7. 凸优化算法

朱天宇

如果要优化的函数是凸的，一般采用 下降算法 。

1 等式约束

1.1 两条路径

$$\begin{aligned} \min \quad & f(\mathsf{x}) \\ s.t. \quad & \mathsf{Ax} = \mathsf{b} \end{aligned} \tag{1}$$

这里 $f(\mathsf{x})$ 是凸的，矩阵 $\mathsf{A} \in \mathbb{R}^{p\times n}, \mathrm{rank}\mathsf{A} = p < n$。

其 KKT 系统为：

$$\begin{cases} \mathsf{Ax}^\star - \mathsf{b} = 0 \\ \nabla f(\mathsf{x}^\star) + \mathsf{A}^\top \nu^\star = 0 \end{cases} \tag{2}$$

这里 ν 是拉格朗日乘子。由于没有不等式约束，所以 KKT 系统中的 1/3/4 式始终满足。剩余的是 2/5 式。上面的式子被称为原问题可行性方程，下面的式子被称为对偶问题可行性方程。这个系统关于 ν 是线性的，关于 $\mathsf{x}^\star$ 是非线性的，因此 KKT 系统是非线性的。

求解这个系统，一般通过牛顿法来处理。原问题 $\min f(\mathsf{x})$ 的牛顿法和求解 KKT 系统的牛顿法是差不多的。原问题的牛顿法通常是在 $f(\mathsf{x})$ 处做 2 阶泰勒展开，变成二次函数，获得当前点处的牛顿方向。展开后的二次函数的导数显然是线性的；而对于 KKT 非线性系统的牛顿迭代，实际上是做一阶近似。对于原问题的二阶近似，会得到一个梯度和 Hessian；对于 KKT 非线性系统的一阶近似，会得到一个 Jacobian 矩阵。

为了确定问题1的牛顿步，在 x 附近作二阶泰勒展开，得到一个关于 s 的式子：

$$\begin{aligned} \min \quad & \hat{f}(\mathsf{x}+\mathsf{s}) = f(\mathsf{x}) + \nabla f(\mathsf{x})^\top + \frac{1}{2}\mathsf{s}^\top \nabla^2 f(\mathsf{x})\mathsf{s} \\ s.t. \quad & \mathsf{A}(\mathsf{x}+\mathsf{s}) = \mathsf{b} \end{aligned} \tag{3}$$

对于这个式子，假设最优解是 $\delta_{\mathsf{x}_{nt}}$。则通过3的 KKT 条件，就可以知道存在关联的最优对偶变量 $w \in \mathbb{R}^p$，满足

$$\begin{bmatrix} \nabla^2 f(\mathsf{x}) & \mathsf{A}^\top \\ \mathsf{A}^\top & 0 \end{bmatrix} \begin{bmatrix} \delta_{\mathsf{x}_{nt}} \\ w \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathsf{x}) \\ 0 \end{bmatrix} \tag{4}$$

从而可以解出牛顿步长。

而如果不对原问题1进行二阶泰勒展开，而是直接处理其 KKT 条件，替换2中的 $\mathsf{x}^\star, \nu^\star$ 为 $\mathsf{x}+\delta_{\mathsf{x}_{nt}}, w$，并且将第二个式子中的梯度项替换为 x 附近的线性近似，就可以获得

$$\begin{cases} \mathsf{A}(\mathsf{x}+\delta_{\mathsf{x}_{nt}}) - \mathsf{b} = 0 \\ \nabla f(\mathsf{x}+\delta_{\mathsf{x}_{nt}}) + \mathsf{A}^\top w \approx \nabla f(\mathsf{x}) + \nabla^2 f(\mathsf{x})\delta_{\mathsf{x}_{nt}} + \mathsf{A}^\top w = 0 \end{cases} \tag{5}$$

可以发现5与4是一模一样的：

原问题1 $\overset{\text{二阶泰勒展开}}{\Rightarrow}$ 二次规划 (QP) 问题3$\Rightarrow$ 问题3的 KKT4$\Rightarrow$ 牛顿迭代步 δ

原问题1$\Rightarrow$ 原问题1的 KKT2 $\overset{\text{一阶近似}}{\Rightarrow}$2的一阶近似展开 $\Rightarrow$ 牛顿迭代步 δ

这两条路径中，第一条路经可以用于无约束问题；

1.2 停机准则

定义下降量 κ

$$\kappa(\mathsf{x}) = \sqrt{\delta_{\mathsf{x}_{nt}^\top} \nabla^2 f(\mathsf{x}) \delta_{\mathsf{x}_{nt}}}$$

因为这个值等于

$$\frac{\mathsf{d}}{\mathsf{d}\alpha} f(\mathsf{x}+\alpha\delta_{\mathsf{x}_{nt}}) \mid_{\alpha=0} = \nabla f(\mathsf{x})^\top \delta_{\mathsf{x}_{nt}} = -\kappa(\mathsf{x})^2$$

如果 $\kappa(\mathsf{x})$ 比较小，则算法停止。

1.3 算法流程 (初始点可行的时候)

Algorithm 1 初始点为可行点的等式约束凸问题的牛顿迭代方法

初始点为 $x \in \mathrm{dom} f$，满足 $Ax = b$，定义 tolerance$\epsilon > 0$
while 1 **do**
  计算牛顿步、下降方向 $\delta_{x_{nt}}$ 和下降量$\kappa(x)$
  if $\kappa^2/2 \leq \epsilon$ **then**
    算法终止
  end if
  通过 backtarcking line search 以确定步长α
  更新：$x \leftarrow x + \alpha\delta_{x_{nt}}$
end while

值得注意的是，这个过程中，初始点必须是可行的。如果起始点不一定可行，要用外点法相关的方法。

1.4 起始点不可行的情况

当 x 是可行点的时候，其天然满足 $Ax = b$ 的约束。起始点是不可行点的时候，不能假定这个条件满足。

这时候，就要寻找一个 δ_x 使得 $A(x + \delta_x) = b$。

记录满足约束的点为 $x^\star = x + \delta_x, Ax^\star = b$，同样将原问题1的 KKT 系统2中的 $x^\star$ 替换掉，并且进行一阶展开，就得到

$$\begin{cases} A(x + \delta_x) - b = 0 \\ \nabla f(x + \delta_x) + A^\top \mu \approx \nabla f(x) + \nabla^2 f(x)\delta_x + A^\top \mu = 0 \end{cases} \tag{6}$$

同样可以得到一个 KKT 系统：

$$\begin{bmatrix} \nabla^2 f(x) & A^\top \\ A^\top & 0 \end{bmatrix} \begin{bmatrix} \delta_x \\ \mu \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ Ax - b \end{bmatrix} \tag{7}$$

注意方程组7和方程组4的区别：等号右边下方，4是 0，而7是 $Ax - b$。

也就是说，当当前点是不可行点的时候，迭代时只需要将这一项进行修改就行了。

解释 将 KKT 系统表达为函数 $r(x^\star, \nu^\star) = 0, r : \mathbb{R}^n \times \mathbb{R}^p \mapsto \mathbb{R}^n \times \mathbb{R}^p$，其包含两个分部：primal 和 dual 的。

$$r(x^\star, \nu^\star) = (r_{dual}(x^\star, \nu^\star), r_{primal}(x^\star, \nu^\star))$$

在这里，r_{dual}, r_{pri} 分别称为对偶残差和原残差。

$$r_{dual}(x, \nu) = \nabla f(x) + A^\top \nu$$

$$r_{primal}(x, \nu) = Ax - b$$

在当前点 $y = (x, \nu)$ 的附近，作 r 的一阶近似展开，就有

$$r(y + \delta_y) \approx \hat{r}(y + \delta_y) = r(y) + J[r(y)]\delta_y$$

这里，J 是 Jacobian 矩阵，因为 KKT 条件当中已经存在一阶信息了，再次作一阶展开，就会出现 Jacobian 矩阵。表示在 y 点处对 r 进行求导。$J[r(y)]$ 的维度是 $(n + p) \times (n + p)$。

定义 $\delta_{y_{pd}}$ 为 $\hat{r}(y + \delta_y) = 0$ 的时候的牛顿迭代步，由于 $\hat{r} = 0$，可知

$$J[r(y)]\delta_{y_{pd}} = -r(y)$$

注意到，$\delta_{y_{pd}} = (\delta_{x_{pd}}, \delta_{y_{pd}})$，同时给出了 primal 和 dual 的步长。

改写一下7式：

$$\begin{bmatrix} \nabla^2 f(x) & A^\top \\ A^\top & 0 \end{bmatrix} \begin{bmatrix} \delta_{x_{pd}} \\ \delta_{\nu_{pd}} \end{bmatrix} = -\begin{bmatrix} r_{dual} \\ r_{pri} \end{bmatrix} = -\begin{bmatrix} \nabla f(x) + A^\top \nu \\ Ax - b \end{bmatrix} \tag{8}$$

这里，$\nu + \delta_{\nu_{pd}} = \mu$

1.5 算法流程 (初始点不可行的时候)

Algorithm 2 初始点为不可行点的等式约束凸问题的牛顿迭代方法

初始点为 $\mathrm{x} \in \mathrm{dom} f$，但是 $\mathrm{Ax} \neq \mathrm{b}$，定义 tolerance$\epsilon > 0$，定义 $\tau \in (0, 1/2), \gamma \in (0, 1)$
while $\mathrm{Ax} \neq \mathrm{b}$ 或者 $\|r(\mathrm{x}, \nu)\|_2 > \epsilon$ **do**
  计算 primal 和 dual 的牛顿步 $\delta_{\mathrm{x}_{nt}}, \delta_{\nu_{nt}}$
  在 $\|r\|_2$ 上做 backtracking line search
  $\alpha \leftarrow 1$
  while $\|r(\mathrm{x} + \alpha\delta_{\mathrm{x}_{nt}}, \nu + \alpha\delta_{\nu_{nt}})\|_2 > (1 - \tau\alpha)\|r(\mathrm{x}, \nu)\|_2$ **do**
    $\alpha \leftarrow \gamma\alpha$
  end while
  $\mathrm{x} \leftarrow \mathrm{x} + \alpha\delta_{\mathrm{x}_{nt}}$
  $\nu \leftarrow \nu + \alpha\delta_{\nu_{nt}}$
end while

2 不等式约束

含不等式约束的凸优化问题的形式为：

$$\begin{aligned} \min \quad & f_0(\mathrm{x}) \\ s.t. \quad & f_i(\mathrm{x}) \leq 0, \quad 1, ..., m \\ & \mathrm{Ax} = \mathrm{b} \end{aligned} \tag{9}$$

其 KKT 系统为

$$\begin{cases} f_i(\mathrm{x}^\star) \leq 0, i = 1, ..., m \\ \mathrm{Ax}^\star = b \\ \lambda_i^\star \geq 0 \\ \lambda_i^\star f_i(\mathrm{x}^\star) = 0, i = 1, ..., m \\ \nabla f_0(\mathrm{x}^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla f_i(\mathrm{x}^\star) + \mathrm{A}^\top \nu^\star = 0 \end{cases} \tag{10}$$

这个系统含有不等式约束，无法像之前一样处理。对于这种情况，有 2 种处理方法：障碍函数 、primal-dual 内点算法

2.1 使用障碍函数

定义 示性函数 ：

$$I_-(u) = \begin{cases} 0, & u \leq 0 \\ \infty, & u > 0 \end{cases}$$

通过示性函数，可以将原问题改写称为仅含有等式约束的问题：

$$\begin{aligned} \min \quad & f_0(\mathrm{x}) + \sum_{i=1}^{m} I_-(f_i(\mathrm{x})) \\ s.t. \quad & \mathrm{Ax} = \mathrm{b} \end{aligned} \tag{11}$$

示性函数不可微不连续，性质不好，所以引入 障碍函数 来进行**逼近**。比如使用

$$\hat{I}_-(u) = -(1/t)\log(-u)$$

这张图显示了障碍函数和示性函数的关系：t 越大，近似程度越高。

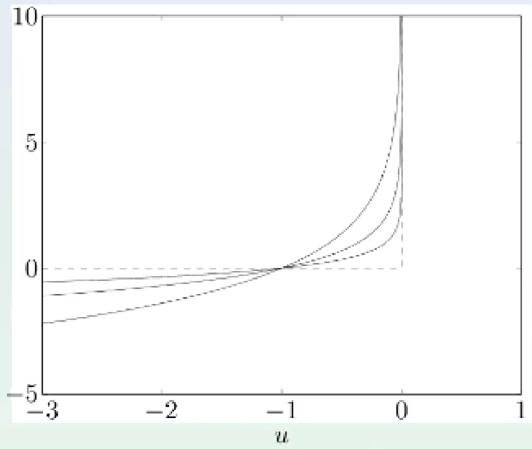

Figure: The dashed lines show the function $I_-(u)$, and the solid curves show $\hat{I}_-(u) = -(1/t)\log(-u)$, for $t = 0.5, 1, 2$. The curve for $t = 2$ gives the best approximation.

代入障碍函数后，方程11就变为：

$$
\begin{aligned}
\min \quad & f_0(\mathsf{x}) + \sum_{i=1}^{m} -(1/t)\log(-f_i(\mathsf{x})) \\
s.t. \quad & \mathsf{Ax} = \mathsf{b}
\end{aligned}
\tag{12}
$$

这是一个等式约束的凸优化问题。

这里，障碍函数被称为 对数障碍函数 ，其表达式、一阶导、二阶导分别是：

$$
\begin{aligned}
\phi(\mathsf{x}) &= -\sum_{i=1}^{m} \log(-f_i(\mathsf{x})) \\
\nabla\phi(\mathsf{x}) &= \sum_{i=1}^{m} \frac{1}{-f_i(\mathsf{x})}\nabla f_i(\mathsf{x}) \\
\nabla^2\phi(\mathsf{x}) &= \sum_{i=1}^{m} \frac{1}{f_i(\mathsf{x})^2}\nabla f_i(\mathsf{x})\nabla f_i(\mathsf{x})^\top + \sum_{i=1}^{m} \frac{1}{-f_i(\mathsf{x})}\nabla^2 f_i(\mathsf{x})
\end{aligned}
\tag{13}
$$

方程12可以通过牛顿法来求解。可以将目标函数乘以 t，则变为

$$
\begin{aligned}
\min \quad & t f_0(\mathsf{x}) + \phi(\mathsf{x}) \\
s.t. \quad & \mathsf{Ax} = \mathsf{b}
\end{aligned}
\tag{14}
$$

中心路径 central path 对于每一个 t，都能对应到一个比较好的 x，因此，定义 $\mathsf{x}^\star(t) = \arg\min_{\mathsf{x}}\{t f_0(\mathsf{x}) + \phi(\mathsf{x})\ s.t.\ \mathsf{Ax} = \mathsf{b}\}$。实际算法中需要仔细调节 t 的数值。

$\mathsf{x}^\star(t)$ 和原问题9不是一回事，是问题14的最优解。问题9的 中心路径 ，被定义为由 $\mathsf{x}^\star(t)$ 构成的，按照 t 的值进行排序的有序点集合 $\{\mathsf{x}^\star(t) | t > 0\}$。另外值得注意的是，当 $t \to \infty$ 的时候，障碍项趋向于不起作用。

在中心路径上的点，始终保证 $\mathsf{x}^\star(t)$ 是可行点。也就是其满足约束

$$
\mathsf{Ax}^\star(t) = \mathsf{b}, \quad f_i(\mathsf{x}^\star(t)) < 0, \quad i = 1, ..., m
$$

并且 $\exists \hat{\nu} \in \mathbb{R}^p$，满足问题14的**最优性条件**：

$$
\begin{aligned}
0 &= t\nabla f_0(\mathsf{x}^\star(t)) + \nabla\phi(\mathsf{x}^\star(t)) + \mathsf{A}^\top\hat{\nu} \\
&= t\nabla f_0(\mathsf{x}^\star(t)) + \sum_{i=1}^{m} \frac{1}{-f_i(\mathsf{x}^\star(t))}\nabla f_i(\mathsf{x}^\star(t)) + \mathsf{A}^\top\hat{\nu}
\end{aligned}
\tag{15}
$$

定义 $\lambda^\star(t) = -\frac{1}{t f_i(\mathsf{x}^\star(t))} > 0, \nu^\star(t) = \frac{\hat{\nu}}{t}$，将之带入式子15中，可以得到

$$
\nabla f_0(\mathsf{x}^\star(t)) + \sum_{i=1}^{m} \lambda_i^\star(t)\nabla f_i(\mathsf{x}^\star(t)) + \mathsf{A}^\top\nu^\star(t) = 0 \tag{16}
$$

可以发现这个式子和原问题9的 KKT 条件10的最后一项是一样的。区别仅仅在于有无 (t)。而这个式子16是原问题的9的拉格朗日函数

$$L(\mathsf{x},\lambda,\nu)=f_0(\mathsf{x})+\sum_{i=1}^{m}\lambda_i f_i(\mathsf{x})+\nu^\top(\mathsf{A}\mathsf{x}-\mathsf{b})$$

的导数。因此，$(\lambda^\star(t),\nu^\star(t))$ 是一对可行的对偶乘子。

所以可以讨论原问题14的**对偶函数**

$$\begin{aligned}g(\lambda^\star(t),\nu^\star(t))&=\min_{\mathsf{x}} L(\mathsf{x},\lambda^\star(t),\nu^\star(t))\\&=f_0(\mathsf{x}^\star(t))+\sum_{i=1}^{m}\lambda_i^\star(t)f_i(\mathsf{x}^\star(t))+\nu^\star(t)^\top(\mathsf{A}\mathsf{x}^\star(t)-\mathsf{b})\\&=f_0(\mathsf{x}^\star(t))-m/t\end{aligned}$$

关于最后一行的化简。回顾定义 $\lambda^\star(t)=-\frac{1}{tf_i(\mathsf{x}^\star(t))}$ 以及 $\mathsf{A}\mathsf{x}^\star(t)-\mathsf{b}=0$。

发现原问题和对偶问题之间存在一个值为 m/t 的 gap。因此，$f_0(\mathsf{x}^\star(t))-v^\star\le m/t$。这样可以说明 $t\to\infty$ 的时候，$\mathsf{x}^\star(t)\to\mathsf{x}^\star$。

从 KKT 出发的理解　假设 x 是问题14的唯一解。那么就必然 $\exists\lambda,\nu$ 使得：

$$\begin{cases}f_i(\mathsf{x})\le 0, i=1,...,m\\\mathsf{A}\mathsf{x}=b\\\lambda_i\ge 0\\-\lambda_i f_i(\mathsf{x})=1/t, i=1,...,m\\\nabla f_0(\mathsf{x})+\sum_{i=1}^{m}\lambda_i\nabla f_i(\mathsf{x})+\mathsf{A}^\top\nu=0\end{cases}\tag{17}$$

和10的 KKT 系统唯一的区别就是互补充实条件 $\lambda_i f_i(\mathsf{x})$ 等号右边的值从 0 变成了 $1/t$，对于一个较大的 t，$\mathsf{x}^\star(t),\lambda^\star(t),\nu^\star(t)$ 接近于 KKT 条件中的最优值。

2.1.1　障碍函数法算法流程

Algorithm 3 障碍函数法

1: 选择可行点 $\mathsf{x}, t\leftarrow t_0>0,\gamma>1,\epsilon>0$
2: **while** 1 **do**
3: 　通过对方程14求最小化，获取 $\mathsf{x}^\star(t)$
4: 　$\mathsf{x}\leftarrow\mathsf{x}^\star(t)$
5: 　**if** $m/t<\epsilon$ **then**
6: 　　算法终止
7: 　**end if**
8: 　增加 t 的值：$t\leftarrow\gamma t$
9: **end while**

获取 $\mathsf{x}^\star(t)$ 的步骤被称为外部迭代，$\mathsf{x}^\star(t)$ 无需精确的计算；选择 γ 是一个 trade-off：外部迭代和内部迭代的次数。t_0 的选择也是一个 trade-off：如果 t_0 过大，那么首次外部迭代需要多轮；t_0 过小，则算法需要额外的外部迭代；

2.2　直接修改 KKT 系统