13. Dual based methods

- augmented Lagrangian method
- ADMM
- distributed optimization via ADMM

Original problem

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & h(\boldsymbol{x}) = \mathbf{0} \end{array} \tag{13.1}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^p \to \mathbb{R}$
- Lagrangian: $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T h(\boldsymbol{x})$ where $\boldsymbol{\lambda} \in \mathbb{R}^p$
- problem is equivalent to (for any $\boldsymbol{\lambda}$)

$$\begin{array}{ll} \text{minimize} & L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T h(\boldsymbol{x}) \\ \text{subject to} & h(\boldsymbol{x}) = \mathbf{0} \end{array}$$

- if $\boldsymbol{x}^\star$ is a solution and a regular point, then

$$\nabla_x L(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star) = \mathbf{0}$$

for some $\boldsymbol{\lambda}^\star$

Augmented Lagrangian formulation

$$\text{minimize} \quad L_\rho(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T h(\boldsymbol{x}) + \frac{\rho}{2}\|h(\boldsymbol{x})\|^2 \qquad (13.2)$$

- $L_\rho(\boldsymbol{x}, \boldsymbol{\lambda})$ is the *augmented Lagrangian* (AL) for problem (13.1)
- solution of the original problem is also a solution of the AL formulation
- AL problem can have other solutions that are not solutions of the original problem
- the idea of the augmented Lagrangian method is that for a large, but finite, ρ, the solution of AL method is also a solution of the original problem
- the augmented Lagrangian method minimizes $L_\rho(\boldsymbol{x}, \boldsymbol{\lambda})$ for a sequence of values of $\boldsymbol{\lambda}$ and ρ

Augmented Lagrangian algorithm

Algorithm Augmented Lagrangian method (equality constraint)

given $\boldsymbol{x}^{(0)}$, $\boldsymbol{\lambda}^{(0)}$, ρ_0, and a solution tolerance $\epsilon > 0$

repeat for $k = 1, 2, \ldots$

1. set $\boldsymbol{x}^{(k+1)}$ to be the (approximate) minimizer of

$$\text{minimize} \quad f(\boldsymbol{x}) + (\boldsymbol{\lambda}^{(k)})^T h(\boldsymbol{x}) + \frac{\rho_k}{2} \|h(\boldsymbol{x})\|^2$$

   using any unconstrained optimization method with initial point $\boldsymbol{x}^{(k)}$

2. update $\boldsymbol{\lambda}^{(k)}$:

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho_k h(\boldsymbol{x}^{(k+1)})$$

3. update ρ_k

   if $\|\nabla L(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k+1)})\| \leq \epsilon$ stop and $\boldsymbol{x}^{(k+1)}$ is output

Updating penalty parameter

- constant $\rho_k = \rho$
- heuristic update:

$$\rho_{k+1} = \begin{cases} \rho_k & \text{if} \quad \|h(\boldsymbol{x}^{(k+1)})\| < 0.25\|h(\boldsymbol{x}^{(k)})\| \\ 2\rho_k & \text{if} \quad \|h(\boldsymbol{x}^{(k+1)})\| \geq 0.25\|h(\boldsymbol{x}^{(k)})\| \end{cases}$$

Multiplier update motivation

the solution $\boldsymbol{x}^{(k+1)}$ satisfies $\nabla_x L_\rho(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k)}) = \mathbf{0}$, *i.e.*

$$\nabla f(\boldsymbol{x}^{(k+1)}) + \sum_{i=1}^{p} \nabla h_i(\boldsymbol{x}^{(k+1)})\lambda_i^{(k)} + \rho_k \sum_{i=1}^{p} \nabla h_i(\boldsymbol{x}^{(k+1)}) h_i(\boldsymbol{x}^{(k+1)})$$

$$= \nabla f(\boldsymbol{x}^{(k+1)}) + \sum_{i=1}^{p} \nabla h_i(\boldsymbol{x}^{(k+1)}) \big(\lambda_i^{(k)} + \rho_k h_i(\boldsymbol{x}^{(k+1)})\big) = \mathbf{0}$$

if we let $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho_k h(\boldsymbol{x}^{(k+1)})$, then

$$\nabla f(\boldsymbol{x}^{(k+1)}) + \sum_{i=1}^{p} \nabla h_i(\boldsymbol{x}^{(k+1)})\lambda_i^{(k+1)} = \mathbf{0}$$

- this implies that $\nabla L(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k+1)}) = \mathbf{0}$ and if $\boldsymbol{x}^{(k+1)}$ is feasible, then we have a candidate solution
- note that ρ should be sufficiently large so that the augmented Lagrangian function has a local minimizer; if ρ is too small, then the unconstrained subproblem may not have a solution

Example

consider applying the augmented Lagrangian method to the problem:

$$\begin{array}{ll} \text{minimize} & e^{3x_1} + e^{-4x_2} \\ \text{subject to} & x_1^2 + x_2^2 = 1 \end{array}$$

starting with the initial points $\boldsymbol{x}^{(0)} = (-1, 1)$ and $\lambda^{(0)} = -1$, we set a constant penalty parameter $\rho_k = 10$

the augmented Lagrangian function is expressed as:

$$L_\rho(\boldsymbol{x}, \lambda) = e^{3x_1} + e^{-4x_2} + \lambda\left(x_1^2 + x_2^2 - 1\right) + (\rho/2)\left(x_1^2 + x_2^2 - 1\right)^2$$

for the inner minimization problems at each iteration, we employ Newton's method with a constant stepsize $\alpha = 1$:

$$\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} + \alpha \nabla^2 L_\rho(\hat{\boldsymbol{x}}, \lambda^{(k)})^{-1} \nabla L_\rho(\hat{\boldsymbol{x}}, \lambda^{(k)})$$

the gradient and Hessian are:

$$\nabla L_\rho(\boldsymbol{x}, \lambda) = \begin{bmatrix} 3e^{3x_1} + 2\lambda x_1 + 2\rho x_1(x_1^2 + x_2^2 - 1) \\ -4e^{-4x_2} + 2\lambda x_2 + 2\rho x_2(x_1^2 + x_2^2 - 1) \end{bmatrix}$$

and

$$\nabla^2 L_\rho(\boldsymbol{x}, \lambda) = \begin{bmatrix} 9e^{3x_1} + 2\lambda + 2\rho(x_1^2 + x_2^2 - 1) + 4\rho x_1^2 & 4\rho x_1 x_2 \\ 4\rho x_1 x_2 & 16e^{-4x_2} + 2\lambda + 2\rho(x_1^2 + x_2^2 - 1) + 4\rho x_2^2 \end{bmatrix}$$

this iteration starts from $\hat{\boldsymbol{x}} = \boldsymbol{x}^{(k)}$ and continues until a stopping criteria is met (*e.g.*, $\|\nabla L_\rho(\hat{\boldsymbol{x}}, \lambda^{(k)})\| < 10^{-4}$)

the value $\boldsymbol{x}^{(k+1)}$ is then set to $\hat{\boldsymbol{x}}$ and the Lagrange multiplier is subsequently updated:

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho\big((x_1^{(k+1)})^2 + (x_2^{(k+1)})^2 - 1\big)$$

after executing the augmented Lagrangian method for 100 iterations, the results are approximately $\boldsymbol{x}^\star = (-0.7483, 0.6633)$ and $\lambda^\star = 0.2123$

MATLAB code implementation

```
rho=10;
x=[-1;1];
lam=-1;
%% AL gradient and Hessian
g=@(x,lam)[3*exp(3*x(1))+2*lam*x(1)+2*rho*x(1)*(x(1)^2+x(2)^2-1);
-4*exp(-4*x(2))+2*lam*x(2)+2*rho*x(2)*(x(1)^2+x(2)^2-1)];
hess=@(x,lam)[9*exp(3*x(1))+2*lam+2*rho*(x(1)^2+x(2)^2-1)+4*rho*x(1)^2 4*rho*x(1)*x(2);
4*rho*x(1)*x(2)  16*exp(-4*x(2))+2*lam+2*rho*(x(1)^2+x(2)^2-1)+4*rho*x(2)^2];
%% AL method
for i=1:100
% Newton inner minimization
while (norm(g(x,lam)) >= 1e-4)
d = -hess(x,lam)\g(x,lam);
x = x+d;
end
% Lagrange update
lam=lam+rho*(x(1)^2+x(2)^2-1);
end
```

Outline

- augmented Lagrangian method
- **ADMM**
- distributed optimization via ADMM

ADMM problem form

the alternating direction method of multiplier (ADMM) problem formulation:

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} & A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c} \end{array}$$

- variables $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{z} \in \mathbb{R}^m$
- $A \in \mathbb{R}^{p\times n}, B \in \mathbb{R}^{p\times m}$, and $\boldsymbol{c} \in \mathbb{R}^p$
- ADMM is a modification of the AL method that is more suitable for large-scale seperable optimization problems (more on this later)

Augmented Lagrangian

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^T(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}) + (\rho/2)\|A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}\|^2$$

ADMM update

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, L_\rho \left(\boldsymbol{x}, \boldsymbol{z}^{(k)}, \boldsymbol{\lambda}^{(k)}\right)$$
$$\boldsymbol{z}^{(k+1)} = \underset{\boldsymbol{z}}{\operatorname{argmin}}\, L_\rho \left(\boldsymbol{x}^{(k+1)}, \boldsymbol{z}, \boldsymbol{\lambda}^{(k)}\right)$$
$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho \left(A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z}^{(k+1)} - \boldsymbol{c}\right)$$

- $\rho > 0$ is the ADMM penalty parameter
- x and z are updated in an alternating or sequential fashion
- this is different from AL method where x and z are minimized jointly

$$(\boldsymbol{x}, \boldsymbol{z}) = \underset{\boldsymbol{x},\boldsymbol{z}}{\operatorname{argmin}} \quad L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}^{(k)})$$

- separating the minimization over x and z allows for decomposition large problems into smaller ones when f or g are separable

ADMM scaled form

ADMM can be written in a more convenient form, by defining the residual $\boldsymbol{r} = A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}$ and $\boldsymbol{u} = (1/\rho)\boldsymbol{\lambda}$, we have

$$\begin{aligned}\boldsymbol{\lambda}^T\boldsymbol{r} + (\rho/2)\|\boldsymbol{r}\|^2 &= (\rho/2)\|\boldsymbol{r} + (1/\rho)\boldsymbol{\lambda}\|^2 - (1/2\rho)\|\boldsymbol{\lambda}\|^2 \\ &= (\rho/2)\|\boldsymbol{r} + \boldsymbol{u}\|^2 - (\rho/2)\|\boldsymbol{u}\|^2\end{aligned}$$

ADMM scaled form

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(f(\boldsymbol{x}) + (\rho/2) \left\| A\boldsymbol{x} + B\boldsymbol{z}^{(k)} - \boldsymbol{c} + \boldsymbol{u}^{(k)} \right\|^2 \right) \\ \boldsymbol{z}^{(k+1)} &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(g(\boldsymbol{z}) + (\rho/2) \left\| A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}^{(k)} \right\|^2 \right) \\ \boldsymbol{u}^{(k+1)} &= \boldsymbol{u}^{(k)} + A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z}^{(k+1)} - \boldsymbol{c}\end{aligned}$$

Example: quadratic programs

$$
\begin{array}{ll}
\text{minimize} & (1/2)\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{r}^T \boldsymbol{x} \\
\text{subject to} & A\boldsymbol{x} = \boldsymbol{b} \\
& \boldsymbol{x} \geq \mathbf{0}
\end{array}
$$

- P is positive semidefinite (reduces to an LP when $P = 0$)
- we can express this problem in the ADMM form:

$$
\begin{array}{ll}
\text{minimize} & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\
\text{subject to} & \boldsymbol{x} - \boldsymbol{z} = \mathbf{0},
\end{array}
$$

where

$$
f(\boldsymbol{x}) = (1/2)\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{r}^T \boldsymbol{x}, \quad \operatorname{dom} f = \{\boldsymbol{x} \mid A\boldsymbol{x} = \boldsymbol{b}\}
$$

is the original objective with restricted domain

- g is the indicator function of the nonnegative orthant $\mathbb{R}^n_+$

the scaled form of ADMM consists of the iterations

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \operatorname*{argmin}_{\boldsymbol{x}} \left(f(\boldsymbol{x}) + (\rho/2) \left\| \boldsymbol{x} - \boldsymbol{z}^{(k)} + \boldsymbol{u}^{(k)} \right\|^2 \right) \\
\boldsymbol{z}^{(k+1)} &= \left(\boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)} \right)_+ \\
\boldsymbol{u}^{(k+1)} &= \boldsymbol{u}^{(k)} + \boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)}
\end{aligned}$$

the $\boldsymbol{x}$-update is an equality-constrained least squares problem with optimality conditions

$$\begin{bmatrix} Q + \rho I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{(k+1)} \\ \boldsymbol{\nu} \end{bmatrix} + \begin{bmatrix} \boldsymbol{r} - \rho \left(\boldsymbol{z}^{(k)} - \boldsymbol{u}^{(k)} \right) \\ -\boldsymbol{b} \end{bmatrix} = \mathbf{0}$$

Norm-one regularized least squares

the **lasso** problem is the ℓ_1 regularized least squares

$$\text{minimize} \quad (1/2)\|A\boldsymbol{x} - \boldsymbol{b}\|^2 + \eta\|\boldsymbol{x}\|_1$$

- $\eta > 0$ is a scalar regularization parameter
- in ADMM form, the lasso problem can be written as

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{x} - \boldsymbol{z} = \boldsymbol{0} \end{array}$$

  where $f(\boldsymbol{x}) = (1/2)\|A\boldsymbol{x} - \boldsymbol{b}\|^2$ and $g(\boldsymbol{z}) = \eta\|\boldsymbol{z}\|_1$

the ADMM iteration is

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \left(A^T A + \rho I\right)^{-1} \left(A^T \boldsymbol{b} + \rho \left(\boldsymbol{z}^{(k)} - \boldsymbol{u}^{(k)}\right)\right) \\
\boldsymbol{z}^{(k+1)} &= S_{\eta/\rho} \left(\boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)}\right) \\
\boldsymbol{u}^{(k+1)} &= \boldsymbol{u}^{(k)} + \boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)}
\end{aligned}$$

where the soft thresholding operator S is defined element-wise as

$$\begin{aligned}
S_\kappa(a) &= \begin{cases} a - \kappa & a > \kappa \\ 0 & |a| \leq \kappa \\ a + \kappa & a < -\kappa \end{cases} \\
&= (a - \kappa)_+ - (-a - \kappa)_+
\end{aligned}$$

Outline

- augmented Lagrangian method
- ADMM
- **distributed optimization via ADMM**

Consensus problem

$$\text{minimize} \quad f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}),$$

- variable $\boldsymbol{x} \in \mathbb{R}^n$
- each $f_i : \mathbb{R}^n \to \mathbb{R}$ represents the ith component of the objective function
- the goal is to solve this problem such that each function f_i can be independently addressed by a distinct processing unit

Example

many classification or regression problems can be formulated as:

$$\text{minimize} \quad \sum_{j=1}^{m} l(\boldsymbol{x}; \xi_j),$$

- $l(\boldsymbol{x}; \xi_j)$ represent the loss function for data ξ_j
- for large m, storing the data on a single machine may not be feasible
- the problem can be solved by distributing the data across multiple workers,

$$f_i(\boldsymbol{x}) = \sum_{j \in \mathcal{J}_i} l(\boldsymbol{x}; \xi_j),$$

where $\mathcal{J}_i$ is the set of training data indices at worker i

Equivalent formulation

to employ ADMM, we introduce local variables $\boldsymbol{x}_i \in \mathbb{R}^n$ handled by each processing unit along with a global variable $\boldsymbol{z}$ (handled by some processing unit):

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^{N} f_i\left(\boldsymbol{x}_i\right) \\ \text{subject to} & \boldsymbol{x}_i - \boldsymbol{z} = \mathbf{0}, \quad i = 1, \ldots, N\end{array}$$

- the constraints ensure that all local variables are equal
- the consensus approach is an efficient strategy to transform additive objectives $\sum_{i=1}^{N} f_i(\boldsymbol{x})$, which are common but non-separable due to the shared variable, into separable objectives $\sum_{i=1}^{N} f_i\left(\boldsymbol{x}_i\right)$
- thus the consensus problem can address problems where objectives and constraints span multiple processors
- each processor solely manages its unique objective and constraint term

ADMM updates

the augmented Lagrangian, given by:

$$L_\rho(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N,\boldsymbol{z},\boldsymbol{\lambda}) = \sum_{i=1}^{N}\left(f_i\left(\boldsymbol{x}_i\right) + \left(\boldsymbol{\lambda}_i\right)^T\left(\boldsymbol{x}_i - \boldsymbol{z}\right) + \frac{\rho}{2}\left\|\boldsymbol{x}_i - \boldsymbol{z}\right\|^2\right)$$

the resulting ADMM algorithm takes the form:

$$\begin{aligned}
\boldsymbol{x}_i^{(k+1)} &= \underset{\boldsymbol{x}_i}{\operatorname{argmin}}\left(f_i\left(\boldsymbol{x}_i\right) + \boldsymbol{\lambda}_i^{(k)T}\left(\boldsymbol{x}_i - \boldsymbol{z}^{(k)}\right) + \frac{\rho}{2}\left\|\boldsymbol{x}_i - \boldsymbol{z}^{(k)}\right\|^2\right) \\
\boldsymbol{z}^{(k+1)} &= \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}_i^{(k+1)} + \frac{1}{\rho}\boldsymbol{\lambda}_i^{(k)}\right) \\
\boldsymbol{\lambda}_i^{(k+1)} &= \boldsymbol{\lambda}_i^{(k)} + \rho\left(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}^{(k+1)}\right)
\end{aligned}$$

- each i undergoes the first and last steps independently
- the processing unit responsible for the global variable z is commonly referred to as the *fusion center* or *central server*

Equivalent simpler update

using an overline to denote the average (across $i = 1, \ldots, N$) of a vector, we can express the $\boldsymbol{z}$-update as:

$$\boldsymbol{z}^{(k+1)} = \bar{\boldsymbol{x}}^{(k+1)} + \frac{1}{\rho}\bar{\boldsymbol{\lambda}}^{(k)}$$

by taking the average of the $\boldsymbol{\lambda}$-update, we get:

$$\bar{\boldsymbol{\lambda}}^{(k+1)} = \bar{\boldsymbol{\lambda}}^{(k)} + \rho\left(\bar{\boldsymbol{x}}^{(k+1)} - \boldsymbol{z}^{(k+1)}\right)$$

upon substituting the first equation into the subsequent one, we obtain that $\bar{\boldsymbol{\lambda}}^{(k+1)} = \mathbf{0}$ for all k

hence $\boldsymbol{z}^{(k)} = \bar{\boldsymbol{x}}^{(k)}$ and ADMM can be reformulated as:

$$\boldsymbol{x}_i^{(k+1)} = \underset{x_i}{\operatorname{argmin}}\left(f_i\left(\boldsymbol{x}_i\right) + \boldsymbol{\lambda}_i^{(k)T}\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^{(k)}\right) + \frac{\rho}{2}\left\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}^{(k)}\right\|^2\right)$$

$$\boldsymbol{\lambda}_i^{(k+1)} = \boldsymbol{\lambda}_i^{(k)} + \rho\left(\boldsymbol{x}_i^{(k+1)} - \bar{\boldsymbol{x}}^{(k+1)}\right)$$

Regularized consensus problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i\left(\boldsymbol{x}_i\right) + g(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{x}_i - \boldsymbol{z} = \mathbf{0}, \quad i = 1, \ldots, N, \end{array}$$

where the objective term g, symbolizes a constraint or regularization (*e.g.*, $g(\boldsymbol{z}) = \|\boldsymbol{z}\|_1$), managed by the central server

for this case, the ADMM method is:

$$\boldsymbol{x}_i^{(k+1)} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \left(f_i\left(\boldsymbol{x}_i\right) + \boldsymbol{\lambda}_i^{(k)T}\left(\boldsymbol{x}_i - \boldsymbol{z}^{(k)}\right) + \frac{\rho}{2}\|\boldsymbol{x}_i - \boldsymbol{z}^{(k)}\|^2 \right)$$

$$\boldsymbol{z}^{(k+1)} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(g(\boldsymbol{z}) + \sum_{i=1}^{N} \left(-\boldsymbol{\lambda}_i^{(k)T}\boldsymbol{z} + \frac{\rho}{2}\|\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}\|^2 \right) \right)$$

$$\boldsymbol{\lambda}_i^{(k+1)} = \boldsymbol{\lambda}_i^{(k)} + \rho\left(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}^{(k+1)}\right)$$

collecting linear and quadratic terms, the z-update can be expressed as:

$$\boldsymbol{z}^{(k+1)} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(g(\boldsymbol{z}) + \frac{N\rho}{2} \|\boldsymbol{z} - \bar{\boldsymbol{x}}^{(k+1)} - \frac{1}{\rho} \bar{\boldsymbol{\lambda}}^{(k)} \|^2 \right)$$

when g is nonzero, we don't typically get that $\bar{\boldsymbol{\lambda}}^{(k)} = \mathbf{0}$, hence $\boldsymbol{\lambda}_i$ terms cannot be eliminated as in the non-regularized case

using the above update form for z, ADMM is:

$$\begin{aligned}
\boldsymbol{x}_i^{(k+1)} &= \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \left(f_i\left(\boldsymbol{x}_i\right) + \boldsymbol{\lambda}_i^{(k)T} \left(\boldsymbol{x}_i - \boldsymbol{z}^{(k)} \right) + \frac{\rho}{2} \|\boldsymbol{x}_i - \boldsymbol{z}^{(k)}\|^2 \right) \\
\boldsymbol{z}^{(k+1)} &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(g(\boldsymbol{z}) + \frac{N\rho}{2} \|\boldsymbol{z} - \bar{\boldsymbol{x}}^{(k+1)} - \frac{1}{\rho} \bar{\boldsymbol{\lambda}}^{(k)} \|^2 \right) \\
\boldsymbol{\lambda}_i^{(k+1)} &= \boldsymbol{\lambda}_i^{(k)} + \rho \left(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}^{(k+1)} \right)
\end{aligned}$$

Examples

- for $g(\boldsymbol{z}) = \eta\|\boldsymbol{z}\|_1$ with $\eta > 0$, the $\boldsymbol{z}$-update translates into a soft threshold operation:

$$\boldsymbol{z}^{(k+1)} = S_{\eta/N\rho}\left(\bar{\boldsymbol{x}}^{(k+1)} - \frac{1}{\rho}\bar{\boldsymbol{\lambda}}^{(k)}\right)$$

- considering g as the indicator function of $\mathbb{R}^n_+$, then

$$\boldsymbol{z}^{(k+1)} = \left(\bar{\boldsymbol{x}}^{(k+1)} - \frac{1}{\rho}\bar{\boldsymbol{\lambda}}^{(k)}\right)_+$$

for this problem, the scaled variant of ADMM, exhibited below, is often more streamlined and manageable compared to its unscaled counterpart:

$$\begin{aligned}
\boldsymbol{x}_i^{(k+1)} &= \underset{\boldsymbol{x}_i}{\operatorname{argmin}}\left(f_i\left(\boldsymbol{x}_i\right) + \frac{\rho}{2}\|\boldsymbol{x}_i - \boldsymbol{z}^{(k)} + \boldsymbol{u}_i^{(k)}\|^2\right) \\
\boldsymbol{z}^{(k+1)} &= \underset{\boldsymbol{z}}{\operatorname{argmin}}\left(g(\boldsymbol{z}) + \frac{N\rho}{2}\|\boldsymbol{z} - \bar{\boldsymbol{x}}^{(k+1)} - \bar{\boldsymbol{u}}^{(k)}\|^2\right) \\
\boldsymbol{u}_i^{(k+1)} &= \boldsymbol{u}_i^{(k)} + \boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}^{(k+1)}
\end{aligned}$$

References and further readings

- S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers,* Foundations and Trends in Machine learning, 2011.

- I. Griva and S. G. Nash and A. Sofer. *Linear and Nonlinear Optimization,* SIAM, 2009.

- E. KP. Chong and S. H. Zak. *An Introduction to Optimization,* John Wiley & Sons, 2013.