADMM-EA is of $\mathcal{O}(1/\epsilon^2)$ complexity.

The proof is complicate but similar to the proof of Theorem IV-1. The key idea is still to prove that

$$\min_{k=0,1,\ldots,K} \Big\{ \|x^{k+1} - x^k\| + \|z^{k+1} - z^k\| \\ + \|Ax(z^k, \lambda^k) - b\| \Big\} \le \mathcal{O}(\epsilon), \tag{204}$$

where $x(z^k, \lambda^k) = \operatorname{argmin}_x P(x, z^k, \lambda^k)$.

I may update the proof details in future :-).

Conclusion

Time to make a conclusion.

- Our study on nonconvex optimization problems is started from the multi-block form $\mathcal{P}_6$, in which all f_i's and g can be nonconvex. However, to guarantee the convergence, we have a global assumption —— All these functions are lower-semicontinuous and g is L-smooth.
- We firstly show LADMM-MB, which is a variant of Bregman ADMM. Then, we use a lot of space to show that under the global assumption, the strongly convexity and smooth property on the Bregman distance functions, and
  - the surjectiveness of B, or
  - $\mathrm{Im}(A_i) \subseteq \mathrm{Im}(B), \forall i \in [m]$,

  LADMM-MB can achieve a $\mathcal{O}(1/\epsilon^2)$ rate.
- If all f_i's and g are L-smooth, then without the above requirements, LADMM-MB can still achieve a $\mathcal{O}(1/\epsilon^2)$ rate.

Conclusion (Cont'd)

- The reason we use Bregman ADMM to solve $\mathcal{P}_6$, rather than the vanilla ADMM, is that, the objective is nonconvex, and thus the subproblems are difficult to solve. However, if we require that
  1. Assumption IV-1 holds, and
  2. the surjectiveness of B holds (or $\mathrm{Im}(A_i) \subseteq \mathrm{Im}(B), \forall i \in [m]$),

  The vanilla ADMM can also achieve a $\mathcal{O}(1/\epsilon^2)$ rate.
- We then introduce a recently developed method, called PADMM-EA. It adopts both proximal method and moving average to solve the nonconvex problems. PADMM-EA can achieve a $\mathcal{O}(1/\epsilon^2)$ rate.

Outline

Stochastic Problems

We consider the following linearly constrained two-block optimization problem $\mathcal{P}_8$:

$$\mathcal{P}_8 : \quad \min_{x_1, x_2} f_1(x_1) + f_2(x_2) \tag{205}$$

$$s.t. \quad A_1 x_1 + A_2 x_2 = b, \tag{206}$$

where $f_1(x_1)$ has the following structure:

$$f_1(x_1) := \mathbb{E}_\xi \left[F(x_1; \xi) \right], \tag{207}$$

where $F(x_1; \xi)$ is a stochastic component indexed by a random variable ξ. $f_1(x_1)$ can also have the following structure:

$$f_1(x_1) := \frac{1}{n} \sum_{i=1}^{n} F_i(x_i). \tag{208}$$

We name $\mathcal{P}_8$ with (207) $\mathcal{P}_8'$, and $\mathcal{P}_8$ with (208) $\mathcal{P}_8''$. For example, $f_1(x_1)$ is a training loss on data and $f_2(x_2)$ is a regularizer.

SADMM

We firstly consider $\mathcal{P}_8'$. *In each iteration, we independently sample a stochastic index* ξ *and compute the stochastic gradient* $\nabla F(x_1;\xi)$. We also denote $\nabla F(x_1;\xi)$ by $\tilde{\nabla} f_1(x_1)$ for simplification.

Now we introduce SADMM (Stochastic ADMM) [ICML, 2013]. It is built on the following approximated augmented Lagrangian function $\hat{L}_\beta^k$:

$$\hat{L}_\beta^k(x_1, x_2, \lambda) := \underbrace{f_1(x_1^k) + \langle \tilde{\nabla} f_1(x_1), x_1 - x_1^k \rangle}_{\text{linearization of } f_1} + f_2(x_2) + \frac{\beta}{2}\|A_1x_1 + A_2x_2 - b + \frac{1}{\beta}\lambda\|^2 + \underbrace{\frac{1}{2\eta_{k+1}}\|x_1 - x_1^k\|^2}_{\text{the proximal term}}. \tag{209}$$

SADMM

SADMM works as follows.

$$x_1^{k+1} = \operatorname*{argmin}_{x_1} \hat{L}_\beta^k(x_1, x_2^k, \lambda^k), \tag{210}$$

$$x_2^{k+1} = \operatorname*{argmin}_{x_2} \hat{L}_\beta^k(x_1^{k+1}, x_2, \lambda^k), , \tag{211}$$

$$\lambda^{k+1} = \lambda^k + \beta(A_1 x_1^{k+1} + A_2 x_2^{k+1} - b). \tag{212}$$

Although not easy, we can prove that SADMM can only achieve $\mathcal{O}(1/\sqrt{K})$ rate. Compared with the $\mathcal{O}(1/K)$-rate of ADMM for $\mathcal{P}_2'$, SADMM is much slower. The result is shown in the following theorem.

Theorem V-1 ($\mathcal{O}(1/\sqrt{K})$ rate of SADMM)

Suppose that f_1 is μ-strongly convex and L-smooth, and f_2 is convex. Assume that the variance of f_1's gradient is uniformly bounded by σ^2, i.e.,

$$\mathbb{E}_\xi\left[\|\nabla F_1(x_1;\xi) - \nabla f_1(x_1)\|^2\right] \le \sigma^2, \forall x_1. \tag{213}$$

SADMM

Theorem V-1 (cont'd) ($\mathcal{O}(1/\sqrt{K})$ rate of SADMM)

Define $D_1 = \|x_1^0 - x_1^*\|$, *and* $D_2 = \|A_2 x_2^0 - A_2 x_2^*\|$.
For the generally convex case, i.e., $\mu = 0$, *set the stepsize* $\eta_k = \frac{1}{2L + \sqrt{k}\sigma/D_1}$,

$$\bar{x}_1^K = \frac{1}{\sum_{k=1}^K \eta_k} \sum_{k=1}^K \eta_k x_1^k, \text{ and } \bar{x}_2^K = \frac{1}{\sum_{k=1}^K \eta_k} \sum_{k=1}^K \eta_k x_2^k,$$

then for any $\rho > 0$ *and sufficiently large* K, *we have*

$$\begin{aligned}&\mathbb{E}[f_1(\bar{x}_1^K)] + \mathbb{E}[f_2(\bar{x}_2^K)] - f_1(x^*) - f_2(x^*) + \rho\mathbb{E}\left[A_1\bar{x}_1^K + A_2\bar{x}_2^K - b\right] \\ &\le \frac{2D_1\sigma\log K}{\sqrt{K}} + \frac{\sigma}{\sqrt{K}}\left[\frac{D_1}{2} + \frac{\rho^2}{2\beta(2LD_1+\sigma)} + \frac{\beta D_2^2}{2(2LD_1+\sigma)}\right].\end{aligned} \tag{214}$$

SADMM

Theorem V-1 (cont'd) ($\mathcal{O}(1/\sqrt{K})$ rate of SADMM)

For the strongly convex case, i.e., $\mu > 0$, set the stepsize $\eta_k = \frac{1}{2L+k\mu}$,

$$\bar{x}_1^K = \frac{1}{K}\sum_{k=1}^{K} x_1^k, \text{ and } \bar{x}_2^K = \frac{1}{K}\sum_{k=1}^{K} x_2^k,$$

then for any $\rho > 0$, we have

$$\begin{aligned}\mathbb{E}[f_1(\bar{x}_1^K)] + \mathbb{E}[f_2(\bar{x}_2^K)] - f_1(x^*) - f_2(x^*) + \rho\mathbb{E}\left[A_1\bar{x}_1^K + A_2\bar{x}_2^K - b\right] \\ \leq \frac{\sigma^2(\log K + 1)}{\mu K} + \frac{1}{K}\left(LD_1^2 + \frac{\rho^2}{2\beta} + \frac{\beta D_2^2}{2}\right).\end{aligned} \tag{215}$$

SVRG-ADMM

SADMM is criticized for its $\mathcal{O}(1/\sqrt{K})$ rate. Nevertheless, it can be accelerated by the variance reduction (VR) technique. *VR is to frequently pre-store a snapshot vector and to control the variance via the snapshot vector and the latest iterate.*

Specifically, we consider the following iterations to solve $\mathcal{P}_8''$:

$$x_{s,1}^{k+1} = \underset{x_1}{\operatorname{argmin}} \Bigg(\underbrace{\langle \tilde{\nabla} f_1(x_{s,1}^k), x_1 - x_{s,1}^k \rangle + f_1(x_{s,1}^k)}_{\textit{linearization of } f_1(x_1)} + \underbrace{\langle \beta(A_1 x_{s,1}^k + A_2 x_{s,2}^k - b) + \lambda_s^k, A_1(x_1 - x_{s,1}^k) \rangle}_{\textit{linearization of the aug. term, constant removed}} + \underbrace{\frac{1}{2\eta_1} \|x_1 - x_{s,1}^k\|^2}_{\textit{the proximal term}} \Bigg) \tag{216}$$

SVRG-ADMM

We consider the following iterations (cont'd):

$$
\begin{aligned}
x_{s,2}^{k+1} &= \operatorname*{argmin}_{x_2} \Big(f_2(x_2) \\
&\quad + \langle \beta(A_1 x_{s,1}^{k+1} + A_2 x_{s,2}^{k} - b) + \lambda_s^k, A_2(x_2 - x_{s,2}^k) \rangle \\
&\quad + \frac{1}{2\eta_2} \|x_2 - x_{s,2}^k\|^2 \Big), \qquad (217) \\
\lambda_s^{k+1} &= \lambda_s^k + \beta(A_1 x_{s,1}^{k+1} + A_2 x_{s,2}^{k+1} - b), \qquad (218)
\end{aligned}
$$

where

$$\eta_1 = \frac{1}{9L + \beta\|A_1\|^2} \text{ and } \eta_2 = \frac{1}{\beta\|A_2\|^2}.$$

SVRG-ADMM

The algorithm is presented below:

```
1. Initialize $x^0_{0,1}, x^0_{0,2}, \lambda^0_0$, and $\tilde{x}_{0,1} \leftarrow x^0_{0,1}$
2. Set epoch length $m$ and stepsize $\eta$
3. for $s = 0, ..., S-1$ do
    3.1 for $k = 0, ..., m-1$ do
        3.1.1 Randomly sample $i_{k,s}$ from $[n]$
        3.1.2 Estimate the gradient at the sample $i_{k,s}$:
              $$\tilde{\nabla} f_1(x^k_{s,1}) = \nabla F_{i_{k,s}}(x^k_{s,1}) - \nabla F_{i_{k,s}}(\tilde{x}_{s,1}) + \frac{1}{n}\sum_{i=1}^{n} \nabla F_i(\tilde{x}_{s,1}) \tag{219}$$
        3.1.3 Update $x^{k+1}_{s,1}$ by (216)
        3.1.4 Update $x^{k+1}_{s,2}$ by (217)
        3.1.5 Update $\lambda^{k+1}_s$ by (218)
    3.2 Update: $\tilde{x}_{s+1,i} = \frac{1}{m}\sum_{k=1}^{m} x^k_{s,i}$, $x^0_{s+1,i} = x^m_{s,i}$, $i = 1, 2$, and
        $\lambda^0_{s+1} = \lambda^m_s$
```

SVRG-ADMM

The algorithm above is called *SVRG-ADMM* [NeurIPS, 2013; IJCAI, 2016]. The main step to reduce the variance is (219), where $\tilde{x}_{s,1}$ is the snapshot vector.

Theorem V-2 ($\mathcal{O}(1/S)$ rate of SVRG-ADMM)

Suppose that F_i is convex and L-smooth for $i \in [n]$ and f_2 is convex. Then SVRG-ADMM is eable to achieve $\mathcal{O}(\frac{1}{S})$ rate.

The result shows that SVRG-ADMM is able to achieve linear convergence rate. The proof details are omitted here.

SADMM-MA

Either SADMM or SVRG-ADMM are transformed to act like a determinisitc algorithm through sampling. It's interesting to fuse them with *Momentum Acceleration*.

Momentum Acceleration works as follows —— the move stepsize of x is not only determined by current (sub)gradient, but also whether the gradients in the past iterations are consistent in each dimension. More details about this can be found here.

SADMM-MA

Specifically, we consider the following problem, which is a variant of $\mathcal{P}_8$:

$$\mathcal{P}_8''' : \quad \min_{x_1,x_2} \left(h_1(x_1) + f_1(x_1) + h_2(x_2) + \frac{1}{n}\sum_{i=1}^{n} F_{2,i}(x_2) \right) \tag{220}$$

$$s.t. \quad A_1x_1 + A_2x_2 = b, \tag{221}$$

where $f_1(x_1)$ and $F_{2,i}(x_2)$ with $i \in [n]$ are convex and L_1-smooth and L_2-smooth, respectively, and $h_1(x_1)$ and $h_2(x_2)$ are also convex and their proximal mappings can be solved efficiently. We define

$$f_2(x_2) = \frac{1}{n}\sum_{i=1}^{n} F_{2,i}(x_2),$$
$$\mathcal{J}_1(x_1) = h_1(x_1) + f_1(x_1), \mathcal{J}_2(x_2) = h_2(x_2) + f_2(x_2),$$
$$x = (x_1^T, x_2^T)^T, A = [A_1, A_2], \text{ and } \mathcal{J}(x) = \mathcal{J}_1(x_1) + \mathcal{J}_2(x_2).$$

SADMM-MA

We then consider the following iterations:

$$x_{s,1}^{k+1} = \operatorname*{argmin}_{x_1} \Bigg(\underbrace{h_1(x_1)}_{\circledast} + \underbrace{\langle \nabla f_1(y_{s,1}^k), x_1 \rangle}_{\circledcirc} + \underbrace{\langle \frac{\beta}{\theta_{1,s}}(A_1 y_{s,1}^k + A_2 y_{s,2}^k - b) + \lambda_s^k, A_1 x_1 \rangle}_{\odot} + \underbrace{\left(\frac{L_1 + \frac{\beta}{\theta_{1,s}}\|A_1\|^2}{2} \right) \|x_1 - y_{s,1}^k\|^2}_{\otimes} \Bigg), \tag{222}$$

where $\theta_{1,s}$ will be used in momentum, and $y_{s,1}^k$ is the extrapolation term of x_1, which will be introduced later. $\circledcirc$ is the linearization of f_1 at $y_{s,1}^k$ (constant removed). $\odot$ is the linearization of the augmented term $\frac{\beta}{2}\|A_1 x_1 + A_2 x_2 - b + \frac{1}{\beta}\lambda\|^2$ (constant removed).

SADMM-MA

(222) is similar to (91). Besides, $\circledast + \otimes$ is the proximal mapping of h_1 at $y_{s,1}^k$. For the update of x_2, we have:

$$\begin{aligned} x_{s,2}^{k+1} = \operatorname*{argmin}_{x_2} \Bigg(& h_2(x_2) + \langle \tilde{\nabla} f_2(y_{s,2}^k), x_2 \rangle \\ & + \langle \frac{\beta}{\theta_{1,s}}(A_1 x_{s,1}^{k+1} + A_2 y_{s,2}^k - b) + \lambda_s^k, A_2 x_2 \rangle \\ & + \left[\frac{(1 + \frac{1}{\varpi\theta_2})L_2 + \frac{\beta}{\theta_{1,s}}\|A_2\|^2}{2} \right] \|x_2 - y_{s,2}^k\|^2 \Bigg), \end{aligned} \tag{223}$$

where θ_2 where be used in momentum, and $y_{s,2}^k$ is the extrapolation term of x_2, and

$$\tilde{\nabla} f_2(y_{s,2}^k) = \frac{1}{\varpi} \sum_{i_{k,s} \in \mathcal{I}_{k,s}} \left(\nabla F_{2,i_{k,s}}(y_{s,2}^k) - \nabla F_{2,i_{k,s}}(\tilde{x}_{s,2}) + \nabla f_2(\tilde{x}_{s,2}) \right),$$

in which ϖ is the mini-batch size of indices sampled, which will be introduced later.

SADMM-MA

Similar to SVRG-ADMM, this new method has two loops. We firstly demonstrate the inner loop:

```
1. for $k = 0, \ldots, m-1$ do
    1.1 Update dual variable:
        $$\lambda_s^k = \tilde{\lambda}_s^k + \frac{\beta\theta_2}{\theta_{1,s}}(A_1 x_{s,1}^k + A_2 x_{s,2}^k - \tilde{b}_s) \quad (224)$$
    1.2 Update $x_{s,1}^{k+1}$ by (222)
    1.3 Update $x_{s,2}^{k+1}$ by (223)
    1.4 Update dual snapshot variable:
        $$\tilde{\lambda}_s^{k+1} = \lambda_s^k + \beta(A_1 x_{s,1}^{k+1} + A_2 x_{s,2}^{k+1} - b) \quad (225)$$
    1.5 Update the extrapolation term $y_s^{k+1}$:
        $$y_s^{k+1} = x_s^{k+1} + (1 - \theta_{1,s} - \theta_2)(x_s^{k+1} - x_s^k) \quad (226)$$
```

SADMM-MA

We then demonstrate the outer loop:

1. *Initialize* $x_0^0 = 0, \tilde{b}_0 = 0, \tilde{\lambda}_0^0 = 0, \tilde{x}_0 = x_0^0, y_0^0 = x_0^0$
2. *Set* $\beta, \tau = 2, c = 2, \theta_{1,s} = \frac{1}{c+\tau s}, \theta_2 = \frac{m-\tau}{\tau(m-1)}$
3. **for** $s = 0, \ldots, S-1$ **do**
   - 3.1 *Do inner loop demonstrated above*
   - 3.2 *Set primal variables* $x_{s+1}^0 = x_s^m$
   - 3.3 *Update primal snapshot variable* $\tilde{x}_{s+1}$*:*

$$\tilde{x}_{s+1} = \frac{1}{m}\Big([1 - \frac{(\tau-1)\theta_{1,s+1}}{\theta_2}]x_s^m + [1 + \frac{(\tau-1)\theta_{1,s+1}}{(m-1)\theta_2}]\sum_{k=1}^{m-1} x_s^k\Big)$$

   - 3.4 *Update dual snapshot variable:*

$$\tilde{\lambda}_{s+1}^0 = \lambda_s^{m-1} + \beta(1-\tau)(A_1 x_{s,1}^m + A_2 x_{s,2}^m - b) \tag{227}$$

$$\tilde{b}_{s+1} = A_1 \tilde{x}_{s+1,1} + A_2 \tilde{x}_{s+1,2} \tag{228}$$

   - 3.5 *Update the extrapolation term:*

$$y_{s+1}^0 = (1-\theta_2)x_s^m + \theta_2 \tilde{x}_{s+1} + \frac{\theta_{1,s+1}}{\theta_{1,s}}\Big[(1-\theta_{1,s})x_s^m - (1-\theta_{1,s}-\theta_2)x_s^{m-1} - \theta_2 \tilde{x}_s\Big] \tag{229}$$

SADMM-MA

The above algorithm is called *SADMM-MA* [NeurIPS, 2017]. We can find that in the outer loop, it maintains several snapshot vectors $\tilde{x}_s$, $\tilde{b}_s$ and the extrapolation terms y_s^0. In the inner loop, the primal variables x_s^k and the snapshot dual variable $\tilde{\lambda}_s^k$ are updated.

The algorithm is complicated. It is proved to have a $\mathcal{O}(1/S)$ rate, which is linear. It runs faster than SVRG-ADMM since it's a fuse of SVRG-ADMM and the Momentum Acceleration technique.

SADMM-NC

In the following, we study the ADMMs for nonconvex stochastic problems. We consider $\mathcal{P}_8'$ under the following assumption:

Assumption V-1

f_1 and f_2 are L_1-smooth and L_2-smooth, respectively. Moreover, the variance of stochastic gradients for f is uniformly bounded by σ^2, i.e.,

$$\mathbb{E}_\xi\left[\|\nabla F_1(x_1;\xi) - \nabla f_1(x_1)\|^2\right] \leq \sigma^2, \forall x_1. \tag{230}$$

Note that both f_1 and f_2 can be nonconvex.

SADMM-NC

We name the following algorithm *SADMM-NC*:

$$x_1^{k+1} = x_1^k - \eta \Bigg(\underbrace{\tilde{\nabla} f(x_1^k) + \beta A_1^T \left({}_1 x_1^k + A_2 x_2^k - b + \frac{\lambda^k}{\beta} \right)}_{\textit{the grad. of (209), proximal term removed}} \Bigg), \tag{231}$$

$$\begin{aligned} x_2^{k+1} &= \operatorname*{argmin}_{x_2} \Big(f_2(x_2) + \langle \lambda^k, A_2 x_2 \rangle \\ &\qquad + \frac{\beta}{2} \|A_1 x_1^{k+1} + A_2 x_2 - b\|^2 + D_\phi(x_2, x_2^k) \Big), \qquad (232) \\ \lambda^{k+1} &= \lambda^k + \beta(A_1 x_1^{k+1} + A_2 x_2^{k+1} - b), \qquad (233) \end{aligned}$$

where

$$\tilde{\nabla} f(x^k) = \frac{1}{\varpi} \sum_{\xi \in \mathcal{I}_k} \nabla F(x^k, \xi) \tag{234}$$

is the mini-batch stochastic estimator of $\nabla f(x^k)$.

SPIDER-ADMM

Unfortunately, SADMM-NC can only achieve $\mathcal{O}(1/\epsilon^4)$ rate under Assumption V-1 and the surjectiveness of A_2. It's very slow.

The Stochastic Path-Integrated Differential Estimator (SPIDER) technique [ICML, 2017, 2019; NeurIPS, 2018] is a radical VR method that is used to *track quantities using reduced stochastic oracles*. In the following, we show how to accelerate SADMM-NC with the SPIDER technique.

SPIDER-ADMM

Now we consider a more general multi-block linearly constrained nonconvex problem $\mathcal{P}_9$:

$$\mathcal{P}_9: \quad \min_{\{x_i\}_{i\in[m]},y} \sum_{i=1}^{m} f_i(x_i) + g(y) \tag{235}$$

$$s.t. \quad \sum_{i=1}^{m} A_i x_i + By = b, \tag{236}$$

where $f_i(x_i) = \mathbb{E}_{\xi_i}[F_i(x_i;\xi_i)]$ for $i \in [m]$, under the following assumption:

Assumption V-2

g is L_0-smooth. For each $i \in [m]$, $F_i(x_i;\xi_i)$ is L_i-smooth w.r.t. x_i for all ξ_i. Moreover, the variance of stochastic gradients of f_i is uniformly bounded by σ^2, i.e.,

$$\mathbb{E}_{\xi_i}\left[\|\nabla F_i(x_i;\xi_i) - \nabla f_i(x_i)\|^2\right] \le \sigma^2, \forall i \in [m]. \tag{237}$$

SPIDER-ADMM

We use the following iterations to solve it:

$$\textit{for each } i \in [m] \textit{ in turn:} \tag{238}$$

$$x_i^{k+1} = x_i^k - \eta\Big[\tilde{\nabla} f_i(x_i^k)$$

$$+ \beta A_i^T\Big(\sum_{j<i} A_j x_j^{k+1} + \sum_{j\geq i} A_j x_j^k + By^k - b + \frac{\lambda^k}{\beta}\Big)\Big] \tag{239}$$

$$y^{k+1} = \operatorname*{argmin}_y \Big(g(y) + \langle \lambda^k, By\rangle$$

$$+ \frac{\beta}{2}\|\sum_i A_i x_i^{k+1} + By - b\|^2 + D_\phi(y, y^k)\Big) \tag{240}$$

$$\lambda^{k+1} = \lambda^k + \beta(\sum_i A_i x_i^{k+1} + By^{k+1} - b). \tag{241}$$

This method is called *SPIDER-ADMM*. It actually has the same shape with SADMM-NC.

SPIDER-ADMM

The difference lies in that how we calculate the estimated stochastic gradient $\tilde{\nabla} f_i(x_i^k)$.

1. For a certian hyper-parameter q, if k is divisible by q, then

$$\tilde{\nabla} f_i(x_i^k) = \frac{1}{\varpi_1} \sum_{\xi_i \in \mathcal{I}_{k,i}} \nabla F_i(x_i^k; \xi_i), \quad \forall i \in [m], \tag{242}$$

   where ϖ_1 is the mini-batch size.

2. Otherwise,

$$\begin{aligned} \tilde{\nabla} f_i(x_i^k) = \frac{1}{\varpi_2} \sum_{\xi_i \in \mathcal{I}_{k,i}} \Big(\nabla F_i(x_i^k; \xi_i) - \nabla F_i(x_i^{k-1}; \xi_i) \Big) \\ + \tilde{\nabla} f_i(x_i^{k-1}), \qquad \forall i \in [m], \end{aligned} \tag{243}$$

   where ϖ_2 is the mini-batch size.

SPIDER-ADMM

Compared with SADMM, SPIDER-ADMM is able to achieve a $\mathcal{O}(1/\epsilon^3)$ rate under Assume V-2 and the surjectiveness of B.

It's worth mentioning that, when f_i has the form of (208), SPIDER-ADMM can achieve a $\mathcal{O}(n + \frac{n^{1/2}}{\epsilon^2})$ rate.

For generic nonconvex optimization, SPIDER-ADMM is more efficient compared with the traditional VR methods, for example, SVRG-ADMM, in the sense that the latter can only achieve a complexity of $\mathcal{O}(\min\{\frac{1}{\epsilon^{10/3}}, n + \frac{n^{2/3}}{\epsilon^2}\})$.

Conclusion

Time to make a conclusion.

- Stochastic problems are problems in which the objective (or part of the objective) has a stochastic component. Nearly all machine learning problems are stochastic problems.
- We firstly introduce SADMM, which is built on an approximated augmented Lagrangian function. Even under strong assumptions, SADMM can only achieve $\mathcal{O}(1/\sqrt{K})$ rate.
- Then we introduce SVRG-ADMM, which is accelerated by the variance reduction (VR) technique. *VR is to frequently pre-store a snapshot vector and to control the variance via the snapshot vector and the latest iterate.* SVRG-ADMM can achieve $\mathcal{O}(1/S)$ rate, which is much faster.

Conclusion (Cont'd)

- We further demonstrate SADMM-MA, which is a combination of SADMM and the Momentum technique. SVRG-ADMM can achieve $\mathcal{O}(1/S)$ rate, and it's faster than SVRG-ADMM.
- SADMM, SVRG-ADMM, and SADMM-MA are good for convex stochastic problems (at least the objective has a convex part). When the problem is nonconvex, they can only achieve $\mathcal{O}(1/\epsilon^4)$ rate.
- We then introduce SPIDER-ADMM, which achieves a speed up on the above SADMMs. It's able to achieve a $\mathcal{O}(1/\epsilon^3)$ rate. Although not very fast, it is a big improvement for stochastic nonconvex problems.

Outline

Distributed Optimization

Consider the following problem $\mathcal{P}_{10}$ in a distributed environment:

$$\mathcal{P}_{10}: \quad \min_{x\in\mathbb{R}^d} f(x) \equiv \sum_{i=1}^{m} f_i(x), \tag{244}$$

where m agents form *a* ***connected*** *and* ***undirected*** *network* and the local function f_i is *only* accessible by agent i due to the storage or privacy reasons. In $\mathcal{P}_{10}$, the agents work collaboratively without revealing their own information to each other. We consider two possible scenarios:

- **Centralized**: There is a centralized master agent and m worker agents. Each worker agent is connected to the master agent
- **Decentralized**: No centralized agent. Each agent only communicates with its neighbors

Centralized Distributed Optimization

In the centralized network, $\mathcal{P}_{10}$ can be reformulated as $\mathcal{P}'_{10}$:

$$\mathcal{P}'_{10}: \quad \min_{\{x_i\}_{i\in[m]},z} \sum_{i=1}^{m} f_i(x_i) \tag{245}$$

$$s.t. \quad x_i = z, \quad i \in [m]. \tag{246}$$

$\mathcal{P}'_{10}$ is obviously solvable by the vanilla ADMM.

The augmented Lagrangian function is:

$$L(x, z, \lambda) = \sum_i \left(f_i(x_i) + \langle \lambda_i, x_i - z \rangle + \frac{\beta}{2} \|x_i - z\|^2 \right). \tag{247}$$

CADMM

We have the following iterations:

$$
\begin{aligned}
z^{k+1} &= \operatorname*{argmin}_z \sum_i \left(\langle \lambda_i^k, x_i^k - z \rangle + \frac{\beta}{2} \|x_i^k - z\|^2 \right) \\
&= \frac{1}{m} \sum_i \left(x_i^k + \frac{1}{\beta} \lambda_i^k \right), \qquad (248) \\
x_i^{k+1} &= \operatorname*{argmin}_{x_i} \left(f_i(x_i) + \langle \lambda^k, x_i - z^{k+1} \rangle + \frac{\beta}{2} \|x_i - z^{k+1}\|^2 \right) \\
&= \operatorname{prox}_{\beta^{-1} f_i} \left(z^{k+1} - \frac{1}{\beta} \lambda_i^k \right), \qquad i \in [m], \qquad (249) \\
\lambda_i^{k+1} &= \lambda_i^k + \beta (x_i^{k+1} - z^{k+1}), \qquad i \in [m], \qquad (250)
\end{aligned}
$$

where $\operatorname{prox}_{\alpha f}(z) := \operatorname{argmin}_x \left\{ f(x) + \frac{1}{2\alpha} \|z - x\|^2 \right\}$ is the formal definition of the proximal operator.

CADMM

The iterations above are carried by different characters. Specifically, for the master:

```
1. for $k = 0, 1, 2, \ldots$ do
    1.1 Wait until receiving $x_i^k$ and $\lambda_i^k$ from all the workers
    1.2 Update $z^{k+1}$ by (248)
    1.3 Send $z^{k+1}$ to all the workers
```

For each worker i:

```
1. Initialize $x_i^0, \lambda_i^0$
2. for $k = 0, 1, 2, \ldots$ do
    2.1 Send $x_i^k$ and $\lambda_i^k$ to the master
    2.2 Wait until receiving $z^{k+1}$ from the master
    2.3 Update $x_i^{k+1}$ and $\lambda_i^{k+1}$ by (249) and (250), respectively
```

We name them CADMM-M and CADMM-W, respectively (Centralized ADMM for the master/worker). From Sec. II, CADMM is able to achieve a linear rate when each f_i is μ-strongly convex and L-smooth.

CLADMM

Obviously, LADMMs can also be applied to $\mathcal{P}'_{10}$, especially when each f_i is L-smooth and the proximal operator in (249) is not easily computable. By adding the Bregman distance with

$$\phi_i(x_i) = \frac{L}{2}\|x_i\|^2 - f_i(x_i) \tag{251}$$

on (249), we have:

$$\begin{aligned} x_i^{k+1} &= \operatorname*{argmin}_{x_i} \Big(f_i(x_i) + \langle \lambda^k, x_i - z^{k+1} \rangle + \frac{\beta}{2}\|x_i - z^{k+1}\|^2 \\ &\qquad\qquad + D_{\phi_i}(x_i, x_i^k) \Big) \\ &= \operatorname*{argmin}_{x_i} \Big(\langle \nabla f_i(x_i^k), x_i - x_i^k \rangle + \frac{L}{2}\|x_i - x_i^k\|^2 \\ &\quad + \langle \lambda_i^k, x_i - z^{k+1} \rangle + \frac{\beta}{2}\|x_i - z^{k+1}\|^2 \Big) \\ &= \frac{1}{L+\beta}\Big(L x_i^k + \beta z^{k+1} - \nabla f_i(x_i^k) - \lambda_i^k \Big), i \in [m]. \end{aligned} \tag{252}$$

Acc-CLADMM

Replacing (249) by (252) while keeping the left unchanged, we get CLADMM-M and CLADMM-W. Similarly, CLADMM is able to achieve a linear rate when each f_i is μ-strongly convex and L-smooth.

Now we consider the Accelerated ADMMs. Specifically, let's extend Acc-LADMM-3 to solve $\mathcal{P}'_{10}$ —— We linearize f_i at the auxiliary variable w_i^k in the x_i update step:

$$\begin{aligned} w_i^k &= \theta x_i^k + (1-\theta)\tilde{x}_i^k, && (253) \\ z^{k+1} &= \operatorname*{argmin}_z \sum_i \left(\langle \lambda_i^k, x_i^k - z \rangle + \frac{\beta\theta}{2}\|x_i^k - z\|^2 \right) \\ &= \frac{1}{m}\sum_i \left(x_i^k + \frac{1}{\beta\theta}\lambda_i^k \right). && (254) \end{aligned}$$

Acc-CLADMM

We linearize f_i at the auxiliary variable w_i^k in the x_i update step (cont'd):

$$\begin{aligned}
x_i^{k+1} &= \operatorname*{argmin}_{x_i} \Bigg(\langle \nabla f_i(w_i^k), x_i - x_i^k \rangle + \langle \lambda_i^k, x_i - z^{k+1} \rangle \\
&\quad + \beta\theta \langle x_i^k - z^{k+1}, x_i \rangle \\
&\quad + \frac{1}{2}\Big(\frac{\theta}{\alpha} + \mu\Big) \Big\| x_i - \frac{1}{\frac{\theta}{\alpha} + \mu}\Big(\frac{\theta}{\alpha} x_i^k + \mu w_i^k\Big) \Big\|^2 \Bigg) \\
&= \frac{1}{\frac{\theta}{\alpha} + \mu} \Big\{ \mu w_i^k + \frac{\theta}{\alpha} x_i^k - \big[\nabla f_i(x_i^k) + \lambda_i^k + \beta\theta(x_i^k - z^{k+1}) \big] \Big\},
\end{aligned} \tag{255}$$

$$\tilde{z}^{k+1} = \theta z^{k+1} + (1-\theta)\tilde{z}^k, \tag{256}$$

$$\tilde{x}_i^{k+1} = \theta x_i^{k+1} + (1-\theta)\tilde{x}_i^k, \tag{257}$$

$$\lambda_i^{k+1} = \lambda_i^k + \beta\theta(x_i^{k+1} - z^{k+1}). \tag{258}$$

Acc-CLADMM

We use the following procedures to consume the above iterations —— For the master, we have:

```
1. Initialize $\tilde{z}^0$
2. for $k = 0, 1, 2, \ldots$ do
    2.1 Wait until receiving $x_i^k$ and $\lambda_i^k$ from all the workers
    2.2 Update $z^{k+1}$ and $\tilde{z}^{k+1}$ by (254) and (256), respectively
    2.3 Send $z^{k+1}$ to all the workers
```

For each worker i:

```
1. Initialize $x_i^0, \lambda_i^0, \tilde{x}_i^0$
2. for $k = 0, 1, 2, \ldots$ do
    2.1 Send $x_i^k$ and $\lambda_i^k$ to the master
    2.2 Wait until receiving $z^{k+1}$ from the master
    2.3 Update $x_i^{k+1}, \tilde{x}_i^{k+1}, \lambda_i^{k+1}$, and $w_i^{k+1}$ by (255), (257), (258), and
        (253), respectively
```

They are called Acc-CLADMM-M and Acc-CLADMM-W, respectively.

Acc-CLADMM

Unsurprisingly, Acc-CLADMM is able to achieve a linear rate when each f_i is μ-strongly convex and L-smooth. The comparisons between the introduced methods for $\mathcal{P}'_{10}$:

Method	Rate
CADMM	$\mathcal{O}(\sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon})$
CLADMM	$\mathcal{O}(\frac{L}{\mu} \log \frac{1}{\epsilon})$
Acc-CLADMM	$\mathcal{O}(\sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon})$

We can find that, CLADMM is slower than CADMM, as it is the cost of linearization. However, Acc-CLADMM is able to remedy that, and it has the same rate as CADMM.

Decentralized Distributed Optimization

Now we study how to solve $\mathcal{P}_{10}$ in a decentralized network.

Formally, we denote the set of edges by $\mathcal{E}$ (recall that the network is undirected), and it it defined as follows —— *Assume that all nodes are ordered from* 1 *to m. For any two nodes i and j, if i and j are directly connected in the network and* $i < j$*, we say* $(i, j) \in \mathcal{E}$*.* To simplify the presentation, we order the edges from 1 to $|\mathcal{E}|$.

For each node i, we denote $\mathcal{N}_i$ as its neighborhood:

$$\mathcal{N}_i := \{j \mid (i, j) \in \mathcal{E} \text{ or } (j, i) \in \mathcal{E}\}, \tag{259}$$

and $d_i = |\mathcal{N}_i|$ as its degree.

Decentralized Distributed Optimization

With the above preliminaries, we can rewrite $\mathcal{P}_{10}$ as the following $\mathcal{P}_{10}''$:

$$\mathcal{P}_{10}'' : \quad \min_{\{x_i\}_{i\in[m]}, \{z_{ij}\}_{(i,j)\in\mathcal{E}}} \sum_{i=1}^{m} f_i(x_i) \tag{260}$$

$$s.t. \quad x_i = z_{ij}, x_j = z_{ij}, \quad (i,j) \in \mathcal{E}. \tag{261}$$

In this case, the augmented Lagrangian function is

$$L(x, z, \lambda) = \sum_i f_i(x_i) + \sum_{(i,j)} \Big(\langle \lambda_{ij}, x_i - z_{ij} \rangle + \langle \gamma_{ij}, x_j - z_{ij} \rangle$$

$$+ \frac{\beta}{2} \|x_i - z_{ij}\|^2 + \frac{\beta}{2} \|x_j - z_{ij}\|^2 \Big). \tag{262}$$

DADMM

We try to use the vanilla ADMM to solve it.

$$x_i^{k+1} = \operatorname*{argmin}_{x_i} \Big[f_i(x_i) + \sum_{j:(i,j)\in\mathcal{E}} \Big(\langle \lambda_{ij}^k, x_i - z_{ij}^k \rangle + \frac{\beta}{2}\|x_i - z_{ij}^k\|^2 \Big) + \sum_{j:(j,i)\in\mathcal{E}} \Big(\langle \gamma_{ji}^k, x_i - z_{ji}^k \rangle + \frac{\beta}{2}\|x_i - z_{ji}^k\|^2 \Big) \Big], \tag{263}$$

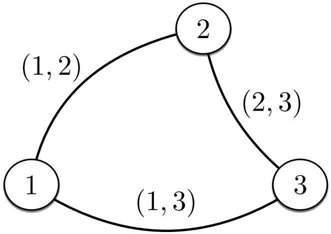
$$z_{ij}^{k+1} = \operatorname*{argmin}_{z_{ij}} \Big(-\langle \lambda_{ij}^k + \gamma_{ij}^k, z_{ij} \rangle + \frac{\beta}{2}\|x_i^{k+1} - z_{ij}\|^2 + \frac{\beta}{2}\|x_j^{k+1} - z_{ij}\|^2 \Big) = \frac{1}{2\beta}(\lambda_{ij}^k + \gamma_{ij}^k) + \frac{1}{2}(x_i^{k+1} + x_j^{k+1}), \tag{264}$$

$$\lambda_{ij}^{k+1} = \lambda_{ij}^k + \beta(x_i^{k+1} - z_{ij}^{k+1}), \tag{265}$$

$$\gamma_{ij}^{k+1} = \gamma_{ij}^k + \beta(x_j^{k+1} - z_{ij}^{k+1}). \tag{266}$$

DADMM

How to consume the above iterations? Here we have a problem —— Who is responsible for updating $\lambda_{ij}, \gamma_{ij}$, and z_{ij} associated with each edge (i, j)? A naive solution is, node i is responsible to update the edges in the set $\mathcal{N}_i' := \{(i, j) \mid \text{if } (i, j) \in \mathcal{E}\}$.



In this figure, node 1 is responsible for updating the variables on $(1, 2)$ and $(1, 3)$; Node 2 is responsible for updating the variables on $(2, 3)$; Node 3 does not have work to do.

DADMM

Then we can directly design the following procedures to consume the above iterations —— For each node i:

```
1. Initialize $x_i^0, \{\lambda_{ij}^0\}_{j\in\mathcal{N}_i'}, \{\gamma_{ij}^0\}_{j\in\mathcal{N}_i'}$, and $\{z_{ij}^0\}_{j\in\mathcal{N}_i'}$
2. Send $x_i^0$ to each neighbor $j \in \mathcal{N}_i$; Send $\{\gamma_{ij}^0\}_{j\in\mathcal{N}_i'}$ and $\{z_{ij}^0\}_{j\in\mathcal{N}_i'}$ to each neighbor $j \in \mathcal{N}_i \backslash \mathcal{N}_i'$
3. Wait until receiving $\{x_j^0\}_{j\in\mathcal{N}_i}$ from each neighbor $j \in \mathcal{N}_i$; Wait until receiving $\gamma_{ij}^0$, and $z_{ij}^0$ from each neighbor $j \in \mathcal{N}_i \backslash \mathcal{N}_i'$
4. for $k = 0, 1, 2, \ldots$ do
    4.1 Update $x_i^{k+1}$ by (263) ($\{\gamma_{ji}^k, z_{ji}^k\}_{j\in\mathcal{N}_i\backslash\mathcal{N}_i'}$ are from the outside)
    4.2 Send $x_i^{k+1}$ to each neighbor $j \in \mathcal{N}_i$
    4.3 Wait until receiving $\{x_j^{k+1}\}_{j\in\mathcal{N}_i}$ from each neighbor $j \in \mathcal{N}_i$
    4.4 for each $j \in \mathcal{N}_i'$ do
        4.4.1 Update $z_{ij}^{k+1}$ by (264)
        4.4.2 Update $\lambda_{ij}^{k+1}$ by (265)
        4.4.3 Update $\gamma_{ij}^{k+1}$ by (266)
    4.5 Send $\{\gamma_{ij}^{k+1}\}_{j\in\mathcal{N}_i'}$ and $\{z_{ij}^{k+1}\}_{j\in\mathcal{N}_i'}$ to each neighbor $j \in \mathcal{N}_i \backslash \mathcal{N}_i'$
    4.6 Wait until receiving $\gamma_{ij}^{k+1}$, and $z_{ij}^{k+1}$ from each neighbor $j \in \mathcal{N}_i \backslash \mathcal{N}_i'$
```

DADMM-S

We call the above procedure DADMM. It's too complicated, and each node has unequal amount of work. Nevertheless, without any additional information, we can simplify it by eliminating $\{z_{ij}, \lambda_{ij}, \gamma_{ij}\}$.

Summing (265) and (266) and using (264), we have

$$\lambda_{ij}^{k+1} + \gamma_{ij}^{k+1} = 0, \forall k \geq 0.$$

Initializing $\lambda_{ij}^{0} = \gamma_{ij}^{0} = 0$, we have

$$\lambda_{ij}^{k} + \gamma_{ij}^{k} = 0, \forall k \geq 0. \tag{267}$$

Plugging it into (264), we have

$$z_{ij}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}), \forall k \geq 0. \tag{268}$$

Similarly, we may initialize $z_{ij}^{0} = \frac{1}{2}(x_i^{0} + x_j^{0})$.

DADMM-S

From (268) and (265), we have

$$\lambda_{ij}^{k+1} = \lambda_{ij}^{k} + \frac{\beta}{2}(x_i^{k+1} - x_j^{k+1}). \tag{269}$$

So we have

$$\lambda_{ij}^{k+1} = \beta \sum_{t=1}^{k+1} \frac{1}{2}(x_i^{t} - x_j^{t}). \tag{270}$$

Similarly, we can get

$$\gamma_{ij}^{k+1} = \beta \sum_{t=1}^{k+1} \frac{1}{2}(x_j^{t} - x_i^{t}). \tag{271}$$

Note that we only define λ_{ij}, γ_{ij}, and z_{ij} for $i < j$. Now we define

$$\lambda_{ij} \equiv \gamma_{ji} \text{ and } z_{ij} \equiv z_{ji}, \quad i > j. \tag{272}$$

DADMM-S

Then (268), (270), and (271) hold for both $i < j$ and $i > j$. In this case, we can simplify (263) to

$$\begin{aligned}
x_i^{k+1} &= \operatorname*{argmin}_{x_i} \left[f_i(x_i) + \sum_{j:(i,j)\in\mathcal{E}} \left(\langle \lambda_{ij}^k - \beta z_{ij}^k, x_i \rangle + \frac{\beta}{2}\|x_i\|^2 \right) \right. \\
&\quad \left. + \sum_{j:(i,j)\in\mathcal{E}} \left(\langle \gamma_{ji}^k - \beta z_{ji}^k, x_i \rangle + \frac{\beta}{2}\|x_i\|^2 \right) \right] \\
&= \operatorname*{argmin}_{x_i} \left[f_i(x_i) + \sum_{j\in\mathcal{N}_i} \left(\langle \lambda_{ij}^k - \beta z_{ij}^k, x_i \rangle + \frac{\beta}{2}\|x_i\|^2 \right) \right] \\
&= \operatorname*{argmin}_{x_i} \left[f_i(x_i) + \sum_{j\in\mathcal{N}_i} \left(\langle \lambda_{ij}^k - \beta z_{ij}^k + \beta x_i^k, x_i \rangle + \frac{\beta}{2}\|x_i - x_i^k\|^2 \right) \right] \\
&= \operatorname*{argmin}_{x_i} \left[f_i(x_i) + \sum_{j\in\mathcal{N}_i} \left(\langle \lambda_{ij}^k + \frac{\beta}{2}(x_i^k - x_j^k), x_i \rangle + \frac{\beta}{2}\|x_i - x_i^k\|^2 \right) \right].
\end{aligned} \tag{273}$$

DADMM-S

We denote the Laplacian matrix of the network as $L \in \mathbb{R}^{m \times m}$ and D as the diagonal degree matrix with $D_{ii} = d_i$. We have

$$0 \leq \alpha^T L \alpha = \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} (\alpha_i - \alpha_j)^2 \leq \sum_{(i,j) \in \mathcal{E}} (\alpha_i^2 + \alpha_j^2) = 2\alpha^T D \alpha,$$

i.e., $0 \preceq L \preceq 2D$. Define

$$X = \begin{pmatrix} x_1^T \\ \vdots \\ x_m^T \end{pmatrix} \in \mathbb{R}^{m \times d}, f(X) = \sum_i f_i(x_i), \tag{274}$$

$$v_i = \sum_{j \in \mathcal{N}_i} \lambda_{ij}, \text{ and } \Upsilon = \begin{pmatrix} v_1^T \\ \vdots \\ v_m^T \end{pmatrix} \in \mathbb{R}^{m \times d}. \tag{275}$$

DADMM-S

Then we have

$$L_i^T X = d_i x_i^T - \sum_{j \in \mathcal{N}_i} x_j^T, \quad \forall i \in [m], \tag{276}$$

where L_i is the i-th column of L. The result can be obtained easily because d_i is the coefficient of the diagonal element and -1 is the coefficient of x_{ij}, $i \neq j$, if there is an edge between i and j. Based on this, (273) can be simplified as

$$\begin{aligned} x_i^{k+1} &= \operatorname*{argmin}_{x_i} \Big[f_i(x_i) + \langle v_i^k, x_i \rangle + \frac{\beta}{2} \Big\langle \sum_{j \in \mathcal{N}_i} L_{ij} x_j^k, x_i \Big\rangle \\ &\quad + \frac{\beta d_i}{2} \|x_i - x_i^k\|^2 \Big] \\ &= \operatorname{prox}_{(\beta d_i)^{-1} f_i} \Big(x_i^k - \frac{1}{\beta d_i} \big(v_i^k + \frac{\beta}{2} \sum_{j \in \mathcal{N}_i} L_{ij} x_j^k \big) \Big). \end{aligned} \tag{277}$$

DADMM-S

Summing (269) over $j \in \mathcal{N}_i$, we have

$$v_i^{k+1} = v_i^k + \frac{\beta}{2} \sum_{j \in \mathcal{N}_i} L_{ij} x_j^{k+1}, i \in [m]. \tag{278}$$

Then based on (277) and (278), we have the following algorithm, named *DADMM-S* (DADMM, Simplified) —— For each node i:

```
1. Initialize $x_i^0$ and $v_i^0$ as 0
2. Send $x_i^0$ to its neighbors
3. Wait until receiving $x_j^0$ from all its neighbors
4. for $k = 0, 1, 2, \ldots$ do
    4.1 Update $x_i^{k+1}$ by (277)
    4.2 Send $x_i^{k+1}$ to its neighbors
    4.3 Wait until receiving $x_j^{k+1}$ from all its neighbors
    4.4 Update $v_i^{k+1}$ by (278)
```

DADMM-S is much simpler than DADMM by greatly reducing the communication overheads.

DADMM-S

Now let's discuss one more interesting thing. (277) and (278) can be written in a compact form:

$$X^{k+1} = \operatorname*{argmin}_X \left(f(X) + \langle \Upsilon^k + \frac{\beta}{2} L X^k, X \rangle + \frac{\beta}{2} \| \sqrt{D}(X - X^k) \|^2 \right), \tag{279}$$

$$\Upsilon^{k+1} = \Upsilon^k + \frac{\beta}{2} L X^{k+1}. \tag{280}$$

Denoting $W = \sqrt{L/2}$, (280) can be rewritten as

$$\Upsilon^{k+1} = \Upsilon^k + \beta W^2 X^{k+1}. \tag{281}$$

Letting $\Upsilon^0 \in \operatorname{span}(W^2)$, we know that

$$\Upsilon^k \in \operatorname{span}(W^2), \forall k \geq 0 \tag{282}$$

and there exists Ω^k such that $\Upsilon^k = W \Omega^k$.

DADMM-S

Then, (279) and (280) can be rewritten as

$$\begin{aligned} X^{k+1} &= \underset{X}{\operatorname{argmin}} \Big(f(X) + \langle \Omega^k, WX \rangle + \beta \langle W^2 X^k, X \rangle \\ &\quad + \frac{\beta}{2} \|\sqrt{D}(X - X^k)\|^2 \Big) \\ &= \underset{X}{\operatorname{argmin}} \Big(f(X) + \langle \Omega^k, WX \rangle + \frac{\beta}{2} \|WX\|^2 + D_\Psi(X, X^k) \Big), \end{aligned} \tag{283}$$

$$\Omega^{k+1} = \Omega^k + \beta W X^{k+1}, \tag{284}$$

with

$$\Psi(X) = \frac{\beta}{2} \|\sqrt{D} X\|^2 - \frac{\beta}{2} \|WX\|^2. \tag{285}$$

DADMM-S

Thus, DADMM-S is exactly using LADMM to solve

$$\min_X f(X), \quad s.t. \quad WX = 0. \tag{286}$$

Under the assumption that each f_i is μ-storngly convex and L_f-smooth DADMM-S is able to achieve a linear convergence rate, more specifically, $\mathcal{O}((\sqrt{\frac{L_f d_{max}}{\mu \sigma_L}} + \frac{d_{max}}{\sigma_L}) \log \frac{1}{\epsilon})$, where $d_{max} = \max\{d_i\}$, and σ_L is the smallest positive eigenvalue of the Laplacian matrix L.

DLADMM

In the following, we also assume that each f_i is μ-storngly convex and L_f-smooth. (263) (or (277)) is actually a proximal mapping of f_i. As what we have done in Sec. II, when the proximal mapping of f_i is not easily computable, we can linearize f_i, which leads to the following iteration:

$$\begin{aligned} x_i^{k+1} = \operatorname*{argmin}_{x_i} \Bigg[& \underbrace{\langle \nabla f_i(x_i^k), x_i - x_i^k \rangle + \frac{L_f}{2}\|x_i - x_i^k\|^2}_{\textit{approximated linearization of } f_i} \\ & + \sum_{j:(i,j)\in\mathcal{E}} \left(\langle \lambda_{ij}^k, x_i - z_{ij}^k \rangle + \frac{\beta}{2}\|x_i - z_{ij}^k\|^2 \right) \\ & + \sum_{j:(j,i)\in\mathcal{E}} \left(\langle \gamma_{ji}^k, x_i - z_{ji}^k \rangle + \frac{\beta}{2}\|x_i - z_{ji}^k\|^2 \right) \Bigg]. \end{aligned} \tag{287}$$

DLADMM

Similarly, (287) can be transformed into

$$\begin{aligned} x_i^{k+1} &= \underset{x_i}{\operatorname{argmin}} \Big[\langle \nabla f_i(x_i^k), x_i - x_i^k \rangle + \frac{L_f}{2} \|x_i - x_i^k\|^2 \\ &\quad + \sum_{j \in \mathcal{N}_i} \Big(\langle \lambda_{ij}^k + \frac{\beta}{2}(x_i^k - x_j^k), x_i \rangle + \frac{\beta}{2} \|x_i - x_i^k\|^2 \Big) \Big] \\ &= x_i^k - \frac{1}{L_f + \beta d_i} \Big\{ \nabla f_i(x_i^k) + \sum_{j \in \mathcal{N}_i} \Big[\lambda_{ij}^k + \frac{\beta}{2}(x_i^k - x_j^k) \Big] \Big\}. \end{aligned} \tag{288}$$

DLADMM

We can also write (288) into a compact form:

$$X^{k+1} = X^k - (L_f I + \beta D)^{-1}\left(\beta W^2 X^k + \nabla f(X^k) + W\Omega^k\right),$$

where I is the identity matrix. We call (288) and (278) *DLADMM* (Decentralized Linearized ADMM).

Similar to DADMM-S, DLADMM is also a speicial case of LADMM, but with a different Bregman distance:

$$\Psi(X) = \underbrace{\frac{L}{2}\|X\|^2 - f(X)}_{\textit{the added term}} + \frac{\beta}{2}\|\sqrt{D}X\|^2 - \frac{\beta}{2}\|WX\|^2. \qquad (289)$$

DLADMM

The procedure of DLADMM is —— For each node i:

```
1. Initialize $x_i^0$ and $v_i^0$ as $0$
2. Send $x_i^0$ to its neighbors
3. Wait until receiving $x_j^0$ from all its neighbors
4. for $k = 0, 1, 2, \ldots$ do
    4.1 Update $x_i^{k+1}$ by (288)
    4.2 Send $x_i^{k+1}$ to its neighbors
    4.3 Wait until receiving $x_j^{k+1}$ from all its neighbors
    4.4 Update $v_i^{k+1}$ by (278)
```

The only difference lies in step 4.1.

Under the same assumption to DADMM-S, DLADMM is also able to achieve a linear convergence rate, but faster. Specifically, the rate is $\mathcal{O}((\frac{L_f}{\mu} + \frac{d_{max}}{\sigma_L}) \log \frac{1}{\epsilon})$.

Acc-DLADMM

Without any doubt, We can also use Accelerated LADMMs to solve $\mathcal{P}''_{10}$. In this case we have

$$Y^k = \theta X^k + (1-\theta)\tilde{X}^k, \tag{290}$$

$$X^{k+1} = \frac{1}{\frac{\theta}{\alpha}+\mu}\left[\mu Y^k + \frac{\theta}{\alpha}X^k - \left(\nabla f(Y^k) + W\Omega^k + \beta\theta W^2 X^k\right)\right], \tag{291}$$

$$\tilde{X}^{k+1} = \theta X^{k+1} + (1-\theta)\tilde{X}^k, \tag{292}$$

$$\Omega^{k+1} = \Omega^k + \beta\theta X^{k+1}. \tag{293}$$

(291) is a direct extension from (109). We name it *Acc-DLADMM*.

Acc-DLADMM

The detailed procedure of Acc-DLADMM is —— For each node i:

```
1. Initialize $x_i^0 = \tilde{x}_i^0$ and $v_i^0 = 0$
2. Send $x_i^0$ to its neighbors
3. Wait until receiving $x_j^0$ from all its neighbors
4. for $k = 0, 1, 2, \ldots$ do
    4.1 Update $y_i^k$ by (290)
    4.2 Update $x_i^{k+1}$ by (291)
    4.3 Update $\tilde{x}_i^{k+1}$ by (292)
    4.4 Send $x_i^{k+1}$ to its neighbors
    4.5 Wait until receiving $x_j^{k+1}$ from all its neighbors
    4.6 Update $v_i^{k+1}$ by (293)
```

Note that the formulas mentioned above are used in their non-compact forms.

Comparisons

Under the same assumption to DADMM-S and DLADMM, Acc-DLADMM is able to achieve a faster rate —— $\mathcal{O}(\sqrt{\frac{L_f d_{max}}{\mu\sigma_L}}\log\frac{1}{\epsilon})$.

For solving $\mathcal{P}_{10}''$, the complexity of different algorithms are compared below:

Method	Rate
DADMM-S	$\mathcal{O}((\sqrt{\frac{L_f d_{max}}{\mu\sigma_L}} + \frac{d_{max}}{\sigma_L})\log\frac{1}{\epsilon})$
DLADMM	$\mathcal{O}((\frac{L_f}{\mu} + \frac{d_{max}}{\sigma_L})\log\frac{1}{\epsilon})$
Acc-DLADMM	$\mathcal{O}(\sqrt{\frac{L_f d_{max}}{\mu\sigma_L}}\log\frac{1}{\epsilon})$

Asynchronous Distributed Optimization

All the CADMMs and DADMMs introduced above are executed in a synchronous manner. That is, the master needs to wait for all the workers (each worker needs to wait for all its neighbors) to finish their updates before it can proceed. Thus, the system proceeds at the pace of the slowest node.

In the following, we introduce the asynchronous ADMMs to reduce the waiting time for centralized optimization problems. They can be easily extended to decentralized problems.

Asynchronous Centralized Distributed Optimization

In the asynchronous setting, the master does not wait for all the workers, but proceeds as long as it receives information from a partial set of workers instead. We denote the partial set at iteration k as $\mathcal{A}^k$, and $\mathcal{A}_c^k$ as the complementary set of $\mathcal{A}^k$, which means the set of workers whose information does not arrive at iteration k.

We use α to lower bound the size of $\mathcal{A}^k$. We require that the master has to receive the updates from every worker at least once in every τ iterations. That is, we do not allow some workers to be absent for a long time. So we make the following bounded delay assumption.

Assumption VI-1

The maximum tolerable delay for all i and k is upper bounded.

Async-CADMM

Denote the upper bound as τ, then it must be that for every worker i,

$$i \in \mathcal{A}^k \cup \mathcal{A}^{k-1} \cup \cdots \cup \mathcal{A}^{\max\{k-\tau+1,0\}}. \tag{294}$$

Similar to CADMM, we have the following iterations:

$$x_i^{k+1} = \begin{cases} \operatorname{argmin}_{x_i} \Big(f_i(x_i) + \langle \lambda_i^{\bar{k}_i+1}, x_i \rangle \\ \qquad\qquad + \frac{\beta}{2} \| x_i - z^{\bar{k}_i+1} \|^2 \Big) & i \in \mathcal{A}^k \\ x_i^k & i \in \mathcal{A}_c^k, \end{cases} \tag{295}$$

$$\lambda_i^{k+1} = \begin{cases} \lambda_i^{\bar{k}_i+1} + \beta(x_i^{k+1} - z^{\bar{k}_i+1}) & i \in \mathcal{A}^k \\ \lambda_i^k & i \in \mathcal{A}_c^k. \end{cases} \tag{296}$$

Async-CADMM

We have the following iterations (cont'd):

$$z^{k+1} = \operatorname*{argmin}_z \Bigg[\sum_{i=1}^{m} \left(\langle \lambda_i^{k+1}, x_i^{k+1} - z \rangle + \frac{\beta}{2} \|x_i^{k+1} - z\|^2 \right) + \frac{\rho}{2} \|z - z^k\|^2 \Bigg]. \tag{297}$$

Note that z is updated with adding an additional proximal term. In the above iterations, we denote $\bar{k}_i$ as *the last iteration before iteration k for which worker $i \in \mathcal{A}^k$ arrives*, i.e., $i \in \mathcal{A}^{\bar{k}_i}$. Thus, for all workers $i \in \mathcal{A}^k$, we have

$$x_i^{\bar{k}_i+1} = x_i^{\bar{k}_i+2} = \cdots = x_i^k, \tag{298}$$

$$\lambda_i^{\bar{k}_i+1} = \lambda_i^{\bar{k}_i+2} = \cdots = \lambda_i^k, \text{ and} \tag{299}$$

$$\max\{k - \tau, 0\} \leq \bar{k}_i < k. \tag{300}$$

Async-CADMM

For each $i \in \mathcal{A}_c^k$, we denote $\tilde{k}_i$ as *the last iteration before iteration k for which worker i arrives*, i.e., $i \in \mathcal{A}^{\tilde{k}_i}$. Under the bounded delay assumption, we have

$$\max\{k - \tau + 1, 0\} \leq \tilde{k}_i < k. \tag{301}$$

Similarly, for all workers $i \in \mathcal{A}_c^k$, we have

$$x_i^{\tilde{k}_i+1} = x_i^{\tilde{k}_i+2} = \cdots = x_i^k = x_i^{k+1}, \tag{302}$$

$$\lambda_i^{\tilde{k}_i+1} = \lambda_i^{\tilde{k}_i+2} = \cdots = \lambda_i^k = \lambda_i^{k+1}. \tag{303}$$

Async-CADMM

We also denote $\hat{k}_i$ as *the last iteration before $\tilde{k}_i$ for which $i \in \mathcal{A}^{\tilde{k}_i}$ arrives*, i.e., $i \in \mathcal{A}^{\hat{k}_i}$. We also have

$$x_i^{\hat{k}_i+1} = x_i^{\hat{k}_i+2} = \cdots = x_i^{\tilde{k}_i}, \tag{304}$$

$$\lambda_i^{\hat{k}_i+1} = \lambda_i^{\hat{k}_i+2} = \cdots = \lambda_i^{\tilde{k}_i}, \quad \text{and} \tag{305}$$

$$\max\{\tilde{k}_i - \tau, 0\} \leq \hat{k}_i < \tilde{k}_i. \tag{306}$$

Then, for all workers $i \in \mathcal{A}_c^k$, we have

$$x_i^{k+1} = x_i^{\tilde{k}_i+1} = \operatorname*{argmin}_{x_i} \left(f_i(x_i) + \langle \lambda_i^{\hat{k}_i+1}, x_i \rangle + \frac{\beta}{2} \| x_i - z^{\hat{k}_i+1} \|^2 \right), \tag{307}$$

$$\lambda_i^{k+1} = \tilde{\lambda}_i^{\tilde{k}_i+1} = \lambda_i^{\hat{k}_i+1} + \beta(x_i^{\tilde{k}_i+1} - z^{\hat{k}_i+1}). \tag{308}$$

Async-CADMM

Based on (295), (296), and (297), for the master we have:

```
1. Initialize the delay counter $\tilde{d}_1^1 = \tilde{d}_2^1 = \ldots = \tilde{d}_m^1 = 0$
2. for $k = 1, 2, \ldots$ do
    2.1 Wait until receiving $\hat{x}_i^k$ and $\hat{\lambda}_i^k$ from workers $i \in \mathcal{A}^k$ such that
        $|\mathcal{A}^k| > \alpha$ and $\tilde{d}_j^k < \tau - 1$ holds for all $j \in \mathcal{A}_c^k$
    2.2 Update the local copy of x: $x_i^{k+1} = \begin{cases} \hat{x}_i^k & i \in \mathcal{A}^k \\ x_i^k & i \in \mathcal{A}_c^k \end{cases}$
    2.3 Update the local copy of $\lambda$: $\lambda_i^{k+1} = \begin{cases} \hat{\lambda}_i^k & i \in \mathcal{A}^k \\ \lambda_i^k & i \in \mathcal{A}_c^k \end{cases}$
    2.4 Update the delay counter: $d_i^{k+1} = \begin{cases} 0 & i \in \mathcal{A}^k \\ \tilde{d}_i^k + 1 & i \in \mathcal{A}_c^k \end{cases}$
    2.5 Update $z^{k+1}$ by (297)
    2.6 Broadcast $z^{k+1}$ to the workers $i \in \mathcal{A}^k$
```

Async-CADMM

For each worker *i*:

```
1. Initialize $\hat{x}_i^0$ and $\hat{\lambda}_i^0$
2. for $k_i = 1, 2, \ldots$ do
    2.1 Wait until receiving $z$ from the master
    2.2 Update $\hat{x}_i^{k_i+1}$ by (295), i.e.,
            $$\hat{x}_i^{k_i+1} = prox\{z - \frac{1}{\beta}\hat{\lambda}_i^{k_i}\}$$
    2.3 Update $\hat{\lambda}_i^{k_i+1}$ by (296), i.e.,
            $$\hat{\lambda}_i^{k_i+1} = \hat{\lambda}_i^{k_i} + \beta(\hat{x}_i^{k_i+1} - z)$$
    2.4 Send $\hat{x}_i^{k_i+1}$ and $\hat{\lambda}_i^{k_i+1}$ to the master
```

Async-CADMM

We name it *Async-CADMM* ——

- The master only updates x_i and λ_i for $i \in \mathcal{A}^k$;
- The delay counter is introduced, and the master has to wait in some cases;
- The master only broadcast z^{k+1} to the arrived workers

When each f_i is convex and L-smooth, Async-CADMM is able to converge to the set of KKT points. Specifically, when each f_i is strongly convex, Async-CADMM can achieve linear convergence rate $\mathcal{O}(\sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon})$.

We believe that in general the asynchronous ADMM needs more iterations than synchronous ADMM. It is unclear whether the time saved per iteration of the asynchronous ADMM can offset the cost of more iterations in theory, although it shows great advantages in practice.

Async-CADMM

What if each f_i in $\mathcal{P}'_{10}$ can be nonconvex? In this case, Async-CADMM can still converge to the set of KKT points. Besides, The synchronous ADMM is a special case of the asynchronous ADMM with $\mathcal{A}_c^k = \varnothing$ and $\bar{k}_i + 1 = k$. Thus, the conclusion also holds for the synchronous ADMM with a much simpler proof.

We leave the asynchronous ADMM for decentralized distributed problems untouched in this slide.

Generally Linear Distributed Optimization

Now we consider generally linear distributed optimization problems. The problem has the form of $\mathcal{P}_3$, i.e.,

$$\min_{\{x_i\}_{i\in[m]}} \sum_{i=1}^{m} f_i(x_i), \quad s.t. \quad \sum_{i=1}^{m} A_i x_i = b.$$

Here we just study the problem in a synchronous and centralized manner. The problem in asynchronous and decentralized scenarios can be extended easily. We can use LADMM-PS introduced in Sec. II to solve it.

Specifically, for the master we have:

```
1. for $k = 0, 1, 2, \ldots$ do
    1.1 Wait until receiving $y_i^{k+1}$ from all the workers
    1.2 Update $s^{k+1}$ by: $s^{k+1} = \sum_i y_i^{k+1}$
    1.3 Update $\lambda^{k+1}$ by: $\lambda^{k+1} = \lambda^k + \beta(s^{k+1} - b)$
    1.4 Send $s^{k+1}$ and $\lambda^{k+1}$ to all the workers
```

CLADMM-PS

For each worker i:

```
1. Initialize $x_i^0$ and $\lambda_i^0$
2. $y_i^0 = A_i x_i^0$
3. Send $y_i^0$ to the master
4. Wait until receiving $s^0$ and $\lambda^0$ from the master
5. for $k = 0, 1, 2, \dots$ do
    5.1 Update $x_i^{k+1}$ by
        $$x_i^{k+1} = \operatorname*{argmin}_{x_i} \Big( f_i(x_i) + \langle \lambda^k, A_i x_i \rangle + \beta \langle A_i^T (s^k - b), x_i \rangle + \frac{m\beta \|A_i\|^2}{2} \|x_i - x_i^k\|^2 \Big) \qquad (309)$$
    5.2 Update $y_i^{k+1}$ by $y_i^{k+1} = A_i x_i^{k+1}$
    5.3 Send $y_i^{k+1}$ to the master
    5.4 Wait until receiving $x^{k+1}$ and $\lambda^{k+1}$ from the master
```

We name it *CLADMM-PS*. The way it updates x_i^{k+1} is the same to (141).

Outline

References

I do not propose anything new in this slide. If you want to use the mentioned algorithms in your work, you should go to the first reference and find the corresponding paper carefully.

1. Lin, Zhouchen, Huan Li, and Cong Fang. *Alternating Direction Method of Multipliers for Machine Learning.* Springer Nature, 2022.
2. Boyd, Stephen, Stephen P. Boyd, and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.
3. Jain, Prateek, and Purushottam Kar. *Non-convex optimization for machine learning.* Foundations and Trends® in Machine Learning 10.3-4 (2017): 142-363.