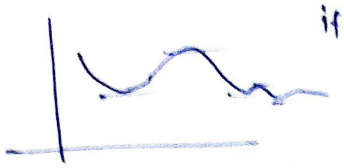
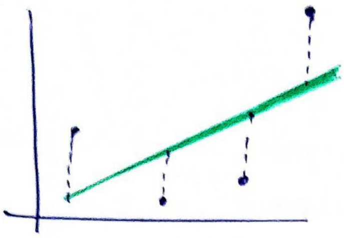
Gradient Descent

11:02am
FRI, Apr, 25



if we provide differentiable func it gives the minimum.

Odinal least square



$\hat{y} = mx_i + b$

$J = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

$J = \sum_{j=1}^{n} (y_i - mx_i - b)^2$

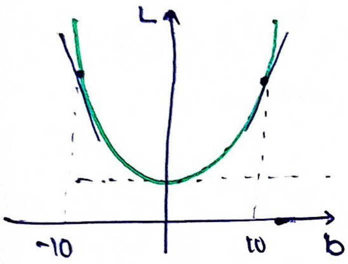
$m = 78.35$

$J(m,b) = \sum (y_i - \overset{78.35}{m}x_i - b)^2$

now only dependent upon $J(b)$

Rel^n of L & b

$L \to (b)^2$



min = ?

Step-1 select a Random b

let b = 0

How algo know where go up↑ down↓

How to find slope of

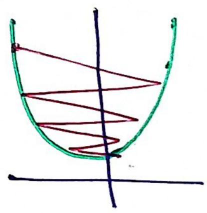
$y = x^2$

$\left.\frac{dy}{dx}\right|_{x=5} = 2x = 10$

$\boxed{b_{new} = b_{old} - \text{slope}}$

if slope (+ve) ⟶ b↓

−ve ⟶ b↑



Due to slope it change very fast so

$b = -10$

let slope = −50

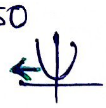
$b_{new} = -10 - (-50)$

$= +40$

b incre

$b_{new} = 10 - (50$

$= -40$



$b_{new} = b_{old} - \eta\,\text{slope}$

η - learning Rate

0.01

eg- $b_{new} = -10 - (0.01 \times -50)$

$= -10 + 0.5$

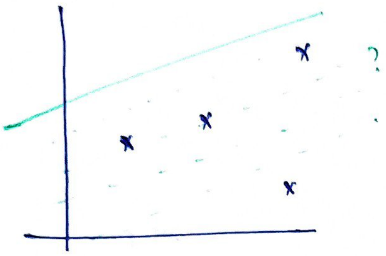
$= -9.5$

? when to stop

$b_{new} - b_{old} \sim 0$

1. diff > 0.0001
2. Iteration (epochs)

Mathematical Formulation



m = 78.35 (Known)

?

(m, b)

Step-1 Start with Random b

for i in epochs:

$b_{new} = b_{old} - \eta \, slope_{b=old}$

$$J = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\frac{d}{db} J = \frac{d}{db} \sum_{i=1}^{n} (y_i - m x_i - b)^2$$

$$= 2 \sum_{i=1}^{n} (y_i - m x_i - b)(-1)$$

$$slope = -2 \sum_{i=1}^{n} (y_i - m x_i - b)$$

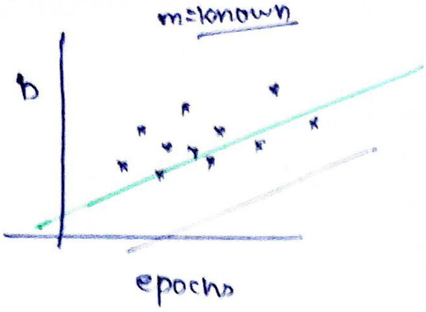
$$slope\Big|_{\substack{b=0 \\ m=78.35}} = -2 \sum_{i=1}^{n} (y_i - 78.35 x_i - 0)$$

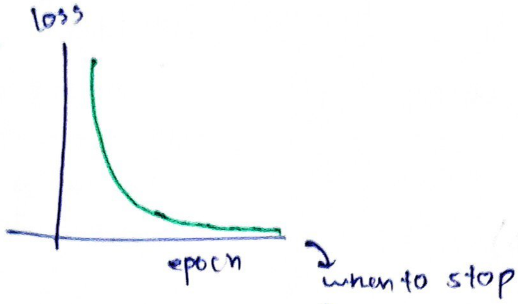
we have 4 point now slove this ↑

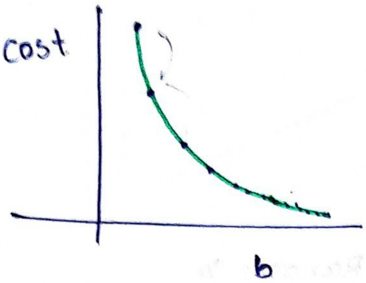
~~is~~ 4 we got slope

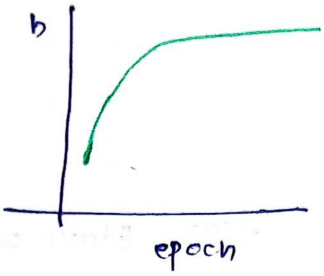
then we got b_{new}.

$$b_{new} = b_{old} - \underbrace{\eta \, slope_{b=old}}_{step\_size}$$









Adding 'm'

S1: Random m & b

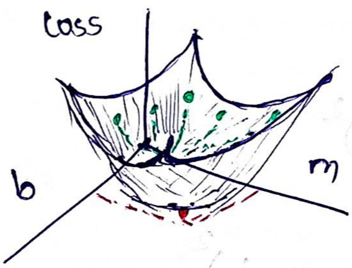
$lr = 0.01$
epochs = 100

S2: for i in epochs:

$$b = b - \eta \overset{b}{\text{slope}}$$

$$m = m - \eta \overset{m}{\text{slope}}$$

Cost fun^c



215 pm

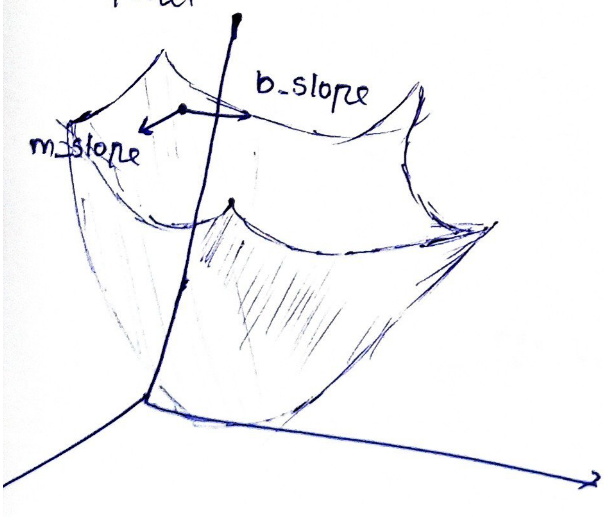
$$J = \sum (Y_i - \hat{Y})^2$$

$$J_{(m,b)} = \sum (Y_i - mX_i - b)^2$$

$$\frac{\partial L}{\partial b} = -2\sum (Y_i - mX_i - b) = b\_slope$$

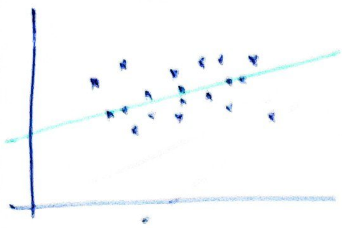
$$\frac{\partial L}{\partial m} = 2\sum (Y_i - mX_i - b)(X_i) = m\_slope$$

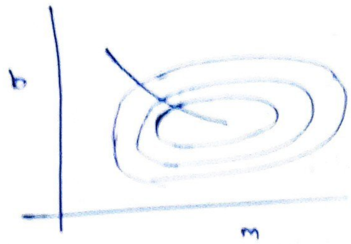
componet

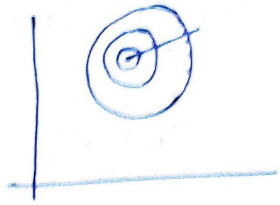


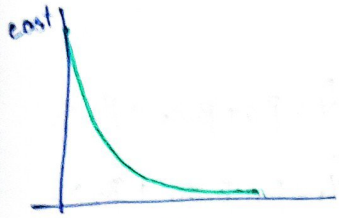
Effect of Data

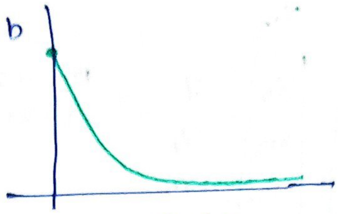
, Standardized Data

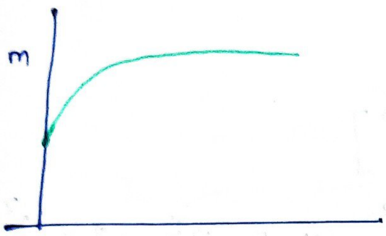








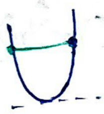




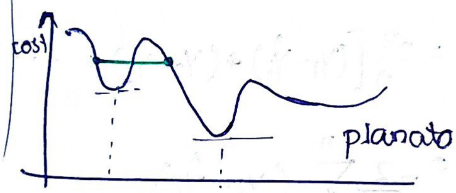
Effect of loss function

$$J = \sum (Y_i - \hat{Y}_i)^2$$

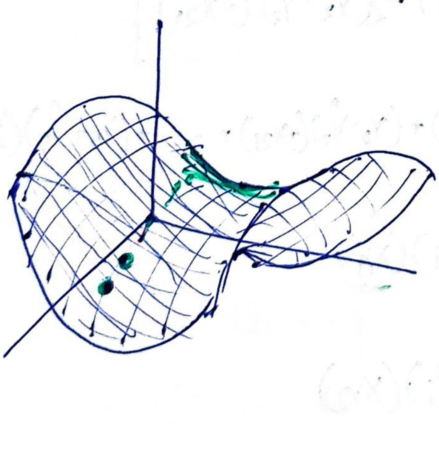
convex func



non covex func



touch two point w/o intercept any other.



312 pm

Type

Gradient Descent

- Batch GD — Total
- Stochastic GD — one data
- Mini-Batch GD — some data

cgpa X_1	iq X_2	lpa Y	
8.1	93	3.2	Y_1
9.2	97	5.2	Y_2

X_{i1}	X_{i2}	
X_1	X_2	Y
X_{11}	X_{12}	Y_1
X_{21}	X_{22}	Y_2

row → 2
col → 2+1

$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2$

$\hat{Y}_1 = \beta_0 + \beta_1 X_{11} + \beta_2 X_{12}$

$\hat{Y}_2 = \beta_0 + \beta_1 X_{21} + \beta_2 X_{22}$

$$\underset{\text{lpa}}{\hat{Y}} = \beta_0 + \beta_1 \underset{\text{cgpa}}{X_1} + \beta_2 \underset{\text{iq}}{X_2}$$

1. Random values
   $\beta_0 = 0$, $\beta_1, \beta_2 = 1$

2. epochs = 100, lr = 0.01

$\beta_0 = \beta_0 - \eta\, b_0\text{-slope}$

$\beta_1 = \beta_1 - \eta\, b_1\text{-slope}$

$\beta_2 = \beta_2 - \eta\, b_2\text{-slope}$

our loss func is

$J(\beta_0, \beta_1, \beta_2)$

$$J = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$= \frac{1}{2}\left[(Y_1 - \hat{Y}_1)^2 + (Y_2 - \hat{Y}_2)^2\right]$$

$$= \frac{1}{2}\left[(Y_1 - \beta_0 - \beta_1 X_{11} - \beta_2 X_{12})^2 + (Y_2 - \beta_0 - \beta_1 X_{21} - \beta_2 X_{22})^2\right]$$

$$\frac{\partial J}{\partial \beta_0} = \frac{1}{2}\left[2(Y_1 - \hat{Y}_1)(-1) + 2(Y_2 - \hat{Y}_2)(-1)\right]$$

$$= \frac{-2}{2}\left[(Y_1 - \hat{Y}_1) + (Y_2 - \hat{Y}_2)\right]$$

let we have n rows - - -

$$= \frac{-2}{n}\left[(Y_1 - \hat{Y}_1) + (Y_2 - \hat{Y}_2) + \cdots + (Y_n - \hat{Y}_n)\right]$$

$$\frac{\partial J}{\partial \beta_0} = \frac{-2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)$$

$$\frac{\partial J}{\partial \beta_1} = \frac{1}{2}\left[2(Y_1 - \hat{Y}_1)(-X_{11}) + 2(Y_2 - \hat{Y}_2)(-X_{21})\right]$$

$$= \frac{-2}{n}\left[(Y_1 - \hat{Y}_1)(X_{11}) + (Y_2 - \hat{Y}_2)(X_{21}) + \cdots + (Y_n - \hat{Y}_n)(X_{n1})\right]$$

$$= \frac{-2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(X_{i1})$$

$$\frac{\partial J}{\partial \beta_2} = \frac{-2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(X_{i2})$$

⋮

$$\frac{\partial J}{\partial \beta_m} = \frac{-2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(X_{im})$$

$$\frac{\partial J}{\partial \beta_0} = \underbrace{\frac{-2}{n}\sum_{i=1}^{n}}_{\text{Mean}}(y_i - \hat{y}_i)$$

y_i: x_train $\hat{y}_i$: prediction

	X_1	X_2	X_3	Y
$\hat{y}_1$	-	-	-	-
$\hat{y}_2$	-	-	-	-

$= -2 \times \frac{1}{n}\sum X_i$

= it is a scalar

$\hat{y}$ = np.dot(xtrain, coeff) + β_0 OR

$\hat{y}$ = np.dot(x_train, coeff) + β_0

$= (353, 10)\ (10, 1) + \beta_0$

$= (353, 1) + \beta_0$

= y_pred

OR

$\hat{y}_1 = \beta_0 + \beta_1 X_{11} + \beta_2 X_{12} + \beta_3 X_{13}$

$\hat{y}_2 = \beta_0 + \beta_1 X_{21} + \beta_2 X_{22} + \beta_3 X_{23}$

$$\hat{y} = \beta_0 + [X_{11}\ X_{12}\ X_{13}]\begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix}$$

$\hat{y}$ = np.dot(coeff_, x_train) + β_0

X_1	X_2	Y	$\hat{Y}$
1	2	5	6
3	4	7	8

all dot $\left[(y_i - \hat{y})_{(1\times2)}\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}_{(2\times2)}\right] \times \frac{-2}{2}$

$$\frac{\partial J}{\partial \beta_1} \ldots\ldots \frac{\partial J}{\partial \beta_m} = \left[(y_i - \hat{y}_i)\ \text{X\_train}\right] \times \frac{-2}{n}$$

n: no of row

$$\frac{\partial J}{\partial \beta_1} = \frac{-2}{n}\sum_{i=1}^{2}(y_i - \hat{y}_1)(x_{11})$$

$y - \hat{y} = [5\ \ 7][6\ \ 8]$

$= [-1\ \ -1]$

$(y - \hat{y})(x_{11}) = [-1\ \ -1]_{1\times2}\begin{bmatrix}1\\ 3\end{bmatrix}_{2\times1}$

$= [(-1\times1 - 3) + (-1\ -3)]$

$= [(-4) + (-4)]$

$= 8.$

$\frac{-2}{2}(-8) = +8$

$$\frac{\partial J}{\partial \beta_2} = \frac{-2}{2}\left[[-1\ \ -1]\begin{bmatrix}2\\ 4\end{bmatrix}\right]$$

$= -1 \times [(-2 - 4) + (-2 - 4)]$

$= \cancel{-8}$

The problem w/ Batch GD

let

$n \rightarrow 100$ $\quad 10^5$

col $\rightarrow 5$ $\quad 10^2$

epoch $\rightarrow 50$ $\quad 10^3$

for 1 col $\rightarrow$ 1000

6 col $\rightarrow$ 6000

Total = 6000×50 $\quad 10^{10}$

1. Slow v.big data
2. Hardware

Stochastic GD

single row one update
less epochs

$$\frac{\partial L}{\partial \beta_0} = \frac{-2}{n} \sum_{i=}^{n} (y_i - \hat{y})$$

we calculate for single row so $n=1$

$$= -2(y_i - \hat{y}_i)$$

Time Comparsion

$e=100$

batch — 100 update

stochastic — $100 \times n$ updates

if the epoch is same then batch GD is faster.

due to this SGD don't required that much epoch so eve at the end it fast.

learning Schedules

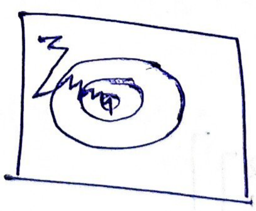
Due to randomness at end also it don't show optimal point so we vary w/ the epocsh.

```
t0, t1 = 5, 50
def learning_rate(t):
        return t0/(t + t1)
for i in range(epochs):
        for j in range(X.shape[0])
              lr = learning_rate(i * X.shape
```

when to use

big Data

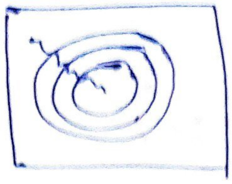
Non convex



9:30 pm

Mini Batch GD

SGD is not optimal soln.



n rows = 1000

batches = 100

means = $\frac{1000}{100}$ = 10 updates/epoch

n rows

batch_size → n 1/epoch BGD

→ 1 n/epoch SGD