14. Oct

LINEAR REGRESSION ALGORITHM

Simple :- $h_\theta(x) = \theta_0 + \theta_1 x$

Multiple :- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$

CONVERGENCE ALGORITHM

Cost Function :- $J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x)^{(i)} - y^{(i)} \right)^2$ } MSE

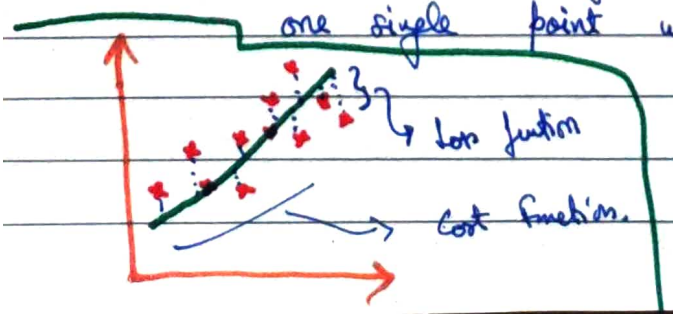
Loss Function :- $\left(h_\theta(x)^{(i)} - y^{(i)} \right)^2 \rightarrow \left(\hat{y}_{(i)} - y_{(i)} \right)^2$

Predicted value Actual value

Difference between **Loss** function and **Cost** function?

↳ In cost function we try to find out the distance b/w each & every point with the predicted value. And we do the summation of all these points. (m)

↳ But in case of Loss function we calculate distance b/w one single point with the predicted value. We take care of each difference.



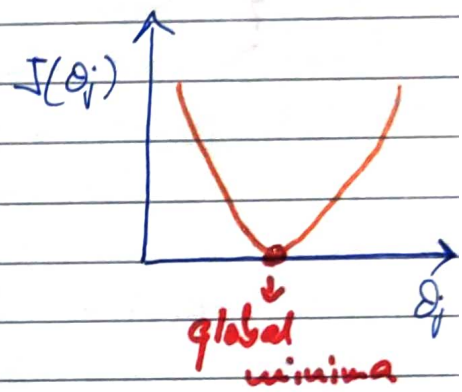
In Convergence Algorithm, we update θ_0 & θ_1 in order to move least fit line.

Repeat Until Convergence

$$\{ \quad \theta_j : \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_j) \qquad j = 1 \text{ to } m \quad \}$$

Q Why we need this?

Ans Because on left hand side, I'm trying to come to my global minima.



The most important thing is to calculate

$$\theta_j : \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J\theta(j)}$$

$J = \theta$

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

since, $h_\theta(x) = \theta_0 + \theta_1 x$

$$\hookrightarrow \frac{\partial}{\partial \theta_0} \left[\frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 x)^{(i)} - y^{(i)} \right)^2 \right]$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left[(\theta_0 + \theta_1 x) - y^{(i)} \right] \times 1$$

$\underline{\underline{J=1}}$

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

$$= \frac{2}{m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 x)^{(i)} - y^{(i)} \right) \times [x]$$

*****

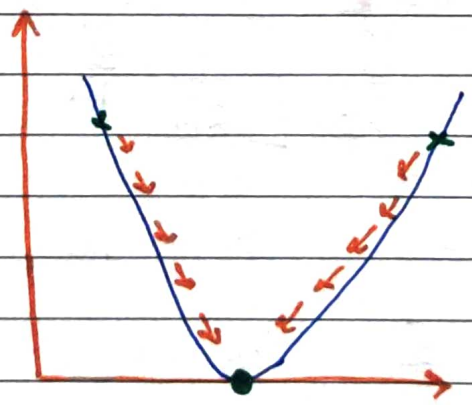
Repeat Until Convergence :-

$$\{$$

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

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

$$\}$$



Global Minima

Types of Cost Functions

1. MSE {Mean Squared Error}

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

↳ quadratic equation

$\hat{y} = \theta_0 + \theta_1 x$

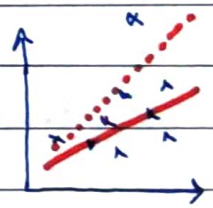
↳ Predicted Value

Advantage

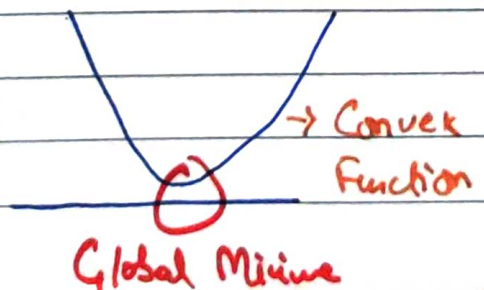
1. This equation is differentiable
2. This equation also has one GLOBAL MINIMA.
3. Our aim is to always work with Convex Function which gives us Parabola with one global Minima. And this is being done by MSE.

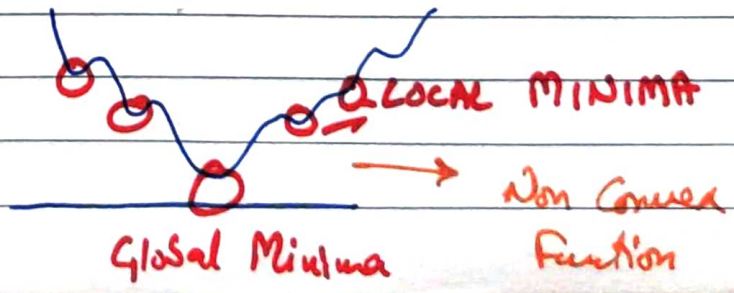
Disadvantage

1. This equation is not robust to outliers. An outlier will greatly increase Cost Function. So outliers must be removed earlier
2. Penalizes the errors & change the units.



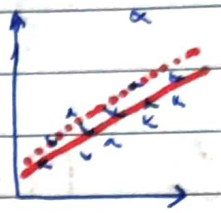
Whenever we plot a quadratic equation, we always get a PARABOLA MSE is a quadratic equation so it ensures a Parabola with one global Minima & NO LOCAL MINIMA!





2. ~~MEA~~ MAE { Mean Absolute Error }

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



Advantages

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

Disadvantages

1. Convergence usually takes more time.
2. optimization is a complex task
3. Time Consuming

3. RMSE { Root Mean Square Error }

$$\rightarrow \sqrt{MSE}$$

Advantages

1. RMSE is differentiable
2. Unit is going to be same

1. Same like MSE it will behave to Outliers

PERFORMANCE METRICS

To check if a model is good or not in Linear Regression, we use Performance Metrics.

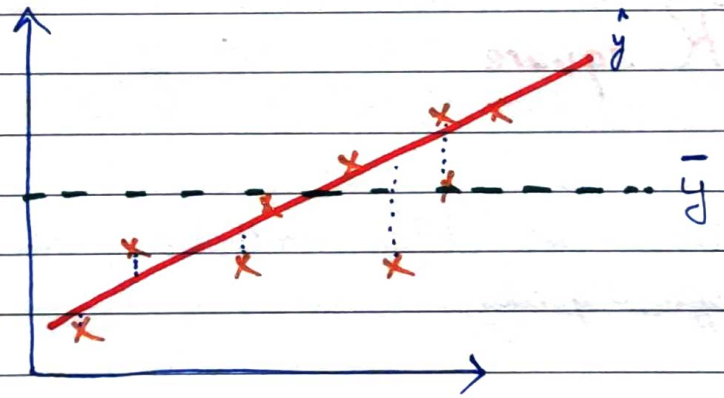
1. R Squared

$$\Rightarrow \boxed{1 - \frac{SS_{RES}}{SS_{Total}}}$$

SS_{res} = Residual sum of sq.
SS_{total} = Average sum of sq.

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

$\bar{y}$ = Average of y
$\hat{y}$ = Predicted value.



(Basic Model)

$SS_{Res} \rightarrow$ Small
$SS_{tot} \rightarrow$ Bigger

$$R \text{ squared} = 1 - \left\{ \frac{\text{smaller No.}}{\text{Bigger No.}} \right\} = \text{small No} \leq 1$$

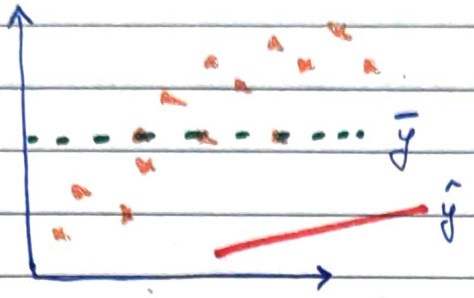
eg. 0.85
↳ 85%

Q Can R squared be negative?

Ans Yes it can be & it means Really Bad Model!

So if SS_{total} is smaller than SS_{RES}, its bad!

eg.



$R^2 = \underline{\underline{-ve}}$

We can see best fit line is placed totally away & $\bar{y}$ is much better.

<u>This is how we check Performance of model.</u>

2. Adjusted R square

Wherever the features are highly correlated, the increase in accuracy will happen quickly.

eg.

	Size of house	$\leftrightarrow$ Price	R^2 65%
+	City location	$\leftrightarrow$ Price	R^2 75%
+	No. of bedroom	$\leftrightarrow$ Price	R^2 85%

But if there is no direct correlation then accuracy may either not increase or mildly increase.

Using R^2, if we add another feature Gender $\leftrightarrow$ Price our R^2 will be 88%. <u>It SHOULDN'T HAVE HAPPENED</u>

That's where we use Adjusted R^2.

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

N = No. of data Points
P = No. of Independent Feature

Size of House | City location | No. of bed | Gender | Price

P=1	Size of House	⟷	Price	R^2 65%	AR^2 = 63%
P=2	+ City location	⟷	Price	R^2 75%	AR^2 73%
P=3	+ No. of bed	⟷	Price	R^2 85%	AR^2 83%
P=4	+ Gender	⟷	Price	R^2 87%	AR^2 80%

Major Difference!

* Adjusted R^2 drops when weak correlation features come up.
* Whereas, R^2 increased (slightly) at its place.

Adjusted R^2 should be used to evaluate the performance.

OVERFITTING & UNDERFITTING

{Bias & variance}

Let say we have 1000 datapoints.

Dataset

1000 Datapoints → TRAINING (700) → Model training

1000 Datapoints → TEST (300) → Model Test

⇓

New dataset

we will use Test dataset at the end.

Picking Training Dataset, will divide into 2 Parts :-

Training Dataset

500 TRAIN → Train the model

200 VALIDATE → Hyperparameter Tuning the model

Our Main Aim is to get:- ?

TRAIN DATA	Very good Accuracy 90%	[Low Bias]
TEST DATA	Very good Accuracy 85%	[Low Variance]

All this creates a Generalized Model

* Whenever we talk about TRAIN Data accuracy, we are point to use Bias

* whenever we talk about TEST Data accuracy, we are going to use variance.

Let say,

TRAIN	Very Good Accuracy (90%)	[Low Bias]
TEST	Very Bad Accuracy (50%)	[High Variance]

⇓

Overfitting

To solve this we will use Hyper parameter Tuning.

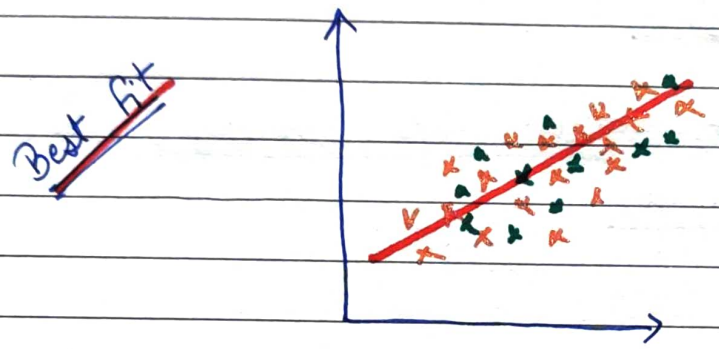
Let say,

TRAIN	Model Accuracy is low	[High Bias]
TEST	Model Accuracy is Low/High	[Low or High variance]

⇓

Underfitting

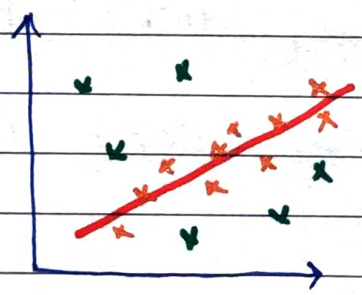
Generalized Model :-



× - Training Data
× - Test Data

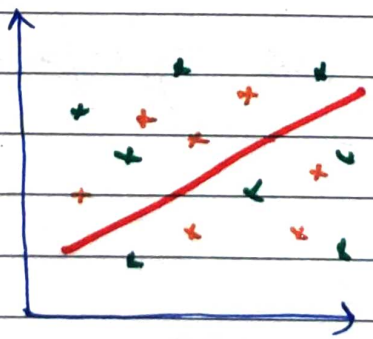
{ Both with High Accuracy }

Overfitting :-



{ Test Data with ~~High~~ Low Acc }

Underfitting :-



{ Both Low Accuracy }