Homework report 1

Name: Ma xiaoqi
grasper: 308932

(the scan of pdf can be enlarged)

Problem 2

Problem 1: ($\vec{\ }$ means vector)

For n points of form: $(\vec{a_1}, b_1) \cdots (\vec{a_n}, b_n)$

$\therefore L_{SH}(\vec{x}, \mu) = \frac{1}{n}\sum_{i=1}^{n} g_i(\vec{x}, \mu)$

$$g_i(\vec{x}, \mu) = \begin{cases} \frac{1}{2} - b_i(\vec{a_i}^T\vec{x} + \mu) & b_i(\vec{a_i}^T\vec{x}+\mu) \le 0 \Rightarrow I_L(i,i) = 1 \\ \frac{1}{2}(1 - b_i\vec{a_i}^T\vec{x} + \mu))^2 & 0 < b_i(\vec{a_i}^T\vec{x}+\mu) \le 1 \Rightarrow I_Q(i,i) = 1 \\ 0 & 1 \le b_i(\vec{a_i}^T\vec{x}+\mu) \Rightarrow I_0(i,i) = 1 \end{cases}$$

(a): In problem a, $\mu = 0$ $\tilde{A}\vec{x} = \underbrace{[b_1\vec{a_1}, \cdots b_n\vec{a_n}]^T}_{n\times p}\underbrace{\vec{x}}_{p\times 1}$ $[\tilde{A}\vec{x}]_{n\times 1}$

~~$\therefore L_{SH}(\vec{x}) = \frac{1}{n}$~~ we use $\vec{1}^T = [1, 1, 1 \cdots 1]$

$$\therefore L_{SH}(\vec{x}) = \frac{1}{n}\left\{\frac{1}{2}\vec{1}^T I_L \tilde{A}\vec{x} + \frac{1}{2}\| I_Q\vec{1} - I_Q\tilde{A}\vec{x}\|^2\right\}$$

$$\therefore \nabla L_{SH}(\vec{x}) = \frac{1}{n}\left\{\nabla(-\vec{1}^T I_L\tilde{A}\vec{x}) + \frac{1}{2}\nabla\| I_Q\vec{1} - I_Q\tilde{A}\vec{x}\|^2\right\}$$

$$= \frac{1}{n}\left\{-(\tilde{A}^T I_L^T\vec{1}) + [-(\tilde{A}^T I_Q^T)(I_Q\vec{1} - I_Q\tilde{A}\vec{x})]\right\}$$

$$= \frac{1}{n}\left\{\tilde{A}^T I_Q^T I_Q\tilde{A}\vec{x} - \tilde{A}^T I_Q^T I_Q\vec{1} - \tilde{A}^T I_L^T\vec{1}\right\} \quad (I_Q^T \cdot I_Q = I_Q)$$

$$= \frac{1}{n}\left\{\tilde{A}^T I_Q(\tilde{A}\vec{x} - \vec{1}) - \tilde{A}^T I_L^T\vec{1}\right\}$$

$\therefore f(\vec{x}) = L_{SH}(\vec{x}) + \frac{\lambda}{2}\|\vec{x}\|^2$ $\nabla f(\vec{x}) = \nabla L_{SH}(\vec{x}) + \lambda\vec{x}$

$\therefore \nabla f(\vec{x}) = \lambda\vec{x} + \frac{1}{n}\tilde{A}^T I_Q(\tilde{A}\vec{x} - \vec{1}) - \frac{1}{n}\tilde{A}^T I_L\vec{1}$ ☆

If there exist L ($L > 0$), $\|\nabla f(\vec{x}) - \nabla f(\vec{y})\| \le L\|\vec{x} - \vec{y}\|$

$$\therefore \|\nabla f(\vec{x}) - \nabla f(\vec{y})\| = \|\lambda\vec{x} - \lambda\vec{y} + \frac{1}{n}\tilde{A}^T I_Q\tilde{A}(\vec{x} - \vec{y}) + 0\|$$

$$= \|\lambda(\vec{x} - \vec{y}) + \frac{1}{n}\tilde{A}^T\tilde{A}(\vec{x} - \vec{y})\| \le \lambda\|\vec{x} - \vec{y}\| + \frac{1}{n}\|\tilde{A}^T\tilde{A}(\vec{x} - \vec{y})\|$$

$$\le [\lambda + \frac{1}{n}\|\tilde{A}^T\|\,\|A\|] \cdot \|\vec{x} - \vec{y}\|$$

$\therefore \|A\| = \|\tilde{A}\|$ $\|A^T\| = \|\tilde{A}^T\|$

$\therefore L = \lambda + \frac{1}{n}\|A^T\|\|A\|$ and ∇f is L-Lipschitz

(b) $\nabla f(\vec{x}) = \lambda\vec{x} + \frac{1}{n}\tilde{A}^T I_Q(\tilde{A}\vec{x} - \vec{1}) - \frac{1}{n}\tilde{A}^T I_L\vec{1}$

$\because I_Q = \mathbb{I}$, cut the constant term.

$\nabla^2 f(\vec{x}) = \nabla(\lambda\vec{x} + \frac{1}{n}\tilde{A}^T\mathbf{I}\tilde{A}\vec{x})$

for the part $\nabla(\lambda\vec{x})$ $\lambda\vec{x} = \begin{bmatrix}\lambda & & \\ & \ddots & \\ 0 & & \lambda\end{bmatrix}_{\text{diagonal}(\lambda)}\vec{x} = \begin{bmatrix}\lambda & \cdots & \\ \vdots & \ddots & \\ 0 & & \lambda\end{bmatrix}_{p\times p}\begin{bmatrix}x_1 \\ \vdots \\ x_p\end{bmatrix} =$

$\therefore \nabla(\lambda\vec{x}) = \nabla(\lambda\mathbb{I}\vec{x}) = (\lambda\mathbb{I})^T = \lambda\mathbb{I}$

for the part $\nabla(\frac{1}{n}\tilde{A}^T I\tilde{A}\vec{x}) = \nabla(\frac{1}{n}\tilde{A}^T\tilde{A}\vec{x}) = \frac{1}{n}(\tilde{A}^T\tilde{A})^T = \frac{1}{n}\tilde{A}^T(\tilde{A}^T)^T = \frac{1}{n}\tilde{A}^T\tilde{A}$

~~$\because \tilde{A}^T = bA^T$ $\tilde{A} = Ab^T$~~ $\because b \in \{1, -1\}$ $\therefore [b_1\vec{a_1}, b_2\vec{a_2} \cdots b_n\vec{a_n}]\begin{bmatrix}b_1\vec{a_1} \\ \vdots \\ b_n\vec{a_n}\end{bmatrix}$

~~$\because b \in \{1, -1\}$~~ $b_i \cdot b_i = 1$ $\therefore \tilde{A}^T\cdot\tilde{A} = A^TA$

$\therefore \nabla^2 f(x) = \lambda\mathbb{I} + \frac{1}{n}A^TA$

$\nabla^2 f(x)$ does not depend on $\vec{x}$ and f is twice-differentiable at $\vec{x}$

~~(b) f(x)~~ because the partial derivations are continuous at x
(∇f)

(c) $f(\vec{x}) = L_{SH}(\vec{x}, M) + \frac{\lambda}{2}\|x\|^2$

So we need to improve the $L_{SH}(\vec{x})$ is convex. ($\mu = 0$)

$\nabla L_{SH}(\vec{x}) = \frac{1}{n}\tilde{A}^T I_Q(\tilde{A}\vec{x} - \vec{T}) - \frac{1}{n}\tilde{A}^T I_L^T \vec{T}$

$\nabla^2 L_{SH}(\vec{x}) = \frac{1}{n}(\tilde{A}^T I_Q \tilde{A})^T = \frac{1}{n}\tilde{A}^T \cdot (I_Q)^T (\tilde{A}^T)^T = \frac{1}{n}\tilde{A}^T I_Q \tilde{A}$

This is the Hessian Matrix of $L_{SH}(\vec{x})$

So we need to prove Matrix $\tilde{A}^T I_Q \tilde{A}$ is positive semi-definite..

We set the Martix $H = \tilde{A}^T I_Q \tilde{A}$

Then, ~~set the~~ For any vector $\vec{P}$, $\vec{P}^T$

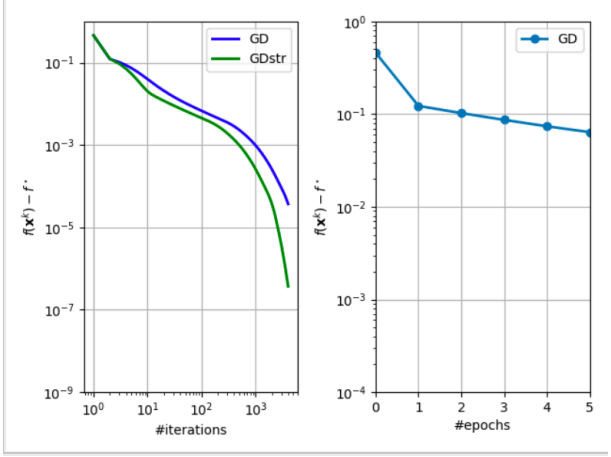
~~$\vec{P} \cdot H \cdot \vec{P}^T$~~ $\vec{P}^T H \vec{P} = \vec{P}^T \tilde{A}^T I_Q^T I_Q \tilde{A} \vec{P} = \|I_Q \tilde{A} \vec{P}\|^2 \geq 0$

∴ H is the positive semi-definite

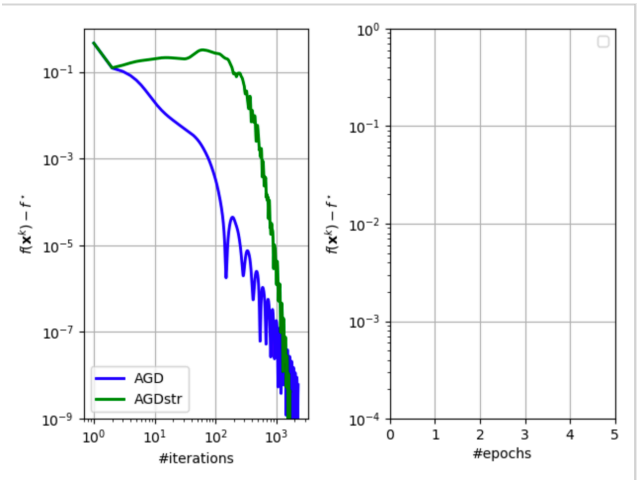
∴ $L_{SH}(\vec{x})$ is convex

$f(\vec{x})$ is λ-strongly convex

(a)GD and GDstr
This problem asks us to implement the GD and GDstr and compare some difference between them.GDstr change the parameter alpha (step-size), and we can get the outcomes as following:
GD:
Set the max iteration is 4000, after 3995 iters,
f(x)=0.039934508
Error w.r.t 0-1 loss: 0.029197080291970802
GDstr
Set the max iteration is 4000, after 3995 iters,
f(x)= 0.039897574
Error w.r.t 0-1 loss: 0.029197080291970802
Although the error is the same, we can find the GDstr converges faster thanGD because the existence of strong convexity(from figure)



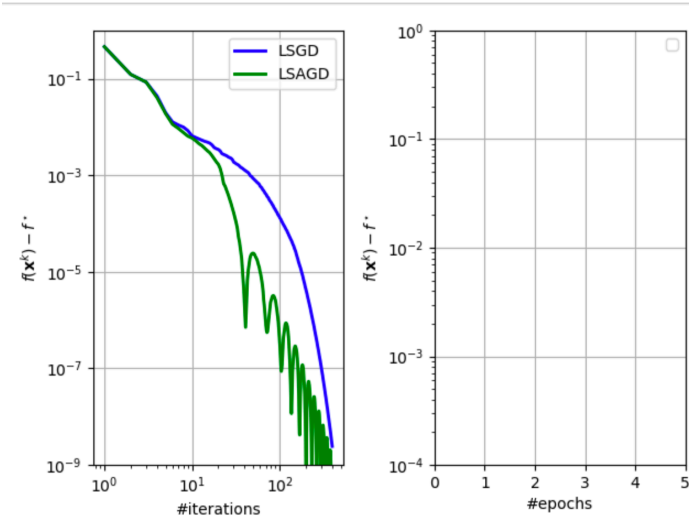
(b)AGD and AGDstr
Implementing the accelerate gradient descent method is to make a faster convergence than gradient decent, I get outcomes as following:
AGD:
Set the max iteration is 2300, after 2295 iters,
f(x) = 0.039897205
Error w.r.t 0-1 loss: 0.029197080291970802
AGDstr:
Set the max iteration is 1700, after 1696,
f(x) = 0.039897203
Error w.r.t 0-1 loss: 0.029197080291970802



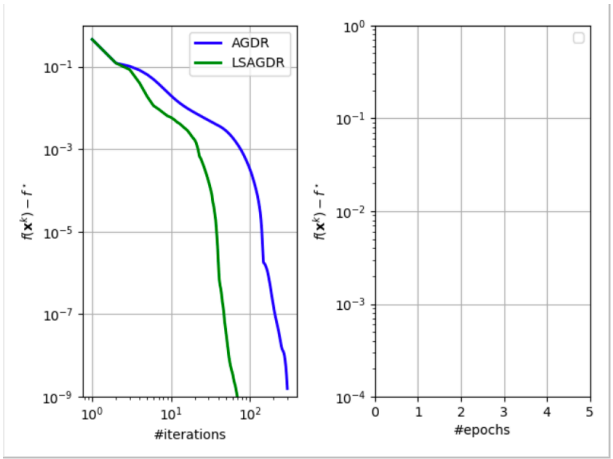
However, after about10^2 iterations, I

find the AGD algorithm performs oscillation by a wide margin, AGDstr also has oscillation but the frequency is low. At the same time, AGDstr just use less iterations than AGD to achieve the same performance.

(c)LSGD and LSAGD
Then, implement a line search procedure which aims at deterring the stepsize of alpha, I get the outcomes as following:
LSGD:
Set the max iteration is 400, after 395 iters,
f(x) = 0.039897202
Error w.r.t 0-1 loss: 0.029197080291970802
LSAGD:
Set the max iteration is 400, after 395 iters,
f(x) = 0.039897199
Error w.r.t 0-1 loss: 0.029197080291970802
I find the LSAGD has the faster convergence than LSGD, but LSAGD performs oscillation after 50 iterations, so we need improvement.



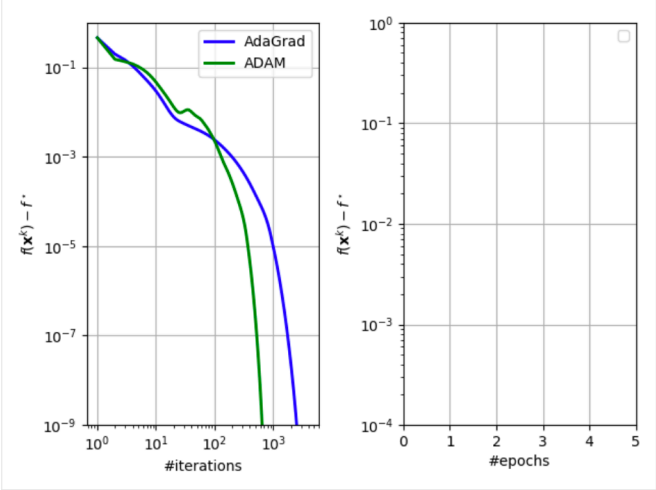
(d)AGDR and LSAGDR
In order to solve the problem that the the accelerated gradient descent can be oscillatory, implement the adaptive restart strategy. I get the outcomes as following:
AGDR: Set the max iteration is 300, after 295 iters,
f(x) = 0.039897201
Error w.r.t 0-1 loss: 0.029197080291970802
LSAGDR:Set the max iteration is 100, after 295 iterations
f(x) = 0.039897199
Error w.r.t 0-1 loss: 0.029197080291970802



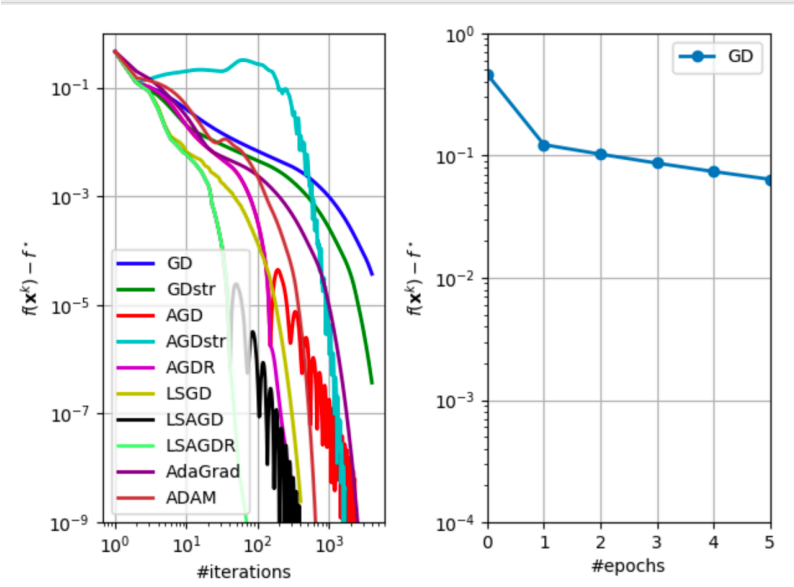
Compared with the AGD, the phenomenon of oscillation has disappeared and LSAGDR converges faster than AGDR

(e)(f)AdaGrad and ADAM

I implement the adaptive optimization method as following:
AdaGrad: Set the max iteration is 4000, after 3995 iterations
f(x) = 0.039897199
Error w.r.t 0-1 loss: 0.029197080291970802
ADAM:Set the max iteration is 4000, after 3995 iterations
f(x) = 0.039897199
Error w.r.t 0-1 loss: 0.029197080291970802



ADAM is better than AdaGrad
Plot about program 2



Problem 3

(a) ① $\because \nabla f(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\vec{x})$

$E_{i_k}[\nabla f_{i_k}(\vec{x})] = \sum_{i=1}^{n} P(i_k = i)\nabla f_i(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\vec{x}) = \nabla f(\vec{x})$

$\therefore \nabla f_{i_k}(\vec{x})$ is an unbiased estimation of $\nabla f(\vec{x})$

② $\nabla f_i(\vec{x}) = \lambda\vec{x} + \vec{1}_{\{0 < b_i \vec{a}_i^T \vec{x} \le 1\}} \vec{a}_i(\vec{a}_i^T\vec{x} - b_i) - \vec{1}_{\{b_i \vec{a}_i^T \vec{x} \le 0\}} b_i \vec{a}_i$

$\therefore \|\nabla f_{i_k}(\vec{x}) - \nabla f_{i_k}(\vec{y})\| = \|\lambda(\vec{x}-\vec{y}) + \vec{a}_{i_k}(\vec{a}_{i_k}^T\vec{x} - b_{i_k}) - \vec{a}_{i_k}(\vec{a}_{i_k}^T\vec{y} - b_{i_k})\|$

$= \|\lambda(\vec{x}-\vec{y}) + \vec{a}_{i_k}\vec{a}_{i_k}^T(\vec{x}-\vec{y})\|$

$\le \lambda\|\vec{x}-\vec{y}\| + \|\vec{a}_{i_k}\vec{a}_{i_k}^T\| \cdot \|\vec{x}-\vec{y}\|$

$= [\lambda + \|\vec{a}_{i_k}\|^2] \cdot \|\vec{x}-\vec{y}\|$

$\therefore L = \lambda + \|\vec{a}_{i_k}\|^2$ and ∇f_{i_k} is Lipschitz continuous

Program 3(4)

(b)(c)(d) stochastic gradient method

In the stochastic gradient method, I use a random number uniformly, so for SGD, I always get the different outcomes.about SGD

SGD(one experiment):

f(x) = 0.109049377

Error w.r.t 0-1 loss: 0.08029197080291971

SAG:

f(x) = 0.047288051

Error w.r.t 0-1 loss: 0.029197080291970802

SVR:

f(x) = 0.039897199

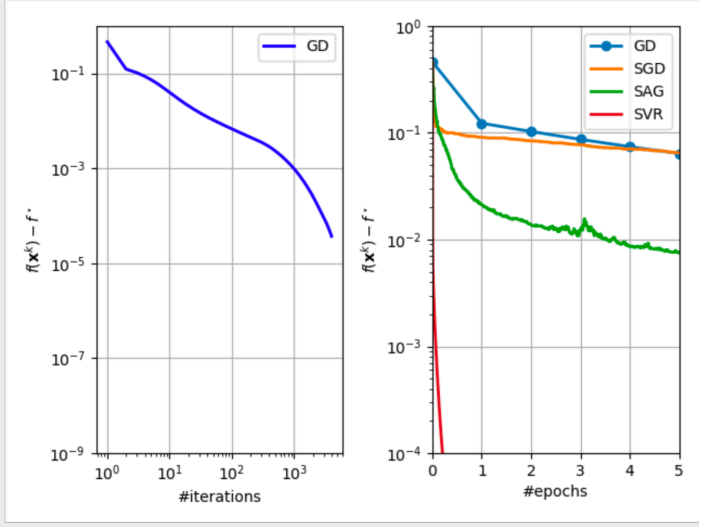
Error w.r.t 0-1 loss: 0.029197080291970802

SGD converges faster than GD but the value of function almost had no change with the change of epochs.

SAG use the average of previous gradients and its convergence rate is bigger than SAG and GD

SVR converges very fast, almost reach as a linear converge rate, so I find the effect of reducing the variance on convergence rate is obvious.

To sum up, SGD choose a single sample randomly as the sample population, so, the error sometimes become larger but when we are dealing with large amounts of data，SGD can help us get better performance.



Final results: