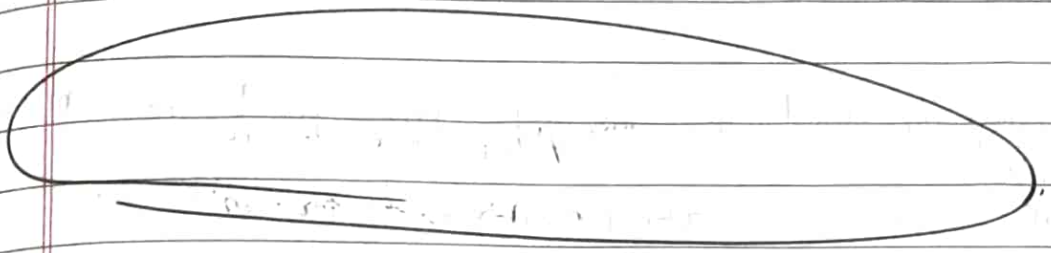
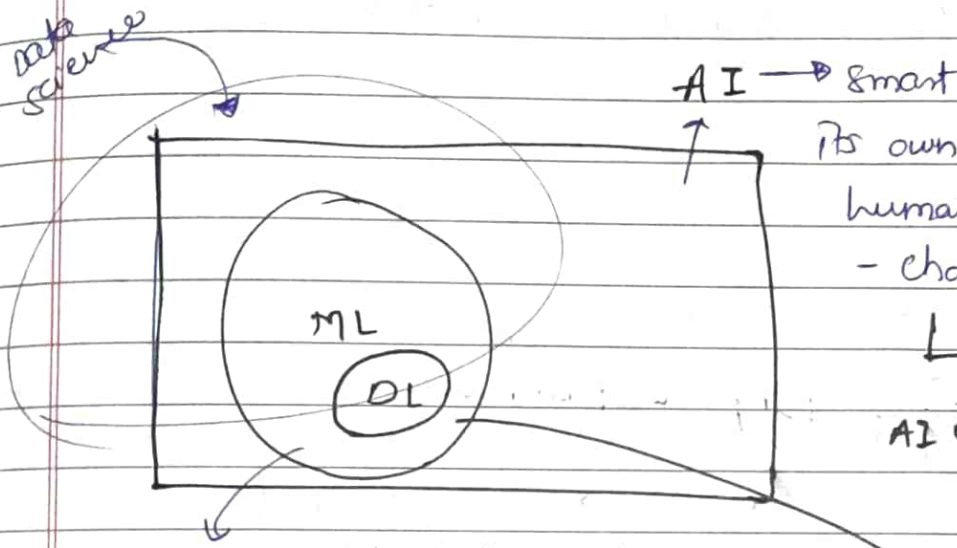
Simple linear regression

Supervised ML → Regression → for continuous value
Supervised ML → Classification → Categorical value

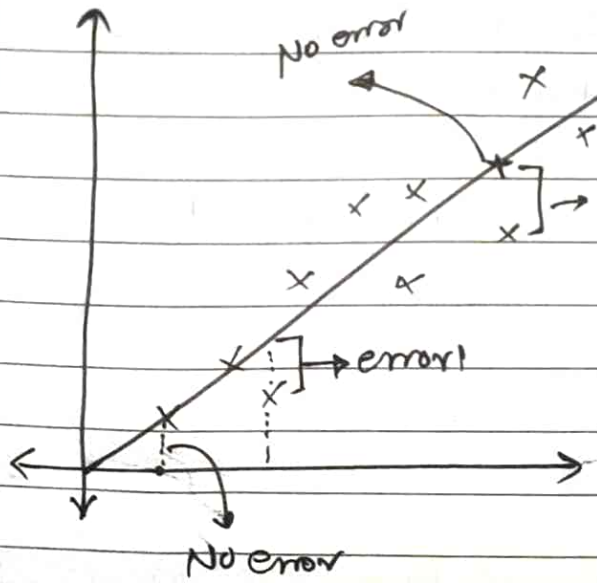




AI → Smart application that perform its own task without any human intervension.
- chatgpt
AI end product

provide stats tool to perform explore, analyze, visualize and perform prediction.
Eg:→ Recomendation system
Netflix
AI powered application.

mimic human brain
- Multilayered Neural Network.



aim:
Create best fit line for data in such a way that error + error2 + ——— errorn have to be minimum,
- In case of two diamensional → it will be bestfitline

- for 4D there will be hyperplane.

classmate Date Page

Equation of basefit line :→

$$\hat{y} : h_\theta(x) = \theta_0 + \theta_1 x_i$$

θ_1 → slope

θ_0 → y-intercept

- By initializing θ_0 and θ_1 we get the first fit plane.
- Then over the iteration we optimize it and for with line there is low error this line/plane is best plate plane or basefit ~~dat~~ plane

Cost function :→

error :

$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - h_\theta(x)\right)^2$$ ——— (mean squared error).

$h_\theta(x)$ → predicted point

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

$\hat{y}_i$ → predicted point

n → no of data point

y_i → actual data point

loss function :→

whenever we calculate the error for one point This is known as loss function

$$= \left(y_i - \hat{y}_i\right)^2$$ ——— (1 data point).

classmate
Date
Page

minimize $J(\theta_0, \theta_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - h_\theta(x_i))^2$

↓ To minimize this equation [value] error

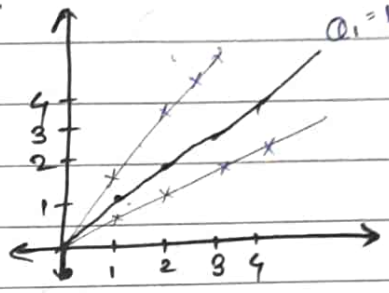
optimation:- [minimize the cost function].

$\theta_1 = 1$

$J(\theta_1) = \frac{1}{n}\sum_{i=1}^{n=4}(y_i - h_\theta(x_i))^2$

$= \frac{1}{4}[(1-1)^2 + (2-2)^2 + (3-3)^2 + (4-4)^2]$

$= 0$ → minimum error [can't be negative]



$\theta_1 = 0.5$

$J(\theta_1) = \frac{1}{n}[y_i - h_\theta(x_i)]^2$

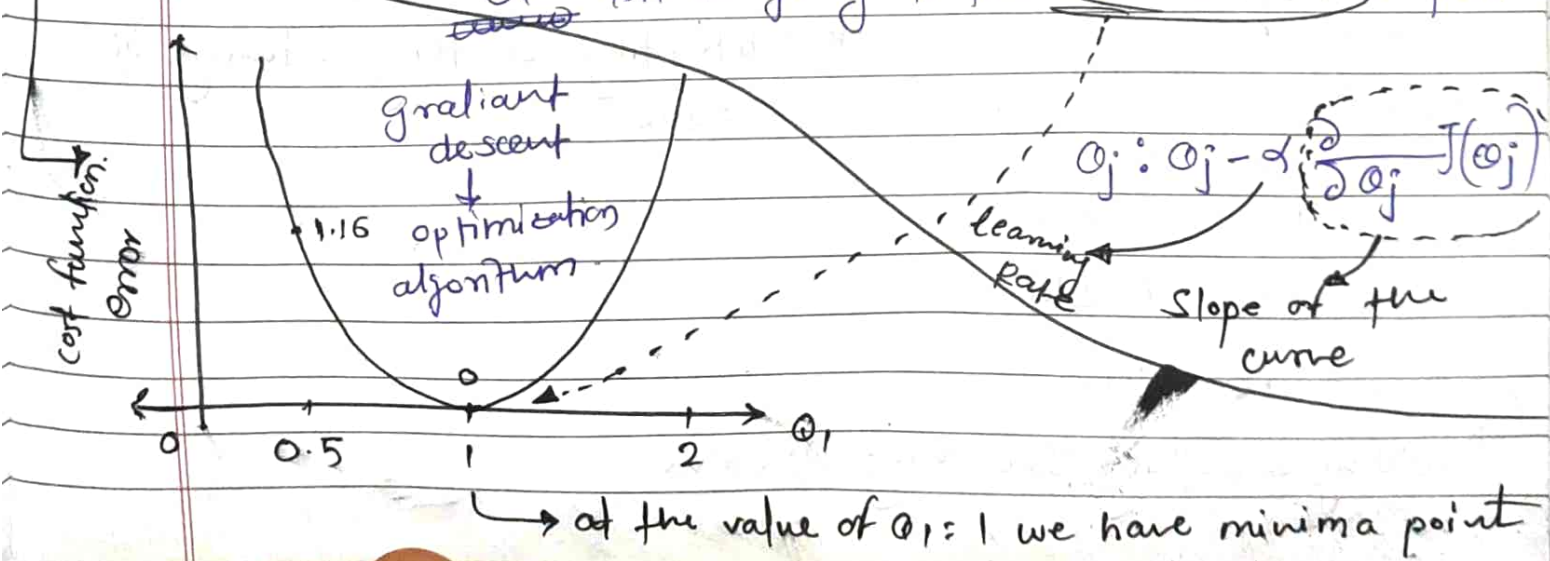
$= \frac{1}{4}[(1-0.5)^2 + (2-1)^2 + (2-(0.5)(3))^2 + (4-2)^2]$

$= 1.16$

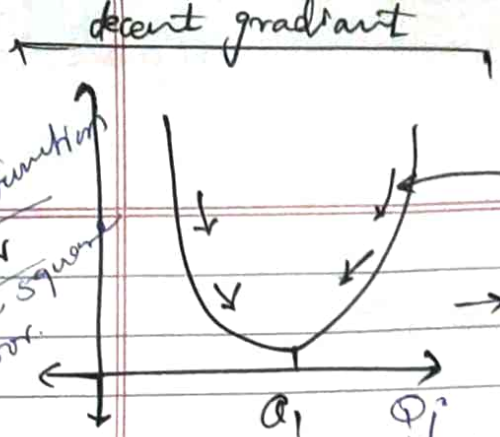
* Convergence algorithm :→

Repeate until Convergence global minima.

→ Repeate the above process with different value of θ_1 till we getting to the minimum error point



→ at the value of $\theta_1 = 1$ we have minima point



Traveling to minimize the error

→ optimization is used to minimize the error

↓

algorithm.

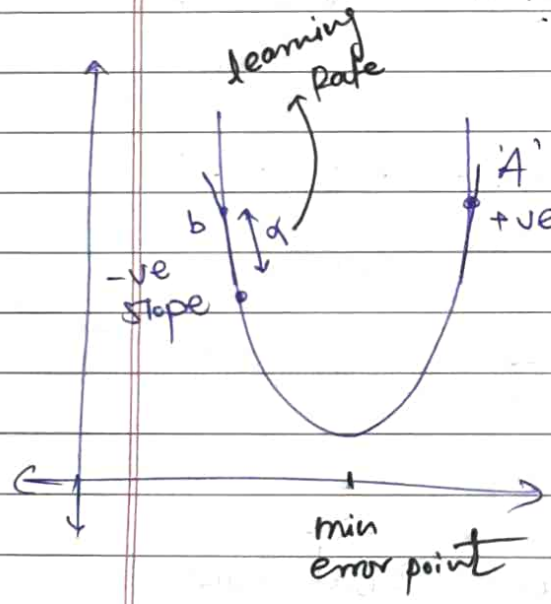
∴ The above traveling to make the error minimum is done by the optimization algorithm

Convergence algorithm

$$\{\ \theta_j := \theta_j - \alpha \left[\frac{\partial}{\partial \theta_j} J(\theta_j) \right] \}$$

α → learning Rate

$\frac{\partial}{\partial \theta_j} J(\theta_j)$ → slope at point



for point 'A'

- To Reduce error we have to decrease 'θ_j'

with above equation

$\theta_j := \theta_j - \alpha$ (+ve value)

↓ Slope

$\therefore \theta_j - \alpha$ (+ve value)

↳ Subtracting something Reducing θ_j value

for point 'b'

- To Reduce error we have to Increase 'θ_j'

with above equation

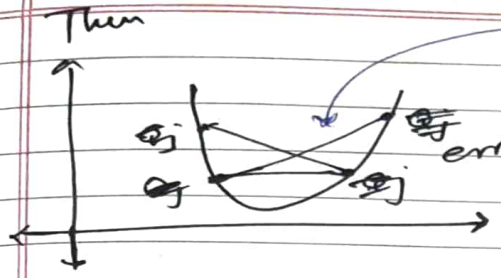
$\theta_j := \theta_j - \alpha$ (-ve slope)

$\therefore \theta_j := \theta_j + \alpha$ (slope)

↳ adding Something Increasing θ_j value

∴ If our learning Rate is too height

(α) is too height



- we cant able to reach at the minimum error point

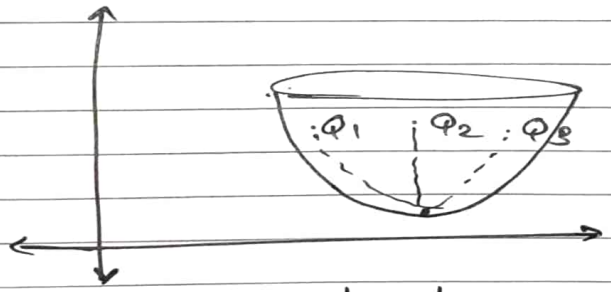
* if α is too small → minimum error

Then it will take too much time to reach the error point

* for multiple independent variable :-

$h_\theta(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 \text{ ——— } \theta_n f_n$

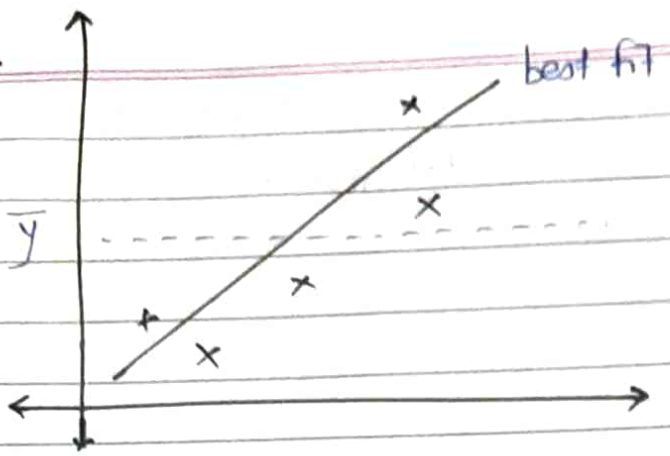
--- multiple leanear regression.



gradiant descent curve.

* Performance Matrix :→

* R-squared



$$R\text{-squared} = \frac{SS_{regression}\ (\text{Best fit line})}{SS_{total}\ (\text{average of } y)}$$

SS → Sum of square Residual or error

SS total → Sum of square total

$$R\text{squared} = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

This term will always greater than numerator

$\therefore$ R squared = 1 - (value < 1)

* Adjusted Rsquared :→

Adjusted Rsquared < R squared

$$\text{Adjusted R squared} = 1 - \frac{(1-R^2)(N-1)}{N-P-1}$$

N: no. of data points

P: No. of independent predictors

Let's understand the adjusted R^2 → with example.

(1) let $R^2 = 80\%$, $N = 11$, $P = 2$

Adjusted R-Squared = $1 - \frac{(1-0.8)(10)}{11-2-1}$

$= 0.75$

$= 75\%$

Highly Correlated	feature	R^2	adjusted R^2
	2	80%	75%
→	3	85%	78%
	4	86%	77%

(2) Now, $P = 3$

~~Then~~ The feature added is highly Co-related Then

$R^2 = 85\%$

adjusted $R^2 = 78\%$

(3) Now, $P = 4$

- added feature not highly correlated

Then

$R^2 = 86\%$

adjusted $R^2 = 77\%$

In linear Regression we use Mean square error as loss function.

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

advantage of MSE

① It is differentiable

② It has one local and one global minima

Disadvantage of MSE

① Not robust to outliers

[to reduce the error occured by the outlier
- algorithm will try ~~to reduce it~~ change the bestfit line]

② MSE changes the unit

[by making the square of it].

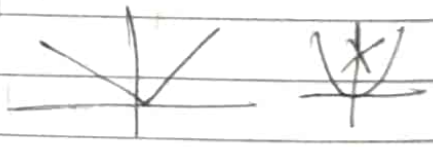
② → Mean Absolute Error :→ (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}|$$

→ Advantage of this :→

1) Robust to outliers
2) It will be in same unit.

disadvantages:
- time complexity is more for optimization
- instead of gradiant decent you get



③ Root mean square error :→

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - y_i)^2}$$

advantage

disadvantage

★ Ridge, Lasso and elastiene :→

Cost function for linear Regression

$$= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x)\right)^2$$

Note :→
always do feature scalling after the train test split only

① **Ridge Regression** → (L2 regularization)

- we are basically overfitting.

Reduce

Cost function :→

$$= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x)\right)^2 + \lambda\sum_{i=1}^{n}(\text{slope})^2$$

- $\frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x)\right)^2$ → 0 (zero) → overfitting
- $\lambda\sum_{i=1}^{n}(\text{slope})^2$ → Hyperparameter

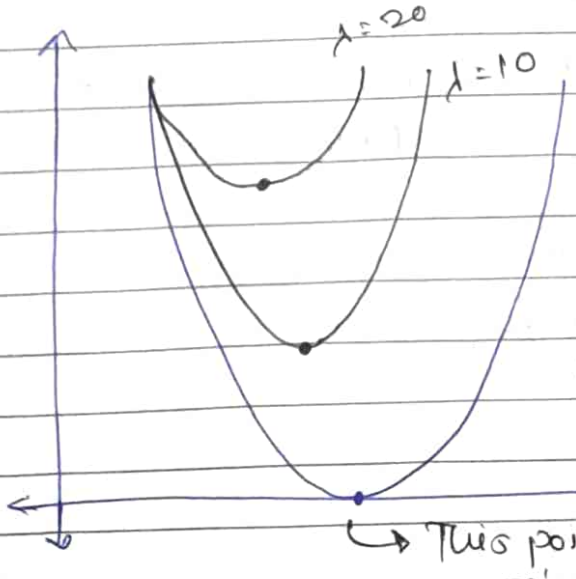
for parameter 2

$$\left[(a_1)^2 + (a_2)^2 + (a_3)^2\right]^2$$

a_1 → for parameter 1
a_2 → for parameter 2
a_3 → for parameter 3

- due to this we always adding error in the cost function of linear Regression
- So basically we reducing the overfitting.

★ Relation between 'λ' and slope :→



- as you can see

λ ↑↑↑ slope ↓↓ curve ↓ error ↑↑↑

and vice versa

② Lasso Regression :→ (L1 Regularization)

- Mainly used for feature selection

Cost function :→

$$= \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x)_i)^2 + \lambda \sum_{i=1}^{n} |\text{slope}|$$

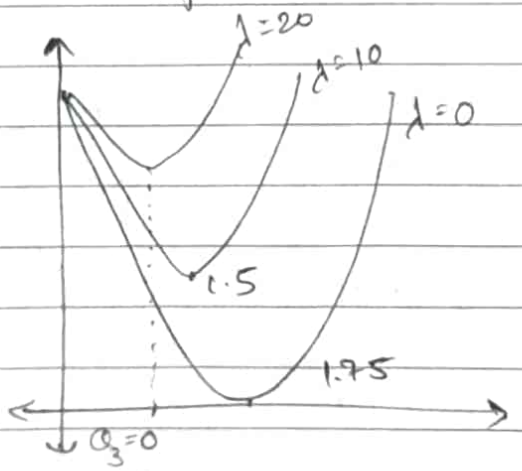
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$$h_\theta(x) : 0.52 + 0.65 x_1 + 0.72 x_2 + 0.12 x_3$$

x_1, x_2, x_3 → features | with unit change in x_i there is 0.65 change in output (prediction)

* So the x_3 contribute much less to predict the output

* Relationship between λ and |slope|



→ will reach to '0' because we use || absolute value not like Ridge.

* So it will remove the impact of less Correlated feature. internally

ElasticNet :→

- Combination of both
  - → Ridge → Reducing overfitting
  - → Lasso → feature selection.

Reducing overfitting.

Cost function :→ $\frac{1}{n}\sum_{i=1}^{n}(y_i - h_\theta(x)_i)^2 + \lambda_1 \sum_{i=1}^{n} (\text{slope})^2 +$

$+ \lambda_2 \sum_{i=1}^{n} |\text{slope}|$

feature selection

sns.pairplot(df)

- plot the scattered plot for all feature in pair.