Algorithms for constrained optimization

There are many, many constrained optimization algorithms, each tuned to the particulars of different classes of problems. We will look at the basics that underlie some of the modern techniques. We will see that the concept of duality both help us understand how these algorithms work, and gives us a way of determining when we are close to the solution.

The three basic techniques presented here, a trick to remove linear equality constraints, barrier methods, and alternating primal-dual methods, all try to replace the constrained program with an unconstrained program (or a series of unconstrained programs).

Eliminating equality constraints

Programs with linear equality constraints can always be written as programs without, regardless of whether there are other inequality constraints — if there are not, the new program is unconstrained. Suppose we are solving

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\ f_0(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$

Let $\boldsymbol{x}_0$ be any feasible point, $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{b}$. Then we can re-write the program above as a search over $\operatorname{Null}(\boldsymbol{A})$:

$$\underset{\boldsymbol{h}\in\mathbb{R}^N}{\text{minimize}}\ f_0(\boldsymbol{x}_0 + \boldsymbol{h}) \quad \text{subject to} \quad \boldsymbol{h} \in \operatorname{Null}(\boldsymbol{A}).$$

Since $\operatorname{Null}(\boldsymbol{A})$ is a linear subspace of dimension $K = N - \operatorname{rank}(\boldsymbol{A})$, there is an $N \times K$ matrix $\boldsymbol{Q}$ whose columns span $\operatorname{Null}(\boldsymbol{A})$ (in fact, there are many such matrices). Note that for such $\boldsymbol{Q}$, we will have $\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{0}$. We can now re-write the program above as

$$\underset{\boldsymbol{w}\in\mathbb{R}^K}{\text{minimize}}\ f_0(\boldsymbol{x}_0 + \boldsymbol{Q}\boldsymbol{w}).$$

Sometimes this method can be very helpful, and sometimes it does not make things much easier — it could be the case that computing $\boldsymbol{x}_0$ and $\boldsymbol{Q}$ is as expensive as just solving the program directly.

Barrier methods

We saw a little of this already. The basic idea is to enforce the inequality constraints by introducing a penalty in the objective that blows up at the boundary of the feasible set. The constrained program

$$\underset{\boldsymbol{x}}{\text{minimize}}\ f_0(\boldsymbol{x}) \quad \text{subject to} \quad f_m(\boldsymbol{x}) \le 0, \quad m = 1, \ldots, M$$

becomes

$$\underset{\boldsymbol{x}}{\text{minimize}}\ f_0(\boldsymbol{x}) + \sum_{m=1}^{M} B(f_m(\boldsymbol{x})),$$

where $\operatorname{dom} B = \mathbb{R}_-$ and $B(x) \to \infty$ as $x \to 0$ from the left. Of course, unless B is the indicator $1_{x<0}$, the new program is an approximation to the original.

An interesting choice is $B(x) = -\frac{1}{\tau}\log(-x)$. This particular barrier function has the properties :

- You can analyze the number of Newton iterations needed for convergence for many f_0 of interest (using self-concordance). This is done very nicely in Chapters 9 and 11 of [BV04].
- The solution[1] $\boldsymbol{x}^\star(\tau)$ of

$$\underset{\boldsymbol{x}}{\text{minimize}}\ \tau f_0(\boldsymbol{x}) - \sum_{m=1}^{M} \log(-f_m(\boldsymbol{x}))$$

[1]We have multiplied the objective by τ to make some of what follows a little easier to express.

can be used to generate a dual-feasible point (and hence a primal-dual gap certificate), and be related to the KKT conditions for the original program.

To appreciate the second point above, we start by taking the gradient of the functional above and setting it equal to zero. We see that

$$\tau \nabla f_0(\boldsymbol{x}^\star(\tau)) + \sum_{m=1}^{M} -\frac{1}{f_m(\boldsymbol{x}^\star(\tau))} \nabla f_m(\boldsymbol{x}^\star(\tau)) = \mathbf{0}.$$

So if we take

$$\lambda_m^\star(\tau) = -\frac{1}{\tau f_m(\boldsymbol{x}^\star(\tau))},\ m = 1, \ldots, M,$$

we have $\lambda_m^\star(\tau) \geq 0$ and

$$f_0(\boldsymbol{x}^\star(\tau)) + \sum_{m=1}^{M} \lambda_m^\star(\tau) \nabla f_m(\boldsymbol{x}^\star(\tau)) = \mathbf{0}.$$

Since $\boldsymbol{x}^\star(\tau)$ is primal feasible, the only KKT condition we are missing is complementary slackness — we have replaced the condition

$$\lambda_m^\star f_m(\boldsymbol{x}^\star) = 0, \quad \text{with} \quad \lambda_m^\star(\tau) f_m(\boldsymbol{x}^\star(\tau)) = -1/\tau.$$

So as τ gets large, we are producing points that meet the KKT conditions.

With this choice of $\boldsymbol{\lambda}^\star(\tau)$, we can also easily compute the dual of the original program:

$$\begin{aligned} g(\boldsymbol{\lambda}^\star(\tau)) &= \inf_{\boldsymbol{x}} \left(f_0(\boldsymbol{x}) + \sum_{m=1}^{M} \lambda_m^\star(\tau) f_m(\boldsymbol{x}) \right) \\ &= f_0(\boldsymbol{x}^\star(\tau)) + \sum_{m=1}^{M} \lambda_m^\star(\tau) f_m(\boldsymbol{x}^\star(\tau)) \\ &= f_0(\boldsymbol{x}^\star(\tau)) - m/\tau. \end{aligned}$$

So then we know that if $p^\star$ is the primal optimal value of the original program,

$$f_0(\boldsymbol{x}^\star(\tau)) - p^\star \;\leq\; f_0(\boldsymbol{x}^\star(\tau)) - g(\boldsymbol{\lambda}^\star(\tau)) \;\leq\; m/\tau.$$

So we know that solving the log-barrier problem gets us within m/τ of the optimal of the original primal objective.

A full discussion of log barrier methods, including some fundamental complexity analysis, can be found in [BV04, Chap. 11]. One interesting theoretical result there is that, with a reasonable way of adjusting τ (multiplying it by 10 at every iteration, for example), the number of log-barrier iterations to make the value of the barrier functional $f_0(\boldsymbol{x}^\star(\tau))$ agree with the minimal value $p^\star$ to the original constrained problem to some precision. The upshot is that there is a very close match after $\sim \sqrt{M}$ iterations. This means that in theory, solving a constrained problem is roughly as expensive as solving $\sqrt{M}$ unconstrained problems. In practice, it is actually much cheaper — standard log barrier iterations take maybe 20–50 iterations to produce good results.

Primal dual interior point methods

These are closely related to log barrier algorithms, but they take a more direct approach towards "solving" the KKT conditions. The general idea is to treat the KKT conditions like a set of nonlinear equations, and solve them using Newton's method.

We start with the same set of relaxed KKT conditions[2] we used with log barrier:

$$\boldsymbol{r}_\tau(\boldsymbol{x},\boldsymbol{\lambda}) = \begin{bmatrix} \nabla f_0(\boldsymbol{x}) + \sum_m \lambda_m \nabla f_m(\boldsymbol{x}) \\ -\lambda_1 f_1(\boldsymbol{x}) - 1/\tau \\ \vdots \\ -\lambda_m f_m(\boldsymbol{x}) - 1/\tau \end{bmatrix}$$

If we find $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ such that the $N+M$-vector $\boldsymbol{r}_\tau(\boldsymbol{x},\boldsymbol{\lambda}) = \boldsymbol{0}$, then we know we have found the same $\boldsymbol{x}^\star(\tau)$, $\boldsymbol{\lambda}^\star(\tau)$ that solve the log barrier problem.

Primal-dual interior point methods take Newton steps to try to make $\boldsymbol{r}_\tau = \boldsymbol{0}$, but they adjust τ at every step. The Newton step is characterized by

$$\boldsymbol{r}_\tau(\boldsymbol{x}+\delta\boldsymbol{x},\boldsymbol{\lambda}+\delta\boldsymbol{\lambda}) \;\approx\; \boldsymbol{r}_\tau(\boldsymbol{x},\boldsymbol{\lambda}) + \boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x},\boldsymbol{\lambda}) \begin{bmatrix} \delta\boldsymbol{x} \\ \delta\boldsymbol{\lambda} \end{bmatrix} = 0,$$

where $\boldsymbol{J}_{\boldsymbol{r}_\tau}$ is the **Jacobian** matrix for the vector-valued function $\boldsymbol{r}_\tau$:

$$\boldsymbol{J}_{\boldsymbol{r}_\tau}(\boldsymbol{x},\boldsymbol{\lambda}) = \begin{bmatrix} \nabla^2 f_0(\boldsymbol{x}) + \sum_m \lambda_m \nabla^2 f_m(\boldsymbol{x}) & \nabla f_1(\boldsymbol{x}) & \nabla f_2(\boldsymbol{x}) & \cdots & \nabla f_m(\boldsymbol{x}) \\ -\lambda_1 \nabla f_1(\boldsymbol{x})^{\mathrm{T}} & -f_1(\boldsymbol{x}) & 0 & \cdots & 0 \\ -\lambda_2 \nabla f_2(\boldsymbol{x})^{\mathrm{T}} & 0 & -f_2(\boldsymbol{x}) & \cdots & 0 \\ \vdots & & & \ddots & \\ -\lambda_M \nabla f_M(\boldsymbol{x})^{\mathrm{T}} & 0 & 0 & \cdots & -f_M(\boldsymbol{x}) \end{bmatrix}$$

[2]We are again only considering inequality constraints; it is straightforward to modify everything we say here to include linear equality constraints.

The update direction is

$$\begin{bmatrix} \delta\boldsymbol{x} \\ \delta\boldsymbol{\lambda} \end{bmatrix} = -\boldsymbol{J}_{\boldsymbol{r}_\tau}^{-1}\boldsymbol{r}_\tau(\boldsymbol{x}, \boldsymbol{\lambda}).$$

With this direction, we can perform a line search. The parameter τ is updated at every step (getting larger) based on an estimate of the duality gap.

A key feature of this type of primal dual method is that the iterates $\boldsymbol{x}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$ do not have to be feasible (although they of course become feasible in the limit).

Details on this particular algorithm, along with a full convergence analysis, can be found in [BV04, Chap. 11.7].

Alternating direction primal dual methods

We will 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,

$$\begin{aligned} f_0(\boldsymbol{x}^\star) &= g(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \\ &= \inf_{\boldsymbol{x}\in\mathbb{R}^N} L(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star), \end{aligned}$$

where L is the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \sum_{m=1}^{M} \lambda_m f_m(\boldsymbol{x}) + \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}).$$

If $L(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ has only one minimizer[3], 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

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

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

[3]Which is the case when f_0 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

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\ f_0(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$

We will assume that the domain of f_0 is all of $\mathbb{R}^N$; again, things are easily modified if this is any open set. The Lagrangian is

$$L(\boldsymbol{x},\boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}),$$

and the dual is

$$\begin{aligned} g(\boldsymbol{\nu}) &= \inf_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\nu}) \\ &= -f_0^*(-\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}) - \langle \boldsymbol{\nu}, \boldsymbol{b}\rangle, \end{aligned}$$

where f_0^* is the Fenchel conjugate of f_0:

$$f_0^*(\boldsymbol{y}) = \sup_{\boldsymbol{x}\in\mathbb{R}^N} \left(\langle \boldsymbol{x}, \boldsymbol{y}\rangle - f_0(\boldsymbol{x})\right).$$

The dual problem is

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

Consider for a moment the problem of maximizing the dual. A reasonable thing to do would be some kind of gradient ascent[4],

$$\boldsymbol{\nu}^{(k+1)} = \boldsymbol{\nu}^{(k)} + t_k \nabla g(\boldsymbol{\nu}^{(k)}),$$

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

where t_k is some appropriate step size. The gradient of g at a point ν_0 is

$$\nabla g(\boldsymbol{\nu}_0) = \nabla_{\boldsymbol{\nu}} \inf_{\boldsymbol{x}} \left(f_0(\boldsymbol{x}) + \langle \boldsymbol{\nu}_0, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \rangle \right),$$

and so if $\boldsymbol{x}^+ = \arg\min_{\boldsymbol{x}} (f_0(\boldsymbol{x}) + \langle \boldsymbol{\nu}_0, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \rangle)$, then

$$\begin{aligned} \nabla_{\boldsymbol{\nu}} g(\boldsymbol{\nu}_0) &= \nabla_{\boldsymbol{\nu}} \left(f_0(\boldsymbol{x}^+) + \langle \boldsymbol{\nu}_0, \boldsymbol{A}\boldsymbol{x}^+ - \boldsymbol{b} \rangle \right) \\ &= \boldsymbol{A}\boldsymbol{x}^+ - \boldsymbol{b}. \end{aligned}$$

This leads naturally to:

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

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

that is repeated until some convergence criteria is met.

This algorithm "works" under certain assumptions on f_0 (that translate to different assumptions on the dual g). In particular, we need $L(\boldsymbol{x}, \boldsymbol{\nu})$ to be bounded for every $\boldsymbol{\nu}$, otherwise the primal update $\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\nu}^{(k)})$ 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

$$\underset{\boldsymbol{x}_1, \boldsymbol{x}_2}{\text{minimize}}\, f_0(\boldsymbol{x}_1) + \langle \boldsymbol{x}_2, \boldsymbol{c} \rangle \quad \text{suject to} \quad \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 = \boldsymbol{b}$$

will have Lagrangian

$$\begin{aligned} L(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{\nu}) &= f_0(\boldsymbol{x}_1) + \langle \boldsymbol{x}_2, \boldsymbol{c} \rangle + \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 - \boldsymbol{b}) \\ &= f_0(\boldsymbol{x}_1) + \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}_1\boldsymbol{x}_1 - \boldsymbol{b}) + (\boldsymbol{c} + \boldsymbol{A}_2^{\mathrm{T}}\boldsymbol{\nu})^{\mathrm{T}}\boldsymbol{x}_2, \end{aligned}$$

which is unbounded below since the last term is linear in $\boldsymbol{x}_2$.

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_0 is **separable**, we can also separate (i.e. parallelize) the primal update. Suppose we can write $f_0(\boldsymbol{x})$ as a sum

$$f_0(\boldsymbol{x}) = \sum_{i=1}^{B} f_i(\boldsymbol{x}_i),$$

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:

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

For fixed $\boldsymbol{\nu}^{(k)}$, the primal update

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

can be separated into B independent updates:

$$\boldsymbol{x}_i^{(k+1)} = \arg\min_{\boldsymbol{x}_i} L_i(\boldsymbol{x}_i,\boldsymbol{\nu})$$

You can imagine broadcasting the current dual iterate to B different processors; they each compute there piece of the primal update $\boldsymbol{x}_i^{(k+1)}$, 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

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\ f_0(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

and

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\ f_0(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

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

The Lagrangian for the second program is

$$L_\rho(\boldsymbol{x}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \boldsymbol{\nu}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}).$$

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

Adding the quadratic term is nice — it makes (under mild conditions on f_0 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 was 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

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \arg\min_{\boldsymbol{x}} L_\rho(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}) \\ \boldsymbol{\nu}^{(k+1)} &= \boldsymbol{\nu}^{(k)} + \rho(\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b})\end{aligned}$$

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:

$$\boldsymbol{A}\boldsymbol{x}^\star = \boldsymbol{b}, \quad \nabla f_0(\boldsymbol{x}^\star) + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}^\star = \mathbf{0}.$$

With ρ as the step size, we have

$$\begin{aligned}\mathbf{0} &= \nabla_{\boldsymbol{x}} L_\rho(\boldsymbol{x}^{(k+1)}, \boldsymbol{\nu}^{(k)}), \quad \text{(since } \boldsymbol{x}^{(k+1)} \text{ is a minimizer)}, \\ &= \nabla f_0(\boldsymbol{x}^{(k+1)}) + \boldsymbol{A}^{\mathrm{T}}\left(\boldsymbol{\nu}^{(k)} + \rho(\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b})\right) \\ &= \nabla f_0(\boldsymbol{x}^{(k+1)}) + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}^{(k+1)}.\end{aligned}$$

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.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.