凸优化和单调变分不等式的收缩算法

第八讲: 基于增广 Lagrange 乘子法的 PPA 收缩算法

Augmented Lagrangian-based PPA contraction methods for constrained convex optimization

南京大学数学系　　何炳生

hebma@nju.edu.cn

1 线性约束凸优化等价的单调变分不等式

考虑一般的线性等式约束凸优化问题

$$\min\{\theta(x) \mid Ax = b,\ x \in \mathcal{X}\}, \tag{1.1}$$

其中 $\theta(x)$ 是凸函数, $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X}$ 是 $\Re^n$ 中的闭凸集. 凸优化问题 (1.1) 的 Lagrange 函数是定义在 $\mathcal{X} \times \Re^m$ 上的

$$L(x, \lambda) = \theta(x) - \lambda^T(Ax - b).$$

记 $\Lambda = \Re^m$, 设 (x^*, λ^*) 是 Lagrange 函数的一个鞍点, 便有

$$L_{\lambda\in\Lambda}(x^*, \lambda) \le L(x^*, \lambda^*) \le L_{x\in\mathcal{X}}(x, \lambda^*).$$

求 Lagrange 函数的一个鞍点等价于求 (x^*, λ^*) 使其满足:

$$\begin{cases} x^* \in \mathcal{X}, \quad \theta(x) - \theta(x^*) + (x - x^*)^T(-A^T\lambda^*) \ge 0, & \forall\, x \in \mathcal{X}, \\ \lambda^* \in \Lambda, \quad (\lambda - \lambda^*)^T(Ax^* - b) \ge 0, & \forall\, \lambda \in \Lambda, \end{cases} \tag{1.2}$$

通过定义

$$u = \begin{pmatrix} x \\ \lambda \end{pmatrix}, \quad F(u) = \begin{pmatrix} -A^T\lambda \\ Ax - b \end{pmatrix} \quad \text{和} \quad \Omega = \mathcal{X} \times \Lambda, \tag{1.3}$$

求解 (1.1) 相当于求解变分不等式

$$\mathsf{VI}(\Omega, F) \quad u^* \in \Omega, \quad \theta(x) - \theta(x^*) + (u - u^*)^T F(u^*) \geq 0, \quad \forall\, u \in \Omega. \tag{1.4}$$

注意到, (1.3) 中的仿射算子 F 是单调的.

对给定的常数 $s > 0$, 定义在 $\Omega = \mathcal{X} \times \Re^m$ 上的

$$\mathcal{L}_A(x, \lambda) = \theta(x) - \lambda^T(Ax - b) + \frac{1}{2s}\|Ax - b\|^2$$

是 (1.1) 的增广 Lagrange 函数. 增广Lagrange 乘子法 [1, 8, 11] 是求等式约束优化问题的很有效的方法之一, 对此, [10] 中第 17 章有详细论述.

求上述等式约束凸优化问题的经典增广 Lagrange 乘子法 (Augmented Lagrangian Method) 的 k-步迭代, 是从给定的 λ^k 出发, 求

$$x^{k+1} = \mathsf{Argmin}\{\mathcal{L}_A(x, \lambda^k) \,|\, x \in \mathcal{X}\},$$

然后以

$$\lambda^{k+1} = \lambda^k - \frac{1}{s}(Ax^{k+1} - b)$$

得到新的迭代点.

设 Ω^* 是 VI(Ω, F) 的解集合. 为考虑收缩算法需要, 我们记

$$\Lambda^* = \{\lambda^* \in \Lambda \,|\, (x^*, \lambda^*) \in \Omega^*\}.$$

如果用经典的增广 Lagrange 方法求解问题 (1.1), 对任意给定的 $b \in \Re^m$, 我们假设子问题

$$\boxed{\min \{\theta(x) + \tfrac{1}{2s}\|Ax - b\|^2 \,|\, x \in \mathcal{X}\}}$$

是容易求解的. 否则的话, 采用基于松弛增广 Lagrange 的收缩方法. 这时, 需要假设对任意给定的 $a \in \Re^n$, 问题

$$\boxed{\min \{\theta(x) + \tfrac{r}{2}\|x - a\|^2 \,|\, x \in \mathcal{X}\}}$$

的解是容易求得的.

2 收缩意义下的增广 Lagrange 乘子法

我们称定义在 $\Omega = \mathcal{X} \times \Re^m$ 上的

$$\mathcal{L}_A(x, \lambda) = \theta(x) - \lambda^T(Ax - b) + \frac{1}{2s}\|Ax - b\|^2$$

为等式约束的问题 $\min\{\theta(x) \mid Ax = b,\ x \in \mathcal{X}\}$ 的增广 Lagrange 函数. 以下是增广 Lagrange 乘子法的框架，它从给定的 λ^k 开始.

等式约束问题的增广 Lagrange 乘子法框架 对给定的 λ^k, 先求

$$\tilde{x}^k = \text{Argmin}\{\theta(x) + \tfrac{1}{2s}\|(Ax - b) - s\lambda^k\|^2 \mid x \in \mathcal{X}\}, \tag{2.1a}$$

然后令

$$\tilde{\lambda}^k = \lambda^k - \tfrac{1}{s}(A\tilde{x}^k - b). \tag{2.1b}$$

经典的增广 Lagrange 乘子法以 $\lambda^{k+1} = \tilde{\lambda}^k$ 完成一次迭代.

我们用收缩算法的观点来分析增广 Lagrange 乘子法, 改成以

$$\lambda^{k+1} = \lambda^k - \gamma(\lambda^k - \tilde{\lambda}^k), \quad \gamma \in (0, 2) \tag{2.2}$$

生成新的迭代点. 由 (2.1a) 生成的 $\tilde{x}^k \in \mathcal{X}$ 满足

$$\theta(x) - \theta(\tilde{x}^k) + (x - \tilde{x}^k)^T\{-A^T\lambda^k + \tfrac{1}{s}A^T(A\tilde{x}^k - b)\} \geq 0, \ \ \forall x \in \mathcal{X}.$$

将 (2.1b) 中的 $\tilde{\lambda}^k = \lambda^k - \frac{1}{s}(A\tilde{x}^k - b)$ 代入上式就有

$$\tilde{x}^k \in \mathcal{X}, \quad \theta(x) - \theta(\tilde{x}^k) + (x - \tilde{x}^k)^T\left(-A^T\tilde{\lambda}^k\right) \geq 0, \ \ \forall x \in \mathcal{X}. \tag{2.3}$$

把 (2.3) 和 (2.1b) 组合在一起, 就是 $\tilde{u}^k = (\tilde{x}^k, \tilde{\lambda}^k) \in \Omega$, 并且

$$\theta(x) - \theta(\tilde{x}^k) + \begin{pmatrix} x - \tilde{x}^k \\ \lambda - \tilde{\lambda}^k \end{pmatrix}^T \left\{ \begin{pmatrix} -A^T\tilde{\lambda}^k \\ A\tilde{x}^k - b \end{pmatrix} + \begin{pmatrix} 0 \\ s(\tilde{\lambda}^k - \lambda^k) \end{pmatrix} \right\} \geq 0, \ \forall\, u \in \Omega. \tag{2.4}$$

将任意的 $u^* = (x^*, \lambda^*)$ 代入上式中的 $u \in \Omega$, 并利用 $F(u)$ 的表达式就有

$$(\tilde{\lambda}^k - \lambda^*)^T(\lambda^k - \tilde{\lambda}^k) \geq \frac{1}{s}\{(\tilde{u}^k - u^*)^T F(\tilde{u}^k) + \theta(\tilde{x}^k) - \theta(x^*)\}. \tag{2.5}$$

利用 F 的单调性和

$$\theta(\tilde{x}^k) - \theta(x^*) + (\tilde{u}^k - u^*)^T F(u^*) \geq 0,$$

推得 (2.5) 式的右端非负. 所以有

$$(\lambda^k - \lambda^*)^T(\lambda^k - \tilde{\lambda}^k) \geq \|\lambda^k - \tilde{\lambda}^k\|^2, \quad \forall\, \lambda^* \in \Lambda^*. \tag{2.6}$$

用 (2.2) 生成新的迭代点 λ^{k+1}, 在收缩意义下我们通常取 $\gamma \in [1, 2)$, 根据 (2.6) 得到

$$\begin{aligned}
\|\lambda^{k+1} - \lambda^*\|^2 &= \|(\lambda^k - \lambda^*) - \gamma(\lambda^k - \tilde{\lambda}^k)\|^2 \\
&= \|\lambda^k - \lambda^*\|^2 - 2\gamma(\lambda^k - \lambda^*)^T(\lambda^k - \tilde{\lambda}^k) + \gamma^2\|\lambda^k - \tilde{\lambda}^k\|^2 \\
&\leq \|\lambda^k - \lambda^*\|^2 - \gamma(2-\gamma)\|\lambda^k - \tilde{\lambda}^k\|^2.
\end{aligned}$$

The sequence $\{\lambda^k\}$ (dual variable) generated by Augmented Lagrangian Method is Fejér monotone.

上述性质说明, 增广 Lagrange 乘子法是关于对偶变量 λ 的 PPA 算法. 用收缩算法的观点来考虑问题, 迭代式 (2.2) 中的 γ 可以在区间 $(0, 2)$ 中自由选取. 在实际计算中, 我们建议取 $\gamma \in [1.2, 1.8]$.

3 基于增广 Lagrange 乘子法的 PPA 算法

在实际问题中, 精确求解子问题 (2.1a) 往往是花费很大的. 我们还是假设只有形似 $\min\{\theta(x)+\frac{r}{2}\|x-a\|^2 \mid x\in\mathcal{X}\}$ 的问题是容易求解的. 如果对 (2.1a) 中的目标函数加上 Proximal 项 $\frac{r}{2}\|x-x^k\|^2$, 要处理的子问题就成为

$$\min\{\theta(x)+\frac{1}{2s}\|(Ax-b)-s\lambda^k\|^2+\frac{r}{2}\|x-x^k\|^2 \mid x\in\mathcal{X}\}. \tag{3.1}$$

再将 (3.1) 中的 $\frac{1}{2s}\|(Ax-b)-s\lambda^k\|^2$ 在 x^k 处做线性化近似就是

$$\frac{1}{2s}\|(Ax^k-b)-s\lambda^k\|^2+\big(\frac{1}{s}A^T[(Ax^k-b)-s\lambda^k]\big)^T(x-x^k).$$

对 (3.1) 中的二次项线性化以后, 子问题就变成

$$\min\{\theta(x)+\big(\frac{1}{s}A^T[(Ax^k-b)-s\lambda^k]\big)^Tx+\frac{r}{2}\|x-x^k\|^2 \mid x\in\mathcal{X}\}. \tag{3.2}$$

基于增广 Lagrange 乘子法的 PPA 算法, 只对 (3.1) 中的二次函数做线性化处理. 每步迭代从给定的 $u^k=(x^k,\lambda^k)$ 开始, 生成 $\tilde{u}^k\in\Omega$.

对二次函数线性化处理 对给定的 (x^k, λ^k), 先求

$$\tilde{x}^k = \text{Argmin}\{\theta(x) + \big(\frac{1}{s}A^T[(Ax^k - b) - s\lambda^k]\big)^T x + \frac{r}{2}\|x - x^k\|^2 \mid x \in \mathcal{X}\} \quad (3.3a)$$

然后令

$$\tilde{\lambda}^k = \lambda^k - \frac{1}{s}(A\tilde{x}^k - b). \quad (3.3b)$$

我们按统一框架考察生成的预测点 $\tilde{u}^k$. 由 (3.3a) 生成的 $\tilde{x}^k \in \mathcal{X}$ 满足

$$\theta(x) - \theta(\tilde{x}^k) + (x - \tilde{x}^k)^T\{-A^T\lambda^k + \frac{1}{s}A^T(Ax^k - b) + r(\tilde{x}^k - x^k)\} \geq 0, \ \forall x \in \mathcal{X}.$$

将 (3.3b) 中的 $\tilde{\lambda}^k = \lambda^k - \frac{1}{s}(A\tilde{x}^k - b)$ 代入上式就有 $\tilde{x}^k \in \mathcal{X}$, 并对$x \in \mathcal{X}$, 有

$$\theta(x) - \theta(\tilde{x}^k) + (x - \tilde{x}^k)^T\big(-A^T\tilde{\lambda}^k + (rI_n - \frac{1}{s}A^TA)(\tilde{x}^k - x^k)\big) \geq 0, \quad (3.4)$$

把 (3.4) 和 (3.3b) 组合在一起, 就是 $\tilde{u}^k = (\tilde{x}^k, \tilde{\lambda}^k) \in \Omega$, 并对所有的 $(x, \lambda) \in \Omega$,

均有

$$\theta(x)-\theta(\tilde{x}^k)+\begin{pmatrix} x-\tilde{x}^k \\ \lambda-\tilde{\lambda}^k \end{pmatrix}^T\left\{\begin{pmatrix} -A^T\tilde{\lambda}^k \\ A\tilde{x}^k-b \end{pmatrix}+\begin{pmatrix} (rI_n-\frac{1}{s}A^TA)(\tilde{x}^k-x^k) \\ s(\tilde{\lambda}^k-\lambda^k) \end{pmatrix}\right\}\geq 0. \tag{3.5}$$

我们记

$$d(u^k,\tilde{u}^k)=\begin{pmatrix} (rI_n-\frac{1}{s}A^TA)(x^k-\tilde{x}^k) \\ s(\lambda^k-\tilde{\lambda}^k) \end{pmatrix}. \tag{3.6}$$

并利用 $F(u)$ 的表达式, 就有

$$\theta(x)-\theta(\tilde{x}^k)+(u-\tilde{u}^k)^T(F(\tilde{u}^k)-d(u^k,\tilde{u}^k))\geq 0,\ \ \forall\, u\in\Omega. \tag{3.7}$$

将任意的 $u^*=(x^*,\lambda^*)$ 代入上式中的 $(x,\lambda)\in\Omega$, 并

$$(\tilde{u}^k-u^*)^Td(u^k,\tilde{u}^k)\geq\theta(\tilde{x}^k)-\theta(x^*)+(\tilde{u}^k-u^*)^TF(\tilde{u}^k). \tag{3.8}$$

利用 F 的单调性和 $\theta(\tilde{x}^k)-\theta(x^*)+(\tilde{u}^k-u^*)^TF(u^*)\geq 0$, 上式右端非负. 进而得到

$$(u^k-u^*)^Td(u^k,\tilde{u}^k)\geq(u^k-\tilde{u}^k)^Td(u^k,\tilde{u}^k). \tag{3.9}$$

对参数 r, s 的要求

对给定的 $s > 0$, 选取 r 使

$$rs > \|A^T A\|. \tag{3.10}$$

在上述条件下, 矩阵

$$G = \begin{pmatrix} rI_n - \frac{1}{s}A^T A & 0 \\ 0 & sI_m \end{pmatrix}, \tag{3.11}$$

是正定的. 利用 (3.6), 就有

$$d(u^k, \tilde{u}^k) = G(u^k - \tilde{u}^k). \tag{3.12}$$

不等式 (3.9) 就化成

$$(u^k - u^*)^T G(u^k - \tilde{u}^k) \geq \|u^k - \tilde{u}^k\|_G^2. \tag{3.13}$$

直接取 $u^{k+1} = \tilde{u}^k$ 为新的迭代点, 就得到 G-模下的 PPA 算法.

基于增广 Lagrange 乘子法的 PPA 算法

PPA 算法用迭代式

$$u^{k+1} = u^k - \gamma(u^k - \tilde{u}^k), \quad \gamma \in (0, 2) \tag{3.14}$$

产生新的迭代点 u^{k+1}, 我们一般取 $\gamma \in [1, 2)$. 利用 (3.13), 序列 $\{u^k\}$ 满足

$$\|u^{k+1} - u^*\|_G^2 \leq \|u^k - u^*\|_G^2 - \gamma(2 - \gamma)\|u^k - \tilde{u}^k\|_G^2.$$

这是保证基于增广 Lagrange 乘子法的 PPA 算法收敛的关键不等式.

使用这一节的算法, 参数 r, s 需要满足条件 $rs > \|A^T A\|$ (见 (3.10)). 因此, 适合用来处理 $\|A^T A\|$ 容易估算的问题.

4 基于增广 Lagrange 乘子法的收缩算法

这一节的基于增广 Lagrange 乘子法的收缩算法, 同样是只对 (3.1) 中的二次函数做线性化处理. 每步迭代从给定的 $u^k = (x^k, \lambda^k)$ 开始, 还是以 (3.3) 生成 $\tilde{u}^k \in \Omega$. 由上一节的分析, 我们得到

$$(u^k - u^*)^T d(u^k, \tilde{u}^k) \geq (u^k - \tilde{u}^k)^T d(u^k, \tilde{u}^k), \tag{4.1}$$

其中

$$d(u^k, \tilde{u}^k) = \begin{pmatrix} (rI_n - \frac{1}{s}A^TA)(x^k - \tilde{x}^k) \\ s(\lambda^k - \tilde{\lambda}^k) \end{pmatrix} \tag{4.2}$$

我们记

$$H = \begin{pmatrix} rI_n & 0 \\ 0 & sI_m \end{pmatrix}. \tag{4.3}$$

并考虑用

$$M(u^k - \tilde{u}^k) = H^{-1}d(u^k, \tilde{u}^k) \tag{4.4}$$

作为寻查方向. 这时

$$M = \begin{pmatrix} I_n - \frac{1}{rs}A^TA & 0 \\ 0 & I_m \end{pmatrix}. \tag{4.5}$$

我们并不要求 M 正定, 因此不再要求 $rs > \|A^TA\|$, 也不能直接将 $\tilde{u}^k$ 取作新的迭代点. 换句话说, 基于增广 Lagrange 乘子法的收缩算法是一种预测－校正方法, 以 (3.3) 生成的 $\tilde{u}^k \in \Omega$ 只是一个预测点.

生成预测点时对参数 r, s 的要求 对给定的 $s > 0$, 要求 r 满足

$$\|A^T A(x^k - \tilde{x}^k)\| \leq rs\nu\|x^k - \tilde{x}^k\|, \quad \nu \in (0, 1). \tag{4.6}$$

如果条件 $rs > \|A^T A\|$ (见 (3.10)) 满足, 则条件 (4.6) 自然满足. 然而, 这里只要求在计算的每次迭代中验证条件 (4.6) 是否满足.

我们考虑 H 模下的收缩算法. 注意到不等式 (4.1) 可以写成

$$(u^k - u^*)^T HM(u^k - \tilde{u}^k) \geq (u^k - \tilde{u}^k)^T HM(u^k - \tilde{u}^k), \ \forall\, u^* \in \Omega^*. \tag{4.7}$$

Lemma 4.1 对给定的 $u^k = (x^k, \lambda^k)$, 设 $\tilde{u}^k \in \Omega$ 由 (3.3) 生成. 在条件 (4.6) 满足的情况下, 我们有

$$\begin{aligned} &(u^k - \tilde{u}^k)^T HM(u^k - \tilde{u}^k) \\ &\quad \geq \quad \frac{1}{2}\{\|M(u^k - \tilde{u}^k)\|_H^2 + (1 - \nu^2)\|u^k - \tilde{u}^k\|_H^2\}. \end{aligned} \tag{4.8}$$

证明. 我们考察

$$2(u^k-\tilde{u}^k)^T HM(u^k-\tilde{u}^k)-\|d(u^k,\tilde{u}^k)\|_H^2.$$

由于

$$\begin{aligned}&2(u^k-\tilde{u}^k)^T HM(u^k-\tilde{u}^k)-\|M(u^k-\tilde{u}^k)\|_H^2\\ =\quad&\big(2(u^k-\tilde{u}^k)-M(u^k-\tilde{u}^k)\big)^T HM(u^k-\tilde{u}^k).\end{aligned}\tag{4.9}$$

利用 H 和 $d(u^k,\tilde{u}^k)$ 的定义(见 (4.3) 和 (4.4)), 对上式的右端进行处理, 由于

$$M(u^k-\tilde{u}^k)\quad=\quad\begin{pmatrix}(I_n-\frac{1}{rs}A^TA)(x^k-\tilde{x}^k)\\(\lambda^k-\tilde{\lambda}^k)\end{pmatrix}$$

和

$$2(u^k-\tilde{u}^k)-M(u^k-\tilde{u}^k)\quad=\quad\begin{pmatrix}(I_n+\frac{1}{rs}A^TA)(x^k-\tilde{x}^k)\\(\lambda^k-\tilde{\lambda}^k)\end{pmatrix}.$$

将这些代入 (4.9) 的右端就得到

$$\begin{aligned} & 2(u^k-\tilde{u}^k)^T HM(u^k-\tilde{u}^k) - \|M(u^k-\tilde{u}^k)\|_H^2 \\ = \quad & r\big(\|x^k-\tilde{x}^k\|^2 - \frac{1}{r^2s^2}\|A^TA(x^k-\tilde{x}^k)\|^2\big) + s\|\lambda^k-\tilde{\lambda}^k\|^2. \end{aligned}$$

在条件 (4.6) 满足的情况下, 从上式得到

$$\begin{aligned} & 2(u^k-\tilde{u}^k)^T HM(u^k-\tilde{u}^k) - \|M(u^k-\tilde{u}^k)\|_H^2 \\ \geq \quad & (1-\nu^2)r\|x^k-\tilde{x}^k\|^2 + s\|\lambda^k-\tilde{\lambda}^k\|^2. \end{aligned} \tag{4.10}$$

从上式马上得到 (4.8), 引理证明完毕.

我们考虑 H 模下的收缩算法. 先考虑取单位步长的**初等的收缩算法**.

初等的收缩算法 对给定的 u^k 和用 (3.3) 生成的 $\tilde{u}^k$, 用

$$u^{k+1} = u^k - M(u^k-\tilde{u}^k) \tag{4.11}$$

生成新的迭代点. 由矩阵 M 的结构 (4.5), 采用初等收缩算法时 $\lambda^{k+1} = \tilde{\lambda}^k$.

根据迭代公式 (4.11) 并使用 (4.1), 就有

$$\begin{aligned}
&\|u^k - u^*\|_H^2 - \|u^{k+1} - u^*\|_H^2 \\
&= \|u^k - u^*\|_H^2 - \|(u^k - u^*) - M(u^k - \tilde{u}^k)\|_H^2 \\
&= 2(u^k - u^*)^T HM(u^k - \tilde{u}^k) - \|M(u^k - \tilde{u}^k)\|_H^2 \\
&\geq 2(u^k - \tilde{u}^k)^T HM(u^k - \tilde{u}^k) - \|M(u^k - \tilde{u}^k)\|_H^2. \qquad (4.12)
\end{aligned}$$

以引理 4.1 的结论 (4.8) 代入 (4.12), 就有下面的定理:

Theorem 4.1 *Let the condition* (4.6) *be satisfied. Then the sequence* $\{u^k = (x^k, \lambda^k)\}$ *generated by the elementary contraction method satisfies*

$$\|u^{k+1} - u^*\|_H^2 \leq \|u^k - u^*\|_H^2 - (1 - \nu^2)\|u^k - \tilde{u}^k\|_H^2. \qquad (4.13)$$

定理 4.1 中的不等式 (4.13) 是保证初等收缩算法收敛的关键不等式.

再考虑通过计算步长确定下一个迭代点的**一般的收缩算法**.

一般的收缩算法 对给定的 u^k 和 (3.3) 生成的 $\tilde{u}^k$, 用

$$u(\alpha) = u^k - \alpha M(u^k - \tilde{u}^k) \qquad (4.14)$$

产生依赖于步长 α 的迭代点. 同样, 对任意给定的 $u^* \in \Omega^*$, 我们定义

$$\vartheta(\alpha) := \|u^k - u^*\|_H^2 - \|u(\alpha) - u^*\|_H^2 \tag{4.15}$$

和

$$q(\alpha) = 2\alpha(u^k - \tilde{u}^k)^T HM(u^k - \tilde{u}^k) - \alpha^2 \|M(u^k - \tilde{u}^k)\|_H^2. \tag{4.16}$$

利用 (4.14) 和 (4.15) 中 $\vartheta(\alpha)$ 的定义以及 (4.1), 可以证明

$$\vartheta(\alpha) \geq q(\alpha) \tag{4.17}$$

同样, 注意到 (4.16) 中的 $q(\alpha)$ 是 α 的二次函数, 它在

$$\alpha^* = \frac{(u^k - \tilde{u}^k)^T HM(u^k - \tilde{u}^k)}{\|M(u^k - \tilde{u}^k)\|_H^2} \tag{4.18}$$

时取得极大值. 从 (4.8) 式知, 在条件 (4.6) 满足的情况下, $\alpha_k^* \geq 1/2$. 在实际计算中, 我们取

$$u^{k+1} = u^k - \gamma \alpha_k^* M(u^k - \tilde{u}^k), \tag{4.19}$$

为新的迭代点, 其中 $\gamma \in [1,2)$ 称为松弛因子. 由 (4.15) 和 (4.17), 有

$$\begin{aligned} \|u^{k+1}-u^*\|_H^2 &\le \|u^k-u^*\|_H^2 - q(\gamma\alpha_k^*) \\ &\le \|u^k-u^*\|_H^2 - \gamma(2-\gamma)\alpha_k^*(u^k-\tilde{u}^k)^T HM(u^k-\tilde{u}^k). \end{aligned} \tag{4.20}$$

根据上式, 由 (4.8) 和 $\alpha_k^* \ge 1/2$, 得到

Theorem 4.2 *The sequence* $\{u^k=(x^k,\lambda^k)\}$ *generated by the general contraction method satisfies*

$$\|u^{k+1}-u^*\|_H^2 \le \|u^k-u^*\|_H^2 - \frac{\gamma(2-\gamma)(1-\nu^2)}{4}\|u^k-\tilde{u}^k\|_H^2. \tag{4.21}$$

定理 4.2 中的不等式 (4.21) 是保证一般收缩算法收敛的关键不等式.

此外, 由 (4.18), (4.20) 和 $\alpha_k^* > 1/2$, 会有

$$\|u^{k+1}-u^*\|_H^2 \le \|u^k-u^*\|_H^2 - \frac{\gamma(2-\gamma)}{4}\|M(u^k-\tilde{u}^k)\|_H^2.$$

5 信息技术领域优化问题中的应用

我们用这一讲 §3 的方法求解第四讲中提到的算例.

5.1 相关性矩阵 (Correlation Matrix) 校正中的应用

对给定的对称矩阵 C, 求 F-模下与 C 距离最近的相关性矩阵, 其数学表达式是

$$\min\Big\{\frac{1}{2}\|X-C\|_F^2 \mid \mathrm{diag}(X)=e,\ X\in S_+^n\Big\}, \tag{5.1}$$

其中 e 表示每个分量都为 1 的 n-维向量, S_+^n 表示 $n\times n$ 正半定锥的集合. 问题 (5.1) 是形如 (1.1) 的等式约束凸优化问题, 其中 $\|A^TA\|=1$.

我们用 $z\in\Re^n$ 作为等式约束 $\mathrm{diag}(X)=e$ 的 Lagrange 乘子.

PPA 算法求解问题(5.1)

对给定的 (X^k,z^k), 用 (3.3) 产生$(\tilde{X}^k,\tilde{z}^k)$:

1. Finding $\tilde{X}^k$ which is the solution of the following minimization problem

$$\min\{\frac{1}{2}\|X - C\|_F^2 + \frac{r}{2}\|X - [X^k + \frac{1}{r}\,\mathrm{diag}(z^{k+\frac{1}{2}})]\|_F^2 | X \in S_+^n\}, \quad (5.2)$$

where

$$z^{k+\frac{1}{2}} = z^k - \frac{1}{s}(\mathrm{diag}(X^k) - e).$$

2. Setting

$$\tilde{z}^k = z^k - \frac{1}{s}(\mathrm{diag}(\tilde{X}^k) - e). \quad (5.3)$$

子问题 (5.2) 求解的具体做法: 化为等价问题

$$\min\{\frac{1}{2}\|X - \frac{1}{1+r}[rX^k + \mathrm{diag}(z^{k+\frac{1}{2}}) + C]\|_F^2 | X \in S_+^n\}.$$

因此我们只要考虑如何求解

$$\tilde{X}^k = \mathrm{Argmin}\{\frac{1}{2}\|X - A\|_F^2 \mid X \in S_+^n\}. \quad (5.4)$$

实际上, 将对称矩阵 A 做标准特征值– 特征向量分解

$$A = V\Lambda V^T, \qquad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n). \quad (5.5)$$

通过

$$\tilde{X} = V\tilde{\Lambda}V^T, \qquad \tilde{\Lambda} = \mathrm{diag}(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_n),$$

就得到 $\tilde{X}$, 其中

$$\tilde{\lambda}_j = \max\{0, \lambda_j\}.$$

因此, 每次迭代的主要工作量是做 (5.5) 中的特征值 (特征向量) 分解 .

数值试验 为生成试验例子, 只要给定对称矩阵 C.

C=rand(n,n); C=(C'+C)-ones(n,n) + eye(n)

这样的矩阵 C 的对角元在 $(0, 2)$ 之间, 非对角元在 $(-1, 1)$ 之间。

Code 7.a. Matlab code for Creating the test examples

```
clear;  close all;
n = 1000;      tol=1e-5;     r=2.0;   s=1.05/r;    gamma=1.5;
rand('state',0);  C=rand(n,n);   C=(C'+C)-ones(n,n) + eye(n);
```

Code 7.1. Matlab code of ALM based classical PPA

```
%%%     Classical PPA for calibrating correlation matrix                %(1)
function PPAC(n,C,r,s,tol)                                              %(2)
X=eye(n);       y=zeros(n,1);     tic;       %%  The initial iterate  %(3)
stopc=1;                          k=0;                                  %(4)
while (stopc>tol && k<=100)          %% Beginning of an Iteration       %(5)
 if mod(k,20)==0 fprintf(' k=%4d   epsm=%9.3e  \n',k,stopc); end;       %(6)
    X0=X;     y0=y;     k=k+1;                                          %(7)
    ya=y0 - (diag(X0)-ones(n,1))/s;                                     %(8)
    A=(X0*r + C + diag(ya))/(1+r);                                      %(9)
    [V,D]=eig(A);   D=max(0,D);  XT=(V*D)*V';      EX=X0-XT;          %(10)
    yt=y0 - (diag(XT)-ones(n,1))/s;                EY=y0-yt           %(11)
    ex=max(max(abs(EX)));  ey=max(abs(EY));   stopc=max(ex,ey);        %(12)
    X= XT;                 y=yt;                                        %(13)
end;                                    % End of an Iteration          %(14)
toc;                       TB = max(abs(diag(X-eye(n))));               %(15)
fprintf(' k=%4d  epsm=%9.3e   max|X_jj - 1|=%8.5f \n',k,stopc,TB); %%
```

做 (5.5) 中的特征值 (特征向量) 分解, 在上述程序中的第 (10) 行用 Matlab 中的语句 [V,D]=eig(A) 实现的, 这是一个计算量大概 $9n^3$ 的运算.

将 Classical PPA 改成 Extended PPA, 只要将第 (13) 行改一下.

Code 7.2 Matlab code of ALM based extended PPA

```
%%%     Extended PPA for calibrating correlation matrix                %(1)
function PPAE(n,C,r,s,tol,gamma)                                       %(2)
X=eye(n);       y=zeros(n,1);    tic;       %%  The initial iterate  %(3)
stopc=1;                        k=0;                                   %(4)
while (stopc>tol && k<=100)          %% Beginning of an Iteration     %(5)
 if mod(k,20)==0 fprintf(' k=%4d   epsm=%9.3e  \n',k,stopc); end;      %(6)
    X0=X;     y0=y;     k=k+1;                                         %(7)
    ya=y0 - (diag(X0)-ones(n,1))/s;                                    %(8)
    A=(X0*r + C + diag(ya))/(1+r);                                     %(9)
    [V,D]=eig(A);   D=max(0,D);  XT=(V*D)*V';      EX=X0-XT;          %(10)
    yt=y0 - (diag(XT)-ones(n,1))/s;                EY=y0-yt           %(11)
    ex=max(max(abs(EX)));  ey=max(abs(EY));   stopc=max(ex,ey);        %(12)
    X=X0 - EX*gamma;           y=y0- EY*gamma;                         %(13)
end;                                        % End of an Iteration    %(14)
toc;                          TB = max(abs(diag(X-eye(n))));           %(15)
fprintf(' k=%4d  epsm=%9.3e   max|X_jj - 1|=%8.5f \n',k,stopc,TB); %%
```

两个不同的方法, 程序都很简单, 用不了几行. 两个程序不同的地方仅仅是第 (13) 行有些差别, 取 $\gamma = 1.5$ 的方法效果却有明显的提高。

矩阵校正问题 (5.1)–使用 Matlab 中的 eig 子程序

$n \times n$ Matrix	Classical PPA		Extended PPA	
$n =$	No. It	CPU Sec.	No. It	CPU Sec.
100	29	0.34	21	0.25
200	32	2.26	24	1.68
500	38	20.70	26	14.04
800	41	86.75	29	61.28
1000	47	182.82	30	117.08
2000	65	1696.50	41	1076.04

The extended PPA converges faster than the classical PPA.

$$\frac{\text{It. No. of Extended PPA}}{\text{It. No. of Classical PPA}} \approx 65\%.$$

♣ 关于相关系数矩阵校正的程序在附件的 Codes-07 的文件夹“矩阵校正”中. 只要运行 demo.m, 输入 n 就可以了. 其中的 ALM_PPAC.m 和 ALM_PPAE.m 分别是 Classical PPA 和 Extended PPA 的子程序.

确实, 用这一讲介绍的 PPA 方法求解相关矩阵矩阵校正问题, 每步迭代的主要计算工作量是对一个对称矩阵用 Matlab 中的标准子程序做 [V,D]=eig(A). 如果改用 Kim TOH 写的 mexeig 做 [V,D]=mexeig(A), 计算时间大为节省.

矩阵校正问题 (5.1)–特征值分解使用 mexeig

$n \times n$ Matrix	Classical PPA		Extended PPA	
$n =$	No. It	CPU Sec.	No. It	CPU Sec.
100	29	0.14	21	0.10
200	32	0.57	24	0.42
500	38	5.64	26	3.92
800	41	20.05	29	14.31
1000	47	41.82	30	27.10
2000	65	401.94	41	255.37

5.2 矩阵完整化方面的应用

设 M 是一个 $m \times n$ 矩阵, Π 是矩阵元素的指标集.

$$\Pi = \{(ij) \,|\, i \in \{1, 2, \ldots, m\},\ j \in \{1, 2, \ldots, n\}\}.$$

矩阵完整化问题是由部分信息获取全部信息. 在适当(实际问题具备的)条件下, 大多数不完整信息的低秩矩阵可以通过求解松弛问题

$$\min\{\|X\|_* \mid X_{ij} = M_{ij},\ (ij) \in \Pi\} \tag{5.6}$$

得到精确恢复. 其中 $\|X\|_*$ 表示矩阵 X 的奇异值的和. 通常称为矩阵 X 的核模—Nuclear Norm.

问题 (5.6) 是形如 (1.1) 的等式约束凸优化问题, 其中 $\|A^T A\| = 1$. 我们将 (5.6) 的等式约束记为 $X_\Pi = M_\Pi$，并用 $Z \in \Re^{m \times n}$ 作为相应的 Lagrange 乘子.

PPA 算法求解问题 (5.6)

对给定的 (X^k, Z^k), 用 (3.3) 产生$(\tilde{X}^k, \tilde{Z}^k)$:

1. Finding $\tilde{X}^k$ which is the solution of the following linear variational inequality

$$\min \left\{ \|X\|_* + \frac{r}{2} \left\| X - \left[X^k + \frac{1}{r} Z_\Pi^{k+\frac{1}{2}} \right] \right\|_F^2 \right\} \tag{5.7}$$

where

$$Z_\Pi^{k+\frac{1}{2}} = Z_\Pi^k - \frac{1}{s}(X_\Pi^k - M_\Pi).$$

2. Updating $\tilde{Z}^k$ by

$$\tilde{Z}_\Pi^k = Z_\Pi^k - \frac{1}{s}(\tilde{X}_\Pi^k - M_\Pi).$$

子问题 (5.7) 求解的具体做法: 只需考虑如何求解

$$\tilde{X}^k = \text{Argmin} \left\{ \frac{1}{r} \|X\|_* + \frac{1}{2} \|X - A\|_F^2 \right\}. \tag{5.8}$$

我们将 A 做 SVD 分解

$$A = U \Lambda V^T,$$

并记

$$\tilde{X} = U \tilde{\Lambda} V^T. \tag{5.9}$$

注意到

$$\frac{1}{r}\|\tilde{X}\|_* + \frac{1}{2}\|\tilde{X} - A\|_F^2 = \frac{1}{r}\|\tilde{\Lambda}\|_* + \frac{1}{2}\|\tilde{\Lambda} - \Lambda\|_F^2.$$

通过

$$\tilde{\lambda}_j = \lambda_j - \min(\lambda_j, \frac{1}{r}), \tag{5.10}$$

就能得到对角矩阵 $\tilde{\Lambda}$ 的对角元 $\tilde{\lambda}_j$, 代入 (5.9) 就得到 (5.8) 的解 $\tilde{X}^k$. 同样, 每次迭代的主要工作量是做一个矩阵的 SVD 分解.

如果以 λ 和 $\tilde{\lambda}$ 分别表示对角矩阵 Λ 和 $\tilde{\Lambda}$ 的对角元生成的向量, 由于 λ 是非负向量, 关系式 (5.10) 也可以写成 Shrinkage 的形式

$$\tilde{\lambda} = \lambda - P_{B_\infty^{1/r}}[\lambda],$$

其中 $B_\infty^{1/r}$ 是无穷模下半径为 $1/r$ 的“圆”（一个立方体）.

结论: 将线性约束的凸优化问题转换成单调变分不等式, 再用这一讲 §3 中介绍的基于增广 Lagrange 乘子法的 PPA 算法求解, 每次迭代中要求解的子问题, 数值代数中都有确定的成熟的方法求解 !

数值试验

数值试验例子取自[2]

一个秩为 ra 的 $n \times n$ 的自由度是 $d_{ra} := ra(2n - ra)$.

生成试验问题：

- 先用高斯同分布 (Gaussian i.i.d) 独立生成两个 $n \times ra$ 的矩阵 M_1 和 M_2, 然后令 $M = M_1 M_2^T$, 则 $n \times n$ 矩阵 M 的秩为 ra.
- 随机选定 M 的 m 个元素作为已知元素, 这些元素的下标集为 Π.

计算结果：

- 矩阵完整化问题的难度与比率 m/d_{ra} 和 m/n^2 都有关系.
- 分别用 Classical PPA 和 Extended PPA ($\gamma = 1.5$) 进行计算.
- 正定矩阵 G (see (3.11)) 中的参数 $r,\ s$ 分别取 $rs = 1.01$ 和 $r = 0.005$.
- 停机准则采用相对误差 $\|X_\Pi^k - M_\Pi\|_F / \|M_\Pi\|_F \leq 10^{-4}$.

我们用 $\gamma = 1.5$ 的Extended PPA 求解, 由于 KKT 条件 Primal 部分((3.5) 的上半

部分) 的不满足量是

$$(rI_n - \frac{1}{s}A^TA)(\tilde{x}^k - x^k).$$

对这个具体问题, $\|A^TA\| = 1$ 并且 $rs \approx 1$, 所以我们检查

$$\text{KKT-Violation} := \max\{r \max_{ij} |X^k_{ij} - \tilde{X}^k_{ij}|\}$$

并在计算结果中列出.

计算时间依赖于用什么 SVD 子程序. 我们分别列出用 Matlab 中的标准 SVD 以及 PROPACK-SVD 的不同计算效果.

♣ 矩阵完整化的程序在附件的 Codes-07 的文件夹"矩阵完整化"中. 只要运行 demo.m 就可以了. 要对不同情形试验, 只要在 demo.m 中用 % 做适当选择. 其中的 PPAC.m 和 PPAE.m 分别是 Classical PPA 和 Extended PPA 的子程序.

Code 7.b. Creating the test examples of Matrix Completion

```
%%  Creating the test examples of the matrix Completion problem      %(1)
clear all;  clc                                                      %(2)
maxIt=100;         tol = 1e-4;                                       %(3)
r=0.005;           s=1.01/r;         gamma=1.5;                      %(4)
  n=200;             ra = 10;          oversampling = 5;             %(5)
% n=1000;   ra=100;      oversampling = 3; %%  Iteration No. 31      %(6)
% n=1000;   ra=50;       oversampling = 4; %%  Iteration No. 36      %(7)
% n=1000;   ra=10;       oversampling = 6; %%  Iteration No. 78      %(8)
%% Generating the test problem                                       %(9)
rs = randseed;         randn('state',rs);                           %(10)
M=randn(n,ra)*randn(ra,n);          %% The matrix will be completed %(11)
df =ra*(n*2-ra);                    %% The freedom of the matrix    %(12)
mo=oversampling;                                                     %(13)
m =min(mo*df,round(.99*n*n));        %% No. of the known elements   %(14)
Omega= randsample(n^2,m);            %% Define the subset Omega     %(15)
fprintf('Matrix: n=%4d  Rank(M)=%3d  Oversampling=%2d \n',n,ra,mo);%(16)
```

我们只对ALM-based extended PPA 给出程序, 若用ALM-based classical PPA,
只要把下面的第(16) 行改成 `X = XT` 第(17) 行改成 `Y(Omega) = YT(Omega)`

Code 7.3. ALM-based extended PPA for Matrix Completion Problem

```
function PPAE(n,r,s,M,Omega,maxIt,tol,gamma)    %  Ititial Process %%(1)
X=zeros(n);     Y=zeros(n);      YT=zeros(n);                          %(2)
  nM0=norm(M(Omega),'fro');    eps=1;  VioKKT=1;  k=0;   tic;          %(3)
%% Minimum nuclear norm solution by PPA method                        %(4)
while  (eps > tol && k<= maxIt)                                       %(5)
 if mod(k,5)==0                                                       %(6)
 fprintf('It=%3d |X-M|/|M|=%9.2e VioKKT=%9.2e\n',k,eps,VioKKT); end;%(7)
  k=k+1;      X0=X;      Y0=Y;                                       %(8)
  Y(Omega)=Y0(Omega)-(X0(Omega)-M(Omega))/s;                         %(9)
  A = X0 + Y/r;                      [U,D,V]=svd(A,0);              %(10)
  D=D-eye(n)/r;     D=max(D,0);     XT=(U*D)*V';        EX=X0-XT;  %(11)
  DXM=XT(Omega)-M(Omega);           eps = norm(DXM,'fro')/nM0;      %(12)
  YT(Omega)=Y0(Omega)-(XT(Omega)-M(Omega))/s;          EY=Y0-YT;  %(13)
  VioKKT =  max(max(abs(EX)))*r;                                    %(14)
   if (eps <= tol)   gamma=1;   end;                                %(15)
  X = X0 - EX*gamma;                                                %(16)
  Y(Omega) = Y0(Omega) - EY(Omega)*gamma;                           %(17)
end;                                                                %(18)
 fprintf('It=%3d  |X-M|/|M|=%9.2e  Vi0KKT=%9.2e \n',k,eps,VioKKT); %(19)
   RelEr=norm((X-M),'fro')/norm(M,'fro');   toc;                    %(20)
 fprintf(' Relative error = %9.2e  Rank(X)=%3d \n',RelEr,rank(X)); %(21)
 fprintf(' Violation of KKT Condition = %9.2e  \n',VioKKT);         %(22)
```

矩阵完整化问题: 用 Matlab-SVD 求解结果

1000×1000 matrix	ALM-based Classical PPA				ALM-based Extended PPA			
(rank, $\frac{m}{d_r}$, $\frac{m}{n^2}$)	No. It.	CPU Sec.	Relative error	KKT-Violation	No. It.	CPU Sec.	Relative error	KKT-Violation
(10, 6, 0.12)	85	989.84	9.62E-5	4.34E-6	77	899.40	9.32E-5	2.79E-6
(50, 4, 0.39)	43	491.35	1.46E-4	1.85E-5	37	425.24	1.21E-4	1.43E-5
(100, 3, 0.58)	36	394.68	1.72E-4	2.78E-5	31	363.04	1.55E-4	3.26E-5

♣ 用 Matlab 中的 SVD, 做一次 SVD 的花费很大, 总耗时与迭代次数成比例.

♣ 用 Extended PPA 与 Classical PPA 相比, 几乎不加额外的负担, 效率还是提高 10% 以上. 因此, 在任何情况下, 我们都提倡用 Extended PPA.

♣ 用 PROPACK [9] 中的 SVD, 快许多, 总耗时主要与问题性质有关. 由于我们主要对迭代次数感兴趣, 我们仅仅报道用 Matlab 做 SVD 的结果, 也附上所需要的子程序. 总耗时主要与问题性质有关.

♣ 注意到, Cai, Candès and Shen [2] 的方法对这三个例子的迭代次数分别是 117, 114 和129 (See the first three examples in Table 5.1 of [2], pp. 1974). 原则上, 每次迭代的主要工作量都是一次 SVD 分解. [2] 中采用不完全分解技术, 节省了总的运行时间.

一点说明 从第六讲到第八讲, 我们主要讲一类 Customized PPA 方法, 求解

$$\min\{\theta(x) \mid Ax = b, x \in \mathcal{X}\} \tag{5.11}$$

这样一类凸优化问题. 用来计算的例子, 都有应用背景. 用 Customized PPA 方法求解, 有很不错的数值结果. 主要原因是这些问题中的线性约束的矩阵 A 的性态比较好 (注意到 $\mathrm{diag}(X) = e$ 和 $X_\Pi = M_\Pi$ 表示成 $Ax = b$ 时, A 只是一个投影矩阵), 加上这些问题中

$$\min\left\{\theta(x) + \frac{r}{2}\|x - a\|^2 \mid x \in \mathcal{X}\right\}$$

这样的子问题是有成熟方法求解的.

但是, 正如 R. Fletcher 在他的著作 Practical Methods of Optimization 中说到,

Indeed there is no general agreement on the best approach and much research is still to be done.

世界上没有一个方法是对所有的问题都是最好的, 我们不能指望用这几讲介绍的方法去求解形如 (5.11) 的一般凸优化问题, 特别是人为构造的难题. 好在一些有应用背景的问题, 常常有它的特殊结构, 简单的方法有时也能凑效.

References

[1] D. P. Bertsekas. Constrained Optimization and Lagrange Multiplier Methods. Academic Press, 1982.

[2] J. F. Cai, E. J. Candès and Z. W. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim., **20**, 1956-1982, 2010.

[3] B.S. He, A class of projection and contraction methods for monotone variational inequalities, Applied Mathematics and Optimization, **35**, 69-76, 1997.

[4] B.S. He, Inexact implicit methods for monotone general variational inequalities, Math. Program. Series A **86**, 199-217, 1999.

[5] B. S. He, X. L. Fu and Z.K. Jiang, Proximal point algorithm using a linear proximal term, JOTA, **141**, 209-239, 2009.

[6] B.S He and L-Z Liao, Improvements of some projection methods for monotone nonlinear variational inequalities, JOTA, **112**, 111-128, 2002

[7] B. S. He and M.-H. Xu, A general framework of contraction methods for monotone variational inequalities, Pacific J. Optimization, **4**, 195-212, 2008.

[8] M. R. Hestenes, Multiplier and gradient methods. Journal of Optimization Theory and Applications,**4**, 302-320, 1969.

[9] R. M. Larsen, Propack − software for large and sparse svd calculation, http://sun.stanford.edu/rmunk/PROPACK/, 2005.

[10] J. Nocedal and S.J. Wright, Numerical Optimization, Springer Verlag, 1999.

[11] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, Optimization. Academic Press, 1969.