# ECE 50024

## Homework 2

Parth Sagar Hasabnis phasabni@purdue.edu

## Exercise 1:

```
male_train = pd.read_csv('male_train_data.csv')
male_train['male_bmi'] = male_train['male_bmi']/10
male_train['male_stature_mm'] = male_train['male_stature_mm']/1000
male_train['y'] = 1
male_train.head(10)
```

| | index | male_bmi | male_stature_mm | y |
|---|---|---|---|---|
| 0 | 0 | 3.00 | 1.679 | 1 |
| 1 | 1 | 2.56 | 1.586 | 1 |
| 2 | 2 | 2.42 | 1.773 | 1 |
| 3 | 3 | 2.74 | 1.816 | 1 |
| 4 | 4 | 2.59 | 1.809 | 1 |
| 5 | 5 | 2.53 | 1.662 | 1 |
| 6 | 6 | 2.27 | 1.829 | 1 |
| 7 | 7 | 2.54 | 1.686 | 1 |
| 8 | 8 | 3.41 | 1.761 | 1 |
| 9 | 9 | 3.34 | 1.797 | 1 |

*Male data*

| | index | female_bmi | female_stature_mm | y |
|---|---|---|---|---|
| 0 | 0 | 2.82 | 1.563 | -1 |
| 1 | 1 | 2.22 | 1.716 | -1 |
| 2 | 2 | 2.71 | 1.484 | -1 |
| 3 | 3 | 2.81 | 1.651 | -1 |
| 4 | 4 | 2.55 | 1.548 | -1 |
| 5 | 5 | 2.30 | 1.665 | -1 |
| 6 | 6 | 3.56 | 1.564 | -1 |
| 7 | 7 | 3.11 | 1.676 | -1 |
| 8 | 8 | 2.46 | 1.690 | -1 |
| 9 | 9 | 4.30 | 1.704 | -1 |

*Female Data*

```
female_train = pd.read_csv('female_train_data.csv')
female_train['female_bmi'] = female_train['female_bmi']/10
female_train['female_stature_mm'] = female_train['female_stature_mm']/1000
female_train['y'] = -1
female_train.head(10)
```

## Exercise 2:

### a)

Homework 2

Exercise 2

a) For the problem:

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{n=1}^{N} \left(y_n - g_\theta(x_n)\right)^2$$

where $g_\theta = \theta^T x$

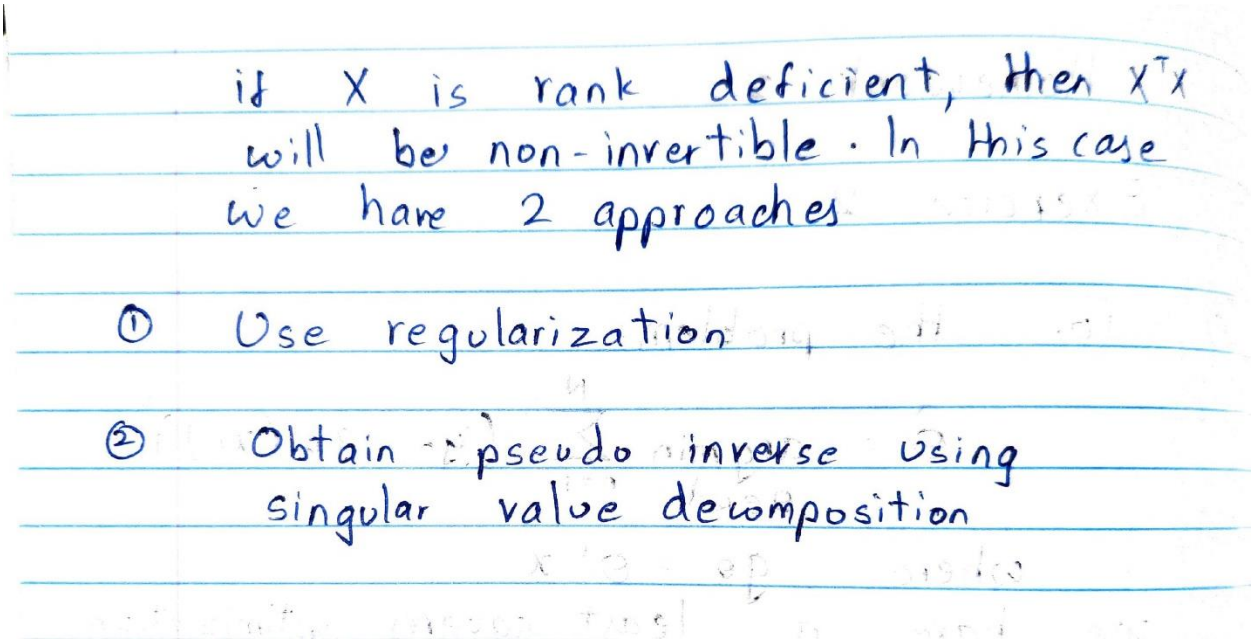
we have a least squares optimization problem which is given by

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \|y - X\theta\|^2$$

The optimal solution for this problem is given by:

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

The solution is the unique global minimum if $X$ is an overdetermined system with full column rank




## b) Analytical Solution in Python

```
theta = np.matmul(X_2.T,X_2)
theta = np.linalg.inv(theta)
temp = np.matmul(X_2.T, y)
theta = np.matmul(theta, temp)

print(theta)

>> [-10.7017505   -0.12339677   6.67486843]
```

## c) Using CVXPY

```
theta = cp.Variable(3)
cost = cp.sum_squares(X_2@theta-y)
prob = cp.Problem(cp.Minimize(cost))
prob.solve()
theta.value

print(theta.value)

>> [-10.7017505   -0.12339677   6.67486843]
```

## d) Gradient Descent Derivation

d) $\mathcal{E}_{train}(\theta) = \|y - X\theta\|^2$

$\nabla_\theta \mathcal{E}_{train}(\theta) = \nabla_\theta \|y - X\theta\|^2$

$= -2X^T(y - X\theta)$

$\nabla_\theta \mathcal{E}_{train}(\theta) = 2X^T(X\theta - y)$

The gradient descent step is given by:

$$\theta^{k+1} = \theta^k - \alpha^k 2X^T(X\theta^k - y)$$

The step size using exact line search is given by:

$$\alpha^k = \arg\min_\alpha \mathcal{E}(\theta + \alpha \bar{d})$$

for steepest descent, $\bar{d} = -\nabla \mathcal{E}(\theta)$

Comparing it to the example covered in Lecture 6,

$$\alpha^k = -\frac{(H\theta^k + c)^T \bar{d}^k}{\bar{d}^k H \bar{d}^k}$$

in this case, $H = X^T X$, $c = -2X^T y$

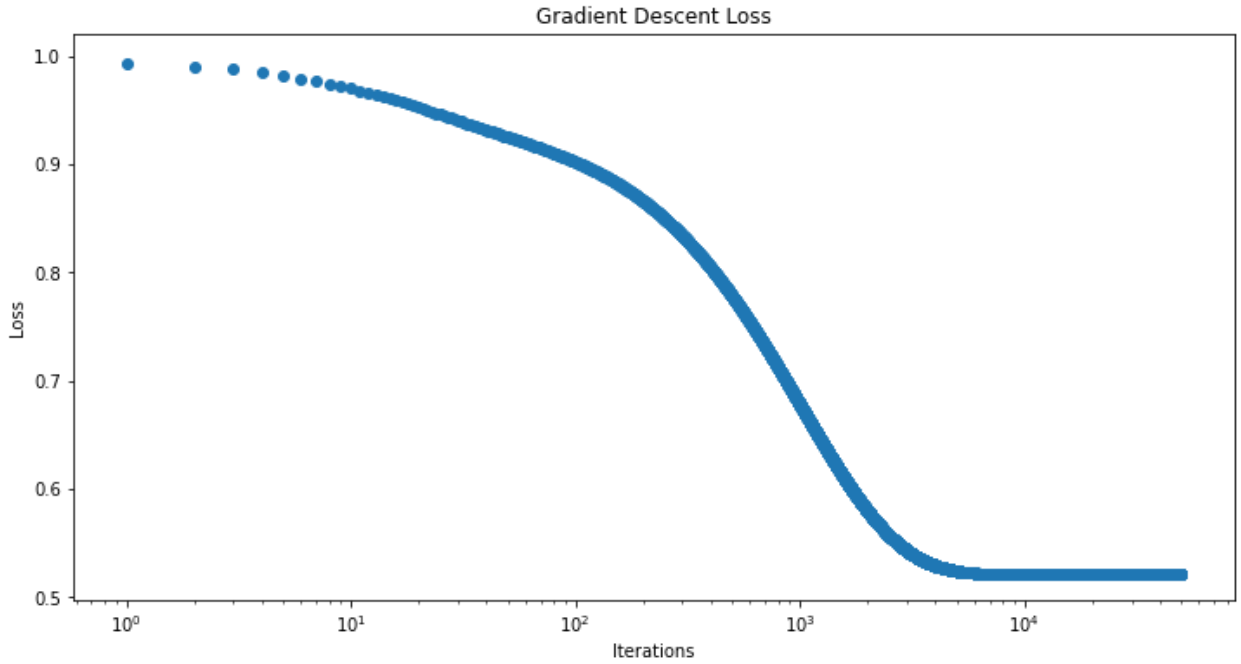
$d = -2X^T(X\theta - y)$

## e) Using Gradient Descent

```
d = 2
N = X_2.shape[0]
itr = 50000
theta = np.zeros(d+1)
cost  = np.zeros(itr)
XtX = np.dot( np.transpose(X_2), X_2)

# Gradient descent
for itr in range(itr):
  dJ     = np.dot(np.transpose(X_2), np.dot(X_2, theta)-y)
  dd     = dJ
  alpha  = np.dot(dJ, dd) / np.dot(np.dot(XtX, dd), dd)
  theta  = theta - alpha*dd
  cost[itr] = np.linalg.norm(np.dot(X_2, theta)-y)**2/N

print(theta)
```

```
>> [-10.7017505   -0.12339677   6.67486843]
```

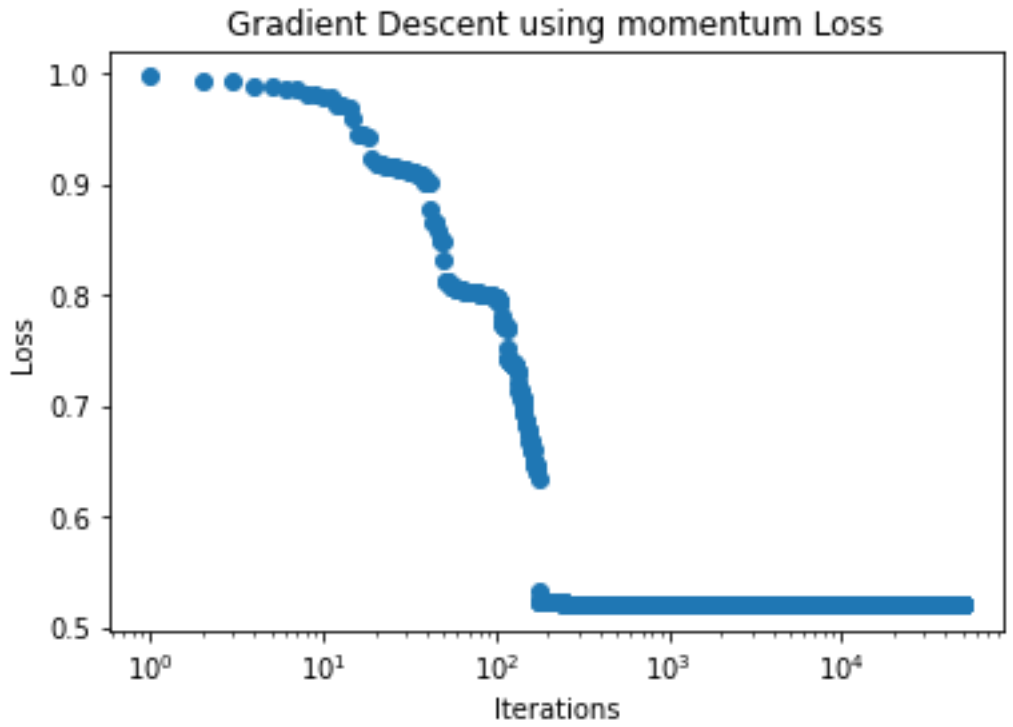
## f)



## g) GD using momentum

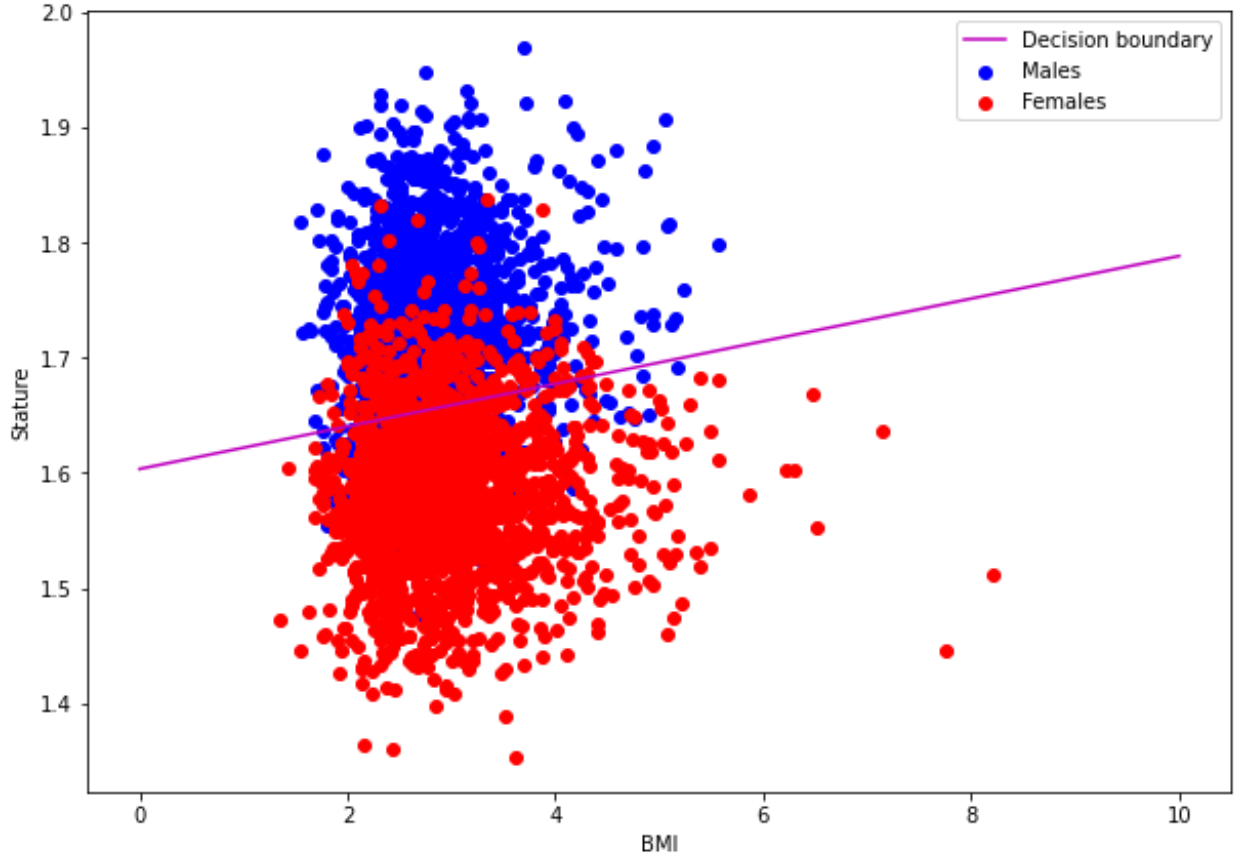
```
d = 2
N = X_2.shape[0]
itr = 50000
theta = np.zeros(d+1)
cost  = np.zeros(itr)
dJ_old = np.zeros(d+1)
XtX = np.dot( np.transpose(X_2), X_2)

beta  = 0.9
for itr in range(itr):
  dJ     = np.dot(np.transpose(X_2), np.dot(X_2, theta)-y)
  dd     = beta*dJ_old + (1-beta)*dJ
  alpha  = np.dot(dJ, dd) / np.dot(np.dot(XtX, dd), dd)
  theta  = theta - alpha*dd
  dJ_old = dJ
  cost[itr] = np.linalg.norm(np.dot(X_2, theta)-y)**2/N

print(theta)

>> [-10.7017505   -0.12339677   6.67486843]
```

## h)



## Exercise 3:

### a)

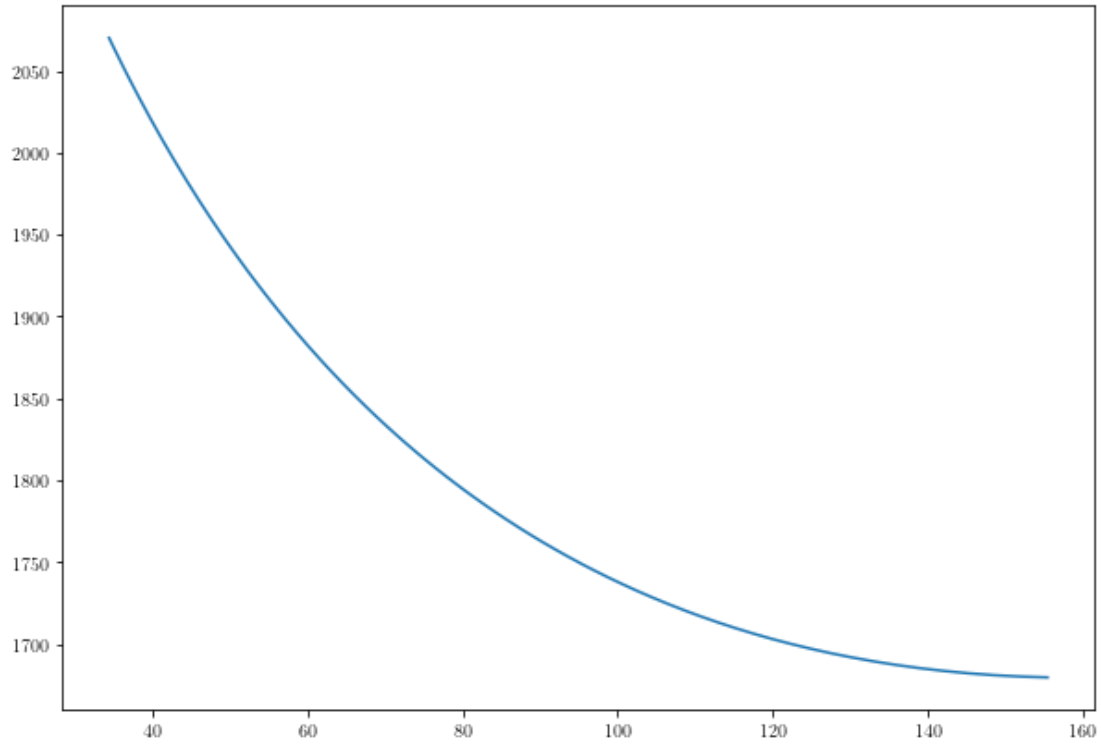



### b) Evaluation metrics

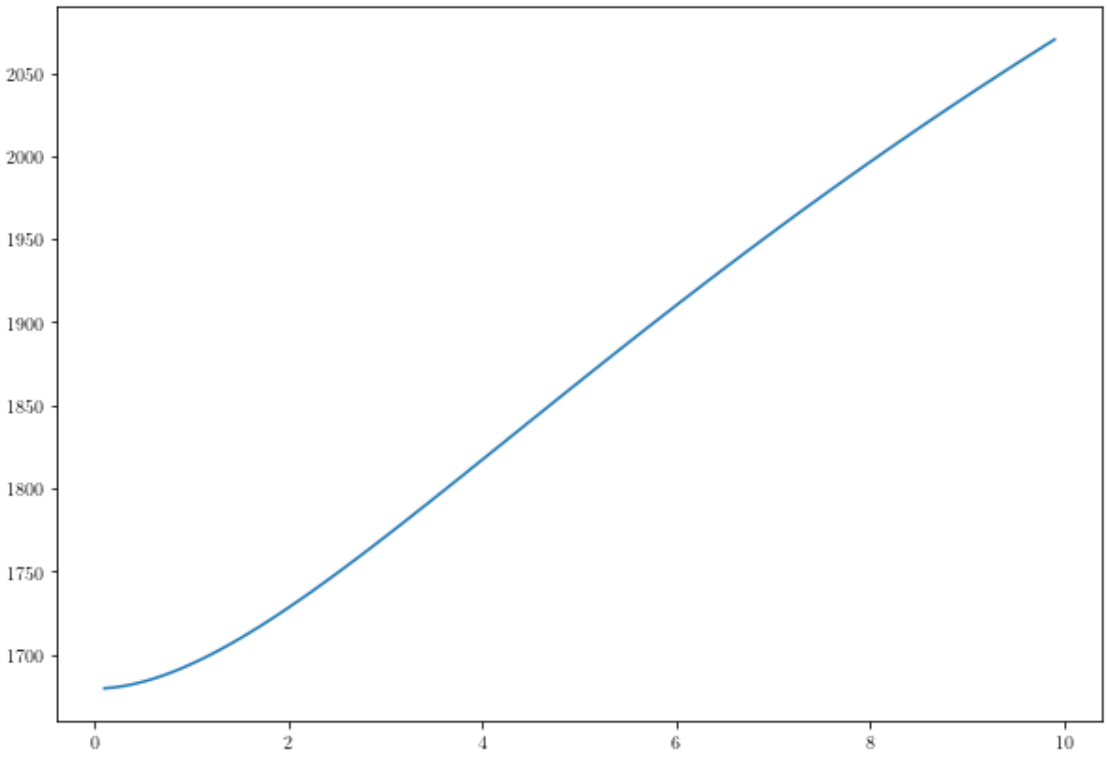
| | |
|---|---|
| Type 1 error | 14.17 % |
| Type 2 error | 14.17 % |
| Precision | 0.85 |
| Recall | 0.82 |

## Exercise 4:

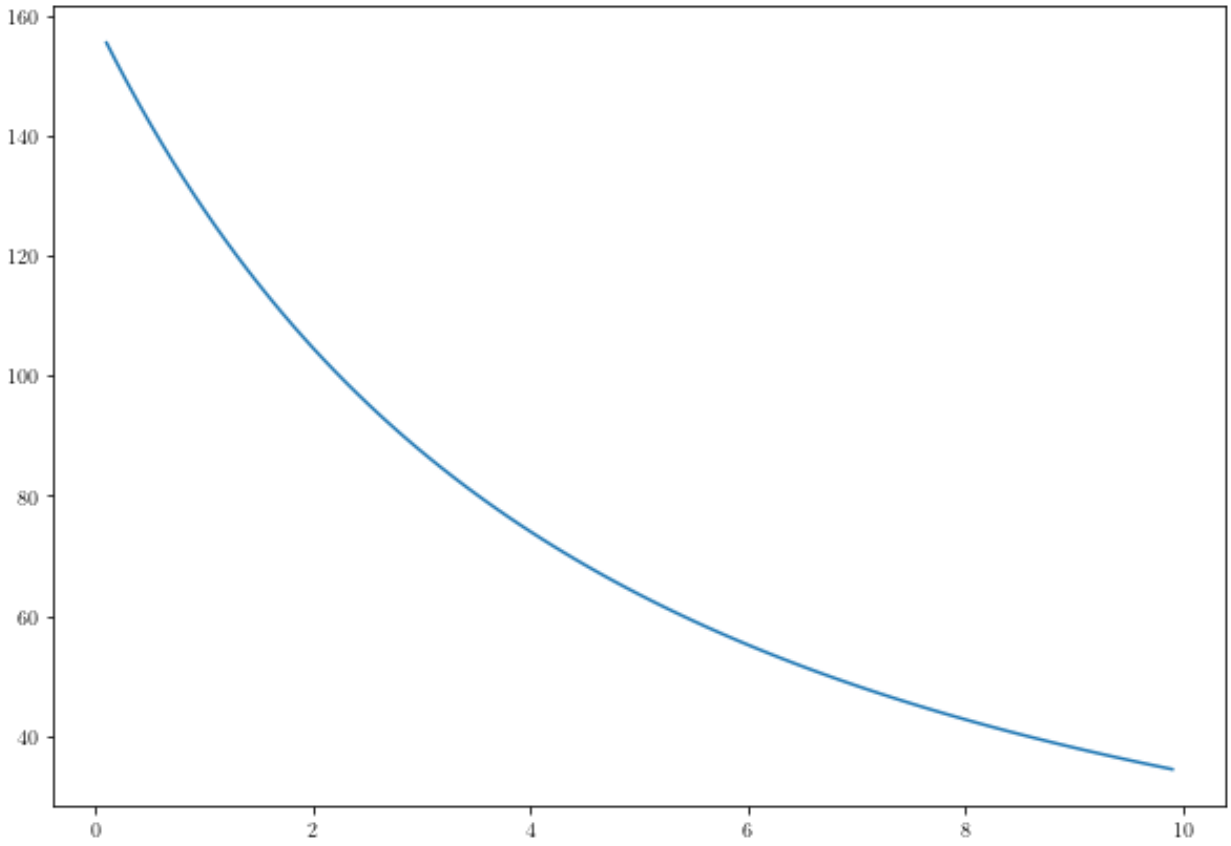
$$\| X\theta - y \|_2^2 \ vs \| \theta_\lambda \|_2^2$$




$$\| X\theta - y \|_2^2 \ vs \lambda$$



$\| \theta_\lambda \|_2^2 \ vs \ \lambda$



Exercise 4

b) i) For eq^n 3

$$\mathcal{L}(\theta) = \|X\theta - y\|_2^2 + \lambda \|\theta\|_2^2$$

For eq^n 4

$$\mathcal{L}(\theta, r_\alpha) = \|X\theta - y\|_2^2 - r_\alpha \left(\alpha - \|\theta\|_2^2\right)$$

For eq^n 5

$$\mathcal{L}(\theta, r_\epsilon) = \|\theta\|_2^2 - r_\epsilon \left[\left[\epsilon - \|X\theta - y\|_2^2\right]\right]$$

ii) For eq^n 3

1) $\nabla_\theta \mathcal{L}(\theta^*) = 0$

$$2X^T(X\theta^* - y) + 2\lambda\theta^* = 0$$

$$(X^TX + \lambda I)\theta^* = X^Ty$$

2) Conditions 2), 3) and 4) do not exist for eq^n 3

For eq^n 4

1) $\nabla_\theta \mathcal{L}(\theta^*, \Gamma_\alpha) = 0$

$$2X^T(X\theta^* - y) - \Gamma_\alpha(-2\theta^*) = 0$$
$$X^TX\theta^* - X^Ty + \Gamma_\alpha\theta^* = 0$$
$$\theta^* = (X^TX + \Gamma_\alpha I)^{-1}X^TY$$

2) $\alpha - \|\theta^*\|_2^2 \geq 0$
$\alpha \geq \|\theta^*\|_2$

3) $\Gamma_\alpha \geq 0$

4) $\Gamma_\alpha \left( \alpha - \|\theta^*\|_2^2 \right) = 0$

$\Gamma_\alpha \cdot \alpha = \Gamma_\alpha \|\theta^*\|_2^2$

For eq^ 5

1) $\nabla_\theta \mathcal{L}(\theta^*, \Gamma_\epsilon) = 0$

$2\theta^* - \Gamma_\epsilon \left( -2X^T (X\theta - y) \right) = 0$

$2\theta^* + 2\Gamma_\epsilon X^T X \theta^* - 2\Gamma_\epsilon X^T y = 0$

$I\theta^* + \Gamma_\epsilon X^T X \theta^* = \Gamma_\epsilon X^T y$

$\theta^* = (I + \Gamma_\epsilon X^T X)^{-1} \Gamma_\epsilon X^T y$

2) $\mathcal{E} - \|X\theta^* - y\|_2^2 \geq 0$

$\mathcal{E} \geq \|X\theta^* - y\|_2^2$

3) $\Gamma_\epsilon \geq 0$

4) $\Gamma_\epsilon \left( \mathcal{E} - \|X\theta^* - y\|_2^2 \right) = 0$

$\Gamma_\epsilon \cdot \mathcal{E} = \Gamma_\epsilon \|X\theta^* - y\|_2^2$

iii) fixing $\lambda > 0$

$$\hat{\theta}_\lambda = (X^T X + \lambda I)^{-1} X^T y$$

if $\hat{\theta}_\lambda$ is a sol$^n$ of eq$^n$ 4, we have

$$\hat{\theta}_\lambda = (X^T X + \Gamma_\alpha I)^{-1} X^T y$$

$$(X^T X + \lambda I)^{-1} X^T y = (X^T X + \Gamma_\alpha I)^{-1} X^T y$$

$$\therefore \Gamma_\alpha = \lambda$$

Using the complementary slackness cod$^n$

$$\lambda ( \alpha - \| \hat{\theta}_\lambda \|_2^2 ) = 0$$

as $\lambda > 0$

$$\alpha = \| \hat{\theta}_\lambda \|_2^2$$

iv) fixing $\lambda > 0$

$$\hat{\theta}_\lambda + \Gamma_\epsilon ( X^T ( X \hat{\theta}_\lambda - y ) ) = 0$$

$$\Gamma_\epsilon = -\hat{\theta}_\lambda \left[ X^T ( X \hat{\theta}_\lambda - y ) \right]^{-1}$$

Substituting $\hat{\theta}_\lambda = (X^T X + \lambda I)^{-1} X^T y$

$$r_\epsilon = -(X^TX + \lambda I)^{-1} X^T y \left[ X^T (X(X^TX + \lambda I)^{-1} X^T y) - y \right]^{-1}$$

Using the complementary slackness property

$$r_\epsilon \left( \epsilon - \| X\hat{\theta}_\lambda - y \|^2 \right) = 0$$
$$\therefore \epsilon = \| X\hat{\theta}_\lambda - y \|^2$$

v) Fixing $\lambda > 0$, we find that $r_\alpha = \lambda$ an $\alpha = \| \hat{\theta}_\lambda \|_2^2$

To prove that $\hat{\theta}_\lambda$ is the sol^ to (4), we need to show that (4) is ~~strictly~~ convex

(4) is convex if $\nabla^2 f(\theta) \geq 0$

$$f(\theta) = \| X\theta - y \|_2^2$$

$$\nabla f(\theta) = 2X^T(X\theta - y)$$

$$\nabla^2 f(\theta) = 2X^T X$$

$X^T X$ is a square matrix and is always positive semi definite

Hence $\nabla^2 f(\theta) \geq 0$

$\therefore$ $f(\theta)$ is a convex func$^n$, and $\hat{\theta}_*$ is the solution to this problem.

## Exercise 5:

Conditional Generative Adverserial Networks (cGANs) addresses a shortcoming of traditional GANS. Traditional GANs generate data by randomly sampling from a latent space and then using a generator network to transform that random input into a new data point. However, the generated data does not follow any given condition, and hence the output can be unpredictable. cGANs, on the other hand, enable the generation of data that is conditioned on a specific input or attribute.

Machine Learning and Deep learning are the premier technologies in today's world, and both of them run on one single currency - data. Hence, the generation of data has become an important problem, as this data can be used to train autonomous vehicles, smart speakers and virtual assistants, wireless communications channels, and solve may more problems. Using cGANs, the generation of all kinds of data has become possible. We can generate conditioned data that was previously difficult or expensive to collect in the real world. For example – the performance and handling of an autonomous vehicle on a slippery road is an experiment that is difficult and dangerous to conduct in real life, but it is essential for autonomous vehicles to be trained on such conditions. Now we can generate artificial testing data using cGANs, and train vehicles using this artificial data.

There exist multiple implementations of cGANs on the internet, and different people have tried to implement it in their own method. I shall start playing around with a PyTorch implementation trained on the MNIST dataset [1].

While I have created discriminator-like classifier networks, I am not familiar with the architecture of Generator networks. Hence my next step would be to learn about them and develop a traditional GAN, before I move towards creating a cGAN.

[1] https://github.com/qbxlvnf11/conditional-GAN

# APPENDIX