Alternating direction primal dual methods

We will now focus on a class of algorithms that work by fixing the dual variables and updating the primal variables $\boldsymbol{x}$, then fixing the primals and updating the dual variables $\boldsymbol{\lambda}, \boldsymbol{\nu}$. An excellent source for this material is [BPC+10]. In fact, what follows here is basically a summary of the first 12 pages of that paper.

We have seen that when we have strong duality (which we will assume throughout), the optimal value of the primal program is equal to the optimal value of the dual program. That is, if $\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star$ are primal/dual optimal points,

$$\begin{aligned} f(\boldsymbol{x}^\star) &= d(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \\ &= \inf_{\boldsymbol{x} \in \mathbb{R}^N} \ \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star), \end{aligned}$$

where $\mathcal{L}$ is the Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \sum_{m=1}^{M} \lambda_m g_m(\boldsymbol{x}) + \langle \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}, \boldsymbol{\nu} \rangle.$$

If $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ has only one minimizer,[1] then we can recover the primal optimal solution $\boldsymbol{x}^\star$ from the dual-optimal solution $\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star$ by solving the *unconstrained* program

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x} \in \mathbb{R}^N} \ \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star).$$

"Alternating" methods search for a saddle point of the Lagrangian by fixing the dual variables $\boldsymbol{\lambda}_k, \boldsymbol{\nu}_k$, minimizing $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_k, \boldsymbol{\nu}_k)$ with respect to $\boldsymbol{x}$, then updating the Lagrange multipliers.

[1] Which is the case when f is strictly convex, and in many other situations.

To start, we will base our discussion on **equality constrained** problems. Incorporating inequality constraints will be natural after we have developed things a bit.

Dual ascent

We want to solve

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \;\; f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$

We will assume that the domain of f is all of $\mathbb{R}^N$; again, things are easily modified if this is any open set. The Lagrangian is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \langle \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}, \boldsymbol{\nu} \rangle,$$

and the dual function is

$$\begin{aligned} d(\boldsymbol{\nu}) &= \inf_{\boldsymbol{x}} \;\; \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}) \\ &= \inf_{\boldsymbol{x}} \;\; f(\boldsymbol{x}) + \langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{\nu} \rangle - \langle \boldsymbol{b}, \boldsymbol{\nu} \rangle \\ &= -\sup_{\boldsymbol{x}} \Big(\langle \boldsymbol{x}, -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu} \rangle - f(\boldsymbol{x}\Big) - \langle \boldsymbol{b}, \boldsymbol{\nu} \rangle \\ &= -f^*(-\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}) - \langle \boldsymbol{b}, \boldsymbol{\nu} \rangle \end{aligned}$$

and the dual problem is

$$\underset{\boldsymbol{\nu}\in\mathbb{R}^N}{\text{maximize}} \;\; d(\boldsymbol{\nu}).$$

Consider for a moment the problem of maximizing the dual. A reasonable thing to do would be some kind of gradient ascent:[2]

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + \alpha_k \nabla d(\boldsymbol{\nu}_k),$$

[2] "Ascent" instead of "descent" because g is concave instead of convex.

where α_k is some appropriate step size. The gradient of d (with respect to $\boldsymbol{\nu}$) at a point $\boldsymbol{\nu}_0$ is

$$\nabla d(\boldsymbol{\nu}_0) = \nabla \inf_{\boldsymbol{x}} \ \left(f(\boldsymbol{x}) + \langle \boldsymbol{\nu}_0, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \rangle \right),$$

and so if $\boldsymbol{x}^+ = \arg\min_{\boldsymbol{x}} \ (f(\boldsymbol{x}) + \langle \boldsymbol{\nu}_0, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \rangle)$, then

$$\begin{aligned} \nabla d(\boldsymbol{\nu}_0) &= \nabla_{\boldsymbol{\nu}} \left(f(\boldsymbol{x}^+) + \langle \boldsymbol{\nu}_0, \boldsymbol{A}\boldsymbol{x}^+ - \boldsymbol{b} \rangle \right) \\ &= \boldsymbol{A}\boldsymbol{x}^+ - \boldsymbol{b}. \end{aligned}$$

This leads naturally to:

The **dual ascent** algorithm consists of the iteration

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \ \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}_k) \\ \boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + \alpha_k(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}) \end{aligned}$$

that is repeated until some convergence criteria is met.

This algorithm "works" under certain assumptions on f (that translate to different assumptions on the dual d). In particular, we need $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu})$ to be bounded for every $\boldsymbol{\nu}$, otherwise the primal update $\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \ \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}_k)$ can fail.

That the Lagrangian is bounded below for every choice of $\boldsymbol{\nu}$ is far from a given. For example, a program of the form

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \langle \boldsymbol{x}, \boldsymbol{c} \rangle \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

will have Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}) &= \langle \boldsymbol{x}, \boldsymbol{c} \rangle + \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \\ &= (\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu})^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{b}, \end{aligned}$$

which is unbounded below since the first term is linear in $\boldsymbol{x}$.

Of course, this algorithm is nicest when we can solve the unconstrained primal update problem efficiently.

Dual decomposition

Dual ascent is simple, and it is pretty much as old an idea as convex optimization itself. But it has a key feature that makes it very attractive (at least as a starting point) for modern computing platforms.

If the objective functional f is **separable**, we can also separate (i.e., parallelize) the primal update. Suppose we can write $f(\boldsymbol{x})$ as a sum

$$f(\boldsymbol{x}) = \sum_{i=1}^{B} f_i(\boldsymbol{x}^{(i)}),$$

where the $\boldsymbol{x}^{(i)} \in \mathbb{R}^{N_i}$ ($\sum_i N_i = N$) are a partition of $\boldsymbol{x}$. We can partition the columns of $\boldsymbol{A}$ in the same way,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^{(1)} & \boldsymbol{A}^{(2)} & \cdots & \boldsymbol{A}^{(B)} \end{bmatrix}$$

so that

$$\boldsymbol{A}\boldsymbol{x} = \sum_{i=1}^{B} \boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)}.$$

Notice that even with this assumption, the primal optimization program itself is not separable, as the constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ tie all of the blocks together – there is still just a single right-hand side vector $\boldsymbol{b}$.

The Lagrangian in this case is also separable:

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}) &= \sum_{i=1}^{B} f_i(\boldsymbol{x}^{(i)}) + \boldsymbol{\nu}^{\mathrm{T}} \left(\sum_{i=1}^{B} \boldsymbol{A}^{(i)} \boldsymbol{x}^{(i)} \right) - \boldsymbol{\nu}^{\mathrm{T}} \boldsymbol{b} \\ &= \sum_{i=1}^{B} \left[f_i(\boldsymbol{x}^{(i)}) + \boldsymbol{\nu}^{\mathrm{T}} \boldsymbol{A}^{(i)} \boldsymbol{x}^{(i)} - \frac{1}{B} \boldsymbol{\nu}^{\mathrm{T}} \boldsymbol{b} \right] \\ &= \sum_{i=1}^{B} \mathcal{L}_i(\boldsymbol{x}^{(i)}, \boldsymbol{\nu}).\end{aligned}$$

For fixed $\boldsymbol{\nu}_k$, the primal update

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \ \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}_k)$$

can be separated into B independent updates:

$$\boldsymbol{x}_{k+1}^{(i)} = \arg\min_{\boldsymbol{x}^{(i)}} \ \mathcal{L}_i(\boldsymbol{x}^{(i)}, \boldsymbol{\nu}_k)$$

You can imagine broadcasting the current dual iterate to B different processors; they each compute there piece of the primal update $\boldsymbol{x}_{k+1}^{(i)}$, and then the results are gathered centrally to compute the dual update. Notice also, though, that the hard part of the dual update, computing $\boldsymbol{A}\boldsymbol{x}_{k+1}$, can also be done in a decentralized manner.

The Method of Multipliers and Augmented Lagrangians

The method of multipliers (MoM) is the same idea as dual ascent, but we smooth out (augment) the Lagrangian to make the primal update more robust.

It should be clear that

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \;\; f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

and

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \;\; f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

have exactly the same set of solutions for all $\rho \geq 0$.

The Lagrangian for the second program is

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

This is called the **augmented Lagrangian** of the original problem.

Adding the quadratic term is nice – it makes (under mild conditions on f with respect to $\boldsymbol{A}$) the primal update minimization well-posed (i.e., makes the dual differentiable).

Notice that the Lagrange multipliers $\boldsymbol{\nu}$ appear in exactly the same way in the augmented Lagrangian as they do in the regular Lagrangian, so the dual update does not change.

The resulting algorithm is called the **method of multipliers**; we iterate

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\nu}_k)$$
$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + \rho(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b})$$

until some convergence criteria is met.

As a bonus, we now have a principled way of selecting the step size for the dual update – just use ρ. To see why this makes sense, recall the KKT conditions for $\boldsymbol{x}^\star$ and $\boldsymbol{\nu}^\star$ to be a solution:

$$\boldsymbol{A}\boldsymbol{x}^\star = \boldsymbol{b}, \quad \nabla f(\boldsymbol{x}^\star) + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}^\star = \boldsymbol{0}.$$

With ρ as the step size, we have

$$\begin{aligned}
\boldsymbol{0} &= \nabla \mathcal{L}_\rho(\boldsymbol{x}_{k+1}, \boldsymbol{\nu}_k), \quad \text{(since } \boldsymbol{x}_{k+1} \text{ is a minimizer)}, \\
&= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^{\mathrm{T}} \left(\boldsymbol{\nu}_k + \rho(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b})\right) \\
&= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}_{k+1}.
\end{aligned}$$

So the dual update maintains the second optimality condition at every step.

The MoM has much better convergence properties than dual ascent, but it is no longer separable. The algorithm we look at next, the alternating direction method of multipliers (ADMM), will build on this idea in such a way that we do have a type of dual decomposition for the smoothed Largrangian. In addition, it can be easily modified to incorporate certain kinds of inequality constraints.

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.