

Assignment - 2 (CSE-S12)

Name: Venkata Subba Narasa Bharath, Meadam

SBU ID: 112672986

Q1)

$$(a) f(x) = \frac{1}{2} \|Ax - b\|_2^2$$

We know, $\|x\|_2^2 = x^T x$

$$\|Ax - b\|_2^2 = (Ax - b)^T (Ax - b)$$

$$= ((Ax)^T - b^T)(Ax - b)$$

$$= (x^T A^T - b^T)(Ax - b)$$

$$= x^T A^T A x - x^T A^T b - b^T A x + b^T b$$

$$= x^T A^T A x - 2x^T A b + b^T b$$

$$\frac{\partial f(x)}{\partial x} = \frac{\partial}{\partial x} \left(\frac{1}{2} (x^T A^T A x - 2x^T A b + b^T b) \right)$$

$$= \frac{1}{2} (2A^T A x - 2A^T b)$$

$$\frac{\partial f(x)}{\partial x} = A^T A x - A^T b$$

$$\nabla^2 f(x) = A^T A$$

Now, to prove Hessian is positive -

Semi-definite (PSD):

$$H z : z^T \nabla^2 f(x) z = z^T (A^T A) z \\ = (Az)^2 \geq 0$$

For it to be μ -strongly convex,

we want $\nabla^2 f(x) > 0$, we have

$\nabla^2 f(x) \geq 0$. So, it is not μ -strongly

convex, i.e. we do not have a

lower bound, but it is indeed L -smooth.

L is the largest eigen-value of
the Hessian. value of mu -0.1

$\therefore f(x) \rightarrow$ NOT M -strongly convex

\rightarrow It is L -smooth

$\rightarrow L =$ largest eigen value of
the Hessian.

\rightarrow Hessian ($\nabla^2 f(x)$) = $A^T A$

b) $F(x) = f(x) + \frac{\rho}{2} \|x\|^2$.

$$F'(x) = f'(x) + \frac{\partial}{\partial x} \left(\frac{\rho}{2} x^T x \right)$$

$$= f'(x) + \frac{\rho}{2} (2x) = f'(x) + \rho x$$

$$\nabla^2 F(x) = \nabla^2 f(x) + \rho I$$

$$\nabla^2 F(x) = A^T A + \rho I$$

To prove $\nabla^2 F(x)$ is μ -strongly convex,

$$\begin{aligned} \nabla^2 F(x) z^T \nabla^2 F(x) z &= z^T (A^T A + \rho I) z \\ &= (z^T A^T A + z^T \rho I) z \\ &= z^T A^T A z + z^T \rho I z \\ &= (Az)^2 + \rho \|z\|^2 \\ &> 0 \quad (\text{as } \rho > 0) \end{aligned}$$

$\therefore \nabla^2 F(x)$ is μ -strongly convex, and μ is the smallest eigen value.

$\therefore F(x) \rightarrow$ It μ -strongly convex
 $\rightarrow \mu = \text{smallest eigen value of } \nabla^2 F(x)$
 \rightarrow It is also L -smooth
 $\rightarrow L = \text{largest eigen value of}$
 the Hessian.
 \rightarrow Hessian ($\nabla^2 F(x)$) = $A^T A + \rho I$

c) exponential convergence:-

$$b=0, \quad p>0$$

$$F(x) = \frac{1}{2} \|Ax\|^2 \rightarrow \frac{p}{2} \|x\|^2$$

$$f(x) = \frac{1}{2} \|Ax\|^2$$

We have already proved, $F(x)$ as
L-smooth and M-strongly convex in
 \mathbb{R}^n

We know that,

$$x^* = \arg \min_{x \in \mathbb{R}^n} F(x)$$

$$x^{t+1} = x^t - \alpha \nabla F(x^t)$$

$$\begin{aligned} \|x^{t+1} - x^*\|_2^2 &= \|x^t - \alpha \nabla F(x^t) - x^*\|_2^2 \\ &= \|x^t - x^* - \alpha \nabla F(x^t)\|_2^2 \\ &= \|(x^t - x^*) - \alpha \nabla F(x^t)\|_2^2 \end{aligned}$$

$$= \|x^+ - x^*\|_2^2 - 2\alpha \langle \nabla F(x^+), x^+ - x^* \rangle \\ + \alpha^2 \|\nabla F(x^+)\|_2^2$$

From, the definition of M -strongly convex,

we have,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M}{2} \|y - x\|^2$$

$$\leq (1 - \alpha M) \|x^+ - x^*\|_2^2 - 2\alpha (F(x^+) - F(x^*)) \\ + \alpha^2 \|\nabla F(x^+)\|_2^2$$

From, the definition of L -smooth

we have,

$$f(x^*) - f(x) \leq \frac{1}{2L} \|\nabla f(x)\|_2^2$$

$$\leq (1 - \alpha M) \|x^+ - x^*\|_2^2 - 2\alpha (F(x^+) - F(x^*)) \\ + 2\alpha^2 L (F(x^+) - F(x^*))$$

$$= (1 - \alpha M) \|x^+ - x^*\|_2^2 - 2\alpha (1 - \alpha L) (F(x^+) - F(x^*))$$

We know that $\alpha < \frac{1}{L} \Rightarrow \alpha L < 1$

$$\Rightarrow -2\alpha(1-\alpha L) < 0.$$

$$\|x^{t+1} - x^*\|_2^2 \leq (1-\alpha M) \|x^t - x^*\|_2^2$$

↓
If we solve this, recurrence relation,
the term $(1-\alpha M)$ gets multiplied throughout

$$F(x^t) - F(x^*) \leq (1-\alpha M)^t$$

$$F(x^t) - F(x^*) \leq c^t$$

$$F(x^t) - F(x^*) = O(c^t)$$

$$c = (1-\alpha M)$$

The idea here is that, if a function is L -smooth and M -strongly convex, the gradient descent is faster and it reaches the minima quickly.

Source: Piazza Post 175

$$\text{D) } A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad P = 0$$

(ii) $u \in \text{null}(A)$, $v \in \text{range}(A^T)$

Definition of null space of A :

$$\text{null}(A) = \{x: Ax = 0\}$$

Definition of Range of A^T :

$$\text{range}(A^T) = \{y: A^T x = y \text{ for some } x\}$$

$$\text{null}(A) \Rightarrow Ax = 0$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x_1 + x_2 = 0 \quad \left. \begin{array}{l} x_1 = -x_2 \\ x_1 + x_2 = 0 \end{array} \right\}$$

$$x = \begin{bmatrix} -k \\ k \end{bmatrix}, \text{ for any } k$$

$$u = \begin{bmatrix} -k \\ k \end{bmatrix} \Rightarrow \text{null space of } A.$$

$$\text{Range}(A^T) \Rightarrow A^T x = y$$

y is the range of A^T

let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\begin{bmatrix} x_1 + x_2 \\ x_1 + x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\Rightarrow y_1 = y_2$$

$$\therefore \text{Range of } A^T = \begin{bmatrix} k \\ k \end{bmatrix} \text{ for any } k.$$

$$x = u + v$$

To find:- $u = \text{Proj}_{\text{null}(A)}(x)$

From, gradient descent, we know,

$$x^{t+1} = x^t - \alpha \nabla F(x^t)$$

Substituting $t=0$

$$x' = x^0 - \alpha \nabla F(x^0)$$

$$\begin{aligned} \text{From (1a)} \quad \nabla F(x) &= A^T Ax - A^T b \\ &= A^T (Ax - b) \end{aligned}$$

$$x' = x^0 - \alpha A^T (Ax - b)$$

This equation is of the form

$$\begin{aligned} u+v \\ u &= x^0 \quad (\text{null space of } A) \\ v &= \alpha A^T (Ax - b) \\ &= A^T (\alpha Ax - \alpha b) \quad (\alpha \rightarrow \text{some scalar}) \\ &= A^T (x) \Rightarrow x = \alpha Ax - \alpha b \\ &\quad (\text{Range space of } A^T) \end{aligned}$$

From, decomposition theorem,

For any $z \in \mathbb{R}^n$, there exists a
UNIQUE u, v where,

$$z = u + v, \quad u \in \text{null}(A), \quad v \in \text{range}(A^\top)$$

$$\therefore z = u + v$$

$$\Rightarrow z' = z^0 - \alpha A^\top (Az - b)$$

(This decomposition is unique)

$$\therefore \text{Projection}_{\text{null}(A)}^{(x^{(0)})} = z^{(0)}$$

$$z^{(1)} = z^{(0)} - \alpha \nabla F(x^{(0)})$$

$$= (x^{(0)} - \alpha A^\top (Ax_0 - b)) - \alpha A^\top (Ax_1 - b)$$

$$= x^{(0)} - \alpha [A^\top A x_0 - A^\top b + A^\top A x_1] \\ - A^\top b$$

$$= \underbrace{x^{(0)}}_u - \alpha A^\top \underbrace{[A^{(0)} + x_1] - 2b}_v$$

$x^{(2)}$ is again in the form of $u+v$ and again this decomposition is unique.

$$\therefore \text{Projection}_{\text{null}(A)}(x^{(2)}) = x^{(0)}$$

Similarly, even after t -iterations,
 THE PROJECTION $\text{null}(A)$ $(x^{(t)})$ DOES NOT
 CHANGE.

ii) The same argument can be used here as well, only the initial vector $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \rightarrow$ which is a part of nullspace (A)

$$x = u+v$$

$$x^{(t+1)} = x^{(t)} - \alpha \nabla F(x^{(t)})$$

The $\text{Proj}_{\text{null}(A)}(x^{(k)})$ even after t -iterations does not change and it remains same. $x^0 = \begin{bmatrix} \cdot \\ \vdots \\ \cdot \end{bmatrix}$

iii) If $\rho = 1$

$$\nabla F(x) = A^T A x_L - A^T b + \rho x$$

$$x^{(1)} = x^{(0)} - \alpha \nabla F(x^{(0)})$$

$$= x^{(0)} - \alpha (A^T A x_L - A^T b + \rho x)$$

$$= x^{(0)} - \alpha A^T (A x_L - b) + \alpha \rho x.$$

$$\text{Again } x_L = u + v$$

$$u \in \text{null}(A), \quad v \in \text{Range}(A^T)$$

THIS DECOMPOSITION IS UNIQUE.

If we prove that $\alpha \rho x$ does not belong to $\text{null}(A)$, we are done.

$$\text{null}(A) = \begin{bmatrix} k \\ -k \end{bmatrix} \text{ for any } k.$$

AS, $x^{(0)} + \alpha P x$ does not belong to $\text{null}(A)$, and also $\text{null}(A)$ and $\text{range}(A^T)$ are orthogonal, we can say decomposition theorem is still true.

∴ even if $\beta=1$, the projections does not change for both

(i) (d) \rightarrow i and (i) (d) \rightarrow ii

there is a coefficient $(1-\alpha)^{t-1}$

$$(Q2) f(\theta) = -\frac{1}{m} \sum_{i=1}^m \log (\sigma(y_i x_i^\top \theta))$$

$$\sigma(s) = \frac{1}{1 + e^{-s}}$$

A function $f(x)$ is convex if:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$$

$$0 \leq \alpha \leq 1$$

Second order definition of convexity

$\forall z: z^\top \nabla^2 f(x) z \geq 0$ where $\nabla^2 f(x)$ is the Hessian

$$f(\theta) = -\frac{1}{m} \sum_{i=1}^m \log (\sigma(y_i x_i^\top \theta))$$

Let us consider the function,

$$h = \log (\sigma(y_i x_i^\top \theta))$$

$$\begin{aligned}\frac{\partial h}{\partial \theta} &= \frac{\partial}{\partial \theta} (\log(\sigma(y_i x_i^\top \theta))) \\ &= \frac{1}{\sigma(y_i x_i^\top \theta)} \frac{\partial}{\partial \theta} (\sigma(y_i x_i^\top \theta))\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial x} (\sigma(x)) &= \sigma(x) (1 - \sigma(x)) \\ \frac{\partial}{\partial \theta} \sigma(y_i x_i^\top \theta) &= \sigma(y_i x_i^\top \theta) (1 - \sigma(y_i x_i^\top \theta)) \\ &\quad * \frac{\partial}{\partial \theta} (y_i x_i^\top \theta) \\ &= y_i x_i^\top \sigma(y_i x_i^\top \theta) (1 - \sigma(y_i x_i^\top \theta)) \\ &\quad y_i x_i^\top \cancel{\sigma(y_i x_i^\top \theta)} (1 - \sigma(y_i x_i^\top \theta))\end{aligned}$$

$$\begin{aligned}\frac{\partial h}{\partial \theta} &= \frac{1}{\sigma(y_i x_i^\top \theta)} y_i x_i^\top (1 - \sigma(y_i x_i^\top \theta)) \\ \frac{\partial f}{\partial \theta} &= -\frac{1}{m} \sum_{i=1}^m (y_i x_i^\top) * (1 - \sigma(y_i x_i^\top \theta))\end{aligned}$$

$$\text{let, } z = y_i x_i, \quad d_i = \sigma(z_i^\top \theta)$$

$$\frac{\partial f}{\partial \theta} = \frac{1}{m} z^\top (\lambda - 1)$$

$$\nabla^2 f = \frac{1}{m} z^\top \text{diag}(\lambda) \text{diag}(1-\lambda) z$$

(source:- HW1)

$$z^\top \nabla^2 f z = z^\top \left(\frac{1}{m} z^\top \text{diag}(\lambda) \text{diag}(1-\lambda) z \right) z$$

$$= \frac{1}{m} (z^\top z)^2 \frac{\text{diag}(\lambda)}{\text{diag}(1-\lambda)}$$

$$\geq 0$$

$\therefore f(\theta)$ is convex

$$(b) R \leq uv^\top$$

Given: $U \in \mathbb{R}^{m \times g}$, $V \in \mathbb{R}^{n \times g}$

$$U^\top \in \mathbb{R}^{g \times n}$$

$$(g=1)$$

Show that, $f(u, v) = \frac{1}{2} \|R - uv^T\|_F^2$ is non-convex.

$$\text{using } x^2 = x^T x$$

$$f(u, v) = \frac{1}{2} \|R - uv^T\|^2$$

$$= \frac{1}{2} (R - uv^T)^T (R - uv^T)$$

$$= \frac{1}{2} [(R^T - v u^T) (R - uv^T)]$$

$$= \frac{1}{2} [R^T R - R^T uv^T - vu^T R + vu^T uv^T]$$

$$= \frac{1}{2} [R^T R - 2R^T uv + u^2 v^2]$$

$$= \frac{1}{2} [R^2 - 2Ruv + v^2 u^2]$$

$$\nabla f(u, v) = \begin{bmatrix} 2uv^2 - 2Ru \\ 2u^2v - 2Rv \end{bmatrix}$$

$$\nabla^2 f(u, v) = \begin{bmatrix} 2v^2 & 4uv - 2R \\ 4uv - 2R & 2u^2 \end{bmatrix}$$

Assume $m=1, n=1, R=1$

$$u = (1x1), v = (1x1)$$

$v = 2, n = 1 \rightarrow \text{assumption}$
 $R = uv - 2$

$$\nabla^2 f(2, 1) = \begin{bmatrix} 2 & u(2)(1) - 2 \\ 4(1)(2) - 2 & 2(2)^2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 6 \\ 6 & 8 \end{bmatrix}$$

$$\lambda_{\min} = 5 - 3\sqrt{5} \Rightarrow \text{NOT PSD}$$

$$= -1.7 < 0$$

$\therefore f$ is non-convex.

$$Q3) B_R = \{x : \|x\| \leq R\}$$

(a)

let $x, y \in B_R$

$$\Rightarrow \|x\| \leq R, \|y\| \leq R$$

for $\lambda \in (0, 1)$

$$z = \lambda x + (1-\lambda)y$$

$$\|z\| = \|\lambda x + (1-\lambda)y\|$$

$$\leq \|\lambda x\| + \|(1-\lambda)y\| \quad (\|a+b\| \leq \|a\| + \|b\|)$$

$$\leq \|x\| \|\lambda\| + \|(1-\lambda)\| \|y\| \quad (\|ka\| = \|k\| \|a\|)$$

$$\leq \lambda \|x\| + (1-\lambda) \|y\| \quad (\lambda \in (0, 1) \Rightarrow (1-\lambda) \in (0, 1))$$

$$\leq \lambda R + (1-\lambda) R \quad (\|x\| \leq R, \|y\| \leq R)$$

$$\leq R$$

$$\therefore z \in B_R$$

$\therefore B_R$ is convex

$$\bar{B}_g = \{x : \|x\| > g\}$$

↑

compliment

$$\text{let } -x, x \in \bar{B}_g$$

$$\Rightarrow \| -x \| > g, \| x \| > g$$

$$z = \alpha(-x) + (1-\alpha)x$$

$$\text{let } \alpha = \frac{1}{2}$$

$$z = \frac{1}{2}(-x) + \left(1 - \frac{1}{2}\right)x = 0$$

$$\|z\| = 0 \quad (\|0\| = 0)$$

$$\|z\| < g \quad (\text{assume initially } g=1)$$

This is a contradiction.

$\therefore \bar{B}_g$ is non-convex.

$$(b) S_g = \{x : f(x) \leq g\}$$

If $f(x)$ is convex:-

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

\uparrow
Convex function property, $\lambda \in (0,1)$

let $x, y \in S_g \Rightarrow f(x) \leq g, f(y) \leq g$

$$z = \lambda x + (1-\lambda)y$$

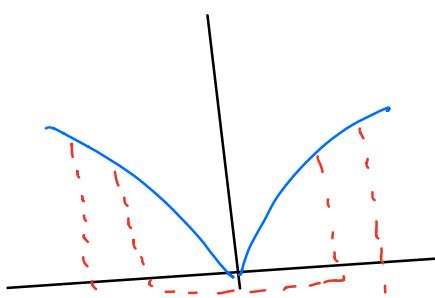
$$\begin{aligned} f(z) &= f(\lambda x + (1-\lambda)y) \\ &\leq \lambda f(x) + (1-\lambda) f(y) \\ &\leq \lambda g + (1-\lambda)g \\ &\leq g \end{aligned}$$

$\therefore S_g$ is convex

If S_g is convex:-

$$S_g = \{x : f(x) \leq g\}$$

$$f(x) = \sqrt{|x|}$$



0.5

we have sub-level sets which are convex, but the function is non-convex.

∴ The opposite (If the function has only convex sets, the function need not to be convex).

(Q4)

a) minimize $\frac{1}{2} \|\theta\|^2 + \rho \phi(s)$

$\theta \in \mathbb{R}^n$, $s \in \mathbb{R}^m$

subject to $y_i x_i^\top \theta = 1 + s_i$
 $i = 1, 2, \dots, m$

$$\phi(s) = \frac{1}{2} \sum_{i=1}^m (\max\{-s_i, 0\})^2$$

Cost-Function :-

$$f(\theta, s) = \frac{1}{2} \|\theta\|^2 + \rho * \frac{1}{2} \sum_{i=1}^m (\max\{-s_i, 0\})^2$$

we have 2 cases :-

i) if $1 - y_i x_i^\top \theta \geq 0$

ii) if $1 - y_i x_i^\top \theta < 0$

if $1 - y_i x_i^\top \theta \geq 0$

$$f(\theta, s) = \frac{1}{2} \|\theta\|^2 + \rho * \frac{1}{2} \sum_{i=1}^m (-s_i)^2$$

$$= \frac{1}{2} \|\theta\|^2 + \rho * \frac{1}{2} \sum_{i=1}^m (1 - y_i x_i^\top \theta)^2$$

if $1 - y_i x_i^\top \theta < 0$

$$f(\theta) = \frac{1}{2} \|\theta\|^2$$

$$\nabla f(\theta) = \theta + \rho \sum_{i=1}^m (1 - y_i x_i^\top \theta) (-y_i x_i^\top)$$

(if $(1 - y_i x_i^\top \theta) \geq 0$)

$$\nabla f(\theta) = \theta \quad (\text{if } (1 - y_i x_i^\top \theta) < 0)$$

$$\nabla^2 f = I + \rho \sum_{i=1}^m (y_i x_i^\top)^2$$

(if $(1 - y_i x_i^\top \theta) \geq 0$)

$$\nabla^2 f = I \quad (\text{if } (1 - y_i x_i^\top \theta) < 0)$$

We can show that Hessian is

always ≥ 1 (more generally - it is PSD)

now, L -value for this is the maximum eigen-value of the hessian -

$$\therefore L = \max(1, S)$$

1.0

(b) Given:-

To minimize $\|\theta\|_2$ subject to $A\theta = b$

We want $\hat{\theta}$,

$$\text{minimize}_{\theta, S} \|\theta - \hat{\theta}\|_2^2 + \|S - \hat{S}\|_2^2$$

subject to $y_i x_i^\top \theta = 1 + s_i$, $i=1, \dots, m$.

Sol :- we want to find the ideal θ, S

so, in the equation $A\theta = b$

θ is a $[2 \times 1]$ vector, and b is a scalar.

$$\therefore A = [1 \times 2]$$

Now,

$$A = \begin{bmatrix} y_i x_i^\top & -1 \end{bmatrix}, \theta = \begin{bmatrix} \theta - \hat{\theta} \\ s - \hat{s} \end{bmatrix}$$

$$A\theta = \begin{bmatrix} y_i x_i^\top & -1 \end{bmatrix}_{1 \times 2} \begin{bmatrix} \theta - \hat{\theta} \\ s - \hat{s} \end{bmatrix}_{2 \times 1}$$

$$= y_i x_i^\top (\theta - \hat{\theta}) - 1 (s - \hat{s})$$

$$= y_i x_i^\top \theta - y_i x_i^\top \hat{\theta} - s + \hat{s}$$

$$\text{Substituting, } -s = 1 - y_i x_i^\top \theta$$

$$= \cancel{y_i x_i^\top \theta} - y_i x_i^\top \hat{\theta} + 1 - y_i x_i^\top \theta + \hat{s}$$

$$b = 1 + \hat{s} - y_i x_i^\top \hat{\theta}$$

The only difference between this equation and $A\theta = b$ previously is

that now we have 2 parameters
 (θ, δ) to optimize against 1-parameter
 previously.

we have chosen over A, θ
 accordingly.

The motivation for $\theta = \begin{bmatrix} \theta - \hat{\theta} \\ s - \hat{s} \end{bmatrix}$
 we differentiate + function w.r.t
 θ, s to form our new θ , i.e.

$$\begin{bmatrix} \theta - \hat{\theta} \\ s - \hat{s} \end{bmatrix}.$$

$$\therefore A = \begin{bmatrix} y_i x_i^\top & -I \end{bmatrix}, \theta = \begin{bmatrix} \theta - \hat{\theta} \\ s - \hat{s} \end{bmatrix}$$

$$b = 1 + \hat{s} - y_i x_i^\top \hat{\theta}$$

```
In [1]: import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from IPython.display import Markdown
from scipy import stats
```

```
In [2]: data = sio.loadmat('arrhythmia/arrhythmia.mat')
```

```
In [3]: X = data['X']
y = data['Y'].T
y[y>=2] = -1
idx_test = data['idx_test']
idx_train = data['idx_train']

Xtrain = X[idx_train[0,:,:]]
ytrain = y[idx_train[0,:,:]]
Xtest = X[idx_test[0,:,:]]
ytest = y[idx_test[0,:,:]]
```

```
In [4]: print("Shape of XTrain:" +str(np.shape(Xtrain)))
print("Shape of YTrain:" +str(np.shape(ytrain)))
print("Shape of XTest:" +str(np.shape(Xtest)))
print("Shape of YTest:" +str(np.shape(ytest)))
```

```
Shape of XTrain:(272, 279)
Shape of YTrain:(272, 1)
Shape of XTest:(180, 279)
Shape of YTest:(180, 1)
```

```
In [5]: np.unique(y, return_counts=True)
```

```
Out[5]: (array([-1.,  1.]), array([207, 245]))
```

Class imbalance

There is not much of a class imbalance as we have:

- 1) 207 affected cases
- 2) 245 Normal cases

The ratio is 45% - 55% (This is almost a perfect balance for both the classes)

```
In [6]: # Computing Number of Normal and positive cases in Training and Test sets
```

```
trainingClassBalance = np.unique(ytrain, return_counts=True)
testClassBalance = np.unique(ytest, return_counts=True)

totalTrainingSamples = len(ytrain)
totalTestSamples = len(ytest)

# For Training Data
trainNormalSamples = round(trainingClassBalance[1][0] * 100 / totalTrainingSamples,2)
trainPosotiveSamples = round(trainingClassBalance[1][1] * 100 / totalTrainingSamples,2)

# For Test Data

testNormalSamples = round(testClassBalance[1][0] * 100 / totalTestSamples,2)
testPosotiveSamples = round(testClassBalance[1][1] * 100 / totalTestSamples,2)
```

```
In [7]: Markdown("# Title")
```

```
Markdown("""
```

```
## Train Class Balance
```

```
Normal Cases: {normal}% Positive Cases {positive}%"
```

```
""".format(normal=trainNormalSamples, positive=trainPosotiveSamples))
```

```
Out[7]:
```

Train Class Balance

```
Normal Cases: 46.32% Positive Cases 53.68%
```

```
In [8]: Markdown("# Title")
```

```
Markdown("""
```

```
## Test Class Balance
```

```
Normal Cases: {normal}% Positive Cases {positive}%"
```

```
""".format(normal=testNormalSamples, positive=testPosotiveSamples))
```

```
Out[8]:
```

Test Class Balance

```
Normal Cases: 45.0% Positive Cases 55.0&
```

```
In [9]: def normalizeData(X):
```

```
    # finding mean
```

```
    mean=np.mean(X, axis=0)
```

```
    # finding std
```

```
    #     std = np.std(X)
```

```
    std = np.std(X, axis=0)
```

```
    m = X.shape[0]
```

```
    X_norm = (X - mean) / std
```

```
    return X_norm
```

```
In [10]: x = data['X']
y = data['y'].T
y[y>=2] = -1
idx_test = data['idx_test']
idx_train = data['idx_train']

Xtrain = X[idx_train[0,:,:],:]
ytrain = y[idx_train[0,:,:],:]
Xtest = X[idx_test[0,:,:],:]
ytest = y[idx_test[0,:,:],:]
```

```
In [11]: Xtrain_Normalized = normalizeData(Xtrain)
Xtest_Normalized = normalizeData(Xtest)
# X = np.append(np.ones((m,1)),X_normalized,axis=1)
```

```
/Users/mvsnbharath/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: RuntimeWarning: divide by zero encountered in true_divide

/Users/mvsnbharath/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: RuntimeWarning: invalid value encountered in true_divide
```

```
In [12]: print(Xtrain_Normalized[0][4])
print(stats.zscore(Xtrain)[0][4])
```

```
-0.8813592157754074
-0.8813592157754074
```

```
/Users/mvsnbharath/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:23
15: RuntimeWarning: divide by zero encountered in true_divide
    return (a - mns) / sstd
/Users/mvsnbharath/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:23
15: RuntimeWarning: invalid value encountered in true_divide
    return (a - mns) / sstd
```

```
In [13]: ## Verification for Normalization with in-built function
if stats.zscore(Xtrain)[0][4] == Xtrain_Normalized[0][4]:
    print("Normalization Verified with the inbuilt function in scipy(z-score)")
else:
    print("Normalization seems to have a bug")
```

```
Normalization Verified with the inbuilt function in scipy(z-score)
```

```
In [14]: def computeLoss(Theta, X, Y,Rho):
    # calculate hinge loss
    m = X.shape[0]
    # equivalent to (1- YX.T.Theta)
    distances = 1 - Y * (np.dot(X, Theta))
    # equivalent to min(-s, 0)
    distances[distances < 0] = 0
    distances = np.square(distances)

    # hinge_loss = Rho * (np.sum(distances) / m)
    hinge_loss = Rho * (np.sum(distances))

    # Total loss
    cost = 1 / 2 * (np.sum(np.square(Theta)) + hinge_loss)
    return cost
```

```
In [15]: # Helper Functions
```

```
# function to calculate margin
def margin(Z):
    return 1/np.linalg.norm(Z)

def computeGradientForS(Theta, X, Y, Rho):

    s = 1 - Y * (np.dot(X, Theta)) # m
    gradient = np.zeros(len(s))
    for i in range(len(s)):
        gradient[i] = Rho * np.max (-1*s[i], 0)
    return gradient

def misClassificationRate(X, Y, theta):
    # we try to calculate how many predictions we missed
    misCalculatedValues = Y * (np.dot(X, theta))
    misCalculatedValues[misCalculatedValues >= 0] = 0
    misCalculatedValues[misCalculatedValues < 0] = 1
    return np.sum(misCalculatedValues)/len(misCalculatedValues)
```

```
In [16]: # Helper method to calculate projections
```

```
def projection(s_hat, theta_hat, X, y):
    m = np.shape(X)[0]
    Z = np.zeros(np.shape(X))

    I = np.identity(m).astype(float)

    for i in range(m):
        Z[i,:] = y[i]*X[i,:]

    x = np.ones(m).reshape(272,1)
    b = -Z.dot(theta_hat)+x+s_hat
    A = np.hstack((Z,-I)).astype(float)
    t = np.linalg.pinv(A).dot(b)
    theta = t[:X.shape[1]] + theta_hat
    s = t[X.shape[1]:] + s_hat

    return s, theta
```

```
In [17]: def gradientDescent(X,Y, theta,s, Rho,num_iters):
    m=len(y)
    lossFunctionHistory = []
    misClassificationRateHistory = []
    marginHistory = []
    learning_rate = max(1, Rho)
    for i in range(num_iters):

        # computing gradient
        gradient_theta = theta
        gradient_s      = computeGradientForS(theta, X, Y ,Rho)
        #gradient = calculate_cost_gradient(weights, x, Y[ind])

        # Updating theta and s values

        theta_hat  = theta - (learning_rate * gradient_theta)
        s_hat      = s      - (learning_rate * gradient_s.reshape(272,1))

        s, theta = projection(s_hat, theta_hat, X, Y)

        lossFunctionHistory.append(computeLoss(theta,X,Y, Rho))
        misClassificationRateHistory.append(misClassificationRate(X,Y, theta))
        marginHistory.append(margin(s_hat))

    return theta , lossFunctionHistory, marginHistory,misClassificationRateHistory
```

```
In [18]: def plotValues(misClassificationRateHistory, marginHistory):
    plt.figure(figsize=(10,5))

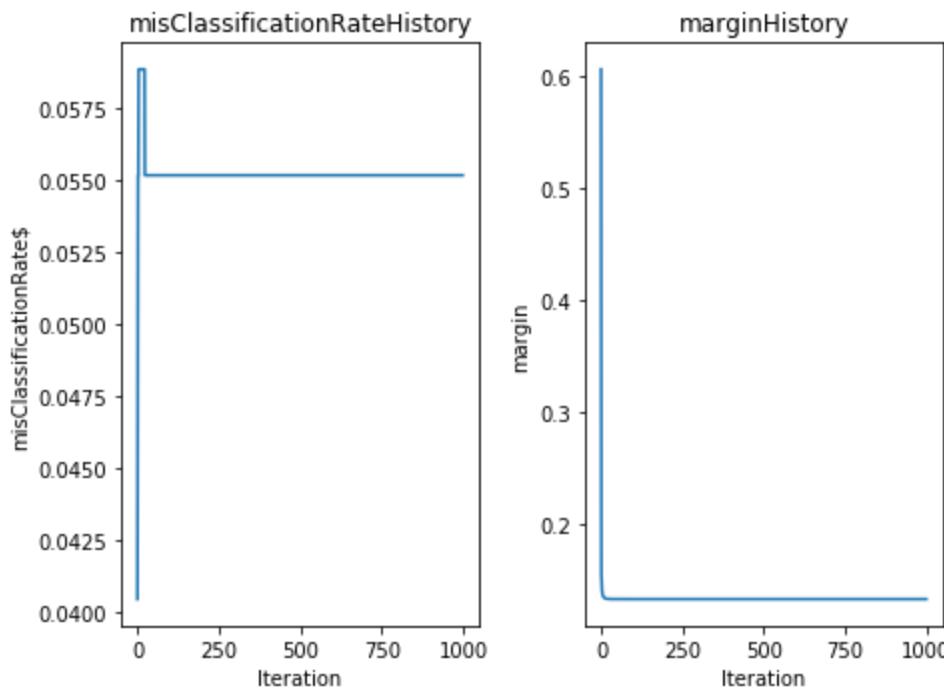
    plt.subplot(1,3,1)
    plt.plot(misClassificationRateHistory)
    plt.xlabel("Iteration")
    plt.ylabel("misClassificationRate$")
    plt.title("misClassificationRateHistory")

    plt.subplot(1,3,2)
    plt.plot(marginHistory)
    plt.xlabel("Iteration")
    plt.ylabel("margin")
    plt.title("marginHistory")

    # plt.subplot(1,3,2)
    # plt.plot(misClassificationRateHistory)
    # plt.xlabel("Iteration")
    # plt.ylabel("$J(\Theta)$")
    # plt.title("misClassificationRateHistory")
    plt.tight_layout()
```

```
In [19]: #print(np.shape(initial_theta))
initial_theta = np.zeros((np.shape(Xtrain)[1],1))
initial_s = np.zeros((np.shape(Xtrain)[0],1))
Rho = 0.1
cost = computeLoss(initial_theta,Xtrain,ytrain,Rho)
#print("Initial theta cost is",cost)

theta , lossFunctionHistory, marginHistory,misClassificationRateHistory = gradientDescent(Xtrain, ytrain, initial_theta, initial_s,Rho, 1000)
plotValues(misClassificationRateHistory, marginHistory)
```



Class imbalance

There is not much of a class imbalance as we have:

- 1) 207 affected cases
- 2) 245 Normal cases

The ratio is 45% - 55% (This is almost a perfect balance for both the classes)

Final Comments

1) I think there are some issues in my implementation of projections, as seen in the graphs.

2) I'll briefly explain the procedure followed:

- 1) Normalize data (Subtract with mean and divide with standard deviation)
- 2) Initialize theta and s with zeros.
- 3) Compute the loss
- 4) Compute the gradient
- 5) Update theta and s with gradient followed by their projections.
- 6) Repeat the process for 1000 iterations with different values of Rho.

3) I may not have the final graph correct but most of my functions I guess are correct and I tried to write a modular code.

4) I request the TAs to consider if it's possible to give partial credit.

I cannot give you partial credit if you do not have any valid results.
I keep this grading policy for all the submissions.
since you do not have any results, your answers for (ii) and (iii)
are unbased to me.

0.5 for Q4.c

4c (ii)

This data is separable,

this may not be a great approach for large values of Rho as we try to minimize the cost function.

If Rho is large, we try to make that term zero and ultimately under-fit the data.

Here, Rho is the parameter of our ridge regression.

We use, ridge regression and a very large value of Rho may not help us to minimize the cost function.

4c (iii)

My expectation from theoretical analysis would be that misclassification rate would be optimal around Rho = 100, that value would give us a good misclassification rate and penalty would also be reduced at that rate.

In []: