

# PRACTICAL OPTIMIZATION ALGORITHMS

徐 翔

数学科学学院  
浙江大学

MAY 15, 2023

## 第十一讲：二次规划(LINEAR AND QUADRATIC PROGRAMMING)

## 简介(INTRODUCTION)

# 简介

- 一般形式：

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } \quad &\begin{cases} a_i^T x = b_i, & i \in \mathcal{E} \\ a_i^T x \geq b_i, & i \in \mathcal{I}. \end{cases} \end{aligned}$$

其中 $G$ 是对称矩阵。

# 简介

- 一般形式：

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } \quad &\begin{cases} a_i^T x = b_i, & i \in \mathcal{E} \\ a_i^T x \geq b_i, & i \in \mathcal{I}. \end{cases} \end{aligned}$$

其中 $G$ 是对称矩阵。

- 二次规划一定可以在有限步内完成，运算量取决于目标函数和不等式约束个数。

# 简介

- 一般形式：

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } \quad &\begin{cases} a_i^T x = b_i, & i \in \mathcal{E} \\ a_i^T x \geq b_i, & i \in \mathcal{I}. \end{cases} \end{aligned}$$

其中 $G$ 是对称矩阵。

- 二次规划一定可以在有限步内完成，运算量取决于目标函数和不等式约束个数。
- 如果 $G$ 是半正定的，即是凸规划问题，那么困难程度等价于线性规划。

# 简介

- 一般形式：

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } \quad &\begin{cases} a_i^T x = b_i, & i \in \mathcal{E} \\ a_i^T x \geq b_i, & i \in \mathcal{I}. \end{cases} \end{aligned}$$

其中 $G$ 是对称矩阵。

- 二次规划一定可以在有限步内完成，运算量取决于目标函数和不等式约束个数。
- 如果 $G$ 是半正定的，即是凸规划问题，那么困难程度等价于线性规划。
- 如果 $G$ 不定，那么困难显著增加，因为会存在多个平衡点或者局部极小解。

# 简介

- 一般形式：

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } \quad &\begin{cases} a_i^T x = b_i, & i \in \mathcal{E} \\ a_i^T x \geq b_i, & i \in \mathcal{I}. \end{cases} \end{aligned}$$

其中 $G$ 是对称矩阵。

- 二次规划一定可以在有限步内完成，运算量取决于目标函数和不等式约束个数。
- 如果 $G$ 是半正定的，即是凸规划问题，那么困难程度等价于线性规划。
- 如果 $G$ 不定，那么困难显著增加，因为会存在多个平衡点或者局部极小解。
- 这里先考虑凸规划问题，即有唯一的全局极小解。



# 仅有等式约束

- 一般形式:

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } Ax &= b, \end{aligned}$$

其中  $A$  是  $m \times n$  的矩阵 (可以看做是非线性约束的线性化 Jacobian, 即  $\nabla c_i(x)$ )。

# 仅有等式约束

- 一般形式：

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } Ax &= b, \end{aligned}$$

其中 $A$ 是 $m \times n$ 的矩阵（可以看做是非线性约束的线性化Jacobian，即 $\nabla c_i(x)$ ）。

- 根据一阶最优性KKT条件，等价于求解如下问题的解

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

# 仅有等式约束

- 一般形式：

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T Gx + x^T c \\ \text{s.t. } Ax &= b, \end{aligned}$$

其中 $A$ 是 $m \times n$ 的矩阵（可以看做是非线性约束的线性化Jacobian，即 $\nabla c_i(x)$ ）。

- 根据一阶最优性KKT条件，等价于求解如下问题的解

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

- 如果我们把计算格式写成  $x^* = x + p$ ,  $x$ 是当前的值,  $p$ 是更新量, 则可以得到如下方程

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix}$$

其中 $h = Ax - b$ ,  $g = c + Gx$ ,  $p = x^* - x$ . 以上系数矩阵称为KKT矩阵。

## 仅有等式约束

回忆一下我们记 $Z$ 是 $A$ 的零空间的基构成的矩阵, 即  $AZ = 0$ ,  $\text{rank}(Z)=n-m$ ,  
 $Z \in R^{n \times (n-m)}$

### 引理1

假设 $A$ 是行满秩的, 并且假设约化后的Hessian矩阵 $Z^T G Z$ 是正定的, 则KKT矩阵是非奇异的, 并且存在唯一的解 $(x^*, \lambda^*)$

# 仅有等式约束

回忆一下我们记 $Z$ 是 $A$ 的零空间的基构成的矩阵, 即  $AZ = 0$ ,  $\text{rank}(Z)=n-m$ ,  
 $Z \in R^{n \times (n-m)}$

## 引理1

假设 $A$ 是行满秩的, 并且假设约化后的Hessian矩阵 $Z^T G Z$ 是正定的, 则KKT矩阵是非奇异的, 并且存在唯一的解 $(x^*, \lambda^*)$

证明(反证法)

# 仅有等式约束

回忆一下我们记 $Z$ 是 $A$ 的零空间的基构成的矩阵, 即  $AZ = 0$ ,  $\text{rank}(Z)=n-m$ ,  $Z \in R^{n \times (n-m)}$

## 引理1

假设 $A$ 是行满秩的, 并且假设约化后的Hessian矩阵 $Z^T G Z$ 是正定的, 则KKT矩阵是非奇异的, 并且存在唯一的解 $(x^*, \lambda^*)$

证明(反证法)

- 假设存在不同时为0的 $w, v$ 满足

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = 0$$

# 仅有等式约束

回忆一下我们记 $Z$ 是 $A$ 的零空间的基构成的矩阵, 即  $AZ = 0$ ,  $\text{rank}(Z)=n-m$ ,  $Z \in R^{n \times (n-m)}$

## 引理1

假设 $A$ 是行满秩的, 并且假设约化后的Hessian矩阵 $Z^T G Z$ 是正定的, 则KKT矩阵是非奇异的, 并且存在唯一的解 $(x^*, \lambda^*)$

证明(反证法)

- 假设存在不同时为0的 $w, v$ 满足

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = 0$$

- 可以得到 $Aw = 0$ 。

# 仅有等式约束

回忆一下我们记 $Z$ 是 $A$ 的零空间的基构成的矩阵, 即  $AZ = 0$ ,  $\text{rank}(Z) = n - m$ ,  $Z \in R^{n \times (n-m)}$

## 引理1

假设 $A$ 是行满秩的, 并且假设约化后的Hessian矩阵 $Z^T G Z$ 是正定的, 则KKT矩阵是非奇异的, 并且存在唯一的解 $(x^*, \lambda^*)$

证明(反证法)

- 假设存在不同时为0的 $w, v$ 满足

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = 0$$

- 可以得到 $Aw = 0$ 。进而可以得到

$$0 = \begin{bmatrix} w \\ v \end{bmatrix}^T \begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = w^T G w$$



# 仅有等式约束

## 仅有等式约束

- 由于 $w$ 在 $A$ 的零空间中, 即 $w = Zu$ . 因此我们可以得到

$$0 = w^T G w = u^T Z^T G Z u$$

## 仅有等式约束

- 由于 $w$ 在 $A$ 的零空间中, 即 $w = Zu$ . 因此我们可以得到

$$0 = w^T G w = u^T Z^T G Z u$$

- 由于 $Z^T G Z$ 是正定的, 因此可以推出  $u = 0$ 。因此 $w = 0$ 。

# 仅有等式约束

- 由于 $w$ 在 $A$ 的零空间中, 即 $w = Zu$ . 因此我们可以得到

$$0 = w^T G w = u^T Z^T G Z u$$

- 由于 $Z^T G Z$ 是正定的, 因此可以推出  $u = 0$ 。因此 $w = 0$ 。
- 再根据

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = 0$$

得到 $A^T v = 0$ 。显然由于 $A$ 是行满秩的, 因此可以得到 $v = 0$ 。

# 仅有等式约束

- 由于 $w$ 在 $A$ 的零空间中, 即 $w = Zu$ . 因此我们可以得到

$$0 = w^T G w = u^T Z^T G Z u$$

- 由于 $Z^T G Z$ 是正定的, 因此可以推出  $u = 0$ 。因此 $w = 0$ 。
- 再根据

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = 0$$

得到 $A^T v = 0$ 。显然由于 $A$ 是行满秩的, 因此可以得到 $v = 0$ 。

- 这与假设不符。因此KKT矩阵一定是非奇异的。

## 定理1:

假设 $A$ 是行满秩的, 并且假设约化后的Hessian矩阵 $Z^T G Z$ 是正定的, 则 $x^*$ 是二次规划问题的**全局唯一解**。

# 直接求解KKT系统

- 下面我们介绍一种直接求解KKT系统的算法。

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix}$$

# 直接求解KKT系统

- 下面我们介绍一种直接求解KKT系统的算法。

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix}$$

- 首先我们的一个观察：只要 $m \geq 1$ ，KKT矩阵一定是非正定的。

# 直接求解KKT系统

- 下面我们介绍一种直接求解KKT系统的算法。

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix}$$

- 首先我们的一个观察：只要  $m \geq 1$ ，KKT矩阵一定是非正定的。
- 定义三个重要的量  $n_+$ ,  $n_-$ ,  $n_0$ ，分别是KKT矩阵的正、负和零特征值的个数，记  $(n_+, n_-, n_0)$  为矩阵的惯性指标(inertia)。

## 定理2

记  $K$  为KKT矩阵，假设  $A$  是行满秩  $m$ 。则  $x^*$  是二次规划问题的

$$\text{inertia}(K) = \text{inertia}(Z^T G Z) + (m, m, 0)$$

因此，如果  $Z^T G Z$  是正定的，那么  $\text{inertia}(K) = (n, m, 0)$ 。



# 矩阵分解

- 最有效的方法是不定对称分解

# 矩阵分解

- 最有效的方法是不定对称分解
- 一个一般的对称矩阵  $K$ ，一定存在如下分解

$$P^T K P = L B L^T$$

其中  $P$  是个初等变换矩阵， $L$  是下三角矩阵（对角元为1）， $B$  是块对角矩阵（每个块最多  $2 \times 2$ ）。

# 矩阵分解

- 最有效的方法是不定对称分解
- 一个一般的对称矩阵  $K$ ，一定存在如下分解

$$P^T K P = L B L^T$$

其中  $P$  是个初等变换矩阵， $L$  是下三角矩阵（对角元为1）， $B$  是块对角矩阵（每个块最多  $2 \times 2$ ）。

- 这个分解的计算量大约与高斯消元法的运算量相当

# 矩阵分解

- 最有效的方法是不定对称分解
- 一个一般的对称矩阵  $K$ ，一定存在如下分解

$$P^T K P = L B L^T$$

其中  $P$  是个初等变换矩阵， $L$  是下三角矩阵（对角元为1）， $B$  是块对角矩阵（每个块最多  $2 \times 2$ ）。

- 这个分解的计算量大约与高斯消元法的运算量相当
- 求解顺序：首先求解  $Lz = P^T \begin{bmatrix} g \\ h \end{bmatrix}$  得到  $z$ 。

# 矩阵分解

- 最有效的方法是不定对称分解
- 一个一般的对称矩阵  $K$ ，一定存在如下分解

$$P^T K P = L B L^T$$

其中  $P$  是个初等变换矩阵， $L$  是下三角矩阵（对角元为1）， $B$  是块对角矩阵（每个块最多  $2 \times 2$ ）。

- 这个分解的计算量大约与高斯消元法的运算量相当
- 求解顺序：首先求解  $Lz = P^T \begin{bmatrix} g \\ h \end{bmatrix}$  得到  $z$ 。然后求解  $B\hat{z} = z$  得到  $\hat{z}$ 。

# 矩阵分解

- 最有效的方法是不定对称分解
- 一个一般的对称矩阵 $K$ ，一定存在如下分解

$$P^T K P = L B L^T$$

其中 $P$ 是个初等变换矩阵， $L$ 是下三角矩阵（对角元为1）， $B$ 是块对角矩阵（每个块最多 $2 \times 2$ ）。

- 这个分解的计算量大约与高斯消元法的运算量相当
- 求解顺序：首先求解  $Lz = P^T \begin{bmatrix} g \\ h \end{bmatrix}$  得到 $z$ 。然后求解 $B\hat{z} = z$ 得到 $\hat{z}$ 。再求解 $L^T \bar{z} = \hat{z}$ 得到 $\bar{z}$ 。

# 矩阵分解

- 最有效的方法是不定对称分解
- 一个一般的对称矩阵  $K$ ，一定存在如下分解

$$P^T K P = L B L^T$$

其中  $P$  是个初等变换矩阵， $L$  是下三角矩阵（对角元为1）， $B$  是块对角矩阵（每个块最多  $2 \times 2$ ）。

- 这个分解的计算量大约与高斯消元法的运算量相当
- 求解顺序：首先求解  $Lz = P^T \begin{bmatrix} g \\ h \end{bmatrix}$  得到  $z$ 。然后求解  $B\hat{z} = z$  得到  $\hat{z}$ 。再求解  $L^T \bar{z} = \hat{z}$  得到  $\bar{z}$ 。最后计算  $\begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = P\bar{z}$ 。

# 矩阵分解

- 最有效的方法是不定对称分解
- 一个一般的对称矩阵  $K$ ，一定存在如下分解

$$P^T K P = L B L^T$$

其中  $P$  是个初等变换矩阵， $L$  是下三角矩阵（对角元为1）， $B$  是块对角矩阵（每个块最多  $2 \times 2$ ）。

- 这个分解的计算量大约与高斯消元法的运算量相当
- 求解顺序：首先求解  $Lz = P^T \begin{bmatrix} g \\ h \end{bmatrix}$  得到  $z$ 。然后求解  $B\hat{z} = z$  得到  $\hat{z}$ 。再求解  $L^T \bar{z} = \hat{z}$  得到  $\bar{z}$ 。最后计算  $\begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = P\bar{z}$ 。
- 其他方法，如 Schur补方法(Schur-Complement)，零空间法 (Null-space) 等。



# 迭代法求解KKT系统

- 如果KKT系统规模很大，一般不适用于直接法求解，而应该使用迭代法求解。

# 迭代法求解KKT系统

- 如果KKT系统规模很大，一般不适用于直接法求解，而应该使用迭代法求解。
- 一般不推荐共轭梯度法，原因是对非正定系统，可能会导致数值不稳定。

# 迭代法求解KKT系统

- 如果KKT系统规模很大，一般不适用于直接法求解，而应该使用迭代法求解。
- 一般不推荐共轭梯度法，原因是对非正定系统，可能会导致数值不稳定。
- 比较好的选择是Krylov子空间法，包括GMRES，QMR，LSQR方法等。

## 不等式约束的二次规划问题

# 不等式约束问题

- 积极集方法(active set method)在1970s广泛应用，比较适合小规模或中等规模问题。

# 不等式约束问题

- 积极集方法(active set method)在1970s广泛应用，比较适合小规模或中等规模问题。
- 内点法(Interior point method)在1990s广泛应用，比较适合大规模问题的计算。

# 不等式约束问题

- 积极集方法(active set method)在1970s广泛应用，比较适合小规模或中等规模问题。
- 内点法(Interior point method)在1990s广泛应用，比较适合大规模问题的计算。
- 内点法不一定是最有效的，因为需要求解很多二次规划子问题。

# 一阶最优性条件

- 写出相应的拉格朗日函数

$$L(x, \lambda) = \frac{1}{2}x^T Gx + x^T c - \sum_{i \in \mathcal{E} \cup \mathcal{J}} \lambda_i (a_i^T x - b_i)$$



# 一阶最优性条件

- 写出相应的拉格朗日函数

$$L(x, \lambda) = \frac{1}{2}x^T Gx + x^T c - \sum_{i \in \mathcal{E} \cup \mathcal{J}} \lambda_i (a_i^T x - b_i)$$

- 活跃集  $\mathcal{A}(x^*)$  包括了

$$\mathcal{A}(x^*) = \{i \in \mathcal{E} \cup \mathcal{J} | a_i^T x^* = b_i\}.$$

# 一阶最优性条件

- 写出相应的拉格朗日函数

$$L(x, \lambda) = \frac{1}{2}x^T Gx + x^T c - \sum_{i \in \mathcal{E} \cup \mathcal{J}} \lambda_i (a_i^T x - b_i)$$

- 活跃集  $\mathcal{A}(x^*)$  包括了

$$\mathcal{A}(x^*) = \{i \in \mathcal{E} \cup \mathcal{J} | a_i^T x^* = b_i\}.$$

- 根据KKT条件，我们得到

$$Gx^* + c - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0,$$

$$a_i^T x^* = b_i, \quad \text{for all } i \in \mathcal{A}(x^*),$$

$$a_i^T x^* \geq b_i, \quad \text{for all } i \in \mathcal{J} \setminus \mathcal{A}(x^*),$$

$$\lambda_i^* \geq 0, \quad \text{for all } i \in \mathcal{J} \cap \mathcal{A}(x^*).$$

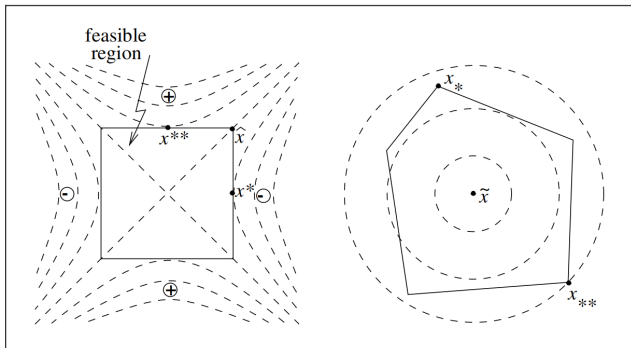
定理3: 如果存在  $\lambda^*$  和  $x^*$  满足上述方程，并且  $G$  是半正定的，则  $x^*$  一定是原二次规划问题的全局唯一解。

# 一阶最优性条件

- 如果 $G$ 不定，那么通常原问题不止一个解，这种情况不再是凸规划问题，而是叫不定二次规划或者负定二次规划。

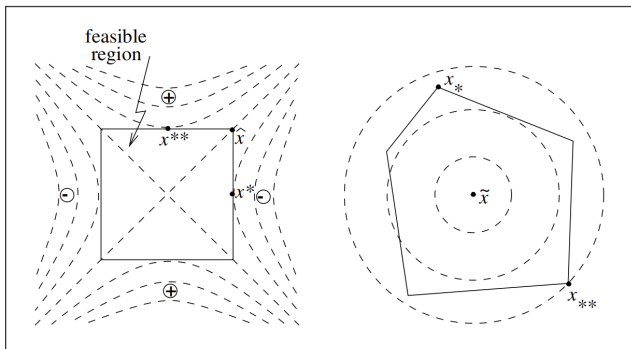
# 一阶最优性条件

- 如果 $G$ 不定，那么通常原问题不止一个解，这种情况不再是凸规划问题，而是叫不定二次规划或者负定二次规划。



# 一阶最优性条件

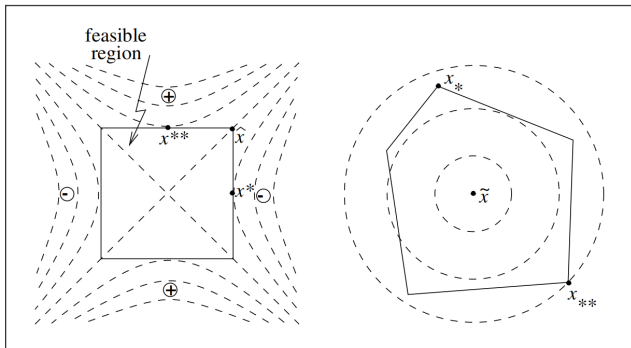
- 如果 $G$ 不定，那么通常原问题不止一个解，这种情况不再是凸规划问题，而是叫不定二次规划或者负定二次规划。



- 左边是不定问题， $x^{**}$ 是极大值， $x^*$ 是个局部极小，中心点是个鞍点(saddle point)。

# 一阶最优性条件

- 如果 $G$ 不定，那么通常原问题不止一个解，这种情况不再是凸规划问题，而是叫不定二次规划或者负定二次规划。



- 左边是不定问题， $x^{**}$ 是极大值， $x^*$ 是个局部极小，中心点是个鞍点(saddle point)。
- 右边是负定问题， $\tilde{x}$ 是全局最大值， $x^*$ ， $x^{**}$ 都是局部极小值点。

# 退化性(Degeneracy)

- 另一个困难是退化性导致的。通常可以分为两种

# 退化性(Degeneracy)

- 另一个困难是退化性导致的。通常可以分为两种
  - 活跃的约束的梯度 $a_i, i \in \mathcal{A}(x^*)$ , 是线性相关的

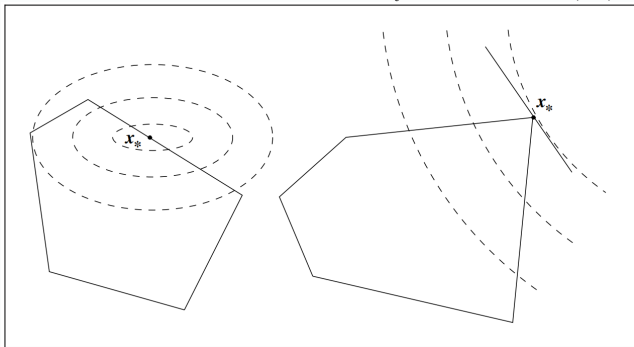


# 退化性(Degeneracy)

- 另一个困难是退化性导致的。通常可以分为两种
  - 活跃的约束的梯度 $a_i, i \in \mathcal{A}(x^*)$ , 是线性相关的
  - 严格的补条件并不满足, 即存在某些 $\lambda_i^* = 0$ 对于 $i \in \mathcal{A}(x^*)$

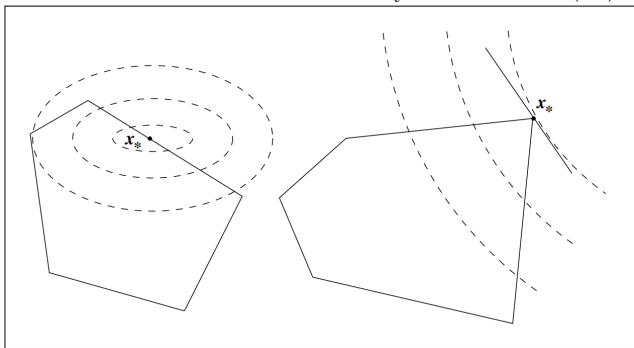
# 退化性(Degeneracy)

- 另一个困难是退化性导致的。通常可以分为两种
  - 活跃的约束的梯度 $a_i, i \in \mathcal{A}(x^*)$ , 是线性相关的
  - 严格的补条件并不满足, 即存在某些 $\lambda_i^* = 0$ 对于 $i \in \mathcal{A}(x^*)$



# 退化性(Degeneracy)

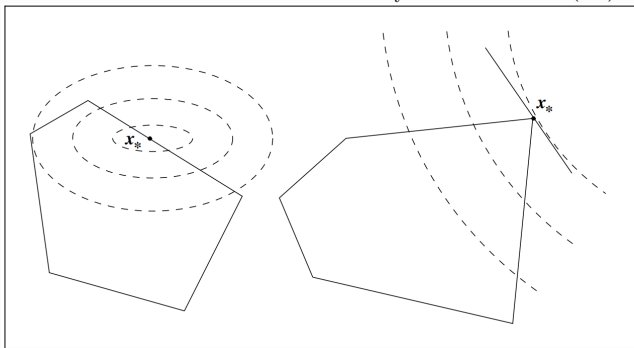
- 另一个困难是退化性导致的。通常可以分为两种
  - 活跃的约束的梯度 $a_i, i \in \mathcal{A}(x^*)$ , 是线性相关的
  - 严格的补条件并不满足, 即存在某些 $\lambda_i^* = 0$ 对于 $i \in \mathcal{A}(x^*)$



- 左边图中在 $x^*$ 处仅有一个活跃的约束, 也是无约束优化问题的解, 因此 $\lambda_i \equiv 0$ .

# 退化性(Degeneracy)

- 另一个困难是退化性导致的。通常可以分为两种
  - 活跃的约束的梯度 $a_i, i \in \mathcal{A}(x^*)$ , 是线性相关的
  - 严格的补条件并不满足, 即存在某些 $\lambda_i^* = 0$ 对于 $i \in \mathcal{A}(x^*)$



- 左边图中在 $x^*$ 处仅有一个活跃的约束, 也是无约束优化问题的解, 因此 $\lambda_i \equiv 0$ .
- 在 $x^*$ 处有三个活跃的约束, 但是在二维向量空间里, 三个必定线性相关。

# 积极集方法求解凸二次规划问题

- 给定当前 $x_k$ ，工作集 $\mathcal{W}_k$ ，假设在工作集中所有的 $a_i$ 是线性无关的.

# 积极集方法求解凸二次规划问题

- 给定当前 $x_k$ ，工作集 $\mathcal{W}_k$ ，假设在工作集中所有的 $a_i$ 是线性无关的.
- 首先检验 $x_k$ 是否是工作集上的最优解，如果不是，需要得到更新一步 $p$ ，通过求解工作集上的相关的 等式约束二次规划问题，不在工作集中的约束先不考虑。

# 积极集方法求解凸二次规划问题

- 给定当前  $x_k$ , 工作集  $\mathcal{W}_k$ , 假设在工作集中所有的  $a_i$  是线性无关的.
- 首先检验  $x_k$  是否是工作集上的最优解, 如果不是, 需要得到更新一步  $p$ , 通过求解工作集上的相关的 等式约束二次规划问题, 不在工作集中的约束先不考虑。
- 具体说来, 设  $p = x - x_k$ ,  $g_k = Gx_k + c$ . 把  $x$  代入原来的目标函数得到

$$q(x) = q(x_k + p) = \frac{1}{2}p^T Gp + g_k^T p + \rho_k, \text{ 这里 } \rho_k = \frac{1}{2}x_k^T Gx_k + c^T x_k$$

# 积极集方法求解凸二次规划问题

- 给定当前 $x_k$ , 工作集 $\mathcal{W}_k$ , 假设在工作集中所有的 $a_i$ 是线性无关的.
- 首先检验 $x_k$ 是否是工作集上的最优解, 如果不是, 需要得到更新一步 $p$ , 通过求解工作集上的相关的 等式约束二次规划问题, 不在工作集中的约束先不考虑。

- 具体说来, 设 $p = x - x_k$ ,  $g_k = Gx_k + c$ . 把 $x$ 代入原来的目标函数得到

$$q(x) = q(x_k + p) = \frac{1}{2}p^T Gp + g_k^T p + \rho_k, \text{ 这里 } \rho_k = \frac{1}{2}x_k^T Gx_k + c^T x_k$$

- 由于 $\rho_k$ 不影响求解, 因此我们可以通过求解如下的等式约束优化子问题得到 $p$

$$\begin{aligned} \min_p \quad & \frac{1}{2}p^T Gp + g_k^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$



# 积极集方法求解凸二次规划问题

- 给定当前 $x_k$ , 工作集 $\mathcal{W}_k$ , 假设在工作集中所有的 $a_i$ 是线性无关的。
- 首先检验 $x_k$ 是否是工作集上的最优解, 如果不是, 需要得到更新一步 $p$ , 通过求解工作集上的相关的 等式约束二次规划问题, 不在工作集中的约束先不考虑。

- 具体说来, 设 $p = x - x_k$ ,  $g_k = Gx_k + c$ . 把 $x$ 代入原来的目标函数得到

$$q(x) = q(x_k + p) = \frac{1}{2}p^T Gp + g_k^T p + \rho_k, \text{ 这里 } \rho_k = \frac{1}{2}x_k^T Gx_k + c^T x_k$$

- 由于 $\rho_k$ 不影响求解, 因此我们可以通过求解如下的等式约束优化子问题得到 $p$

$$\begin{aligned} \min_p \quad & \frac{1}{2}p^T Gp + g_k^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$

- 假设该问题的解是 $p_k \neq 0$ , 我们需要确定沿着该方向可以走多远。如果 $x_k + p_k$ 对于任何约束条件是可行的, 那可以接受  $x_{k+1} = x_k + p_k$ .  
否则...

# 积极集方法求解凸二次规划问题

- 否则可以设  $x_{k+1} = x_k + \alpha_k p_k$ ,  $\alpha \in (0, 1]$

# 积极集方法求解凸二次规划问题

- 否则可以设  $x_{k+1} = x_k + \alpha_k p_k$ ,  $\alpha \in (0, 1]$
- 当  $i \notin \mathcal{W}_k$ , 我们可以显式地得到  $\alpha_k$ 。因为当  $i \in \mathcal{W}_k$  时, 约束条件会自动满足 无论  $\alpha_k$  取多少。

# 积极集方法求解凸二次规划问题

- 否则可以设  $x_{k+1} = x_k + \alpha_k p_k$ ,  $\alpha \in (0, 1]$
- 当  $i \notin \mathcal{W}_k$ , 我们可以显式地得到  $\alpha_k$ 。因为当  $i \in \mathcal{W}_k$  时, 约束条件会自动满足 无论  $\alpha_k$  取多少。
- 如果  $a_i^T p_k \geq 0$  当  $i \notin \mathcal{W}_k$ , 对任意的  $\alpha_k \geq 0$ , 都有  $a_i^T (x_k + \alpha_k p_k) \geq a_i^T x_k \geq b_i$ .

# 积极集方法求解凸二次规划问题

- 否则可以设  $x_{k+1} = x_k + \alpha_k p_k$ ,  $\alpha \in (0, 1]$
- 当  $i \notin \mathcal{W}_k$ , 我们可以显式地得到  $\alpha_k$ 。因为当  $i \in \mathcal{W}_k$  时, 约束条件会自动满足 无论  $\alpha_k$  取多少。
- 如果  $a_i^T p_k \geq 0$  当  $i \notin \mathcal{W}_k$ , 对任意的  $\alpha_k \geq 0$ , 都有  $a_i^T (x_k + \alpha_k p_k) \geq a_i^T x_k \geq b_i$ 。
- 如果  $a_i^T p_k < 0$  当  $i \notin \mathcal{W}_k$ , 为了  $a_i^T (x_k + \alpha_k p_k) \geq b_i$ , 必须

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k}.$$

# 积极集方法求解凸二次规划问题

- 否则可以设  $x_{k+1} = x_k + \alpha_k p_k$ ,  $\alpha \in (0, 1]$
- 当  $i \notin \mathcal{W}_k$ , 我们可以显式地得到  $\alpha_k$ 。因为当  $i \in \mathcal{W}_k$  时, 约束条件会自动满足 无论  $\alpha_k$  取多少。
- 如果  $a_i^T p_k \geq 0$  当  $i \notin \mathcal{W}_k$ , 对任意的  $\alpha_k \geq 0$ , 都有  $a_i^T (x_k + \alpha_k p_k) \geq a_i^T x_k \geq b_i$ 。
- 如果  $a_i^T p_k < 0$  当  $i \notin \mathcal{W}_k$ , 为了  $a_i^T (x_k + \alpha_k p_k) \geq b_i$ , 必须

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k}.$$

- 因此为了保证在可行域内, 且能让目标函数最大可能地降低 我们取

$$\alpha_k = \min \left( 1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right).$$

# 积极集方法求解凸二次规划问题

- 否则可以设  $x_{k+1} = x_k + \alpha_k p_k$ ,  $\alpha \in (0, 1]$
- 当  $i \notin \mathcal{W}_k$ , 我们可以显式地得到  $\alpha_k$ 。因为当  $i \in \mathcal{W}_k$  时, 约束条件会自动满足 无论  $\alpha_k$  取多少。
- 如果  $a_i^T p_k \geq 0$  当  $i \notin \mathcal{W}_k$ , 对任意的  $\alpha_k \geq 0$ , 都有  $a_i^T (x_k + \alpha_k p_k) \geq a_i^T x_k \geq b_i$ 。
- 如果  $a_i^T p_k < 0$  当  $i \notin \mathcal{W}_k$ , 为了  $a_i^T (x_k + \alpha_k p_k) \geq b_i$ , 必须

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k}.$$

- 因此为了保证在可行域内, 且能让目标函数最大可能地降低 我们取

$$\alpha_k = \min \left( 1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right).$$

- 我们称上式中取到最小值的那个  $i$  的约束为 **blocking constrains**。

# 积极集方法求解凸二次规划问题

- 否则可以设  $x_{k+1} = x_k + \alpha_k p_k$ ,  $\alpha \in (0, 1]$
- 当  $i \notin \mathcal{W}_k$ , 我们可以显式地得到  $\alpha_k$ 。因为当  $i \in \mathcal{W}_k$  时, 约束条件会自动满足 无论  $\alpha_k$  取多少。
- 如果  $a_i^T p_k \geq 0$  当  $i \notin \mathcal{W}_k$ , 对任意的  $\alpha_k \geq 0$ , 都有  $a_i^T (x_k + \alpha_k p_k) \geq a_i^T x_k \geq b_i$ 。
- 如果  $a_i^T p_k < 0$  当  $i \notin \mathcal{W}_k$ , 为了  $a_i^T (x_k + \alpha_k p_k) \geq b_i$ , 必须

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k}.$$

- 因此为了保证在可行域内, 且能让目标函数最大可能地降低 我们取

$$\alpha_k = \min \left( 1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right).$$

- 我们称上式中取到最小值的那个  $i$  的约束为 **blocking constrains**.
- 如果  $\alpha_k = 1$ , 则没有 **blocking constrains**.



# 积极集方法求解凸二次规划问题

- 如果 $\alpha_k < 1$ , 则把block constrain加入到工作集 $\mathcal{W}_k$ , 构成新的 $\mathcal{W}_{k+1}$ .

# 积极集方法求解凸二次规划问题

- 如果  $\alpha_k < 1$ , 则把block constrain加入到工作集 $\mathcal{W}_k$ , 构成新的 $\mathcal{W}_{k+1}$ .
- 重复以上过程, 直到关于 $p$ 我们找到一个 $\hat{x}$ 能够在当前的工作集 $\hat{\mathcal{W}}$ 上极小化目标函数。

# 积极集方法求解凸二次规划问题

- 如果  $\alpha_k < 1$ , 则把block constrain加入到工作集  $\mathcal{W}_k$ , 构成新的  $\mathcal{W}_{k+1}$ .
- 重复以上过程, 直到关于  $p$  我们找到一个  $\hat{x}$  能够在当前的工作集  $\hat{\mathcal{W}}$  上极小化目标函数。
- 这个很容易校验, 因为子问题

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + g_k^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$

的解是  $p = 0$ .

# 积极集方法求解凸二次规划问题

- 如果  $\alpha_k < 1$ , 则把block constrain加入到工作集  $\mathcal{W}_k$ , 构成新的  $\mathcal{W}_{k+1}$ .
- 重复以上过程, 直到关于  $p$  我们找到一个  $\hat{x}$  能够在当前的工作集  $\hat{\mathcal{W}}$  上极小化目标函数。
- 这个很容易校验, 因为子问题

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + g_k^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$

的解是  $p = 0$ .

- 当  $p = 0$  时, 可得

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + c$$

对于某些拉格朗日乘子  $\hat{\lambda}_i$ .

# 积极集方法求解凸二次规划问题

- 如果  $\alpha_k < 1$ , 则把block constrain加入到工作集  $\mathcal{W}_k$ , 构成新的  $\mathcal{W}_{k+1}$ .
- 重复以上过程, 直到关于  $p$  我们找到一个  $\hat{x}$  能够在当前的工作集  $\hat{\mathcal{W}}$  上极小化目标函数。
- 这个很容易校验, 因为子问题

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + g_k^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$

的解是  $p = 0$ .

- 当  $p = 0$  时, 可得

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + c$$

对于某些拉格朗日乘子  $\hat{\lambda}_i$ .

- 如果不在工作集中的  $\hat{\lambda}_i$  取为 0, 当  $i \notin \hat{\mathcal{W}}$ . 那么  $(\hat{x}, \hat{\lambda}_i)$  就是满足 KKT 条件的第一个条件, 即  $\nabla L(\hat{x}, \hat{\lambda}) = 0$ .

# 积极集方法求解凸二次规划问题

- 如果  $\alpha_k < 1$ , 则把block constrain加入到工作集  $\mathcal{W}_k$ , 构成新的  $\mathcal{W}_{k+1}$ .
- 重复以上过程, 直到关于  $p$  我们找到一个  $\hat{x}$  能够在当前的工作集  $\hat{\mathcal{W}}$  上极小化目标函数。
- 这个很容易校验, 因为子问题

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + g_k^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$

的解是  $p = 0$ .

- 当  $p = 0$  时, 可得

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + c$$

对于某些拉格朗日乘子  $\hat{\lambda}_i$ .

- 如果不在工作集中的  $\hat{\lambda}_i$  取为 0, 当  $i \notin \hat{\mathcal{W}}$ . 那么  $(\hat{x}, \hat{\lambda}_i)$  就是 满足KKT条件的第一个条件, 即  $\nabla L(\hat{x}, \hat{\lambda}) = 0$ .
- 由于控制  $\alpha_k$  保持落在可行域内, 因此KKT的第二、三个条件也满足。

# 积极集方法求解凸二次规划问题

- 最后考虑 $\lambda_i$ 的符号。

# 积极集方法求解凸二次规划问题

- 最后考虑 $\lambda_i$ 的符号。
- 主要考虑工作集中的不等式约束那些, 即 $i \in \hat{\mathcal{W}} \cap \mathcal{J}$ . 如果这些 $\hat{\lambda}_i \geq 0$ , 那么KKT的第四个条件也满足了, 我们可以确定 $(\hat{x}, \hat{\lambda})$ 是满足KKT条件的。



# 积极集方法求解凸二次规划问题

- 最后考虑 $\lambda_i$ 的符号。
- 主要考虑工作集中的不等式约束那些, 即 $i \in \hat{\mathcal{W}} \cap \mathcal{J}$ . 如果这些 $\hat{\lambda}_i \geq 0$ , 那么KKT的第四个条件也满足了, 我们可以确定 $(\hat{x}, \hat{\lambda})$ 是满足KKT条件的。
- 事实上, 如果某个 $\hat{\lambda}_j < 0, j \in \hat{\mathcal{W}} \cap \mathcal{J}$ , 那么我们可以通过在工作集中去掉 $j$ , 然后重新在新的工作集上求解子问题。

# 积极集方法求解凸二次规划问题

- 最后考虑 $\lambda_i$ 的符号。
- 主要考虑工作集中的不等式约束那些, 即 $i \in \hat{\mathcal{W}} \cap \mathcal{J}$ . 如果这些 $\hat{\lambda}_i \geq 0$ , 那么KKT的第四个条件也满足了, 我们可以确定 $(\hat{x}, \hat{\lambda})$ 是满足KKT条件的。
- 事实上, 如果某个 $\hat{\lambda}_j < 0, j \in \hat{\mathcal{W}} \cap \mathcal{J}$ , 那么我们可以通过在工作集中去掉 $j$ , 然后重新在新的工作集上求解子问题。

## 定理4

假设 $\hat{x}$ 在工作集 $\hat{\mathcal{W}}$ 上满足一阶最优性条件。再假设当前工作集中所有的线性约束条件 $a_i$ 线性无关, 并且存在某个 $j$ , 对应的拉格朗日乘子 $\hat{\lambda}_j < 0$ . 假设 $p$ 是在当前工作集中去掉 $j$ 后的解 即:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (G \hat{x} + c)^T p, \\ \text{s.t.} \quad & a_i^T p = 0, \text{ for all } i \in \hat{\mathcal{W}} \text{ and } i \neq j. \end{aligned}$$

# 积极集方法求解凸二次规划问题

- 最后考虑 $\lambda_i$ 的符号。
- 主要考虑工作集中的不等式约束那些, 即 $i \in \hat{\mathcal{W}} \cap \mathcal{J}$ . 如果这些 $\hat{\lambda}_i \geq 0$ , 那么KKT的第四个条件也满足了, 我们可以确定 $(\hat{x}, \hat{\lambda})$ 是满足KKT条件的。
- 事实上, 如果某个 $\hat{\lambda}_j < 0, j \in \hat{\mathcal{W}} \cap \mathcal{J}$ , 那么我们可以通过在工作集中去掉 $j$ , 然后重新在新的工作集上求解子问题。

## 定理4

假设 $\hat{x}$ 在工作集 $\hat{\mathcal{W}}$ 上满足一阶最优性条件。再假设当前工作集中所有的线性约束条件 $a_i$ 线性无关, 并且存在某个 $j$ , 对应的拉格朗日乘子 $\hat{\lambda}_j < 0$ . 假设 $p$ 是在当前工作集中去掉 $j$ 后的解 即:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (G \hat{x} + c)^T p, \\ \text{s.t.} \quad & a_i^T p = 0, \text{ for all } i \in \hat{\mathcal{W}} \text{ and } i \neq j. \end{aligned}$$

则 $p$ 是可行的方向,  $a_j^T p \geq 0$ . 并且如果 $p$ 满足二阶的充分条件, 那么可以得到 $a_j^T p > 0$ , 并且 $p$ 是目标函数 $q(x)$ 的一个下降方向。

# 积极集方法求解凸二次规划问题

- 当 $\lambda_j < 0$ 时，说明第 $j$ 个约束并不是活跃的，目标函数可以沿着该方向继续下降。

# 积极集方法求解凸二次规划问题

- 当 $\lambda_j < 0$ 时，说明第 $j$ 个约束并不是活跃的，目标函数可以沿着该方向继续下降。
- 当有多个 $\lambda_j < 0$ 时，通常选取绝对值最大的那个，因为沿着该方向是下降最多的。

# 积极集算法实现

ALGORITHM 1: (凸二次规划的积极集方法)

初始点 $x_0$ 和初始的工作集 $\mathcal{W}_0$ ; 设 $k = 0$ ;

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点 $x_0$ 和初始的工作集 $\mathcal{W}_0$ ; 设 $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
    求解子问题得到 $p_k$

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
    求解子问题得到  $p_k$   
    **if**  $p_k = 0$



# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
    求解子问题得到  $p_k$   
    **if**  $p_k = 0$   
        计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**  
              $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**  
              $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$   
             设  $x_{k+1} \leftarrow x_k$ ,  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**  
              $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$   
             设  $x_{k+1} \leftarrow x_k$ ,  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;  
     **else**  $p_k \neq 0$ ;

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**  
              $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$   
             设  $x_{k+1} \leftarrow x_k$ ,  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;  
     **else**  $p_k \neq 0$ ;  
         计算  $\alpha_k$ ,  $x_{k+1} \leftarrow x_k + \alpha_k p_k$



# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**  
              $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$   
             设  $x_{k+1} \leftarrow x_k$ ,  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;  
     **else**  $p_k \neq 0$ ;  
         计算  $\alpha_k$ ,  $x_{k+1} \leftarrow x_k + \alpha_k p_k$   
         **if** 存在 blocking constraints;

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;

**for**  $k = 0, 1, 2, \dots$

    求解子问题得到  $p_k$

**if**  $p_k = 0$

        计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;

**if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$

**Stop** with  $x^* = x_k$ ;

**else**

$j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$

            设  $x_{k+1} \leftarrow x_k$ ,  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;

**else**  $p_k \neq 0$ ;

        计算  $\alpha_k$ ,  $x_{k+1} \leftarrow x_k + \alpha_k p_k$

**if** 存在 blocking constrains;

            Obtain  $\mathcal{W}_{k+1}$  by adding one of the blocking constrains to  $\mathcal{W}_k$

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**  
              $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$   
             设  $x_{k+1} \leftarrow x_k$ ,  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;  
     **else**  $p_k \neq 0$ ;  
         计算  $\alpha_k$ ,  $x_{k+1} \leftarrow x_k + \alpha_k p_k$   
         **if** 存在 blocking constraints;  
             Obtain  $\mathcal{W}_{k+1}$  by adding one of the blocking constraints to  $\mathcal{W}_k$   
         **else**  
              $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$

# 积极集算法实现

## ALGORITHM 1: (凸二次规划的积极集方法)

初始点  $x_0$  和初始的工作集  $\mathcal{W}_0$ ; 设  $k = 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
     求解子问题得到  $p_k$   
     **if**  $p_k = 0$   
         计算相应的拉格朗日乘子  $\hat{\lambda}_i$  with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;  
         **if**  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{J}$   
             **Stop** with  $x^* = x_k$ ;  
         **else**  
              $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$   
             设  $x_{k+1} \leftarrow x_k$ ,  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;  
     **else**  $p_k \neq 0$ ;  
         计算  $\alpha_k$ ,  $x_{k+1} \leftarrow x_k + \alpha_k p_k$   
         **if** 存在 blocking constraints;  
             Obtain  $\mathcal{W}_{k+1}$  by adding one of the blocking constraints to  $\mathcal{W}_k$   
         **else**  
              $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$   
**end for**

# 积极集算法实现

- 如何确定初始值? (“Phase I”方法,使用线性规划的解来作为二次规划的初值)

# 积极集算法实现

- 如何确定初始值? (“Phase I”方法,使用线性规划的解来作为二次规划的初值)
- 给定一个 $\tilde{x}$ , 定义如下问题

$$\begin{aligned}
 & \min_{(x,z)} \quad e^T z \\
 & s.t. \quad a_i^T x + \gamma_i z_i = b_i, \quad i \in \mathcal{E}, \\
 & \quad \quad a_i^T x + \gamma_i z_i \geq b_i, \quad i \in \mathcal{J}, \\
 & \quad \quad z \geq 0,
 \end{aligned}$$

其中 $e = (1, \dots, 1)^T$ ,  $\gamma_i = -\text{sign}(a_i^T \tilde{x} - b_i)$  for  $i \in \mathcal{E}$  and  $\gamma_i = 1$  for  $i \in \mathcal{J}$ .

# 积极集算法实现

- 如何确定初始值? (“Phase I”方法,使用线性规划的解来作为二次规划的初值)
- 给定一个 $\tilde{x}$ , 定义如下问题

$$\begin{aligned} \min_{(x,z)} \quad & e^T z \\ \text{s.t.} \quad & a_i^T x + \gamma_i z_i = b_i, \quad i \in \mathcal{E}, \\ & a_i^T x + \gamma_i z_i \geq b_i, \quad i \in \mathcal{J}, \\ & z \geq 0, \end{aligned}$$

其中 $e = (1, \dots, 1)^T$ ,  $\gamma_i = -\text{sign}(a_i^T \tilde{x} - b_i)$  for  $i \in \mathcal{E}$  and  $\gamma_i = 1$  for  $i \in \mathcal{J}$ .

- 可以通过线性规划求解上述子问题, 初始值可以选取

$$x = \tilde{x}, \quad z_i = |a_i^T \tilde{x} - b_i| (i \in \mathcal{E}), \quad z_i = \max(b_i - a_i^T \tilde{x}, 0) (i \in \mathcal{J}).$$

# 积极集算法实现

- 如何确定初始值? (“Phase I”方法,使用线性规划的解来作为二次规划的初值)
- 给定一个 $\tilde{x}$ , 定义如下问题

$$\begin{aligned} \min_{(x,z)} \quad & e^T z \\ \text{s.t.} \quad & a_i^T x + \gamma_i z_i = b_i, \quad i \in \mathcal{E}, \\ & a_i^T x + \gamma_i z_i \geq b_i, \quad i \in \mathcal{J}, \\ & z \geq 0, \end{aligned}$$

其中 $e = (1, \dots, 1)^T$ ,  $\gamma_i = -\text{sign}(a_i^T \tilde{x} - b_i)$  for  $i \in \mathcal{E}$  and  $\gamma_i = 1$  for  $i \in \mathcal{J}$ .

- 可以通过线性规划求解上述子问题, 初始值可以选取

$$x = \tilde{x}, \quad z_i = |a_i^T \tilde{x} - b_i| (i \in \mathcal{E}), \quad z_i = \max(b_i - a_i^T \tilde{x}, 0) (i \in \mathcal{J}).$$

- 可以验证, 如果 $\tilde{x}$ 是原问题可行域内的一个点, 则 $(\tilde{x}, 0)$ 是上述子问题的一个最优解。



# 积极集算法实现

- 另一种选初值的方法是通过罚函数方法。（在目标函数里增加一项，使得解落到不可行区域的测度为0）

# 积极集算法实现

- 另一种选初值的方法是通过罚函数方法。（在目标函数里增加一项，使得解落到不可行区域的测度为0）
- 引入人工变量很大的 $M$ 和 $\eta$ ，定义如下问题

$$\begin{aligned}
 \min_{x, \eta} \quad & \frac{1}{2}x^T Gx + x^T c + M\eta \\
 \text{s.t.} \quad & (a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & -(a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & b_i - a_i^T x \leq \eta, \quad i \in \mathcal{J} \\
 & 0 \leq \eta.
 \end{aligned}$$

# 积极集算法实现

- 另一种选初值的方法是通过罚函数方法。（在目标函数里增加一项，使得解落到不可行区域的测度为0）
- 引入人工变量很大的 $M$ 和 $\eta$ ，定义如下问题

$$\begin{aligned}
 \min_{x, \eta} \quad & \frac{1}{2} x^T G x + x^T c + M \eta \\
 \text{s.t.} \quad & (a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & -(a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & b_i - a_i^T x \leq \eta, \quad i \in \mathcal{J} \\
 & 0 \leq \eta.
 \end{aligned}$$

- 根据罚函数方法，当 $M$ 充分大， $\eta = 0$ 时，上述问题的解就是原问题的解。

# 积极集算法实现

- 另一种选初值的方法是通过罚函数方法。（在目标函数里增加一项，使得解落到不可行区域的测度为0）
- 引入人工变量很大的 $M$ 和 $\eta$ ，定义如下问题

$$\begin{aligned}
 \min_{x, \eta} \quad & \frac{1}{2} x^T G x + x^T c + M \eta \\
 \text{s.t.} \quad & (a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & -(a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & b_i - a_i^T x \leq \eta, \quad i \in \mathcal{J} \\
 & 0 \leq \eta.
 \end{aligned}$$

- 根据罚函数方法，当 $M$ 充分大， $\eta = 0$ 时，上述问题的解就是原问题的解。
- 我们的策略是选取一个比较大的 $M$ ，通过常规方法求解上述问题。

# 积极集算法实现

- 另一种选初值的方法是通过罚函数方法。（在目标函数里增加一项，使得解落到不可行区域的测度为0）
- 引入人工变量很大的 $M$ 和 $\eta$ ，定义如下问题

$$\begin{aligned}
 \min_{x, \eta} \quad & \frac{1}{2} x^T G x + x^T c + M \eta \\
 \text{s.t.} \quad & (a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & -(a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E} \\
 & b_i - a_i^T x \leq \eta, \quad i \in \mathcal{J} \\
 & 0 \leq \eta.
 \end{aligned}$$

- 根据罚函数方法，当 $M$ 充分大， $\eta = 0$ 时，上述问题的解就是原问题的解。
- 我们的策略是选取一个比较大的 $M$ ，通过常规方法求解上述问题。
- 如果我们找到一个解， $\eta > 0$ ，那我们可以继续增大 $M$ 继续寻找。

# 内点法



$$\begin{array}{ll} \min_x & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} & Ax \geq b. \end{array}$$

其中 $G$ 是对称半正定的矩阵， $A$ 是 $m \times n$ 的矩阵。

# 内点法



$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & Ax \geq b. \end{aligned}$$

其中 $G$ 是对称半正定的矩阵， $A$ 是 $m \times n$ 的矩阵。

## • 对应的KKT系统

$$Gx - A^T \lambda + c = 0,$$

$$Ax - b \geq 0,$$

$$(Ax - b)_i \lambda_i = 0, \quad i = 1, 2, \dots, m,$$

$$\lambda \geq 0.$$

# 内点法



$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & Ax \geq b. \end{aligned}$$

其中 $G$ 是对称半正定的矩阵， $A$ 是 $m \times n$ 的矩阵。

• 对应的KKT系统

引入变量 $y = Ax - b$

$$Gx - A^T \lambda + c = 0,$$

$$\rightarrow Gx - A^T \lambda + c = 0,$$

$$Ax - b \geq 0,$$

$$\rightarrow Ax - b - y = 0,$$

$$(Ax - b)_i \lambda_i = 0, \quad i = 1, 2, \dots, m,$$

$$\rightarrow y_i \lambda_i = 0, \quad i = 1, 2, \dots, m,$$

$$\lambda \geq 0.$$

$$\rightarrow (y, \lambda) \geq 0.$$



# 内点法: 求解主-对偶系统

- 线性规划中内点法可以推广到二次规划

## 内点法: 求解主-对偶系统

- 线性规划中内点法可以推广到二次规划
- 定义一个新的变量  $\mu = \frac{y^T \lambda}{m}$ , 再定义扰动的KKT系统

# 内点法: 求解主-对偶系统

- 线性规划中内点法可以推广到二次规划
- 定义一个新的变量  $\mu = \frac{y^T \lambda}{m}$ , 再定义扰动的KKT系统

$$F(x, y, \lambda; \sigma\mu) = \begin{bmatrix} Gx - A^T \lambda + c \\ Ax - y - b \\ \mathcal{Y} \Lambda e - \sigma\mu e \end{bmatrix} = 0$$

其中  $\sigma \in [0, 1]$ ,  $\mathcal{Y} = \text{diag}(y_1, \dots, y_m)$ ,  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m)$ ,  
 $e = (1, \dots, 1)^T$

# 内点法: 求解主-对偶系统

- 线性规划中内点法可以推广到二次规划
- 定义一个新的变量  $\mu = \frac{y^T \lambda}{m}$ , 再定义扰动的KKT系统

$$F(x, y, \lambda; \sigma\mu) = \begin{bmatrix} Gx - A^T \lambda + c \\ Ax - y - b \\ \mathcal{Y} \Lambda e - \sigma\mu e \end{bmatrix} = 0$$

其中  $\sigma \in [0, 1]$ ,  $\mathcal{Y} = \text{diag}(y_1, \dots, y_m)$ ,  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m)$ ,  $e = (1, \dots, 1)^T$

- 当  $\sigma\mu \rightarrow 0$ , 则上述问题的解趋于原二次规划问题的解。

# 内点法: 求解主-对偶系统

- 线性规划中内点法可以推广到二次规划
- 定义一个新的变量  $\mu = \frac{y^T \lambda}{m}$ , 再定义扰动的KKT系统

$$F(x, y, \lambda; \sigma\mu) = \begin{bmatrix} Gx - A^T \lambda + c \\ Ax - y - b \\ y \Lambda e - \sigma\mu e \end{bmatrix} = 0$$

其中  $\sigma \in [0, 1]$ ,  $y = \text{diag}(y_1, \dots, y_m)$ ,  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m)$ ,  
 $e = (1, \dots, 1)^T$

- 当  $\sigma\mu \rightarrow 0$ , 则上述问题的解趋于原二次规划问题的解。
- 固定  $\mu$ , 使用牛顿法求解上述系统, 我们可以得到如下的线性对偶系统

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda y e + \sigma\mu e \end{bmatrix} \quad (12.1)$$

其中  $r_d = Gx - A^T \lambda + c$ ,  $r_p = Ax - y - b$ .

# 内点法: 求解主-对偶系统

- 线性规划中内点法可以推广到二次规划
- 定义一个新的变量  $\mu = \frac{y^T \lambda}{m}$ , 再定义扰动的KKT系统

$$F(x, y, \lambda; \sigma\mu) = \begin{bmatrix} Gx - A^T \lambda + c \\ Ax - y - b \\ \mathcal{Y} \Lambda e - \sigma\mu e \end{bmatrix} = 0$$

其中  $\sigma \in [0, 1]$ ,  $\mathcal{Y} = \text{diag}(y_1, \dots, y_m)$ ,  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m)$ ,  
 $e = (1, \dots, 1)^T$

- 当  $\sigma\mu \rightarrow 0$ , 则上述问题的解趋于原二次规划问题的解。
- 固定  $\mu$ , 使用牛顿法求解上述系统, 我们可以得到如下的线性对偶系统

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda \mathcal{Y} e + \sigma\mu e \end{bmatrix} \quad (12.1)$$

其中  $r_d = Gx - A^T \lambda + c$ ,  $r_p = Ax - y - b$ .

- 迭代格式:  $(x^+, y^+, \lambda^+) = (x, y, \lambda) + \alpha(\Delta x, \Delta y, \Delta \lambda)$   
 $\alpha$  的选取使得  $(y^+, \lambda^+) > 0$

## 内点法：求解主-对偶系统

- 在内点法中，最主要的工作量是求解扰动的KKT系统  $F(x, y, \lambda; \sigma\mu) = 0$ 。

## 内点法：求解主-对偶系统

- 在内点法中，最主要的工作量是求解扰动的KKT系统  $F(x, y, \lambda; \sigma\mu) = 0$ 。
- 但是与线性规划中的内点相比，求解要更困难。可以使用直接矩阵分解法，或者使用合适的预条件求解器。



## 内点法: 求解主-对偶系统

- 在内点法中, 最主要的工作量是求解扰动的KKT系统  $F(x, y, \lambda; \sigma\mu) = 0$ 。
- 但是与线性规划中的内点相比, 求解要更困难。可以使用直接矩阵分解法, 或者使用合适的预条件求解器。
- 首先我们把相应的牛顿迭代系统写成一个紧凑的形式

$$\begin{bmatrix} G & -A^T \\ A & \Lambda^{-1}\mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p + (-y + \sigma\mu\Lambda^{-1}e) \end{bmatrix}$$

## 内点法: 求解主-对偶系统

- 在内点法中, 最主要的工作量是求解扰动的KKT系统  $F(x, y, \lambda; \sigma\mu) = 0$ 。
- 但是与线性规划中的内点相比, 求解要更困难。可以使用直接矩阵分解法, 或者使用合适的预条件求解器。
- 首先我们把相应的牛顿迭代系统写成一个紧凑的形式

$$\begin{bmatrix} G & -A^T \\ A & \Lambda^{-1}\mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p + (-y + \sigma\mu\Lambda^{-1}e) \end{bmatrix}$$

- 通过计算, 我们可以得到如下系统

$$(G + A^T\mathcal{Y}^{-1}\Lambda A)\Delta x = -r_d + A^T\mathcal{Y}^{-1}\Lambda[-r_p - y + \sigma\mu\Lambda^{-1}e].$$

## 内点法: 求解主-对偶系统

- 在内点法中, 最主要的工作量是求解扰动的KKT系统  $F(x, y, \lambda; \sigma\mu) = 0$ 。
- 但是与线性规划中的内点相比, 求解要更困难。可以使用直接矩阵分解法, 或者使用合适的预条件求解器。
- 首先我们把相应的牛顿迭代系统写成一个紧凑的形式

$$\begin{bmatrix} G & -A^T \\ A & \Lambda^{-1}\mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p + (-y + \sigma\mu\Lambda^{-1}e) \end{bmatrix}$$

- 通过计算, 我们可以得到如下系统

$$(G + A^T\mathcal{Y}^{-1}\Lambda A)\Delta x = -r_d + A^T\mathcal{Y}^{-1}\Lambda[-r_p - y + \sigma\mu\Lambda^{-1}e].$$

- 上面这个方程可以用修正Cholesky算法求解, 而且当  $A^T\mathcal{Y}^{-1}\Lambda A$  不像  $G$  那样稠密, 不等式约束较多时, 该算法更有效, 计算量会显著低于直接求解紧凑形式。

## 内点法:如何选步长?

- 在线性规划内点法中, 对于主-对偶系统选取不同的 $\alpha^{pri}$ 和 $\alpha^{dual}$ , 可以使算法更有效

## 内点法:如何选步长?

- 在线性规划内点法中, 对于主-对偶系统选取不同的 $\alpha^{pri}$ 和 $\alpha^{dual}$ , 可以使算法更有效
- 在二次规划内点法中, 我们定义如下的迭代

$$(x^+, y^+) = (x, y) + \alpha^{pri}(\Delta x, \Delta y), \lambda^+ = \lambda + \alpha^{dual} \Delta \lambda,$$

注意 $\alpha$ 的选取使得 $(x^+, y^+) > 0$ 。

## 内点法:如何选步长?

- 在线性规划内点法中, 对于主-对偶系统选取不同的 $\alpha^{pri}$ 和 $\alpha^{dual}$ , 可以使算法更有效
- 在二次规划内点法中, 我们定义如下的迭代

$$(x^+, y^+) = (x, y) + \alpha^{pri}(\Delta x, \Delta y), \lambda^+ = \lambda + \alpha^{dual} \Delta \lambda,$$

注意 $\alpha$ 的选取使得 $(x^+, y^+) > 0$ 。

- 先计算更新后的残量

$$r_p^+ = (1 - \alpha^{pri})r_p, \quad r_d^+ = (1 - \alpha^{dual})r_d + (\alpha^{pri} - \alpha^{dual})G\Delta x$$

## 内点法:如何选步长?

- 在线性规划内点法中, 对于主-对偶系统选取不同的 $\alpha^{pri}$ 和 $\alpha^{dual}$ , 可以使算法更有效
- 在二次规划内点法中, 我们定义如下的迭代

$$(x^+, y^+) = (x, y) + \alpha^{pri}(\Delta x, \Delta y), \lambda^+ = \lambda + \alpha^{dual} \Delta \lambda,$$

注意 $\alpha$ 的选取使得 $(x^+, y^+) > 0$ 。

- 先计算更新后的残量

$$r_p^+ = (1 - \alpha^{pri})r_p, \quad r_d^+ = (1 - \alpha^{dual})r_d + (\alpha^{pri} - \alpha^{dual})G\Delta x$$

- 如果 $\alpha^{pri} = \alpha^{dual} = \alpha$ , 则两个残量都线性下降。

## 内点法:如何选步长?

- 在线性规划内点法中, 对于主-对偶系统选取不同的 $\alpha^{pri}$ 和 $\alpha^{dual}$ , 可以使算法更有效
- 在二次规划内点法中, 我们定义如下的迭代

$$(x^+, y^+) = (x, y) + \alpha^{pri}(\Delta x, \Delta y), \lambda^+ = \lambda + \alpha^{dual} \Delta \lambda,$$

注意 $\alpha$ 的选取使得 $(x^+, y^+) > 0$ 。

- 先计算更新后的残量

$$r_p^+ = (1 - \alpha^{pri})r_p, \quad r_d^+ = (1 - \alpha^{dual})r_d + (\alpha^{pri} - \alpha^{dual})G\Delta x$$

- 如果 $\alpha^{pri} = \alpha^{dual} = \alpha$ , 则两个残量都线性下降。
- 如果取不一样的值, 有可能会导导致其中对偶系统的残量上升, 进而导致算法发散。



## 内点法:如何选步长?

- 一种选择是选取相同的步长

$$\alpha = \min(\alpha_{\tau}^{pri}, \alpha_{\tau}^{dual}),$$

$$\alpha_{\tau}^{pri} = \max\{\alpha \in (0, 1] : y + \alpha\Delta y \geq (1 - \tau)y\},$$

$$\alpha_{\tau}^{dual} = \max\{\alpha \in (0, 1] : \lambda + \alpha\Delta\lambda \geq (1 - \tau)\lambda\},$$

其中  $\tau \in (0, 1)$ , 是用于控制我们到最大步长的距离(是满足  $y + \alpha\Delta y \geq 0$  和  $\lambda + \alpha\Delta\lambda \geq 0$  的最大  $\alpha$ )。

## 内点法:如何选步长?

- 一种选择是选取相同的步长

$$\begin{aligned}\alpha &= \min(\alpha_{\tau}^{pri}, \alpha_{\tau}^{dual}), \\ \alpha_{\tau}^{pri} &= \max\{\alpha \in (0, 1] : y + \alpha\Delta y \geq (1 - \tau)y\}, \\ \alpha_{\tau}^{dual} &= \max\{\alpha \in (0, 1] : \lambda + \alpha\Delta\lambda \geq (1 - \tau)\lambda\},\end{aligned}$$

其中  $\tau \in (0, 1)$ , 是用于控制我们到最大步长的距离(是满足  $y + \alpha\Delta y \geq 0$  和  $\lambda + \alpha\Delta\lambda \geq 0$  的最大  $\alpha$ )。

- 但是一般的数值例子显示, 对主-对偶系统选取不同的  $\alpha$ , 可以得到更快的收敛性。

## 内点法:如何选步长?

- 一种选择是选取相同的步长

$$\begin{aligned}\alpha &= \min(\alpha_{\tau}^{pri}, \alpha_{\tau}^{dual}), \\ \alpha_{\tau}^{pri} &= \max\{\alpha \in (0, 1] : y + \alpha\Delta y \geq (1 - \tau)y\}, \\ \alpha_{\tau}^{dual} &= \max\{\alpha \in (0, 1] : \lambda + \alpha\Delta\lambda \geq (1 - \tau)\lambda\},\end{aligned}$$

其中  $\tau \in (0, 1)$ , 是用于控制我们到最大步长的距离(是满足  $y + \alpha\Delta y \geq 0$  和  $\lambda + \alpha\Delta\lambda \geq 0$  的最大  $\alpha$ )。

- 但是一般的数值例子显示, 对主-对偶系统选取不同的  $\alpha$ , 可以得到更快的收敛性。
- 一种选择不同步长的策略

$$\begin{aligned}\min \quad & \|Gx^+ - A^T\lambda^+ + c\|_2^2 + \|Ax^+ - y^+ - b\|_2^2 + (y^+)^T z^+ \\ \text{s.t.} \quad & 0 \leq \alpha^{pri} \leq \alpha_{\tau}^{pri}, \quad 0 \leq \alpha^{dual} \leq \alpha_{\tau}^{dual}\end{aligned}$$

# 内点法:实用的主-对偶算法

- 目前最有效的算法是预估-校正算法（起源于线性规划、推广到二次规划）

# 内点法:实用的主-对偶算法

- 目前最有效的算法是预估-校正算法（起源于线性规划、推广到二次规划）
- 首先设 $\sigma = 0$ ，计算一个仿射伸缩步( $\Delta x^{aff}, \Delta y^{aff}, \Delta \lambda^{aff}$ )

# 内点法:实用的主-对偶算法

- 目前最有效的算法是预估-校正算法（起源于线性规划、推广到二次规划）
- 首先设 $\sigma = 0$ ，计算一个仿射伸缩步 $(\Delta x^{aff}, \Delta y^{aff}, \Delta \lambda^{aff})$
- 然后基于此，再计算校正系统：

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda y_e - \Delta \lambda^{aff} \Delta y^{aff} e + \sigma \mu e \end{bmatrix} \quad (12.2)$$

# 内点法:实用的主-对偶算法

## Algorithm 2: QP的预估-校正算法

计算  $(x_0, y_0, \lambda_0)$  with  $(y_0, \lambda_0) > 0$ ;

**for**  $k = 0, 1, 2, \dots$

Set  $(x, y, \lambda) = (x_k, y_k, \lambda_k)$ , 求解主-对偶系统(12.1) with  $\sigma = 0$ ,  
得到  $(\Delta x^{aff}, \Delta y^{aff}, \Delta \lambda^{aff})$

计算  $\mu = \frac{y^T \lambda}{m}$

计算  $\hat{\alpha}_{aff} = \max\{\alpha \in (0, 1] : (y, \lambda) + \alpha(\Delta y^{aff}, \Delta \lambda^{aff}) \geq 0\}$ ;

计算  $\mu_{aff} = (y + \hat{\alpha}^{aff} \Delta y^{aff})^T (\lambda + \hat{\alpha}^{aff} \Delta \lambda^{aff}) / m$ ;

Set  $\sigma = (\mu_{aff} / \mu)^3$ ;

求解上述的校正系统(12.2)得到  $(\Delta x, \Delta y, \Delta \lambda)$ ;

选择  $\tau_k \in (0, 1)$ , set  $\hat{\alpha} = \min(\alpha_{\tau_k}^{pri}, \alpha_{\tau_k}^{dual})$ ;

Set  $(x_{k+1}, y_{k+1}, \lambda_{k+1}) = (x_k, y_k, \lambda_k) + \hat{\alpha}(\Delta x, \Delta y, \Delta \lambda)$ ;

**end (for)**

# 内点法:实用的主-对偶算法

- 可以让  $\tau_k \rightarrow 1$ , 加速算法的收敛性。



# 内点法:实用的主-对偶算法

- 可以让  $\tau_k \rightarrow 1$ , 加速算法的收敛性。
- 如何选取初值?

# 内点法:实用的主-对偶算法

- 可以让  $\tau_k \rightarrow 1$ , 加速算法的收敛性。
- 如何选取初值?
  - 对于任意给定的初值  $(\bar{x}, \bar{y}, \bar{\lambda})$ , 首先计算仿射伸缩步  $(\Delta x^{aff}, \Delta y^{aff}, \Delta \lambda^{aff})$

# 内点法:实用的主-对偶算法

- 可以让  $\tau_k \rightarrow 1$ , 加速算法的收敛性。
- 如何选取初值?
  - 对于任意给定的初值  $(\bar{x}, \bar{y}, \bar{\lambda})$ , 首先计算仿射伸缩步  $(\Delta x^{aff}, \Delta y^{aff}, \Delta \lambda^{aff})$
  - Set

$$y_0 = \max(1, |\bar{y} + \Delta y^{aff}|),$$

$$\lambda_0 = \max(1, |\bar{\lambda} + \Delta \lambda^{aff}|),$$

$$x_0 = \bar{x}.$$

# 活跃集方法与内点法的比较

# 活跃集方法与内点法的比较

- 活跃集方法每一步计算量比较小，但需要很多步。

# 活跃集方法与内点法的比较

- 活跃集方法每一步计算量比较小，但需要很多步。  
内点法每一步需要比较多的计算量，但总的迭代步数较少。

# 活跃集方法与内点法的比较

- 活跃集方法每一步计算量比较小，但需要很多步。  
内点法每一步需要比较大的计算量，但总的迭代步数较少。
- 活跃集方法通常比较难实现，特别是求解过程中会更新矩阵分解，而且利用 $A$ 和 $G$ 的稀疏性。

# 活跃集方法与内点法的比较

- 活跃集方法每一步计算量比较小，但需要很多步。  
内点法每一步需要比较多的计算量，但总的迭代步数较少。
- 活跃集方法通常比较难实现，特别是求解过程中会更新矩阵分解，而且要利用 $A$ 和 $G$ 的稀疏性。  
在内点法中，不会更新矩阵分解，如果 $G$ 和 $A$ 有特殊的稀疏结构，那么可以使用高效的线性系统求解器。



# 活跃集方法与内点法的比较

- 活跃集方法每一步计算量比较小，但需要很多步。  
内点法每一步需要比较多的计算量，但总的迭代步数较少。
- 活跃集方法通常比较难实现，特别是求解过程中会更新矩阵分解，而且要利用 $A$ 和 $G$ 的稀疏性。  
在内点法中，不会更新矩阵分解，如果 $G$ 和 $A$ 有特殊的稀疏结构，那么可以使用高效的线性系统求解器。
- 对于大规模系统，内点法的效率要更高。

# 活跃集方法与内点法的比较

- 活跃集方法每一步计算量比较小，但需要很多步。  
内点法每一步需要比较多的计算量，但总的迭代步数较少。
- 活跃集方法通常比较难实现，特别是求解过程中会更新矩阵分解，而且要利用 $A$ 和 $G$ 的稀疏性。  
在内点法中，不会更新矩阵分解，如果 $G$ 和 $A$ 有特殊的稀疏结构，那么可以使用高效的线性系统求解器。
- 对于大规模系统，内点法的效率要更高。
- 如果初值选取的很好，那么活跃集方法的收敛要更快。

# 梯度投影法

- 活跃集方法每次只能更新一个约束条件，当约束条件规模较大的时候，收敛较慢。

# 梯度投影法

- 活跃集方法每次只能更新一个约束条件，当约束条件规模较大的时候，收敛较慢。
- 梯度投影法可以快速更新活跃集。特别是对一些简单的不等式约束条件。

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & l \leq x \leq u. \end{aligned}$$

# 梯度投影法

- 活跃集方法每次只能更新一个约束条件，当约束条件规模较大的时候，收敛较慢。
- 梯度投影法可以快速更新活跃集。特别是对一些简单的不等式约束条件。

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & l \leq x \leq u. \end{aligned}$$

- 不需要假设 $G$ 的半正定性，梯度投影法适用于不定情形。

# 梯度投影法

- 活跃集方法每次只能更新一个约束条件，当约束条件规模较大的时候，收敛较慢。
- 梯度投影法可以快速更新活跃集。特别是对一些简单的不等式约束条件。

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & l \leq x \leq u. \end{aligned}$$

- 不需要假设 $G$ 的半正定性，梯度投影法适用于不定情形。
- 可行区域是矩形区域。如果有些分量没有下界或上界，默认为 $\infty$ 。

# 梯度投影法

- 活跃集方法每次只能更新一个约束条件，当约束条件规模较大的时候，收敛较慢。
- 梯度投影法可以快速更新活跃集。特别是对一些简单的不等式约束条件。

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & l \leq x \leq u. \end{aligned}$$

- 不需要假设 $G$ 的半正定性，梯度投影法适用于不定情形。
- 可行区域是矩形区域。如果有些分量没有下界或上界，默认为 $\infty$ 。
- 在梯度投影法中，每一步迭代包含两步

# 梯度投影法

- 活跃集方法每次只能更新一个约束条件，当约束条件规模较大的时候，收敛较慢。
- 梯度投影法可以快速更新活跃集。特别是对一些简单的不等式约束条件。

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & l \leq x \leq u. \end{aligned}$$

- 不需要假设 $G$ 的半正定性，梯度投影法适用于不定情形。
- 可行区域是矩形区域。如果有些分量没有下界或上界，默认为 $\infty$ 。
- 在梯度投影法中，每一步迭代包含两步
  - 从当前点 $x$ 出发，沿着最速下降方向 $-g = -(Gx + c)$ 搜索，当遇到边界，为了留在可行区域内，搜索方向会“弯曲”。继续沿着得到的分片线性路径，找到第一个局部极小值点，记为Cauchy点 $x^c$ 。此时 $x^c$ 处的活跃的约束集记为工作集 $\mathcal{A}(x^c)$ 。



# 梯度投影法

- 活跃集方法每次只能更新一个约束条件，当约束条件规模较大的时候，收敛较慢。
- 梯度投影法可以快速更新活跃集。特别是对一些简单的不等式约束条件。

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & l \leq x \leq u. \end{aligned}$$

- 不需要假设 $G$ 的半正定性，梯度投影法适用于不定情形。
- 可行区域是矩形区域。如果有些分量没有下界或上界，默认为 $\infty$ 。
- 在梯度投影法中，每一步迭代包含两步
  - 从当前点 $x$ 出发，沿着最速下降方向 $-g = -(Gx + c)$ 搜索，当遇到边界，为了留在可行区域内，搜索方向会“弯曲”。继续沿着得到的分片线性路径，找到第一个局部极小值点，记为Cauchy点 $x^c$ 。此时 $x^c$ 处的活跃的约束集记为工作集 $\mathcal{A}(x^c)$ 。
  - 通过求解活跃的分量 $x_i, i \in \mathcal{A}(x^c)$ ，确定Cauchy点落在哪个面上。

# 梯度投影法：计算Cauchy点

- 我们可以显式地计算出分片线性路径

# 梯度投影法：计算Cauchy点

- 我们可以显式地计算出分片线性路径
- 定义如下的投影算子

$$P(x, l, u)_i = \begin{cases} l_i & \text{if } x_i < l_i, \\ x_i & \text{if } x_i \in [l_i, u_i], \\ u_i & \text{if } x_i > u_i, \end{cases}$$

# 梯度投影法：计算Cauchy点

- 我们可以显式地计算出分片线性路径
- 定义如下的投影算子

$$P(x, l, u)_i = \begin{cases} l_i & \text{if } x_i < l_i, \\ x_i & \text{if } x_i \in [l_i, u_i], \\ u_i & \text{if } x_i > u_i, \end{cases}$$

- 定义分片线性路径

$$x(t) = P(x - tg, l, u)$$

# 梯度投影法：计算Cauchy点

- 我们可以显式地计算出分片线性路径
- 定义如下的投影算子

$$P(x, l, u)_i = \begin{cases} l_i & \text{if } x_i < l_i, \\ x_i & \text{if } x_i \in [l_i, u_i], \\ u_i & \text{if } x_i > u_i, \end{cases}$$

- 定义分片线性路径

$$x(t) = P(x - tg, l, u)$$

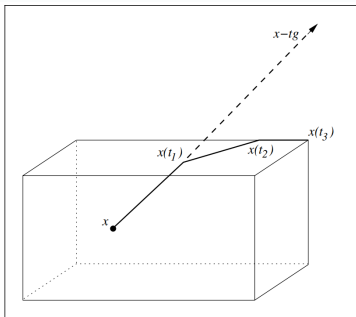
# 梯度投影法：计算Cauchy点

- 我们可以显式地计算出分片线性路径
- 定义如下的投影算子

$$P(x, l, u)_i = \begin{cases} l_i & \text{if } x_i < l_i, \\ x_i & \text{if } x_i \in [l_i, u_i], \\ u_i & \text{if } x_i > u_i, \end{cases}$$

- 定义分片线性路径

$$x(t) = P(x - tg, l, u)$$



定义：Cauchy点

分片二次函数 $q(x(t))$ ,  $t > 0$ 的第一个局部极小值点

# 梯度投影法：计算Cauchy点

- 通过检查每一段 $x(t)$ 来得到极小值点。

# 梯度投影法：计算Cauchy点

- 通过检查每一段 $x(t)$ 来得到极小值点。
- 首先确定间断点(breakpoints)，可以显式地得到

$$\bar{t}_i = \begin{cases} (x_i - u_i)/g_i & \text{if } g_i < 0 \text{ and } u_i < +\infty, \\ (x_i - l_i)/g_i & \text{if } g_i > 0 \text{ and } l_i > -\infty, \\ \infty & \text{otherwise.} \end{cases}$$



# 梯度投影法：计算Cauchy点

- 通过检查每一段 $x(t)$ 来得到极小值点。
- 首先确定间断点(breakpoints)，可以显式地得到

$$\bar{t}_i = \begin{cases} (x_i - u_i)/g_i & \text{if } g_i < 0 \text{ and } u_i < +\infty, \\ (x_i - l_i)/g_i & \text{if } g_i > 0 \text{ and } l_i > -\infty, \\ \infty & \text{otherwise.} \end{cases}$$

- 因此对任意的 $t$ ，可以得到

$$x_i(t) = \begin{cases} x_i - tg_i & \text{if } t \leq \bar{t}_i, \\ x_i - \bar{t}_i g_i & \text{otherwise.} \end{cases}$$

# 梯度投影法：计算Cauchy点

## 梯度投影法：计算Cauchy点

- 把 $\{\bar{t}_1, \dots, \bar{t}_n\}$ 中重复的 $\bar{t}_i$ 以及0去除掉，得到缩减的间断点集合 $\{t_1, \dots, t_l\}$ 满足 $0 < t_1 < t_2 < \dots$ .

# 梯度投影法：计算Cauchy点

- 把 $\{\bar{t}_1, \dots, \bar{t}_n\}$ 中重复的 $\bar{t}_i$ 以及0去除掉，得到缩减的间断点集合 $\{t_1, \dots, t_l\}$ 满足 $0 < t_1 < t_2 < \dots$ .
- 逐个 $[0, t_1], [t_1, t_2], \dots$ 检验。假定我们已经检查到 $t_{j-1}$ ，并且还没招到局部极小值，那么针对 $[t_{j-1}, t_j]$ ，我们有

$$\begin{aligned}
 x(t) &= x(t_{j-1}) + \Delta t p^{j-1} \\
 \Delta t &= t - t_{j-1} \in [0, t_j - t_{j-1}] \\
 p_i^{j-1} &= \begin{cases} -g_i & \text{if } t_{j-1} < \bar{t}_i, \\ 0 & \text{otherwise.} \end{cases}
 \end{aligned}$$

## 梯度投影法：计算Cauchy点

- 把 $\{\bar{t}_1, \dots, \bar{t}_n\}$ 中重复的 $\bar{t}_i$ 以及0去除掉，得到缩减的间断点集合 $\{t_1, \dots, t_l\}$ 满足 $0 < t_1 < t_2 < \dots$ .
- 逐个 $[0, t_1], [t_1, t_2], \dots$ 检验。假定我们已经检查到 $t_{j-1}$ ，并且还没招到局部极小值，那么针对 $[t_{j-1}, t_j]$ ，我们有

$$\begin{aligned} x(t) &= x(t_{j-1}) + \Delta t p^{j-1} \\ \Delta t &= t - t_{j-1} \in [0, t_j - t_{j-1}] \\ p_i^{j-1} &= \begin{cases} -g_i & \text{if } t_{j-1} < \bar{t}_i, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- 在每个线段上 $[x(t_{j-1}), x(t_j)]$ 上，我们有

$$q(x(t)) = c^T(x(t)) + \frac{1}{2}(x(t))^T G(x(t)), \text{ with } (x(t)) = x(t_{j-1}) + (\Delta t)p^{j-1}$$

# 梯度投影法：计算Cauchy点

- 展开后根据 $\Delta t$ 的幂次合并同类项得到

# 梯度投影法：计算Cauchy点

- 展开后根据 $\Delta t$ 的幂次合并同类项得到

$$q(x(t)) = f_{j-1} + f'_{j-1}\Delta + \frac{1}{2}f''_{j-1}(\Delta t)^2, \Delta t \in [0, t_j - t_{j-1}]$$

$$f_{j-1} = c^T x(t_{j-1}) + \frac{1}{2}x(t_{j-1})^T G x(t_{j-1}),$$

$$f'_{j-1} = c^T p^{j-1} + x(t_{j-1})^T G p^{j-1},$$

$$f''_{j-1} = (p^{j-1})^T G p^{j-1}.$$

# 梯度投影法：计算Cauchy点

- 展开后根据 $\Delta t$ 的幂次合并同类项得到

$$q(x(t)) = f_{j-1} + f'_{j-1}\Delta + \frac{1}{2}f''_{j-1}(\Delta t)^2, \quad \Delta t \in [0, t_j - t_{j-1}]$$

$$f_{j-1} = c^T x(t_{j-1}) + \frac{1}{2}x(t_{j-1})^T G x(t_{j-1}),$$

$$f'_{j-1} = c^T p^{j-1} + x(t_{j-1})^T G p^{j-1},$$

$$f''_{j-1} = (p^{j-1})^T G p^{j-1}.$$

- 对 $\Delta t$ 求导并设为0，可以得到

$$\Delta t^* = -\frac{f'_{j-1}}{f''_{j-1}}.$$



# 梯度投影法：计算Cauchy点

- 展开后根据 $\Delta t$ 的幂次合并同类项得到

$$q(x(t)) = f_{j-1} + f'_{j-1}\Delta + \frac{1}{2}f''_{j-1}(\Delta t)^2, \Delta t \in [0, t_j - t_{j-1}]$$

$$f_{j-1} = c^T x(t_{j-1}) + \frac{1}{2}x(t_{j-1})^T G x(t_{j-1}),$$

$$f'_{j-1} = c^T p^{j-1} + x(t_{j-1})^T G p^{j-1},$$

$$f''_{j-1} = (p^{j-1})^T G p^{j-1}.$$

- 对 $\Delta t$ 求导并设为0，可以得到

$$\Delta t^* = -\frac{f'_{j-1}}{f''_{j-1}}.$$

- (1) 如果 $f'_{j-1} > 0$ ，在 $t = t_{j-1}$ 处取到极小值点，

# 梯度投影法：计算Cauchy点

- 展开后根据 $\Delta t$ 的幂次合并同类项得到

$$q(x(t)) = f_{j-1} + f'_{j-1}\Delta + \frac{1}{2}f''_{j-1}(\Delta t)^2, \Delta t \in [0, t_j - t_{j-1}]$$

$$f_{j-1} = c^T x(t_{j-1}) + \frac{1}{2}x(t_{j-1})^T G x(t_{j-1}),$$

$$f'_{j-1} = c^T p^{j-1} + x(t_{j-1})^T G p^{j-1},$$

$$f''_{j-1} = (p^{j-1})^T G p^{j-1}.$$

- 对 $\Delta t$ 求导并设为0，可以得到

$$\Delta t^* = -\frac{f'_{j-1}}{f''_{j-1}}.$$

- (1) 如果 $f'_{j-1} > 0$ ，在 $t = t_{j-1}$ 处取到极小值点，
- (2)  $\Delta t^* \in [0, t_j - t_{j-1}]$ ，在 $t = t_{j-1} + \Delta t^*$ 处取到极小值，

# 梯度投影法：计算Cauchy点

- 展开后根据 $\Delta t$ 的幂次合并同类项得到

$$q(x(t)) = f_{j-1} + f'_{j-1}\Delta + \frac{1}{2}f''_{j-1}(\Delta t)^2, \Delta t \in [0, t_j - t_{j-1}]$$

$$f_{j-1} = c^T x(t_{j-1}) + \frac{1}{2}x(t_{j-1})^T Gx(t_{j-1}),$$

$$f'_{j-1} = c^T p^{j-1} + x(t_{j-1})^T Gp^{j-1},$$

$$f''_{j-1} = (p^{j-1})^T Gp^{j-1}.$$

- 对 $\Delta t$ 求导并设为0，可以得到

$$\Delta t^* = -\frac{f'_{j-1}}{f''_{j-1}}.$$

- (1) 如果 $f'_{j-1} > 0$ ，在 $t = t_{j-1}$ 处取到极小值点，
- (2)  $\Delta t^* \in [0, t_j - t_{j-1}]$ ，在 $t = t_{j-1} + \Delta t^*$ 处取到极小值，
- (3) 其他情况不存在极小值，继续搜索 $[t_j, t_{j+1}]$ 。

## 梯度投影法：计算Cauchy点

- 展开后根据 $\Delta t$ 的幂次合并同类项得到

$$q(x(t)) = f_{j-1} + f'_{j-1}\Delta + \frac{1}{2}f''_{j-1}(\Delta t)^2, \quad \Delta t \in [0, t_j - t_{j-1}]$$

$$f_{j-1} = c^T x(t_{j-1}) + \frac{1}{2}x(t_{j-1})^T G x(t_{j-1}),$$

$$f'_{j-1} = c^T p^{j-1} + x(t_{j-1})^T G p^{j-1},$$

$$f''_{j-1} = (p^{j-1})^T G p^{j-1}.$$

- 对 $\Delta t$ 求导并设为0，可以得到

$$\Delta t^* = -\frac{f'_{j-1}}{f''_{j-1}}.$$

- (1) 如果 $f'_{j-1} > 0$ ，在 $t = t_{j-1}$ 处取到极小值点，
  - (2)  $\Delta t^* \in [0, t_j - t_{j-1}]$ ，在 $t = t_{j-1} + \Delta t^*$ 处取到极小值，
  - (3) 其他情况不存在极小值，继续搜索 $[t_j, t_{j+1}]$ 。
- 在下一段中，重新计算新的方向得到 $p^j$ ，并计算 $f_j, f'_j, f''_j$ 。计算量不大。

# 梯度投影法：子空间中优化

- 第一步计算得到了Cauchy点后，记为 $x^c$ ，可以得到相应的活跃集

$$\mathcal{A}(x^c) = \{i | x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

# 梯度投影法：子空间中优化

- 第一步计算得到了Cauchy点后，记为 $x^c$ ，可以得到相应的活跃集

$$\mathcal{A}(x^c) = \{i | x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

- 剩下的分量可以通过求解如下子问题系统

# 梯度投影法：子空间中优化

- 第一步计算得到了Cauchy点后，记为 $x^c$ ，可以得到相应的活跃集

$$\mathcal{A}(x^c) = \{i | x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

- 剩下的分量可以通过求解如下子问题系统

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & x_i = x_i^c, \quad i \in \mathcal{A}(x^c), \\ & l_i \leq x \leq u_i, \quad i \notin \mathcal{A}(x^c). \end{aligned}$$

# 梯度投影法：子空间中优化

- 第一步计算得到了Cauchy点后，记为 $x^c$ ，可以得到相应的活跃集

$$\mathcal{A}(x^c) = \{i | x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

- 剩下的分量可以通过求解如下子问题系统

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & x_i = x_i^c, \quad i \in \mathcal{A}(x^c), \\ & l_i \leq x \leq u_i, \quad i \notin \mathcal{A}(x^c). \end{aligned}$$

- 我们并不精确求解上述子问题。因为针对不等式约束问题的难度，几乎与原问题一样。



# 梯度投影法：子空间中优化

- 第一步计算得到了Cauchy点后，记为 $x^c$ ，可以得到相应的活跃集

$$\mathcal{A}(x^c) = \{i | x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

- 剩下的分量可以通过求解如下子问题系统

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & x_i = x_i^c, \quad i \in \mathcal{A}(x^c), \\ & l_i \leq x \leq u_i, \quad i \notin \mathcal{A}(x^c). \end{aligned}$$

- 我们并不精确求解上述子问题。因为针对不等式约束问题的难度，几乎与原问题一样。
- 事实上，我们只需要寻找一个可行的 $x^+$ ，满足  $q(x^+) \leq q(x^c)$  即可。

# 梯度投影法：子空间中优化

- 第一步计算得到了Cauchy点后，记为 $x^c$ ，可以得到相应的活跃集

$$\mathcal{A}(x^c) = \{i | x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

- 剩下的分量可以通过求解如下子问题系统

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & x_i = x_i^c, \quad i \in \mathcal{A}(x^c), \\ & l_i \leq x \leq u_i, \quad i \notin \mathcal{A}(x^c). \end{aligned}$$

- 我们并不精确求解上述子问题。因为针对不等式约束问题的难度，几乎与原问题一样。
- 事实上，我们只需要寻找一个可行的 $x^+$ ，满足 $q(x^+) \leq q(x^c)$ 即可。
- 一个策略是介于取最简单 $x^+ = x^c$ 和精确求解子问题系统中间的方法。可以用CG法求解上述等式约束优化问题，只要满足不等式就停止。

# 梯度投影法：子空间中优化

- 第一步计算得到了Cauchy点后，记为 $x^c$ ，可以得到相应的活跃集

$$\mathcal{A}(x^c) = \{i | x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

- 剩下的分量可以通过求解如下子问题系统

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & x_i = x_i^c, \quad i \in \mathcal{A}(x^c), \\ & l_i \leq x \leq u_i, \quad i \notin \mathcal{A}(x^c). \end{aligned}$$

- 我们并不精确求解上述子问题。因为针对不等式约束问题的难度，几乎与原问题一样。
- 事实上，我们只需要寻找一个可行的 $x^+$ ，满足 $q(x^+) \leq q(x^c)$ 即可。
- 一个策略是介于取最简单 $x^+ = x^c$ 和精确求解子问题系统中间的方法。可以用CG法求解上述等式约束优化问题，只要满足不等式就停止。
- 或者是不用考虑不等式约束，求解后直接投影到可行区域内。

# 梯度投影法算法实现

## ALGORITHM 3: GRADIENT PROJECTION METHOD FOR QP

计算一个可行的初始点  $x_0$ ;

**for**  $k = 0, 1, 2, \dots$

**if**  $x_k$  满足KKT条件

**stop** with solution  $x^* = x_k$  ;

    Set  $x = x_k$  and find the Cauchy point  $x^c$ ;

    Find an approximate solution  $x^+$ , such that  $q(x^+) \leq q(x^c)$ ;  
        and  $x^+$  is feasible;

$x_{k+1} \leftarrow x^+$ ;

**end (for)**

# 梯度投影法

- 梯度投影法原则上也可应用于一般线性约束，但需要的计算量更大。

# 梯度投影法

- 梯度投影法原则上也可应用于一般线性约束，但需要的计算量更大。
- 例如，当线性约束变成  $a_i^T x \geq b_i$ ，对于给定的  $\bar{x}$  需要求解如下的子问题

$$\max_x \|x - \bar{x}\| \quad \text{subject to } a_i^T x \geq b_i, \text{ for all } i \in \mathcal{I}.$$

# 梯度投影法

- 梯度投影法原则上也可应用于一般线性约束，但需要的计算量更大。
- 例如，当线性约束变成  $a_i^T x \geq b_i$ ，对于给定的  $\bar{x}$  需要求解如下的子问题

$$\max_x \|x - \bar{x}\| \quad \text{subject to } a_i^T x \geq b_i, \text{ for all } i \in \mathcal{I}.$$

- 求解这个子问题的难度几乎与原问题相当，所以在这种情况下很少使用梯度投影法。

THANKS FOR YOUR ATTENTION