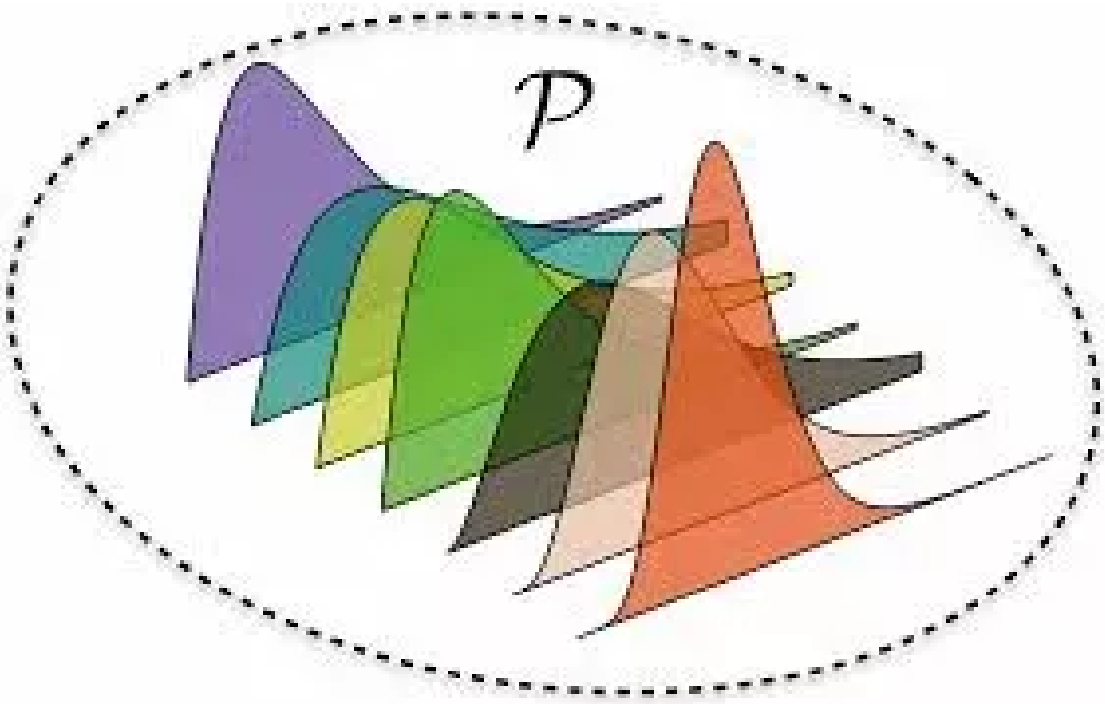
优化 | 基于ADMM的OSQP求解器介绍

Original 运筹OR帷幄 运筹OR帷幄 7/26

收录于话题
#运筹学

70个

↑↑↑↑↑点击上方蓝色字关注我们!



『运筹OR帷幄』原创

作者：陈宇文

编者按

经典永不过时，虽然ADMM算法很久之前就被提及，但现在仍然有诸多研究围绕着ADMM算法展开。本文对ADMM算法的求解器OSQP进行了详细介绍，包含了ADMM算法的框架，求解器是如何对将问题转化为标准的ADMM框架及发散思维的关键比较。

基于ADMM的OSQP求解器介绍

前几天刚看了@门泊东吴写的基于ADMM的Splitting Conic Solver (SCS) [1] 的推文 "大规模锥优化之Splitting Conic Solver(SCS)" (附1)，便在这儿安利一波另外一个基于ADMM的开源solver，名字叫Operator Splitting Quadratic Program (OSQP)。两个求解器底层idea同出一门（都有Stephen Boyd影子，大牛就是大牛，无处不在...），但是深入比较时又会发现各有各的特点。如果大家对ADMM不太了解，可以先看看之前@含章写的 ADMM算法的详细推导过程 (附2)和@门泊东吴在 交替方向乘子法（ADMM）算法的流程和原理是怎样的？ (附3)中的回答。当然有时间、英文好的同学可以直接去啃Stephen Boyd关于ADMM总结的小册子 [2]，之后的讨论中我们也会反复联系 [2]中出现的内容.（本文有关的都是凸优化问题）

ADMM基本思想回顾

首先，我们在这儿大致回忆一下ADMM的基本框架。假设原凸优化问题模型为

$$\begin{aligned} \min \quad & f(X)+g(Y) \\ \text{s.t.} \quad & BX=Y, \quad X\in\mathcal{X}, Y\in\mathcal{Y} \end{aligned}$$

与之相关的维度为 $X\in\mathbb{R}^N, Y\in\mathbb{R}^M, B\in\mathbb{R}^{M\times N}$。对应的增广拉格朗日 L_ρ 则为：

$$L_\rho(X,Y,\Lambda) := f(X)+g(Y)+\Lambda^\top(BX-Y)+\frac{\rho}{2}||BX-Y||^2$$

ADMM算法可以看成是一种primal-dual algorithm，在一个内循环周期内依次进行原变量 (primal) X,Y 以及对偶变量 (dual) Λ 的更新: 首先求解一个有关 X 的子问题，然后求解一个有关 Y 的子问题，最后再对 Λ 进行一次误差更正。具体步骤如下：

$$\begin{aligned} X_{k+1} &= \arg\min_{X\in\mathcal{X}} L_\rho(X,Y_k,\Lambda_k) \\ Y_{k+1} &= \arg\min_{Y\in\mathcal{Y}} L_\rho(X_{k+1},Y,\Lambda_k) \\ \Lambda_{k+1} &= \Lambda_k+\rho(BX_{k+1}-Y_{k+1}) \end{aligned} \tag{1}$$

在接下来的文章中，我们会一步步阐述如何将OSQP变形后的ADMM算法化成 (1) 中最基本的框架，但同时我也会指出OSQP的算法底层会和其他哪些经典的方法有着内在联系。这可能是我觉得算法中最有趣的事情：由繁入简，许多看起来复杂的算法最后都能简化统一到一个经典的算法类别；由简入繁，通过融合不同算法的底层思想，往往我们又能设计出许多精巧优美的新算法。

基于上述这个理念，文章接下来也将按照这个逻辑进行叙述：第一部分，我们先简单粗暴地给出OSQP的底层算法；第二部分，我们将解释如何将OSQP算法转化成标准的ADMM算法框架 (1)；第三部分，我将试着根据自己的理解将OSQP算法中的每一个模块与之前介绍的 SCS 进行关联和比较，同时发散思维，尝试将OSQP算法与自己曾经读过的一些优化知识进行穿插联系（没事就喜欢瞎折腾...）。

（第一部分）OSQP底层算法介绍

OSQP求解器求解的模型如下：

$$\begin{aligned} \min \quad & \frac{1}{2}x^\top Px + q^\top x \\ \text{s.t.} \quad & l \le Ax \le u \end{aligned} \tag{2}$$

上述问题可以写成如下的更加一般的形式：

$$\begin{aligned} \min \quad & \frac{1}{2}x^\top Px + q^\top x \\ \text{s.t.} & Ax = z, \quad z \in \mathcal{C} \end{aligned} \tag{3}$$

若取 $\mathcal{C} := [l, u]$，那么问题(3)就简化成了问题(2).

接下来我们先给出**OSQP算法实现**：

$$\begin{aligned} (x^{k+1}, \nu^{k+1}) &\leftarrow \text{solve linear system} \\ \tilde{z}^{k+1} &\leftarrow z^k + \rho^{-1}(\nu^{k+1} - y^k) \\ z^{k+1} &\leftarrow \Pi_{\mathcal{C}}(\tilde{z}^{k+1} + \rho^{-1}y^k) \\ y^{k+1} &\leftarrow y^k + \rho(\tilde{z}^{k+1} - z^{k+1}) \end{aligned} \tag{4}$$

$\Pi_{\mathcal{C}}(x)$ 的含义是 x 在集合 $\mathcal{C}$ 上的投影，ρ 是增广拉格朗日方法中的系数。此外，上述第一步中对应的求解线性系统 (solve linear system) 对应的系统如下：

$$\begin{bmatrix} P + \sigma I & A^T \\ A & -\rho^{-1}I \end{bmatrix} \begin{bmatrix} x^{k+1} \\ \nu^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \rho^{-1}y^k \end{bmatrix} \tag{5}$$

乍一看，OSQP算法(4)与基本版本的ADMM算法很相似：(x, y) ，$(\tilde{x}, \tilde{y})$ 分别对应(1)中的变量 X, Y，可以将求解线性方程是求解一个关于二次型优化子问题，将投影过程 $\Pi_{\mathcal{C}}$ 可以看成对于 z 的一个优化子问题；好像先更新变量 (primal) x, z 再更新变量 (primal) $\tilde{x}, \tilde{z}$，最后更新对偶变量 (dual) y ..... 但是仔细一看好像不太对劲：如果 (x, z) 是一组 primal 变量，$(\tilde{x}, \tilde{z})$ 是另外一组primal 变量，那么为什么 $\tilde{z}^{k+1}$ 更新夹在 x^{k+1} z^{k+1} 更新的中间而不是有序地放在 (x, z) 更新之后？（怎么依次有序的变成混搭的了？？？）另外 $\tilde{x}^{k+1}$ 的更新到哪儿去了？（等等，好像少了什么...） ν^{k+1} 又是个什么变量？（这又是哪里多出来的...）..... 如果一直对着(1)和(4)来回看，朦胧之中直觉告诉你这和ADMM有关，但是为什么又有这么多不一致的地方呢？ **（简单≠易懂）**

这就是接下来第二部分要讲的内容了：因为求解器主页上的给出的解释是进行了 瘦身 的（偷工减料了...），想要了解OSQP算法全貌还得回去读论文 [3]。（该做的一个都躲不掉）

（第二部分）由繁入简：统一到ADMM经典框架

论文 [3] 中算法的原貌本来应该是这样的：问题 (3) 的首先写成如下的**等价形式**：

$$\begin{aligned} \min \quad & \frac{1}{2}\tilde{x}^\top P\tilde{x} + q^\top \tilde{x} + \mathcal{I}_{Ax=z}(\tilde{x}, \tilde{z}) + \mathcal{I}_{\mathcal{C}}(z) \\ \text{s.t.} \quad & (\tilde{x}, \tilde{z}) = (x, z) \end{aligned} \tag{6}$$

其中 $\mathcal{I}_{Ax=z}$ 和 $\mathcal{I}_{\mathcal{C}}$ 是我们常说的 indicator functions：

$$\mathcal{I}_{Ax=z}(x, z) = \begin{cases} 0 & Ax = z \\ +\infty & \text{otherwise} \end{cases}, \qquad \mathcal{I}_{\mathcal{C}}(z) = \begin{cases} 0 & z \in \mathcal{C} \\ +\infty & \text{otherwise} \end{cases}.$$

问题 (6) 对应的**增广Lagrange $L_{\rho,\sigma}$** 如下所示：

$$
\begin{aligned}
&L_{\rho,\sigma}(\tilde{x},\tilde{z},x,z,w,y)\\
=&\frac{1}{2}\tilde{x}^\top P\tilde{x}+q^\top\tilde{x}+\mathcal{I}_{Ax=z}(\tilde{x},\tilde{z})+\mathcal{I}_{\mathcal{C}}(z)+w^\top(\tilde{x}-x)\\
&+y^\top(\tilde{z}-z)+\frac{\sigma}{2}\|\tilde{x}-x\|^2+\frac{\rho}{2}||\tilde{z}-z||^2
\end{aligned}
$$

在这儿，我们选取 σ,ρ 分别作为 $\tilde{x}=x,\tilde{z}=z$ 的两个等式对应的增广lagrange系数。

接下来我们来详细讨论一下OSQP算法与基本的ADMM框架 (1) 之间的关系了，两者之间的变量对应关系如下：

$$
\begin{aligned}
X=(\tilde{x},\tilde{z}),&\qquad \mathcal{X}:=\{A\tilde{x}=\tilde{z}|\tilde{x},\tilde{z}\}\\
Y=(x,z),&\qquad \mathcal{Y}:=\{x\in\mathbf{R}^n,z\in\mathcal{C}\}\\
\Lambda=(w,y)&
\end{aligned}
$$

同时矩阵 B 取为单位矩阵。注意到，在这儿我们又将 (6)中的indication function $\mathcal{I}_{Ax=z}$, $\mathcal{I}_{\mathcal{C}}$ 分别看成变量 $(\tilde{x},\tilde{z})$, (x,z) 的作用域。由此,仿照(1)的ADMM框架，我们可以得出**问题(6)对应的ADMM算法的完整形式**如下

$$
\begin{aligned}
(\tilde{x}^{k+1},\tilde{z}^{k+1})\leftarrow&\arg\min_{(\tilde{x},\tilde{z}):A\tilde{x}=\tilde{z}}\frac{1}{2}\tilde{x}^\top P\tilde{x}+q^\top\tilde{x} &(7)\\
&+\frac{\sigma}{2}\|\tilde{x}-x^k+\sigma^{-1}w^k\|_2^2+\frac{\rho}{2}\|\tilde{z}-z^k+\rho^{-1}y^k\|_2^2\\
x^{k+1}\leftarrow&\tilde{x}^{k+1}+\sigma^{-1}w^k &(8)\\
z^{k+1}\leftarrow&\Pi_{\mathcal{C}}\left(\tilde{z}^{k+1}+\rho^{-1}y^k\right) &(9)\\
w^{k+1}\leftarrow&w^k+\sigma\left(\tilde{x}^{k+1}-x^{k+1}\right) &(10)\\
y^{k+1}\leftarrow&y^k+\rho\left(\tilde{z}^{k+1}-z^{k+1}\right) &(11)
\end{aligned}
$$

其中 (7)的objective function 是对 $\arg\min L_{\rho,\sigma}$ 问题的一种常见变形。

（跑个题：细心的读者可能会发现上述的算法和文献 [3] 的原始算法有稍微差异，例如论文中步骤 (8) 和 (9) 的 $\tilde{x}^{k+1}$, $\tilde{z}^{k+1}$ 应该写成 $\alpha\tilde{x}^{k+1}+(1-\alpha)x^k$, $\alpha\tilde{z}^{k+1}+(1-\alpha)z^k$ 的形式，为了方便算法理解，我们在这儿取 $\alpha=1$。但实际上 α 可以取 $[0,2]$ 中的任何值。我们将在第三部分对 α 的含义进行简单的讨论。）

到这儿，算是初步把完整的OSQP算法与咱们熟悉的ADMM框架 (1) 结合在了一起（嗯，的确是熟悉的配方～～）。

我们剩下需要做的是，**如何将OSQP完整的算法 (7)-(11) 简化成公式 (4) 中所对应的算法呢?**

算法简化

当设初始变量 $w^0=0$ 时，依据公式 (8) 和 (10)， 我们发现 w^{k+1} **将恒为0，！！！ 因此可以从算法中剔除掉。** 接着，我们很容易通过 (8) 得到 $x^{k+1}=\tilde{x}^{k+1}$。这时，(7) 中的子问题可以进一步简化成：

$$\begin{aligned}\min_{(\tilde{x},\tilde{z}):A\tilde{x}=\tilde{z}} \quad & \frac{1}{2}\tilde{x}^\top P\tilde{x}+q^\top\tilde{x}+\frac{\sigma}{2}\|\tilde{x}-x^k\|_2^2 \\ & +\frac{\rho}{2}\|\tilde{z}-z^k+\rho^{-1}y^k\|_2^2\end{aligned} \tag{12}$$

对应地，算法(7)-(11) 可以初步简化成

$$\begin{aligned}(x^{k+1},\tilde{z}^{k+1}) &\leftarrow \arg\min_{(\tilde{x},\tilde{z}):A\tilde{x}=\tilde{z}} \frac{1}{2}\tilde{x}^\top P\tilde{x}+q^\top\tilde{x}+\frac{\sigma}{2}\|\tilde{x}-x^k\|_2^2 \\ &\qquad +\frac{\rho}{2}\|\tilde{z}-z^k+\rho^{-1}y^k\|_2^2 \\ z^{k+1} &\leftarrow \Pi_{\mathcal{C}}\left(\tilde{z}^{k+1}+\rho^{-1}y^k\right) \\ y^{k+1} &\leftarrow y^k+\rho\left(\tilde{z}^{k+1}-z^{k+1}\right)\end{aligned} \tag{13}$$

到这儿，对比 (13) 和 (4)，我们发现唯一的区别可能就是 $\tilde{z}^{k+1}$ 的更新，以及多出来的变量 ν^{k+1}。 **这也就是说如果我们能够说明（13）中的第一步与（4）中的前两步等价，那我们也可以说明算法（7)-(11）与（4）等价** 。（真相就要浮出水面了...）

那么如何得到两者是等价的呢？这时候就又回到了**最优化判断中常用的KKT条件**了： **对于一个二次型优化问题，假如对应的限制条件是affine形式，那么最终的KKT形式可以写成矩阵形式** 。对应 (13) 中第一步子优化问题的KKT条件形式如下：

$$\begin{aligned}P\tilde{x}^{k+1}+q+\sigma(\tilde{x}^{k+1}-x^k)+A^\top\nu^{k+1}&=0 \\ \rho(\tilde{z}^{k+1}-z^k)+y^k-\nu^{k+1}&=0 \\ A\tilde{x}^{k+1}-\tilde{z}^{k+1}&=0\end{aligned} \tag{14}$$

上述等式中，$\tilde{x}^{k+1},\tilde{z}^{k+1}$ 是primal最优解，ν^{k+1} 是dual最优解（ν 对应的是子问题 (12) 中 $A\tilde{x}=\tilde{z}$ 条件的对偶变量）。将上式中的 $\tilde{z}^{k+1}$ 进行消元以及利用 $x^{k+1}=\tilde{x}^{k+1}$，我们可以得到如下的矩阵形式：

$$\begin{bmatrix}P+\sigma I & A^T \\ A & -\rho^{-1}I\end{bmatrix}\begin{bmatrix}x^{k+1} \\ \nu^{k+1}\end{bmatrix}=\begin{bmatrix}\sigma x^k-q \\ z^k-\rho^{-1}y^k\end{bmatrix} \tag{15}$$

这个和公式 (5)中的矩阵求解完全一致！！！

除此之外，当通过矩阵计算得到 ν^{k+1} 后，我们可以通过(14)中的第二个等式还原得到 $\tilde{z}^{k+1}$ 的值：

$$\tilde{z}^{k+1}=z^k+\rho^{-1}(\nu^{k+1}-y^k)$$

这也就是(4)中的第二步。由此我们说明了 (13) 中的第一步与 (4) 中的前两步等价，进而说明算法 (13) 与算法 (4) 是一样的。同时，我们还知道 (13) 是完整的OSQP底层算法 (7)-(11)的简化形式，所以 (4)同样是(7)-(11) 的简化形式。

（第三部分）由简入繁：OSQP算法与其他算法的联系

介绍完了OSQP底层算法的大致思想，这一部分更加类似我的个人学习总结。如果把OSQP算法进行拆解细究，其实会发现精妙的算法其实往往就是几个**经典的结果进行交叉融合**的产物，

很多细节其实早就有迹可寻，而我在这一部分想做的就是尝试提取出算法的几个支撑点，并尝试对它们进行溯源。同时因为这部分的侧重点是**建立与其他知识点的联系**，所以在这儿也就不对涉及到的知识点细细展开了。喜欢刨根问底的同学可以根据我每一部分提供的相关内容的链接进行深入研究。

求解线性系统子问题

当我们将OSQP与SCS算法进行对比时，首先最直观的区别是OSQP [3] 求解的是二次型cost凸优化问题而SCS [1] 求解的是一个线性cost凸优化问题。但是看似不同的问题模型底层求解思想却是一致的： **在每一个内循环(inner loop)求解线性系统子问题(solving linear system)实际上对应的是一个二次型cost的子问题的求解(两者之前是等价的)。**

这个思想其实Stephen Boyd 在文献 [2] 中就已经提到了（见 [2] 中4.2.5部分）。更早的， 这个思想其实在有关 共轭梯度（Conjugate Gradient）算法的相关资料中就出现过了！！！ （再早我也不没去考证了...）通过引入新的拉格朗日算子求导分析二次型子问题取到最优解时的KKT条件，也就是**类似于 (14), (15) 的矩阵形式**。区别在于，SCS中，求解线性系统对应的是一个简单的投影子问题(见 [1] 的3.2.2部分)，而在OSQP中，这个投影子问题多出来了一个二次型目标函数 $\frac{1}{2}x^\top Px + q^\top x$。细心的读者可以发现，相比较SCS中的矩阵而言，OSQP的线性系统求解步骤中的矩阵左上角多出了一个P模块，同时，对角线上的单位矩阵前面多了系数 σ, ρ (来源于增广拉格朗日法的系数)。

$$K := \begin{bmatrix} P + \sigma I & A^\top \\ A & -\rho^{-1} I \end{bmatrix}$$

涉及更加底层的计算问题时，求解线性系统与SCS [1] 中求解线性系统思路差不多，分为**直接法 (direct method)** 和**间接法 (indirect method)** 两种，而采用两种不同的矩阵计算方法是为了**根据不同的问题模型尽可能地加速计算**（求解器的终极奥义）：

直接法：我们先将(15)看成是矩阵形式 K。对应 (15) 中 K 是quasi-definite的 （第一个对角阵 $P + \sigma I$ 是正定矩阵，第二个对角阵 $-\rho^{-1} I$ 是负定矩阵），可以采用 LDL分解 （一种Cholesky分解的一般形式），将矩阵 K 分解为 K 的形式，其中 L 是三角矩阵和不存在零元素的对角矩阵 D，然后再利用三角矩阵性质，进行向前或向后替换求解 x（跟LU分解类似）：首先求解 $Ly = t$ 中的 y，其中 y 代表 $DL^\top x$，然后再求解 $L^\top x = D^{-1} y$ 中的 x。

直接法的好处在于，当矩阵 P, A 和增广系数 σ, ρ 固定时，我们只需要在初始化的时候进行一次LDL分解 （**factorization**），然后将分解得到的矩阵 L, D^{-1} 存储起来，以后每一次 (4) 的计算只需要进行两次向前（向后）替换求解，每一次矩阵运算的计算复杂度仅为 $\mathcal{O}(n^2)$ （假设矩阵 K 不存在其他特殊结构）。与之相对的，如果我们想要在计算过程中采用变化的 σ, ρ，就得重新做矩阵的LDL分解，而这是**直接法中最耗时的部分**。这时，选择接下来的间接法可能更加合适。

间接法：间接法求解另外一个等价的线性系统 （消去 (15)中的 ν^{k+1}）：

$$\left(P+\sigma I+\rho A^TA\right)x^{k+1}=\sigma x^k-q+A^T(\rho z^k-y^k)$$

相比于直接法，间接法求解基于之前提到过的 共轭梯度（Conjugate Gradient）算法 ，不需要进行矩阵分解，因此也可以更加灵活的调整系数 σ,ρ，甚至矩阵 A。同时，当 K 维度很大时，进行矩阵分解的计算量太大了，采用间接法也能大大提高矩阵系统的计算速度。

和Douglas–Rachford Splitting的关系

细心的读者如果去阅读算法原文 [3]，会发现算法的完整形式应该是如下形式：

$$\begin{aligned}(\tilde{x}^{k+1},\nu^{k+1})&\leftarrow \text{solve linear system}\\ \tilde{z}^{k+1}&\leftarrow z^k+\rho^{-1}(\nu^{k+1}-y^k)\\ x^{k+1}&\leftarrow \alpha\tilde{x}^{k+1}+(1-\alpha)x^k\\ z^{k+1}&\leftarrow \Pi_{\mathcal{C}}(\alpha\tilde{z}^{k+1}+(1-\alpha)z^k+\rho^{-1}y^k)\\ y^{k+1}&\leftarrow y^k+\rho(\alpha\tilde{z}^{k+1}+(1-\alpha)z^k-z^{k+1})\end{aligned}\tag{16}$$

对比 (7)-(11)和 (16)，我们发现唯一多出来的区别在于 (16) 中标注为红色的部分。直观上来看，似乎是多出了一步线性组合的步骤：

$$\begin{pmatrix}x\\z\end{pmatrix}\leftarrow\alpha\begin{pmatrix}\tilde{x}\\\tilde{z}\end{pmatrix}+(1-\alpha)\begin{pmatrix}x\\z\end{pmatrix}$$

但是我们前文提到过，**α 的取值可以取 **$[0,2]$ 不是 $[0,1]$（$\alpha>1$ 时被称为 over relaxation），这个取值区间又是如何确立的呢？而算法 (16) 的收敛性又是如何保障的呢？

在OSQP的相关论文 [4] 中，作者还给出了OSQP算法的另外一种基于Douglas–Rachford Splitting方法的解释，能够让我们更好地理解 $\alpha\in[0,2]$ 的原因。不熟悉Douglas–Rachford Splitting的同学可以先阅读@邓博之前写的有关 算子理论的文章 (附4)或者Stephen Boyd的有关单调算子（monotone operator）的课件 [5] 有个初步了解，更深入的算子理论了解可以阅读 [6]，[7]。

OSQP的Douglas–Rachford Splitting方法的解释如下：

$$T_{DR}=(1-\frac{\alpha}{2})\mathrm{Id}+\frac{\alpha}{2}(2\mathrm{prox}_f-\mathrm{Id})\circ(2\mathrm{prox}_g-\mathrm{Id})\tag{17}$$

其中 Id 是单位算子（$x=\mathrm{Id}(x)$），prox 是我们熟知的临近算子（proximal operator）。函数 f,g 的定义如下：

$$f(x,z)=\frac{1}{2}x^\top Px+q^\top x+\mathcal{I}_{Ax=z}(x,z),g(x,z)=\mathcal{I}_{\mathcal{C}}(z)$$

在这儿，算子 T_{DR} 是作用在变量 $s:=(x,v)$ 上的，即算法迭代过程为 $s^{k+1}=T_{DR}(s^k)$。新的变量 v 和变量 z 有关（$z=\Pi_{\mathcal{C}}(v)$，之后会有详细说明)。因**此运算符 (17)最后取两部分线性组合（对应系数$1-\frac{\alpha}{2}$ 和 $\frac{\alpha}{2}$）对应 (16)的第三行和第四行中标红部分的线性组合**。注意到这时 $\frac{\alpha}{2},1-\frac{\alpha}{2}\in[0,1]$，是一个凸组合（convex combination）。我们称它为**$\alpha/2$-平均算子（$\alpha/2$-averaged）**。

有没有感觉很神奇？ **一个这么复杂的算法可以简洁地由一行运算符表示而成** 。这也是我个人觉得operator splitting method最有趣的原因。（数学就是越简洁越美）如果我们想让 (17)的表达更

加简洁，我们可以引入另外一个新的算子 $C := 2\mathrm{prox} - \mathrm{Id}$（被称作**Cayley operator**），这样 (17) 就可以进一步简化为：

$$T_{DR} = (1 - \frac{\alpha}{2})\mathrm{Id} + \frac{\alpha}{2} C_f \circ C_g \tag{18}$$

那写成算子形式和我们求解优化问题有什么联系呢？原问题(6)可以写成如下形式

$$\min_{x,z} f(x,z) + g(x,z)$$

根据凸优化理论，我们知道最优解 (x^*, z^*) 满足 $0 \in \partial f(x^*, z^*) + \partial g(x^*, z^*)$，***而这个问题可以等价为求解算子 $C_f \circ C_g$ 的不动点问题***：

$$0 \in \partial f(x,z) + \partial g(x,z) \iff C_f \circ C_g(s) = s,\ (x,z) = \mathrm{prox}_g(s) \tag{19}$$

（注意：因为Calyley算子 C_f, C_g 是非扩张的（nonexpansive），所以 $C_f \circ C_g$ 也是非扩张的。详细证明可见 Stanford EE364b 的课程网站的课件[5] （monotone operator splitting methods）。）***而将算子 $C_f \circ C_g$ 的不动点 $\{s \mid C_f \circ C_g(s) = s\}$ 与 Id 进行凸组合（如 (18)），那么新的算子 T_{DR} 迭代将收敛到其不动点，并且同时也是 $C_f \circ C_g$ 的不动点。***（*注意：$C_f \circ C_g$ 是非扩张不能保证 $C_f \circ C_g$ 的迭代能收敛到 $C_f \circ C_g$ 的不动点！！！这也是为什么要多一步凸组合保证收敛性的原因*）

与此同时，我们注意到，由于 $g(x,z) = \mathcal{I}_{\mathcal{C}}(z)$， prox_g ***算子其实就是*** $(x,z) = (x, \Pi_{\mathcal{C}}(v))$。再结合(17),(18),(19)，我们就不难理解为什么***算子 T_{DR} 的变量是 $s = (x,v)$ 而不是*** (x,z) 了。

可行性检验 (Infeasibility Detection)

和SCS [1] 一样，由于进行的是**求解器算法设计，除了求解可行最优解外我们还需要考虑我们能不能在算法中判断是否原问题有可行解**。文献 [4] 专门讨论了OSQP可行性检验的问题，在这儿我就不详细阐述了（其实是底层细节我自己还没看太懂，汗.....）。我想提的是OSQP可行性检验 [4] 和SCS可行性检验的**基本思想**都是 **Theorems of alternatives**，在这我大概对 Theorems of alternatives 的几个重要概念介绍一下（可以类比duality gap），详细内容可见 Stephen Boyd 的 convex optimization 教材中 [8] 中的5.8和5.9.4部分。学艺不精，还希望熟悉这方面的大佬进行指正。

在介绍可行性检验之前，我们先定义两个集合：

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \in \mathcal{C}\}, \tag{20}$$

$$\mathcal{D} = \{y \in \mathbb{R}^m \mid A^\top y = 0, \quad S_{\mathcal{C}}(y) < 0\}, \tag{21}$$

其中 $S_{\mathcal{C}}(y) := \max_{z \in \mathcal{C}} \langle y, z \rangle$。根据 (20) 和 (21)，我们给出有关 theorem of alternatives 中的两个基本概念：

（1）**Weak alternatives**：集合(20),(21)两者中最多有一个成立。这个性质总是成立的（无论问题是否为凸）。

（2）**Strong alternatives**：集合 (20),(21) 两者中一定是一个成立另外一个不成立。在凸优化问题中，如果满足一定条件检验（constraint qualification）[8]，那么weak alternatives可以变成strong alternatives（这么看似乎theorem of alternatives 和对偶定理也有着某种相关性？）。**而这也是OSQP和SCS求解器检验是否有可行解的理论基础。*通过检验 $\mathcal{P},\mathcal{D}$ 哪个集合非空即可确定是否原问题有可行解。***

细致讨论$\mathcal{P},\mathcal{D}$含义之前，我先来介绍一下原问题 (3)的对偶问题为：

$$\begin{aligned} \max_{(x,y)} \quad & -\frac{1}{2}x^\top Px - S_{\mathcal{C}}(y) \\ \text{s.t.} \quad & Px + A^\top y = -q,\ y \in (C^\infty)^\circ \end{aligned} \tag{22}$$

C^∞ 是集合 C 的退化锥（recession cone），加上$'\circ'$ 意味着取 C^∞ 的极锥（polar cone），意味着 y 属于集合 C 的退化锥的极锥。**根据 $S_{\mathcal{C}}(y)$ 的定义，(21)要求 $S_{\mathcal{C}}(y) < 0$ 其实已经隐含了 $y \in (C^\infty)^\circ$ 。** 我们不难猜想，集合 $\mathcal{D}$ 中的 y 和对偶问题 (22)中的对偶变量 y 有着某种联系。事实上也的确是这种联系：

我们回过头来看 (20),(21) 的集合 $\mathcal{P},\mathcal{D}$。结合 (3)，很容易看出 ***$\mathcal{P}$ 对应的是原问题 (3)的可行解的集合***，若假设 $\mathcal{P},\mathcal{D}$ 满足 strong alternatives，那么***$\mathcal{D}$ 对应的是原问题 (3) 不存在可行解的情况。*** 我们有两种方式理解 $\mathcal{P},\mathcal{D}$ 满足 strong alternatives：

（1）[4] 是用**平面分割集合的思想**进行的证明：

若设 $\exists\, \bar{y} \in \mathcal{D}$，则

$$\inf_x \ \langle \bar{y}, Ax \rangle = \inf_x \langle A^\top \bar{y}, x \rangle = 0$$

所以平面 $\langle \bar{y}, z \rangle = 0$ 将集合 $\{Ax | x \in \mathbb{R}^n\}$ 和集合 $\mathcal{C}$ 分割开，也就意味着 $\mathcal{P}$ 是空集。反之，若 $\exists\, \bar{x} \in \mathcal{P}$，可以反证法推出 $\mathcal{D}$ 必为空集。因此，$\mathcal{P},\mathcal{D}$ 满足 strong alternatives。

（2）根据**对偶理论**进行直观的理解（不是严格的证明）。如果 $\mathcal{D}$ 非空且对偶问题(22)存在可行解，那么 $\exists\, \bar{y} \in \mathcal{D}$，$\bar{y}$ 将是对偶问题 (22) 的一个无界方向，意味着对偶问题可以取值 $+\infty$，根据对偶理论 $q^* \le p^*$， $q^* = +\infty$ 说明原问题(3)没有可行解，即 $\mathcal{P}$ 为空集。

同理，仿照 (20),(21)，我们也可以定义关于对偶问题 (22) 的可行性检验的一组集合，这儿就不再详细讨论，有兴趣的读者可自行阅读 [4]。

总结

写这篇文章的目的是介绍OSQP求解器，作为之前SCS求解器（同样基于ADMM的框架）或者是说ADMM小册子 [2] 的后续延伸阅读，在第二部分尝试着阐述OSQP算法与经典ADMM之间的联系，同时在第三部分尝试着将OSQP中自己觉得比较有意思的三个模块（二次型问题等价于线性方程组求解、算子理论的解释、可行性检验）进行提炼，稍微加以讨论。而如果要继续深究，提炼出的每一模块又会牵涉到更多细节问题，如对方程组求解还涉及对矩阵进行预处理

（preconditioning）来加速、具体的算法终止条件设计（termination criteria）等等，在这就不一一展开了。感觉很多东西真的是靠着时间的积累，对于像 [2] 这样的经典文章，每次间隔一段时间积累了新的知识后再重新去阅读，又会发现之前觉得的某些不起眼的论述其实是如此的精妙。也只有大牛们的经验见识才能写出 [2] 这样对于某个知识点如此详尽解释的经典之作吧。

参考文献

[1] O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. 169(3), 1042–1068, 2016.

[2] 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, 2011.

[3] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S.: OSQP: an operator splitting solver for quadratic programs. Mathematical Programming Computation, 2020

[4] G. Banjac, P. Goulart, B. Stellato, and S. Boyd. Infeasibility detection in the alternating direction method of multipliers for convex optimization. Journal of Optimization Theory and Applications, 183(2), 490-519, 2019.

[5] http://web.stanford.edu/class/ee364b/lectures.html

[6] Eckstein, J., Bertsekas, D.P.: On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. Mathematical Programming, 55(1), 293–318, 1992.

[7] H. H. Bauschke and P. L. Combettes. Convex Analysis and Monotone Operator Theory in Hilbert Spaces. Springer, 1st edition, 2011.

[8] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004).

附录

1. https://mp.weixin.qq.com/s?__biz=Mzg2MTA0NzA0Mw==&mid=2247495356&idx=1&sn=a5e7a6b22e2a1e919ae906709b22f194&chksm=ce1f-bed0f96837c66b9d029ce3d11791b891064388e5500877d14114cd9ac1d38d-b7460c5e03&scene=126&sessionid=1592828955&key=202-

fa661b52134ec373b82a1c6a31c8549864b31ba16e27dd7cbd-d5e8ae09b9d8d6346b1f0c9bf1103b4133cf4d7fcdf38d6f73e011f1b-b9084e2b662662c0971465584b274f3e6340cab5e78b15c464&ascene=1&uin=NzM0MTc0OTA3&devicetype=Windows+10+x64&version=62090070&lang=zh_CN&exportkey=AdFDcK8EhCVahEPALh3fock%3D&pass_ticket=0zZ6hoz1KhkQzp%2F8A832TKrBAnFC5nOt-N9vfFuqDtt6XvMROiRxpt%2FOM5%2FG%2BbU3n

2. https://www.zhihu.com/question/309568920/answer/580226096
3. https://www.zhihu.com/question/36566112
4. https://zhuanlan.zhihu.com/p/150605754

相关文章推荐

来自优化小学生对基于ADMM的求解大规模锥优化问题分布式算法的介绍：Splitting Conic Solver(SCS)，以及其中用到的Homogeneous Self-Dual Embedding。
点击蓝字标题，即可阅读《优化 | 大规模锥优化之Splitting Conic Solver(SCS)》

版块招聘信息：
关于我们：『运筹OR帷幄』团队掠影
请将简历发送至：
operations_r@163.com
欢迎加入我们这个大家庭！

本文福利

可以在 **本公众号后台** 回复关键词：**“ 网盘 ”**获取大量由我平台编辑精心整理的学习资料，如果觉得有用， 请勿吝啬你的留言和赞哦！

—— 完 ——

文章须知

文章作者：陈宇文

责任编辑：苏向阳

审核编辑：阿春

微信编辑：玖蓁

本文由『运筹OR帷幄』原创发布

如需转载请在公众号后台获取转载须知

Read more

喜欢此内容的人还喜欢

优化 | 对偶问题下的几类优化方法

运筹OR帷幄

金税四期真来了，加强税务监控不能乱避税，企业和会计将有多大的影响?

会计联盟超级群

小个子女生怎么穿才显高？看看这四个日系时尚博主怎么搭配

荷兰微生活