最优化第三次作业

第三次作业题

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 19 日

西安交通大學实验报告

专业： 大数据管理与应用
姓名： 郅啸淇
学号： 2184114639
日期： 2022 年 10 月 19 日
地点： 寝室

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：	第三次作业题	实验类型：	完成作业	同组学生姓名：	Nobody

一、 HW1

1. Theorem1

$D_\phi(x,y)+D_\phi(z,x)-D_\phi(z,y)$
$=\phi(x)-\phi(y)-\nabla\phi(y)^T(x-y)+\phi(z)-\phi(x)-\nabla\phi(x)^T(z-x)-\phi(z)+\phi(y)+\nabla\phi(y)^T(z-y)$
$=\nabla\phi(y)^T(z-y-x+y)-\nabla\phi(x)^T(z-x)$
$=(\nabla\phi(x)-\nabla\phi(y))^T(x-z)$
得证

2. Theorem3

由一般最优性条件 $\langle\nabla f(x^*),y-x^*\rangle\geq 0$
则对于 $D_\phi(x,y)$ 的最优值有 $\langle\nabla D_\phi(x^*,y),(y-x^*)\rangle\geq 0$
由条件有
$\pi_\omega^\phi(y)=argmin_{x\in\omega}D_\omega(x,y)$
$\nabla D_\phi(x,y)=\nabla\phi(x)-\nabla\phi(y)$
代入 D_ϕ 最优性条件即有
$(\nabla\phi(\pi_\omega^\phi(y))-\nabla\phi(y))^T(\pi_\omega^\phi(y)-z)\leq 0$
得证

3. Theorem4

由 Theorem1 有
$(\nabla\phi(\pi_\omega^\phi(y))-\nabla\phi(y))^T(\pi_\omega^\phi(y)-z)=D_\phi(z,\pi_\omega^\phi(y))+D_\phi(\pi_\omega^\phi,y)-D_\phi(z,y)\leq 0$
故有 $D_\phi(z,y)\geq D_\phi(z,\pi_\omega^\phi(y))+D_\phi(\pi_\omega^\phi(y),y)$
得证

二、 HW2

1. Optimization of Quadratic Program

$$\min_x \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}\|^2 \quad s.t. \quad A\mathbf{x}_0 = \mathbf{b} \tag{1}$$

使用 Lagrange 乘数法，然后使用 KKT 条件求解 x^*

$L(x,\lambda) = \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}\| + \lambda(A\mathbf{x} - b)$

KKT 条件

$$\begin{cases} \frac{\partial L}{\partial x} = -(\mathbf{x}_0 - \mathbf{x}) + A\lambda = 0 \\ \frac{\partial L}{\partial \lambda} = A\mathbf{x} - b = 0 \end{cases}$$

求解得

$\lambda = A(A^TA)^{-1} - (A^TA)^{-1}b$

$x^* = x_0 - A(A^TA)^{-1}(Ax_0 - b)$

得证

2. Projected Gradient Descent Algorithm

上一题的二次规划最优解即向量对超平面的投影

即 $\pi_c(x) = x - A(A^TA)^{-1}(Ax_0 - b)$

那么优化问题

$$\min_x f(x) \quad s.t. \quad A\mathbf{x}_0 = \mathbf{b} \tag{2}$$

的投影梯度下降迭代算法

三、 HW3

1. I

$E_{D_t}\|g^t\|^2 = E\|\frac{1}{n_b}\sum_{i\in D_t}\nabla f_i(x^t)\|^2$

$= \frac{1}{n_b^2}(\sum_{i\in D_t}^{n_b} E[\|\nabla f_i(x^t)\|^2] + \sum_{i\in D_t, j\in D_t, i\neq j}^{n_b}(E[\|\nabla f_i(x^t)\|^2] * E[\|\nabla f_j(x^t)\|^2]))$

$= \frac{1}{n_b^2}(n_b\sigma^2 + n_b(n_b-1)\|\nabla f(x^t)\| * \|\nabla f(x^t)\|)$

$= \frac{\sigma^2}{n_b} + \|\nabla f(x^t)\|^2$

得证

2. II

在 A1 条件下，有

$f(x^{t+1}) \le f(x^t) + \langle \nabla f(x^t), x^{t+1} - x^t\rangle + \frac{\beta}{2}\|x^{t+1} - x^t\|^2$

$= f(x^t) - S_t\langle \nabla f(x^t), \nabla f_{i_t}(x^t)\rangle + \frac{\beta S_t^2}{2}\|\nabla f_{i_t}(x^t)\|^2$

$= f(x^t) - S_t\langle \nabla f(x^t), \frac{1}{n_b}\sum_{i\in D_t}\nabla f_i(x^t)\rangle + \frac{\beta S_t^2}{2}\|\frac{1}{n_b}\sum_{i\in D_t}\nabla f_i(x^t)\|^2$

对式子两边取期望，即得证。

3. III

在 A1 和 A2 的条件下
$E[f(x^{t+1})-f(x^t)] \le \frac{\beta S_t^2}{2}E_{D_t}[\|g^t\|^2]-S_t\nabla f(x)^T E_{D_t}(g^t)$
$\le \frac{\beta S_t^2}{2}(\frac{\sigma^2}{n_b}+\|\nabla f(x^t)\|^2)-S_t\|\nabla f(x^t)\|^2$
$= \frac{\beta S_t^2}{2n_b}\sigma^2-(s-\frac{\beta S_t^2}{2})\|\nabla f(x^t)\|^2$
得证

4. IV

在 A1 和 A2 的条件下
令 $S_t \in (0, \frac{1}{\beta}]$
有引理
$f(x)-f^* \le \frac{1}{2\alpha}\|\nabla f(x^t)\|^2$
则有
$E_{D_t}[f(x^{t+1})-f(x^t)] = \frac{\beta S_t^2}{2n_b}\sigma^2-(s-\frac{\beta S_t^2}{2})\|\nabla f(x^t)\|$
$\le \frac{\beta S_t^2}{2n_b}\sigma^2-\alpha S_t(f(x^t)-f^*)$
$\Rightarrow E_{D_t}[f(x^{t+1})-f^*+f^*-f(x^t)] \le \frac{\beta S_t^2}{2n_b}\sigma^2-\alpha S_t(f(x^t)-f^*)$
$\Rightarrow E_{D_t}[f(x^{t+1})-f^*] \le \frac{\beta S_t^2}{2n_b}\sigma^2-(1-\alpha)S_t(f(x^t)-f^*)$
$\Rightarrow E_{D_t}[f(x^{t+1})-f^*]-\frac{\beta S_t}{2n_b\alpha(2-\beta S_t)}\sigma^2 \le \frac{\beta S_t^2}{2n_b}\sigma^2-\frac{\beta S_t}{2n_b\alpha(2-\beta S_t)}\sigma^2+(1-\alpha)S_t(f(x^t)-f^*)$
$\le (1-\alpha S_t(2-\beta S_t))(f(x^t)-f^*-\frac{\beta S_t}{2n_b\alpha(2-\beta S_t)}\sigma^2)$
对两边取期望即得证

四、 HW4

逻辑回归对应的优化问题可以写成
$\min_{x\in\mathbf{R}} = \frac{1}{N}\sum_{i=1} Nf_i(x) = \frac{1}{N}\sum_{i=1} Nln(1+exp(-b_i * a_i^T x))+\lambda\|x\|_2^2$

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv('a9a.csv')
data = data.values
n = data.shape[0]
data2 = []
for i in range(n):
    a = data[i][0].split()
    data2.append(a)
y = []
for i in range(n):
    temp = int(data2[i][0])
    y.append(temp)
y = np.array(y)
y = y.reshape(-1, 1)
c = []
```

```python
for i in range(n):
    b = []
    for j in range(1, len(data2[i])):
        index = data2[i][j].find(':')
        b.append(int(data2[i][j][0:index]))
    c.append(b)
m = max(max(i) for i in c)
X = np.zeros((n, m))
for i in range(n):
    for j in range(len(c[i])):
        X[i, c[i][j] - 1] = 1
X = np.matrix(X)

import math


def SGD_fix(A, b, lam):
    k = np.random.randint(0, A.shape[0])
    t = 0
    x = np.zeros((A.shape[1], 1))
    F = []
    sum = 0
    for i in range(A.shape[0]):
        sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
    f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
    F.append(f)
    deltaF = 1
    while abs(deltaF / f) > 0.0000001:
        k = np.random.randint(0, A.shape[0])
        x = x - 0.001 * ((-math.exp(-b[k, :][0] * A[k, :] @ x) * b[k, :][0] * A[k, :].T) / (
                1 + math.exp(-b[k, :][0] * A[k, :] @ x)) + 2 * lam * x)
        sum = 0
        for i in range(A.shape[0]):
            sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
        f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
        F.append(f)
        t = t + 1
        deltaF = F[t] - F[t - 1]
        print(deltaF / f, t)
    plt.scatter(list(range(len(F))), F, s=5)
    plt.show()
    print(f)


SGD_fix(X, y, 0.01 / X.shape[0])


def SGD_dr(A, b, lam):
    k = np.random.randint(0, A.shape[0])
    t = 0
```

```python
    x = np.zeros((A.shape[1], 1))
    F = []
    sum = 0
    for i in range(A.shape[0]):
        sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
    f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
    F.append(f)
    deltaF = 1
    while abs(deltaF / f) > 0.0000001:
        t = t + 1
        k = np.random.randint(0, A.shape[0])
        x = x - 1 / t * ((-math.exp(-b[k, :][0] * A[k, :] @ x) * b[k, :][0] * A[k, :].T) / (
                1 + math.exp(-b[k, :][0] * A[k, :] @ x)) + 2 * lam * x)
        sum = 0
        for i in range(A.shape[0]):
            sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
        f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
        F.append(f)
        deltaF = F[t] - F[t - 1]
        print(deltaF / f, t)
    plt.scatter(list(range(len(F))), F, s=5)
    plt.show()
    print(f)


SGD_dr(X, y, 0.01 / X.shape[0])


def SVRG(A, b, lam):
    t = 0
    x = np.zeros((A.shape[1], 1))
    y = x
    F = []
    sum = 0
    for i in range(A.shape[0]):
        sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
    f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
    F.append(f)
    deltaF = 1
    grad = 0
    for i in range(A.shape[0]):
        grad = grad + (-math.exp(-b[i, :][0] * A[i, :] @ x) * b[i, :][0] * A[i, :].T) / (
                1 + math.exp(-b[i, :][0] * A[i, :] @ x)) + 2 * lam * x
    gradF = grad / A.shape[0]
    T = 0
    while abs(deltaF / f) > 0.00001:
        #    while T<10000:
        xs = []
        t = 0
        grad = 0
```

```
        for i in range(A.shape[0]):
            grad = grad + (-math.exp(-b[i, :][0] * A[i, :] @ y) * b[i, :][0] * A[i, :].T) / (
                    1 + math.exp(-b[i, :][0] * A[i, :] @ y)) + 2 * lam * y
        grady = grad / A.shape[0]
        while t < 20:
            t = t + 1
            T = T + 1
            k = np.random.randint(0, A.shape[0])
            gradkx = (-math.exp(-b[k, :][0] * A[k, :] @ x) * b[k, :][0] * A[k, :].T) / (
                    1 + math.exp(-b[k, :][0] * A[k, :] @ x)) + 2 * lam * x
            gradky = (-math.exp(-b[k, :][0] * A[k, :] @ y) * b[k, :][0] * A[k, :].T) / (
                    1 + math.exp(-b[k, :][0] * A[k, :] @ y)) + 2 * lam * y
            v = gradkx - (gradky - grady)
            x = x - 0.01 * v
            xs.append(x)
            sum = 0
            for i in range(A.shape[0]):
                sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
            f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
            F.append(f)
            deltaF = F[T] - F[T - 1]
            print(deltaF / f, T)
        sumx = 0
        for i in range(len(xs)):
            sumx = sumx + xs[i]
        y = sumx / len(xs)
    plt.scatter(list(range(len(F))), F, s=5)
    plt.show()
    print(f)


SVRG(X, y, 0.01 / X.shape[0])
```

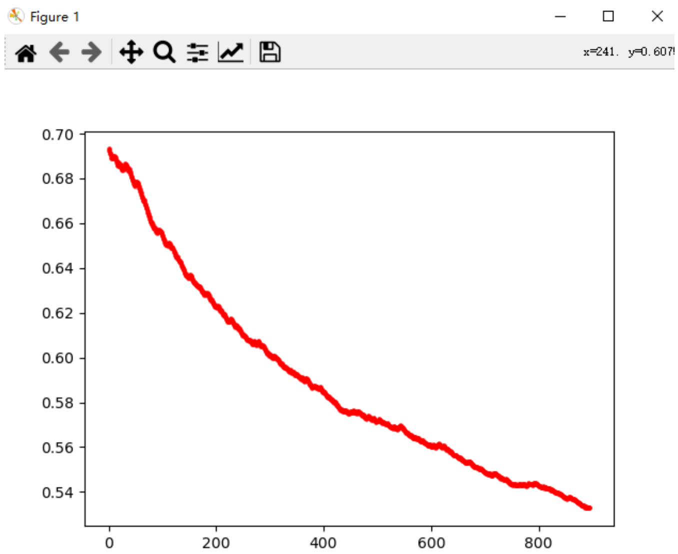

图 1: Result of Fixed Step SDG

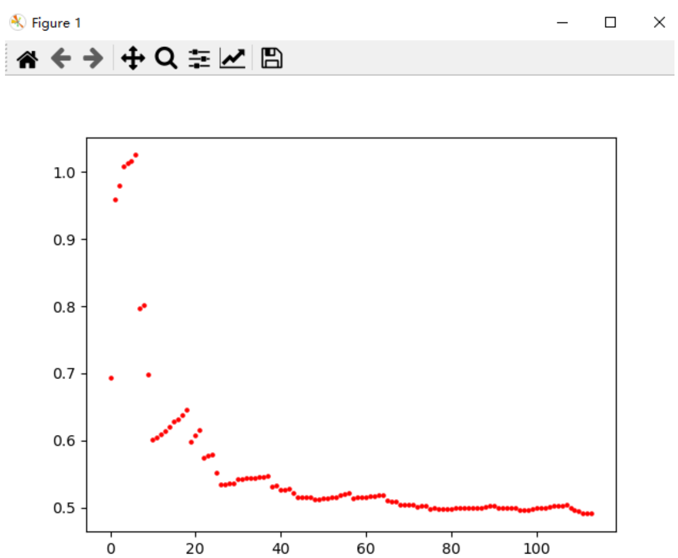

图 2: Result of Descent Step SDG

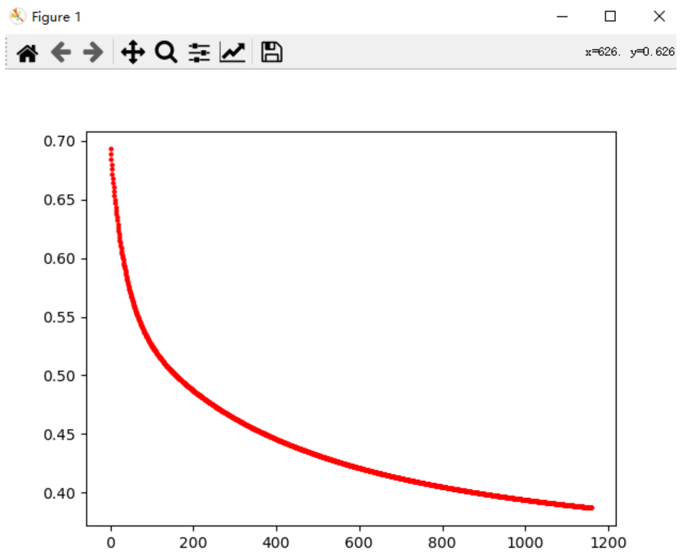

图 3: Result of SVRG