Homework 3: Steepest-descent method

สมาชิกกลุ่ม 4

1. เมธาวี สกุล 6220422078
2. อธิวัฒน์ หิรัญวรวงศ์กุล 6220422080
3. กิตติศักดิ์ สุคันธรัตน์ 6220422081
4. พรทิพย์ แก้วแหวน 6220422085
5. อภิญญา เกตุหนู 6220422086

Problem 1

$$minimize\ f(x) = \log\left(\sum_{i=1}^{m} e^{a_i^T x + b_i}\right)$$

ข้อมูลจากไฟล์ csv

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,100} \\ \vdots & \ddots & \vdots \\ a_{300,1} & \cdots & a_{300,100} \end{bmatrix}_{300x100} \qquad B = \begin{bmatrix} b_{1,1} \\ \vdots \\ b_{300,1} \end{bmatrix}_{300x1}$$

จากโจทย์

$$minimize\ f(x) = \log\left(\sum_{i=1}^{m} e^{a_i^T x + b_i}\right)$$

จะได้ว่า

$$A = \begin{bmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix}_{100x300} \qquad B = \begin{bmatrix} b_{1,1} \\ \vdots \\ b_{300,1} \end{bmatrix}_{300x1}$$

ให้ m = 300

พิจารณาที่ i = 1 จะได้

$$a_{(1)} = \begin{bmatrix} a_{1,1} \\ \vdots \\ a_{100,1} \end{bmatrix}_{100x1} \qquad b_{(1)} = [b_{1,1}]_{1x1}$$

$$a_{(1)}^T = [a_{1,1} \quad \cdots \quad a_{1,100}]_{1x100} \qquad x = \begin{bmatrix} x_1 \\ \vdots \\ x_{100} \end{bmatrix}_{100x1}$$

$$a_{(1)}^T x + b_{(1)} = [a_{1,1} \quad \cdots \quad a_{100,1}]_{1x100} \begin{bmatrix} x_1 \\ \vdots \\ x_{100} \end{bmatrix}_{100x1} + [b_{1,1}]_{1x1}$$

สำหรับ $\sum_{i=1}^{m} e^{a_i{}^T x+b_i}$

พิจารณาที่ $a_i{}^T x + b_i$ เมื่อ $i = 1, \dots, m$

$$a_i{}^T x + b_i = \begin{bmatrix} a_{1,1} & \cdots & a_{1,100} \\ \vdots & \ddots & \vdots \\ a_{300,1} & \cdots & a_{300,100} \end{bmatrix}_{300\ x\ 100} \cdot \begin{bmatrix} x_{1,1} \\ \vdots \\ x_{100,1} \end{bmatrix}_{100\ x\ 1} + \begin{bmatrix} b_{1,1} \\ \vdots \\ b_{300,1} \end{bmatrix}_{300\ x\ 1}$$

$$= \begin{bmatrix} \vdots \end{bmatrix}_{300\ x\ 1}$$

$$f(x) = \log\left(\sum_{i=1}^{m} e^{a_i{}^T x+b_i}\right)$$

$\nabla f(x)$ เมื่อ $x \in \mathbb{R}^n$ โดยที่ n = 100

พิจารณาที่ x_1

$$\frac{\partial f(x)}{\partial x_1} = \frac{1}{\sum_{i=1}^{m} e^{a_i{}^T x+b_i}} \times \sum_{i=1}^{m} e^{a_i{}^T x+b_i} \times a_{i1}$$

พิจารณาที่ x_2

$$\frac{\partial f(x)}{\partial x_2} = \frac{1}{\sum_{i=1}^{m} e^{a_i{}^T x+b_i}} \times \sum_{i=1}^{m} e^{a_i{}^T x+b_i} \times a_{i2}$$

พิจารณาที่ x_n

$$\frac{\partial f(x)}{\partial x_n} = \frac{1}{\sum_{i=1}^{m} e^{a_i{}^T x+b_i}} \times \sum_{i=1}^{m} e^{a_i{}^T x+b_i} \times a_{in}$$

จะได้ว่า $\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}_{n\ x\ 1}$

จาก General optimization algorithm

$$x^{(k+1)} = x^{(k)} + t_k \Delta \mathrm{x}^{(\mathrm{k})}$$

หา Steepest-descent method โดยการแทน

$$\Delta x^{(k)} = -\nabla f(x^{(k)})$$

หา FISTA โดยจะมีการเพิ่ม γ เข้ามาเพื่อถ่วงน้ำหนัก

Initial $x^{(0)}$ and set $y^{(1)} = x^{(0)}$, $\gamma_1 = 1$, $t_k = 0.1$

$$x^{(k)} = y^{(k)} - t_k \nabla \mathrm{f}(\mathrm{y}^{(\mathrm{k})})$$

$$\gamma_{k+1} = \frac{1 + \sqrt{1 + 4\gamma_k^2}}{2}$$

$$y^{(k+1)} = x^{(k)} + \left(\frac{\gamma_k - 1}{\gamma_{k+1}}\right)\left(x^{(k)} - x^{(k-1)}\right)$$

<u>Coding</u>

- $f(x)$

```
def f(x):
  return np.log(np.sum(np.exp((A @ x) + B)))
```

- $\nabla f(x)$

```
def f_diff1(x):
  results = np.empty_like(x)
  for i in range(A.shape[1]):
      a = A[:,i]
      result = np.exp((A @ x) + B).reshape(A.shape[0],)
      result = (1/np.sum(np.exp((A @ x) + B))) * np.dot(result,a)
      results[i] = [result]
  return results
```

- $find\ f(x)\ using\ Steepest\ descent$

```
def steepest_descent(x0,t=0.1,k=200):
  x_hist = [x0]
  for i in range(k):
      if i == 0:
          x_prev = x0
      else:
          x_prev = x_hist[i-1]
      x_prev_delta = -1 * f_diff1(x_prev)
      x_hist.append(x_prev + (t * x_prev_delta))
  return [f(x) for x in x_hist]
```

- $find\ f(x)\ using\ FISTA$

```
def fista(x0, t = 0.1, k = 200):
  x_hist = [x0]
  gamma_curr = 1
  y_next = x0

  for i in range(k):
      x = y_next - (t*f_diff1(y_next))
      gamma_next = (1 + math.sqrt(1 + (4*(math.pow(gamma_curr,2))))))/2
      gamma_curr = gamma_next
      y_next = x + (((gamma_curr - 1)/gamma_next)*(x - x_hist[-1]))
      x_hist.append(x)
  return [f(x) for x in x_hist]
```

- $find\ p^*\ from\ \min\ of\ f(x)\ using\ FISTA\ at\ k = 1000\ iterations$
- $initial\ x$

```
# x is in R^n
n = 100
f_opt = min(fista(np.random.rand(n,1),0.1,1000))
print('p*:',f_opt)
```

```
p*: 5.610898008828295
```

- $initial\ x_0, t_0, k(\text{maximum}\ interation)$

```
# intital
t = 0.1
max_interations = 200
x0 = np.random.rand(n,1)
```

- $find\ f(x)\ using\ Steepest\ descent\ and\ FISTA\ in\ each\ interation$

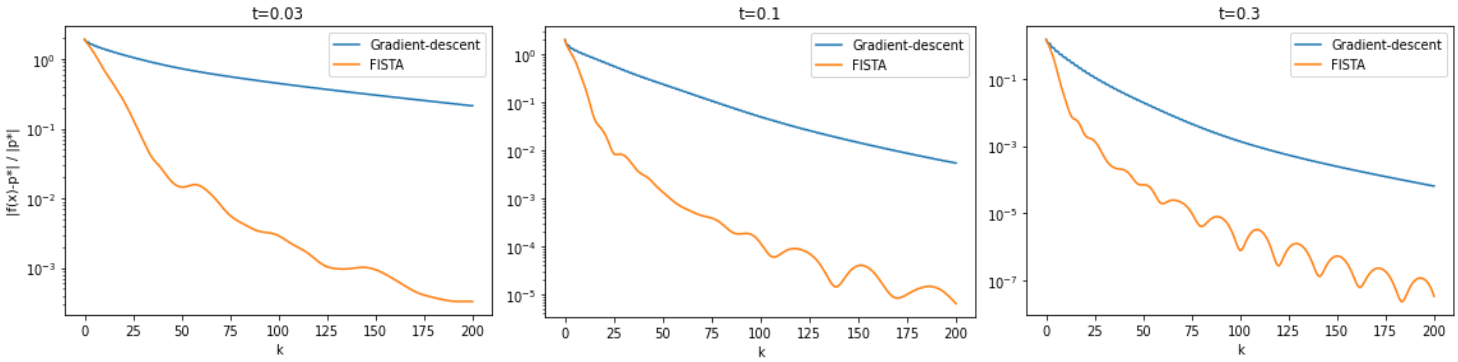
```
f_hist_SD = steepest_descent(x0,t,max_interations)
f_hist_FISTA = fista(x0,t,max_interations)
```

- $compare\ \frac{|f(x)-p^*|}{|p^*|}$

```
f_err_SD = [abs(f_curr - f_opt)/abs(f_opt) for f_curr in f_hist_SD]
f_err_FISTA = [abs(f_curr - f_opt)/abs(f_opt) for f_curr in f_hist_FISTA]

import matplotlib.pyplot as plt
plt.plot([x for x in range(max_interations+1)], f_err_SD, label='Gradient-descent
')
plt.plot([x for x in range(max_interations+1)], f_err_FISTA, label='FISTA')
plt.yscale('log')
plt.xlabel("k")
plt.ylabel("|f(x)-p*| / |p*|")
plt.legend(loc='upper right')
plt.show()
```

Result

จากกราฟที่ได้ จะเห็นว่าวิธีของ FISTA สามารถ Converge ได้เร็วกว่า วิธีของ Gradient-descent และเมื่อทดลองปรับค่า t=0.03 จะพบว่าการ Converge จะช้ากว่า t=0.1 และเมื่อปรับค่า t=0.3 จะพบว่าการ Converge จะเร็วมากกว่า t=0.1 ดังนั้นจะเห็นว่าการกำหนดค่า t นั้นมีผลต่อการ Converge ของ $f(x)$



Problem 2

$\min_{x} \frac{1}{2} \left(f(x) \coloneqq \frac{1}{2}(x_1^2 + ax_2^2)\right.$ using steepest descent method

2.1 Derived the step size from the exact line search

$$\nabla f(x) = \frac{1}{2}\begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2x_1 + 0 \\ 0 + 2ax_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ ax_2 \end{bmatrix}$$

General optimization algorithm $x^{(k+1)} = x^k + t_k \Delta x^{(k)}$

Steepest-descent method $\Delta x^{(k)} = -\nabla f\left(x^{(k)}\right)$

$$\therefore\ x^{(k+1)} = x^k - t_k \nabla f\left(x^{(k)}\right)$$

Substituted $-\nabla f(x)$ in steepest descent equation

$$x^{(k+1)} = x^k - t_k \begin{bmatrix} x_1 \\ ax_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} t_k x_1 \\ t_k a x_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 - t_k x_1 \\ x_2 - t_k a x_2 \end{bmatrix} = \begin{bmatrix} (1 - t_k)x_1 \\ (1 - t_k a)x_2 \end{bmatrix}$$

$$f\left(x^{(k+1)}\right) = \frac{1}{2}\left[((1 - t_k)x_1)^2 + a\left((1 - t_k a)x_2\right)^2\right]$$

Exact the step size from the exact line search by choosing t_k from

$$t_k = arg\min_{t \geq 0} f(x^{(k)} + t\Delta x^{(k)})$$

Hence, using the first derivative to find extrema by setting function equal to zero.

Note:

When the first derivative technique of a function equal to 0, the function is at the minimum point. However, the technique can’t distinguish between the global minimum and the global maximum.

$$\frac{df\left(x^{(k+1)}\right)}{dt} = \frac{1}{2}[2x_1(1-t)(-x_1) + 2ax_2(1-at)(-ax_2)]$$
$$= -x_1^2(1-t) - a^2x_2^2(1-at)$$

Set equation to zero

$$-x_1^2(1-t) - a^2x_2^2(1-at) = 0$$
$$-x_1^2 + x_1^2t - (a^2x_2^2 - a^3x_2^2t) = 0$$
$$x_1^2t + a^3x_2^2t = x_1^2 + a^2x_2^2$$
$$t = \frac{x_1^2 + a^2x_2^2}{x_1^2 + a^3x_2^2}$$

$$\therefore\ t_k = \frac{x_1^{(k)^2} + a^2x_2^{(k)^2}}{x_1^{(k)^2} + a^3x_2^{(k)^2}}$$ __________implemented in code

2.2 Express the steepest-descent update rule by using the exact line search method.

From $\nabla f(x) = \begin{bmatrix} x_1 \\ ax_2 \end{bmatrix} \rightarrow x^t = x - t\nabla f(x) = \begin{bmatrix} x_1 - tx_1 \\ x_2 - tax_2 \end{bmatrix} = \begin{bmatrix} (1-t)x_1 \\ (1-ta)x_2 \end{bmatrix}$

Hence, t

$$= \begin{bmatrix} \left(1 - \left(\dfrac{x_1^{(k)^2} + a^2x_2^{(k)^2}}{x_1^{(k)^2} + a^3x_2^{(k)^2}}\right)\right)x_1 \\ \hline \left(1 - \left(\dfrac{x_1^{(k)^2} + a^2x_2^{(k)^2}}{x_1^{(k)^2} + a^3x_2^{(k)^2}}\right)a\right)x_2 \end{bmatrix}$$

2.3 Use $x^{(0)} = \begin{bmatrix} 9 \\ 1 \end{bmatrix}$. Plot the sequences of $x^{(k)}$ on R^2 plane and the function contour and use a = 2 and a = 10 (two graphs).

Step 1: Import necessary python library

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

Step 2: Define functions

2.1 Objective function

```
def f(x,a):
    x1 = float(x[0])
    x2 = float(x[1])
    return 1/2 * (math.pow(x1, 2) + (a * math.pow(x2, 2)))
```

2.2 Δx

```
def delta_x(x,a):
    x1 = float(x[0]) * -1
    x2 = float(x[1]) * -1 * a
    return np.array([[x1],[x2]])
```

2.3 Steepest Descent function

```
def steepest_descent(x0,a,k=200):
    x_hist = [x0]
    x_next = x0
    for i in range(k):
        x_next = x_next + (t(x_next,a)*delta_x(x_next,a))
        x_hist.append(x_next)
    return x_hist,[f(x,a) for x in x_hist]
```

Step 4: Run Steepest descent method function

4.1 Use a1 = 2

```
a1 = 2
x_hist_a1, f_hist_a1 = steepest_descent(x0, a1, k)
```

4.2 Use a2 = 10

```
a2 = 10
x_hist_a2, f_hist_a2 = steepest_descent(x0, a2, k)
```

Step 5: Plot contour to compare the behavior

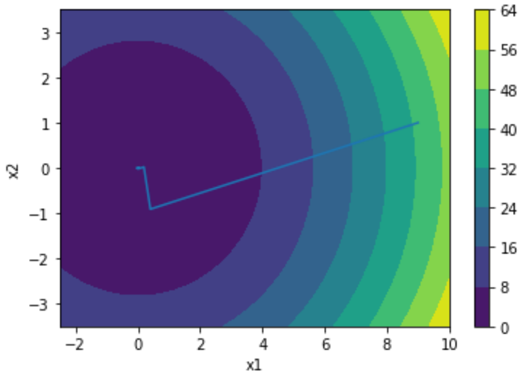
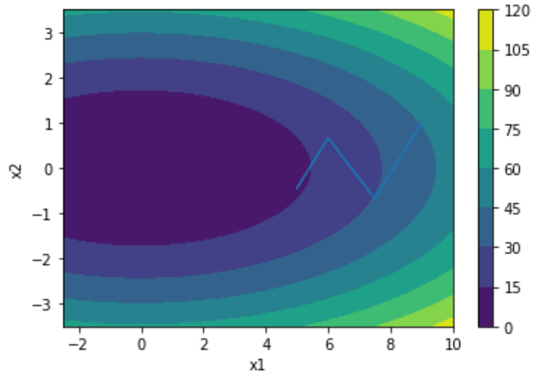
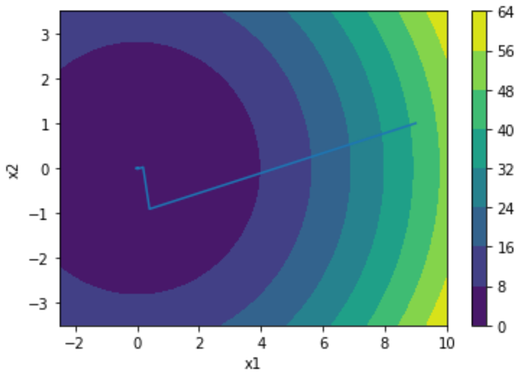
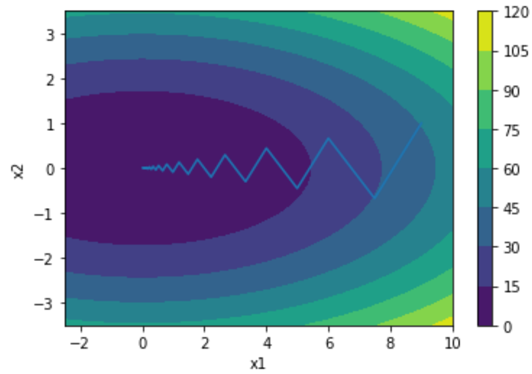
5.1 Plot a1 = 2

```
x1_plot_a1 = [float(x[0]) for x in x_hist_a1]
x2_plot_a1 = [float(x[1]) for x in x_hist_a1]

xlist = np.linspace(-5.0, 10.0, 100)
ylist = np.linspace(-1.5, 1.5, 100)
X, Y = np.meshgrid(xlist, ylist)
Z = 1/2 * (X**2 + (a * Y**2))
fig, ax = plt.subplots(1,1)
cp = ax.contourf(X, Y, Z)
fig.colorbar(cp)
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.plot(x1_plot_a1, x2_plot_a1)
plt.show()
```

5.1 Plot a2 = 10

```
x1_plot_a2 = [float(x[0]) for x in x_hist_a2]
x2_plot_a2 = [float(x[1]) for x in x_hist_a2]

xlist = np.linspace(-5.0, 10.0, 100)
ylist = np.linspace(-1.5, 1.5, 100)
X, Y = np.meshgrid(xlist, ylist)
Z = 1/2 * (X**2 + (a * Y**2))
fig, ax = plt.subplots(1,1)
cp = ax.contourf(X, Y, Z)
fig.colorbar(cp)
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.plot(x1_plot_a2, x2_plot_a2)
plt.show()
```

Result

k	a=2	a=10
3	 Figure 1 Show the behavior of $x^{(k)}$ when a = 2, k=3	 Figure 2 Show the behavior of $x^{(k)}$ when a = 10, k=3
30	 Figure 3 Show the behavior of $x^{(k)}$ when a = 2, k=30	 Figure 4 Show the behavior of $x^{(k)}$ when a = 10, k=30

จากรูปภาพที่ 1 และรูปภาพที่ 2 ค่า a ส่งผลกับ step size เพราะว่าเมื่อพิจารณาที่ a =2 นั้น ค่า step size ที่ iteration แรกนั้นมีค่าใกล้เคียง 1 (0.955) เมื่อนำ step size ไปคูณกับ gradient ทิศทางตรงกันข้าม $\begin{bmatrix}-9\\-2\end{bmatrix}$ ผลลัพธ์ของ vector output จึงขยับเข้ามาใกล้จุด optimum

แต่เมื่อพิจารณาที่ a = 10 ค่า step size ที่ iteration แรกได้นั้นมีค่าแค่ 0.167 เมื่อนำ step size ที่ได้ไปคูณกับ gradient ทิศทางตรงกันข้าม $\begin{bmatrix}-9\\-10\end{bmatrix}$ ผลลัพธ์จึงมีทิศที่เบี่ยงออกไปทาง gradient ทิศตรงข้าม โดยที่ vector ผลลัพธ์จะมีขนาดเล็กเพราะค่า step size ที่น้อย

และเมื่อพิจารณาที่ iteration (k) ค่า a ที่น้อยส่งผลให้ลู่เข้า (converge) สู่จุด optimum ได้เร็ว แต่ในทางกลับกันเมื่อค่า a สูงขึ้นส่งผลให้ลู่เข้าสู่จุด optimum ได้ช้ากว่า จากรูปภาพที่ 4 เมื่อ a = 10 ต้องมีการทำ iteration ถึง 30 รอบจึงจะลู่เข้าจุด optimum และส่วนหนึ่งก็มาจากผลของค่า a ที่มากทำให้ gradient ของ vector ทิศทางตรงกันข้ามนั้นเบี่ยงออกไปไกลด้วยเช่นกัน