面试中常见的凸优化相关问题的整理

优化问题的基本形式

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

无约束优化问题

线搜索

梯度法

牛顿法

带约束优化问题

对偶问题

Lagrange 对偶函数和对偶问题

拉格朗日对偶函数是优化问题中引入的一个概念，旨在通过转换原始问题来简化求解过程。它是拉格朗日乘子法在凸优化中的推广，尤其适用于处理带约束的优化问题。

考虑一个带有约束的标准优化问题：

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \cdots, m, \\ & h_j(x) = 0, \quad j = 1, \cdots, n. \end{aligned}$$

其中，$f(x)$ 是目标函数，$g_i(x)$ 是不等式约束，$h_j(x)$ 是等式约束。

拉格朗日函数（Lagrangian Function）通过引入拉格朗日乘子将约束条件整合到目标函数中，形式如下：

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_i \lambda_i g_i(x) + \sum_j \nu_j h_j(x)$$

其中，λ_i 是不等式约束对应的拉格朗日乘子，ν_j 是等式约束对应的拉格朗日乘子。

为了得到拉格朗日对偶函数，我们对拉格朗日函数在 x 上取极小值(下确界)：

$$g(\lambda, \nu) = \inf_x \mathcal{L}(x, \lambda, \nu)$$

这个函数 $g(\lambda, \nu)$ 就是拉格朗日对偶函数。 它表示对于任意的拉格朗日乘子组合 (λ, ν)，目标函数在考虑约束条件的情况下能够达到的最小值。

拉格朗日对偶函数的重要性在于，它将原始优化问题转换成一个对偶问题：

$$\begin{aligned} \max_{\lambda>0, \nu} \quad & g(\lambda, \nu) \\ \text{s.t.} \quad & \lambda_i \geq 0, \quad i = 1, \cdots, m. \end{aligned}$$

- 即使原问题不是凸的，对偶函数一定是凹的。

对偶问题通常比原始问题更容易求解，特别是在凸优化问题中，原始问题与对偶问题的最优解在某些条件下是一致的（即存在“强对偶性”），这意味着可以通过求解对偶问题来间接求解原问题。

总结来说，拉格朗日对偶函数是一种通过引入乘子并考虑约束条件的最小化，将复杂的优化问题转化为易处理的对偶形式，从而简化求解过程的工具。

弱对偶性和对偶间隙

弱对偶性和**对偶间隙**是优化理论中对原始问题和对偶问题之间关系的两个重要概念。

弱对偶性表明，对于任意一个优化问题，**对偶问题的最优值始终小于或等于原问题的最优值**。这一性质普遍适用于带有约束的优化问题。

具体来说，假设我们有以下优化问题：

- 原问题（也称为主问题）：

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \cdots, m, \\ & h_j(x) = 0, \quad j = 1, \cdots, n, \end{aligned}$$

- 对偶问题：

$$\begin{aligned} \max \quad & g(\lambda, \nu) \\ \text{s.t.} \quad & \lambda_i \geq 0, \quad i = 1, \cdots, m, \end{aligned}$$

在这种情况下，弱对偶性可以表示为：

$$g(\lambda^*, \nu^*) \leq f(x^*),$$

其中，x^* 是原问题的最优解，(λ^*, ν^*) 是对偶问题的最优解。即对偶问题所得到的最优值（最大值）不会超过原问题的最优值（最小值）。

弱对偶性对任何优化问题都成立，即使没有满足一些特殊条件（如凸性）。

对偶间隙（Duality Gap）是指原问题的最优值与对偶问题的最优值之间的差异。数学上，若原问题的最优值是 $f(x^*)$，对偶问题的最优值是 $g(\lambda^*, \nu^*)$，那么对偶间隙定义为：

$$\text{对偶间隙} = f(x^*) - g(\lambda^*, \nu^*)$$

如果对偶间隙为0，则意味着原问题和对偶问题的最优值相等，此时存在所谓的**强对偶性**。即：

$$f(x^*) = g(\lambda^*, \nu^*)$$

总结：

- **弱对偶性**：对偶问题的最优值永远小于或等于原问题的最优值。
- **对偶间隙**：原问题最优值与对偶问题最优值之间的差距，若对偶间隙为0，则表明存在强对偶性。

弱对偶性在所有优化问题中都成立，而对偶间隙则描述了原问题和对偶问题之间的差异，当对偶间隙为0时，表示我们通过对偶问题成功解决了原问题。

强对偶性和Slater约束准则

强对偶性和**Slater约束条件**在凸优化理论中起着非常重要的作用，尤其是当我们讨论原问题和对偶问题之间的关系时。

1. 强对偶性

强对偶性指的是原问题的最优解与对偶问题的最优解相等，即两者的最优值是相同的。若存在强对偶性，说明原问题的最优值与对偶问题的最优值没有差距，对偶间隙为零。

数学上，假设 x^* 是原问题的最优解，(λ^*, ν^*) 是对偶问题的最优解，则**强对偶性**表示：

$$f(x^*) = g(\lambda^*, \nu^*),$$

其中 $f(x^*)$ 是原问题的最小值，$g(\lambda^*, \nu^*)$ 是对偶问题的最大值。

如果优化问题具有强对偶性，则我们可以通过求解对偶问题来间接求解原问题。在凸优化中，强对偶性在某些条件下成立，而**Slater约束条件**是确保强对偶性成立的常用条件之一。

2. Slater约束条件（Slater's Condition）

Slater约束条件是一个适用于凸优化问题的充分条件，用于保证强对偶性成立。

考虑一个凸优化问题：

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \cdots, m, \\ & h_j(x) = 0, \quad j = 1, \cdots, n, \end{aligned}$$

其中，目标函数 $f(x)$ 是凸的，约束 $g_i(x)$ 是凸函数，$h_j(x)$ 是仿射函数（线性约束的一种）。

Slater约束条件描述如下：

- **如果问题是凸优化问题，并且所有不等式约束 $g_i(x)$ 是严格凸的，那么存在一个可行解 x' 使得：**

$$g_i(x') < 0, \quad i = 1, \cdots, m,$$

  即，存在一个严格可行的点 x'，它满足所有不等式约束的严格形式。

当满足Slater条件时，对于凸优化问题可以确保**强对偶性成立**，也就是说，原问题的最优解与对偶问题的最优解是相等的。

总结：

- **强对偶性**：原问题的最优值与对偶问题的最优值相等，即对偶间隙为0。这通常可以通过求解对偶问题来间接求解原问题。
- **Slater约束条件**：这是一个充分条件，保证在凸优化问题中强对偶性成立。如果存在一个严格满足不等式约束的可行点（即严格可行解），那么该凸优化问题将具有强对偶性。

KKT条件

对于上述优化问题，如果在某点 x^* 处满足一定的正则性条件（如满足约束资格条件），则 x^* 是最优解的必要条件为：

1. **可行性条件（Primal Feasibility）**：

$$g_i(x^*) \leq 0, \quad \forall i = 1, 2, \ldots, m$$

$$h_j(x^*) = 0, \quad \forall j = 1, 2, \ldots, p$$

2. **对偶可行性条件（Dual Feasibility）**：
   存在拉格朗日乘子 $\lambda_i \geq 0$ 对于所有不等式约束，满足：

$$\lambda_i \geq 0, \quad \forall i = 1, 2, \ldots, m$$

3. **互补松弛条件（Complementary Slackness）**：

$$\lambda_i \cdot g_i(x^*) = 0, \quad \forall i = 1, 2, \ldots, m$$

   这意味着对于每一个不等式约束，要么约束是紧的（即 $g_i(x^*) = 0$），要么对应的拉格朗日乘子为零。
4. **梯度平衡条件（Stationarity）**：

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum_{j=1}^{p} \mu_j \nabla h_j(x^*) = 0$$

   其中，μ_j 是等式约束的拉格朗日乘子。

解释与应用

- **可行性条件**确保解 x^* 满足所有的约束。
- **对偶可行性条件**保证了拉格朗日乘子 λ_i 的非负性，这与不等式约束的方向相关。
- **互补松弛条件**表明只有在约束紧时，对应的乘子才可能非零，反之则乘子为零。
- **梯度平衡条件**表示在最优点，目标函数的梯度可以表示为约束函数梯度的线性组合，表明没有进一步改进的方向。

罚函数法

罚函数法的主要思想是将带有约束的优化问题转化为无约束优化问题来解决。具体做法是通过增加一个惩罚项来构造一个新的目标函数，该惩罚项用于惩罚违反约束条件的解。随着算法的迭代，罚因子逐渐增大，迫使最终解趋近于满足约束条件。

针对凸优化问题：

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \cdots, m, \\ & h_j(x) = 0, \quad j = 1, \cdots, n, \end{aligned}$$

我们可以使用二次罚函数法来解决该问题。

1. 二次罚函数定义：
为了将该约束优化问题转化为无约束优化问题，定义二次罚函数 $P(x, \sigma)$ 如下：

$$P(x, \sigma) = f(x) + \frac{\sigma}{2}\left[\sum_{j=1}^{n} h_j^2(x) + \sum_{i=1}^{m} \max(g_i(x), 0)^2\right]$$

其中：

- $h_j(x)$ 表示等式约束；
- $g_i(x)$ 表示不等式约束；
- 罚因子 $\sigma > 0$ 控制对约束违规的惩罚。

2. 二次罚函数法算法：
使用罚函数法解决该问题的步骤可以概括为：

1. 初始化：给定初始罚因子 $\sigma_1 > 0$，初始点 x^0，设定罚因子增长系数 $\rho > 1$，并初始化迭代计数 $k \leftarrow 1$。
2. While 未达到收敛准则：
   i. 以当前迭代点 x^k 为初始点，求解下一个迭代点 x^{k+1}：

$$x^{k+1} = \arg\min_{x} P(x, \sigma_k)$$

   ii. 更新罚因子：

$$\sigma_{k+1} = \rho \sigma_k$$

   iii. 增加迭代次数：$k \leftarrow k + 1$
3. End while

3. 罚函数法的优点：

- **易于实现**：通过将带约束的问题转化为无约束问题，避免了直接处理复杂的约束条件，便于使用无约束优化算法来求解。
- **统一处理不同类型的约束**：罚函数法能够同时处理等式约束和不等式约束。

4. 罚函数法的缺点：

- **罚因子选择困难**：如图中提到，罚因子 σ_k 的取值需要谨慎处理。若罚因子增长过快，子问题会变得难以求解，尤其是在初始迭代中可能导致解不稳定或跳跃。若增长过慢，算法需要更多的迭代次数才能收敛，计算效率降低。

- **收敛性问题**：当罚因子逐渐增大时，子问题求解的精度要求变得非常高，求解精度不够时可能导致解不收敛或精度不佳。
- **数值不稳定性**：在处理二次罚函数时，罚因子趋向无穷大，容易导致条件数恶化，海瑟矩阵（Hessian 矩阵）可能变得接近奇异，求解过程变得数值不稳定。
- **外点问题**：罚函数法会产生外点，即在迭代过程中，算法的中间解通常位于可行域之外，直到迭代过程接近结束时才逐步逼近可行解。这意味着在算法早期解并不具备物理可行性，可能不适合处理需要实时满足约束条件的问题。

总的来说，罚函数法是一种有效但需要谨慎调节参数的优化方法，适用于约束优化问题，但在实际应用中可能因为数值问题、罚因子选择以及收敛速度等问题受到限制。

增广拉格朗日法

等式约束的增广拉格朗日函数法

增广拉格朗日函数法通过罚函数和拉格朗日乘子相结合，逐步逼近原问题的最优解。该方法既保留了罚函数法的优点，又能通过逐步调整乘子来避免过大的罚因子导致的数值不稳定。

1. 增广拉格朗日函数的构造

对于带有等式约束的优化问题，增广拉格朗日函数的定义如下：

$$L_\sigma(x, \lambda) = f(x) + \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{1}{2}\sigma \sum_{i \in \mathcal{E}} c_i^2(x)$$

其中：

- $f(x)$ 是目标函数；
- λ_i 是等式约束 $c_i(x) = 0$ 的拉格朗日乘子；
- σ 是罚因子，用于控制对违反约束的惩罚。

2. 求解过程（等式约束增广拉格朗日法的流程）

增广拉格朗日函数法

1. **初始化**：
   - 选择初始点 x^0，乘子 λ^0，罚因子 $\sigma_0 > 0$，更新系数 $\rho > 0$，约束违度常数 $\epsilon > 0$ 和精度要求 $\eta_k > 0$；
   - 设迭代次数 $k = 0$。
2. **主循环**：
   - 设 x^k 为初始点，求解：

$$\min_x L_{\sigma_k}(x, \lambda^k)$$

得到满足精度要求的解 x^{k+1}，即：

$$\|\nabla_x L_{\sigma_k}(x^{k+1}, \lambda^k)\| \leq \eta_k$$

3. **检查约束违度**：
   - 若 $\|c(x^{k+1})\| \leq \epsilon$，则返回近似解 x^{k+1}, λ^k，并终止迭代。
4. **更新乘子**：
   - 乘子更新公式为：

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_k c_i(x^{k+1}), \quad \forall i \in \mathcal{E}$$

5. **更新罚因子**：

$$\sigma_{k+1} = \rho \sigma_k$$

6. **循环迭代**：返回到步骤 2，直到满足终止条件。

3. 注意点

- **罚因子增长**：罚因子 σ_k 随迭代逐渐增大，以逐步减少约束违度。过大的罚因子会导致数值不稳定，因此 ρ 的选择需要合理。
- **乘子更新**：拉格朗日乘子 λ 在每一步迭代中通过约束的残差 $c(x^{k+1})$ 更新，这一更新公式确保在迭代过程中乘子逐渐逼近最优值。
- **终止条件**：算法的终止条件基于约束违度 $\|c(x^{k+1})\|$ 和梯度的大小。当违度足够小且梯度满足精度要求时，算法结束。
- **精度控制**：通过调节 η_k 来控制求解精度，精度要求越高，求解过程越精确，但计算代价可能增加。

一般约束的增广拉格朗日函数法问题的构造和算法流程

1. 增广拉格朗日函数的构造

对于一般约束优化问题：

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & c_i(x) = 0, \quad i \in \mathcal{E}, \\ & c_i(x) \leq 0, \quad i \in \mathcal{I}, \end{aligned}$$

引入松弛变量 $s_i \geq 0, i \in \mathcal{I}$，将不等式约束转化为等式约束，并保持非负约束，优化问题变为：

$$
\begin{aligned}
\min \quad & f(x) \\
\text{s.t.} \quad & c_i(x) = 0, \quad i \in \mathcal{E}, \\
& c_i(x) + s_i = 0, \quad i \in \mathcal{I}, \\
& s_i \geq 0, \quad i \in \mathcal{I}.
\end{aligned}
$$

构造拉格朗日函数：

$$
L(x, s, \lambda, \mu) = f(x) + \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \sum_{i \in \mathcal{I}} \mu_i (c_i(x) + s_i), \quad s_i \geq 0, \forall i \in \mathcal{I}.
$$

然后将问题中的等式约束通过二次罚函数引入，定义为 $p(x, s)$：

$$
p(x, s) = \sum_{i \in \mathcal{E}} c_i^2(x) + \sum_{i \in \mathcal{I}} (c_i(x) + s_i)^2.
$$

增广拉格朗日函数 $L_\sigma(x, s, \lambda, \mu)$ 定义为：

$$
L_\sigma(x, s, \lambda, \mu) = f(x) + \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \sum_{i \in \mathcal{I}} \mu_i (c_i(x) + s_i) + \frac{\sigma}{2} p(x, s).
$$

2. 求解过程（等一般约束增广拉格朗日法的流程）

根据图中的算法6.5，总结出增广拉格朗日函数法的求解步骤如下：

1. **初始化**：
   - 选择初始点 x^0，拉格朗日乘子 λ^0，μ^0，罚因子 $\sigma_0 > 0$，精度要求 $\eta > 0$，常数 $0 < \alpha \leq \beta < 1$，增大因子 $\rho > 1$。
   - 设置精度参数：$\eta_0 = \frac{1}{\sigma_0}, \epsilon_0 = \frac{1}{\sigma_0^\alpha}$，并令 $k = 0$。
2. **主循环**：
   - 对 $k = 0, 1, 2, \ldots$，执行以下步骤：
3. **求解子问题**：
   - 以 x^k 为初始点，求解优化问题：

     $$
     \min_x L_{\sigma_k}(x, \lambda^k, \mu^k),
     $$

     满足精度条件：

     $$
     \|\nabla_x L_{\sigma_k}(x^{k+1}, \lambda^k, \mu^k)\|_2 \leq \eta_k.
     $$
4. **检查约束违度**：
   - 计算约束违度：

$$v_k(x^{k+1}) = \sqrt{\sum_{i\in\mathcal{E}} c_i^2(x^{k+1}) + \sum_{i\in\mathcal{I}} \max(c_i(x^{k+1}), -\frac{\mu_i^k}{\sigma_k})^2}.$$

- 如果 $v_k(x^{k+1}) \leq \epsilon_k$，则：
  - 如果 $\|\nabla_x L_{\sigma_k}(x^{k+1}, \lambda^k, \mu^k)\|_2 \leq \eta_k$，则终止迭代，返回解 $x^{k+1}, \lambda^k, \mu^k$。
  - 否则更新乘子：

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_k c_i(x^{k+1}), \quad i \in \mathcal{E},$$

$$\mu_i^{k+1} = \max\left\{\mu_i^k + \sigma_k c_i(x^{k+1}), 0\right\}, \quad i \in \mathcal{I}.$$

  - 罚因子保持不变 $\sigma_{k+1} = \sigma_k$。

5. **罚因子调整**：
   - 如果约束违度 $v_k(x^{k+1}) > \epsilon_k$，则更新罚因子：

$$\sigma_{k+1} = \rho\sigma_k,$$

   同时更新精度要求：

$$\eta_{k+1} = \frac{\eta_k}{\sigma_{k+1}}, \quad \epsilon_{k+1} = \frac{\epsilon_k}{\sigma_{k+1}^\beta}.$$

   拉格朗日乘子保持不变：$\lambda^{k+1} = \lambda^k$。
6. **迭代终止**：直到满足精度要求停止。

算法流程中用到的公式：

- 增广拉格朗日函数：

$$L_\sigma(x, s, \lambda, \mu) = f(x) + \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \sum_{i\in\mathcal{I}} \mu_i(c_i(x) + s_i) + \frac{\sigma}{2} p(x, s),$$

  其中 $p(x, s)$ 为：

$$p(x, s) = \sum_{i\in\mathcal{E}} c_i^2(x) + \sum_{i\in\mathcal{I}} (c_i(x) + s_i)^2.$$

- 拉格朗日乘子的更新：

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_k c_i(x^{k+1}), \quad i \in \mathcal{E},$$

$$\mu_i^{k+1} = \max\left\{\mu_i^k + \sigma_k c_i(x^{k+1}), 0\right\}, \quad i \in \mathcal{I}.$$

- 约束违度计算公式：

$$v_k(x^{k+1}) = \sqrt{\sum_{i\in\mathcal{E}} c_i^2(x^{k+1}) + \sum_{i\in\mathcal{I}} \max(c_i(x^{k+1}), -\frac{\mu_i^k}{\sigma_k})^2}.$$