# Optimization

## Problem

Sai Ashish Somayajula[1]

[1]EE16BTECH11043

November 14, 2019

# Problem

- Find w such that,

$$L = ||y - Xw||^2 \tag{1}$$

is minimized.
where,

$$X = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \tag{2}$$

$$y = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix} \tag{3}$$

# Least Squares Method

- $$\begin{aligned} \widehat{w} &= \min_{w} ||y - Xw||^2 \qquad (4) \\ &= (X^T X)^{-1} X^T y \qquad (5) \end{aligned}$$

# Gradient Descent

-
$$w_{n+1} = w_n - \mu \frac{\partial L}{\partial w} \tag{6}$$

  Till,

$$w_{n+1} \approx w_n \tag{7}$$

# Python code

```
#Gradient Descent
#Least Squares
import numpy as np
c = 2
cur_w = np.array([[1.0],[1.0]])
# The algorithm starts at w = [1,1]
gamma = 0.01 # step size multiplier
precision = 1e-6
previous_step_size = 1
max_iters = 10000
# maximum number of iterations
iters = 0 #iteration counter

X = np.array([[1.0,0.0],[1.0,1.0],[1.0,2.0]])
y = np.array([[6.0],[0.0],[0.0]])
```

# Python code

```
df = lambda x, y, w:
        (-2.0*np.matmul(x.T,(y-np.matmul(x,w))))

while (previous_step_size > precision)
                & (iters < max_iters):
    prev_w = cur_w
    cur_w -= gamma * df(X,y,prev_w)
    previous_step_size =
                np.linalg.norm(cur_w - prev_w)
    iters+=1
```

# Python code

```
# solution from Gradient Descent
Gradient_descent = curr_w

# equation for least squares
least_squares = np.matmul(
np.linalg.inv(np.matmul(X.T,X)),(np.matmul(X.T,y)))
```

# Answer

Answer:

- The minimum occurs at(gradient Descent)

$$w = \begin{pmatrix} 1.0 \\ 0.84 \end{pmatrix} \tag{8}$$

The minimum occurs at(least squares)

$$w = \begin{pmatrix} 1.0 \\ 0.84 \end{pmatrix} \tag{9}$$