Alternating direction method of multipliers (ADMM)

ADMM extends the method of multipliers in such a way that we get back some of the decomposability (i.e., ability to parallelize) of standard dual ascent algorithms. It also gives us a flexible framework for incorporating many types of convex constraints, though we will again focus on linear equality constraints to start. See [BPC+10] for a more complete discussion.

ADMM **splits** the optimization variables into two parts, $\boldsymbol{x}$ and $\boldsymbol{z}$, and solves programs of the form

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \;\; f(\boldsymbol{x}) + g(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}.$$

The basic idea is to rotate through 3 steps:

1. Minimize the (augmented) Lagrangian over $\boldsymbol{x}$ with $\boldsymbol{z}$ and the Lagrange multipliers $\boldsymbol{\nu}$ fixed.
2. Minimize the (augmented) Lagrangian over $\boldsymbol{z}$ with $\boldsymbol{x}$ and $\boldsymbol{\nu}$ fixed.
3. Update the Lagrange multipliers using gradient ascent as before.

If the splitting is done in a careful manner, it can happen that each of the subproblems above can be easily computed. We can also handle general convex constraints (more on this later).

To make the three steps above more explicit: the augmented Lagrangian is

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\|_2^2,$$

and the general ADMM iteration is

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}_k, \boldsymbol{\nu}_k) \\
\boldsymbol{z}_{k+1} &= \arg\min_{\boldsymbol{z}} \ \mathcal{L}_\rho(\boldsymbol{x}_{k+1}, \boldsymbol{z}, \boldsymbol{\nu}_k) \\
\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + \rho(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{c}).
\end{aligned}$$

The only real difference between ADMM and MoM is the we are splitting the primal minimization into two parts instead of optimizing over $(\boldsymbol{x}, \boldsymbol{z})$ jointly.

Scaled form.

We can write the ADMM iterations in a more convenient form by substituting

$$\boldsymbol{\mu} = \frac{1}{\rho}\boldsymbol{\nu}.$$

By "completing the square" we have that

$$\boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c})+\frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c}\|_2^2 = \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c}+\boldsymbol{\mu}\|_2^2-\frac{\rho}{2}\|\boldsymbol{\mu}\|_2^2,$$

and so we can write:

ADMM:

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left(f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}_k - \boldsymbol{c} + \boldsymbol{\mu}_k\|_2^2 \right) \\
\boldsymbol{z}_{k+1} &= \arg\min_{\boldsymbol{z}} \left(g(\boldsymbol{z}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{\mu}_k\|_2^2 \right) \\
\boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{c}
\end{aligned}$$

Example: The LASSO

Recall the LASSO:

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\tau\|\boldsymbol{x}\|_1.$$

Taking

$$f(\boldsymbol{x})=\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 \quad \text{and} \quad g(\boldsymbol{z})=\tau\|\boldsymbol{z}\|_1,$$

we can rewrite this in ADMM form as

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \ f(\boldsymbol{x})+g(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{x}-\boldsymbol{z}=\boldsymbol{0}.$$

The $\boldsymbol{x}$ update is

$$\boldsymbol{x}_{k+1}=\arg\min_{\boldsymbol{x}}\left(\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\frac{\rho}{2}\|\boldsymbol{x}-\boldsymbol{z}_k+\boldsymbol{\mu}_k\|_2^2\right).$$

With both $\boldsymbol{z}_k$ and $\boldsymbol{\mu}_k$ fixed, this is a regularized least-squares problem and is equivalent to:

$$\min_{\boldsymbol{x}} \ \left\|\begin{bmatrix}\boldsymbol{A}\\ \sqrt{\rho}\,\mathbf{I}\end{bmatrix}\boldsymbol{x}-\begin{bmatrix}\boldsymbol{b}\\ \sqrt{\rho}(\boldsymbol{z}_k-\boldsymbol{\mu}_k)\end{bmatrix}\right\|_2^2.$$

This problem has a closed-form solution:

$$\begin{aligned}\boldsymbol{x}_{k+1}&=\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}+\rho\mathbf{I}\right)^{-1}\begin{bmatrix}\boldsymbol{A}^{\mathrm{T}} & \sqrt{\rho}\,\mathbf{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{b}\\ \sqrt{\rho}(\boldsymbol{z}_k-\boldsymbol{\mu}_k)\end{bmatrix}\\ &=\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}+\rho\mathbf{I}\right)^{-1}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}+\rho(\boldsymbol{z}_k-\boldsymbol{\mu}_k)\right)\end{aligned}$$

The $\boldsymbol{z}$ update problem is:

$$\underset{\boldsymbol{z}}{\text{minimize}} \ \tau\|\boldsymbol{z}\|_1 + \frac{\rho}{2}\|\boldsymbol{z} - \boldsymbol{x}_{k+1} - \boldsymbol{\mu}_k\|_2^2.$$

You may recognize this: it is the proximal operator for the ℓ_1-norm, which as we have seen before has closed form solution:

$$\boldsymbol{z}_{k+1} = T_{\tau/\rho}(\boldsymbol{x}_{k+1} + \boldsymbol{\mu}_k),$$

where $T_\lambda(\cdot)$ is the term-by-tern soft-thresholding operator,

$$(T_\lambda(\boldsymbol{v}))[n] = \begin{cases} v[n] - \lambda, & v[n] > \lambda, \\ 0, & |v[n]| \leq \lambda, \\ v[n] + \lambda, & v[n] < -\lambda. \end{cases}$$

To summarize:

ADMM iterations for the LASSO

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \rho\mathbf{I}\right)^{-1}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b} + \rho(\boldsymbol{z}_k - \boldsymbol{\mu}_k)), \\ \boldsymbol{z}_{k+1} &= T_{\tau/\rho}(\boldsymbol{x}_{k+1} + \boldsymbol{\mu}_k), \\ \boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1}. \end{aligned}$$

Convergence properties

We will state one convergence result. If the following two conditions hold:

1. f and g are closed, proper, and convex (i.e., their epigraphs are nonempty closed convex sets),
2. strong duality holds,

then

- $\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{z}_k - \boldsymbol{c} \to \boldsymbol{0}$ as $k \to \infty$. That is, the primal iterates are asymptotically feasible.
- $f(\boldsymbol{x}_k) + g(\boldsymbol{z}_k) \to f(\boldsymbol{x}^\star) + g(\boldsymbol{x}^\star)$ as $k \to \infty$. That is, the value of the objective function approaches the optimal value asymptotically.
- $\boldsymbol{\nu}_k \to \boldsymbol{\nu}^\star$ as $k \to \infty$, where $\boldsymbol{\nu}^\star$ is a dual optimal point.

Under additional assumptions, we can also have convergence to a primal optimal point, $(\boldsymbol{x}_k, \boldsymbol{z}_k) \to (\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ as $k \to \infty$.

See [BPC+10, Section 3.2] for further discussion and references.

Convex constraints

Using a technique that we have seen before, we can write the general program

$$\underset{\boldsymbol{x}\in\mathcal{C}}{\text{minimize}} \;\; f(\boldsymbol{x}),$$

where $\mathcal{C}$ is a closed convex set, in ADMM form as

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \;\; f(\boldsymbol{x}) + g(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{x} - \boldsymbol{z} = \boldsymbol{0},$$

where $g(\boldsymbol{z})$ is the indicator function for $\mathcal{C}$:

$$g(\boldsymbol{z}) = \begin{cases} 0, & \boldsymbol{z} \in \mathcal{C}, \\ \infty, & \boldsymbol{z} \notin \mathcal{C}. \end{cases}$$

Note that in this case, the $\boldsymbol{z}$ update is a closest-point-to-a-convex-set problem. For fixed $\boldsymbol{v} \in \mathbb{R}^N$:

$$\arg\min_{\boldsymbol{z}} \;\; g(\boldsymbol{z}) + \frac{\rho}{2}\|\boldsymbol{z} - \boldsymbol{v}\|_2^2 = \arg\min_{\boldsymbol{z}\in\mathcal{C}} \;\; \|\boldsymbol{z} - \boldsymbol{v}\|_2 = \mathcal{P}_{\mathcal{C}}(\boldsymbol{v}).$$

ADMM iteration for general convex constraints:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left(f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{z}_k + \boldsymbol{\mu}_k\|_2^2 \right), \\ \boldsymbol{z}_{k+1} &= \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{x}_{k+1} + \boldsymbol{\mu}_k\right), \\ \boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1}. \end{aligned}$$

Of course, this algorithm is most attractive when we have a fast method for computing $\mathcal{P}_{\mathcal{C}}(\cdot)$.

Example: Basis Pursuit

As we have seen before, a good proxy for finding the sparsest solution to an underdetermined system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is to solve

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \|\boldsymbol{x}\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$

To put this in ADMM form, we are solving

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \ f(\boldsymbol{x}) + g(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{x} - \boldsymbol{z} = \boldsymbol{0},$$

with

$$f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1, \quad \text{and} \quad g(\boldsymbol{z}) = \begin{cases} 0, & \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}, \\ \infty, & \text{otherwise.} \end{cases}$$

The projection onto $\mathcal{C} = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ can be given in closed form using the pseudo-inverse c of $\boldsymbol{A}$ as

$$\begin{aligned} \mathcal{P}_{\mathcal{C}}(\boldsymbol{v}) &= \boldsymbol{A}^{\dagger}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{v}) + \boldsymbol{v} \\ &= (\mathbf{I} - \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{A})\boldsymbol{v} + \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{b}, \end{aligned}$$

where the last equality comes from $\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}$ when $\boldsymbol{A}$ has full row rank.

The updates in this case are

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left(\|\boldsymbol{x}\|_1 + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{z}_k + \boldsymbol{\mu}_k\|_2^2 \right) \\ &= T_{1/\rho}(\boldsymbol{z}_k - \boldsymbol{\mu}_k) \\ \boldsymbol{z}_{k+1} &= (\mathbf{I} - \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{A})(\boldsymbol{x}_{k+1} + \boldsymbol{\mu}_k) + \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{b} \\ \boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1}. \end{aligned}$$

Example: Linear programming

Consider the general linear program

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \ \ \boldsymbol{x} \geq \boldsymbol{0},$$

where $\boldsymbol{A}$ is an $M \times N$ matrix with full row rank.[1] We can put this in ADMM form by first eliminating the equality constraints, then introducing the indicator function for the non-negativity constraint.

Let $\boldsymbol{Q}$ be an $N \times (N - M)$ matrix whose columns span Null($\boldsymbol{A}$), and let $\boldsymbol{x}_0$ be any point such that $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{b}$. Then we can re-write the LP as

$$\underset{\boldsymbol{w}}{\text{minimize}} \ \boldsymbol{c}^{\mathrm{T}}(\boldsymbol{x}_0 + \boldsymbol{Q}\boldsymbol{w}) \quad \text{subject to} \quad \boldsymbol{x}_0 + \boldsymbol{Q}\boldsymbol{w} \geq \boldsymbol{0},$$

which we can write in ADMM form as

$$\underset{\boldsymbol{w}}{\text{minimize}} \ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_0 + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{w} + g(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{Q}\boldsymbol{w} - \boldsymbol{z} = -\boldsymbol{x}_0,$$

where

$$g(\boldsymbol{z}) = \begin{cases} 0, & \boldsymbol{z} \geq \boldsymbol{0}, \\ \infty, & \text{otherwise.} \end{cases}$$

(We can drop the $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_0$ from the objective since it does not depend on either of the optimization variables.)

Notice that when $\boldsymbol{Q}$ has full column rank, the program

$$\underset{\boldsymbol{w}}{\text{minimize}} \ \boldsymbol{v}^{\mathrm{T}}\boldsymbol{w} + \frac{1}{2}\|\boldsymbol{Q}\boldsymbol{w} - \boldsymbol{y}\|_2^2,$$

[1] The full row rank assumption is not at all essential; I am just making it to keep things streamlined.

has the closed-form solution

$$\boldsymbol{w}^\star = (\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q})^{-1}(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{v}).$$

Also, the projection onto the non-negative orthant $\mathcal{C} = \{\boldsymbol{x} : \boldsymbol{x} \geq \boldsymbol{0}\}$ is

$$\mathcal{P}_{\mathcal{C}}(\boldsymbol{v}) = (\boldsymbol{v})_+, \quad \text{or} \quad (\mathcal{P}_{\mathcal{C}}(\boldsymbol{v}))[n] = \begin{cases} v[n], & v[n] \geq 0, \\ 0, & v[n] < 0. \end{cases}$$

For the general linear program, then, the ADMM iterations are

$$\begin{aligned}
\boldsymbol{w}_{k+1} &= \arg\min_{\boldsymbol{w}} \left(\frac{1}{\rho}\boldsymbol{c}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{w} + \frac{1}{2}\|\boldsymbol{Q}\boldsymbol{w} - \boldsymbol{z}_k + \boldsymbol{x}_0 + \boldsymbol{\mu}_k\|_2^2 \right) \\
&= (\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q})^{-1} \left[\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{z}_k - \boldsymbol{x}_0 - \boldsymbol{\mu}_k) - \frac{1}{\rho}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{c} \right], \\
\boldsymbol{z}_{k+1} &= P_{\mathcal{C}}(\boldsymbol{Q}\boldsymbol{w}_{k+1} + \boldsymbol{x}_0 + \boldsymbol{\mu}_k) \\
&= (\boldsymbol{Q}\boldsymbol{w}_{k+1} + \boldsymbol{x}_0 + \boldsymbol{\mu}_k)_+ \\
\boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \boldsymbol{Q}\boldsymbol{w}_{k+1} - \boldsymbol{z}_{k+1} + \boldsymbol{x}_0.
\end{aligned}$$

Notice that especially when the columns of $\boldsymbol{Q}$ are orthogonal, $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}$, all of these steps are very simple.

References

[BPC+10] 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*, 3(1):1–122, 2010.