最优化第四次作业

第四次作业题

课程名称：	最优化理论与算法 II
姓名：	郅啸淇
学院：	管理学院
专业：	大数据管理与应用
学号：	2184114639
指导老师：	Xiangyu Chang

2022 年 10 月 27 日

专业： 大数据管理与应用
姓名： 郅啸淇
学号： 2184114639
日期： 2022 年 10 月 27 日
地点： 寝室

西安交通大學实验报告

课程名称： 最优化理论与算法 II　指导老师： Xiangyu Chang　成绩： ??
实验名称： 第四次作业题　实验类型： 完成作业　同组学生姓名： Nobody

一、 HW1

LASSO Problem $x^* = argmin_x\{\frac{1}{2}\|A\mathbf{x} - b\|^2 + \lambda\|\mathbf{x}\|_1\}$
定义 $f_k^t(x_k) = f(x_{<k}^{t+1}, x_k, x_{>k}^t)$
原 LASSO 问题等价于 $\min_x\{\frac{1}{2}\|b_i - \sum_{i=1}^n x_iA_i\|^2 + \lambda\sum_{i=1}^n x_i\}$
其中 $A = [A_1, A_2, ..., A_n], b_i = b - \sum_{j\neq i} x_jA_j$
当 $i = 1$,$x_2, ...x_n$ 固定时
原式为 $\min_x\{\frac{1}{2}\|b_1 - x_1A_1\|^2 + \lambda x_1\}$
令 $g(x) = \frac{1}{2}\|b_1 - x_1A_1\|^2 + \lambda x_1$
当 $x_1 > 0$ 时，$\nabla_x g(x) = -A_1(b_1 - x_1A_1) + \lambda = 0$
$x_1 = (A^TA)^{-1}(A_1b_1 - \lambda)$
当 $x_1 < 0$ 时，$x_1 = (A^TA)^{-1}(A_1b_1 + \lambda)$
当 $x_1 = 0$ 时，$0 \in -A_1^T(b_1 - x_1A_1) + \lambda\partial|0|$

故 $x_i = \begin{cases} (A^TA)^{-1}(A_nb_n - \lambda) & x_i > 0 \\ (A^TA)^{-1}(A_nb_n + \lambda) & x_i < 0 \\ 0 & |A_i^Tb_i^T| \leq \lambda \end{cases}$

二、 HW2

Fused LASSO
$\min_x\{\frac{1}{2}\|Ax - b\|^2 + \lambda\|Bx\|\}$
等价于

$$\begin{aligned} &\min_x\{\frac{1}{2}\|Ax - b\|^2 + \lambda\|z\|\} \\ &s.t. \quad Bx - z = 0 \end{aligned} \tag{1}$$

增广 Lagrange 函数为 $L_\rho(x, z, v) = \frac{1}{2}\|Ax - b\|^2 + \lambda\|z\| + v^T(Bx - z) + \frac{\rho}{2}\|Bx - z\|^2$

$\begin{cases} x^{t+1} = argmin_x L_\rho(x, z^t, v^t) \\ z^{t+1} = argmin_x L_\rho(x^{t+1}, z, v^t) \\ v^{t+1} = v^t + \rho(Bx^{t+1} - z^{t+1}) \end{cases}$

令 $u = \frac{v}{\rho}$
$x^{t+1} = argmin_x\{\frac{1}{2}\|Ax - b\|^2 + \frac{\rho}{2}\|Bx - z^t + u^t\|^2\}$
令 $g(x) = \frac{1}{2}\|Ax - b\|^2 + \frac{\rho}{2}\|Bx - z^t + u^t\|^2$
$\nabla g(x) = A^T(Ax - b) + \rho(x - z^t + u^t) = 0$

$x^{t+1} = (A^TA+\rho I)^{-1}(A^Tb+\rho z^t-\rho u^t)$
$z^{t+1} = argmin_x\{\lambda\|z\|+\frac{\rho}{2}\|Bx^{t+1}+u^t-z\|^2\}$
$\Rightarrow prox_{\frac{\lambda}{\rho}\|z\|}(Bx^{t+1}+u^t)$
$= sign(Bx^{t+1}+u^t)(|Bx^{t+1}+u^t|-\frac{\lambda}{\rho})_+$
$u^{t+1} = u^t+Bx^{t+1}-z^{t+1}$

$$\begin{cases} x^{t+1} = (A^TA+\rho I)^{-1}(A^Tb+\rho z^t-\rho u^t) \\ z^{t+1} = sign(Bx^{t+1}+u^t)(|Bx^{t+1}+u^t|-\frac{\lambda}{\rho})_+ \\ u^{t+1} = u^t+Bx^{t+1}-z^{t+1} \end{cases}$$

三、 HW3

原问题

$$\begin{aligned} &\min_x\{\|x\|_1\} \\ &s.t. \quad Ax=b \end{aligned} \tag{2}$$

等价于

$$\begin{aligned} &\min_x\{\|z\|+f(x)\} \\ &s.t. \quad x-z=0 \end{aligned} \tag{3}$$

其中 $f(x) = \begin{cases} 0, if Ax=b \\ +\infty, if Ax\neq b \end{cases}$

$L_\rho(x,z,v) = \begin{cases} \|z\|+v^T(x-z)+\frac{\rho}{2}\|x-z\|^2 \\ +\infty, if Ax\neq b \end{cases}$

$$\begin{cases} x^{t+1} = argmin_x L_\rho(x,z^t,v^t) \\ z^{t+1} = argmin_x L_\rho(x^{t+1},z,v^t) \\ v^{t+1} = v^t+\rho(Bx^{t+1}-z^{t+1}) \end{cases}$$

$x^{t+1} = argmin_x\{v^T(x-z^t)+\frac{\rho}{2}\|x-z^t\|^2\}$
$= argmin_x\{\frac{\rho}{2}\|x-z^t+u^t\|^2\}$
$= \pi_\omega(z^t-u^t)$
其中 $\omega = \{x|Ax=b\}$
由上次作业结论 $x^{t+1} = (z^t-u^t)-A^T(A^TA)^{-1}[A(z^t-u^t)-b]$
$z^{t+1} = argmin_x\{\|z\|+\frac{\rho}{2}\|x^{t+1}+u^t-z^t\|^2\}$
$= prox_{\frac{\|z\|}{\rho}}\{x^{t+1}-u^t\}$
$= sign(x^{t+1}-u^t)(|x^{t+1}-u^t|-\frac{1}{\rho})_+$
$u^{t+1} = u^t+x^{t+1}-z^{t+1}$

$$\begin{cases} x^{t+1} = (z^t-u^t)-A^T(A^TA)^{-1}[A(z^t-u^t)-b] \\ z^{t+1} = sign(x^{t+1}-u^t)(|x^{t+1}-u^t|-\frac{1}{\rho})_+ \\ u^{t+1} = u^t+x^{t+1}-z^{t+1} \end{cases}$$

四、 HW4

```python
import numpy as np
import matplotlib.pyplot as plt
```

```python

np.random.seed(2022) # set a constant seed to get samerandom matrixs
A = np.random.rand(500, 100)
x_ = np.zeros([100, 1])
x_[:5, 0] += np.array([i + 1 for i in range(5)]) # x_denotes expected x
b = np.matmul(A, x_) + np.random.randn(500, 1) * 0.1 # add a noise to b
lam = 10 # try some different values in {0.1, 1, 10}


def fx(A, x, b, mu):
    f = 1 / 2 * np.linalg.norm(A @ x - b, ord=2) ** 2 + mu * np.linalg.norm(x, ord=1)
    return f


def Beta(A):
    return max(np.linalg.eig(A.T @ A)[0])


def z(A, x, b):
    beta = Beta(A)
    z = (np.eye(len(x)) - A.T @ A / beta) @ x + A.T @ b / beta
    return z


def xp(z, mu, A):
    temp = abs(z) - mu / Beta(A)
    for i in range(len(temp)):
        if temp[i] > 0:
            temp[i] = temp[i]
        else:
            temp[i] = 0
    xp = np.sign(z) * temp
    return xp


def prox(A, x, b, mu, ml):
    k = 0
    fmin = fx(A, x, b, mu)
    fk = fmin
    f_list = [fk]
    while k < ml:
        k = k + 1
        x = xp(z(A, x, b), mu, A)
        fk = fx(A, x, b, mu)
        f_list.append(fk)
        if fk < fmin:
            fmin = fk
    plt.scatter(list(range(len(f_list))), f_list, s=5, color="red")
    plt.show()
    print("迭代结果为：", fmin)
```

```python


# prox(A,x_,b,lam,100)
# prox(A,x_,b,1,1000)
# prox(A,x_,b,0.1,1000)
def BCD(A, x, b, mu):
    k = 0
    y = np.ones([100, 1])
    fk = fx(A, x, b, mu)
    f_list = [fk]
    while k < 100:
        y = x
        k = k + 1
        for i in range(len(x)):
            if x[i][0] > 0:
                x[i][0] = 1 / (A[:, i].T @ A[:, i]) * (A[:, i].T @ b2(A, x, b, i) - mu)
            elif x[i][0] < 0:
                x[i][0] = 1 / (A[:, i].T @ A[:, i]) * (A[:, i].T @ b2(A, x, b, i) + mu)
            elif abs(A[:, i].T @ b2(A, x, b, i)) <= mu:
                x[i][0] = 0
        fk = fx(A, x, b, mu)
        f_list.append(fk)
    plt.scatter(list(range(len(f_list))), f_list, s=5)
    plt.show()
    print("迭代结果为: ", fx(A, x, b, mu))


def b2(A, x, b, n):
    sum = np.zeros((500, 1))
    for i in range(n):
        sum = sum + x[i][0] * A[:, i]
    for i in range(n + 1, len(x)):
        sum = sum + x[i][0] * A[:, i]
    b2 = b - sum
    return b2


A = np.matrix(A)


# BCD(A, x_, b, 10)
# BCD(A, x_, b, 1)
# BCD(A, x_, b, 0.1)

def fxz(A, x, z, b, lam):
    f = 1 / 2 * np.linalg.norm(A @ x - b, ord=2) ** 2 + lam * np.linalg.norm(z, ord=1)
    return f


def Beta(A):
```

```
    return max(np.linalg.eig(A.T @ A)[0])


def xp(z, lam, A):
    temp = abs(z) - lam / Beta(A)
    for i in range(len(temp)):
        if temp[i] > 0:
            temp[i] = temp[i]
        else:
            temp[i] = 0
    xp = np.sign(z) * temp
    return xp


def ADMM(A, x, b, lam):
    mu = np.ones([100, 1])
    rho = Beta(A)
    rho_i = np.identity(A.shape[1]) * rho
    z = x
    k = 0
    F = []
    f = fxz(A, x, z, b, lam)
    while k < 100:
        x = np.linalg.inv(A.T @ A + rho_i) @ (A.T @ b + rho * (z - mu))
        z = xp(x + mu, lam, A)
        mu = mu + x - z
        k = k + 1
        deltaf = (f - fxz(A, x, z, b, lam)) / fxz(A, x, z, b, lam)
        f = fxz(A, x, z, b, lam)
        F.append(f)
    plt.scatter(list(range(0, 100)), F, s=5)
    plt.show()
    print(fxz(A, x, z, b, lam))


np.random.seed(2022)
A = np.random.rand(500, 100)
# ADMM(A, x_, b, 10)
# ADMM(A, x_, b, 1)
# ADMM(A, x_, b, 0.1)
```

1. Proximal Gradient Descent

$\lambda = 10$ 时，迭代结果为:152.23667337896808

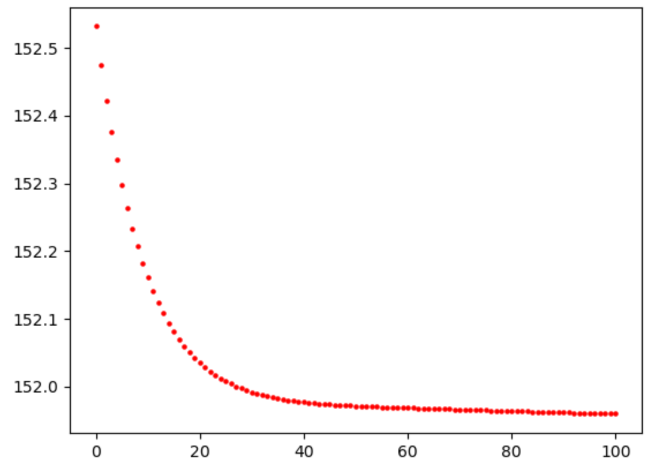

图 1: lambda = 10

$\lambda = 1$ 时，迭代结果为:17.47128983481235

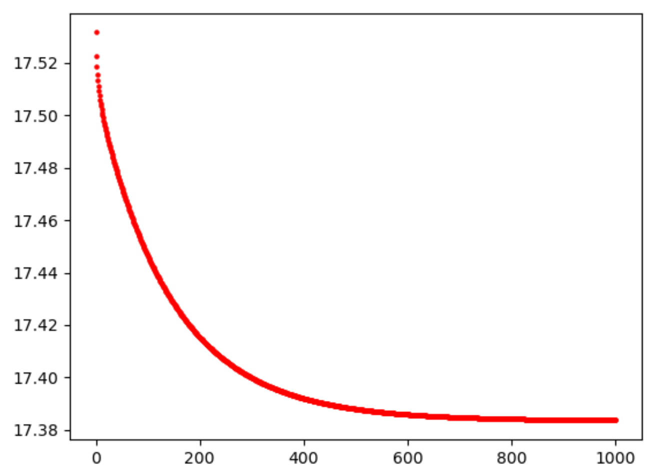

图 2: lambda = 1

$\lambda = 0.1$ 时，迭代结果为:3.587676594826708

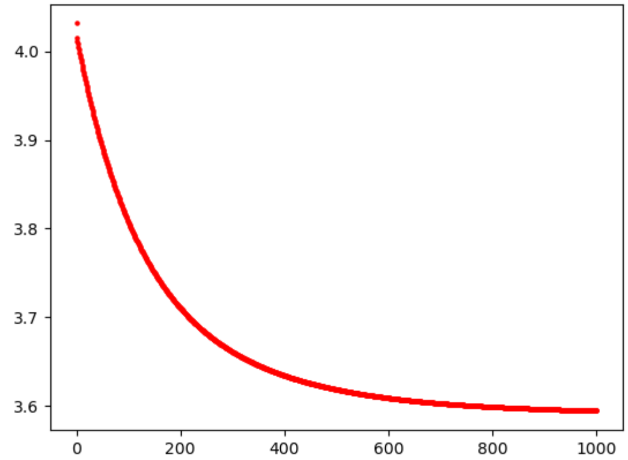

图 3: lambda = 0.1

2. BCD

$\lambda = 10$ 时，迭代结果为:152.23046561943306

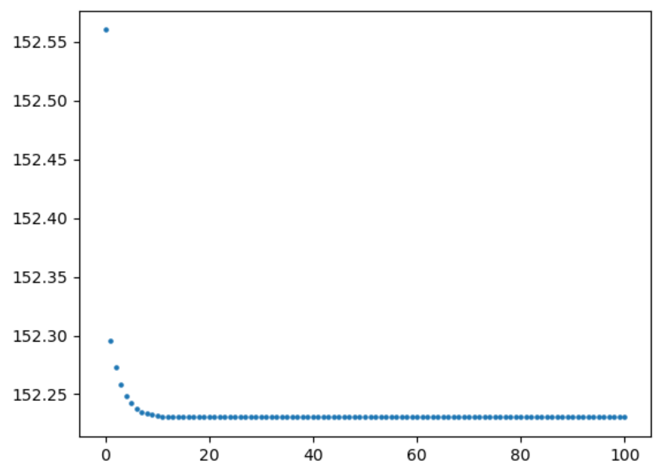

图 4: lambda = 10

$\lambda = 1$ 时，迭代结果为:17.5370839635338

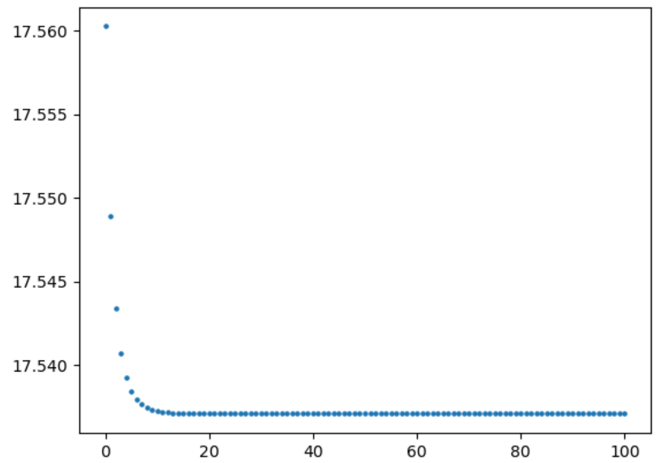

图 5: lambda = 1

$\lambda = 0.1$ 时，迭代结果为:4.0347736846143825

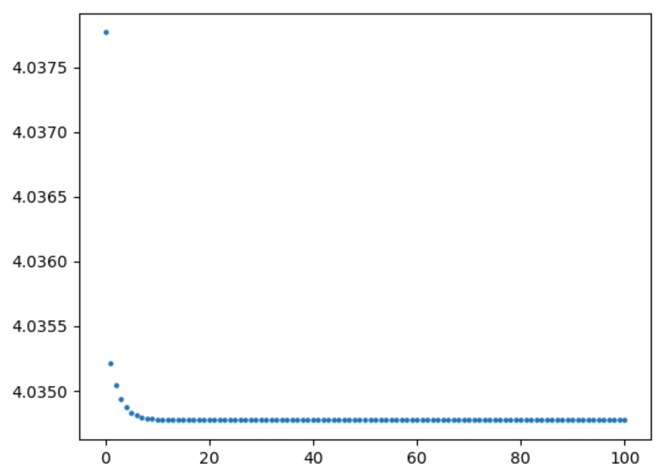

图 6: lambda = 0.1

3. ADMM

$\lambda = 10$ 时，迭代结果为:152.2382124270402

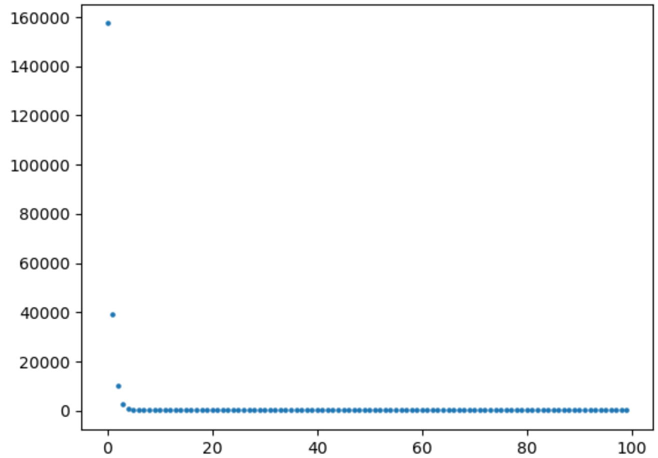

图 7: lambda = 10

$\lambda = 1$ 时，迭代结果为:17.51542818309732

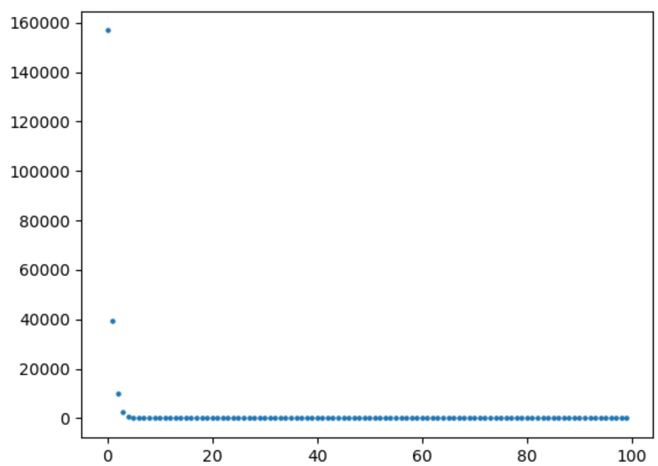

图 8: lambda = 1

$\lambda = 0.1$ 时，迭代结果为:3.821736337446727

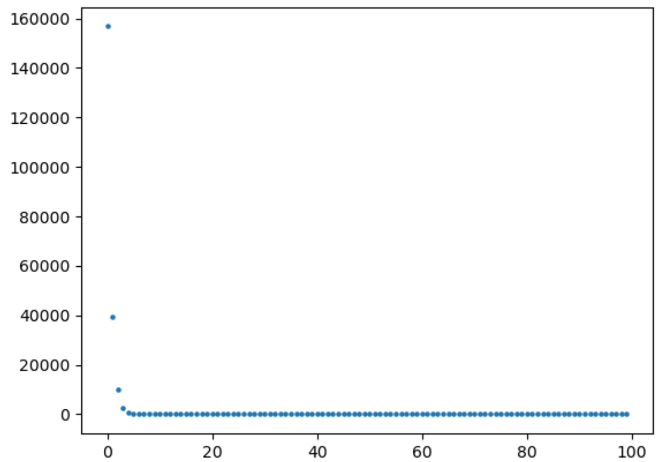
160000
140000
120000
100000
80000
60000
40000
20000
0
0
20
40
60
80
100

图 9: lambda = 0.1