Algorithms for constrained optimization

- Penalty Methods – Choice of penalty function

The absolute value penalty function may not be differentiable at points $\mathbf{x}$ where $g_i(\mathbf{x}) = 0$. Therefore, in such cases we cannot use techniques for optimization that involve derivatives. A form of the penalty function that is guaranteed to be differentiable is the *Courant-Beltrami penalty function*, given by

$$P(\mathbf{x}) = \sum_{i=1}^{p} (g_i^+(\mathbf{x}))^2.$$

Algorithms for constrained optimization

- Penalty Methods - Convergence

Denote by $\mathbf{x}^*$ a solution (global minimizer) to the problem. Let P be a penalty function for the problem. For each $k = 1, 2, ...$, let $\gamma_k \in \mathbb{R}$ be a given positive constant. Define an associated function $q(\gamma_k, \cdot) : \mathbb{R}^n \to \mathbb{R}$ by

$$q(\gamma_k, \mathbf{x}) = f(\mathbf{x}) + \gamma_k P(\mathbf{x}).$$

For each k, we can write the following associated unconstrained optimization problem:

$$\min q(\gamma_k, \mathbf{x}).$$

Denote by $\mathbf{x}^{(k)}$ a minimizer of $q(\gamma_k, \mathbf{x})$.

Algorithms for constrained optimization

- Penalty Methods - Convergence

Lemma 1. *Suppose that* $\{\gamma_k\}$ *is a nondecreasing sequence; that is, for each* k, *we have* $\gamma_k < \gamma_{k+1}$. *Then, for each* k *we have*

1. $q(\gamma_{k+1}, \mathbf{x}^{(k+1)}) \geq q(\gamma_k, \mathbf{x}^{(k)})$.
2. $P(\mathbf{x}^{(k+1)}) \leq P(\mathbf{x}^{(k)})$.
3. $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$.
4. $f(\mathbf{x}^*) \geq q(\gamma_k, \mathbf{x}^{(k)}) \geq f(\mathbf{x}^{(k)})$.

Theorem 2. *Suppose that the objective function* f *is continuous and* $\gamma_k \to \infty$ *as* $k \to \infty$. *Then, the limit of any convergent subsequence of the sequence* $\{\mathbf{x}^{(k)}\}$ *is a solution to the constrained optimization problem.*

Algorithms for constrained optimization

- Penalty Methods – Exact penalty

We desire an exact solution to the original constrained problem by solving the associated unconstrained problem $\min_{\mathbf{x}}\ f(\boldsymbol{x}) + \gamma P(\boldsymbol{x})$ with a finite $\gamma > 0$. It turns out that indeed this can be accomplished, in which case we say that the penalty function is *exact*. However, it is necessary that exact penalty functions be nondifferentiable.

Example:

$$\begin{aligned} &\min\ f(x) \\ &\ s.t.\ x \in [0, 1], \end{aligned}$$

where $f(x) = 5 - 3x$.

Algorithms for constrained optimization

- Penalty Methods – Exact penalty

Proposition 1. *Consider the problem*

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\ s.t.\ \mathbf{x} \in \Omega, \end{aligned}$$

with $\Omega \subset \mathbb{R}^n$ *convex. Suppose that the minimizer* $\mathbf{x}^*$ *lies on the boundary of* Ω *and there exists a feasible direction* $\mathbf{d}$ *at* $\mathbf{x}^*$ *such that* $\mathbf{d}^T \nabla f(\mathbf{x}^*) > 0$*. If* P *is an exact penalty function, then* P *is not differentiable at* $\mathbf{x}^*$*.*

Proof. We use contraposition. Suppose that P is differentiable at $\mathbf{x}^*$. Then, $\mathbf{d}^T \nabla P(\mathbf{x}^*) = 0$, because $P(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega$. Hence, if we let $g = f + \gamma P$, then $\mathbf{d}^T \nabla g(\mathbf{x}^*) > 0$ for all finite $\gamma > 0$, which implies that $\nabla g(\mathbf{x}^*) \neq 0$. Hence, $\mathbf{x}^*$ is not a local minimizer of g, and thus P is not an exact penalty function. $\square$

Algorithms for constrained optimization

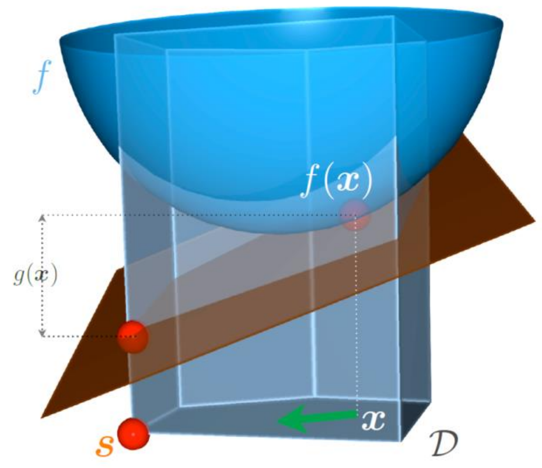
- Frank-Wolfe Algorithm

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{D}, \tag{1}$$

where f is convex and continuously differentiable and $\mathcal{D}$ is a compact convex set.

$$\boxed{f(\mathbf{x}) \approx f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle}$$

$$\begin{aligned} \mathbf{g}_k &= \underset{\mathbf{g} \in \mathcal{D}}{\operatorname{argmin}} \langle \mathbf{g}, \nabla f(\mathbf{x}_k) \rangle, \\ \mathbf{x}_{k+1} &= (1 - \gamma_k)\mathbf{x}_k + \gamma_k \mathbf{g}_k, \quad \text{where } \gamma_k = \frac{2}{k+2}. \end{aligned} \tag{2}$$



Algorithms for constrained optimization

- Frank-Wolfe Algorithm

The Frank-Wolfe algorithm is advantageous when

$$\mathbf{g}_k = \underset{\mathbf{g} \in \mathcal{D}}{\operatorname{argmin}} \langle \mathbf{g}, \nabla f(\mathbf{x}_k) \rangle$$

is easy to compute, e.g., when $\mathcal{D}$ is the unit ball of a norm, such as nuclear norm.

Algorithms for constrained optimization

- Frank-Wolfe Algorithm - Convergence

Lemma 1. *For an iteration* $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma(\mathbf{g}_k - \mathbf{x}_k)$ *with an arbitrary stepsize* $\gamma \in [0, 1]$, *it holds that*

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \gamma g(\mathbf{x}_k) + \frac{\gamma^2}{2} C_f, \tag{3}$$

if $\mathbf{g}_k$ *is an approximate linear minimizer, i.e.,*

$$\langle \mathbf{g}_k, \nabla f(\mathbf{x}_k) \rangle = \min_{\hat{\mathbf{g}}_k \in \mathcal{D}} \langle \hat{\mathbf{g}}_k, \nabla f(\mathbf{x}_k) \rangle.$$

Here $g(\mathbf{x})$ *is the duality gap defined as*

$$g(\mathbf{x}) = \max_{\mathbf{g} \in \mathcal{D}} \langle \mathbf{x} - \mathbf{g}, \nabla f(\mathbf{x}) \rangle. \tag{4}$$

Algorithms for constrained optimization

- Frank-Wolfe Algorithm - Convergence

Theorem 1. *For each $k \geq 1$, the iterate $\mathbf{x}_k$ in procedure (2) satisfies*

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{2C_f}{k+2}, \tag{5}$$

where $\mathbf{x}^* \in \mathcal{D}$ is an optimal solution to problem (1) and C_f is the curvature constant defined as

$$C_f = \sup_{\mathbf{x},\mathbf{s}\in\mathcal{D},\gamma\in[0,1],\mathbf{y}=\mathbf{x}+\gamma(\mathbf{s}-\mathbf{x})} \frac{2}{\gamma^2}(f(\mathbf{y}) - f(\mathbf{x}) - \langle \mathbf{y}-\mathbf{x}, \nabla f(\mathbf{x})\rangle). \tag{6}$$

$\boxed{\text{For } L\text{-smooth } f,\ C_f \leq L \max_{\mathbf{s}\in\mathcal{D}} \|\mathbf{s}-\mathbf{x}\|^2.}$

Algorithms for constrained optimization

- Frank-Wolfe Algorithm - Example

$$\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{y} - \sum_{i=1}^{N} x_i \mathbf{d}_i\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{7}$$

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2, \quad \Leftrightarrow \quad &&\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{y}/s - \mathbf{D}\mathbf{x}\|_2^2, \\ &s.t. \quad \|\mathbf{x}\|_1 \le s. && s.t. \quad \|\mathbf{x}\|_1 \le 1. \end{aligned} \tag{8}$$

$$\nabla f(\mathbf{x}) = \mathbf{D}^\top(\mathbf{D}\mathbf{x} - \mathbf{y}). \tag{9}$$

Now assume that $\mathbf{z}_t = \mathbf{D}\mathbf{x}_t - \mathbf{y} \in \mathbb{R}^n$ is already computed, then to compute $(\mathbf{u}_t \in \operatorname{argmin}_{\mathbf{u}\in\mathcal{X}} \nabla f(\mathbf{x}_t)^\top \mathbf{u})$ one needs to find the coordinate $i_t \in [N]$ that maximizes $|[\nabla f(\mathbf{x}_t)](i)|$ which can be done by maximizing $\mathbf{d}_i^\top \mathbf{z}_t$ and $-\mathbf{d}_i^\top \mathbf{z}_t$.

$$\begin{aligned} \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_\infty &= \|\mathbf{D}^\top \mathbf{D}(\mathbf{x} - \mathbf{y})\|_\infty = \max_{1\le i\le N} \left|\mathbf{d}_i^\top \mathbf{D}(\mathbf{x} - \mathbf{y})\right| \\ &\le \max_{1\le i\le N} \|\mathbf{d}_i^\top \mathbf{D}\|_\infty \|\mathbf{x} - \mathbf{y}\|_1 \le m^2 \|\mathbf{x} - \mathbf{y}\|_1. \end{aligned}$$

Algorithms for constrained optimization

- Alternating Direction Method

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}), \quad \text{s.t.} \quad \mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}) = \mathbf{c}, \tag{1}$$

where f and g are convex functions and $\mathcal{A}$ and $\mathcal{B}$ are linear mappings. It is a variant of the Lagrange Multiplier method. It first constructs an augmented Lagrangian function:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \langle \boldsymbol{\lambda}, \mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}) - \mathbf{c} \rangle + \frac{\beta}{2} \|\mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}) - \mathbf{c}\|^2, \tag{2}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier and $\beta > 0$ is the penalty parameter.

Algorithms for constrained optimization

- Alternating Direction Method

$$\begin{aligned} \mathbf{x}_{k+1} &= \operatorname*{argmin}_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}_k, \boldsymbol{\lambda}_k) \\ &= \operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}) + \frac{\beta}{2}\|\mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}_k) - \mathbf{c} + \boldsymbol{\lambda}_k/\beta\|^2, \end{aligned} \tag{3}$$

$$\begin{aligned} \mathbf{y}_{k+1} &= \operatorname*{argmin}_{\mathbf{y}} \mathcal{L}(\mathbf{x}_{k+1}, \mathbf{y}, \boldsymbol{\lambda}_k) \\ &= \operatorname*{argmin}_{\mathbf{y}} g(\mathbf{y}) + \frac{\beta}{2}\|\mathcal{B}(\mathbf{y}) + \mathcal{A}(\mathbf{x}_{k+1}) - \mathbf{c} + \boldsymbol{\lambda}_k/\beta\|^2. \end{aligned} \tag{4}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \beta(\mathcal{A}(\mathbf{x}_{k+1}) + \mathcal{B}(\mathbf{y}_{k+1}) - \mathbf{c}). \tag{5}$$

Convergence: deferred to Lineared Alternating Direction Method (LADM).

Algorithms for constrained optimization

- Alternating Direction Method - Example

$$\mathbf{E}_{k+1} = \underset{\mathbf{E}}{\operatorname{argmin}}\, \lambda \|\mathbf{E}\|_1 + \frac{\beta}{2}\|\mathbf{D} - \mathbf{A}_k - \mathbf{E} + \mathbf{\Lambda}_k/\beta\|_F^2, \tag{6}$$

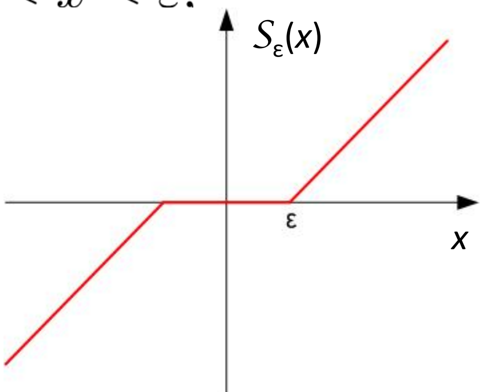
where $\mathbf{\Lambda}$ is the Lagrange multiplier.

$$\mathbf{E}_{k+1} = \mathcal{S}_{\lambda\beta^{-1}}(\mathbf{D} - \mathbf{A}_k + \mathbf{\Lambda}_k/\beta), \tag{7}$$

where

$$\mathcal{S}_\varepsilon(x) = \operatorname{sgn}(x)\max(|x| - \varepsilon, 0) = \begin{cases} x - \varepsilon, & \text{if } x > \varepsilon, \\ x + \varepsilon, & \text{if } x < -\varepsilon, \\ 0, & \text{if } -\varepsilon < x < \varepsilon. \end{cases} \tag{8}$$

is the soft thresholding operator.



Algorithms for constrained optimization

- Alternating Direction Method - Example

$$\mathbf{A}_{k+1} = \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{A}\|_* + \frac{\beta}{2}\|\mathbf{D} - \mathbf{A} - \mathbf{E}_{k+1} + \mathbf{\Lambda}_k/\beta\|_F^2, \tag{9}$$

Singular Value Thresholding (SVT): suppose that the SVD of $\mathbf{W} = \mathbf{D} - \mathbf{E}_{k+1} + \mathbf{\Lambda}_k/\beta_k$ is $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, then the optimal solution is $\mathbf{A} = \mathbf{U}\mathcal{S}_{\beta^{-1}}(\mathbf{\Sigma})\mathbf{V}^T$.

We only need to compute singular values greater than β^{-1} and their corresponding singular vectors. This can be achieved by `svds()` in MATLAB and accordingly the computation cost reduces to $O(rmn)$, where r is the expected rank of the optimal $\mathbf{A}$. It is worth noting that `svds()` can only provide expected number of leading singular values and their singular vectors. So we have to dynamically predict the value of r when calling `svds()`.

Alternating Direction Method (ADM)

Model Problem:

$$\begin{aligned} \min_{\mathbf{x}_1,\mathbf{x}_2} \quad & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2), \\ s.t. \quad & \mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2) = \mathbf{b}, \end{aligned}$$

where f_i are convex functions and $\mathcal{A}_i$ are linear mappings.

$$\begin{aligned} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\lambda}) \quad = \quad & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \langle \boldsymbol{\lambda}, \mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2) - \mathbf{b} \rangle \\ & + \frac{\beta}{2} \|\mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2) - \mathbf{b}\|_F^2, \end{aligned}$$

$$\begin{aligned} \mathbf{x}_1^{k+1} \quad &= \quad \arg\min_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2^k, \boldsymbol{\lambda}^k), \\ \mathbf{x}_2^{k+1} \quad &= \quad \arg\min_{\mathbf{x}_2} \mathcal{L}(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \boldsymbol{\lambda}^k), \\ \boldsymbol{\lambda}^{k+1} \quad &= \quad \boldsymbol{\lambda}^k + \beta_k [\mathcal{A}_1(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2^{k+1}) - \mathbf{b}]. \end{aligned}$$

Assume: Easy

Update β_k

Linearized Alternating Direction Method (LADM)

$$\begin{aligned}
\mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} f_1(\boldsymbol{x}_1) + \tfrac{\beta_k}{2}\|\mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b} + \boldsymbol{\lambda}_k/\beta_k\|^2, \\
\mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2} f_2(\mathbf{x}_2) + \tfrac{\beta_k}{2}\|\mathcal{A}_2(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2) - \mathbf{b} + \boldsymbol{\lambda}_k/\beta_k\|^2
\end{aligned}$$

$$\boxed{\min_{\mathbf{x}} f_1(\mathbf{x}) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{w}\|_F^2} \qquad \boxed{\min_{\mathbf{x}} f_2(\mathbf{x}) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{w}\|_F^2}$$

Proximal Operator

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\beta}{2}\|\mathbf{x} - \mathbf{w}\|^2 = \mathcal{S}_{\beta^{-1}}(\mathbf{w}),$$

$$\mathcal{S}_\varepsilon(x) = \operatorname{sgn}(x)\max(|x| - \varepsilon, 0).$$

$$\operatorname*{argmin}_{\mathbf{X}} \|\mathbf{X}\|_* + \frac{\varepsilon}{2}\|\mathbf{X} - \mathbf{W}\|_F^2 = \Theta_{\varepsilon^{-1}}(\mathbf{W}) = \mathbf{U}\mathcal{S}_{\varepsilon^{-1}}(\mathbf{\Sigma})\mathbf{V}^T,$$

where $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ is the singular value decomposition (SVD) of $\mathbf{W}$.

Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

Linearized Alternating Direction Method (LADM)

Introducing auxiliary variables:

$$
\begin{aligned}
\min_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} \quad & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2), \\
s.t. \quad & \mathbf{x}_1 = \mathbf{x}_3, \mathbf{x}_2 = \mathbf{x}_4, \mathcal{A}_1(\mathbf{x}_3) + \mathcal{A}_2(\mathbf{x}_4) = \mathbf{b}.
\end{aligned}
$$

$$
\begin{aligned}
& \tilde{\mathcal{L}}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \lambda_1, \lambda_2, \lambda_3) \\
= \quad & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \langle \lambda_1, \mathbf{x}_1 - \mathbf{x}_3 \rangle + \langle \lambda_2, \mathbf{x}_2 - \mathbf{x}_4 \rangle + \langle \lambda_3, \mathcal{A}_1(\mathbf{x}_3) + \mathcal{A}_2(\mathbf{x}_4) - \mathbf{b} \rangle \\
& + \frac{\beta}{2} \left(\|\mathbf{x}_1 - \mathbf{x}_3\|_F^2 + \|\mathbf{x}_2 - \mathbf{x}_4\|_F^2 + \|\mathcal{A}_1(\mathbf{x}_3) + \mathcal{A}_2(\mathbf{x}_4) - \mathbf{b}\|_F^2 \right),
\end{aligned}
$$

Three drawbacks:

1. More blocks $\longrightarrow$ more memory & slower convergence.

2. Matrix inversion is expensive.

3. Convergence is NOT guaranteed!

Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

Linearized Alternating Direction Method (LADM)

$$
\begin{aligned}
\mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) + \frac{\beta_k}{2}\|\mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b} + \lambda_k/\beta_k\|^2, \\
\mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2} f_2(\mathbf{x}_2) + \frac{\beta_k}{2}\|\mathcal{A}_2(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2) - \mathbf{b} + \lambda_k/\beta_k\|^2
\end{aligned}
$$

$$\boxed{\min_{\mathbf{x}} f_1(\mathbf{x}) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{w}\|_F^2} \qquad \boxed{\min_{\mathbf{x}} f_2(\mathbf{x}) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{w}\|_F^2}$$

- Linearize the quadratic term

$$
\begin{aligned}
\mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) + \langle \mathcal{A}_1^*(\lambda_k) + \beta_k \mathcal{A}_1^*(\mathcal{A}_1(\mathbf{x}_1^k) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b}), \mathbf{x}_1 - \mathbf{x}_1^k\rangle \\
&\quad + \frac{\beta_k\eta_1}{2}\|\mathbf{x}_1 - \mathbf{x}_1^k\|^2 \\
&= \arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) \\
&\quad + \frac{\beta_k\eta_1}{2}\|\mathbf{x}_1 - \mathbf{x}_1^k + \mathcal{A}_1^*(\lambda_k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^k) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b}))/(\beta_k\eta_1)\|^2,
\end{aligned}
$$

$$\boxed{\langle \mathcal{A}^*(x), y\rangle = \langle x, \mathcal{A}(y)\rangle, \quad \forall x, y} \leftarrow \text{Adjoint Operator}$$

$$
\begin{aligned}
\mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2} f_2(\mathbf{x}_2) \\
&\quad + \frac{\beta_k\eta_2}{2}\|\mathbf{x}_2 - \mathbf{x}_2^k + \mathcal{A}_2^*(\lambda_k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b}))/(\beta_k\eta_2)\|^2.
\end{aligned}
$$

Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

LADM with Adaptive Penalty (LADMAP)

Theorem: If $\{\beta_k\}$ is non-decreasing and upper bounded, $\eta_i > \|\mathcal{A}_i\|^2$, $i = 1, 2$, then the sequence $\{(\mathbf{x}_1^k, \mathbf{x}_2^k, \lambda_k)\}$ converges to a KKT point of the model problem.

Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

LADM with Adaptive Penalty (LADMAP)

- Adaptive Penalty

$$
\begin{aligned}
\mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) \\
&\quad + \frac{\beta_k \eta_1}{2} \|\mathbf{x}_1 - \mathbf{x}_1^k + \mathcal{A}_1^*(\lambda_k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^k) + \mathcal{A}_2(\mathbf{x}_1^k) - \mathbf{b}))/(\beta_k \eta_1)\|^2, \\
\mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2} f_2(\mathbf{x}_2) \\
&\quad + \frac{\beta_k \eta_2}{2} \|\mathbf{x}_2 - \mathbf{x}_2^k + \mathcal{A}_2^*(\lambda_k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b}))/(\beta_k \eta_2)\|^2.
\end{aligned}
$$

$$\Downarrow$$

$$-\beta_k \eta_1(\mathbf{x}_1^{k+1} - \mathbf{x}_1^k) - \mathcal{A}_1^*(\lambda_k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^k) + \mathcal{A}_2(\mathbf{x}_1^k) - \mathbf{b})) \in \partial f_1(\mathbf{x}_1^{k+1})$$

$$-\beta_k \eta_2(\mathbf{x}_2^{k+1} - \mathbf{x}_2^k) - \mathcal{A}_2^*(\lambda_k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b})) \in \partial f_2(\mathbf{x}_2^{k+1})$$

KKT condition: $\exists(\mathbf{x}^*, \mathbf{y}^*, \lambda^*)$ such that

$$
\begin{gathered}
\mathcal{A}_1(\mathbf{x}_1^*) + \mathcal{A}_2(\mathbf{x}_2^*) - \mathbf{b} = \mathbf{0}, \\
-\mathcal{A}_1^*(\lambda^*) \in \partial f_1(\mathbf{x}_1^*), -\mathcal{A}_2^*(\lambda^*) \in \partial f_2(\mathbf{x}_2^*).
\end{gathered}
$$

Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

LADM with Adaptive Penalty (LADMAP)

Both $\beta_k\eta_1\|\mathbf{x}_1^{k+1}-\mathbf{x}_1^k\|/\|\mathcal{A}_1^*(\mathbf{b})\|$ and $\beta_k\eta_2\|\mathbf{x}_2^{k+1}-\mathbf{x}_2^k\|/\|\mathcal{A}_2^*(\mathbf{b})\|$ should be small.

$$\eta_i = \|\mathcal{A}_i\|^2 \quad \Rightarrow \quad \text{Approximate } \|\mathcal{A}_i^*(\mathbf{b})\| \text{ by } \sqrt{\eta_i}\|\mathbf{b}\|$$

- Adaptive Penalty

$$\beta_{k+1} = \min(\beta_{\max}, \rho\beta_k),$$

$$\rho = \begin{cases} \rho_0, & \text{if } \beta_k \max(\sqrt{\eta_1}\|\mathbf{x}_1^{k+1}-\mathbf{x}_1^k\|_F, \sqrt{\eta_2}\|\mathbf{x}_2^{k+1}-\mathbf{x}_2^k\|_F)/\|\mathbf{b}\|_F < \varepsilon_2, \\ 1, & \text{otherwise,} \end{cases}$$

where $\rho_0 \geq 1$ is a constant.

- Loop until

$$\|\mathcal{A}_1(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2^{k+1}) - \mathbf{b}\|_F < \varepsilon_1,$$

$$\beta_k \max(\sqrt{\eta_1}\|\mathbf{x}_1^{k+1}-\mathbf{x}_1^k\|_F, \sqrt{\eta_2}\|\mathbf{x}_2^{k+1}-\mathbf{x}_2^k\|)/\|\mathbf{b}\|_F < \varepsilon_2.$$

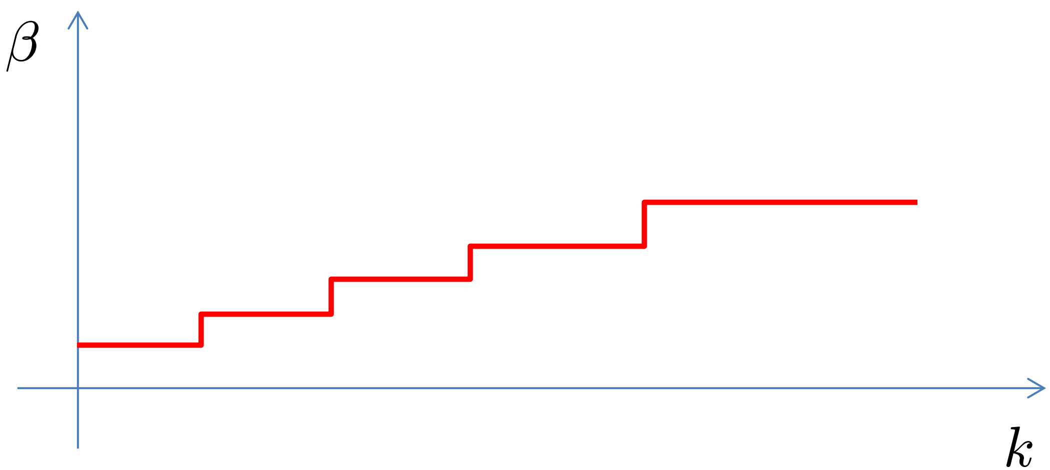
Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

LADM with Adaptive Penalty (LADMAP)

- Choice of parameters

1. $\beta_0 = \alpha\varepsilon_2$, where $\alpha \propto$ the size of $\mathbf{b}$. β_0 should not be too large, so that β_k increases in the first few iterations.

2. $\rho_0 \geq 1$ should be chosen such that β_k increases steadily (but not necessarily every iteration).



Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

LADM with Adaptive Penalty (LADMAP)

- An example (LRR):

$$\min_{Z,E} \|Z\|_* + \mu\|E\|_1, \quad s.t. \quad X = XZ + E.$$

$$\mathcal{A}_1(Z) = XZ, \quad \mathcal{A}_2(E) = E.$$

$$\mathcal{A}_1^*(Z) = X^T Z, \quad \mathcal{A}_2^*(E) = E, \eta_1 = \|X\|_2^2, \eta_2 = 1.$$

Lin et al., *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation,* NIPS 2011.

Experiment

Table 1: Comparison among APG, ADM, LADM and LADMAP on the synthetic data. For each quadruple $(s, p, d, \tilde{r})$, the LRR problem, with $\mu = 0.1$, was solved for the same data using different algorithms. We present typical running time (in $\times 10^3$ seconds), iteration number, relative error (%) of output solution $(\hat{\mathbf{E}}, \hat{\mathbf{Z}})$ and the clustering accuracy (%) of tested algorithms, respectively.

Size $(s, p, d, \tilde{r})$	Method	Time	Iter.	$\frac{\|\hat{\mathbf{Z}}-\mathbf{Z}_0\|}{\|\mathbf{Z}_0\|}$	$\frac{\|\hat{\mathbf{E}}-\mathbf{E}_0\|}{\|\mathbf{E}_0\|}$	Acc.
(10, 20,200, 5)	APG	0.0332	110	2.2079	1.5096	81.5
	ADM	0.0529	176	0.5491	0.5093	**90.0**
	LADM	0.0603	194	**0.5480**	**0.5024**	**90.0**
	LADMAP	0.0145	**46**	**0.5480**	**0.5024**	**90.0**
(15, 20,300, 5)	APG	0.0869	106	2.4824	1.0341	80.0
	ADM	0.1526	185	0.6519	0.4078	83.7
	LADM	0.2943	363	**0.6518**	**0.4076**	**86.7**
	LADMAP	0.0336	**41**	**0.6518**	**0.4076**	**86.7**
(20, 25, 500, 5)	APG	1.8837	117	2.8905	2.4017	72.4
	ADM	3.7139	225	1.1191	1.0170	80.0
	LADM	8.1574	508	**0.6379**	**0.4268**	80.0
	LADMAP	0.7762	**40**	**0.6379**	**0.4268**	**84.6**
(30, 30, 900, 5)	APG	6.1252	116	3.0667	0.9199	69.4
	ADM	11.7185	220	0.6865	0.4866	**76.0**
	LADM	N.A.	N.A.	N.A.	N.A.	N.A.
	LADMAP	2.3891	**44**	**0.6864**	**0.4294**	**80.1**

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

- Model problem:

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}.$$

$$\min_{\mathbf{X}} \|\mathbf{X}\|_* + \frac{1}{2\mu}\|\mathbf{b} - \mathcal{P}\ \ (\mathbf{X})\|^2, \quad s.t. \quad \mathbf{X} \geq 0,$$

$$\Downarrow$$

$$\min_{\mathbf{X},\mathbf{e}} \|\mathbf{X}\|_* + \frac{1}{2\mu}\|\mathbf{e}\|^2, \quad s.t. \quad \mathbf{b} = \mathcal{P}\ \ (\mathbf{X}) + \mathbf{e},\ \mathbf{X} \geq 0,$$

$$\Downarrow$$

$$\min_{\mathbf{X},\mathbf{Y},\mathbf{e}} \|\mathbf{X}\|_* + \frac{1}{2\mu}\|\mathbf{e}\|^2, \quad s.t. \quad \mathbf{b} = \mathcal{P}\ \ (\mathbf{Y}) + \mathbf{e},\ \mathbf{X} = \mathbf{Y},\ \mathbf{Y} \geq 0,$$

$$\Downarrow$$

$$\min_{\mathbf{X},\mathbf{Y},\mathbf{e}} \|\mathbf{X}\|_* + \frac{1}{2\mu}\|\mathbf{e}\|^2 + \chi_{\mathbf{Y}\geq 0}(\mathbf{Y}), \quad s.t. \quad \mathbf{b} = \mathcal{P}\ \ (\mathbf{Y}) + \mathbf{e},\ \mathbf{X} = \mathbf{Y}.$$

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

- Can we naively generalize two-block LADMAP for multi-block problems?

$\boxed{\text{No!}}$

Actually, the naive generalization of LADMAP may be divergent, e.g., when applied to the following problem with $n \geq 5$:

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_n} \sum_{i=1}^{n} \|\mathbf{x}_i\|_1, \quad s.t. \quad \sum_{i=1}^{n} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}.$$

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

C. Chen et al. *The Direct Extension of ADMM for Multi-block Convex Minimization Problems is Not Necessarily Convergent*. Preprint.

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

$$\mathbf{x}_i^{k+1} = \operatorname*{argmin}_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\eta_i \beta_k}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \mathcal{A}_i^* \left(\lambda^k + \beta_k \left(\boxed{\sum_{j=1}^{n} \mathcal{A}_i(\mathbf{x}_j^k)} - \mathbf{b} \right) \right) / (\eta_i \beta_k) \right\|^2,$$

$$i = 1, \cdots, n,$$

$$\boxed{\sum_{j=1}^{i-1} \mathcal{A}_i(\mathbf{x}_j^{k+1}) + \sum_{j=i}^{n} \mathcal{A}_i(\mathbf{x}_j^k)}$$

$$\lambda^{k+1} = \lambda^k + \beta_k \left(\sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b} \right),$$

Parallel!

$$\beta_{k+1} = \min(\beta_{\max}, \rho \beta_k),$$

where

$$\rho = \begin{cases} \rho_0, & \text{if } \beta_k \max\left(\left\{ \sqrt{\eta_i} \left\| \mathbf{x}_i^{k+1} - \mathbf{x}_i^k \right\|, i = 1, \cdots, n \right\} \right) / \|\mathbf{b}\| < \varepsilon_2, \\ 1, & \text{otherwise}, \end{cases}$$

with $\rho_0 > 1$ being a constant and $0 < \varepsilon_2 \ll 1$ being a threshold.

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

Theorem: If $\{\beta_k\}$ is non-decreasing and upper bounded, $\boxed{\eta_i > n\|\mathcal{A}_i\|^2}$, $i = 1, \cdots, n$, then $\{(\{\mathbf{x}_i^k\}, \lambda^k)\}$ generated by LADMPSAP converges to a KKT point of the problem.

Remark: When $n = 2$, LADMPSAP is weaker than LADMAP:

$$\eta_i > 2\|A_i\|^2 \text{ vs. } \eta_i > \|A_i\|^2.$$

- Related work: He & Yuan, *Linearized Alternating Direction Method with Gaussian Back Substitution for Separable Convex Programming*, preprint.

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

- Model problem:

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \ \ s.t. \ \ \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}, \ \ \mathbf{x}_i \in X_i, i = 1, \cdots, n,$$

where $X_i \subseteq \mathbb{R}^{d_i}$ is a closed convex set.

$$\Downarrow$$

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_{2n}} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \sum_{i=n+1}^{2n} \chi_{\mathbf{x}_i \in X_{i-n}}(\mathbf{x}_i), \ \ s.t. \ \ \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}, \mathbf{x}_i = \mathbf{x}_{n+i}, i = 1, \cdots, n.$$

Theorem: If $\{\beta_k\}$ is non-decreasing and upper bounded, $\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{2n}$ are auxiliary variables, $\boxed{\eta_i > n\|\mathcal{A}_i\|^2 + 2, \ \eta_{n+i} > 2, \ i = 1, \cdots, n}$, then $\{(\{\mathbf{x}_i^k\}, \lambda^k)\}$ generated by LADMPSAP converges to a KKT point of the problem.

$$\boxed{\eta_i > 2n(\|\mathcal{A}_i\|^2 + 1), \eta_{n+i} > 2n, i = 1, \cdots, n}$$

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

Experiment

$$\min_{\mathbf{X}} \|\mathbf{X}\|_* + \frac{1}{2\mu}\|\mathbf{b} - \mathcal{P}\ \ (\mathbf{X})\|^2, \quad s.t. \quad \mathbf{X} \geq 0,$$

$$\Downarrow$$

$$\min_{\mathbf{X},\mathbf{Y},\mathbf{e}} \|\mathbf{X}\|_* + \frac{1}{2\mu}\|\mathbf{e}\|^2 + \chi_{\mathbf{Y}\geq 0}(\mathbf{Y}), \quad s.t. \quad \mathbf{b} = \mathcal{P}\ \ (\mathbf{Y}) + \mathbf{e},\ \mathbf{X} = \mathbf{Y}.$$

(a) Original (b) Corrupted (c) FPCA (d) LADM (e) LADMPSAP

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

Experiment

Table 1: Numerical comparison on the NMC problem with synthetic data, average of 10 runs. q, t and d_r denote, respectively, sample ratio, the number of measurements $t = q(mn)$ and the "degree of freedom" defined by $d_r = r(m+n-r)$ for a matrix with rank r and q. Here we set $m = n$ and fix $r = 10$ in all the tests.

X			LADM				LADMPSAP			
n	q	t/d_r	Iter.	Time(s)	RelErr	FA	Iter.	Time(s)	RelErr	FA
1000	20%	10.05	375	177.92	1.35E-5	6.21E-4	58	24.94	9.67E-6	0
	10%	5.03	1000	459.70	4.60E-5	6.50E-4	109	42.68	1.72E-5	0
5000	20%	50.05	229	1613.68	1.08E-5	1.93E-4	49	369.96	9.05E-6	0
	10%	25.03	539	2028.14	1.20E-5	7.70E-5	89	365.26	9.76E-6	0
10000	10%	50.03	463	6679.59	1.11E-5	4.18E-5	89	1584.39	1.03E-5	0

Table 1: Numerical comparison on the image inpainting problem.

Method	#Iter.	Time(s)	PSNR	FA
FPCA	179	228.99	27.77dB	9.41E-4
LADM	228	207.95	26.98dB	2.92E-3
LADMPSAP	143	134.89	31.39dB	0

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

Enhanced convergence results:

Theorem 1: If $\{\beta_k\}$ is non-decreasing and $\sum\limits_{k=1}^{+\infty} \beta_k^{-1} = +\infty$, $\eta_i > n\|\mathcal{A}_i\|^2$, $\boxed{\partial f_i(\mathbf{x}) \text{ is bounded}}$, $i = 1, \cdots, n$, then the sequence $\{\mathbf{x}_i^k\}$ generated by LADMPSAP converges to an optimal solution to the model problem.

Theorem 2: If $\{\beta_k\}$ is non-decreasing, $\eta_i > n\|\mathcal{A}_i\|^2$, $\boxed{\partial f_i(\mathbf{x}) \text{ is bounded}}$, $i = 1, \cdots, n$, then $\sum\limits_{k=1}^{+\infty} \beta_k^{-1} = +\infty$ is also the necessary condition for the global convergence of $\{\mathbf{x}_i^k\}$ generated by LADMPSAP to an optimal solution to the model problem.

With the above analysis, when all the subgradients of the component objective functions are bounded we can remove the upper bound $\beta_{\max}$.

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

Define $\mathbf{x} = (\mathbf{x}_1^T, \cdots, \mathbf{x}_n^T)^T$, $\mathbf{x}^* = ((\mathbf{x}_1^*)^T, \cdots, (\mathbf{x}_2^*)^T)^T$ and $f(\mathbf{x}) = \sum\limits_{i=1}^{n} f_i(\mathbf{x}_i)$, where $(\mathbf{x}_1^*, \cdots, \mathbf{x}_2^*, \lambda^*)$ is a KKT point of the model problem.

Proposition: $\tilde{\mathbf{x}}$ is an optimal solution to the model problem iff there exists $\alpha > 0$, such that

$$f(\tilde{\mathbf{x}}) - f(\mathbf{x}^*) + \sum_{i=1}^{n} \langle \mathcal{A}_i^*(\lambda^*), \tilde{\mathbf{x}}_i - \mathbf{x}_i^* \rangle + \alpha \left\| \sum_{i=1}^{n} \mathcal{A}_i(\tilde{\mathbf{x}}_i) - \mathbf{b} \right\|^2 = 0.$$

Our criterion for checking the optimality of a solution is much simpler than that in He et al. 2011, which has to compare with all $(\mathbf{x}_1, \cdots, \mathbf{x}_n, \lambda) \in \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_n} \times \mathbb{R}^m$.

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

B. S. He and X. Yuan. *On the O(1/t) convergence rate of alternating direction method*. Preprint, 2011.

LADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

Theorem 3: Define $\bar{\mathbf{x}}^K = \sum_{k=0}^{K} \gamma_k \mathbf{x}^{k+1}$, where $\gamma_k = \beta_k^{-1} / \sum_{j=0}^{K} \beta_j^{-1}$. Then

$$f(\bar{\mathbf{x}}^K) - f(\mathbf{x}^*) + \sum_{i=1}^{n} \langle \mathcal{A}_i^*(\lambda^*), \bar{\mathbf{x}}_i^K - \mathbf{x}_i^* \rangle + \frac{\alpha\beta_0}{2} \left\| \sum_{i=1}^{n} \mathcal{A}_i(\bar{\mathbf{x}}_i^K) - \mathbf{b} \right\|^2 <= C_0 / \left(2 \sum_{k=0}^{K} \beta_k^{-1} \right), \tag{1}$$

where $\alpha^{-1} = (n+1) \max\left(1, \left\{ \frac{\|\mathcal{A}_i\|^2}{\eta_i - n\|\mathcal{A}_i\|^2}, i = 1, \cdots, n \right\}\right)$ and $C_0 = \sum_{i=1}^{n} \eta_i \left\| \mathbf{x}_i^0 - \mathbf{x}_i^* \right\|^2$ $+ \beta_0^{-2} \left\| \lambda^0 - \lambda^* \right\|^2$.

A much simpler proof of convergence rate (in ergodic sense)!

Lin et al., *Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning*, ML, 2015.

B. S. He and X. Yuan. *On the O(1/t) convergence rate of alternating direction method*. Preprint, 2011.

Proximal LADMPSAP

- Even more general problem:

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}.$$

$$f_i(\mathbf{x}_i) = g_i(\mathbf{x}_i) + h_i(\mathbf{x}_i),$$

where both g_i and h_i are convex, g_i is $C^{1,1}$:

$$\|\nabla g_i(\mathbf{x}) - \nabla g_i(\mathbf{y})\| \leq L_i \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^{d_i},$$

and h_i may not be differentiable but its proximal operation is easily solvable.

Proximal LADMPSAP

- Linearize the augmented term to obtain:

$$\mathbf{x}_i^{k+1} = \operatorname*{argmin}_{\mathbf{x}_i} h_i(\mathbf{x}_i) + g_i(\mathbf{x}_i) + \frac{\sigma_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \mathcal{A}_i^\dagger(\hat{\lambda}^k)/\sigma_i^{(k)} \right\|^2, \quad i = 1, \cdots, n,$$

- Further linearize g_i:

$$\begin{aligned} \mathbf{x}_i^{k+1} &= \operatorname*{argmin}_{\mathbf{x}_i} h_i(\mathbf{x}_i) + g_i(\mathbf{x}_i^k) + \frac{\sigma_i^{(k)}}{2} \left\| \mathcal{A}_i^\dagger(\hat{\lambda}^k)/\sigma_i^{(k)} \right\|^2 \\ &\quad + \langle \nabla g_i(\mathbf{x}_i^k) + \mathcal{A}_i^\dagger(\hat{\lambda}^k), \mathbf{x}_i - \mathbf{x}_i^k \rangle + \frac{\tau_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k \right\|^2 \\ &= \operatorname*{argmin}_{\mathbf{x}_i} h_i(\mathbf{x}_i) + \tfrac{\tau_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \tfrac{1}{\tau_i^{(k)}} [\mathcal{A}_i^\dagger(\hat{\lambda}^k) + \nabla g_i(\mathbf{x}_i^k)] \right\|^2. \end{aligned}$$

- Convergence condition:

$\tau_i^{(k)} = T_i + \beta_k \eta_i$, where $T_i \geq L_i$ and $\eta_i > n\|\mathcal{A}_i\|^2$ are both positive constants.

Experiment

- Group Sparse Logistic Regression with Overlap

$$\min_{\mathbf{w},b} \frac{1}{s}\sum_{i=1}^{s} \log\left(1+\exp\left(-y_i(\mathbf{w}^T\mathbf{x}_i+b)\right)\right)+\mu\sum_{j=1}^{t}\|\mathbf{S}_j\mathbf{w}\|, \tag{1}$$

where $\mathbf{x}_i$ and y_i, $i=1,\cdots,s$, are the training data and labels, respectively, and $\mathbf{w}$ and b parameterize the linear classifier. $\mathbf{S}_j$, $j=1,\cdots,t$, are the selection matrices, with only one 1 at each row and the rest entries are all zeros. The groups of entries, $\mathbf{S}_j\mathbf{w}$, $j=1,\cdots,t$, may overlap each other.

Introducing $\bar{\mathbf{w}}=(\mathbf{w}^T,b)^T$, $\bar{\mathbf{x}}_i=(\mathbf{x}_i^T,1)^T$, $\mathbf{z}=(\mathbf{z}_1^T,\mathbf{z}_2^T,\cdots,\mathbf{z}_t^T)^T$, and $\bar{\mathbf{S}}=(\mathbf{S},\mathbf{0})$, where $\mathbf{S}=(\mathbf{S}_1^T,\cdots,\mathbf{S}_t^T)^T$, (1) can be rewritten as

$$\min_{\bar{\mathbf{w}},\mathbf{z}} \frac{1}{s}\sum_{i=1}^{s} \log\left(1+\exp\left(-y_i(\bar{\mathbf{w}}^T\bar{\mathbf{x}}_i)\right)\right)+\mu\sum_{j=1}^{t}\|\mathbf{z}_j\|, \quad s.t. \quad \mathbf{z}=\bar{\mathbf{S}}\bar{\mathbf{w}}, \tag{2}$$

The Lipschitz constant of the gradient of logistic function with respect to $\bar{\mathbf{w}}$ can be proven to be $L_{\bar{w}} \cdot \ \frac{1}{4s}\|\bar{\mathbf{X}}\|_2^2$, where $\bar{\mathbf{X}}=(\bar{\mathbf{x}}_1,\bar{\mathbf{x}}_2,\cdots,\bar{\mathbf{x}}_s)$.

Experiment

$(s,\, p,\, t,\, q)$	Method	Time	#Iter.	$\frac{\|\hat{\bar{\mathbf{w}}}-\bar{\mathbf{w}}^*\|}{\|\bar{\mathbf{w}}^*\|}$	$\frac{\|\hat{\mathbf{z}}-\mathbf{z}^*\|}{\|\mathbf{z}^*\|}$
(300, 901, 100, 10)	ADM	294.15	43	0.4800	0.4790
	LADM	229.03	43	0.5331	0.5320
	LADMPS	105.50	47	0.2088	0.2094
	LADMPSAP	57.46	39	0.0371	0.0368
	pLADMPSAP	**1.97**	141	**0.0112**	**0.0112**
(450, 1351, 150, 15)	ADM	450.96	33	0.4337	0.4343
	LADM	437.12	36	0.5126	0.5133
	LADMPS	201.30	39	0.1938	0.1937
	LADMPSAP	136.64	37	0.0321	0.0306
	pLADMPSAP	**4.16**	150	**0.0131**	**0.0131**
(600, 1801, 200, 20)	ADM	1617.09	62	1.4299	1.4365
	LADM	1486.23	63	1.5200	1.5279
	LADMPS	494.52	46	0.4915	0.4936
	LADMPSAP	216.45	32	0.0787	0.0783
	pLADMPSAP	**5.77**	127	**0.0276**	**0.0277**