Chapter 7. Constrained Optimization

- Algorithms for constrained optimization
- Frank-Wolfe method
- Alternating direction method (ADM)
- Linearized alternating direction method (LADM)
- Proximal Linearized alternating direction method (PLADM)
- Coordinate descent and block coordinate descent

Algorithms for constrained optimization

- Projections

$$\min \quad f(\mathbf{x}) \tag{1}$$

$$s.t. \quad \mathbf{x} \in \Omega. \tag{2}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

where $\mathbf{d}^{(k)}$ is typically a function of $\nabla f(\mathbf{x}^{(k)})$, may not work. A simple modification involves the introduction of a projection:

$$\mathbf{x}^{(k+1)} = \mathcal{P}_\Omega(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}).$$

The projected version of the gradient algorithm has the form

$$\mathbf{x}^{(k+1)} = \mathcal{P}_\Omega(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})). \tag{3}$$

We refer to the above as the *projected gradient algorithm*.

Algorithms for constrained optimization

- Analysis

Properties of projection onto a convex set $\Omega \subset \mathbb{R}^n$. Denote $[\mathbf{x}]^+ = P_\Omega(\mathbf{x})$.

1. Given some $\mathbf{x} \in \mathbb{R}^n$, a vector $\mathbf{x}^* \in \Omega$ is equal to $[\mathbf{x}]^+$ iff

$$\langle \mathbf{y} - \mathbf{x}^*, \mathbf{x} - \mathbf{x}^* \rangle \leq 0, \quad \forall \mathbf{y} \in \Omega.$$

2. The projection is nonexpansive, i.e.,

$$\|[\mathbf{x}]^+ - [\mathbf{y}]^+\| \leq \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

Algorithms for constrained optimization

- Analysis

Proposition 1. *If both f and Ω are convex and $\mathbf{x}^*$ is a stationary point of the iteration (3), i.e.,*

$$\mathbf{x}^* = [\mathbf{x}^* - \alpha \nabla f(\mathbf{x}^*)]^+,$$

then $\mathbf{x}^*$ is an optimal solution to problem (1)-(2).

Theorem 1. *Assume that both f and Ω are convex and f is further L-smooth. Let $\{\mathbf{x}^k\}$ be a sequence generated by the iteration (3) with $\alpha_k \equiv \alpha$, where $\alpha \in (0, 2/L)$. Then every limit point of $\{\mathbf{x}^k\}$ is stationary.*

Algorithms for constrained optimization

- Analysis

Proof: By the property of projection,

$$\langle \mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k) - \mathbf{x}^{k+1}, \mathbf{x} - \mathbf{x}^{k+1} \rangle \leq 0, \quad \forall \mathbf{x} \in \Omega.$$

Applying the above with $\mathbf{x} = \mathbf{x}^k$, we obtain

$$\langle \nabla f(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \rangle \leq \frac{1}{\alpha} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2.$$

Then by the L-smoothness of f,

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leq \langle \nabla f(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \rangle + \frac{L}{2} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 \leq \left(\frac{L}{2} - \frac{1}{\alpha} \right) \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2.$$

Since $\alpha < 2/L$, the RHS of the above is nonpositive. So if $\{\mathbf{x}^k\}$ has a limit point, the LHS tends to 0 and thus $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \to 0$. This implies that for every limit point $\tilde{\mathbf{x}}$ of $\{\mathbf{x}^k\}$ we have $[\tilde{\mathbf{x}} - \alpha \nabla f(\tilde{\mathbf{x}})]^+ = \tilde{\mathbf{x}}$. So $\tilde{\mathbf{x}}$ is stationary.

Algorithms for constrained optimization

- Projections - Example

$$\min \quad \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}$$

$$s.t. \quad \|\mathbf{x}\|^2 = 1,$$

where $\mathbf{Q} = \mathbf{Q}^T \succ \mathbf{0}$. Suppose that we apply a fixed-step-size projected gradient algorithm to this problem.

Algorithms for constrained optimization

- Projected Gradient Methods with Linear Constraints

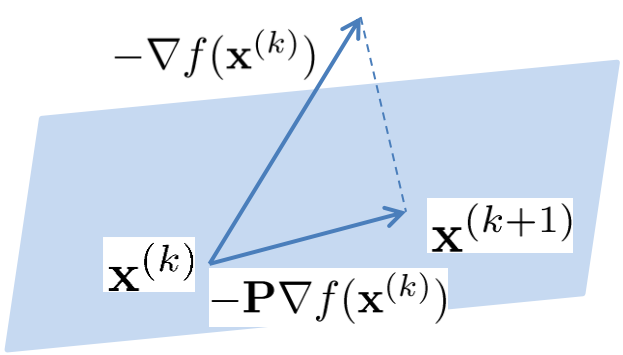
$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $m < n$, $\operatorname{rank} \mathbf{A} = m$, $\mathbf{b} \in \mathbb{R}^m$.

$$\mathbf{P} = \mathbf{I}_n - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A},$$

$$\mathcal{P}_\Omega(\mathbf{x}) = \mathbf{x} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{P}\mathbf{x} + \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}.$$

$$\begin{aligned} \mathbf{x}^{(k)} \in \Omega \Longrightarrow \mathbf{x}^{(k+1)} =& \mathcal{P}_\Omega(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})) \\ =& [\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}](\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})) \\ &+ \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \\ =& \mathbf{x}^{(k)} - \alpha_k \mathbf{P}\nabla f(\mathbf{x}^{(k)}). \end{aligned}$$



Algorithms for constrained optimization

- Solving equality constrained problems via the dual

Another approach to solving (1) is to solve the dual, and then recover the optimal primal variable $\mathbf{x}^*$. The dual function of (1) is:

$$\begin{aligned} g(\boldsymbol{\nu}) &= -\mathbf{b}^T\boldsymbol{\nu} + \inf_{\mathbf{x}}(f(\mathbf{x}) + \boldsymbol{\nu}^T\mathbf{A}\mathbf{x}) \\ &= -\mathbf{b}^T\boldsymbol{\nu} - \sup_{\mathbf{x}}((-\mathbf{A}^T\boldsymbol{\nu})^T\mathbf{x} - f(\mathbf{x})) = -\mathbf{b}^T\boldsymbol{\nu} - f^*(-\mathbf{A}^T\boldsymbol{\nu}), \end{aligned}$$

where f^* is the conjugate of f. So the dual problem is:

$$\max_{\boldsymbol{\nu}} -\mathbf{b}^T\boldsymbol{\nu} - f^*(-\mathbf{A}^T\boldsymbol{\nu}).$$

Since by assumption there is an optimal point, the problem is strictly feasible, so Slater's condition holds. Therefore strong duality holds, and the dual optimum is attained, i.e., there exists a $\boldsymbol{\nu}^*$ with $g(\boldsymbol{\nu}^*) = p^*$.

Once we find an optimal dual variable $\boldsymbol{\nu}^*$, we reconstruct an optimal primal solution $\mathbf{x}^*$ from it.

Algorithms for constrained optimization

- Solving equality constrained problems via the dual

Example: Equality constrained analytic center. We consider the problem

$$\min_{\mathbf{x}} \; f(\mathbf{x}) = -\sum_{i=1}^{n} \log x_i, \quad s.t. \; \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{p \times n}$, with implicit constraint $\mathbf{x} > \mathbf{0}$. Using

$$f^*(\mathbf{y}) = \sum_{i=1}^{n} (-1 - \log(-y_i)) = -n - \sum_{i=1}^{n} \log(-y_i)$$

(with $\mathrm{dom} f^* = -\mathbb{R}^n_{++}$), the dual problem is

$$\max_{\boldsymbol{\nu}} g(\boldsymbol{\nu}) = -\mathbf{b}^T \boldsymbol{\nu} + n + \sum_{i=1}^{n} \log(\mathbf{A}^T \boldsymbol{\nu})_i, \tag{5}$$

with implicit constraint $\mathbf{A}^T \boldsymbol{\nu} > \mathbf{0}$.

Algorithms for constrained optimization

- Solving equality constrained problems via the dual

Here we can easily solve the dual feasibility equation, i.e., find the $\mathbf{x}$ that minimizes $L(\mathbf{x}, \boldsymbol{\nu})$:

$$\nabla f(\mathbf{x}) + \mathbf{A}^T \boldsymbol{\nu} = -(1/x_1, \cdots, 1/x_n) + \mathbf{A}^T \boldsymbol{\nu} = \mathbf{0},$$

and so

$$x_i(\boldsymbol{\nu}) = 1/(\mathbf{A}^T \boldsymbol{\nu})_i. \tag{6}$$

To solve the equality constrained analytic centering problem (4), we solve the (unconstrained) dual problem (5), and then recover the optimal solution of (4) via (6).

Algorithms for constrained optimization

- Dual Derivatives and Subgradients – Model Problem

We focus on the primal problem:

$$\begin{aligned} &\min_{\mathbf{x}} \; f(\mathbf{x}) \\ &s.t. \; \mathbf{x} \in \mathcal{X}, g_j(\mathbf{x}) \leq 0, j = 1, \cdots, r, \end{aligned} \tag{10}$$

and its dual

$$\begin{aligned} &\max_{\boldsymbol{\mu}} \; q(\boldsymbol{\mu}) \\ &s.t. \; \boldsymbol{\mu} \geq \mathbf{0}, \end{aligned} \tag{11}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g_j : \mathbb{R}^n \to \mathbb{R}$ are given functions, $\mathcal{X}$ is a subset of $\mathbb{R}^n$, and

$$q(\boldsymbol{\mu}) = \inf_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x}, \boldsymbol{\mu}) = \inf_{\mathbf{x} \in \mathcal{X}} \left\{ f(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}) \right\}$$

is the dual function.

Algorithms for constrained optimization

- Dual Derivatives and Subgradients

For a given $\boldsymbol{\mu} \in \mathbb{R}^r$, suppose that $\mathbf{x}_\mu$ minimizes the Lagrangian $L(\mathbf{x}, \boldsymbol{\mu})$,

$$\mathbf{x}_\mu = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x}, \boldsymbol{\mu}) = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{X}} \left\{ f(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}) \right\}.$$

An important fact is that $\mathbf{g}(\mathbf{x}_\mu)$ is a supergradient of the dual function q at $\boldsymbol{\mu}$:

obtained *essentially at no cost*

$$q(\bar{\boldsymbol{\mu}}) \leq q(\boldsymbol{\mu}) + (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \mathbf{g}(\mathbf{x}_\mu), \quad \forall \bar{\boldsymbol{\mu}} \in \mathbb{R}^r. \tag{10}$$

To see this, we use the definition of q and $\mathbf{x}_\mu$ to write for all $\bar{\boldsymbol{\mu}} \in \mathbb{R}^r$,

$$\begin{aligned} q(\bar{\boldsymbol{\mu}}) = \inf_{\mathbf{x} \in \mathcal{X}} \left\{ f(\mathbf{x}) + \bar{\boldsymbol{\mu}}^T \mathbf{g}(\mathbf{x}) \right\} &\leq f(\mathbf{x}_\mu) + \bar{\boldsymbol{\mu}}^T \mathbf{g}(\mathbf{x}_\mu) \\ &= f(\mathbf{x}_\mu) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}_\mu) + (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \mathbf{g}(\mathbf{x}_\mu) \\ &= q(\boldsymbol{\mu}) + (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \mathbf{g}(\mathbf{x}_\mu). \end{aligned}$$

Note that this calculation is valid for all $\boldsymbol{\mu} \in \mathbb{R}^r$ for which there is a minimizing vector $\mathbf{x}_\mu$, regardless of whether $\boldsymbol{\mu} \geq \mathbf{0}$.

Algorithms for constrained optimization

- Dual Derivatives and Subgradients

Proposition 1. *Let $\mathcal{X}$ be a compact set, and let f and $\mathbf{g}$ be continuous over $\mathcal{X}$. Assume also that for every $\boldsymbol{\mu} \in \mathbb{R}^r$, $L(\mathbf{x}, \boldsymbol{\mu})$ is minimized over $\mathbf{x} \in \mathcal{X}$ at a unique point $\mathbf{x}_\mu$. Then q is everywhere continuously differentiable and*

$$\nabla q(\boldsymbol{\mu}) = \mathbf{g}(\mathbf{x}_\mu), \quad \forall \boldsymbol{\mu} \in \mathbb{R}^r.$$

Algorithms for constrained optimization

- Dual Ascend

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}), \\ &s.t. \quad \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned} \tag{10}$$

The Lagrangian is

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}),$$

and the dual function is

$$d(\boldsymbol{\lambda}) = \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}) = -f^*(-\mathbf{A}^T - \mathbf{y}) - \mathbf{b}^T\boldsymbol{\lambda}.$$

The dual problem is

$$\max_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}).$$

Assuming that strong duality holds, we can recover $\mathbf{x}^*$ via

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}^*),$$

Algorithms for constrained optimization

- Dual Ascend

We first find $\mathbf{x}^+ = \underset{\mathbf{x}}{\operatorname{argmin}}\, L(\mathbf{x}, \boldsymbol{\lambda})$; then we have $\nabla d(\boldsymbol{\lambda}) = \mathbf{A}\mathbf{x}^+ - \mathbf{b}$. The dual ascent method:

may not have a solution!

primal update

$$\mathbf{x}_{k+1} = \boxed{\underset{\mathbf{x}}{\operatorname{argmin}}\, L(\mathbf{x}, \boldsymbol{\lambda}_k)}, \tag{10a}$$

dual update

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}), \tag{10b}$$

Algorithms for constrained optimization

- Dual Decomposition

$$f(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i),$$

where $\mathbf{x} = (\mathbf{x}_1^T, \cdots, \mathbf{x}_N^T)^T$. $\mathbf{A} = [\mathbf{A}_1 \cdots \mathbf{A}_N]$. The Lagrangian can be written as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} L_i(\mathbf{x}_i, \boldsymbol{\lambda}) = \sum_{i=1}^{N} (f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T \mathbf{A}_i \mathbf{x}_i - (1/N)\boldsymbol{\lambda}^T \mathbf{b}).$$

parallel

$$\boxed{\mathbf{x}_{i,k+1} = \underset{\mathbf{x}_i}{\operatorname{argmin}}\, L_i(\mathbf{x}_i, \boldsymbol{\lambda}_k),} \quad i = 1, ..., N, \tag{10a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}). \tag{10b}$$

Algorithms for constrained optimization

- Augmented Lagrangians and the Method of Multipliers

$$L_\beta(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + (\beta/2)\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2. \tag{10}$$

It can be viewed as the Lagrangian of

penalty parameter

$$\min_{\mathbf{x}} \; f(\mathbf{x}) + (\beta/2)\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2,$$
$$s.t. \; \mathbf{A}\mathbf{x} = \mathbf{b}.$$

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}} L_\beta(\mathbf{x}, \boldsymbol{\lambda}_k), \tag{11a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \beta(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}), \tag{11b}$$

Algorithms for constrained optimization

- Augmented Lagrangians and the Method of Multipliers – why the stepsize is β?

$$\mathbf{A}\mathbf{x}^* - \mathbf{b} = \mathbf{0}, \nabla f(\mathbf{x}^*) + \mathbf{A}^T \boldsymbol{\lambda}^* = \mathbf{0},$$

respectively. By definition, $\mathbf{x}_{k+1}$ minimizes $L_\beta(\mathbf{x}, \boldsymbol{\lambda}_k)$, so

$$\begin{aligned}
\mathbf{0} &= \nabla_x L_\beta(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_k) \\
&= \nabla_x f(\mathbf{x}_{k+1}) + \mathbf{A}^T(\boldsymbol{\lambda}_k + \beta(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}) \\
&= \nabla_x f(\mathbf{x}_{k+1}) + \mathbf{A}^T \boldsymbol{\lambda}_{k+1}.
\end{aligned}$$

We see that by using β as the step size in the dual update, the iterate $(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1})$ is dual feasible.

Algorithms for constrained optimization

- Augmented Lagrangian Method of Multipliers

For any $\boldsymbol{\lambda}$, we have $d(\boldsymbol{\lambda}) \leq d_\beta(\boldsymbol{\lambda})$. Moreover, for any $\boldsymbol{\lambda}$, we have $d_\beta(\boldsymbol{\lambda}) \leq f(\mathbf{x}^*)$. Since $d(\boldsymbol{\lambda}^*) = f(\mathbf{x}^*)$, we know $d(\boldsymbol{\lambda}^*) = d_\beta(\boldsymbol{\lambda}^*) = f(\mathbf{x}^*)$. We have a better smoothness property of $d_\beta(\boldsymbol{\lambda})$ than $d(\boldsymbol{\lambda})$.

Proposition 1. *Let $\mathcal{D}(\boldsymbol{\lambda})$ denote the optimal solution set of $\min_{\mathbf{x}} L_\beta(\mathbf{x}, \boldsymbol{\lambda})$. Then $\mathbf{A}\mathbf{x}$ is invariant over $\mathcal{D}(\boldsymbol{\lambda})$. Moreover, $d_\beta(\boldsymbol{\lambda})$ is differentiable and $\nabla d_\beta(\boldsymbol{\lambda}) = \mathbf{A}\mathbf{x}(\boldsymbol{\lambda}) - \mathbf{b}$, where $\mathbf{x}(\boldsymbol{\lambda}) \in \mathcal{D}(\boldsymbol{\lambda})$ is any minimizer of $\min_{\mathbf{x}} L_\beta(\mathbf{x}, \boldsymbol{\lambda})$. We also have that d_β is $\frac{1}{\beta}$-smooth, i.e.,*

$$\|\nabla d_\beta(\boldsymbol{\lambda}) - \nabla d_\beta(\boldsymbol{\lambda}')\| \leq \frac{1}{\beta}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}'\|. \tag{10}$$

Algorithms for constrained optimization

- Penalty Methods

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &s.t.\ \mathbf{x} \in \Omega. \end{aligned} \tag{11}$$

$$\min\ f(\mathbf{x}) + \boxed{\gamma}\boxed{P(\mathbf{x})}, \tag{12}$$

penalty parameter

penalty function

Definition 1. *A function $P : \mathbb{R}^n \to \mathbb{R}$ is called a* penalty function *for the constrained optimization problem above if it satisfies the following three conditions:*

1. *P is continuous.*
2. *$P(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$.*
3. *$P(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is feasible (i.e., $\mathbf{x} \in \Omega$).*

Algorithms for constrained optimization

- Penalty Methods – Choice of penalty function

Example.

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ s.t. \quad & g_i(\mathbf{x}) \leq 0, i = 1, \cdots, p. \end{aligned}$$

A possible choice for P is

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

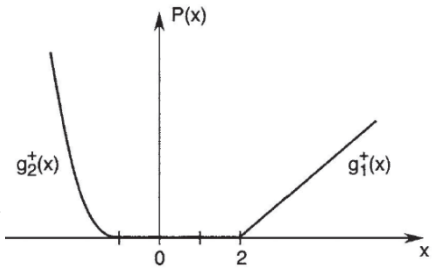
where

$$g_i^+(\mathbf{x}) = \max\{0, g_i(x)\} = \begin{cases} 0, & g_i(\mathbf{x}) \leq 0 \\ g_i(\mathbf{x}), & g_i(\mathbf{x}) > 0. \end{cases}$$

We refer to this penalty function as the absolute value penalty function, because it is equal to $\sum |g_i(\mathbf{x})|$, where the summation is taken over all constraints that are violated at $\mathbf{x}$.

Algorithms for constrained optimization

- Penalty Methods – Choice of penalty function



Example: Let $g_1, g_2 : \mathbb{R} \to \mathbb{R}$ be defined by $g_1(x) = x - 2, g_2(x) = -(x+1)^3$. The feasible set defined by $\{x \in \mathbb{R} : g_1(x) \leq 0, g_2(x) \leq 0\}$ is simply the interval $[-1, 2]$. In this example, we have

$$g_1^+(x) = \max\{0, g_1(x)\} = \begin{cases} 0, & x \leq 2 \\ x - 2, & otherwise, \end{cases}$$

$$g_2^+(x) = \max\{0, g_2(x)\} = \begin{cases} 0, & x \geq -1 \\ -(x+1)^3, & otherwise, \end{cases}$$

and

$$P(x) = g_1^+(x) + g_2^+(x) = \begin{cases} x - 2, & x > 2 \\ 0, & -1 \leq x \leq 2 \\ -(x+1)^3, & x < -1. \end{cases}$$

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

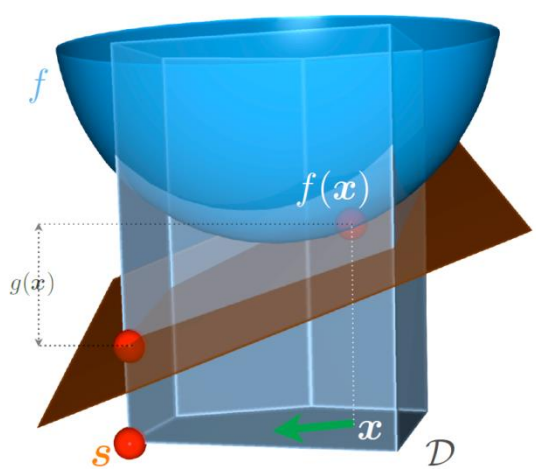
- Frank-Wolfe Algorithm

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

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

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

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



Algorithms for constrained optimization

- Frank-Wolfe Algorithm

The Frank-Wolfe algorithm is advantageous when

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

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

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

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

Algorithms for constrained optimization

- Frank-Wolfe Algorithm - Example

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

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

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

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

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

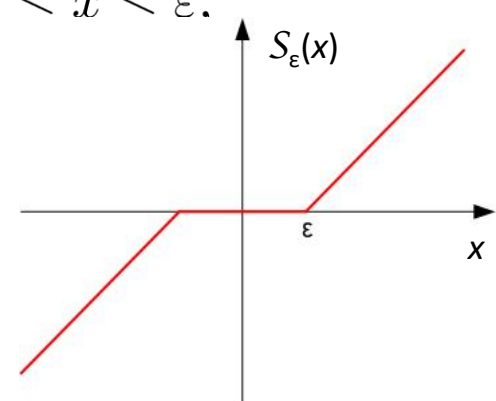
where $\mathbf{\Lambda}$ is the Lagrange multiplier.

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

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.

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

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

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

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

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

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

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)

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

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

- Linearize the quadratic term

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

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

Adjoint Operator

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

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)

- An example (LRR):

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

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

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

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:

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

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

$$\Downarrow$$

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

$$\Downarrow$$

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

$$\Downarrow$$

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

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?

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)

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

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

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

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

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

Parallel!

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:

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

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

Experiment

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

$$\Downarrow$$

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

(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.

$\mathbf{X}$			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 2: 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.

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:

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

- Further linearize g_i:

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

- 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 $L_{\bar{w}} \le \frac{1}{4s}\|\bar{\mathbf{X}}\|_2^2$, where $\bar{\mathbf{X}}=(\bar{\mathbf{x}}_1,\bar{\mathbf{x}}_2,\cdots,\bar{\mathbf{x}}_s)$.

Experiment

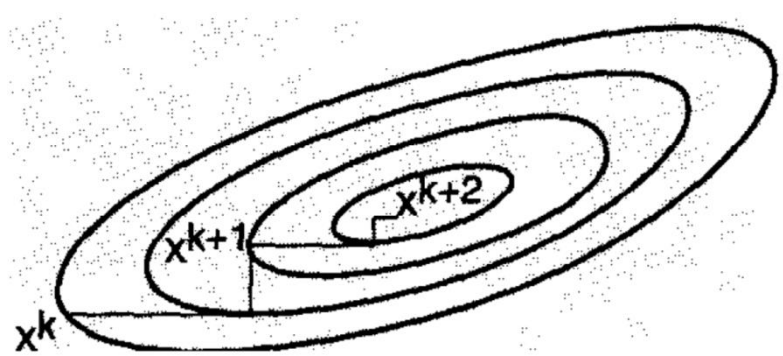
$(s,\ p,\ t,\ q)$	Method	Time	#Iter.	$\frac{\lVert\hat{\bar{\mathbf{w}}}-\bar{\mathbf{w}}^*\rVert}{\lVert\bar{\mathbf{w}}^*\rVert}$	$\frac{\lVert\hat{\mathbf{z}}-\mathbf{z}^*\rVert}{\lVert\mathbf{z}^*\rVert}$
(300, 901, 100, 10)	ADM	294.15	43	0.4800	0.4790
	LADM	229.03	43	0.5331	0.5320
	LADMPS	105.50	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**

Coordinate Descent and Block Coordinate Descent

- Coordinate Descent

The cost is minimized along one coordinate direction at each iteration. The order in which coordinates are chosen may vary in the course of the algorithm. In the case where this order is cyclical, given $\mathbf{x}^k$, the i-th coordinate of $\mathbf{x}^{k+1}$ is determined by

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}\, f(x_1^{k+1}, x_2^{k+1}, \cdots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \cdots, x_n^k). \tag{1}$$



Coordinate Descent and Block Coordinate Descent

- Block Coordinate Descent

$$\begin{aligned} &\min_{\mathbf{x}} \ f(\mathbf{x}), \\ &s.t. \ \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{2}$$

where $\mathcal{X}$ is a Cartesian product of closed convex sets $\mathcal{X}_1, \cdots, \mathcal{X}_m$:

$$\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m. \tag{3}$$

We assume that $\mathcal{X}_i$ is a closed convex subset of $\mathbb{R}^{n_i}$ and $n = n_1 + \cdots + n_m$. The vector $\mathbf{x}$ is partitioned as

$$\mathbf{x} = (\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_m^T)^T,$$

where each $\mathbf{x}_i$ belong to $\mathbb{R}^{n_i}$, so the constraint $\mathbf{x} \in \mathcal{X}$ is equivalent to

$$\mathbf{x}_i \in \mathcal{X}_i, \quad i = 1 \cdots, m.$$

Coordinate Descent and Block Coordinate Descent

- Block Coordinate Descent

Let us assume that for every $\mathbf{x} \in \mathcal{X}$ and every $i = 1, \cdots, m$, the optimization problem:

$$\min_{\mathbf{x}} \; f(\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \boldsymbol{\xi}, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_m),$$
$$s.t. \; \boldsymbol{\xi} \in \mathcal{X}_i,$$

has at least one solution. The following algorithm, known as *block coordinate descent* or *nonlinear Gauss-Seidel* method, generates the next iterate $\mathbf{x}^{k+1} = (\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \cdots, \mathbf{x}_m^{k+1})^T$, given the current iterate $\mathbf{x}^k = (\mathbf{x}_1^k, \mathbf{x}_2^k, \cdots, \mathbf{x}_m^k)^T$, according to the iteration

$$\mathbf{x}_i^{k+1} = \operatorname*{argmin}_{\boldsymbol{\xi} \in \mathcal{X}_i} \; f(\mathbf{x}_1^{k+1}, \cdots, \mathbf{x}_{i-1}^{k+1}, \boldsymbol{\xi}, \mathbf{x}_{i+1}^k, \cdots, \mathbf{x}_m^k), \quad i = 1, \cdots, m. \tag{4}$$

Coordinate Descent and Block Coordinate Descent

- Block Coordinate Descent - Convergence

Proposition 1 (Convergence of Block Coordinate Descent). *Suppose that f is continuously differentiable over the set $\mathcal{X}$ of equation (3). Furthermore, suppose that for each i and $\mathbf{x} \in \mathcal{X}$, the minimum below*

$$\min_{\boldsymbol{\xi} \in \mathcal{X}_i} f(\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \boldsymbol{\xi}, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_m)$$

is uniquely attained. Let $\{\mathbf{x}^k\}$ be the sequence generated by the block coordinate descent method (4). Then every accumulate point of $\{\mathbf{x}^k\}$ is a stationary point.

A counterexample.

Coordinate Descent and Block Coordinate Descent

- Block Coordinate Descent - Examples

Dictionary learning:

$$\min_{\mathbf{D},\mathbf{X}} \frac{1}{2}\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_1, \quad s.t. \quad \|\mathbf{d}_i\|_2 = 1, i = 1, \cdots, K.$$

Low-rank matrix completion:

$$\min_{\mathbf{U},\mathbf{V},\mathbf{A}} \frac{1}{2}\|\mathbf{U}\mathbf{V}^T-\mathbf{A}\|_F^2, \quad s.t. \quad \mathcal{P}_\Omega(\mathbf{A}) = \mathcal{P}_\Omega(\mathbf{D}).$$

Robust Matrix Factorization:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{W}\odot(\mathbf{U}\mathbf{V}^T-\mathbf{M})\|_1 + R_u(\mathbf{U}) + R_v(\mathbf{V}), \quad s.t. \quad \mathbf{U}\in\mathcal{U}, \mathbf{V}\in\mathcal{V}.$$

Truncated Nuclear Norm Minimization:

$$\min_{\mathbf{X}} \|\mathbf{X}\|_r + f(\mathbf{X}), \text{ where } \|\mathbf{X}\|_r = \sum_{i=r+1}^{\min(m,n)} \sigma_i(\mathbf{X}).$$

Coordinate Descent and Block Coordinate Descent

- Block Coordinate Descent - Examples

AdaBoost:

$$\min_{\boldsymbol{\beta},\boldsymbol{\gamma}} \sum_{i=1}^{l} \ell\left(y_i, \sum_{j=1}^{M} \beta_j f_j(\mathbf{x}_i;\boldsymbol{\gamma}_j)\right),$$

$F_k(\mathbf{x}) = \sum_{j=1}^{k} \beta_j f_j(\mathbf{x};\boldsymbol{\gamma}_j)$ is the general additive model;
$(\mathbf{x}_i, y_i), i = 1, \cdots, l,$ are the training samples;
$\ell(y_i, y) = \exp(-y_i y)$ is the loss.

Coordinate Descent and Block Coordinate Descent

- Block Coordinate Descent - Examples

$$(\beta_j, \boldsymbol{\gamma}_j) = \underset{\beta,\boldsymbol{\gamma}}{\operatorname{argmin}} \sum_{i=1}^{l} \ell\left(y_i, F_{j-1}(\mathbf{x}_i) + \beta f_j(\mathbf{x}_i; \boldsymbol{\gamma})\right)$$

$$= \underset{\beta,\boldsymbol{\gamma}}{\operatorname{argmin}} \sum_{i=1}^{l} \exp\left[-y_i(F_{j-1}(\mathbf{x}_i) + \beta f_j(\mathbf{x}_i; \gamma))\right]$$

$$= \underset{\beta,\boldsymbol{\gamma}}{\operatorname{argmin}} \sum_{i=1}^{l} w_i^j \exp\left(-y_i \beta f_j(\mathbf{x}_i; \gamma)\right))$$

$$\boxed{w_i^j = \exp\left(-y_i F_{j-1}(\mathbf{x}_i)\right)}$$

$$\boxed{w_i^{j+1} = w_i^j \exp\left(-y_i \beta_j f_j(\mathbf{x}_i; \boldsymbol{\gamma}_j)\right)}$$

Fixing β, $\boldsymbol{\gamma}_j = \underset{\boldsymbol{\gamma}}{\operatorname{argmin}} \sum_{i=1}^{l} w_i^j \exp\left(-y_i \beta f_j(\mathbf{x}_i; \gamma)\right) = \underset{\boldsymbol{\gamma}_j}{\operatorname{argmin}} \sum_{i=1}^{l} w_i^j I(y_i \neq f_j(\mathbf{x}_i; \boldsymbol{\gamma}))$.

Fixing $\boldsymbol{\gamma}$, $\beta_j = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{l} w_i^j \exp\left(-y_i \beta f_j(\mathbf{x}_i; \gamma)\right) = \frac{1}{2} \log \frac{1 - \mathrm{err}_j}{\mathrm{err}_j}$,

where $\mathrm{err}_j = \left(\sum_{i=1}^{l} w_i^j I(y_i \neq f_j(\mathbf{x}_i; \boldsymbol{\gamma}_j)) \right) / \sum_{i=1}^{l} w_i^j$.