



西安交通大学
XI'AN JIAOTONG UNIVERSITY

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：鄧嘯淇

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

西安交通大学实验报告

专业: 大数据管理及应用
姓名: 鄧嘯淇
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 Chen 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

$$\begin{aligned} & D_\phi(x, y) + D_\phi(z, x) - D_\phi(z, y) \\ &= \phi(x) - \phi(y) - \nabla\phi(y)^T(x - y) + \phi(z) - \phi(x) - \nabla\phi(x)^T(z - x) - \phi(z) + \phi(y) + \nabla\phi(y)^T(z - y) \\ &= \nabla\phi(y)^T(z - y - x + y) - \nabla\phi(x)^T(z - x) \\ &= (\nabla\phi(x) - \nabla\phi(y))^T(x - z) \end{aligned}$$

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
 $\pi_\omega^\phi(y) = \operatorname{argmin}_{x \in \omega} D_\omega(x, y)$
 $\nabla D_\phi(x, y) = \nabla\phi(x) - \nabla\phi(y)$
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
 $(\nabla\phi(\pi_\omega^\phi(y)) - \nabla\phi(y))^T(\pi_\omega^\phi(y) - z) = D_\phi(z, \pi_\omega^\phi(y)) + D_\phi(\pi_\omega^\phi(y), y) - D_\phi(z, y) \leq 0$
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

$$\begin{aligned} \min_x \quad & \frac{1}{2} \|\mathbf{x}_0 - \mathbf{x}\|^2 \\ \text{s.t.} \quad & A\mathbf{x}_0 = \mathbf{b} \end{aligned} \quad (1)$$

使用 Lagrange 乘数法，然后使用 KKT 条件求解 x^*

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

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

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

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

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

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

那么优化问题

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & A\mathbf{x}_0 = \mathbf{b} \end{aligned} \quad (2)$$

的投影梯度下降迭代算法

三、 HW3

1. I

$$\begin{aligned} E_{D_t} \|g^t\|^2 &= E \left\| \frac{1}{n_b} \sum_{i \in D_t} \nabla f_i(x^t) \right\|^2 \\ &= \frac{1}{n_b^2} \left(\sum_{i \in D_t} E[\|\nabla f_i(x^t)\|^2] + \sum_{i \in D_t, j \in D_t, i \neq j} (E[\|\nabla f_i(x^t)\|^2] * E[\|\nabla f_j(x^t)\|^2]) \right) \\ &= \frac{1}{n_b^2} (n_b \sigma^2 + n_b(n_b - 1) \|\nabla f(x^t)\| * \|\nabla f(x^t)\|) \\ &= \frac{\sigma^2}{n_b} + \|\nabla f(x^t)\|^2 \end{aligned}$$

得证

2. II

在 A1 条件下，有

$$\begin{aligned} f(x^{t+1}) &\leq f(x^t) + \langle \nabla f(x^t), x^{t+1} - x^t \rangle + \frac{\beta}{2} \|x^{t+1} - x^t\|^2 \\ &= f(x^t) - S_t \langle \nabla f(x^t), \nabla f_{i_t}(x^t) \rangle + \frac{\beta S_t^2}{2} \|\nabla f_{i_t}(x^t)\|^2 \\ &= f(x^t) - S_t \langle \nabla f(x^t), \frac{1}{n_b} \sum_{i \in D_t} \nabla f_i(x^t) \rangle + \frac{\beta S_t^2}{2} \left\| \frac{1}{n_b} \sum_{i \in D_t} \nabla f_i(x^t) \right\|^2 \end{aligned}$$

对式子两边取期望，即得证。

3. III

在 A1 和 A2 的条件下

$$\begin{aligned}
 E[f(x^{t+1}) - f(x^t)] &\leq \frac{\beta S_t^2}{2} E_{D_t}[\|g^t\|^2] - S_t \nabla f(x)^T E_{D_t}(g^t) \\
 &\leq \frac{\beta S_t^2}{2} (\frac{\sigma^2}{n_b} + \|\nabla f(x^t)\|^2) - S_t \|\nabla f(x^t)\|^2 \\
 &= \frac{\beta S_t^2}{2n_b} \sigma^2 - (s - \frac{\beta S_t^2}{2}) \|\nabla f(x^t)\|^2
 \end{aligned}$$

得证

4. IV

在 A1 和 A2 的条件下

令 $S_t \in (0, \frac{1}{\beta}]$

有引理

$$f(x) - f^* \leq \frac{1}{2\alpha} \|\nabla f(x^t)\|^2$$

则有

$$\begin{aligned}
 E_{D_t}[f(x^{t+1}) - f(x^t)] &= \frac{\beta S_t^2}{2n_b} \sigma^2 - (s - \frac{\beta S_t^2}{2}) \|\nabla f(x^t)\|^2 \\
 &\leq \frac{\beta S_t^2}{2n_b} \sigma^2 - \alpha S_t (f(x^t) - f^*) \\
 \Rightarrow E_{D_t}[f(x^{t+1}) - f^* + f^* - f(x^t)] &\leq \frac{\beta S_t^2}{2n_b} \sigma^2 - \alpha S_t (f(x^t) - f^*) \\
 \Rightarrow E_{D_t}[f(x^{t+1}) - f^*] &\leq \frac{\beta S_t^2}{2n_b} \sigma^2 - (1 - \alpha) S_t (f(x^t) - f^*) \\
 \Rightarrow E_{D_t}[f(x^{t+1}) - f^*] - \frac{\beta S_t}{2n_b \alpha (2 - \beta S_t)} \sigma^2 &\leq \frac{\beta S_t^2}{2n_b} \sigma^2 - \frac{\beta S_t}{2n_b \alpha (2 - \beta S_t)} \sigma^2 + (1 - \alpha) S_t (f(x^t) - f^*) \\
 &\leq (1 - \alpha S_t (2 - \beta S_t)) (f(x^t) - f^* - \frac{\beta S_t}{2n_b \alpha (2 - \beta S_t)} \sigma^2)
 \end{aligned}$$

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

$$\min_{x \in \mathbf{R}} = \frac{1}{N} \sum_{i=1} N f_i(x) = \frac{1}{N} \sum_{i=1} N \ln(1 + \exp(-b_i * a_i^T x)) + \lambda \|x\|_2^2$$

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 data = pd.read_csv('a9a.csv')
6 data = data.values
7 n = data.shape[0]
8 data2 = []
9 for i in range(n):
10     a = data[i][0].split()
11     data2.append(a)
12 y = []
13 for i in range(n):
14     temp = int(data2[i][0])
15     y.append(temp)
16 y = np.array(y)
17 y = y.reshape(-1, 1)
18 c = []

```

```
19 for i in range(n):
20     b = []
21     for j in range(1, len(data2[i])):
22         index = data2[i][j].find(':')
23         b.append(int(data2[i][j][0:index]))
24     c.append(b)
25 m = max(max(i) for i in c)
26 X = np.zeros((n, m))
27 for i in range(n):
28     for j in range(len(c[i])):
29         X[i, c[i][j] - 1] = 1
30 X = np.matrix(X)
31
32 import math
33
34
35 def SGD_fix(A, b, lam):
36     k = np.random.randint(0, A.shape[0])
37     t = 0
38     x = np.zeros((A.shape[1], 1))
39     F = []
40     sum = 0
41     for i in range(A.shape[0]):
42         sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
43     f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
44     F.append(f)
45     deltaF = 1
46     while abs(deltaF / f) > 0.0000001:
47         k = np.random.randint(0, A.shape[0])
48         x = x - 0.001 * ((-math.exp(-b[k, :][0] * A[k, :] @ x) * b[k, :][0] * A[k, :].T) / (
49             1 + math.exp(-b[k, :][0] * A[k, :] @ x)) + 2 * lam * x)
50         sum = 0
51         for i in range(A.shape[0]):
52             sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
53         f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
54         F.append(f)
55         t = t + 1
56         deltaF = F[t] - F[t - 1]
57         print(deltaF / f, t)
58     plt.scatter(list(range(len(F))), F, s=5)
59     plt.show()
60     print(f)
61
62
63 SGD_fix(X, y, 0.01 / X.shape[0])
64
65
66 def SGD_dr(A, b, lam):
67     k = np.random.randint(0, A.shape[0])
68     t = 0
```

```

69     x = np.zeros((A.shape[1], 1))
70     F = []
71     sum = 0
72     for i in range(A.shape[0]):
73         sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
74     f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
75     F.append(f)
76     deltaF = 1
77     while abs(deltaF / f) > 0.0000001:
78         t = t + 1
79         k = np.random.randint(0, A.shape[0])
80         x = x - 1 / t * ((-math.exp(-b[k, :][0] * A[k, :] @ x) * b[k, :][0] * A[k, :].T) / (
81             1 + math.exp(-b[k, :][0] * A[k, :] @ x)) + 2 * lam * x)
82         sum = 0
83         for i in range(A.shape[0]):
84             sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
85         f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
86         F.append(f)
87         deltaF = F[t] - F[t - 1]
88         print(deltaF / f, t)
89     plt.scatter(list(range(len(F))), F, s=5)
90     plt.show()
91     print(f)
92
93
94 SGD_dr(X, y, 0.01 / X.shape[0])
95
96
97 def SVRG(A, b, lam):
98     t = 0
99     x = np.zeros((A.shape[1], 1))
100     y = x
101     F = []
102     sum = 0
103     for i in range(A.shape[0]):
104         sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
105     f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
106     F.append(f)
107     deltaF = 1
108     grad = 0
109     for i in range(A.shape[0]):
110         grad = grad + (-math.exp(-b[i, :][0] * A[i, :] @ x) * b[i, :][0] * A[i, :].T) / (
111             1 + math.exp(-b[i, :][0] * A[i, :] @ x)) + 2 * lam * x
112     gradF = grad / A.shape[0]
113     T = 0
114     while abs(deltaF / f) > 0.00001:
115         # while T<10000:
116         xs = []
117         t = 0
118         grad = 0

```

```
119     for i in range(A.shape[0]):
120         grad = grad + (-math.exp(-b[i, :][0] * A[i, :] @ y) * b[i, :][0] * A[i, :].T) / (
121             1 + math.exp(-b[i, :][0] * A[i, :] @ y)) + 2 * lam * y
122     grady = grad / A.shape[0]
123     while t < 20:
124         t = t + 1
125         T = T + 1
126         k = np.random.randint(0, A.shape[0])
127         gradkx = (-math.exp(-b[k, :][0] * A[k, :] @ x) * b[k, :][0] * A[k, :].T) / (
128             1 + math.exp(-b[k, :][0] * A[k, :] @ x)) + 2 * lam * x
129         gradky = (-math.exp(-b[k, :][0] * A[k, :] @ y) * b[k, :][0] * A[k, :].T) / (
130             1 + math.exp(-b[k, :][0] * A[k, :] @ y)) + 2 * lam * y
131         v = gradkx - (gradky - grady)
132         x = x - 0.01 * v
133         xs.append(x)
134         sum = 0
135         for i in range(A.shape[0]):
136             sum = sum + math.log(1 + math.exp(-b[i, :][0] * A[i, :] @ x))
137         f = (1 / A.shape[0]) * sum + lam * np.linalg.norm(x, ord=2) ** 2
138         F.append(f)
139         deltaF = F[T] - F[T - 1]
140         print(deltaF / f, T)
141     sumx = 0
142     for i in range(len(xs)):
143         sumx = sumx + xs[i]
144     y = sumx / len(xs)
145     plt.scatter(list(range(len(F))), F, s=5)
146     plt.show()
147     print(f)
148
149
150 SVRG(X, y, 0.01 / X.shape[0])
```

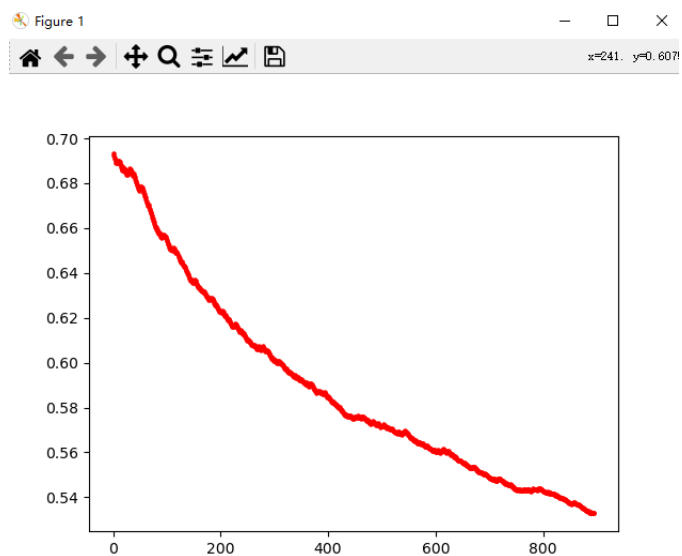


图 1: Result of Fixed Step SDG

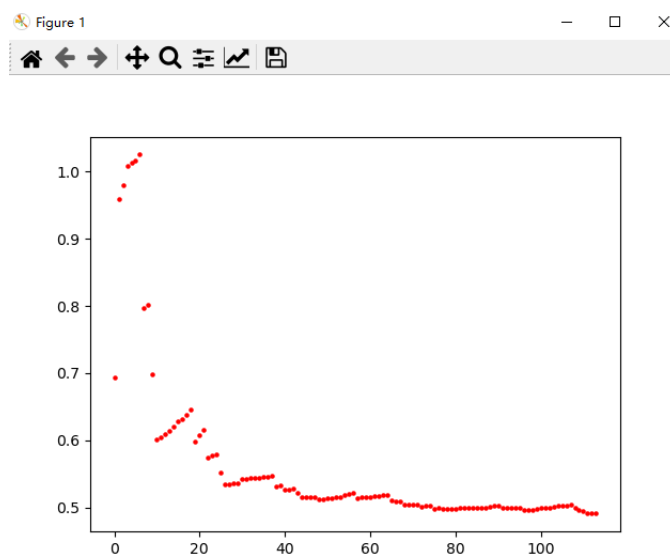


图 2: Result of Descent Step SDG

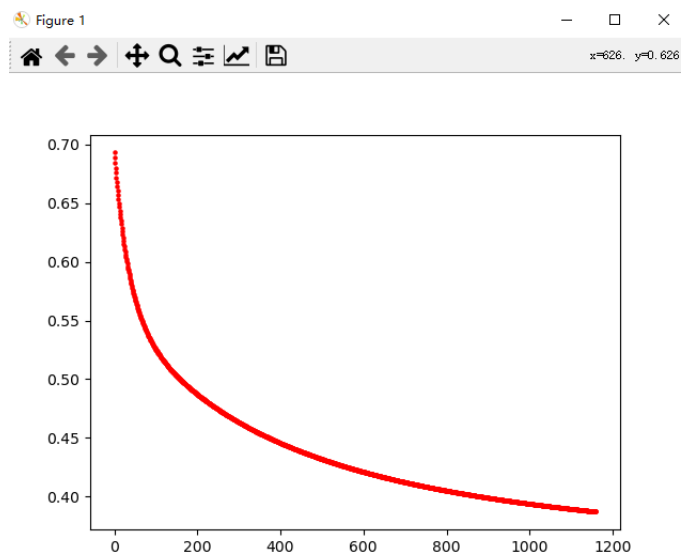


图 3: Result of SVRG