Assignment - 2 (CSE - 512)

Name: Venkata Subba Narasa Bharath, Meadam

SBU ID: 112672986

Q1)

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

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

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

$$= \left((Ax)^T - b^T\right)(Ax - b)$$

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

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

$$= x^T A^T Ax - 2x^T Ab + bb^T$$

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

$$= \frac{1}{2}\left(2A^TAx - 2A^Tb\right)$$

$$\frac{\partial f(x)}{\partial x} = A^TAx - A^Tb$$

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

Now, to Prove Hessian is positive-

Semi-definite (PSD):

$$\forall z: \; z^T \nabla^2 f(x) z = z^T (A^TA) z$$

$$= (Az)^2 \geq 0$$

For it to be M-strongly convex,

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

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

$\therefore f(x) \rightarrow$ NOT μ-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) $F(x) = f(x) + \frac{\rho}{2} \|x\|^2$.

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

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

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

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

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

$\forall z \quad z^T \nabla^2 F(x) z = z^T (A^T A + \rho) z$

$= (z^T A^T A + z^T \rho) z$

$= z^T A^T A z + z^T \rho z$

$= (Az)^2 + \rho (z)^2$

$> 0 \quad (\text{as } \rho > 0)$

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

$\therefore F(x) \rightarrow$ It μ-strongly convex

$\rightarrow \mu =$ smallest eigen value of $\nabla^2 F(x)$

$\rightarrow$ It is also L-smooth

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

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

c) exponential convergence:-

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

$$F(x) = \frac{1}{2}\|Ax\|^2 + \frac{\rho}{2}\|x\|^2$$

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

we have already proved, $F(x)$ as L-smooth and M- strongly convex in $1(b)$

we know that,

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

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

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

$$= \|x^t - x^* - \alpha \nabla F(x^t)\|_2^2$$

$$= \|(x^t - x^*) - \alpha \nabla F(x^t)\|_2^2$$

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

From the definition of M-strongly convex, we have,

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

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

From the definition of L-smooth we have,

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

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

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

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

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

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

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

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

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

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

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

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

d) $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ $\quad p = 0$

(i)

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

Definition of nullspace of A:

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

Definition of Range of A^T:

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

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

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

$x_1 + x_2 = 0$ $\quad \Big\} \; x_1 = -x_2$

$x_1 + x_2 = 0$

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

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

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

y is the range of A^T

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

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

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

$\Rightarrow y_1 = y_2$

$\therefore$ Range of $A^T = \begin{bmatrix} k \\ k \end{bmatrix}$ 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 $t = 0$

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

From (1a) $\nabla F(x) = A^T Ax - A^T b$

$= A^T (Ax - b)$

$$x^1 = x^0 - \alpha A^T (Ax - b)$$

This equation is of the form

$u + v$

$u = x^0$ (null space of A)

$v = \alpha A^T (Ax - b)$

$= A^T (\alpha Ax - \alpha b)$ ($\alpha \rightarrow$ some scalar)

$= A^T (x) \Rightarrow x = \alpha Ax - \alpha b$

(Range space of A^T)

From, decomposition theorem,

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

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

$\therefore x = u + v$

$\Rightarrow x^1 = x^0 - \alpha A^T (Ax - b)$

(This decomposition is unique)

$\therefore$ Projection$_{\text{null}(A)}(x^{(1)}) = x^{(0)}$

$x^{(2)} = x^{(1)} - \alpha \nabla F(x^{(1)})$

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

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

$= \underbrace{x^{(0)}}_{u} - \alpha A^T \underbrace{\left[A(x_0 + x_1) - 2b \right]}_{v}$

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

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

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

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

$x = u+v$

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

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

iii) If $\rho = 1$

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

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

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

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

Again $x = u + v$

$u \to \text{null}(A)$, $v = \text{Range}(A^T)$

THIS DECOMPOSITION IS UNIQUE.

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

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

As, $x^{(0)} + \alpha \delta x$ does not belong to null (A), and also null (A) and 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) → i and (i) (d) → (ii)

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

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

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

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

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

let us consider the function,

$$h = \log\left(\sigma\left(y_i x_i^T \theta\right)\right)$$

$$\frac{\partial h}{\partial \theta} = \frac{\partial}{\partial \theta} \left(\log \left(\sigma(y_i x_i^T \theta) \right) \right)$$

$$= \frac{1}{\sigma(y_i x_i^T \theta)} \frac{\partial}{\partial \theta} \left(\sigma(y_i x_i^T \theta) \right)$$

$$\frac{\partial}{\partial x} (\sigma(x)) = \sigma(x) (1 - \sigma(x))$$

$$\frac{\partial}{\partial \theta} \sigma(y_i x_i^T \theta) = \sigma(y_i x_i^T \theta) (1 - \sigma(y_i x_i^T \theta)) * \frac{\partial}{\partial \theta} (y_i x_i^T \theta)$$

$$= y_i x_i^T \sigma(y_i x_i^T \theta) (1 - \sigma(y_i x_i^T \theta))$$

$$\frac{\partial h}{\partial \theta} = \frac{1}{\cancel{\sigma(y_i x_i^T \theta)}} \; y_i x_i^T \; \cancel{\sigma(y_i x_i^T \theta)} (1 - \sigma(y_i x_i^T \theta))$$

$$\frac{\partial f}{\partial \theta} = \frac{-1}{m} \sum_{i=1}^{m} (y_i x_i^T) * (1 - \sigma(y_i x_i^T \theta))$$

let, $z = y_i x_i$, $d_i = \sigma(z_i^T \theta)$

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

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

(source:- HW1)

$$\forall q \quad q^T \nabla^2 f q = q^T \left(\frac{1}{m} z^T \text{diag}(d) \text{diag}(1-d)\right) z \cdot q$$

$$= \frac{1}{m} (q \cdot z)^2 \text{diag}(d) \text{diag}(1-d)$$

$$\geq 0$$

$\therefore f(\theta)$ is convex

(b) $R \leq UV^T$

Given:- $U \in R^{m \times q}$, $V \in R^{n \times q}$

$V^T \in R^{q \times n}$

$(q = 1)$

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

using, $x^2 = x^T x$

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

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

$$= \frac{1}{2}\left[(R^T - vu^T)(R - uv^T)\right]$$

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

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

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

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

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

Assume $m=1,\ n=1,\ q=1$

$u = (1\times1),\ v = (1\times1)$

$u=2,\ v=1$ → assumption

$R = u \times v = 2$

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

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

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

$= -1.7 < 0$

∴ f is non-convex.

Q3) $B_r = \{x : \|x\| \le r\}$

(a)

let $x, y \in B_r$

$\Rightarrow \|x\| \le r \quad , \|y\| \le r$

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

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

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

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

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

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

$\le \lambda r + (1-\lambda) r$ $\quad (\|x\| \le r, \ \|y\| \le r)$

$\le r$

$\therefore z \in B_r$

$\therefore B_r$ is convex

$\bar{B}_r$ $= \{x : \|x\| > r\}$

↑ compliment

let $-x, x \in B_r$

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

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

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

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

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

$\|z\| < r$ (assume initially $r = 1$)

This is a contradiction.

$\therefore \bar{B}_r$ is non-CONVEX.

(b) $S_\alpha = \{x : f(x) \le \alpha\}$

If $f(x)$ is convex:-

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

↑ convex function property, $\lambda \in (0,1)$

let $x, y \in S_\alpha \Rightarrow f(x) \le \alpha, \; f(y) \le \alpha$

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

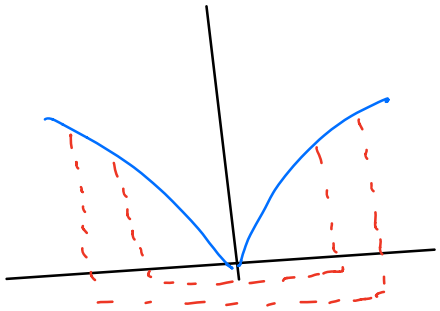
$$\begin{aligned} f(z) &= f(\lambda x + (1-\lambda) y) \\ &\le \lambda f(x) + (1-\lambda) f(y) \\ &\le \lambda \alpha + (1-\lambda) \alpha \\ &\le \alpha \end{aligned}$$

$\therefore S_\alpha$ is convex

If S_α is convex:-

$S_\alpha = \{x : f(x \le \alpha\}$

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



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

$\theta \in R^n,\ s \in R^m$

subject to $y_i x_i^T \theta = 1 + s_i$

$i = 1, 2, \ldots, m$

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

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 $1 - y_i x_i^T \theta \geq 0$

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

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

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

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

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

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

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

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

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

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

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

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

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.

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

(b) Given:-

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

we want to,

minimize $\|\theta - \vec{\theta}\|_2^2 + \|s - \vec{s}\|_2^2$
θ, s

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

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,

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

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

$$= y_i x_i^T (\theta - \hat{\theta}) - I(s - \hat{s})$$

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

Substituting, $-s = 1 - y_i x_i^T \theta$

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

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

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

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

we have chosen our A, θ accordingly.

The motivation for $\theta = \begin{bmatrix} \theta - \hat{\theta} \\ s - \hat{s} \end{bmatrix}$

we differentiate t function w.r.t θ, s to form our new θ, i.e.

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

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

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

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 / totalTrainingSamp
les,2)
trainPosotiveSamples = round(trainingClassBalance[1][1] * 100 / totalTrainingSamp
les,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 Normalioxation 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
```

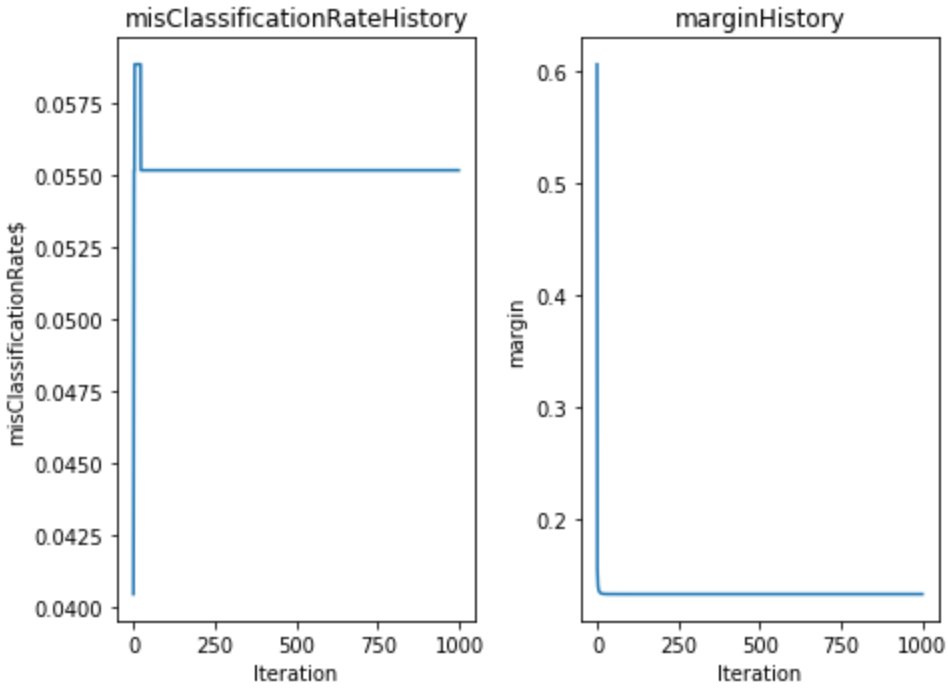
In [15]:
```
# Helper Functions

# 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,misClassificationRateHistor
y
```

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 = gradien
tDescent(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 dicide 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.

4c (ii)

This data is seperable,

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 [ ]: