数值分析

范数

- **什么是范数?**

我们知道距离的定义是一个宽泛的概念，只要满足非负、自反、三角不等式就可以称之为距离。范数是一种强化了的距离概念，它在定义上比距离多了一条数乘的运算法则。有时候为了便于理解，我们可以把范数当作距离来理解。

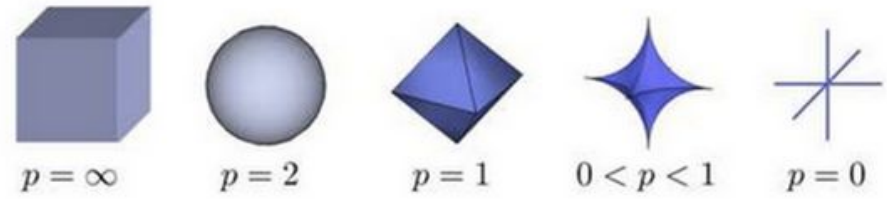
在数学上，范数包括向量范数和矩阵范数，向量范数表征向量空间中向量的大小，矩阵范数表征矩阵引起变化的大小。一种非严密的解释就是，对应向量范数，向量空间中的向量都是有大小的，这个大小如何度量，就是用范数来度量的，不同的范数都可以来度量这个大小，就好比米和尺都可以来度量远近一样；对于矩阵范数，在线性代数中，我们知道，通过运算AX=B，可以将向量X变化为B，矩阵范数就是来度量这个变化大小的。

- **几种范数的定义和含义**

LP范数:

$$Lp = \sqrt[p]{\sum_{1}^{n} x_i^p}\ ,\ x = (x_1, x_2, \cdots, x_n)$$

根据P 的变化，范数也有着不同的变化，一个经典的有关P范数的变化图



向量范数

`L0` 范数

当P=0时，也就是 `L0` 范数，由上面可知，`L0` 范数并不是一个真正的范数，它主要被用来度量向量中非零元素的个数。用上面的L-P定义可以得到的L-0的定义为:

$$||\mathbf{x}||_0 = \sqrt[0]{\sum_{i=1}^{n} x_i^0}$$

在实际应用中，由于`L0`范数本身不容易有一个好的数学表示形式，给出上面问题的形式化表示是一个很难的问题，故被人认为是一个`NP`难问题。所以在实际情况中，`L0`的最优问题会被放宽到`L1`或`L2`下的最优化。

`L1` 范数

`L1`范数是我们经常见到的一种范数，它的定义如下:

$$||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$$

表示向量x中非零元素的绝对值之和。

`L1`范数有很多的名字，例如我们熟悉的曼哈顿距离、最小绝对误差等。使用L1范数可以度量两个向量间的差异，如绝对误差和（Sum of Absolute Difference）:

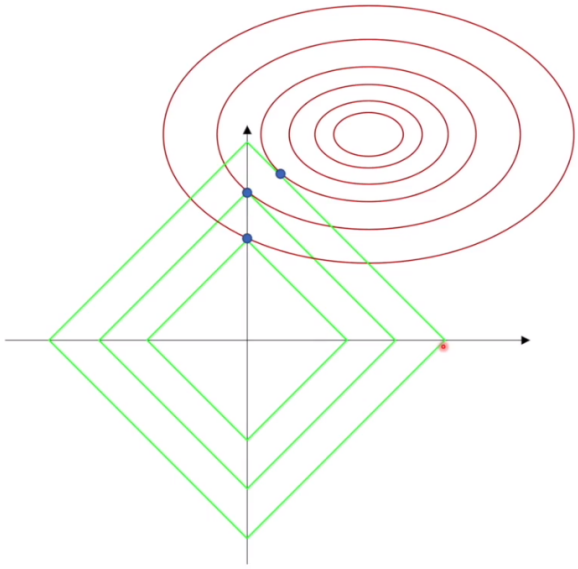
$$SAD(x_1, x_2) = \sum_{i}^{n} |x_{1i} - x_{2i}|$$

对于`L1`范数，它的优化问题如下:

$$min||\mathbf{x}||_1$$
$$s.t.\ Ax = b$$

由于L1范数的天然性质，对L1优化的解是一个稀疏解，因此L1范数也被叫做稀疏规则算子。通过L1可以实现特征的稀疏，去掉一些没有信息的特征，例如在对用户的电影爱好做分类的时候，用户有100个特征，可能只有十几个特征是对分类有用的，大部分特征如身高体重等可能都是无用的，利用L1范数就可以过滤掉。

- 从几何角度来解释L1正则化的稀疏作用



L1 正则化，带来的结果是，他在某一个坐标轴上的取值为零，而只在某一个轴上有数值，相当于把特征之间的关系去耦合了，通过此性质我可就可以对特征进行分类处理，把为零的特征过滤掉只保留想要的特征

L2 范数

L2范数是我们最常见最常用的范数了，我们用的最多的度量距离欧氏距离就是一种L2范数，它的定义如下：

$$||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

表示向量元素的平方和再开平方。

像L1范数一样，L2也可以度量两个向量间的差异，如平方差和（Sum of Squared Difference）：

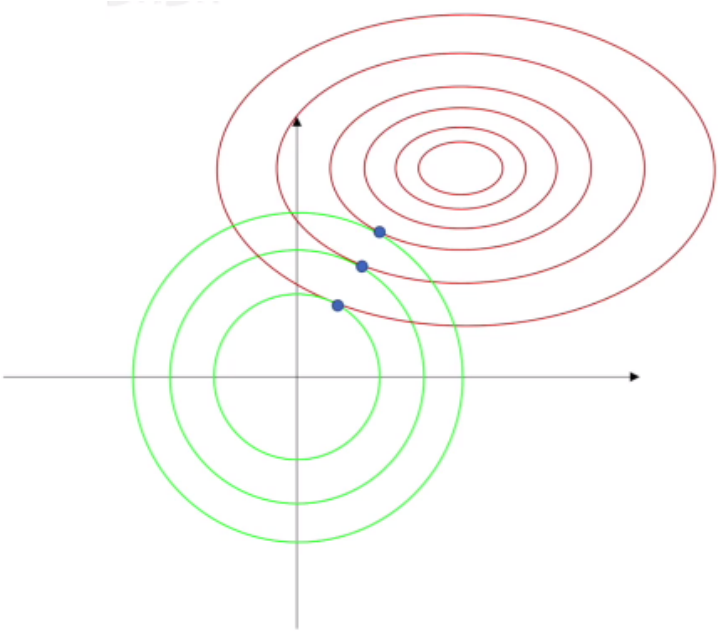
$$SSD(x_1, x_2) = \sum_{i=1}^{n} (x_{1i} - x_{2i})^2$$

对于L2范数，它的优化问题如下：

$$min||x||_2$$
$$s.t.\ Ax = b$$

L2范数通常会被用来做优化目标函数的正则化项，防止模型为了迎合训练集而过于复杂造成过拟合的情况，从而提高模型的泛化能力。

- 从几何角度来解释 L2 正则化的防止过拟合的作用



二维平面下`L2`正则化的函数图形是个圆（绝对值的平方和，是个圆），与方形相比，被磨去了棱角。因为不太可能出现多数w都为0的情况，这就是为什么`L2`正则化不具有稀疏性的原因。

拟合过程中通常都倾向于让权值尽可能小，最后构造一个所有参数都比较小的模型。因为一般认为参数值小的模型比较简单，能适应不同的数据集，也在一定程度上避免了过拟合现象。可以设想一下对于一个线性回归方程，若参数很大，那么只要数据偏移一点点，就会对结果造成很大的影响；但如果参数足够小，数据偏移得多一点也不会对结果造成什么影响，专业一点的说法是**抗扰动能力强**，而`L2`正则化正好可以获得值很小的参数。

L-∞ 范数

当$P=\infty$时，也就是L_∞范数，它主要被用来度量向量元素的最大值，与`L0`一样，通常情况下表示为

$$||\mathbf{x}||_\infty = max(|x_i|)$$

举例

一范数二范数也是用来度量一个整体，比如两个个班的人比较高度，你可以用班里面最高的人（无穷范数）去比较，也可以用班里所有人的身高总和比较（一范数），也可以求平均（类似二范数）。

总结

L-0范数：用来统计向量中非零元素的个数。
L-1范数：向量中所有元素的绝对值之和。可用于优化中去除没有取值的信息，又称稀疏规则算子。
L-2范数：典型应用——欧式距离。可用于优化正则化项，避免过拟合。
L-∞范数：计算向量中的最大值。

矩阵范数

```
A=[2, 3, -5, -7;
   4, 6, 8, -4;
   6,-11,-3,16];
```

1. 矩阵的1-范数（列模）,矩阵的每一列上的元素绝对值先求和，再从中取个最大的，（列和最大）；即矩阵A的1-范数为：27
2. 矩阵的2-范数（谱模）,其中 为的特征值；矩阵的最大特征值开平方根。
3. 矩阵的无穷范数（行模）,矩阵的每一行上的元素绝对值先求和，再从中取个最大的，（行和最大）

总结

1. 矩阵的核范数：矩阵的奇异值（将矩阵`svd`分解）之和，这个范数可以用来低秩表示（因为最小化核范数，相当于最小化矩阵的秩——低秩）；
2. 矩阵的`L0`范数：矩阵的非0元素的个数，通常用它来表示稀疏，`L0`范数越小0元素越多，也就越稀疏。
3. 矩阵的`L1`范数：矩阵中的每个元素绝对值之和，它是`L0`范数的最优凸近似，因此它也可以近似表示稀疏；
4. 矩阵的`F`范数：矩阵的各个元素平方之和再开平方根，它通常也叫做矩阵的`L2`范数，它有点在它是一个凸函数，可以求导求解，易于计算；
5. 矩阵的`L21`范数：矩阵先以每一列为单位，求每一列的F范数（也可认为是向量的2范数），然后再将得到的结果求`L1`范数（也可认为是向量的1范数），很容易看出它是介于`L1`和`L2`之间的一种范数

增广拉格朗日法

拉格朗日函数

- 对于优化问题
  $\min\ f(x)$
  $s.t.\ Ax=b$
- 把约束乘以一个系数放在目标函数上，得到拉格朗日函数为
  $L(x,v)=f(x)+v^T(Ax-b)$

增广拉格朗日函数

- Augmented Lagrangian Method进一步将约束的二范数平方放在了目标函数上，得到的增广拉格朗日函数为如下形式
  $L_c(x,v)=f(x)+v^T(Ax-b)+\frac{c}{2}\|Ax-b\|_2^2$
- 以上L_c也是如下问题的拉格朗日函数
  $\min\ f(x)+\frac{c}{2}\|Ax-b\|_2^2$
  $s.t.\ Ax=b$

这个问题与原问题有着相同的最优解，增加2范数不会影响解的结果

增广拉格朗日法

- Augmented Lagrangian Method, `ALM` 通过交替优化原变量与对偶变量求解问题
  $x^{k+1}=\arg\min_x L_c(x,v^k)$
  $v^{k+1}=v^k+c(Ax^{k+1}-b)$
  其中$c=1$或是递增的序列
- **性质**：1）当$v=v^\star$，则$\forall c>0$，$x^\star=\arg\min_x L_c(x,v^\star)$
  2）当$c\to\infty$，则$\forall v$，$x^\star=\arg\min_x L_c(x,v)$

交替方向乘子法--`ADMM`

方法的引出

- 使用上面讲的增广拉格朗日法求解
  $\min f(x)+g(x)$
  这里$f(x),g(x)$都是凸函数，此时目标函数有两个，整体优化可能不太方便，现在想要交替优化两项，考虑将这个优化问题写为
  $\min\ f(x)+g(z)$
  $s.t.\ x=z$
- 以上问题的增广拉格朗日函数为
  $L_c(x,z,v)=f(x)+g(z)+v^T(x-z)+0.5c\|x-z\|_2^2$
- 使用增广拉格朗日函数交替优化原变量x,z，和对偶变量v，交替优化步骤如下
  1）$\{x^{k+1},z^{k+1}\}=\arg\min_{x,z} f(x)+g(z)+v^T(x-z)+0.5c\|x-z\|_2^2$
  2）$v^{k+1}=v^k+c(x^{k+1}-z^{k+1})$
- 在上面步骤1）中，优化了两个变量，现在把这一步运用分块坐标轮换法，进一步地交替优化x,z，拆解为两步，并且将增广拉格朗日函数后两项配方法合并
  $x^{k+1|t+1}=\arg\min_x f(x)+\frac{c}{2}\|x-z^{k+1|t}+\frac{v^k}{c}\|_2^2$
  $z^{k+1|t+1}=\arg\min_z g(z)+\frac{c}{2}\|z-x^{k+1|t+1}-\frac{v^k}{c}\|_2^2$

这两步多次交替迭代，然后再迭代一次对偶变量v，这种方法即是大名鼎鼎的**交替方向乘子法**(Alternating Direction Method of Multipliers, `ADMM`)。

ADMM举例

ADMM算法一般用于解决如下的凸优化问题:

$$min f(x)+g(y)$$

$$s.t. Ax+By=c$$

其中，$x \in R^n$ 为目标函数 $f(x)$ 的优化变量，$y \in R^m$ 为目标函数 $g(y)$ 的优化变量，$A \in R^{p \times n}$ ，$B \in R^{p \times m}$ ，$c \in R^p$ 。函数 f ，g 是凸函数。

它的增广拉格朗日函数如下:

$$L_\rho(x,y,\lambda)=f(x)+g(y)+\lambda^T(Ax+By-c)+(\rho/2)||Ax+By-c||_2^2, \quad \rho>0$$

其中，λ 称为拉格朗日乘子，ρ 是惩罚参数且 $\rho>0$ 。此时，用ADMM算法进行求解，则过程如下:

$$x^{k+1}:=argminL_\rho(x,y,\lambda)$$
$$y^{k+1}:=argminL_\rho(x,y,\lambda)$$
$$\lambda:=\lambda^k+\rho(Ax^{k+1}+By^{k+1}-c)$$

第一步简化:

通过公式 $2a^Tb+||b||_2^2=||a+b||_2^2-||a||_2^2$ 替换增广拉格朗日函数中的线性项 $\lambda^T(Ax+By-c)$ 和二次项 $\rho||Ax+By-c||_2^2$

$$\lambda^T(Ax+By-c)+\rho||Ax+By-c||_2^2=\rho/2||AX+by-c+\rho/\lambda||_2^2-\rho/2||\lambda/\rho||_2^2$$

于是ADMM求解过程可以简化为如下形式:

$$x^{k+1}:=argmin(f(x)+\rho/2||Ax+By^k-c+\lambda^k/\rho||_2^2)$$
$$y^{k+1}:=argmin(g(y)+\rho/2||Ax^{k+1}+By-c+\lambda^k/\rho||_2^2)$$
$$\lambda:=\lambda^k+\rho(Ax^{k+1}+By^{k+1}-c)$$

第二步简化:

令缩放对偶变量为 $u=(1/\rho)\lambda$，于是ADMM求解过程再次简化为如下形式:

$$x^{k+1}:=argmin(f(x)+\rho/2||Ax+By^k-c+u^k||_2^2)$$
$$y^{k+1}:=argmin(g(y)+\rho/2||Ax^{k+1}+By-c+u^k||_2^2)$$
$$u^{k+1}:=u^k+Ax^{k+1}+By^{k+1}-c$$

- **分布式计算：**从以上三步迭代可以看到：更新x_i, v_i时，与x_{-i}, v_{-i}, f_{-i}无关，即n个下标索引的计算可以分配到n台计算机/线程/进程分别进行计算，只在更新z时，需要汇总各个计算节点的数据，更新z后，再将结果分发到各个计算节点。这也是ADMM的一大优点。
- 总结来说就是，构造增广拉格朗日函数，然后用交替求解计算(固定某些变量，再求解目标变量)，当达到某个阈值时，求解结束

内点法(Interior Method)

简介

内点法是一种处理带约束优化问题的方法，其在线性规划，二次规划，非线性规划等问题上都有着很好的表现。内点法或障碍法是解决线性和非线性凸优化问题的一类算法。通过在目标函数中增加一个障碍项来防止违反不等式约束，该障碍项导致最佳无约束值位于可行空间中。

- 线性规划单纯形法：通过一系列迭代达到最优解，迭代点沿着可行多面体的边界从一个顶点到另一个顶点，直到得到最优解。一般而言单纯形法每次迭代的开销相对内点法来说较小，但所需迭代次数较多。
- 线性规划内点法：同样是通过一系列迭代达到最优解，但其是从多面体内部逐渐收敛到最优解。一般而言内点法每次迭代的开销相对单纯形法来说较大，但所需迭代次数较少。

内点法并不仅仅用于线性规划的求解，值得一提的是内点法的很多思想有着更广泛的应用，例如障碍函数法的思想。线性规划问题的一般形式为

$$\begin{aligned}&\min c^T x\\&\text{s.t. } Ax \preceq b\end{aligned}$$

这里，目标函数为线性函数 $c^T x : \mathbb{R}^n \to \mathbb{R}$，约束条件为 $A_{ij}x_j \leq b_i, i = 1, 2, \ldots, m.$ 矩阵 A 为 $m \times n$ 的满秩矩阵，其中 m 为约束条件的个数， n 为变量的个数。通常约束条件的个数大于变量的个数，所以有 $m > n$.

障碍法(Barrier Method)

1.线性规划问题的等价(近似)表述

这个线性规划问题可以重新表述为计算 $\min f(x)$ ，其中

$$f(x) = c^T x + \sum_{i=1}^{m} I(A_{ij}x_j - b_i)$$

这里，我们使用了一个indicator函数，定义为

$$I(u) = \begin{cases} 0 & \text{if } x \leq 0 \\ \infty & \text{if } x > 0 \end{cases}$$

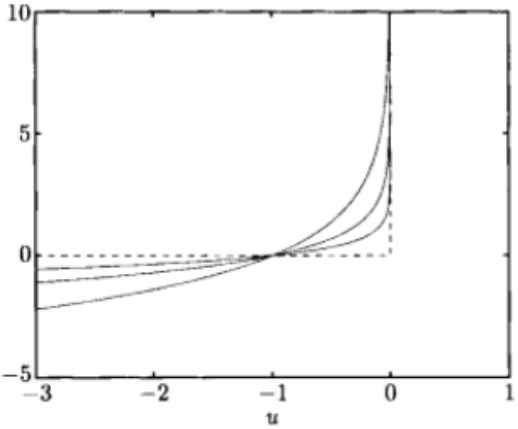


图 11.1　虚线表示函数 $I_-(u)$，实曲线分别表示 $t = 0.5,\ 1,\ 2$ 所对应的 $\widehat{I}_-(u) = -(1/t)\log(-u)$。对应于 $t = 2$ 的曲线给出最好的近似。

引入这个函数的意义在于可以将约束条件直接写入到目标函数里面，这样我们直接求新的函数的极小值就可以了，而不必借助于未知乘子。 但是这里有一个问题，那就是indicator函数存在不可求导的点，因此在求函数极小值的时候我们没法通过普通的微分法来确定函数的极小值。为了规避这个问题，我们可以用一个光滑的函数来近似这个indicator函数。一个不错的选择是用 $I_t(u) = -\frac{1}{t}\log(-u)$ 来代替indicator函数。 $I_t(u)$ 只有在 $u < 0$ 的时候有定义，我们规定当 $u > 0$ 的时候 $I_t(u) = \infty$. 而且参数 $t > 0$ 越大，函数 $I_t(u)$ 就越接近于 $I(u)$. 所以我们可以通过调节 t 的值来调节这个函数的近似程度。使用这个近似的indicator函数，我们新的的目标函数可以写作

$$f(x) = tc^T x - \sum_{i=1}^{m} \log(-A_{ij}x_j + b_i).$$

因为线性函数是凸函数，并且 $I_t(u)$ 也是凸函数，所以 $f(x)$ 是凸函数，因此我们可以很容易用凸优化的经典方法得到该函数的极小值。

2.计算函数的梯度和 `Hessian` 矩阵

为了求函数的极小值，根据微积分的经典结果，只需令函数的梯度等于零，然后计算梯度为零时对应的解 $x^\star$.

函数的梯度为

$$\frac{\partial f}{\partial x_k} = tc_k + \sum_{i=1}^{m} \frac{-A_{ik}}{A_{ij}x_j - b_i}$$

`Hessian` 矩阵为

$$\frac{\partial^2 f}{\partial x_k \partial x_l} = \sum_{i=1}^{m} \frac{A_{ik}A_{il}}{\left(A_{ij}x_j - b_i\right)^2}$$

定义对角型矩阵为

$$D_{ij} = \delta_{ij} \frac{1}{\left(A_{ik}x_k - b_i\right)^2}$$

于是`Hessian`矩阵可以写作

$$H_f = A^T D A$$

因为 D 为正定矩阵，所以 `Hessian` 矩阵至少为半正定矩阵。所以函数 $f(x)$ 是一个凸函数。而且矩阵 D 为可逆矩阵，矩阵 A 满秩，所以 `Hessian` 矩阵为可逆矩阵。于是函数 $f(x)$ 为强凸函数。所以，要计算 $\nabla f = 0$ 的根，我们可以用高效的**牛顿迭代法**。

3.牛顿迭代法

> Newton Method求解近似问题的算法框架
>
> 重复进行:
>
> 1. 中心点步骤:
>   从初始值 x 开始，采用Newton Method在 $Ax = b$ 的约束下极小化 $tf_0 + \phi$，最终确定 $x^*(t)$
> 2. 改进： $x := x^*(t)$
> 3. 停止准则。 如果 $m/t < \epsilon$ 则退出
> 4. 增加 t ： $t := \mu t, \quad \mu > 1$
>
> 根据在中心路径小节中的分析，近似问题的最优解与原始问题的最优解的误差不超过 m/t，所以在求解时，我们从较小的 t 开始，不断增大 t 直至收敛。此外，我们把每次求出的最优解 $x^*(t)$ 当作下一个 $t := \mu t$ 的初值，这样Newton Method会收敛更快，即Step 1.中心点步骤的运行时间可以当作常数对待，而不是一个耗时的迭代循环。

我们现在的目标是计算 $\nabla f = 0$ 的根。因为Hessian矩阵可逆，所以我们可以用牛顿迭代法求解。牛顿迭代法为

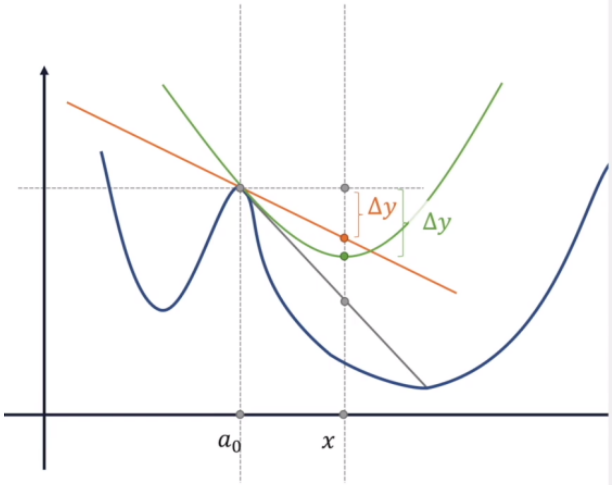
$$x^{(n+1)} = x^{(n)} - H^{-1}\nabla f$$

在这个迭代过程中，参数 t 为固定的。每对应一个 t，我们都可以得到一个解 $x_t^\star$. 如果我们扫描参数 t，我们就可以得到一系列的解。 其中最大的 t 的对应的解应该最精确。

一个更好的算法是选取一个比较小的初始参数 t_0 ， 求出这个参数对应的解 $x_{t_0}^\star$. 然后增加 t ，用之前得到的解 $x_{t_0}^\star$ 来初始化当前 t 所对应的牛顿迭代法的试解。这样算出来的解应该比直接计算 t 所对应的解更加精确。这样逐步迭代从小 t_0 到大 t 可以得到一系列的解，最后得到的解 $x_t^\star$ 称作是淬火解。这个解应该可以满足我们的精度需求。

牛顿法的补充

一维变量情况



$f(x)$泰勒展开保留到二阶导：

$$f(x) = J(a_0) + J'(a_0)(x - a_0) + \frac{1}{2}J''(a_0)(x - a_0)^2$$

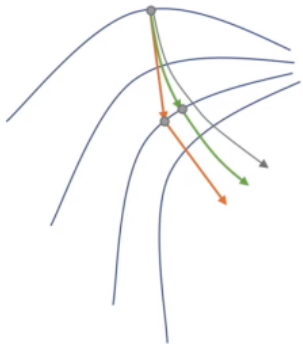
$f(x)$求极值，令$f'(x) = 0$

$$f'(x) = J'(a_0) + J''(a_0)(x - a_0) = 0$$

$$x = a_0 - \frac{J'(a_0)}{J''(a_0)}$$

$$W = W - \eta \frac{J'(W)}{J''(W)}$$

二维变量情况



$$W = W - \nabla^2 J(W)^{-1} \cdot \nabla J(W)$$

$\nabla^2 J(W)$也就是Hessian矩阵H(W)

$$\mathrm{H}(W) = \begin{bmatrix} \frac{\partial}{\partial W_1}\frac{\partial f}{\partial W_1} & \frac{\partial}{\partial W_2}\frac{\partial f}{\partial W_1} & \cdots & \frac{\partial}{\partial W_n}\frac{\partial f}{\partial W_1} \\ \frac{\partial}{\partial W_1}\frac{\partial f}{\partial W_2} & \frac{\partial}{\partial W_2}\frac{\partial f}{\partial W_2} & & \frac{\partial}{\partial W_n}\frac{\partial f}{\partial W_2} \\ \vdots & & \ddots & \vdots \\ \frac{\partial}{\partial W_1}\frac{\partial f}{\partial W_n} & \frac{\partial}{\partial W_2}\frac{\partial f}{\partial W_n} & \cdots & \frac{\partial}{\partial W_n}\frac{\partial f}{\partial W_n} \end{bmatrix}$$

原始-对偶法(Primal-Dual Method)

原问题

$$\begin{aligned} &\min c^T x \\ &\text{subject to } Ax = b \\ &x \geq 0 \end{aligned} \tag{9}$$

对偶问题

$$\begin{aligned} &\max b^T y \\ &\text{subject to } A^T y + s = c \\ &s \geq 0 \end{aligned} \tag{10}$$

`kkt`条件

$$\begin{aligned} A^T y + s &= c && (1) \\ Ax &= b && (2) \\ x_i s_i &= 0, i = 1, \ldots, n && (3) \\ (x, s) &\geq 0 && (4) \end{aligned}$$

其中

$$X = \mathrm{diag}(x_1, x_2, \ldots, x_n), \quad S = \mathrm{diag}(s_1, s_2, \ldots, s_n), \quad e = (1, 1, \ldots 1)^T \tag{5}$$

- 上述`xs`已经是一个非线性系统，一种实用的方法是采用牛顿法
- 这里需要定义一个量来检验当前的迭代点与最优点(x,s)的差距。在Barrier Method中，使用 duality gap 的上界 `m/t` 来检验的，在 Primal Dual Method 中，定义一个新的 duality measure 来进行某种衡量

$$u = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i s_i = \frac{x^T}{n} \tag{6}$$

上述 `u` 也被成为`对偶间隙`，当`u->0`时，`(x,s)`将接近可行域的边界，假设给定当前的可行点`(x,y,s)`,寻找下一个点

$$(x', y', s') = (x, y, s) + (\Delta x, \Delta y, \Delta s) \tag{7}$$

使得如下条件成立

$$A^T y' + s' = c, \qquad s' > 0 \tag{8}$$
$$Ax' = b, \qquad x' > 0 \tag{9}$$
$$x_i s_i = 0, \qquad i = 1, \ldots, n \tag{10}$$
$$x_i' s_i' = \sigma u, \qquad i = 1, \ldots, n; 0 < \sigma < 1 \tag{11}$$

上述问题也被成为**扰动`kkt`条件**，可进一步简化转化为矩阵形式为

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c \\ -XSe + \sigma \mu e \end{bmatrix} \tag{12}$$

其中

$$r_b = Ax - b, \quad r_c = A^T y + s - c \tag{13}$$

可以看出，Barrier Method 中控制 t 使得 duality gap 为 `m/t`，而在Primal Dual 内点法中控制 `σμ` ，二者是一致的。

- 当 `σμ` 小于一定阈值，则计算结束

`Ipopt` 求解思路

原问题

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0. \end{aligned}$$

1.处理不等式约束

作为内点法会将不等式问题装换为等式问题, 会构造了一个 `ln(x)` 的障碍函数，来代替`x>=0`

$$\min_{x \in \mathbb{R}^n} \quad \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \ln(x_i) \tag{4a}$$
$$\text{s.t.} \quad c(x) = 0, \tag{4b}$$

当 `xi` 小于零时，目标函数值为无穷大，因此，原问题的最优解任然会在 `x>=0` 的定义域内。 障碍函数的近似度项取决于障碍参数 μ 的大小，即在 `u -> 0` 时，新构造的问题最优解会收敛到原问题的最优解为 。因此，求解新问题的思路就是，通过不同的取值`u`，解决一系列不同的障碍函数构成的问题(**由u的取值不同决定**)以及用户指定的起始点，不断迭代u,直到满足`KKT`条件，那么问题得到解决。

2.处理等式约束

需要满足如下`KKT`条件

$$\text{增广拉格朗日函数：}\ L(x,y,z)=f(x)+c(x)y+u\cdot\sum_{i=0}^{n-1}ln(x) \tag{14}$$

$$\begin{aligned}\nabla f(x)+\nabla c(x)y-z&=0 &(5a)\\ c(x)&=0 &(5b)\\ XZe-\mu e&=0 &(5c)\\ x,z&\geq 0 &(5d)\end{aligned}$$

其中`令Z = u / X`，`e`为单位列向量，所以`z`也需要被纳入优化变量中，由(`5c`)可推断出这是个非线性系统，应为是x，z两个变量的乘积，一般采用牛顿法来求解。

2.1 将`KKT`条件转换为矩阵形式求解

Newton迭代的步长`step (Δxk, Δyk, Δzk)`

$$\begin{bmatrix}W_k+X_k^{-1}Z_k+\delta_x I & \nabla c(x_k)\\ \nabla c(x_k)^T & 0\end{bmatrix}\begin{pmatrix}\Delta x_k\\ \Delta y_k\end{pmatrix}=-\begin{pmatrix}\nabla\varphi_\mu(x_k)+\nabla c(x_k)y_k\\ c(x_k)\end{pmatrix}$$

其中 $\delta_x=0$ and $\Delta z_k=\mu X_k^{-1}e-z_k-X_k^{-1}Z_k\Delta x_k$.

$$W_k=\nabla^2 f(x_k)+\sum_{j=1}^{m}y_k^{(j)}\nabla^2 c^{(j)}(x_k)$$

3.评估结果

3.1 首先判断走过步长后的值，是否增加了一个足够大的值

$$x_k+\alpha_k^{x,\max}\Delta x_k\geq(1-\tau)x_k,\qquad z_k+\alpha_k^{z,\max}\Delta z_k\geq(1-\tau)z_k$$

3.2 如何满足条件，则更新(α的确定由过滤器类来确定)

$$x_{k+1}=x_k+\alpha_{k,l}^{x}\Delta x_k,\quad y_{k+1}=y_k+\alpha_{k,l}^{x}\Delta y_k,\quad z_{k+1}=z_k+\alpha_k^{z,\max}\Delta z_k$$

4.判断收敛准则

当容差满足`kkt`条件的时收敛

$$\begin{aligned}\max|\nabla f(x)+\nabla c(x)\lambda-z|&\leq\epsilon_{tol}\\ \max|c(x)|&\leq\epsilon_{tol}\\ \max|XZe-\mu e|&\leq\epsilon_{tol}\end{aligned}$$

计算结果输出

参数	意义
iter	迭代次数
objective	目标函数的当前值 (max-norm)
inf_pr	当前原始不可行性（最大范数）
inf_du	当前双重不可行性（最大范数）
lg(mu)	`log10` 的障碍函数的参数 u
||d||	原始搜索方向的无穷范数
lg(rg)	log10 的 Hessian 扰动 δx
alpha_du , 双步长 α	对偶步长αk
alpha_pr	原始步长αx
ls	回溯步数(α的大小要依赖于之前的数据)

col #	header	meaning
1	`iter`	iteration counter k (`r`: restoration phase)
2	`objective`	current value of objective function, $f(x_k)$
3	`inf_pr`	current primal infeasibility (max-norm), $\|c(x_k)\|_\infty$
4	`inf_du`	current dual infeasibility (max-norm), $\|\text{Eqn. } (5a)\|_\infty$
5	`lg(mu)`	$\log_{10}$ of current barrier parameter μ
6	`||d||`	max-norm of the primal search direction, $\|\Delta x_k\|_\infty$
7	`lg(rg)`	$\log_{10}$ of Hessian perturbation δ_x (--: none, $\delta_x = 0$)
8	`alpha_du`	dual step size $\alpha_k^{z,\max}$
9	`alpha_pr`	primal step size α_k^x
10	`ls`	number of backtracking steps $l+1$

`warm_start`

在求解之前我们往往对最优点处的有效约束了解很少,其实在一些应用中，我们需要求解一系列类似的 `QP` 命题，这个时候我们往往对最优点处的有效约束有一个初始猜测，因此通过这种方式可以实现算法的热启动（Warm Start），从而加速算法的收敛

example

这个例子可以用简单的几何方法求解出来。用数值方法可以得到同样的结果（在一定的精度范围之内）。之所以要用数值方法求解这个简单的问题，是因为数值方法对更复杂的问题同样有效，而简单的几何方法对复杂问题却已经不适用了。

例 1

$$min\frac{1}{12}(x_1+1)^3+x_2$$

$$s.t\,x_1-1\geq 0;x_2\geq 0$$

令 $G(x,r_k)=\frac{1}{12}(x_1+1)^3+x_2+r_k(\frac{1}{x_1-1}+\frac{1}{x_2})\,(1)$

用解析法求解：（其实就是高数中的求偏导并令其等于零）

$$\frac{\partial G(x)}{\partial x_1}=\frac{1}{4}(x_1+1)^2-\frac{r_k}{x_1-1}=0\,(2)$$

$$\frac{\partial G(x)}{\partial x_2}=1-\frac{r_k}{x_2^2}=0\,(3)$$

由 (1) 式可得，$x_1=\sqrt{1+2\sqrt{r_k}}$；由 (2) 式可得，$x_2=\sqrt{r_k}$，故:

$$\bar{x_{r_k}} = (x_1, x_2) = (\sqrt{1+2\sqrt{r_k}}, \sqrt{r_k})$$

当 $r_k \to 0$ 时，$\bar{x_{r_k}} \to \bar{x} = (1,0)$，$\bar{x}$ 就是原问题的最优解。

例 2

$$\begin{gathered}\min x+y \\ \text{s.t.} \\ x+2y \leq 1 \\ 2x+y \leq 1 \\ x \geq 0 \\ y \geq 0\end{gathered}$$

也就是

$$c = \begin{pmatrix}1\\1\end{pmatrix}, A = \begin{pmatrix}1 & 2\\2 & 1\\-1 & 0\\0 & -1\end{pmatrix}, b = \begin{pmatrix}1\\1\\0\\0\end{pmatrix}.$$

梯度为

$$\nabla f(x) = tc - A^T \begin{pmatrix}\frac{1}{x+2y-1}\\\frac{1}{2x+y-1}\\\frac{1}{-x}\\\frac{1}{-y}\end{pmatrix}$$

Hessian矩阵为

$$H_f(x) = A^T D A$$

其中，对角型矩阵 D 为

$$D = \begin{pmatrix}\frac{1}{(x+2y-1)^2} & 0 & 0 & 0\\0 & \frac{1}{(2x+y-1)^2} & 0 & 0\\0 & 0 & \frac{1}{x^2} & 0\\0 & 0 & 0 & \frac{1}{y^2}\end{pmatrix}$$

对于一个固定的参数 t，选择一个恰当的初始解 $x^{(0)}$，代入牛顿迭代公式

$$x^{(n+1)} = x^{(n)} - H_f(x^{(n)})^{-1}\nabla f(x^{(n)}), n \geq 0$$

可以得到一个依赖于参数 t 的解 $x_t^\star$.

用几何方法很容易求得这个例子的解为 $(x^\star, y^\star) = (0,0)$. 所以我们期待，当 $\lim\limits_{t\to\infty}(x_t^\star, y_t^\star) = (0,0)$.

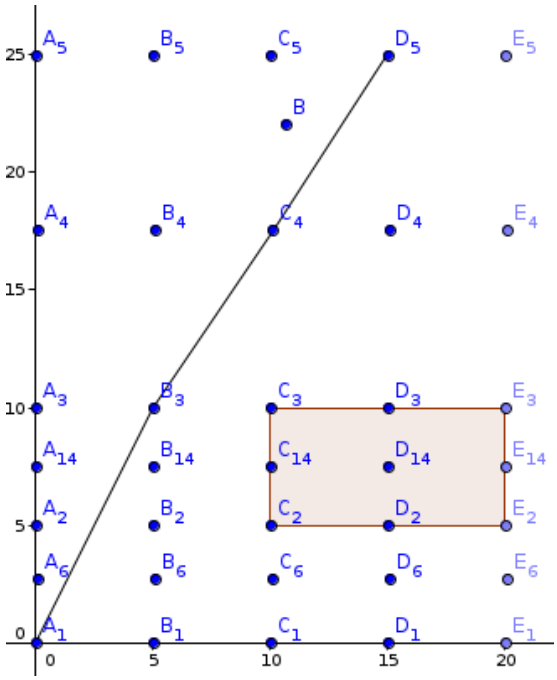
总结

- **对于最简单的无约束凸优化问题**，可以采用Gradient Descent或者Newton Method求解
- **对于稍复杂的等式约束凸优化问题**，可以对目标函数进行二阶Taylor近似，然后采用Newton Method求解`KKT`最优条件；
- **对于最复杂的带不等式约束问题**，则时引入对数障碍函数，转化为带等式约束的凸优化问题，然后采用Newton Method方法迭代求解。

速度优化

DP搜索

通过`DP`搜索出一系列点集合，相当于一条粗略的轨迹



针对`Dp`搜索出来的轨迹进行平滑，但是由于st图在构建时，在距离自车较近的范围内对s的采样间隔为0.1,而离自车较远的距离的采样间隔为1.0

piecewise_jerk_speed_nonlinear_optimizer

流程

- `OptimizeByQP`

首先对DP采集到的点集合进行平滑(按0.1采样)(`QP`求解)，为下面的非线性优化提供位置，速度，加速度的迭代初始值

$$w_{speed}\cdot\sum_{i=0}^{n-1}s_i^2+w'_{speed}\cdot\sum_{i=0}^{n-1}{s'_i}^2+w''_{speed}\cdot\sum_{i=0}^{n-1}{s''_i}^2+w'''_{speed}\cdot\sum_{i=0}^{n-2}(\frac{{s_{i+1}}''-{s_i}''}{\Delta s})^2 \tag{15}$$

- `CheckSpeedLimitFeasibility`
  限速检查，会检查规划周期的第一个点是否满足当前道路的的限速要求
  - 曲率曲线平滑`SmoothPathCurvature`(`QP`求解)

$$w_k\cdot\sum_{i=0}^{n-1}x_i^2+w'_k\cdot\sum_{i=0}^{n-1}{x'_i}^2+w''_k\cdot\sum_{i=0}^{n-1}{x''_i}^2+w'''_k\cdot\sum_{i=0}^{n-2}(\frac{{x_{i+1}}''-{x_i}''}{\Delta s})^2 \tag{16}$$

  - 限速曲线平滑`SmoothSpeedLimit`(`QP`求解)

$$w_{sl}\cdot\sum_{i=0}^{n-1}x_i^2+w'_{sl}\cdot\sum_{i=0}^{n-1}{x'_i}^2+w''_{sl}\cdot\sum_{i=0}^{n-1}{x''_i}^2+w'''_{sl}\cdot\sum_{i=0}^{n-2}(\frac{{x_{i+1}}''-{x_i}''}{\Delta s})^2 \tag{17}$$

为什么要对曲率和限速曲线进行平滑?

曲率曲线和限速曲线都是对纵向位移s的函数，而优化的变量正是s，所以在二次规划之前通过动态规划的粗轨迹提前计算出每个时间t时的曲率，得到每个时间t的曲率惩罚权重p。

- `OptimizeByNLP` 非线性优化入口

```
ptr_interface->set_safety_bounds(s_bounds_);

// Set weights and reference values
const auto& config = config_.piecewise_jerk_nonlinear_speed_config();

ptr_interface->set_curvature_curve(smoothed_path_curvature_);

// TODO(Hongyi): add debug_info for speed_limit fitting curve
ptr_interface->set_speed_limit_curve(smoothed_speed_limit_);
```

- 传入st图中的可行上下边界到对象中
- 传入上一步 `QP` 平滑后的曲率曲线和速度曲线
- 设置优化变量 `s` 的参考值，如果没有使用dp优化的参考线，就直接把优化变量个数和路径总长传入，开st空间内直接搜索

```
if (FLAGS_use_smoothed_dp_guide_line) {
    ptr_interface->set_reference_spatial_distance(*distance);
    // TODO(Jinyun): move to confs
    ptr_interface->set_w_reference_spatial_distance(10.0);
  } else {
    std::vector<double> spatial_potantial(num_of_knots_, total_length_);
    ptr_interface->set_reference_spatial_distance(spatial_potantial);
    ptr_interface->set_w_reference_spatial_distance(
        config.s_potential_weight());
  }
```

- 设置`wart_start`加入求解

```
void PiecewiseJerkSpeedNonlinearIpoptInterface::set_warm_start(
    const std::vector<std::vector<double>> &speed_profile) {
  x_warm_start_ = speed_profile;
}
```

- 设置优化变量的权重，包括 `s_bound`、加速度、加加速度、曲率、参考速度

- Ipopt 求解

数学建模

优化变量

变量包含了每个固定间隔时间点的位置、速度、加速度。此外非线性规划中如果打开了软约束`FLAGS_use_soft_bound_in_nonlinear_speed_opt`，还会有几个关于s软约束的松弛变量

$$s_0,\cdots s_{n-1},s_0'\cdots s_{n-1}',s_0'',\cdots,s_{n-1}'',s_{lowerSlack(0)},\cdots s_{lowerSlack(n-1)},,s_{upperSlack(0)},\cdots s_{upperSlack(n-1)}, \tag{18}$$

代价函数

$$\begin{aligned} cost\ \ function = & w_{ref}\cdot\sum_{i=0}^{n-1}(s_i-s_{refi})^2+w_{ref}'\cdot\sum_{i=0}^{n-1}(s_i'-s_{refi}')^2+w_{ref}''\cdot\sum_{i=0}^{n-1}s_i''^2+w_{ref}'''\cdot\sum_{i=0}^{n-2}(\frac{s_{i+1}''-s_i''}{\Delta t})^2+ \\ & w_{centrAcc}\cdot\sum_{i=0}^{n-1}(s_i'^2\cdot kappa(s_i))^2+ \\ & w_{end}\cdot\sum_{i=0}^{n-1}(s_{end}-s_{n-1})^2+w_{end}'\cdot\sum_{i=0}^{n-1}(s_{end}'-s_{n-1}')^2+w_{end}''\cdot\sum_{i=0}^{n-1}(s_{end}''-s_{n-1}'')^2+ \\ & w_s\sum_{i=0}^{n-1}s_{lowerSlack}+w_s\sum_{i=0}^{n-1}s_{upperSlack} \end{aligned} \tag{19}$$

> 原因一是引入松弛变量后的问题与原问题等价，最优解不改变，不等式$\boldsymbol{x_1+x_2\le 0}$通过引入松弛变量$\boldsymbol{s_1}$后变为$\boldsymbol{x_1+x_2+s_1=0}$且$\boldsymbol{s_1\ge 0}$，引入前后完全等价。原因二是通过引入松弛变量可以将原问题可行域扩展到更大的空间。。

约束

- 对单调性的约束(车只允许向前开)

$$0 \leq s_{i+1} - s_i \leq s_i' \cdot \Delta t \tag{20}$$

- 对`Jerk`的约束

$$-s_{i_{max}}''' \leq \frac{s_i'' - s_i''}{\Delta t} \leq s_{i_{max}}''' \tag{21}$$

- 连续性约束(位置相等 & 速度相等)

$$\begin{aligned} s_{i+1}' - s_i' - \frac{1}{2}\Delta t * s_i'' - \frac{1}{2}\Delta t * s_{i+1}'' = 0 \\ s_{i+1} - s_i - \Delta t \cdot s_i' - \frac{1}{3}\Delta t^2 \cdot s_i'' - \frac{1}{6}\Delta t^2 \cdot s_{i+1}'' = 0 \end{aligned} \tag{22}$$

- 对`speed_limit`的约束(非线性)

$$s_i' - vboundfunc(s_i) \leq 0.0 \tag{23}$$

- 设置安全边界(加入松弛因子,合并到代价函数中)

$$soft_lower_i - lowerslack_i \leq s_i \leq softupper_i + lowerslack_i \tag{24}$$

目标函数的梯度

原始

$$\frac{\delta costfunc}{\delta s_i} = 2 \cdot w_{ref} \cdot (s_i - s_{refi}) + 2 \cdot w_{centrAcc} \cdot s_i^5 \cdot kappa(s_i) \cdot kappa'(s_i) \tag{25}$$

一阶

$$\frac{\delta costfunc}{\delta s_i'} = 2 \cdot w_{ref}' \cdot (s_i' - s_{refi}') + 4 \cdot w_{centrAcc} \cdot s_i^3 \cdot kappa^2(s_i) \tag{26}$$

二阶

$$\frac{\delta costfunc}{\delta s_i''} = 2 \cdot w_{ref}'' \cdot s_i'' - 2 \cdot w_{rfe}'' \cdot \frac{s_{i+1}'' - s_i'}{\Delta t^2} \tag{27}$$

松弛变量

$$\begin{aligned} \frac{\delta costfunc}{\delta s_{lowerslack}} = s_{lowerSlack} \\ \frac{\delta costfunc}{ds_{upperslack}} = s_{upperSlack} \end{aligned} \tag{28}$$

约束条件的雅克比矩阵

对于速度约束:

$$\boldsymbol{g_1 = \dot{s}_i - speed\_limit(s_i)}$$

$$\boldsymbol{\frac{\delta g_1}{d\dot{s}_i} = 1}$$

$$\boldsymbol{\frac{\delta g_1}{ds_i} = -speed\_limit(s_i)}$$

对于位置软约束:

$$\boldsymbol{g_2 = s_i - s\_lower_i + slack\_lower_i}$$

$$\boldsymbol{\frac{\delta g_2}{ds_i} = 1}$$

$$\boldsymbol{\frac{\delta g_2}{dslack\_lower_i} = 1}$$

$$g_3 = s_i - s\_upper_i + slack\_upper_i$$
$$\frac{\delta g_2}{ds_i} = 1$$
$$\frac{\delta g_2}{dslack\_upper_i} = -1$$

对于jerk约束：

$$g_3 = \frac{\ddot{s}_{i+1} - \ddot{s}_i}{\Delta t}$$
$$\frac{\delta g_3}{d\ddot{s}_i} = \frac{-1}{\Delta t}$$
$$\frac{\delta g_3}{d\ddot{s}_{i+1}} = \frac{1}{\Delta t}$$

微分关系的等式约束

$$g_4 = s_{i+1} - s_i - \Delta t\dot{s}_i - \frac{\Delta t^2\ddot{s}_i}{3} - \frac{\Delta t^3\ddot{s}_{i+1}}{6}$$
$$\frac{\delta g_4}{ds_i} = -1$$
$$\frac{\delta g_4}{ds_{i+1}} = 1$$
$$\frac{\delta g_4}{ddots_i} = -\Delta t$$
$$\frac{\delta g_4}{d\ddot{s}_i} = -\frac{\Delta t^2}{3}$$
$$\frac{\delta g_4}{d\ddot{s}_{i+1}} = -\frac{\Delta t^3}{6}$$

$$g_5 = \dot{s}_{i+1} - \dot{s}_i - \Delta t\ddot{s}_i - \frac{\Delta t(\ddot{s}_{i+1} - \ddot{s}_i)}{2}$$
$$\frac{\delta g_5}{d\dot{s}_i} = -1$$
$$\frac{\delta g_5}{d\dot{s}_{i+1}} = 1$$
$$\frac{\delta g_5}{d\ddot{s}_i} = -\frac{\Delta t}{2}$$
$$\frac{\delta g_5}{d\ddot{s}_{i+1}} = -\frac{\Delta t}{2}$$

对于位置约束

$$g_6 = s_{i+1} - s_i$$
$$\frac{\delta g_6}{ds_{i+1}} = 1$$
$$\frac{\delta g_6}{ds_i} = -1$$

`Ipopt` 求解最优化问题

原始-对偶法(Primal-Dual Method)

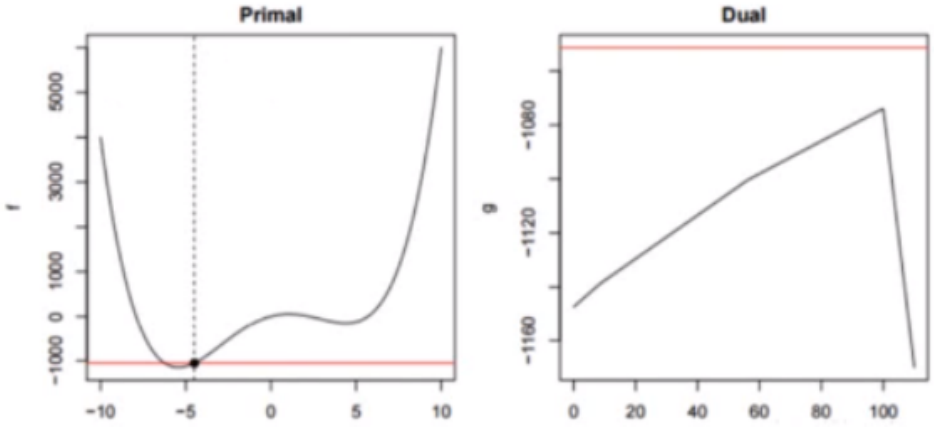
原问题

$$\begin{aligned} &\min c^T x \\ &\text{subject to } Ax = b \\ &x \geq 0 \end{aligned} \tag{9}$$

对偶问题

$$\begin{aligned} &\max b^T y \\ &\text{subject to } A^T y + s = c \\ &s \geq 0 \end{aligned} \tag{10}$$

$$\begin{aligned} &\min_x \ (x^4 - 50x^2 + 100x) \\ &s.t. \quad x \geq 4.5 \end{aligned}$$



`kkt`条件

$$A^T y + s = c \tag{29}$$
$$Ax = b \tag{30}$$
$$x_i s_i = 0, i = 1, \ldots, n \tag{31}$$
$$(x, s) \geq 0 \tag{32}$$

其中

$$X = \text{diag}(x_1, x_2, \ldots, x_n), \quad S = \text{diag}(s_1, s_2, \ldots, s_n), \quad e = (1, 1, \ldots 1)^T \tag{33}$$

- 上述`xs`已经是一个非线性系统，一种实用的方法是采用牛顿法
- 这里需要定义一个量来检验当前的迭代点与最优点(x,s)的差距。在Barrier Method中，使用 duality gap 的上界 `m/t` 来检验的，在 Primal Dual Method 中，定义一个新的 duality measure 来进行某种衡量

$$u = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i s_i = \frac{x^T}{n} \tag{34}$$

上述 `u` 也被成为 `对偶间隙`，当`u->0`时，`(x,s)`将接近可行域的边界，假设给定当前的可行点`(x,y,s)`,寻找下一个点

$$(x', y', s') = (x, y, s) + (\Delta x, \Delta y, \Delta s) \tag{35}$$

使得如下条件成立

$$A^T y' + s' = c, \qquad s' > 0 \tag{36}$$
$$Ax' = b, \qquad x' > 0 \tag{37}$$
$$x_i s_i = 0, \qquad i = 1, \ldots, n \tag{38}$$
$$x_i' s_i' = \sigma u, \qquad i = 1, \ldots, n; 0 < \sigma < 1 \tag{39}$$

上述问题也被成为**扰动`kkt`条件**，可进一步简化转化为矩阵形式为

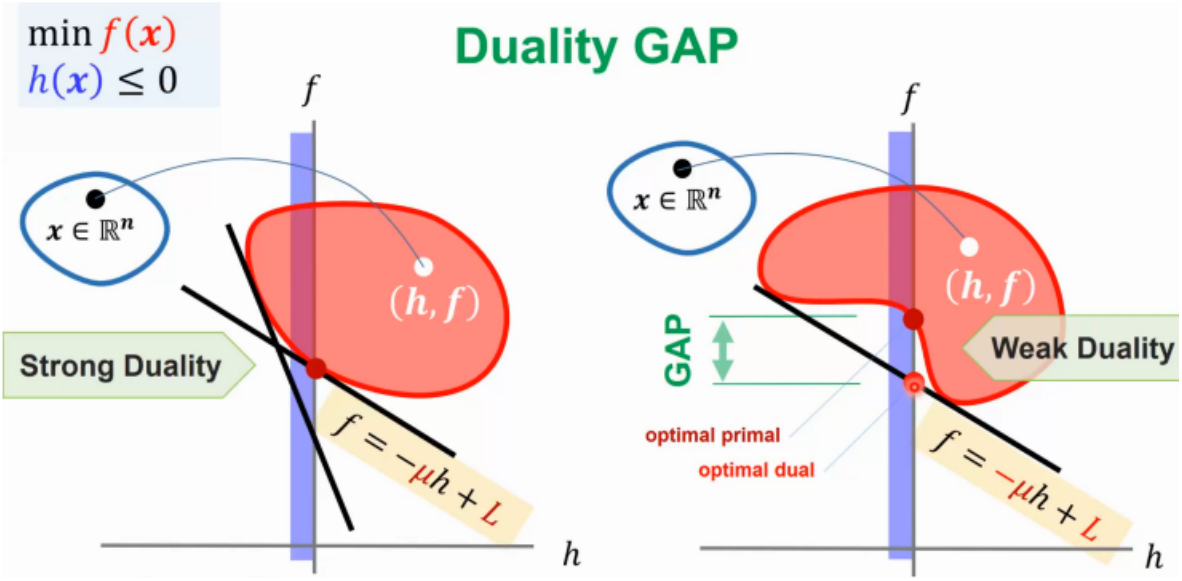
$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c \\ -XSe + \sigma \mu e \end{bmatrix} \tag{40}$$

其中

$$r_b = Ax - b, \quad r_c = A^T y + s - c \tag{41}$$

可以看出，Barrier Method 中控制 t 使得 duality gap 为 `m/t`，而在Primal Dual 内点法中控制 `σμ`，二者是一致的。

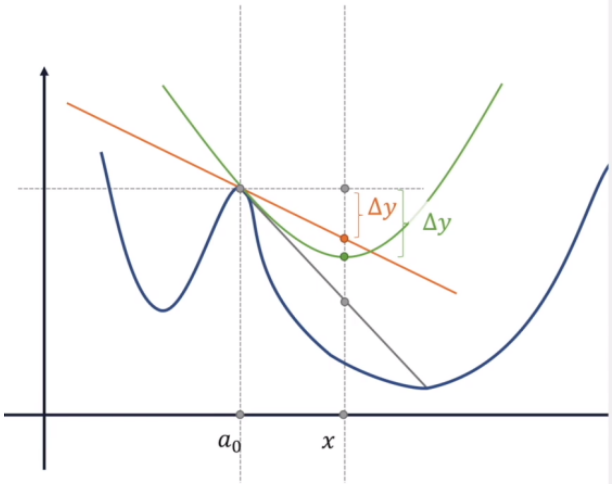
- 当 `σμ` 小于一定阈值，则计算结束



牛顿迭代法

上述矩阵形式不是线性方程组，需要用到牛顿迭代法求解

一维变量情况



$f(x)$泰勒展开保留到二阶导：

$$f(x) = J(a_0) + J'(a_0)(x - a_0) + \frac{1}{2}J''(a_0)(x - a_0)^2$$

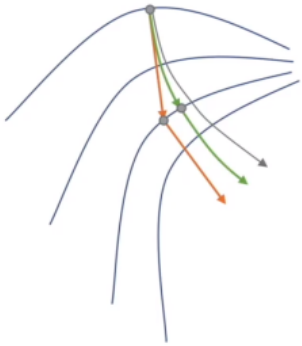
$f(x)$求极值，令$f'(x) = 0$

$$f'(x) = J'(a_0) + J''(a_0)(x - a_0) = 0$$

$$x = a_0 - \frac{J'(a_0)}{J''(a_0)}$$

$$W = W - \eta \frac{J'(W)}{J''(W)}$$

二维变量情况



$$W = W - \nabla^2 J(W)^{-1} \cdot \nabla J(W)$$

$\nabla^2 J(W)$也就是Hessian矩阵H(W)

$$\mathrm{H}(W) = \begin{bmatrix} \frac{\partial}{\partial W_1}\frac{\partial f}{\partial W_1} & \frac{\partial}{\partial W_2}\frac{\partial f}{\partial W_1} & \cdots & \frac{\partial}{\partial W_n}\frac{\partial f}{\partial W_1} \\ \frac{\partial}{\partial W_1}\frac{\partial f}{\partial W_2} & \frac{\partial}{\partial W_2}\frac{\partial f}{\partial W_2} & & \frac{\partial}{\partial W_n}\frac{\partial f}{\partial W_2} \\ \vdots & & \ddots & \vdots \\ \frac{\partial}{\partial W_1}\frac{\partial f}{\partial W_n} & \frac{\partial}{\partial W_2}\frac{\partial f}{\partial W_n} & \cdots & \frac{\partial}{\partial W_n}\frac{\partial f}{\partial W_n} \end{bmatrix}$$

`Ipopt` 接口

`get_nlp_info`

对该函数对优化问题的维度进行设置

- n=4，变量x个数
- m=2，约束条件个数
- nnz_jac_g=8，Jacobian非零个数
- Nnz_h_lag = 10，Hessian非零个数

`get_bounds_info`

设置优化变量的上下限范围以及约束条件的上下限值

- x_l[i]设置xi的下界值
- x_u[i]设置xi的上界值
- g_l[i]设置约束i的下界值
- g_u[i]设置约束i的上界值

`get_start_point`

设置优化变量的初始点

- x[i]设置第i个变量的初始迭代值

`eval_f`

设置代价函数

- object_value 设置目标函数计算方式

`eval_grad_f`

设置目标函数的梯度

- grad_f[i]设置目标函数对第i个变量的偏导

`eval_g`

设置约束条件

- g[i]约束条件i

`eval_jac_g`

设置约束条件的雅克比矩阵

- `iRow`和`jCol`设置非零行列的坐标
- Values设置矩阵迭代值，如果values==NULL，即尚未初始化时，需要设置雅克比矩阵哪些下标位置非零

`eval_h`

设置二阶梯度海森矩阵

- `iRow`和`jCol`设置非零行列的坐标
- obj_factor为目标函数系数
- lambda[i]为第i个约束的拉格朗日乘子
- values设置矩阵的迭代求值

`finalize_solution`

求解

- status为返回的求解状态
- obj_value最优值
- x:最优解变量取值
- z_l 拉格朗日乘子上界
- z_u 拉格朗日乘子下届
- lambda 最优解拉格朗日乘子取值