作业一

姓名：谢泽钰

学号：2020012544

前言

这个问题常常被描述为 l_0 范数最小化问题：即在满足 $Ax=b$ 的条件下，寻找具有最少非零元素的向量 x。

为了处理这个问题，一种著名的数学模型是通过 l_1 范数最小化，即 LASSO 问题或基追踪问题。

为了解决 LASSO 问题或基追踪问题，我们提出了一个非凸松弛模型，该模型的目标是最小化一个由两部分组成的函数：第一部分是 Ax 与 b 之间的欧几里德范数的平方，第二部分是一个与向量 x 的每个分量相关的非凸替代函数 $g(x_i)$ 之和。我们使用参数 μ 调整两部分的相对重要性。

这个模型的特点是，它考虑了数据服从高斯随机分布的情况，并使用非凸替代函数来处理问题。

模型

为解决 l_0 范数最小化问题

$$\min_x \|x\|_0 \quad s.t.\ Ax=b \tag{1}$$

我们考虑如下非凸松弛模型

$$\min_x \frac{1}{2}\|Ax-b\|_2^2+\mu\sum_{i=1}^n g(x_i) \tag{2}$$

其中数据符合高斯随机分布

$$f(x|\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3}$$

并取 $g(\cdot)$ 为 Capped l_1 函数（参数 $\gamma>0$ ），也即

$$g(\theta)=\begin{cases}|\theta|, & \text{if } |\theta|\le\gamma,\\ \gamma, & \text{if } |\theta|\ge\gamma.\end{cases} \tag{4}$$

求解

理论分析

首先，我们可以写出该问题的拉格朗日函数：

$$L(x,\lambda)=\frac{1}{2}\|Ax-b\|_2^2+\mu\sum_{i=1}^n g(x_i)+\sum_{i=1}^n \lambda_i x_i \tag{5}$$

其中 $\lambda=[\lambda_1,\lambda_2,\ldots,\lambda_n]$ 是拉格朗日乘子向量。

接下来，我们可以建立该问题的最优性条件，即对拉格朗日函数求偏导数并令其为零。

$$\frac{\partial L}{\partial x_i} = A^T(Ax-b) + \mu g'(x_i) + \lambda_i = 0, \quad \text{for } i = 1, 2, \ldots, n \tag{6}$$

这里 $g'(x_i)$ 是 $g(x_i)$ 对 x_i 的导数，当 $g(\cdot)$ 为 Capped l_1 函数（$\gamma > 0$）时有

$$g'(\theta) = \begin{cases} \frac{\theta}{|\theta|}, & \text{if } -\gamma < \theta < \gamma, \\ 0, & \text{if } \theta \le -\gamma \text{ or } \theta \ge \gamma. \end{cases} \tag{7}$$

然后，我们使用次梯度法进行求解，其迭代步骤如下：

$$x^{(k+1)} = prox_{\gamma\mu g}(x^{(k)} - \gamma A^T(Ax^{(k)} - b + \lambda^{(k)})) \tag{8}$$

$$\lambda^{(k+1)} = \lambda^{(k)} + \alpha(Ax^{(k+1)} - b) \tag{9}$$

其中，$\gamma > 0$ 是步长参数，$\alpha > 0$ 是拉格朗日乘子更新的步长参数。

因为 $g(\cdot)$ 为 Capped l_1 函数（$\gamma > 0$）故 $prox$ 运算符可选取为：

$$prox_{\gamma\mu g}(x_i) = \text{sign}(x_i) \cdot \max\{|x_i| - \gamma\mu, 0\} \tag{10}$$

代码实现

```
import numpy as np

def g_prime(x_i, gamma):
    return np.where((-gamma < x_i) & (x_i < gamma) & (x_i != 0), x_i / np.abs(x_i), 0)

def prox_operator(x, gamma, mu):
    return np.sign(x) * np.maximum(np.abs(x) - gamma * mu, 0)

def solve_nonconvex_relaxation(A, b, mu, gamma, alpha, max_iter):
    n = A.shape[1]
    x = np.zeros((n, 1))
    lambda_vec = np.zeros((n, 1))

    for _ in range(max_iter):

        # 更新 x
        grad_x = A.T @ (A @ x - b) + mu * np.array([g_prime(xi, gamma) for xi in x]) +
lambda_vec
        x = prox_operator(x - gamma * grad_x, gamma, mu)

        # 更新 Lagrange 乘子
        lambda_vec = lambda_vec + alpha * (A @ x - b)

    return x

# 示例用法
A = np.array([[1, 2], [3, 4]])
b = np.array([5, 6])
mu = 0.1
gamma = 0.01
alpha = 0.1
```

```
max_iter = 100

result = solve_nonconvex_relaxation(A, b, mu, gamma, alpha, max_iter)
print("Optimal solution x:", result)
```

收敛性分析

使用如下定义的目标函数可证明上述算法的收敛性:

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \mu \sum_{i=1}^{n} g(x_i) \tag{11}$$

可以证明在每次迭代中，目标函数值是下降的。