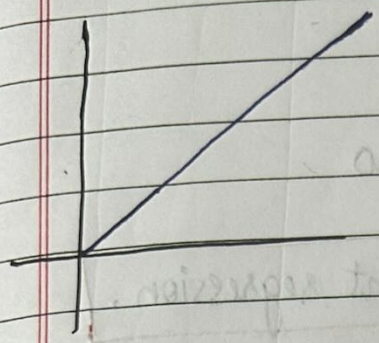
LINEAR REGRESSION

→ Linear Regression Equation- $y = mx + c$

Standard Equation
or
Straight line equation



$\frac{y_2 - y_1}{x_2 - x_1}$ } slope

* I love Python
0 1 2 → encoding

Machine learning sochta hai

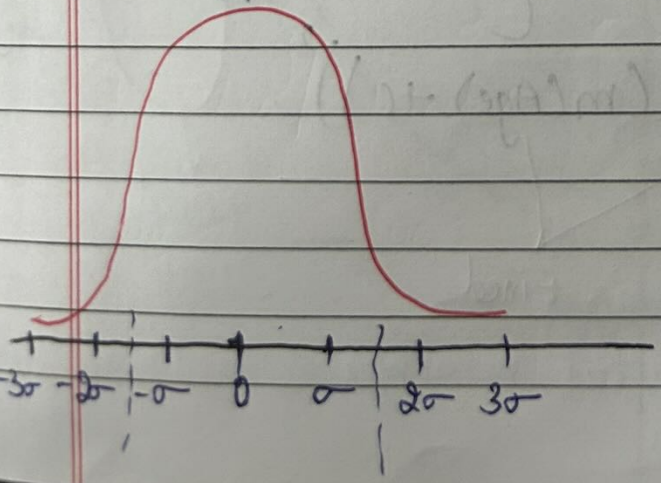
0 < 1 < (2) → gives priority

0 | 1 | 010 ↑

⇒ If distributions are skewed → NaN values with median
normal → NaN values with mean

⇒ Scaling should be same for all columns.

⇒ Har column ko 0 & 1 ke beech mein leke aate hai



⇒ ~~-σ~~ -σ & σ → 68%.
2 deviation ~~2~~ & 2
→ 97%.

9/1/23

Date / /
Page No.

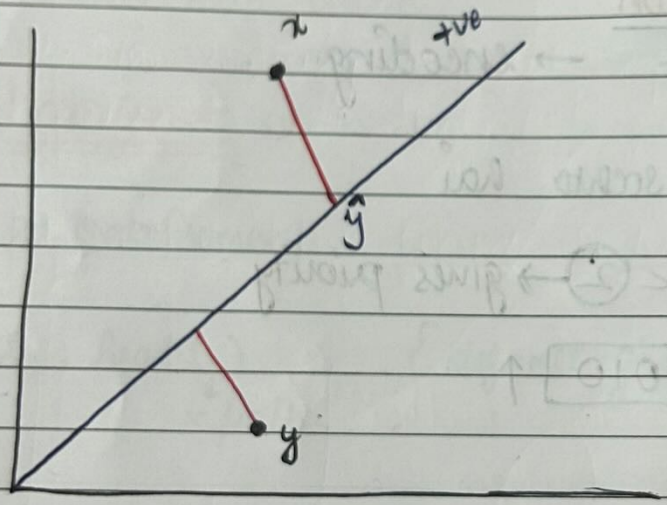
* How much far the predicted value is far from the actual value is loss

$$\text{Single (loss)} = \sum_{i=1}^{n} (y - \hat{y})$$

choose line jiska loss minimum ho

* Linear regression also called univasient regression.

10/1/23



$$L = \frac{\sum_{i=1} (y - \hat{y})^2}{N}$$

minimize this

Eg	Age	Salary	Diff.
	21	89	
	22	88	
	23	70	

$$\left\{ L = \frac{\sum (y - (mx + c))^2}{N} \right\}$$

Parameters

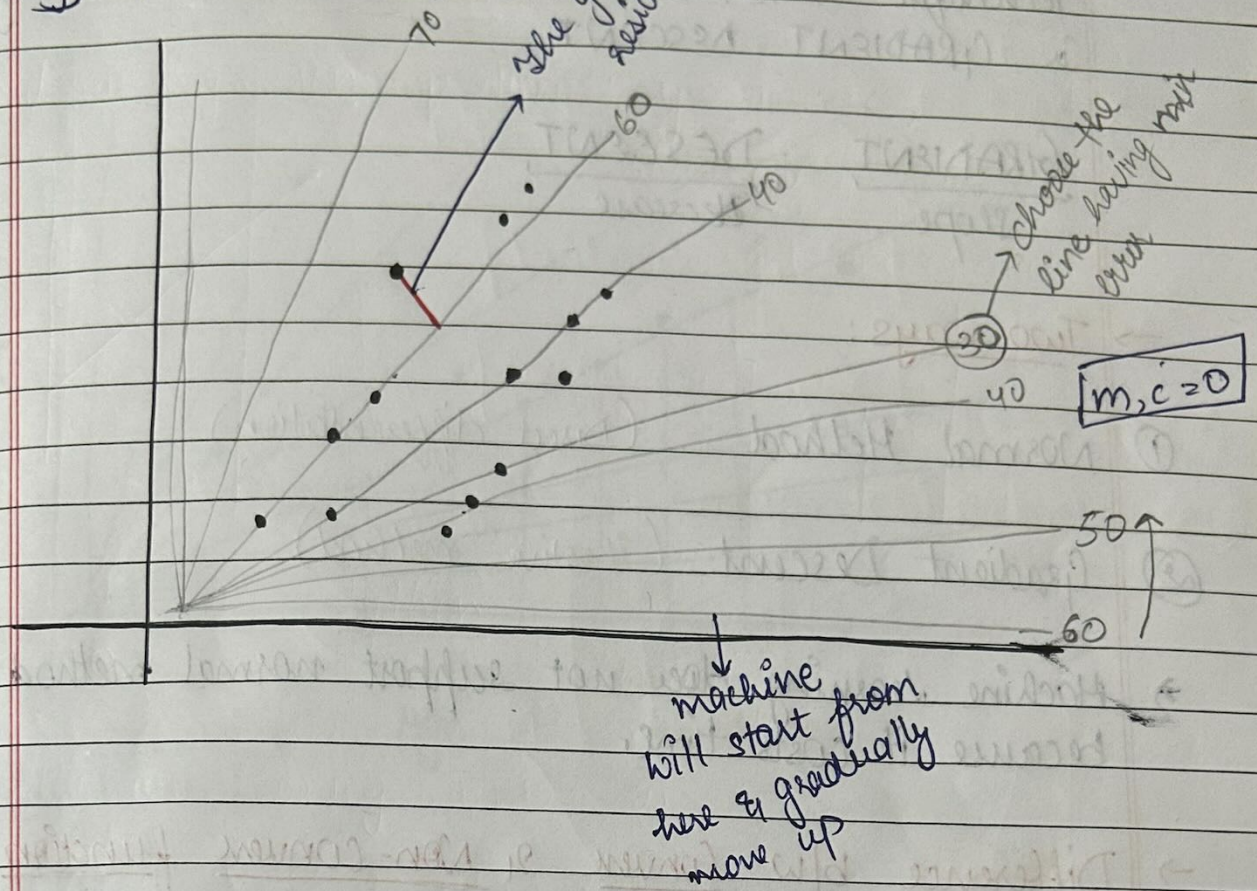
$$L = \frac{\sum_{i=1} (\text{Salary} - (m(\text{Age}) + c))^2}{N}$$

Fixed

Objective = Find m, c

Date / /
Page No.

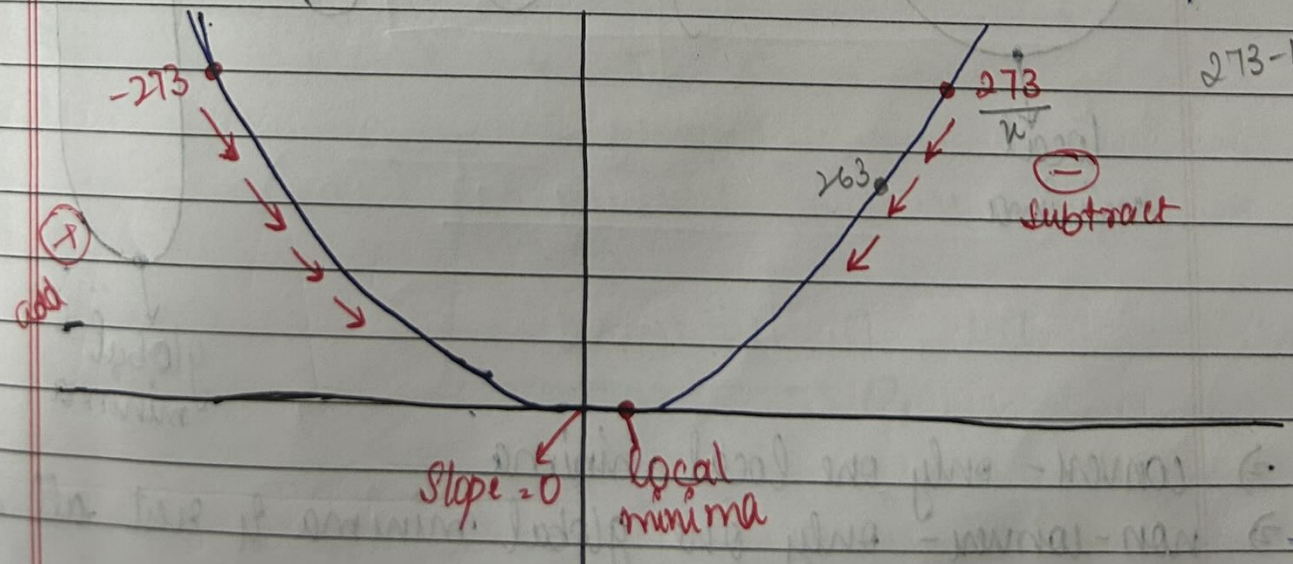
Eg) Dataset (Plotted)



⇒ L = Distance of all points from line

⇒ $L = \underbrace{\sum_{i=1}^{n} (y - (mx+c))}_{\text{minimize}}$

Eg) $y = 4x^2$ {Let slope = 10}



Technique which tells which direction, we have to go, is GRADIENT DESCENT

GRADIENT (slope) DESCENT (decrease)

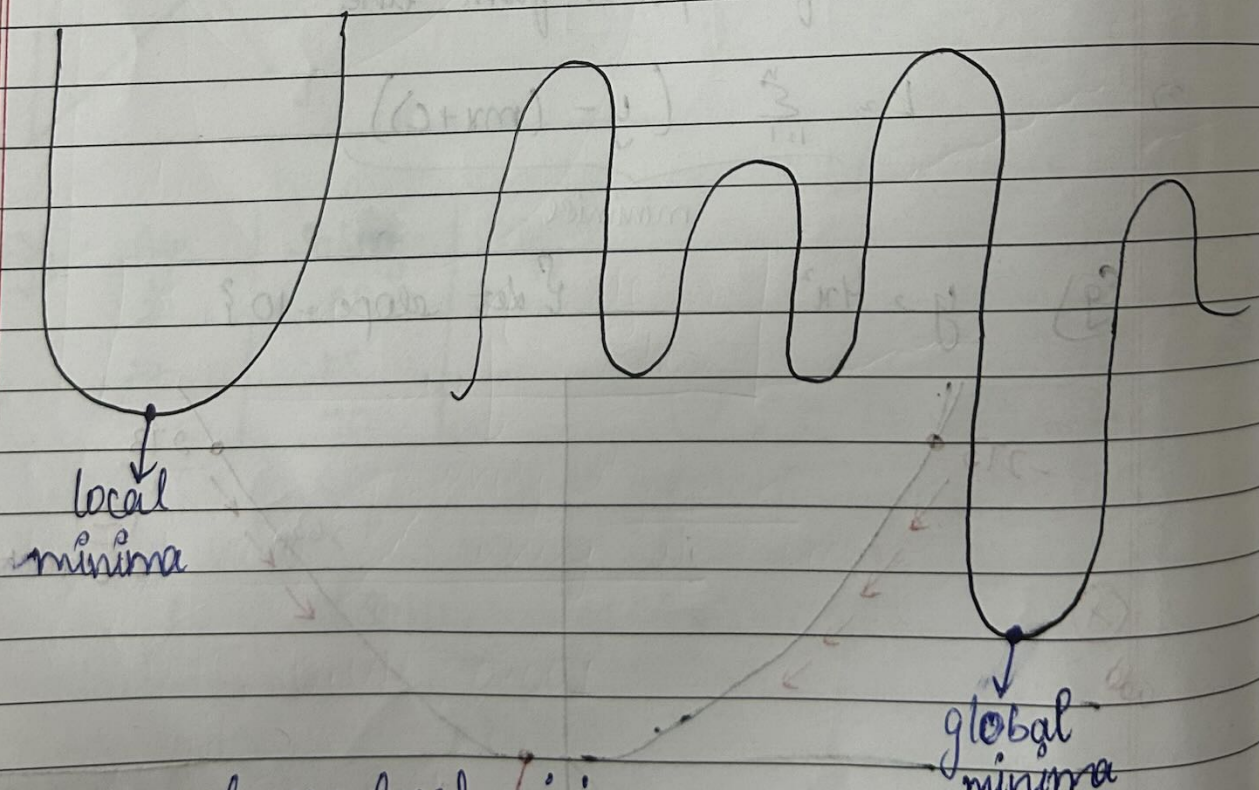
→ Two ways:

① Normal Method (Direct differentiation)

② Gradient Descent (iterative method)

⇒ Machine learning does not support normal method, because it costs time.

→ Difference b/w Convex & Non-convex function

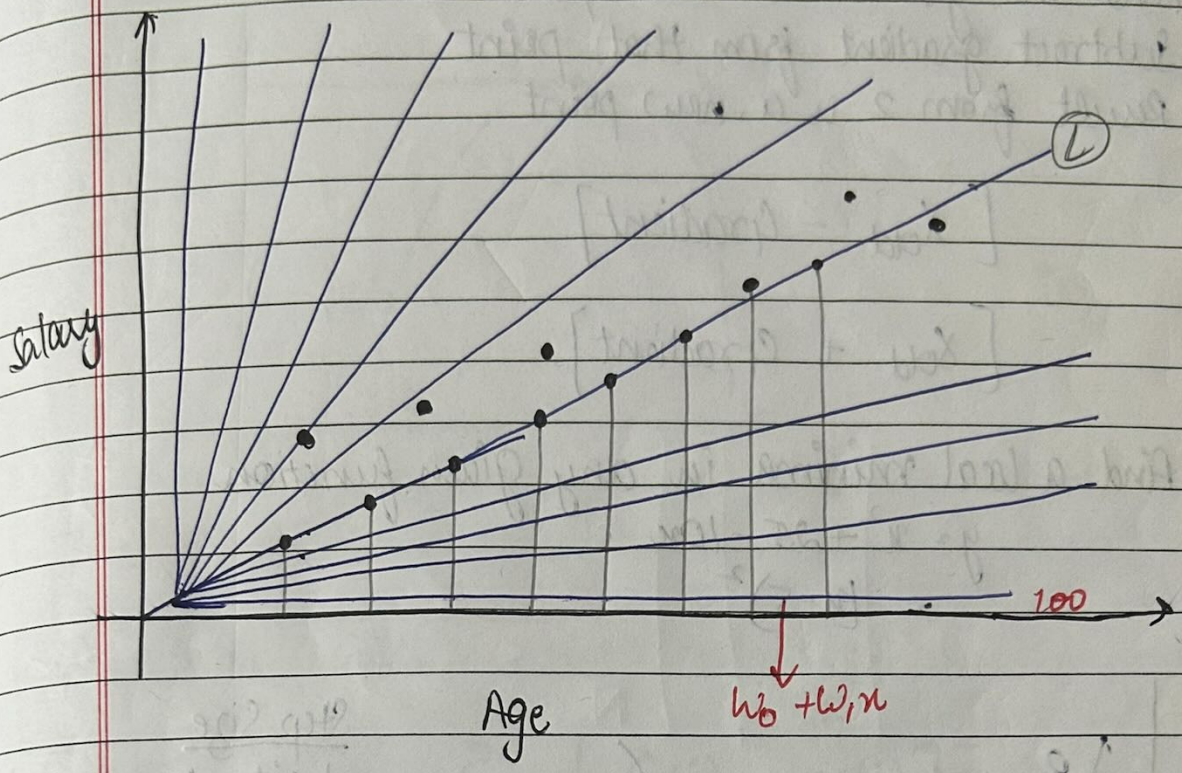


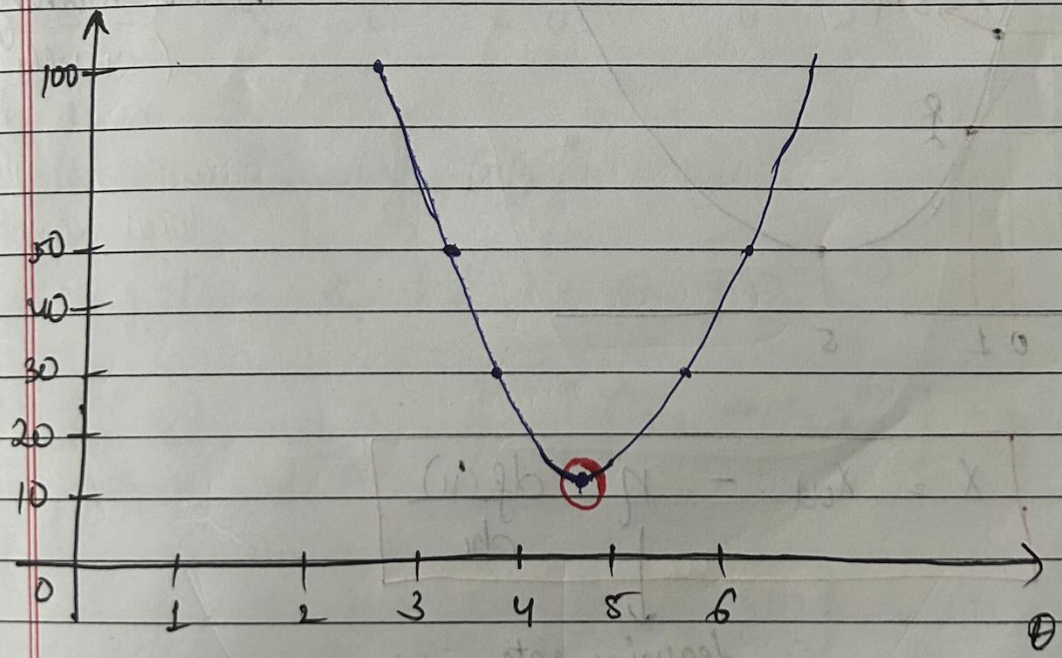
⇒ convex - only one local minima

⇒ non-convex - only one global minima & rest all are local minima

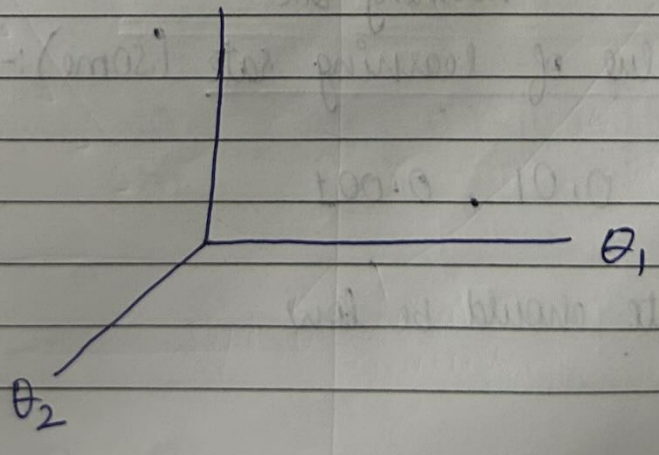
Date / /
Page No.

⇒ Curvature of Loss Function





⇒



→ Steps :-

i) Find the gradient at that point
ii) Subtract gradient from that point
iii) Result from 2 is a new point

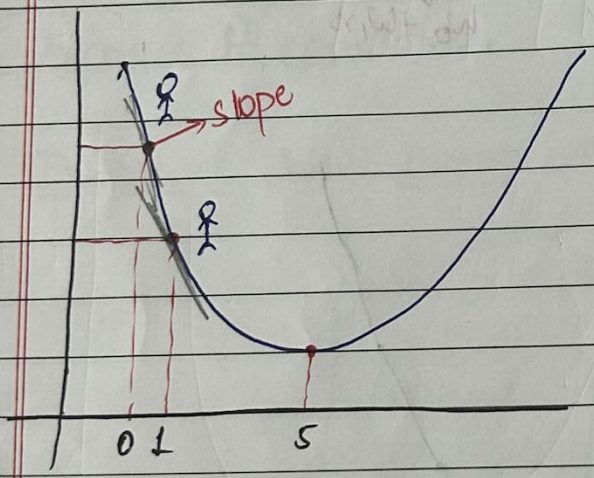
$[x_{old} - Gradient]$

$[x_{old} + Gradient]$

Ques) Find a local minima in any given function

$y = x^2 + 25 - 10x$

$= (x-5)^2$



Step Size decides learning rate (it tells how much steps are required to reach local minima)

$$x = x_{old} - \eta * \frac{df(x)}{dx}$$

η : learning rate

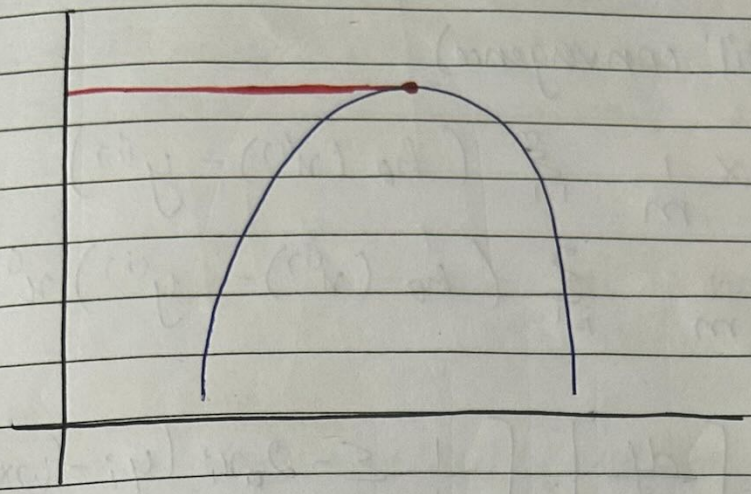
⇒ Standard value of learning rate (some) :-

0.1, 0.01, 0.001

⇒ Learning rate should be low

Date / /
Page No.

* For maxima, Gradient Ascent



→ Steps:

① θ = init()

② $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [\hat{y}(i) - y(i)]^2$

③ Update θ using GD

$$\hat{y}(i) = h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

* (i) $\frac{\partial J(\theta)}{\partial \theta_0} = \frac{2}{m} \sum_{i=1}^{m} [[\theta_0 + \theta_1 x^{(i)}] - y^{(i)}]$

$$= \frac{2}{m} \sum_{i=1}^{m} [\hat{y}(i) - y(i)]$$

(ii) $\frac{dJ(\theta)}{d\theta_1} = \frac{2}{m} \sum_{i=1}^{m} [\hat{y}(i) - y(i)] x^{(i)}$

Date / /
Page No.

Gradient Descent Algorithm

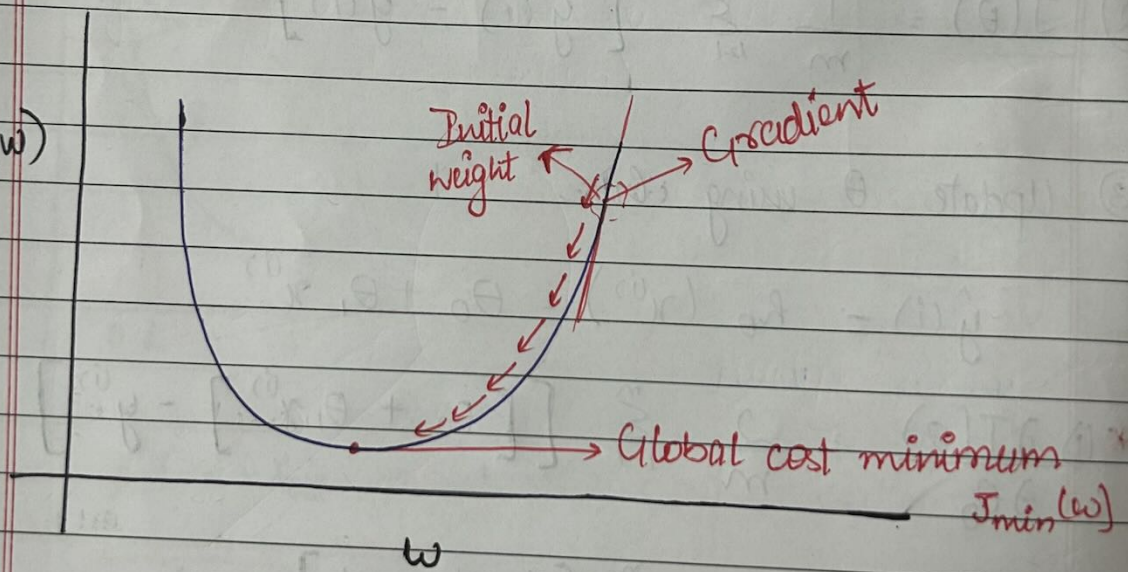
(repeat until convergence)

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

12/1/23

* $$J'(m, b) = \begin{bmatrix} \frac{df}{dw} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum -2x_i (y_i - (wx_i + b)) \\ \frac{1}{N} \sum -2(y_i - (wx_i + b)) \end{bmatrix}$$



Assumptions of Linear Regression

i) Linear relationship
ii) No multicollinearity
iii) Normality of residual
iv) Homoscedacity
v) No auto correlation

13/1/23

Date / /
Page No.

→ **Linear relationship / Linearity**

There should be linear relationship b/w independent & dependent variables, bcoz it suggests that a change in Y due to change in X is constant, regardless the value of X.

{non-linear → log, exponential to convert into linear}

→ **No multicollinearity**

Independent variables should not be correlated, absence of this phenomenon is known as multicollinearity.

Multicollinearity → Tolerance $(1-R^2)$ [coefficient of determination] $[T < 0.1$, exists$]$

↳ VIF $\left(\frac{1}{1-R^2}\right)$ $[VIF > 10$, exists$]$

{variance Inflation Factor}

→ **Normality of residual / Normal distribution of error**

Error terms must be normally distributed.

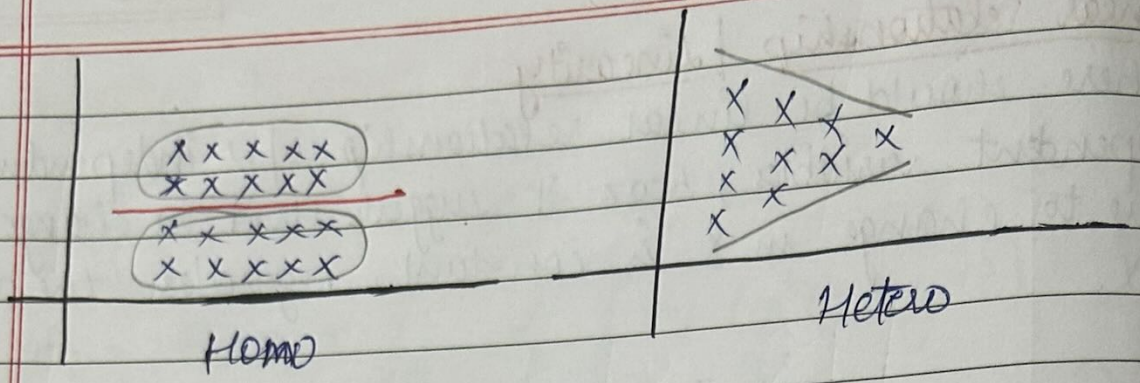
Normal distribution of error → Analytical (small dataset)

↳ Graphical (large not well)

Quantile Plot (to check normal distribution)

→ **Homoskedasticity**

Constant variance of the residuals

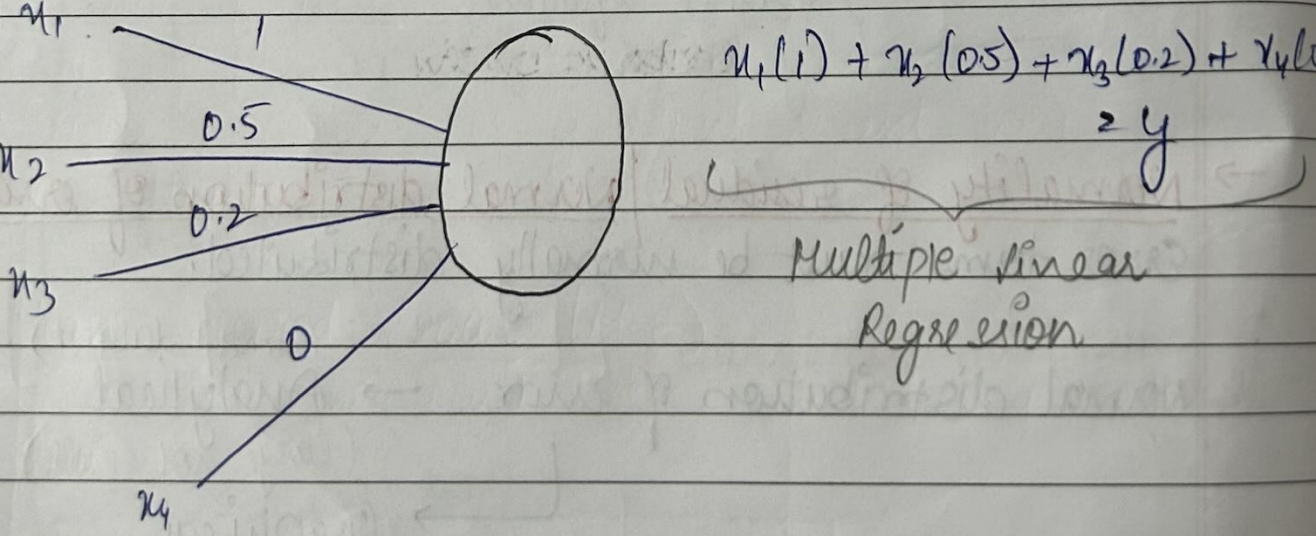


→ **No autocorrelation**
There should be no correlation b/w the residual (error) terms.

16/1/23

Ques) How to check performance of the model?
↔ Evaluation Metrics

* **Weights** - gives importance to inputs.
(for feature selection)



The incoming weights to the neuron is a measure to what the neuron cares about.

Types of Gradient Descent

Gradient Descent

Batch Gradient Descent

- → computes of whole training data
- → slow & computationaly expensive
- → not for huge training dataset

Stochastic Gradient Descent
(stochastic - random)

computes on single training sample
fast & less computationally expensive
can be used for huge training data.

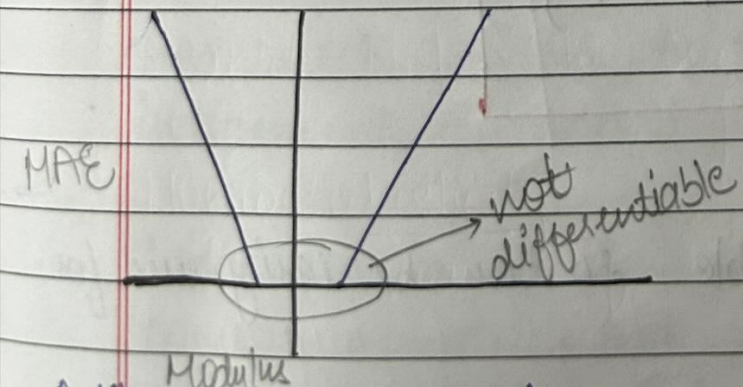
* GD
  - Batch
  - Stochastic
  - Mini-Batch (does in batches)

* Fastest - mini-batch gradient descent

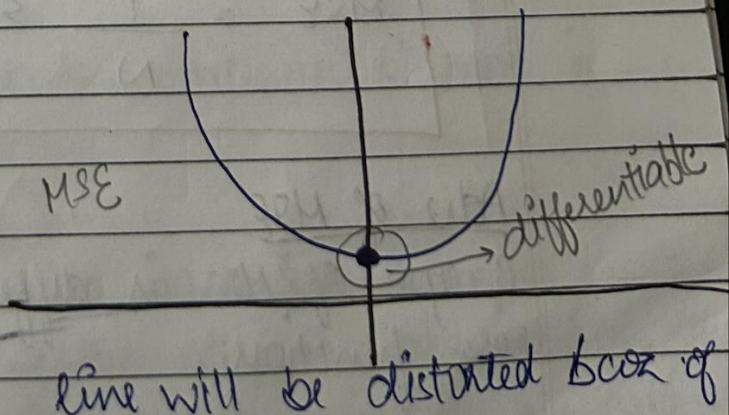
2) Evaluation Metrics

→ **Mean Absolute Error (MAE)**

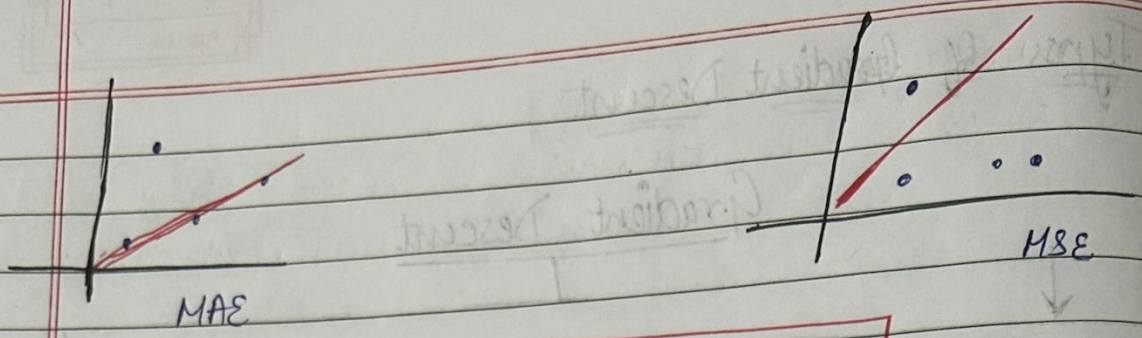
MAE calculates diff b/w actual & predicted values.



Outlies won't affect or have an impact on the line



line will be distorted bcoz of outliers



$$MAE = \frac{1}{N} \sum |y - \hat{y}|$$

- $\frac{1}{N}$: Divide by total no. of data points
- $\sum$: sum of Absolute value of residual
- y : Actual output
- $\hat{y}$: Predicted output

Adv of MAE

→ gives same unit as output variable
→ It is most robust (strong) to outliers

Disadv of MAE

Graph of MAE is not differentiable, so we have to apply various optimisers like gradient descent which can be differentiable.

→ **Mean Squared Error (MSE)**

It represents the squared distance b/w actual & predicted values.

$$MSE = \frac{1}{N} \sum (y - \hat{y})^2$$

Adv of MSE

Graph of MSE is differentiable, so can be easily used for loss function.

Date / /
Page No.

Dis adv of MSE

→ Output unit is squared unit. (Eg $m \rightarrow m^2$)

→ It penalises the outliers most if calculated MSE is bigger, ∴ it is not robust to outliers.

→ Root Mean Squared Error (RMSE)

It is square root of mean squared error

$$RMSE = \sqrt{MSE}$$

$$RMSE = \sqrt{\frac{1}{N} \Sigma (Y - \hat{Y})^2}$$

Adv of RMSE

same unit as output variable

Dis adv of RMSE

not robust to outliers

* Mostly preferred in deep learning

→ R Squared (R2) | Coefficient of Determination

It tells performance of the model, not the loss in absolute sense

MAE & MSE depends on the content, whereas R^2 score is independent of content.

With the help of R-squared we have baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call

a threshold which is fixed at 0.5. So, basically R^2 calculates how must regression line is better than a mean line.

→ also known as Goodness of fit

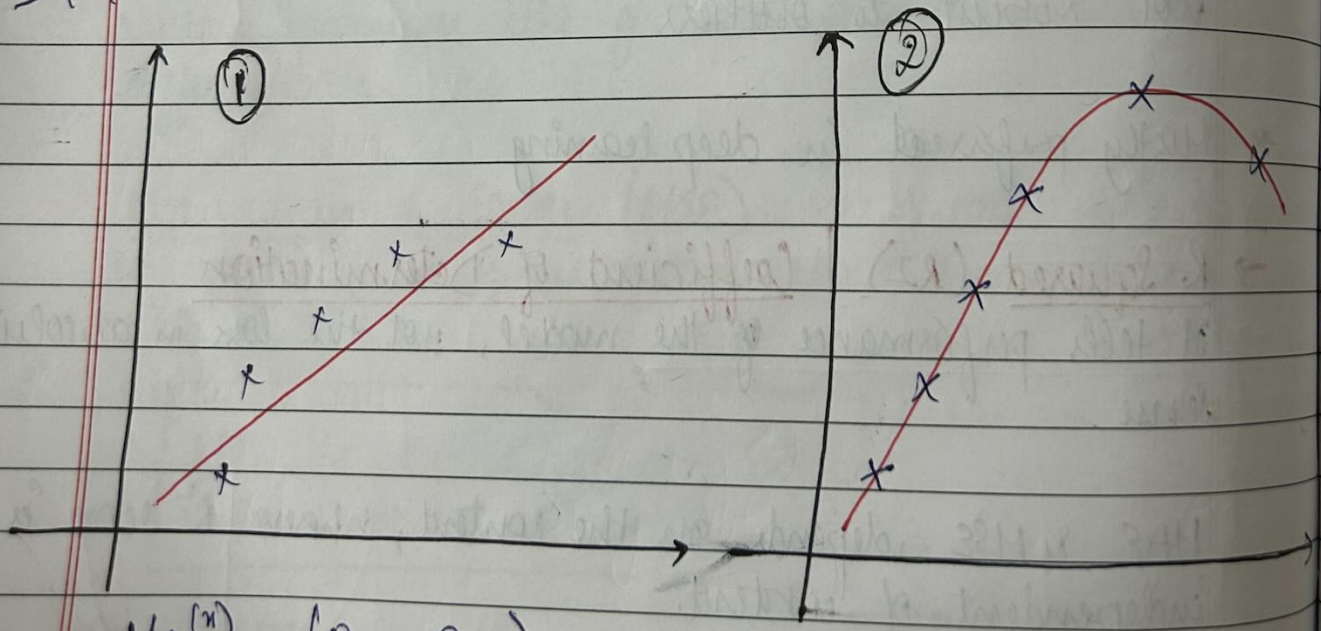
$$R^2 = 1 - \frac{RSS}{TSS}$$

RSS → sum of square of residuals
TSS → total sum of squres
R^2 → coefficient of determination

→ $R = 1$ (Perfect line)
$R = 0$ (worst line)
$R < 0$ $R > 0$

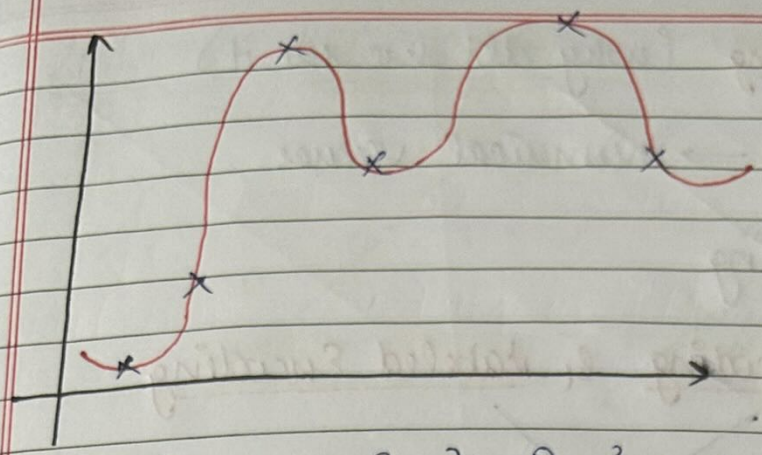
{ R should be near to 1 $R \to 1$ }

17/1/23.



$M\theta^{(x)} = (\theta_0 + \theta_1 x)$
1st hypothesis is giving me straight line which is underfitting

$M\theta^{(x)} = \theta_0 + \theta_1 x + \theta_2 x^2$
Here we are increasing the order of polynomial

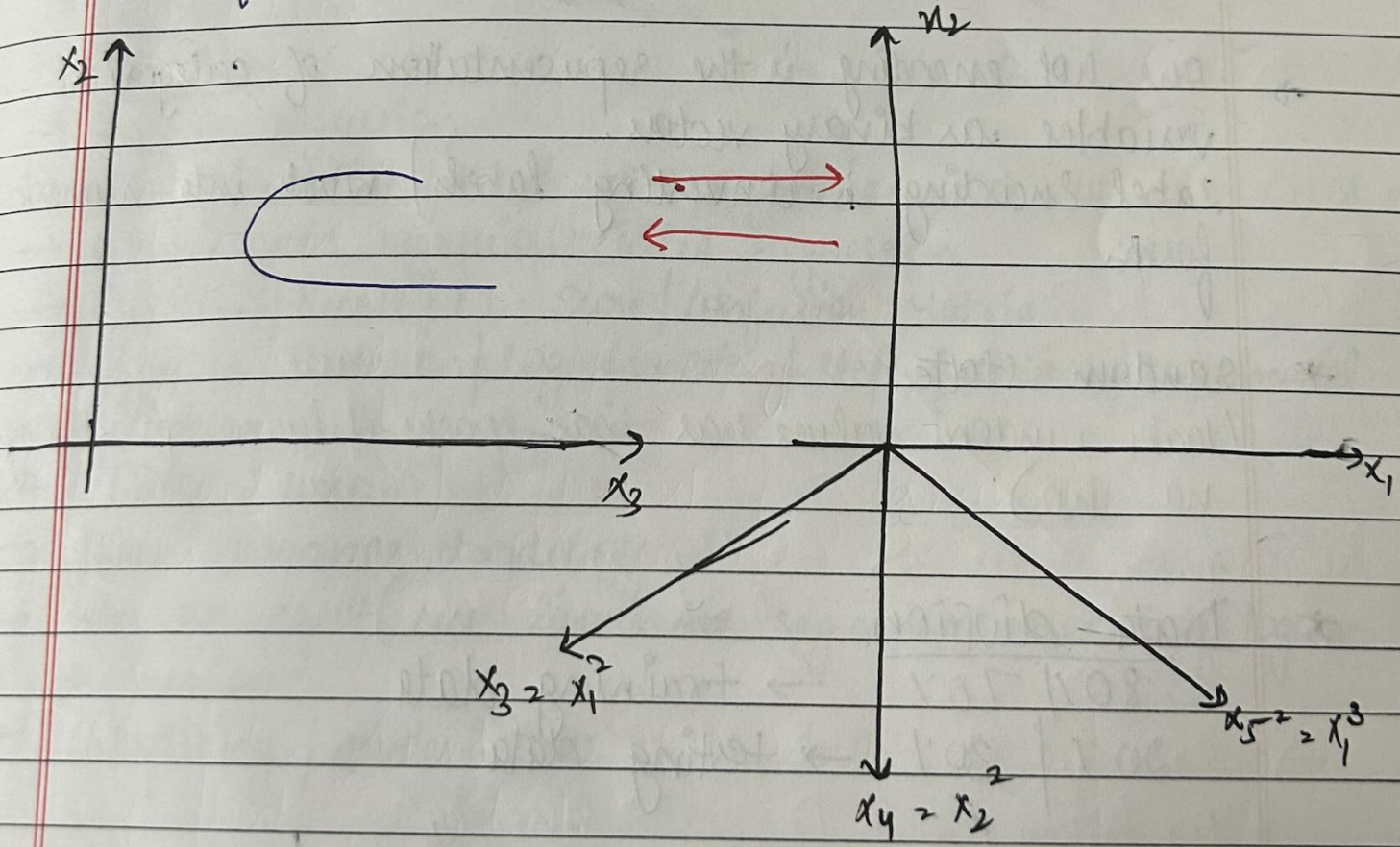


$M_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6$

Order of poly = 6 (Perfect)

→ Original data (quadratic) = $(x_1, x_2) \in R^2$

→ Transformed data (linear) = $f(x) = (x_1, x^2, x^3 \text{---} x^n)$



* numpy > dataframe (faster)
* Weights tells me importance of features
  - it tells us that feature is not imp.

Date / /
Page No.

* One Hot Encoding (why did we use it)

↓

Categorical values → numerical values

↑

Labelled Encoding

=> One Hot Encoding & Labeled Encoding

Categorical to vectors

=> Jahan pe order hai / meaning hai / comparison hai / rank hai / greater - less hai, wahan pe one hot encoding use hoti hai

=> One hot encoding is the representation of categorical variables as binary vectors.
Label Encoding is converting labels / words into numeric form.

* random state
(taaki random values har baar epoche chalaane pr change na ho)

* Data division
80% / 70% → training data
20% / 30% → testing data

=> Due to one hot encoding, parameters / inputs gets increased in number.
∴ we get multiple m values.